B4A

Beginner's Guide

Edition 3.3

Last update : 2016.10.26

Main contributors: Klaus Christl (klaus), Erel Uziel (Erel)
Reviewers : Hubert Brandel, David Williamson, Bill Mackin, Narayanan Neelakantan.

To search for a given word or sentence use the Search function in the Edit menu.

All the source code and files needed (layouts, images etc.) of the example projects in this guide are included in the SourceCode folder.

Updated for B4A version 6.30 .

A more advanced guide can be downloaded User's Guide.

1 Getting started

B4A is a simple yet powerful development environment that targets Android devices.
B4A language is similar to Visual Basic language with additional support for objects.
B4A compiled applications are native Android applications; there are no extra runtimes or dependencies.
Unlike other IDE's, B4A is 100% focused on Android development.
B4A includes a powerful GUI designer with built-in support for multiple screens and orientations.
No XML writing is required.
You can develop and debug with:
- a real device connected via B4Abridge
- a real device connected via USBcable
- or an Android emulator.

B4A has a rich set of libraries that make it easy to develop advanced applications.
This includes: SQL databases, GPS, Serial ports (Bluetooth), Camera, XML parsing, Web services (HTTP), Services (background tasks), JSON, Animations, Network (TCP and UDP), Text To Speech (TTS), Voice Recognition, WebView, AdMob (ads), Charts, OpenGL, Graphics and more.

Android 1.6 and above are supported (including tablets).

1.1 Trial version

Look at this page for instructions how to use the trial version: www.b4x.com/b4a.html

1.2 Installing B4A and Android SDK

B4A depends on two additional (free) components:
- Java JDK
- Android SDK

Installation instructions:
The first step should be to install the **Java JDK**, as Android SDK requires it as well.
Note that there is no problem with having several versions of Java installed on the same computer.

- Open the Java 8 JDK download link.
- Check the Accept License Agreement radio button.
- Select **"Windows x86"** in the platforms list (for 64 bit machines as well).

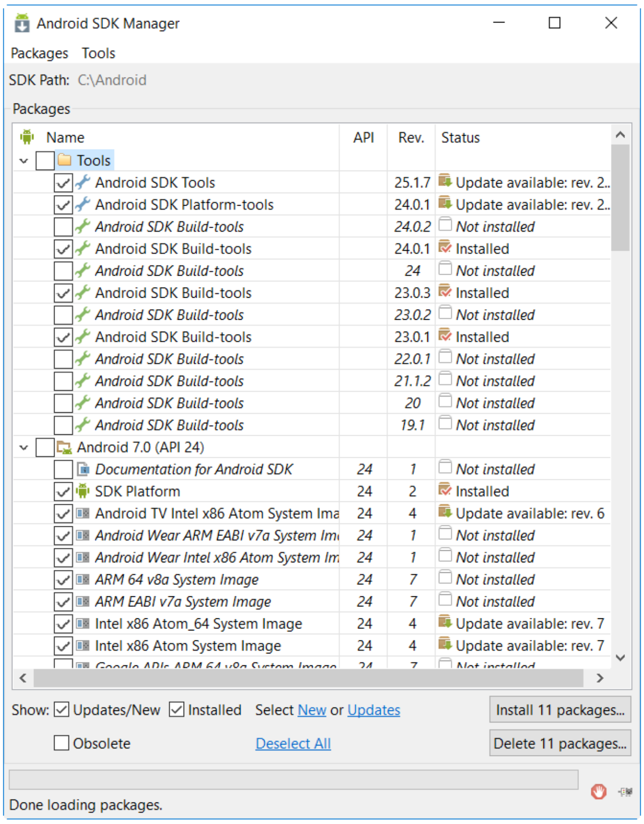
Android SDK doesn't work with Java 64bit JDK.
You should install the regular JDK for 64-bit computers as well.

- Download the file and install it.

The next step is to install the Android SDK and a platform:

- Install the SDK . **The SDK doesn't always behave properly when it is installed in a path with embedded spaces (like Program Files). It is recommended to install it to a custom folder similar to C:\Android**.
- You should now install the platform tools and at least one platform image. Use the latest one or at least API 8.

You can also install Google USB Driver if you need to connect a physical device with USB. A list of other drivers is available here.
Note that B4A allows you to connect to any device over the local network with B4A-Bridge tool.



Android SDK Manager

Packages Tools

SDK Path: C:\Android

Packages

Name	API	Rev.	Status
☐ Tools			
☑ Android SDK Tools		25.1.7	Update available: rev. 2..
☑ Android SDK Platform-tools		24.0.1	Update available: rev. 2..
☐ *Android SDK Build-tools*		*24.0.2*	☐ *Not installed*
☑ Android SDK Build-tools		24.0.1	Installed
☐ *Android SDK Build-tools*		*24*	☐ *Not installed*
☑ Android SDK Build-tools		23.0.3	Installed
☐ *Android SDK Build-tools*		*23.0.2*	☐ *Not installed*
☑ Android SDK Build-tools		23.0.1	Installed
☐ *Android SDK Build-tools*		*22.0.1*	☐ *Not installed*
☐ *Android SDK Build-tools*		*21.1.2*	☐ *Not installed*
☐ *Android SDK Build-tools*		*20*	☐ *Not installed*
☐ *Android SDK Build-tools*		*19.1*	☐ *Not installed*
☐ Android 7.0 (API 24)			
☐ *Documentation for Android SDK*	*24*	*1*	☐ *Not installed*
☑ SDK Platform	24	2	Installed
☑ Android TV Intel x86 Atom System Ima	24	4	Update available: rev. 6
☑ *Android Wear ARM EABI v7a System Im*	*24*	*1*	☐ *Not installed*
☑ *Android Wear Intel x86 Atom System Im*	*24*	*1*	☐ *Not installed*
☑ *ARM 64 v8a System Image*	*24*	*7*	☐ *Not installed*
☑ *ARM EABI v7a System Image*	*24*	*7*	☐ *Not installed*
☑ Intel x86 Atom_64 System Image	24	4	Update available: rev. 7
☑ Intel x86 Atom System Image	24	4	Update available: rev. 7
☐ *Google APIs ARM 64 v8a System Image*	*24*	*7*	☐ *Not installed*

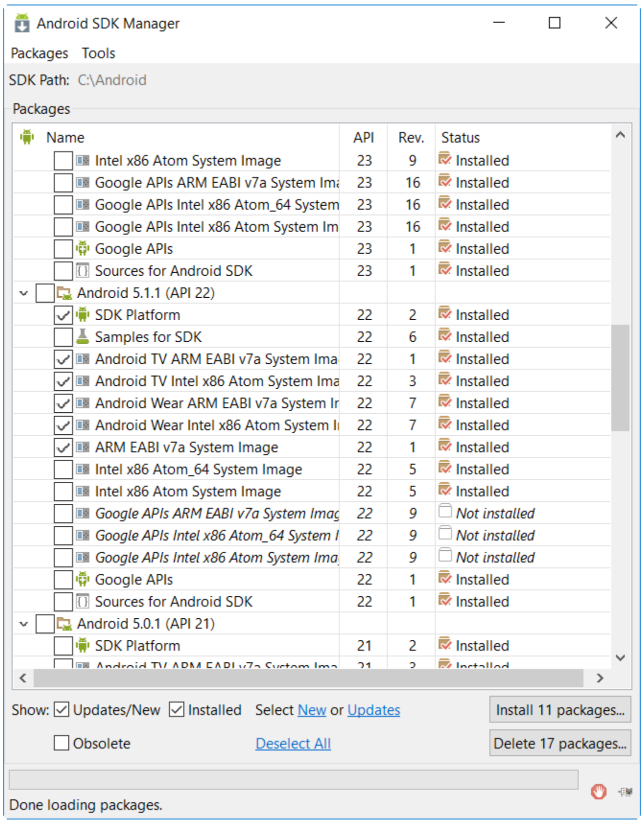
Show: ☑ Updates/New ☑ Installed Select New or Updates [Install 11 packages...]

☐ Obsolete Deselect All [Delete 11 packages...]

Done loading packages.

A screen similar to this will be shown.

Select the API version you want to download.
In the example, I choose API 24.



Android SDK Manager

Packages Tools

SDK Path: C:\Android

Packages

Name	API	Rev.	Status
☐ Intel x86 Atom System Image	23	9	Installed
☐ Google APIs ARM EABI v7a System Ima	23	16	Installed
☐ Google APIs Intel x86 Atom_64 System	23	16	Installed
☐ Google APIs Intel x86 Atom System Im	23	16	Installed
☐ Google APIs	23	1	Installed
☐ Sources for Android SDK	23	1	Installed
☐ Android 5.1.1 (API 22)			
☑ SDK Platform	22	2	Installed
☐ Samples for SDK	22	6	Installed
☑ Android TV ARM EABI v7a System Ima	22	1	Installed
☑ Android TV Intel x86 Atom System Ima	22	3	Installed
☑ Android Wear ARM EABI v7a System Ir	22	7	Installed
☑ Android Wear Intel x86 Atom System Ii	22	7	Installed
☑ ARM EABI v7a System Image	22	1	Installed
☐ Intel x86 Atom_64 System Image	22	5	Installed
☐ Intel x86 Atom System Image	22	5	Installed
☐ *Google APIs ARM EABI v7a System Imag*	*22*	*9*	*Not installed*
☐ *Google APIs Intel x86 Atom_64 System I*	*22*	*9*	*Not installed*
☐ *Google APIs Intel x86 Atom System Ima*	*22*	*9*	*Not installed*
☐ Google APIs	22	1	Installed
☐ Sources for Android SDK	22	1	Installed
☐ Android 5.0.1 (API 21)			
☐ SDK Platform	21	2	Installed
☐ Android TV ARM EABI v7a System Ima	21	2	Installed

Show: ☑ Updates/New ☑ Installed Select New or Updates Install 11 packages...

☐ Obsolete Deselect All Delete 17 packages...

Done loading packages.

You can select several APIs and install them in parallel.

In this example, API 22 is also selected.

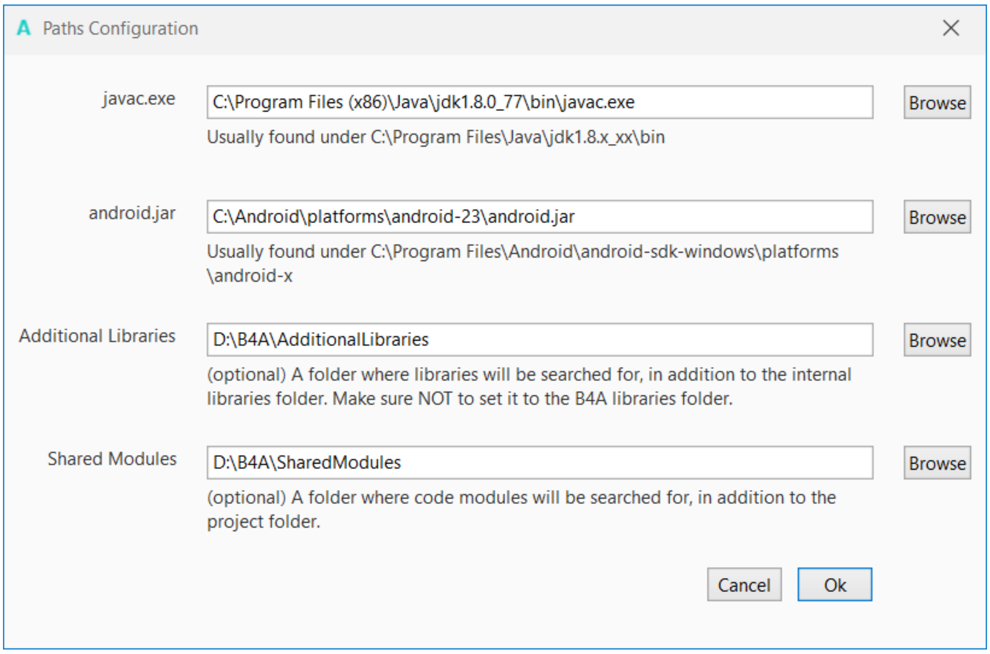
Note that you can install more packages later.

- Press on Install Selected and install both packages.

If you want to connect a device with USB you might also download the Google USB driver.

1.2.1 Install and configure B4A

- Download and install B4A.
- Open B4A.
- Choose Tools menu - Configure Paths.



Paths Configuration

javac.exe: C:\Program Files (x86)\Java\jdk1.8.0_77\bin\javac.exe
Usually found under C:\Program Files\Java\jdk1.8.x_xx\bin

android.jar: C:\Android\platforms\android-23\android.jar
Usually found under C:\Program Files\Android\android-sdk-windows\platforms \android-x

Additional Libraries: D:\B4A\AdditionalLibraries
(optional) A folder where libraries will be searched for, in addition to the internal libraries folder. Make sure NOT to set it to the B4A libraries folder.

Shared Modules: D:\B4A\SharedModules
(optional) A folder where code modules will be searched for, in addition to the project folder.

- Use the browse buttons to locate "javac.exe" and "android.jar"
javac is located under <java folder>\bin.
android.jar is located under <android-sdk-windows>\platforms\android-2**1.**
The folder depends on where you installed the Android SDK,
It should be: C:\Android\platforms\android-21\android.jar
or C:\Android\platforms\android-8\android.jar.
The number depends on the Android version you loaded.

On older versions it could be under:
C:\Android\android-sdk-windows\platforms\android-8\android.jar.
On Windows 64 bit, Java will probably be installed under C:\Program Files (x86).

It is recommended to create a specific folder for Additional libraries.
B4A utilizes two types of libraries:

- Standard libraries, which come with B4A and are located in the Libraries folder of B4A.
  These libraries are automatically updated when you install a new version of B4A.
- Additional libraries, which are not part of B4A, and are mostly written by members. These libraries should be saved in a specific folder different from the standard libraries folder.
  More details in Chapter 14.7.2 Additional libraries folder.

Shared modules: Module files can be shared between different projects and must therefore be saved in a specific folder. More details in 11.5 Shared modules.

Common errors
- Windows XP - "Basic4Android.exe Application could not be initialised correctly error 0xc0000135" on start-up. B4A requires .Net Framework 4.0 or above.
Windows XP users who didn't install it before should first install the framework.

1.3 Installing B4A Bridge

B4A Bridge is the advised link between B4A and your device(s).

It is made of two components. One component runs on the device and allows the second component which is part of the IDE to connect and communicate with the device.
The connection is done over the local network or with a Bluetooth connection.

Once connected, B4A-Bridge supports all of the IDE features which include: installing applications, viewing the logs, debugging and the visual designer (taking screenshots is not supported).

Android doesn't allow applications to quietly install other applications, therefore when you run your application using B4A-Bridge you will see a dialog asking you to approve the installation.

Getting started with B4A-Bridge
1. First you need to install B4A-Bridge on your device.
B4A-Bridge can be downloaded here: http://www.basic4ppc.com/android/files/b4a_bridge.apk.
Some browsers may treat this file as a zip file. In that case you should restore its apk extension.

B4A-Bridge is also available in Google Play and Amazon Market. Search for: B4A Bridge.
Note that you need to allow installation of applications from "Unknown sources". This is done by choosing Settings from the Home screen - Manage Applications.

B4A-Bridge requires writable storage card. It is not possible to install applications without it.

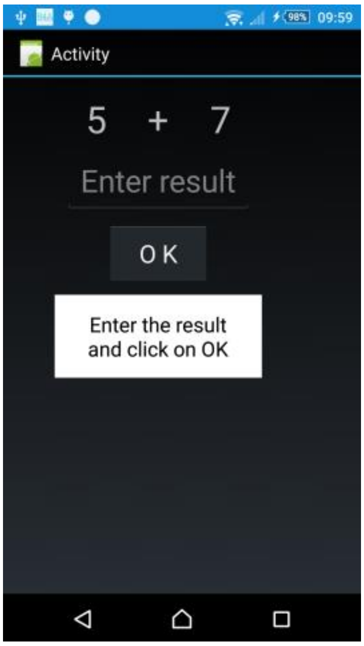
See chapter onnecting a real device via B4A Bridge on how to connect your device to the IDE.

2 My first program (MyFirstProgram.b4a)

Let us write our first program. The suggested program is a math trainer for kids.

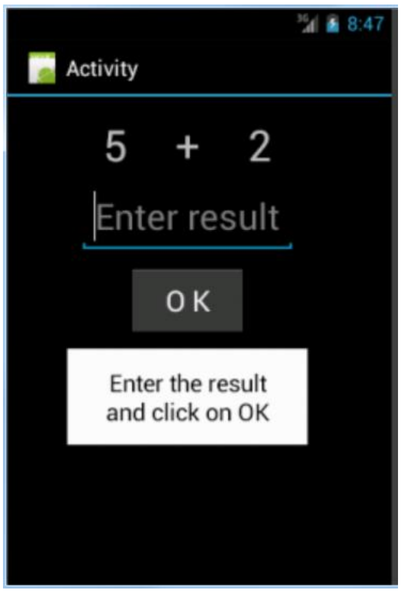
The project is available in the SourceCode folder:
SourceCode\MyFirstProgram\ MyFirstProgram.b4a

The look of the screen is different depending on the Android version of the devices, also with Emulators.



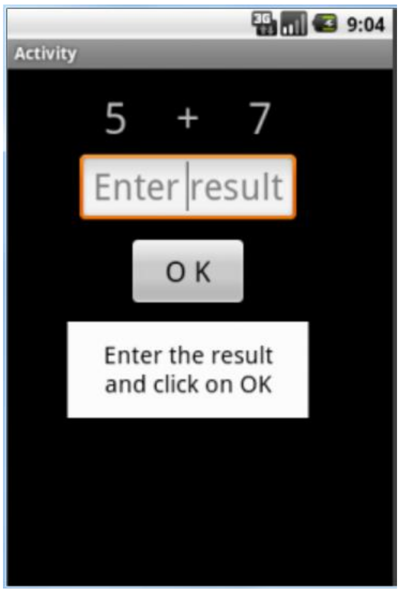
Enter the result and click on OK

Sony xperia z1



Enter the result and click on OK

Emulator Android version 4.2



Enter the result and click on OK

Emulator Android version 2.2

On the screen, we will have:
- 2 Labels displaying randomly generated numbers (between 1 and 9)
- 1 Label with the math sign (+)
- 1 EditText view where the user must enter the result
- 1 Button, used to either confirm when the user has finished entering the result or generate a new calculation.
- 1 Label with a comment about the result.

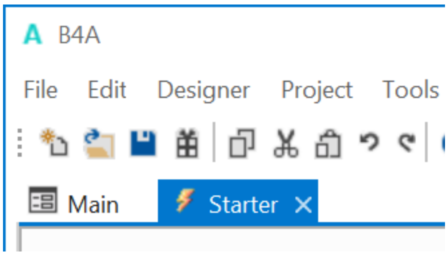
In Android:

- Label	is an object to show text.
- EditText	is an object allowing the user to enter text.
- Button	is an object allowing user actions.

We will design the layout of the user interface with the VisualDesigner and go step by step through the whole process.

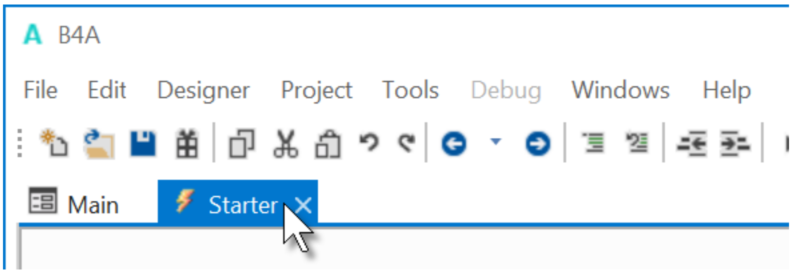
Run the IDE B4A 5.02

When you open the IDE you will see on the top left two Tabs Main and Starter.

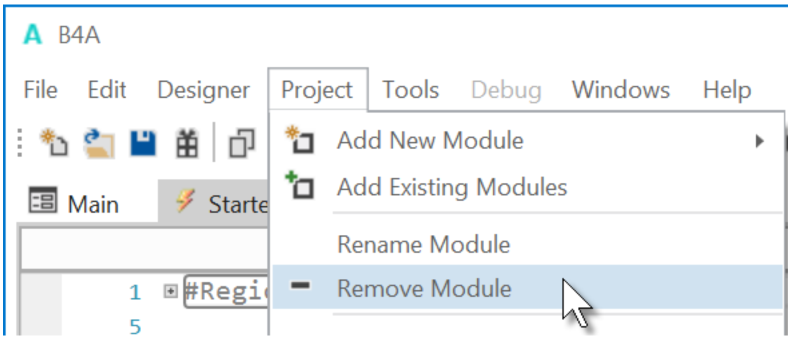


- Main × Is the Main module for B4A which is the normal starting module. Its name cannot be changed.
- Starter Is a Service, which is startet when the program is lauched.

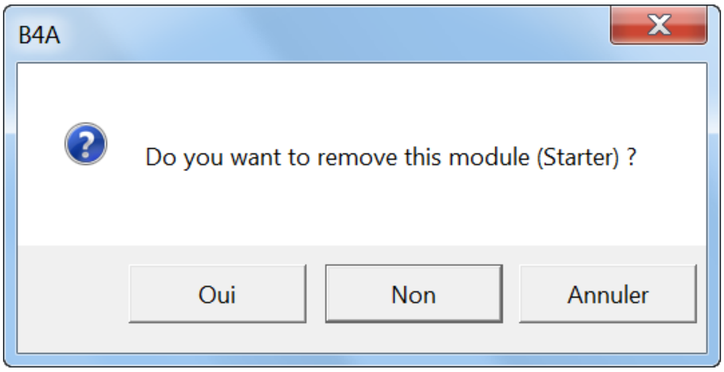
For our first program we don't need this Starter service, so we delete it.



Click on Starter × tab to select the service.

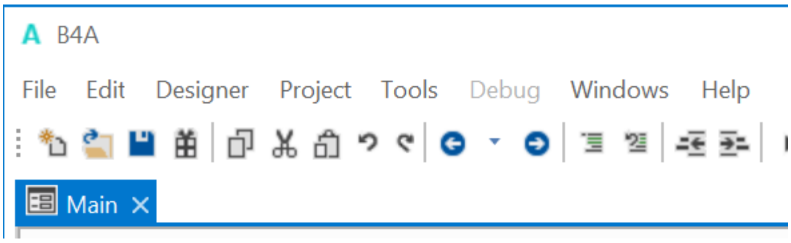


In the menu Project click on Remove Module.



Do you want to remove this module (Starter) ?

You are asked if you really want to remove it.
Click on OUI (YES)



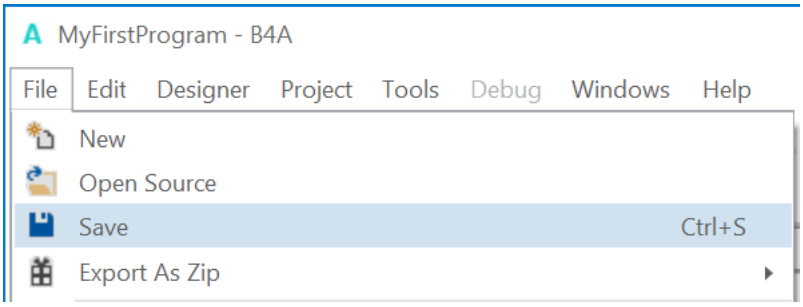
The Starter service module is removed.

You could also leave the Starter service, the removal is not mandatory.

Save the project.

You must save the project before you can run the Designer.

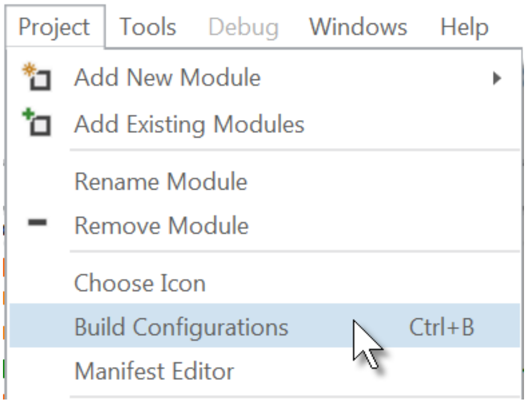
Create a new folder MyFirstProgram and save the project with the name MyFirstProgram.



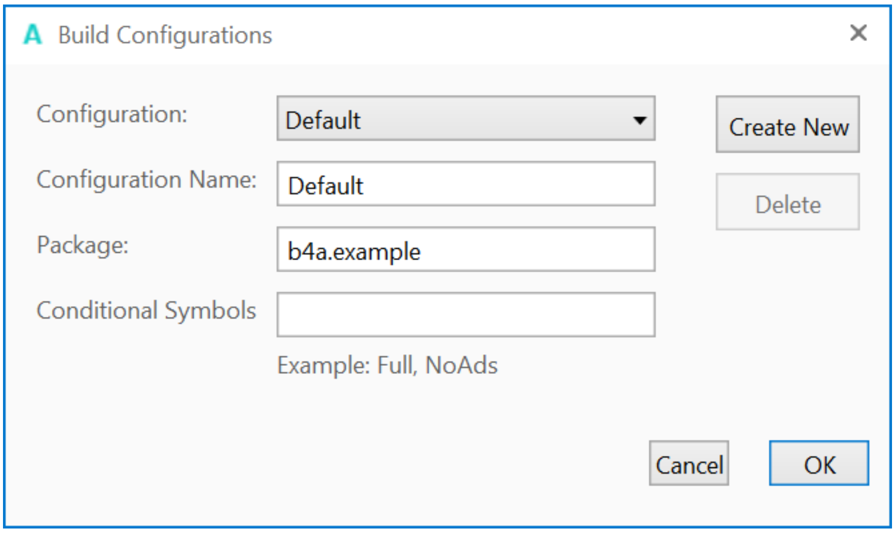
Set the Package Name.

Each program needs a package name.

In the menu Project click on Build Configurations.



This window appears:



Configuration: Default

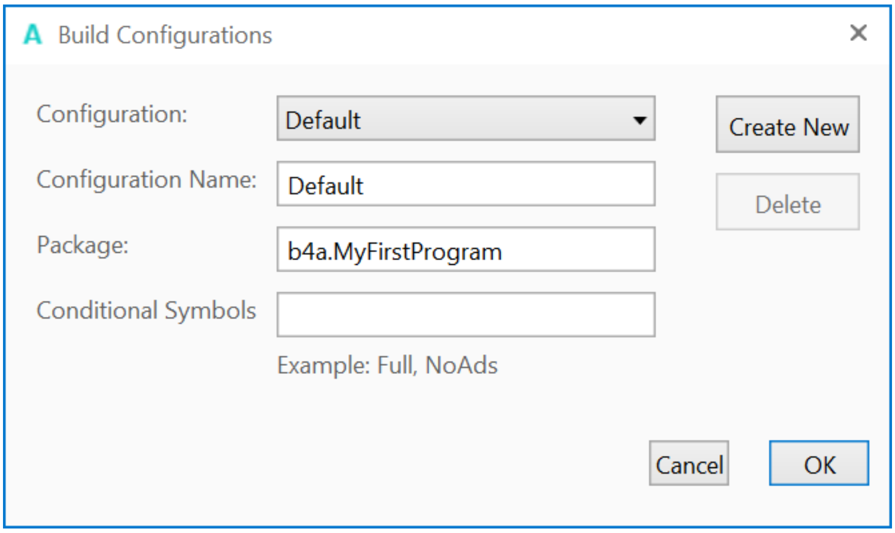
Configuration Name: Default

Package: b4a.example

Conditional Symbols:

Example: Full, NoAds

The default name is b4a.example.



Configuration: Default

Configuration Name: Default

Package: b4a.MyFirstProgram

Conditional Symbols:

Example: Full, NoAds

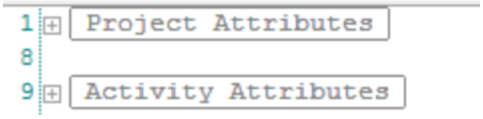
We will change it to b4a.MyFirstProgram.

And click on OK.

Set the Application Label.

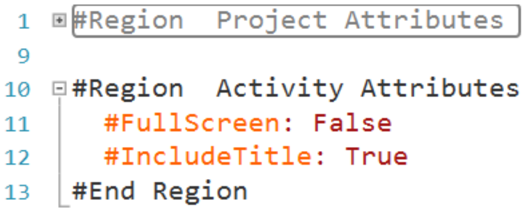
The Application label is the name of the program that will be shown on the device.

On top of the code screen you see these two lines showing two 'regions'.



```
⊞ Project Attributes

⊞ Activity Attributes
```

Regions are code parts which can be collapsed or extended.
Clicking on ⊞ will expand the Region.
Clicking on ⊟ will collapse the Region.
Regions are explained in Chapter Collapse a Region.



```
⊞ #Region  Project Attributes

⊟ #Region  Activity Attributes
     #FullScreen: False
     #IncludeTitle: True
  #End Region
```

```
#Region  Project Attributes
	#ApplicationLabel: B4A Example
	#VersionCode: 1
	#VersionName: 
	'SupportedOrientations possible values: unspecified, landscape or portrait.
	#SupportedOrientations: unspecified
	#CanInstallToExternalStorage: False
#End Region

#Region  Activity Attributes
	#FullScreen: False
	#IncludeTitle: True
#End Region
```

The default name is `B4A Example`, but we will change it to `MyFirstProgram` for naming consistency.

Change this line:

```
   #ApplicationLabel: B4A Example
```

to

```
   #ApplicationLabel: MyFirstProgram
```

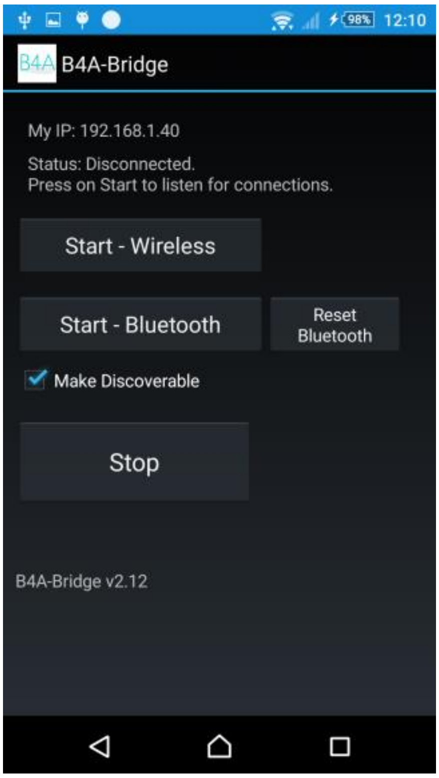
The other lines are explained in Chapter Code header Project Attributes / Activity Attributes.

Connect a device

To test the program you should connect a device to the IDE.
The best connection is via B4A-Bridge.

It is also possible to connect an Emulator.

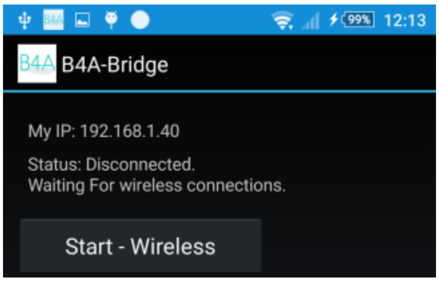
On the device run B4A-Bridge. If you haven't it on your device it's the right moment to install it.



My IP: 192.168.1.40
Status: Disconnected.
Press on Start to listen for connections.

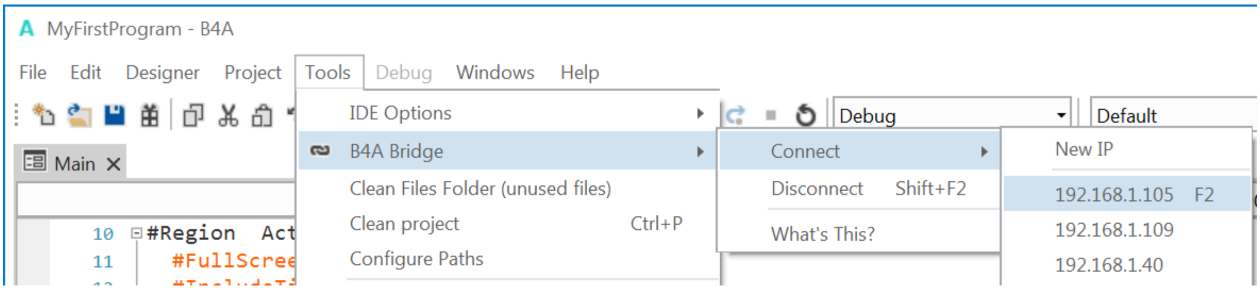
B4A-Bridge v2.12

Click on Start - Wireless.



My IP: 192.168.1.40
Status: Disconnected.
Waiting For wireless connections.

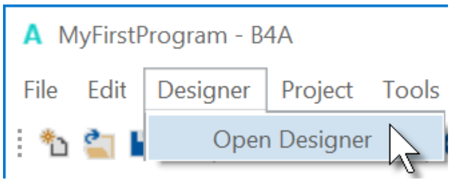
You will see Status: Disconnected. Waiting For wireless connections. on top.



```
#Region  Act
   #FullScree
```

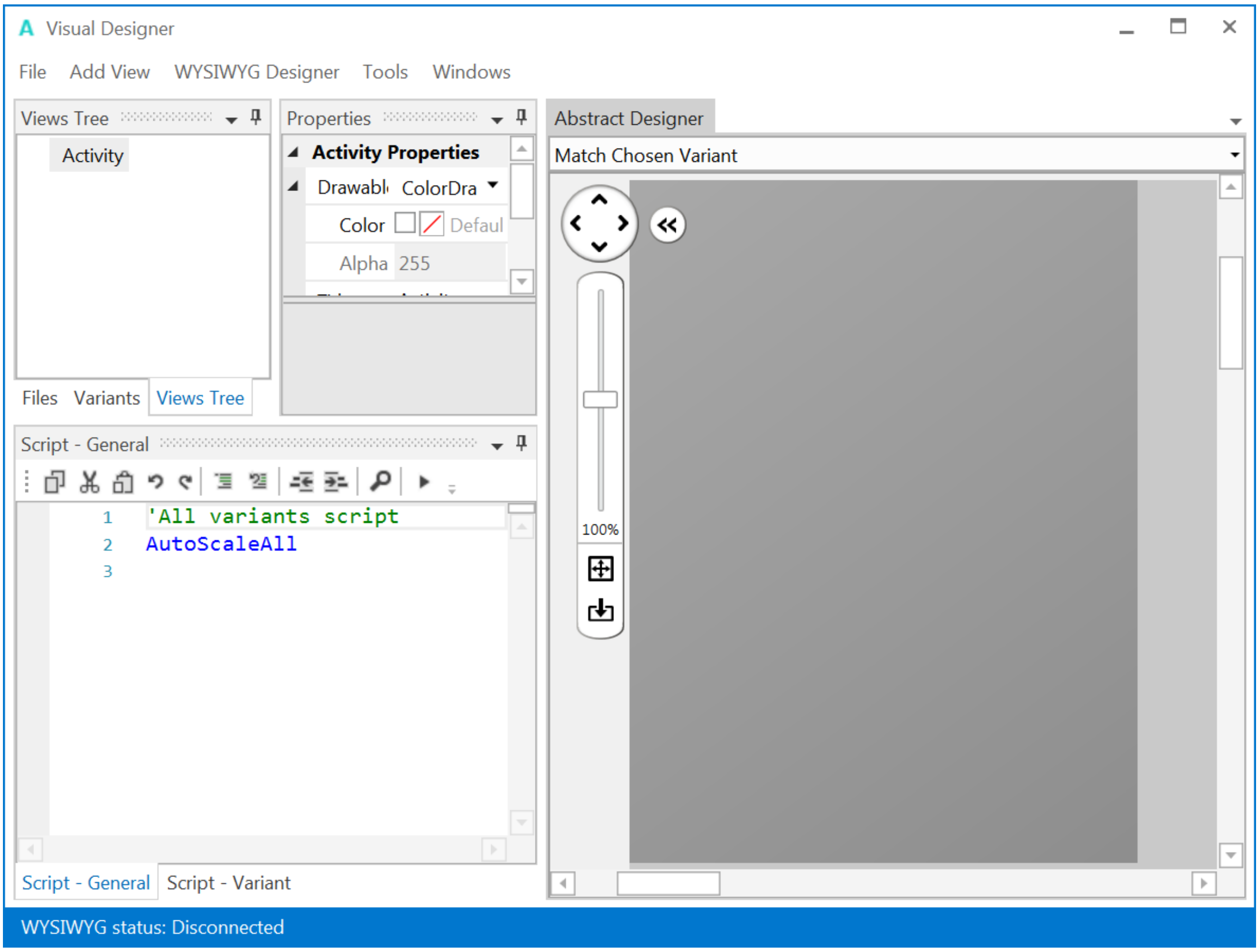
In the IDE click on the address of the device you want to connect.
The address is shown on the B4A-Bridge screen on the device.

In the IDE run the Designer.



Click on Open Designer.

The Visual Designer looks like this.



```
'All variants script
AutoScaleAll
```

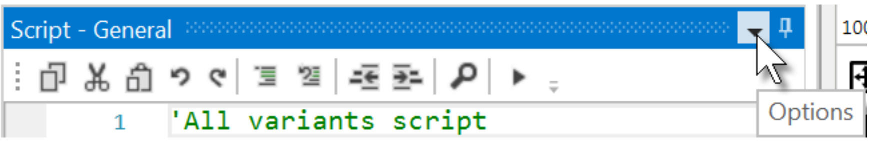
There are different windows:

- Views Tree shows all views as a tree.
- Properties shows all properties of the selected view.
- Abstract Designer shows the views on a screen
- Script - General allows to ‘fine tune’ the layouts.

The Designer is explained in detail in the chapter The Designer.

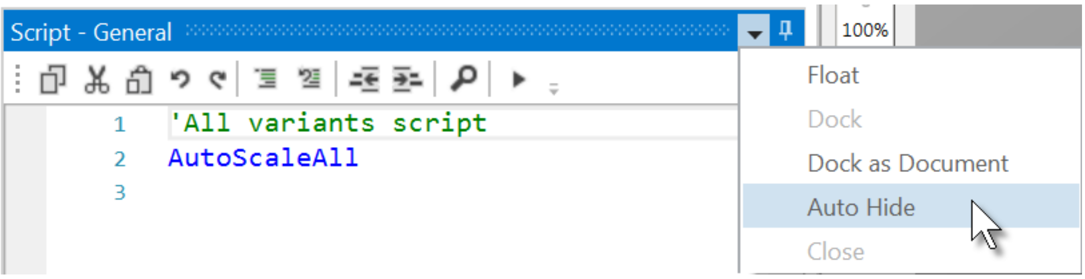
In this first project we will only look at the three first windows.

So we hide the Script- General window to increase the size of the two other windows on top. Click on ▾.



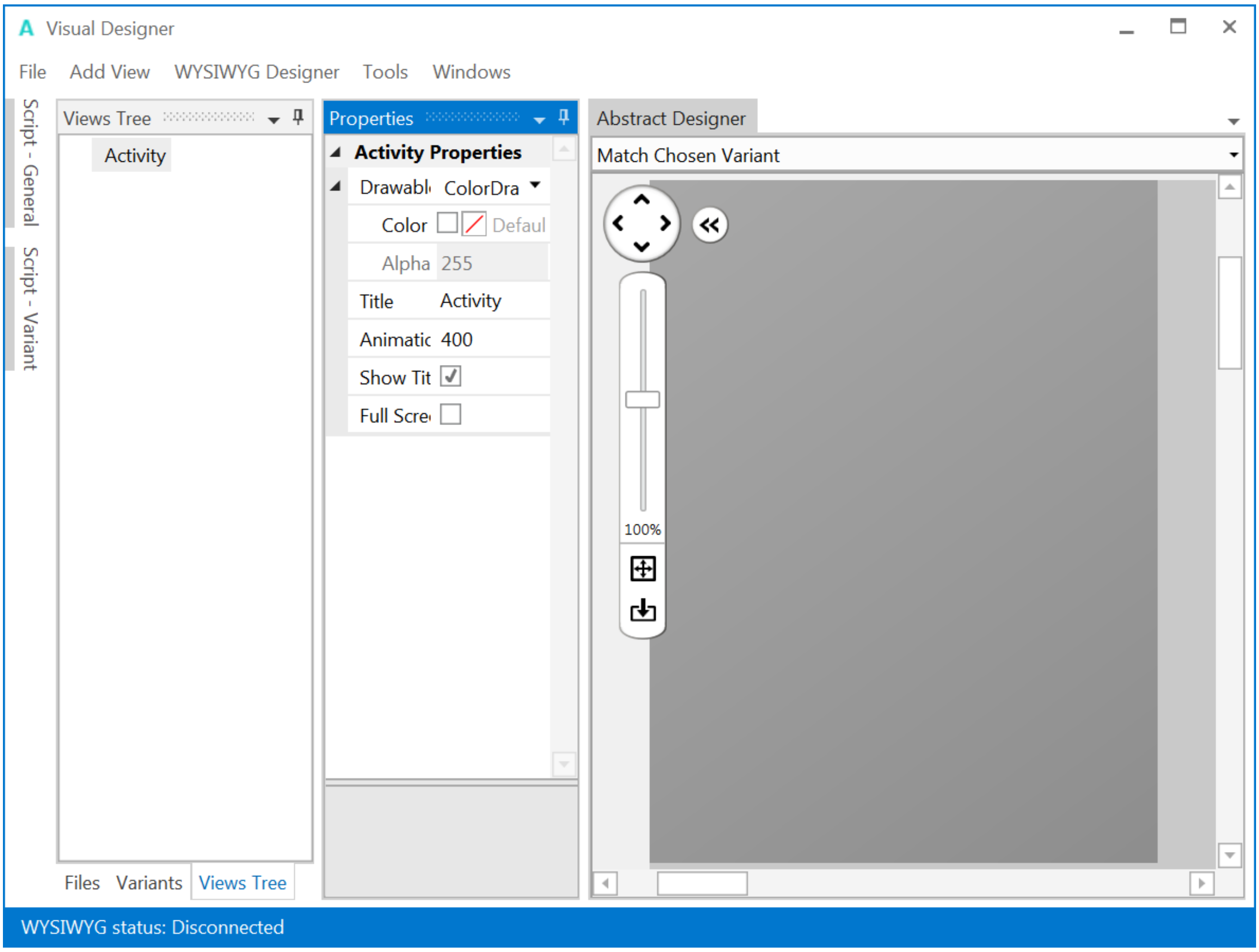
```
'All variants script
```

And on Auto Hide .



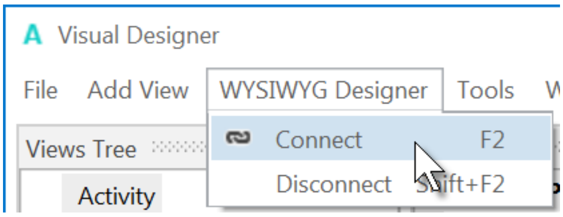
```
'All variants script
AutoScaleAll
```

The Designer will look like this.



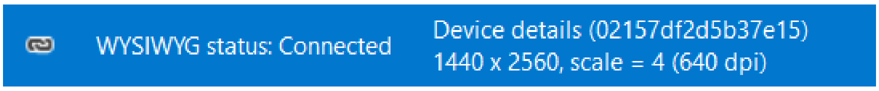
Activity Properties	
Drawabl	ColorDra
Color	□ Defaul
Alpha	255
Title	Activity
Animatic	400
Show Tit	☑
Full Scre	□

To shows the views on the device you must connect the device to the Designer.



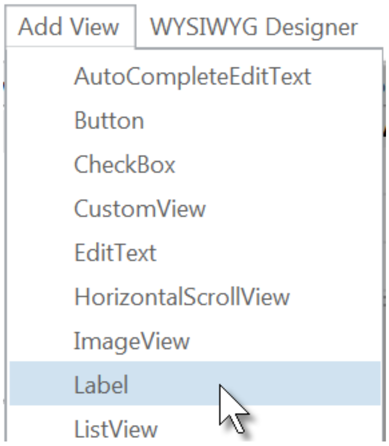
Wait until the Designer and the device are connected. This can take some time, so be patient.

You will see the state of the Designer here on the bottom of the Designer with the parameters of the connected device:


WYSIWYG status: Connected

Device details (02157df2d5b37e15)
1440 x 2560, scale = 4 (640 dpi)

Now we will add the 2 Labels for the numbers.
In the Designer, add a Label.

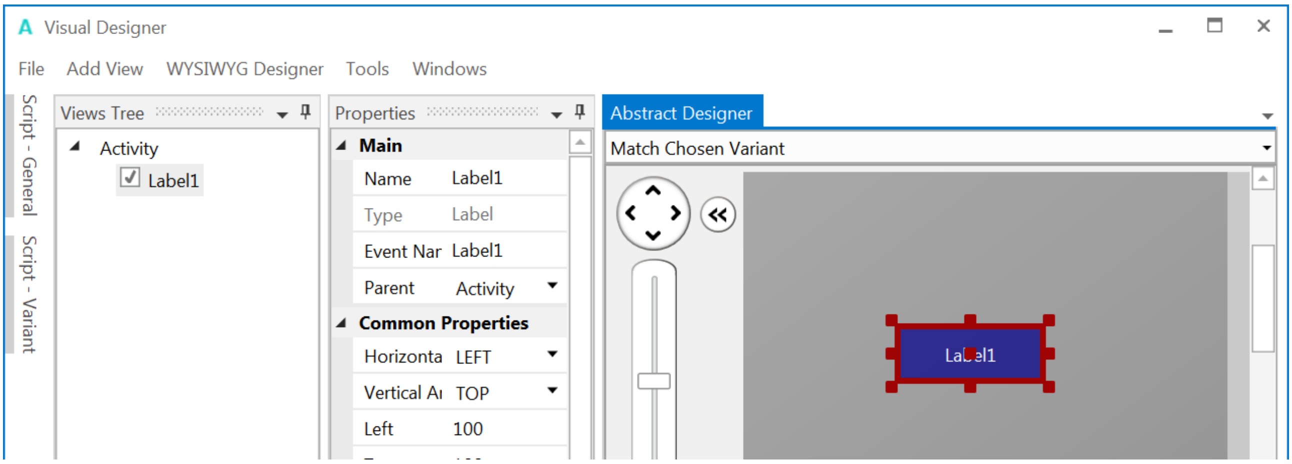


We see the Label with the default name `Label1` in following windows:

Views Tree

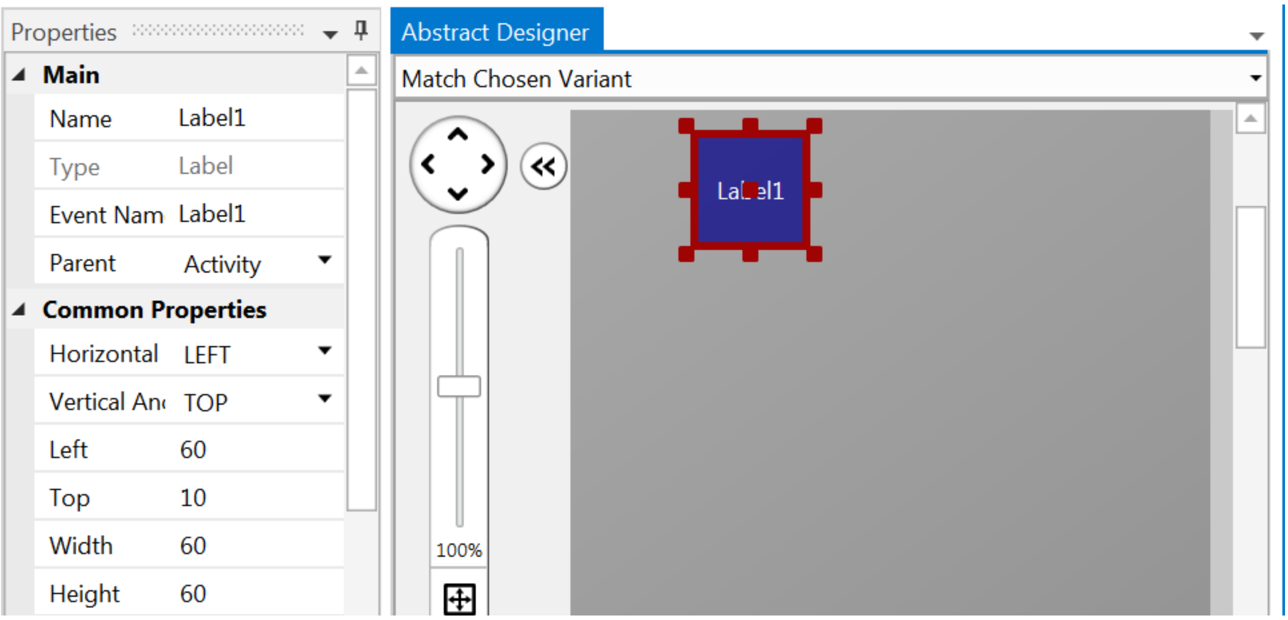
Properties
with its default properties.

Abstract Designer
at its default position and Default dimensions.



Main	
Name	Label1
Type	Label
Event Nar	Label1
Parent	Activity
Common Properties	
Horizonta	LEFT
Vertical Ai	TOP
Left	100

Resize and move the Label with the red squares like this.



Main

Name	Label1
Type	Label
Event Nam	Label1
Parent	Activity

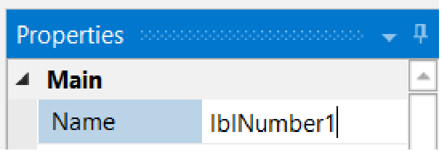
Common Properties

Horizontal	LEFT
Vertical An	TOP
Left	60
Top	10
Width	60
Height	60

The new properties Left, Top, Width and Height are directly updated in the Properties window.
You can also modify the Left, Top, Width and Height properties directly in the Properties window.

Let us change the properties of this first Label according to our requirements.

By default, the name is Label with a number, here Label1, let us change its name to lblNumber1.
The three letters 'lbl' at the beginning mean 'Label', and 'Number1' means the first number.
It is recommended to use meaningful names for views so we know directly what kind of view it is and its purpose.

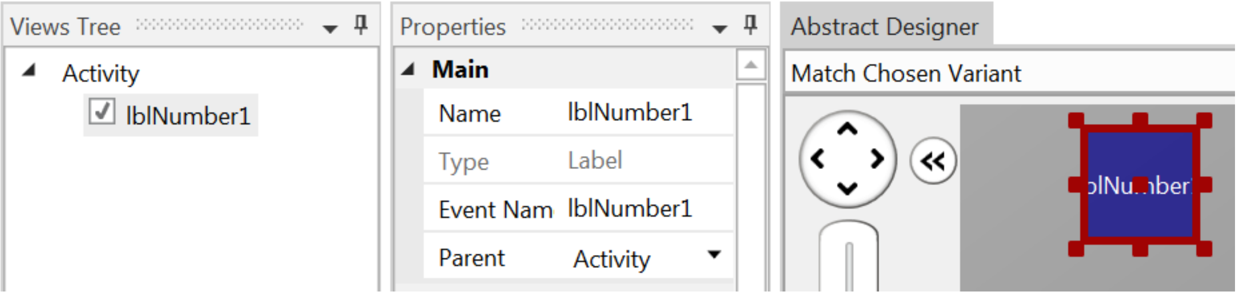


Main

Name	lblNumber1

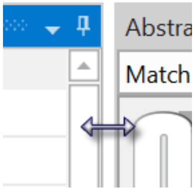
Pressing the 'Return' key or clicking elsewhere will update the name in the other windows and change the Event Name property.

Main:	Main module.
Name:	Name of the view.
Type:	Type of the view. In this case, Label, which is not editable.
Event Name:	Generic name of the routines that handle the events of the Label.
Parent:	Parent view the Label belongs to.



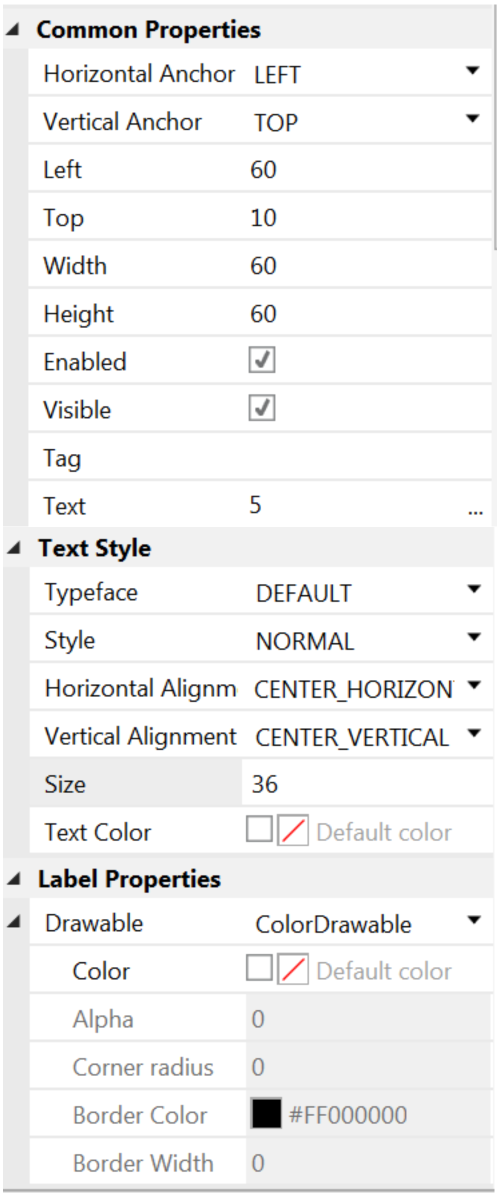
Main

Name	lblNumber1
Type	Label
Event Nam	lblNumber1
Parent	Activity



To better see the other properties we enlarge the Properties window.

Let us check and change the other properties:



Common Properties	
Horizontal Anchor	LEFT
Vertical Anchor	TOP
Left	60
Top	10
Width	60
Height	60
Enabled	☑
Visible	☑
Tag	
Text	5 ...
Text Style	
Typeface	DEFAULT
Style	NORMAL
Horizontal Alignm	CENTER_HORIZON
Vertical Alignment	CENTER_VERTICAL
Size	36
Text Color	□ Default color
Label Properties	
Drawable	ColorDrawable
Color	□ Default color
Alpha	0
Corner radius	0
Border Color	■ #FF000000
Border Width	0

Left, Top, Width and Height are OK.
Or if the values are not the same you should change them.

Enabled, Visible are OK

Tag, we leave empty.
Text, we set a default number, say 5

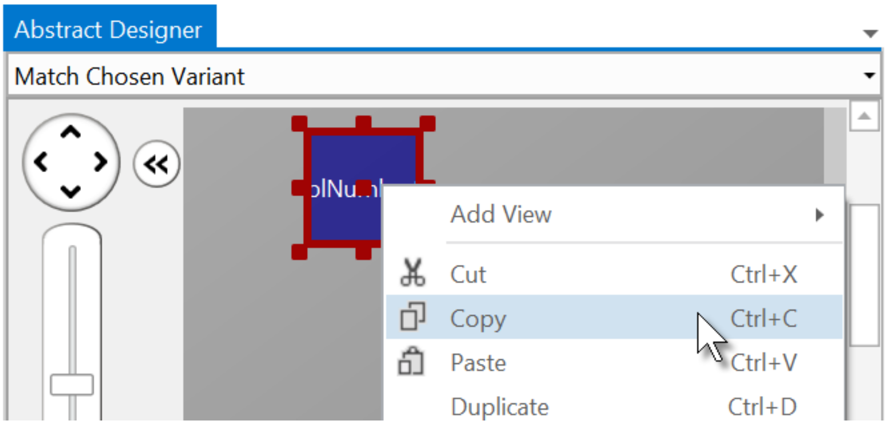
Typeface, Style are OK

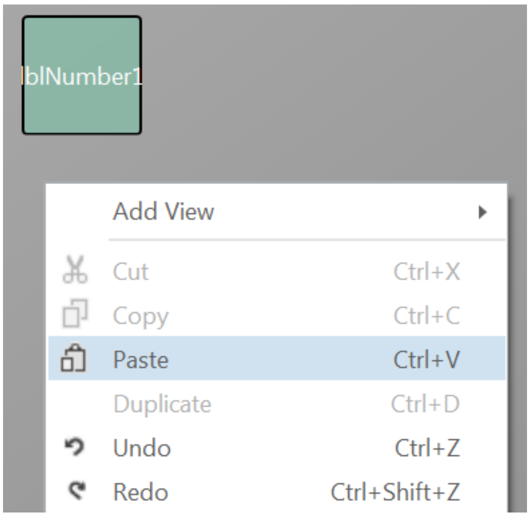
Horizontal Alignment, we set to CENTER_HORIZONTAL
Vertical Alignment, we leave CENTER_VERTICAL.
Size, we set to 36

We leave all the other properties as they are.

We need a second Label similar to the first one. Instead of adding a new one, we copy the first one with the same properties. Only the Name and Left properties will change.

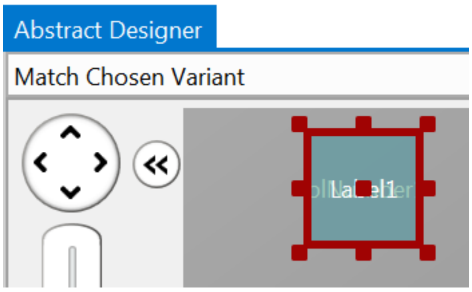
Right click in the Abstract Designer on lblNumber1 and click on Copy

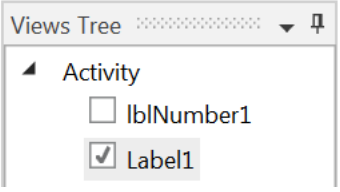




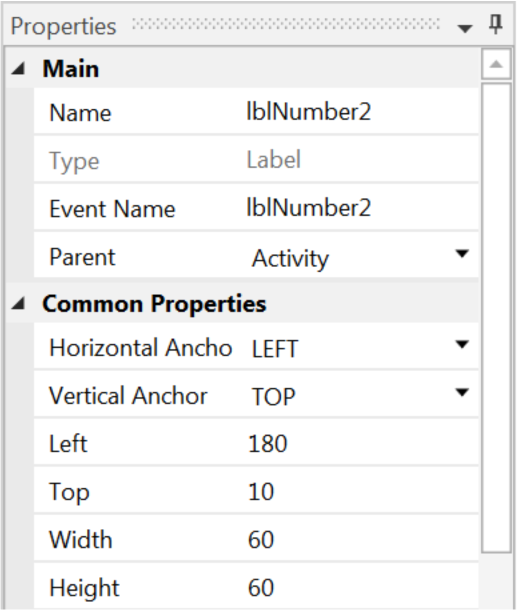
Click somewhere else in the Abstract Designer and right click again and click on Paste.

The new label covers the previous one.





We see the new label added in the Views Tree.



Properties

Main

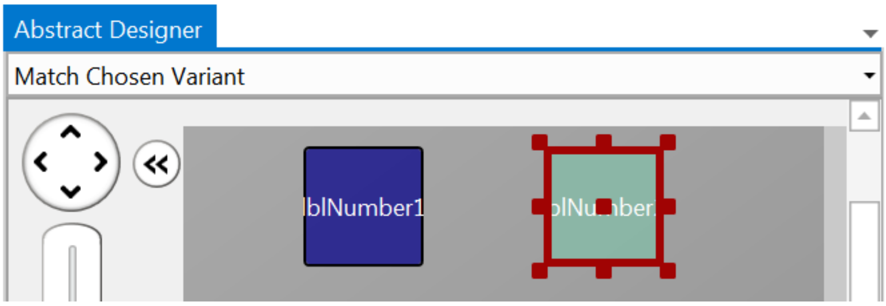
Name	lblNumber2
Type	Label
Event Name	lblNumber2
Parent	Activity

Common Properties

Horizontal Ancho	LEFT
Vertical Anchor	TOP
Left	180
Top	10
Width	60
Height	60

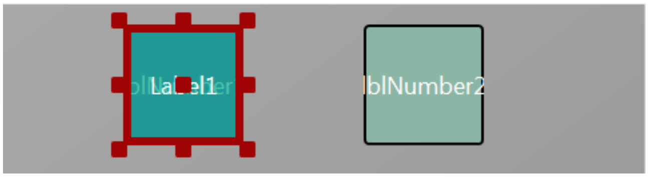
Change its name to lblNumber2.

Change the Left property to 180.

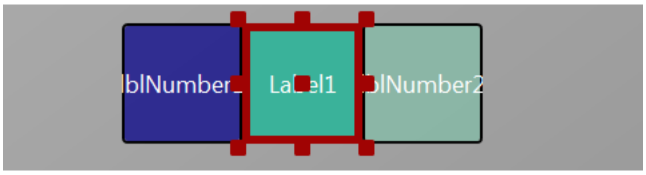


The new label with its new name and at its new position

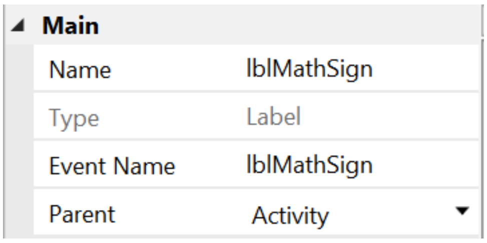
Now we add a 3rd Label for the math sign. We copy once again lblNumber1.
In the Abstract Designer right click on lblNumber1, click on Copy.
Click somewhere else, right click again and click on Paste.



The new label covers lblNumber1.

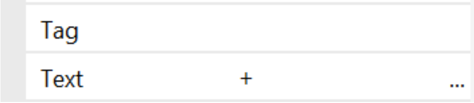


Position it between the first two Labels and change its name to lblMathSign and its Text property to '+'.



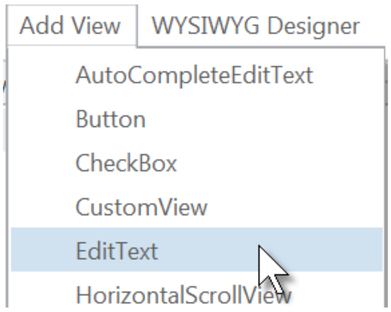
Main

Name	lblMathSign
Type	Label
Event Name	lblMathSign
Parent	Activity

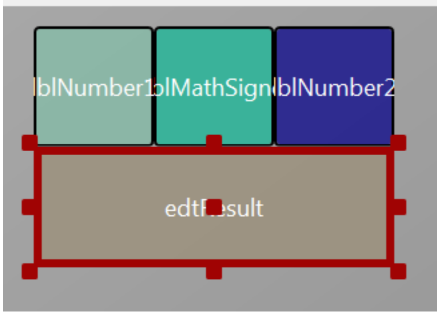


Tag	
Text	+

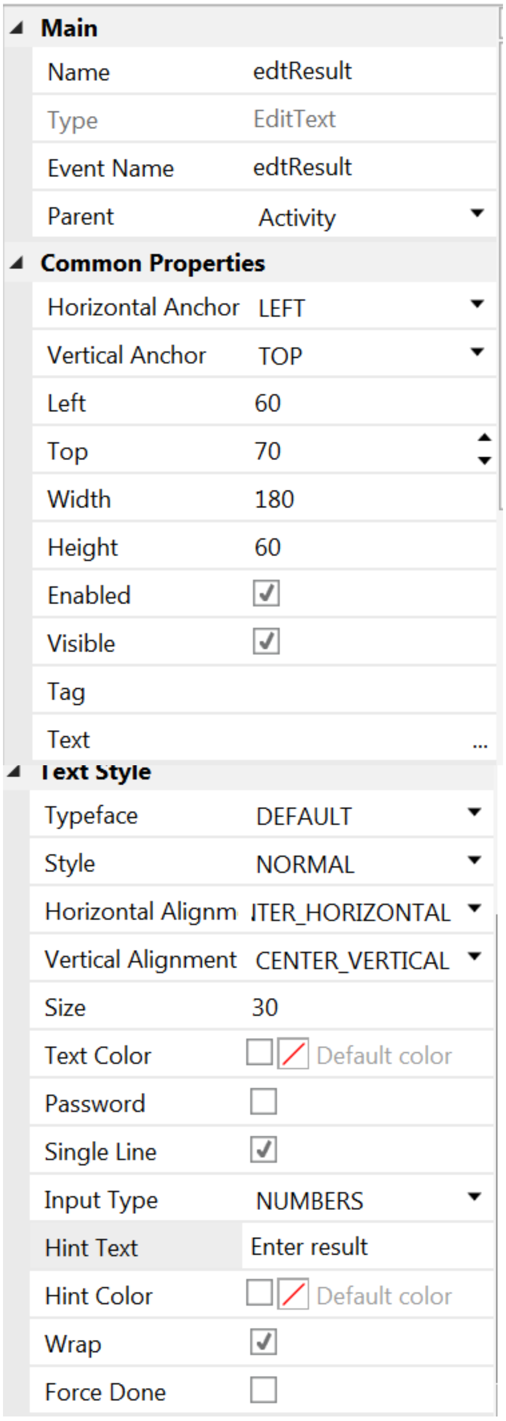
Now let us add an EditText view.



In the Designer Add View menu click on EditText .



Position it below the three Labels and change its name to `edtResult`. `'edt'` means EditText and `'Result'` for its purpose.



Main	
Name	edtResult
Type	EditText
Event Name	edtResult
Parent	Activity
Common Properties	
Horizontal Anchor	LEFT
Vertical Anchor	TOP
Left	60
Top	70
Width	180
Height	60
Enabled	☑
Visible	☑
Tag	
Text	...
Text Style	
Typeface	DEFAULT
Style	NORMAL
Horizontal Alignm	ITER_HORIZONTAL
Vertical Alignment	CENTER_VERTICAL
Size	30
Text Color	□ Default color
Password	□
Single Line	☑
Input Type	NUMBERS
Hint Text	Enter result
Hint Color	□ Default color
Wrap	☑
Force Done	□

Let us change these properties.
Name to edtResult

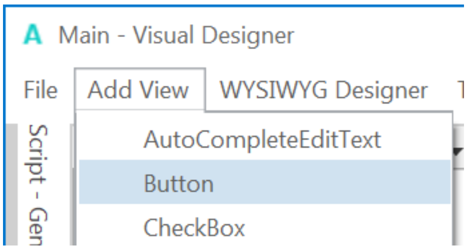
Horizontal Alignment to CENTER_HORIZONTAL

Text Size to 30

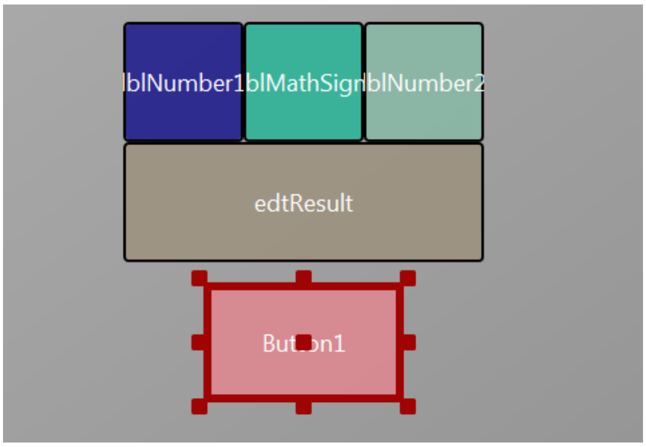
Input Type to NUMBERS
Hint Text to Enter result

Setting Input Type to NUMBERS lets the user enter only numbers.

Hint Text represents the text shown in the EditText view if no text is entered. After making these changes, you should see something like this.

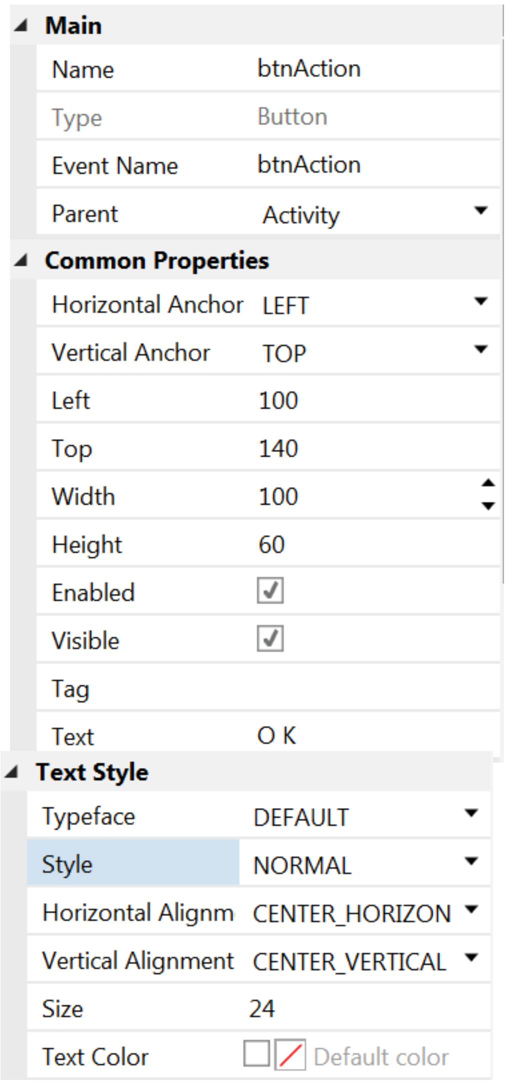


Now, let's add the Button which, when pressed, will either check the result the user supplied as an answer, or will generate a new math problem, depending on the user's input.



Position it below the EditText view. Resize it and change following properties:

Set the properties like below.



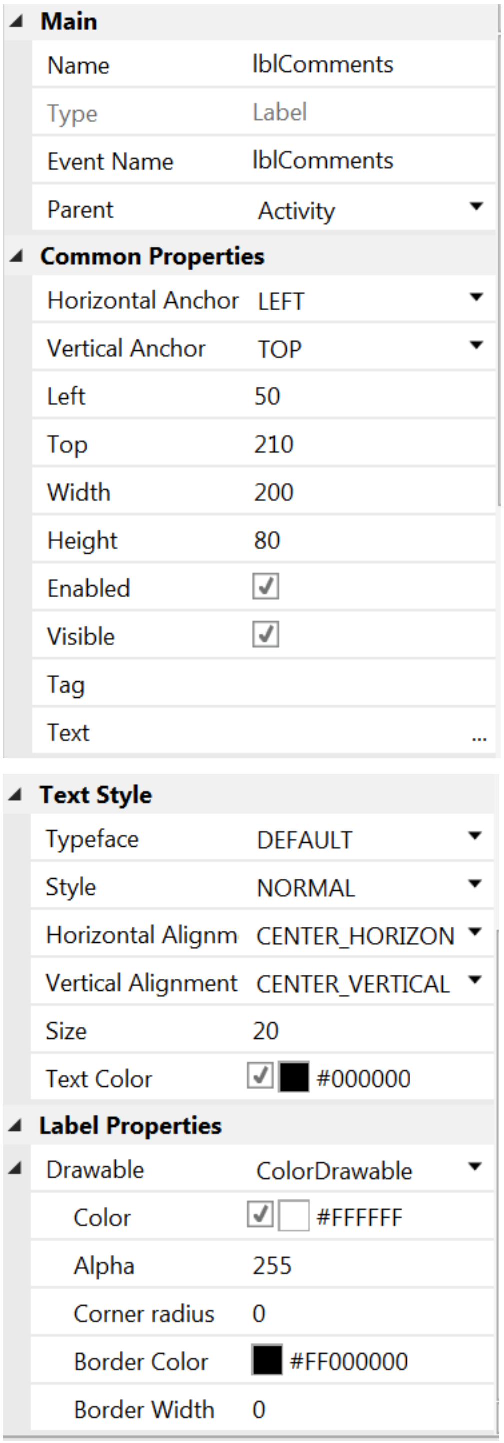
Main	
Name	btnAction
Type	Button
Event Name	btnAction
Parent	Activity
Common Properties	
Horizontal Anchor	LEFT
Vertical Anchor	TOP
Left	100
Top	140
Width	100
Height	60
Enabled	☑
Visible	☑
Tag	
Text	O K
Text Style	
Typeface	DEFAULT
Style	NORMAL
Horizontal Alignm	CENTER_HORIZON
Vertical Alignment	CENTER_VERTICAL
Size	24
Text Color	□ Default color

Name to btnAction

Text to O K (with a space between O and K)

Text Size to 24

Let us add the last Label for the comments. Position it below the Button and resize it.



Main	
Name	lblComments
Type	Label
Event Name	lblComments
Parent	Activity
Common Properties	
Horizontal Anchor	LEFT
Vertical Anchor	TOP
Left	50
Top	210
Width	200
Height	80
Enabled	☑
Visible	☑
Tag	
Text	...
Text Style	
Typeface	DEFAULT
Style	NORMAL
Horizontal Alignm	CENTER_HORIZON
Vertical Alignment	CENTER_VERTICAL
Size	20
Text Color	☑ ■ #000000
Label Properties	
Drawable	ColorDrawable
Color	☑ □ #FFFFFF
Alpha	255
Corner radius	0
Border Color	■ #FF000000
Border Width	0

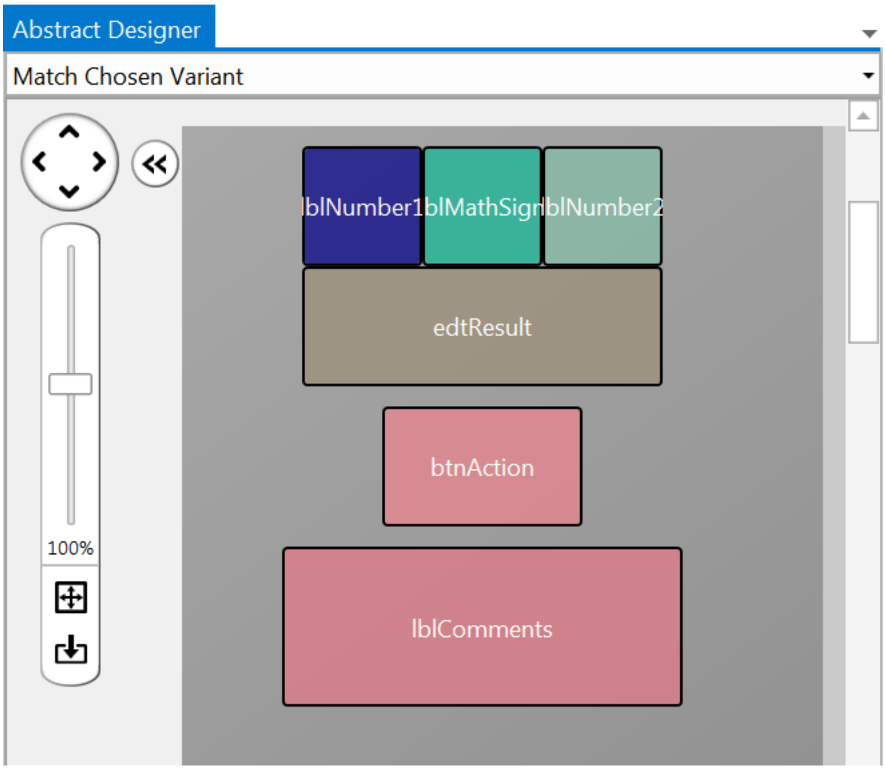
Change the following properties:
Name to lblComments

Horizontal Alignment CENTER_HORIZONTAL

Text Color to #000000
We set the Text Color property to Black (#000000).

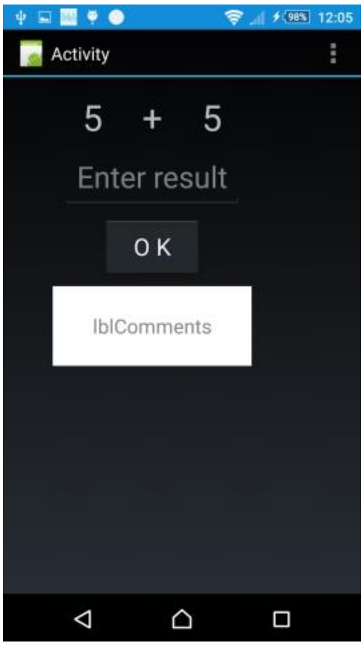
Color to #FFFFFF
Alpha to 255

By default, the Label background color is black and transparent.
We set it to white and opaque Alpha = 255.

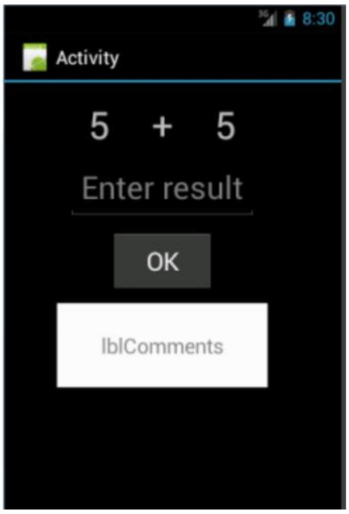


The result will look like this in the Designer.

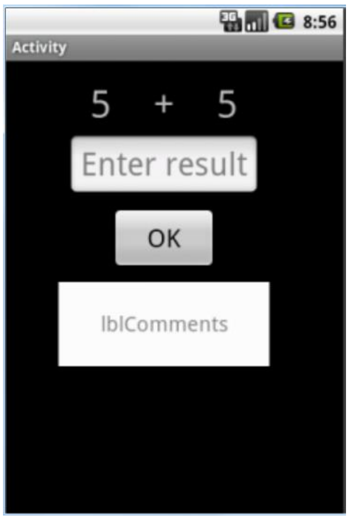
And on a device or Emulator.



Sony xperia z1

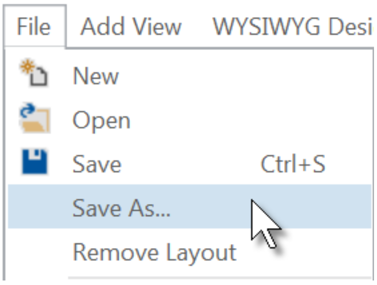


Android 4.2 Emulator

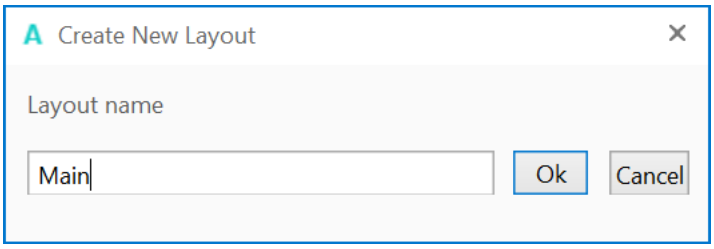


Android 2.2 Emulator

Let us save the layout in a file.



In the File menu click on Save As... and save it with the name 'Main'.



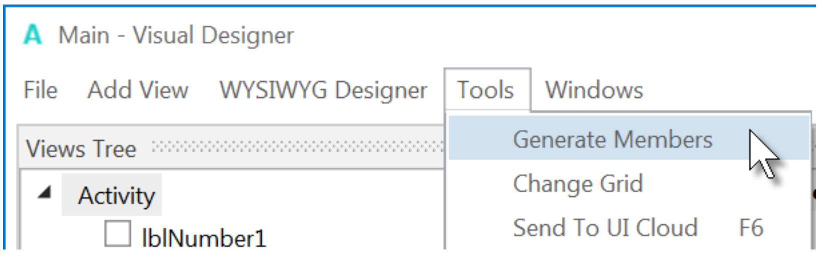
Layout name

Main

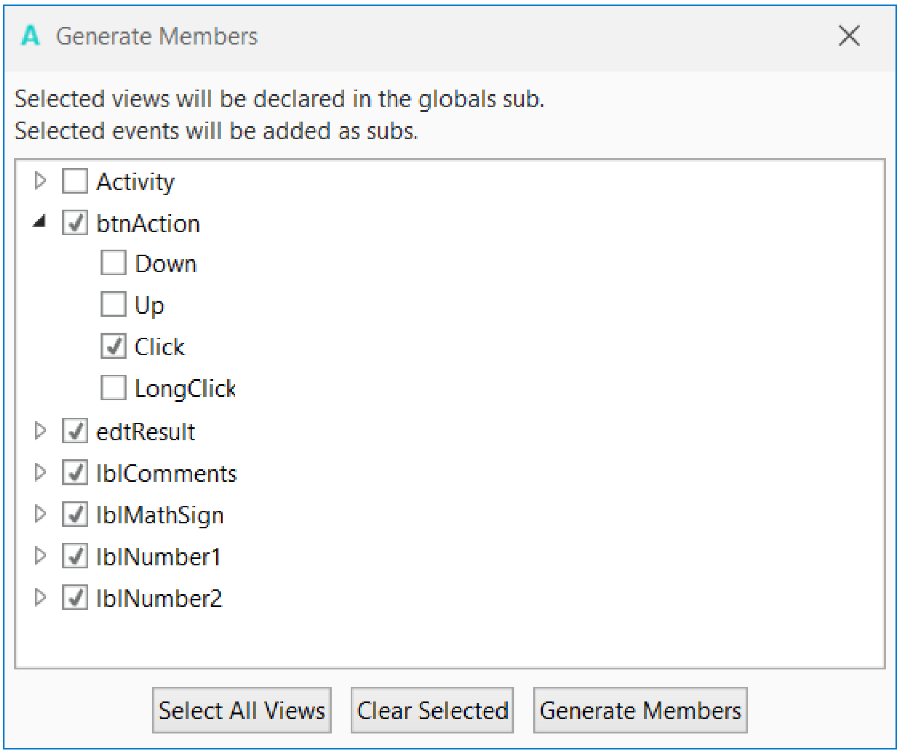
Click on Ok .

To write the routines for the project, we need to reference the Views in the code.
This can be done with the *Generate Members* tool in the Designer.

The *Generate Members* tool automatically generates references and subroutine frames.



In the Tools menu click on Generate Members to open the generator.



Selected views will be declared in the globals sub.
Selected events will be added as subs.

- ▷ □ Activity
- ◢ ☑ btnAction
  - □ Down
  - □ Up
  - ☑ Click
  - □ LongClick
- ▷ ☑ edtResult
- ▷ ☑ lblComments
- ▷ ☑ lblMathSign
- ▷ ☑ lblNumber1
- ▷ ☑ lblNumber2

Here we find all the views added to the current layout.
We check all views and check the Click event for the btnAction Button.
Checking a view ▷ ☑ edtResult generates its reference in the Globals Sub routine in the code.
This is needed to make the view recognized by the system and allow the autocomplete function.

```
  Private btnAction As Button
  Private edtResult As EditText
  Private lblComments As Label
  Private lblMathSign As Label
  Private lblNumber1 As Label
  Private lblNumber2 As Label
```

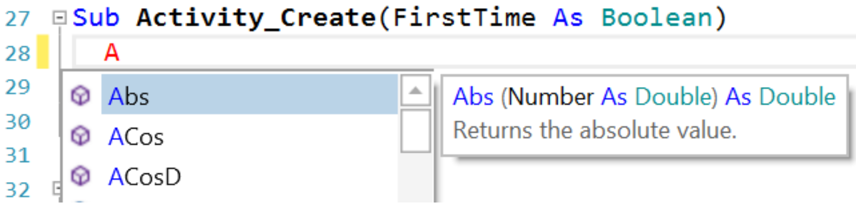
Clicking on an event of a view ☑ Click generates the Sub frame for this event.

```
Sub btnAction_Click

End Sub
```

Click on Generate Members to generate the references and Sub frames.

Now we go back to the IDE to enter the code.
First, we need our Activity to load our layout file. Within the “Activity_Create” sub, do the following. You can remove the lines in green.
We will enter this line of code `Activity.LoadLayout(“Main”)`.

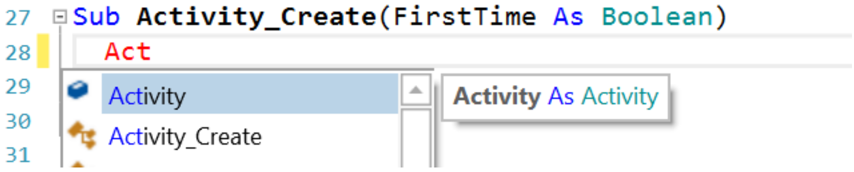
- Enter 'A’, as soon as you begin typing the autocomplete function shows you all keywords beginning with ‘a’.



```
Sub Activity_Create(FirstTime As Boolean)
    A
```

Abs (Number As Double) As Double
Returns the absolute value.

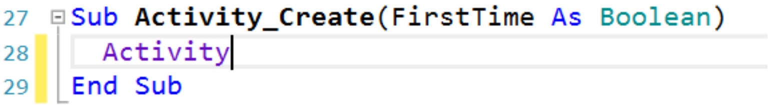
- Continue typing ‘Act’.



```
Sub Activity_Create(FirstTime As Boolean)
    Act
```

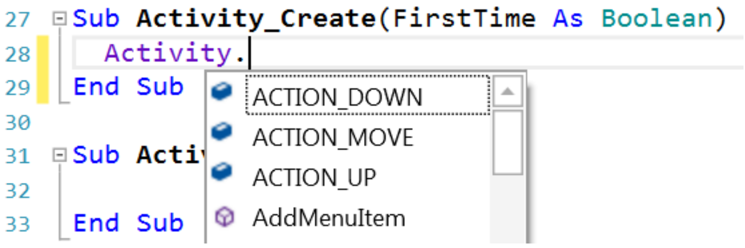
Activity As Activity

- Press ‘Return’ or click on Activity.



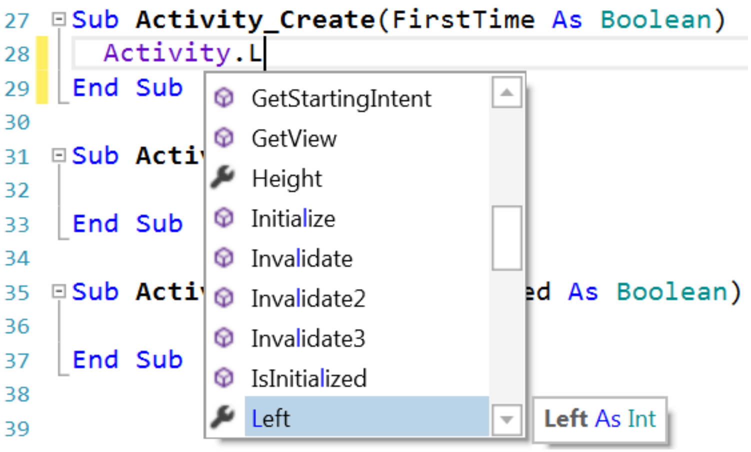
```
Sub Activity_Create(FirstTime As Boolean)
    Activity
End Sub
```

- We have the word Activity, now enter a dot.



```
Sub Activity_Create(FirstTime As Boolean)
    Activity.
End Sub

Sub Acti

End Sub
```

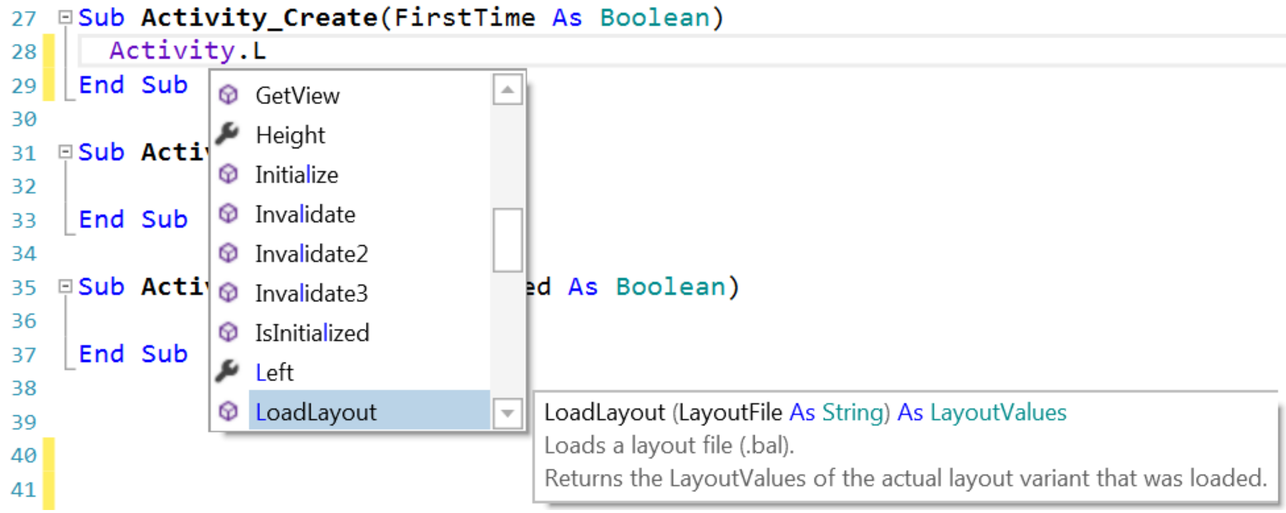
- The autocomplete function shows all the possible properties of the view.
- Enter 'L' , and the autocomplete function shows the properties beginning with 'L'



```
Sub Activity_Create(FirstTime As Boolean)
    Activity.L
End Sub

Sub Acti

End Sub

Sub Acti                                    ed As Boolean)

End Sub
```

Left As Int

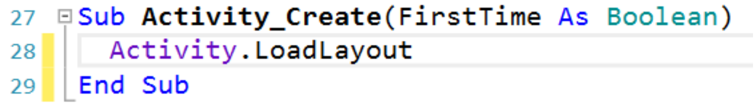
- Press the down arrow key, and LoadLayout will be highlighted with the online help for the given property or method.



```
Sub Activity_Create(FirstTime As Boolean)
  Activity.L
End Sub

Sub Acti

End Sub

Sub Acti                                    ed As Boolean)

End Sub
```

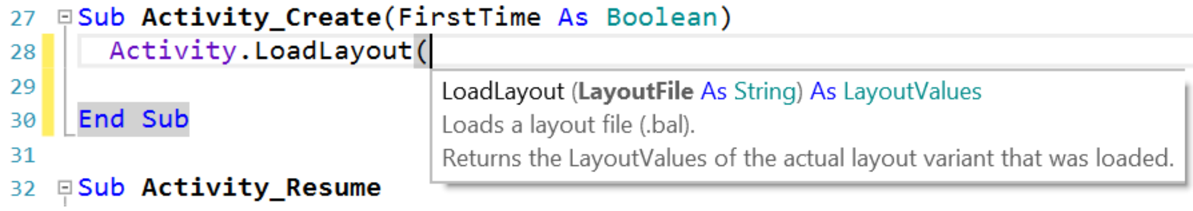
LoadLayout (LayoutFile As String) As LayoutValues
Loads a layout file (.bal).
Returns the LayoutValues of the actual layout variant that was loaded.

- Press 'Return' to add LoadLayout.



```
Sub Activity_Create(FirstTime As Boolean)
  Activity.LoadLayout
End Sub
```

- Press '(' to display the online help showing the needed properties for the method.



```
Sub Activity_Create(FirstTime As Boolean)
  Activity.LoadLayout(

End Sub

Sub Activity_Resume
```

LoadLayout (**LayoutFile** As String) As LayoutValues
Loads a layout file (.bal).
Returns the LayoutValues of the actual layout variant that was loaded.

- Enter "Main")

```
Sub Activity_Create(FirstTime As Boolean)
   Activity.LoadLayout("Main")
End Sub
```

We want to generate a new problem as soon as the program starts. Therefore, we add a call to the New subroutine.

```
Sub Activity_Create(FirstTime As Boolean)
   Activity.LoadLayout("Main")
   New
End Sub
```

Generating a new problem means generating two new random values between 1 and 9 (inclusive) for Number1 and Number2, then showing the values using the lblNumber1 and lblNumber2 ‘Text’ properties.

To do this we enter following code:
In Sub Globals we add two variables for the two numbers.

```
   Public Number1, Number2 As Int
End Sub
```

And the 'New' Subroutine:

```
Sub New
   Number1 = Rnd(1, 10)                 ' Generates a random number between 1 and 9
   Number2 = Rnd(1, 10)                 ' Generates a random number between 1 and 9
   lblNumber1.Text = Number1            ' Displays Number1 in label lblNumber1
   lblNumber2.Text = Number2            ' Displays Number2 in label lblNumber2
   lblComments.Text = "Enter the result" & CRLF & "and click on OK"
   edtResult.Text = ""                  ' Sets edtResult.Text to empty
End Sub
```

The following line of code generates a random number from '1' (inclusive) to '10' (exclusive):
`Rnd(1, 10)`
The following line displays the comment in the `lblComments` view:
`lblComments.Text = "Enter the result" & CRLF & "and click on OK"`
`CRLF` is the LineFeed character.

Now we add the code for the Button click event.

We have two cases:
- When the Button text is equal to "O K" (with a space between O and K), it means that a new problem is displayed, and the program is waiting for the user to enter a result and press the Button.
- When the Button text is equal to "NEW", it means that the user has entered a correct answer and when the user clicks on the Button a new problem will be generated.

```
Sub btnAction_Click
   If btnAction.Text = "O K" Then
      If edtResult.Text = "" Then
         Msgbox("No result entered","E R R O R")
      Else
         CheckResult
      End If
   Else
      New
      btnAction.Text = "O K"
   End If
End Sub
```

`If btnAction.Text = "O K" Then` checks if the Button text equals "O K"
If yes then we check if the EditText is empty.
If yes, we display a MessageBox telling the user that there is no result in the EditText view.
If no, we check if the result is correct or if it is false.
If no then we generate a new problem, set the Button text to "O K" and clear the EditText view.

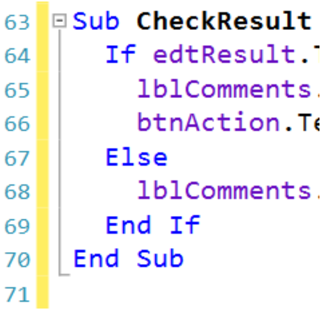
The last routine checks the result.

```
Sub CheckResult
   If edtResult.Text = Number1 + Number2 Then
      lblComments.Text = "G O O D  result" & CRLF & "Click on NEW"
      btnAction.Text = "N E W"
   Else
      lblComments.Text = "W R O N G  result" & CRLF & "Enter a new result" & CRLF & "and click OK"
   End If
End Sub
```

With `If edtResult.Text = Number1 + Number2 Then` we check if the entered result is correct.

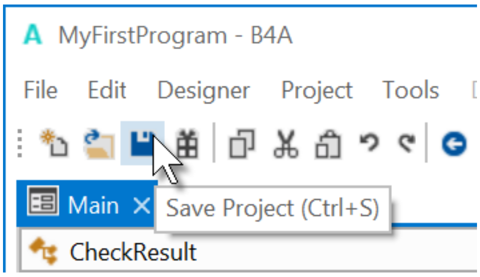
If yes, we display in the lblComments label the text below:
'G O O D result'
'Click on NEW'
and we change the Button text to "N E W ".
If no, we display in the lblComments label the text below:
W R O N G result
Enter a new result
and click OK

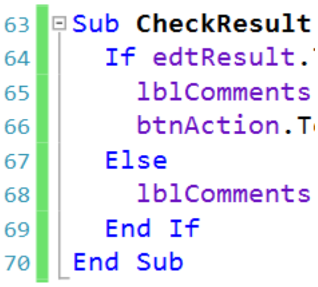
On the left side of the editor you see a yellow line.
This means that the code was modified.



```
Sub CheckResult
   If edtResult.
      lblComments.
      btnAction.Te
   Else
      lblComments.
   End If
End Sub
```

If we click on 💾 to save the project the yellow line becomes green showing a modified code but already saved. You can also press Ctrl + S to save the project.



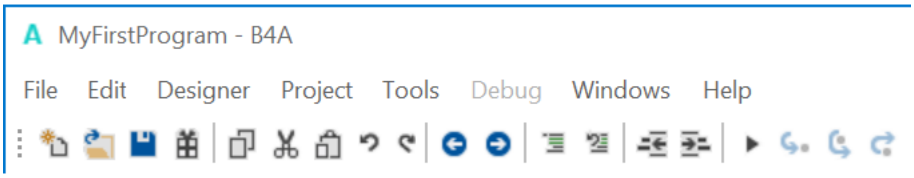


```
Sub CheckResult
   If edtResult.
      lblComments
      btnAction.T
   Else
      lblComments
   End If
End Sub
```

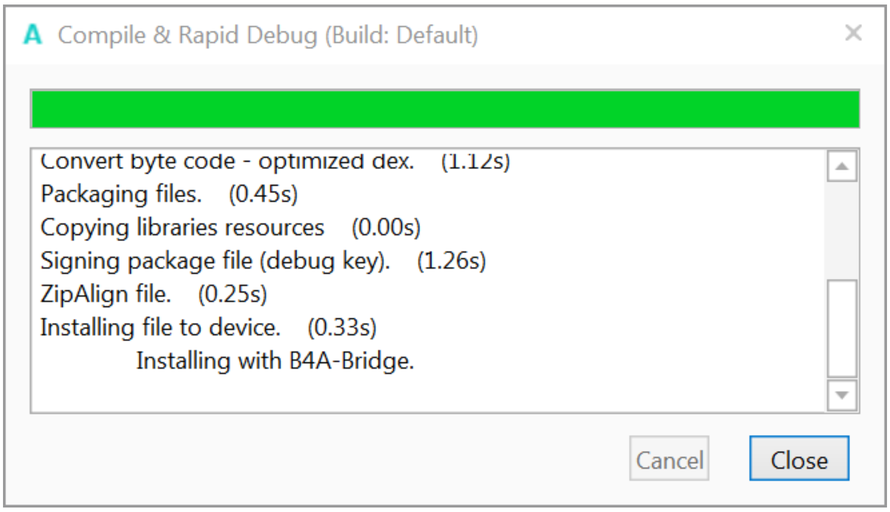
If we leave the IDE and load it again the green line disappears.

Let us now compile the program and transfer it to the Device.

In the IDE on top click on ▶ :



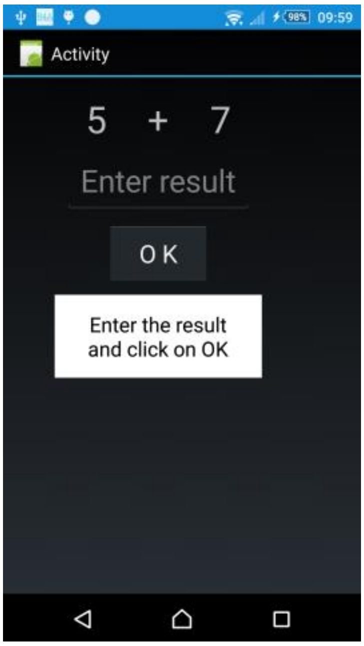
The program is going to be compiled.



```
Convert byte code - optimized dex.    (1.12s)
Packaging files.    (0.45s)
Copying libraries resources    (0.00s)
Signing package file (debug key).    (1.26s)
ZipAlign file.    (0.25s)
Installing file to device.    (0.33s)
            Installing with B4A-Bridge.
```

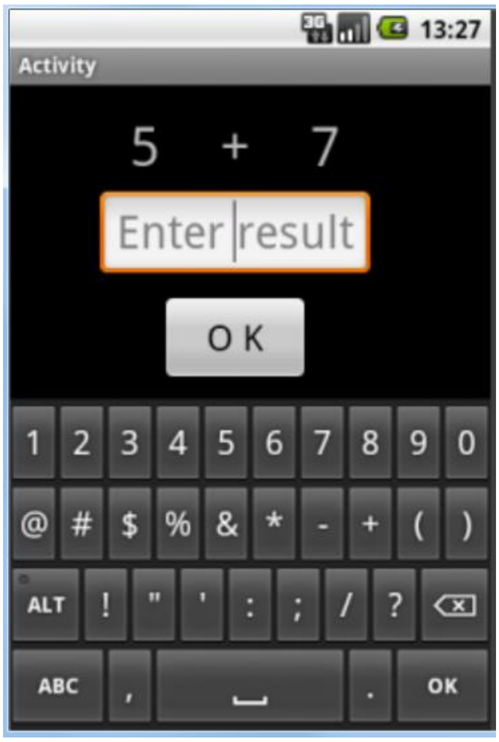
When the Close button becomes enabled as in message box, above, the compiling and transfer is finished.

Looking at the device, you should see something similar to the image below, with different numbers.



The screenshot may look different depending on the device and the Android version.

Of course, we could make aesthetic improvements in the layout, but this was not the main issue for the first program.



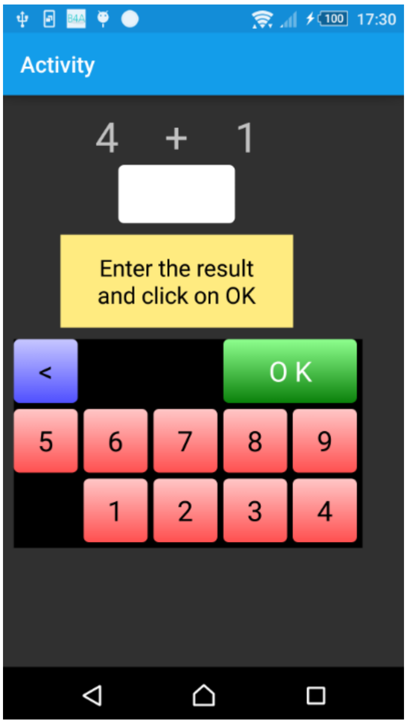
On a real device, you need to use the virtual keyboard.
Click on the EditText view to show the keyboard.

On some devices the current layout has the disadvantage that the comment label is covered by the virtual keyboard.

This will be improved in the next chapter,
'Second program', where we create our own keyboard.

3 Second program (SecondProgram.b4a)

The project is available in the SourceCode folder:
SourceCode\SecondProgram\SecondProgram.b4a



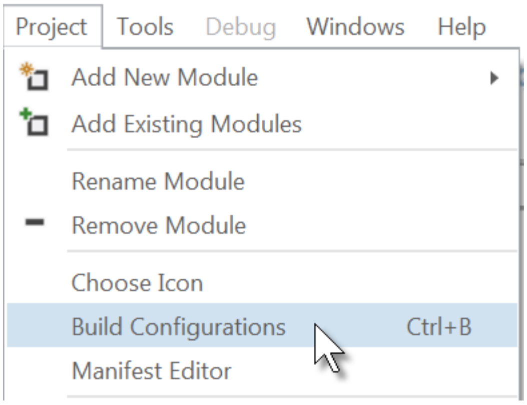
Improvements to “ My first program”.

We will add a numeric keyboard to the layout to avoid the use of the virtual keyboard.

Create a new folder called “SecondProgram”. Copy all the files and folders from MyFirstProgram to the new SecondProgram folder and rename the program files MyFirstProgram.b4a to SecondProgram.b4a and MyFirstProgram.b4a.meta to SecondProgram.b4a.meta.

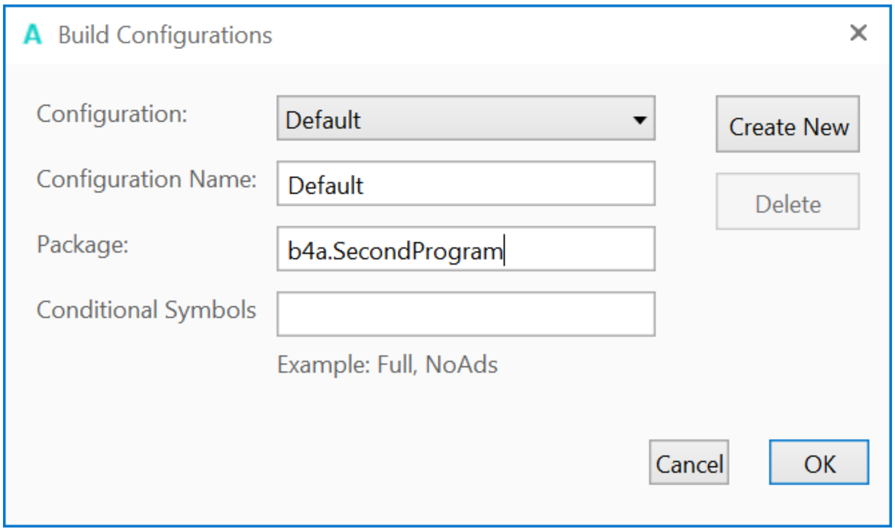
Load this new program in the IDE.

We need to change the Package Name.



In the IDE Project menu.

Click on Build Configurations



Configuration: Default

Configuration Name: Default

Package: b4a.SecondProgram

Conditional Symbols

Example: Full, NoAds

Change the Package name to

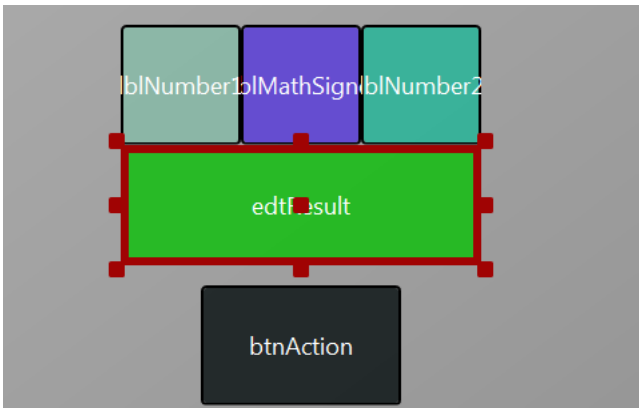
b4a.SecondProgram.

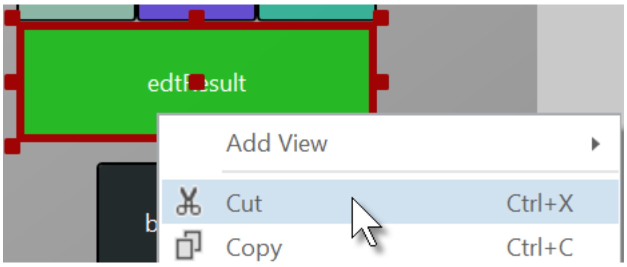
Click on OK.

Then we must change the ApplicationLabel on the very top of the code.

```
#Region  Project Attributes
   #ApplicationLabel: SecondProgram
```

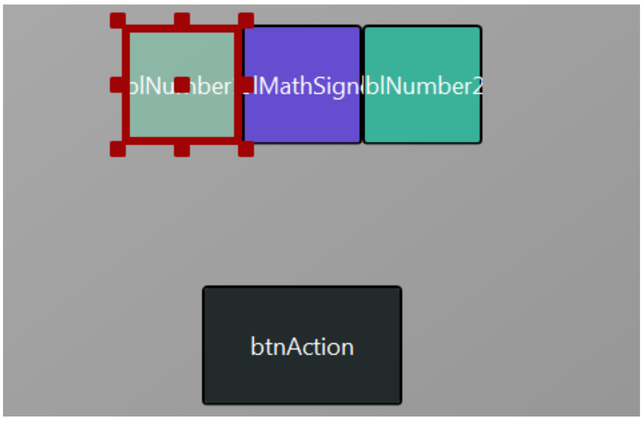
We want to replace the edtResult EditText view by a new Label.
Run the Visual Designer. If you want you can already connect the device or an Emulator.
In the Abstract Designer, click on the edtResult view.

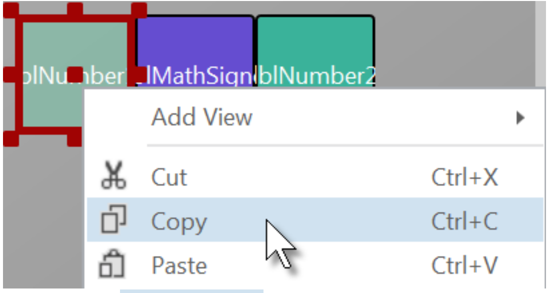




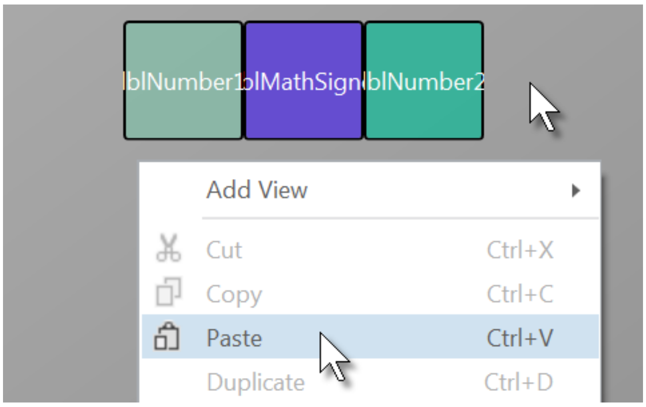
Right click on edtResult and click on Cut.

Right click on lblNumber1 to select it.



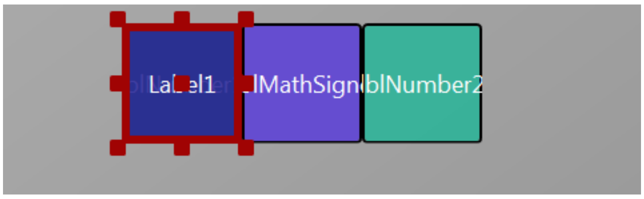


Click on Copy.

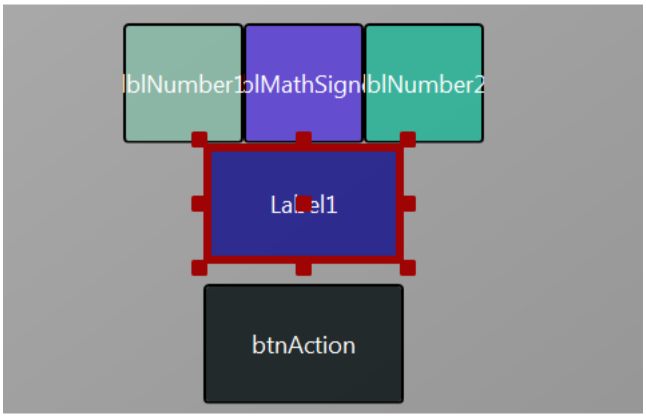


Right click somewhere else outsides a View.

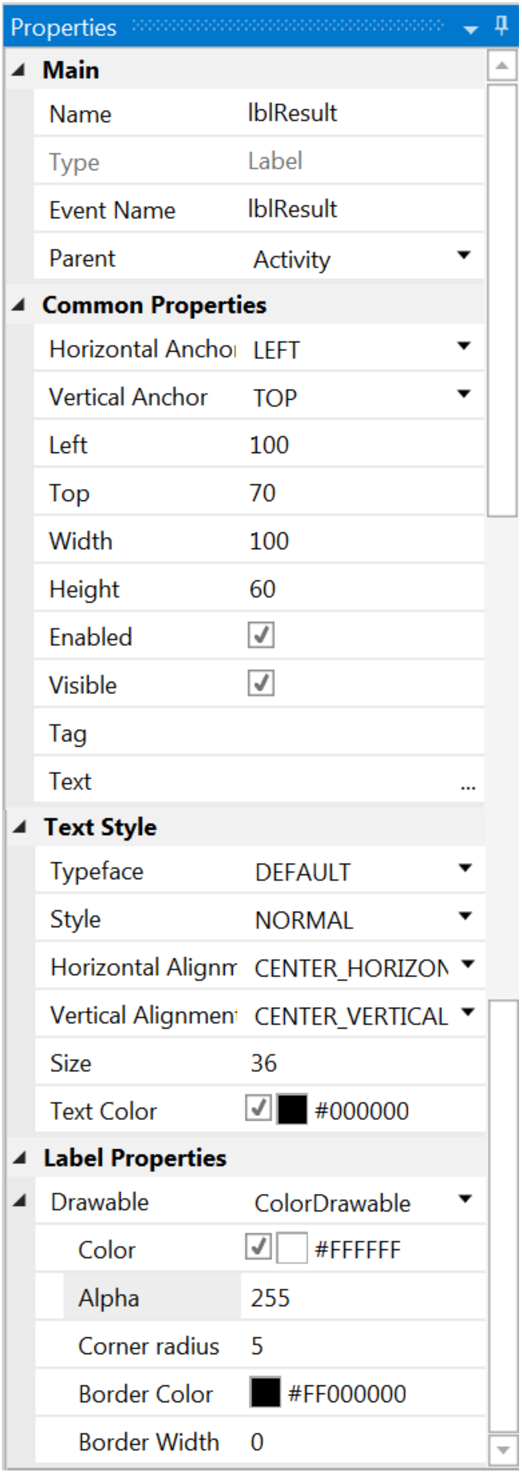
And click on Paste.



The new label covers lblNumber1.



Move it between the upper labels and the button and resize it.



Main	
Name	lblResult
Type	Label
Event Name	lblResult
Parent	Activity
Common Properties	
Horizontal Anchor	LEFT
Vertical Anchor	TOP
Left	100
Top	70
Width	100
Height	60
Enabled	☑
Visible	☑
Tag	
Text	...
Text Style	
Typeface	DEFAULT
Style	NORMAL
Horizontal Alignm	CENTER_HORIZON
Vertical Alignmen	CENTER_VERTICAL
Size	36
Text Color	☑ ■ #000000
Label Properties	
Drawable	ColorDrawable
Color	☑ □ #FFFFFF
Alpha	255
Corner radius	5
Border Color	■ #FF000000
Border Width	0

Modify the following properties:

Name to lblResult

Change the Left, Top, Width and Height properties if they are not the same as in the image.

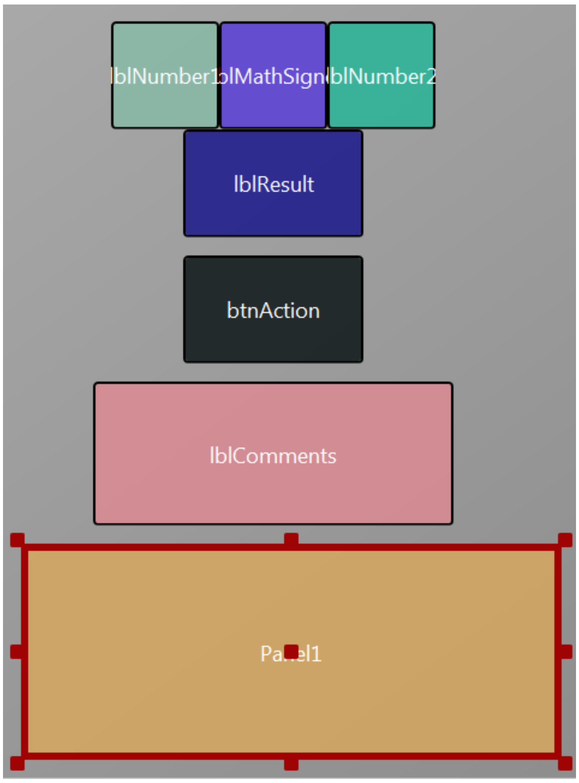
Text to " " blank character

Text Color to Black #000000

Color to White #FFFFFF

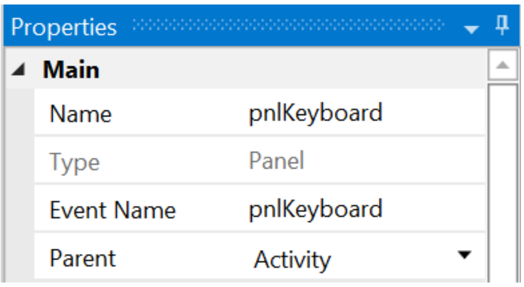
Alpha to 255

Corner Radius to 5



Now we add a Panel for the keyboard buttons.

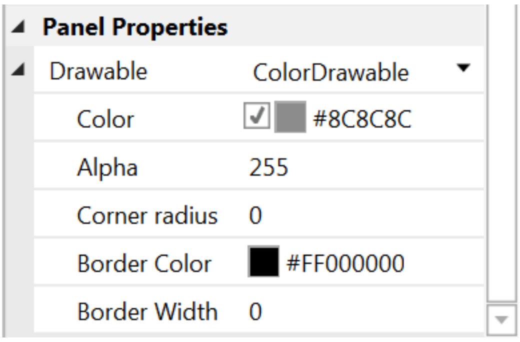
Position and resize it as in the image.



Main

Name	pnlKeyboard
Type	Panel
Event Name	pnlKeyboard
Parent	Activity

Change its Name to pnlKeyboard
"pnl" for Panel, the view type.

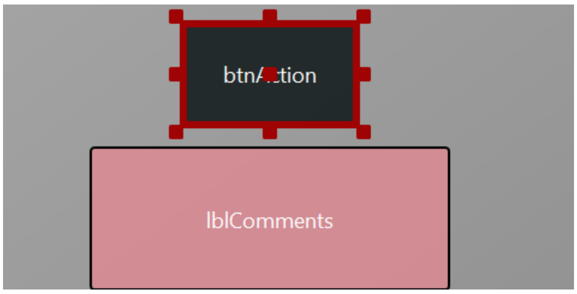


Panel Properties

Drawable	ColorDrawable
Color	☑ #8C8C8C
Alpha	255
Corner radius	0
Border Color	#FF000000
Border Width	0

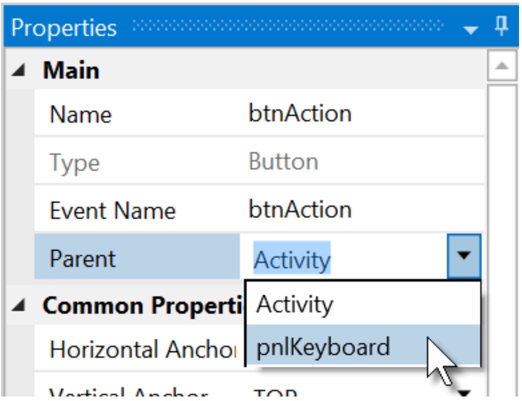
Change

Color	to #8C8C8C
Corner radius	to 0



We will move the btnAction button from the Activity to the pnlKeyboard Panel.

Click on btnAction.



Properties

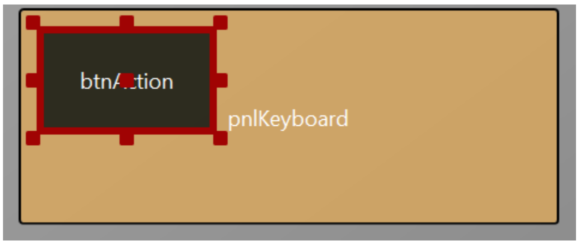
Main

Name	btnAction
Type	Button
Event Name	btnAction
Parent	Activity

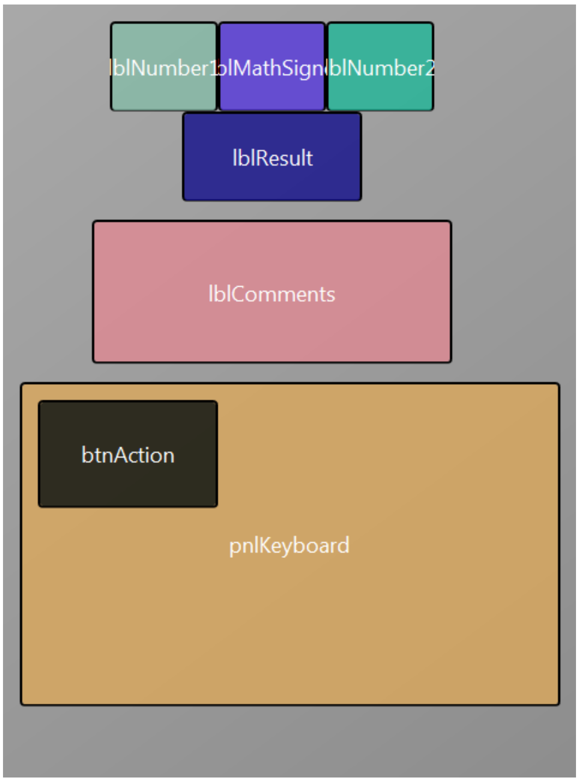
Activity
pnlKeyboard

Common Properti
Horizontal Ancho
Vertical Anchor TOP

and in the Parent list click on pnlKeyboard.

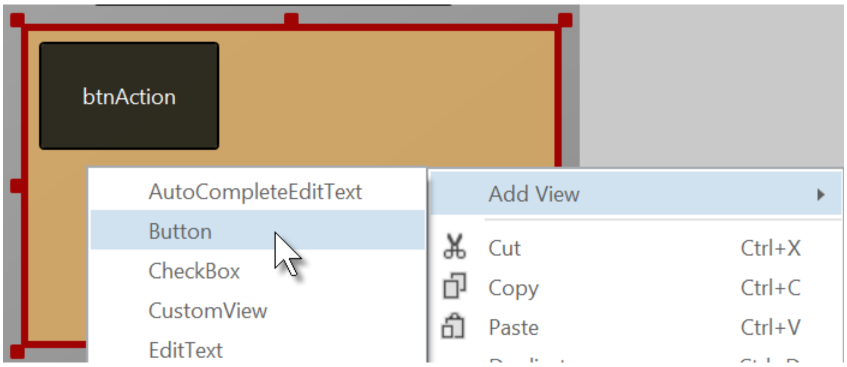


The button now belongs to the Panel.



Now we rearrange the views to get some more space for the keyboard.

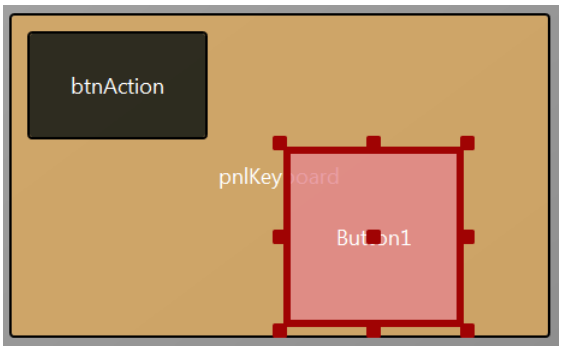
Set the Height property of the 4 Labels to 50 instead of 60.
Set the Top property of label lblResult to 60.
Set the Top property of label lblComments to 120.
Set the Top property of panel pnlKeyboard to 210.
Set the Height property of panel pnlKeyboard to 180.



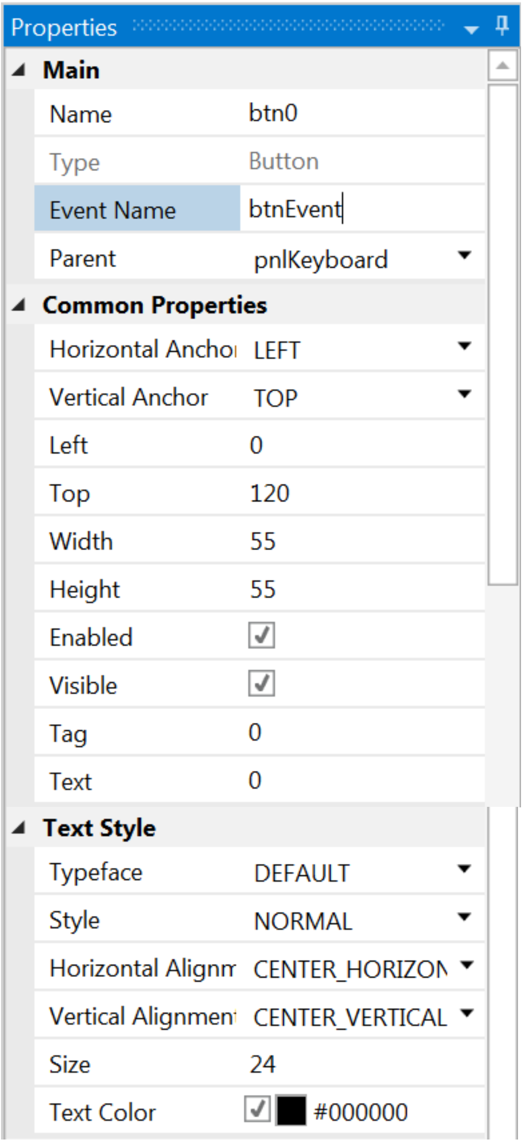
Right click on the pnlKeyboard and click on Add View

And click on Button.

to add a new button.



The new button is added.

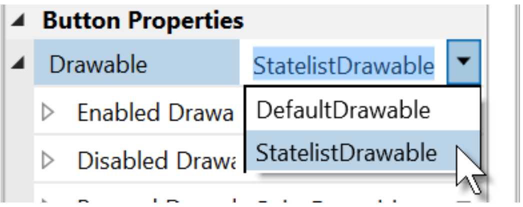


Properties

Main	
Name	btn0
Type	Button
Event Name	btnEvent
Parent	pnlKeyboard
Common Properties	
Horizontal Ancho	LEFT
Vertical Anchor	TOP
Left	0
Top	120
Width	55
Height	55
Enabled	☑
Visible	☑
Tag	0
Text	0
Text Style	
Typeface	DEFAULT
Style	NORMAL
Horizontal Alignm	CENTER_HORIZON
Vertical Alignmen	CENTER_VERTICAL
Size	24
Text Color	☑ ■ #000000

Change the following properties:

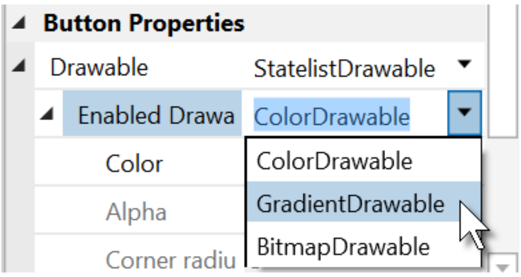
Name	to btn0
Event name	to btnEvent
Left	to 0
Top	to 120
Width	to 55
Height	to 55
Tag	to 0
Text	to 0
Size	to 24
TextColor	to Black #000000



Button Properties	
Drawable	StatelistDrawable
Enabled Drawa	DefaultDrawable
Disabled Draw	StatelistDrawable

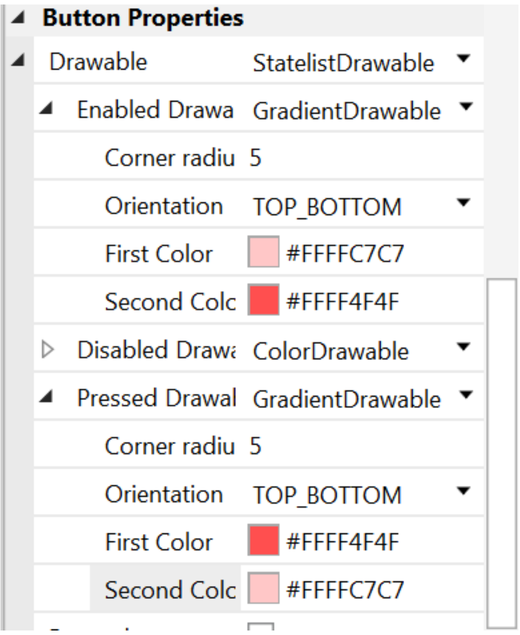
Now we want to change the button colors.

Click on StatelistDrawable.



Button Properties	
Drawable	StatelistDrawable
Enabled Drawa	ColorDrawable
Color	ColorDrawable
Alpha	GradientDrawable
Corner radiu	BitmapDrawable

In Enabled Drawable
click on GradientDrawable.



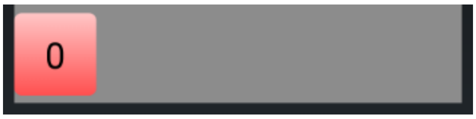
Button Properties	
Drawable	StatelistDrawable
Enabled Drawa	GradientDrawable
Corner radiu	5
Orientation	TOP_BOTTOM
First Color	#FFFFC7C7
Second Colc	#FFFF4F4F
Disabled Draw	ColorDrawable
Pressed Drawal	GradientDrawable
Corner radiu	5
Orientation	TOP_BOTTOM
First Color	#FFFF4F4F
Second Colc	#FFFFC7C7

Change the following properties:

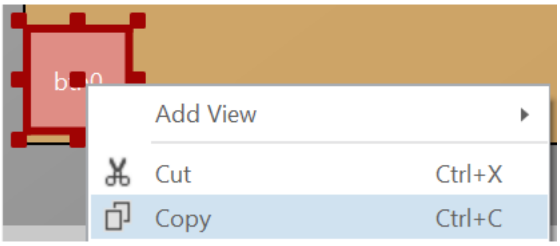
Orientation to TOP_BOTTOM
First Color
Second Color

Pressed Drawable to GradientDrawable

Orientation to TOP_BOTTOM
First Color
Second Color

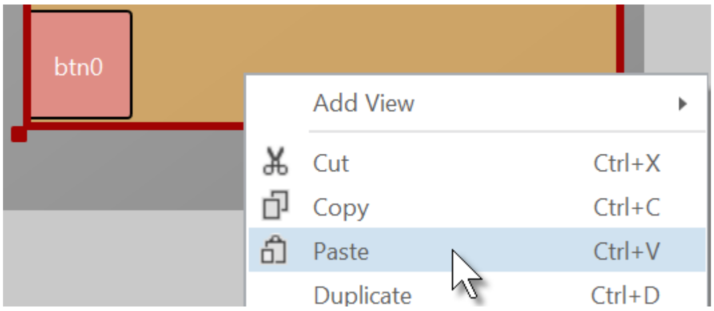


If you have connected a device the button looks now like this.



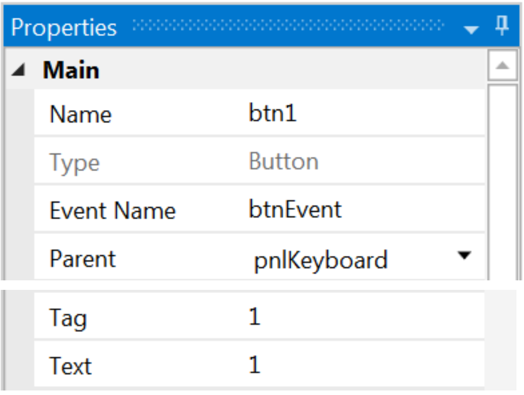
Now we duplicate btn0 and position the new one beside button btn0 with a small space.

Right click on btn0 and click on Copy .



Click on the pnlKeyboard view and click on Paste .

Move the new Button next to the previous one.



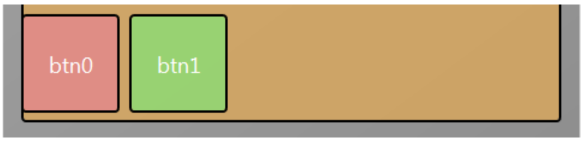
Name	btn1
Type	Button
Event Name	btnEvent
Parent	pnlKeyboard
Tag	1
Text	1

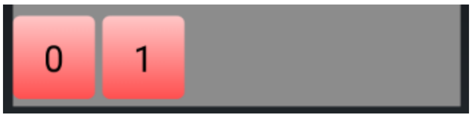
Change the following properties:

Name	to btn1
Tag	to 1
Text	to 1

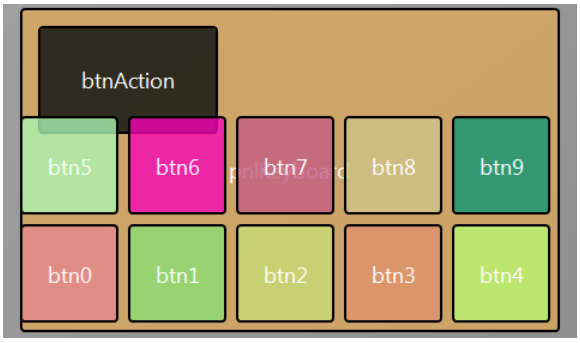
And the result.

In the Abstract Designer and on the device.



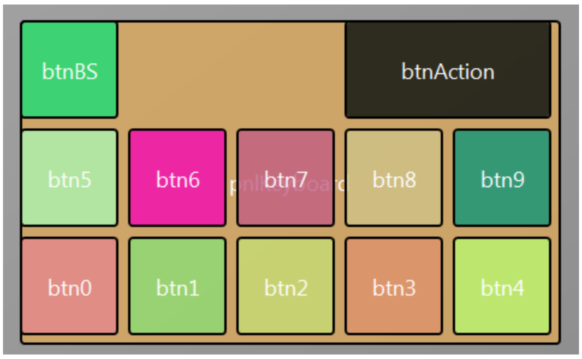


Let us add 8 more Buttons and position them like in the image.



Change following properties:

Name	btn2 ,	btn3 ,	btn4	etc.
Tag	2 ,	3 ,	4	etc.
Text	2 ,	3 ,	4	etc.



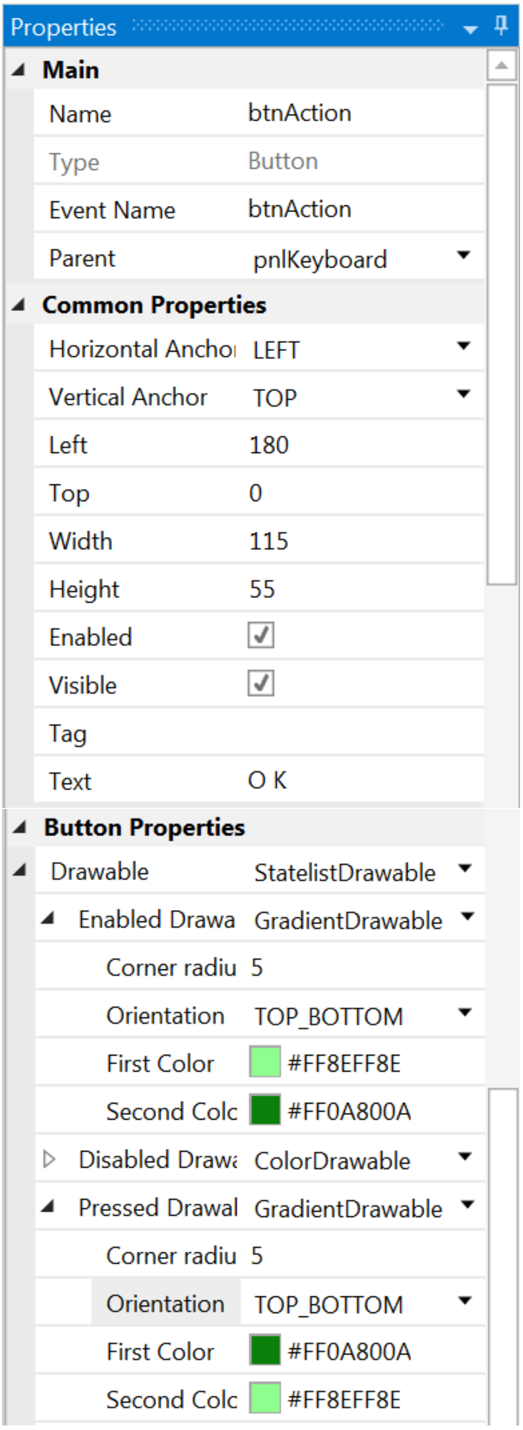
To create the BackSpace button, duplicate one of the number buttons, and position it like in the image.

Resize and position btnAction.

Change the pnlKeyboard Color to Black #000000.

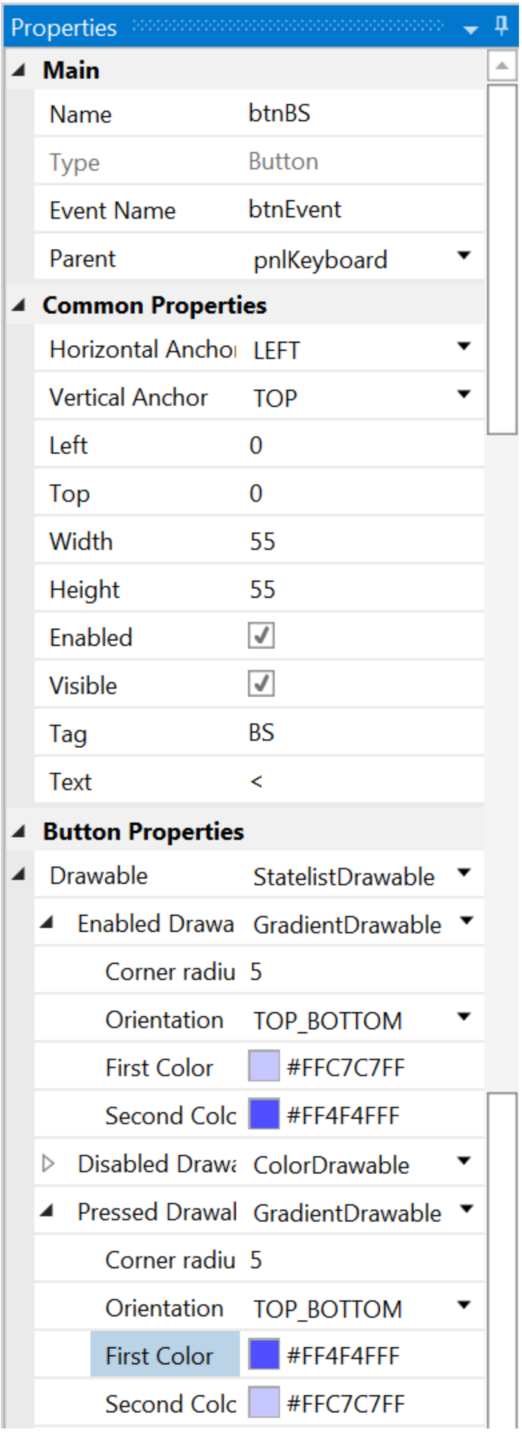
Change their Name, Tag, Text and Color properties as below.

btnBS <



Main	
Name	btnAction
Type	Button
Event Name	btnAction
Parent	pnlKeyboard
Common Properties	
Horizontal Ancho	LEFT
Vertical Anchor	TOP
Left	180
Top	0
Width	115
Height	55
Enabled	☑
Visible	☑
Tag	
Text	O K
Button Properties	
Drawable	StatelistDrawable
Enabled Drawa	GradientDrawable
Corner radiu	5
Orientation	TOP_BOTTOM
First Color	🟩 #FF8EFF8E
Second Colc	🟩 #FF0A800A
Disabled Draw	ColorDrawable
Pressed Drawal	GradientDrawable
Corner radiu	5
Orientation	TOP_BOTTOM
First Color	🟩 #FF0A800A
Second Colc	🟩 #FF8EFF8E

btnAction O K

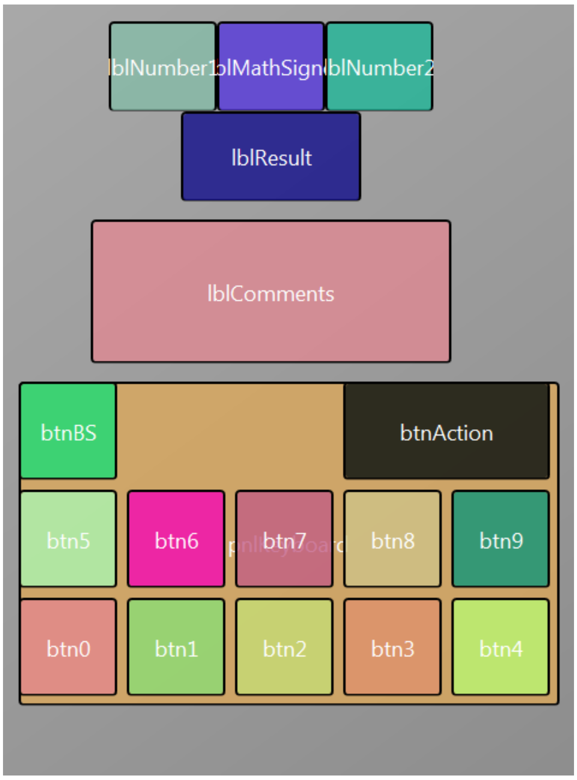


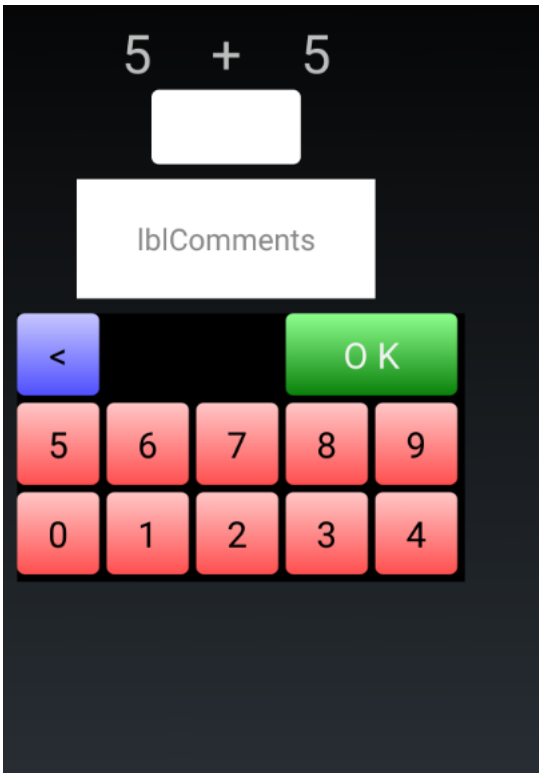
Main	
Name	btnBS
Type	Button
Event Name	btnEvent
Parent	pnlKeyboard
Common Properties	
Horizontal Ancho	LEFT
Vertical Anchor	TOP
Left	0
Top	0
Width	55
Height	55
Enabled	☑
Visible	☑
Tag	BS
Text	<
Button Properties	
Drawable	StatelistDrawable
Enabled Drawa	GradientDrawable
Corner radiu	5
Orientation	TOP_BOTTOM
First Color	🟪 #FFC7C7FF
Second Colc	🟦 #FF4F4FFF
Disabled Draw	ColorDrawable
Pressed Drawal	GradientDrawable
Corner radiu	5
Orientation	TOP_BOTTOM
First Color	🟦 #FF4F4FFF
Second Colc	🟪 #FFC7C7FF

Set the Color property of panel pnlKeyboard to Black.

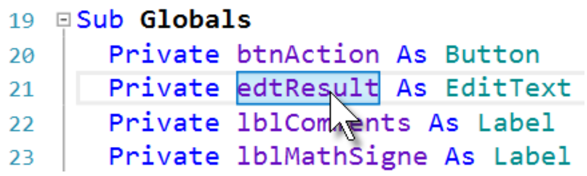
The finished new layout.

In the Abstract Designer and on the device.



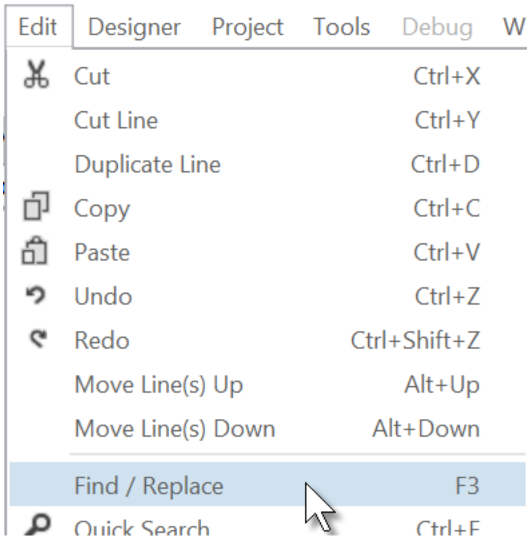


Now we will update the code.
First, we must replace the edtResult by lblResult because we replaced the EditText view by a Label.

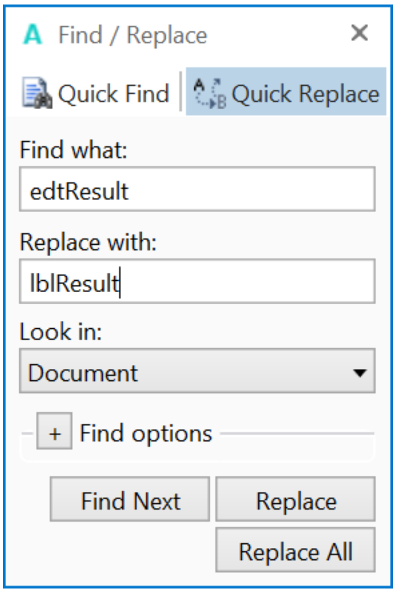


```
Sub Globals
    Private btnAction As Button
    Private edtResult As EditText
    Private lblComments As Label
    Private lblMathSigne As Label
```

Double click on edtResult to select it.



In the Edit menu click on Find / Replace or press F3.



Find what: edtResult

Replace with: lblResult

Look in: Document

Enter ‘lblResult’ in the Replace with field.

Click on Replace All

We also need to change its view type form EditText to Label.

```
Private lblResult As Label
```

Now we write the routine that handles the Click events of the Buttons.
The Event Name for all buttons, except btnAction, is "btnEvent".
The routine name for the associated click event will be btnEvent_Click.
Enter the following code:

```
Sub btnEvent_Click

End Sub
```

We need to know what button raised the event. For this, we use the Sender object which is a special object that holds the object reference of the view that generated the event in the event routine.

```
Sub btnEvent_Click
   Private btnSender As Button

   btnSender = Sender

   Select btnSender.Tag
   Case "BS"
   Case Else
   End Select
End Sub
```

To have access to the properties of the view that raised the event we declare a local variable `Private btnSender As Button`.

And set `btnSender = Sender`.

Then, to differentiate between the backspace button and the numeric buttons we use a Select / Case / End Select structure and use the Tag property of the buttons.
Remember, when we added the different buttons we set their Tag property to BS, 0, 1, 2 etc.

`Select btnSender.Tag`	sets the variable to test.
`Case "BS"`	checks if it is the button with the "BS" tag value.
`Case Else`	handles all the other buttons.

Now we add the code for the numeric buttons.
We want to add the value of the button to the text in the lblResult Label.

```
   Select btnSender.Tag
   Case "BS"
   Case Else
      lblResult.Text = lblResult.Text & btnSender.Text
   End Select
End Sub
```

This is done in this line

```
   lblResult.Text = lblResult.Text & btnSender.Text
```

The "&" character means concatenation, so we just append to the already existing text the value of the Text property of the button that raised the event.

Now we add the code for the BackSpace button.

```
   Select btnSender.Tag
   Case "BS"
      If lblResult.Text.Length >0 Then
         lblResult.Text = lblResult.Text.SubString2(0, lblResult.Text.Length - 1)
      End If
   Case Else
      lblResult.Text = lblResult.Text & btnSender.Text
   End Select
End Sub
```

When clicking on the BS button we must remove the last character from the existing text in lblResult.
However, this is only valid if the length of the text is bigger than 0. This is checked with:

```
If lblResult.Text.Length >0 Then
```

To remove the last character we use the SubString2 function.

```
lblResult.Text = lblResult.Text.SubString2(0,lblResult.Text.Length - 1)
```

SubString2(BeginIndex, EndIndex) extracts a new string beginning at BeginIndex (inclusive) until EndIndex (exclusive).

Now the whole routine is finished.

```
Sub btnEvent_Click
   Private btnSender As Button

   btnSender = Sender

   Select btnSender.Tag
   Case "BS"
      If lblResult.Text.Length >0 Then
         lblResult.Text = lblResult.Text.SubString2(0,lblResult.Text.Length - 1)
      End If
   Case Else
      lblResult.Text = lblResult.Text & btnSender.Text
   End Select
End Sub
```

We can try to improve the user interface of the program by adding some colors to the lblComments Label.

Let us set:

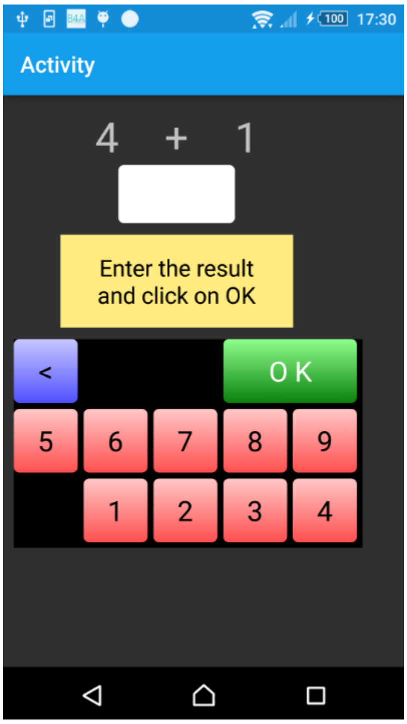
- Yellow for a new problem
- Light Green for a GOOD answer
- Light Red for a WRONG answer.

Let us first modify the New routine, where we add the line `lblResult.Text = ""`.

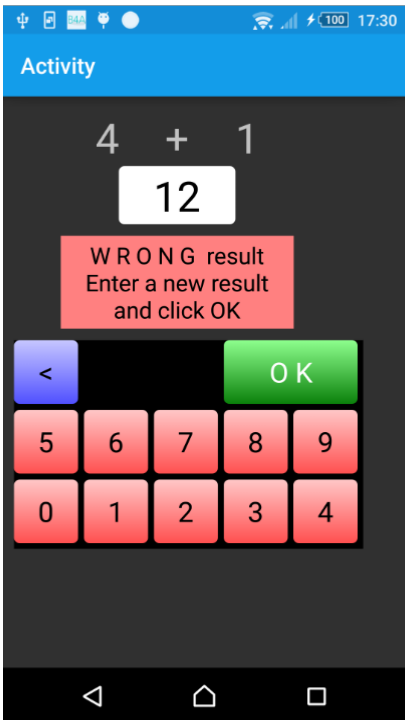
```
Sub New
   Number1 = Rnd(1, 10)           ' Generates a random number between 1 and 9
   Number2 = Rnd(1, 10)           ' Generates a random number between 1 and 9
   lblNumber1.Text = Number1      ' Displays Number1 in label lblNumber1
   lblNumber2.Text = Number2      ' Displays Number2 in label lblNumber2
   lblComments.Text = "Enter the result" & CRLF & "and click on OK"
   lblComments.Color = Colors.RGB(255,235,128)       ' yellow color
   lblResult.Text = ""            ' Sets lblResult.Text to empty
End Sub
```

And in the CheckResult routine we add lines 76 and 80.

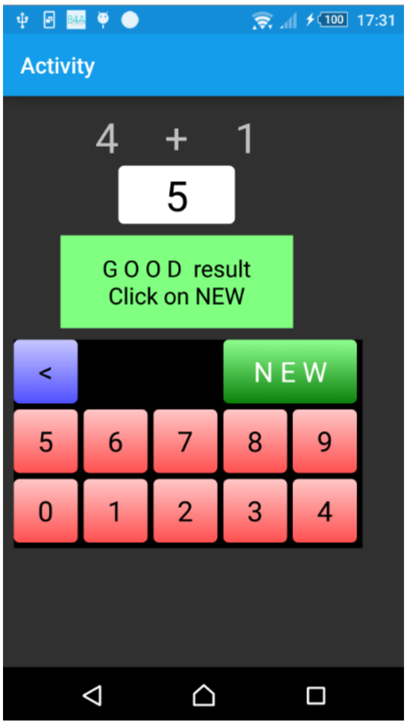
```
Sub CheckResult
   If lblResult.Text = Number1 + Number2 Then
      lblComments.Text = "G O O D  result" & CRLF & "Click on NEW"
      lblComments.Color = Colors.RGB(128,255,128)       ' light green color
      btnAction.Text = "N E W"
   Else
      lblComments.Text = "W R O N G  result" & CRLF & "Enter a new result" & CRLF & "and click OK"
      lblComments.Color = Colors.RGB(255,128,128)       ' light red color
   End If
End Sub
```



Enter the result
and click on OK



W R O N G result
Enter a new result
and click OK



G O O D result
Click on NEW

Another improvement would be to hide the '0' button to avoid entering a leading '0'.
For this, we hide the button in the New subroutine in line `btn0.Visible = False`.

```
Sub New
   Number1 = Rnd(1, 10)         ' Generates a random number between 1 and 9
   Number2 = Rnd(1, 10)         ' Generates a random number between 1 and 9
   lblNumber1.Text = Number1    ' Displays Number1 in label lblNumber1
   lblNumber2.Text = Number2    ' Displays Number2 in label lblNumber2
   lblComments.Text = "Enter the result" & CRLF & "and click on OK"
   lblComments.Color = Colors.RGB(255,235,128)      ' yellow color
   lblResult.Text = ""          ' Sets lblResult.Text to empty
   btn0.Visible = False
End Sub
```

In addition, in the btnEvent_Click subroutine, we hide the button if the length of the text in lblResult is equal to zero and show it if the length is greater than zero, lines 98 to 102.

```
Sub btnEvent_Click
   Private btnSender As Button

   btnSender = Sender

   Select btnSender.Tag
   Case "BS"
      If lblResult.Text.Length >0 Then
         lblResult.Text = lblResult.Text.SubString2(0,lblResult.Text.Length - 1)
      End If
   Case Else
      lblResult.Text = lblResult.Text & btnSender.Tag
   End Select

   If lblResult.Text.Length = 0 Then
      btn0.Visible = False
   Else
      btn0.Visible = True
   End If
End Sub
```

As we are accessing btn0 in the code we need to declare it in the Globals routine.

Modify line 25 like below:

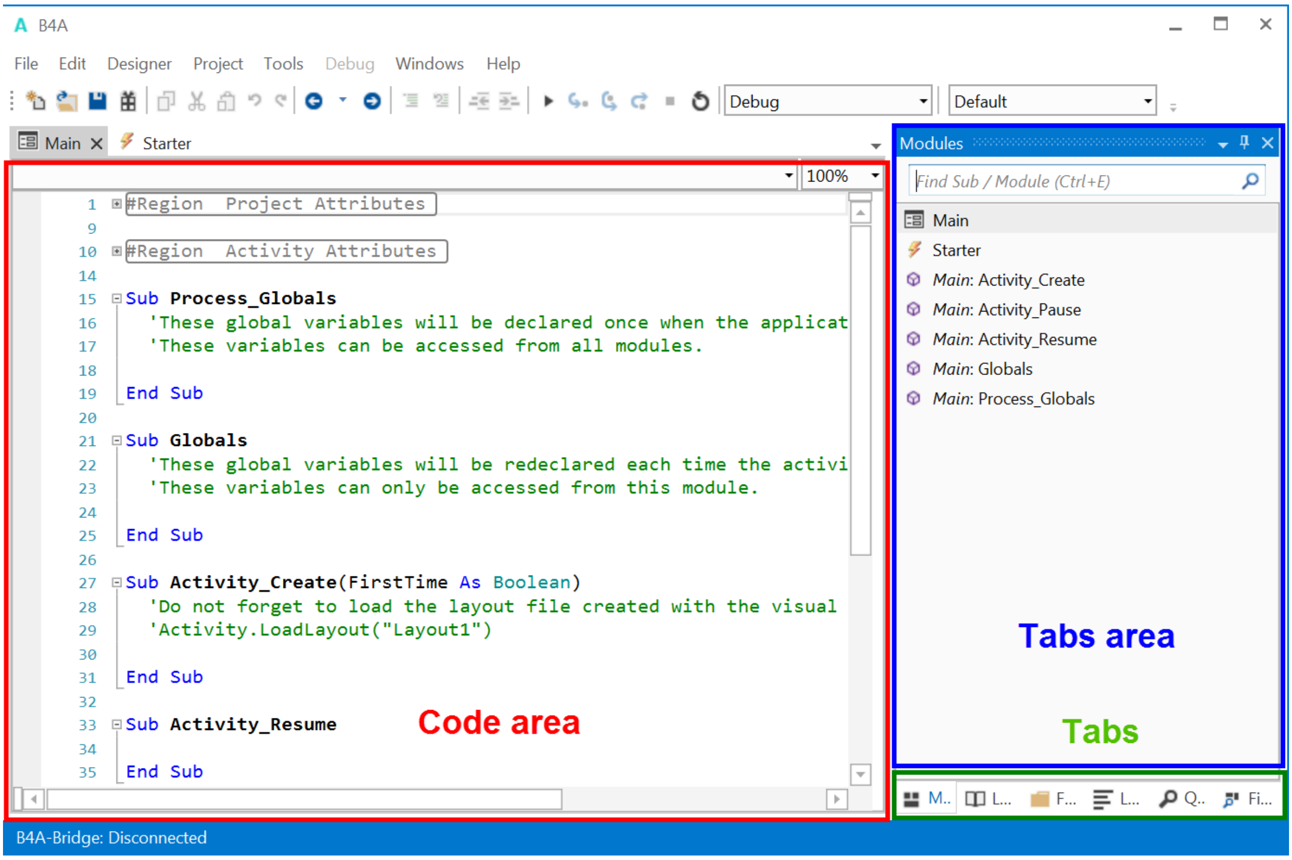
```
   Private btnAction, btn0 As Button
```

Run the program to check the result.

4 The IDE

The **I**ntegrated **D**evelopment **E**nvironment.

When you run the IDE you will get a form like the image below:

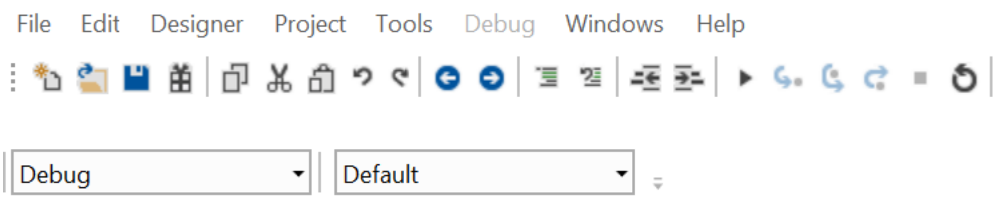


```
#Region  Project Attributes

#Region  Activity Attributes

Sub Process_Globals
	'These global variables will be declared once when the applicat
	'These variables can be accessed from all modules.

End Sub

Sub Globals
	'These global variables will be redeclared each time the activi
	'These variables can only be accessed from this module.

End Sub

Sub Activity_Create(FirstTime As Boolean)
	'Do not forget to load the layout file created with the visual
	'Activity.LoadLayout("Layout1")

End Sub

Sub Activity_Resume

End Sub
```

You see 3 main areas:

• Code area	The code editor
• Tab area	The content of this area depends on the selected Tab.
• Tabs	Tabs for different settings.

4.1 Menu and Toolbar



4.1.1 Toolbar

Generates a new empty project [Ctrl + N].

Loads a project.

Saves the current project [Ctrl + S].

Export As Zip.

Copies the selected text to the clipboard [Ctrl + C].

Cuts the selected text and copies it to the clipboard [Ctrl + X].

Pastes the text in the clipboard at the cursor position [Ctrl + V].

Undoes the last operation [Ctrl + Z].

Redoes the previous operation [Ctrl + Shift + Z].

Navigate backwards [Alt + Left].

Navigate forwards [Alt + Right].

Block Comment [Ctrl + Q].

Block Uncomment [Ctrl + W].

Decrease the indentation of the selected lines.

Increase the indentation of the selected lines.

Runs the compiler [F5].

Step In [F8].

Step Over [F9].

Step Out [F10].

Stop.

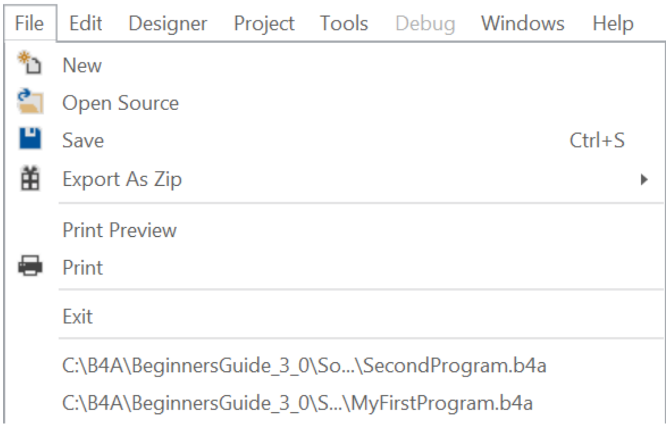
Restart [F11].

These 5 functions are active only when the debugger is active.

Compiler options list and Debugging.

Conditional compiling options.

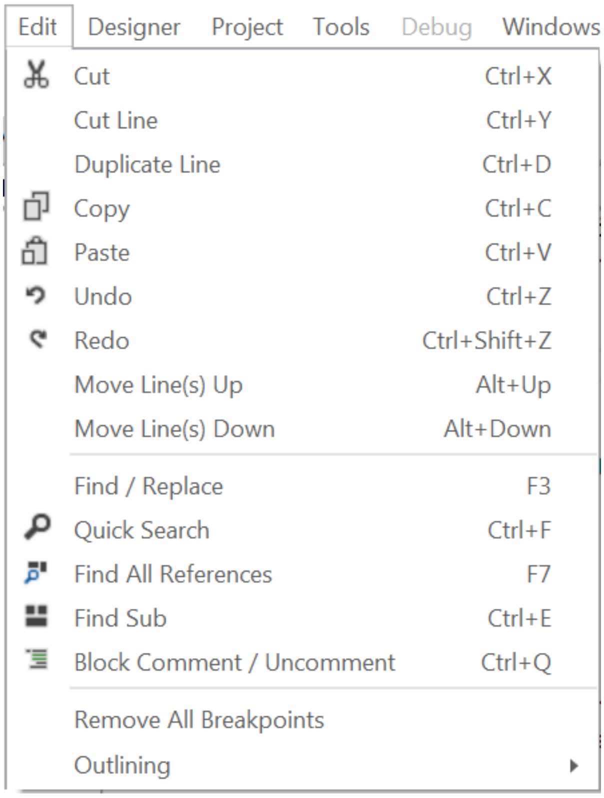
4.1.2 File menu



New Generates a new empty project.
Open Source Loads a project.
Save Saves the current project.
Export As Zip Exports the whole project in a zip file.
Print Preview Preview of the print.
Print Prints the whole code of the selected Module.
Exit Leaves the IDE.

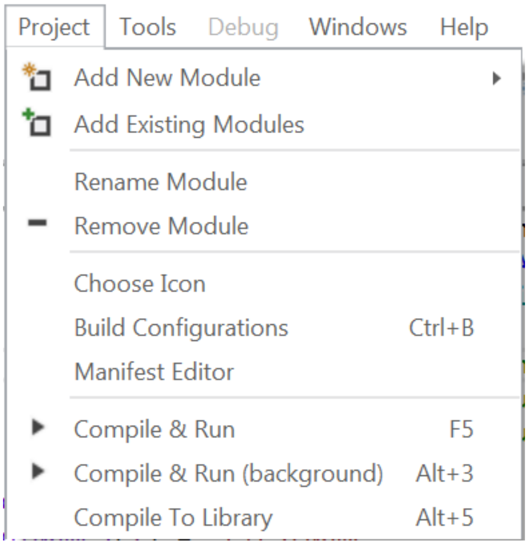
List of last loaded programs.

4.1.3 Edit menu



Cut Cuts the selected text and copies it to the clipboard.
Cut Line Cuts the line at the cursor position.
Copy Copies the selected text to the clipboard.
Paste Pastes the text in the clipboard at the cursor position.
Undo Undoes the last operation.
Redo Redoes the previous operation.
Move Line(s) Up Moves the selected lines upwards.
Move Line(s) Down Moves the selected lines downwards.
Find / Replace Activates the Find and Replace function.
Quick Search Quick Search
Find All References Find All References
Find Sub Find Sub
Block Comment / Uncomment Comment / Uncomment the selected lines.
Remove All Breakpoints Breakpoints.
Outlining Collapse the whole code.

4.1.4 Project menu



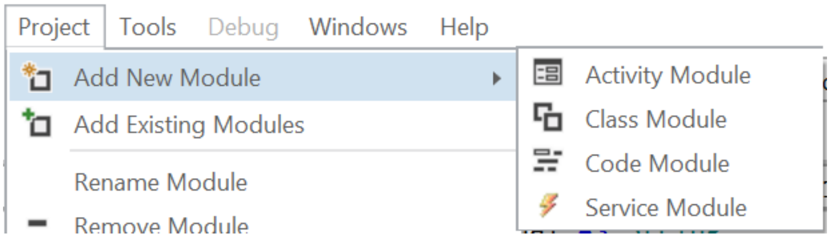
Adds a new module
Adds an existing module

Changes the module name
Removes the current module

Chooses an icon for the program.
Changes the package name.
Runs the Manifest Editor.

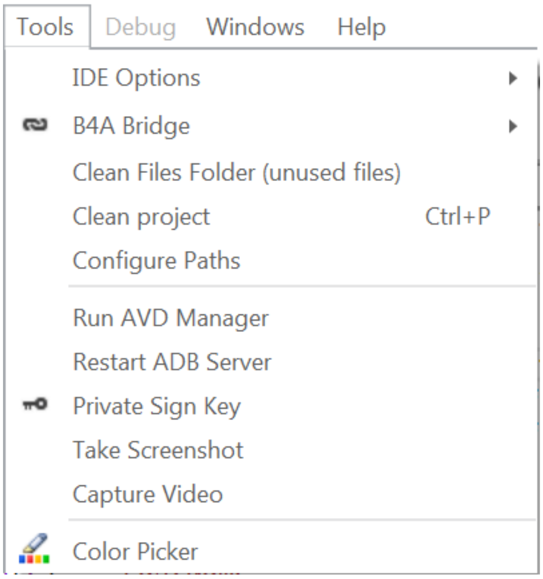
Compile and run the project.
Compile and run the project in the background.
Compile to a library.

4.1.4.1 Add a new module



Activity module
Class module
Code module
Service module

4.1.5 Tools menu



IDE Options see below

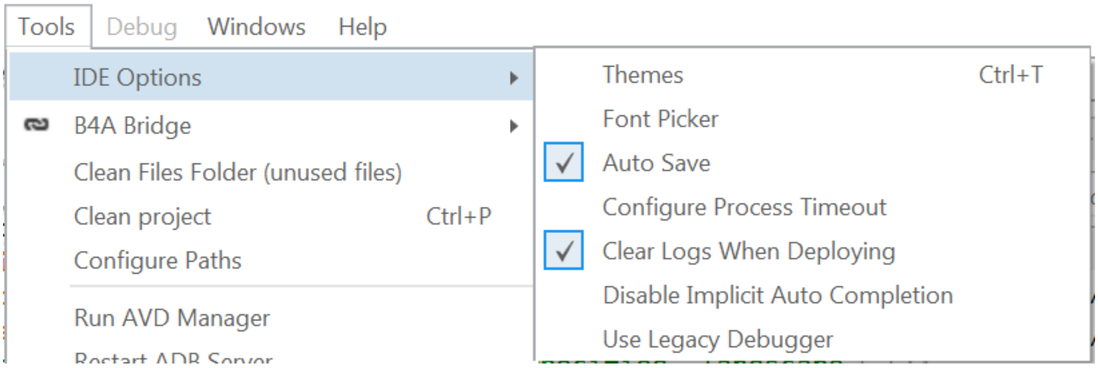
B4A Bridge, connection with Bluetooth or Wifi
Clean Files Folder (unused files)
Clean Project
Configure Paths

Run AVD Manager

Take Screenshot
Capture a video

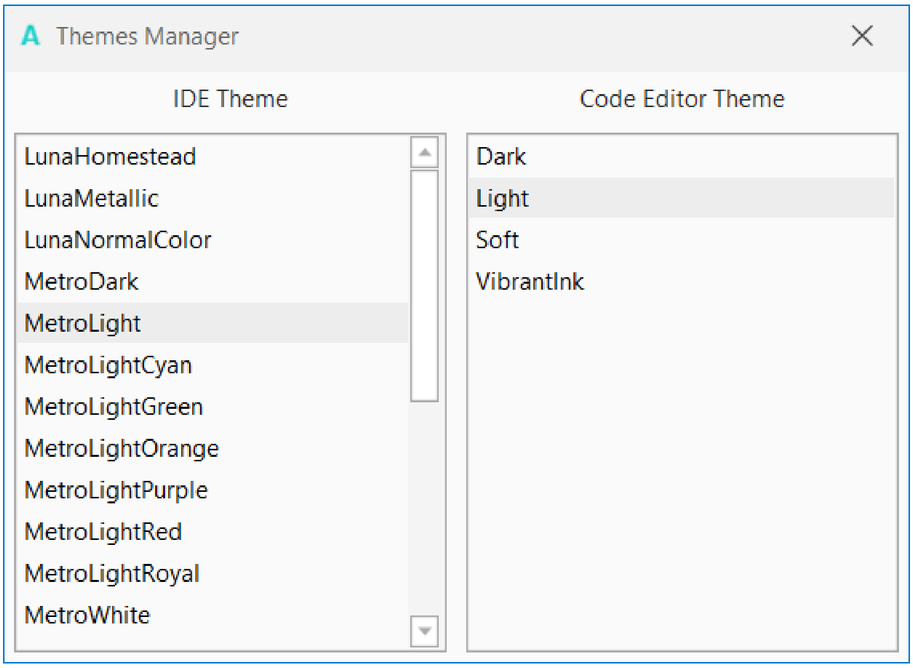
Show the Color Picker

4.1.5.1 IDE Options



Themes.
Font Picker.
Auto Save Saves the program every time you run it.
Configure Process Timeout
Clear Logs When Deploying Removes all Log statements when compiled in Release mode.
Disable Implicit Auto Completion.
Use Legacy Debugger Use the legacy Debugger instead of the rapid Debugger.

4.1.5.1.1 Themes

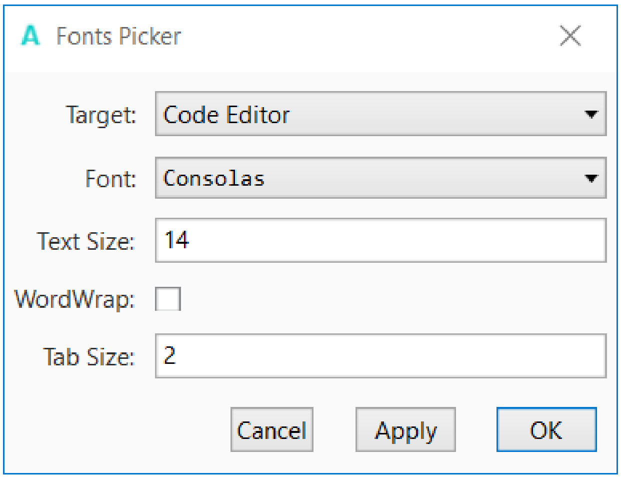


You can select different themes for the IDE.

The default theme is MetroLight.

When you select one you see directly the new colors.

4.1.5.1.2 Font Picker


Fonts Picker

Target: Code Editor

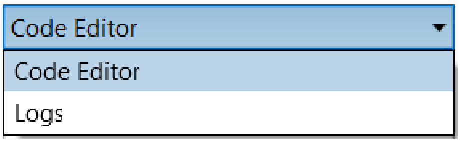
Font: Consolas

Text Size: 14

WordWrap: □

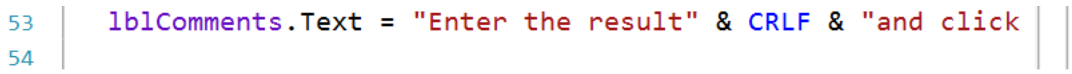
Tab Size: 2

You can select the target Code Editior or Logs.



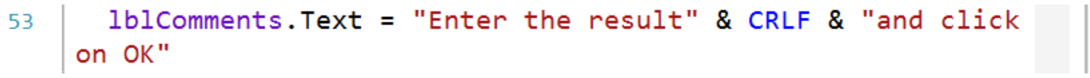
Different fonts.
Enter the text size.
Select WordWrap
Enter the Tab size.

4.1.5.1.2.1 Word wrap



```
    lblComments.Text = "Enter the result" & CRLF & "and click
```

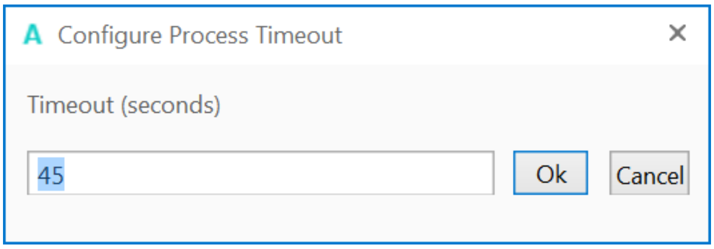
Without word wrap. The end of the line is hidden.



```
    lblComments.Text = "Enter the result" & CRLF & "and click
on OK"
```

With word wrap. The end of the line is wrapped to the next line.

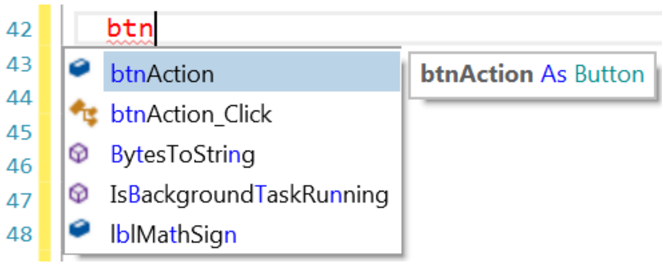
4.1.5.1.2.2 Configure Process Timeout


Configure Process Timeout

Timeout (seconds): 45

Sometimes the compilation needs more time. If you get a message 'Process timeout' you can increase the time.

4.1.5.1.2.3 Disable Implicit Auto Completion



```
    btn
```

If Disable Implicit Auto Completion is unchecked you will see a drop down list with possible words during typing.

If checked ☑ Disable Implicit Auto Completion you won't see the auto completion list.

4.1.5.2 Take Screenshot

The Take Screenshot function can be called from the:

- Tools menu when the IDE is in edit mode
- Debug menu when the IDE is in debug mode

Note: This function works only with USB connetion not with B4A-Bridge !

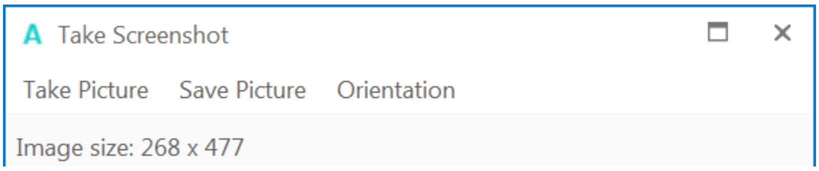


Image size: 268 x 477

Clicking on Take Screenshot shows this window.

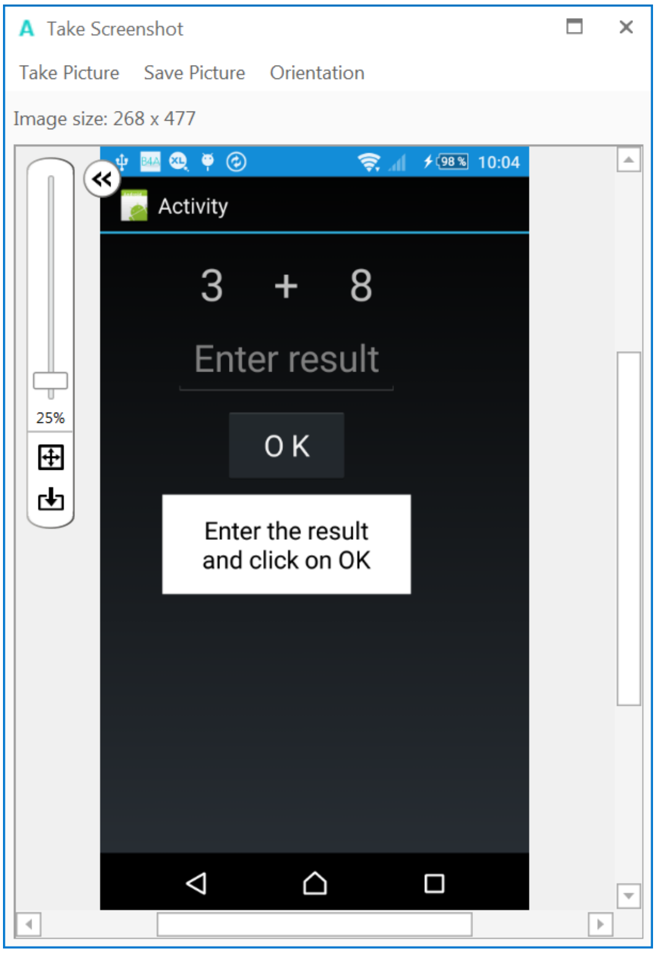


Image size: 268 x 477

3 + 8

Enter result

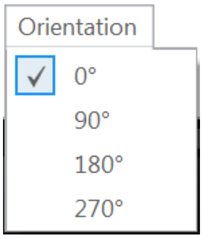
OK

Enter the result and click on OK

Click on Take Picture to take the screenshot picture from the device.

You can resize the image with the cursor on the left side.

You can save the image with Save Picture as a PNG file.



And you can change the orientation of the picture.

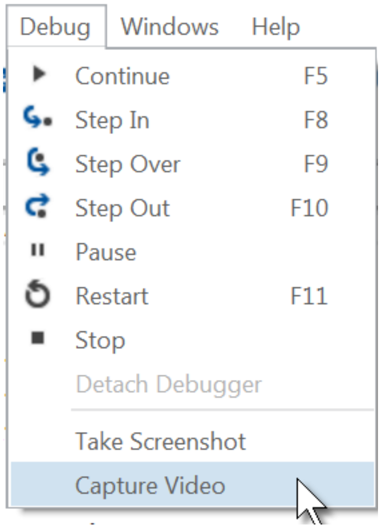
Copy To Clipboard

Right click on the image to copy the image to the clipboard.

4.1.5.3 Create Video

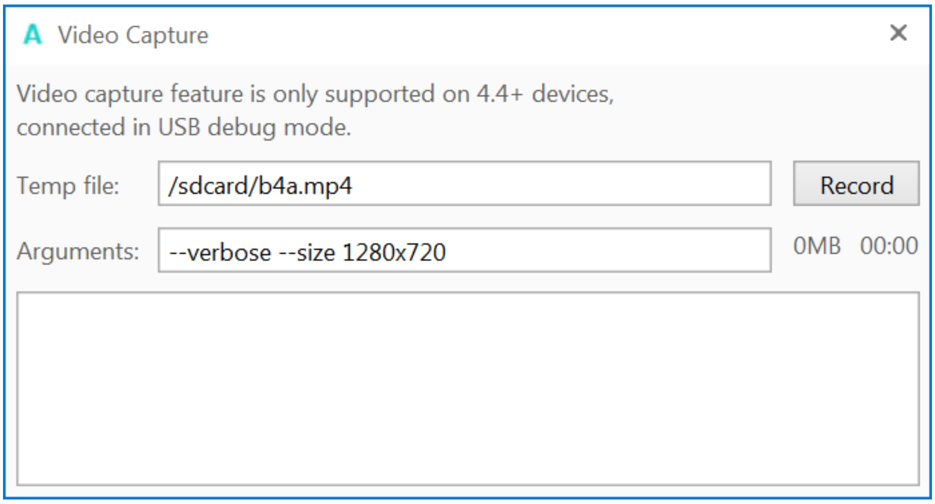
You can run your program and record a video when you use it.

Note: This function works only with USB connetion not with B4A-Bridge !



In the Debug menu click on Capture Video.

The sceen below will be dispayed:



Video capture feature is only supported on 4.4+ devices, connected in USB debug mode.

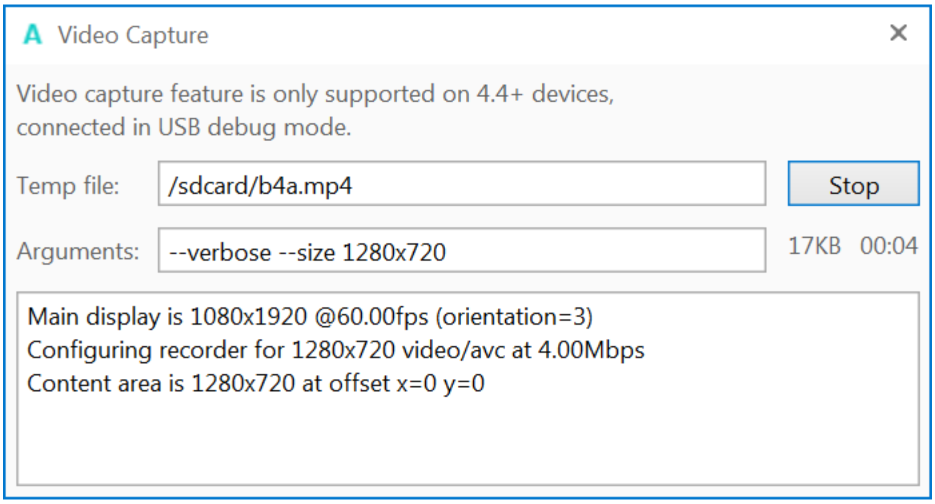
Temp file: /sdcard/b4a.mp4

Arguments: --verbose --size 1280x720

0MB 00:00

Click on Record to begin recording.

A screen similar to this one will be dispaled:



Video capture feature is only supported on 4.4+ devices, connected in USB debug mode.

Temp file: /sdcard/b4a.mp4

Arguments: --verbose --size 1280x720

17KB 00:04

```
Main display is 1080x1920 @60.00fps (orientation=3)
Configuring recorder for 1280x720 video/avc at 4.00Mbps
Content area is 1280x720 at offset x=0 y=0
```

Click on Stop to stop recording.

You will be asked where you want to save the file on the computer.

4.1.5.4 Clean Files Folder (unused files)

Deletes files that are located under the Files folder but are not used by the project (it will not delete any file referenced by any of the project layouts). A list of unused files will be displayed before deletion (and you may cancel the operation).

4.1.5.5 Clean Project

Deletes all files that are generated during compilation.

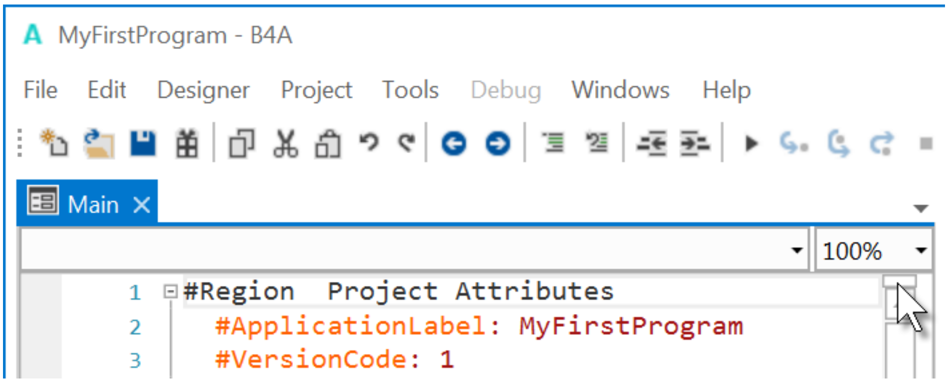
4.2 Code area

The code of the selected module is displayed in this area and can be edited.
The examples below are based on the code of the SecondProgram.

4.2.1 Split the code area

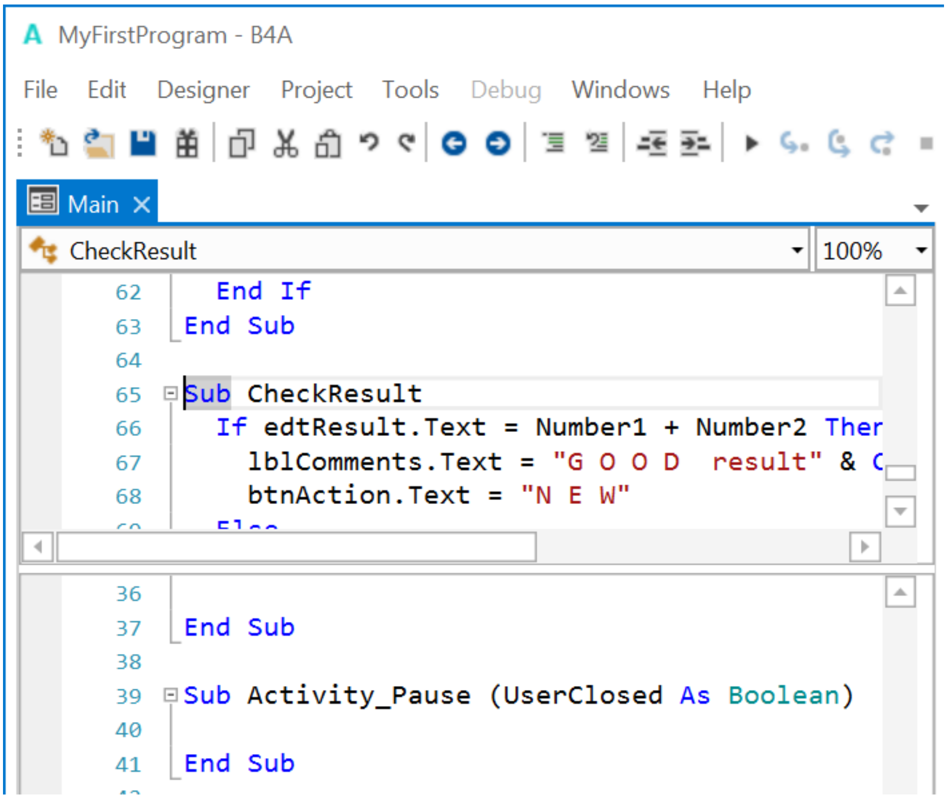
It is possible to split the code area into two parts allowing to edit two different code parts at the same time.

Move the small rectangle below the zoom level.



```
#Region  Project Attributes
   #ApplicationLabel: MyFirstProgram
   #VersionCode: 1
```

And the result.



```
    End If
End Sub

Sub CheckResult
    If edtResult.Text = Number1 + Number2 Ther
        lblComments.Text = "G O O D  result" & C
        btnAction.Text = "N E W"
```

```

End Sub

Sub Activity_Pause (UserClosed As Boolean)

End Sub
```

4.2.2 Code header Project Attributes / Activity Attributes

A code header, with general settings, is added at the beginning of the code.

4.2.2.1 Project Attributes

Attributes that are valid for the whole project. Displayed only in the Main module.

```
#Region  Project Attributes
  #ApplicationLabel: SecondProgram
  #VersionCode: 1
  #VersionName:
  'SupportedOrientations possible values: unspecified, landscape or portrait.
  #SupportedOrientations: unspecified
  #CanInstallToExternalStorage: False
#End Region
```

`#ApplicationLabel:` The name which will be displayed below the program icon on the device.
`#VersionCode:` The version of the code, it is not displayed.
`#VersionName:` You can add a name for the version.
`#SupportedOrientations:` You can limit the whole program to a given orientation.
`#CanInstallToExternalStorage:` If you want to install the program on an external storage card you must set this attribute to True.

You can add or change the values to your needs.

4.2.2.2 Activity Attributes

Valid for the current activity.

```
#Region  Activity Attributes
  #FullScreen: False
  #IncludeTitle: True
#End Region
```

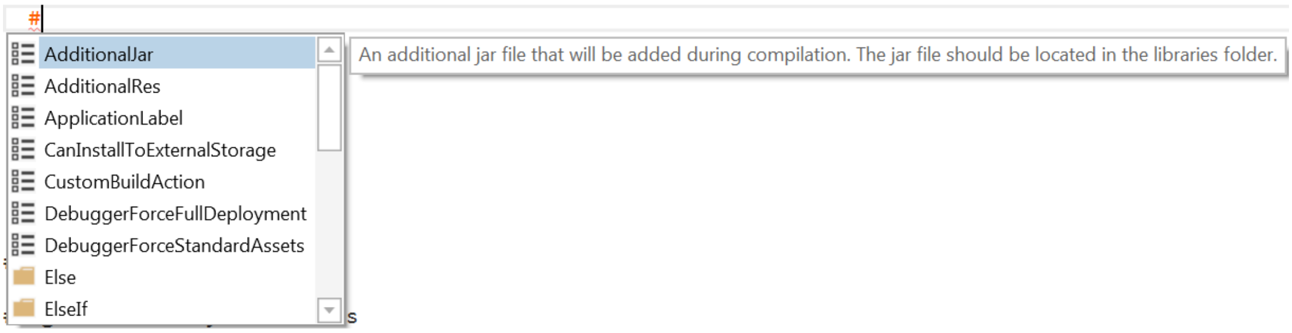
When you add a new Activity you'll find the Activity Attributes region on top.

```
#Region  Activity Attributes
  #FullScreen: False
  #IncludeTitle: True
#End Region
```

When you add a new Service you'll find the Service Attributes header.

```
#Region  Service Attributes
  #StartAtBoot: False
#End Region
```

When you want to add a new Attribute you can just write # and the inline help shows all possibilities.



```
#
AdditionalJar
AdditionalRes
ApplicationLabel
CanInstallToExternalStorage
CustomBuildAction
DebuggerForceFullDeployment
DebuggerForceStandardAssets
Else
ElseIf
```

An additional jar file that will be added during compilation. The jar file should be located in the libraries folder.

Note the two different icons:

☰ Attributes.

🗀 Conditional compilation and region keywords.

When you load a project saved with a version of B4A older than 2.5 then the header will look like this:

```
#Region Module Attributes
   #FullScreen: False
   #IncludeTitle: True
   #ApplicationLabel: MyFirstProgram
   #VersionCode: 1
   #VersionName:
   #SupportedOrientations: unspecified
   #CanInstallToExternalStorage: False
#End Region
```

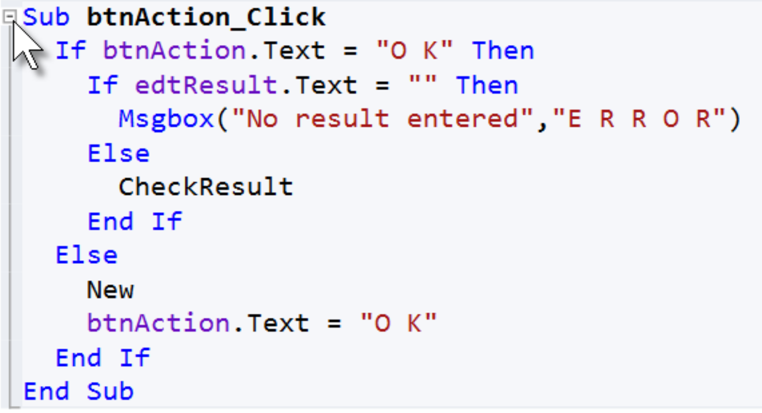
4.2.3 Undo – Redo ↶ ↷

In the IDE it is possible to undo the previous operations and redo undone operations.
Click on ↶ to undo and on ↷ to redo.

4.2.4 Collapse a subroutine

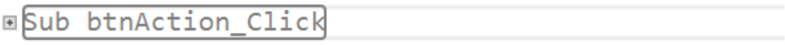
A subroutine can be collapsed to minimize the number of lines displayed.

The btnAction_Click routine expanded.



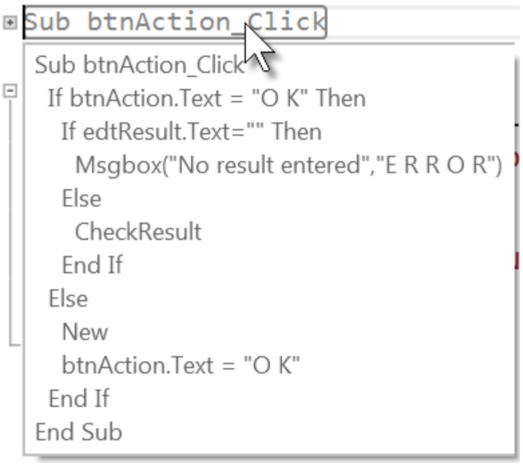
```
Sub btnAction_Click
  If btnAction.Text = "O K" Then
    If edtResult.Text = "" Then
      Msgbox("No result entered","E R R O R")
    Else
      CheckResult
    End If
  Else
    New
    btnAction.Text = "O K"
  End If
End Sub
```

Click on ⊟ to collapse the subroutine.



```
Sub btnAction_Click
```

The btnAction_Click routine collapsed.



```
Sub btnAction_Click
```

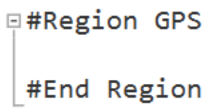
```
Sub btnAction_Click
 If btnAction.Text = "O K" Then
  If edtResult.Text="" Then
   Msgbox("No result entered","E R R O R")
  Else
   CheckResult
  End If
 Else
  New
  btnAction.Text = "O K"
 End If
End Sub
```

Hovering with the mouse over the collapsed routine name shows its content.

4.2.5 Collapse a Region

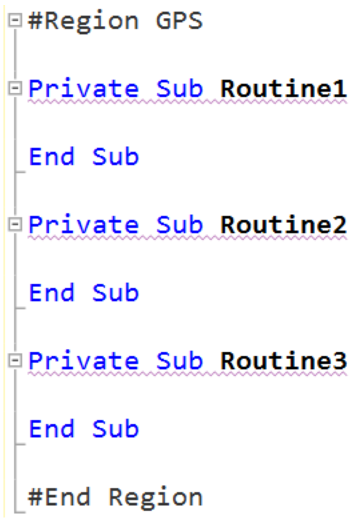
You can define ‘Regions’ in the code, which can be collapsed.

Example:



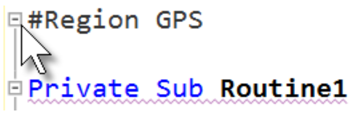
```
#Region GPS

#End Region
```

`#Region GPS` sets the beginning of a region and
`#End Region` the end



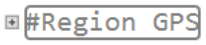
```
#Region GPS

Private Sub Routine1

End Sub

Private Sub Routine2

End Sub

Private Sub Routine3

End Sub

#End Region
```

Then you can add the subroutines between the two limits.



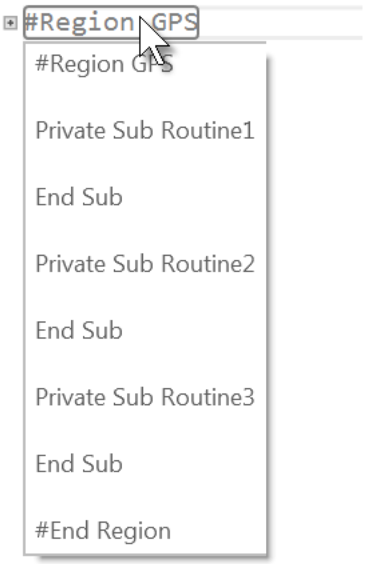
```
#Region GPS

Private Sub Routine1
```

Then click on ⊟ to collapse the whole region.



```
#Region GPS
```

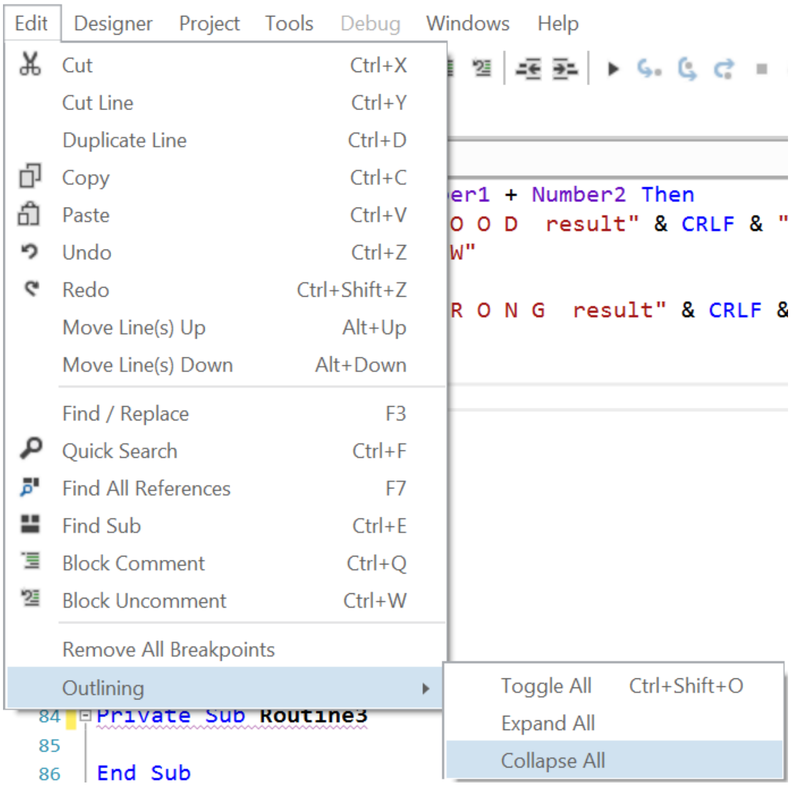
Hovering over GPS



```
#Region GPS
#Region GPS

Private Sub Routine1

End Sub

Private Sub Routine2

End Sub

Private Sub Routine3

End Sub

#End Region
```

shows the code. For big regions not all the code is displayed.

4.2.6 Collapse the entire code



```
er1 + Number2 Then
O O D  result" & CRLF & "
W"
R O N G  result" & CRLF &
84 Private Sub Routine3
85
86   End Sub
```

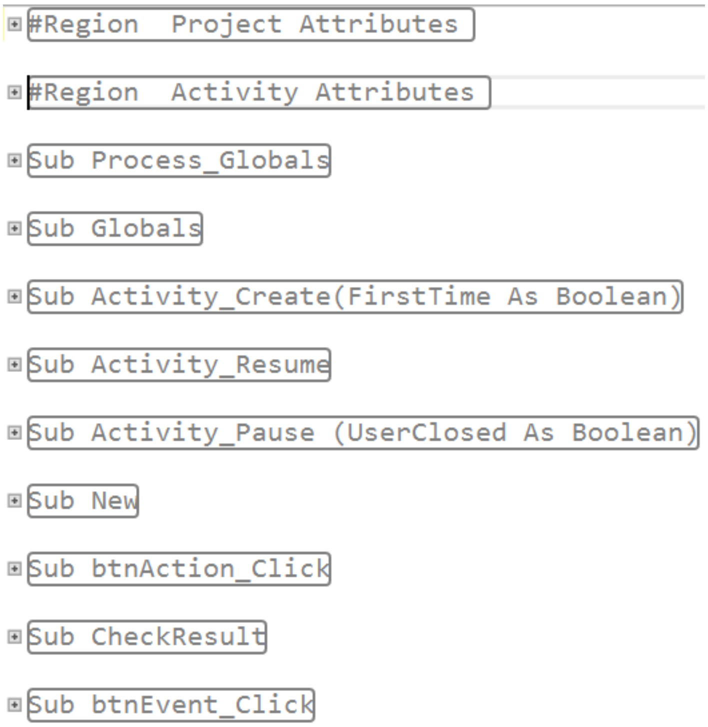
In the Edit / Outlining menu there are three functions:

- Toggle All
Expands the collapsed routines and collapses the expanded routines and regions.

- Expand All
Expands the entire code

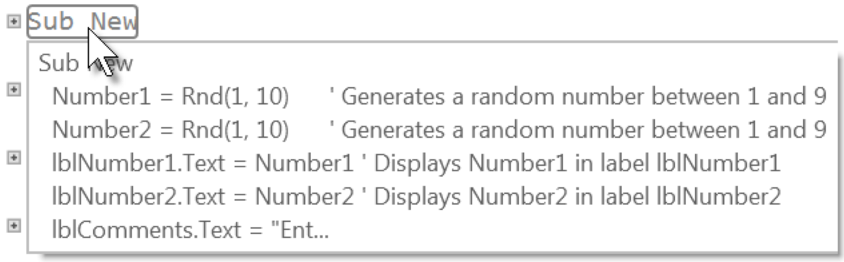
- Collapse All
Collapses the entire code.

Click on Collapse All.



```
#Region  Project Attributes

#Region  Activity Attributes

Sub Process_Globals

Sub Globals

Sub Activity_Create(FirstTime As Boolean)

Sub Activity_Resume

Sub Activity_Pause (UserClosed As Boolean)

Sub New

Sub btnAction_Click

Sub CheckResult

Sub btnEvent_Click
```

The whole code collapsed.



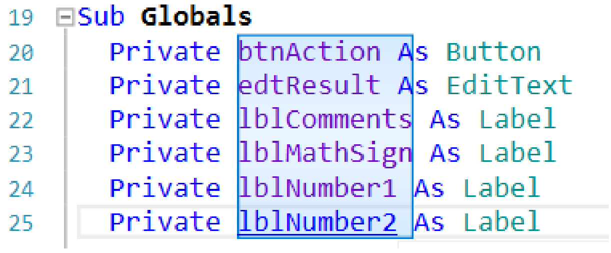
```
Sub New
Sub New
  Number1 = Rnd(1, 10)    ' Generates a random number between 1 and 9
  Number2 = Rnd(1, 10)    ' Generates a random number between 1 and 9
  lblNumber1.Text = Number1 ' Displays Number1 in label lblNumber1
  lblNumber2.Text = Number2 ' Displays Number2 in label lblNumber2
  lblComments.Text = "Ent...
```

Hovering with the mouse over a subroutine shows the beginning of its content.

4.2.7 Copy a selected bloc of text

It is possible to copy a selected bloc of text to the clipboard.

To select the bloc press Alt and move the mouse cursor.

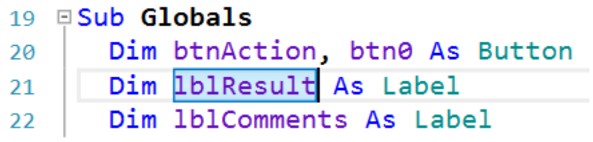


```
Sub Globals
    Private btnAction As Button
    Private edtResult As EditText
    Private lblComments As Label
    Private lblMathSign As Label
    Private lblNumber1 As Label
    Private lblNumber2 As Label
```

4.2.8 Find / Replace

The example uses the code from the SecondProgram project.

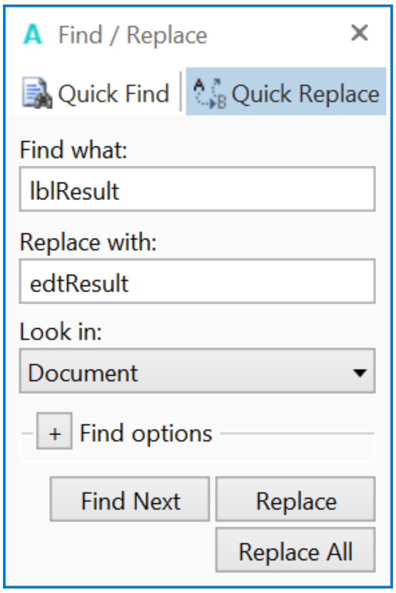
Let’s replace `lblResult` by `edtResult`.



```
Sub Globals
    Dim btnAction, btn0 As Button
    Dim lblResult As Label
    Dim lblComments As Label
```

In the code select `lblResult`.

Press F3 or click on Find / Replace in the Edit menu.



Find what: lblResult

Replace with: edtResult

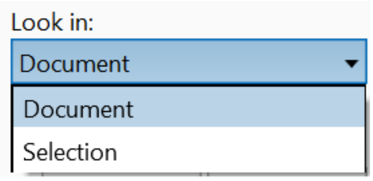
Look in: Document

This window will be displayed

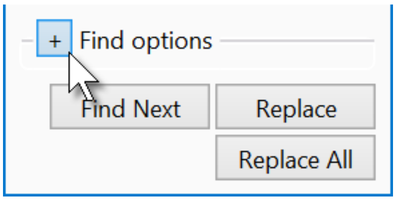
Enter `edtResult` in the ‘Replace with’ field.

Now, you can either:

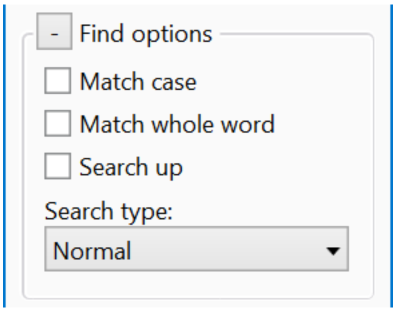
- Find Next find the next occurrence.
- Replace replace the current occurrence and find the next one.
- Replace All replace all occurrencies.



You can search either in a Selection or in the Document, which means in the selected module not the whole document.



You can select Find options, click on + .



☐ Match case

☐ Match whole word

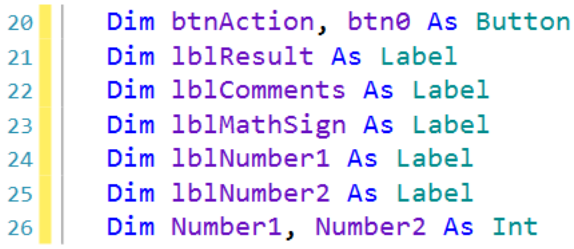
☐ Search up

Search type: Normal

These options are self-explanatory.

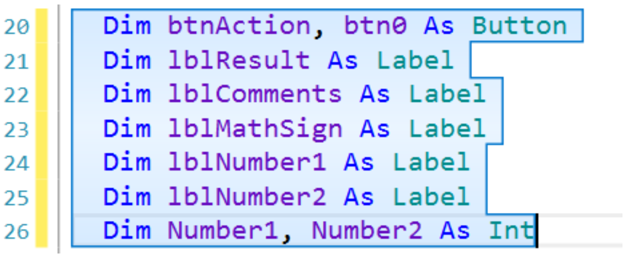
4.2.9 Commenting and uncommenting code ☰ ☰

A selected part of the code can be set to comment lines or set to normal.



```
Dim btnAction, btn0 As Button
Dim lblResult As Label
Dim lblComments As Label
Dim lblMathSign As Label
Dim lblNumber1 As Label
Dim lblNumber2 As Label
Dim Number1, Number2 As Int
```

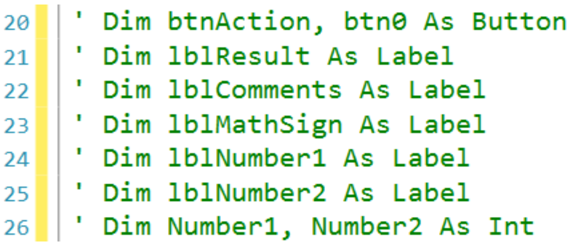
Original code



```
Dim btnAction, btn0 As Button
Dim lblResult As Label
Dim lblComments As Label
Dim lblMathSign As Label
Dim lblNumber1 As Label
Dim lblNumber2 As Label
Dim Number1, Number2 As Int
```

Select the code.

Click on ☰ or Ctrl + Q.



```
' Dim btnAction, btn0 As Button
' Dim lblResult As Label
' Dim lblComments As Label
' Dim lblMathSign As Label
' Dim lblNumber1 As Label
' Dim lblNumber2 As Label
' Dim Number1, Number2 As Int
```

The selected lines set as comments.

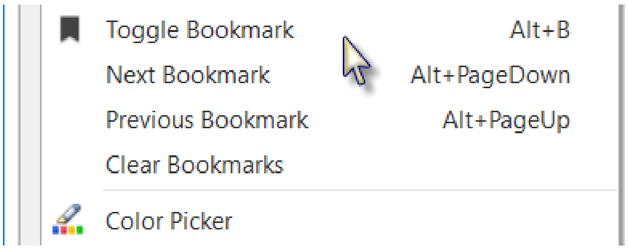
To set the lines to normal,
select the lines and click on ☰ or Ctrl + W.

4.2.10 Bookmarks

You can set 'bookmarks' anywhere in the code and jump forward and backwards between these bookmarks.

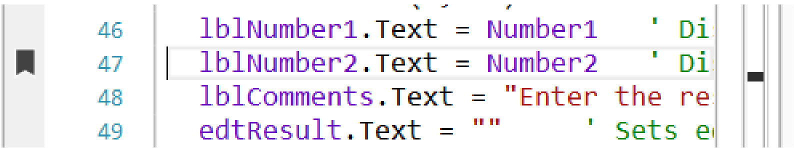
To set or clear a bookmark, select the line and press Alt + B.

Or right click on the line where you want to set a bookmark.



You will get a pop up menu, click on Toggle Bookmark to activate or deactivate a bookmark.

You will see this mark on the left of the line and a small black line in the right slider:



```
lblNumber1.Text = Number1    ' Di
lblNumber2.Text = Number2    ' Di
lblComments.Text = "Enter the re
edtResult.Text = ""        ' Sets e
```

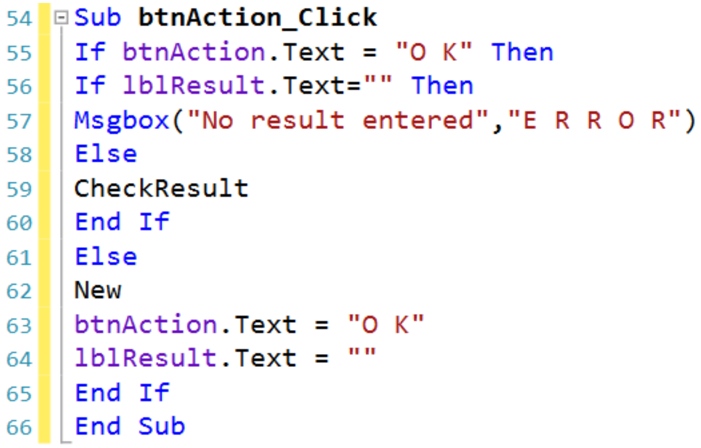
To jump to the next bookmark press Alt + PageDown
or right click and click on Next Bookmark Alt+PageDown

To jump to the previous bookmark press on Alt + PageUp
or right click and click on Previous Bookmark Alt+PageUp

To clear all bookmarks right click and click on Clear Bookmarks

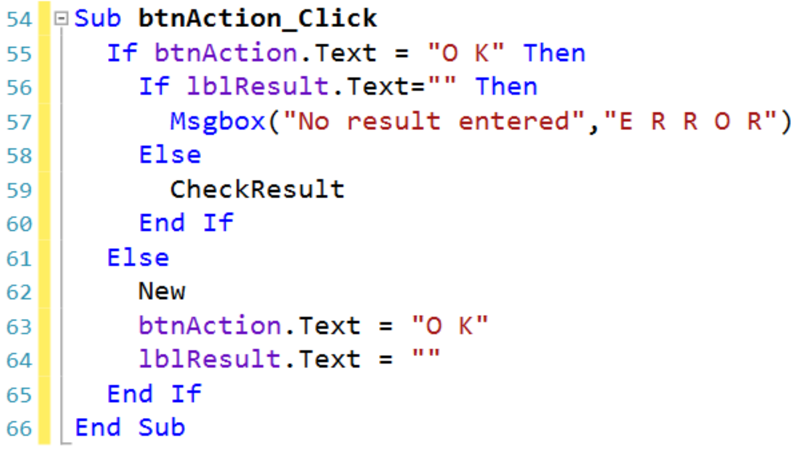
4.2.11 Indentation

A good practice is to use indentation of code parts.
For example for subroutines, loops, structures etc.



```
Sub btnAction_Click
If btnAction.Text = "O K" Then
If lblResult.Text="" Then
Msgbox("No result entered","E R R O R")
Else
CheckResult
End If
Else
New
btnAction.Text = "O K"
lblResult.Text = ""
End If
End Sub
```

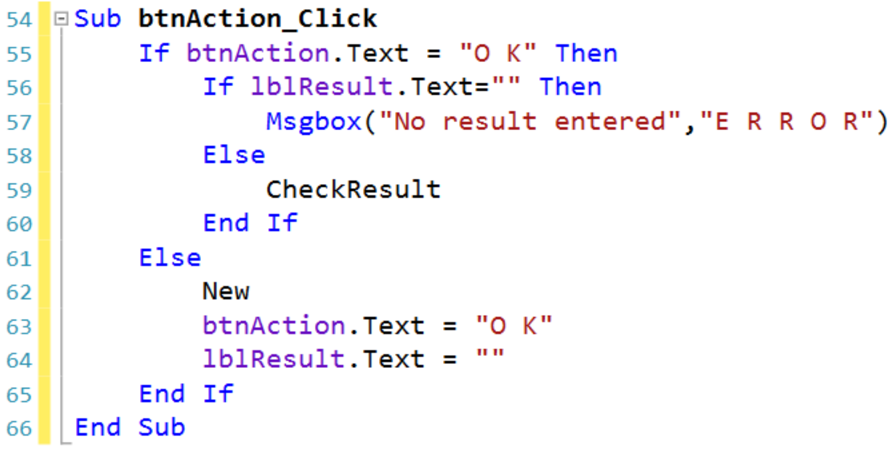
This code is difficult to read because the structure of the code is not obvious.



```
Sub btnAction_Click
  If btnAction.Text = "O K" Then
    If lblResult.Text="" Then
      Msgbox("No result entered","E R R O R")
    Else
      CheckResult
    End If
  Else
    New
    btnAction.Text = "O K"
    lblResult.Text = ""
  End If
End Sub
```

This code is much easier to read, the structure of the code is in evidence.

A tabulation value of 2 for the indentation is a good value.

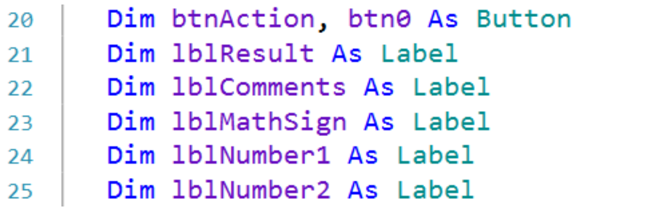


```
Sub btnAction_Click
    If btnAction.Text = "O K" Then
        If lblResult.Text="" Then
            Msgbox("No result entered","E R R O R")
        Else
            CheckResult
        End If
    Else
        New
        btnAction.Text = "O K"
        lblResult.Text = ""
    End If
End Sub
```

Example with an indentation of 4

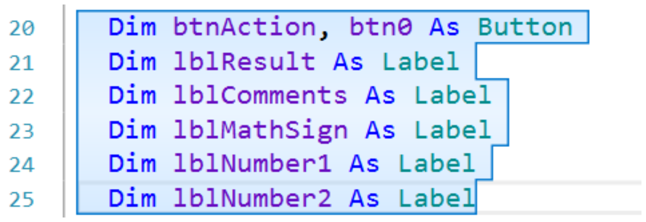
Personally,
I prefer a value of 2.

Whole blocks of code can be indented forth and back at once.



```
  Dim btnAction, btn0 As Button
  Dim lblResult As Label
  Dim lblComments As Label
  Dim lblMathSign As Label
  Dim lblNumber1 As Label
  Dim lblNumber2 As Label
```

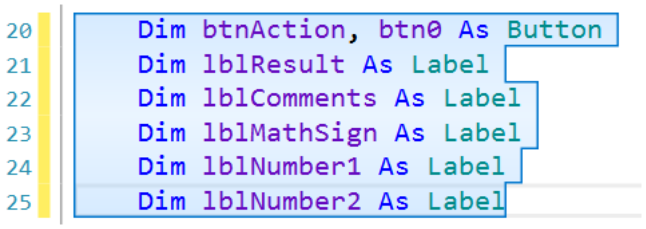
Original code.



```
  Dim btnAction, btn0 As Button
  Dim lblResult As Label
  Dim lblComments As Label
  Dim lblMathSign As Label
  Dim lblNumber1 As Label
  Dim lblNumber2 As Label
```

Select the code block.

Click on ⇥ .

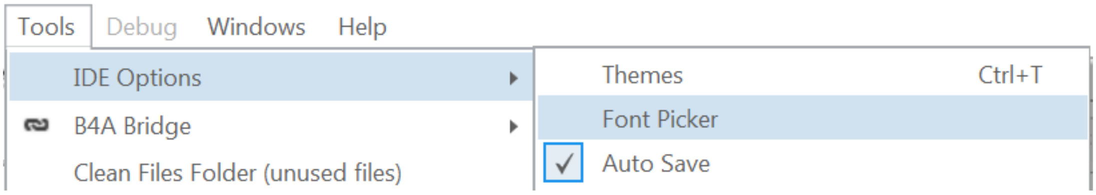


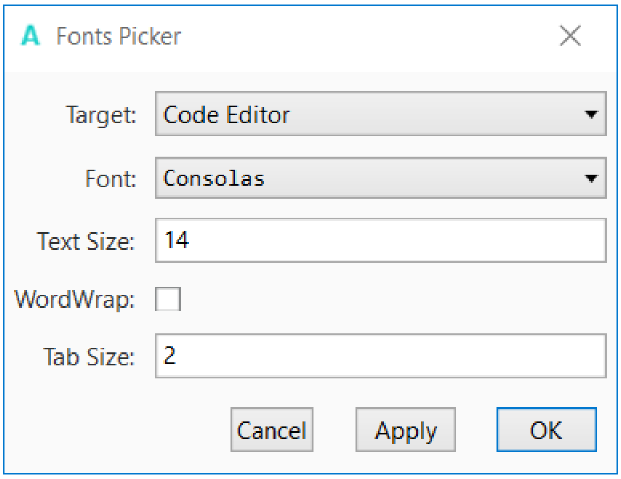
```
    Dim btnAction, btn0 As Button
    Dim lblResult As Label
    Dim lblComments As Label
    Dim lblMathSign As Label
    Dim lblNumber1 As Label
    Dim lblNumber2 As Label
```

The whole block has moved one tabulation to the right.

To move a block to the left.
Select the code block and click on ⇤ .

The indentation value can be changed in the Tools menu IDE Options / Font Picker.





Target: Code Editor

Font: Consolas

Text Size: 14

WordWrap: □

Tab Size: 2

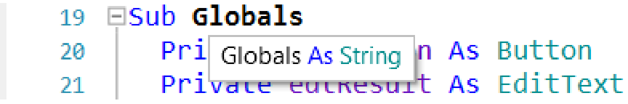
Enter the value and click on OK.

4.2.12 Documentation tool tips while hovering over code elements

When you hover over code elements the on line help is displayed.

Examples:

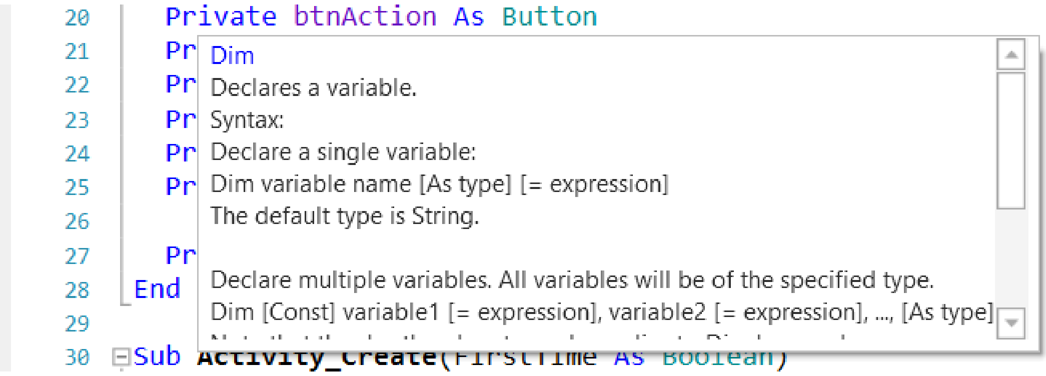
Hovering over Globals:



```
Sub Globals
    Pri              n As Button
    Private edtResult As EditText
```

Globals As String

Hovering over Private:



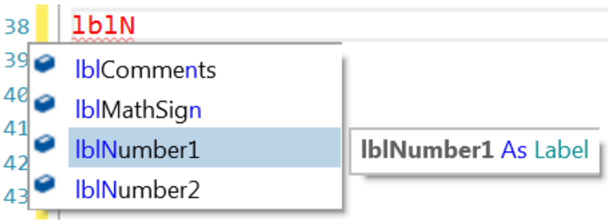
```
    Private btnAction As Button
    Pr
    Pr
    Pr
    Pr
    Pr

    Pr
End

Sub Activity_Create(FirstTime As Boolean)
```

Dim
Declares a variable.
Syntax:
Declare a single variable:
Dim variable name [As type] [= expression]
The default type is String.

Declare multiple variables. All variables will be of the specified type.
Dim [Const] variable1 [= expression], variable2 [= expression], ..., [As type]

4.2.13 Auto Completion

A very useful tool is the Auto Completion function.
Example with the SecondProgram code:



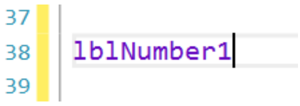
```
lblN
```

lblNumber1 As Label

Let us write lblN.

All variables, views and property names beginning with the letters already written are shown in a popup menu with the online help for the highlighted variable, view or property name.

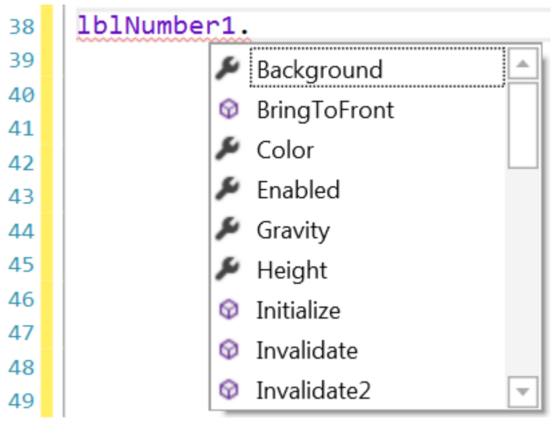
To choose lblNumber1 press Return.



```
lblNumber1
```

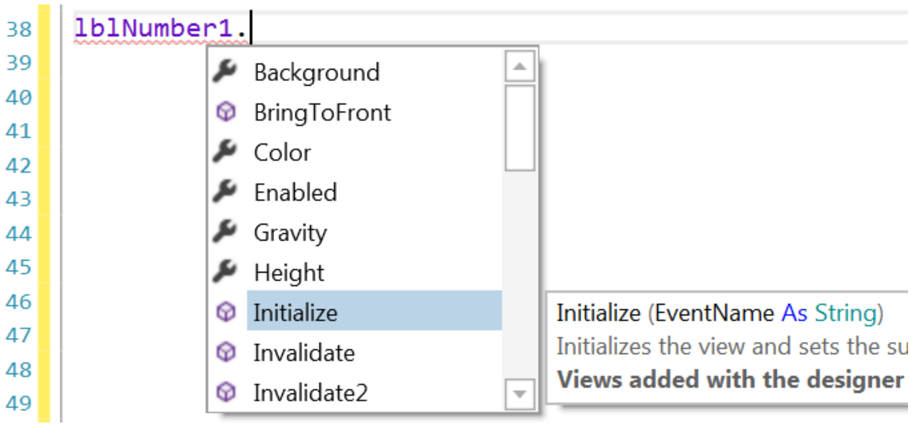
The selected name is completed.

To choose lblNumber2 double click on it or press the down arrow and press Return.



```
lblNumber1.
```

After pressing "." all properties and methods of the view are displayed in a popup menu.



```
lblNumber1.
```

Initialize (EventName As String)
Initializes the view and sets the su
Views added with the designer

When selecting an item, the internal help is displayed

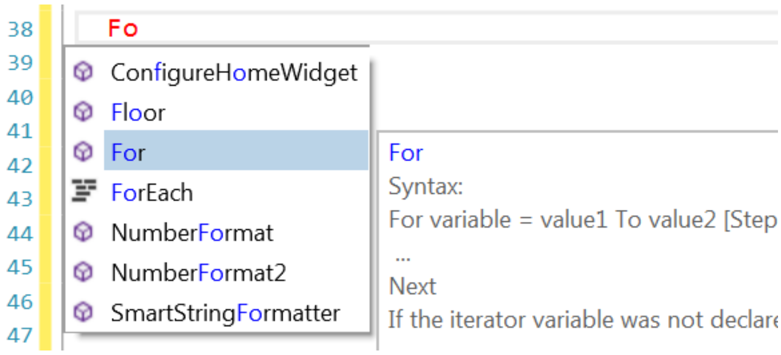
Pressing on the up / down arrows selects the previous or next item with its help.

Pressing a character updates the list and shows the parameter beginning with that character.

Structures are also completed.

Examples:

For / Next



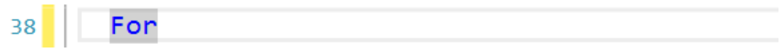
```
Fo
```

For
Syntax:
For variable = value1 To value2 [Step
...
Next
If the iterator variable was not declare

Tpye Fo

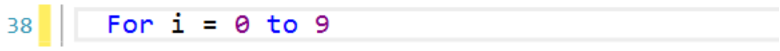
You get For with the help.

Press Return.



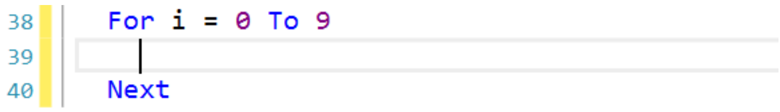
```
For
```

For is completed.



```
For i = 0 to 9
```

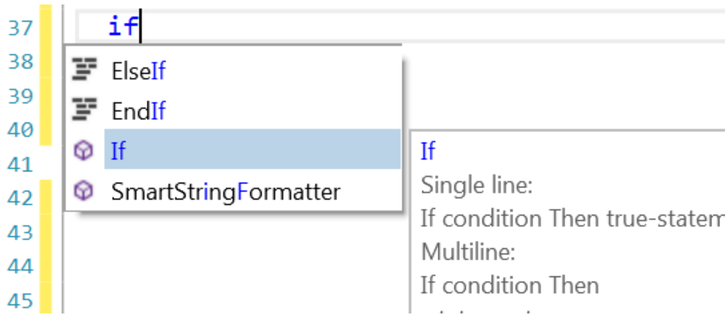
Write the rest of the instruction.
And press Return.



```
For i = 0 To 9
    
Next
```

Next is automatically added and the cursor is in the next line idented.

If / Then



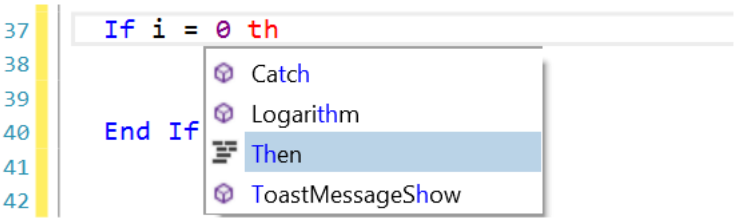
```
if
```

If
Single line:
If condition Then true-statem
Multiline:
If condition Then

Type 'if'.

You get If with the help.

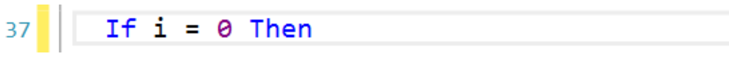
Press Return and continue typing like in the example.



```
If i = 0 th



End If
```

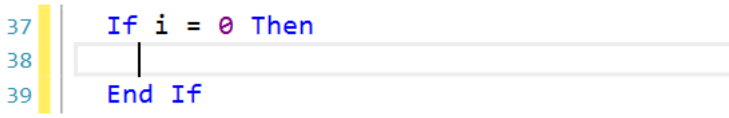
After th you get Then with its help.

Press Return.



```
If i = 0 Then
```

And press Return again.



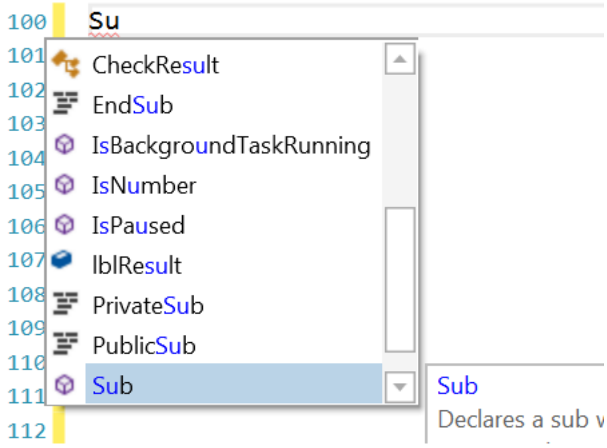
```
If i = 0 Then
    
End If
```

`End If` is automatically added
and the cursor is in the next line idented.

The best way to learn it is to 'play' with it.

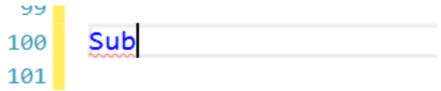
Another very powerful Autocomplete, function allows you to create event subroutines.

In the example below we want to create the Click event for the bntOK button.
Write ’Su’ and the Auto Completion displays all keywords containing the two characters.



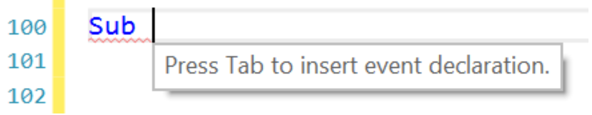
```
Su
```

Press Return to select Sub.



```
Sub
```

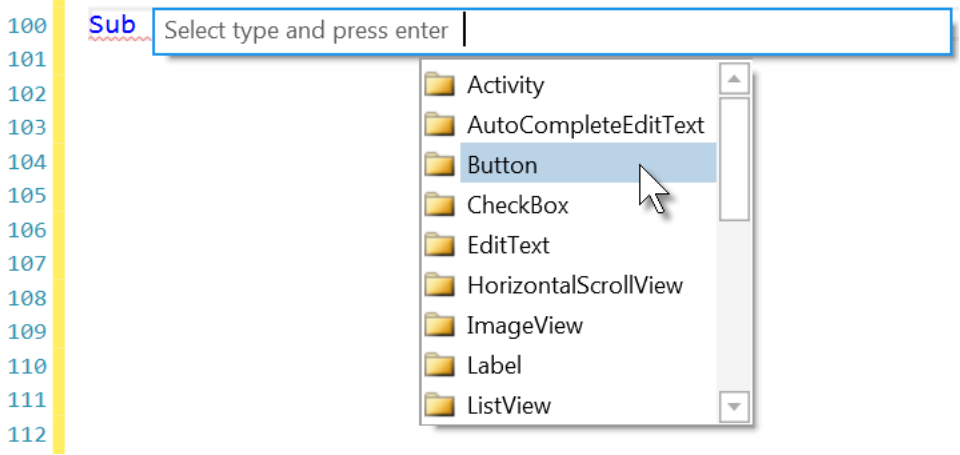
Press blank.



```
Sub
```

Press Tab to insert event declaration.

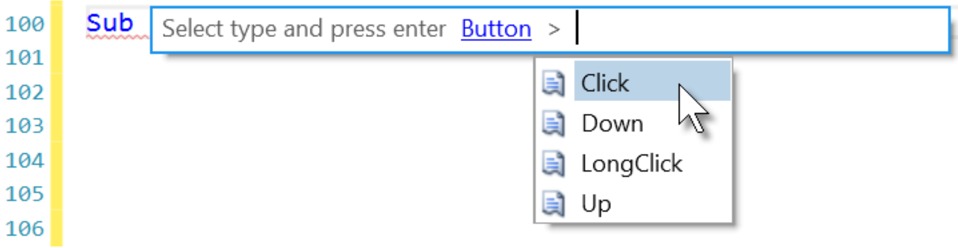
Press Tab and select the view type, select Button.



```
Sub
```

Select type and press enter

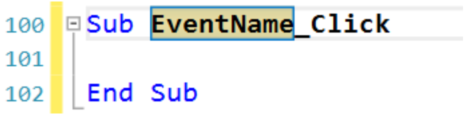
All events for a Button are displayed, select Click.



```
Sub
```

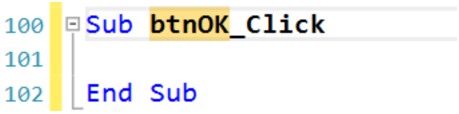
Select type and press enter Button >

The subroutine frame is generated.



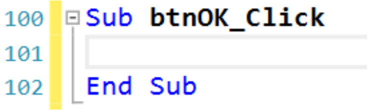
```
Sub EventName_Click

End Sub
```

Modify 'EventName' to the event name of the button, in our example `btnOK`.



```
Sub btnOK_Click

End Sub
```

Press Return and the routine is ready.



```
Sub btnOK_Click

End Sub
```

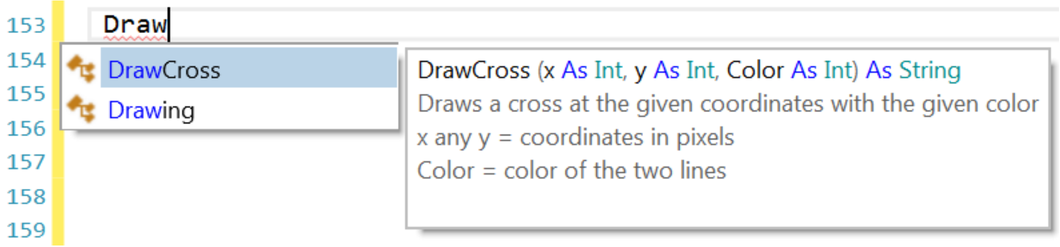
4.2.14 Built in documentation

Another useful function is the built-in documentation.

Comments above subs, such as:

```
'Draws a cross at the given coordinates with the given color
'x any y = coordinates in pixels
'Color = color of the two lines
Sub DrawCross(x As Int, y As Int, Color As Int)
      Private d = 3dip As Int

      cvsLayer(2).DrawLine(x - d, y, x + d, y, Color, 1)
      cvsLayer(2).DrawLine(x, y - d, x, y + d, Color, 1)
End Sub
```

Will automatically appear in the auto complete pop-up window:



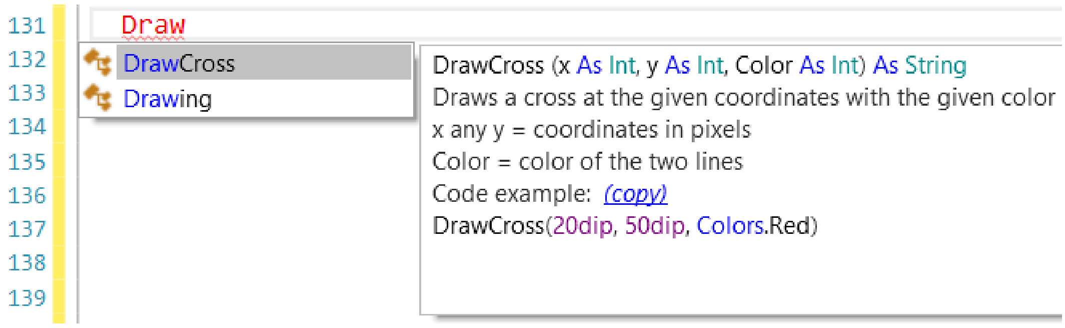
```
Draw
```

DrawCross (x As Int, y As Int, Color As Int) As String
Draws a cross at the given coordinates with the given color
x any y = coordinates in pixels
Color = color of the two lines

If you want to add a code example you can use <code> </code> tags:

```
'Draws a cross at the given coordinates with the given color
'x any y = coordinates in pixels
'Color = color of the two lines
'Code example: <code>
'DarwCross(20dip, 50dip, Colors.Red)
'</code>
Sub DrawCross(x As Int, y As Int, Color As Int)
   Private d = 3dip As Int

   cvsLayer(2).DrawLine(x - d, y, x + d, y, Color, 1)
   cvsLayer(2).DrawLine(x, y - d, x, y + d, Color, 1)
End Sub
```

The code will be syntax highlighted:



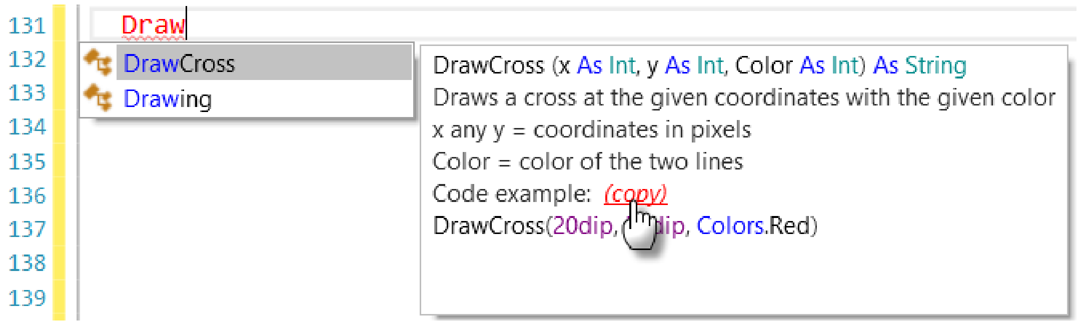
```
Draw
```

DrawCross (x As Int, y As Int, Color As Int) As String
Draws a cross at the given coordinates with the given color
x any y = coordinates in pixels
Color = color of the two lines
Code example: (copy)
DrawCross(20dip, 50dip, Colors.Red)

4.2.14.1 Copy code examples

You can copy the code example in your code.

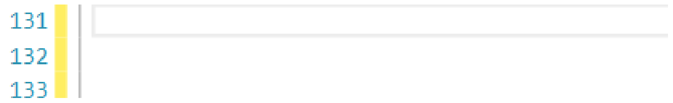
When hovering over (copy) you can copy the code example to the clipboard.



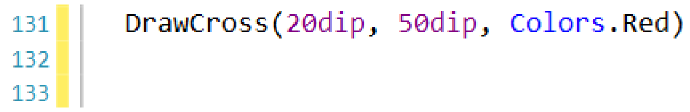
DrawCross (x As Int, y As Int, Color As Int) As String
Draws a cross at the given coordinates with the given color
x any y = coordinates in pixels
Color = color of the two lines
Code example: (copy)

```
DrawCross(20dip, 50dip, Colors.Red)
```

Remove `Draw`



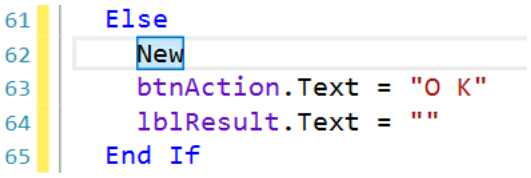
And copy.



```
DrawCross(20dip, 50dip, Colors.Red)
```

4.2.15 Jump to a subroutine

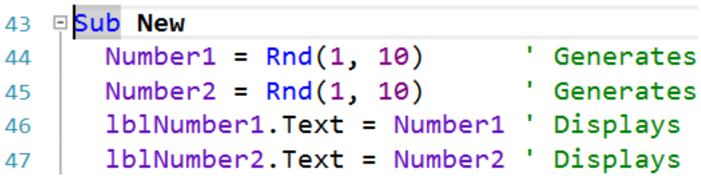
Sometimes it is useful to jump from a subroutine call to the subroutine definition.
This can easily be done :



```
Else
    New
    btnAction.Text = "O K"
    lblResult.Text = ""
End If
```

Select the text of the subroutine call.

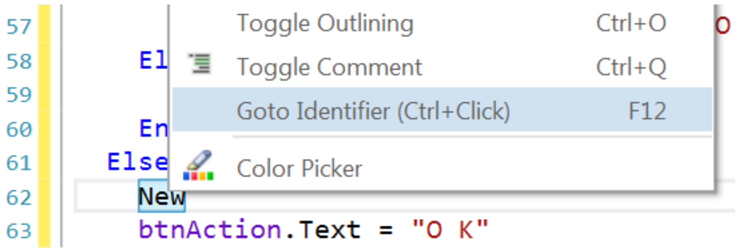
Press Ctrl and Click.



```
Sub New
    Number1 = Rnd(1, 10)        ' Generates
    Number2 = Rnd(1, 10)        ' Generates
    lblNumber1.Text = Number1 ' Displays
    lblNumber2.Text = Number2 ' Displays
```

And you are there.

Another method.

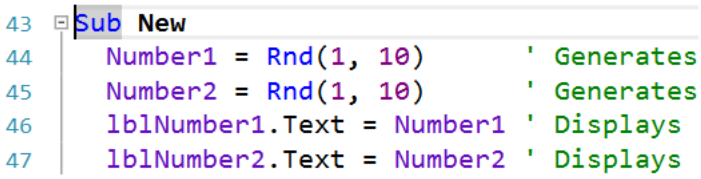


```
    El
    En
Else
    New
    btnAction.Text = "O K"
```

Select the text of the subroutine call.

Right click on the selected text.

Click on `Goto Identifier`.

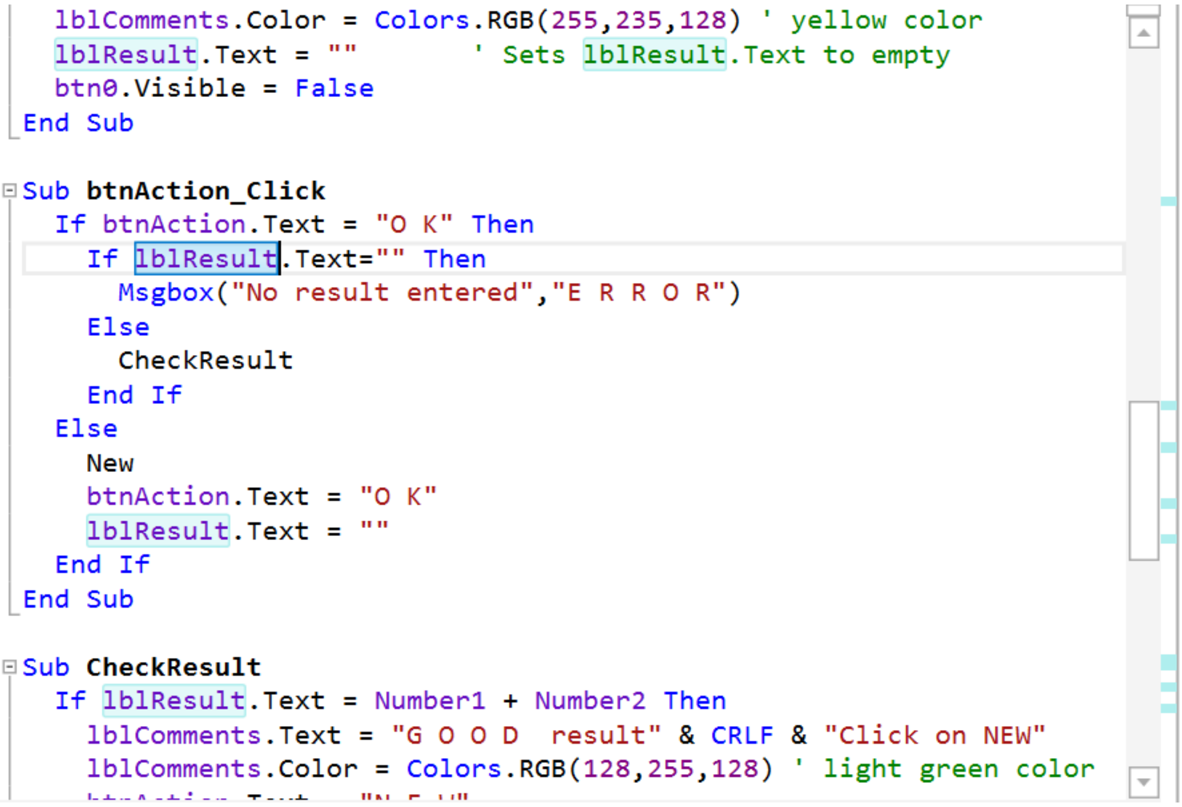


```
Sub New
    Number1 = Rnd(1, 10)        ' Generates
    Number2 = Rnd(1, 10)        ' Generates
    lblNumber1.Text = Number1 ' Displays
    lblNumber2.Text = Number2 ' Displays
```

And you are there.

4.2.16 Highlighting occurrences of words

When you select a single word, it is highlighted in dark blue and all the other occurrences in the code are highlighted in light blue and in the scroll view on the right side.
With the slider you can move up or down the code to go to the other occurrences.



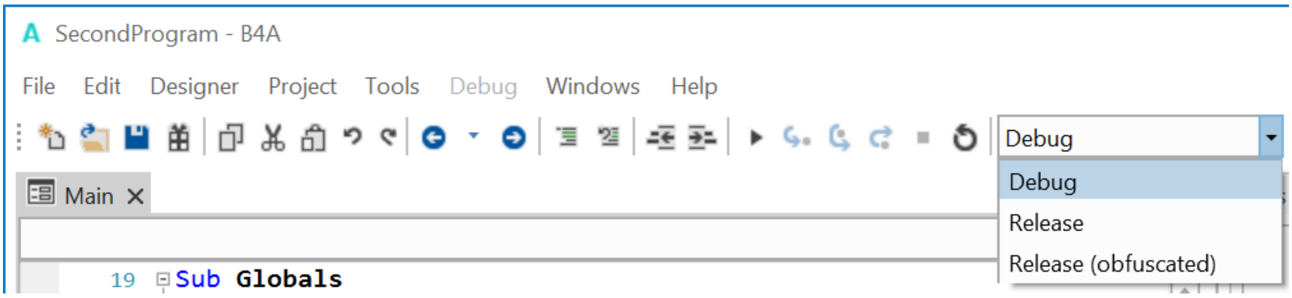
```
  lblComments.Color = Colors.RGB(255,235,128) ' yellow color
  lblResult.Text = ""         ' Sets lblResult.Text to empty
  btn0.Visible = False
End Sub

Sub btnAction_Click
  If btnAction.Text = "O K" Then
    If lblResult.Text="" Then
      Msgbox("No result entered","E R R O R")
    Else
      CheckResult
    End If
  Else
    New
    btnAction.Text = "O K"
    lblResult.Text = ""
  End If
End Sub

Sub CheckResult
  If lblResult.Text = Number1 + Number2 Then
    lblComments.Text = "G O O D  result" & CRLF & "Click on NEW"
    lblComments.Color = Colors.RGB(128,255,128) ' light green color
    btnAction.Text = "N E W"
```

4.2.17 Compiler mode

Besides the toolbar there is a drop down list to select the compiler mode.

Thes are:

- Debug
- Release
- Release (obfuscated)



```
Sub Globals
```

Debugging is explained in detail in the Debugging chapter.

4.2.17.1 Release and Release (obfuscated) modes

To distribute your project you must compile it with:

- Release
  The debugger code will not be added to the apk file.
- Release (obfuscated)
  The debugger code will not be added to the apk file,
  but the program file will be modified. See below.

During compilation B4A generates Java code which is then compiled with the Java compiler and converted to Dalvik (Android byte code format).
There are tools that allow decompilation of Dalvik byte code into Java code.

The purpose of obfuscation is to make the decompiled code less readable, harder to understand and make it more difficult to extract strings like developer account keys.

It is important to understand how the obfuscator works.
The obfuscator does two things:

Strings obfuscation
Any string written in Process_Globals sub (and only in this sub) will be obfuscated, making it much harder to extract important keys. The strings are deobfuscated at runtime.
Note that several keys are used during obfuscation including the package name, version name and version code. Modifying these values with the manifest editor will break the deobfuscation process.

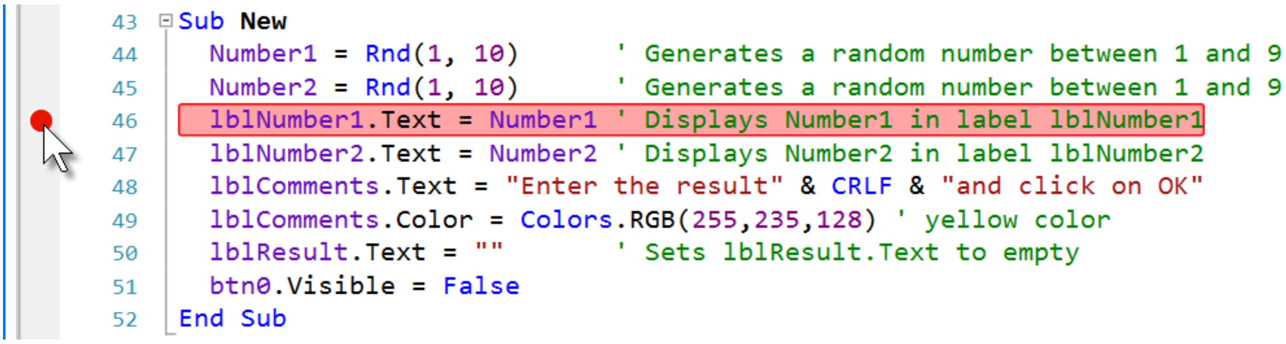
Variables renaming
The names of global variables and subs are converted to meaningless strings. Local variables are not affected as their names are lost anyway during the compilation.
The following identifiers are **not** renamed:
- Identifiers that contain an underscore (required for the events handlers).
- Subs that appear in CallSub statements. When a sub name appears as a static string, the identifier be kept as it is.
- Designer views names.

Tip: If, for some reason, you wish to prevent obfuscation of an identifier, include an underscore character in the name.

A file named ObfuscatorMap.txt will be created under the Objects folder. This file maps the original identifiers names to the obfuscated names. This mapping can be helpful in analysing crash reports.

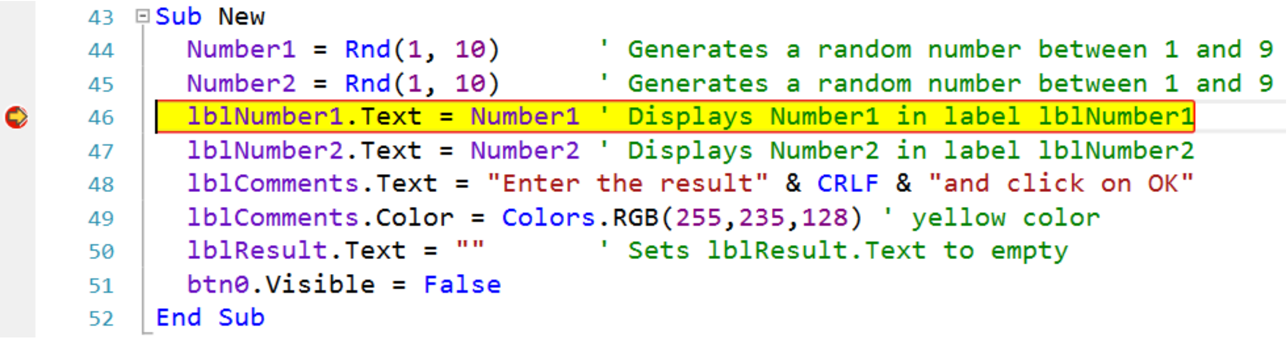
4.2.18 Breakpoints

Clicking on a line in the left margin adds a breakpoint. When the program is running it stops at the first breakpoint.
Breakpoints are ignored in Globals, Process_Globals and Activity_Pause.
The IDE behaves differently depending on the debug mode. The examples below are for the *rapid debug* mode.



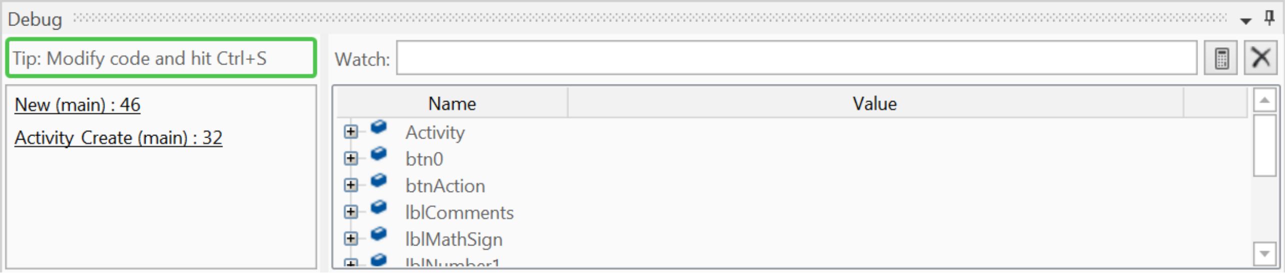
```
43 Sub New
44     Number1 = Rnd(1, 10)        ' Generates a random number between 1 and 9
45     Number2 = Rnd(1, 10)        ' Generates a random number between 1 and 9
46     lblNumber1.Text = Number1  ' Displays Number1 in label lblNumber1
47     lblNumber2.Text = Number2  ' Displays Number2 in label lblNumber2
48     lblComments.Text = "Enter the result" & CRLF & "and click on OK"
49     lblComments.Color = Colors.RGB(255,235,128) ' yellow color
50     lblResult.Text = ""         ' Sets lblResult.Text to empty
51     btn0.Visible = False
52 End Sub
```

Run the program, the program stops at the breakpoint and the IDE looks like below. The line where the program stops is highlighted in yellow.

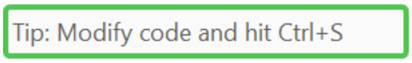
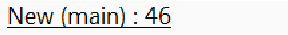
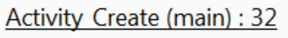


```
43 Sub New
44     Number1 = Rnd(1, 10)        ' Generates a random number between 1 and 9
45     Number2 = Rnd(1, 10)        ' Generates a random number between 1 and 9
46     lblNumber1.Text = Number1  ' Displays Number1 in label lblNumber1
47     lblNumber2.Text = Number2  ' Displays Number2 in label lblNumber2
48     lblComments.Text = "Enter the result" & CRLF & "and click on OK"
49     lblComments.Color = Colors.RGB(255,235,128) ' yellow color
50     lblResult.Text = ""         ' Sets lblResult.Text to empty
51     btn0.Visible = False
52 End Sub
```

At the bottom of the IDE you find other information.

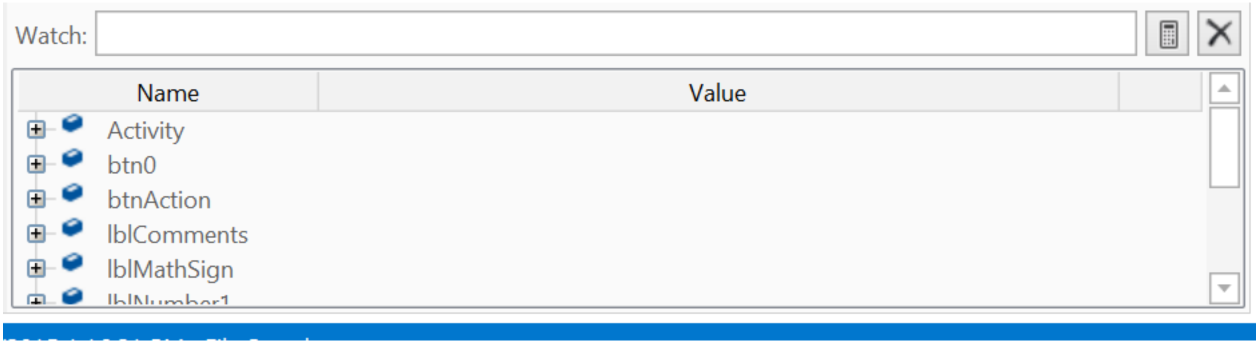


The Debugger is connected. In the left part of the Debugger window we find:

-  Tip: Modify code and hit Ctrl+S — A button to update the program after a code modification.
-  New (main) : 46 — The name of the routine where the Debugger stopped the program. New in the module Main in line 46.
-  Activity Create (main) : 32 — Caller of the “New” routine:
  Activity_Create in the module Main routine in line 32.

Clicking on these links moves the cursor to the given line.

In the right part of the Debugger window we find the list of all Views and Variables with their values.



Watch:

Name	Value
Activity	
btn0	
btnAction	
lblComments	
lblMathSign	
lblNumber1	

In the Toolbar, at the top of the IDE the navigation buttons are enabled.

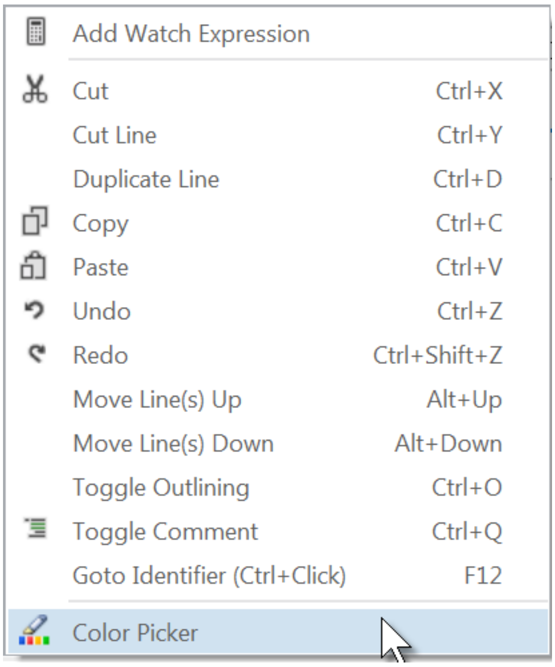
▶	Run the program		Runs the program, no action in Debug (rapid)
	Step In	F8	Executes the next statement.
	Step Out	F9	Leaves the current subroutine.
	Step Over	F10	Steps over the subroutine call.
■	Stop		Stops the program.
↻	Restart	F11	Restarts the program.

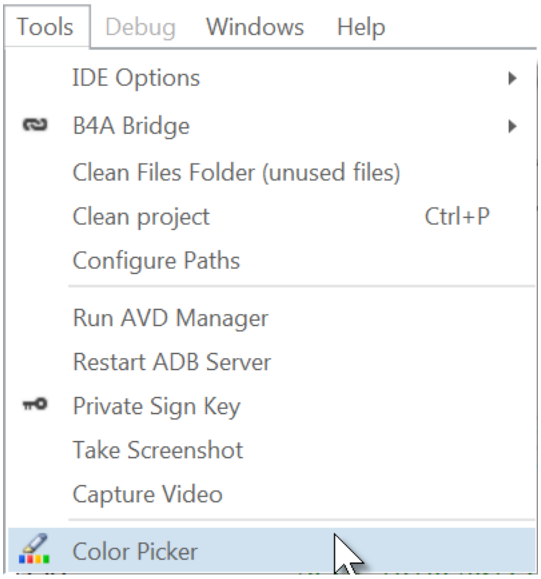
For more details look at Debug (rapid) mode.

4.2.19 Color Picker

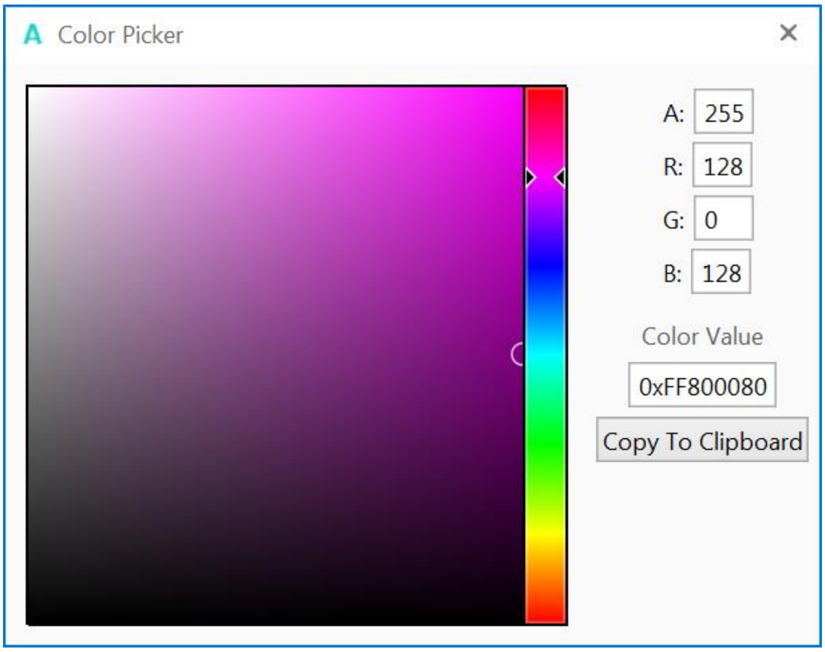
In the code, right click to show the popup menu below.

Or, in the menu Tools.





Click on Color Picker to show the Color Picker.



A: 255
R: 128
G: 0
B: 128

0xFF800080

You can move the cursor in the square and the rectangular areas.

Or enter the A R G B values.

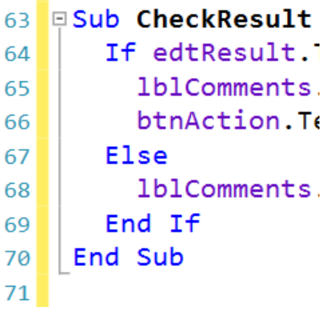
Copy the value to the Clipboard.

You can then paste the value into the code.

4.2.20 Colors in the left side

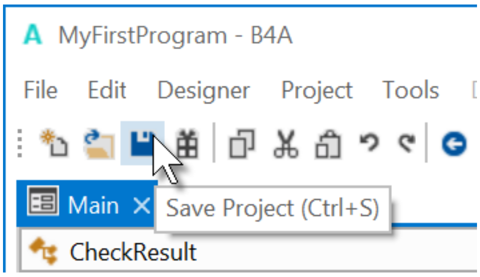
Sometimes, you will see yellow or green vertical lines in the left side od the IDE.

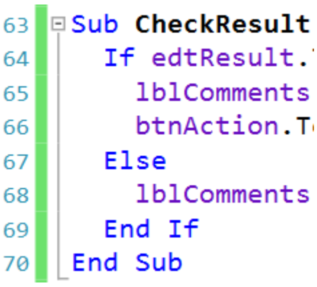
As soon as you modify a line it will be marked with a yellow vertical line on the right of the line number meaning that this line was modified.



```
Sub CheckResult
  If edtResult.
    lblComments
    btnAction.Te
  Else
    lblComments
  End If
End Sub
```

If we click on 💾 to save the project the yellow lines become green showing a modified code but already saved. You can also press Ctrl + S to save the project.





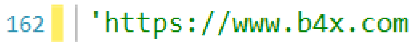
```
Sub CheckResult
  If edtResult.
    lblComments
    btnAction.T
  Else
    lblComments
  End If
End Sub
```

If we leave the IDE and load the project again the green lines disappear.

4.2.21 URLs in comments and strings are ctrl-clickable

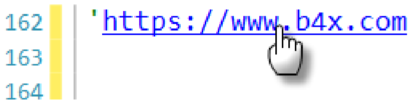
URLs in comments and strings are ctrl-clickable.

In a comment:



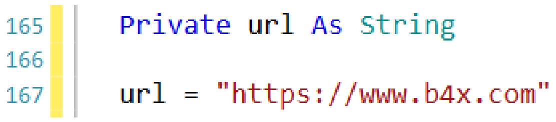
```
'https://www.b4x.com
```

If the cursor is on the line and you press Ctrl the url is highlighted in blue and if you click on it the url it is executed. Hovering over the line with Ctrl pressed does also highlight the url.



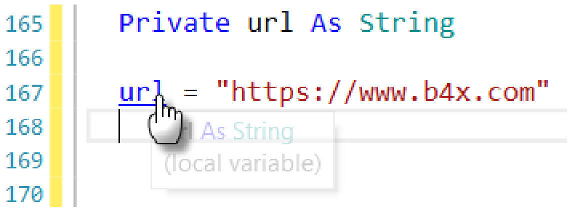
```
'https://www.b4x.com
```

In a String:



```
Private url As String

url = "https://www.b4x.com"
```

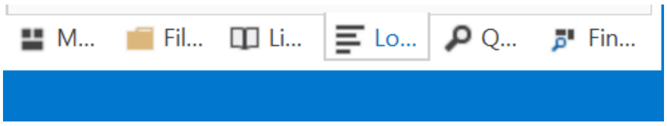
The cursor must be over the String variable and not over text.



```
Private url As String

url = "https://www.b4x.com"
```

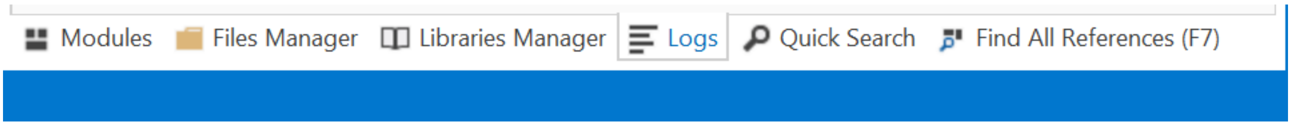
4.3 Tabs

There are 6 tabs at the bottom right corner of the IDE that displays different windows.



The short version.

The wide version.



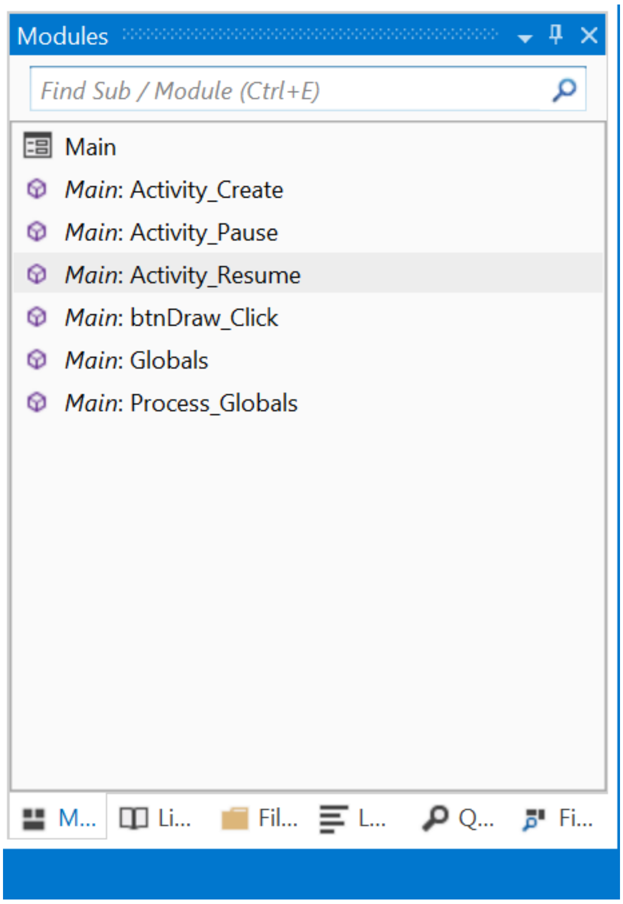
The 6 Tabs are:

- Modules
- Files Manager
- Libraries Manager
- Logs
- Quick Search
- Find All References

Each Tab has its own window.
By default they are displayed in the Tab area on the right side of the IDE, only one at the same time.
These windows can be closed, hidden or floating, see next chapter.

4.3.1 Floating Tab windows

When you start the default IDE all Tab windows are docked in the Tab area.



Modules

Find Sub / Module (Ctrl+E)

Main

Main: Activity_Create

Main: Activity_Pause

Main: Activity_Resume

Main: btnDraw_Click

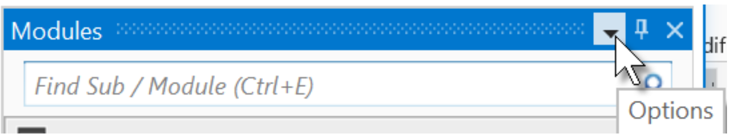
Main: Globals

Main: Process_Globals

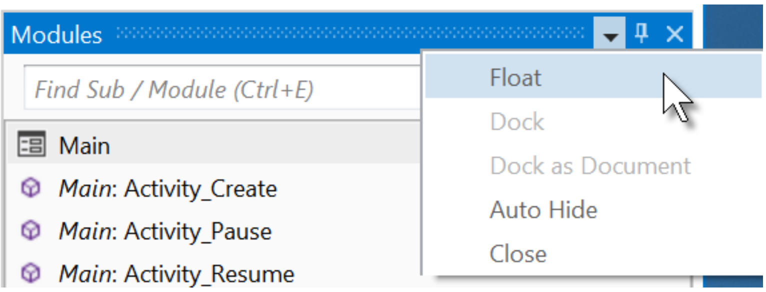
You can set each Tab window as a separate floating window.

4.3.2 Float ▾

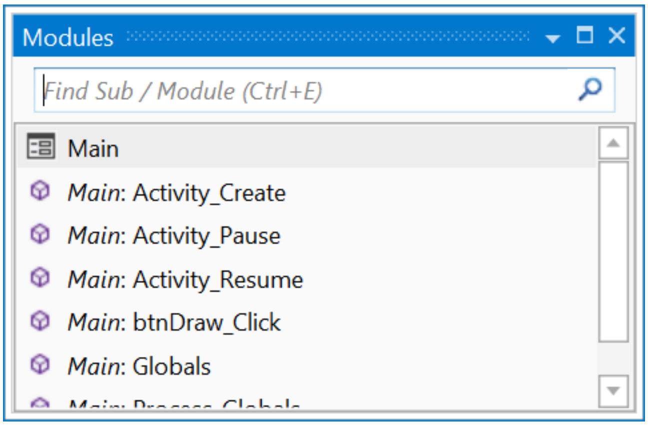
To set the Modules Tab window to floating click in the title on ▾.



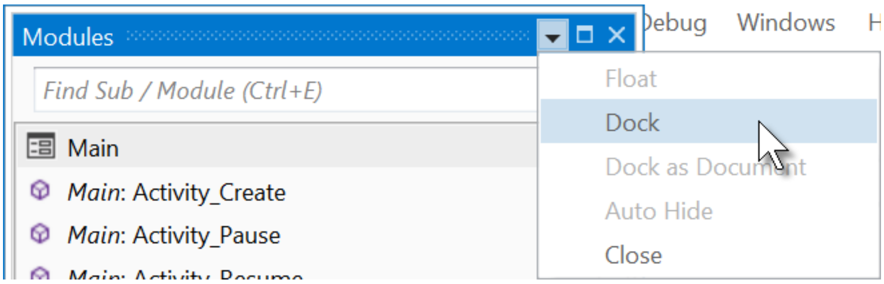
Click on Float.



The Modules Tab Window is now floating, you can place it where you want on the screen even on a second monitor.

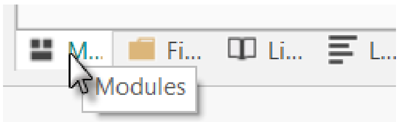


To dock it back to the Tab area click on Dock.

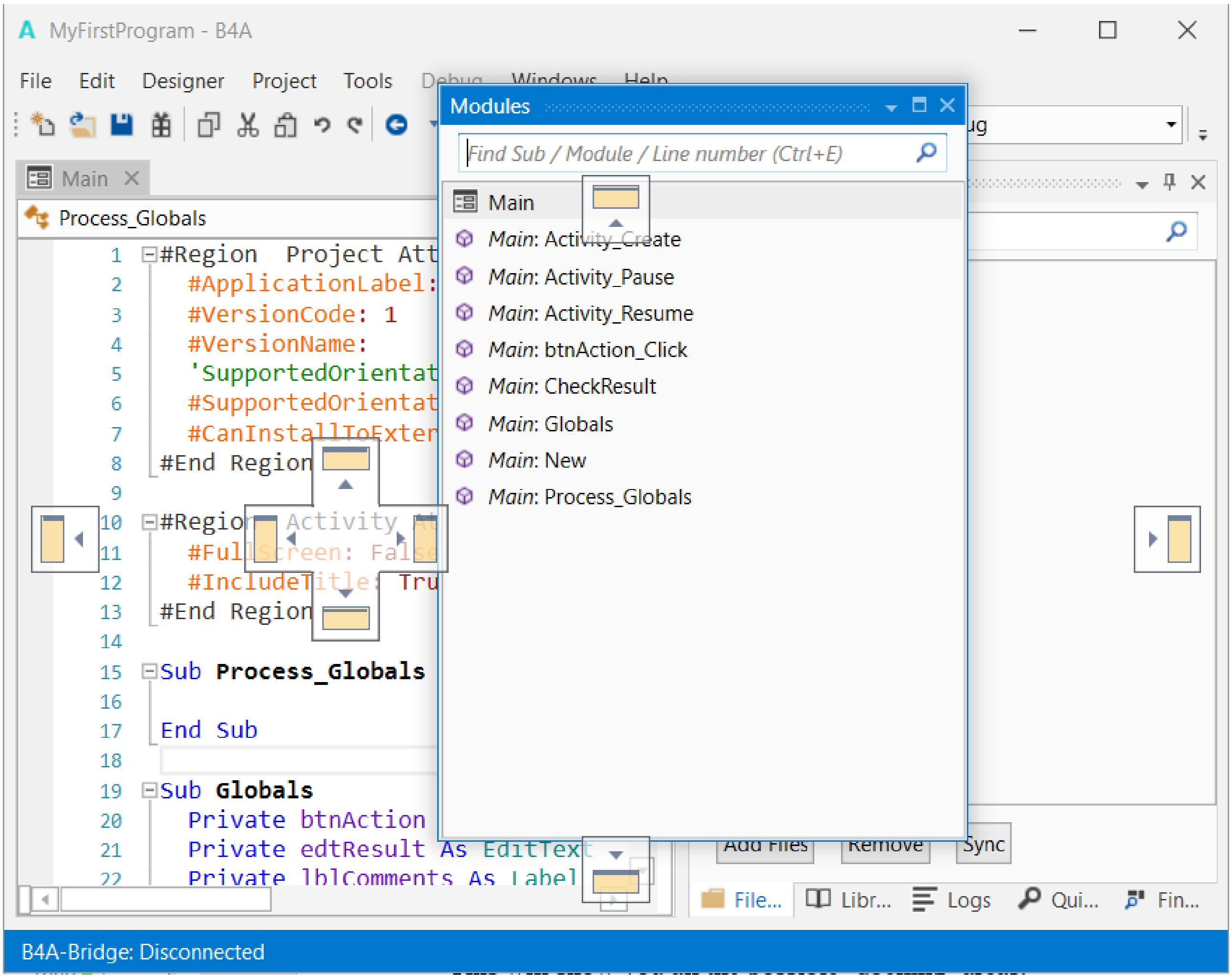


To show the Tabs again click either on Dock in the Options or on Reset in the IDE Window menu.

You can also click on a Tab and while maintaining the mouse down, move the Tab.

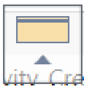


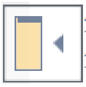
This will show you all the possible 'docking' areas.

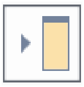


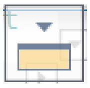
```
#Region  Project Att
   #ApplicationLabel:
   #VersionCode: 1
   #VersionName:
   'SupportedOrientat
   #SupportedOrientat
   #CanInstallToExter
#End Region

#Regio  Activity At
   #FullScreen: False
   #IncludeTitle: Tru
#End Region

Sub Process_Globals

End Sub

Sub Globals
   Private btnAction
   Private edtResult As EditText
   Private lblComments As Label
```

Docking areas:

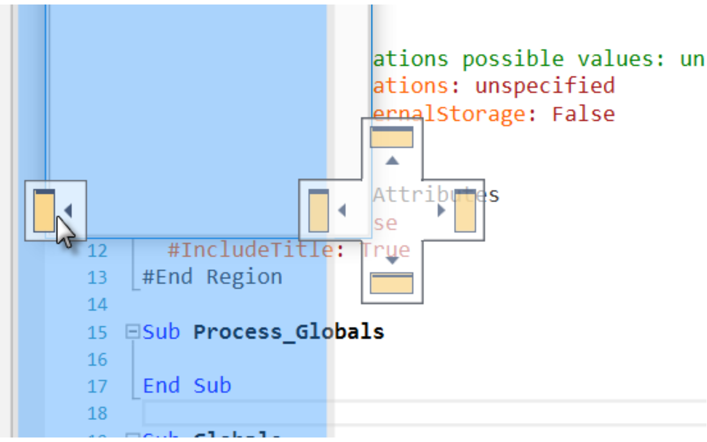

Top


Left


Right

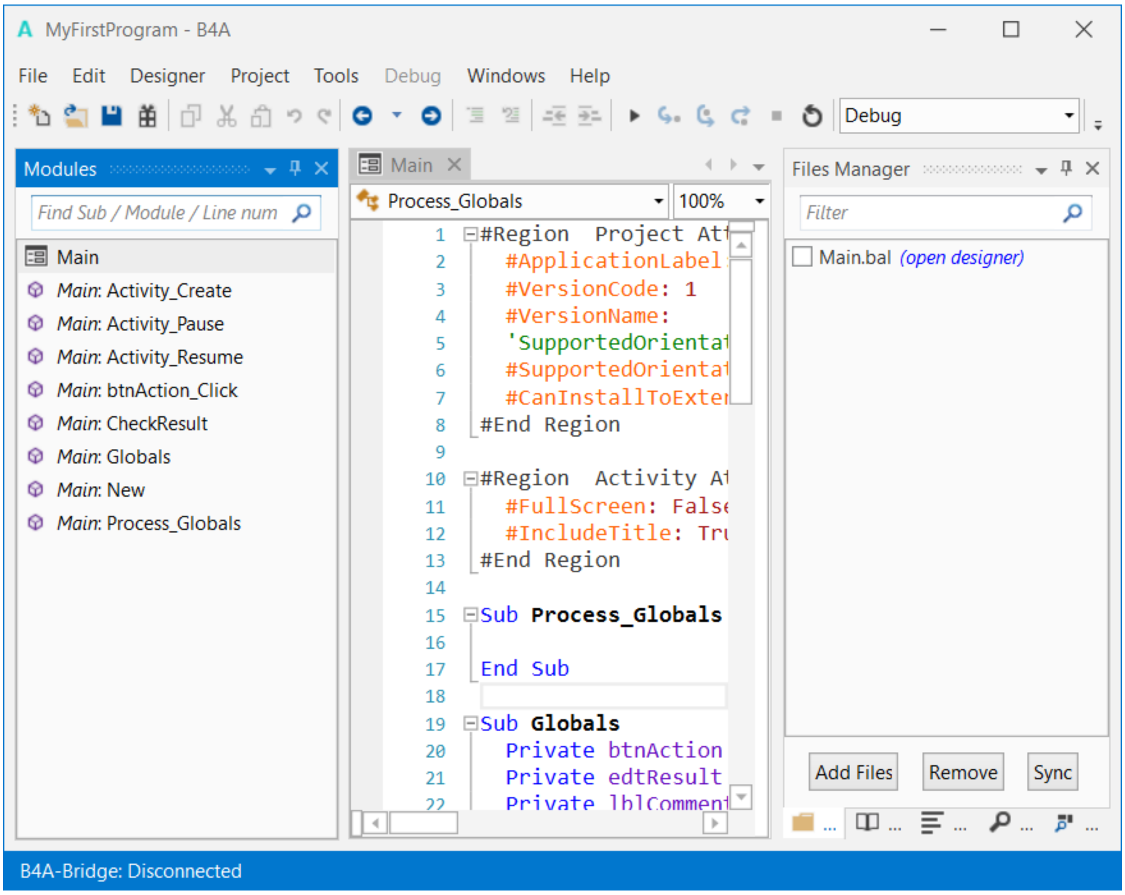

Bottom

If you mouve the mouse onto one of the docking area symbol, the Tab window will be either on top, on the left, the right or on the bottom.



```
ations possible values: un
ations: unspecified
ernalStorage: False

Attributes
se
  #IncludeTitle: True
#End Region

Sub Process_Globals

End Sub
```

And the result.



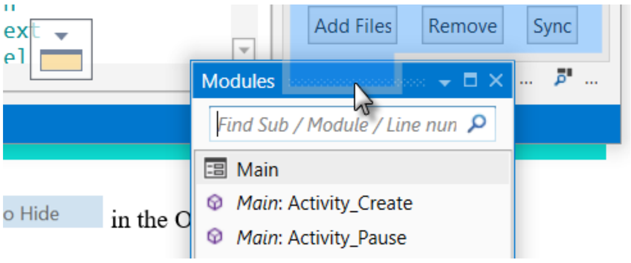
Main
Main: Activity_Create
Main: Activity_Pause
Main: Activity_Resume
Main: btnAction_Click
Main: CheckResult
Main: Globals
Main: New
Main: Process_Globals

□ Main.bal *(open designer)*

```
#Region  Project At
    #ApplicationLabel
    #VersionCode: 1
    #VersionName:
    'SupportedOrienta
    #SupportedOrienta
    #CanInstallToExte
#End Region

#Region  Activity At
    #FullScreen: False
    #IncludeTitle: Tru
#End Region

Sub Process_Globals

End Sub

Sub Globals
    Private btnAction
    Private edtResult
    Private lblComment
```

B4A-Bridge: Disconnected

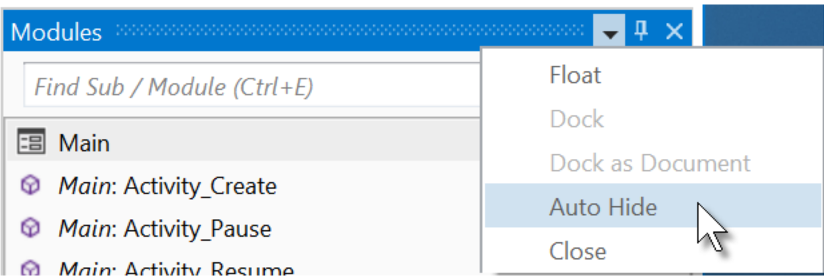
To bring it back to the Tabs, click on the window title and move it back to the Tabs.

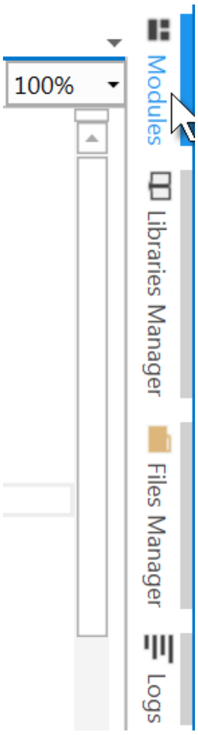


Main
Main: Activity_Create
Main: Activity_Pause

4.3.3 Auto Hide 🖈

Click on 🖈 in the title or click on Auto Hide in the Options.

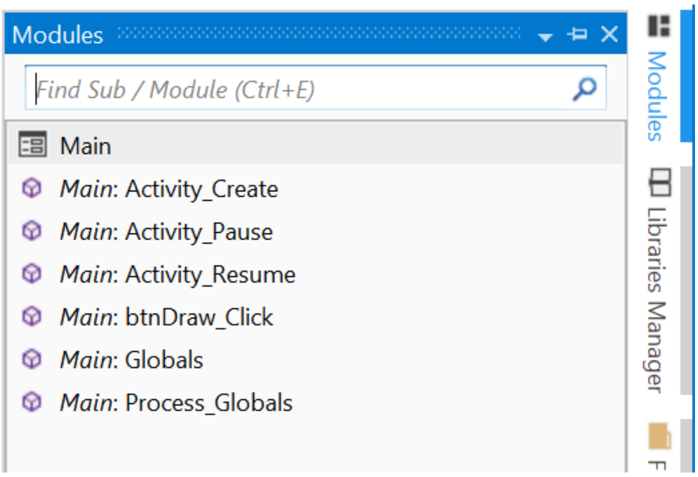




The Tabs move from the bottom of the screen vertically on the right side of the screen and the Tab window is hidden.

Hovering over a Tab highlights it in blue.

Click on a Tab to show it.

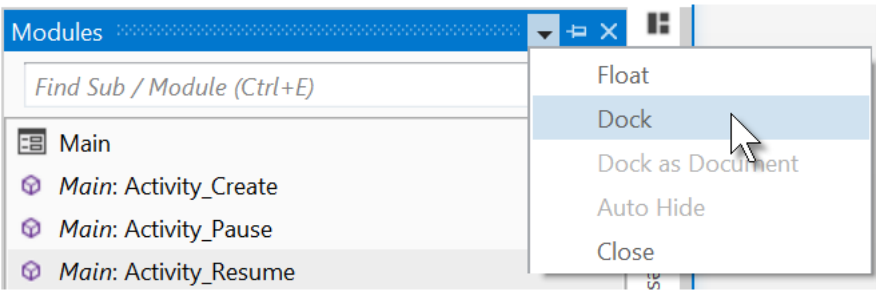


The selected Tab is displayed.

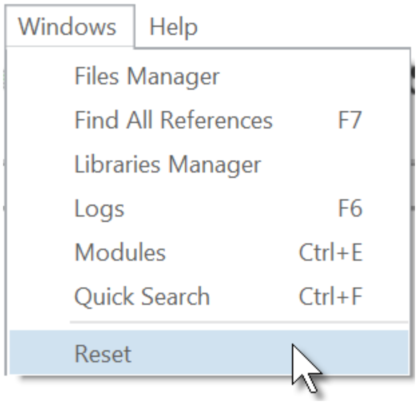
As soon as you click on something in the IDE the Tab is hidden again.

To move the Tabs back to the lower right corner:

Click on Dock in the Options.



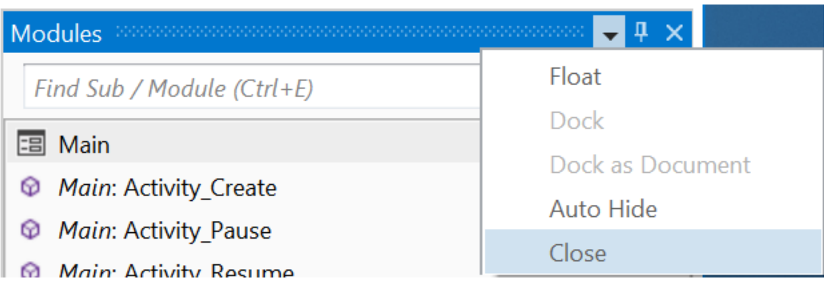
Or click on Reset in the IDE Windows menu.

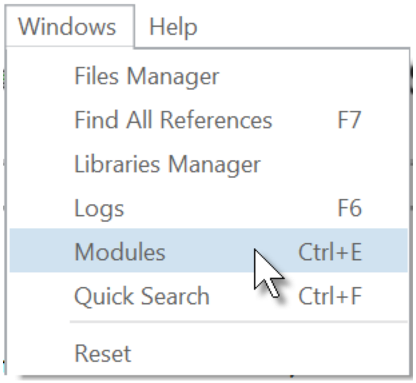


4.3.4 Close

You can close a window, hide it.

Click on ✕ in the title or on Close in the Options.

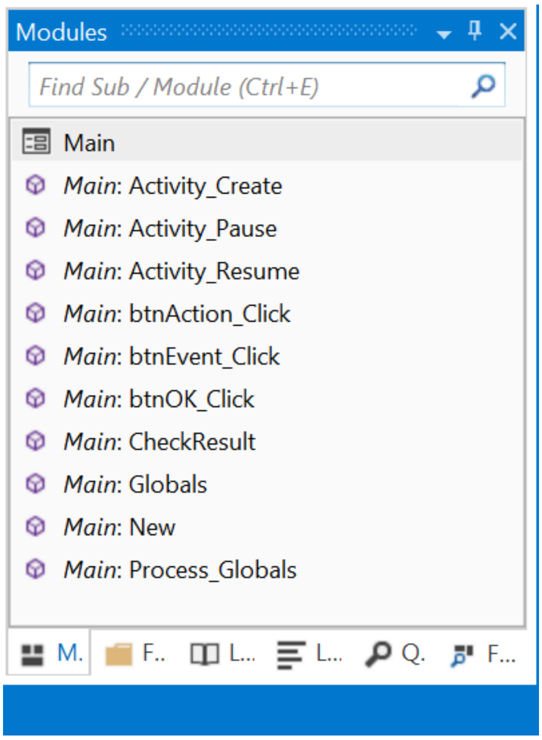




To show it again, in the Windows menu click on the module name you want to show, Modules in our example.

4.3.5 Modules and subroutine lists Modules

All the modules of the project and all subroutines of the selected module are listed in the Modules window. The picture below has been reduced in height.


Modules

Find Sub / Module (Ctrl+E)

- Main
- *Main*: Activity_Create
- *Main*: Activity_Pause
- *Main*: Activity_Resume
- *Main*: btnAction_Click
- *Main*: btnEvent_Click
- *Main*: btnOK_Click
- *Main*: CheckResult
- *Main*: Globals
- *Main*: New
- *Main*: Process_Globals

Find Sub / Module (Ctrl + E)

Module list on top.
Clicking on a module shows its code in the code area.

Find Sub Tool (Ctrl + E)	see below
Find All References (F7)	see below

Subroutine list of the selected module.
Clicking on a subroutine shows its code in the middle of the code area.

In the IDE, in the bottom right corner.

To show a hidden module, click on the module name in the module list.

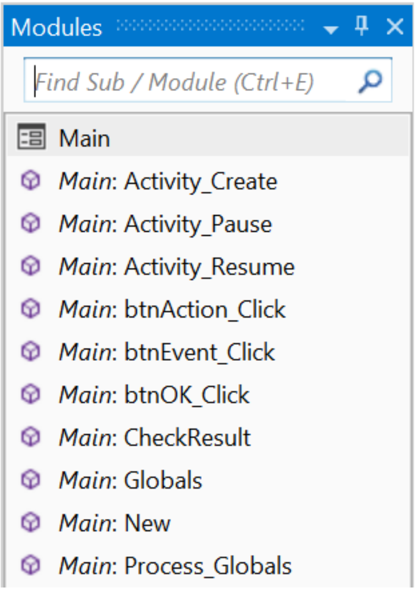
4.3.5.1 Find Sub / Module (Ctrl + E)

The *Find Sub / Module* function is a search engine, on the Top of the Modules Tab, to find subroutines or Modules with a given name or with a given part of the name.

You can press Ctrl + E in the code to select the Modules Tab with the *Find Sub / Module* function.

Example with the code of the SecondProgram example.

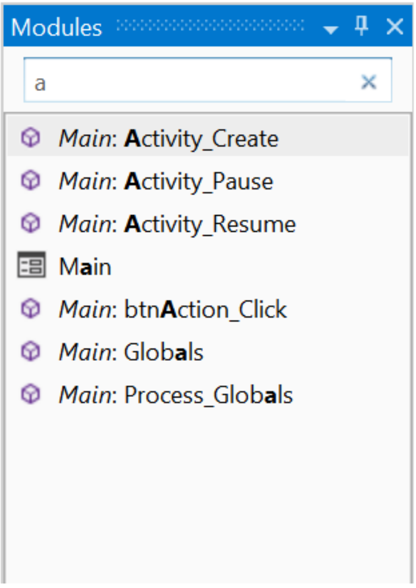
No text



Main
Main: Activity_Create
Main: Activity_Pause
Main: Activity_Resume
Main: btnAction_Click
Main: btnEvent_Click
Main: btnOK_Click
Main: CheckResult
Main: Globals
Main: New
Main: Process_Globals

Shows all modules and all routines of the selected Module.

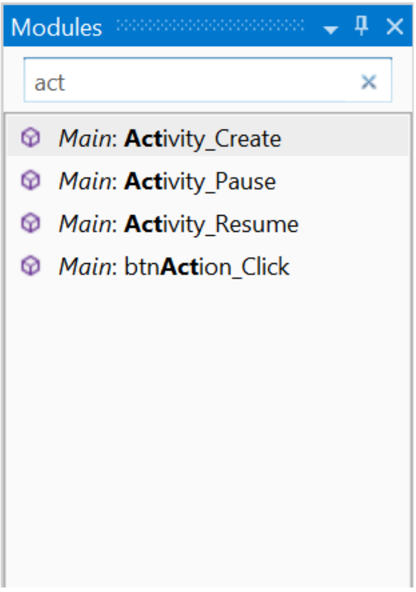
only the character 'a'



Main: **A**ctivity_Create
Main: **A**ctivity_Pause
Main: **A**ctivity_Resume
M**a**in
Main: btn**A**ction_Click
Main: Glob**a**ls
Main: Process_Glob**a**ls

Shows all modules and routines containing 'a'.

text 'act'



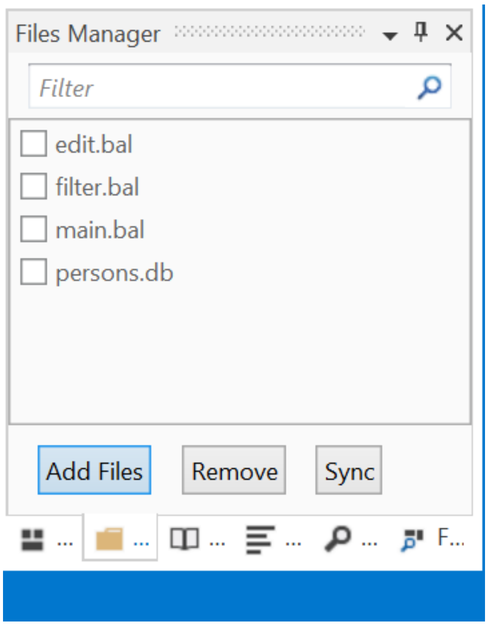
Main: **Act**ivity_Create
Main: **Act**ivity_Pause
Main: **Act**ivity_Resume
Main: btn**Act**ion_Click

Shows all modules and routines containing 'act'.

Clicking on one item shows the code of the selected module or routine, even if it's in another module than the current one.

4.3.6 Files Manager Files Manager

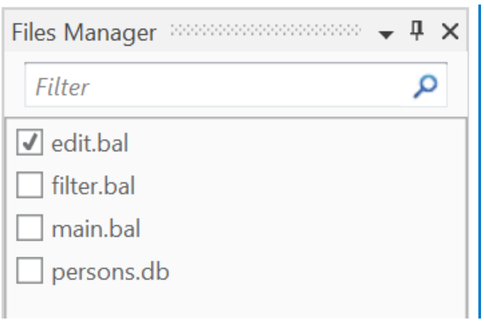
This window lists all the files that have been added to the project.
These files are saved in the ‘Files’ subfolder under your main project folder.
These can be any kind of files: layouts, images, texts, etc.



Click on Add Files to add files to the list.
The files in that subfolder can be accessed from your program by using the reference File.DirAssets.

Or click on Sync to add all the files from the projects Files folder into the File Tab.

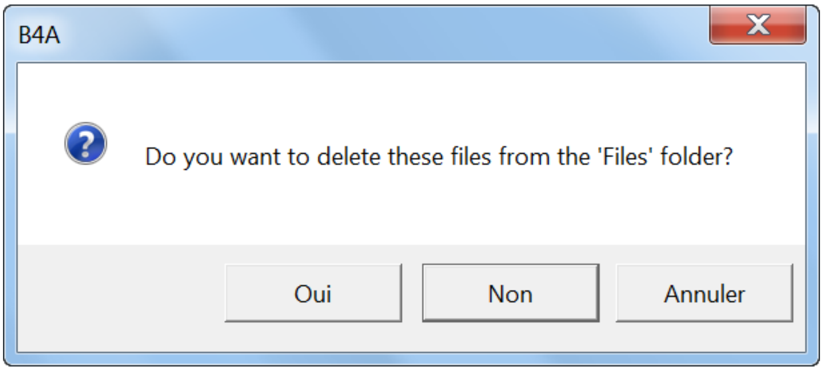
In the IDE, in the bottom right corner.



Checking one or more files enables the

Remove button.

Clicking on this button removes the selected files from the list and, if you want, from the Files folder of the project.



Do you want to delete these files from the 'Files' folder?

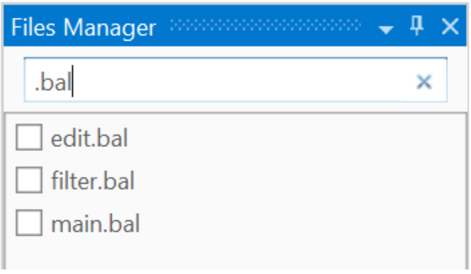
You are asked if you want to delete the files from the 'Files' folder.

Oui	=	Yes
Non	=	No
Annuler	=	Cancel

Make sure to have a copy of the files you remove, because they are removed from the Files folder, but not transferred to the Recycle Bin, which means that they are definitely lost if you don't make a copy.

See chapter Files for file handling.

On top of the Files Manager window you can filter the files list.

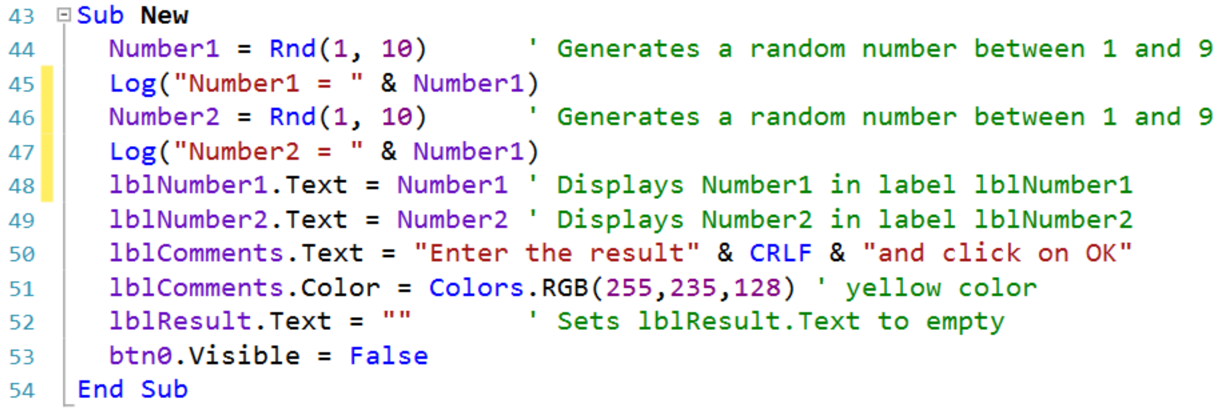


Enter ‘.bal’ to filter all layout files,

4.3.7 Logs Logs

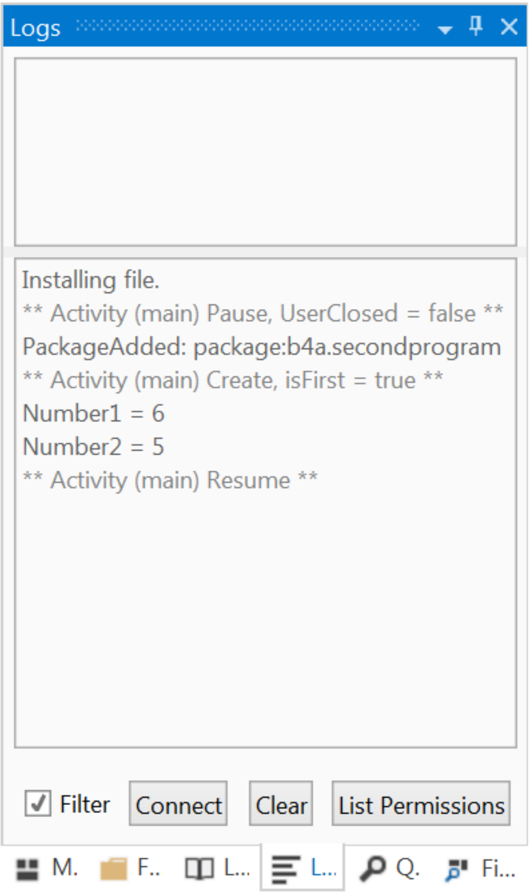
Display of Log comments generated by the program when it is running.

We add the two lines 44 and 46 in the program 'SecondProgram' in the 'New' routine.
The number of the lines may be different from yours.



```
43 Sub New
44     Number1 = Rnd(1, 10)        ' Generates a random number between 1 and 9
45     Log("Number1 = " & Number1)
46     Number2 = Rnd(1, 10)        ' Generates a random number between 1 and 9
47     Log("Number2 = " & Number1)
48     lblNumber1.Text = Number1 ' Displays Number1 in label lblNumber1
49     lblNumber2.Text = Number2 ' Displays Number2 in label lblNumber2
50     lblComments.Text = "Enter the result" & CRLF & "and click on OK"
51     lblComments.Color = Colors.RGB(255,235,128) ' yellow color
52     lblResult.Text = ""         ' Sets lblResult.Text to empty
53     btn0.Visible = False
54 End Sub
```

Run the program.



```
Installing file.
** Activity (main) Pause, UserClosed = false **
PackageAdded: package:b4a.secondprogram
** Activity (main) Create, isFirst = true **
Number1 = 6
Number2 = 5
** Activity (main) Resume **
```

Click on Connect to connect the logger.

The top area of the window shows Compile Warnings see next page.

In the lower area of the window we see the flow of the program.

Installing file.
** Activity (main) Pause, UserClosed = false **
Package Added: package:b4a.secondprogram
** Activity (main) Create, isFirst = true **
Number1 = 6 First log message
Number2 = 5 Second log message
** Activity (main) Resume **

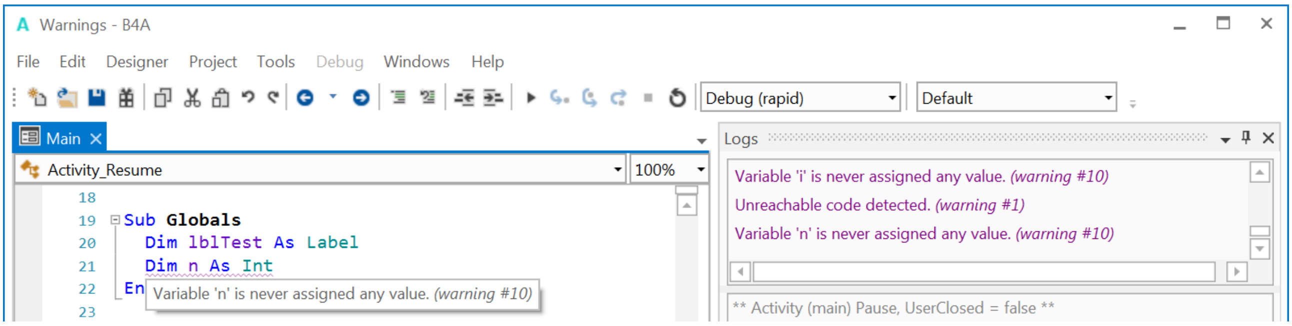
☑ Filter When *Filter* is checked you will only see messages related to your program. When it is unchecked you will see all the messages running in the system. If you are encountering an error and do not see any relevant message in the log, it is worth unchecking the filter option and looking for an error message

Click on Clear to clear the Logs window.

4.3.7.1 Compile Warnings

B4A includes a warning engine. The purpose of the warning engine is to find potential programming mistakes as soon as possible. The examples are from the Warnings project.

The compile-time warnings appear above the logs and in the code itself when hovering with the cursor above the code line.
The code lines which cause a warning are underlined like this `Dim i As Int` .



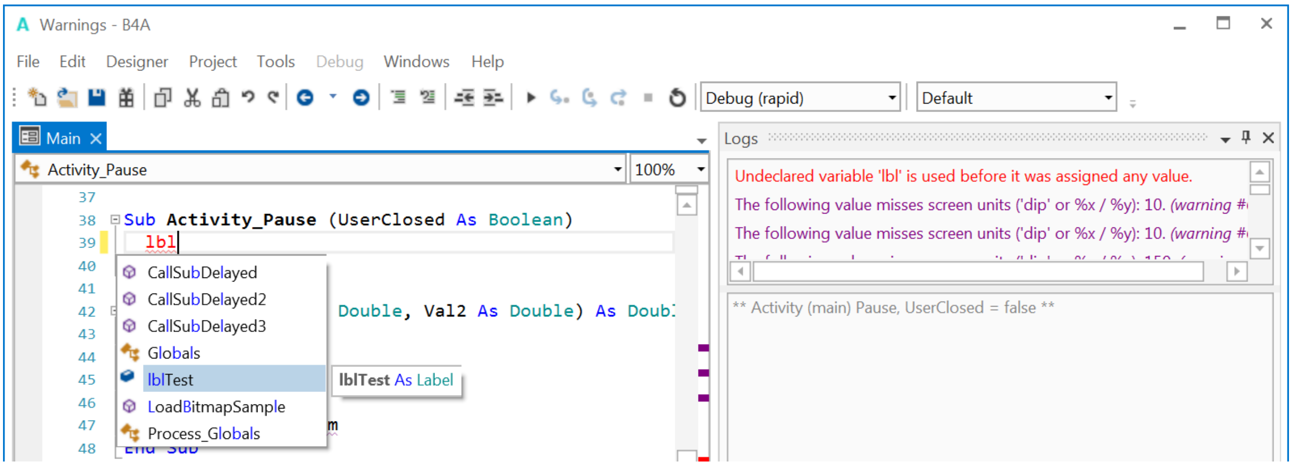
```
Sub Globals
    Dim lblTest As Label
    Dim n As Int
En
```

Variable 'n' is never assigned any value. *(warning #10)*

```
Variable 'i' is never assigned any value. (warning #10)
Unreachable code detected. (warning #1)
Variable 'n' is never assigned any value. (warning #10)
** Activity (main) Pause, UserClosed = false **
```

Clicking on the warning in the list will take you to the relevant code.

The warning engine runs as soon as you type.



```
Sub Activity_Pause (UserClosed As Boolean)
    lbl
                        Double, Val2 As Double) As Doubl
                     m
End Sub
```

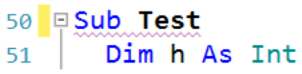
```
Undeclared variable 'lbl' is used before it was assigned any value.
The following value misses screen units ('dip' or %x / %y): 10. (warning #
The following value misses screen units ('dip' or %x / %y): 10. (warning #
** Activity (main) Pause, UserClosed = false **
```

Typing for example ‘lbl’ at the beginning of a line shows immediately:

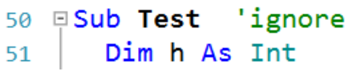
- `lbl` in red, because lbl was not declared.
- a warning Undeclared variable 'lbl' is used before it was assigned any value.
- the auto complete pop up window with suggestion containing the written characters.

4.3.7.1.1 Ignoring warnings

You, as the developer, can choose to ignore any warning. Adding an "ignore" comment will disable all the warnings for that specific line:



```
Sub Test
    Dim h As Int
```



```
Sub Test  'ignore
    Dim h As Int
```

You can also disable warnings from a specific type in the module by adding the #IgnoreWarning attribute in the Project Attributes or Module Attributes regions.

For example, to disable warnings #10 and #12:

```
#Region  Project Attributes
  #ApplicationLabel: Warnings
  #VersionCode: 1
  #VersionName:
  'SupportedOrientations possible values: unspecified, landscape or portrait.
  #SupportedOrientations: unspecified
  #CanInstallToExternalStorage: False
  #IgnoreWarnings: 10, 12
#End Region
```

You find the warning numbers at the end of each warning line.

4.3.7.1.2 List of warnings

1: Unreachable code detected.
2: Not all code paths return a value.
3: Return type (in Sub signature) should be set explicitly.
4: Return value is missing. Default value will be used instead.
5: Variable declaration type is missing. String type will be used.
6: The following value misses screen units ('dip' or %x / %y): {1}.
7: Object converted to String. This is probably a programming mistake.
8: Undeclared variable '{1}'.
9: Unused variable '{1}'.
10: Variable '{1}' is never assigned any value.
11: Variable '{1}' was not initialized.
12: Sub '{1}' is not used.
13: Variable '{1}' should be declared in Sub Process_Globals.
14: File '{1}' in Files folder was not added to the Files tab.\nYou should either delete it or add it to the project.\nYou can choose Tools - Clean unused files.
15: File '{1}' is not used.
16: Layout file '{1}' is not used. Are you missing a call to Activity.LoadLayout?
17: File '{1}' is missing from the Files tab.
18: TextSize value should not be scaled as it is scaled internally.
19: Empty Catch block. You should at least add Log(LastException.Message).
20: View '{1}' was added with the designer. You should not initialize it.
21: Cannot access view's dimension before it is added to its parent.
22: Types do not match.
23: Modal dialogs are not allowed in Sub Activity_Pause. It will be ignored.
24: Accessing fields from other modules in Sub Process_Globals can be dangerous as the initialization order is not deterministic.
28: It is recommended to use a custom theme or the default theme.
Remove SetApplicationAttribute(android:theme, “@android:style/Theme.Holo”) from the manifest editior.
32: Library ‘xxxx’ is not used.

'Runtime warnings
1001: Panel.LoadLayout should only be called after the panel was added to its parent.
1002: The same object was added to the list. You should call Dim again to create a new object.
1003: Object was already initialized.
1004: FullScreen or IncludeTitle properties in layout file do not match the activity attributes settings.

1: Unreachable code detected.

There is some code which will never be executed.
This can happen if you have some code in a Sub after a Return statement.

2: Not all code paths return a value.

```
Sub Calc(Val1 As Double, Val2 As Double, Operation As String) As Double
   Select Operation
   Case "Add"
      Return (Val1 + Val2)
   Case "Sub"
      Return (Val1 - Val2)
   Case "Mult"
      Return (Val1 * Val2)
   Case "Div"

   End Select
End Sub
```

In the `Case "Div"` path no value is returned !

Other example:
Wrong code

```
Sub Activity_KeyPress(KeyCode As Int) As Boolean
   Private Answ As Int
   Private Txt As String

   If KeyCode = KeyCodes.KEYCODE_BACK Then' Checks if the KeyCode is BackKey
      Txt = "Do you really want to quit the program ?"
      Answ = Msgbox2(Txt,"A T T E N T I O N","Yes","","No",Null) ' MessageBox
      If Answ = DialogResponse.POSITIVE Then  ' If return value is Yes then
        Return False ' Return = False  the Event will not be consumed
      Else                       ' we leave the program
        Return True              ' Return = True    the Event will be consumed to avoid
      End If                     ' leaving the program
   End If
End Sub
```

Correct code

```
Sub Activity_KeyPress(KeyCode As Int) As Boolean
   Private Answ As Int
   Private Txt As String

   If KeyCode = KeyCodes.KEYCODE_BACK Then' Checks if the KeyCode is BackKey
      Txt = "Do you really want to quit the program ?"
      Answ = Msgbox2(Txt,"A T T E N T I O N","Yes","","No",Null) ' MessageBox
      If Answ = DialogResponse.POSITIVE Then  ' If return value is Yes then
        Return False ' Return = False  the Event will not be consumed
      Else                       ' we leave the program
        Return True              ' Return = True    the Event will be consumed to avoid
      End If                     ' leaving the program
   Else
      Return True                ' Return = True    the Event will be consumed to avoid
   End If                        ' leaving the program
End Sub
```

3: Return type (in Sub signature) should be set explicitly.

Wrong code

```
Sub Calc(Val1 As Double, Val2 As Double, Operation As String)
```

Correct code

```
Sub Calc(Val1 As Double, Val2 As Double, Operation As String) As Double
```

The return type must be declared !

4: Return value is missing. Default value will be used instead.

Wrong code

```
Sub CalcSum(Val1 As Double, Val2 As Double) As Double
  Private Sum As Double

  Sum = Val1 + Val2
  Return
End Sub
```

Correct code

```
Sub CalcSum(Val1 As Double, Val2 As Double) As Double
  Private Sum As Double

  Sum = Val1 + Val2
  Return Sum
End Sub
```

5: Variable declaration type is missing. String type will be used.

Wrong code

```
Sub Calc(Val1, Val2 As Double, Operation As String) As Double
```

Correct code

```
Sub Calc(Val1 As Double, Val2 As Double, Operation As String) As Double
```

In sub declarations each variable needs its own type declaration.
But in Private, Public or Dim declarations it's allowed, in the line below both variables are Doubles:

```
Private Val1, Val2 As Double
```

6: The following value misses screen units ('dip' or %x / %y): {1}.

Wrong code

```
Activity.AddView(lblTest, 10, 10, 150, 50)
```

Correct code

```
Activity.AddView(lblTest, 10dip, 10dip, 150dip, 50dip)
```

In the example above you will get four warnings, one for each value.
For view dimensions you should use dip, %x or %y values.
See chapter 5.1 Special functions like 50%x, 50dip

7: Object converted to String. This is probably a programming mistake.

8: Undeclared variable '{1}'.

Wrong code

```
Sub SetHeight
  h = 10dip
End Sub
```

Correct code

```
Sub SetHeight
  Private h As Int
  h = 10dip
End Sub
```

The variable `h` was not declared. You see it also with the red color.

9: Unused variable '{1}'.

```
Sub SetHeight
   Private h As Int
   h = 10dip
End Sub
```

This warning tells that the variable h is not used.
It is declared and assigned a value, but it is not used !

This code gives no warning because variable h is used:

```
Sub SetHeight
   Private h As Int
   h = 10dip
   lblTest.Height = h
End Sub
```

10: Variable '{1}' is never assigned any value.

```
Sub Test
   Private h As Int

End Sub
```

This warning shows that the variable h is declared but not assigned any value.
Correct code see above.

11: Variable '{1}' was not initialized.

Wrong code

```
   Private lst As List
   lst.Add("Test1")
```

Correct code

```
   Private lst As List
   lst.Initialize
   lst.Add("Test1")
```

Variables (objects) like List or Map must be initialized before they can be used.
Views added by code must also be initialized before they can be added to a parent view.

12: Sub '{1}' is not used.

This warning is displayed if a Sub routine is never used.

13: Variable '{1}' should be declared in Sub Process_Globals.

Wrong code :

```
Sub Globals
   Public Timer1 As Timer
   Public GPS1 As GPS
```

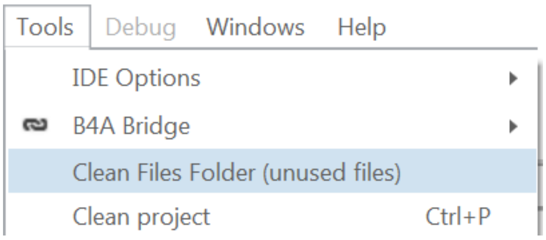
Correct code :

```
Sub Process_Globals
   Public Timer1 As Timer
   Public GPS1 As GPS
```

Certain objects like Timers and GPS should be declared in Process_Globals, not in Globals.

14: File '{1}' in Files folder was not added to the Files tab.

You are using a file which is in the Files folder, but was not added to the Files tab.
You should:
- Make a backup copy.
- Delete it from the Files subfolder.
- Add it to the project in the Files tab.
- Use Clean Files Folder (unused files) in the Tools menu.

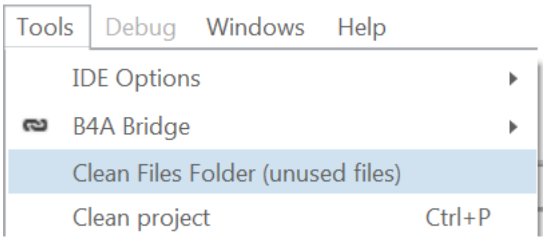


15: File '{1}' is not used.

You have files in the Files folder that are not used.
You should remove them from the Files folder.
Or you can clean the Files folder from within the Tools menu (see above).

16: Layout file '{1}' is not used. Are you missing a call to Activity.LoadLayout?

You have a layout file in the Files folder that is not used.
You should add LoadLayout or you can remove the layout file from the Files folder.
Or you can clean the Files folder in the Tools menu.



17: File '{1}' is missing from the Files tab.

The given file is in the Files tab but is missing in the Files folder. You should add it.
See chapter 4.3.2 Files

18: TextSize value should not be scaled as it is scaled internally.

Wrong code

```
lblTest.TextSize = 16dip
```

Correct code

```
lblTest.TextSize = 16
```

TextSize values are pixel and density independent. Their unit is the typographic point, a typographic unit, and must be given absolute values and not dip values.

19: Empty Catch block. You should at least add Log(LastException.Message).

Wrong code

```
Try
   imvImage.Bitmap = LoadBitmap(File.DirRootExternal, "image.jpg")
Catch

End Try
```

Correct code

```
Try
   imvImage.Bitmap = LoadBitmap(File.DirRootExternal, "image.jpg")
Catch
   Log(LastException.Message)
End Try
```

It is recommended to add at least `Log(LastException.Message)` in the Catch block instead of leaving it empty.

20: View '{1}' was added with the designer. You should not initialize it.

A View defined with the Designer in a layout file must not be initialized !
Only views added by code need to be initialized.

21: Cannot access view's dimension before it is added to its parent.

You must add a view to a parent view before you can access its dimensions.
When you add a view by code its dimensions are defined when you add it with AddView.

22: Types do not match.

23: Modal dialogs are not allowed in Sub Activity_Pause. It will be ignored.

Modal dialogs like MessageBox should not be used in the Activity_Pause routine.

24: Accessing fields from other modules in Sub Process_Globals can be dangerous as the initialization order is not deterministic.

28: It is recommended to use a custom theme or the default theme.
Remove SetApplicationAttribute(android:theme, “@android:style/Theme.Holo”) from the manifest editior.

This was set automatically in older versions of B4A. No more needed.

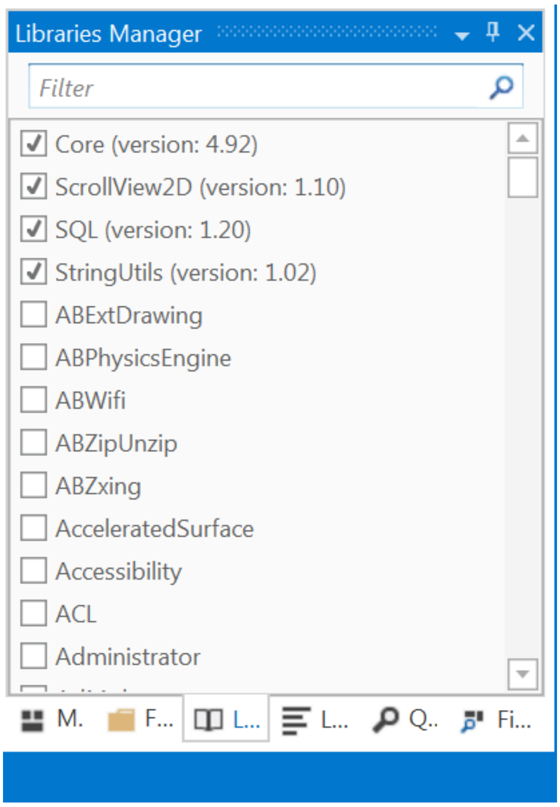
32: Library ‘xxxx’ is not used.

Remove the unused library.

4.3.8 Libraries Manager Libraries Manager

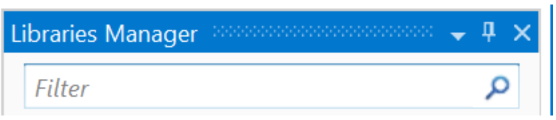
The “Libraries Manager” Tab contains a list of the available libraries that can be used in the project.

Check the libraries you need for your project.
Make sure that you have the latest version of the libraries.

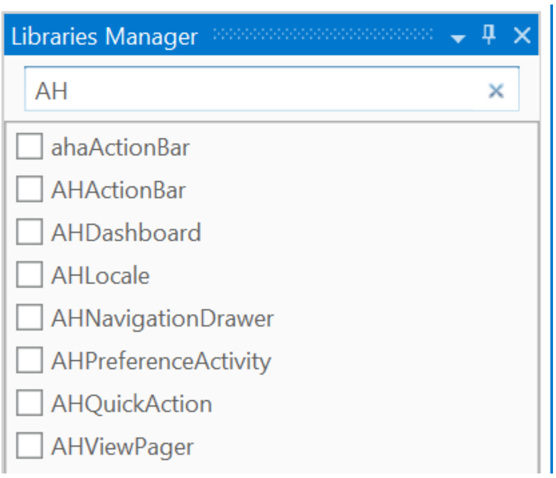


☑ Core (version: 4.92)
☑ ScrollView2D (version: 1.10)
☑ SQL (version: 1.20)
☑ StringUtils (version: 1.02)
□ ABExtDrawing
□ ABPhysicsEngine
□ ABWifi
□ ABZipUnzip
□ ABZxing
□ AcceleratedSurface
□ Accessibility
□ ACL
□ Administrator

The used libraries are shown on top of the list.
As soon as you select one it moves to the top of the list.



On the top of the Tab you find a field to filter the libraries.



AH

□ ahaActionBar
□ AHActionBar
□ AHDashboard
□ AHLocale
□ AHNavigationDrawer
□ AHPreferenceActivity
□ AHQuickAction
□ AHViewPager

Enter ‘AH’ and you get all libraries beginning with AH.

The list of all additional libraries can be found here: Additional Libraries.

The documentation for libraries can be found here: B4A - Libraries.
Clicking on a link in the list shows the documentation.

Look also at chapter Libraries.

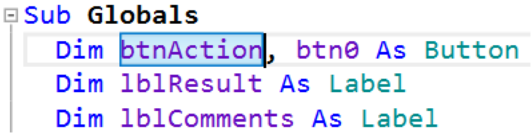
4.3.9 Quick Search Quick Search

Quick Search allows to search for any text occurrences in the code of the whole project.
Examples with the SecondProgram code.

Several possibilities to select the Quick Search function:

- Ctrl + F, the easiest and most efficient way.
- Click on the Quick Search Tab in the lower right corner of the IDE.
- Click on Quick Search Ctrl+F in the Edit menu.

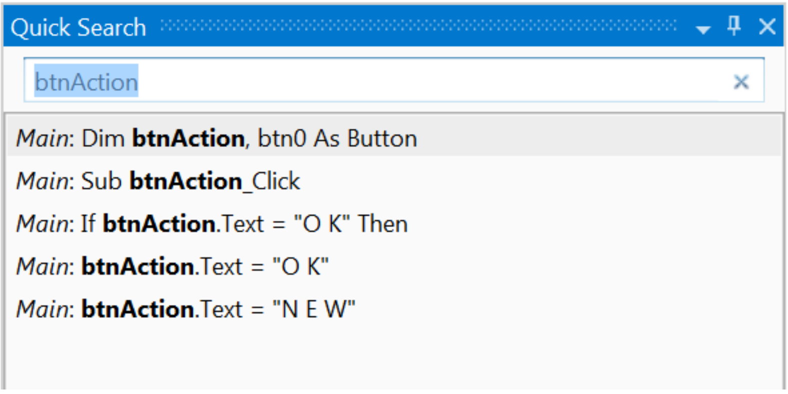
Example:



```
Sub Globals
    Dim btnAction, btn0 As Button
    Dim lblResult As Label
    Dim lblComments As Label
```

In the code double click on `btnAction` to select it and press Ctrl + F.

You get the window below in the Tab area.



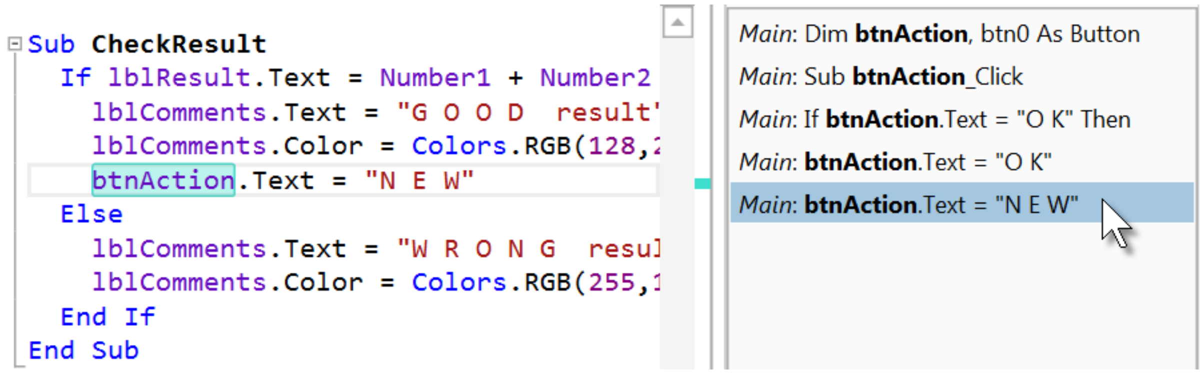
btnAction

Main: Dim **btnAction**, btn0 As Button
Main: Sub **btnAction**_Click
Main: If **btnAction**.Text = "O K" Then
Main: **btnAction**.Text = "O K"
Main: **btnAction**.Text = "N E W"

The list shows the occurrences in all Modules.

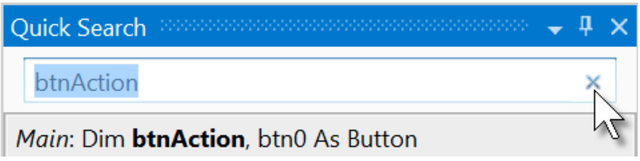
In each line you find the Module name and the line content.

Clicking on a line in the list moves the cursor directly to the selected occurrence in the code.



```
Sub CheckResult
    If lblResult.Text = Number1 + Number2
        lblComments.Text = "G O O D  result'
        lblComments.Color = Colors.RGB(128,2
        btnAction.Text = "N E W"
    Else
        lblComments.Text = "W R O N G  resul
        lblComments.Color = Colors.RGB(255,1
    End If
End Sub
```

Main: Dim **btnAction**, btn0 As Button
Main: Sub **btnAction**_Click
Main: If **btnAction**.Text = "O K" Then
Main: **btnAction**.Text = "O K"
Main: **btnAction**.Text = "N E W"



btnAction

Main: Dim **btnAction**, btn0 As Button

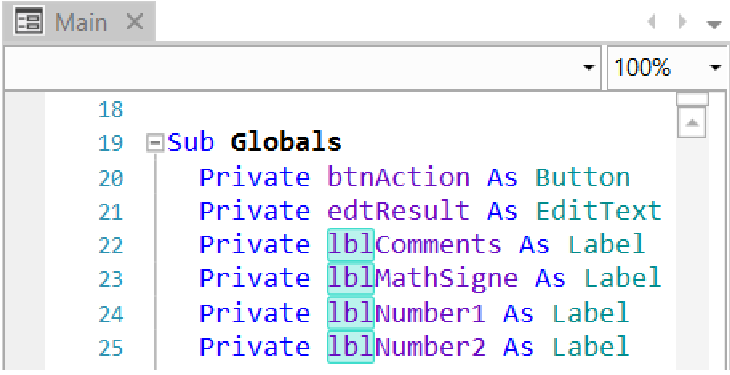
To remove the selection click on × on the top right corner of the Quick Search window.

You can also enter any text in the search field:

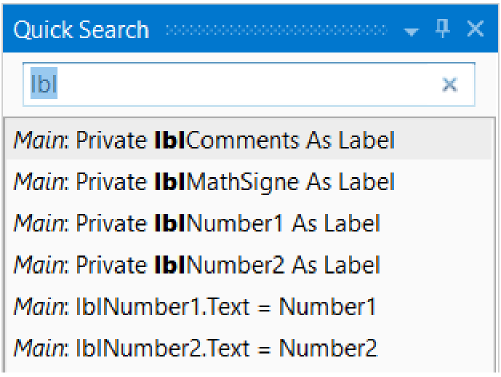
As an example, enter *lbl* in the Search field and you get the window below where you find all lines containing the text you entered, *lbl* in this example.

The search text is highlighted in all code lines containing this text.

Clicking on one of the lines in the list jumps directly to this line in the IDE.

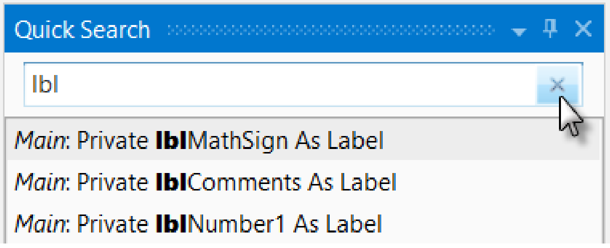


```
Sub Globals
    Private btnAction As Button
    Private edtResult As EditText
    Private lblComments As Label
    Private lblMathSigne As Label
    Private lblNumber1 As Label
    Private lblNumber2 As Label
```



lbl

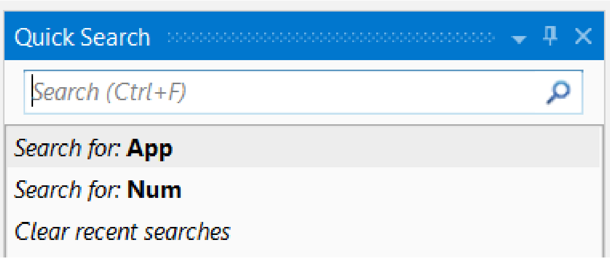
Main: Private **lbl**Comments As Label
Main: Private **lbl**MathSigne As Label
Main: Private **lbl**Number1 As Label
Main: Private **lbl**Number2 As Label
Main: lblNumber1.Text = Number1
Main: lblNumber2.Text = Number2



lbl

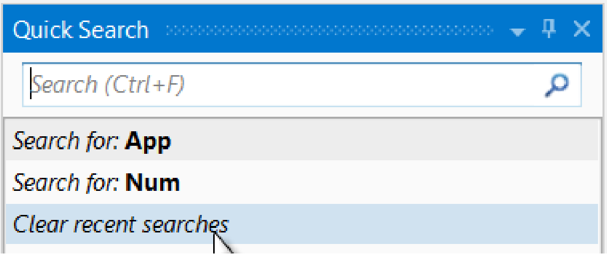
Main: Private **lbl**MathSign As Label
Main: Private **lbl**Comments As Label
Main: Private **lbl**Number1 As Label

Click on × to remove a search.



Search for: **App**
Search for: **Num**
Clear recent searches

You will see a list of the last searches.



Search for: **App**
Search for: **Num**
Clear recent searches

Click on *Clear recent searches* to remove all recent searches.

Items are added to the recent items when:
1. You select one of the results or click enter which selects the first result.
2. You select text in your code and click on Ctrl + F to search for it.

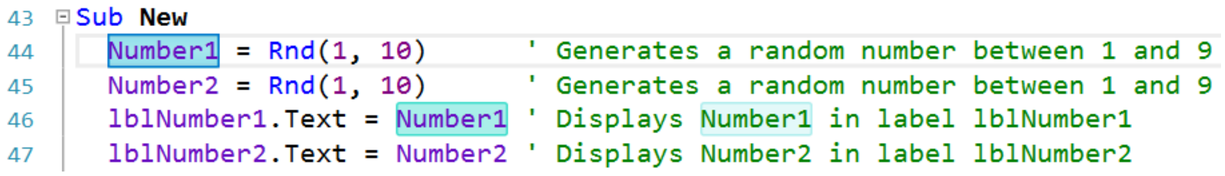
4.3.10 Find All References (F7) Find All References (F7)

This is a search engine to find all references for a given object (view, variable).

Click on the Find All References (F7) Tab or press F7 to get the screen below showing a list of all code lines with the selected reference or the first object in the current line.

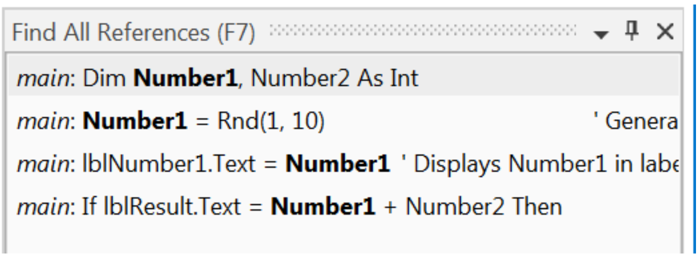
Example with the code of SecondProgram.

Select in the code in line 49 `Number1`.



```
43 Sub New
44     Number1 = Rnd(1, 10)         ' Generates a random number between 1 and 9
45     Number2 = Rnd(1, 10)         ' Generates a random number between 1 and 9
46     lblNumber1.Text = Number1 ' Displays Number1 in label lblNumber1
47     lblNumber2.Text = Number2 ' Displays Number2 in label lblNumber2
```

Click on Find All References (F7) or press F7 and you get the list below with all code lines containing the selected object.



main: Dim **Number1**, Number2 As Int

main: **Number1** = Rnd(1, 10) ' Genera

main: lblNumber1.Text = **Number1** ' Displays Number1 in labe

main: If lblResult.Text = **Number1** + Number2 Then

Clicking on a line in the list shows that line in the middle of the IDE code area.

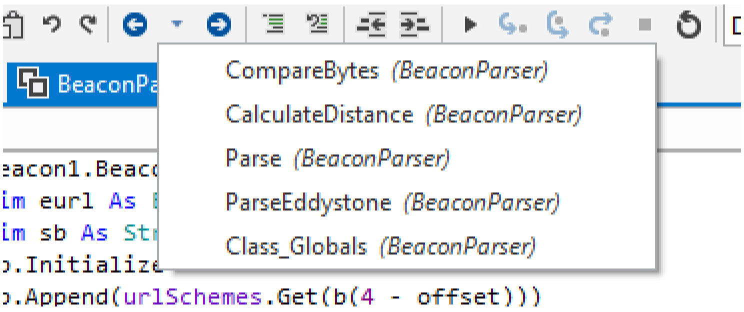
4.4 Navigation in the IDE

4.4.1 Alt + Left / Alt + Right Move backwards and forwards

Moves backwards and forwards based on the navigation stack. This is useful to jump back and forth between the last recent subs.

4.4.2 Alt + N Navigation stack menu

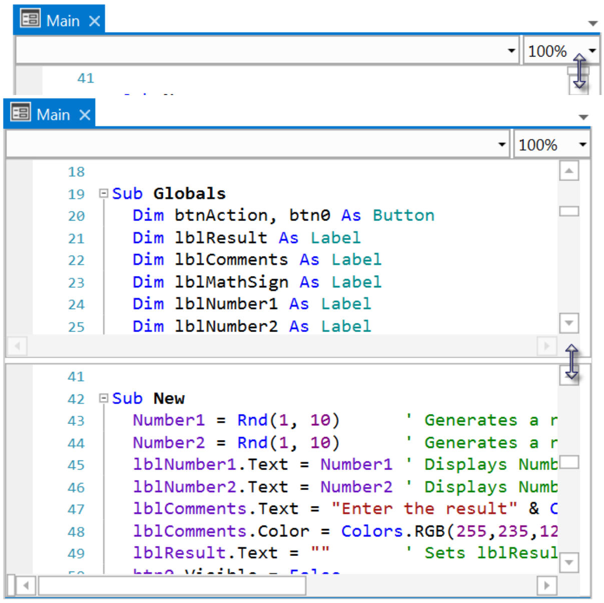
Opens the navigation stack menu. You can then choose the location with the up and down keys.


BeaconPa
CompareBytes (BeaconParser)
CalculateDistance (BeaconParser)
Parse (BeaconParser)
ParseEddystone (BeaconParser)
Class_Globals (BeaconParser)

```
eacon1.Beac
im eurl As 
im sb As Str
).Initialize
).Append(urlSchemes.Get(b(4 - offset)))
```

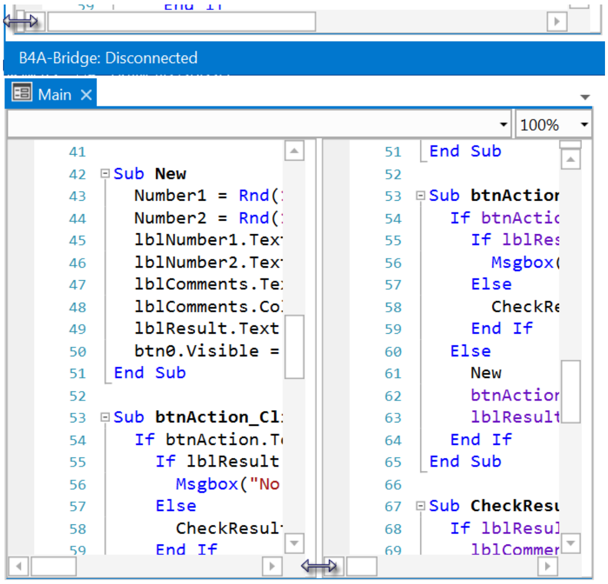
4.4.3 Split the screen

If you are working on two locations in the same module then you can split the code editor (it can be split again vertically):



```
Sub Globals
    Dim btnAction, btn0 As Button
    Dim lblResult As Label
    Dim lblComments As Label
    Dim lblMathSign As Label
    Dim lblNumber1 As Label
    Dim lblNumber2 As Label
```

```
Sub New
    Number1 = Rnd(1, 10)        ' Generates a r
    Number2 = Rnd(1, 10)        ' Generates a r
    lblNumber1.Text = Number1   ' Displays Numb
    lblNumber2.Text = Number2   ' Displays Numb
    lblComments.Text = "Enter the result" & C
    lblComments.Color = Colors.RGB(255,235,12
    lblResult.Text = ""         ' Sets lblResul
```

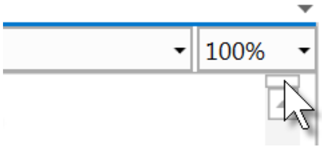


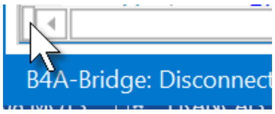
```
Sub New
    Number1 = Rnd(
    Number2 = Rnd(
    lblNumber1.Tex
    lblNumber2.Tex
    lblComments.Te
    lblComments.Co
    lblResult.Text
    btn0.Visible =
End Sub

Sub btnAction_Cl
    If btnAction.T
        If lblResult
            Msgbox("No
        Else
            CheckResul
        End If
```

```
End Sub

Sub btnAction
    If btnActic
        If lblRes
            Msgbox(
        Else
            CheckRe
        End If
    Else
        New
        btnAction
        lblResult
    End If
End Sub

Sub CheckResu
    If lblResul
        lblComme
```

Horizontally Vertically

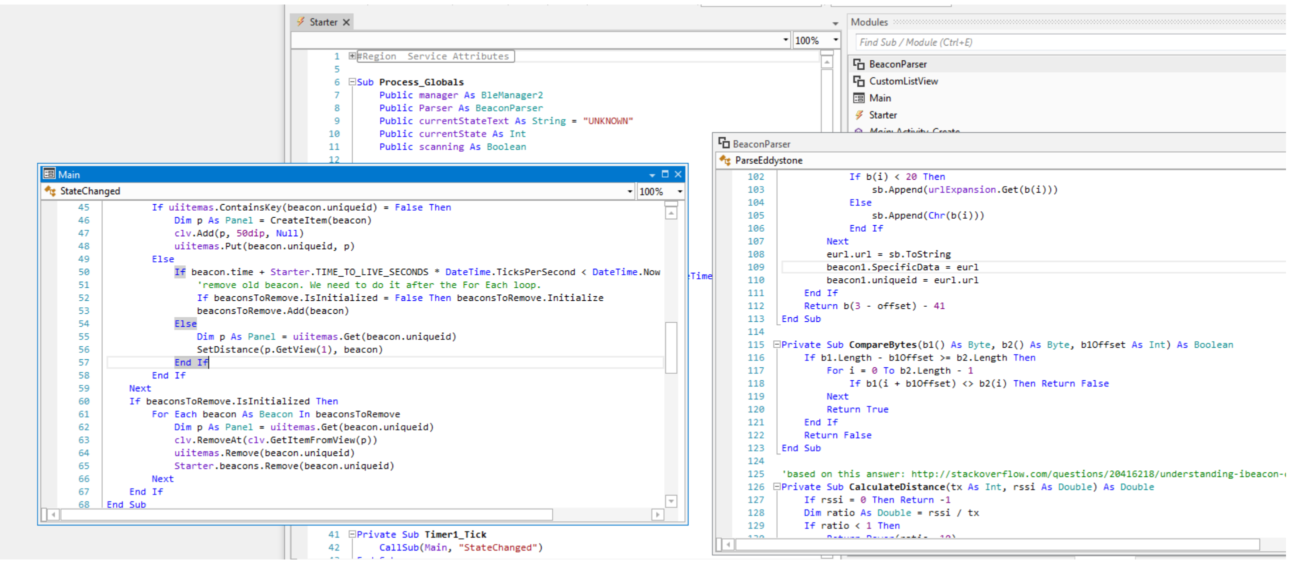
You can also double click on the small rectangles to split the screen.





4.4.4 Multiple windows

If you are working with multiple modules you can move the modules out of the main IDE as separate windows.



```
#Region  Service Attributes

Sub Process_Globals
    Public manager As BleManager2
    Public Parser As BeaconParser
    Public currentStateText As String = "UNKNOWN"
    Public currentState As Int
    Public scanning As Boolean
```

```
            If uiitemas.ContainsKey(beacon.uniqueid) = False Then
                Dim p As Panel = CreateItem(beacon)
                clv.Add(p, 50dip, Null)
                uiitemas.Put(beacon.uniqueid, p)
            Else
                If beacon.time + Starter.TIME_TO_LIVE_SECONDS * DateTime.TicksPerSecond < DateTime.Now
                    'remove old beacon. We need to do it after the For Each loop.
                    If beaconsToRemove.IsInitialized = False Then beaconsToRemove.Initialize
                    beaconsToRemove.Add(beacon)
                Else
                    Dim p As Panel = uiitemas.Get(beacon.uniqueid)
                    SetDistance(p.GetView(1), beacon)
                End If
            End If
        Next
        If beaconsToRemove.IsInitialized Then
            For Each beacon As Beacon In beaconsToRemove
                Dim p As Panel = uiitemas.Get(beacon.uniqueid)
                clv.RemoveAt(clv.GetItemFromView(p))
                uiitemas.Remove(beacon.uniqueid)
                Starter.beacons.Remove(beacon.uniqueid)
            Next
        End If
End Sub
```

```
Private Sub Timer1_Tick
    CallSub(Main, "StateChanged")
```

```
            If b(i) < 20 Then
                sb.Append(urlExpansion.Get(b(i)))
            Else
                sb.Append(Chr(b(i)))
            End If
        Next
        eurl.url = sb.ToString
        beacon1.SpecificData = eurl
        beacon1.uniqueid = eurl.url
    End If
    Return b(3 - offset) - 41
End Sub

Private Sub CompareBytes(b1() As Byte, b2() As Byte, b1Offset As Int) As Boolean
    If b1.Length - b1Offset >= b2.Length Then
        For i = 0 To b2.Length - 1
            If b1(i + b1Offset) <> b2(i) Then Return False
        Next
        Return True
    End If
    Return False
End Sub

'based on this answer: http://stackoverflow.com/questions/20416218/understanding-ibeacon-
Private Sub CalculateDistance(tx As Int, rssi As Double) As Double
    If rssi = 0 Then Return -1
    Dim ratio As Double = rssi / tx
    If ratio < 1 Then
```

4.4.5 Ctrl + E Search for sub or module

Ctrl + E - searches for sub or module. Very useful when working with large projects.

4.4.6 Ctrl + Click on any sub or variable

Ctrl + Click on any sub or variable to jump to the declaration location.

4.4.7 F7 - Find all references

Not exactly related to navigation but is also useful when working with large projects.
Details in Find all references.

4.4.8 Ctrl + F Quick Search

Ctrl + F - Index based quick search. Details in Quick Search.

5 Screen sizes and resolutions

There exist many different screen sizes with different resolutions and pixel densities.
We must explain the difference between the following parameters.

- Physical screen size	Ex: 3.6 " diagonal
- Resolution in pixels	Ex: 320 / 480
- Density pixels per inch	Ex: 160
- Scale	Ex: 1

The standard screen is 320 / 480 pixels, density 160 pixels/inch and scale 1.
There exist other screens with almost the same physical size but with a higher resolution (for example 480 / 640 pixels with a density of 240 pixels/inch and a scale of 1.5).
Tablets have bigger physical sizes but can have a density similar to the standard screen.
Example: 7.2 " screen diagonal, 640 / 960 pixels and a density of 160 pixels/inch.

A non-exhaustive list of screens:

Diagonal	Resolution	Density	Scale	W / H Ratio
3.5	320 / 480	160	1	3 / 2
3.5	480 / 720	240	1.5	3 / 2
3.9	480 / 800	240	1.5	5 / 3
3.5	240 / 320	120	0.75	4 / 3
5	1080 / 1920	480	3	4 / 3
7	640 / 960	160	1	3 / 2
7	800 / 1280	160	1	16 / 9
10	768 / 1024	160	1	4 / 3
10	800 / 1280	160	1	16 / 10
10	1200 / 1920	240	1.5	16 / 10

Let us compare the following resolutions:

1)	320 / 480 / 160	screen ~3.5"	standard density 160	scale 1
2)	480 / 800 / 240	screen ~3.5"	density 240	scale 1.5
3)	640 / 960 / 320	screen ~3.5"	density 320	scale 2
4)	640 / 960 / 160	~7" screen	standard density 160	scale 1

In cases 1) 2) and 3) the physical sizes of the screens are the same but the density of the pixels is different.
In cases 1) and 4) the densities are the same, but the physical dimensions of the screen in case 4) are double the dimensions of screen 1), yielding 4 times the area, and 4 times total number of pixels.

Let us look at the physical size of a button with 80 / 80 dip (dip = **d**ensity **i**ndependent **p**ixel).

	dips	pixels	inch	
1)	80	80	0.5	
2)	80	120	0.5	
2)	--	80	0.375	dimension given in pixels not in dips
3)	80	160	0.5	
3)	--	80	0.25	dimension given in pixels not in dips
4)	80	80	0.5	

It is possible to generate special Emulators with special sizes, resolutions and densities.

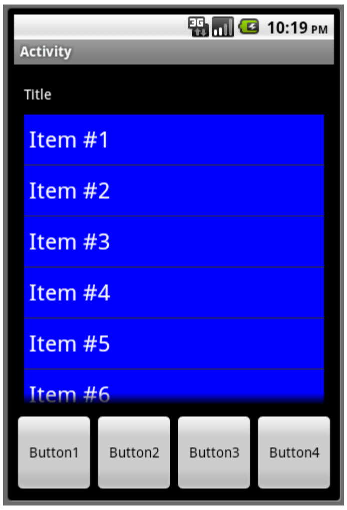
A same layout can fit into different screen resolutions, but with some restrictions.

We will use the TestLayouts program to test the same layout with different screen resolutions.
The source code is in the <Guide>\SourceCode\TestLayouts directory.

The different resolutions are:

Screen resolution	Density	H / W ratio	Equivalent height pixels	Pixel diff.
240 / 320	120	4 / 3	360	- 40
320 / 480	160	3 / 2	480	0
480 / 800	240	5 / 3	720	+ 80

The reference resolution is 320 / 480 with a density of 160.
If we calculate, for the two other resolutions, the equivalent height using the same H/W ratio we get the equivalent height in pixels and the difference in pixels.
This means that with the same layout file for all three resolutions there will be 40 pixels missing with the 240/320 resolution and 80 extra pixels with the 480/800 resolution.



The original layout in the standard 320/480 pixels density 160 Emulator is the following.

To make the tests we need three emulators:
- 320 / 480 density 160
- 240 / 320 density 120
- 480 / 800 density 240

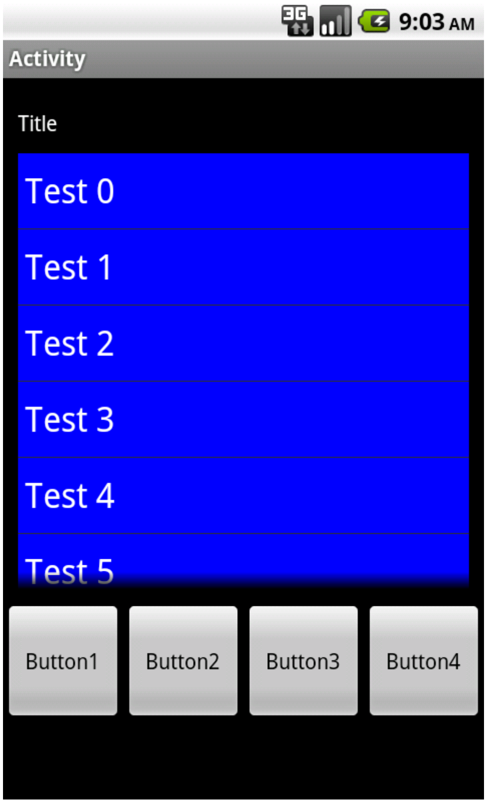
If you do not have these emulators, you must create them in the AVD Manager. Look here to Create a new Emulator.

And the code is:

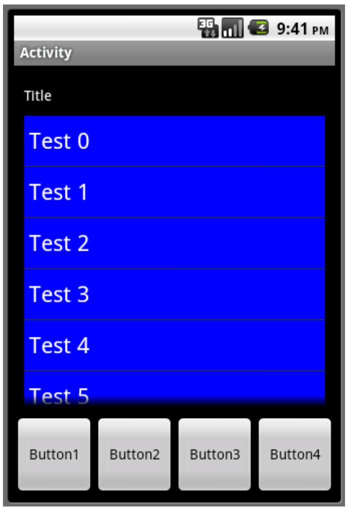
```
Sub Globals
' These global variables will be redeclared each time the activity Is created.
' These variables can only be accessed from this module.
   Private ListView1 As ListView
   Private pnlToolBox As Panel
End Sub

Sub Activity_Create(FirstTime As Boolean)
   Private i As Int

   Activity.LoadLayout("MainLayout")

   For i=0 To 10
      ListView1.AddSingleLine("Test "&i)
   Next
   If Activity.Height > Activity.Width Then
'     pnlToolBox.Top = Activity.Height - pnlToolBox.Height
'     ListView1.Height = pnlToolBox.Top - ListView1.Top - 10dip
   Else
'     pnlToolBox.Left = Activity.Width - pnlToolBox.Width
'     ListView1.Width=pnlToolBox.Left - ListView1.Left - 10dip
   End If
End Sub
```

Note that lines 38 and 39 are commented out (lines 41 and 42 too for landscape)!

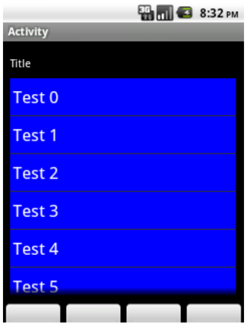
Tests with the three Emulators with different resolutions and different densities.



480 / 800 240



320 / 480 160



240 / 320 120

The image sizes are reduced by a factor of 0.5 for easier comparison.

What we see:

- with the standard resolution, the image in the emulator is equal to the original layout.
- with the 240/320 resolution we see that there are the 'expected' 40 pixels missing.
- with the 480/800 resolution, we see that there are the 'expected' 80 extra pixel.

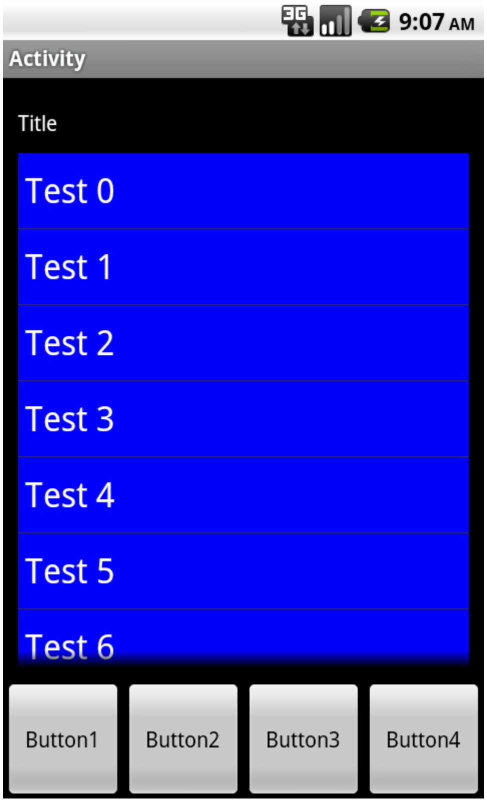
The numbers of items in the ListView are the same for all three resolutions.

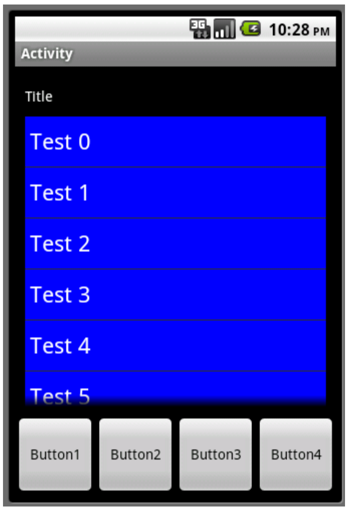
Second test with lines 38 and 39 activated (and 41 and 42 for landscape).

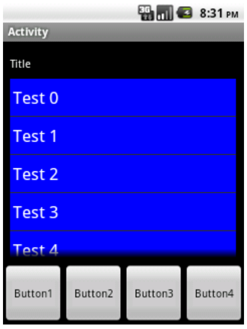
```
If Activity.Height > Activity.Width Then
   pnlToolBox.Top = Activity.Height - pnlToolBox.Height
   ListView1.Height = pnlToolBox.Top - ListView1.Top - 10dip
Else
   pnlToolBox.Left = Activity.Width - pnlToolBox.Width
   ListView1.Width = pnlToolBox.Left - ListView1.Left - 10dip
End If
```

In line 38 we calculate the top of the pnlToolBox panel according to the screen height.
In line 39 we calculate the ListView height according to the top of the pnlToolBox.

Another solution to this problem is using Anchors.







What we see:

- with the standard resolution, the image in the emulator is still equal to the original layout.
- with the 240/320 resolution we see that the buttons are at the bottom of the screen but the ListView height is shortened.
- with the 480/800 resolution we see that the buttons are at the bottom and the ListView is higher.

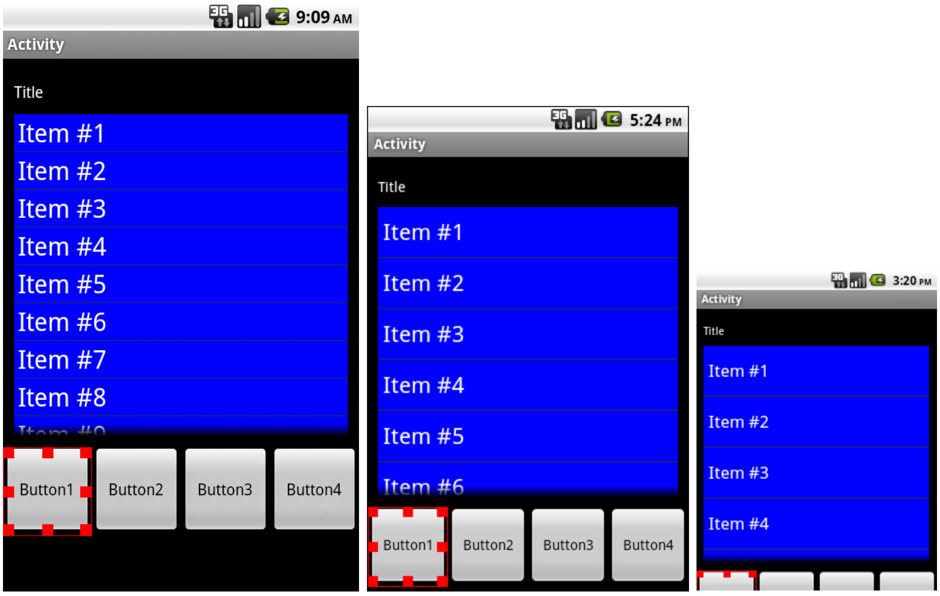
The numbers of items in the ListView is different in the three layouts because the ListView height has been adapted to the different relative screen heights.

In the first test, the number of items in the ListView were the same !

These examples show that it is not easy to have one layout for different screen resolutions.
In the example above it was relatively easy because the view in the middle is easily adjustable.

Even when we load the layout file in the three emulators with resolutions 480 / 800, 320 / 480 and 240 / 320 pixels the layout is stretched or compressed according to the screen size, but of course we also see extra or missing pixels depending on the different relative screen heights.

The Android OS auto scale system adjusts the Left, Top, Width, Height, FontSize and other properties with the scale factor but does NOT resize the vertical positions nor the heights of the views proportional to the screen height. The same is valid for the width in landscape mode.



Using Anchors.

Another and better solution to adjust and position the views according to the different screen heights is to use the Anchor function in the Designer. Anchors are described in detail in the Anchors chapter.

The TestLayoutsAnchors project shows it.

In the Designer we set:

We set:

- The vertical Anchor for `pnlToolBox` to BOTTOM.
- The horizontal Anchor for `ListView1` to BOTH.
- The vertical Anchor for `ListView1` to BOTH.

5.1 Special functions like 50%x, 50dip

There are special functions to accommodate different screen sizes and resolution.

5.1.1 PerXToCurrent, PerYToCurrent - 50%x

PerXToCurrent(Percentage As Float) or 50%x

`PerXToCurrent(50)` means 50% of the Activity width.
It can be written as a shortcut: 50%x.
50%x is equal to `Activity.Width * 0.5`

`PerYToCurrent(30)` means 30% of the Activity height.
It can be written as a shortcut: 30%y.
30%y is equal to `Activity.Height * 0.3`

In the Designer Scripts 100%x and 100%y refer to the dimensions of the view where the layout file is loaded.
If the layout file is loaded:

- onto the Activity then 100%x = Activity.Width and 100%y = Activity.Height
- onto a Panel then 100%x = Panel.Width and 100%y = Panel.Height

5.1.2 DipToCurrent - 50dip

DipToCurrent(Length As Int) or 50dip

DipToCurrent calculates a dimension with the given Length according to the scale of the current device.

DipToCurrent(50) is equal to 50 * DeviceScale
It can be written as a shortcut: 50dip **d**ensity **i**ndependent **p**ixel

The 'standard' resolution is 160 dpi (**d**ots **p**er **i**nch) and scale 1.
No spaces between the number and dip!
If we have a Button with a dimension of 50 * 50 pixels standard scale, to define its dimensions we should set Button1.Width = 50dip and Button1.Height = 50dip.
Depending on the scale, the Button dimension will be:

Scale	Pixels
1	50 * 50
1.5	75 * 75

Example:

```
Private Button1 As Button
Button1.Initialize("Button1")
Activity.AddView(Button1, 20%x, 30%y, 100dip, 50dip)
```

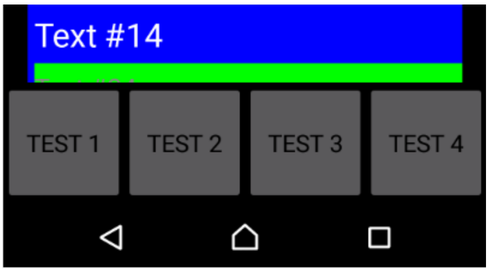
The values for the Left, Top, Width and Height properties in the Designer are considered as dip values.

5.1.3 LayoutValues.ApproximateScreenSize

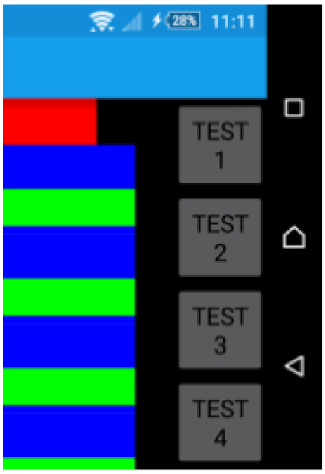
To get the approximate screen size in the code you should use the LayoutValues object.

```
Private lv As LayoutValues
lv = GetDeviceLayoutValues
```

`lv.ApproximateScreenSize` returns the approximate screen size, it’s the size of the screen without the soft buttons area. The values can be somewhat different in portrait and in landscape. On some devices the soft buttons area is smaller in landscape than in portrait.



Portrait



Landscape

5.1.4 ActivitySize in the DesignerScripts

In the DesignerScripts exist the `ActivitySize` keyword which returns the size of the Activity.
This value is different from `GetDeviceLayoutValues`, which returns the screen size and not the Activity size.
The values in portrait and in landscape are also slightly different, look above.

5.2 Working with different screen sizes / number of layouts

With the big number of devices, screen sizes, and resolutions on the market, it becomes more and more complicated to design a project that looks nice on all available devices.
There is no universal method to manage this problem. The method you choose depends on:

- What kind of project you are designing.
- What devices and screen sizes you are targeting.
- What you want to show on the different screens.
  - The same layout but stretched according to the screen sizes, or
  - Different layout variants for the different sizes. On a big screen more views can be displayed at the same time.

Summary of the different physical screen sizes, where each size can have different resolutions, densities scales and aspect ratios.

	resolution	density	scale	aspect ratio	
~ 3.0'' - 4.0 ''					
	320 / 240	120	0.75	4 / 3	1.333
	480 / 320	160	1	3 / 2	1.5
	640 / 480	240	1.5	4 / 3	1.333
	800 / 480	240	1.5	5 / 3	1.667
	854 / 480	240	1.5	16 / 9	1.78
	960 / 540	240	1.5	16 / 9	1.78
	960 / 640	240	1.5	3 / 2	1.5
	1280 / 720	320	2	16 / 9	1.78
~ 5.5 ''					
	1280 / 800	240	1.5	16 / 10	1.6
~ 7 ''					
	1024 / 600	160	1		1.71
	1280 / 800	160	1	16 / 10	1.6
~ 10 ''					
	1024 / 600	160	1		1.71
	1024 / 768	160	1	4 / 3	1.333
	1280 / 800	160	1	16 / 10	1.6
	1920 / 1200	240	1.5	16 / 10	1.6

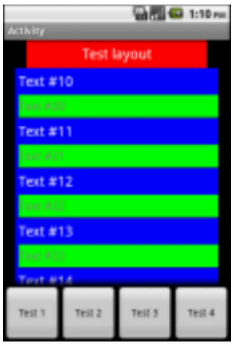
Depending on what you want to display on the different screens you can either:

- Design different layout variants.
  Two layout variants (portrait and landscape) for each dimension.
  The views are automatically resized for the different densities.
  However, you may need to take into account different width/height ratios (aspect ratios).
  These adjustment could be done in the code or would need two more layout variants.
- Calculate all view dimensions and positions in the code using `%x` , `%y` and `dip` dimensions.

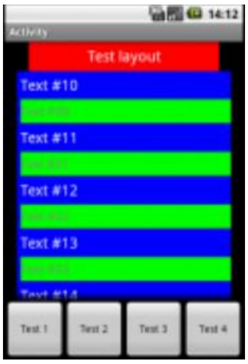
For comparison:

- A 5.5’’ screen has a surface about 2.5 times bigger than a 3.5" screen.
- A 7" screen has a surface about 4 times bigger than a 3.5" screen.
- A 10" screen has a surface about 9 times bigger than a 3.5" screen.

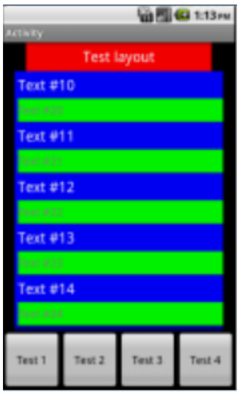
The examples below show the same layout stretched in the code to fit the different screen sizes.
The source code is OneLayoutStretched, the images are Emulator screenshots.



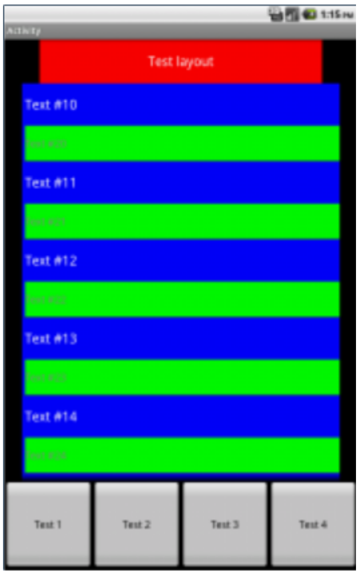
~3.5" 240 / 320



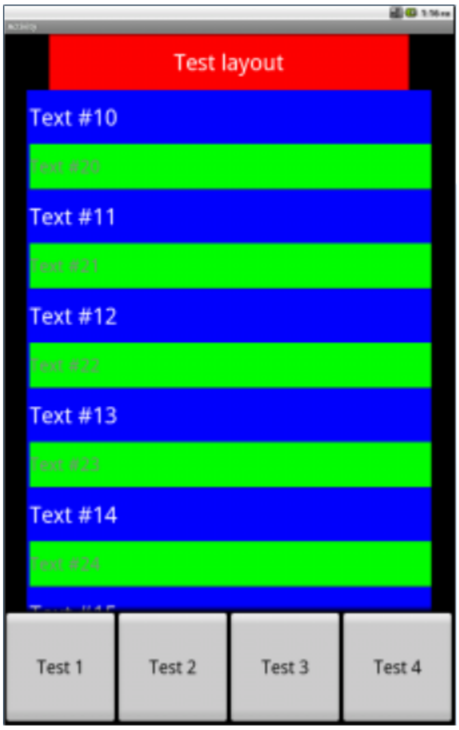
~3.5" 320 / 480



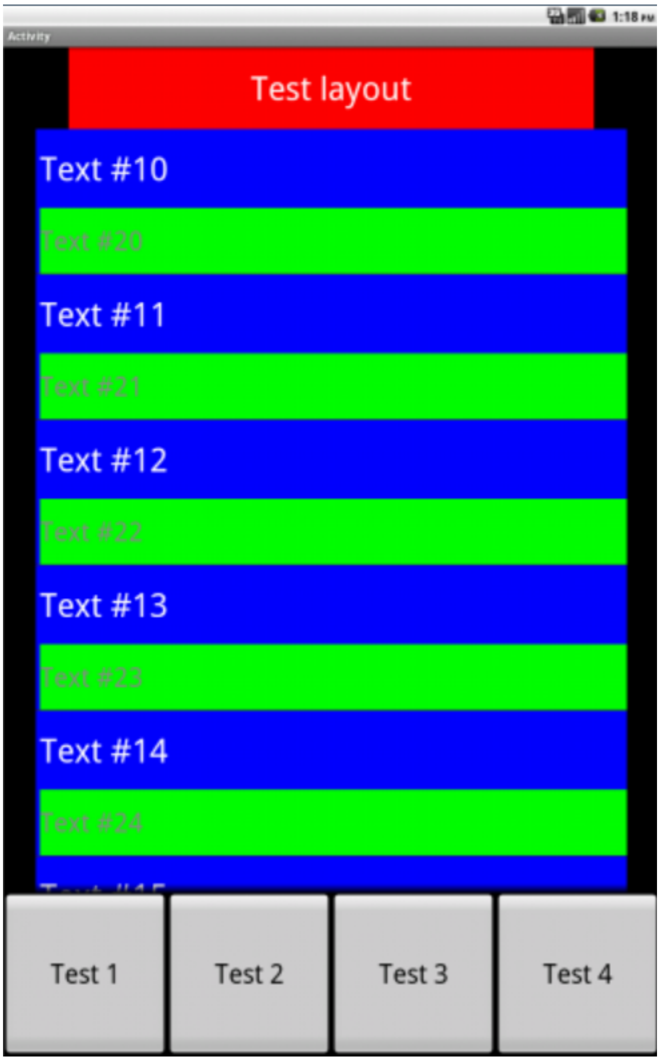
~3.5" 480 / 800



~5.5" 800 / 1280 / 240



7 " 800 / 1280 / 160



10 " 800 / 1280 / 160

It's probably not the best solution to have the same layout stretched for all screen sizes.
It could be more interesting to show more views on bigger screens.

Code to adjust the layout, we adjust the views positions and dimensions according to the Activity size in pixels and the text sizes according to the approximate screen size.

```
Sub InitLayout
   ' Gets the approximate screen size
   ' and calculates the screen size ratio
   ' according to the current screen size
   lv = GetDeviceLayoutValues
   ScreenSizeRatio = lv.ApproximateScreenSize / 3.5 '3.5 = standard screen size

   Private Width, Height As Int

   If Activity.Width > Activity.Height Then
      Height = 100%y / 4
      ' if the height is smaller than 80dip we set the width to 80dip
      Width = Max(80dip * ScreenSizeRatio, Height)

      pnlToolbox.Left = 100%x - Width
      pnlToolbox.Width = Width
      pnlToolbox.Height = 100%y
      pnlToolbox.Top = 0

      btnTest1.Left = 0
      btnTest1.Width = Width
      btnTest1.Top = 0
      btnTest1.Height = Height

      btnTest2.Left = 0
      btnTest2.Width = Width
      btnTest2.Top = Height
      btnTest2.Height = Height

      btnTest3.Left = 0
      btnTest3.Width = Width
      btnTest3.Top = 2 * Height
      btnTest3.Height = Height

      btnTest4.Left = 0
      btnTest4.Width = Width
      btnTest4.Top = 3 * Height
      btnTest4.Height = Height

      lblTitle.Left = 10%x
      lblTitle.Width = 80%x - Width
      lblTitle.Top = 0
      lblTitle.Height = 100%y / 8

      lstTest.Left = 5%x
      lstTest.Width = 90%x - Width
      lstTest.Top = lblTitle.Height
      lstTest.Height = 100%y - lstTest.Top

      lstTest.TwoLinesLayout.Label.Height = 12%y
      lstTest.TwoLinesLayout.Label.Top = 0
      lstTest.TwoLinesLayout.SecondLabel.Height = 10%y
      lstTest.TwoLinesLayout.SecondLabel.Top = 12%y
      lstTest.TwoLinesLayout.ItemHeight = 24%y
   Else
```

```
        'Width and height are the same
        Height = 100%x / 4
        pnlToolbox.Left = 0
        pnlToolbox.Width = 100%x
        pnlToolbox.Height = Height
        pnlToolbox.Top = 100%y - Height

        btnTest1.Left = 0
        btnTest1.Width = Height
        btnTest1.Top = 0
        btnTest1.Height = Height

        btnTest2.Left = Height
        btnTest2.Width = Height
        btnTest2.Top = 0
        btnTest2.Height = Height

        btnTest3.Left = 2 * Height
        btnTest3.Width = Height
        btnTest3.Top = 0
        btnTest3.Height = Height

        btnTest4.Left = 3 * Height
        btnTest4.Width = Height
        btnTest4.Top = 0
        btnTest4.Height = Height

        lblTitle.Left = 10%x
        lblTitle.Width = 80%x
        lblTitle.Top = 0
        lblTitle.Height = 100%x / 8

        lstTest.Left = 5%x
        lstTest.Width = 90%x
        lstTest.Top = lblTitle.Height
        lstTest.Height = pnlToolbox.Top - lstTest.Top

        lstTest.TwoLinesLayout.Label.Height = 12%x
        lstTest.TwoLinesLayout.Label.Top = 0
        lstTest.TwoLinesLayout.SecondLabel.Height = 10%x
        lstTest.TwoLinesLayout.SecondLabel.Top = 12%x
        lstTest.TwoLinesLayout.ItemHeight = 22%x
    End If

    btnTest1.Text = "Test 1"
    btnTest1.TextSize = 16 * ScreenSizeRatio
    btnTest2.Text = "Test 2"
    btnTest2.TextSize = 16 * ScreenSizeRatio
    btnTest3.Text = "Test 3"
    btnTest3.TextSize = 16 * ScreenSizeRatio
    btnTest4.Text = "Test 4"
    btnTest4.TextSize = 16 * ScreenSizeRatio

    lblTitle.Text = "Test layout"
    lblTitle.TextSize = 20 * ScreenSizeRatio

    lstTest.TwoLinesLayout.Label.TextSize = 20 * ScreenSizeRatio
    lstTest.TwoLinesLayout.Label.Gravity = Gravity.CENTER_VERTICAL
    lstTest.TwoLinesLayout.Label.Color = Colors.Blue
    lstTest.TwoLinesLayout.SecondLabel.TextSize = 16 * ScreenSizeRatio
    lstTest.TwoLinesLayout.SecondLabel.Gravity = Gravity.CENTER_VERTICAL
    lstTest.TwoLinesLayout.SecondLabel.Color = Colors.Green
End Sub
```

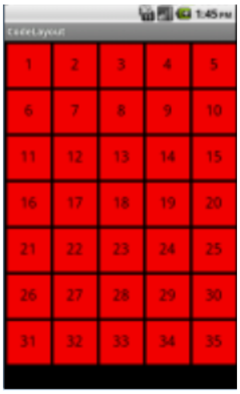
A better approach is to use DesignerScripts in the Visual Designer to separate layout adjustments from the code.
Part of the code from the `InitLayout` routine will be moved to the DesignerScripts.
This is shown in the OneLayoutStretchedDS project.

Code which remains in the `InitLayout` routine, this is needed for the ListView layout.

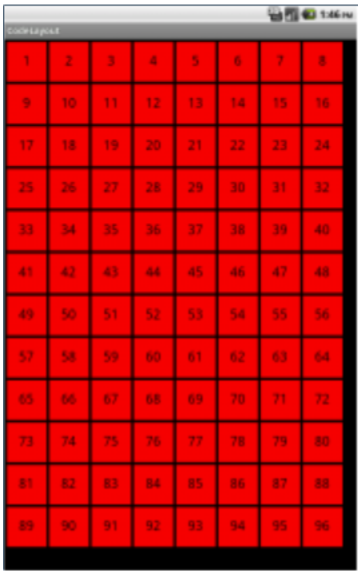
```
Sub InitLayout
   ' Gets the approximate screen size
   ' and calculates the screen size ratio
   ' according to the current screen size
   lv = GetDeviceLayoutValues
   ScreenSizeRatio = lv.ApproximateScreenSize / 3.5 '3.5 = standard screen size

   If Activity.Width > Activity.Height Then
      lstTest.TwoLinesLayout.Label.Height = 12%y
      lstTest.TwoLinesLayout.Label.Top = 0
      lstTest.TwoLinesLayout.SecondLabel.Height = 10%y
      lstTest.TwoLinesLayout.SecondLabel.Top = 12%y
      lstTest.TwoLinesLayout.ItemHeight = 24%y
   Else
      lstTest.TwoLinesLayout.Label.Height = 12%x
      lstTest.TwoLinesLayout.Label.Top = 0
      lstTest.TwoLinesLayout.SecondLabel.Height = 10%x
      lstTest.TwoLinesLayout.SecondLabel.Top = 12%x
      lstTest.TwoLinesLayout.ItemHeight = 22%x
   End If

   lstTest.TwoLinesLayout.Label.TextSize = 20 * ScreenSizeRatio
   lstTest.TwoLinesLayout.Label.Gravity = Gravity.CENTER_VERTICAL
   lstTest.TwoLinesLayout.Label.Color = Colors.Blue
   lstTest.TwoLinesLayout.SecondLabel.TextSize = 16 * ScreenSizeRatio
   lstTest.TwoLinesLayout.SecondLabel.Gravity = Gravity.CENTER_VERTICAL
   lstTest.TwoLinesLayout.SecondLabel.Color = Colors.Green
End Sub
```

The code in the DesignerScripts is almost the same as in the code, the only difference is that we use the `ActivitySize` value instead of the `lv.ApproximateScreenSize` value.

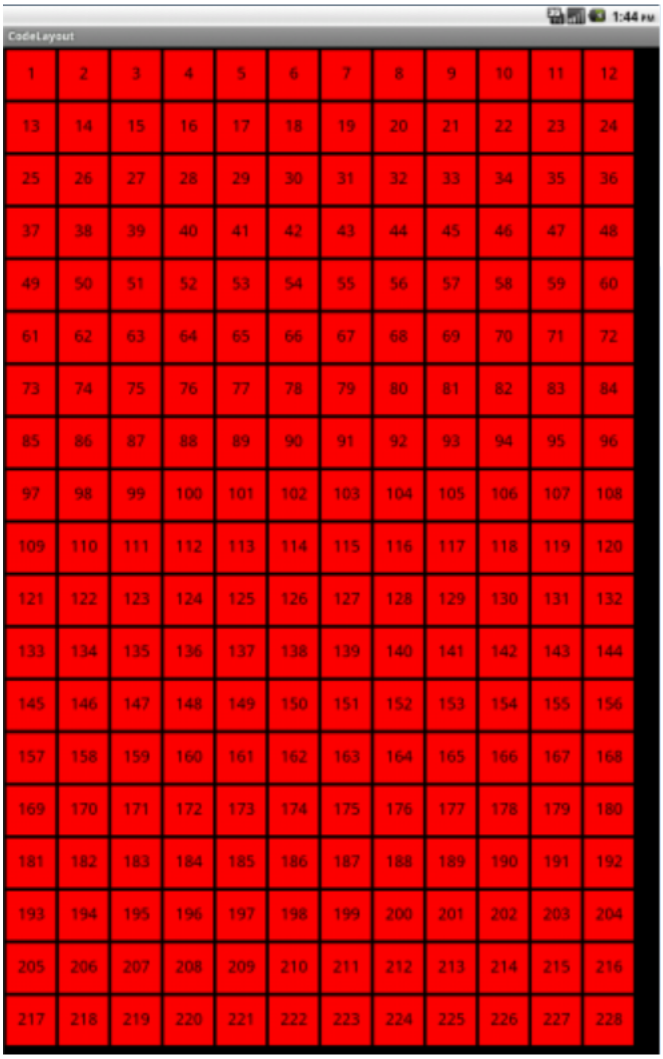
In the examples below we see the display of a grid with buttons. The physical button dimensions are almost the same. The source code is CodeLayout.



3.5"



5.5"



10"

The source code:

```
Private i, j, k, nx, ny, x0, x1, x2 As Int

x0 = 4dip
x1 = 60dip
x2 = x0 + x1

nx = Floor(Activity.Width / x2) - 1
ny = Floor(Activity.Height / x2) - 1
k = 0
For j = 0 To ny
   For i = 0 To nx
      k = k + 1
      Private btn As Button
      btn.Initialize("btn")
      btn.Color = Colors.Red
      Activity.AddView(btn, x0 + i * x2, x0 + j * x2, x1, x1)
      btn.Text = k
      btn.TextSize = 20
   Next
Next
```

If you want to display more views on bigger screens you must define two layout variants (one for portrait and one for landscape) for each screen size and resolution. This can become quite cumbersome.
A compromise could be made by defining the layouts partly in the layout and partly in the Designer Script.
The adaptation of different aspect ratios could be done in the Designer Script rather than in separate layout variants.

As already mentioned, there is no universal rule, the solution depends on different factors.
As a developer, you must define your needs and requirements as a function of :

- What kind of project you are designing.
- What kind of data you are treating, displaying, editing, etc.
- What devices and screen sizes you are targeting.
- What you want to show on the different screens.
    - The same layout, but stretched according to the screen size, or
    - Different layout variants for different sizes. On a big screen more views can be displayed at the same time.

5.3 Screen orientations

Three different screen orientation values can be defined:

- Portrait only
- Landscape only
- Both

These orientations can be defined either:

- **In the code on top in the Project Attributes region.**

  ```
  #Region  Project Attributes
     #ApplicationLabel: MyFirstProgram
     #VersionCode: 1
     #VersionName:
     #SupportedOrientations: unspecified
     #CanInstallToExternalStorage: False
  #End Region
  ```

  In this line:

  ```
     #SupportedOrientations: unspecified
  ```

  The possible orientation values are :

  ```
     #SupportedOrientations: unspecified Both
     #SupportedOrientations: portrait
     #SupportedOrientations: landscape
  ```

- **In the code with the Phone library**
  - Landscape
    `Phone1.SetScreenOrientation(0)`
  - Portrait
    `Phone1.SetScreenOrientation(1)`
  - Both
    `Phone1.SetScreenOrientation(-1)`

5.4 Supporting multiple screens - tips and best practices

There are several features in B4A and the Visual Designer that help you target Android phones and tablets with different screen sizes and resolutions. The purpose of this page is to collect tips and best practices that will help you create flexible layouts.

If you are not familiar with the designer script feature then please read this chapter
Designer Scripts

5.4.1 Advices

Below a few advices.

5.4.1.1 'dip' units

It is very simple. You should always use 'dip' units when specifying the size or position of a view (control). This way the view's **physical** position and size will be the same on any device.
This is correct for both regular code and designer script. The IDE gives a Warning for that.

```
Button1.Width = 100        'WRONG!
Button1.Width = 100dip     'Good job!
```

Note that text size is measured in physical units. So you should NOT use 'dip' units with text size values.

5.4.1.2 Use only a few layout variants

It is easy to create many variants. However it is very difficult to maintain a layout made of many variants. You should use anchors and the designer script feature to adjust (or fine tune) your layout instead of creating many variants.

5.4.1.3 Understand the meaning of scale (dots per inch)

There are many questions starting with "I have a device with 480x800 screen...". There is no meaning to these dimensions without the scale value.

A scale of 1.0 means that there are 160 dots (pixels) per inch.
The scale values can be one of the following values: 0.75, 1.0, 1.5, 2 and 3.
Most phones today have a scale of 1.5 (160 * 1.5 = 240 dots per inch).
Most tablets have a scale of 1.0, and some have a scale of 1.5.

5.4.1.4 "Normalized" variants

Normalized variants are variants with a scale of 1.0.
The layout you create with the designer is scaled (not stretched or resized) automatically. This means that the layout will look exactly the same on two phones with the same physical size. The scale doesn't matter.
It is highly recommended to work and design your layout with normalized variants only.
For example a variant of 480x800, scale=1.5 matches the normalized variant: 320x533, scale=1.0 (divide each value by the scale value). Now it is easy to see that this device is slightly longer than the "standard" variant: 320x480, scale=1.0.

5.4.1.5 Scaling strategy

You should decide what will happen with your layout when it runs on a larger device.
Usually some views will be docked to the edges. This can be done easily with the designer script.
For example, to dock a button to the right side:

```
Button1.Right = 100%x
```

Some views should fill the available area.
This is done with SetTopAndButton and SetLeftAndRight methods.

```
'Make an EditText fill the available height between two buttons:
EditText1.SetTopAndBottom(Button1.Bottom, Button2.Top)

'Make a Button fill the entire PARENT panel:
Button1.SetLeftAndRight(0, Parent1.Width)
Button1.SetTopAndBottom(0, Parent1.Height)
```

5.4.1.6 How to change the views size and text size? AutoScale

Larger devices offer a lot more available space. The result is that even if the physical size of a view is the same, it just "feels" smaller.
Some developers use %x and %y to specify the views size. However the result is far from being perfect. The layout will just be stretched.
The solution is to combine the "dock and fill" strategy with a smart algorithm that increases the views size and text size based on the running device’s physical size.
This is done with the AutoScale algorithm in the Designer Scripts.

We treat the standard variant (320 x 480, scale = 1.0) as the base variant.
AutoScale calculates a scale with the equations below:

```
delta = ((100%x + 100%y) / (320dip + 430dip) - 1)
rate = 0.3 'value between 0 to 1.
scale = 1 + rate * delta
```

You can play with the value of 'rate'. The rate determines the change amount in relation to the device physical size.
Value of 0 means no change at all. Value of 1 is similar to using %x and %y: If the physical size is twice the size of the standard phone then the size will be twice the original size.
Values between 0.2 and 0.5 seem to give good results.
The abstract designer is useful to quickly test the effect of this value.

If done properly this saves the need to create many variants.
Your layout will look good on all devices.

6 Connecting a real device

There are different means to connect a real device:

- USB
  Needs that the device supports ADB debugging.
  Need to activate USB Debugging on the device.
- B4A Bridge
  - via WiFi
  - via Bluetooth till B4A version 4.3 it is no more available since version 5.00.

6.1 Connecting via B4A Bridge

It is always recommended to use a real device instead of an Android emulator which is very slow compared to a real device (especially with applications installation).

However not all devices support ADB debugging. This is the reason for the B4A-Bridge tool.
B4A-Bridge is made of two components. One component runs on the device and allows the second component which is part of the IDE to connect and communicate with the device.
The connection is done over a network (B4A-Bridge cannot work if there is no network available).

Once connected, B4A-Bridge supports all of the IDE features which include: installing applications, viewing LogCat and the visual designer.

Android doesn't allow applications to quietly install other applications, therefore when you run your application using B4A-Bridge you will see a dialog asking for your approval.

6.1.1 Getting started with B4A-Bridge

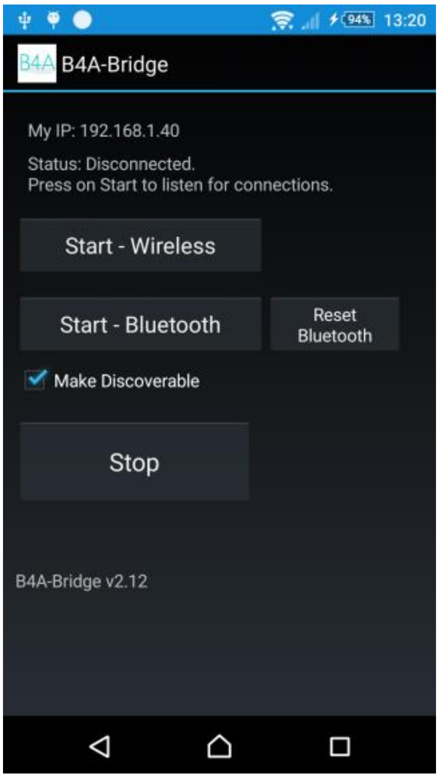
First you need to install B4A-Bridge on your device.

B4A-Bridge can be downloaded here: http://www.basic4ppc.com/android/files/b4a_bridge.apk.

B4A-Bridge is also available on Play Store. Search for: B4A Bridge.
Note that you need to allow install of applications from "Unknown sources". This is done by choosing Settings from the Home screen - Manage Applications.

B4A-Bridge requires writable storage card. It is not possible to install applications without it.

6.1.2 Run B4A-Bridge on your device.



B4A-Bridge

My IP: 192.168.1.40

Status: Disconnected.
Press on Start to listen for connections.

☑ Make Discoverable

B4A-Bridge v2.12

It will display a screen similar to:

Status will be: Press on Start to listen for connections.

Since B4A version 5.00 Bluetooth is no more supported.

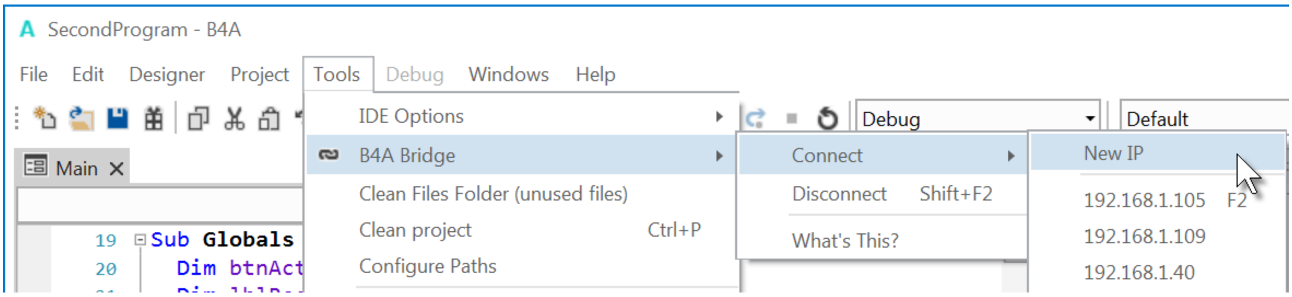
Press Start - Wireless for wireless connection.
The status will change to Waiting for wireless connections

With B4A till version 4.30 you could also select Start - Bluetooth.
The Make Discoverable checkbox will make your device Bluetooth discoverable for 5 minutes.
This is only needed if the device and computer weren't paired before.

Note that B4A-Bridge was written with B4A.

6.1.3 Wireless connections

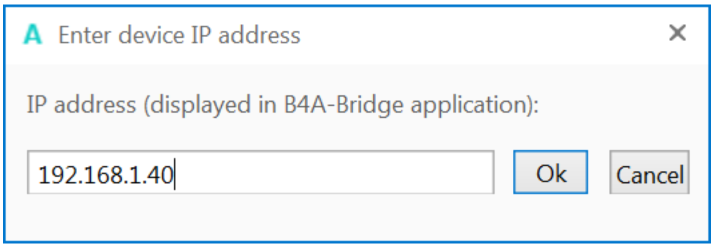
In the IDE menu Tools select New IP.
If the address already exists click directly on this address.
If this device was already connected before you can simply press F2 to connect it.



```
Sub Globals
    Dim btnAct
```

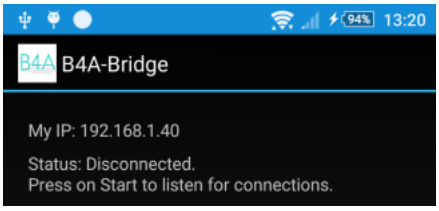
Enter the IP of the device, you find it on top of the B4A-Bridge screen on the device.

In some cases the address displayed may be the mobile network address. In that case you can find the local wireless address in the wireless advanced settings page.


Enter device IP address

IP address (displayed in B4A-Bridge application):

192.168.1.40

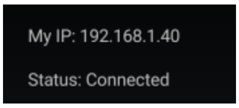


My IP: 192.168.1.40

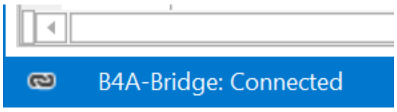
Status: Disconnected.
Press on Start to listen for connections.

Click on Ok, the device is connected to the IDE.

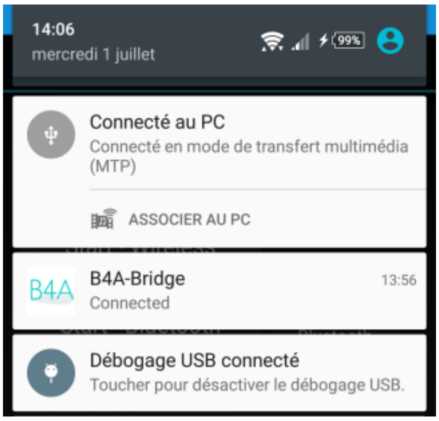
You see that the status changed on both,
the device and the IDE in the lower left corner.


My IP: 192.168.1.40

Status: Connected


B4A-Bridge: Connected.

B4A-Bridge keeps running as a service until you press on the Stop button.



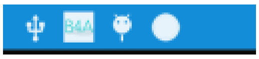
Connecté au PC
Connecté en mode de transfert multimédia (MTP)

ASSOCIER AU PC

B4A-Bridge 13:56
Connected

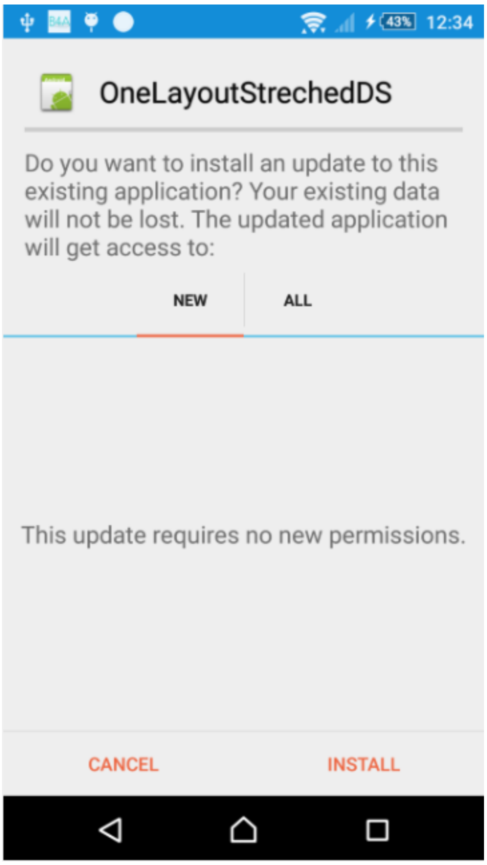
Débogage USB connecté
Toucher pour désactiver le débogage USB.

You can always reach it by opening the notifications screen.



You see B4A-Bridge with the current status.

Note that the Internet permission are automatically added in debug mode.



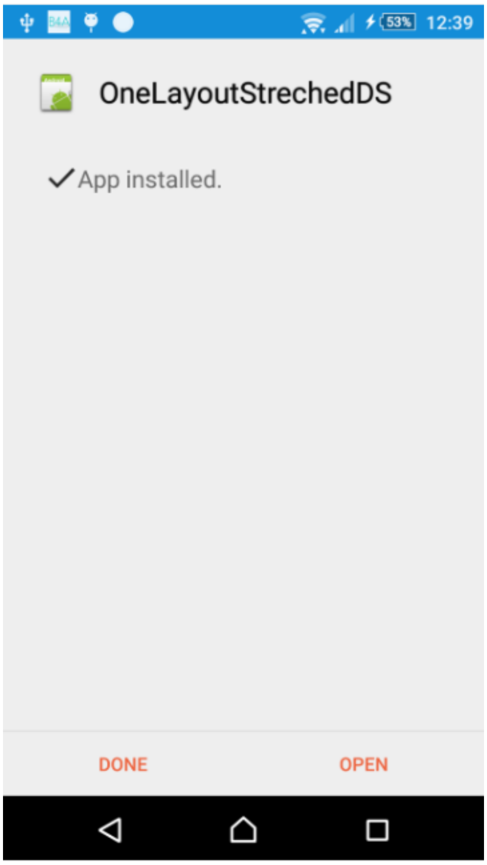
OneLayoutStrechedDS

Do you want to install an update to this existing application? Your existing data will not be lost. The updated application will get access to:

This update requires no new permissions.

When you run an application you are required to approve the installation. You will usually see a screens like the picture.

Press on INSTALL to install the program.



OneLayoutStrechedDS

✓App installed.

If you pressed on INSTALL you will see a screen like in picture.

On this screen you should choose OPEN to start the application.

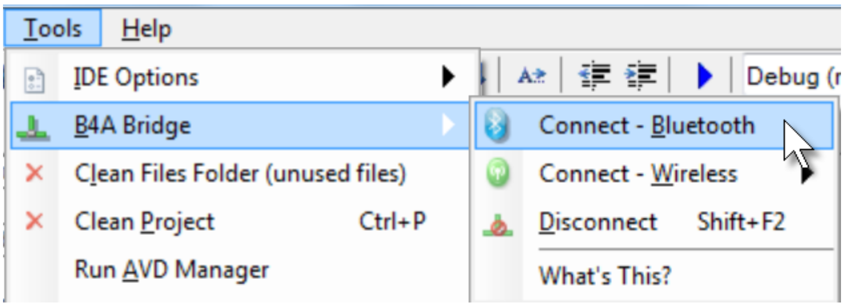
If you try to install an existing application signed with a different key, the install will fail (without any meaningful message). You should first uninstall the existing application. Go to the home screen - Settings - Applications - Manage applications - choose the application - Uninstall.

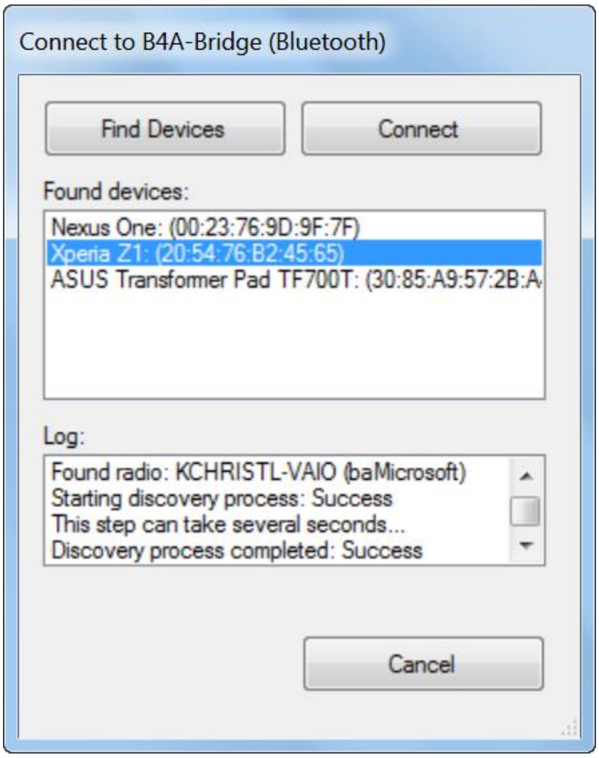
Once you finished developing you should press on the Stop button in B4A-Bridge in order to save battery.

6.1.4 Bluetooth connections

Since B4A version 5.00 Bluetooth is no more supported.

In the IDE menu Tools click on Connect - Bluetooth.





Found devices:

Nexus One: (00:23:76:9D:9F:7F)
Xperia Z1: (20:54:76:B2:45:65)
ASUS Transformer Pad TF700T: (30:85:A9:57:2B:A

Log:

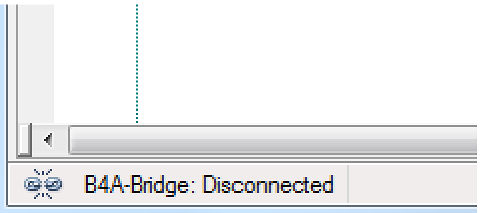
```
Found radio: KCHRISTL-VAIO (baMicrosoft)
Starting discovery process: Success
This step can take several seconds...
Discovery process completed: Success
```

Fist click on Find Devices Find Devices.

All paired devices and new devices in discoverable mode will be listed.
You should choose the correct one and press on Connect Connect.

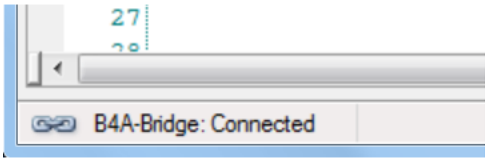
Assuming that the connection succeeded the dialog will be closed.

The status bar at the bottom of the screen shows the current status:



B4A-Bridge: Disconnected

or



B4A-Bridge: Connected

When B4A-Bridge gets connected it first checks if the designer application needs to be updated. In that case it will first install the designer application.

B4A-Bridge keeps running as a service until you press on the Stop button.
You can always reach it by opening the notifications screen:

6.1.4.1 Bluetooth tips

- Unfortunately many devices, especially older devices running Android 2.1 or 2.2 have all kinds of issues with Bluetooth connections and especially with multiple connections. All kinds of workarounds were implemented because of these issues. Still however there are devices (HTC desire for example) that do not work reliably enough.

- The Reset Bluetooth button disables and then enables the Bluetooth adapters. You should try it if there are connections problems.

- If your connection is not stable then you should avoid using the debugger or designer. Both the debugger and the designer create an additional connection.

Note that the Bluetooth permission and Internet permission are automatically added in debug mode.

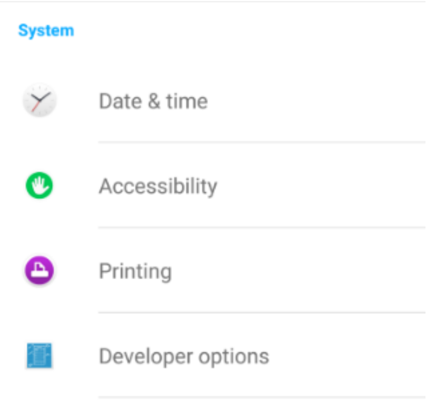
6.2 Connecting via USB

You should download Google USB Driver in the Android SDK Manager.
If this driver doesn't work you must search for a specific driver for your device.

To be able to connect a device with USB you must activate USB Debugging.
This is also need if you use an Emulator.

In this state on some older devices you will not be able to access the SD card from the PC.
If you want to access the SD card you must uncheck USB Debugging.

Launch Settings



System

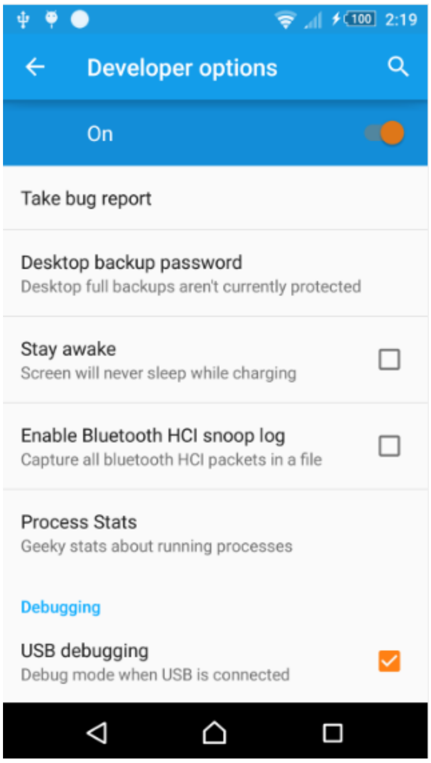
Date & time

Accessibility

Printing

Developer options

In System, select Developer options.



Developer options

On

Take bug report

Desktop backup password
Desktop full backups aren't currently protected

Stay awake ☐
Screen will never sleep while charging

Enable Bluetooth HCI snoop log ☐
Capture all bluetooth HCI packets in a file

Process Stats
Geeky stats about running processes

Debugging

USB debugging ☑
Debug mode when USB is connected

Select On.

Check USB Debugging

The device will automatically be recognized by the IDE.

7 Emulators

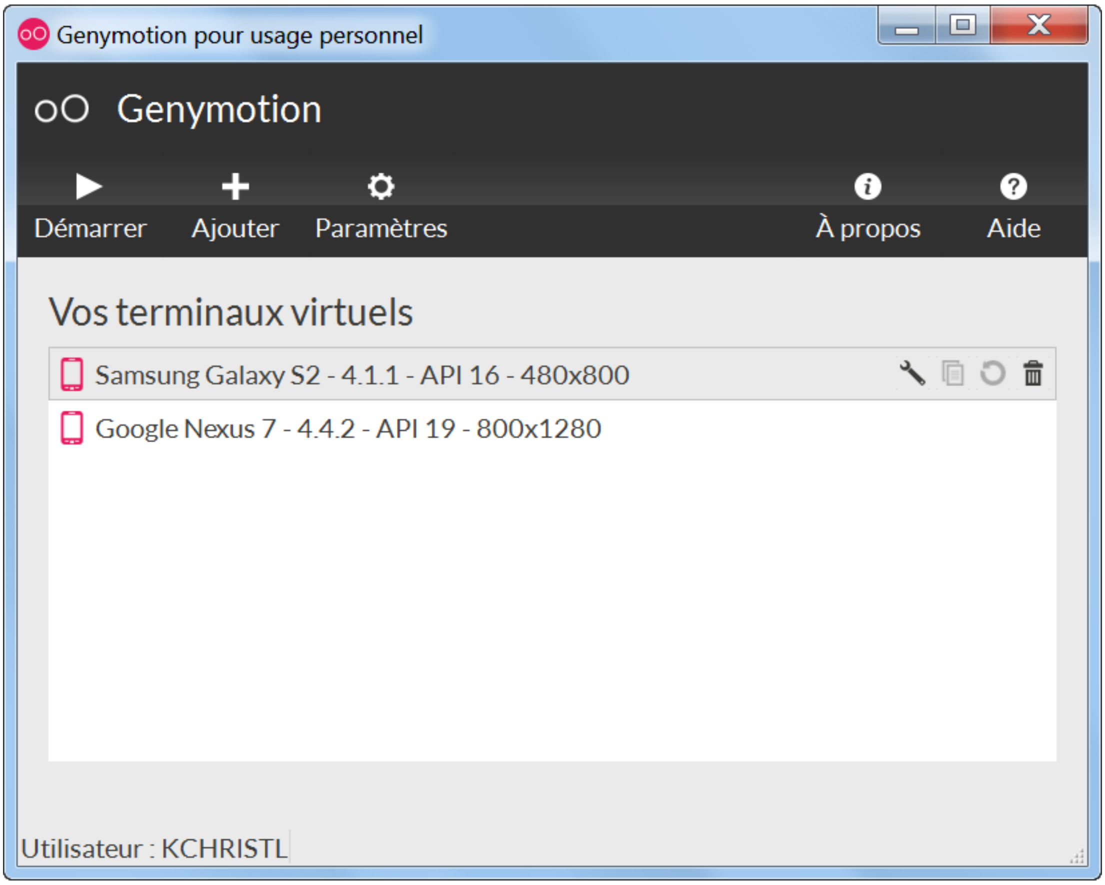
The Emulator or Virtual Device is a program that simulates devices on the PC.
It is always better to use real devices to test your projects. But in some cases, to test it on other 'devices' it could be useful.

There exist two emulators:

- The Android Emulator, which is very very slow you should use it only if there is no other solution.
- The Genymotion Emulator, it's a third party emulator much faster than the Android emulator.

7.1 Genymotion Emulator

You can download the Genymotion Emulator HERE.


Genymotion pour usage personnel

Vos terminaux virtuels

Samsung Galaxy S2 - 4.1.1 - API 16 - 480x800

Google Nexus 7 - 4.4.2 - API 19 - 800x1280

Utilisateur : KCHRISTL

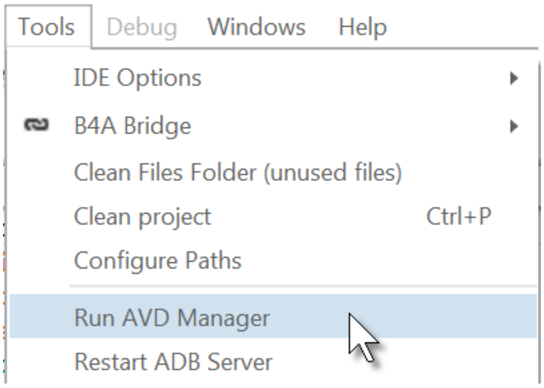
The User's Guide is HERE.

7.2 Android Emulator

7.2.1 Create a new Emulator

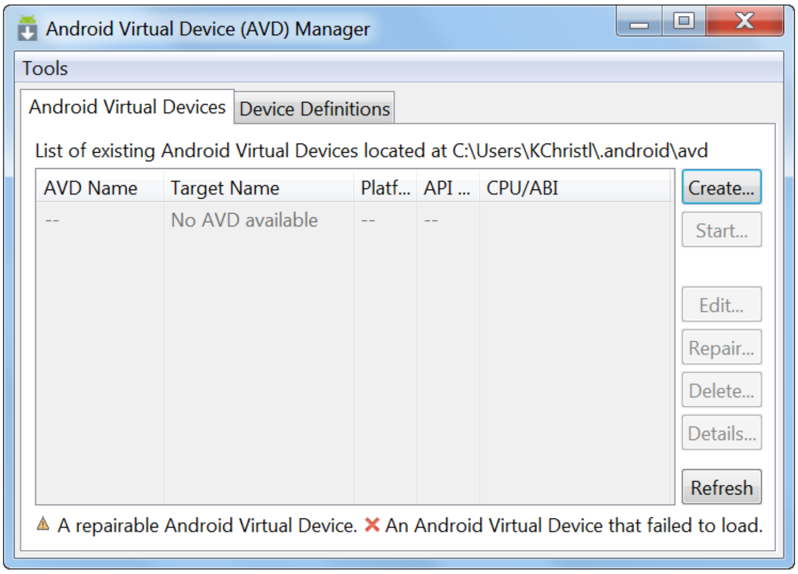
Let us add a new Emulator with a resolution of 480 / 800 pixels, density 240.

In the IDE menu Tools click on Run AVD Manager to run the AVD Manager.



Be patient it’s everything but fast.

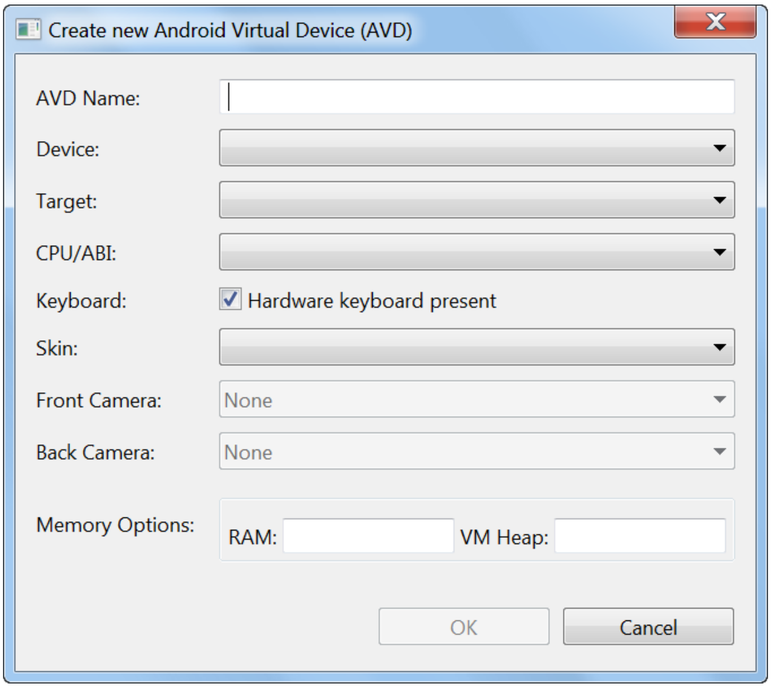
In the AVD Manager Click on Create... .



List of existing Android Virtual Devices located at C:\Users\KChristl\.android\avd

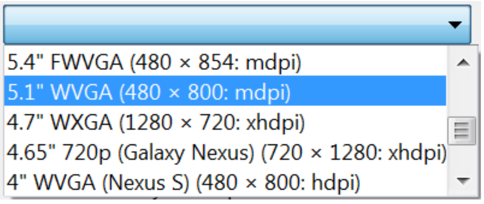
AVD Name	Target Name	Platf...	API ...	CPU/ABI
--	No AVD available	--	--	

⚠ A repairable Android Virtual Device. ✕ An Android Virtual Device that failed to load.

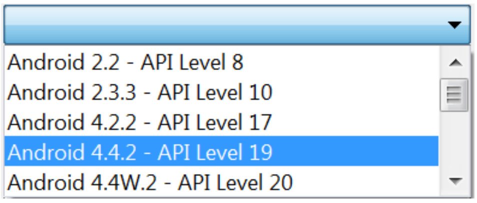


AVD Name:
Enter the name *Emul480_800*
No spaces !

Device: Select 5.1 WVGA

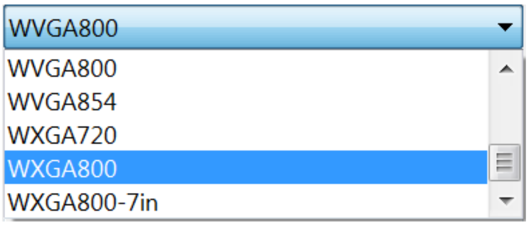

5.4" FWVGA (480 × 854: mdpi)
5.1" WVGA (480 × 800: mdpi)
4.7" WXGA (1280 × 720: xhdpi)
4.65" 720p (Galaxy Nexus) (720 × 1280: xhdpi)
4" WVGA (Nexus S) (480 × 800: hdpi)

Target:
Select Android 4.4.2 – API Level 19


Android 2.2 - API Level 8
Android 2.3.3 - API Level 10
Android 4.2.2 - API Level 17
Android 4.4.2 - API Level 19
Android 4.4W.2 - API Level 20

CPU/ABI: Will be set automatically.

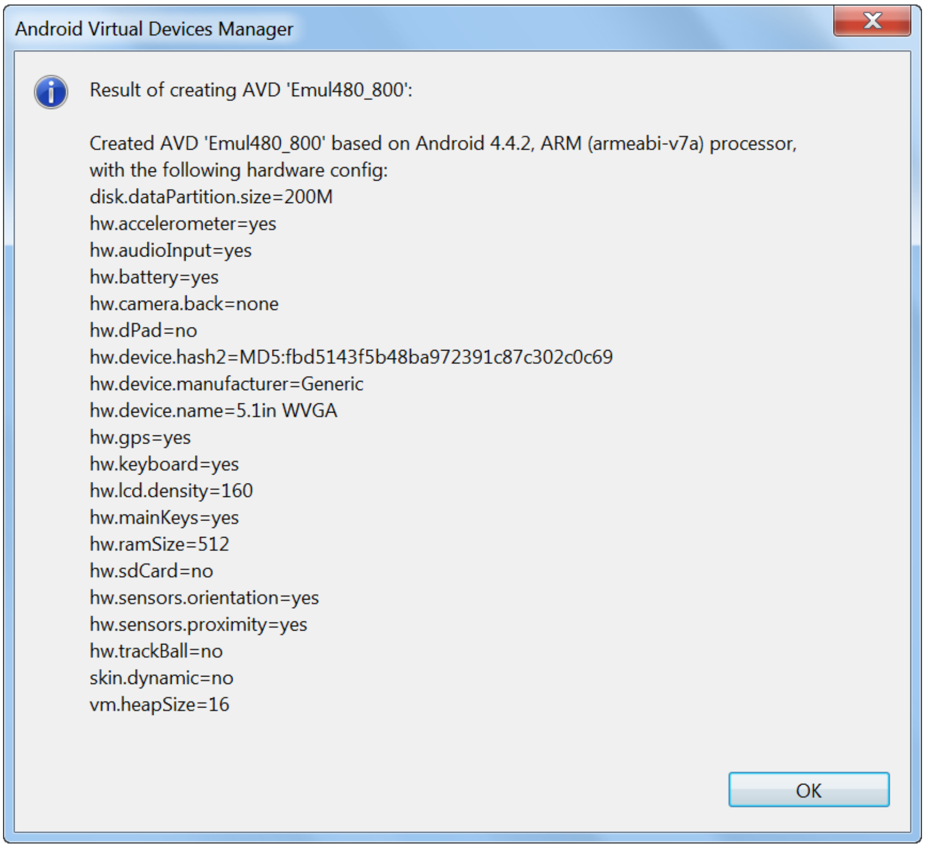
Skin: Select WVGA800



HVGA	320 / 480
QVGA	240 / 320
WQVGA400	240 / 400
WQVGA432	240 / 432
WVGA800	480 / 800
WVGA854	480 / 854

Memory Options: Are set by default, but can be changed.
RAM: 512 VM Heap : 16

You will see a window similar to this.

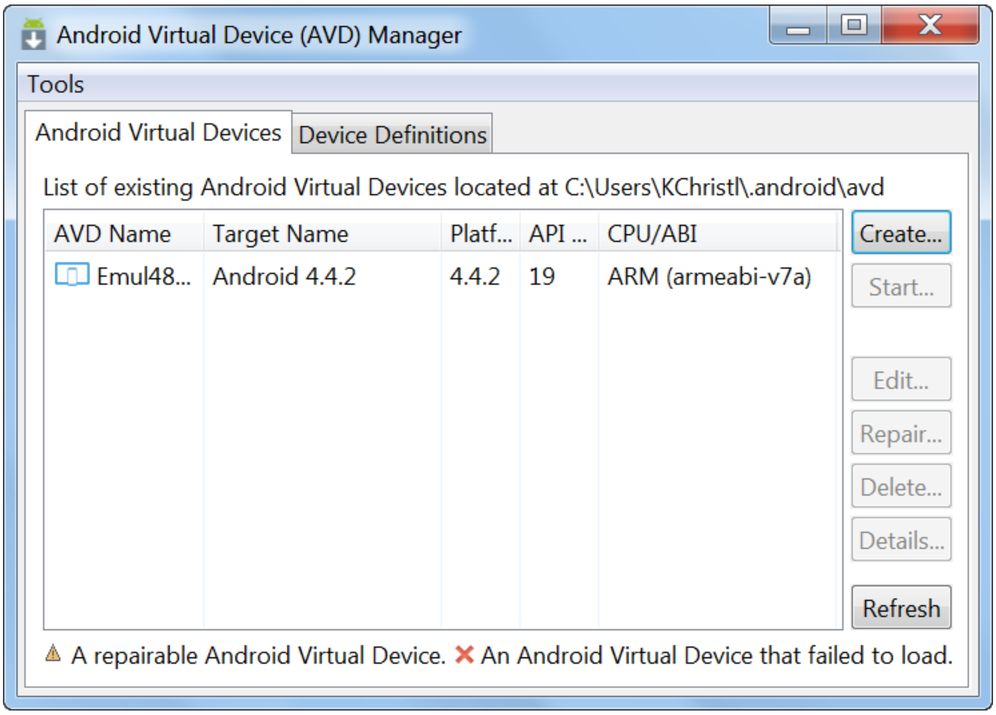


Result of creating AVD 'Emul480_800':

```
Created AVD 'Emul480_800' based on Android 4.4.2, ARM (armeabi-v7a) processor,
with the following hardware config:
disk.dataPartition.size=200M
hw.accelerometer=yes
hw.audioInput=yes
hw.battery=yes
hw.camera.back=none
hw.dPad=no
hw.device.hash2=MD5:fbd5143f5b48ba972391c87c302c0c69
hw.device.manufacturer=Generic
hw.device.name=5.1in WVGA
hw.gps=yes
hw.keyboard=yes
hw.lcd.density=160
hw.mainKeys=yes
hw.ramSize=512
hw.sdCard=no
hw.sensors.orientation=yes
hw.sensors.proximity=yes
hw.trackBall=no
skin.dynamic=no
vm.heapSize=16
```

Click OK

The new Emulator is added.



List of existing Android Virtual Devices located at C:\Users\KChristl\.android\avd

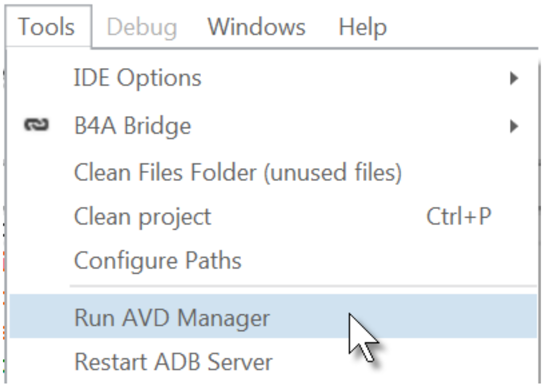
AVD Name	Target Name	Platf...	API ...	CPU/ABI
Emul48...	Android 4.4.2	4.4.2	19	ARM (armeabi-v7a)

⚠ A repairable Android Virtual Device. ✕ An Android Virtual Device that failed to load.

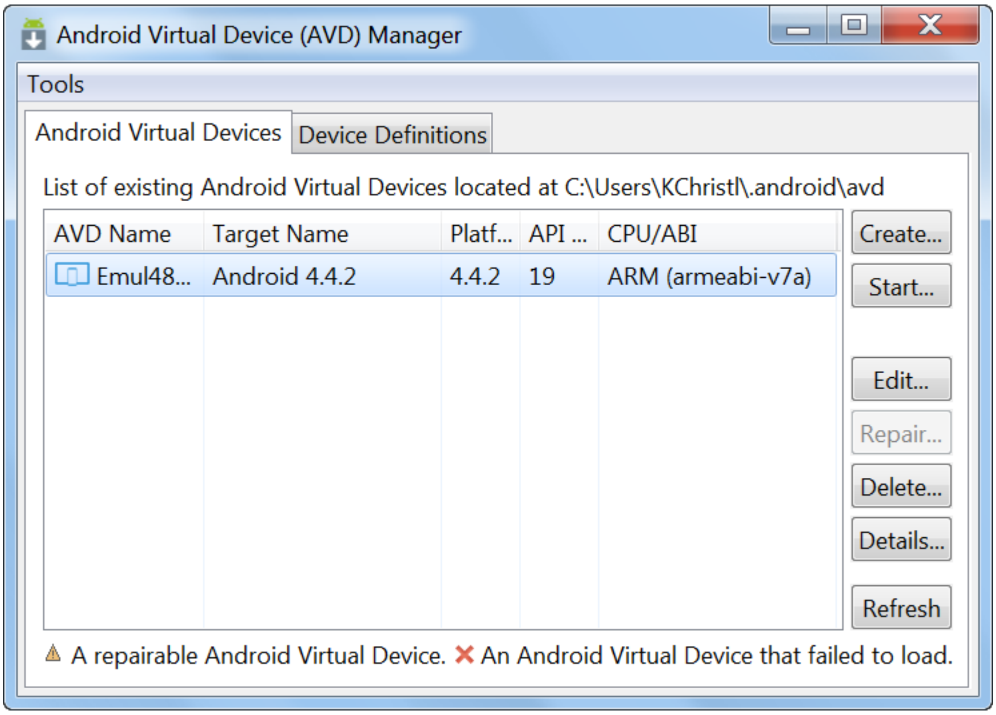
7.2.2 Launch an Android Emulator

To launch an Emulator click in the IDE in the Tools menu on Run AVD Manager.

Select the desired emulator.



The Android 4.4.2 portrait Emulator in this case.

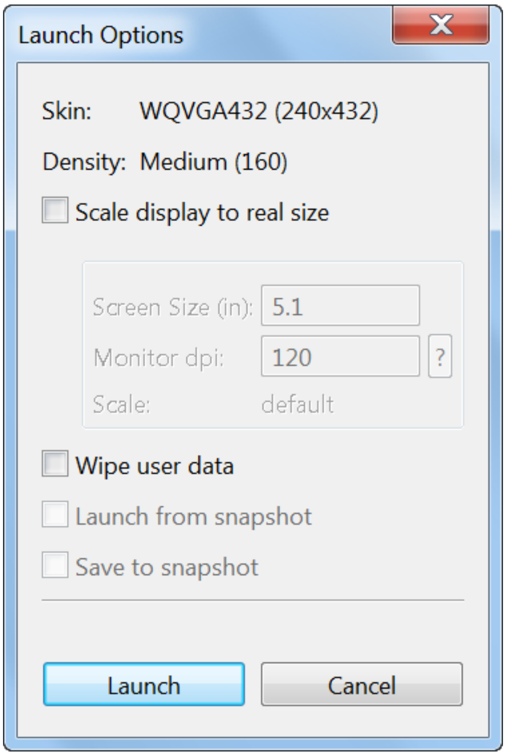


List of existing Android Virtual Devices located at C:\Users\KChristl\.android\avd

AVD Name	Target Name	Platf...	API ...	CPU/ABI
Emul48...	Android 4.4.2	4.4.2	19	ARM (armeabi-v7a)

⚠ A repairable Android Virtual Device. ✖ An Android Virtual Device that failed to load.

Click on Start...



Launch Options

Skin: WQVGA432 (240x432)

Density: Medium (160)

□ Scale display to real size

Screen Size (in): 5.1

Monitor dpi: 120

Scale: default

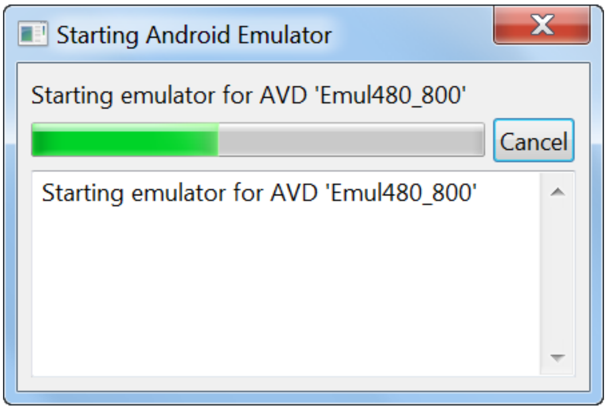
□ Wipe user data

□ Launch from snapshot

□ Save to snapshot

Click on

You will see a window like this:

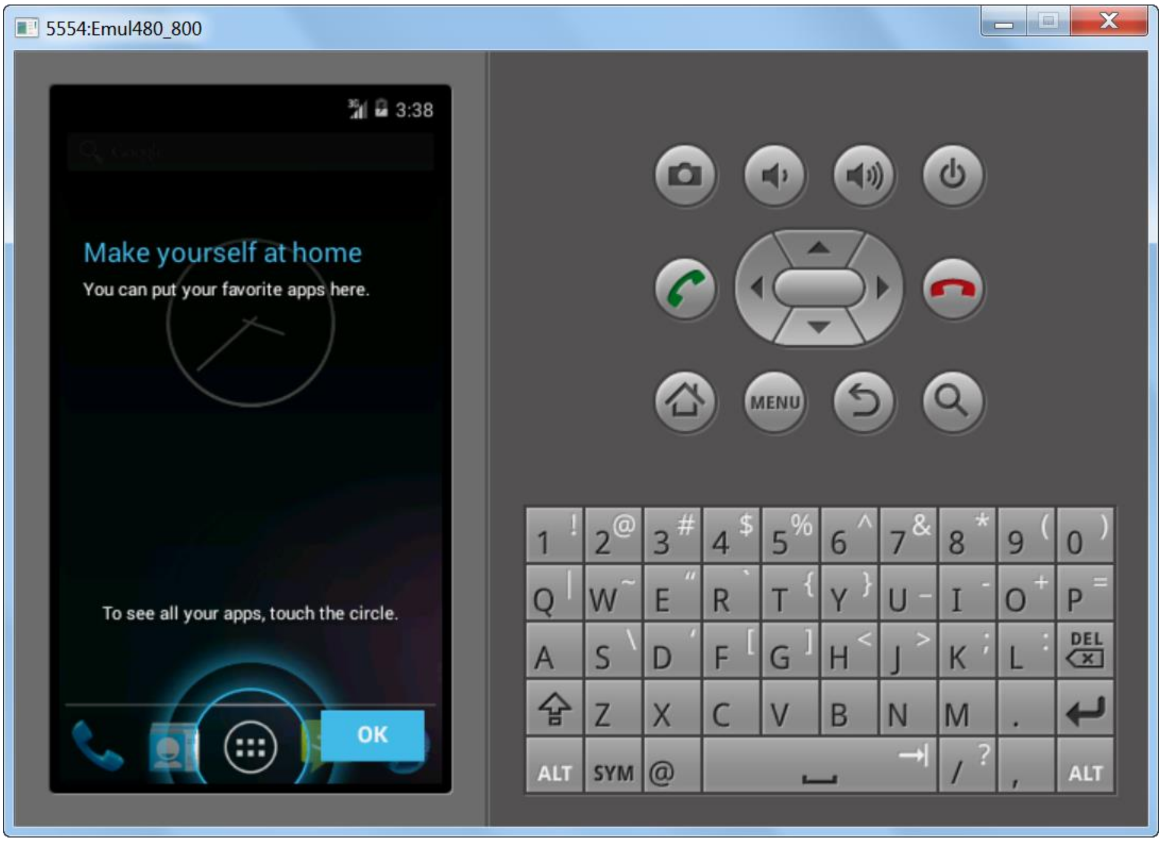


Starting Android Emulator

Starting emulator for AVD 'Emul480_800'

Starting emulator for AVD 'Emul480_800'

Wait until the Emulator is ready, this will take quite some time can be several minutes !?.
The screen is different for different versions of Android.



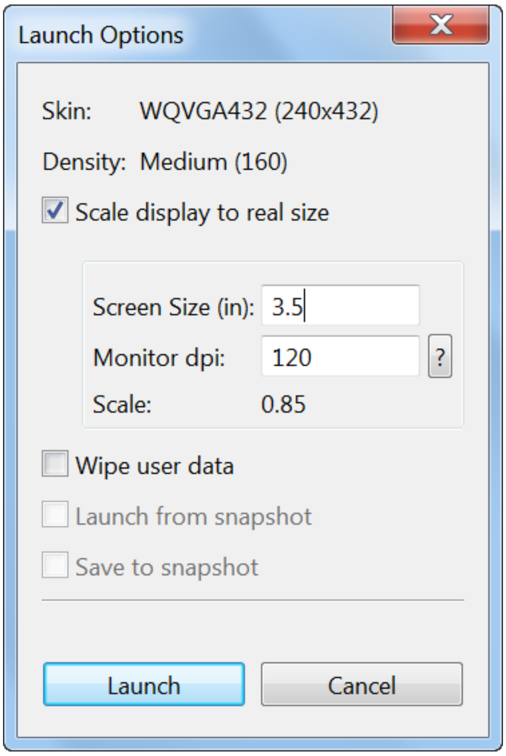
5554:Emul480_800

Make yourself at home

You can put your favorite apps here.

To see all your apps, touch the circle.

The physical size of the Emulator on the computer screen can be changed in the Launch Option window. This can be useful for big screen emulators.



Launch Options

Skin: WQVGA432 (240x432)

Density: Medium (160)

☑ Scale display to real size

Screen Size (in): 3.5

Monitor dpi: 120

Scale: 0.85

□ Wipe user data

□ Launch from snapshot

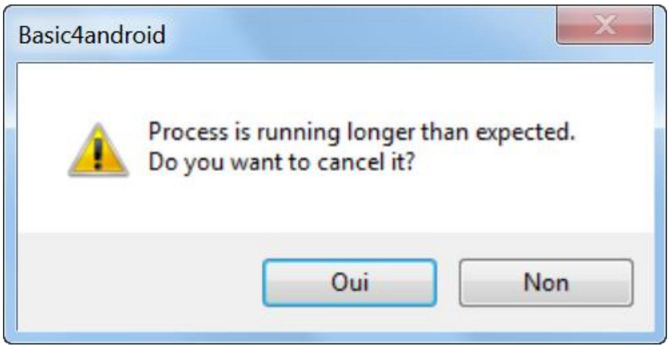
□ Save to snapshot

In the example the size is set to 3.5 inches.

7.2.3 Android Emulator problems

Unfortunately, the Emulator is quite slow and sometimes a pain.

When you either run the program or connect to the Emulator from the Designer, sometimes you will see the message below.



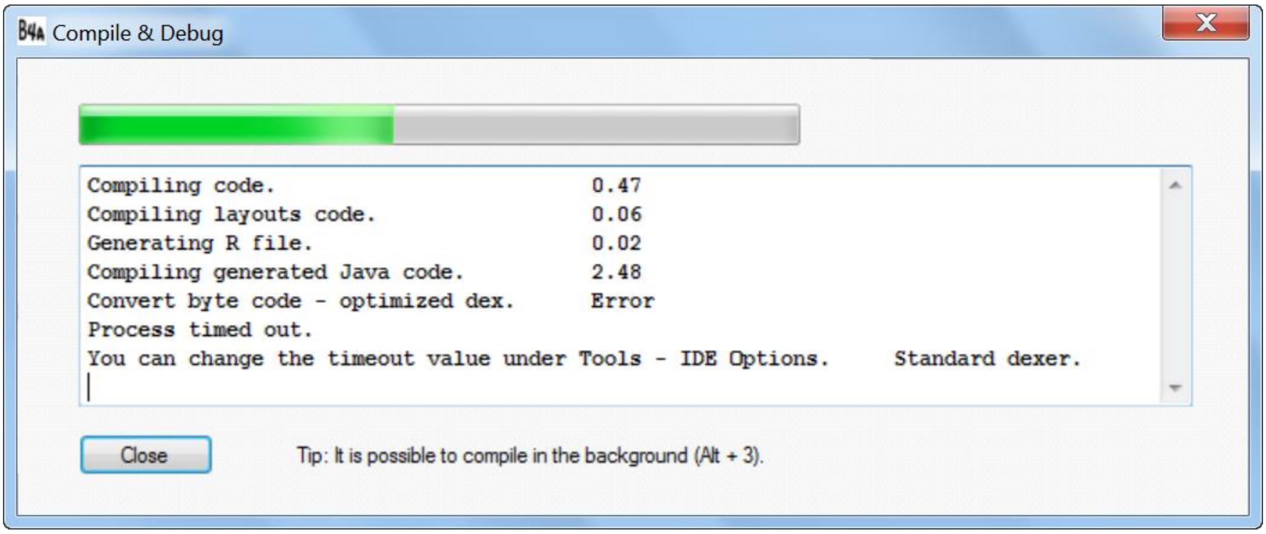
Basic4android

Process is running longer than expected.
Do you want to cancel it?

You have two options:
- Yes (Oui) to cancel the process.
- No (Non) Continue the process.

Most times when clicking Non, the process will succeed.

However, even after having clicked Non, sometimes you will see following message.



Compile & Debug

```
Compiling code.                         0.47
Compiling layouts code.                 0.06
Generating R file.                      0.02
Compiling generated Java code.          2.48
Convert byte code - optimized dex.      Error
Process timed out.
You can change the timeout value under Tools - IDE Options.     Standard dexer.
```

Tip: It is possible to compile in the background (Alt + 3).

In most cases, if you run the program once more, the connection to the Emulator will be established and it will work properly.

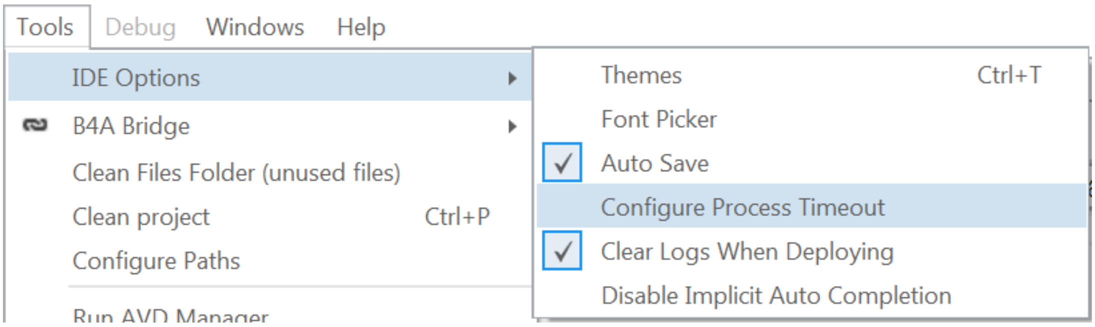
This often happens when the Emulator is still running a program or if the Emulator is still connected to another project. In this case press the back button until you reach the Emulator's home screen and try again.

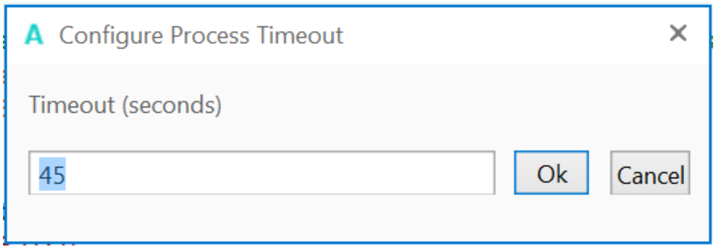
If this happens for a second time, close the current Emulator and run it again from the AVD Manager.

If the first message above appears too often you can increase the process timeout value.

7.2.4 Process timeout

In the IDE Tools menu.





Timeout (seconds)

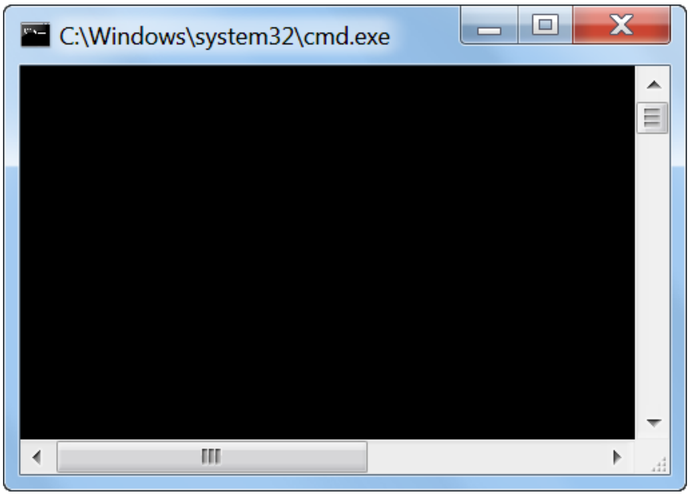
45

Set the ProcessesTimeout (seconds) parameter to a higher value.
I set it to 45 seconds.

7.2.5 Exchanging files with the PC

To get access to files in the Emulator you can use the Dalvik Debug Monitor.
The name is ddms.bat and it is located in the folder where you copied the Android SDK.
Example: C:\Android\android-sdk-windows\tools.

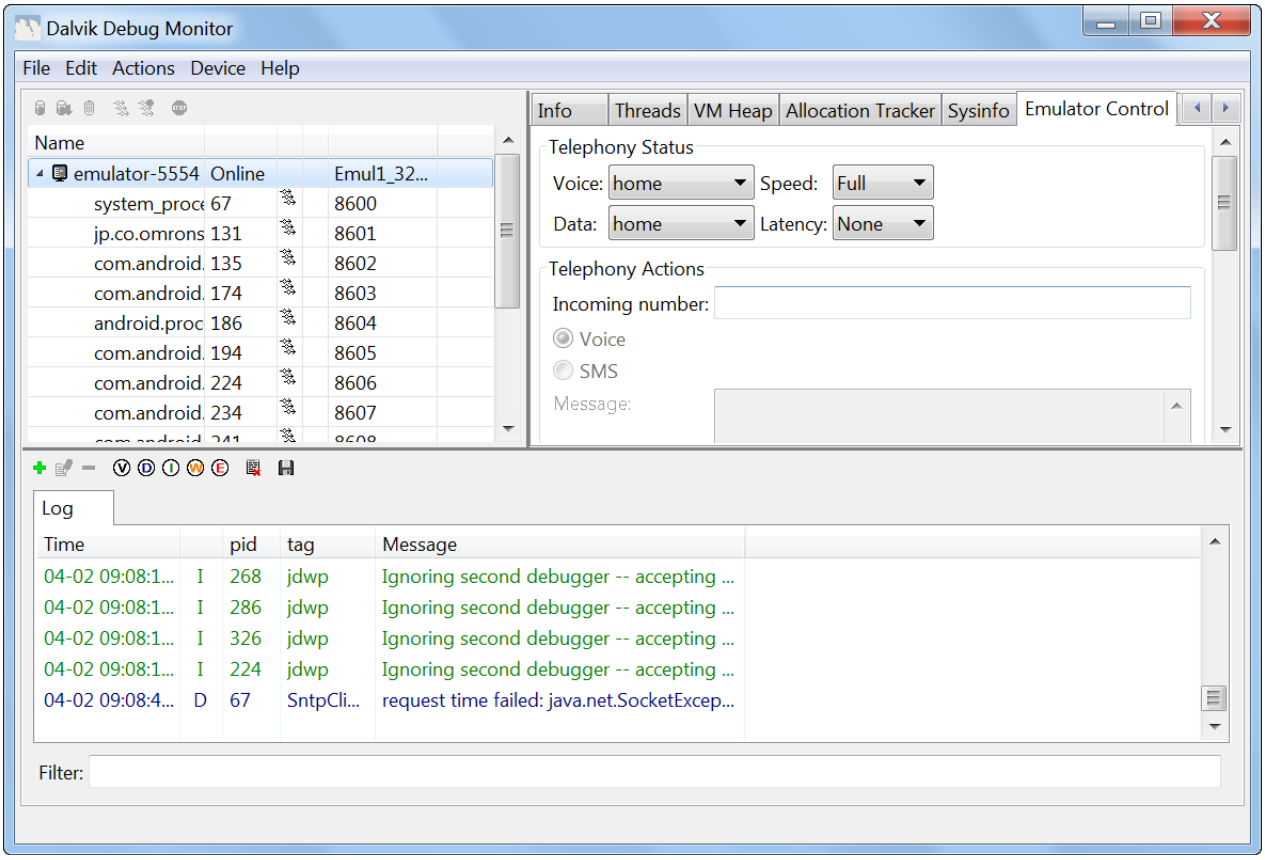
Make sure that Emulator is running.
Run the ddms.bat file:



A window like this one will appear.

Wait a moment.

The Dalvik Debug Monitor will be displayed.



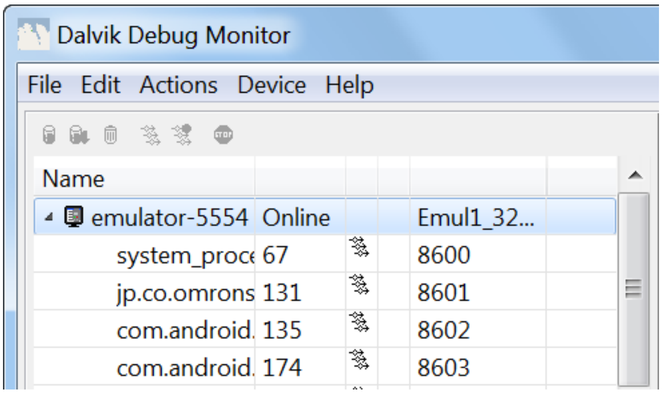
Name			
emulator-5554	Online		Emul1_32...
system_proce	67		8600
jp.co.omrons	131		8601
com.android.	135		8602
com.android.	174		8603
android.proc	186		8604
com.android.	194		8605
com.android.	224		8606
com.android.	234		8607
com.android.	241		8608

Voice: home Speed: Full
Data: home Latency: None

Incoming number:
○ Voice
○ SMS
Message:

Time		pid	tag	Message
04-02 09:08:1...	I	268	jdwp	Ignoring second debugger -- accepting ...
04-02 09:08:1...	I	286	jdwp	Ignoring second debugger -- accepting ...
04-02 09:08:1...	I	326	jdwp	Ignoring second debugger -- accepting ...
04-02 09:08:1...	I	224	jdwp	Ignoring second debugger -- accepting ...
04-02 09:08:4...	D	67	SntpCli...	request time failed: java.net.SocketExcep...

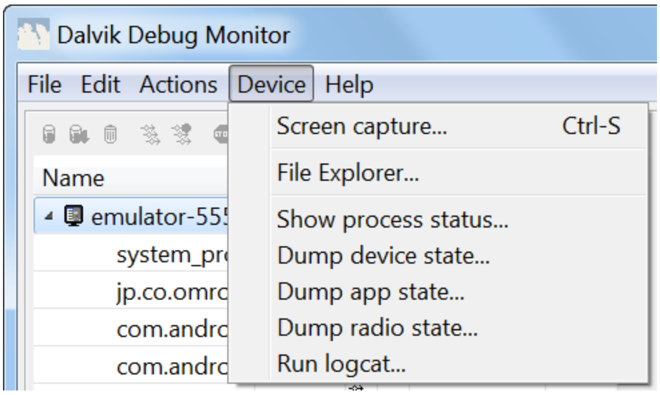
Filter:



Name			
emulator-5554	Online		Emul1_32...
system_proce	67		8600
jp.co.omrons	131		8601
com.android.	135		8602
com.android.	174		8603

In the upper left corner you should see a reference to the Emulator.

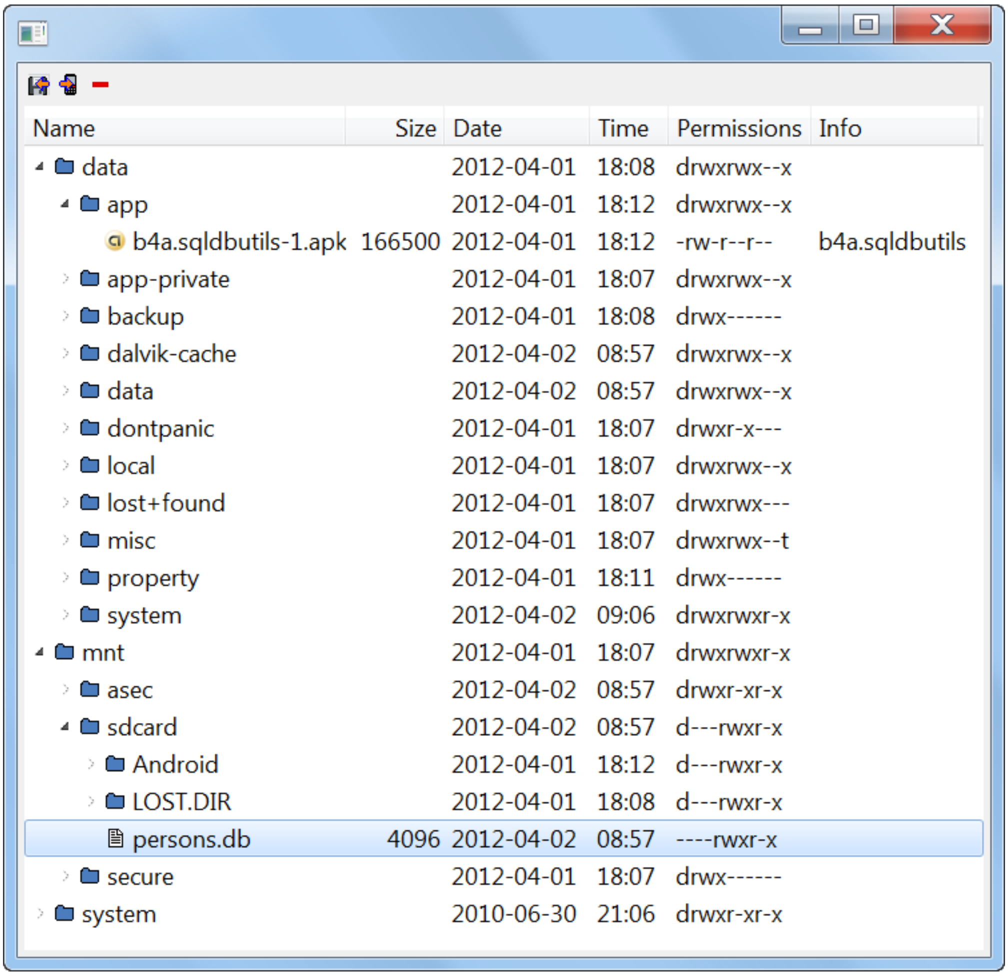
Select it.



Then in the menu Device

select File Explorer...

The Device File Explorer will be displayed:



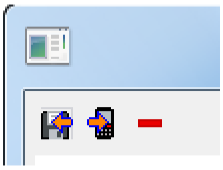
Name	Size	Date	Time	Permissions	Info
data		2012-04-01	18:08	drwxrwx--x	
app		2012-04-01	18:12	drwxrwx--x	
b4a.sqldbutils-1.apk	166500	2012-04-01	18:12	-rw-r--r--	b4a.sqldbutils
app-private		2012-04-01	18:07	drwxrwx--x	
backup		2012-04-01	18:08	drwx------	
dalvik-cache		2012-04-02	08:57	drwxrwx--x	
data		2012-04-02	08:57	drwxrwx--x	
dontpanic		2012-04-01	18:07	drwxr-x---	
local		2012-04-01	18:07	drwxrwx--x	
lost+found		2012-04-01	18:07	drwxrwx---	
misc		2012-04-01	18:07	drwxrwx--t	
property		2012-04-01	18:11	drwx------	
system		2012-04-02	09:06	drwxrwxr-x	
mnt		2012-04-01	18:07	drwxrwxr-x	
asec		2012-04-02	08:57	drwxr-xr-x	
sdcard		2012-04-02	08:57	d---rwxr-x	
Android		2012-04-01	18:12	d---rwxr-x	
LOST.DIR		2012-04-01	18:08	d---rwxr-x	
persons.db	4096	2012-04-02	08:57	----rwxr-x	
secure		2012-04-01	18:07	drwx------	
system		2010-06-30	21:06	drwxr-xr-x	

You see several folders.

In data\app you'll find applications.

mnt\sdcard is the DirRootExternal folder.
In the example the file persons.db is a database copied in a B4A program from DirAssets to DirRootExternal.

In the upper left corner you see three icons:



Pull file from device, copies the file to the PC

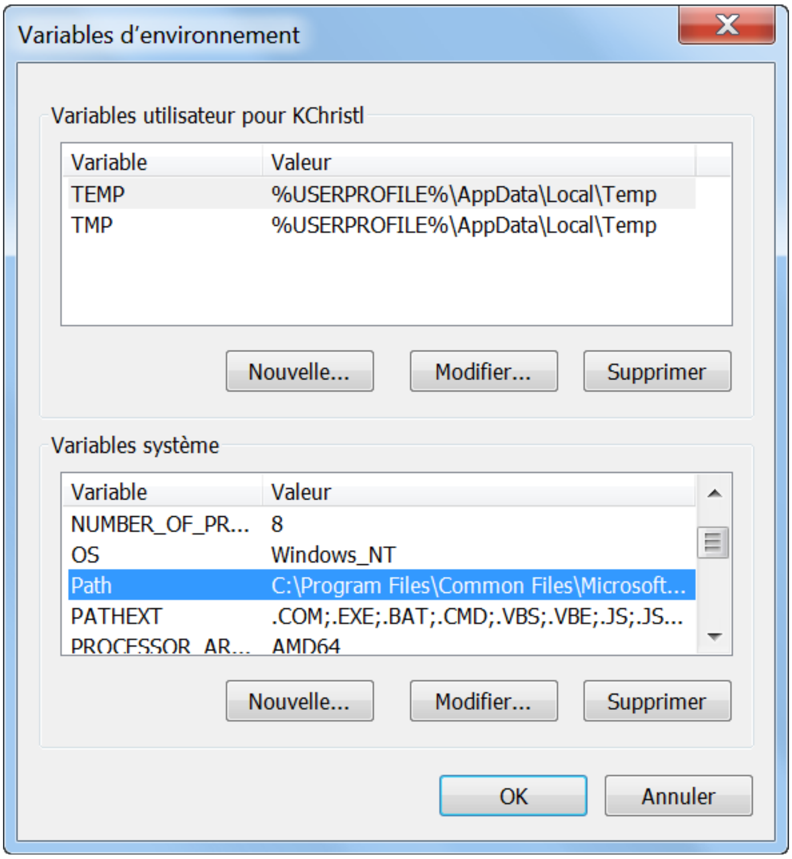
Push file onto device, copies a file to the device

− Deletes the file

Clicking on either or shows the standard Windows file explorer to select the destination or source folder for the selected file.

If the Dalvik Debug Monitor doesn't run you need to add the path where the ddms.bat file is located to the environment variables.

- From the desktop, right-click **My Computer** and click **Properties**.
- In the System Properties window, click on the **Advanced tab**.
- In the Advanced section, click the **Environment Variables** button.
- Finally, in the Environment Variables window (as shown below), highlight the **Path** variable in the Systems Variable section and click the **Edit** (Modifier) button. Add or modify the path lines with the paths you wish the computer to access. Each different directory is separated with a semicolon as shown below.



Variables d'environnement

Variables utilisateur pour KChristl

Variable	Valeur
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

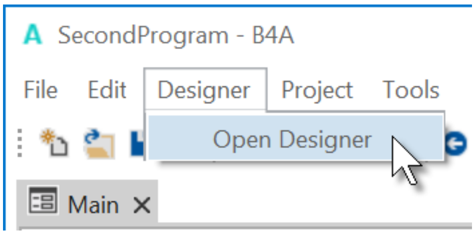
Variables système

Variable	Valeur
NUMBER_OF_PR...	8
OS	Windows_NT
Path	C:\Program Files\Common Files\Microsoft...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JS...
PROCESSOR_AR...	AMD64

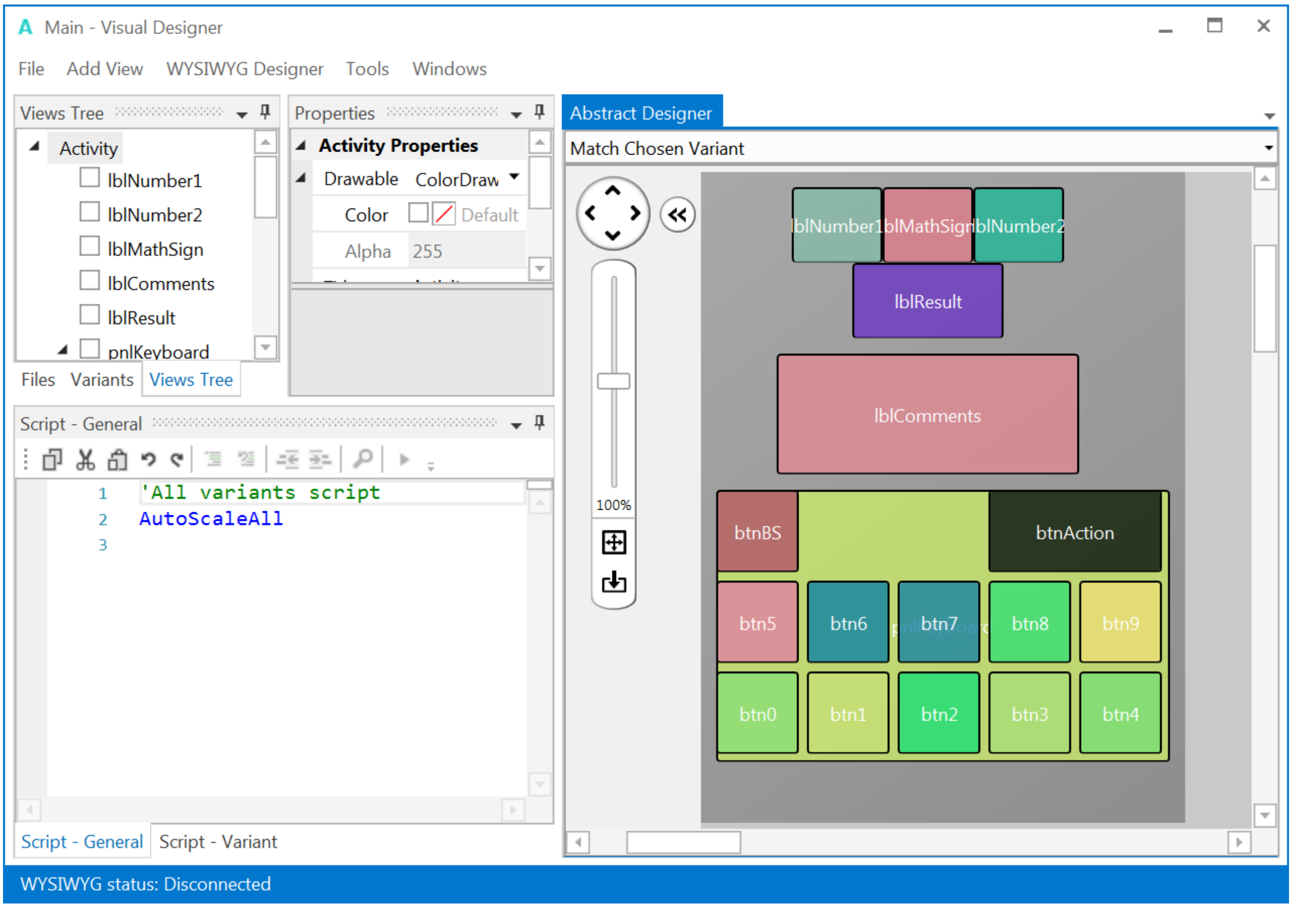
8 The Visual Designer

The Visual Designer allows generating layouts with either the Abstract Designer or with a real device. You can also use Emulators.

Launch the Designer in the IDE Menu Designer.



The default Visual Designer looks like this, the layout in the Abstract Designer is from the SecondProgram project.



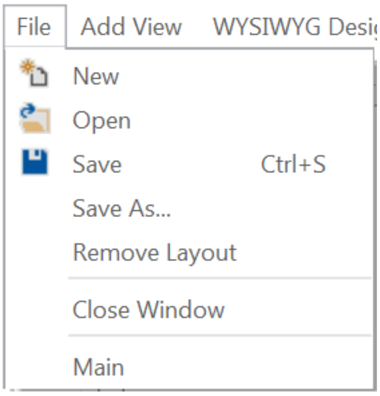
Drawable: ColorDrawable
Color: Default
Alpha: 255

```
'All variants script
AutoScaleAll
```

WYSIWYG status: Disconnected

8.1 The menu

8.1.1 File menu



New Opens a new empty layout.

Open Opens an existing layout.

Save Saves the current layout.

Save As… Saves the current layout with a new name.

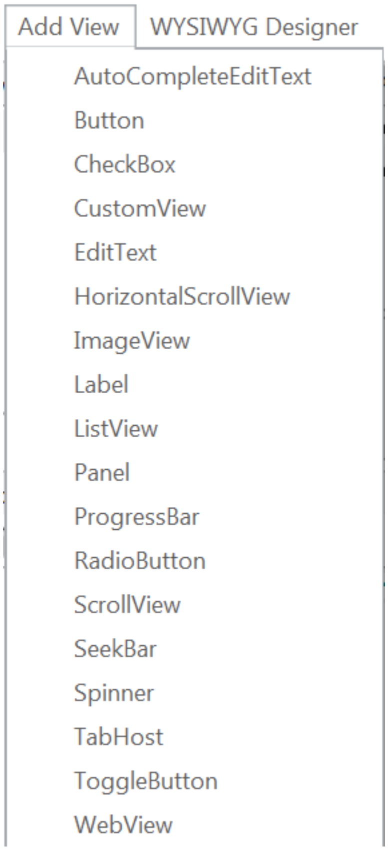
Remove Layout Removes the layout from the Files directory.

Close Window Closes the Visual Designer.

Main Layout file list, in this case only one file, 'Main'.

8.1.2 AddView menu

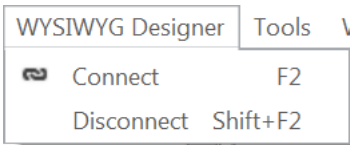
This menu allows you to add views to the current layout.



AutoCompleteEditText	adds an AutoCompleteEditText
Button	adds a Button
CheckBox	adds a CheckBox
CustomView	adds a CustomView
EditText	adds an EditText
HorizontalScrollView	adds a HorizontalScrollView
ImageView	adds an ImageView
Label	adds a Label
ListView	adds a ListView
Panel	adds a Panel
ProgressBar	adds a ProgressBar
RadioButton	adds a RadioButton
ScrollView	adds a Scrollview
SeekBar	adds a SeekBar
Spinner	adds a Spinner
TabHost	adds a TabHost
ToggleButton	adds a ToggleButton
WebView	adds a WebView

8.1.3 WYSIWYG Designer menu

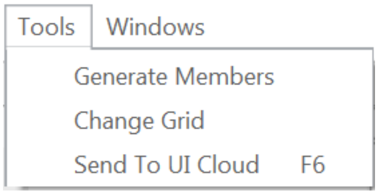
Connects a real device or an Emulator.



Connects a device or an Emulator to the Visual Desiger.
Disconnect From Device / Emulator.

For details on how to connect a device look at chapter
6 Connecting a real device or at chapter 7 Emulators.

8.1.4 The Tools menu

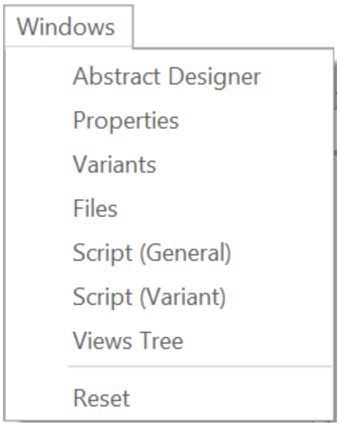


Generate Members	Members generator
Change Grid	Allows to change the grid size

Send To UI Cloud.

Bring to Front	Brings the selected View to front
Send To Back	Sends the selected View to back
Duplicate Selected View	Duplicates the selected View
Remove Selected View	Removes the selected View

8.1.5 Windows menu



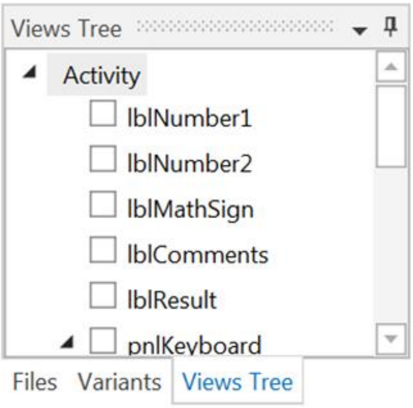
Shows the Abstract Designer window.
Shows the Properties window.
Shows the Variants window.
Shows the Files window.
Shows the Script (General) window.
Shows the Script (Variant) window.
Shows the Views window.

Resets the Visual Designer layout to the default layout.

8.2 Visual Designer Windows

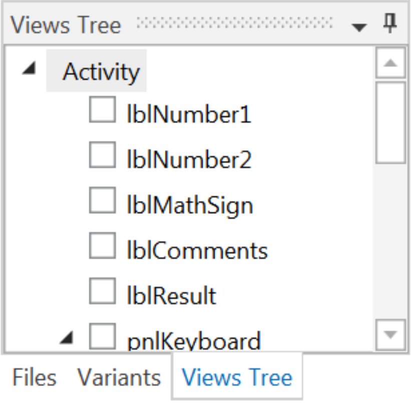
The Visual Designer is composed of different windows.

8.2.1 Views windows Views Tree / Files / Variants



In this Window three windows are combined:
Files, Variants and Views Tree.

8.2.1.1 Views Tree window



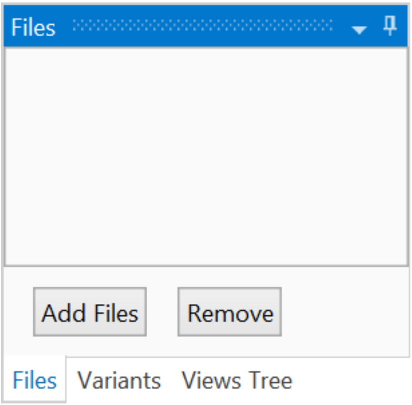
Shows all views of the selected layout in a tree.

When you select a view in the list, all the properties of the selected view are displayed in the Properties window.

You can select several Views at the same time and change common properties.

The selected views are highlighted in the Abstract Designer.

8.2.1.2 Files Windows

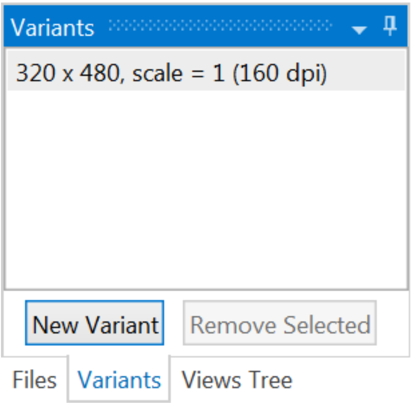


Used to add or remove files to the Visual Designer, mainly image files.

File handling is explained in the Image Files chapter.

These files are copied to the Files folder of the project and can be accessed in the code in the File.DirAssets folder.

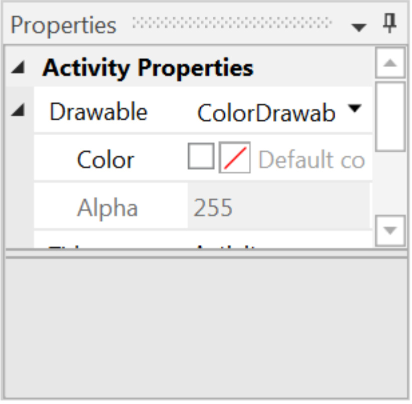
8.2.1.3 Variants window



Used to add and remove layout variants.

Layout variants are explained in the Layout variants chapter.

8.2.2 Properties window



Drawable	ColorDrawab
Color	Default co
Alpha	255

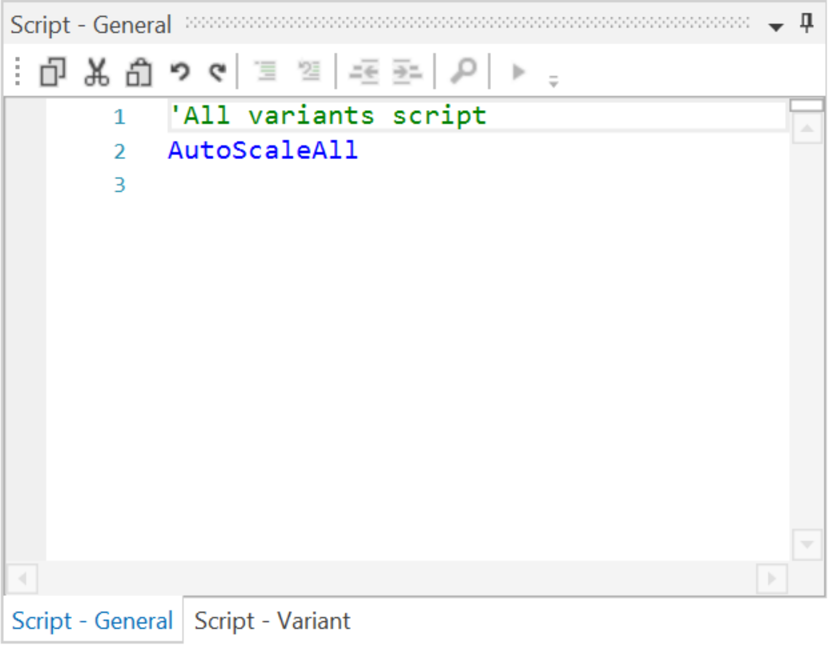
The Properties window shows all properties of the selected View.

The Properties are explained in the Properties list chapter.

8.2.3 Script (General) / (Variant) windows

In the Scrip windows you can add code to position and resize Views.
Two windows are available:

- **Script - General** Code valid for all layout variants.
- **Script - Variant** Specific code for the selected variant.



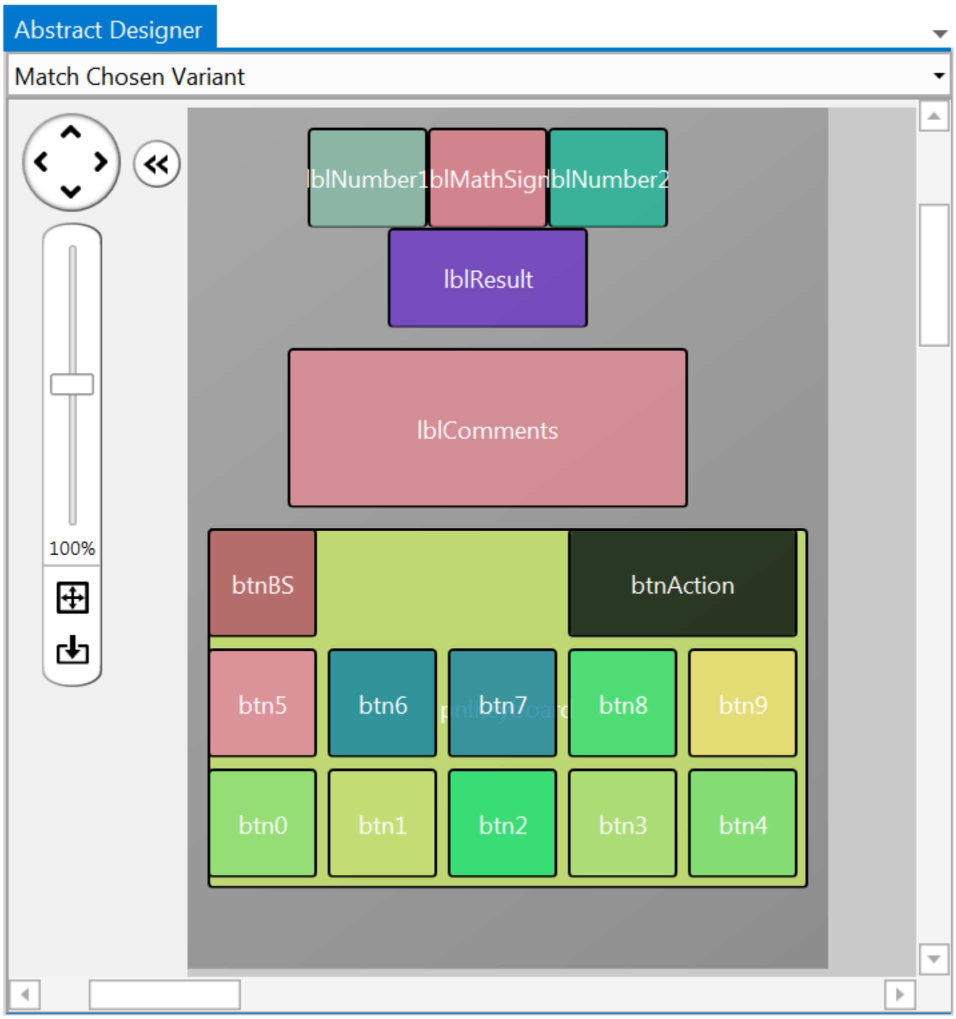
```
'All variants script
AutoScaleAll
```

Script code is explained in the Designer Scripts chapter.

8.2.4 Abstract Designer window

The Abstract Designer allows to select, position and resize Views.
It is not a WYSIWYG Designer, for this you need to connect a real device or an Emulator.

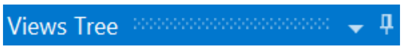
The displayed layout in the picture below is from the SecondProgram project.

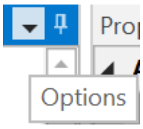


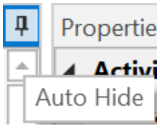
8.3 Floating windows

You can define your own Visual Designer layout, rearrange the windows in size and position, docked or floating.

On top of each window two icons allow you to manage the behaviour of this window.

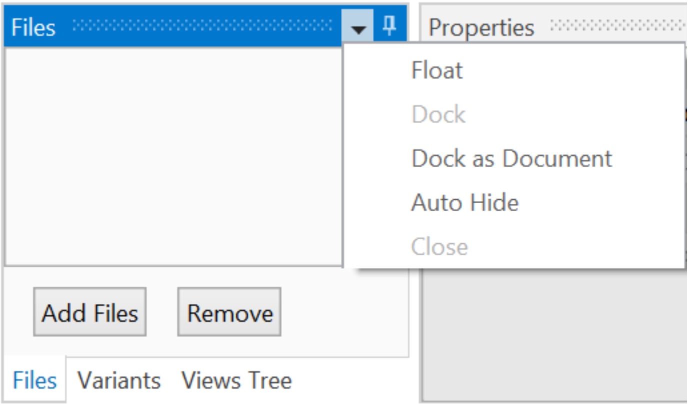






▾ Options.

Example with the Files window:

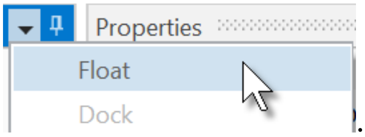


Float sets the window to Float, independent of the Visual Designer window.

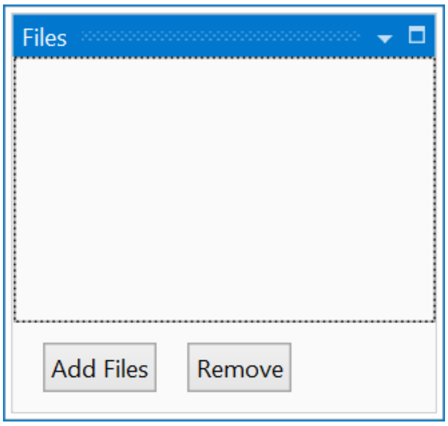
Dock as Document

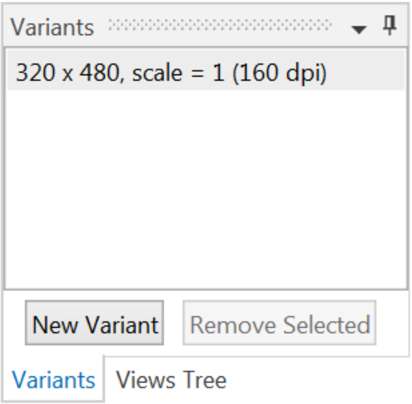
Auto Hide

8.3.1 Float

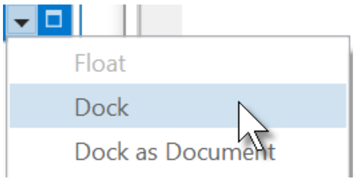
Clock on .

The Files windows is independent from the Visual Designer and is removed from the Views window.





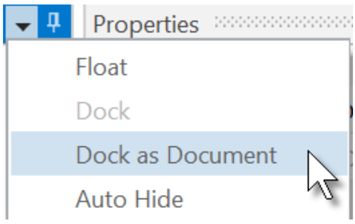
8.3.2 Dock

Click on 

.

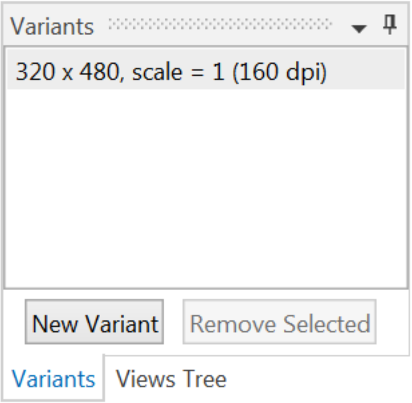
The window is moved back to the Views window.

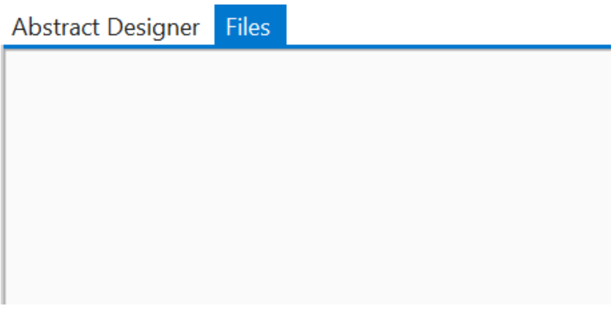
8.3.3 Dock as Document

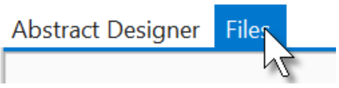
Click on 

.

The window is removed from its parent window and added to the Abstract Designer window.

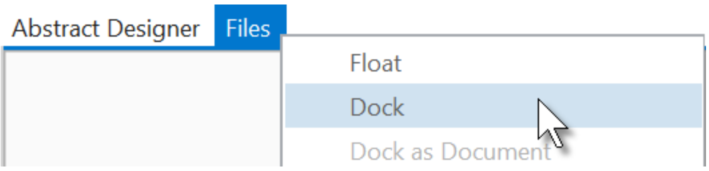




Right click on 

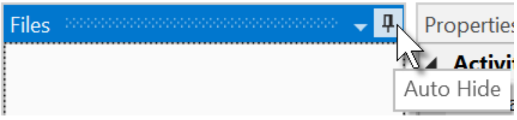
and on

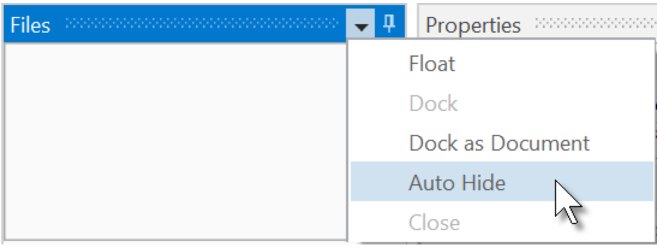
to move it back to its parent window.



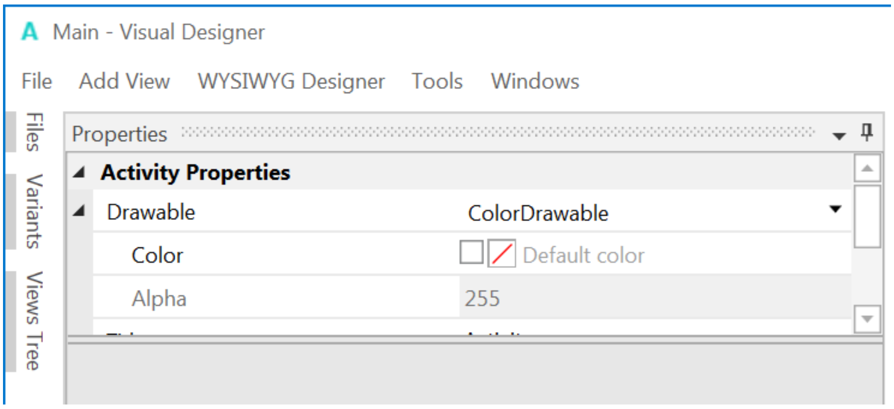
8.3.4 Auto Hide

Click either on 📌 or on Auto Hide .





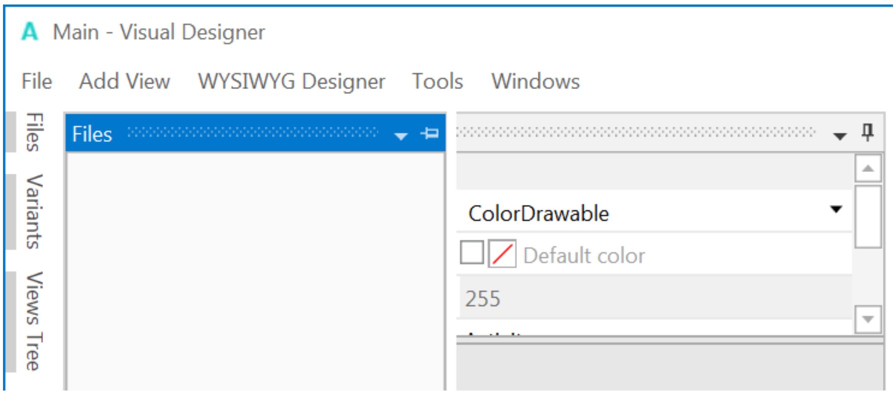
The three windows Files, Variants and Views Tree are moved as Tabs to the left border of the Visual Designer. The Properties window width is increased.



Activity Properties

Drawable	ColorDrawable
Color	□ Default color
Alpha	255

Click on a Tab to show the window.



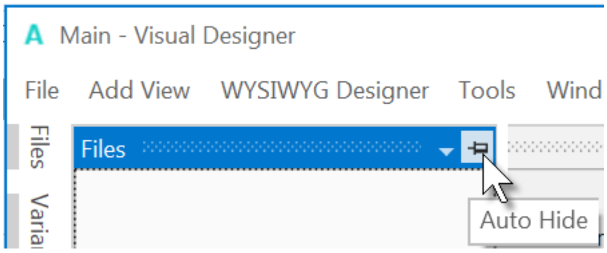
ColorDrawable

□ Default color

255

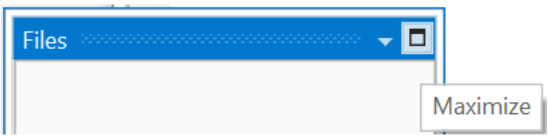
When you click somewhere else, outsides the selected window, hides it automatically.

Click on 📌 in the title to move the windows back to their previous position.



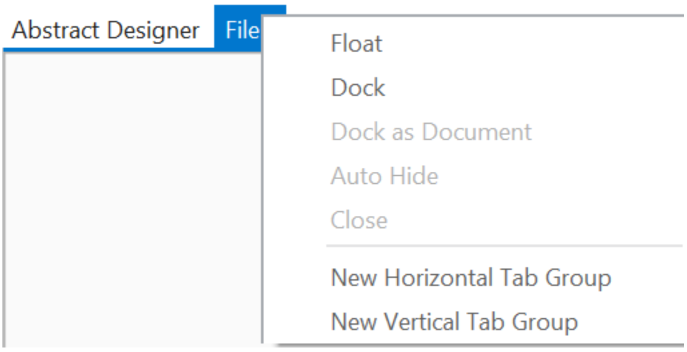
8.3.5 Maximize

Floating windows can be maximized.

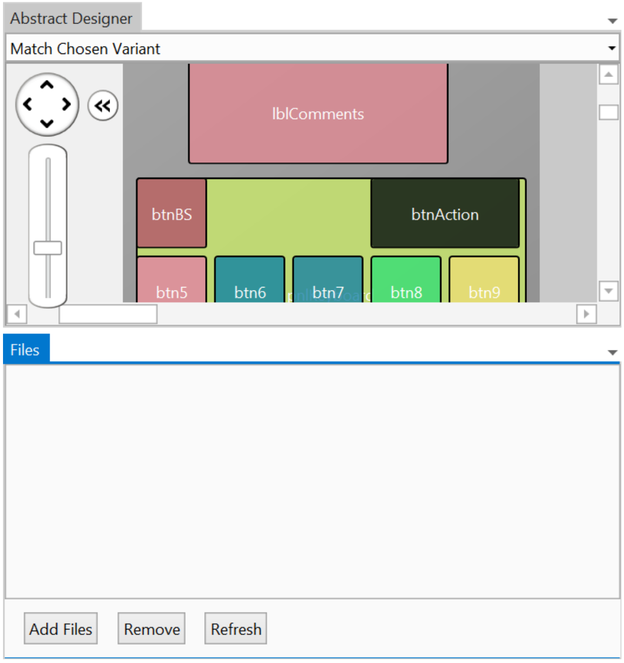


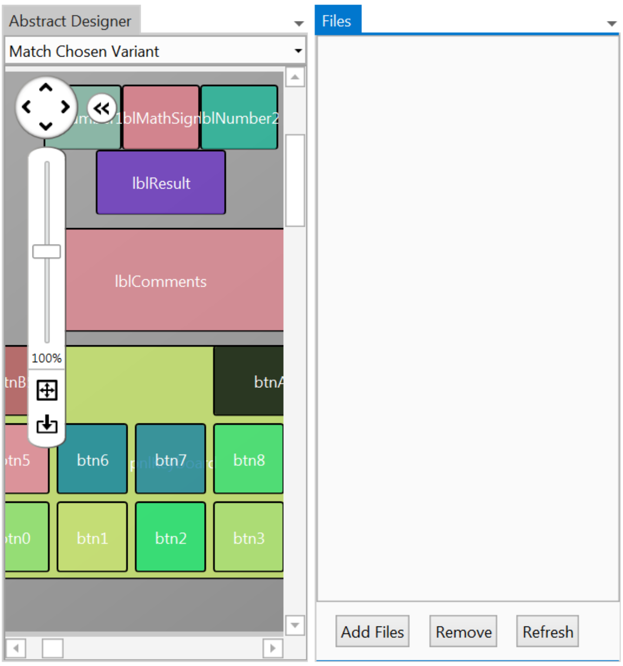
8.3.6 New Horizontal / Vertical Tab Group

When a window is set as *Dock as Document* two other options are available.



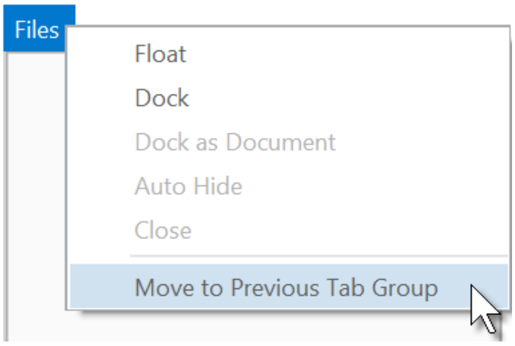
New Horizontal Tab Group
New Vertctal Tab Group





New Horizontal Tab Group New Vertctal Tab Group

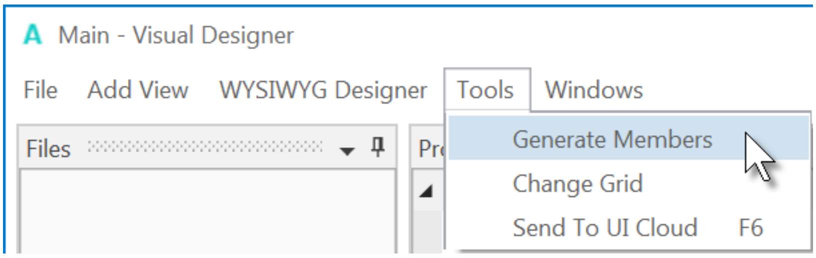
To remove Tab Group right click on Files and click on Move to Previous Tab Group .



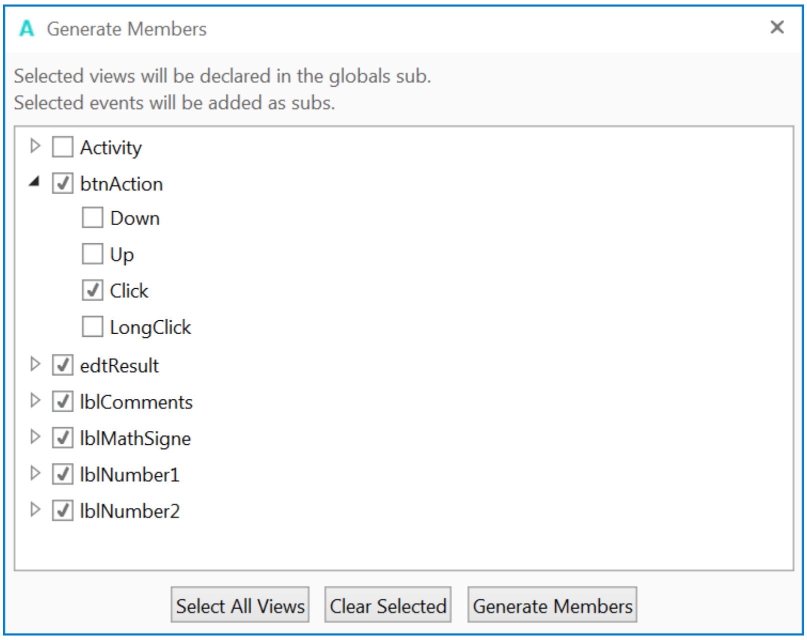
8.4 Tools

8.4.1 Generate Members

Generates declaration statements and subroutines frames. A similar function exists in the Abstract Designer context menu. The example is based on the MyFirstProgram project.



Click on Generate Members to open the generator.



Selected views will be declared in the globals sub.
Selected events will be added as subs.

- ▷ ☐ Activity
- ◢ ☑ btnAction
  - ☐ Down
  - ☐ Up
  - ☑ Click
  - ☐ LongClick
- ▷ ☑ edtResult
- ▷ ☑ lblComments
- ▷ ☑ lblMathSigne
- ▷ ☑ lblNumber1
- ▷ ☑ lblNumber2

Here we find all the views added to the current layout (MyFirstProgram).
We check all views and check the Click event for the btnAction Button.

Checking a view ▷ ☑ edtResult generates its reference in the Globals Sub routine in the code.
This is needed to make the view recognized by the system and allow the autocomplete function.

Variable declarations in Globals

```
Sub Globals
   Private btnAction As Button
   Private edtResult As EditText
   Private lblComments As Label
   Private lblMathSign As Label
   Private lblNumber1 As Label
   Private lblNumber2 As Label
```

Clicking on an event of a view ☑ Click generates the Sub frame for this event.

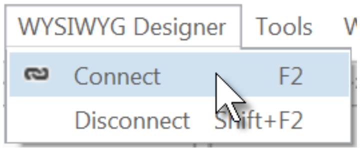
```
Sub btnAction_Click

End Sub
```

Click on Generate Members to generate the references and Sub frames.

Click on Select All Views to select all vies in the list,

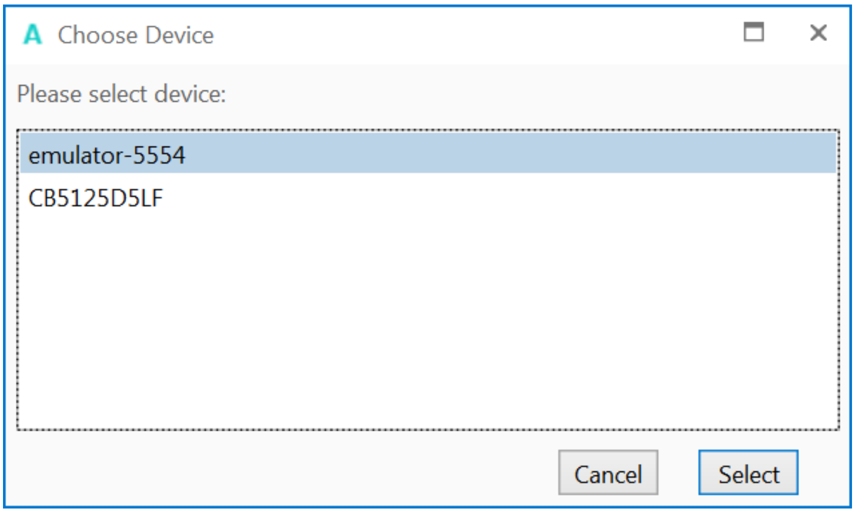
Click on Clear Selected to clear the current selections.

8.4.2 Connect device or emulator



To connect a device or an emulator click Connect in the WYSIWYG Designer menu or pres F2.

If different devices or Emulators are connected, you will be asked which device or Emulator you want to connect to.


Choose Device

Please select device:

emulator-5554
CB5125D5LF

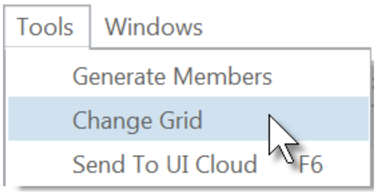
Select an emulator or a device in the list.

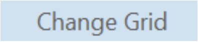
Click on Select to confirm.

To disconnect it click on Disconnect Shift+F2 in the WYSIWYG Designer menu or press SHIST + F2.

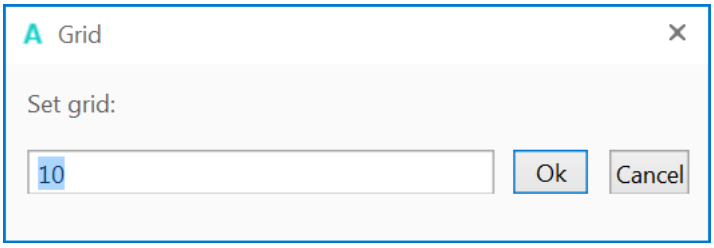
8.4.3 Change grid

The grid is an invisible grid with a given size. The default grid size is 10 pixels. That means that all positions and dimensions of a view will be set to values in steps corresponding to the grid size. Moving a view will be done in steps equal to the grid size.



In the Tools menu click on .

You can change the grid size to the value you want.

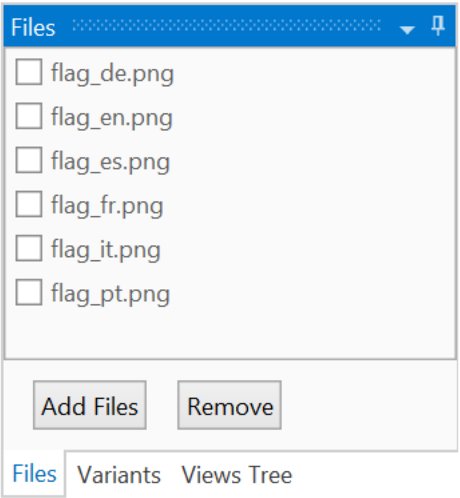


Set grid: 10

The value is saved in the layout file, you will get the same value when you reload this layout.

The default value when you start a new project is 10.

8.5 Image files



□ flag_de.png
□ flag_en.png
□ flag_es.png
□ flag_fr.png
□ flag_it.png
□ flag_pt.png

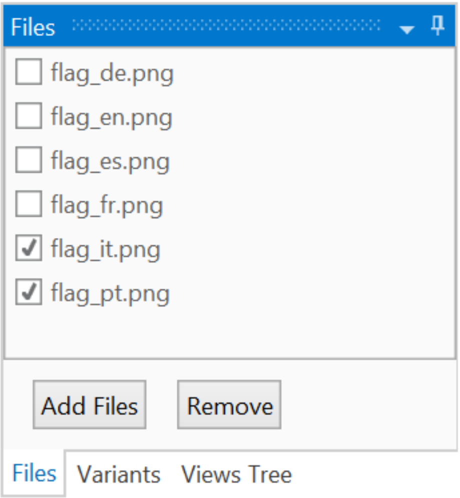
You can add image files to the layout.

Click on Add Files to select the files(s) to add.

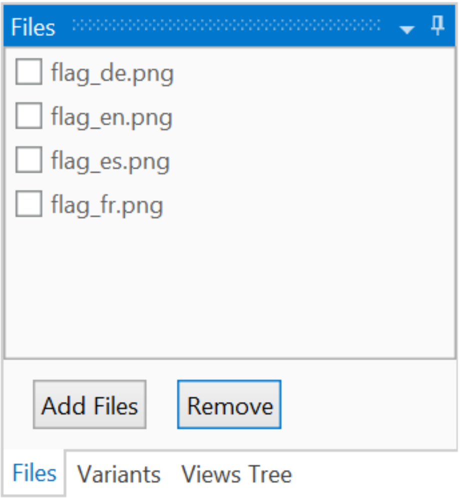
These files will be listed in the Image Files list.

These files are saved to the Files folder of the project and can be accessed in the code in the Files.DirAssets folder.

To remove files, check the files to remove and click on Remove.

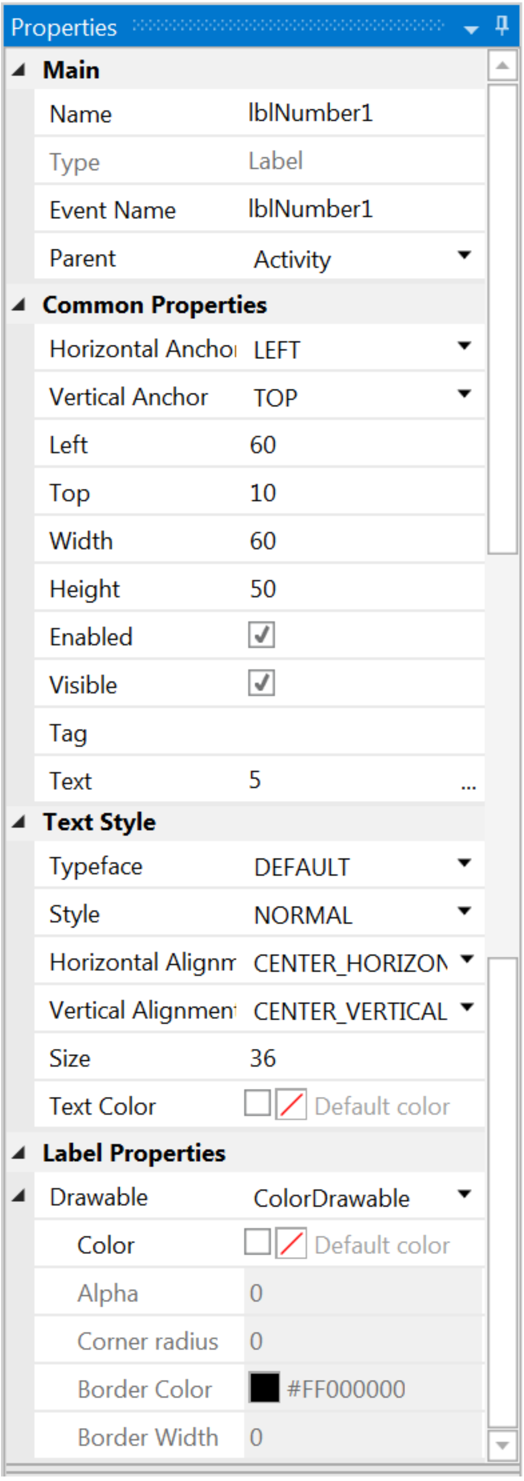


□ flag_de.png
□ flag_en.png
□ flag_es.png
□ flag_fr.png
☑ flag_it.png
☑ flag_pt.png



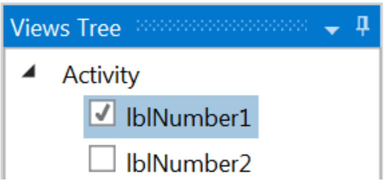
□ flag_de.png
□ flag_en.png
□ flag_es.png
□ flag_fr.png

8.6 Properties list



Properties

Main	
Name	lblNumber1
Type	Label
Event Name	lblNumber1
Parent	Activity
Common Properties	
Horizontal Anchoı	LEFT
Vertical Anchor	TOP
Left	60
Top	10
Width	60
Height	50
Enabled	☑
Visible	☑
Tag	
Text	5 ...
Text Style	
Typeface	DEFAULT
Style	NORMAL
Horizontal Alignm	CENTER_HORIZON
Vertical Alignmen	CENTER_VERTICAL
Size	36
Text Color	□ Default color
Label Properties	
Drawable	ColorDrawable
Color	□ Default color
Alpha	0
Corner radius	0
Border Color	■ #FF000000
Border Width	0



Views Tree

- Activity
  - ☑ lblNumber1
  - □ lblNumber2

Select for example lblNumber1 in the list.

All the properties of lblNumber1 are displayed.
These are organized in groups.

All properties can be modified directly in the list.

All properties in the Main group and some of the properties in the other groups are common to all view types.

Explanation of some general properties for all types of Views:

8.6.1 Main properties

Name Name of the view. It is good practice to give meaningful names. Common usage is to give a 3 character prefix and add the purpose of the view. In the example, the view is of type Label and its purpose is to enter a result. So we give it the name "lblResult", "lbl" for Label and "Result" for the purpose. This does not take much time during the design of the layout but saves a lot time during coding and maintenance of the program.

Type Type of the view, not editable. It is not possible to change the type of a view. If you need to, you must remove the view and add a new one.

Event Name Generic name for the subroutines that manages the view's events. By default, the Event Name is the same as the view's name like in the example. The Events of several Views can be redirected to a same subroutine. In that case you must enter the name of that routine. Look at the SecondProgram example for the Click event management for the buttons of the keyboard, the btnEvent_Click routine.

Parent Name of the parent view. Activity, in the example. The parent view can be changed in selecting the new one in the list.

8.6.2 Common properties

HorizontalAnchor Horizontal Anchor function. Possible values LEFT, RIGHT or BOTH

VerticalAnchor Vertical Anchor function. Possible values TOP, BOTTOM or BOTH

Left X coordinate of the left edge of the View from the left edge of its parent View, in pixels (the pixels are in reality dips, density independent pixels).

Top Y coordinate of the upper edge of the View from the upper edge of its parent View, in pixels (the pixels are in reality dips, density independent pixels).

Width Width of the View in pixels (the pixels are in reality dips, density independent pixels).

Height Height of the View in pixels (the pixels are in reality dips, density independent pixels).

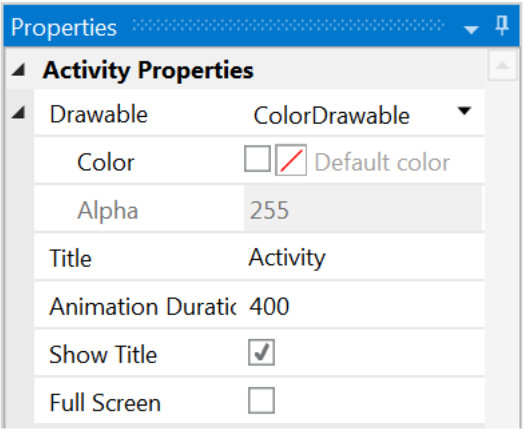
Enabled Enables or disables the use of the View Ex: Enabled = True

Visible Determines if the View is visible to the user or not.

Tag This is a place holder which can used to store additional data. Tag can simply be text but can also be any other kind of object.
Tag is used in the SecondProgram example for the numeric buttons click events management in the btnEvent_Click routine.

Text The text which will be displayed in the View, this property is only available for views having a Text property.

8.6.3 Activity properties



Properties

Activity Properties

Drawable	ColorDrawable
Color	□ Default color
Alpha	255
Title	Activity
Animation Duratic	400
Show Title	☑
Full Screen	□

Drawable Sets the Activity background Drawable, the default property is ColorDrawable.

Title Sets the activity title text.

Animation Duration Sets the animation duration in milliseconds.
When you launch the program the Activity is not shown directly but grows with the given duration. If you set this value to '0' the Activity will be shown instantly.

Show Title Changes the Abstract Designer height.
This setting does not change the Activity property,
only the Abstract Designer height.

Full Screen Changes the Abstract Designer height.
This setting does not change the Activity property,
only the Abstract Designer height.

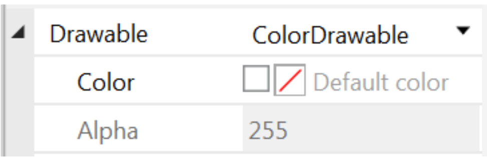
To not show the titles or set full screen, you need to set these two properties in the Module code in the `Activity Attributes` or `Module Attributes` Regions:

```
#Region  Activity Attributes
   #FullScreen: False
   #IncludeTitle: True
#End Region
```

Checking or unchecking the last two properties only changes the visible screen size in the Abstract Designer.

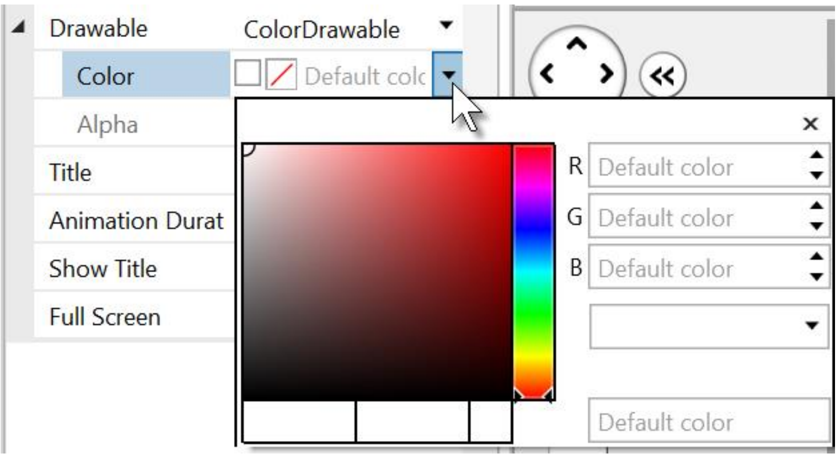
8.6.4 Color properties

For some properties, like ColorDrawable color, TextColor, you can select a color.



Drawable	ColorDrawable
Color	□ Default color
Alpha	255

By default the Default color is selected.
And Alpha, the transparency factor, is set to 255 which means fully opaque.

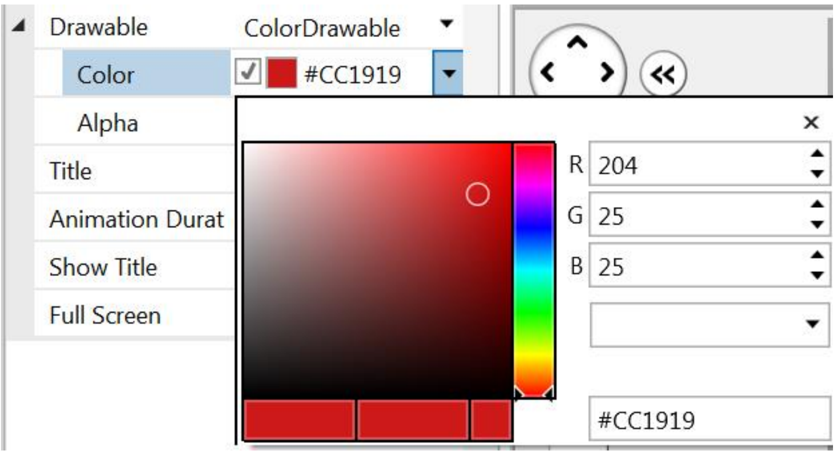


Drawable	ColorDrawable
Color	□ Default colc
Alpha	
Title	
Animation Durat	
Show Title	
Full Screen	

R Default color
G Default color
B Default color
Default color

Click on ▾ to select another color.

The color picker is displayed.



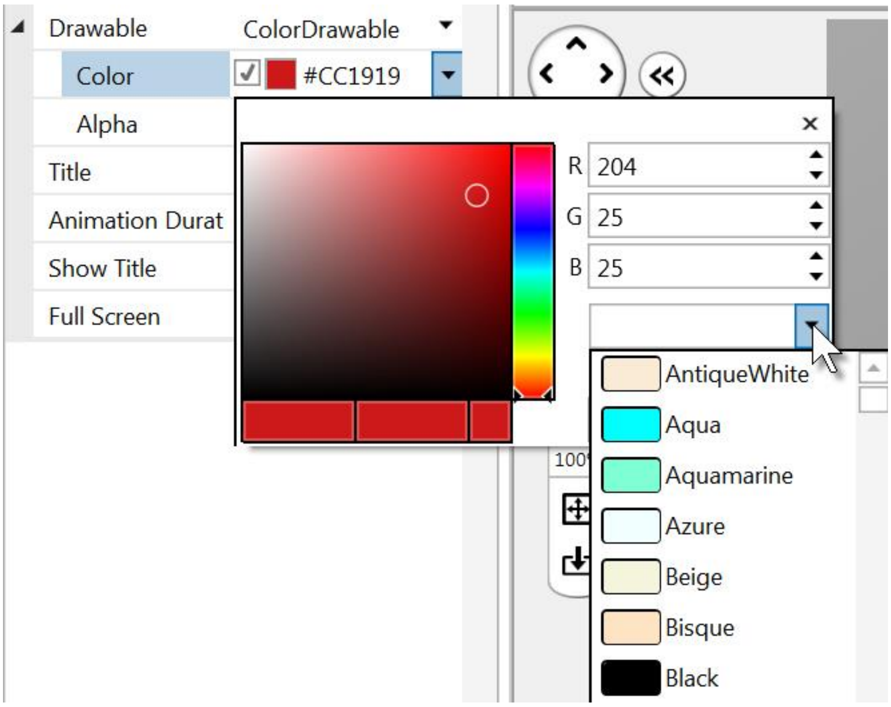
Drawable	ColorDrawable
Color	☑ 🟥 #CC1919
Alpha	
Title	
Animation Durat	
Show Title	
Full Screen	

R 204
G 25
B 25
#CC1919

You can either:

- Move the vertical slider to select a color.
- Move the small circle to select the 'darkness'.
- Enter the RGB values.
- Select a predefined color.
- Enter the hex value.

Select a predefined color.



Drawable ColorDrawable

Color ☑ #CC1919

Alpha

Title

Animation Durat

Show Title

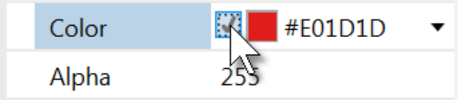
Full Screen

R 204

G 25

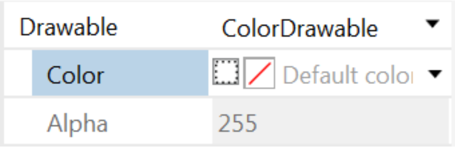
B 25

To reset the default color click on ☑.



Color ☑ #E01D1D

Alpha 255

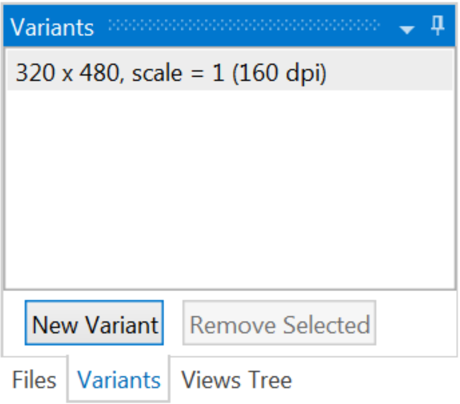


Drawable ColorDrawable

Color □ Default colo

Alpha 255

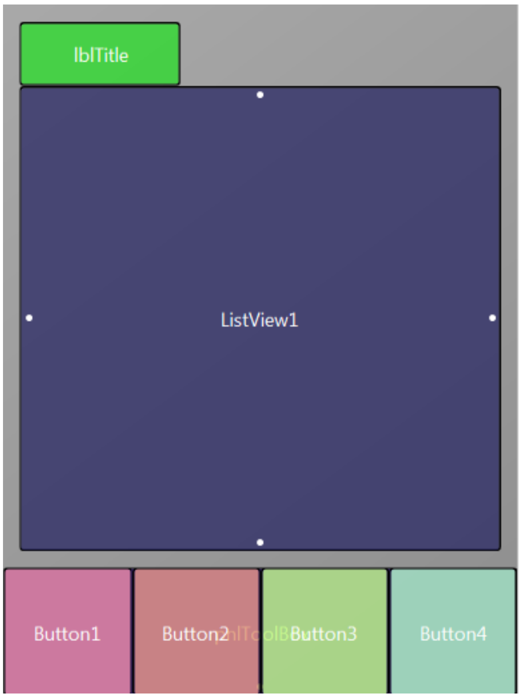
8.7 Layout variants

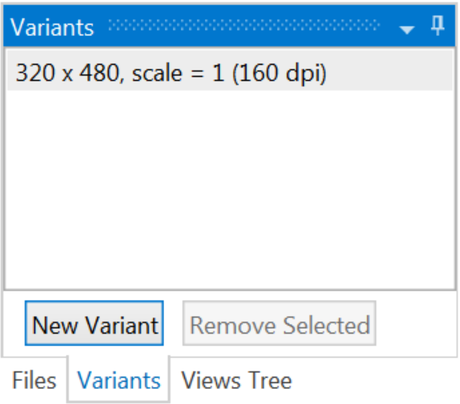


Different layout variants can be managed in a same layout file.

Let us make an example based on the TestLayoutsAnchors project
(which can be found under the Guide\SourceCode\TestLayoutsAnchors directory):
- Create a new folder and name it TestLayoutVariants.
- Copy the whole contents of the TestLayoutsAnchors folder.
- Rename the TestLayoutAnchors.b4a file to TestLayoutVariants.b4a.
- Rename the TestLayoutAnchors.b4a.meta file to TestLayoutVariants.b4a.meta.
- Run the IDE.
- Run the Visual Designer.

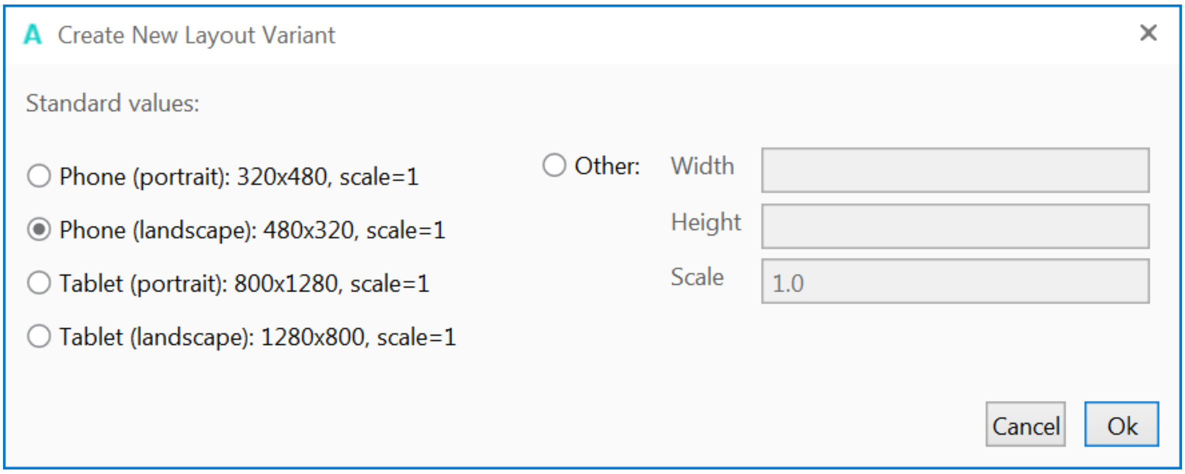
The layout in the Abstract Designer should look like this.





320 x 480, scale = 1 (160 dpi)

In the Designer, click on New Variant.



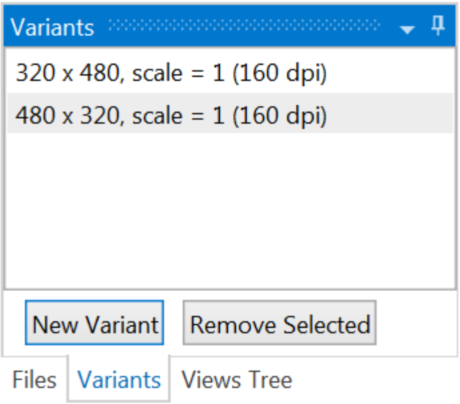
Standard values:

- ○ Phone (portrait): 320x480, scale=1
- ● Phone (landscape): 480x320, scale=1
- ○ Tablet (portrait): 800x1280, scale=1
- ○ Tablet (landscape): 1280x800, scale=1

○ Other: Width Height Scale 1.0

Select: Phone (landscape):480 x 320, scale = 1

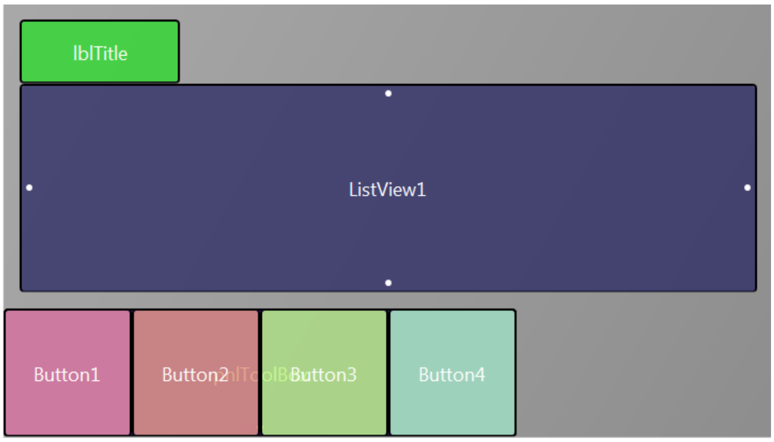
Click on Ok.



320 x 480, scale = 1 (160 dpi)
480 x 320, scale = 1 (160 dpi)

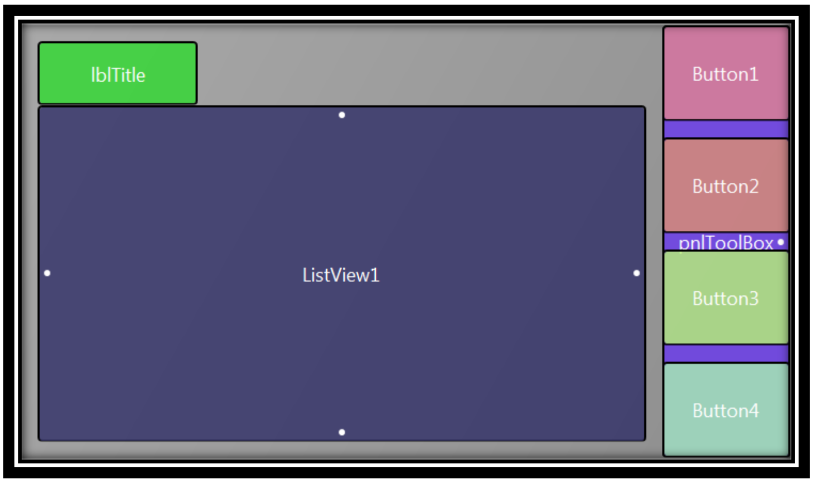
The new variant is added.

In the Abstract Designer you'll see something like this.



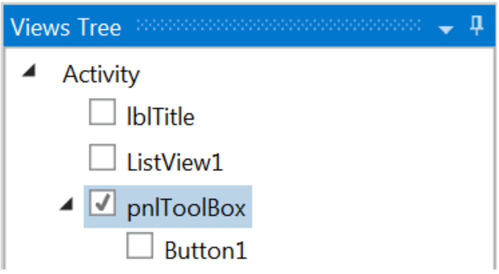
We see that the anchors work well.
pnlToolBox is still at the bottom of the screen and ListView1 is stretched the fill almost the whole screen width.

But for the landscape variant we want the ToolBox at the right side of the screen.



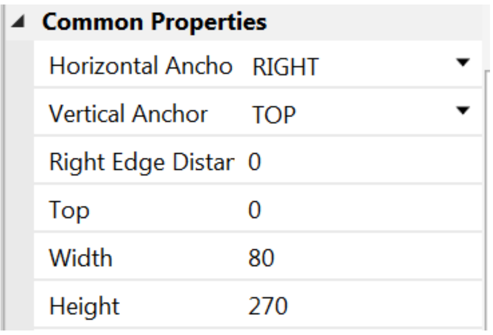
We:
- Reduce the width of ListView1 to get space for the ToolBox.

- Move pnlToolBox to the right side of the screen, change the Button heights and rearrange them vertically like in the picture.



- Activity
  - □ lblTitle
  - □ ListView1
  - ☑ pnlToolBox
    - □ Button1

- Select pnlToolBox



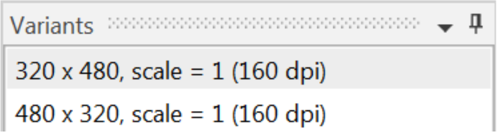
Common Properties

Horizontal Ancho	RIGHT
Vertical Anchor	TOP
Right Edge Distar	0
Top	0
Width	80
Height	270

- Set the pnlToolBox Horizontal Anchor to RIGHT.
  Set the Right Edge Distance to 0
  Set the pnlToolBox vertical anchor to TOP.
  Adjust the button heights and their Top properties accordingly.

To always show pnlToolBox in the middle of the screen we add following code in the Script – Variant window.

For portrait:



Variants

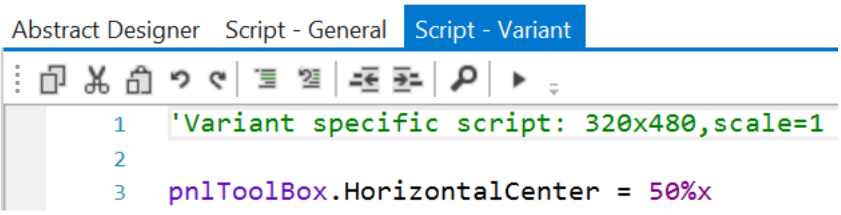
320 x 480, scale = 1 (160 dpi)

480 x 320, scale = 1 (160 dpi)

Select the portrait variant.

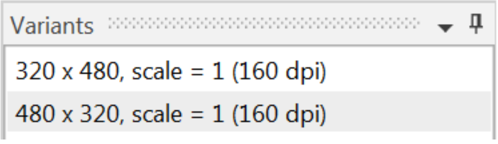
And add this code

`pnlToolBox.HorizontalCenter = 50%x.`



```
'Variant specific script: 320x480,scale=1

pnlToolBox.HorizontalCenter = 50%x
```

For landscape :



Variants

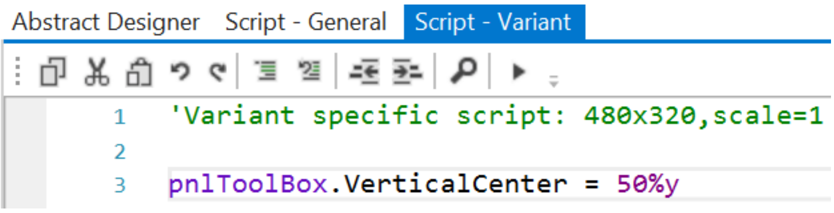
320 x 480, scale = 1 (160 dpi)

480 x 320, scale = 1 (160 dpi)

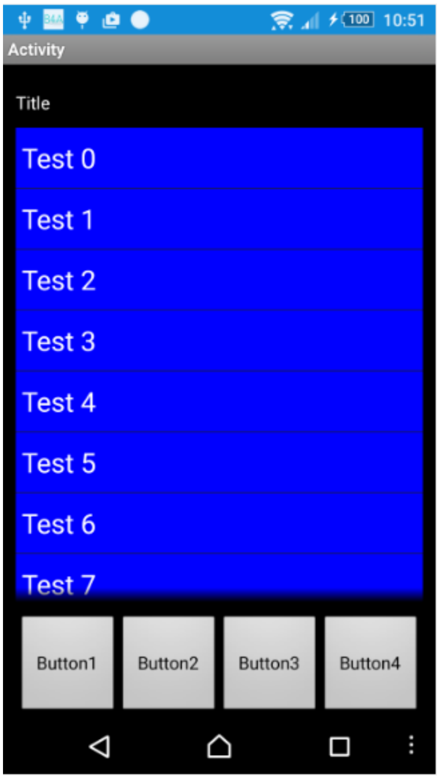
Select the landscape variant.

And add this code

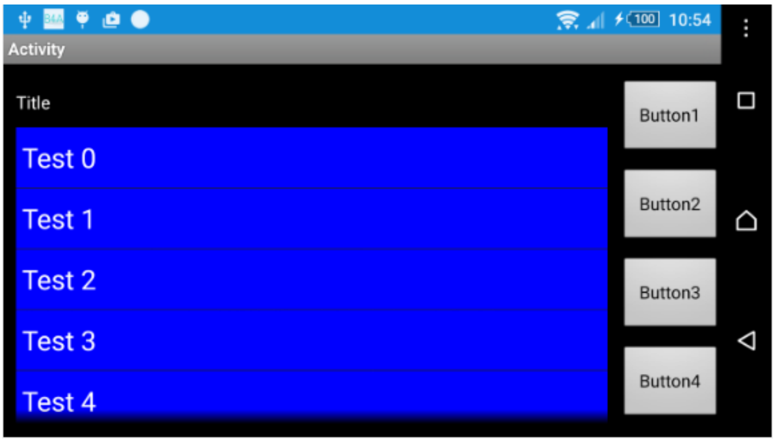
`pnlToolBox.VerticalCenter = 50%y.`



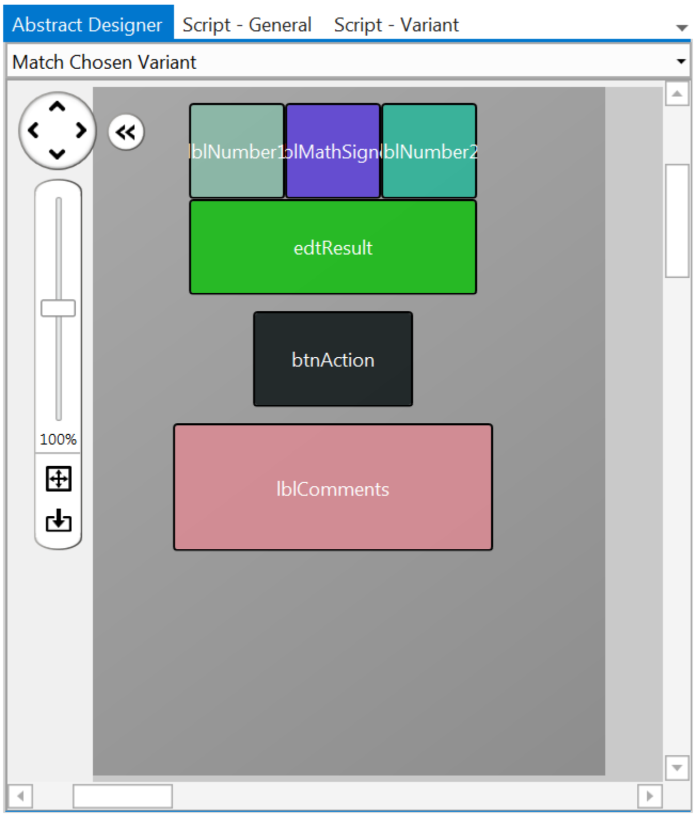
```
'Variant specific script: 480x320,scale=1

pnlToolBox.VerticalCenter = 50%y
```



And the result on a device.



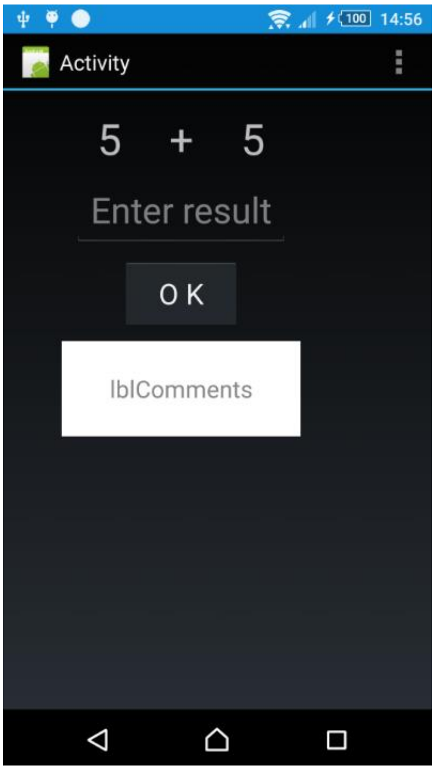
8.8 The Abstract Designer

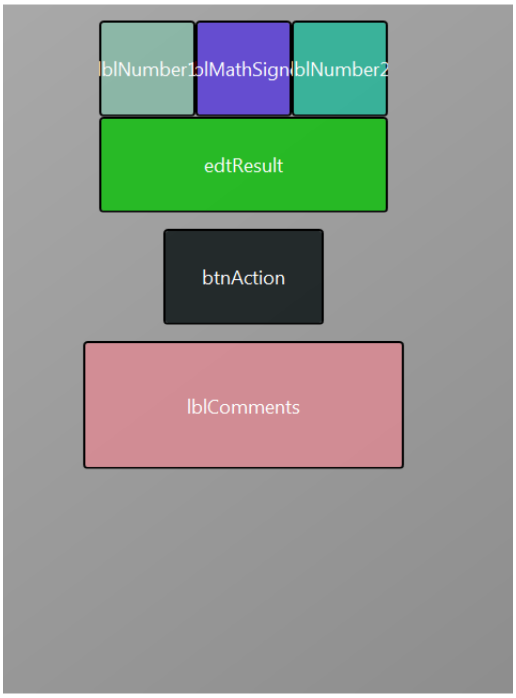


The Abstract Designer is a tool that shows the layout.
Its main purpose is to create different layout variants.
It is much faster than the Emulator.

The different views are not shown with their exact shape but only as coloured rectangles.
Clicking on a view shows its properties in the Designer.

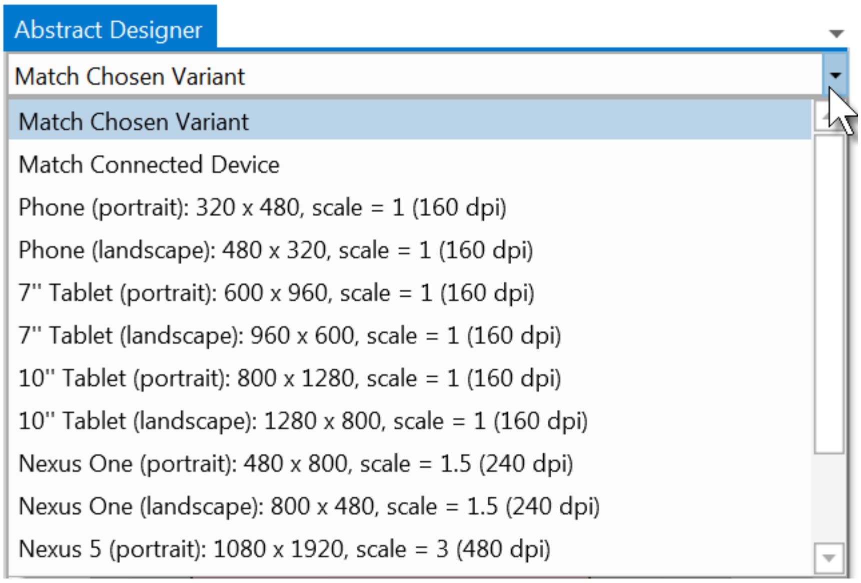
Device Abstract Designer





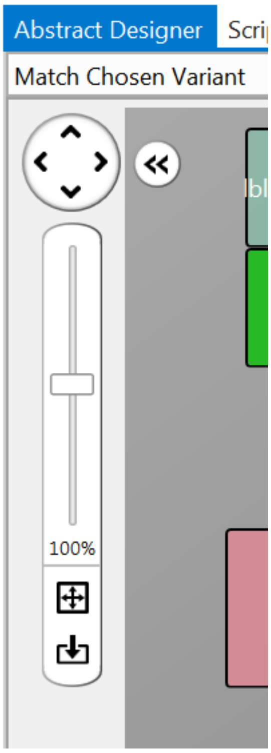
8.8.1 Selection of a screen size

On top you can select different screen sizes:


Abstract Designer
Match Chosen Variant
Match Chosen Variant
Match Connected Device
Phone (portrait): 320 x 480, scale = 1 (160 dpi)
Phone (landscape): 480 x 320, scale = 1 (160 dpi)
7'' Tablet (portrait): 600 x 960, scale = 1 (160 dpi)
7'' Tablet (landscape): 960 x 600, scale = 1 (160 dpi)
10'' Tablet (portrait): 800 x 1280, scale = 1 (160 dpi)
10'' Tablet (landscape): 1280 x 800, scale = 1 (160 dpi)
Nexus One (portrait): 480 x 800, scale = 1.5 (240 dpi)
Nexus One (landscape): 800 x 480, scale = 1.5 (240 dpi)
Nexus 5 (portrait): 1080 x 1920, scale = 3 (480 dpi)

- Match chosen Variant. Matches the variant selected in the Variant window.
- Match Connected Device. Matches the size of the connected device or emulator.
- Different ‘standard’ sizes. This allows see how a layout looks on s different screen.

8.8.2 Zoom



With ✥ you can move the virtual screen in the four directions.

With « you can hide the zoom cursor and show it again with ».

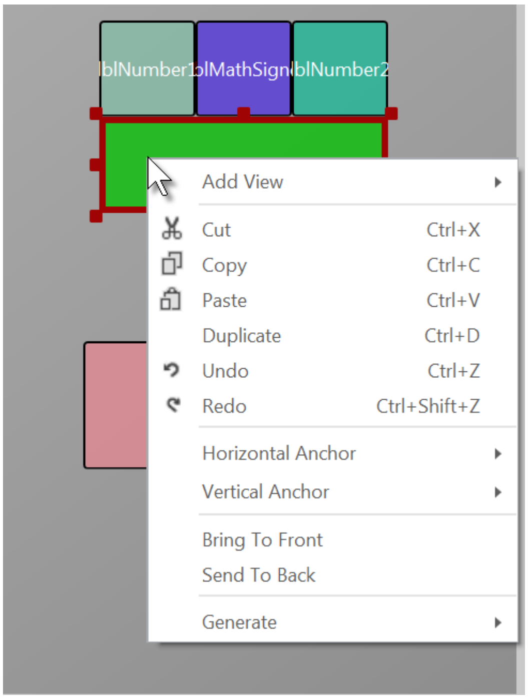
With the cursor you can set the zoom level you want.

With ⊞ you can zoom to fit the selected screen size.
With ⤓ you can reset the zoom back to 100%.

With the bottom and side cursors you can move the layout vertically or horizontally.

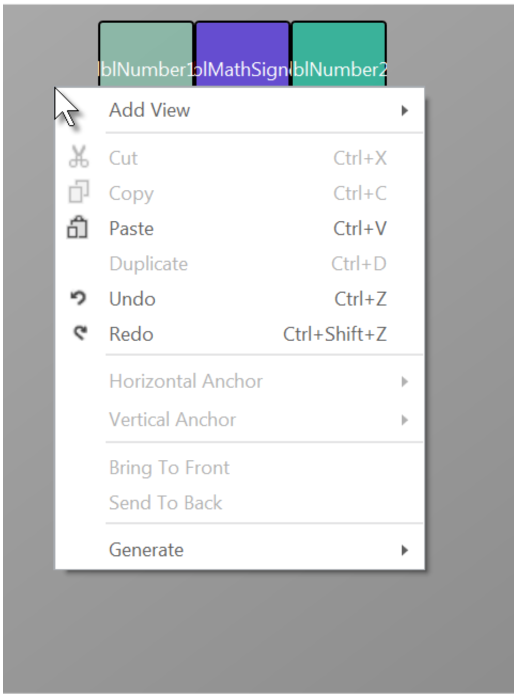
8.8.3 Context menus

Right clicking on a view shows a context menu.



Add View
Cut
Copy
Paste
Duplicate
Undo
Redo
Horizontal Anchor
Vertical Anchor
Bring To Front
Send To Back
Generate

Right clicking somewhere on the Activity area shows the context menu with some functions disabled which are not relevant for an Activity.

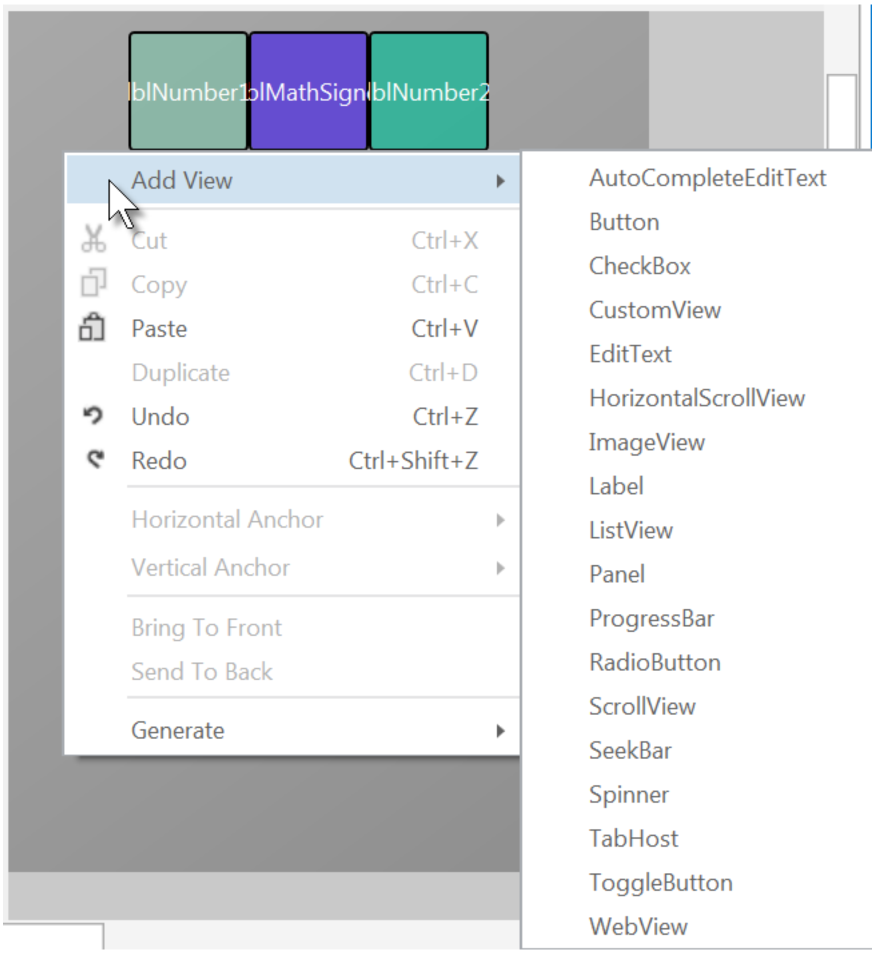


Only Add View, Paste, Undo, Redo and Generate are available.

8.8.3.1 Add View

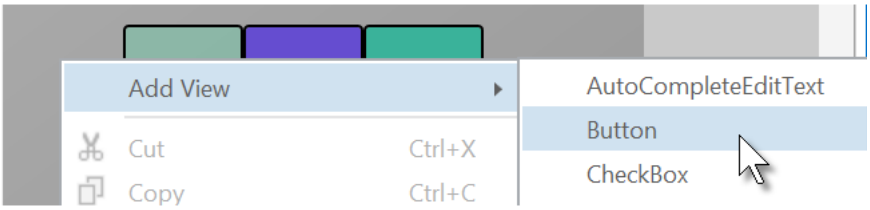
Right click somewhere and move the cursor onto Add View.

This function is the same as the Add View function in the Visual Designer menu.

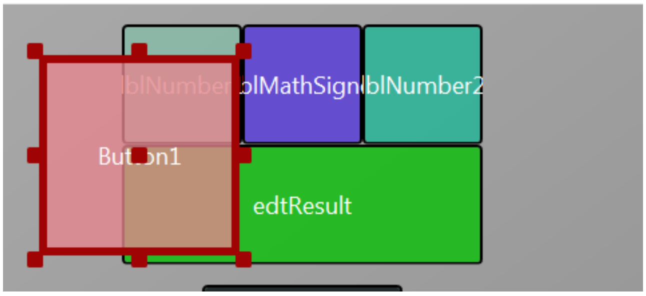


The list of all available views is displayed.

Click on the desired view to add it.

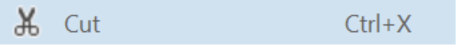


Example for a Button.



The Button is added to the layout.

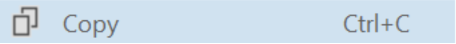
8.8.3.2 Cut



removes the selected view from the layout.
If you selected a Panel, it will be removed with all its child views!

If you cut it by accident click on ↶ or press Ctrl+Z to recover it.

8.8.3.3 Copy



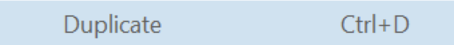
copies the selected view into the clipboard.
If you selected a Panel, it will be copied with all its child views!

8.8.3.4 Paste

copies the content of the clipboard.
If you selected a Panel, it will be pasted with all its child views!

Before you paste a view you must select where you want to paste it. This can be either onto the Activity or onto a Panel.

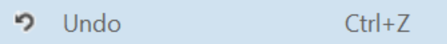
8.8.3.5 Duplicate

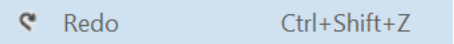


Duplicates the selected view, it is added over itself.
Duplicate is a shortcut of Copy and Paste.

If you selected a Panel, it will be duplicated with all its child views!

8.8.3.6 Undo / Redo

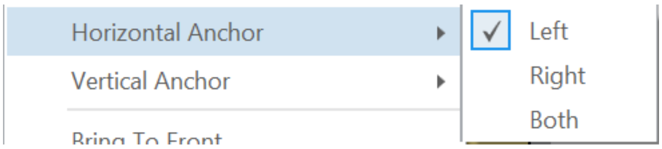




These two functions allow you to undo or redo the last operations.

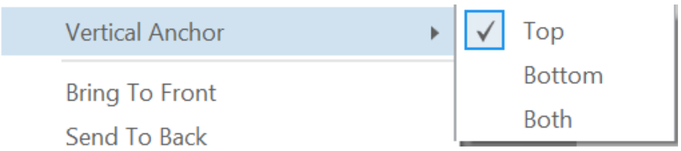
8.8.3.7 Horizontal Anchor

You can set the horizontal anchor in the context menu instead of changing it in the Properties window.
The current anchor is checked.

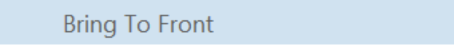


8.8.3.8 Vertical Anchor

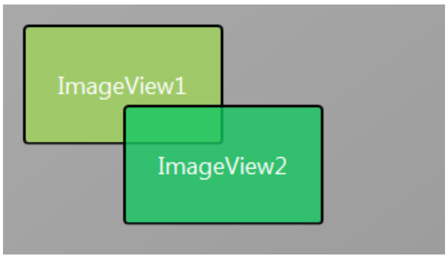
You can set the vertical anchor in the context menu instead of changing it in the Properties window. The current anchor is checked.



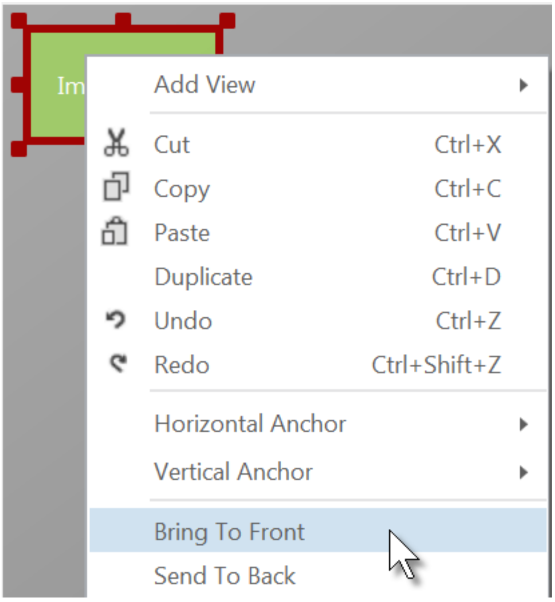
8.8.3.9 Bring To Front



Moves the selected View on top of the layout.

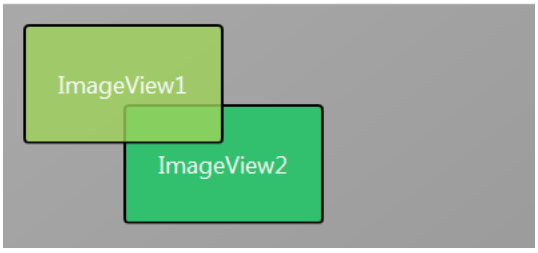


In the picture ImageView2 is over ImageView1.
You see it with the border color.



Right click on ImageView1 and click on to move ImageView1 to front of all other views.

And the result:



8.8.3.10 Send To Back

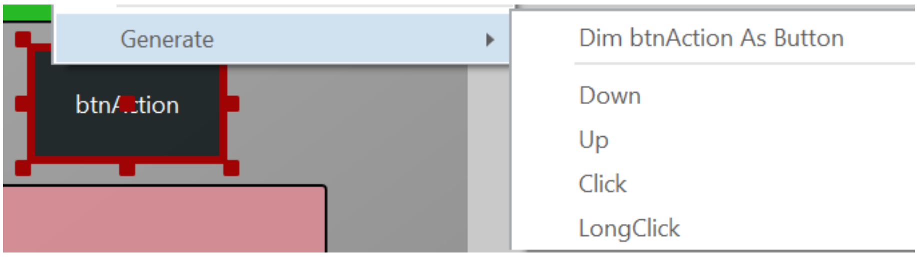
is the inverse function of Bring To Front function above.

8.8.3.11 Generate

Generates the declaration statement or an event routine for the selected View. It is a shortcut of the Generate Members function in the VisualDesigner Tools menu but only for the selected view.

A popup menu allows you to select what code you want to generate, the possibilities depend on the type of the selected view.

Example with a Button:



Dim btnAction As Button

Generates the declaration statement in the Globals routine.

```
Private btnAction As Button
```

Down

Generates the Down event routine frame.

```
Sub btnAction_Down

End Sub
```

Up

Generates the Up event routine frame.

```
Sub btnAction_Up

End Sub
```

Click

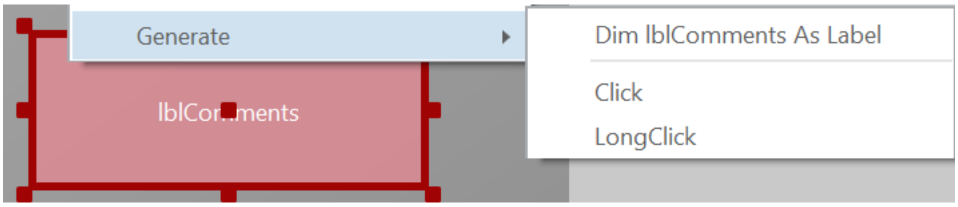
Generates the Click event routine frame.

```
Sub btnAction_Click

End Sub
```

LongClick

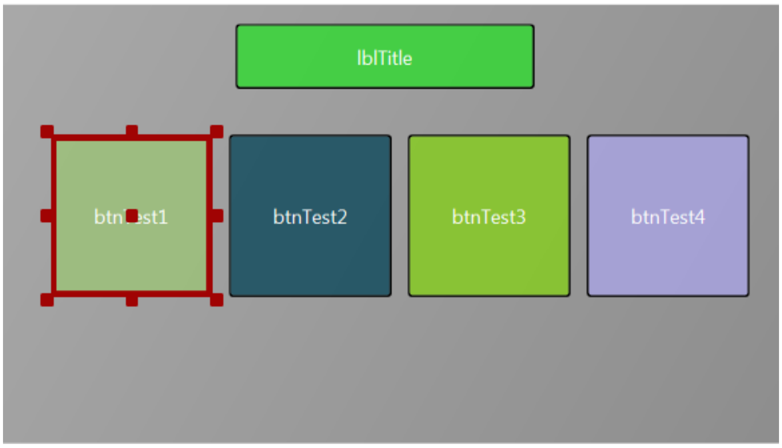
Generates the LongClick event routine frame.

```
Sub btnTest1_LongClick

End Sub
```

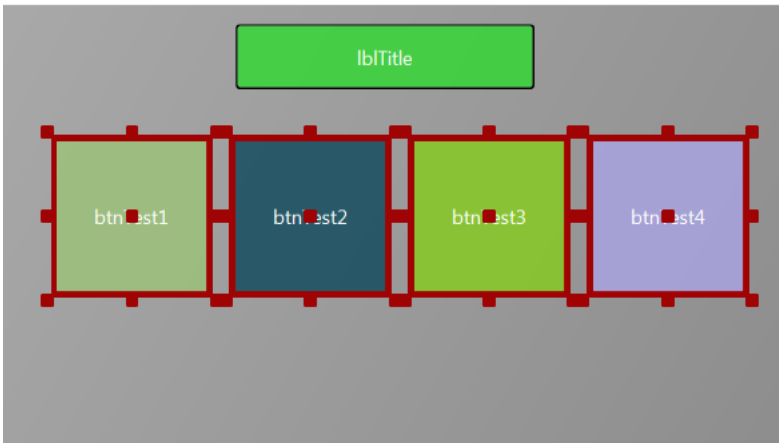


Example with a Label.

8.8.4 Select views



Select a single view:
Click on the view

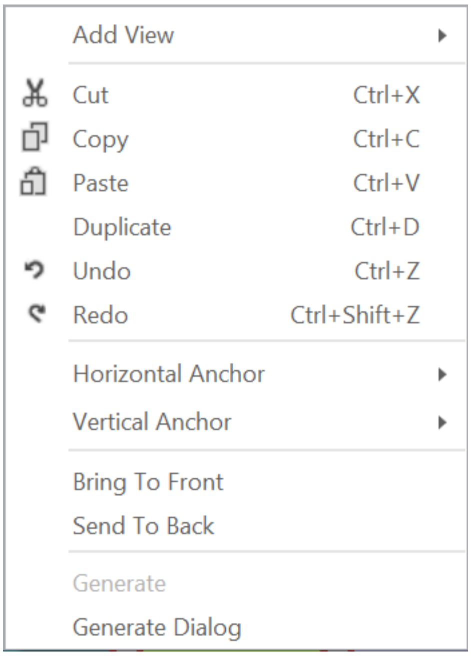


Select several views:
Click on the first view.
Press the Ctrl key,
Click the following views.

The selected views are highlighted.

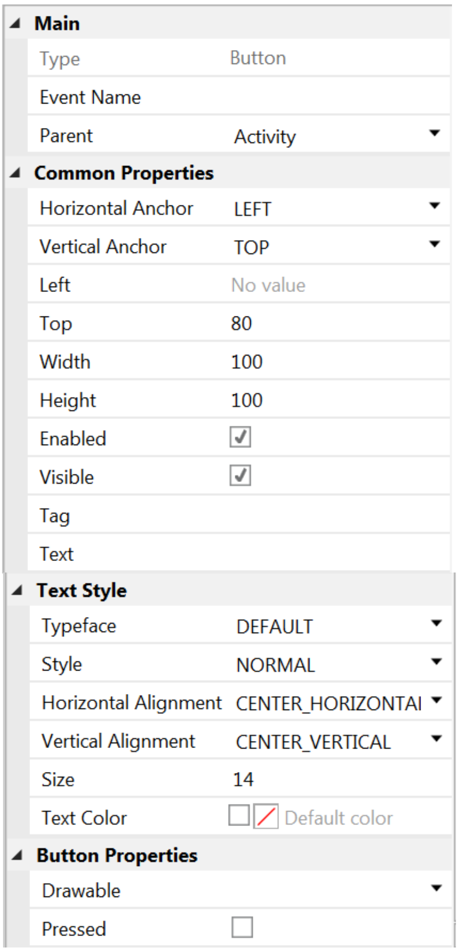
After the selection you can:

- Move the selected views with the arrow keys of the keyboard in the four directions.
- Right click on one of the selected view to show the contect menu.



The functions are the same as for a single view, but a new function, GenerateDialog, is available to Generate Members.
This is the same function as in the Visual Designer Tools menu.

- In the Properties window you can change all properties common to the selected views.

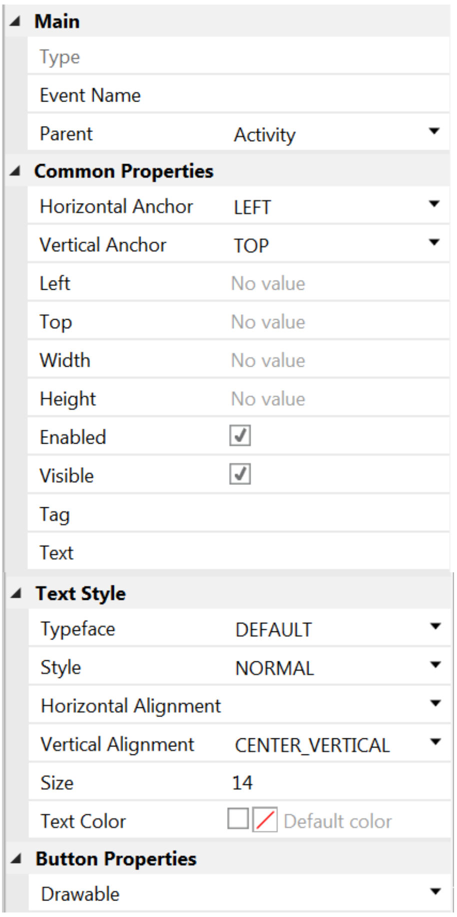


Main	
Type	Button
Event Name	
Parent	Activity
Common Properties	
Horizontal Anchor	LEFT
Vertical Anchor	TOP
Left	No value
Top	80
Width	100
Height	100
Enabled	☑
Visible	☑
Tag	
Text	
Text Style	
Typeface	DEFAULT
Style	NORMAL
Horizontal Alignment	CENTER_HORIZONTAl
Vertical Alignment	CENTER_VERTICAL
Size	14
Text Color	□ Default color
Button Properties	
Drawable	
Pressed	□

You can change the parent view.

You can change all these properties because they are the same for the four views selected in the example.

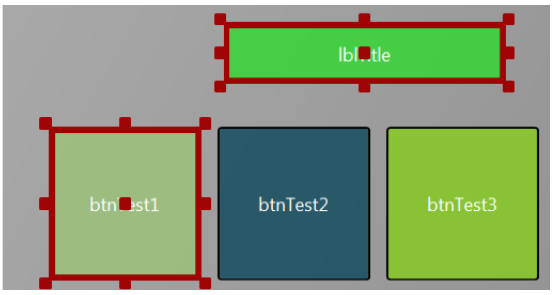
Changing, for example, the Height property will change it for all the selected views.



Main	
Type	
Event Name	
Parent	Activity
Common Properties	
Horizontal Anchor	LEFT
Vertical Anchor	TOP
Left	No value
Top	No value
Width	No value
Height	No value
Enabled	☑
Visible	☑
Tag	
Text	
Text Style	
Typeface	DEFAULT
Style	NORMAL
Horizontal Alignment	
Vertical Alignment	CENTER_VERTICAL
Size	14
Text Color	□ Default color
Button Properties	
Drawable	

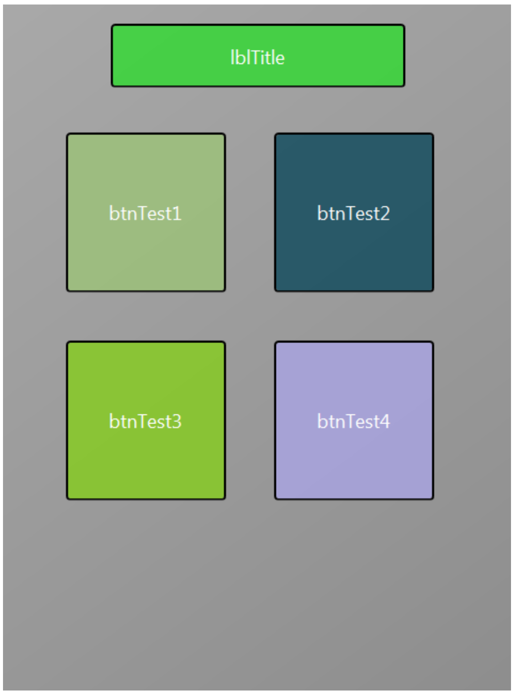
If you select views of different types, only the properties common to the selected views can be changed.

Example with a Label and a Button.



8.8.5 Example

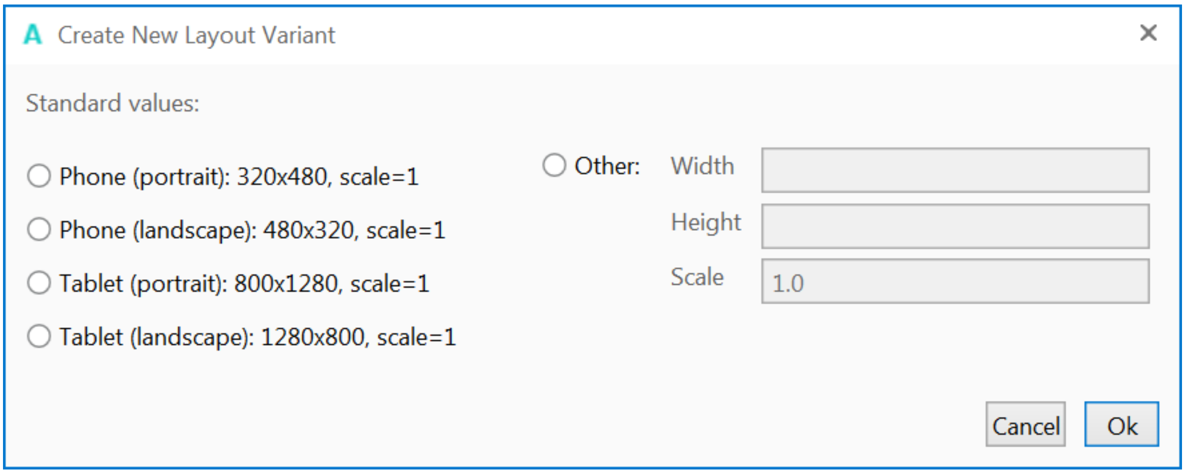
Let us take a simple example with a layout in portrait mode, like the image below.
This example project is in the SourceCode folder in the AbstractDesigner subfolder.



Now we would like to make a landscape variant.

In the Variant window click on New Variant.

A selection window is displayed.



Create New Layout Variant

Standard values:

○ Phone (portrait): 320x480, scale=1

○ Phone (landscape): 480x320, scale=1

○ Tablet (portrait): 800x1280, scale=1

○ Tablet (landscape): 1280x800, scale=1

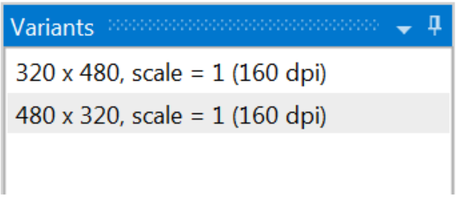
○ Other: Width

Height

Scale 1.0

Select ◉ Phone (landscape): 480x320, scale=1 .

In the Variant window the new variant is displayed.

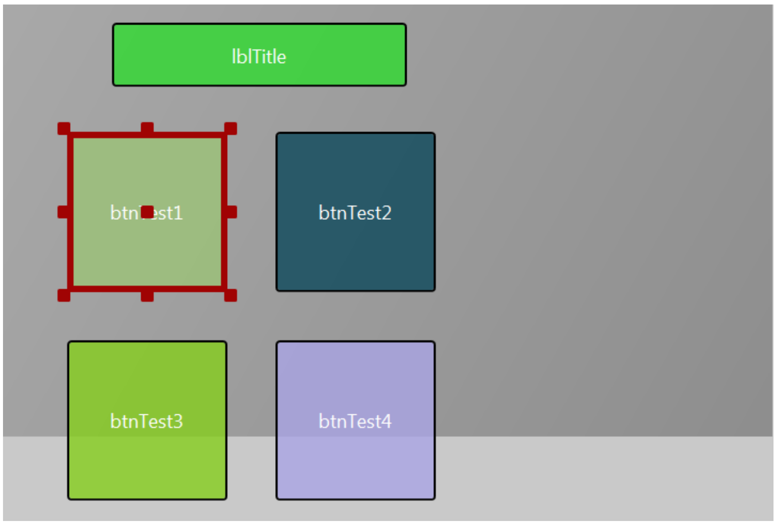


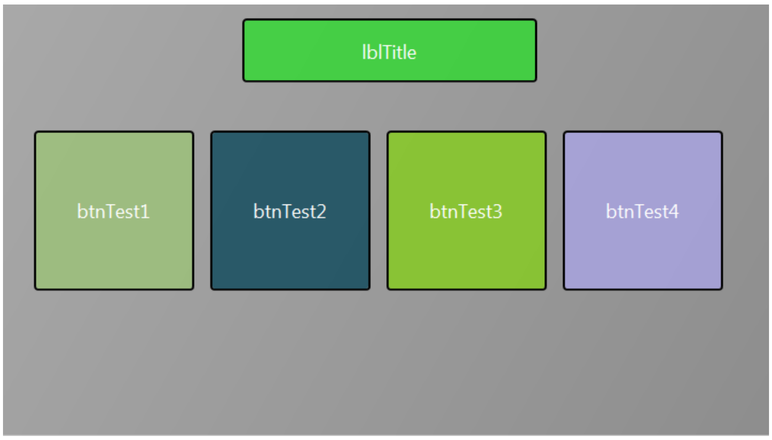
Variants

320 x 480, scale = 1 (160 dpi)

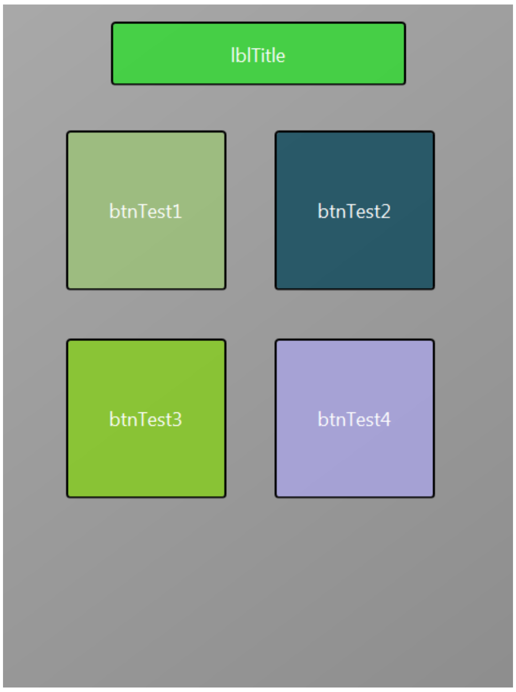
480 x 320, scale = 1 (160 dpi)

The Abstract Designer looks now like this:

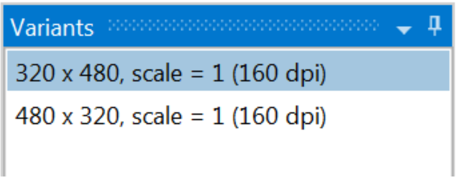




Now we rearrange the views to fit the new orientation.



If you select in the Variant window the previous variant



320 x 480, scale = 1 (160 dpi)
480 x 320, scale = 1 (160 dpi)

you will see the layout on the left.

8.9 Adding views by code

It is also possible to add views by code instead of using the Designer with a device, the Abstract Designer or an Emulator.
Advantage: you have full control of the view.
Disadvantage: you have to define almost everything.

The source code is in the source code directory: AddViewsByCode

For the positions and dimensions of the views on the screen two special options are available:

- dip **d**ensity **i**ndependent **p**ixels.
  100dip = DipToCurrent(100) `DipToCurrent` is a Keyword `dip` is the Shortcut
  100dip = 100 / 160 * device density
  The default density is 160 dpi **d**ots **p**er **i**nch (pixels per inch)
  Densities in Android:
    - 120 scale 0.75
    - 160 scale 1 default
    - 240 scale 1.5
    - 320 scale 2

- %x and %y represent distances proportional to the active screen width and height.
  20%x = 0.2 * Activity.Width
  90%y = 0.9 * Activity.Height
  20%x = PerXToCurrent(20) `PerXToCurrent` is a Keyword `%x` is the Shortcut
  90%y = PerYToCurrent(90)

Example:
Let us put a Label on top of the screen and a Panel below it with a Label and a Button on it:

```
Sub Globals
  Private lblTitle, lblPanelTitle As Label
  Private pnlTest As Panel
  Private btnTest As Button
End Sub
```

```
Sub Activity_Create(FirstTime As Boolean)
  lblTitle.Initialize("")
  lblTitle.Color = Colors.Red
  lblTitle.TextSize = 20
  lblTitle.TextColor = Colors.Blue
  lblTitle.Gravity = Gravity.CENTER_HORIZONTAL + Gravity.CENTER_VERTICAL
  lblTitle.Text = "Title"
  Activity.AddView(lblTitle, 20%x, 10dip, 60%x, 30dip)

  pnlTest.Initialize("")
  pnlTest.Color = Colors.Blue

  btnTest.Initialize("btnTest")
  btnTest.Text = "Test"

  lblPanelTitle.Initialize("")
  lblPanelTitle.Color = Colors.Red
  lblPanelTitle.TextSize = 16
  lblPanelTitle.TextColor = Colors.Blue
  lblPanelTitle.Gravity = Gravity.CENTER_HORIZONTAL + Gravity.CENTER_VERTICAL
  lblPanelTitle.Text = "Panel test"

  Activity.AddView(pnlTest, 0, lblTitle.Top+lblTitle.Height+10dip, 100%x, 50%y)
  pnlTest.AddView(lblPanelTitle, 20dip, 10dip, 100dip, 30dip)
  pnlTest.AddView(btnTest, 50dip, 50dip, 100dip, 60dip)
End Sub
```

Declaring the views.

```
Private lblTitle, lblPanelTitle As Label
Private pnlTest As Panel
Private btnTest As Button
```

Initializing the title label:

`lblTitle.Initialize("")` Initializes the Label, no EventName required.

`lblTitle.Color = Colors.Red` Sets the Background color to red.

`lblTitle.TextSize = 20` Sets the text size to 20.

`lblTitle.TextColor = Colors.Blue`Sets the text color to blue.

`lblTitle.Gravity = Gravity.CENTER_HORIZONTAL + Gravity.CENTER_VERTICAL`
Sets the label gravity.

`lblTitle.Text = "Title"` Sets the label text to 'Title'.

`Activity.AddView(lblTitle, 20%x, 10dip, 60%x, 30dip)` Adds the view to the activity.

If the Label had been added in the Designer, all the above code wouldn't have been necessary because the properties would already have been defined in the Designer.
In the `Activity.AddView` line we see that:

- the Left property is set to 20%x, 20% of Activity.Width.
- the Top property is set to 10dip, 10 density independent pixels.
- the Width property is set to 60%x, 60% of Activity.Width
- the Height property is set to 30dip, 30 density independent pixels.

`pnlTest.Initialize("")` Initializes the Panel, no EventName required.

`pnlTest.Color = Colors.Blue` Sets the Background color to blue.

`btnTest.Initialize("btnTest")` Initializes the Button, EventName = btnTest.

`btnTest.Text = "Test"` Sets the button text to "Test"

`lblPanelTitle.Initialize("")`

```
lblPanelTitle.Color = Colors.Red
lblPanelTitle.TextSize = 16
lblPanelTitle.TextColor = Colors.Blue
lblPanelTitle.Gravity = Gravity.CENTER_HORIZONTAL + Gravity.CENTER_VERTICAL
lblPanelTitle.Text = "Panel test"
```

Similar to the title Label.

```
Activity.AddView(pnlTest,0,lblTitle.Top + lblTitle.Height + 10dip, 100%x, 50%y)
```

Adds the Panel pnlTest to the Activity.

- the Left property is set to 0
- the Top property is set to 10dips below the title Label
- the Width property is set to 100%x, the total Activity.Width
- the Height property is set to 50%y, half the Activity.Height

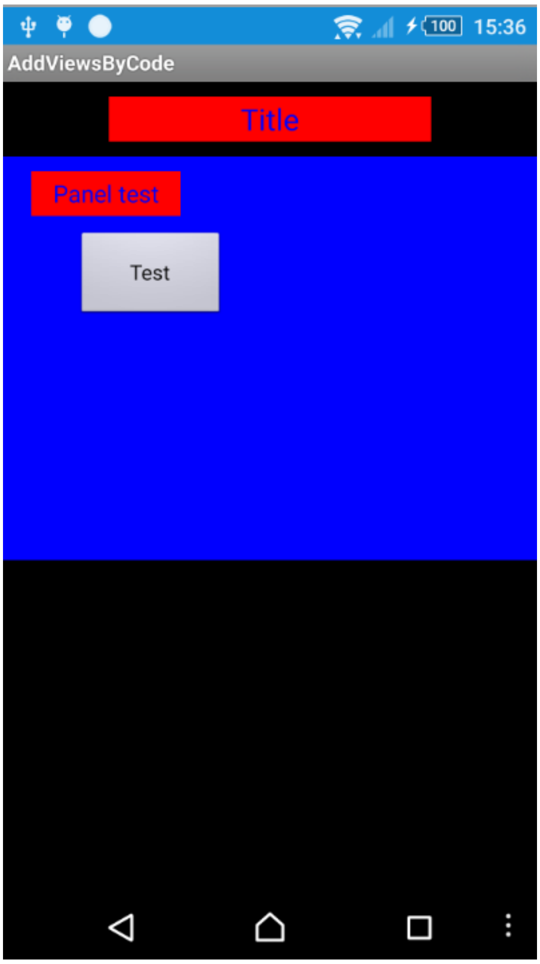
```
pnlTest.AddView(lblPanelTitle, 20dip, 10dip, 100dip, 30dip)
```

Adds the Label lblPanelTitle to the Panel pnlTest at the given position and with the given dimensions in dips.

```
pnlTest.AddView(btnTest, 50dip, 50dip, 100dip, 60dip)
```

Adds the Button btnTest to the Panel pnlTest at the given position and with the given dimensions in dips.

And the result on the device:



8.10 Designer Scripts

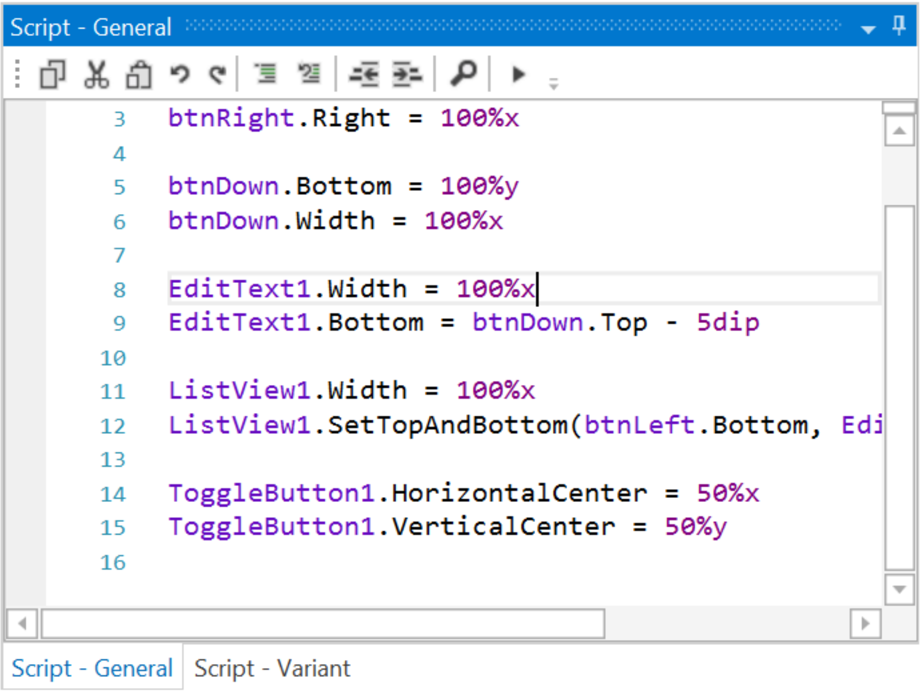
One of the most common issues that Android developers face is the need to adapt the user interface to devices with different screen sizes.
As described in the visual designer tutorial, you can create multiple layout variants to match different screens.
However it is not feasible nor recommended to create many layout variants.

The Designer Scripts will help you fine tune your layout and easily adjust it to different screens and resolutions.

The idea is to combine the usefulness of the visual designer with the flexibility and power of programming code.

You can write a simple script to adjust the layout based on the dimensions of the current device and immediately see the results. No need to compile and install the full program each time.

You can also immediately see the results in the Abstract Designer. This allows you to test your layout on many different screen sizes.



```
btnRight.Right = 100%x

btnDown.Bottom = 100%y
btnDown.Width = 100%x

EditText1.Width = 100%x
EditText1.Bottom = btnDown.Top - 5dip

ListView1.Width = 100%x
ListView1.SetTopAndBottom(btnLeft.Bottom, Edi

ToggleButton1.HorizontalCenter = 50%x
ToggleButton1.VerticalCenter = 50%y

```

Every layout file can include script code. The script is written inside the Visual Designer in the Script window:

There are two types of scripts:

- Script – General, the general script that will be applied to all variants.
- Script – Variant, specific code can be written for each variant.

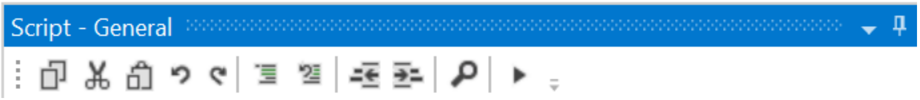
Once you press on the Run Script button (or F5), the script is executed and the connected device / emulator and abstract designer will show the updated layout.

The same thing happens when you run your compiled program. The (now compiled) script is executed after the layout is loaded.

The general script is first executed followed by the variant specific script.

The script language is very simple and is optimized for managing the layout.

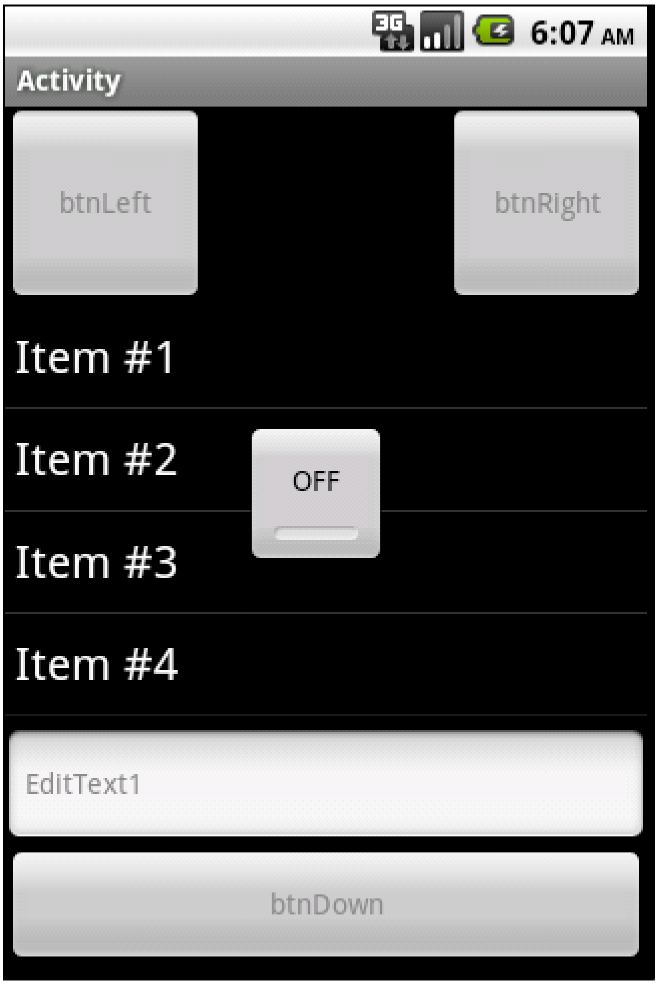
8.10.1 The menu



⧉	Ctrl + C	Copy
✂	Ctrl + X	Cut
📋	Ctrl + V	Paste
↶	Ctrl + Z	Undo
↷	Ctrl + Shift + Z	Redo
≡	Ctrl + Q	Block Comment
≡	Ctrl + W	Block Uncomment
⇤		Outdent
⇥		Indent
🔍	F3	Find / Replace
▶	F5	Run

Example

In this example we will build the following layout:
The source code is in the DesignerScripts folder.



btnLeft and btnRight should be located in the top corners.
btnDown should be located at the bottom and fill the entire width.
ListView1 should fill the entire available area.
ToggleButton1 should be located exactly in the centre.

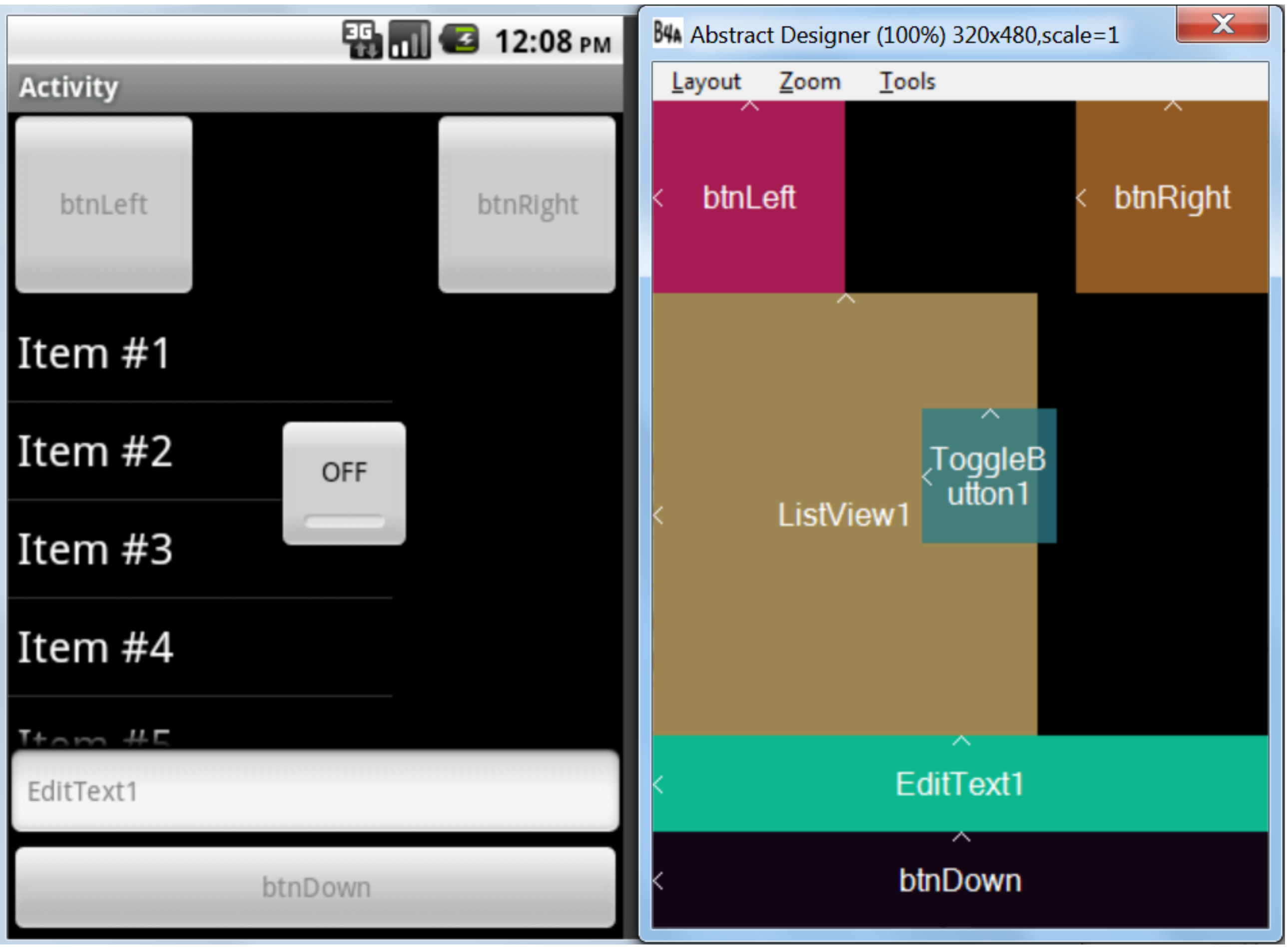
The first step is to add the views and position them with the visual designer (you do not need to be 100% accurate).
Now we will select the designer scripts tab and add the code.
Note that the views are locked when the designer scripts tab is selected.

The same can be done with Anchors, a new feature sine B4A version 3.20.

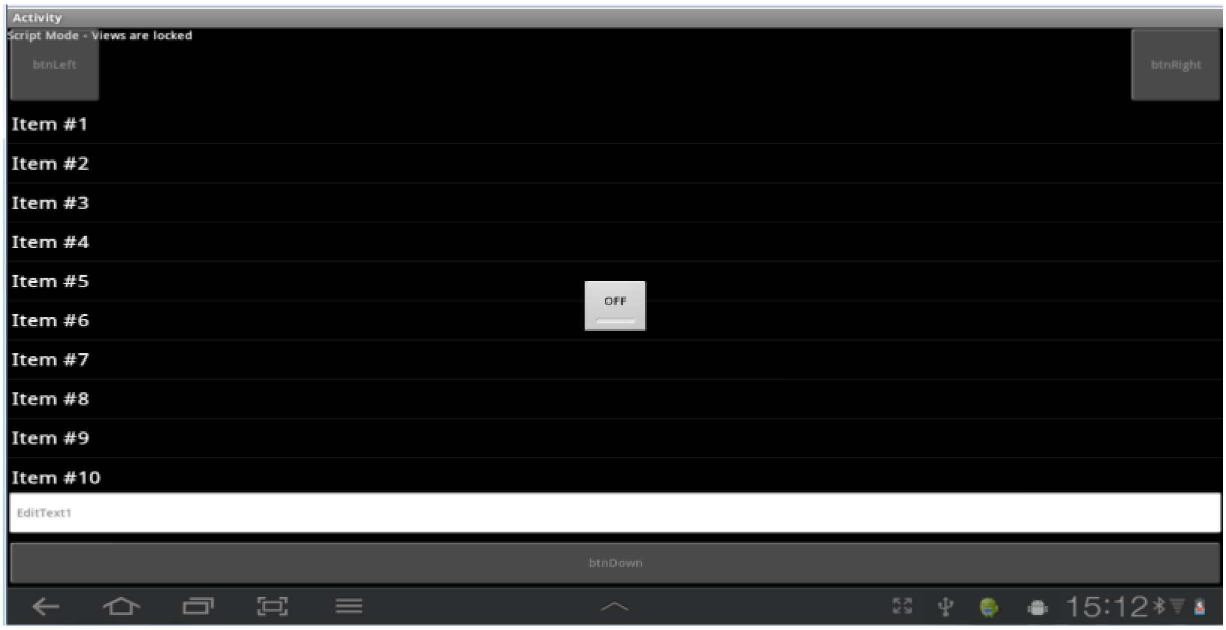
The code in this case is:

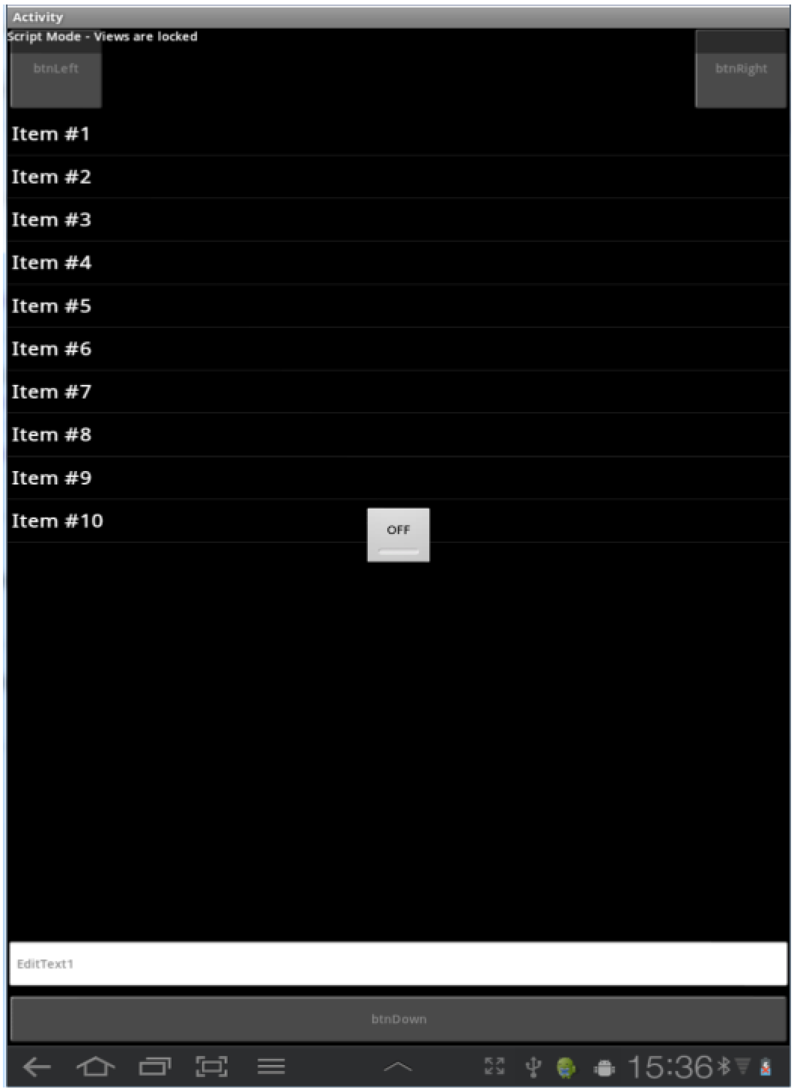
```
'All variants script
btnRight.Right = 100%x

btnDown.Bottom = 100%y
btnDown.Width = 100%x

EditText1.Width = 100%x
EditText1.Bottom = btnDown.Top - 5dip

ListView1.Width = 100%x
ListView1.SetTopAndBottom(btnLeft.Bottom, EditText1.Top)

ToggleButton1.HorizontalCenter = 50%x
ToggleButton1.VerticalCenter = 50%y
```

The result:



10 " tablet


Activity
Script Mode - Views are locked
btnLeft
btnRight
Item #1
Item #2
Item #3
Item #4
Item #5
Item #6
Item #7
Item #8
Item #9
Item #10
OFF
EditText1
btnDown
15:12


Activity
Script Mode - Views are locked
btnLeft
btnRight
Item #1
Item #2
Item #3
Item #4
Item #5
Item #6
Item #7
Item #8
Item #9
Item #10
OFF
EditText1
btnDown
15:36

8.10.2 Supported Properties

The following properties are supported:
- **`Left / Right / Top / Bottom / HorizontalCenter / VerticalCenter`** – Gets or sets the view's position. The view's width or height will not be changed.

- **`Width / Height`** - Gets or Sets the view's width or height.

- **`TextSize`** - Gets or sets the text size.
You should not use 'dip' units with this value as it is already measured in physical units.

- **`Text`** - Gets or sets the view's text. TextSize and Text properties are only available to views that show text.

- **`Image`** - Sets the image file (write-only). Only supported by ImageView.

- **`Visible`** - Gets or sets the view's visible property.

8.10.3 Supported Methods

- **`SetLeftAndRight`** (Left, Right) - Sets the view's left and right properties. This method changes the width of the view based on the two values.

- **`SetTopAndBottom`** (Top, Bottom) - Sets the view's top and bottom properties. This method changes the height of the view based on the two values.

8.10.4 Supported Keywords

- **`And / Or`** - Same as the standard And / Or keywords.

- **`False / True`** - Same as the standard False / True keywords.

- **`Min / Max`** - Same as the standard Min / Max keywords.

- **`Landscape / Portrait`** - Detects if the layout is in landscape or portrait.
Can be used with If / Then.

- **`AutoScale`** - Autoscales a view based on the device physical size. Example: AutoScale(Button1)

- **`AutoScaleAll`** - Autoscales all layout views.

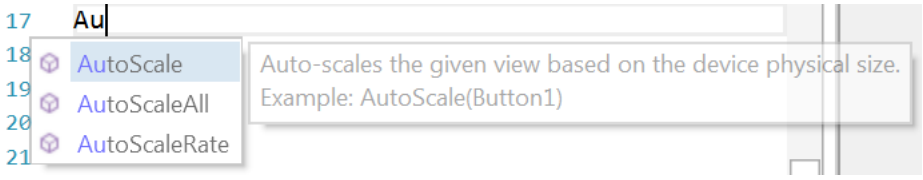
- **`AutoScaleRate`** - Sets the scaling rate, a value between 0 and 1. The default value is 0.3
Example : `AutoScaleRate(0.5)`

- **`ActivitySize`** - Returns the approximate activity size measured in inches.

- **`If . Else If . Else . Then`** condition blocks - Both single line and multiline statements are supported. The syntax is the same as the regular If blocks.

8.10.5 Autocomplete

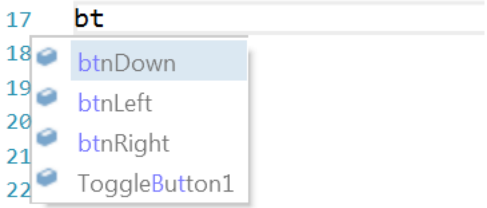
When you begin typing, the Autocomplet function shows all possible keywords or view names containing the written text with the help of the selected keyword.
Example: Au, shows all AutoScale methods.



```
Au
```

Auto-scales the given view based on the device physical size.
Example: AutoScale(Button1)

Example: bt, shows all buttons.



```
bt
```

8.10.6 Notes and tips

- %x and %y values are relative to the view that loads the layout.
Usually it will be the activity. However if you use Panel.LoadLayout then it will be relative to this panel.

- **Use 'dip' units for all specified sizes (except of TextSize).** By using 'dip' units the values will be scaled correctly on devices with higher or lower resolution.

- In most cases it is not recommended to create variants with scales other than 1.0. When you add such a variant you will be given an option to add a normalized variant instead with a scale of 1.0.

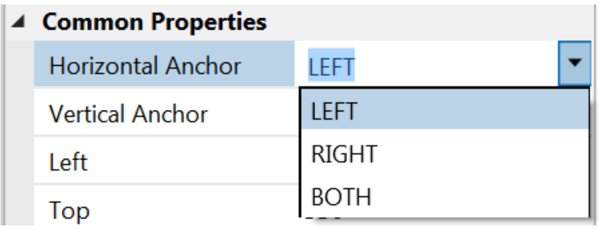
- Variables - You can use variables in the script. You do not need to declare the variables before using them (there is no Private, Public nor Dim keyword in the script).

- Activity.RerunDesignerScript (LayoutFile As String, Width As Int, Height As Int) - In some cases it is desirable to run the script code again during the program. For example you may want to update the layout when the soft keyboard becomes visible. Activity.RerunDesignerScript method allows you to run the script again and specify the width and height that will represent 100%x and 100%y.
In order for this method to work all the views referenced in the script must be declared in Sub Globals.
Note that this method should **not** be used to handle screen orientation changes. In that case the activity will be recreated and the script will run during the Activity.LoadLayout call.

8.11 Anchors

The Horizontal Anchor and Vertical Anchor properties are very powerful to adapt to different screen sizes.

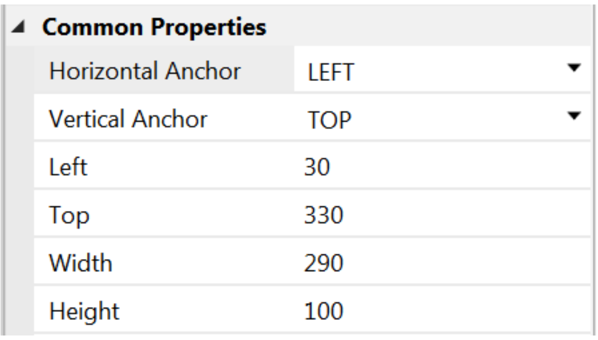
8.11.1 Horizontal Anchor



Common Properties	
Horizontal Anchor	LEFT
Vertical Anchor	
Left	
Top	

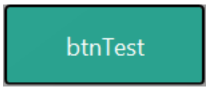
The horizontal anchor property can take three values:

- LEFT



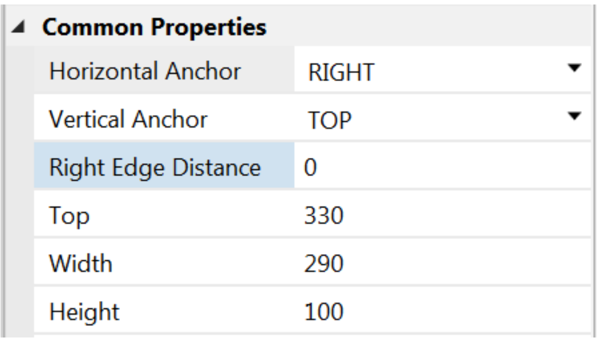
Common Properties	
Horizontal Anchor	LEFT
Vertical Anchor	TOP
Left	30
Top	330
Width	290
Height	100

This is the default value.
The left edge is anchored to the left edge of the parent view with the distance given in the Left property.



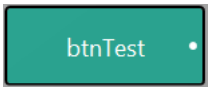
No anchor is shown.

- RIGHT



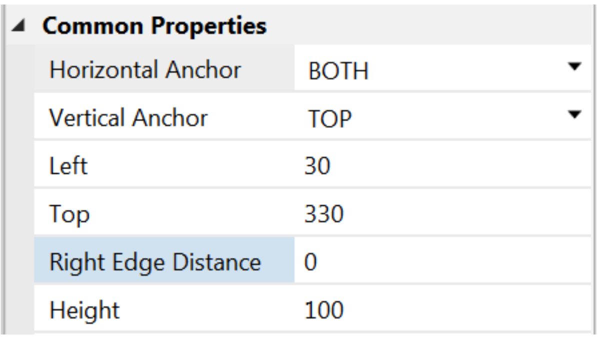
Common Properties	
Horizontal Anchor	RIGHT
Vertical Anchor	TOP
Right Edge Distance	0
Top	330
Width	290
Height	100

The right edge is anchored to the right edge of the parent view with the distance given in the Right Edge Distance property.
The Left property is no longer available because it is defined by the width and the right anchor !



The dot on the right edge shows the anchor.

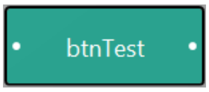
- BOTH



Common Properties	
Horizontal Anchor	BOTH
Vertical Anchor	TOP
Left	30
Top	330
Right Edge Distance	0
Height	100

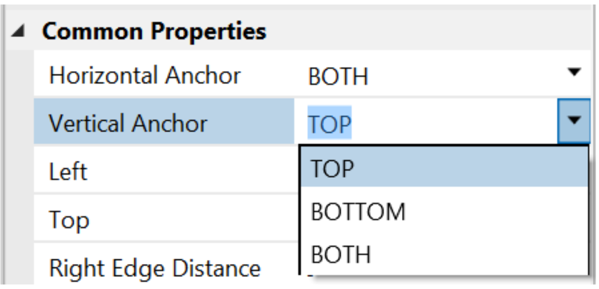
Both edges are anchored.
The Width property is no longer available because it is defined by the anchors !

Setting the Horizontal Anchor property to BOTH is similar to using the SetLeftAndRight function in the Designer Scripts.



The dots on the two edges show the anchors.

8.11.2 Vertical Anchor

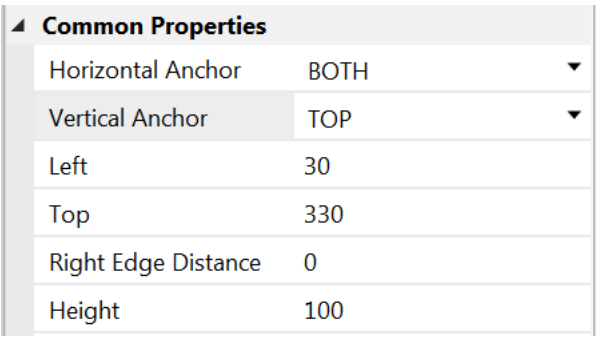


Common Properties

Horizontal Anchor	BOTH
Vertical Anchor	TOP
Left	
Top	
Right Edge Distance	

The vertical anchor property can take three values:

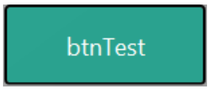
- TOP



Common Properties

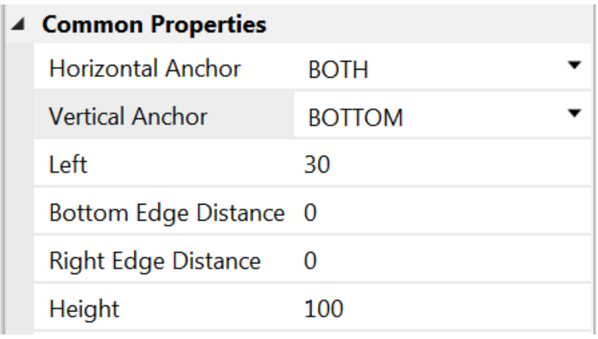
Horizontal Anchor	BOTH
Vertical Anchor	TOP
Left	30
Top	330
Right Edge Distance	0
Height	100

This is the default value.
The top edge is anchored to the top edge of the parent view with the distance given in the Top property.



No anchor is shown.

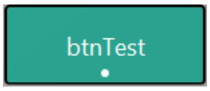
- BOTTOM



Common Properties

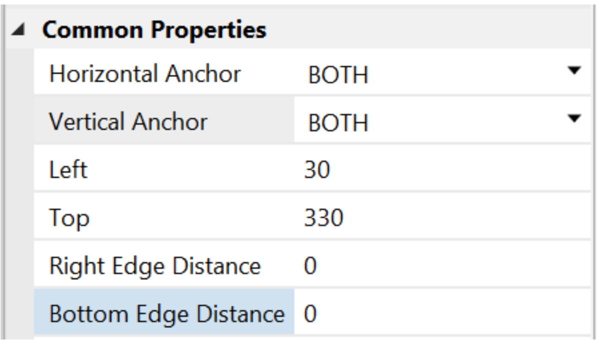
Horizontal Anchor	BOTH
Vertical Anchor	BOTTOM
Left	30
Bottom Edge Distance	0
Right Edge Distance	0
Height	100

The bottom edge is anchored to the bottom edge of the parent view with the distance given in the Bottom Edge Distance property.
The Top property is no longer available because it is defined by the Height and the bottom anchor !



The dot on the bottom edge shows the anchor.

- BOTH

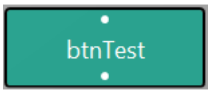


Common Properties

Horizontal Anchor	BOTH
Vertical Anchor	BOTH
Left	30
Top	330
Right Edge Distance	0
Bottom Edge Distance	0

Both edges are anchored.
The Height property is no longer available because it is defined by the anchors !

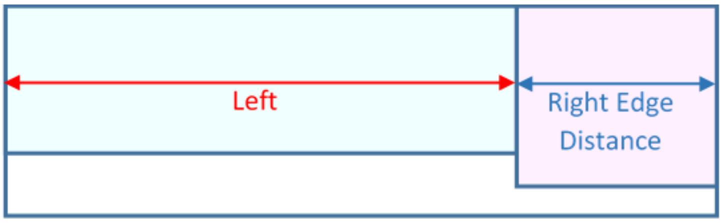
Setting the Vertical Anchor property to BOTH is similar to using the SetTopAndBottom function in the Designer Scripts.



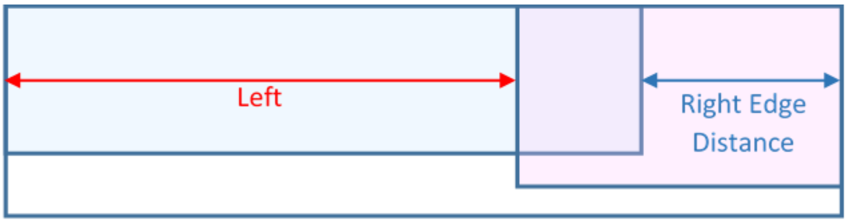
The dots on the two edges show the anchors.

What happens when we set the horizontal anchor of the two views below to BOTH and change the parent view width?

The left view's right edge is anchored to the right edge of the parent view with the Right Edge Distance.
The right view's left edge is anchored to the left edge of the parent view with the Left distance.



If we increase the width of the parent view we get the layout below.



The left view's right edge is still at the Right Edge Distance from the parent view's right edge.
The right view's left edge is still at the Left distance from the parent view's left edge.
The result is an overlapping of both views.

In this case you must adjust the views in the Designer Scripts with the `SetLeftAndRight` method!

For example:

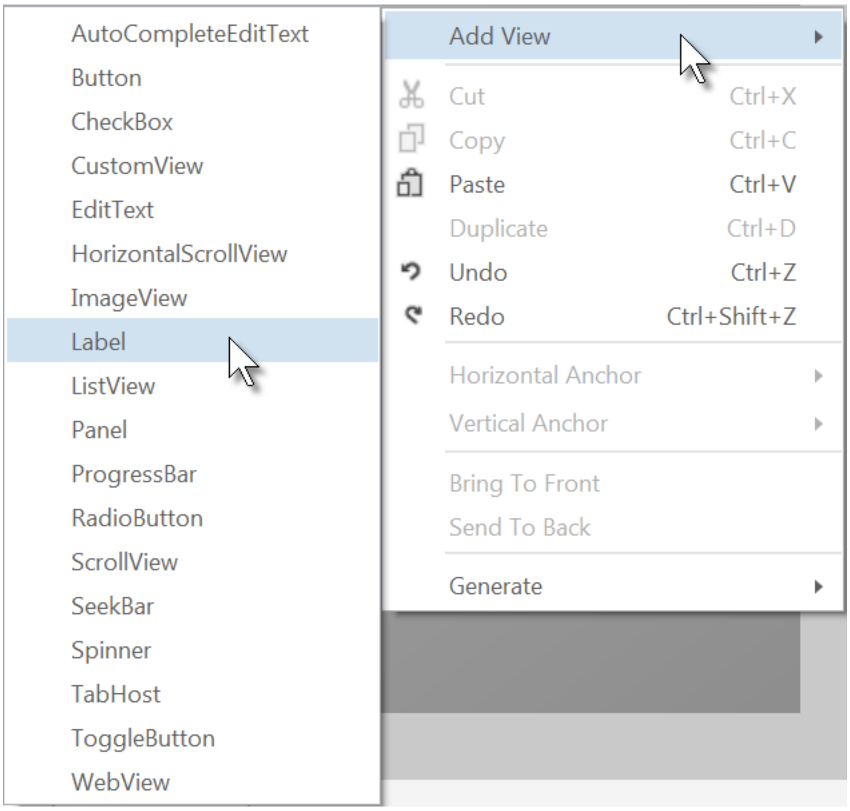
```
LeftView.SetLeftAndRight(0, 67%x)
RightView.SetLeftAndRight(33%x, 100%x)
```

8.11.3 First example

The examples shown in this chapter are based on the *DesignerAnchor* project.

First we add a label on top of the screen which should cover the whole width and stay on top.

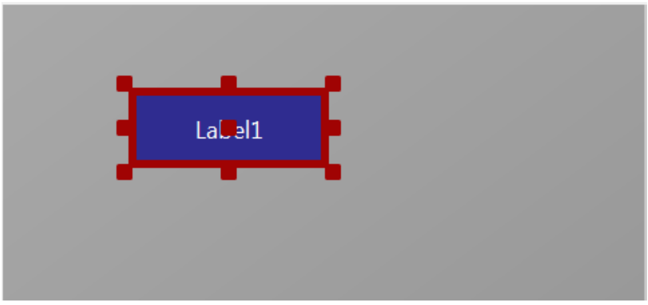
In the AbstractDesigner right-click somewhere on the screen, the menu below will be displayed:



Click on Add View.

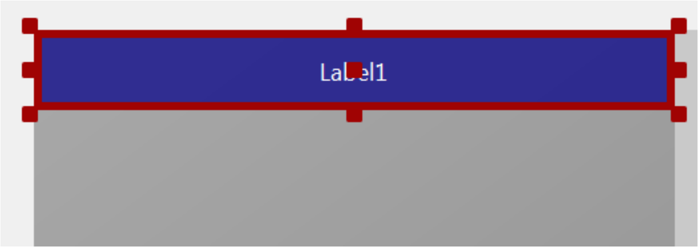
On the left side appears the list of the views you can add to the layout.

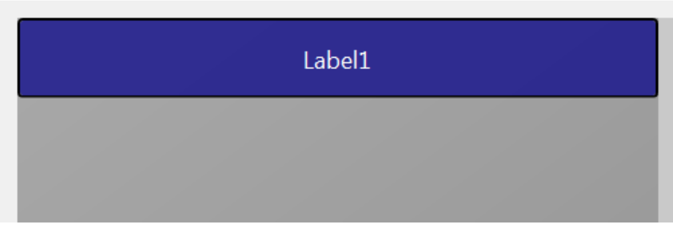
Click on Label.



The Label is added.

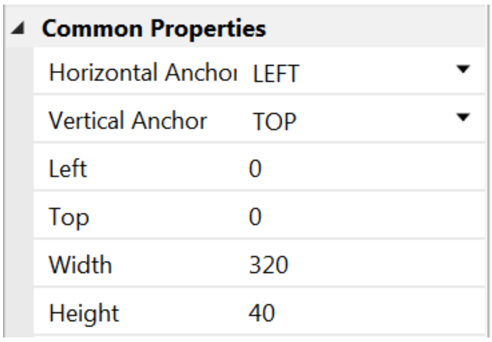
Move the label’s upper left corner to the upper left corner of the screen and stretch it to fill the whole width of the screen.





Click somewhere else on the screen to remove the red anchors.

No anchors are displayed.



◢ **Common Properties**

Horizontal Anchoı	LEFT ▾
Vertical Anchor	TOP ▾
Left	0
Top	0
Width	320
Height	40

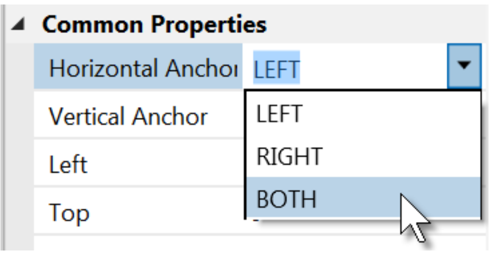
Click again on the Label and we see these properties:

Left = 0

Top = 0

Width = 320 full layout width

Height = 40

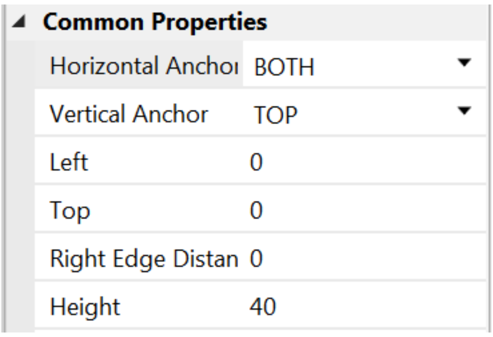


◢ **Common Properties**

Horizontal Anchoı	LEFT ▾
Vertical Anchor	LEFT
Left	RIGHT
Top	BOTH

Now we change the ‘Horizontal Anchor’ property :

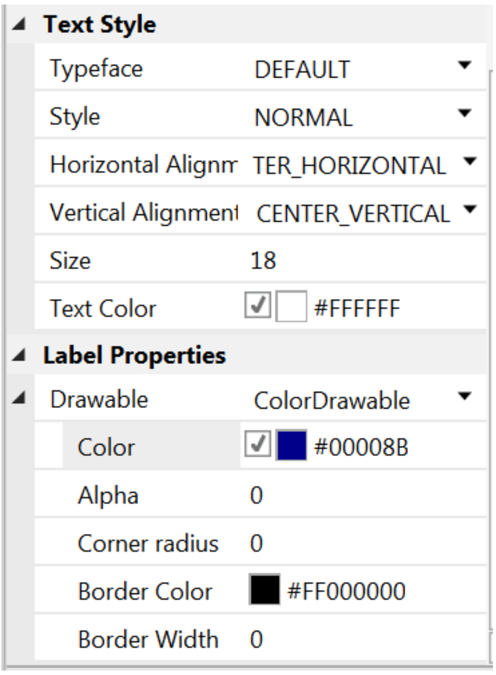
Click on BOTH.



◢ **Common Properties**

Horizontal Anchoı	BOTH ▾
Vertical Anchor	TOP ▾
Left	0
Top	0
Right Edge Distan	0
Height	40

We see that the properties changed:
Left, Top and Height are still the same.
But Width has disappeared and is replaced by
Right Edge Distance = 0
Its value = 0 because the right edge is on the right edge of the screen.



◢ **Text Style**

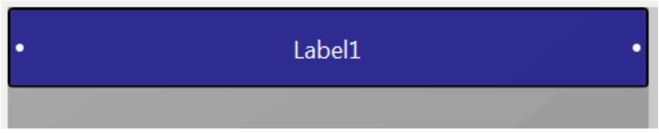
Typeface	DEFAULT ▾
Style	NORMAL ▾
Horizontal Alignm	TER_HORIZONTAL ▾
Vertical Alignmen	CENTER_VERTICAL ▾
Size	18
Text Color	☑ □ #FFFFFF

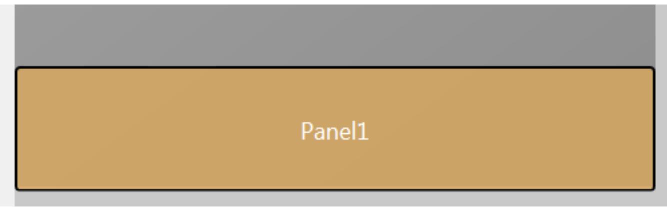
◢ **Label Properties**

◢ Drawable	ColorDrawable ▾
Color	☑ ■ #00008B
Alpha	0
Corner radius	0
Border Color	■ #FF000000
Border Width	0

Set the other properties like in the picture.

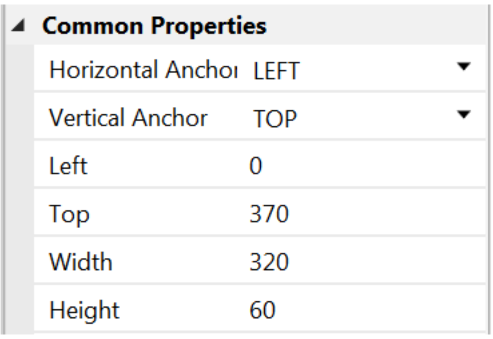
Now we see the two anchors on the left and the right edge.





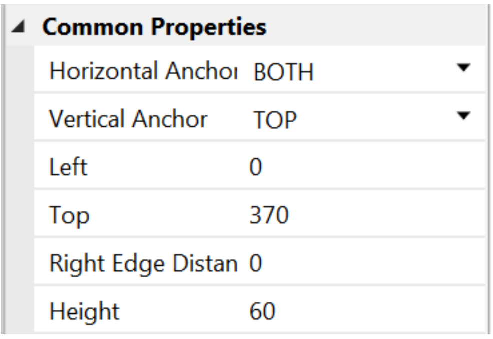
Now, let us add a Panel at the bottom of the screen covering also the whole screen width.

The properties look like in the picture.

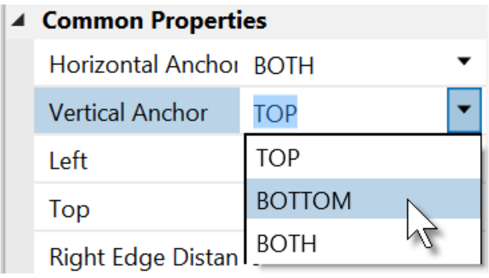


Common Properties	
Horizontal Anchoı	LEFT
Vertical Anchor	TOP
Left	0
Top	370
Width	320
Height	60

We set the Horizontal Anchor to BOTH.
Same as for Label1.

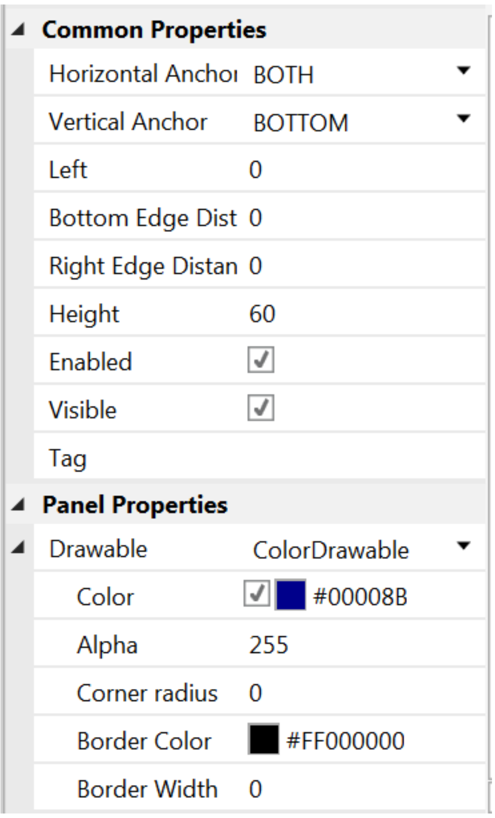


Common Properties	
Horizontal Anchoı	BOTH
Vertical Anchor	TOP
Left	0
Top	370
Right Edge Distan	0
Height	60



Common Properties	
Horizontal Anchoı	BOTH
Vertical Anchor	TOP
Left	
Top	
Right Edge Distan	

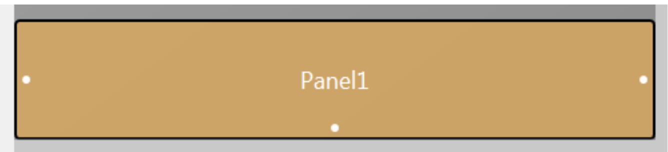
We set the Vertical Anchor to BOTTOM.



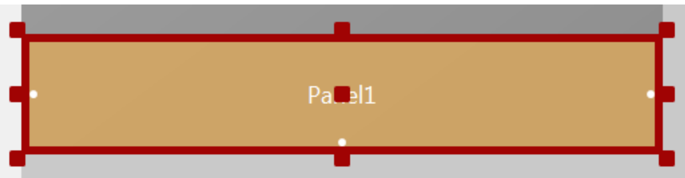
Common Properties	
Horizontal Anchoı	BOTH
Vertical Anchor	BOTTOM
Left	0
Bottom Edge Dist	0
Right Edge Distan	0
Height	60
Enabled	☑
Visible	☑
Tag	
Panel Properties	
Drawable	ColorDrawable
Color	☑ ■ #00008B
Alpha	255
Corner radius	0
Border Color	■ #FF000000
Border Width	0

The Top property is replaced by the:
Bottom Edge Distance = 0 property.
Its value = 0 because we anchor the bottom edge of Panel1 to the screens bottom edge.

We see the three anchors.

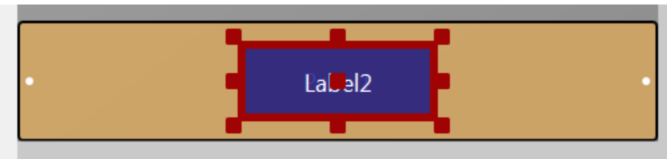


And set the other properties like this.

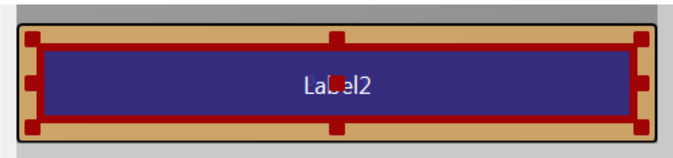


Now we add a second label onto Panel1.

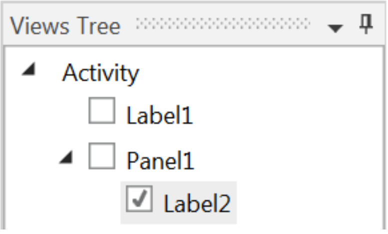
Click on Panel1 to select it.



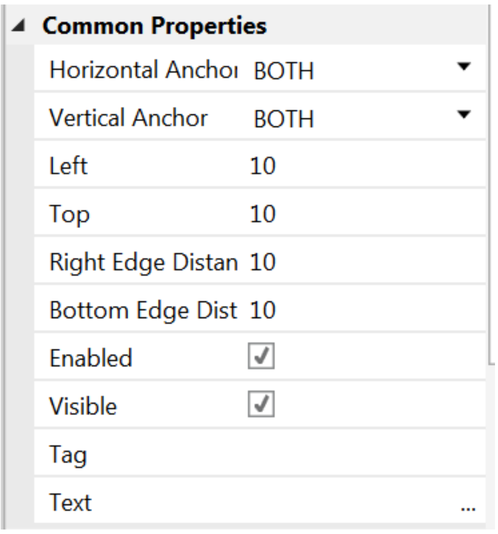
Add the label.



Move and size the label like in the picture with the Left, Top, Width and Height properties like in the list below.



In the Views Tree window we see that Label2 is shifted to the right because its parent view is Panel1 and not the Activity like for Label1 and Panel1!



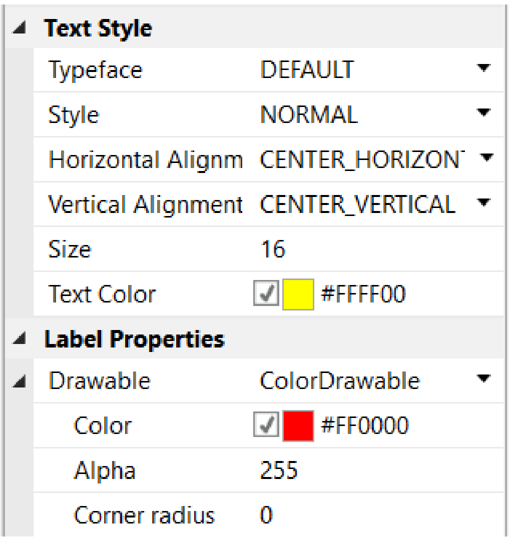
Common Properties

Horizontal Anchor	BOTH
Vertical Anchor	BOTH
Left	10
Top	10
Right Edge Distan	10
Bottom Edge Dist	10
Enabled	☑
Visible	☑
Tag	
Text	...

Set the Horizontal and Vertical Anchors to BOTH.

The properties
Left = 10 and
Top = 10 remain the same.

Right Edge Distance = 10 and
Bottom Edge Distance = 10
The two values are equal to 10 because we want a 'frame' around Label2.



Text Style

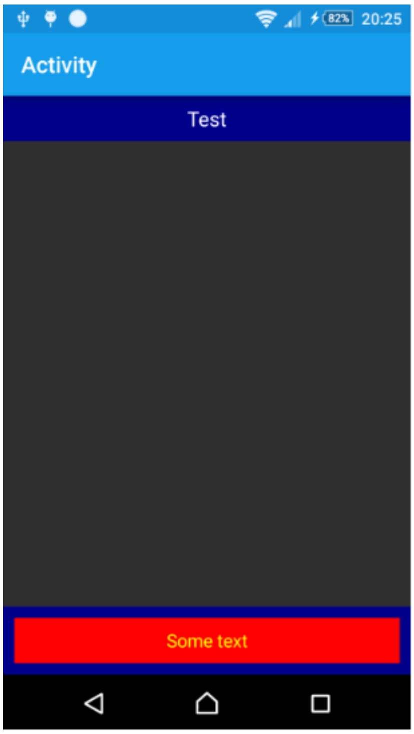
Typeface	DEFAULT
Style	NORMAL
Horizontal Alignm	CENTER_HORIZON
Vertical Alignment	CENTER_VERTICAL
Size	16
Text Color	☑ 🟨 #FFFF00

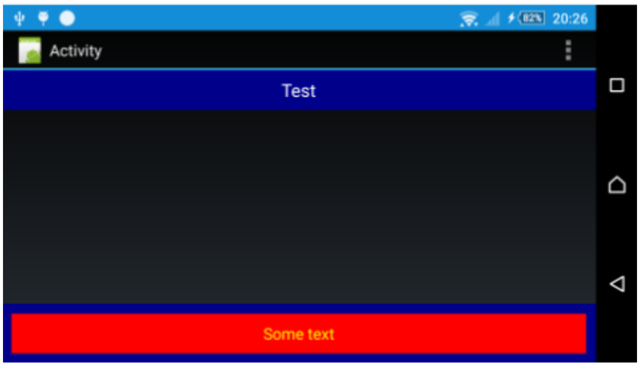
Label Properties

Drawable	ColorDrawable
Color	☑ 🟥 #FF0000
Alpha	255
Corner radius	0

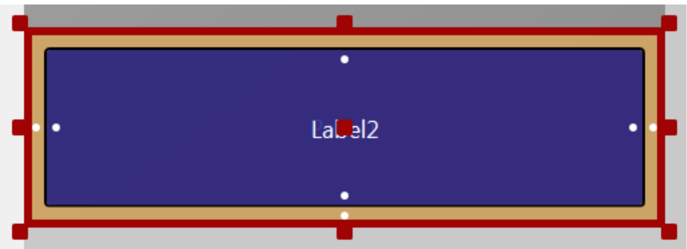
Set the other properties like in the picture.

And the result looks like the pictures below in portrait and landscape screen orientations.

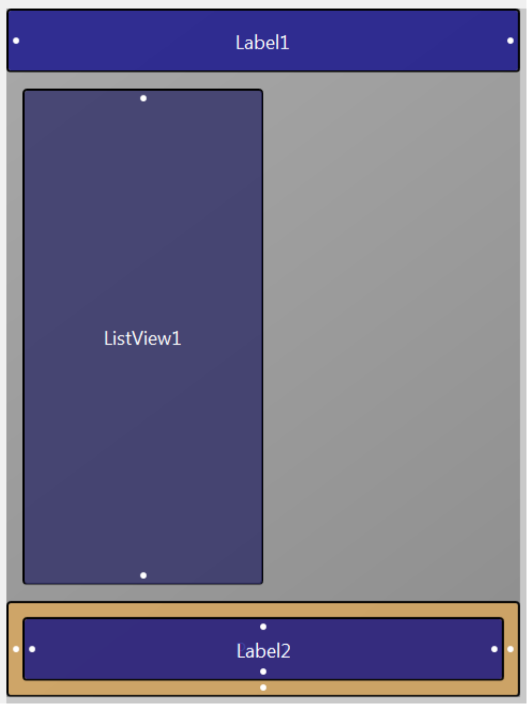




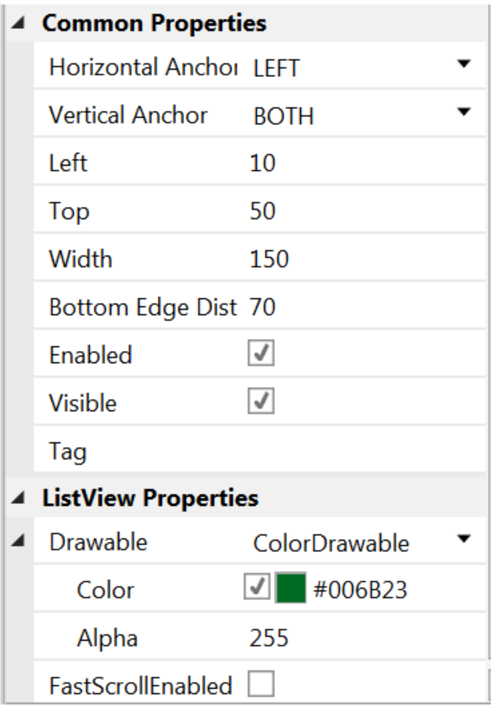
To demonstrate the anchor feature we move, in the Abstract Designer, the top edge of Panel1 upwards.



We see that the bottom edge of Label2 remains at its place !



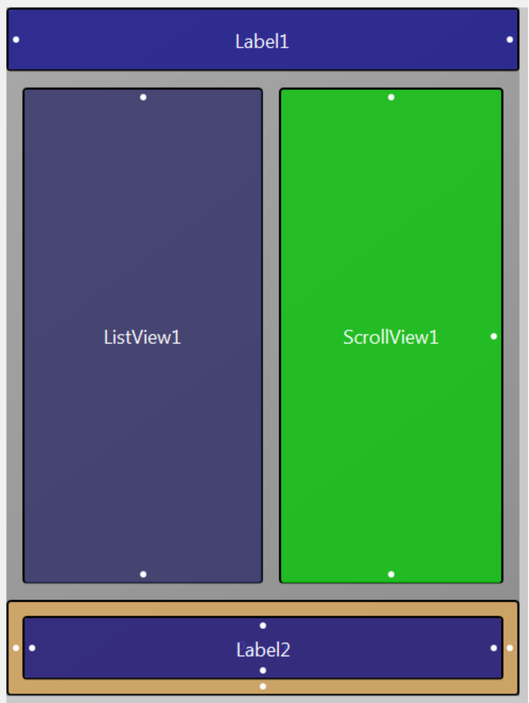
Now, we add a ListView onto the left half of the screen and vertically positioned between Label1 and Panel1 leaving a small space.



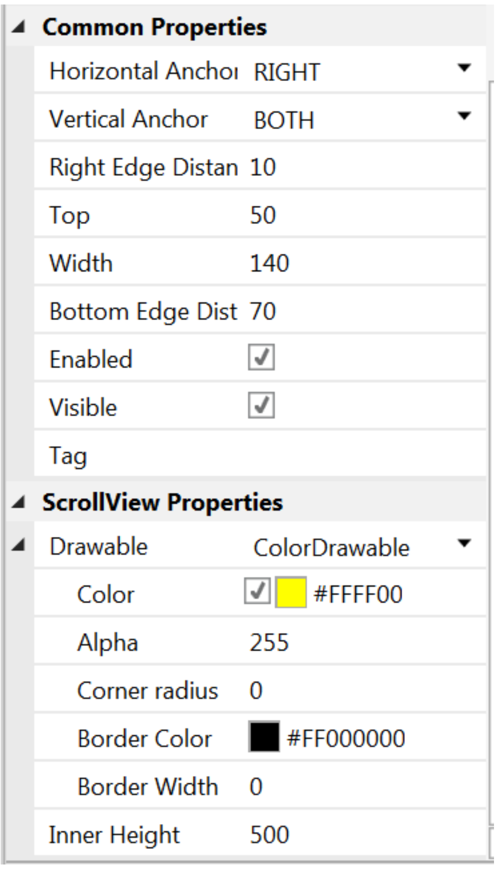
Common Properties	
Horizontal Anchoı	LEFT ▾
Vertical Anchor	BOTH ▾
Left	10
Top	50
Width	150
Bottom Edge Dist	70
Enabled	☑
Visible	☑
Tag	
ListView Properties	
Drawable	ColorDrawable ▾
Color	☑ 🟩 #006B23
Alpha	255
FastScrollEnabled	☐

We set the vertical anchor to BOTH.

And set the other properties like in the picture.



Now, we add a ScrollView on the right half of the screen also positioned between Label1 and Panel1 leaving a small space.



Common Properties	
Horizontal Anchoı	RIGHT ▾
Vertical Anchor	BOTH ▾
Right Edge Distan	10
Top	50
Width	140
Bottom Edge Dist	70
Enabled	☑
Visible	☑
Tag	
ScrollView Properties	
Drawable	ColorDrawable ▾
Color	☑ 🟨 #FFFF00
Alpha	255
Corner radius	0
Border Color	⬛ #FF000000
Border Width	0
Inner Height	500

We set the horizontal anchor to RIGHT.
We set the vertical anchor to BOTH.
And set the other properties like in the picture.

In the code we:

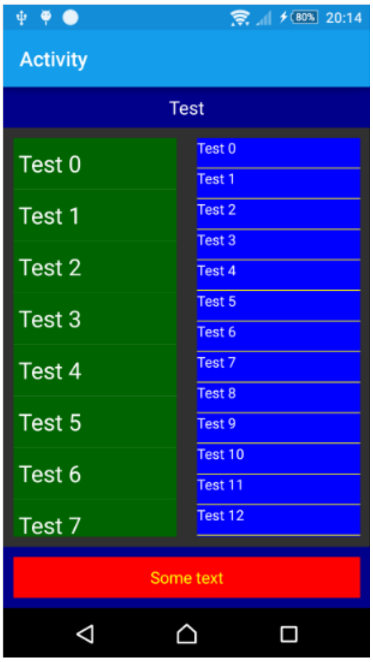
- Load the layout.
- Fill the ListView and the ScrollView.

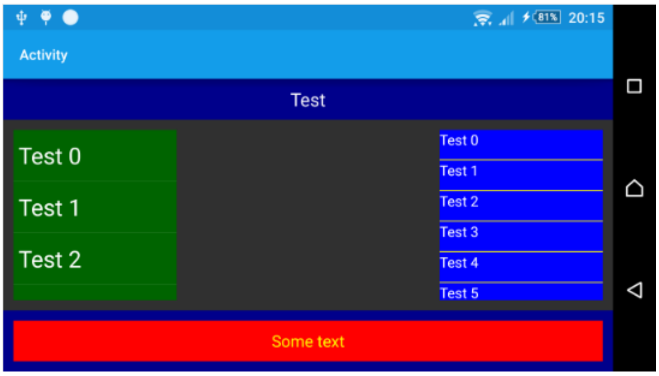
```
Sub Activity_Create(FirstTime As Boolean)
   Activity.LoadLayout("Main")
   FillListView
   FillScrollView
End Sub
```

The two filling routines.

```
Sub FillListView
   Private i As Int

   For i = 0 To 20
      ListView1.AddSingleLine("Test " & i)
   Next
End Sub

Sub FillScrollView
   Private i As Int
   Private lblHeight = 30dip As Int

   For i = 0 To 20
      Private lbl As Label
      lbl.Initialize("lbl")
      ScrollView1.Panel.AddView(lbl, 0, i*lblHeight, 100%x-20dip, lblHeight-1dip)
      lbl.Color = Colors.Blue
      lbl.TextColor = Colors.White
      lbl.Text = "Test " & i
      lbl.Tag = i
   Next
   ScrollView1.Panel.Height = i * lblHeight
End Sub
```

And the result:
In portrait and landscape screen orientations.





We see that the anchors work fine.
But, we see that there is a big gap between the ListView and the ScrollView.

Script - General

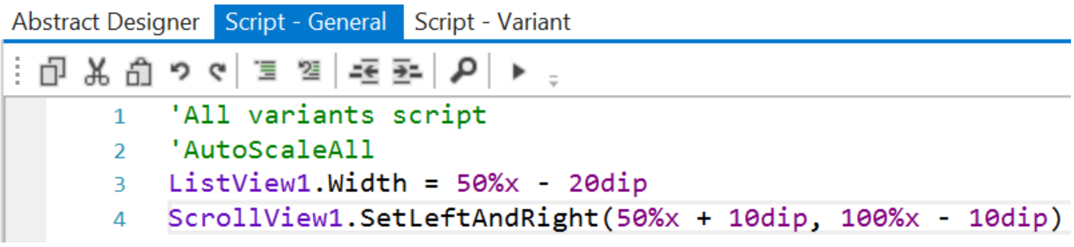
Why do we have this gap ?

Because we set the Horizontal Anchor of the ListView to LEFT
and the Horizontal Anchor of the ScrollView to RIGHT.

But the Width property remains the same and that’s why we get the gap between the two views when the screen width is wider than the layout screen width.

To adjust the width we add two lines in the DesignerScripts.

Click on Designer Scripts to show the DesignerScripts window.



```
'All variants script
'AutoScaleAll
ListView1.Width = 50%x - 20dip
ScrollView1.SetLeftAndRight(50%x + 10dip, 100%x - 10dip)
```

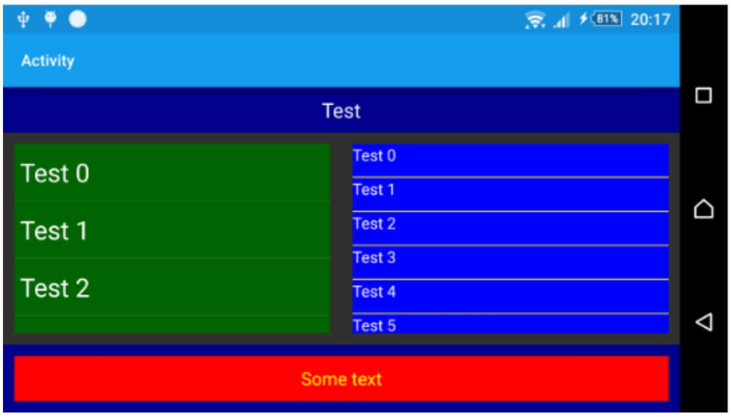
Here we comment AutoScaleAll and add the following two lines:

```
'AutoScaleAll
ListView1.Width = 50%x - 20dip
ScrollView1.SetLeftAndRight(50%x + 10dip, 100%x - 10dip)
```

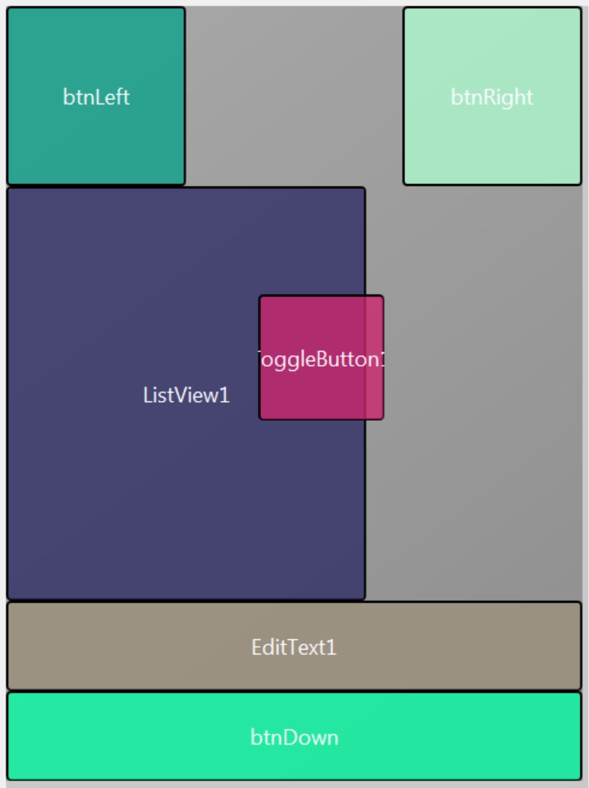
The anchors are valid in the AbstractDesigner but not in Designer Scripts.

For ListView1 it’s enough to set its Width property.
But for ScrollView1 we need to define both properties Left and Right which is done with `SetLeftAndRight` because the RIGHT anchor is lost.

And the new result in landscape orientation.

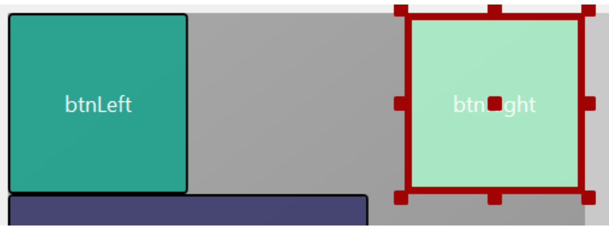


8.11.4 Second example

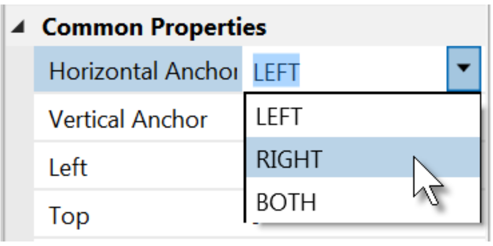


We do the same exercise as in chapter 8.10 Designer Scripts but with the Anchor functions.

This is the layout file from the DesignerScripts example.



Click on btnRight.



Common Properties

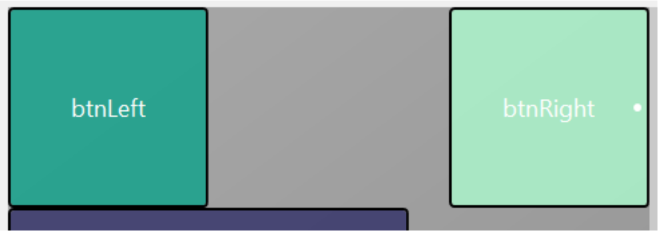
Horizontal Anchor LEFT

Vertical Anchor

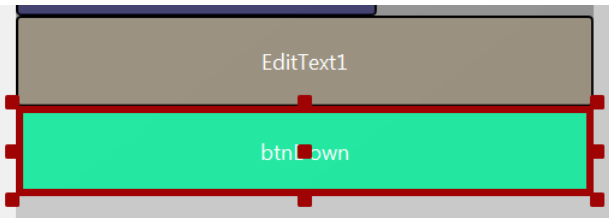
Left

Top

And set the Horizontal Anchor property to RIGHT.

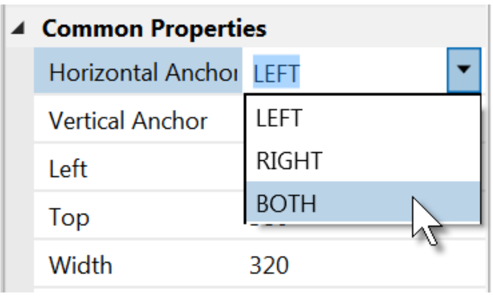


The RIGHT anchor arrow is now displayed.



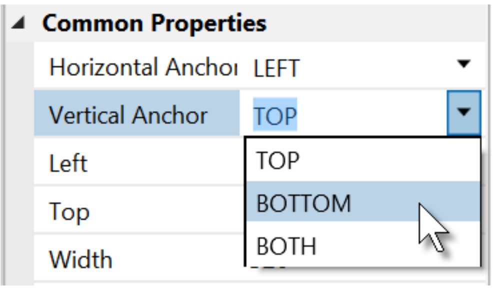
Click on btnDown.

Set the Horizontal Anchor property to BOTH.

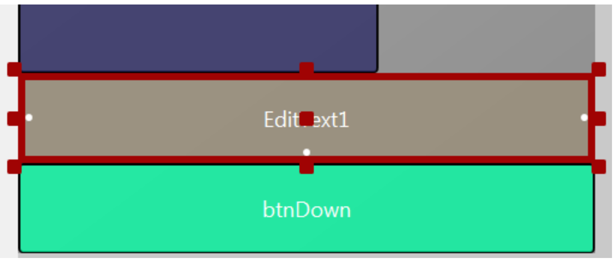

Common Properties

Horizontal Anchoı	LEFT
Vertical Anchor	
Left	
Top	
Width	320

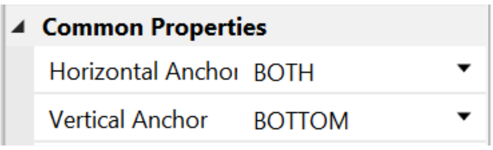
Set the Vertical Anchor property to BOTTOM.


Common Properties

Horizontal Anchoı	LEFT
Vertical Anchor	TOP
Left	
Top	
Width	

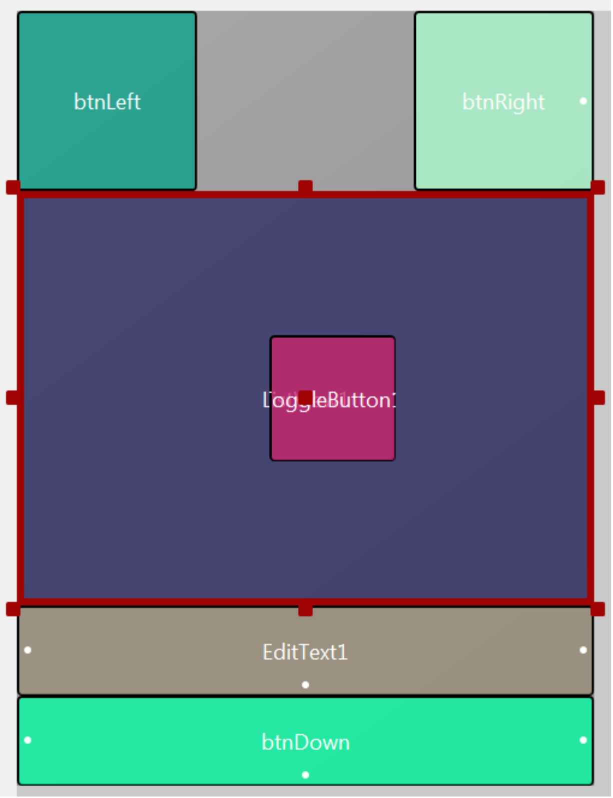


Click on EditText1.


Common Properties

Horizontal Anchoı	BOTH
Vertical Anchor	BOTTOM

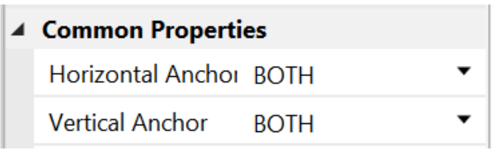
Set the anchors to BOTH and BOTTOM.



Click on ListView1

Move the right edge to the right edge of the screen

and set both anchors to BOTH.

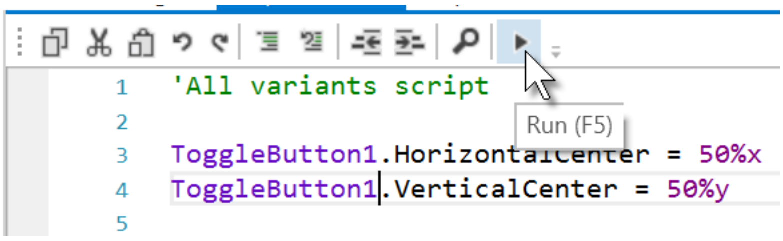

Common Properties

Horizontal Anchoı	BOTH
Vertical Anchor	BOTH

Remains ToggleButton1.
For this view we need the DesignerScript we cannot adjust it with anchors.

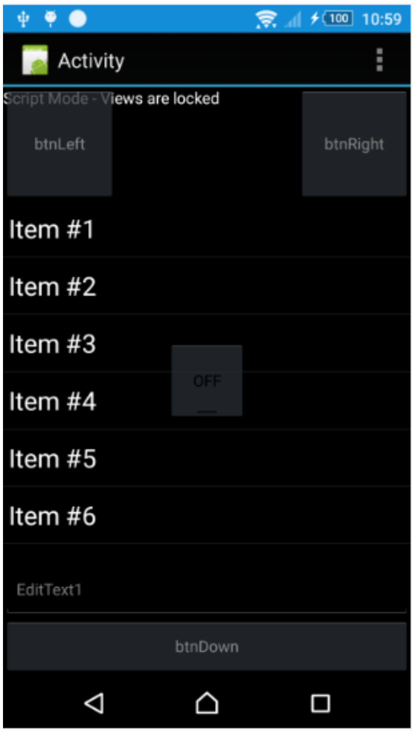
```
'All variants script

ToggleButton1.HorizontalCenter = 50%x
ToggleButton1.VerticalCenter = 50%y
```

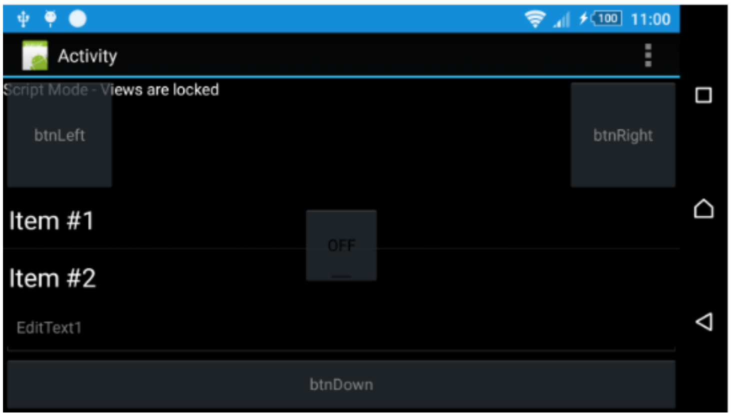
In the Designer Script window click on ▶ to run the script.



```
'All variants script

ToggleButton1.HorizontalCenter = 50%x
ToggleButton1.VerticalCenter = 50%y

```

And the result in portrait and landscape.





8.12 AutoScale

AutoScale includes three functions:

- AutoScaleRate(rate)
- AutoScale
- AutoScaleAll

Larger devices offer a lot more available space. The result is that even if the physical size of a view is the same, it just "feel" smaller.
Some developers use %x and %y to specify the views size. However the result is far from being perfect. The layout will just be stretched.
The solution is to combine the "dock and fill" strategy with a smart algorithm that increases the views size and text size based on the running device physical size.

The AutoScale function is based on the standard variant (320 x 480, scale = 1.0).
Since B4A version 3.2 AutoScale takes into account the dimensions of the variant defined in the layout.
For other screen sizes and resolutions AutoScale calculates a scaling factor based on the equations below.

```
delta = ((100%x + 100%y) / (320dip + 430dip) - 1)
rate = 0.3 'value between 0 to 1.
scale = 1 + rate * delta
```

AutoScale multiplies the Left / Top / Width and Height properties by the scale value.
If the view has a Text property this one is also multiplied by the scale value.

You can play with the 'rate' value. The rate determines the change amount in relation to the device physical size.
Value of 0 means no change at all. Value of 1 is almost similar to using %x and %y: If the physical size is twice the size of the standard phone then the size will be twice the original size.
Values between 0.2 and 0.5 seem to give good results. The default value is 0.3.
Be careful when you ‘downsize’ a layout defined for a big screen to a small screen. The views may become very small.
Note: The size of the CheckBox and RadioButton images is the same for all screen sizes.

The abstract designer is useful to quickly test the effect of this value.

Functions:

- AutoScaleRate(rate) Sets the rate value for above equations.
  Example: `AutoScaleRate(0.5)` Sets the rate value to 0.5.

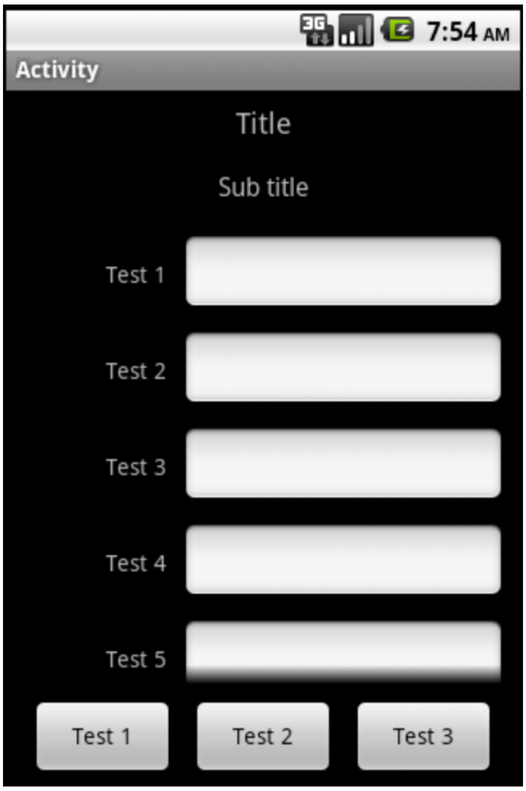
- AutoScale(View) Scales the given view.
  Example : `AutoScale(btnTest1)`
  This is equivalent to :

```
btnTest1.Left = btnTest1.Left * scale
btnTest1.Top = btnTest1.Top * scale
btnTest1.Width = btnTest1.Width * scale
btnTest1.Height = btnTest1.Height * scale
btnTest1.TextSize = btnTest1.TextSize * scale
```

- AutoScaleAll Scales all the views in the selected layout

8.12.1 Simple AutoScale example with only one layout variant

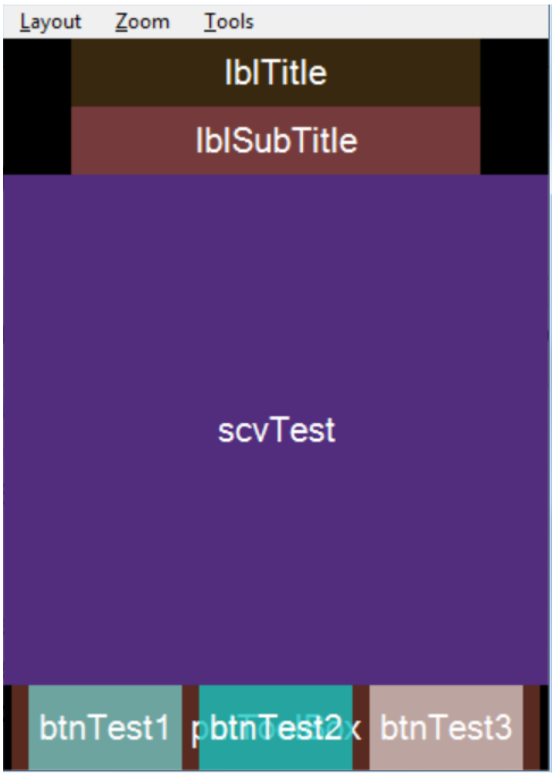
We will AutoScale a simple example with the layout below, source code AutoScaleExample1:

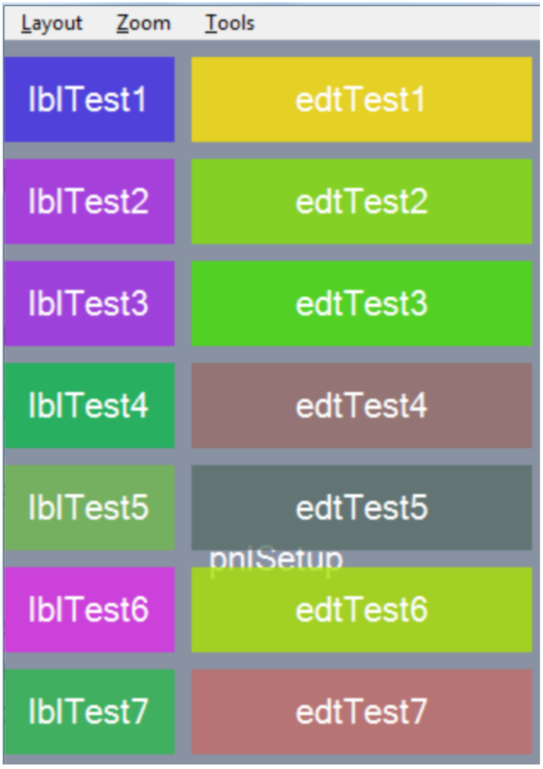


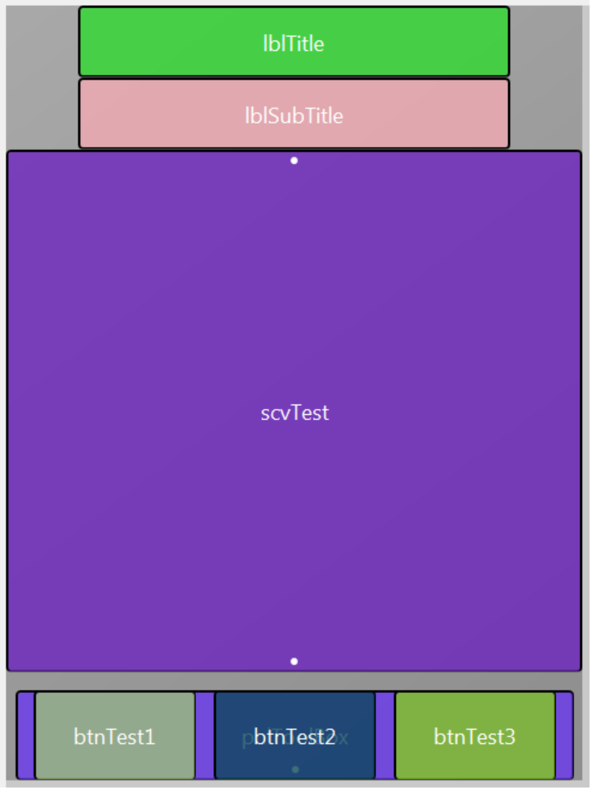
We have:

- 2 Labels on the top of the screen :
  - lblTitle
  - lblSubTitle
- 1 ScrollView in the middle of the screen :
  - scvTest containing
    - one Panel pnlSetup with
    - 10 Labels lblTest1 to lblTest10
    - 10 EditTexts edtTest1 to edtTest10
- 1 Panel at the bottom of the screen :
  - pnlToolBox
  - Containing 3 Buttons
    - btnTest1
    - btnTest2
    - btnTest3

We have two layout files *Main* for the main screen and *Panel* for the ScrollView content with only one layout variant 320 x 480 scale = 1 (160dip) for each.







Main layout file:

We want to have the:

- Two Labels on the top of the screen and centred horizontally on the screen.
- ToolBox Panel on the bottom of the screen and centred horizontally.
- ScrollView filling the space between the SubTitle Label and the ToolBox Panel.

Note: Look at the anchors especially for the ToolBox and the ScrollView.

First we set the AutoScaleRate to 0.5 with:

```
AutoScaleRate(0.5)
```

and AutoScale all views with:

```
AutoScaleAll
```

The two Labels are already on top so there is no need to change the Top property for different screen sizes.

But we need to centre them on the screen with:

```
lblTitle.HorizontalCenter = 50%x
lblSubTitle.HorizontalCenter = 50%x
```

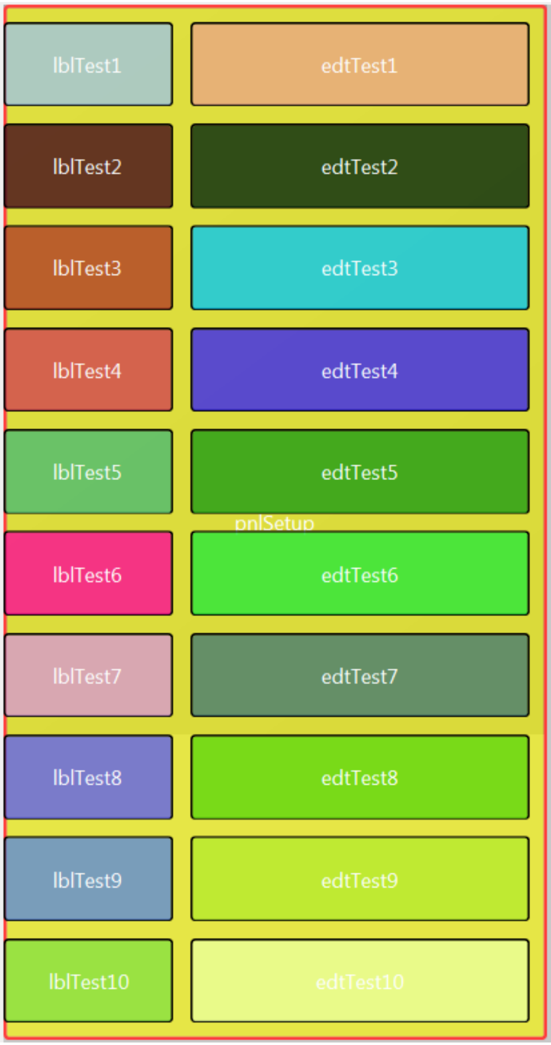
Then we centre the ToolBox with:

```
pnlToolBox.HorizontalCenter = 50%x
```

And we set the Vertical Anchor property of the ToolBox to BOTTOM to ‘anchor’ it to the bottom of the screen.
This is needed because not all screens have the same width / height ratio and in landscape orientation it would even not be visible.

Then we set the Vertical Anchor property of the ScrollView to BOTH because we want it to fill the space between lblSubTitle and pnlToolBox.
We set the Bottom Edge Distance property to 60 to leave a small space of 10dip between the ScrollView and the ToolBox.
Code in the Designer Scripts of the Main layout in the area for All variants script:

```
'All variants script

'Set the rate value to 0.5
AutoScaleRate(1)

'Scale all the views in the layout
AutoScaleAll

'Center the Labels horizontally to the middle of the screen
lblTitle.HorizontalCenter = 50%x
lblSubTitle.HorizontalCenter = 50%x

'Center the ToolBox Panel horizontally to the middle of the screen
pnlToolBox.HorizontalCenter = 50%x

'Center the ScrollView horizontally to the middle of the screen
scvTest.HorizontalCenter = 50%x
```



Panel layout file:

All the Label and EditText views are on a Panel.
This is needed because they occupy more space than the screen size.
This layout file is loaded into the ScrollView.Panel.

For this layout file we set also the AutoScaleRate value to 0.5 with:
`AutoScaleRate(0.5)`
and AutoScale all views with:
`AutoScaleAll`

There is no need to modify any view after autoscaling.

Code in the Designer Scripts of the Panel layout in the area for ‘All variants script’:
The whole code is very simply:

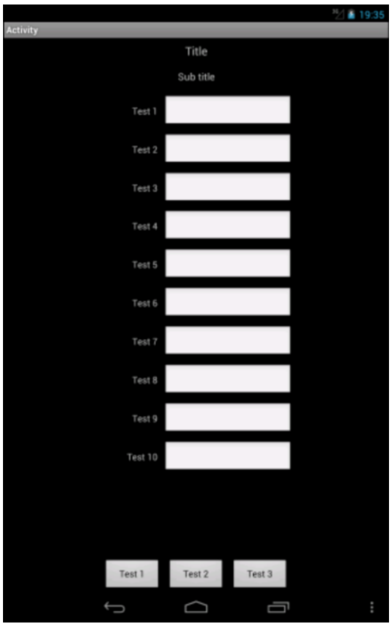
```
'All variants script
AutoScaleRate(0.5)
AutoScaleAll
```

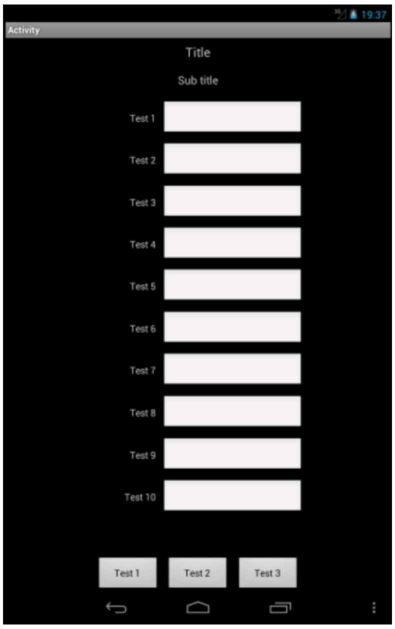
In the program the code is the following:

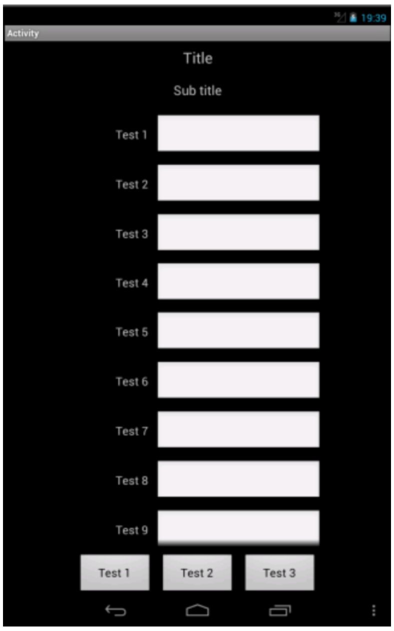
```
Sub Activity_Create(FirstTime As Boolean)
  ' load the Main layout file
  Activity.LoadLayout("Main")

  ' load the ScrollView.Panel layout file
  scvTest.Panel.LoadLayout("Panel")

  ' set the ScrollView.Panel.Height to the pnlSetup Panel height
  scvTest.Panel.Height = pnlSetup.Height
End Sub
```

We load the Main layout file into the Activity with `Activity.LoadLayout("Main")`.
We load the Panel layout file into the ScrollView with `scvTest.Panel.LoadLayout("Panel")`.
We set the ScrollView.Panel.Height to the height of the Panel in the layout file with:
`scvTest.Panel.Height = pnlSetup.Height`

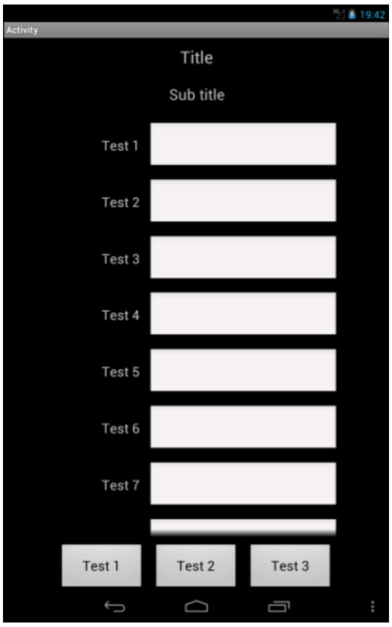
Screenshots of an 800/1280 10" screen Emulator with different Rate values:
All the images have been downsized.

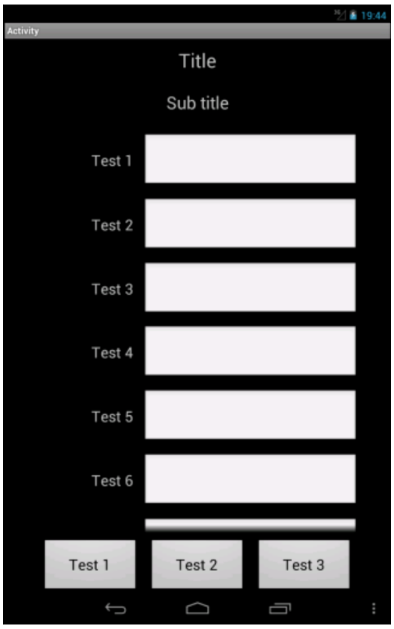


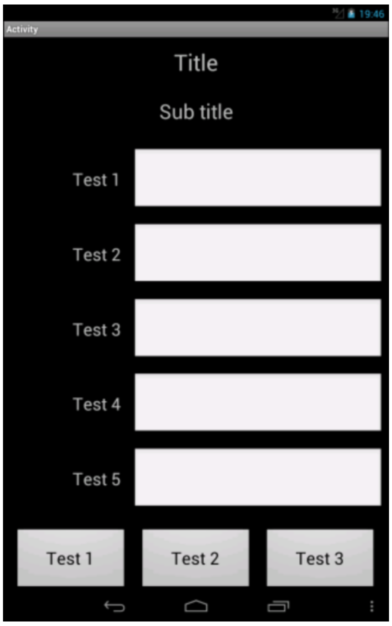




Rate = 0 Rate = 0.1 Rate = 0.3

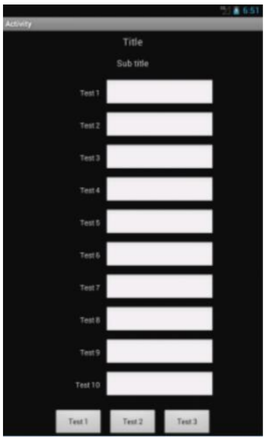


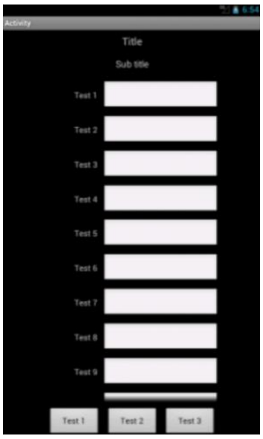


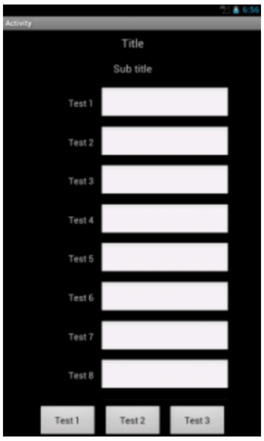


Rate = 0.5 Rate = 0.7 Rate = 1.0

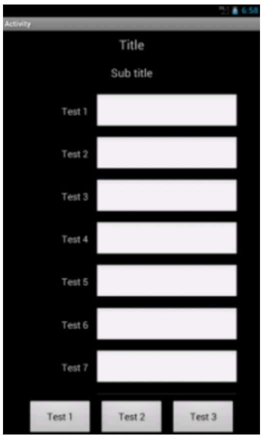
Screenshots of an 480/800 7" screen Emulator with different Rate values:

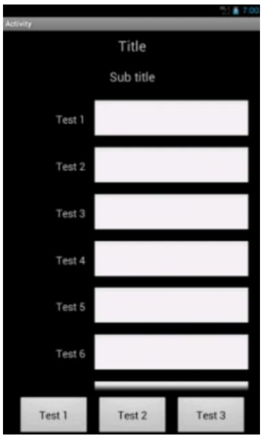


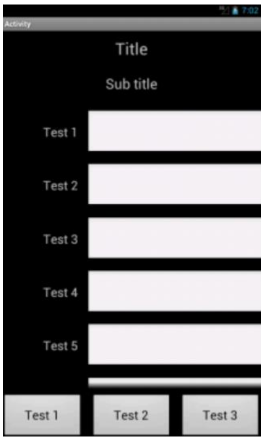




Rate = 0 Rate = 0.1 Rate = 0.3

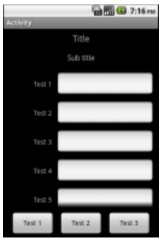






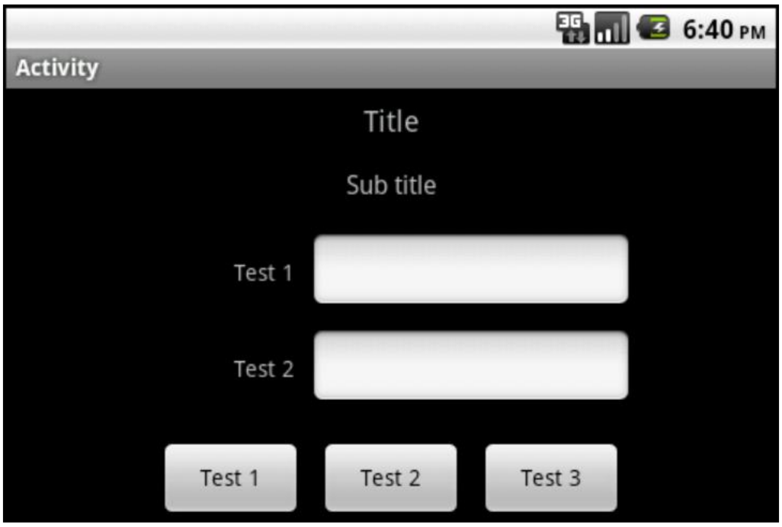
Rate = 0.5 Rate = 0.7 Rate = 1

Screenshots of a 320/480 3.5" screen Emulator. The Rate value has no influence.

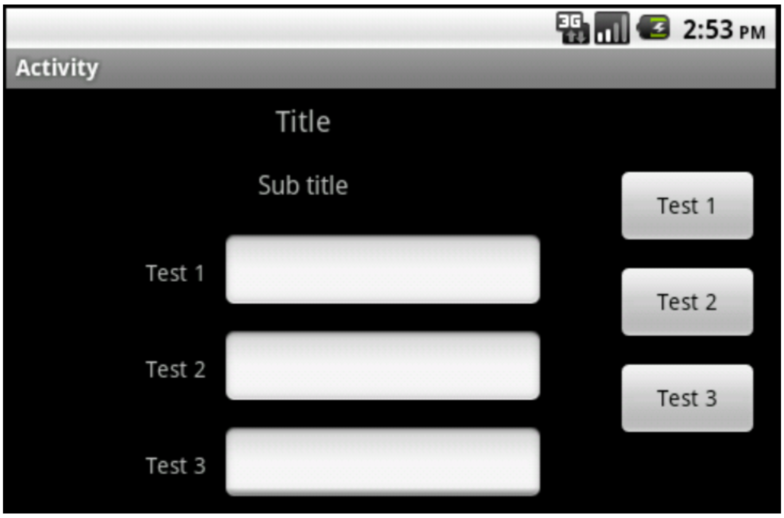


8.12.2 Same AutoScale example with portrait and landscape layout variants

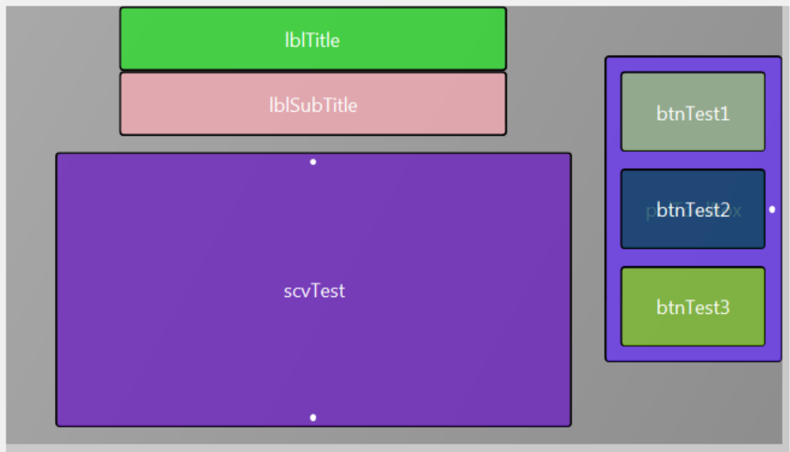
Source code AutoScaleExample2:



The previous example doesn't look good on smartphone screens with landscape orientation.



So we make a new layout variant for landscape where we move the ToolBox with the Buttons to the right side of the screen.



The layout variant in the Main layout file.
Note: Look at the anchors especially for the ToolBox and the ScrollView.

The code in the Designer Script must be changed:

For the portrait variant in the Main layout file we keep in the `All variants script` area only the code below:

```
'All variants script
AutoScaleRate(0.5)
AutoScaleAll
```

Setting the rate value and autoscaling all the views.

All the other code is moved to the `Variant specific script: 320x480,scale=1` area:

```
'Variant specific script: 320x480,scale=1

'Center the Labels horizonally to the middle of the screen
lblTitle.HorizontalCenter = 50%x
lblSubTitle.HorizontalCenter = 50%x

'Center the ToolBox Panel horizontally to the middle of the screen
pnlToolBox.HorizontalCenter = 50%x

'Center the ScrollView horizontally to the middle of the screen
scvTest.HorizontalCenter = 50%x
```

For the landscape variant we have in the `All variants script` area the same code as for the portrait variant:

```
'All variants script
AutoScaleRate(0.5)
AutoScaleAll
```

And in the `'Variant specific script: 480x320,scale=1` area:

We centre the Title and SubTitle Labels to the middle of the space between the left screen border and the left ToolBox boarder with:

```
lblTitle.HorizontalCenter = pnlToolBox.Left / 2
lblSubTitle.HorizontalCenter = pnlToolBox.Left / 2
```

We centre the ToolBox vertically to the middle of the screen height with:

```
pnlToolBox.VerticalCenter = 50%y
```

We set the right border of the ToolBox to right border of the screen with:

```
pnlToolBox.Right= 100%x
```

We set the Vertical Anchor property of the ScrollView to BOTH to fill the space between the bottom SubTitle Label border and the bottom screen border with.

And the whole code:

```
'Variant specific script: 480x320,scale=1

'Center the ToolBox Panel vertically
pnlToolBox.VerticalCenter = 50%y

'Center the Labels horizontally to the middle
'of the space between the left screen border
'and the left boarder of the ToolBox Panel
lblTitle.HorizontalCenter = pnlToolBox.Left / 2
lblSubTitle.HorizontalCenter = pnlToolBox.Left / 2

'Center the ScrollView horizontally to the middle
'of the space between the left screen border
'and the left ToolBox Panel border
scvTest.HorizontalCenter = pnlToolBox.Left / 2
```

For the Panel layout file:

The code for the portrait variant remains the same.

We add the same code for the landscape variant:

```
'All variants script
AutoScaleRate(0.5)
AutoScaleAll
```

Here too, no code in the `'Variant specific script: 480x320,scale=1` area.

8.12.3 AutoScale more advanced examples

This chapter is dedicated for more advanced users, the code is not explained here.
Source code AutoScaleExample4.

The AutoScale function in the Designer Scripts scales only views added in the Designer but not views added in the code.

To overcome this drawback I wrote the Scale Code module included in the example code.

There are two other drawbacks:

- the internal Labels of ListViews are not scaled.
- with the Designer Scripts AutoScale function for some smartphone screen sizes especially the 480 x 800 scale 1.5 screen the scaling is not optimal (at least for me).
  With AutoScale on a screen with a resolution of 480 x 800 scale 1.5 (the standard screen is 320 x 480 scale 1) the views are stretched too much horizontally and not enough vertically because of the different width/height ratio.

I added in the Scale Module a new set of equations with two scale factors one for X and one for Y.
For smartphone screens (< 6") the views are scaled according to the screen width and the screen height without the rate factor. For bigger screens the scale factors are modified with the rate factor.
For the big screens a rate value of 0 means no scaling and a value of 1 is equivalent to a scaling with %x and %y.

The AutoScale function in the code module scales also the internal views in ScrollViews, ListViews and scales the TextSize property of Spinners.

The *Scale* code module contains following functions:

Initialize	Calculates the scale factors
SetRate(Rate)	Sets a new Rate value and calculates the scale factors
ScaleView(View)	Scales the given view with its child views with the new equations.
ScaleViewDS(View)	Scales the given view with its child views with the Designer Scripts equations
ScaleAll(Activity, True)	Scales all views of the given Activity or Panel with the new equations.
ScaleAllDS(Activity, True)	Scales all views of the given Activity or Panel with the Designer Scripts equations
GetDevicePhysicalSize	Gets the approximate physical size of the device
GetScaleDS	Returns the Designer Scripts scale factor
GetScaleX	Returns the X scale factor
GetScaleX_L	Returns the X scale landscape factor independant of the current orientation
GetScaleX_P	Returns the X scale portrait factor independant of the current orientation
GetScaleY	Returns the Y scale factor
GetScaleY_L	Returns the Y scale landscape factor independant of the current orientation
GetScaleY_P	Returns the Y scale portrait factor independant of the current orientation
Bottom(View)	Returns the Bottom coordinate of the View

- Right(View) Returns the Right coordinate of the View
- HorizontalCenter(View, x1, x2) Centres the View horizontally the view between two coordinates
- HorizontalCenter2(V1, V2, V3) Centres the View V1 horizontally between two views V2 and V3
- VerticalCenter(View, x1, x2) Centres the View horizontally the view between two coordinates
- VerticalCenter2(V1, V2, V3) Centres the View V1 horizontally between two views V2 and V3
- IsActivity(View) Returns True if the View is an activity
- IsPanel(View) Returns True if the View is a Panel
- SetRight(View, xRight) Sets the Left property of the view according to the given right coordinate xRight and the views Width property.
- SetBottom(View, yBottom) Sets the Top property of the view according to the given bottom coordinate yBottom and the views Height property.
- SetLeftAndRight(View, xLeft, xRight) Sets the Left and Width properties of view View according to the xLeft and xRight coordinates.
- SetLeftAndRight2(V1, VL, dxL, VR, dxR) Sets the Left and Width properties of view V1 between the views VL and VR with the given spaces dxL and dxR.
- SetTopAndBottom(View, yTop, yBottom) Sets the Top and Height properties of view View according to the yTop and yBottom coordinates.
- SetTopAndBottom2(V1, VT, dyT, VB, dyB) Sets the Top and Height properties of view V1 between the views VT and VB with the given spaces dyT and dyB.

The module is part of the attached AutoScaleExample4 project.

The project contains following activities showing different examples of the use of either Designer Scripts AutoScale or scaling with the Code Module.
Activities:

- Main Main screen with an image and buttons.
- Setup The setup screen from the GPSExample program.
- About An about screen example.
- DBWebView A database table in a WebView with a modified DBUtils version scaling the table text size.
- DBScrollView A database table in a ScrollView.
- Keyboard A keyboard with views added in the code.
- ListView A ListView with the internal Labels and Bitmap scaled.
- Calculator A calculator layout from the RPNCalc project without the functions scaled with the new equations.
- Calculator1 Same as Calculator but scaled with the Designer Scripts equations.
- Positionning Example with the different positioning routines.

Code modules:

- Scale The scaling module
- DBUtils The modified DBUtils module

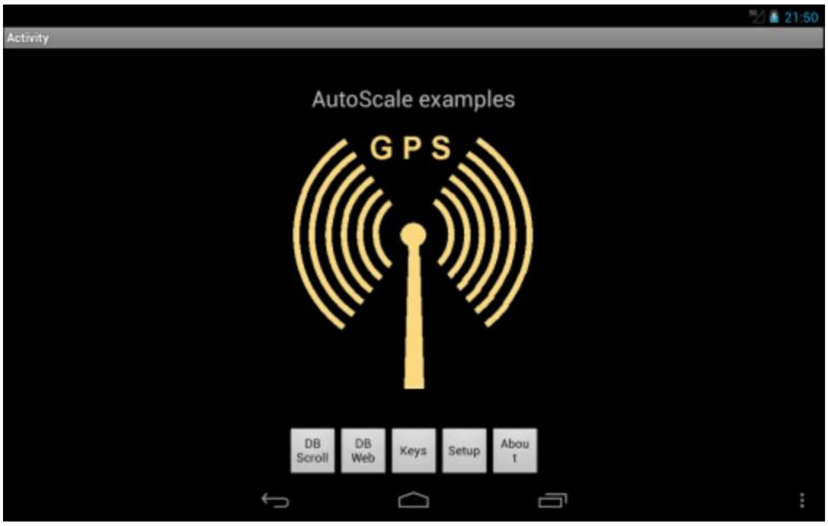
If you run Calculator and Calculator1 on a 480 x 800 scale 1.5 device you'll see the difference between Designer Scripts scaling and the scaling with the new equations.

If you don't need all routines in the Scale module you can remove those not needed

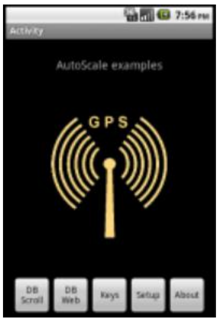
The Scale module scales also ScrollView2D views, if you don't use such a view you must comment the corresponding lines or remove them.
On the following pages you'll see some screenshots of following Emulators with Rate = 0.5:

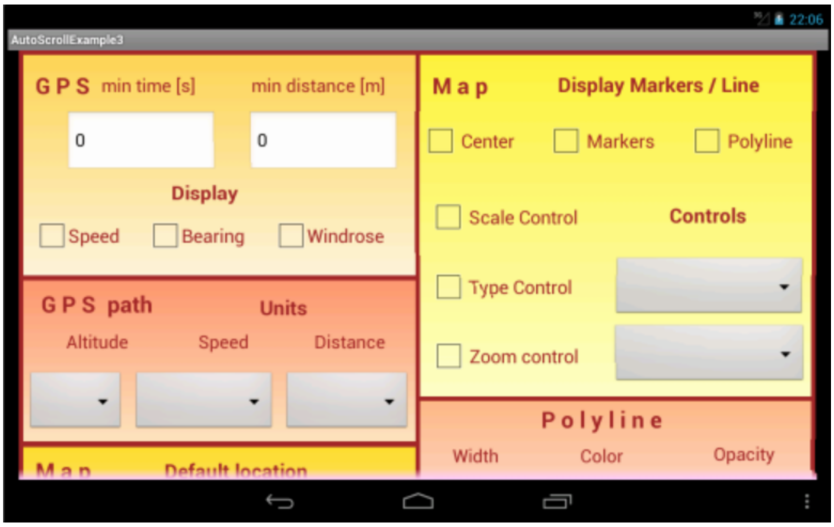
- 1280 x 800 10" tablet
- 480 x 320 3.5" smartphone

For some examples the smartphone screen is limited to portrait orientation only depending on the screen sizes and layouts.

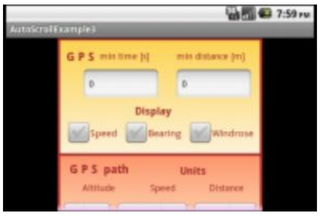


Main:

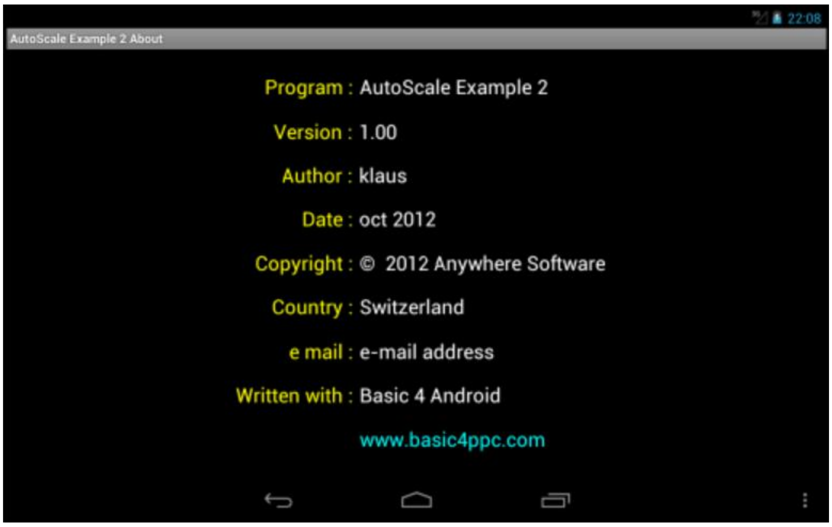




Setup:



About:



AutoScale Example 2 About

Program : AutoScale Example 2

Version : 1.00

Author : klaus

Date : oct 2012

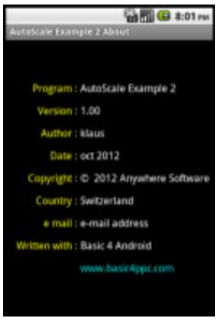
Copyright : © 2012 Anywhere Software

Country : Switzerland

e mail : e-mail address

Written with : Basic 4 Android

www.basic4ppc.com



AutoScale Example 2 About

Program : AutoScale Example 2

Version : 1.00

Author : klaus

Date : oct 2012

Copyright : © 2012 Anywhere Software

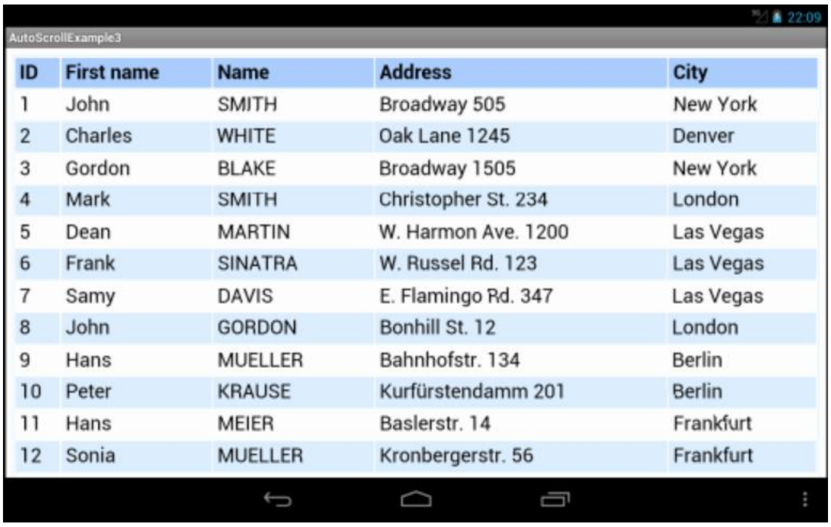
Country : Switzerland

e mail : e-mail address

Written with : Basic 4 Android

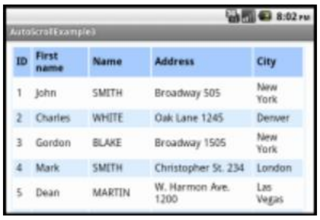
www.basic4ppc.com

DBWebView:



AutoScrollExample3

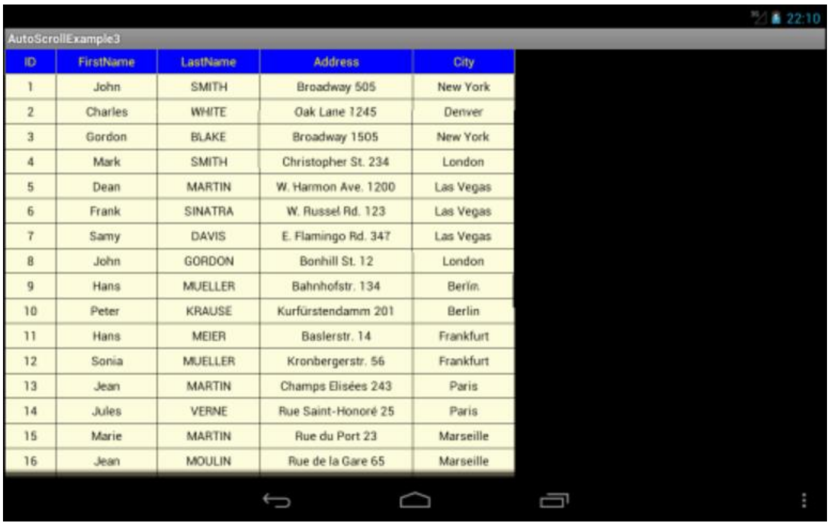
ID	First name	Name	Address	City
1	John	SMITH	Broadway 505	New York
2	Charles	WHITE	Oak Lane 1245	Denver
3	Gordon	BLAKE	Broadway 1505	New York
4	Mark	SMITH	Christopher St. 234	London
5	Dean	MARTIN	W. Harmon Ave. 1200	Las Vegas
6	Frank	SINATRA	W. Russel Rd. 123	Las Vegas
7	Samy	DAVIS	E. Flamingo Rd. 347	Las Vegas
8	John	GORDON	Bonhill St. 12	London
9	Hans	MUELLER	Bahnhofstr. 134	Berlin
10	Peter	KRAUSE	Kurfürstendamm 201	Berlin
11	Hans	MEIER	Baslerstr. 14	Frankfurt
12	Sonia	MUELLER	Kronbergerstr. 56	Frankfurt



AutoScrollExample3

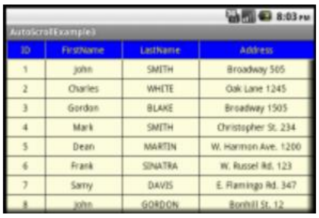
ID	First name	Name	Address	City
1	John	SMITH	Broadway 505	New York
2	Charles	WHITE	Oak Lane 1245	Denver
3	Gordon	BLAKE	Broadway 1505	New York
4	Mark	SMITH	Christopher St. 234	London
5	Dean	MARTIN	W. Harmon Ave. 1200	Las Vegas

DBScrollView:



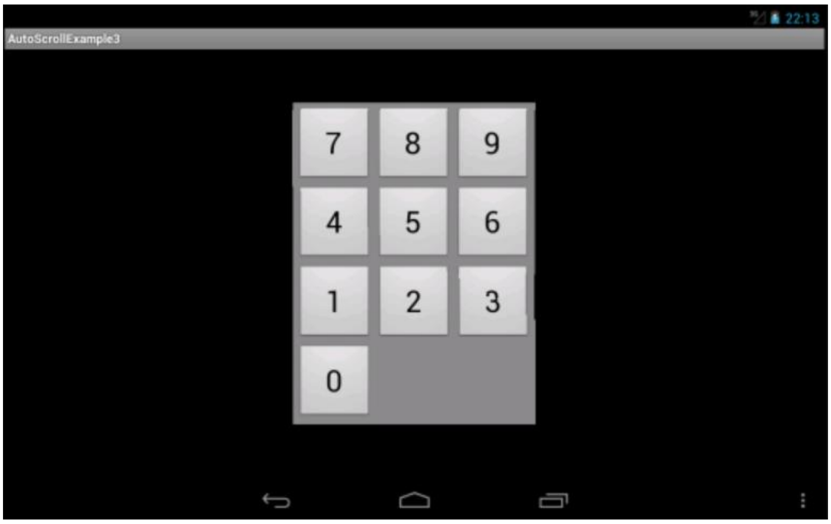
AutoScrollExample3

ID	FirstName	LastName	Address	City
1	John	SMITH	Broadway 505	New York
2	Charles	WHITE	Oak Lane 1245	Denver
3	Gordon	BLAKE	Broadway 1505	New York
4	Mark	SMITH	Christopher St. 234	London
5	Dean	MARTIN	W. Harmon Ave. 1200	Las Vegas
6	Frank	SINATRA	W. Russel Rd. 123	Las Vegas
7	Samy	DAVIS	E. Flamingo Rd. 347	Las Vegas
8	John	GORDON	Bonhill St. 12	London
9	Hans	MUELLER	Bahnhofstr. 134	Berlin
10	Peter	KRAUSE	Kurfürstendamm 201	Berlin
11	Hans	MEIER	Baslerstr. 14	Frankfurt
12	Sonia	MUELLER	Kronbergerstr. 56	Frankfurt
13	Jean	MARTIN	Champs Elisées 243	Paris
14	Jules	VERNE	Rue Saint-Honoré 25	Paris
15	Marie	MARTIN	Rue du Port 23	Marseille
16	Jean	MOULIN	Rue de la Gare 65	Marseille

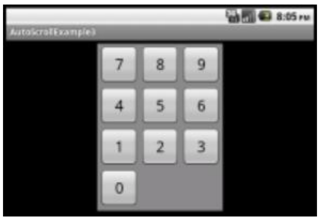


AutoScrollExample3

ID	FirstName	LastName	Address
1	John	SMITH	Broadway 505
2	Charles	WHITE	Oak Lane 1245
3	Gordon	BLAKE	Broadway 1505
4	Mark	SMITH	Christopher St. 234
5	Dean	MARTIN	W. Harmon Ave. 1200
6	Frank	SINATRA	W. Russel Rd. 123
7	Samy	DAVIS	E. Flamingo Rd. 347
8	John	GORDON	Bonhill St. 12

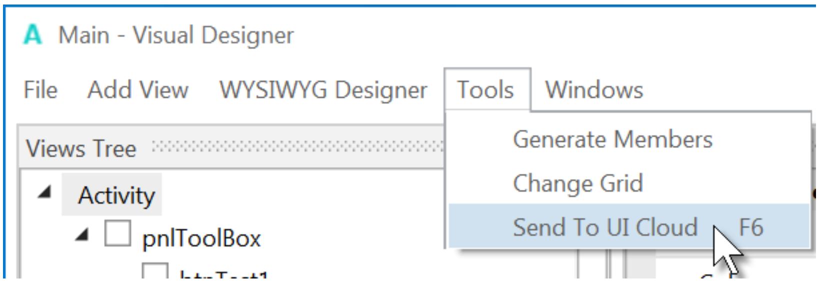

22:13
AutoScrollExample3
7
8
9
4
5
6
1
2
3
0

Keyboard:


8:05 PM
AutoScrollExample3
7
8
9
4
5
6
1
2
3
0

8.13 UI Cloud

With UI Cloud you can check how layouts look on different devices.



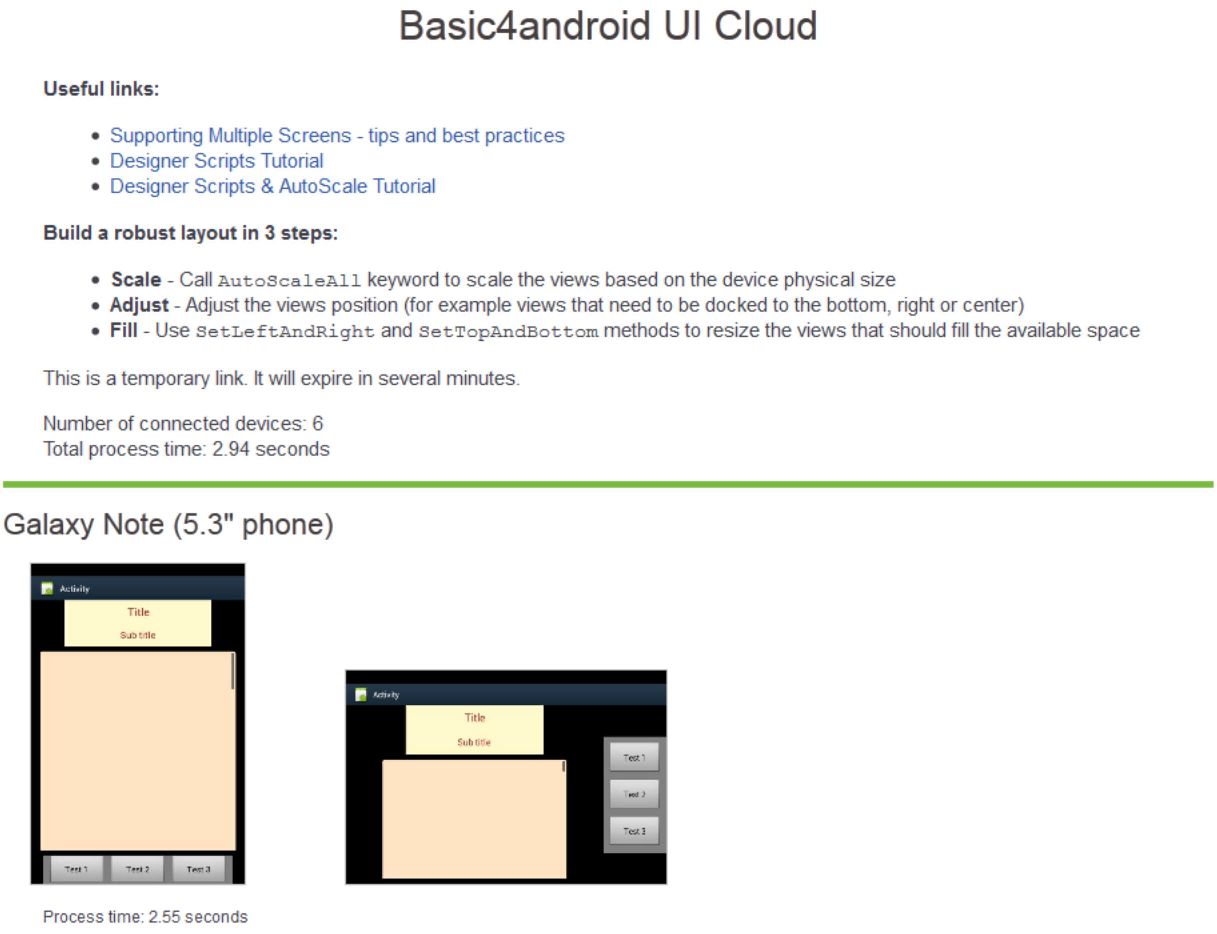
When you have defined a layout in the Designer Scripts you can send it to the UI Cloud in the tools menu.

The layout file is be sent to the B4A site and you get a page showing your layout on different devices with different screen resolutions and densities.

It's a very convenient tool to check the layout look without needing to have physical devices.

UI Cloud checks only layouts defined in the Designer, not layouts defined in the code !

Example of a UI Cloud screen:



Basic4android UI Cloud

Useful links:

- Supporting Multiple Screens - tips and best practices
- Designer Scripts Tutorial
- Designer Scripts & AutoScale Tutorial

Build a robust layout in 3 steps:

- **Scale** - Call `AutoScaleAll` keyword to scale the views based on the device physical size
- **Adjust** - Adjust the views position (for example views that need to be docked to the bottom, right or center)
- **Fill** - Use `SetLeftAndRight` and `SetTopAndBottom` methods to resize the views that should fill the available space

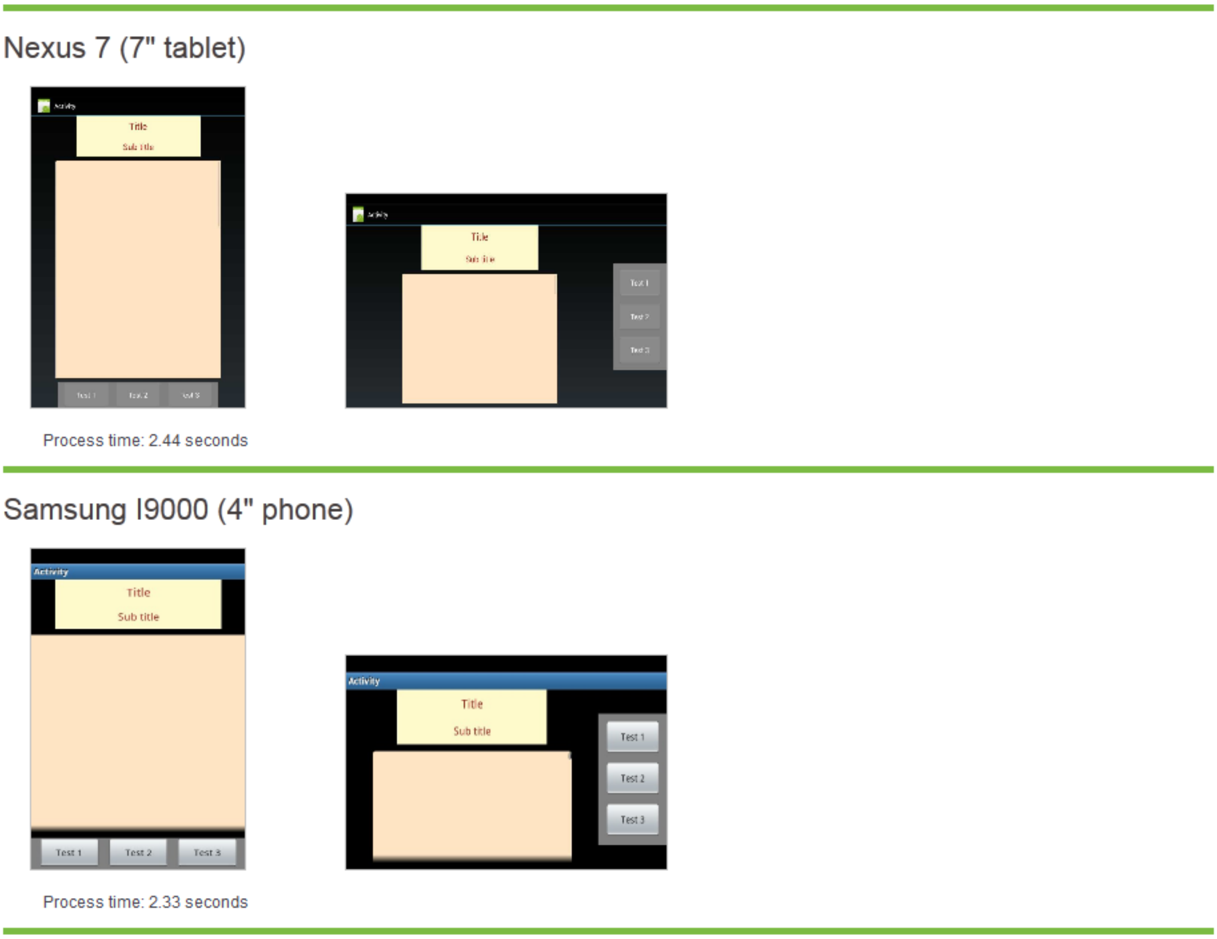
This is a temporary link. It will expire in several minutes.

Number of connected devices: 6
Total process time: 2.94 seconds

Galaxy Note (5.3" phone)

Process time: 2.55 seconds

Some other devices:



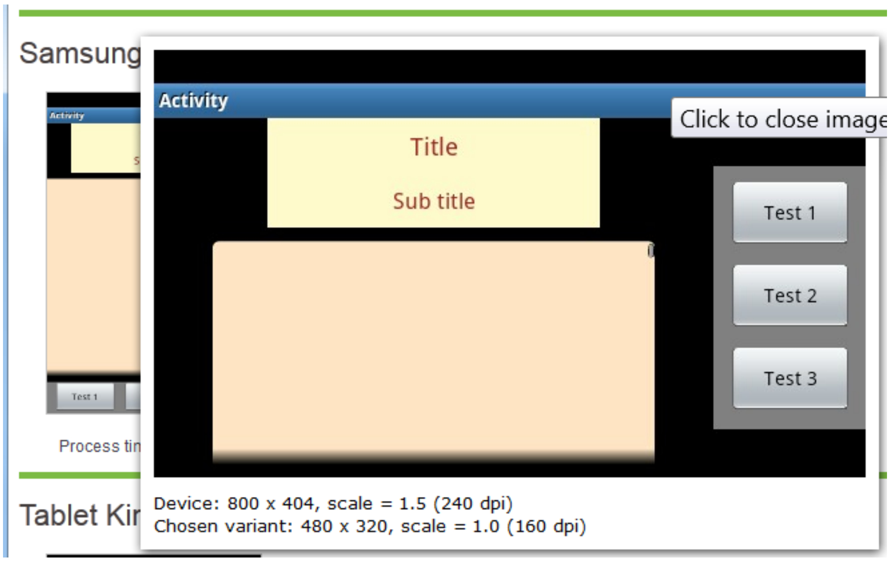
Nexus 7 (7" tablet)

Process time: 2.44 seconds

Samsung I9000 (4" phone)

Process time: 2.33 seconds

You can click on an image to show it in real size:



Click to close image

Device: 800 x 404, scale = 1.5 (240 dpi)
Chosen variant: 480 x 320, scale = 1.0 (160 dpi)

9 Process and Activity life cycle

Let's start simple:
Each B4A program runs in its own process.
A process has one main thread which is also named the UI thread which lives as long as the process lives. A process can also have more threads which are useful for background tasks.

A process starts when the user launches your application, assuming that it is not running already in the background.

The process end is less determinant. It will happen sometime after the user or system has closed all the activities.
If for example you have one activity and the user pressed on the back key, the activity gets closed. Later when the phone gets low on memory (and eventually it will happen) the process will quit.
If the user launches your program again and the process was not killed then the same process will be reused.

A B4A application is made of one or more activities.

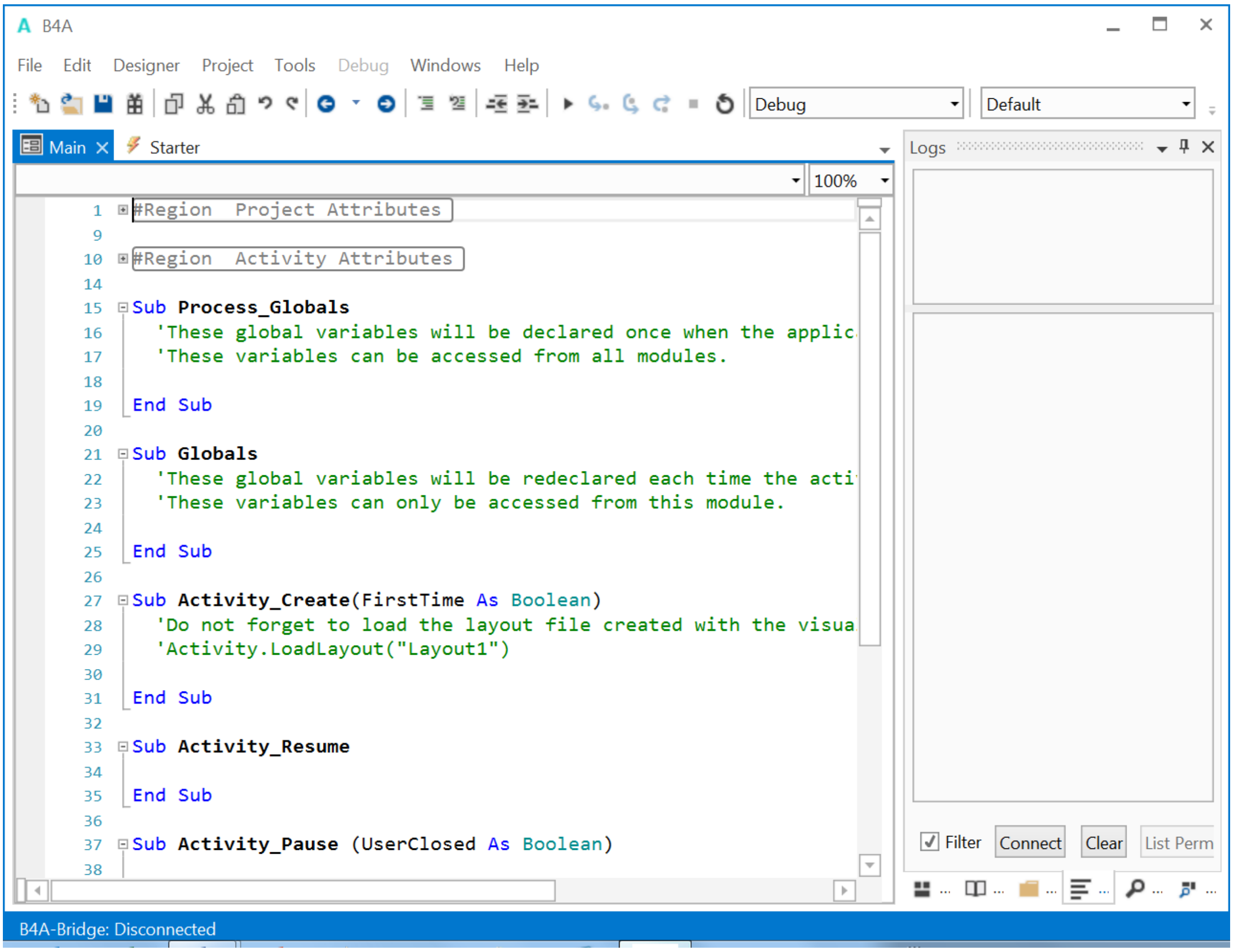
Activities are somewhat similar to Windows Forms.

One major difference is that, while an activity is not in the foreground it can be killed in order to preserve memory. Usually you will want to save the state of the activity before it gets lost. Either in a persistent storage or in memory that is associated with the process.
Later this activity will be recreated when needed.

Another delicate point happens when there is a major configuration change in the device. The most common is an orientation change (the user rotates the device). When such a change occurs the current activities are destroyed and then recreated. Now it is possible to create the activity according to the new configuration (for example, we now know the new screen dimensions).

9.1 Program Start

When we start a new program we get following template:



```
#Region  Project Attributes

#Region  Activity Attributes

Sub Process_Globals
    'These global variables will be declared once when the applic
    'These variables can be accessed from all modules.

End Sub

Sub Globals
    'These global variables will be redeclared each time the acti
    'These variables can only be accessed from this module.

End Sub

Sub Activity_Create(FirstTime As Boolean)
    'Do not forget to load the layout file created with the visua
    'Activity.LoadLayout("Layout1")

End Sub

Sub Activity_Resume

End Sub

Sub Activity_Pause (UserClosed As Boolean)

```

B4A-Bridge: Disconnected

On the top left we see two module Tabs :
Main Activity
Starter Service

The Starter Service is used to declare all ProcessGlobal variables and these variables are accessible from any module in the project.
The Main Activity is the starting activity, it cannot be removed.

Variables can be either global or local. Local variables are variables that are declared inside a sub other than Process_Globals or Globals.
Local variables are local to the containing sub or module. Once the sub ends, these variables no longer exist.
Global variables can be accessed from all subs in the containing module.

There are two types of global variables.
Process variables (accessible from all modules) and activity variables (accessible from a single module).

9.2 Process global variables

These variables live as long as the process lives.
You should declare these variables as Public inside Sub Process_Globals of the Starter Service like.

```
Sub Process_Globals
  'These global variables will be declared once when the application starts.
  'These variables can be accessed from all modules.
  Public MyVariable  = "Test" As String
```

This sub is called once when the process starts.
These variables are the only "public" variables. Which means that they can be accessed from other modules as well.

There is also a Process_Globals routines in each Activity module.
If you need variables, valid only in the Activity, which are initialized only once when the program is lauched you should put them in the Activity’s Process_Globals routine (this is true for all activities, not just the first activity).

However, not all types of objects can be declared as process variables.
All of the views for example cannot be declared as process variables.
The reason is that we do not want to hold a reference to objects that should be destroyed together with the activity.
In other words, when the activity is destroyed, all of the views that are contained in the activity are destroyed as well. If we didn't do this, and kept a reference to a view after the Activity was destroyed, the garbage collector would not be able to free the resource and we would have a memory leak.
The compiler enforces this requirement.

9.3 Activity variables

These variables are owned by the activity.
You should declare these variables inside Sub Globals.
These variables are "Private" and can only be accessed from the current activity module.
All object types can be declared as activity variables.
Every time the activity is created, Sub Globals is called (before Activity_Create).
These variables exist as long as the activity exists.

9.4 Starter service

One of the challenges that developers of any non-small Android app need to deal with, is the multiple possible entry points.

During development in almost all cases the application will start from the Main activity.
Many programs start with code similar to:

```
Sub Activity_Create (FirstTime As Boolean)
   If FirstTime Then
      SQL.Initialize(...)
      SomeBitmap = LoadBitmap(...)
      'additional code that loads application-wide resources
   End If
End Sub
```

Everything seems to work fine during development. However the app "strangely" crashes from time to time on the end user device.
The reason for those crashes is that the OS can start the process from a different activity or service.
For example if you use StartServiceAt and the OS kills the process while it is in the background.
Now the SQL object and the other resources will not be initialized.

Starting from B4A v5.20 there is a new feature named Starter service that provides a single and consistent entry point. If the Starter service exists then the process will always start from this service.

The Starter service will be created and started and only then the activity or service that were supposed to be started will start.
This means that the Starter service is the best place to initialize all the application-wide resources.
Other modules can safely access these resources.
The Starter service should be the default location for all the public process global variables. SQL objects, data read from files and bitmaps used by multiple activities should all be initialized in the Service_Create sub of the Starter service.

Notes

- The Starter service is identified by its name. You can add a new service named Starter to an existing project and it will be the program entry point.
  This is done by selecting Project > Add New Module > Service Module.
- This is an optional feature. You can remove the Starter service.
- You can call StopService(Me) in Service_Start if you don't want the service to keep on running. However this means that the service will not be able to handle events (for example you will not be able to use the asynchronous SQL methods).
- The starter service should be excluded from compiled libraries. Its #ExcludeFromLibrary attribute is set to True by default in the Service Attributes region.

9.5 Program flow

The program flow is the following:

- **Main Process_Globals** Process_Globals routines of the Main modules
  Here we declare all Private variables and objects for the Main module.

- **Starter Sevice Process_Globals** If the service exists, it is run.
  Here we declare all Public Process Global variables and objects like SQL, Bitmaps etc.

- **Other Activity Main Process_Globals** Process_Globals routines of other modules
  Here we declare all Private variables and objects for the given module.

- **Starter Service Service_Create** If the service exists, it is run.
  Here we initialize all Public Process Global variables and objects like SQL, Bitmaps etc.

- **Starter Sevice Service_Start** If the service exists, it is run.
  We can leave this routine empty.

- **Globals**
  Here we declare all Private variables for the given Activity.

- **Sub Activity_Create**
  Here we load layouts and initialize activity objects added by code

- **Activity_Resume**
  This routine is run every time the activity changes its state.

- **Activity_Pause**
  This routine is run when the Activity is paused, like orientation change, lauch of another activity etc.

You can ‘play’ with the program ProgramFlow in the SourceCode folder to see the program flow in different situations.

Look at the difference of ProcessGolbal and Global variables of the Activities.
Run the program.
You’ll see that:
MainPG = 2
MainG = 2
Press button Change values, the two variables have now the value of 3.
Change the orientation of the device.
Now you’ll see that the values are:
MainPG = 3
MainG = 2
Why ?
When you change the orientation the current Activity is destroyed and regenerated.
This means that Sub Globals is called and the variable MainG is reinitialized to the value of 2 !

The same happens of course with Activity 2.

9.6 Globals versus FirstTime

In any Activity, Process_Globals and Globals should be used to declare variables.
You can also set the values of "simple" variables (numeric, strings and booleans).

You should not put any other code there.
You should instead put the code in Activity_Create.

9.7 Sub Activity_Create (FirstTime As Boolean)

This sub is called when the activity is created.
The activity is created

- when the user first launches the application
- the device configuration has changed (user rotated the device) and the activity was destroyed
- when the activity was in the background and the OS decided to destroy it in order to free memory.

The primary purpose of this sub is to load or create the layout (among other uses).
The FirstTime parameter tells us if this is the first time that this activity is created. First time relates to the current process.
You can use FirstTime to run all kinds of initializations related to the process variables.
For example if you have a file with a list of values that you need to read, you can read it if FirstTime is True and store the list as a process variable by declaring the list in Sub Process_Globals
Now we know that this list will be available as long as the process lives and there is no need to reload it even when the activity is recreated.

To summarize, you can test whether FirstTime is True and then initialize the process variables that are declared in the Activity’s Sub Process_Globals.

9.8 Variable declaration summary

Which variable should we declare where and where do we initialize our variables:

- Variables and none user interface objects you want to access from several modules.
  Like SQL, Maps, Lists, Bitmaps etc.
  These must be declared as Public in Starter Process_Globals like:

```
Sub Process_Globals
   Public SQL1 As SQL
   Public Origin = 0 As Int
   Public MyBitmap As Bitmap
End Sub
```

  And initialized in Starter Service_Create like:

```
Sub Service_Create
   SQL1.Initialize(...)
   MyBitmap.Initialize(...)
End Sub
```

- Variables accessible from all Subs in an Activity which should be initialized only once.
  These must be declared as Private in Activity Process_Globals like:

```
Sub Process_Globals
   Private MyList As List
   Private MyMap As Int
End Sub
```

  And initialized in Activty_Create like:

```
Sub Activity_Create
   MyList.Initialize
   MyMap.Initialize
End Sub
```

- Variables in a Class or Code module
  These are mostly declared as Private, you can declare them as Public if you want them being accessible from outsides the Class.
  Class mudules are explained in detail in the User's Guide.

- User interface objects
  These must be declared in the Activity module where they are used in Globals like:

```
Sub Globals
   Private btnGoToAct2, btnChangeValues As Button
   Private lblStarterPG, lblMainPG, lblMainG  As Label
End Sub
```

Simple variables like Int, Double String and Boolean can be initialized directly in the declaration line, even in Process_Globals routines.
Example:

```
Public Origin = 0 as Int
```

No code should be written in Process_Globals routines !

9.9 Sub Activity_Resume
Sub Activity_Pause (UserClosed As Boolean)

Activity_Resume is called right after Activity_Create finishes or after resuming a paused activity (activity moved to the background and now it returns to the foreground).
Note that when you open a different activity (by calling StartActivity), the current activity is first paused and then the other activity will be created if needed and (always) resumed.

Each time the activity moves from the foreground to the background Activity_Pause is called.
Activity_Pause is also called when the activity is in the foreground and a configuration change occurs (which leads to the activity getting paused and then destroyed).
Activity_Pause is the last place to save important information.
Generally there are two types of mechanisms that allow you to save the activity state.
Information that is only relevant to the current application instance can be stored in one or more process variables.
Other information should be stored in a persistent storage (file or database).
For example, if the user changed some settings you should save the changes to a persistent storage at this point. Otherwise the changes may be lost.

Activity_Pause is called every time the activity moves from the foreground to the background. This can happen because:

1. A different activity was started.
2. The Home button was pressed.
3. A configuration changed event was raised (orientation changed for example).
4. The Back button was pressed.

In scenarios 1 and 2, the activity will be paused and for now kept in memory as it is expected to be reused later.

In scenario 3 the activity will be paused, destroyed and then created (and resumed) again.

In scenario 4 the activity will be paused and destroyed. **Pressing on the Back button is similar to closing the activity**. In this case you do **not** need to save any instance specific information (the position of pacman in a PacMan game for example).

The UserClosed parameter will be true in this scenario and false in all other. Note that it will also be true when you call Activity.Finish. This method pauses and destroys the current activity, similar to the Back button.

You can use UserClosed parameter to decide which data to save and also whether to reset any related process variables to their initial state (move pacman position to the center if the position is a process variable).

9.10 Activity.Finish / ExitApplication

Some explanations on how and when to use Activity.Finish and ExitApplication.

An interesting article about the functioning of Android can be found here: Multitasking the Android way.

Most applications should not use ExitApplication but prefer Activity.Finish which lets the OS decide when the process is killed.
You should use it only if you really need to fully kill the process.

When should we use Activity.Finish and when not ?
Let us consider following example without any Activity.Finish:

- **Main activity**
  - StartActivity(SecondActivity)
- **SecondActivity activity**
  - StartActivity(ThirdActivity)
- **ThirdActivity activity**
  - Click on Back button
  - The OS goes back to previous activity, SecondActivity
- **SecondActivity activity**
  - Click on Back button
  - The OS goes back to previous activity, Main
- **Main activity**
  - Click on Back button
  - The OS leaves the program

Let us now consider following example with Activity.Finish before each StartActivity:

- **Main activity**
  - Activity.Finish
  - StartActivity(SecondActivity)
- **SecondActivity activity**
  - Activity.Finish
  - StartActivity(ThirdActivity)
- **ThirdActivity activity**
  - Click on Back button
  - The OS leaves the program

We should use Activity.Finish before starting another activity only if we don't want to go back to this activity with the Back button.

10 Variables and objects

A **variable** is a symbolic name given to some known or unknown quantity or information, for the purpose of allowing the name to be used independently of the information it represents. A variable name in computer source code usually associated with a data storage location and thus also its contents, and these may change during the course of program execution (source Wikipedia).

B4A type system is derived directly from Java type system.
There are two types of variables: primitives and non-primitives types.
Primitives include the numeric types: Byte, Short, Int, Long, Float and Double.
Primitives also include: Boolean and Char.

10.1 Variable Types

List of types with their ranges:

B4A	Type	min value	max value
Boolean	boolean	False	True
Byte	integer 8 bits	- 2^{7} -128	2^{7} - 1 127
Short	integer 16 bits	- 2^{15} - 32768	2^{15} -1 32767
Int	integer 32 bits	- 2^{31} -2147483648	2^{31} -1 2147483647
Long	long integer 64 bits	- 2^{63} -9223372036854775808	2^{63} -1 9223372036854775807
Float	floating point number 32 bits	- 2^{-149} 1.4E-45	$(2 - 2^{-23}) * 2^{127}$ 3.4028235 E 38
Double	double precision number 64 bits	- 2^{-1074} 2.2250738585072014 E - 308	$(2 - 2^{-52}) * 2^{1023}$ 1.7976931348623157 E 308
Char	character		
String	array of characters		

Primitive types are always passed by value to other subs or when assigned to other variables.
For example:

```
Sub S1
  Private A As Int
  A = 12                 The variable A = 12
  S2(A)        It's passed by value to routine S2
  Log(A)  ' Prints 12    Variable A still equals 12, even though B was changed in routine S2.
End Sub

Sub S2(B As Int)         Variable B = 12
  B = 45                 Its value is changed to B = 45
End Sub
```

All other types, including arrays of primitive types and strings are categorized as non-primitive types.
When you pass a non-primitive to a sub or when you assign it to a different variable, a copy of the reference is passed.
This means that the data itself isn't duplicated.
It is slightly different than passing by reference as you cannot change the reference of the original variable.

All types can be treated as Objects.
Collections like lists and maps work with Objects and therefore can store any value.
Here is an example of a common mistake, where the developer tries to add several arrays to a list:

```
Private arr(3) As Int
Private List1 As List
List1.Initialize
For i = 1 To 5
   arr(0) = i * 2
   arr(1) = i * 2
   arr(2) = i * 2
   List1.Add(arr)    'Add the whole array as a single item
Next
arr = List1.Get(0) 'get the first item from the list
Log(arr(0))          'What will be printed here???
```

You may expect it to print 2. However it will print 10.
We have created a single array and added 5 references of this array to the list.
The values in the single array are the values set in the last iteration.
To fix this we need to create a new array each iteration.
This is done by calling Private each iteration:

```
Private arr(3) As Int 'This call is redundant in this case.
Private List1 As List
List1.Initialize
For i = 1 To 5
   Private arr(3) As Int
   arr(0) = i * 2
   arr(1) = i * 2
   arr(2) = i * 2
   List1.Add(arr) 'Add the whole array as a single item
Next
arr = List1.Get(0) 'get the first item from the list
Log(arr(0)) 'Will print 2
```

Tip: You can use agraham's CollectionsExtra library to copy an array.

10.2 Names of variables

It is up to you to give any name to a variable, except reserved words.
A variable name must begin with a letter and must be composed by the following characters A-Z, a-z, 0-9, and underscore "_", no spaces, no brackets etc.
Variable names are case insensitive, that means that Index and index refer to the same variable.

But it is good practice to give them meaningful names.
Example:

Interest = Capital * Rate / 100	is meaningful
n1 = n2 * n3 / 100	not meaningful

For Views it is useful to add to the name a three character prefix that defines its type.
Examples:

`lblCapital`	lbl > Label	Capital > purpose
`edtInterest`	edt > EditText	Interest > purpose
`btnNext`	btn > Button	Next > purpose

10.3 Declaring variables

10.3.1 Simple variables

Variables are declared with the `Private` or the `Public` keyword followed by the variable name and the `As` keyword and followed by the variable type. For details look at chapter 10.5 Scope.
There exist the `Dim` keyword, this is maintained for compatibility.

Examples:

```
Private Capital As Double
Private Interest As Double
Private Rate As Double
```

Declares three variables as Double, double precision numbers.

```
Private i As Int
Private j As Int
Private k As Int
```

Declares three variables as Int, integer numbers.

```
Private edtCapital As EditText
Private edtInterest As EditText
Private edtRate As EditText
```

Declares three variables as EditText views.

```
Private btnNext As Button
Private btnPrev As Button
```

Declares two variables as Button views.

The same variables can also be declared in a short way.

```
Private Capital, Interest, Rate As Double
Private i, j, k As Int
Private edtCapital, edtInterest, edtRate As EditText
Private btnNext, btnPrev As Button
```

The names of the variables separated by commas and followed by the type declaration.

Following variable declarations are valid:

```
Private i = 0, j = 2, k = 5 As Int

Private txt = "test" As String, value = 1.05 As Double, flag = False As Boolean
```

View names must be declared if we want to use them in the code.
For example, if we want to change the text in an EditText view in the code, like
`edtCapital.Text = "1200"`,
we need to reference this EditText view by its name `edtCapital`, this is done with the `Private` declaration.
If we never make any reference to this EditText view anywhere in the code no declaration is needed.
Using an event routine for that view doesn't need a declaration either.

To allocate a value to a variable write its name followed by the equal sign and followed by the value, like:

```
Capital = 1200
LastName = "SMITH"
```

Note that for `Capital` we wrote just `1200` because `Capital` is a number.
But for `LastName` we wrote `"SMITH"` because `LastName` is a string.
Strings must always be written between double quotes.

10.3.2 Array variables

Arrays are collections of data or objects that can be selected by indices. Arrays can have multiple dimensions.
The declaration contains the `Private` or the `Public` keyword followed by the variable name `LastName`, the number of items between brackets `(50)`, the keyword `As` and the variable type `String`.
For details look at chapter 10.5 Scope. There exist the `Dim` keyword, this is maintained for compatibility.

Examples:

`Public LastName(50) As String`	One dimension array of strings, total number of items 50.
`Public Matrix(3, 3) As Double`	Two dimensions array of Doubles, total number of items 9.
`Public Data(3, 5, 10) As Int`	Three dimensions array of integers, total number of items 150.

The first index of each dimension in an array is 0.
`LastName(0), Matrix(0,0), Data(0,0,0)`

The last index is equal to the number of items in each dimension minus 1.
`LastName(49), Matrix(2,2), Data(2,4,9)`

```
Public LastName(10) As String
Public FirstName(10) As String
Public Address(10) As String
Public City(10) As String
```

or

```
Public LastName(10), FirstName(10), Address(10), City(10) As String
```

This example shows how to access all items in a three dimensional array.

```
For i = 0 To 2
   For j = 0 To 2
      For k = 0 To 2
         Data(i, j, k) = ...
      Next
   Next
Next
```

A more versatile way to declare arrays is to use variables.

```
Public NbPers = 10 As Int
Public LastName(NbPers) As String
Public FirstName(NbPers) As String
Public Address(NbPers) As String
Public City(NbPers) As String
```

We declare the variable `Public NbPers = 10 As Int` and set its value to 10.
Then we declare the arrays with this variable instead of the number 10 as before.
The big advantage is if at some point we need to change the number of items, we change only ONE value.

For the Data array we could use the following code.

```
Public NbX = 2 As Int
Public NbY = 5 As Int
Public NbZ = 10 As Int
Public Data(NbX, NbY, NbZ) As Int
```

And the access routine.

```
For i = 0 To NbX - 1
   For j = 0 To NbY -
      For k = 0 To NbZ - 1
         Data(i, j, k) = ...
      Next
   Next
Next
```

Filling an array with the Array keyword :

```
Public Name() As String
Name = Array As String("Miller", "Smith", "Johnson", "Jordan")
```

10.3.3 Array of views (objects)

Views or objects can also be in an Array. The following code shows an example:

In the example below the Buttons are added to the Activity by code.

```
Sub Globals
   Private Buttons() As Button
End Sub

Sub Activity_Create(FirstTime As Boolean)
   Private i As Int

   For i = 0 To 6
      Buttons(i).Initialize("Buttons")
      Activity.AddView(Buttons(i), 10dip, 10dip + i * 60dip, 150dip, 50dip)
      Buttons(i).Tag = i + 1
      Buttons(i).Text = "Test " & (i + 1)
   Next
End Sub

Sub Buttons_Click
   Private btn As Button

   btn = Sender

   Activity.Title = "Button " & btn.Tag & " clicked"
End Sub
```

The Buttons could also have been added in a layout file, in that case they must neither be initialized, nor added to the Activity and the Text and Tag properties should also be set in the Designer.
In that case the code would look like this:

```
Sub Globals
   Private b1, b2, b3, b4, b5, b6, b7 As Button
   Private Buttons() As Button
End Sub

Sub Activity_Create(FirstTime As Boolean)
   Private i As Int

   Buttons = Array As Button(b1, b2, b3, b4, b5, b6, b7)
End Sub

Sub Buttons_Click
   Private btn As Button

   btn = Sender

   Activity.Title = "Button " & btn.Tag & " clicked"
End Sub
```

10.3.4 Type variables

A Type cannot be private. Once declared it is available everywhere (similar to Class modules).
The best place to declare them is in the Process_Globals routine in the Main module.

Let us reuse the example with the data of a person.
Instead of declaring each parameter separately, we can define a personal type variable with the Type keyword:

```
Public NbUsers = 10 As Int
Type Person(LastName As String, FirstName As String. Address As String, City As String)
Public User(NbUsers) As Person
Public CurrentUser As Person
```

The new personal type is Person , then we declare either single variables or arrays of this personal type.
To access a particular item use following code.

```
CurrentUser.FirstName
CurrentUser.LastName

User(1).LastName
User(1).FirstName
```

The variable name, followed by a dot and the desired parameter.
If the variable is an array then the name is followed by the desired index between brackets.

It is possible to assign a typed variable to another variable of the same type, as shown below.

```
CurrentUser = User(1)
```

10.4 Casting

B4A casts types automatically as needed. It also converts numbers to strings and vice versa automatically.
In many cases you need to explicitly cast an Object to a specific type.
This can be done by assigning the Object to a variable of the required type.
For example, Sender keyword references an Object which is the object that raised the event.
The following code changes the color of the pressed button.
Note that there are multiple buttons that share the same event sub.

```
Sub Globals
   Private Btn1, Btn2, Btn3 As Button
End Sub

Sub Activity_Create(FirstTime As Boolean)
   Btn1.Initialize("Btn")
   Btn2.Initialize("Btn")
   Btn3.Initialize("Btn")
   Activity.AddView(Btn1, 10dip, 10dip, 200dip, 50dip)
   Activity.AddView(Btn2, 10dip, 70dip, 200dip, 50dip)
   Activity.AddView(Btn3, 10dip, 130dip, 200dip, 50dip)
End Sub

Sub Btn_Click
   Private btn As Button
   btn = Sender     ' Cast the Object to Button
   btn.Color = Colors.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255))
End Sub
```

The above code could also be written more elegantly:

```
Sub Globals

End Sub

Sub Activity_Create(FirstTime As Boolean)
   Private i As Int
   For i = 0 To 9   ' create 10 Buttons
      Private Btn As Button
      Btn.Initialize("Btn")
      Activity.AddView(Btn, 10dip, 10dip + 60dip * i, 200dip, 50dip)
   Next
End Sub

Sub Btn_Click
   Private btn As Button
   btn = Sender     ' Cast the Object to Button
   btn.Color = Colors.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255))
End Sub
```

10.5 Scope

10.5.1 Process variables

These variables live as long as the process lives.
You should declare these variables inside Sub Process_Globals.
This sub is called once when the process starts (this is true for all activities, not just the first activity).
These variables are the only "public" variables. Which means that they can be accessed from other modules as well.
However, not all types of objects can be declared as process variables.
For example, views cannot be declared as process variables.
The reason is that we do not want to hold a reference to objects that should be destroyed together with the activity.
In other words, once the activity is being destroyed, all of the views which are contained in the activity are being destroyed as well.
If we hold a reference to a view, the garbage collector would not be able to free the resource and we will have a memory leak. The compiler enforces this requirement.

To access process global variables in other modules than the module where they were declared their names must have the module name they were declared as a prefix.
Example:
Variable defined in a module with the name : *MyModule*

```
Sub Process_Globals
  Public MyVar As String
End Sub
```

Accessing the variable in *MyModule* module:

```
  MyVar = "Text"
```

Accessing the variable in any other module:

```
  MyModule.MyVar = "Text"
```

Variables can be declared with:

```
Dim MyVar As String
```

In this case the variable is public same as Public.

It is good practice to declare the variables like this:

```
Public MyVar As String
```

This variable is public.

It is possible to declare private variables in Sub Process_Globals like this:

```
Private MyVar As String
```

The variable is private to the activity or the module where it is declared.
For Activities it is better to declare them in Sub Globals.

For variables declared in Class modules in Sub Class_Globals the same rules as above are valid.

```
  Public MyVarPublic As String        ' public
  Private MyVarPublic As String       ' private
  Dim MyVar As String                 ' public like Public
```

Using Dim in Sub Class_Globals is not recommended !

10.5.2 Activity variables

These variables are contained by the activity.
You should declare these variables inside Sub Globals.
These variables are "private" and can only be accessed from the current activity module.
All object types can be declared as activity variables.
Every time the activity is created, Sub Globals is called (before Activity_Create).
These variables exist as long as the activity exists.

10.5.3 Local variables

Variables declared in a subroutine are local to this subroutine.
They are "private" and can only be accessed from within the subroutine where they were declared.
All objects types can be declared as local variables.
At each call of the subroutine the local variables are initialized to their default value or to any other value you have defined in the code and are 'destroyed' when the subroutine is exited.

10.6 Tips

A view can be assigned to a variable so you can easily change the common properties of the view.

For example, the following code disables all views that are direct children of the activity:

```
For i = 0 To Activity.NumberOfViews - 1
   Private v As View
   v = Activity.GetView(i)
   v.Enabled = False
Next
```

If we only want to disable buttons:

```
For i = 0 To Activity.NumberOfViews - 1
   Private v As View
   v = Activity.GetView(i)
   If v Is Button Then ' check whether it is a Button
      v.Enabled = False
   End If
Next
```

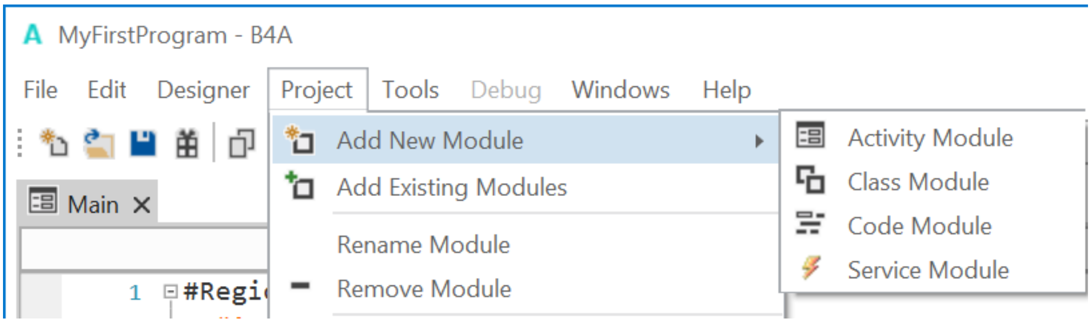
11 Modules

At least one module exists, the main one.
Its name is always **Main** and cannot be changed.

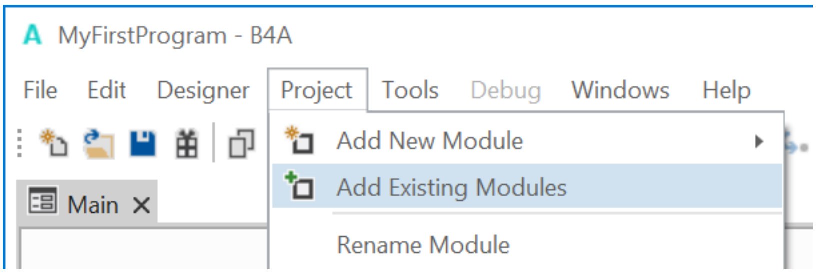
There do exist four different types of modules:

- Activity modules
- Class modules
- Code modules
- Service modules

To add a new module click on either Activity, Class, Code or Service Module in the IDE menu Project / Add New Module.



To add an existing module click on Add Existing Module in the IDE menu Project.

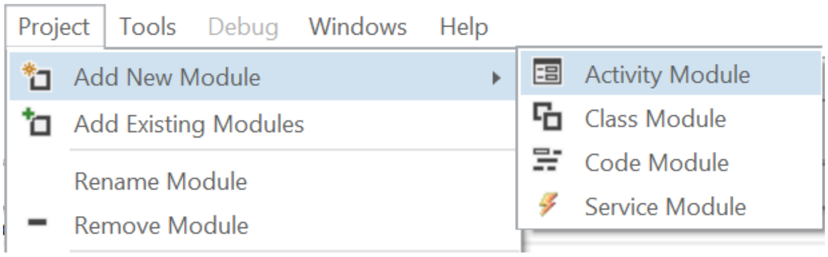


11.1 Activity modules

Each Activity has its own module. For a better knowledge of Activity life cycle have a look at the Process and Activity life cycle chapter.

You can add either an existing module or a new module.

To add a new Activity module click on:



The example is explained in detail in the chapter: Program with 3 Activities.

To access any object or variable in a module other than the module where they were declared you must add the module name as a prefix to the object or variable name separated by a dot.

Examples from the ThreeActivityExample program:
Variables Value1 and Value2 are declared in Main module in Sub Process_Globals.

```
Sub Process_Globals
   Public Value1, Value2, Value3 As String
End Sub
```

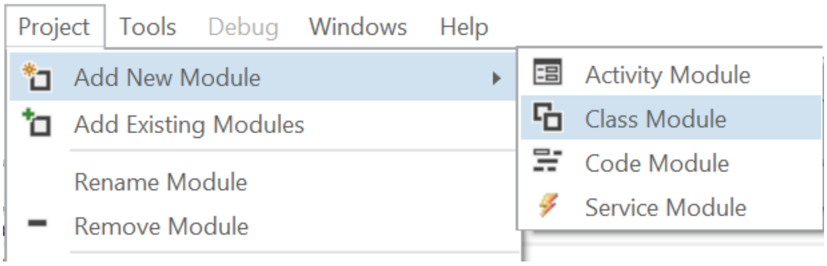
To access these variables from another module the name is Main.Value1 or Main.Value2.

```
Sub Activity_Pause (UserClosed As Boolean)
   Main.Value2 = edtValue2_P2.Text      ' Sets edtValue_P2.Text to the
End Sub                                 ' Process Global variable Value2
```

It is NOT possible to access any view from another activity module, because when a new activity is started the current activity is paused and it’s no longer accessible !

11.2 Class modules

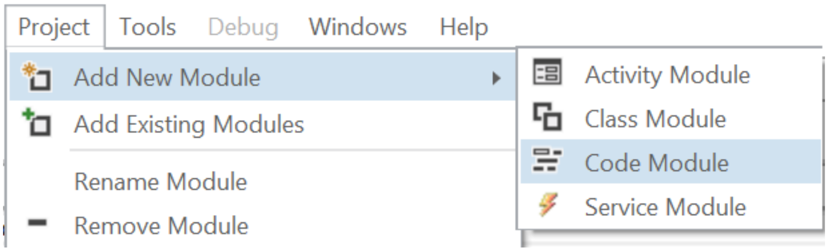
Class modules are explained in detail in the User's Guide.



11.3 Code modules

Code modules contain code only. No activity is allowed in Code modules.
The purpose and advantage of code modules is sharing same code in different programs, mainly for calculations or other general management.
Some code modules, called utilities, are already published by Erel in the forum:

- DBUtils, Working with Android databases have been greatly simplified with the use of these database management utilities.
  The DBUtils module is explained in the User's Guide.
- HttpUtils, Android web services are now simple.
- StateManager, helps managing Android application settings and state.



11.4 Service modules

Service modules play an important role in the application and process life cycle.
Start with this tutorial if you haven't read it before: Android Process and activities life cycle
Code written in an activity module is paused once the activity is not visible.
So by only using activities it is not possible to run any code while your application is not visible.
Services life cycle is (almost) not affected by the current visible activity. This allows you to run tasks in the background.
Services usually use the status bar notifications to interact with the user. Services do not have any other visible elements. Services also cannot show any dialog (except of toast messages).
Note that when an error occurs in a service code you will not see the "Do you want to continue?" dialog. Android's regular "Process has crashed" message will appear instead.

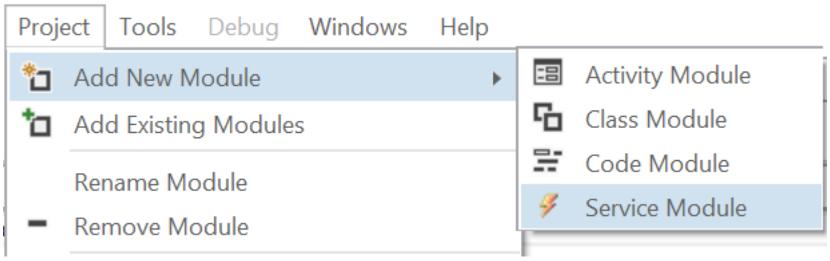
Before delving into the details I would like to say that **using services is simpler than it may first sound. In fact for many tasks it is easier to work with a service instead of an activity** as a service is not paused and resumed all the time and services are not recreated when the user rotates the screen. There is nothing special with code written in service.
Code in a service module runs in the same process and the same thread as all other code.

It is important to understand how Android chooses which process to kill when it is low on memory (a new process will later be created as needed).
A process can be in one of the three following states:
- Foreground - The user currently sees one of the process activities.
- Background - None of the activities of the process are visible, however there is a started service.
- Paused - There are no visible activities and no started services.

Paused processes are the first to be killed when needed. If there is still not enough memory, background processes will be killed.
Foreground processes will usually not be killed.

As you will soon see a service can also bring a process to the foreground.

Adding a service module is done by choosing Project - Add New Module - Service Module.



The template for new services is:

```
Sub Process_Globals

End Sub

Sub Service_Create

End Sub

Sub Service_Start (StartingIntent As Intent)

End Sub

Sub Service_Destroy

End Sub
```

Sub Process_Globals is the place to declare the service global variables. A service module does not have a Globals sub because it does not support Activity objects.
Sub process globals should only be used to declare variables. It should not run any other code as it might fail. This is true for other modules as well.
Note that Process_Global variables are kept as long as the process runs and are accessible from other modules.

Sub Service_Create is called when the service is first started. This is the place to initialize and set the process global variables. Once a service is started it stays alive until you call StopService or until the whole process is destroyed.

Sub Service_Start is called **each time** you call StartService (or StartServiceAt). When this subs runs the process is moved to the foreground state. Which means that the OS will not kill your process until this sub finishes running. If you want to run some code every couple of minutes / hours you should schedule the next task with StartServiceAt inside this sub.

Sub Service_Destroy is called when you call StopService. The service will not be running after this sub until you call StartService again (which will run Sub Service_Create followed by Sub Service_Start).

Service use cases

As I see it, there are four main reasons to use services.
- Separating UI code with "business" or logic code. Writing the non-UI code in a service is easier than implementing it inside an Activity module because the service is not paused, resumed or (usually) recreated like an Activity.
You can call StartService during Activity_Create and from now on work with the service module.
A good design is to make the activity fetch the required data from the service in Sub Activity_Resume. The activity can fetch data stored in a process global variable or it can call a service Sub with CallSub method.

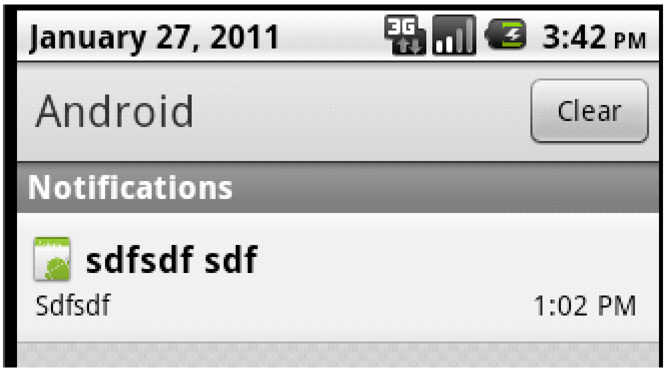
- Running a long operation. For example downloading a large file from the internet. In this case you can call Service.StartForeground (from the service module). This will move your activity to the foreground state and will make sure that the OS doesn't kill it. Make sure to eventually call Service.StopForeground.

- Scheduling a repeating task. By calling StartServiceAt you can schedule your service to run at a specific time. You can call StartServiceAt in Sub Service_Start to schedule the next time and create a repeating task (for example a task that checks for updates every couple of minutes).

- Run a service after boot. Set `#StartAtBoot: True` in the `#Region  Service Attributes` and your service will run after boot is completed.

Notifications

Status bar notifications can be displayed by activities and services.
Usually services use notifications to interact with the user. The notification displays an icon in the status bar. When the user pulls the status bar they see the notification message.



Example of a notification (using the default icon):

The user can press on the message, which will open an activity as configured by the Notification object.

The notification icon is an image file which you should manually put in the following folder:
<project folder>\Object\res\drawable.

Accessing other modules

Process global objects are public and can be accessed from other modules.
Using CallSub method you can also call a sub in a different module.
It is however limited to non-paused modules. This means that one activity can never access a sub of a different activity as there could only be one running activity.
However an activity can access a running service and a service can access a running activity.
Note that if the target component is paused then an empty string returns.
No exception is thrown.
You can use IsPause to check if the target module is paused.

For example if a service has downloaded some new information it can call:

If the Main activity is running it will fetch the data from the service process global variables and will update the display.
It is also possible to pass the new information to the activity sub. However it is better to keep the information as a process global variable. This allows the activity to call RefreshData whenever it want and fetch the information (as the activity might be paused when the new information arrived).

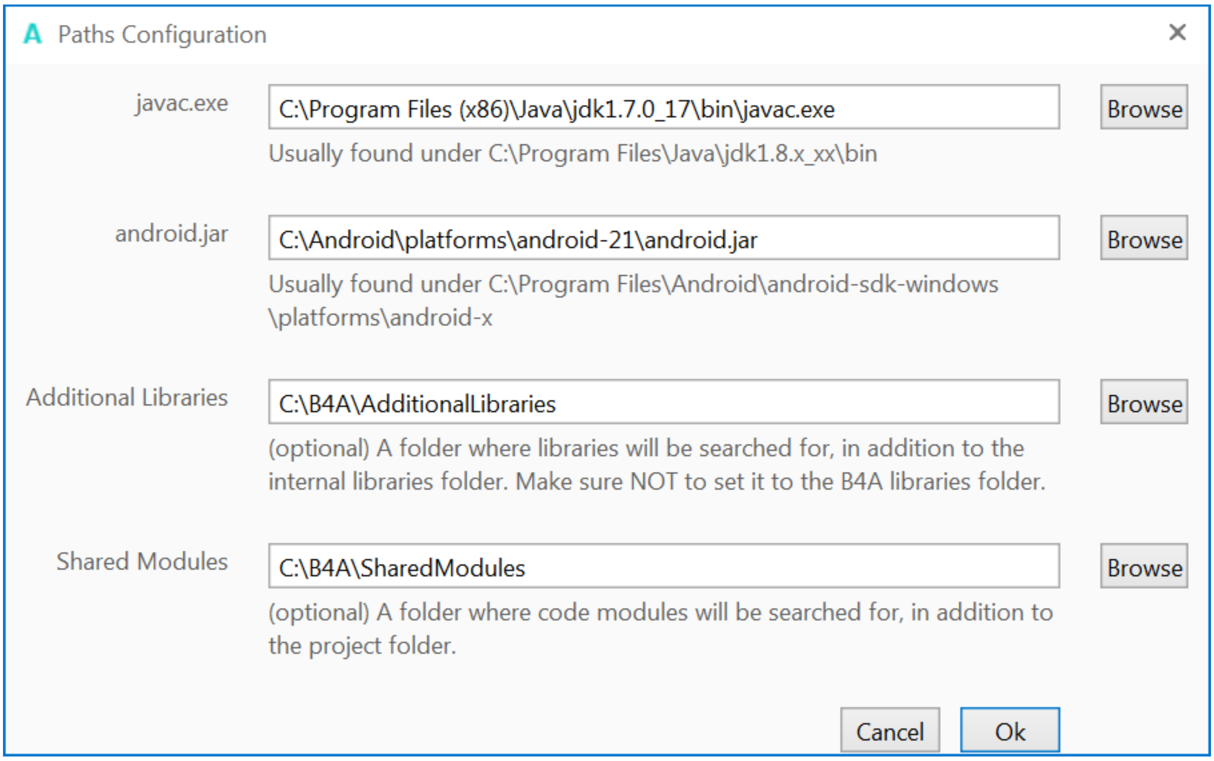
Note that it is not possible to use CallSub to access subs of a Code module.

Examples:
[Downloading a file using a service module]
[Periodically checking Twitter feeds]

11.5 Shared modules

It is possible to share modules between different applications.

These module files must be stored in a specific ‘Shared Modules’ folder which must be defined in the IDE menu *Tools* - *Configure Paths*.



Paths Configuration

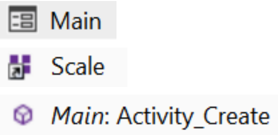
javac.exe: C:\Program Files (x86)\Java\jdk1.7.0_17\bin\javac.exe
Usually found under C:\Program Files\Java\jdk1.8.x_xx\bin

android.jar: C:\Android\platforms\android-21\android.jar
Usually found under C:\Program Files\Android\android-sdk-windows \platforms\android-x

Additional Libraries: C:\B4A\AdditionalLibraries
(optional) A folder where libraries will be searched for, in addition to the internal libraries folder. Make sure NOT to set it to the B4A libraries folder.

Shared Modules: C:\B4A\SharedModules
(optional) A folder where code modules will be searched for, in addition to the project folder.

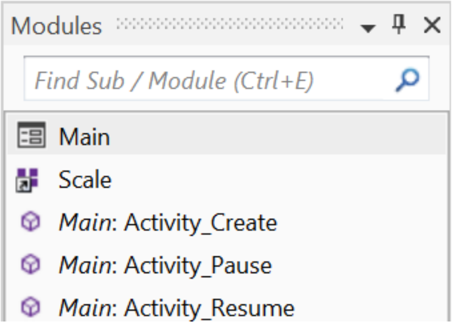
You can see that a module was loaded from the shared folder in the list of modules with the icon:



Module

Shared module

Routine, the routine is in the Main module in the example.



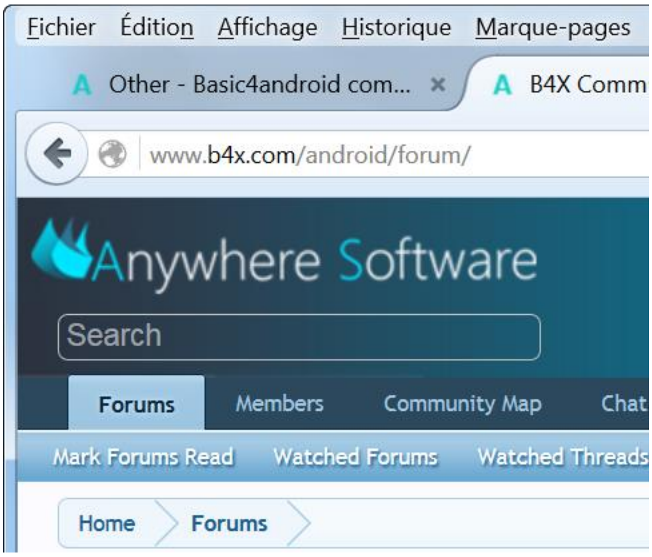
Adding a shared module to a project is done in the same way as adding a non-shared module.
You choose Project -> Add Existing Module. If the module file is in the shared folder then the module will be loaded as a shared module and will not be copied to the project folder.

If you want to convert a non-shared module to a shared module then you need to manually move the module file to the shared modules folder and reload the project

12 Help tools

To find answers to many questions about B4A the following tools are very useful.

12.1 Search function in the forum

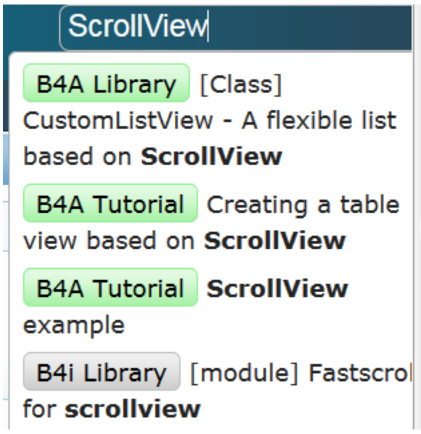


In the upper left corner you find the searchbox for the forum. Depending on the window size it can be on the top right side.

Enter a question or any keywords and press 'Return'.

The function shows you the posts that match your request.

Example: Enter the keyword ScrollView :



ScrollView

B4A Library [Class] CustomListView - A flexible list based on **ScrollView**

B4A Tutorial Creating a table view based on **ScrollView**

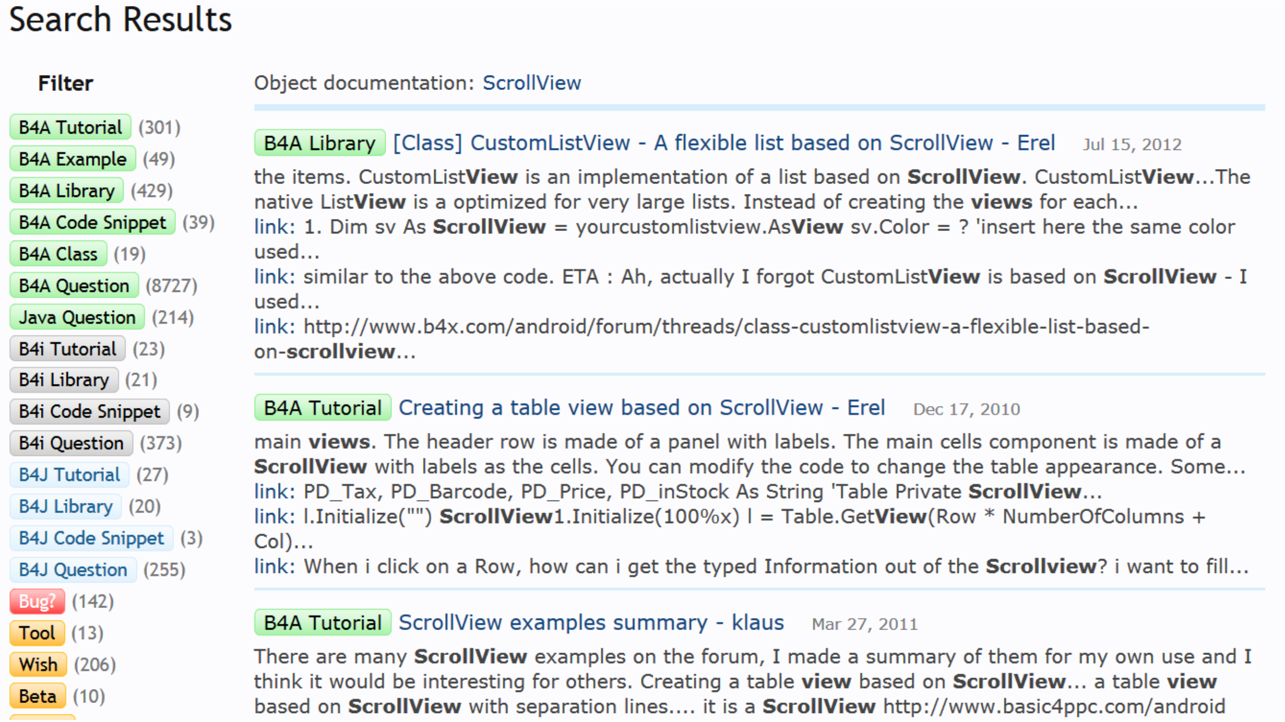
B4A Tutorial **ScrollView** example

B4i Library [module] Fastscrol for **scrollview**

A list of the result is displayed below the search box.

Click on an item to show the thread.

And the result:



Search Results

Filter

B4A Tutorial	(301)
B4A Example	(49)
B4A Library	(429)
B4A Code Snippet	(39)
B4A Class	(19)
B4A Question	(8727)
Java Question	(214)
B4i Tutorial	(23)
B4i Library	(21)
B4i Code Snippet	(9)
B4i Question	(373)
B4J Tutorial	(27)
B4J Library	(20)
B4J Code Snippet	(3)
B4J Question	(255)
Bug?	(142)
Tool	(13)
Wish	(206)
Beta	(10)

Object documentation: ScrollView

B4A Library [Class] CustomListView - A flexible list based on ScrollView - Erel Jul 15, 2012

the items. CustomList**View** is an implementation of a list based on **ScrollView**. CustomList**View**...The native List**View** is a optimized for very large lists. Instead of creating the **views** for each...
link: 1. Dim sv As **ScrollView** = yourcustomlistview.As**View** sv.Color = ? 'insert here the same color used...
link: similar to the above code. ETA : Ah, actually I forgot CustomList**View** is based on **ScrollView** - I used...
link: http://www.b4x.com/android/forum/threads/class-customlistview-a-flexible-list-based-on-**scrollview**...

B4A Tutorial Creating a table view based on ScrollView - Erel Dec 17, 2010

main **views**. The header row is made of a panel with labels. The main cells component is made of a **ScrollView** with labels as the cells. You can modify the code to change the table appearance. Some...
link: PD_Tax, PD_Barcode, PD_Price, PD_inStock As String 'Table Private **ScrollView**...
link: l.Initialize("") **ScrollView**1.Initialize(100%x) l = Table.Get**View**(Row * NumberOfColumns + Col)...
link: When i click on a Row, how can i get the typed Information out of the **Scrollview**? i want to fill...

B4A Tutorial ScrollView examples summary - klaus Mar 27, 2011

There are many **ScrollView** examples on the forum, I made a summary of them for my own use and I think it would be interesting for others. Creating a table **view** based on **ScrollView**... a table **view** based on **ScrollView** with separation lines.... it is a **ScrollView** http://www.basic4ppc.com/android

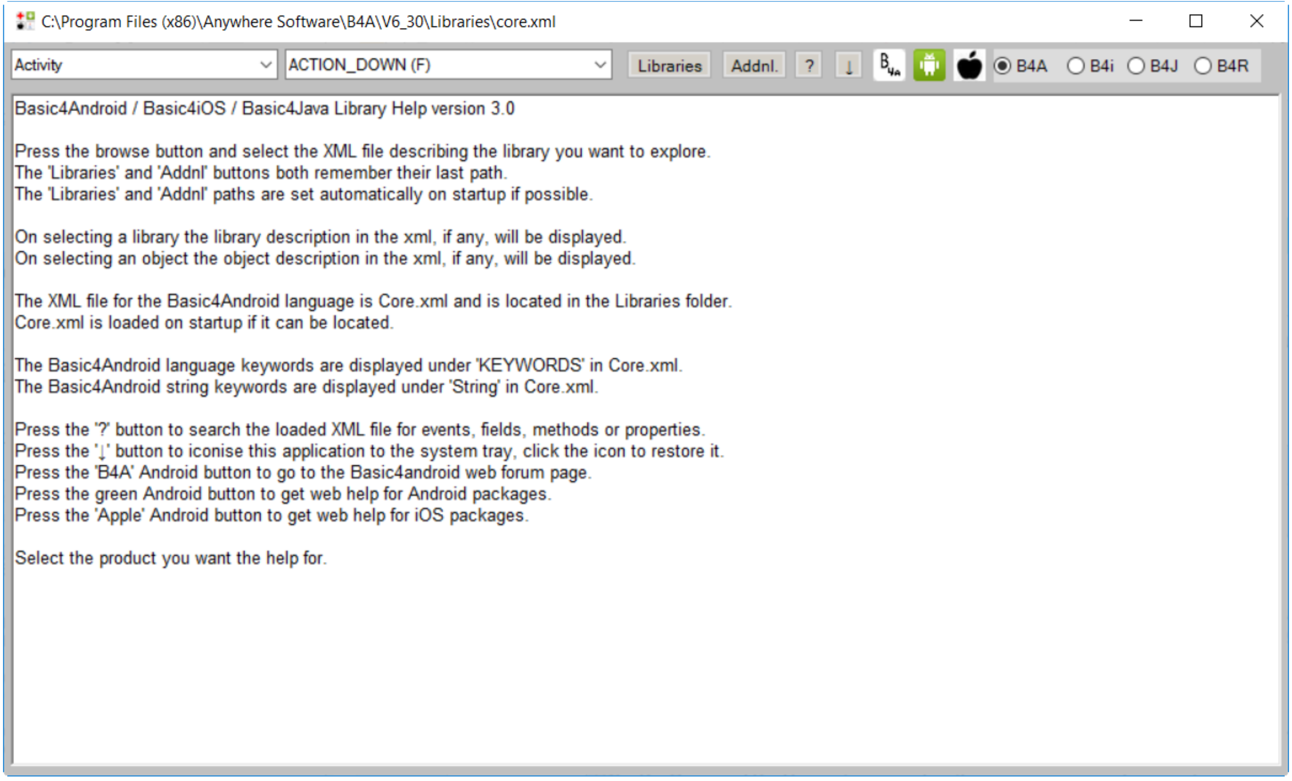
On the left you have a list of forums which you can filter.

Click on the title to show the selected post.

12.2 B4x Help Viewer

This program shows xml help files. It was originally written by Andrew Graham (agrham) for B4A. I modified it, with Andrews' agreement, to show B4A, B4J, B4i and B4R xml help files.

The program can be downloaded from the forum.


C:\Program Files (x86)\Anywhere Software\B4A\V6_30\Libraries\core.xml
Activity
ACTION_DOWN (F)
Libraries
Addnl.
?
↓
● B4A ○ B4i ○ B4J ○ B4R

Basic4Android / Basic4iOS / Basic4Java Library Help version 3.0

Press the browse button and select the XML file describing the library you want to explore.
The 'Libraries' and 'Addnl' buttons both remember their last path.
The 'Libraries' and 'Addnl' paths are set automatically on startup if possible.

On selecting a library the library description in the xml, if any, will be displayed.
On selecting an object the object description in the xml, if any, will be displayed.

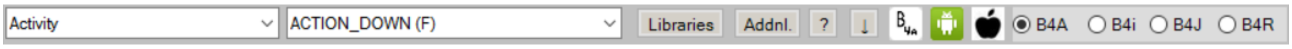
The XML file for the Basic4Android language is Core.xml and is located in the Libraries folder.
Core.xml is loaded on startup if it can be located.

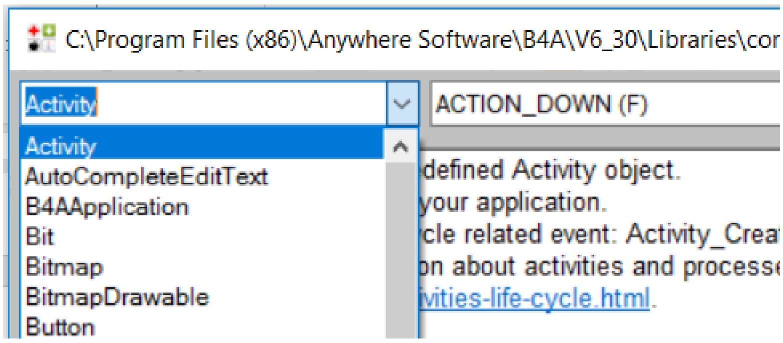
The Basic4Android language keywords are displayed under 'KEYWORDS' in Core.xml.
The Basic4Android string keywords are displayed under 'String' in Core.xml.

Press the '?' button to search the loaded XML file for events, fields, methods or properties.
Press the '↓' button to iconise this application to the system tray, click the icon to restore it.
Press the 'B4A' Android button to go to the Basic4android web forum page.
Press the green Android button to get web help for Android packages.
Press the 'Apple' Android button to get web help for iOS packages.

Select the product you want the help for.

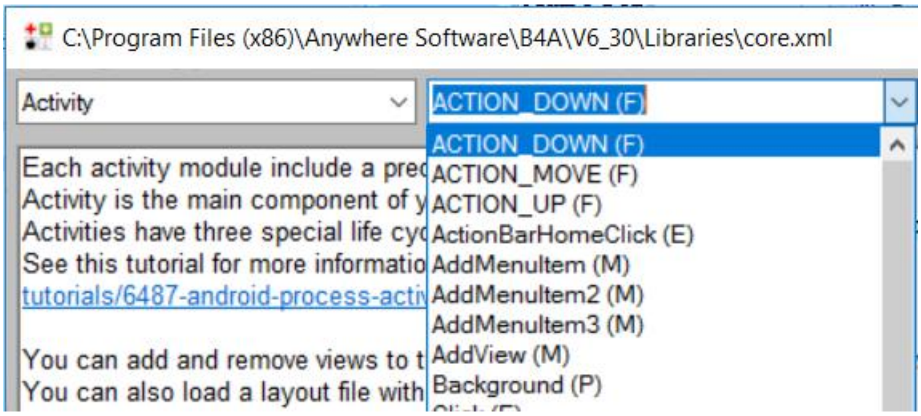
On top we find:





defined Activity object.
your application.
cle related event: Activity_Crea
on about activities and process
vities-life-cycle.html.

In the upper left corner a drop down list shows the different objects included in the selected library.



Each activity module include a pre
Activity is the main component of y
Activities have three special life cy
See this tutorial for more informatio
tutorials/6487-android-process-acti

You can add and remove views to t
You can also load a layout file with

Besides the objects list you find another drop down list with the

- methods(M)
- events(E)
- properties(P)
- fields(F) constants

for the selected object.

Libraries Select the standard library to display.

Addnl. Select the additional library to display.

? Search engine to find the object for a given keyword.

↓ Closes B4AHelp

B4A Launches the forum 'Online Community'.

Launches the Android Developers site.

Launches the iOS developer’s site.

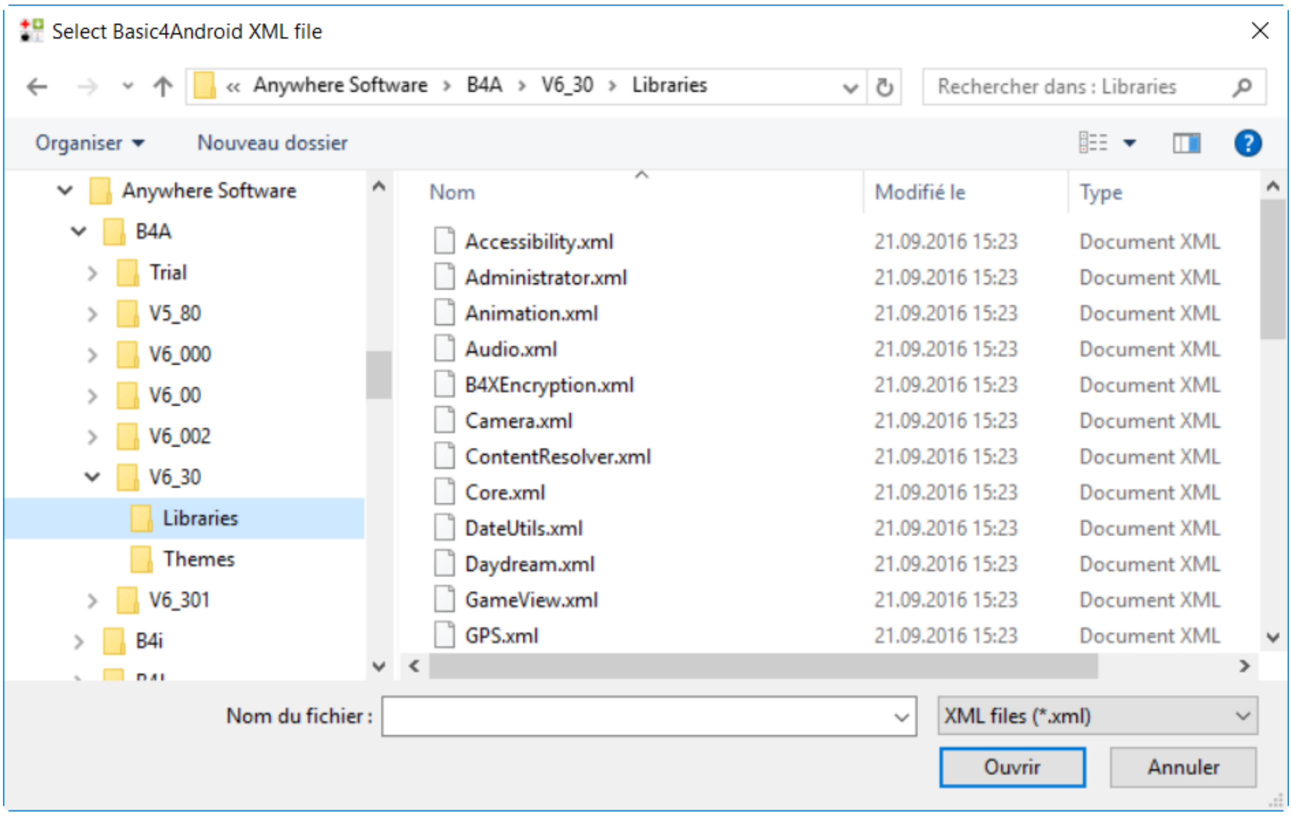
○ B4A B4A help files.

● B4i B4i help files.

○ B4J B4J help files.

○ B4R B4R help files.

Libraries Standard libraries



Nom	Modifié le	Type
Accessibility.xml	21.09.2016 15:23	Document XML
Administrator.xml	21.09.2016 15:23	Document XML
Animation.xml	21.09.2016 15:23	Document XML
Audio.xml	21.09.2016 15:23	Document XML
B4XEncryption.xml	21.09.2016 15:23	Document XML
Camera.xml	21.09.2016 15:23	Document XML
ContentResolver.xml	21.09.2016 15:23	Document XML
Core.xml	21.09.2016 15:23	Document XML
DateUtils.xml	21.09.2016 15:23	Document XML
Daydream.xml	21.09.2016 15:23	Document XML
GameView.xml	21.09.2016 15:23	Document XML
GPS.xml	21.09.2016 15:23	Document XML

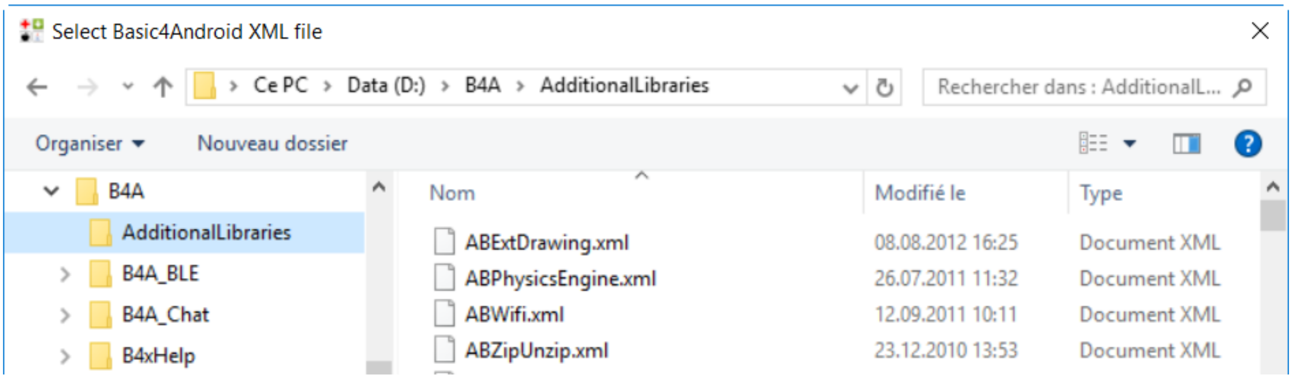
Select the library to display and click on Ouvrir (Open).

Here « Anywhere Software › B4A › V6_30 › Libraries you can select the directory where the standard libraries are saved.

Once selected the directory is saved for the next start of the program.

Addnl. Additional libraries.

The same also for the additional libraries.



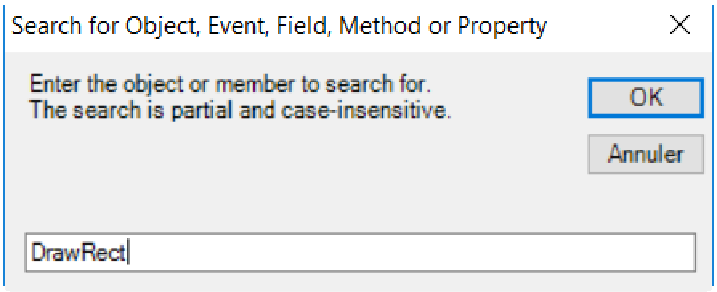
Select Basic4Android XML file

Ce PC > Data (D:) > B4A > AdditionalLibraries
Rechercher dans : AdditionalL...
Organiser ▾
Nouveau dossier
B4A
AdditionalLibraries
B4A_BLE
B4A_Chat
B4xHelp

Nom	Modifié le	Type
ABExtDrawing.xml	08.08.2012 16:25	Document XML
ABPhysicsEngine.xml	26.07.2011 11:32	Document XML
ABWifi.xml	12.09.2011 10:11	Document XML
ABZipUnzip.xml	23.12.2010 13:53	Document XML

Here you can select the directory for the additional libraries.

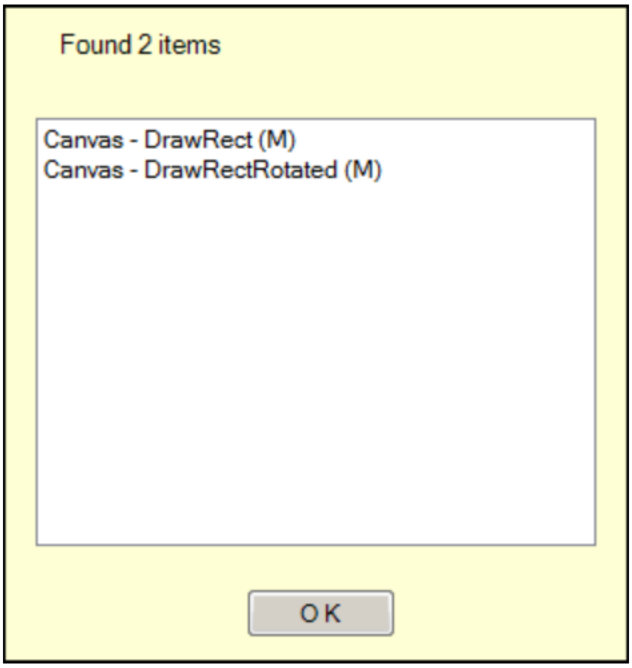
? Search engine for the selected library.


Search for Object, Event, Field, Method or Property

Enter the object or member to search for.
The search is partial and case-insensitive.

DrawRect

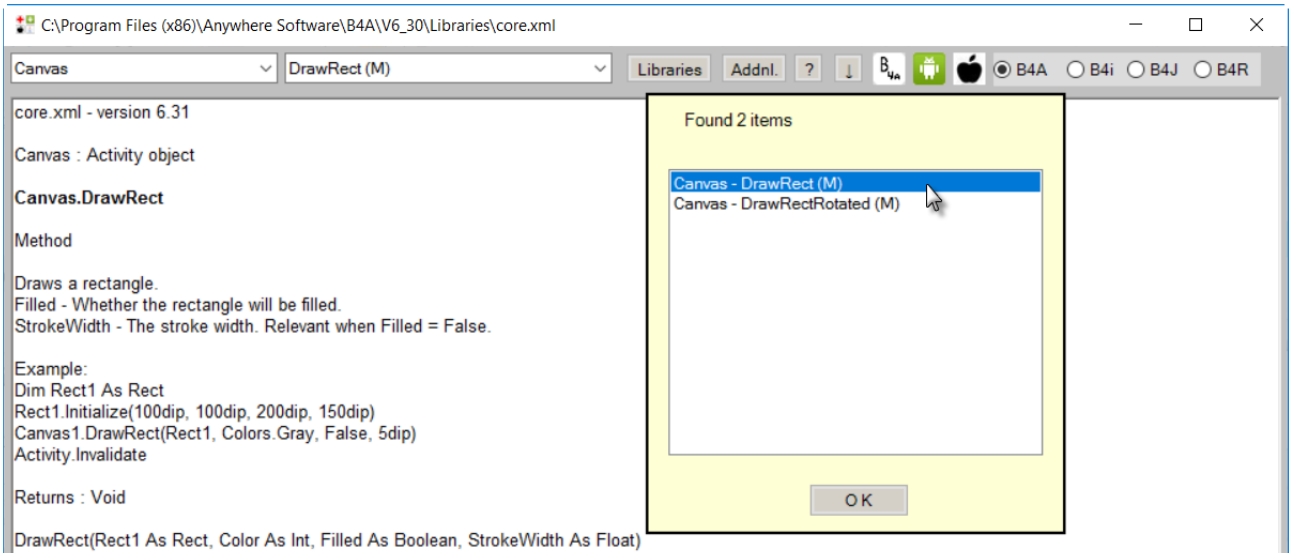
Example:
Selected library: Core
Enter *DrawRect*


Found 2 items

Canvas - DrawRect (M)
Canvas - DrawRectRotated (M)

And the result.

We get the object Canvas and two methods.


C:\Program Files (x86)\Anywhere Software\B4A\V6_30\Libraries\core.xml

Canvas | DrawRect (M)

core.xml - version 6.31

Canvas : Activity object

Canvas.DrawRect

Method

Draws a rectangle.
Filled - Whether the rectangle will be filled.
StrokeWidth - The stroke width. Relevant when Filled = False.

Example:
Dim Rect1 As Rect
Rect1.Initialize(100dip, 100dip, 200dip, 150dip)
Canvas1.DrawRect(Rect1, Colors.Gray, False, 5dip)
Activity.Invalidate

Returns : Void

DrawRect(Rect1 As Rect, Color As Int, Filled As Boolean, StrokeWidth As Float)

Found 2 items

Canvas - DrawRect (M)
Canvas - DrawRectRotated (M)

Click on an item in the list to show its help.

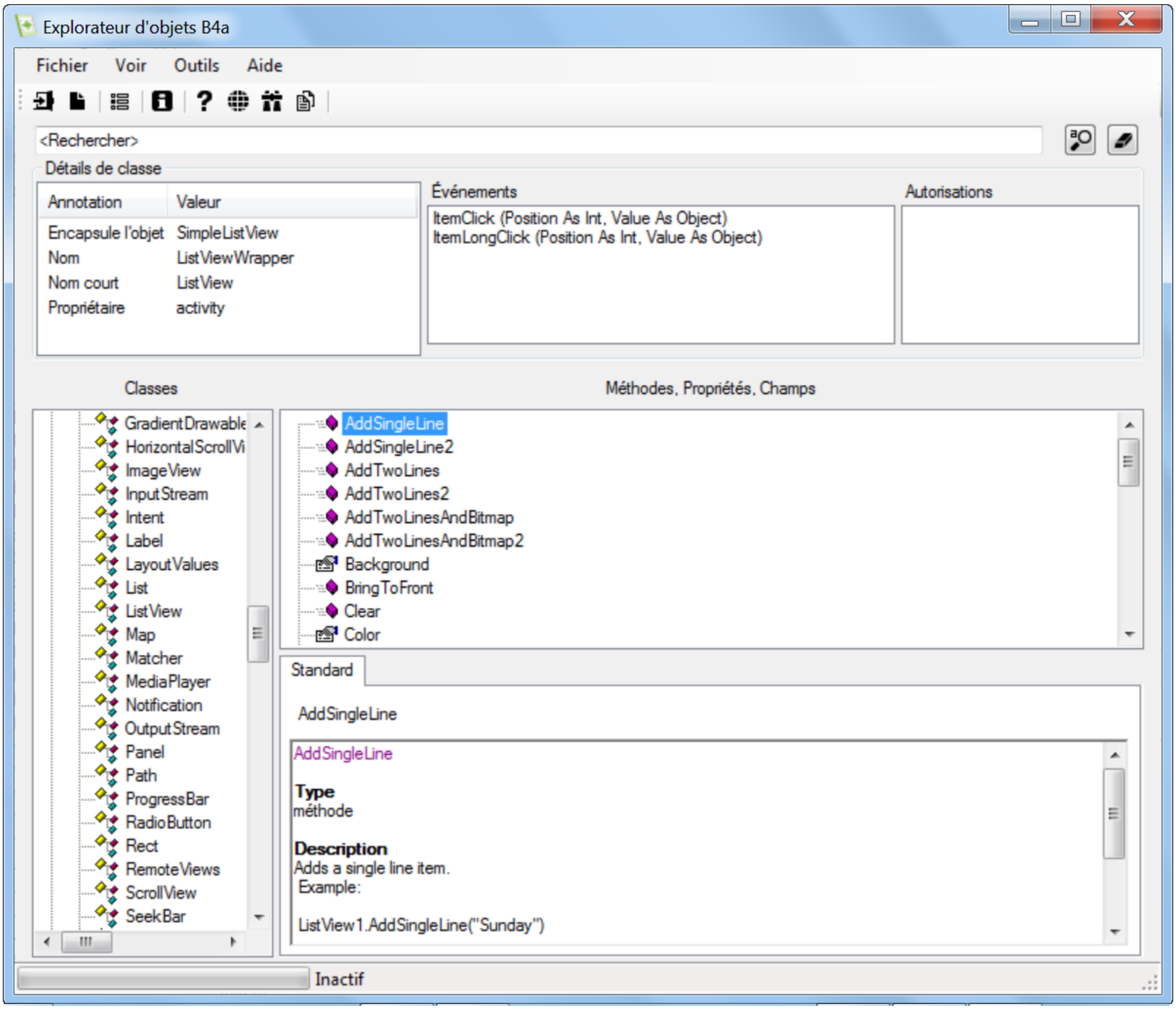
Click on OK to leave the search result list.

12.3 Help documentation - B4A Object Browser

This is also a standalone Windows program showing the help files of libraries.

It has been written by Vader and can be downloaded here.

A pdf documentation on how to use the program is part of the download.



Annotation	Valeur
Encapsule l'objet	SimpleListView
Nom	ListViewWrapper
Nom court	ListView
Propriétaire	activity

ItemClick (Position As Int, Value As Object)
ItemLongClick (Position As Int, Value As Object)

GradientDrawable
HorizontalScrollVi
ImageView
InputStream
Intent
Label
LayoutValues
List
ListView
Map
Matcher
MediaPlayer
Notification
OutputStream
Panel
Path
ProgressBar
RadioButton
Rect
RemoteViews
ScrollView
SeekBar

AddSingleLine
AddSingleLine2
AddTwoLines
AddTwoLines2
AddTwoLinesAndBitmap
AddTwoLinesAndBitmap2
Background
BringToFront
Clear
Color

AddSingleLine

AddSingleLine

Type
méthode

Description
Adds a single line item.
Example:

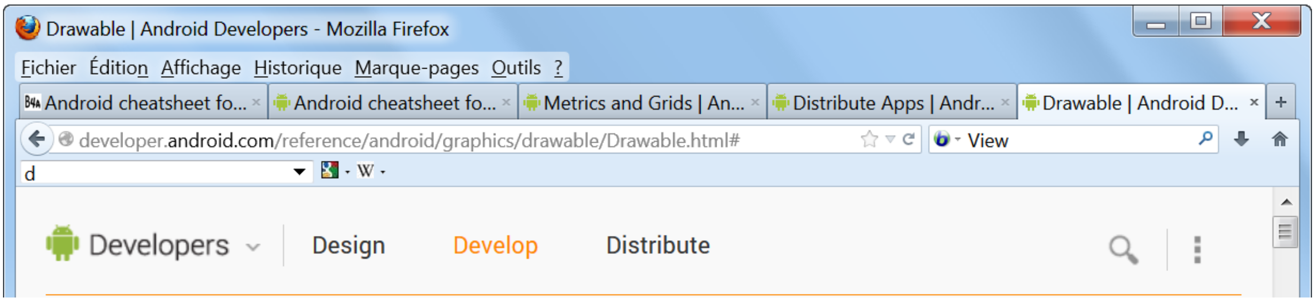
ListView1.AddSingleLine("Sunday")

12.4 Useful links

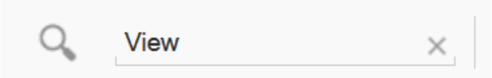
A useful link for layout graphics. Android cheat sheet for graphic designers

Android Developers. Design Develop Distribute

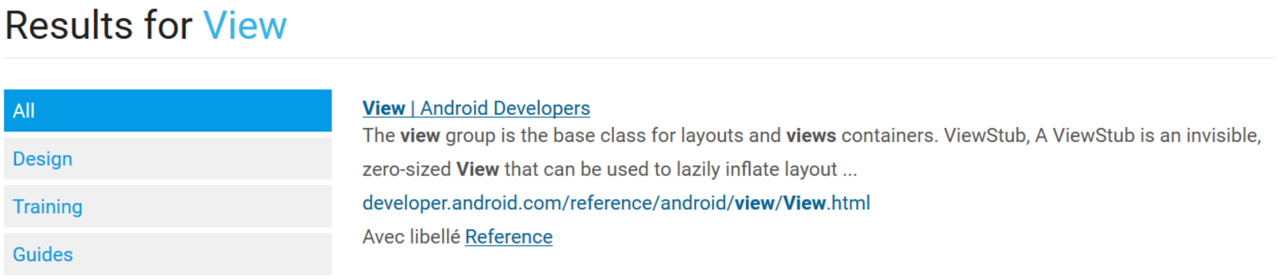
Android Developers searching for any request.


Drawable | Android Developers - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils ?
Android cheatsheet fo... × Android cheatsheet fo... × Metrics and Grids | An... × Distribute Apps | Andr... × Drawable | Android D... × +
developer.android.com/reference/android/graphics/drawable/Drawable.html#
View
d
W
Developers
Design
Develop
Distribute

In the upper right corner you find the search field.

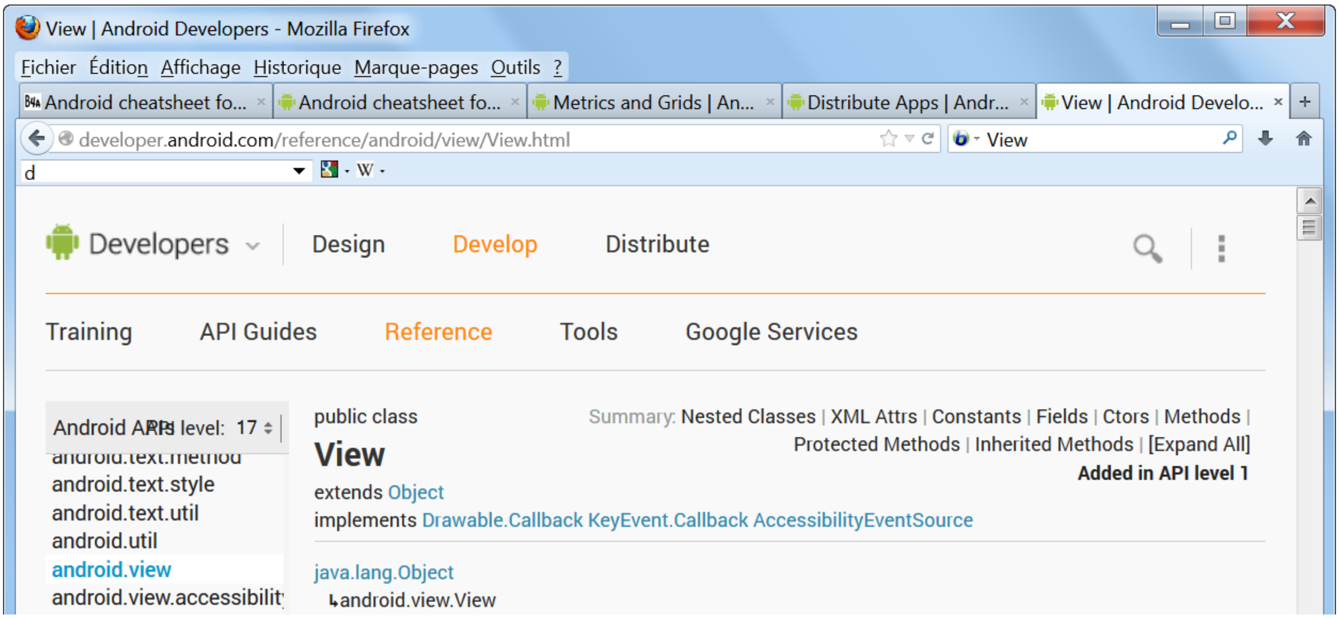
Enter View in the field:


Click on the link View | Android Developers.


Results for View

View | Android Developers
The **view** group is the base class for layouts and **views** containers. ViewStub, A ViewStub is an invisible, zero-sized **View** that can be used to lazily inflate layout ...
developer.android.com/reference/android/**view**/**View**.html
Avec libellé Reference

And you get all the information about Views.


View | Android Developers - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils ?
Android cheatsheet fo... × Android cheatsheet fo... × Metrics and Grids | An... × Distribute Apps | Andr... × View | Android Develo... × +
developer.android.com/reference/android/view/View.html
View
d
W
Developers
Design
Develop
Distribute
Training
API Guides
Reference
Tools
Google Services
Android APIs level: 17
android.text.method
android.text.style
android.text.util
android.util
android.view
android.view.accessibilit

public class
View
extends Object
implements Drawable.Callback KeyEvent.Callback AccessibilityEventSource

Summary: Nested Classes | XML Attrs | Constants | Fields | Ctors | Methods | Protected Methods | Inherited Methods | [Expand All]
Added in API level 1

java.lang.Object
↳android.view.View

12.5 Books

B4A book

Written by Philip Brown under the pseudo Wyken Seagrave.

http://www.b4x.com/android/forum/threads/book-now-available-for-b4a-version-5-02.55292/#post-348887

MagBook Build your own Android App.

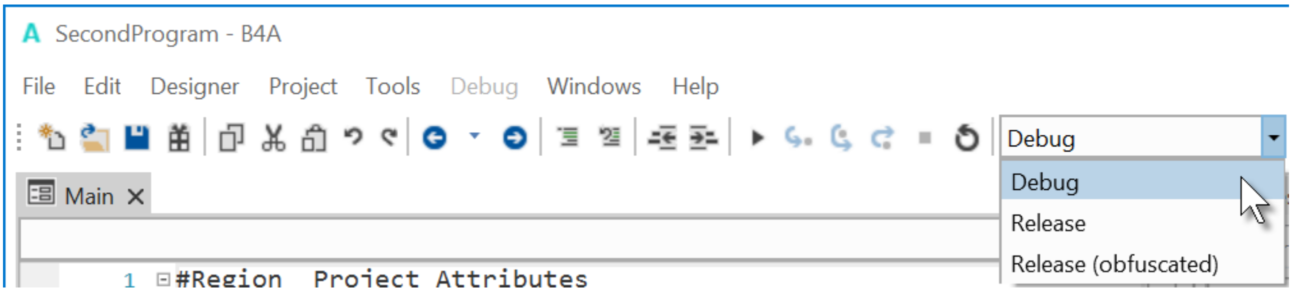
Written by Nigel Whitfield.

http://www.magbooks.com/product/build-your-own-android-app/

13 Debugging

Debugging is an important part when developing.

To allow debugging you must activate the debugging mode *Debug* on top of the IDE.



```
#Region  Project Attributes
```

Notes about the debugger:

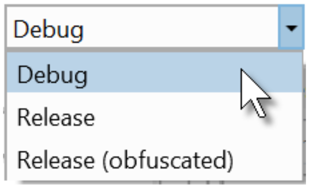
- Breakpoints in the following subs will be ignored: Globals, Process_Globals and Activity_Pause.
- Services - Breakpoints that appear after a call to StartService will be ignored. Breakpoints set in Service_Create and Service_Start will pause the program for up to a specific time (about 12 seconds). This is to prevent the OS from killing the Service.
- Events that fire when the program is paused will be executed. Breakpoints in the event code will be ignored (only when the program is already paused).
- The data sent from the device to the IDE is limited in size. Long strings may be truncated.
- When the debugger is running in *rapid* mode, you can change the code and run the changes.
- When the debugger is running in *legacy* mode, the IDE is read-only. The user cannot change any of the program text.

The two major utilities for debugging are:

Breakpoints - You can mark lines of codes as breakpoints. This is done by pressing on the grey area left of the line.
The program will pause when it reaches a breakpoint and will allow you to inspect the current state.

Logging - The Logs tab at the right pane is very useful. It shows messages related to the components life cycle and it can also show messages that are printed with the Log keyword. You should press on the Connect button to connect to the device logs. Note that there is a Filter checkbox. When it is checked you will only see messages related to your program. When it is unchecked you will see all the messages running in the system. If you are encountering an error and do not see any relevant message in the log, it is worth unchecking the filter option and looking for an error message.
Note that the log is maintained by the device. When you connect to a device you will also see previous messages.

13.1 Debug mode

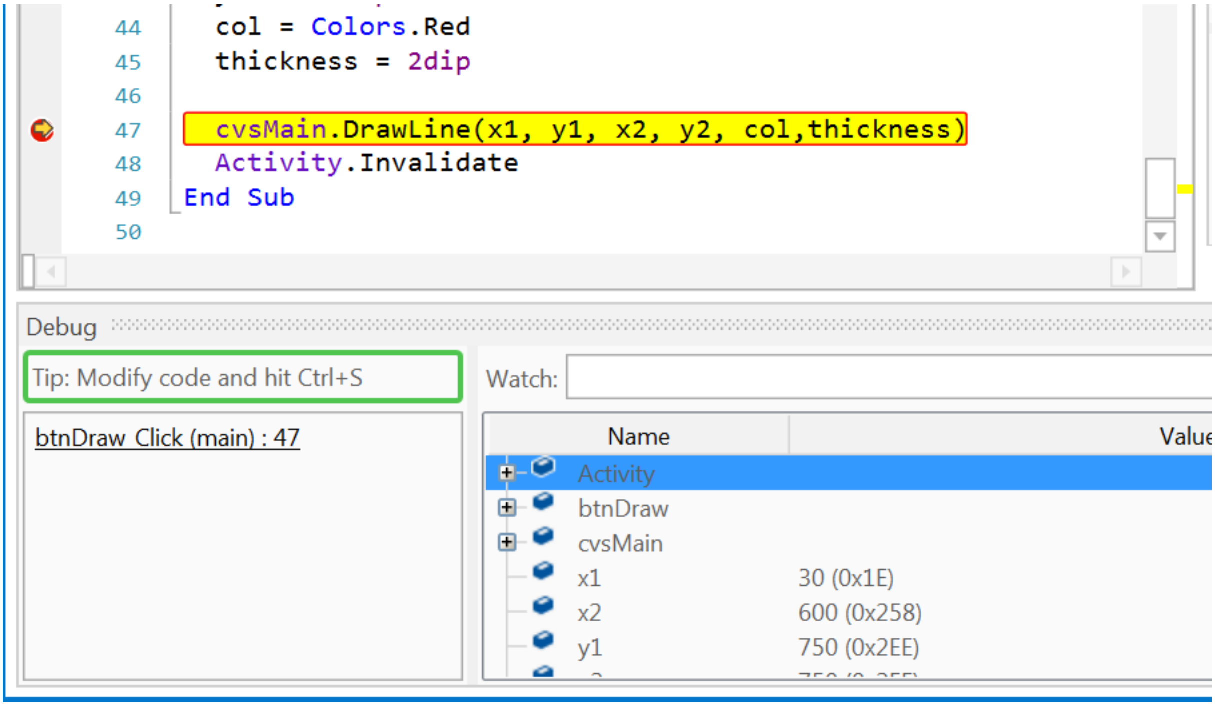


The debugger is very sophisticated with features not available in any other native Android development tool.

It will increase your productivity as it makes the "write code -> test result" cycle much quicker.

13.1.1 Debugger advantages

- Very quick compilation and installation. Usually in less than one or two seconds.
- In most cases (after the first installation) there is no need to reinstall the APK. This means that the deployment is much quicker. With B4A-Bridge there is no need to approve the installation.
- Hot code swapping (edit and continue). You can modify the code while the app runs, hit Save and the code will be updated.
- Watch Expressions feature.
- Powerful variables browser:



```
    col = Colors.Red
    thickness = 2dip

    cvsMain.DrawLine(x1, y1, x2, y2, col,thickness)
    Activity.Invalidate
End Sub
```

Name	Value
Activity	
btnDraw	
cvsMain	
x1	30 (0x1E)
x2	600 (0x258)
y1	750 (0x2EE)

13.1.2 Debugger Limitations

- The runtime execution in this mode is almost as rapid like non debugging. The debugger is not suitable for debugging "real-time" games or CPU intensive tasks. This is why the Legacy debugger is kept.
- Hot code swapping is very powerful. You can even add subs or modify existing subs. However you cannot add or remove global variables.
- Unlike the legacy debugger, the app cannot run when the IDE is not connected. It will wait for 10 seconds for the IDE to connect and then exit.

So how does the debugger work?

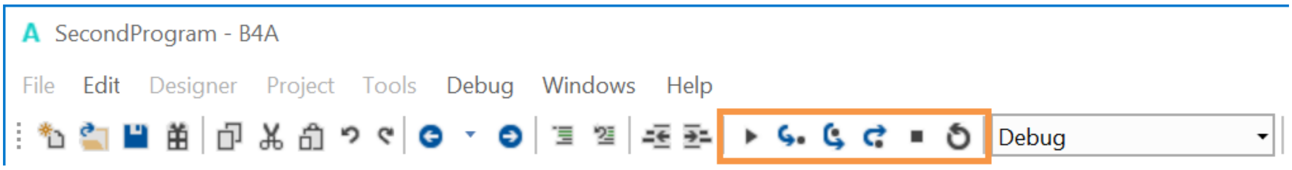
I will start with an explanation about the legacy debugger. When you compile in Debug (legacy) the B4A compiler creates a regular Android APK. However in addition to the program code the compiler generates instrumented code. This means that for each line it adds a **runtime** check to test whether there is a breakpoint on that line. If there is a breakpoint the program pauses by showing a modal dialog and the variables data is sent to the IDE over the connection.

The debugger works differently. When you compile your code it creates two applications. A device application (shell) and a standard Java application (debugger engine).

Your code resides in the debugger engine. The debugger engine runs on the desktop. The shell application is like a mini-virtual machine. The debugger engine connects to the shell app and sends the instructions to the shell app. The interesting part of this is that your code is not executed on the device. It is executed on the desktop.
In most cases the shell app can be reused. If for example you add a new file or edit the manifest file with the manifest editor then the compiler will create a new shell app and will reinstall it automatically during compilation.

13.1.3 Debug Toolbar

The debug toolbar is at the right side of the IDE toolbar.



Debug Toolbar: ▶ ↪ ↷ ↩ ■ ↻

▶	Run the program	F5	Runs the program, no action in Debug (rapid)
↪	Step In	F8	Executes the next statement.
↷	Step Over	F9	Executes a routine without jumping in it.
↩	Step Out	F10	Finishes executing the rest of a routine.
■	Stop		Stops the program.
↻	Restart	F11	Restarts the program.

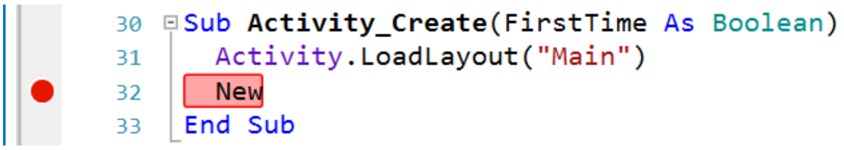
The examples below are shown in the SecondProgram project.

13.1.3.1 Run ▶ F5

Runs the program,
If the program is stopped at a breakpoint the program runs until the next breakpoint or completes running.

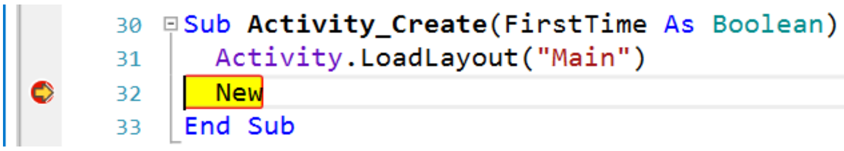
13.1.3.2 Step In F8

The debugger executes the code step by step.



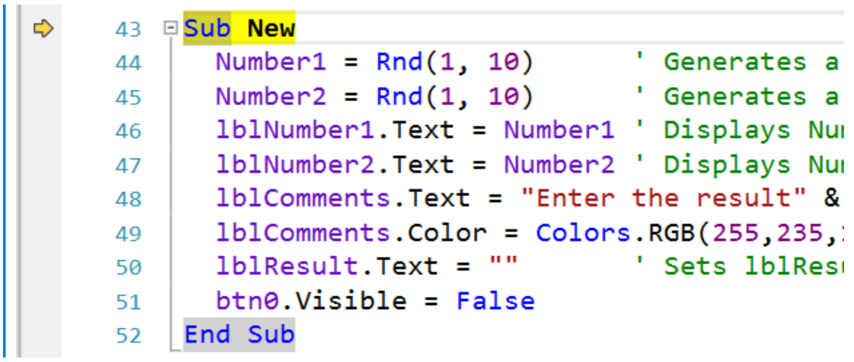
```
30 ⊟Sub Activity_Create(FirstTime As Boolean)
31      Activity.LoadLayout("Main")
32      New
33   End Sub
```

In the SecondProgram project we set a Breakpoint at line 32 `New`.



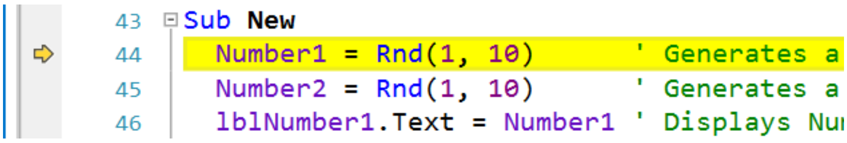
```
30 ⊟Sub Activity_Create(FirstTime As Boolean)
31      Activity.LoadLayout("Main")
32      New
33   End Sub
```

We run the program, it will stop executing at line 32 `New`.



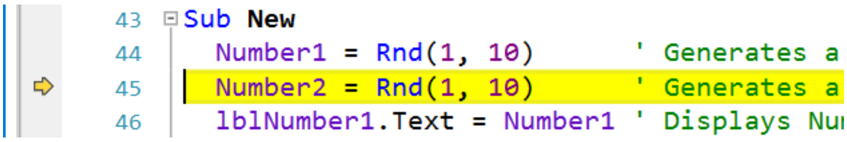
```
43 ⊟Sub New
44      Number1 = Rnd(1, 10)        ' Generates a
45      Number2 = Rnd(1, 10)        ' Generates a
46      lblNumber1.Text = Number1 ' Displays Nu
47      lblNumber2.Text = Number2 ' Displays Nu
48      lblComments.Text = "Enter the result" &
49      lblComments.Color = Colors.RGB(255,235,
50      lblResult.Text = ""         ' Sets lblRes
51      btn0.Visible = False
52   End Sub
```

Click on .
The debugger executes the next line, `Sub New` in this case.



```
43 ⊟Sub New
44      Number1 = Rnd(1, 10)        ' Generates a
45      Number2 = Rnd(1, 10)        ' Generates a
46      lblNumber1.Text = Number1 ' Displays Nu
```

Click once more on .
The debugger executes the next line, `Number1 =...`

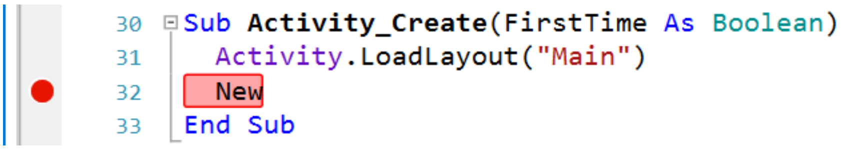


```
43 ⊟Sub New
44      Number1 = Rnd(1, 10)        ' Generates a
45      Number2 = Rnd(1, 10)        ' Generates a
46      lblNumber1.Text = Number1 ' Displays Nu
```

Click once more on .
The debugger executes the next line, `Number2 =...`

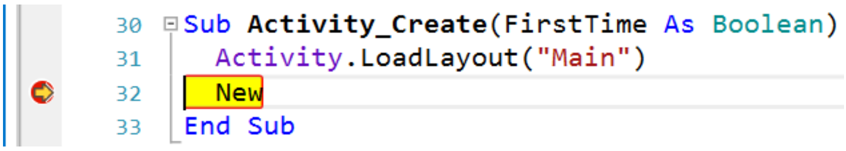
13.1.3.3 Step Over F9

If the current line is a sub calling line the debugger executes the code in this subroutine and jumps to the line after the calling line.



```
30 Sub Activity_Create(FirstTime As Boolean)
31   Activity.LoadLayout("Main")
32   New
33 End Sub
```

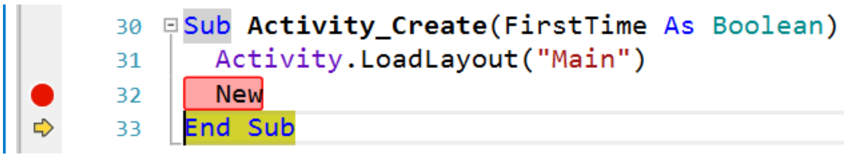
In the SecondProgram project we set a Breakpoint at line 32 `New`.



```
30 Sub Activity_Create(FirstTime As Boolean)
31   Activity.LoadLayout("Main")
32   New
33 End Sub
```

We run the program, it will stop executing at line 32 `New`.

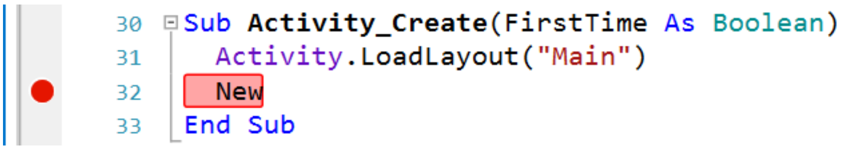
Click on .
The debugger executes the code in `New` and jumpes directly to the next line which is `End Sub` of `Activity_Create`.



```
30 Sub Activity_Create(FirstTime As Boolean)
31   Activity.LoadLayout("Main")
32   New
33 End Sub
```

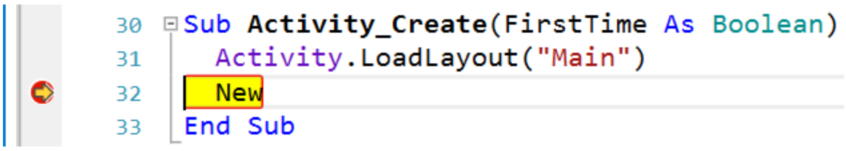
13.1.3.4 Step Out F10

If the current line is in a subroutine the debugger finishes executing the rest of the code and jumps to the next line after the subs' calling line.



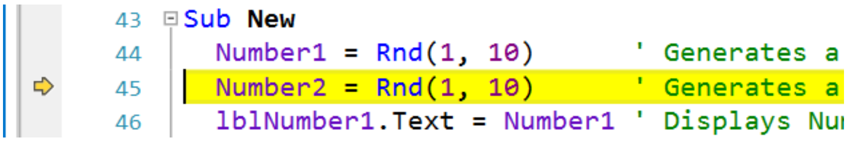
```
30 Sub Activity_Create(FirstTime As Boolean)
31   Activity.LoadLayout("Main")
32   New
33 End Sub
```

In the SecondProgram project we set a Breakpoint at line 32 `New`.



```
30 Sub Activity_Create(FirstTime As Boolean)
31   Activity.LoadLayout("Main")
32   New
33 End Sub
```

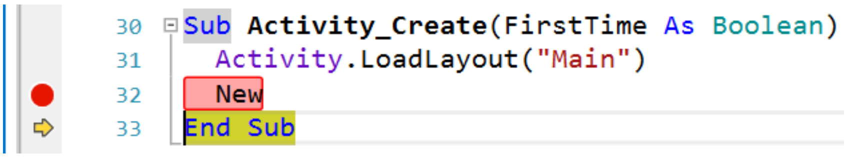
We run the program, it will stop executing at line 32 `New`.



```
43 Sub New
44   Number1 = Rnd(1, 10)        ' Generates a
45   Number2 = Rnd(1, 10)        ' Generates a
46   lblNumber1.Text = Number1   ' Displays Nu
```

We go step by step with to a line in the subroutine.

Click on .
The debugger executes the rest of the code in the subroutine and jumps to the next line which is `End Sub` of `Activity_Create`.



```
30 Sub Activity_Create(FirstTime As Boolean)
31   Activity.LoadLayout("Main")
32   New
33 End Sub
```

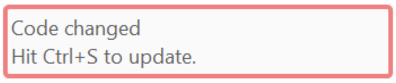
13.1.3.5 Stop ■

Stops the program and leaves the Rapid Debugger.

13.1.3.6 Restart ↻ F11

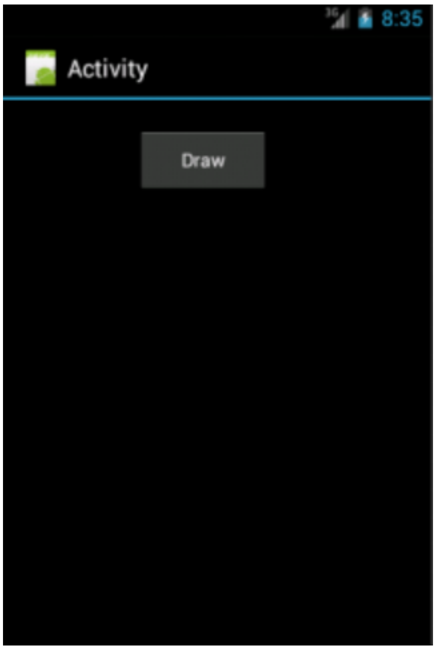
Restarts the program remaining in the Rapid Debugger.
Executes Process_Globals, Globals, Activity_Create and reloads the layout.

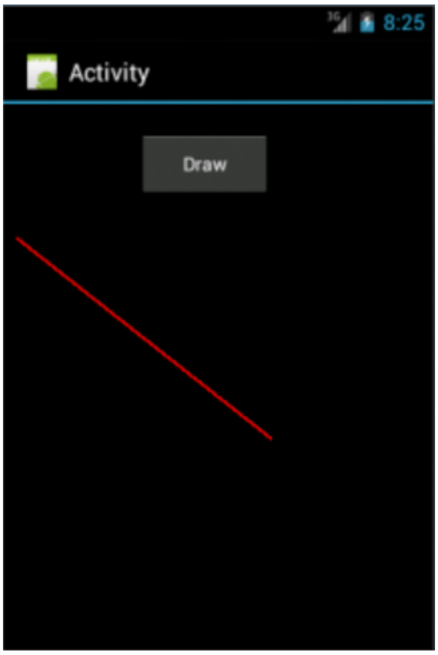
This is useful if you changed a layout file.

It is different from  Code changed
Hit Ctrl+S to update. explained in the next chapter.

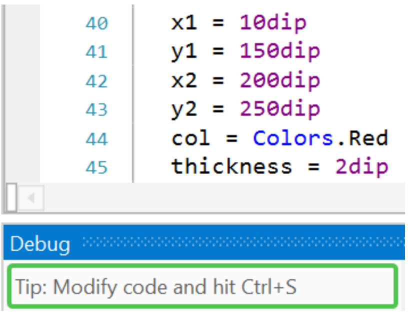
13.1.4 Small debug example

The code used is *DebugRapid.b4a* is in the *DebugRapid* folder in the source code folder:





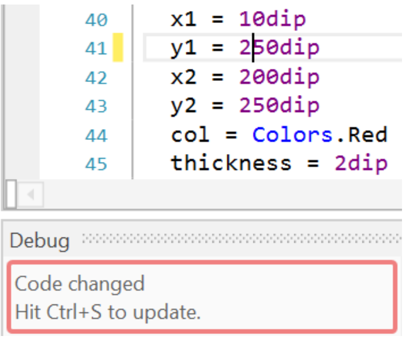
If you click on the Draw button the red line is drawn.



```
40    x1 = 10dip
41    y1 = 150dip
42    x2 = 200dip
43    y2 = 250dip
44    col = Colors.Red
45    thickness = 2dip
```

Debug

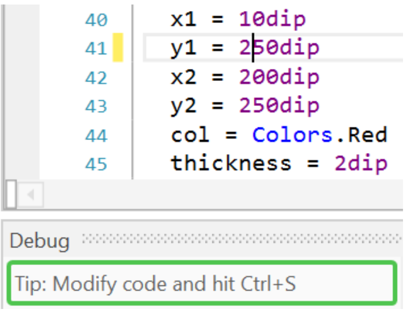
Tip: Modify code and hit Ctrl+S



```
40    x1 = 10dip
41    y1 = 250dip
42    x2 = 200dip
43    y2 = 250dip
44    col = Colors.Red
45    thickness = 2dip
```

Debug

Code changed
Hit Ctrl+S to update.



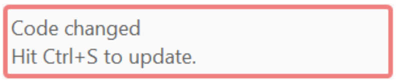
```
40    x1 = 10dip
41    y1 = 250dip
42    x2 = 200dip
43    y2 = 250dip
44    col = Colors.Red
45    thickness = 2dip
```

Debug

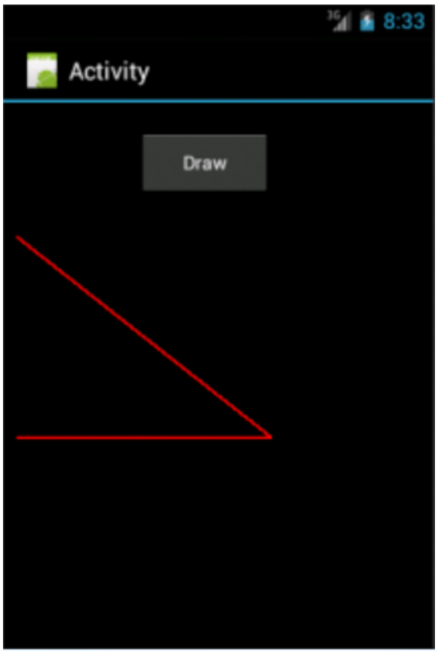
Tip: Modify code and hit Ctrl+S

In the code change line 41 `y1 = 150dip` to `y1 = 250dip` and hit Ctrl + s

or click on



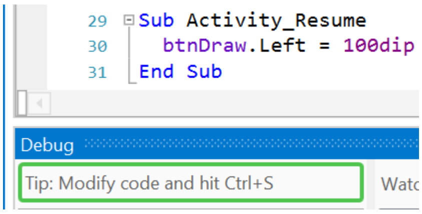
Code changed
Hit Ctrl+S to update.



Click on the Draw button, a new line with the new coordinate is drawn without rerunning the program.

In the Debug rapid mode the Restart button ↻ allows to restart the program.

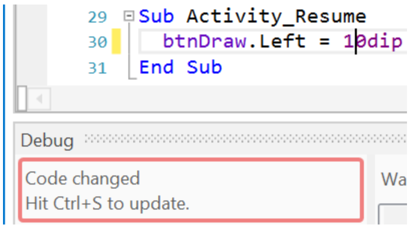
Changing the position of `btnDraw` in the code:



```
29 Sub Activity_Resume
30     btnDraw.Left = 100dip
31 End Sub
```

Debug

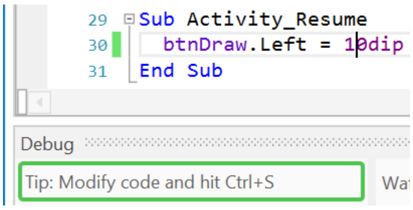
Tip: Modify code and hit Ctrl+S



```
29 Sub Activity_Resume
30     btnDraw.Left = 10dip
31 End Sub
```

Debug

Code changed
Hit Ctrl+S to update.

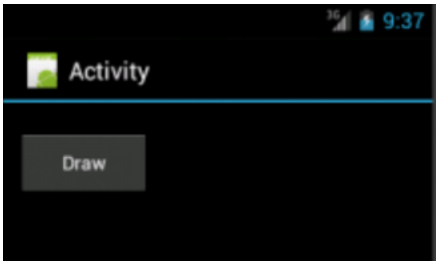


```
29 Sub Activity_Resume
30     btnDraw.Left = 10dip
31 End Sub
```

Debug

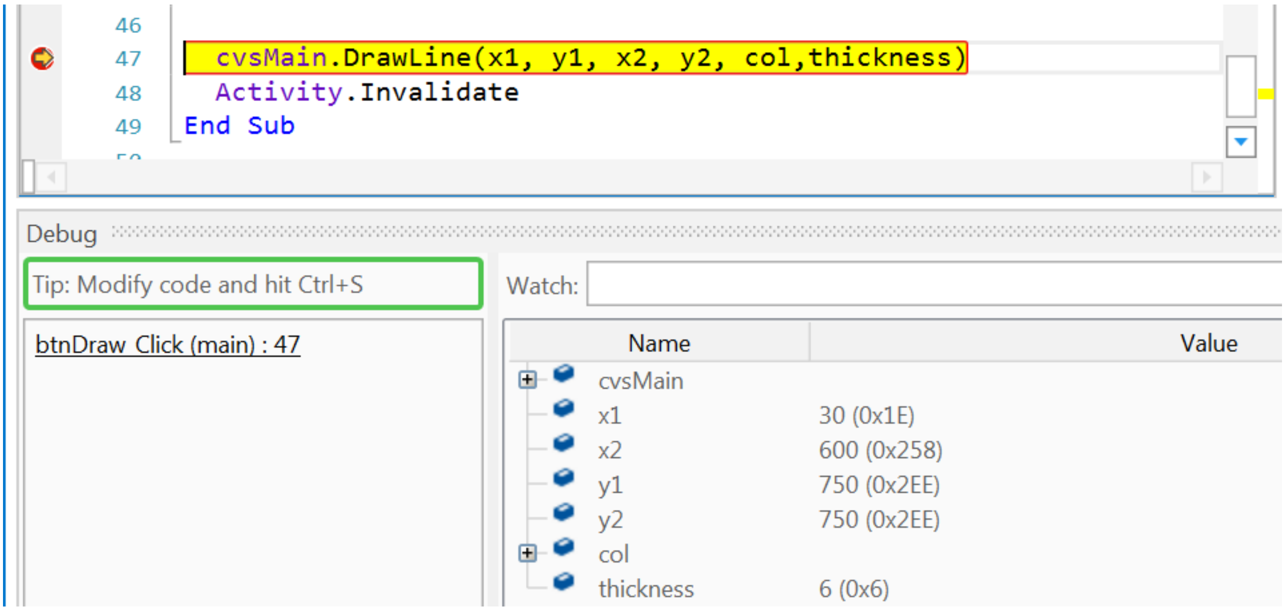
Tip: Modify code and hit Ctrl+S

In the code change line 30 `btnDraw.Left = 100dip`
to `btnDraw.Left = 10dip`
and click on ↻
in the Toolbar
to restart the program



The Button has moved without stopping and rerunning the program.

If you set a breakpoint in the code and run it, you will find a window in the lower part of the screen showing all objects with their properties and all variables with their current values.



```
46
47     cvsMain.DrawLine(x1, y1, x2, y2, col,thickness)
48     Activity.Invalidate
49 End Sub
```

Debug

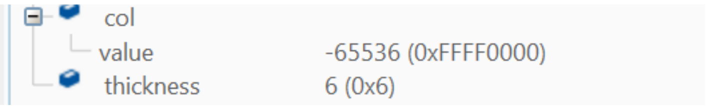
Tip: Modify code and hit Ctrl+S

btnDraw_Click (main) : 47

Watch:

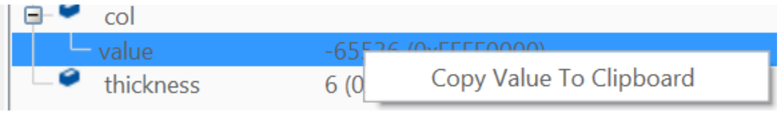
Name	Value
⊞ cvsMain	
x1	30 (0x1E)
x2	600 (0x258)
y1	750 (0x2EE)
y2	750 (0x2EE)
⊞ col	
thickness	6 (0x6)

You can open and close objects or variables by clicking on ⊞ or ⊟ .



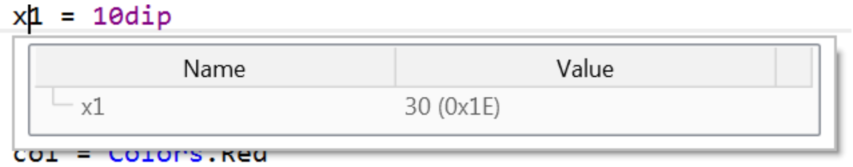
⊟ col	
value	-65536 (0xFFFF0000)
thickness	6 (0x6)

Right click on an item to copy its value to the clipboard.



⊟ col	
value	-65536 (0xFFFF0000)
thickness	6 (0x6)

Hovering over a variable shows its name and value in a pop up window.

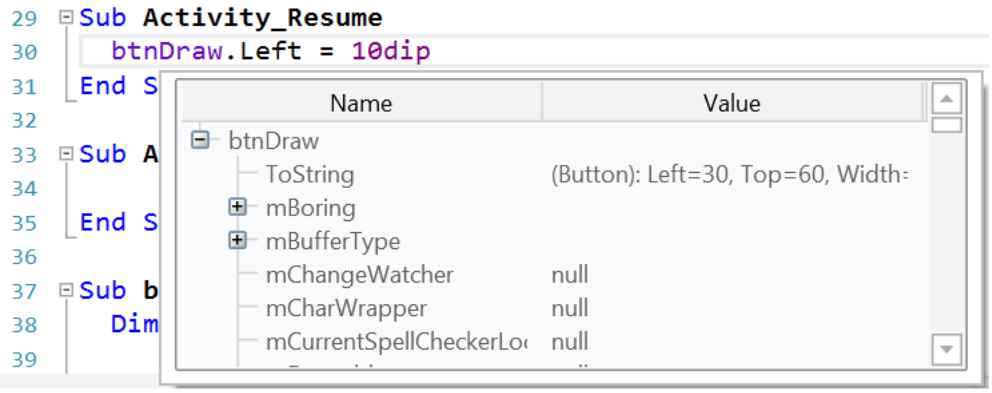


```
x1 = 10dip
col = Colors.Red
```

Name	Value
x1	30 (0x1E)

If the value is truncated hover over Value, the whole content will be displayed.

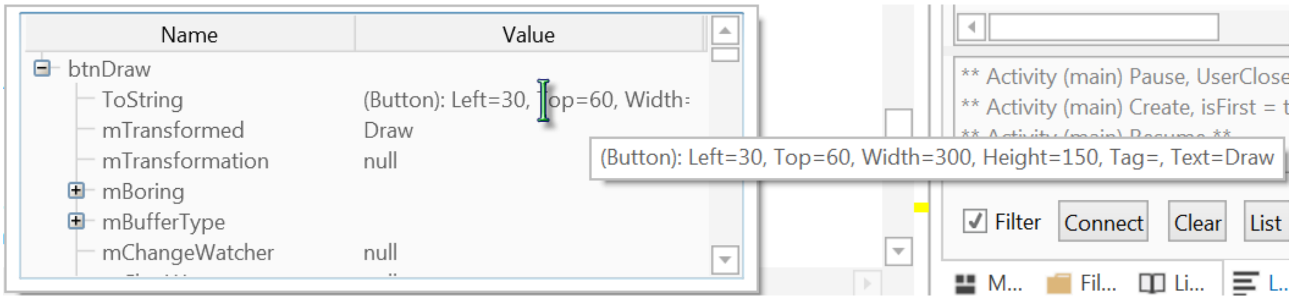
Hovering over an object in the code, **`btnDraw`** in the example, shows all properties of this object in a pop up window.



```
Sub Activity_Resume
    btnDraw.Left = 10dip
End S

Sub A

End S

Sub b
    Dim
```

Name	Value
⊟ btnDraw	
ToString	(Button): Left=30, Top=60, Width=
⊞ mBoring	
⊞ mBufferType	
mChangeWatcher	null
mCharWrapper	null
mCurrentSpellCheckerLo	null

Hovering over a line in the pop up window shows its complete text.



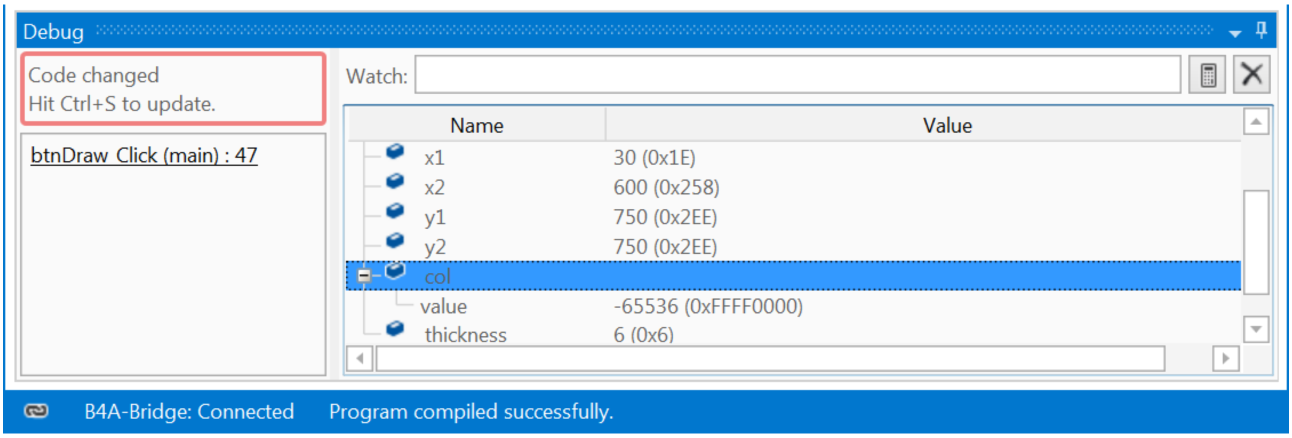
Name	Value
⊟ btnDraw	
ToString	(Button): Left=30, Top=60, Width=
mTransformed	Draw
mTransformation	null
⊞ mBoring	
⊞ mBufferType	
mChangeWatcher	null

(Button): Left=30, Top=60, Width=300, Height=150, Tag=, Text=Draw

```
** Activity (main) Pause, UserClose
** Activity (main) Create, isFirst = t
** Activity (main) Resume **
```

13.1.5 Watch Expressions feature

At the right side of the Debug window we find a TextBox Watch:

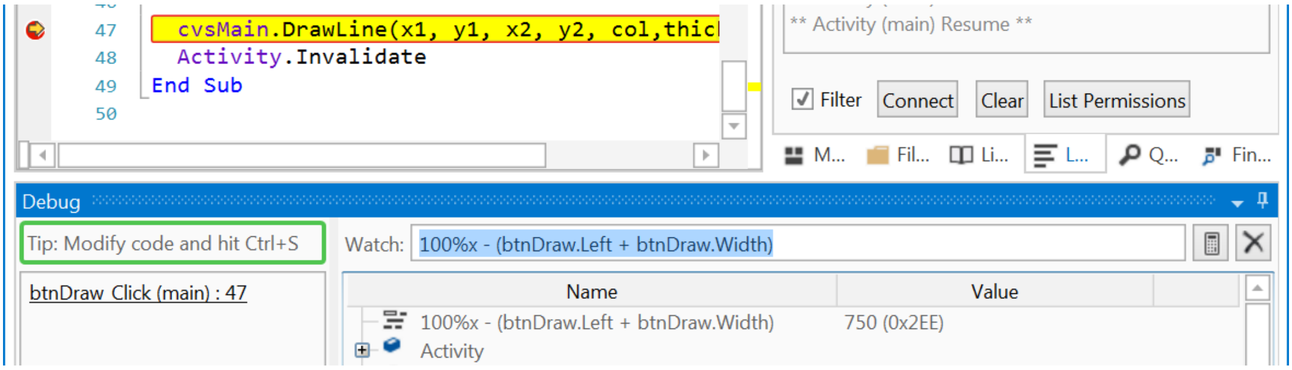


Name	Value
x1	30 (0x1E)
x2	600 (0x258)
y1	750 (0x2EE)
y2	750 (0x2EE)
col	
value	-65536 (0xFFFF0000)
thickness	6 (0x6)

To watch the Right coordinate of btnDraw we enter

`100%x – (btnDraw.Left + btnDraw.Width)` and click on 🖩.

This adds a new watch expression in the list with its value.



```
    cvsMain.DrawLine(x1, y1, x2, y2, col,thic
    Activity.Invalidate
End Sub
```

** Activity (main) Resume **
☑ Filter
Connect
Clear
List Permissions
M...
Fil...
Li...
L...
Q...
Fin...
Debug
Tip: Modify code and hit Ctrl+S
btnDraw_Click (main) : 47

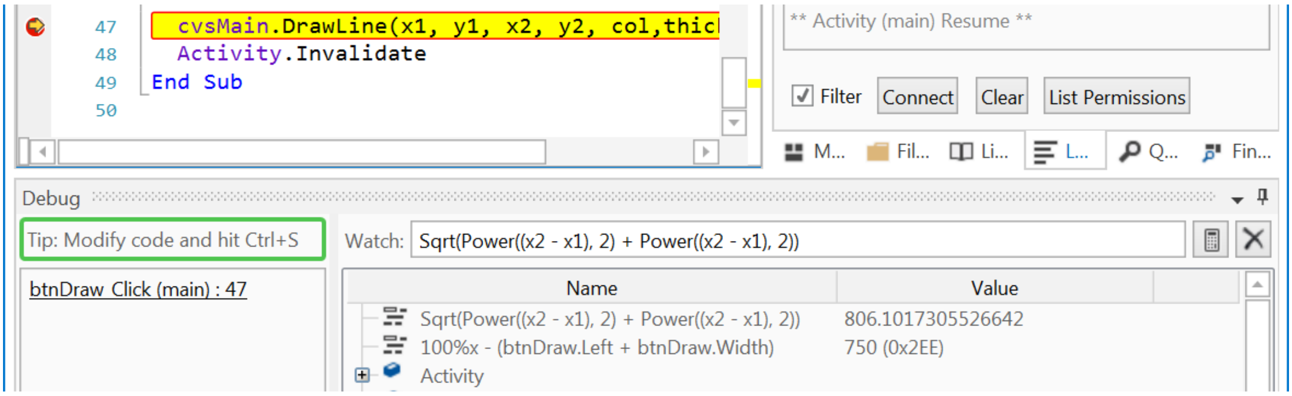
Watch: 100%x - (btnDraw.Left + btnDraw.Width)

Name	Value
100%x - (btnDraw.Left + btnDraw.Width)	750 (0x2EE)
Activity	

Clicking on ✕ removes all watch expressions.

Now we want to know the length of the line in pixels:
We add `Sqrt(Power((x2 - x1), 2) + Power((x2 - x1), 2))`

And click on 🖩 .



```
    cvsMain.DrawLine(x1, y1, x2, y2, col,thic
    Activity.Invalidate
End Sub
```

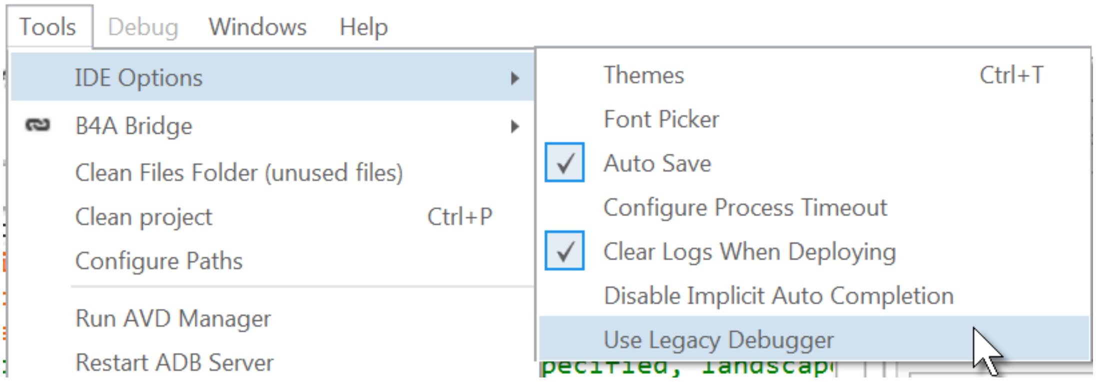
** Activity (main) Resume **
☑ Filter
Connect
Clear
List Permissions
M...
Fil...
Li...
L...
Q...
Fin...
Debug
Tip: Modify code and hit Ctrl+S
btnDraw_Click (main) : 47

Watch: Sqrt(Power((x2 - x1), 2) + Power((x2 - x1), 2))

Name	Value
Sqrt(Power((x2 - x1), 2) + Power((x2 - x1), 2))	806.1017305526642
100%x - (btnDraw.Left + btnDraw.Width)	750 (0x2EE)
Activity	

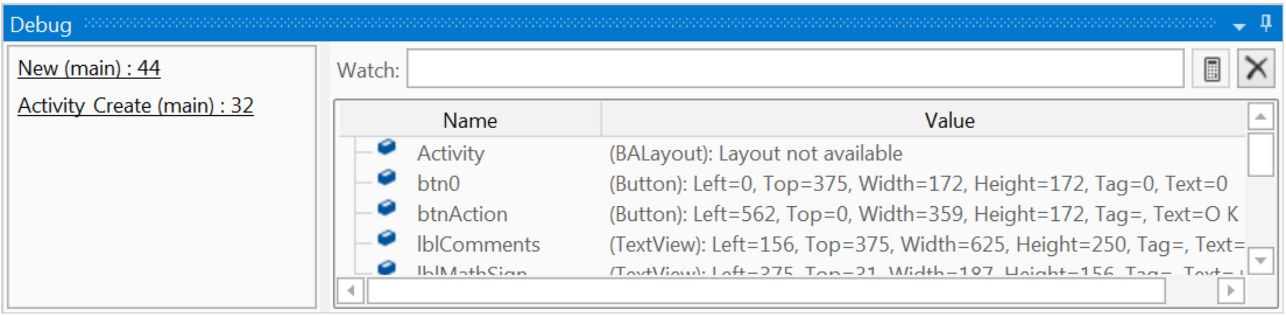
13.2 Debug (legacy) mode

In some cases the legacy Debugger can be useful, can select it in the Tools menu under IDE options.

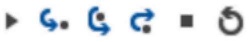


Debug(legacy): When this option is selected then the compiled code will contain debugging code. The debugging code allows the IDE to connect to the program and inspect it while it runs.
When the program starts, it will wait for up to 10 seconds for the IDE to connect. Usually the IDE will connect immediately. However if you run your program manually from the phone you will see it waiting.
The name of the compiled APK file will end with _DEBUG.apk. You should not distribute this apk file as it contains the debugging code which adds a significant overhead.
To distribute files you must select the *Release* or the *Release (obfuscated)* option.

When we run the program with the Debug (legacy) option, the IDE will open the debugger module at the bottom of the screen:



Name	Value
Activity	(BALayout): Layout not available
btn0	(Button): Left=0, Top=375, Width=172, Height=172, Tag=0, Text=0
btnAction	(Button): Left=562, Top=0, Width=359, Height=172, Tag=, Text=O K
lblComments	(TextView): Left=156, Top=375, Width=625, Height=250, Tag=, Text=
lblMathSign	(TextView): Left=375, Top=21, Width=187, Height=156, Tag=, Text=

The navigation buttons in the Toolbar are enabled  .
These work similar to the Debug (rapid) mode.

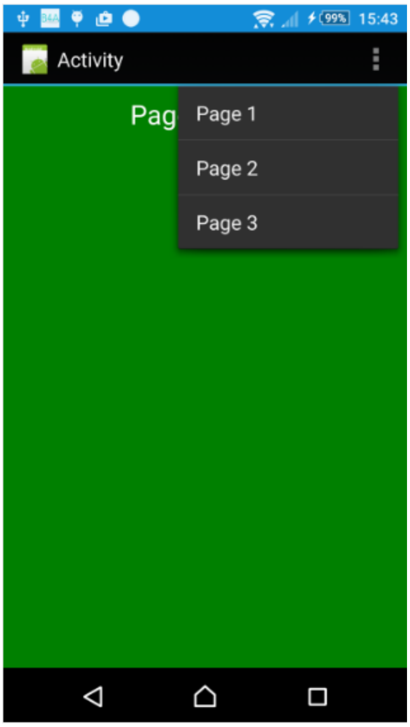
14 Example programs

14.1 User interfaces

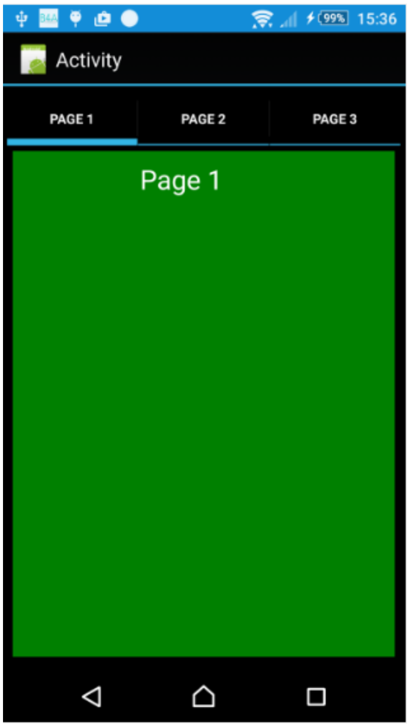
Let us make three different user interfaces to select three different screens.

The three user interfaces are:

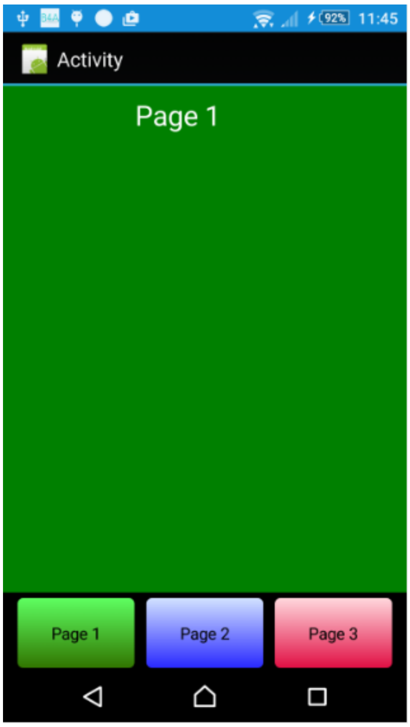
Menu



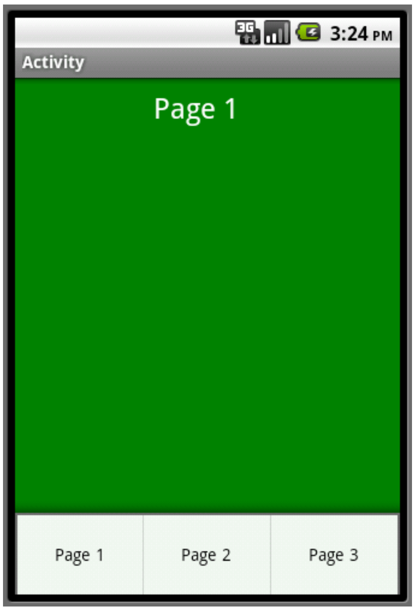
TabHost view



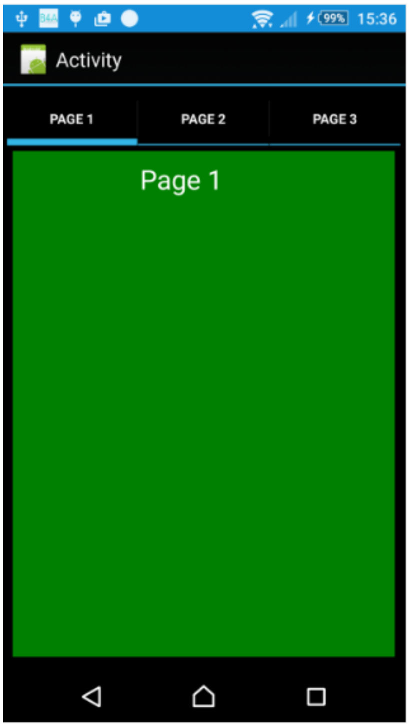
Button toolbox

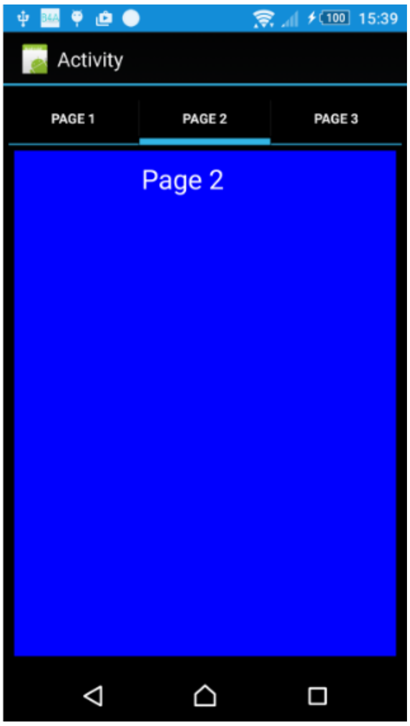


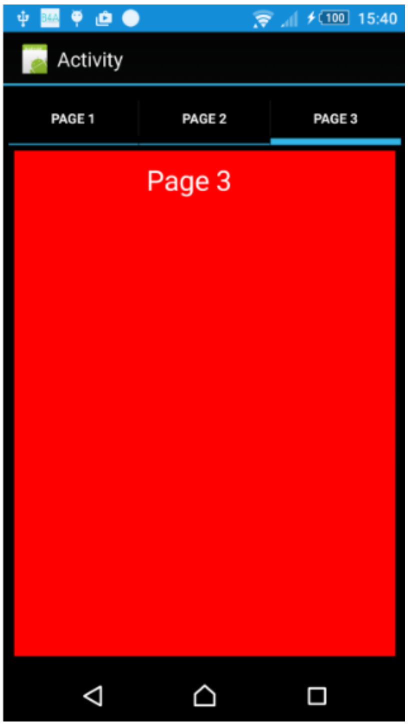
The menu layout can have different looks depending on the Android version



For each test program there is a Main layout.







For each of the three pages there are separate layout files Page1, Page2 and Page3.
Each layout file is loaded to a Panel or a TabHost panel.
These layouts can contain whatever views you need.

14.1.1 Menu example (UserInterfaceMenu.b4a)

The test program is: UserInterfaceMenu.b4a. The code is self-explanatory.

1. Each page is on a Panel, pnlPage1, pnlPage2 and pnlPage3.
2. The Panels are added by code.
3. The page layout files are loaded to the Panels.
4. The Menu items are added to the Activity.
5. One Click event routine for each Menu item.
   It could also be done in one routine (like in UserInterfaceButtonToolbox.b4a).

```
Sub Globals
   Private pnlPage1, pnlPage2, pnlPage3 As Panel    ' Declares the three panels
End Sub

Sub Activity_Create(FirstTime As Boolean)
   Activity.LoadLayout("Main")                 ' Loads "Main" layout file

   pnlPage1.Initialize("")                     ' Initializes pnlPage1
   pnlPage1.LoadLayout("Page1")                ' Loads "Page1" layout file
   Activity.AddView(pnlPage1,0,0,100%x,100%y) ' Adds pnlPage1 to Activity
   pnlPage1.Visible=True                              ' Sets pnlPage1 to Visible

   pnlPage2.Initialize("")                     ' Initializes pnlPage2
   pnlPage2.LoadLayout("Page2")                ' Loads "Page2" layout file
   Activity.AddView(pnlPage2,0,0,100%x,100%y) ' Adds pnlPage1 to Activity
   pnlPage2.Visible=False                      ' Sets pnlPage1 to Visible

   pnlPage3.Initialize("")                     ' Initializes pnlPage3
   pnlPage3.LoadLayout("Page3")                ' Loads "Page3" layout file
   Activity.AddView(pnlPage3,0,0,100%x,100%y) ' Adds pnlPage1 to Activity
   pnlPage3.Visible=False                      ' Sets pnlPage1 to Visible

   Activity.AddMenuItem("Page 1","mnuPage1")  ' Adds menu item mnuPage1
   Activity.AddMenuItem("Page 2","mnuPage2")  ' Adds menu item mnuPage2
   Activity.AddMenuItem("Page 3","mnuPage3")  ' Adds menu item mnuPage3
End Sub

Sub mnuPage1_Click
   pnlPage2.Visible = False                    ' Hides pnlPage2
   pnlPage3.Visible = False                    ' Hides pnlPage3
   pnlPage1.Visible = True                     ' Sets pnlPage1 to Visible
End Sub

Sub mnuPage2_Click
   pnlPage1.Visible = False                    ' Hides pnlPage1
   pnlPage3.Visible = False                    ' Hides pnlPage3
   pnlPage2.Visible = True                     ' Sets pnlPage2 to Visible
End Sub

Sub mnuPage3_Click
   pnlPage1.Visible = False                    ' Hides pnlPage1
   pnlPage2.Visible = False                    ' Hides pnlPage2
   pnlPage3.Visible = True                     ' Sets pnlPage3 to Visible
End Sub
```

14.1.2 TabHost example (UserInterfaceTabHost.b4a)

The test program is: UserInterfaceTabHost.b4a The code is self-explanatory.

1. Each page is on a TabHost panel.
2. The TabHost view is in the Main layout.
3. The TabHost panels are added with the Page layout files.

```
Sub Globals
   Private tbhPages As TabHost                      ' Declares the TabHost view
End Sub

Sub Activity_Create(FirstTime As Boolean)
   Activity.LoadLayout("Main")               ' Loads "Main" layout file

   tbhPages.AddTab("Page 1","Page1")         ' Adds Page1 on the first Tab
   tbhPages.AddTab("Page 2","Page2")         ' Adds Page2 on the second Tab
   tbhPages.AddTab("Page 3","Page3")         ' Adds Page3 on the third Tab
 End Sub
```

14.1.3 Button toolbox example (UserInterfaceButtonToolbox.b4a)

The test program is: UserInterfaceButtonToolbox.b4a The code is self-explanatory.

1. Each page is on a Panel, pnlPage1, pnlPage2 and pnlPage3.
2. The Panels are added by code.
3. The page layout files are loaded to the Panels.
4. The Buttons are in the Main layout on the pnlToolBox panel.
5. One Click event routine for all Buttons.

```
Sub Globals
   Private pnlPage1, pnlPage2, pnlPage3 As Panel    ' Declares the three panels
   Private pnlToolbox As Panel
End Sub

Sub Activity_Create(FirstTime As Boolean)
   Private PanelHeight As Float

   Activity.LoadLayout("Main")                ' Loads "Main" layout file
                                              ' Calculates the top of the     Toolbox
   pnlToolbox.Top = Activity.Height - pnlToolbox.Height
   PanelHeight = pnlToolbox.Top - 5dip        ' Calculates the Panel height

   pnlPage1.Initialize("")                    ' Initializes pnlPage1
   pnlPage1.LoadLayout("Page1")               ' Loads "Page1" layout file
   Activity.AddView(pnlPage1,0,0,100%x,PanelHeight) ' Adds pnlPage1
   pnlPage1.Visible=True                      ' Sets pnlPage1 to Visible

   pnlPage2.Initialize("")                    ' Initializes pnlPage2
   pnlPage2.LoadLayout("Page2")               ' Loads "Page2" layout file
   Activity.AddView(pnlPage2,0,0,100%x,PanelHeight) ' Adds pnlPage2
   pnlPage2.Visible=False                     ' Sets pnlPage1 to Visible

   pnlPage3.Initialize("")                    ' Initializes pnlPage3
   pnlPage3.LoadLayout("Page3")               ' Loads "Page3" layout file
   Activity.AddView(pnlPage3,0,0,100%x,PanelHeight) ' Adds pnlPage3
   pnlPage3.Visible=False                     ' Sets pnlPage1 to Visible
End Sub

Sub btnPage_Click
   Private Send As Button              ' Declares Send as a Button

   Send = Sender                       ' Sets Sender to Send
                                       ' Sender is the view that raised the event
   pnlPage1.Visible=False              ' Hides pnlPage1
   pnlPage2.Visible=False              ' Hides pnlPage2
   pnlPage3.Visible=False              ' Hides pnlPage3

   Select Send.Tag                     ' Selects the buttons tag
   Case "1"                            ' If Tag = 1, btnPage1
     pnlPage1.Visible=True             ' Sets pnlPage1 visible
   Case "2"                            ' If Tag = 2, btnPage1
     pnlPage2.Visible=True             ' Sets pnlPage2 visible
   Case "3"                            ' If Tag = 3, btnPage1
     pnlPage3.Visible=True             ' Sets pnlPage3 visible
   End Select
End Sub
```

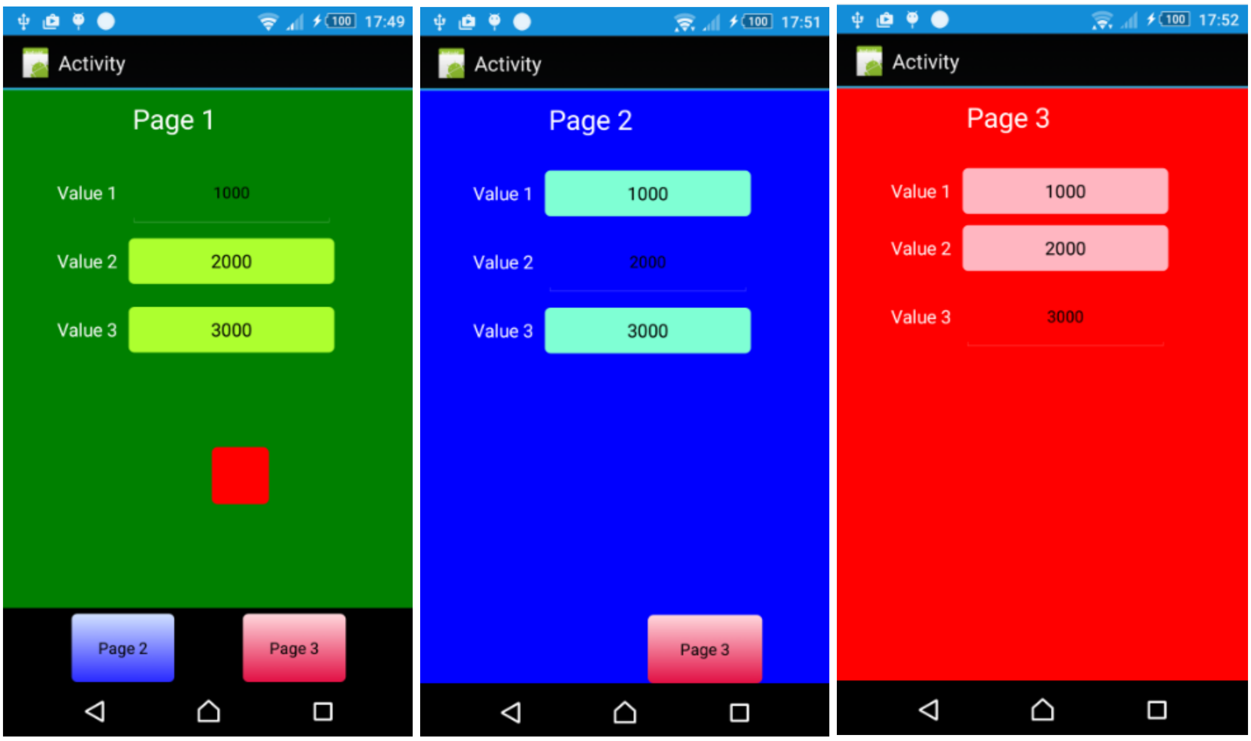
14.2 Program with 3 Activities (ThreeActivityExample.b4a)

The test program is: ThreeActivityExample.b4a

The goal of the program is:

- to show how to manage several Activities.
- working with Process Global variables across different Activities. The variables can be changed in different activities, but are available over the whole project.
- change layout properties, in moving a small red panel over the screen.
- save and load the layout properties of the small red panel with a Map object so the square will keep the same position after changing a page or restarting the program.

The program looks like below:



Page 1
Value 1 1000
Value 2 2000
Value 3 3000

Page 2
Value 1 1000
Value 2 2000
Value 3 3000

Page 3
Value 1 1000
Value 2 2000
Value 3 3000

We have:

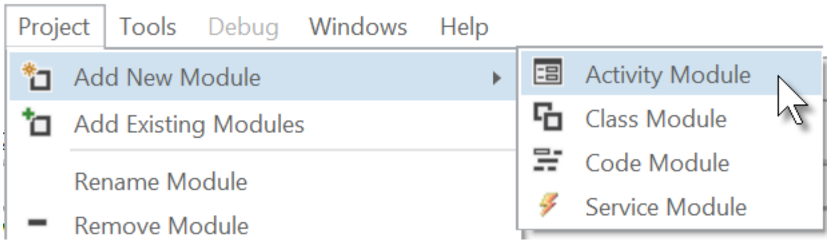
- 3 pages, each one in its own Activity.
- 3 process global variables, Value1, Value2 and Value3
- on each page 1 EditText view to modify the Value variable with page index.
- 2 Labels to display the two other variables.
- on Page1 a small red square Panel to move around.

We can:

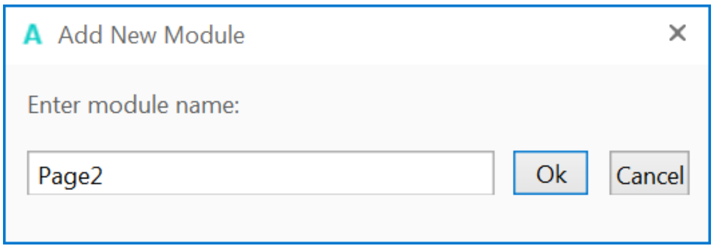
- Change Value1 in Page1.
- Change Value2 in Page2.
- Change Value3 in Page3.
- Move the small red square over the screen.
- Select either Page2 or Page3 on Page1.

Let us take the example with the Button toolbox (UserInterfaceButtonToolbox.b4a).
Instead of having our three pages on three panels we will use 3 activities.
Main, Page2 and Page 3.

For this we must create two new Activity Modules: Page2 and Page3.



In the IDE click on Activity Module.

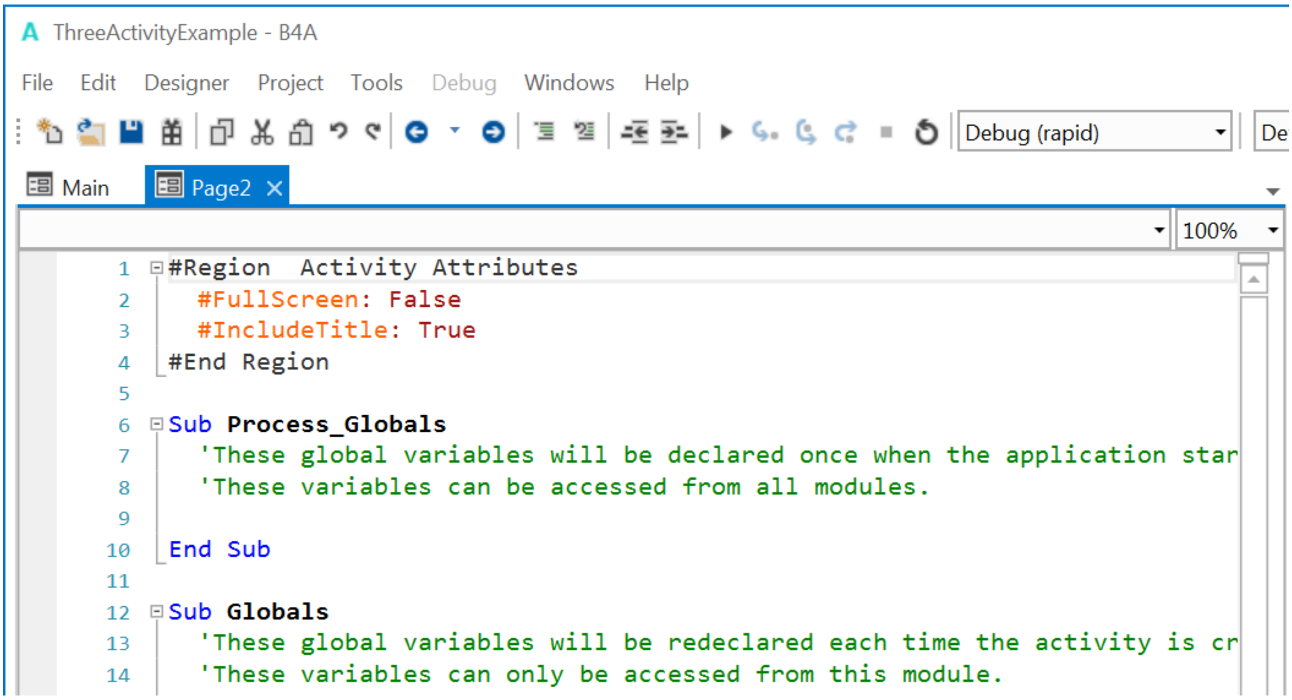


Enter module name: Page2

Enter the name Page2

and click on Ok.

A new module is added to the project.



```
#Region  Activity Attributes
	#FullScreen: False
	#IncludeTitle: True
#End Region

Sub Process_Globals
	'These global variables will be declared once when the application star
	'These variables can be accessed from all modules.

End Sub

Sub Globals
	'These global variables will be redeclared each time the activity is cr
	'These variables can only be accessed from this module.
```

Modify the code of Page2 module as below:

```
'Activity module
Sub Process_Globals

End Sub

Sub Globals
  Private lblValue1_P2, lblValue3_P2 As Label
  Private edtValue2_P2 As EditText
End Sub

Sub Activity_Create(FirstTime As Boolean)
  Activity.LoadLayout("Page2")        ' Loads "Page2" layout file
End Sub

Sub Activity_Resume
  lblValue1_P2.Text = Main.Value1     ' Sets Main.Value1 to lblValue1_P2.Text
  edtValue2_P2.Text = Main.Value2     ' Sets Main.Value2 to edtValue2_P2.Text
  lblValue3_P2.Text = Main.Value3     ' Sets Main.Value3 to lblValue3_P2.Text
End Sub

Sub Activity_Pause (UserClosed As Boolean)
  Main.Value2 = edtValue2_P2.Text     ' Sets edtValue2_P2.Text to the
End Sub                               ' Process_Global variable  Value2
```

Add now a new module "Page3" the same way as Page2 and modify the code like below:

```
'Activity module
Sub Process_Globals

End Sub

Sub Globals
  Private lblValue1_P3, lblValue2_P3 As Label
  Private edtValue3_P3 As EditText
End Sub

Sub Activity_Create(FirstTime As Boolean)
  Activity.LoadLayout("Page3")        ' Loads "Page3" layout file
End Sub

Sub Activity_Resume
  lblValue1_P3.Text = Main.Value1     ' Sets Main.Value1 to lblValue1_P3.Text
  lblValue2_P3.Text = Main.Value2     ' Sets Main.Value2 to lblValue2_P3.Text
  edtValue3_P3.Text = Main.Value3     ' Sets Main.Value3 to edtValue3_P3.Text
End Sub

Sub Activity_Pause (UserClosed As Boolean)
  Main.Value3 = edtValue3_P3.Text     ' Sets edtValue3_P2.Text to the
End Sub                               ' Process_Global variable  Value3
```

These codes are self-explanatory.

Let us modify the code of the Main module:

In Sub Process-Globals we add following variables.
Value1, Value2 and Value3 to save some values.
mapMoveTopLeft as a Map object to save the Left and Top parameter of the small red square.

```
Sub Process_Globals
  Public Value1, Value2, Value3 As String    ' Declares the Value variables
                                              ' as Process_Global variables
  Public mapMoveTopLeft As Map                ' Declares the Map as Process_Global
End Sub
```

In Globals we have the variables below:
lblValue2_P1 is the Label to display Value2 on page1.
lblValue3_P1 is the Label to display Value3 on page1.
edtValue1_P1 is the EditText to enter Value1 on page1.
pnlPage1 is the container for the Page1 layout.
pnlMove is the small red square.
X0, Y0, X1 and Y1 are used to memorize initial coordinates when moving the red square.

```
Sub Globals
  Private lblValue2_P1, lblValue3_P1 As Label      ' Declares the Views
  Private edtValue1_P1 As EditText
  Private pnlPage1 As Panel
  Private pnlToolbox, pnlMove As Panel

  Private X0, Y0, X1, Y1 As Float                  ' Coordinate variables
End Sub
```

Sub Activity_Create is modified like below:

When the routine is called for the first time, we initialize the three Value variables.

```
Sub Activity_Create(FirstTime As Boolean)
  Private PanelHeight As Float

  Activity.LoadLayout("Main")                      ' Loads "Main" layout file
    ' Calculates the top of theToolbox
  pnlToolbox.Top = Activity.Height - pnlToolbox.Height
  PanelHeight = pnlToolbox.Top - 5dip              ' Calculates the Panel height
  pnlPage1.Initialize("")                          ' Initializes pnlPage1
  pnlPage1.LoadLayout("Page1")                     ' Loads "Page1" layout file
  Activity.AddView(pnlPage1,0,0,100%x,PanelHeight) ' Adds pnlPage1

  If FirstTime = True Then      ' If Activity_Create is called the first time
    Value1 = 1000               ' we initialize the three Values
    Value2 = 2000
    Value3 = 3000
  End If
End Sub
```

In Sub Activity_Resume we initialize the properties of the views of Activity Main.
Init.txt is the file with the Left and Top properties of the small red square pnlMove.
If the file exists we read it and set the Left and Top properties of pnlMove.
If the file doesn't exist we initialize the Map object and set the two first properties to the Left and Top properties of pnlMove.

```
Sub Activity_Resume
   edtValue1_P1.Text = Value1             ' Attribues Value1 to edtValue1_P1.Text
   lblValue2_P1.Text = Value2
   lblValue3_P1.Text = Value3
   'If the file Init.txt exists, we read it.
   'It contains the values of the Left and Top properties of pnlMove
   If File.Exists(File.DirInternal,"Init.txt") Then
      mapMoveTopLeft = File.ReadMap(File.DirInternal,"Init.txt")
      pnlMove.Left = mapMoveTopLeft.Get("Left")      ' set pnlMove.Left parameter
      pnlMove.Top = mapMoveTopLeft.Get("Top")        ' set pnlMove.Top parameter
   Else                                              ' If the file doesn't exsit
      mapMoveTopLeft.Initialize                      ' We initialize the Map
      mapMoveTopLeft.Put("Left",pnlMove.Left)        ' Setting the Left parameter
      mapMoveTopLeft.Put("Top",pnlMove.Top)          ' Setting the Top parameter
   End If
End Sub
```

When the "Main" Activity is paused, due to either a page change or the program close, we:

- set variable Value1 to the edtValue1.Text content.
- save the Map to file Init.txt.

```
Sub Activity_Pause (UserClosed As Boolean)
   Value1 = edtValue1_P1.Text                      ' get Value1 from edtValue1_P1.Text
   File.WriteMap(File.DirInternal,"Init.txt",mapMoveTopLeft) ' Saves the Map
End Sub
```

To go back to Page1 from either Page2 or Page3, the user must press the Back key. To avoid that the program stops when the user clicks, by inadvertence, one time too much, we check in Sub Activity_KeyPress what key was pressed. And if it's the Back key we display a message in a MessageBox asking the user if he really wants to quit the program. If Yes, then we set the Return value to False that means that the event is sent back to the OS to close the program. If the answer is No, we set the Return value to True, that means that we 'consume' the event and the OS will not stop the program.

```
Sub Activity_KeyPress(KeyCode As Int) As Boolean
   Private Answ As Int
   Private Txt As String

   If KeyCode = KeyCodes.KEYCODE_BACK Then' Checks if the KeyCode is BackKey
      Txt = "Do you really want to quit the program ?"
      Answ = Msgbox2(Txt,"A T T E N T I O N","Yes","","No",Null) ' MessageBox
      If Answ = DialogResponse.POSITIVE Then  ' If return value is Yes then
         Return False       ' Return = False  the Event will not be consumed
      Else                  '                 we leave the program
         Return True        ' Return = True   the Event will be consumed to avoid
      End If                '                 leaving the program
   End If
End Sub
```

To show how to manage layout properties we have the small red square, pnlMove, which can be moved on the screen. The position of pnlMove is handled in Sub Activity_Touch where we get three parameters:

- Action holding the value of the action the user made.
  `ACTION_DOWN` the user touches the screen.
  `ACTION_MOVE` the user moves on the screen
  `ACTION_UP` the user leaves the screen
- X the X coordinate of the finger on the screen.
- Y the Y coordinate of the finger on the screen.

To be able to move pnlMove we do the following:

- when Action is equal to ACTION_DOWN, the user touches the screen
  we memorize the coordinates of the finger and the coordinates of the upper left corner of pnlMove (lines 76 to 79).

- when Action is equal to ACTION_MOVE the user moves his finger on the screen,
  we calculate the relative displacement, dX and dY, in both directions and set the new Left and Top properties of pnlMove (lines 82 to 85).

- when Action is equal to ACTION_UP, the user leaves the screen and
  we update the two properties in the Map object (lines 88 and 89=.

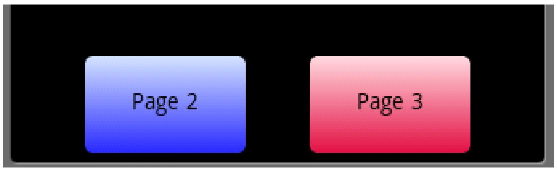
```
Sub Activity_Touch (Action As Int, X As Float, Y As Float)
  Private dX, dY As Float

  Select Action                        ' Selects the Action parameter
  Case Activity.ACTION_DOWN            ' Checks if ACTION_DOWN
    X0 = X                             ' Memorizes the X coordinate
    Y0 = Y                             ' Memorizes the Y coordinate
    X1 = pnlMove.Left                  ' Memorizes the Left coordinate
    Y1 = pnlMove.Top                   ' Memorizes the Top coordinate

  Case Activity.ACTION_MOVE            ' Checks if ACTION_MOVE
    dX = X - X0                        ' Calculates the X distance moved
    dY = Y - Y0                        ' Calculates the X distance moved
    pnlMove.Left = X1 + dX             ' Sets the new Left coordinate
    pnlMove.Top = Y1 + dY              ' Sets the new Top coordinate

  Case Activity.ACTION_UP              ' Checks if ACTION_UP
    mapMoveTopLeft.Put("Left",pnlMove.Left) ' Memorizes Left in the Map
    mapMoveTopLeft.Put("Top",pnlMove.Top)   ' Memorizes Top in the Map
  End Select
End Sub
```

In Sub btnPage_Click we start the Page Activity according to what button was pressed.

- We declare a new Button object, Send.
- We attribute Sender to Send.
  Sender is the button view that raised the event.
- Depending on the Tag value of the sender object we start the correct Activity.

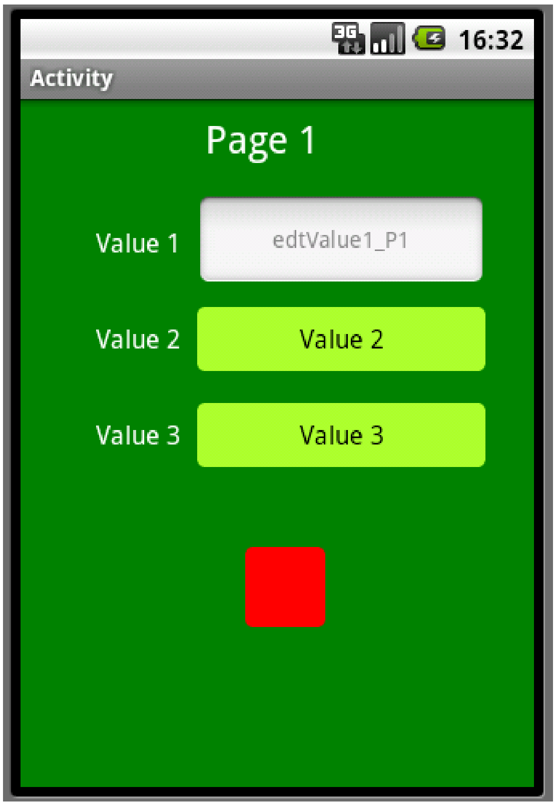
```
Sub btnPage_Click
   Private Send As Button         ' Declares Send as a Button

   Send = Sender                  ' Sets Sender to Send
                                  ' Sender is the view that raised the event
   Select Send.Tag                ' Selects the buttons tag
   Case "2"                       ' If Tag = 2, btnPage2
      StartActivity("Page2")      ' Calls Page2 Activity
   Case "3"                       ' If Tag = 3, btnPage3
      StartActivity("Page3")      ' Calls Page3 Activity
   End Select
End Sub
```

We still need to modify the four layout files:



Main:

We remove btnPage1, as it is no longer needed.
Enlarge the two remaining buttons and reposition them



Page 1:

We add the views like in the image at the left.

Similar for Page 2 and Page 3.

The layout files are in the project.

14.3 ScrollView examples

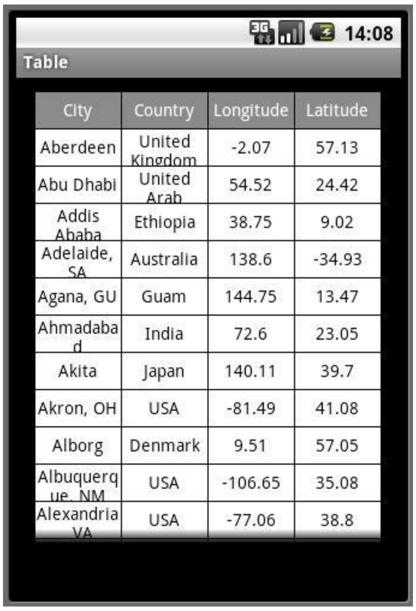
ScrollView is a very versatile view to display lists of objects holding data or user interface views.

ListViews are, currently, limited to two lines of text and an image per data set.

ScrollViews have an internal Panel, bigger than the screen, which can be scrolled vertically and holds any type of views either as one layout or as lists of view sets.

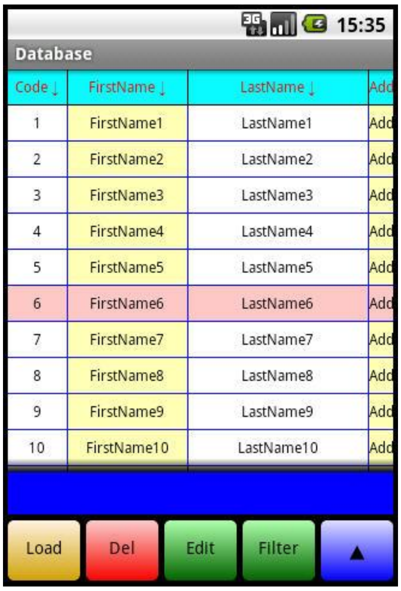
Some screenshots of examples: (a summary of ScrollView examples)

Gridline in TableView ...Scrollview



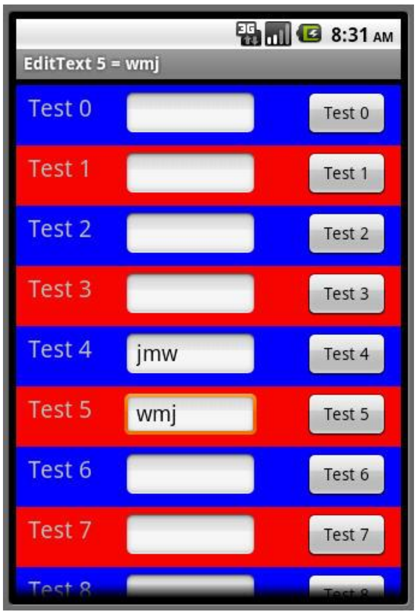
City	Country	Longitude	Latitude
Aberdeen	United Kingdom	-2.07	57.13
Abu Dhabi	United Arab	54.52	24.42
Addis Ababa	Ethiopia	38.75	9.02
Adelaide, SA	Australia	138.6	-34.93
Agana, GU	Guam	144.75	13.47
Ahmadabad	India	72.6	23.05
Akita	Japan	140.11	39.7
Akron, OH	USA	-81.49	41.08
Alborg	Denmark	9.51	57.05
Albuquerque, NM	USA	-106.65	35.08
Alexandria VA	USA	-77.06	38.8

SQLLiteDB



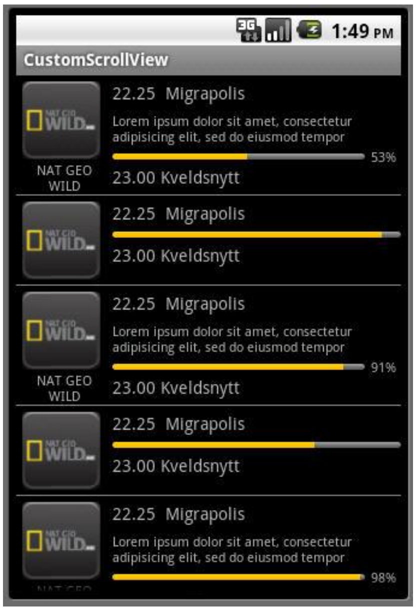
Code ↓	FirstName ↓	LastName ↓	Add
1	FirstName1	LastName1	Add
2	FirstName2	LastName2	Add
3	FirstName3	LastName3	Add
4	FirstName4	LastName4	Add
5	FirstName5	LastName5	Add
6	FirstName6	LastName6	Add
7	FirstName7	LastName7	Add
8	FirstName8	LastName8	Add
9	FirstName9	LastName9	Add
10	FirstName10	LastName10	Add

ScrollView, layouts ...



Test 0		Test 0
Test 1		Test 1
Test 2		Test 2
Test 3		Test 3
Test 4	jmw	Test 4
Test 5	wmj	Test 5
Test 6		Test 6
Test 7		Test 7
Test 8		Test 8

Another ScrollView example



22.25 Migrapolis
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
53%
23.00 Kveldsnytt

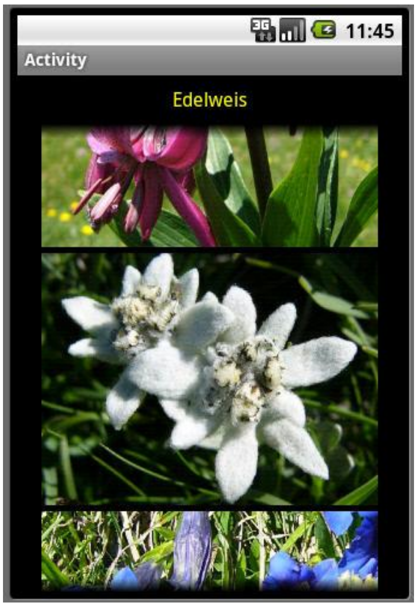
22.25 Migrapolis
23.00 Kveldsnytt

22.25 Migrapolis
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
91%
23.00 Kveldsnytt

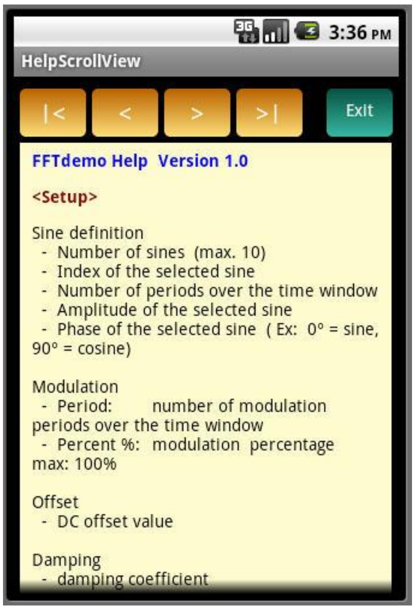
22.25 Migrapolis
23.00 Kveldsnytt

22.25 Migrapolis
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
98%

Add imageview ...



HelpScrollView



FFTdemo Help Version 1.0

<Setup>

Sine definition
- Number of sines (max. 10)
- Index of the selected sine
- Number of periods over the time window
- Amplitude of the selected sine
- Phase of the selected sine (Ex: 0° = sine, 90° = cosine)

Modulation
- Period: number of modulation periods over the time window
- Percent %: modulation percentage max: 100%

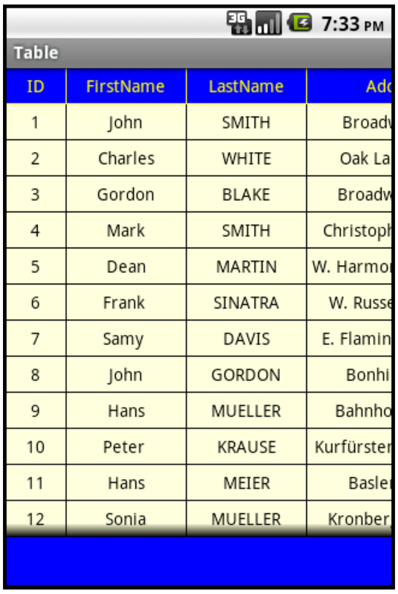
Offset
- DC offset value

Damping
- damping coefficient

14.3.1 ScrollView example program

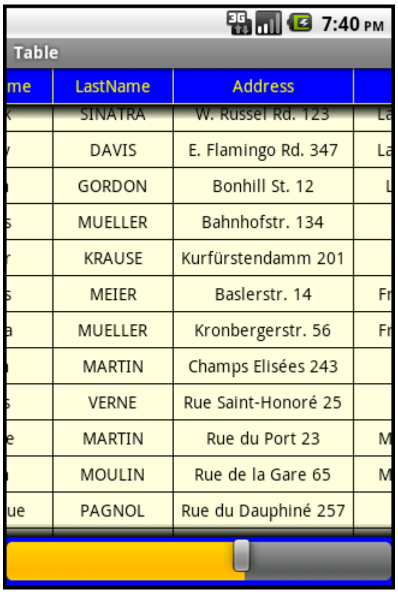
Let us make a ScrollView example with following functions:

The source code is in the ScrollViewExample folder.



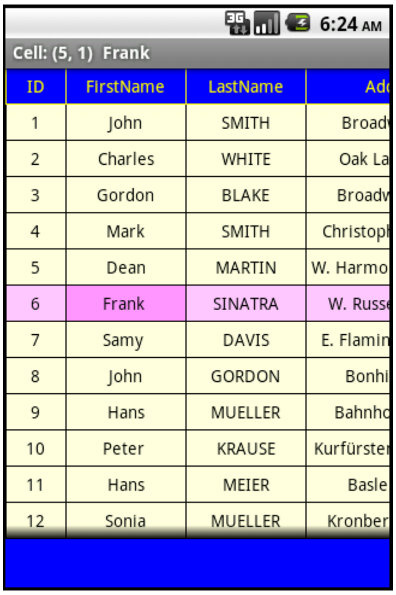
Table

ID	FirstName	LastName	Add
1	John	SMITH	Broadv
2	Charles	WHITE	Oak La
3	Gordon	BLAKE	Broadw
4	Mark	SMITH	Christoph
5	Dean	MARTIN	W. Harmo
6	Frank	SINATRA	W. Russe
7	Samy	DAVIS	E. Flamin
8	John	GORDON	Bonhi
9	Hans	MUELLER	Bahnho
10	Peter	KRAUSE	Kurfürster
11	Hans	MEIER	Basle
12	Sonia	MUELLER	Kronber



Table

me	LastName	Address	
k	SINATRA	W. Russel Rd. 123	La
y	DAVIS	E. Flamingo Rd. 347	La
	GORDON	Bonhill St. 12	L
s	MUELLER	Bahnhofstr. 134	
r	KRAUSE	Kurfürstendamm 201	
s	MEIER	Baslerstr. 14	Fr
a	MUELLER	Kronbergerstr. 56	Fr
	MARTIN	Champs Elisées 243	
s	VERNE	Rue Saint-Honoré 25	
e	MARTIN	Rue du Port 23	M
	MOULIN	Rue de la Gare 65	M
ue	PAGNOL	Rue du Dauphiné 257	



Cell: (5, 1) Frank

ID	FirstName	LastName	Add
1	John	SMITH	Broad
2	Charles	WHITE	Oak La
3	Gordon	BLAKE	Broadv
4	Mark	SMITH	Christopl
5	Dean	MARTIN	W. Harmo
6	Frank	SINATRA	W. Russe
7	Samy	DAVIS	E. Flamin
8	John	GORDON	Bonhi
9	Hans	MUELLER	Bahnhc
10	Peter	KRAUSE	Kurfürste
11	Hans	MEIER	Basle
12	Sonia	MUELLER	Kronber

- Read a csv file and display it in a table based on a ScrollView.
- The ScrollView can be scrolled vertically with the standard scrolling function of the ScrollView.
- The ScrollView can also be scrolled horizontally with a Seekbar or dynamically with the finger on the lower blue rectangle (SeekBar visible or not).
- Clicking on a cell highlights the row and the cell, this routine allows adding other functions related to a row or a cell.
- Clicking on a header, displays the column, this routine allows adding functions related to a column.

We define the following variables and assign their default values:

`StringUtils1`	Used to read the csv file.
`NumberOfColumns`	Number of columns.
`RowHeight`	Height of a row in the ScrollView.
`RowLineWidth`	Width of the lines between the rows.
`RowHeight_1`	Internal height of a row `RowHeight_1 = RowHeight - RowLineWidth`
`ColLineWidth`	Width of the lines between the columns.
`ColumnWidth()`	Width of the different columns as an array
`ColumnWidth_1()`	Internal width of the different columns.
`TotalColumnWidth()`	Coordinates of the left border of a column as an array.
`HeaderColor`	Headers background color.
`HeaderFontColor`	Headers font color.
`HeaderLineColor`	Headers line color.
`LineColor`	Cell line color.
`CellColor`	Cell background line color.
`FontColor`	Cell font line color.
`Alignment`	Text alignment of the text in the headers and cells.
`SelectedRow`	Index of the selected row.
`SelectedRowColor`	Color of the selected row.
`SelectedCellColor`	Color of the selected cell.
`Type RowCol (Row As Int, Col As Int)`	Define a custom variable that contains a row and a column index.
`MoveLeft0`	Used for the horizontal scrolling.
`MoveX0`	Used for the horizontal scrolling.
`MoveX1`	Used for the horizontal scrolling.
`DeltaScroll`	Used for the horizontal scrolling.
`DeltaX`	Used for the horizontal scrolling.
`Time0`	Used for the horizontal scrolling.

Personally, I prefer working with variables rather than with values. The maintenance and modification of a program is much easier with variables than with numerical values.

```
Sub Process_Globals
   Public StringUtils1 As StringUtils

   Public NumberOfColumns = 5 As Int
   Public NumberOfRows As Int
   Public RowHeight = 30dip As Int
   Public RowLineWidth = 1dip As Int
   Public RowHeight_1 = RowHeight - RowLineWidth As Int
   Public ColLineWidth = 1dip As Int
   Public ColumnWidth(NumberOfColumns) As Int
      ColumnWidth(0) = 50dip
      ColumnWidth(1) = 100dip
      ColumnWidth(2) = 100dip
      ColumnWidth(3) = 150dip
      ColumnWidth(4) = 100dip
   Public ColumnWidth_1(NumberOfColumns) As Int
   Public TotalColumnWidth(NumberOfColumns + 1) As Int

   Public HeaderColor = Colors.Blue As Int
   Public HeaderFontColor = Colors.Yellow As Int
   Public HeaderLineColor = Colors.Yellow As Int
   Public LineColor = Colors.Black As Int
   Public CellColor = Colors.RGB(255,255,220) As Int
   Public FontColor = Colors.Black As Int
   Public FontSize = 14 As Float
   Public Alignment = Gravity.CENTER As Int
   Public SelectedRow = -1 As Int
   Public SelectedRowColor = Colors.RGB(255,196,255) As Int
   Public SelectedCellColor = Colors.RGB(255,150,255) As Int
   Type RowCol (Row As Int, Col As Int)
   Public MoveLeft0, MoveX0, MoveX1, DeltaScroll, DeltaX As Float
   Public Time0 As Long

   Public Timer1 As Timer
End Sub
```

Now we define the Views for the program:

`scvPersons`	ScrollView to display the data.
`pnlHeader`	Panel to display the headers.
`skbScroll`	Seekbar to scroll the ScrollView and Header.
`pnlScroll`	Panel for the 'dynamic' horizontal scrolling.
`Timer1`	Timer used for the 'dynamic' horizontal scrolling.

```
Sub Globals
   Private scvPersons As ScrollView
   Private pnlHeader As Panel
   Private skbScroll As SeekBar
   Private pnlScroll As Panel
End Sub
```

Now we initialize the different Views and variables, we:

- Initialize the panel for the horizontal scrolling.
- Initialize the SeekBar for the horizontal scrolling.
- Initialize the ScrollView
- Initialize the internal column width and the left coordinates for each column and the total width of all columns.
- Initialize the ScrollView width.
- Set the Max parameter for the Seekbar
- Set the index of the selected row to -1, no row selected.
- Load the csv file, set the headers and fill the ScrollView.
- Initialize the Timer for the horizontal scrolling.

```
Sub Activity_Create(FirstTime As Boolean)
  Private i As Int

  pnlScroll.Initialize("pnlScroll")          ' initialize the scroll panel
  Activity.AddView(pnlScroll, 0, Activity.Height - 40dip, 100%x, 40dip)
  pnlScroll.Color = Colors.Blue

  skbScroll.Initialize("skbScroll")          ' initilaize the Seekbar
  Activity.AddView(skbScroll, 0, Activity.Height - 40dip, 100%x, 40dip)
  skbScroll.Visible = True

  scvPersons.Initialize(0)                   ' initialize the ScrollView
  scvPersons.Panel.Color = LineColor
  Activity.AddView(scvPersons,0,RowHeight,100%x,pnlScroll.Top-RowHeight)

  ' initialze the internal column width and left coordinates
  TotalColumnWidth(0) = ColLineWidth
  For i = 0 To NumberOfColumns - 1
    ColumnWidth_1(i) = ColumnWidth(i) - ColLineWidth
    TotalColumnWidth(i + 1) = TotalColumnWidth(i) + ColumnWidth(i)
  Next

  ' initializes the ScrollView width
  scvPersons.Width = TotalColumnWidth(NumberOfColumns)

  ' initializes the Seekbar max value
  skbScroll.Max = scvPersons.Width - Activity.Width

  SelectedRow = -1                     ' sets the selected row index to -1

  ' loads the csv file
  LoadTableFromCSV(File.DirAssets, "persons.csv", True)
' SaveTableToCSV(File.DirRootExternal, "persons.csv")

  Timer1.Initialize("Timer1",100)
End Sub
```

Then we read the csv file, fill the headers and the table (ScrollView).

- First, if the headers exist, we read the csv file with the headers.
- Or, if the headers do not exist, we read the csv file without the headers and set the default header names to Col1, Col2 etc.
- Get the number of columns.
- Display the headers `SetHeader(h)`.
- Display the table, by adding the different rows to the ScrollView `AddRow(row)`.

```
Sub LoadTableFromCSV(Dir As String,Filename As String,HeadersExist As Boolean)
  ClearAll  'Clears the previous table and loads the CSV file to the table
  Private List1 As List
  Private h() As String
  If HeadersExist Then
    ' Reads the csv file
    Private headers As List
    List1 = StringUtils1.LoadCSV2(Dir, Filename, ",", headers)
    ' Sets the header names of the columns
    Private h(headers.Size) As String
    For i = 0 To headers.Size - 1
      h(i) = headers.Get(i)
    Next
  Else
    ' Reads the csv file
    List1 = StringUtils1.LoadCSV(Dir, Filename, ",")
    ' Sets default header names
    Private firstRow() As String
    firstRow = List1.Get(0)
    Private h(firstRow.Length)
    For i = 0 To firstRow.Length - 1
      h(i) = "Col" & (i + 1)
    Next
  End If
  NumberOfColumns = h.Length    ' Gets the number of columns
  SetHeader(h)             ' Sets the headers

  NumberOfRows = 0
  For i = 0 To List1.Size - 1
    ' Fills the table
    Private row() As String
    row = List1.Get(i)
    AddRow(row)
  Next
End Sub
```

To display the headers we:

- Initialize the header panel.
- Set the header panel color to the header line color.
- Initialize a Label for each column name.
- Set the different properties for the labels.
- Add the Labels onto the header panel.
- Add the header panel to the Activity

```
Sub SetHeader(Values() As String)
   'Set the headers values
   If pnlHeader.IsInitialized Then Return 'should only be called once
   pnlHeader.Initialize("")
   pnlHeader.Color = HeaderLineColor
   For i = 0 To NumberOfColumns - 1
      Private l As Label
      l.Initialize("Header")
      l.Text = Values(i)
      l.Gravity = Gravity.CENTER
      l.TextSize = FontSize
      l.Color = HeaderColor
      l.TextColor = HeaderFontColor
      l.Tag = i
      pnlHeader.AddView(l,TotalColumnWidth(i),0,ColumnWidth_1(i),RowHeight_1)
   Next
   Activity.AddView(pnlHeader,scvPersons.Left,0,scvPersons.Width,RowHeight)
End Sub
```

Filling a row of the ScrollView with the AddRow routine:

- First we check if the number of cells is equal to the number of columns.
- Initialize a Label for each cell in the row.
- Set the different properties of the cell.
- Initialize a RowCol variable, rc, for the label tag.
- Set rc.Row to the row index and rc.Col to the column index.
- Set the label tag to rc.
- Add each label to the ScrollView.
- Set the height of the internal panel of the ScrollView.

```
Sub AddRow(Values() As String)
   'Adds a row to the table
   If Values.Length <> NumberOfColumns Then
      Log("Wrong number of values.")
      Return
   End If

   For i = 0 To NumberOfColumns - 1
      Private l As Label
      l.Initialize("cell")
      l.Text = Values(i)
      l.Gravity = Alignment
      l.TextSize = FontSize
      l.TextColor = FontColor
      l.Color=CellColor
      Private rc As RowCol
      rc.Initialize
      rc.Col = i
      rc.Row = NumberOfRows
      l.Tag = rc
      scvPersons.Panel.AddView(l,TotalColumnWidth(i), RowHeight * NumberOfRows, _
      ColumnWidth_1(i), RowHeight_1)
   Next
   NumberOfRows = NumberOfRows + 1
   scvPersons.Panel.Height = NumberOfRows * RowHeight
End Sub
```

Note: an underscore character at the end of a line means 'continue same instruction next line'.

Other functions:

- Cell_Click
  Click event of one of the cells in the table.
    - Declare rc as a RowCol variable and declare l as a Label
    - Set l equal to the Sender, the View that raised the event
    - Set rc equal to the Sender Tag parameter
    - Call the SelectRow routine
    - Display in the Activities title the row and column indexes and the cell content.

```
Sub Cell_Click
  Private rc As RowCol
  Private l As Label

  l = Sender
  rc = l.Tag
  SelectRow(rc)
  Activity.Title = "Cell: ("&rc.Row&", "& rc.Col&") "&GetCell(rc.Row, rc.Col)
End Sub
```

- Header_Click
  Click event of one of the header cells in the table.
    - Declare l as a Label and declare col as an integer.
    - Set I equal to the Sender.
    - Set col equal to the Sender Tag parameter, which is the column index.
    - Display the selected column in the Activity title.

```
Sub Header_Click
  Private l As Label
  Private col As Int

  l = Sender
  col = l.Tag
  Activity.Title = "Header clicked: " & col
End Sub
```

- SelectRow
  This routine manages the colors of the selected row and cell.
  It is called from the Cell_Click routine
  - Declare col as an integer.
  - If there is a row selected, set the normal cell color.
  - Set the SelectedRow variable to the new selected row index.
  - Set the selected row and selected cell colors.

```
Sub SelectRow(rc As RowCol)
   Private col As Int

   'Removes the color of previously selected row
   If SelectedRow > -1 Then
      For col = 0 To NumberOfColumns - 1
         GetView(SelectedRow, col).Color = CellColor
      Next
   End If

   SelectedRow = rc.Row

   'Sets the color of the selected row and selected cell
   For col = 0 To NumberOfColumns - 1
      If col = rc.col Then
         GetView(rc.Row, col).Color = SelectedCellColor
      Else
         GetView(rc.Row, col).Color = SelectedRowColor
      End If
   Next
End Sub
```

- GetView
  Gets the Label object for the given row and column.
  - Declare l as a Label.
  - Gets the View in the given row and column, the view index in the ScrollView panel is equal to `Row * NumberOfColumns + Col`.
  - Returns the Label.

```
Sub GetView(Row As Int, Col As Int) As Label
   'Returns the label in the specific cell
   Private l As Label

   l = scvPersons.Panel.GetView(Row * NumberOfColumns + Col)
   Return l
End Sub
```

- GetCell
  Gets the text of the Label for the given row and column.
  - Gets the View in the given row and column.
  - Return the Views Text parameter.

```
Sub GetCell(Row As Int, Col As Int) As String
   'Gets the value of the given cell
   Return GetView(Row, Col).Text
End Sub
```

- SetCell (not used in the program)
  Sets the text of the Label for the given row and column.
  - Gets the View in the given row and column
  - Sets the Views Text parameter to the given value

```
Sub SetCell(Row As Int, Col As Int, Value As String)
   'Sets the value of the given cell
   GetView(Row, Col).Text = Value
End Sub
```

- ClearAll
  - Removes all Views (Labels) from the ScrollView Panel
  - Sets the ScrollView Panel Height to 0
  - Sets the selected row index to -1, no row selected

```
Sub ClearAll
   'Clears the table
   For i = scvPersons.Panel.NumberOfViews -1 To 0 Step -1
      scvPersons.Panel.RemoveViewAt(i)
   Next
   scvPersons.Panel.Height = 0
   SelectedRow = -1
End Sub
```

- Horizontal moving with the SeekBar
  - Sets the Left parameter of the Header panel and the ScrollView.
  - The SeekBar Max value was set to
    `skbScroll.Max = scvPersons.Width - Activity.Width.`

```
Sub skbScroll_ValueChanged (Value As Int, UserChanged As Boolean)
   'Moves the ScrollView horizontally
   pnlHeader.Left = - Value
   scvPersons.Left = - Value
End Sub
```

- Horizontal scrolling with the scroll panel.
  - pnlScroll_Touch and Timer1_Tick.
  - I leave it up to you to find how these work.

    The basic principle is to calculate the speed between ACTION_DOWN and ACTION_UP and in the Timer routine to move dynamically the header and the Scrollview and reducing the speed.

For the horizontal moving we could use ScrollView2D instead of the standard vertical scrollview. This would allow to move the table in both directions simultaneously.

Another approach could be to add a HorizontalScrollView into the vertical ScrollView, this would allow to move in both directions but not simultaneously. This depends on the beginning of the moving if it’s horizontally only horizontal moving is allowed and if it’s vertically only vertical moving is allowed.

15 Basic language

In computer programming, **BASIC** (an acronym which stands for **B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode) is a family of high-level programming languages designed to be easy to use.
The original Dartmouth BASIC was designed in 1964 by John George Kemeny and Thomas Eugene Kurtz at Dartmouth College in New Hampshire, USA to provide computer access to non-science students. At the time, nearly all use of computers required writing custom software, which was something only scientists and mathematicians tended to do. The language and its variants became widespread on microcomputers in the late 1970s and 1980s.
BASIC remains popular to this day in a handful of highly modified dialects and new languages influenced by BASIC such as Microsoft Visual Basic. (source Wikipedia).

15.1 Expressions

An expression in a programming language is a combination of explicit values, constants, variables, operators, and functions that are interpreted according to the particular rules of precedence and of association for a particular programming language, which computes and then produces (returns) another value. This process, like for mathematical expressions, is called evaluation. The value can be of various types, such as numerical, string, and logical (source Wikipedia).

For example, 2 + 3 is an arithmetic and programming expression which evaluates to 5. A variable is an expression because it is a pointer to a value in memory, so y + 6 is an expression. An example of a relational expression is 4 = 4 which evaluates to True (source Wikipedia).

15.1.1 Mathematical expressions

Operator	Example	Precedence level	Operation
+	`x + y`	3	Addition
–	`x - y`	3	Subtraction
*	`x * y`	2	Multiplication
/	`x / y`	2	Division
`Mod`	`x Mod y`	2	Modulo
`Power`	`Power(x,y)` x^y	1	Power of

Precedence level: In an expression, operations with level 1 are evaluated before operations with level 2, which are evaluated before operations with level 3.

Examples:

4 + 5 * 3 + 2 = 21 > 4 + 15 + 2

(4 + 5) * (3 + 2) = 45 > 9 * 5

$(4 + 5)^2$ * (3 + 2) = 405 > 9^2 * 5 > 81 * 5
Power(4+5,2)*(3+2)

11 Mod 4 = 3 > Mod is the remainder of 10 / 4

23^3 Power(23,3) > 23 at the power of 3

$- 2^2 = - 4$
$(-2)^2 = 4$

15.1.2 Relational expressions

In computer science in relational expressions an operator tests some kind of relation between two entities. These include numerical equality (e.g., 5 = 5) and inequalities (e.g., 4 >= 3).
In B4A these operators return **True** or **False**, depending on whether the conditional relationship between the two operands holds or not (source Wikipedia).

Operator	Example	Used to test
=	x = y	the equivalence of two values
<>	x <> y	the negated equivalence of two values
>	x > y	if the value of the left expression is greater than that of the right
<	x < y	if the value of the left expression is less than that of the right
>=	x >= y	if the value of the left expression is greater than or equal to that of the right
<=	x <= y	if the value of the left expression is less than or equal to that of the right

15.1.3 Boolean expressions

In computer science, a Boolean expression is an expression that produces a Boolean value when evaluated, i.e. one of **True** or **False**. A Boolean expression may be composed of a combination of the Boolean constants **True** or **False**, Boolean-typed variables, Boolean-valued operators, and Boolean-valued functions (source Wikipedia).

Boolean operators are used in conditional statements such as IF-Then and Select-Case.

Operator	Comment
Or	Boolean Or Z = X Or Y Z = True if X or Y is equal to True or both are True
And	Boolean And Z = X And Y Z = True if X and Y are both equal to True
Not ( )	Boolean Not X = True Y = Not(X) > Y = False

		Or	And
X	Y	Z	Z
False	False	False	False
True	False	True	False
False	True	True	False
True	True	True	True

15.2 Conditional statements

Different conditional statements are available in Basic.

15.2.1 If – Then – End If

The **If-Then-Else** structure allows to operate conditional tests and execute different code sections according to the test result.
General case:

```
If test1 Then
   ' code1
Else If test2 Then
   ' code2
Else
   ' code3
End If
```

The **If-Then-Else** structure works as follows:

1. When reaching the line with the **If** keyword, **test1** is executed.
2. If the test result is **True**, then **code1** is executed until the line with the **Else If** keyword.
   And jumps to the line following the **End If** keyword and continues.
3. If the result is **False**, then **test2** is executed.
4. If the test result is **True**, then **code2** is executed until the line with the **Else** keyword.
   And jumps to the line following the **End If** keyword and continues.
5. If the result is **False**, then **code3** is executed and continues at the line following the **End If** keyword.

The tests can be any kind of conditional test with two possibilities **True** or **False**.
Some examples:

```
If b = 0 Then
   a = 0
End If
```

The simplest **If-Then** structure.

```
If b = 0 Then a = 0
```

The same but in one line.

```
If b = 0 Then
   a = 0
Else
   a = 1
End If
```

The simplest **If-Then-Else** structure.

```
If b = 0 Then a = 0 Else a = 1
```

The same but in one line.

Personally, I prefer the structure on several lines, better readable.
An old habit from HP Basic some decades ago, this Basic accepted only one instruction per line.

Note. Difference between:

B4A	VB
Else If	**ElseIf**

In B4A there is a blank character between **Else** and **If**.

Some users try to use this notation:

```
  If b = 0 Then a = 0 : c = 1
```

There is a big difference between B4A and VB that gives errors :
The above statements is equivalent to :

```
     B4A                                    VB
  If b = 0 Then                          If b = 0 Then
    a = 0                                  a = 0
  End If                                   c = 1
  c = 1                                  End If
```

The colon character ' : ' in the line above is treated in B4A like a CarriageReturn CR character.

This structure throws an error.

```
Sub Plus1 : x = x + 1 : End Sub
```

You cannot have a Sub declaration and End Sub on the same line.

15.2.2 Select – Case

The **Select - Case** structure allows to compare a **TestExpression** with other **Expressions** and to execute different code sections according to the matches between the **TestExpression** and **Expressions**.

General case:

```
Select TestExpression
Case ExpressionList1
   ' code1
Case ExpressionList2
   ' code2
Case Else
   ' code3
End Select
```

TestExpression is the expression to test.

ExpressionList1 is a list of expressions to compare to **TestExpression**
ExpressionList2 is another list of expressions to compare to **TestExpression**

The **Select - Case** structure works as follows:

1. The **TestExpression** is evaluated.
2. If one element in the **ExpressionList1** matches **TestExpression** then executes **code1** and continues at the line following the **End Select** keyword.
3. If one element in the **ExpressionList2** matches **TestExpression** then executes **code2** and continues at the line following the **End Select** keyword.
4. For no expression matches **TestExpression** executes **code3** and continues at the line following the **End Select** keyword.

TestExpression can be any expression or value.
ExpressionList1 is a list of any expressions or values.

Examples:

```
Select Value
Case 1, 2, 3, 4
```

The Value variable is a numeric value.

```
Select a + b
Case 12, 24
```

The **TestExpression** is the sum of a + b

```
Select Txt.CharAt
Case "A", "B", "C"
```

The **TestExpression** is a character at

```
Sub Activity_Touch (Action As Int, X As Float, Y As Float)
   Select Action
   Case Activity.ACTION_DOWN

   Case Activity.ACTION_MOVE

   Case Activity.ACTION_UP

   End Select
End Sub
```

Note. Differences between:

B4A	VB
`Select Value`	`Select Case Value`
`Case 1,2,3,4,8,9,10`	`Case 1 To 4 , 8 To 9`

In VB the keyword `Case` is added after the `Select` keyword.
VB accepts `Case 1 To 4` , this is not implemented in B4A.

15.3 Loop structures

Different loop structures are available in Basic.

15.3.1 For – Next

In a **For–Next** loop a same code will be executed a certain number of times.
Example:

```
For i = n1 To n2 Step n3

   ' Specific code

Next
```

i	incremental variable
n1	initial value
n2	final value
n3	step

The **For–Next** loop works as below:

1. At the beginning, the incremental variable **i** is equal to the initial value **n1**.
   i = n1
2. The specific code between the **For** and **Next** keywords is executed.
3. When reaching **Next**, the incremental variable **i** is incremented by the step value **n3**.
   i = i + n3.
4. The program jumps back to **For**, compares if the incremental variable **i** is lower or equal to the final value **n2**.
   test if i <= n2
5. If **Yes**, the program continues at step 2, the line following the **For** keyword.
6. If **No**, the program continues at the line following the **Next** keyword.

If the step value is equal to '+1' the step keyword is not needed.

```
For i = 0 To 10

Next
```

is the same as

```
For i = 0 To 10 Step 1

Next
```

The step variable can be negative.

```
For i = n3 To 0 Step -1

Next
```

It is possible to exit a For – Next loop with the `Exit` keyword.

```
For i = 0 To 10
   ' code
   If A = 0 Then Exit
   ' code
Next
```

In this example, if the variable a equals 0

Then exit the loop.

Note : Differences between

B4A	VB
`Next`	`Next i`
`Exit`	`Exit For`

In VB :

- The increment variable is added after the **`Next`** Keyword.
- The loop type is specified after the **`Exit`** keyword.

15.3.2 For - Each

It is a variant of the For - Next loop.

Example:

```
For Each n As Type In Array

   ' Specific code

Next
```

n	variable any type or object
Type	type of variable n
Array	Array of values or objects

The **For–Each** loop works as below:

1. At the beginning, **n** gets the value of the first element in the Array.
   n = Array(0)
2. The specific code between the **`For`** and **`Next`** keywords is executed.
3. When reaching **Next**, the program checks if **n** is the last element in the array.
4. If **No**, the variable **n** gets the next value in the Array and continues at step 2, the line following the **`For`** keyword.
   n = Array(next)
5. If **Yes**, the program continues at the line following the **`Next`** keyword.

Example For - Each :

```
Private Numbers() As Int
Private Sum As Int

Numbers = Array As Int(1, 3, 5 , 2, 9)

Sum = 0
For Each n As Int In Numbers
   Sum = Sum + n
Next
```

Same example but with a For - Next loop :

```
Private Numbers() As Int
Private Sum As Int
Private i As Int

Numbers = Array As Int(1, 3, 5 , 2, 9)

Sum = 0
For i = 0 To Numbers.Length - 1
   Sum = Sum + Numbers(i)
Next
```

This example shows the power of the For - Each loop :

```
For Each lbl As Label In Activity
   lbl.TextSize = 20
Next
```

Same example with a For - Next loop :

```
For i = 0 To Activity.NumberOfViews - 1
   Private v As View
   v = Activity.GetView(i)
   If v Is Label Then
      Private lbl As Label
      lbl = v
      lbl.TextSize = 20
   End If
Next
```

15.3.3 Do - Loop

Several configurations exist:

```
Do While test
   ' code
Loop
```

test is any expression
Executes the **code** while **test** is **True**

```
Do Until test
   ' code
Loop
```

test is any expression
Executes the **code** until **test** is **True**

The **Do While -Loop** loop works as below :

1. At the beginning, **test** is evaluated.
2. If **True**, then executes **code**
3. If **False** continues at the line following the **Loop** keyword.

The **Do Until -Loop** loop works as below :

1. At the beginning, **test** is evaluated.
2. If **False**, then executes **code**
3. If **True** continues at the line following the **Loop** keyword.

It is possible to exit a Do-Loop structure with the Exit keyword.

```
Do While test
   ' code
   If a = 0 Then Exit
   ' code
Loop
```

If **a = 0** then exit the loop

Examples :

Do Until Loop :

```
Private i, n As Int

i = 0
Do Until i = 10
   ' code
   i = i + 1
Loop
```

Do While Loop :

```
Private i, n As Int

i = 0
Do While i < 10
   ' code
   i = i + 1
Loop
```

Read a text file and fill a List :

```
Private lstText As List
Private line As String
Private tr As TextReader

tr.Initialize(File.OpenInput(File.DirInternal, "test.txt"))
lstText.Initialize
line = tr.ReadLine
Do While line <> Null
   lstText.Add(line)
   line = tr.ReadLine
Loop

tr.Close
```

Note : Difference between:

B4A	VB
`Exit`	`Exit Loop`

In VB the loop type is specified after the **`Exit`** keyword.

VB accepts also the following loops, which are not supported in B4A.

```
Do                          Do
  ' code                      ' code
Loop While test             Loop Until test
```

15.4 Subs

A Subroutine (“Sub”) is a piece of code. It can be any length, and it has a distinctive name and a defined scope (in the means of variables scope discussed earlier). In B4A code, a subroutine is called “Sub”, and is equivalent to procedures, functions, methods and subs in other programming languages. The lines of code inside a Sub are executed from first to last, as described in the program flow chapter.
It is not recommended to have Subs with a large amount of code, they get less readable.

15.4.1 Declaring

A Sub is declared in the following way:

```
Sub CalcInterest(Capital As Double, Rate As Double) As Double
   Return Capital * Rate / 100
End Sub
```

It starts with the keyword **Sub**, followed by the Sub’s name, followed by a parameter list, followed by the return type and ends with the keywords **End Sub**.
Subs are always declared at the top level of the module, you cannot nest two Subs one inside the other.

15.4.2 Calling a Sub

When you want to execute the lines of code in a Sub, you simply write the Sub’s name.

For example:

```
   Interest = CalcInterest(1234, 5.2)
```

Interest	Value returned by the Sub.
`CalcInterest`	Sub name.
`1235`	Capital value transmitted to the Sub.
`5.25`	Rate value transmitted to the Sub.

15.4.3 Calling a Sub from another module

A subroutine declared in a code module can be accessed from any other module but the name of the routine must have the name of the module where it was declared as a prefix.

Example: If the `CalcInterest` routine is declared in module `MyModule` then calling the routine must be :

```
   Interest = MyModule.CalcInterest(1234, 5.2)
```

instead of:

```
   Interest = CalcInterest(1234, 5.2)
```

15.4.4 Naming

Basically, you can name a Sub any name that’s legal for a variable. It is recommended to name the Sub with a significant name, like **`CalcInterest`** in the example, so you can tell what it does from reading the code.
There is no limit on the number of Subs you can add to your program, but it is not allowed to have two Subs with the same name in the same module.

```
Sub CalcInterest(Capital As Double, Rate As Double) As Double
   Return Capital * Rate / 100
End Sub
```

15.4.5 Parameters

Parameters can be transmitted to the Sub. The list follows the sub name. The parameter list is put in brackets.
The parameter types should be declared directly in the list.

```
Sub CalcInterest(Capital As Double, Rate As Double) As Double
   Return Capital * Rate / 100
End Sub
```

In B4A, the parameters are transmitted by value and not by reference.

15.4.6 Returned value

A sub can return a value, this can be any object.
Returning a value is done with the Return keyword.
The type of the return value is added after the parameter list.

```
Sub CalcInterest(Capital As Double, Rate As Double) As Double
   Return Capital * Rate / 100
End Sub
```

15.5 Events

In Object-oriented programming we have objects which can react on different user actions called events.
The number and the type of events an object can raise depend on the type of the object.
User interface objects are called 'Views' in Android.

Summary of the events for different views:

	Events																	
Views	Click	LongClick	Touch	Down	Up	KeyPress	KeyUp	ItemClick	ItemLongClick	CheckedChange	EnterPressed	FocusChanged	TextChanged	ScrollChanged	ValueChanged	TabChanged	OverrideUrl	PageFinished
Activity	🟩	🟩	🟩			🟩	🟩											
Button	🟩	🟩		🟩	🟩													
CheckBox										🟩								
EditText											🟩	🟩	🟩					
HorizontalScrollView														🟩				
ImageView	🟩	🟩																
Label	🟩	🟩																
ListView								🟩	🟩									
Panel	🟩	🟩	🟩															
RadioButton										🟩								
ScrollView														🟩				
SeekBar															🟩			
Spinner								🟩										
TabHost	🟩	🟩														🟩		
ToggleButton										🟩								
WebView																	🟩	🟩

The most common events are:

- **Click** Event raised when the user clicks on the view.
  Example:
  ```
  Sub Button1_Click
    ' Your code
  End Sub
  ```

- **LongClick** Event raised when the user clicks on the view and holds it pressed for a while.
  Example:
  ```
  Sub Button1_LongClick
    ' Your code
  End Sub
  ```

- **Touch** `(Action As Int, X As Float, Y As Float)`
  Event raised when the user touches the screen.

  Three different actions are handled:
  - Activity.Action_DOWN, the user touches the screen.
  - Activity.Action_MOVE, the user moves the finger without leaving the screen.
  - Activity.Action_UP, the user leaves the screen.

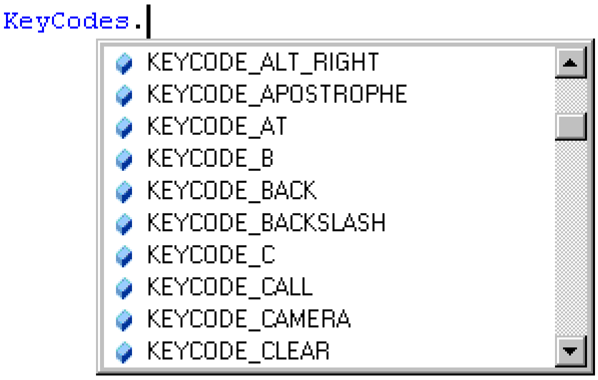
  The X an Y coordinates of the finger position are given.

  Example:
  ```
  Sub Activity_Touch (Action As Int, X As Float, Y As Float)
    Select Action
    Case Activity.ACTION_DOWN
      ' Your code for DOWN action
    Case Activity.ACTION_MOVE
      ' Your code for MOVE action
    Case Activity.ACTION_UP
      ' Your code for UP action
    End Select
  End Sub
  ```

- **CheckChanged** `(Checked As Boolean)`
  Event raised when the user clicks on a CheckBox or a RadioButton
  Checked is equal to True if the view is checked or False if not checked.

  Example:
  ```
  Sub CheckBox1_CheckedChange(Checked As Boolean)
    If Checked = True Then
      ' Your code if checked
    Else
      ' Your code if not checked
    End If
  End Sub
  ```

- **KeyPress** (KeyCode As Int) As Boolean
  Event raised when the user presses a physical or virtual key.
  KeyCode is the code of the pressed key, you can get them with the KeyCodes keyword.



```
KeyCodes.
  KEYCODE_ALT_RIGHT
  KEYCODE_APOSTROPHE
  KEYCODE_AT
  KEYCODE_B
  KEYCODE_BACK
  KEYCODE_BACKSLASH
  KEYCODE_C
  KEYCODE_CALL
  KEYCODE_CAMERA
  KEYCODE_CLEAR
```

The event can return either:
- True, the event is 'consumed', considered by the operating system as already executed and no further action is taken.
- False, the event is not consumed and transmitted to the system for further actions.

Example:

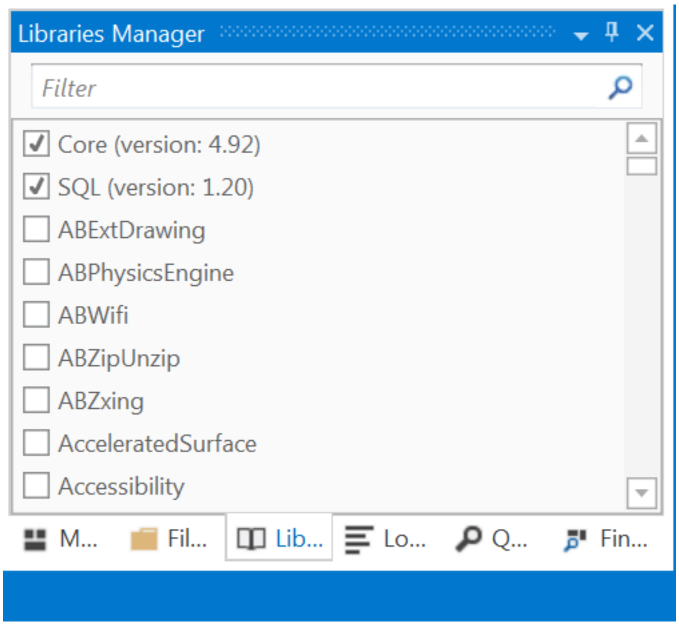
```
Sub Activity_KeyPress(KeyCode As Int) As Boolean
  Private Answ As Int
  Private Txt As String

  If KeyCode = KeyCodes.KEYCODE_BACK Then     ' Checks if KeyCode is BackKey
    Txt = "Do you really want to quit the program ?"
    Answ = Msgbox2(Txt,"A T T E N T I O N","Yes","","No",Null)' MessageBox
    If Answ = DialogResponse.POSITIVE Then  ' If return value is Yes then
      Return False     ' Return = False  the Event will not be consumed
    Else               '                  we leave the program
      Return True      ' Return = True   the Event will be consumed to avoid
    End If             '                 leaving the program
  End If
End Sub
```

15.6 Libraries

Libraries add more objects and functionalities to B4A.
Some of these libraries are shipped with B4A and are part of the standard development system.
Other, often developed by users like Andrew Graham (agraham), can be downloaded (by registered users only) to add supplementary functionalities to the B4A development environment.

When you need a library, you have to:

- Check it in the Libs Tab, if you already have the library.
- For additional libraries, check if it's the latest version.
  You can check the versions in the B4A Documentation Page.
  To find the library files use a query like
  http://www.b4x.com/search?query=betterdialogs+library
  in your internet browser.
- If **yes**, then check the library in the list to select it.



☑ Core (version: 4.92)
☑ SQL (version: 1.20)
□ ABExtDrawing
□ ABPhysicsEngine
□ ABWifi
□ ABZipUnzip
□ ABZxing
□ AcceleratedSurface
□ Accessibility

- If **no**, download the library, unzip it and copy the <LibraryName>.jar and <LibraryName>.xml files to the additional libraries folder.
- Right click in the Lib area and click on Refresh and check the library in the list to select it.

15.6.1 Standard libraries

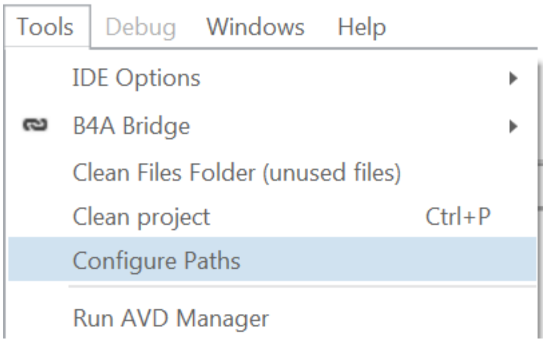
The standard B4A libraries are saved in the Libraries folder in the B4A program folder.
Normally in: C:\Program Files\Anywhere Software\B4A\Libraries

15.6.2 Additional libraries folder

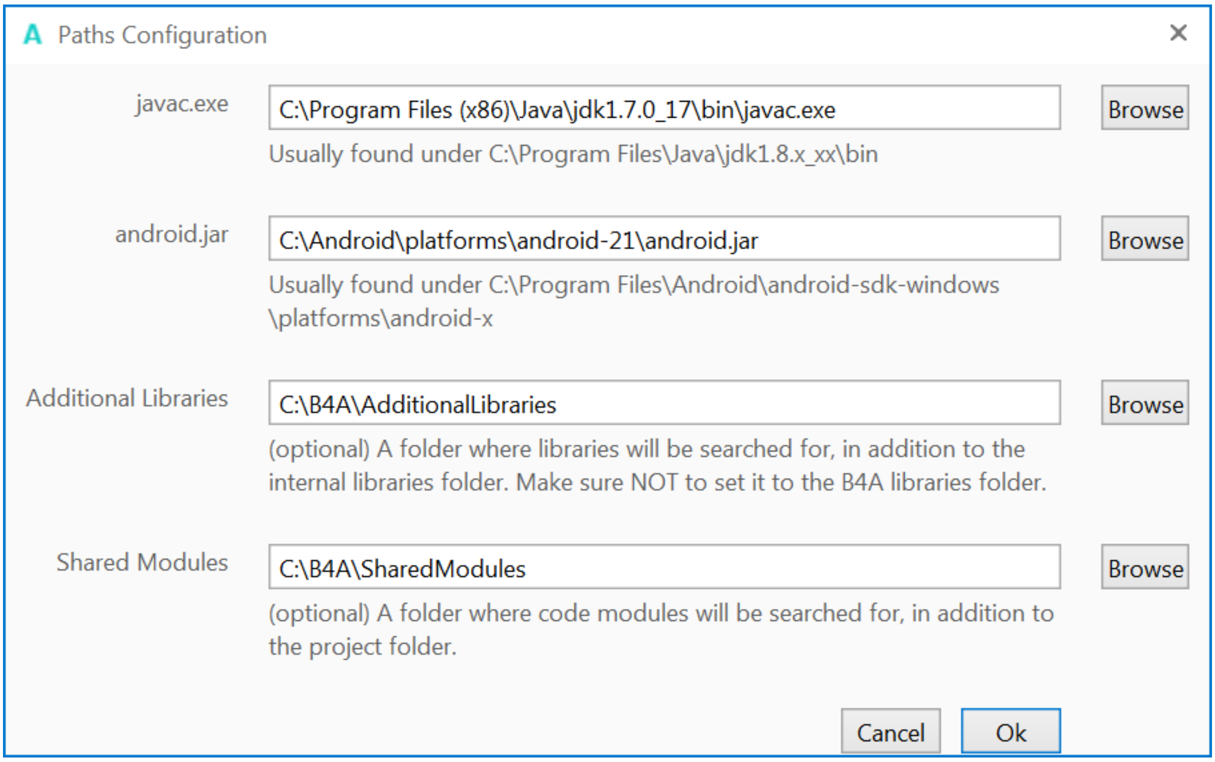
For the additional libraries it is useful to setup a special folder to save them somewhere else.
For example: D:\B4A\AddLibraries

When you install a new version of B4A, all standard libraries are automatically updated, but the additional libraries are not included. The advantage of the special folder is that you don't need to care about them because this folder is not affected when you install the new version of B4A.
The additional libraries are not systematically updated with new version of B4A.

When the IDE starts, it looks first for the available libraries in the Libraries folder of B4A and then in the folder for the additional libraries.



To setup the special additional libraries folder click in the IDE menu on Tools / Configure Paths.



Paths Configuration

javac.exe: C:\Program Files (x86)\Java\jdk1.7.0_17\bin\javac.exe
Usually found under C:\Program Files\Java\jdk1.8.x_xx\bin

android.jar: C:\Android\platforms\android-21\android.jar
Usually found under C:\Program Files\Android\android-sdk-windows \platforms\android-x

Additional Libraries: C:\B4A\AdditionalLibraries
(optional) A folder where libraries will be searched for, in addition to the internal libraries folder. Make sure NOT to set it to the B4A libraries folder.

Shared Modules: C:\B4A\SharedModules
(optional) A folder where code modules will be searched for, in addition to the project folder.

Enter the folder name and click on Ok.

15.6.3 Load and update a Library

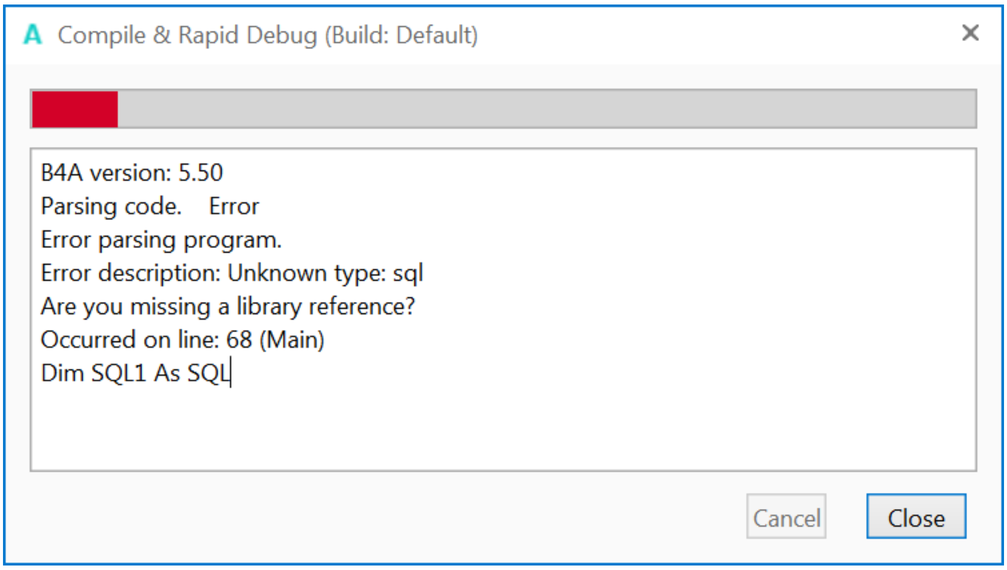
A list of the official and additional libraries with links to the relevant help documentation can be found on the B4x site in the B4A Documentation page: List of Libraries
To find the library files use a query like http://www.b4x.com/search?query=betterdialogs+library in your internet browser.

To load or update a library follow the steps below:

- Download the library zip file somewhere.
- Unzip it.
- Copy the xxx.jar and xxx.xml files to the
  - B4A Library folder for a standard B4A library
  - Additional libraries folder for an additional library.
- Right click in the libraries list in the Lib Tab and click on Refresh and select the library.

15.6.4 Error message "Are you missing a library reference?"

If you get a message similar to this, means that you forgot to check the specified library in the Lib Tab list !



```
B4A version: 5.50
Parsing code.    Error
Error parsing program.
Error description: Unknown type: sql
Are you missing a library reference?
Occurred on line: 68 (Main)
Dim SQL1 As SQL
```

15.7 String manipulation

B4A allows string manipulations like other Basic languages but with some differences.

These manipulations can be done directly on a string.
Example:

```
  txt = "123,234,45,23"
  txt = txt.Replace(",", ";")
```

Result: `123;234;45;23`

The different functions are:

- **CharAt(Index)** Returns the character at the given index.
- **CompareTo(Other)** Lexicographically compares the string with the Other string.
- **Contains(SearchFor)** Tests whether the string contains the given SearchFor string.
- **EndsWith(Suffix)** Returns True if the string ends with the given Suffix substring.
- **EqualsIgnoreCase(Other)** Returns True if both strings are equal ignoring their case.
- **GetBytes(Charset)** Encodes the Charset string into a new array of bytes.
- **IndexOf(SearchFor)** Returns the index of the first occurrence of SearchFor in the string. The index is 0 based. Returns -1 if no occurrence is found.
- **IndexOf2(SearchFor, Index)** Returns the index of the first occurrence of SearchFor in the string. Starts searching from the given index.
  The index is 0 based. Returns -1 if no occurrence is found.
- **LastIndexOf(SearchFor)** Returns the index of the first occurrence of SearchFor in the string. The search starts at the end of the string and advances to the beginning.
  The index is 0 based. Returns -1 if no occurrence is found.
- **LastIndexOf2(SearchFor)** Returns the index of the first occurrence of SearchFor in the string. The search starts at the given index and advances to the beginning.
  The index is 0 based. Returns -1 if no occurrence is found.
- **Length** Returns the length, number of characters, of the string.
- **Replace(Target, Replacement)** Returns a new string resulting from the replacement of all the occurrences of Target with Replacement.
- **StartsWith(Prefix)** Returns True if this string starts with the given Prefix.
- **Substring(BeginIndex)** Returns a new string which is a substring of the original string. The new string will include the character at BeginIndex and will extend to the end of the string.
- **Substring2(BeginIndex, EndIndex)** Returns a new string which is a substring of the original string. The new string will include the character at BeginIndex and will extend to the character at EndIndex, not including the last character.
  Note that EndIndex is the end index and not the length like in other languages.
- **ToLowerCase** Returns a new string which is the result of lower casing this string.
- **ToUpperCase** Returns a new string which is the result of upper casing this string.
- **Trim** Returns a copy of the original string without any leading or trailing white spaces.

Note: The string functions are case sensitive.
If you want to use case insensitive functions you should use either ToLowerCase or ToUpperCase.

Example:

```
NewString = OriginalString.ToLowerCase.StartsWith("pre")
```

15.8 Number formatting

Number formatting, display numbers as strings with different formats, there are two keywords:

- **NumberFormat**(Number As Double, MinimumIntegers As Int, MaximumFractions As Int)
  `NumberFormat(12345.6789, 0, 2)` = 12,345.68
  `NumberFormat(1, 3 ,0)` = 001
  `NumberFormat(Value, 3 ,0)` variables can be used.
  `NumberFormat(Value + 10, 3 ,0)` arithmetic operations can be used.
  `NumberFormat((lblscore.Text + 10), 0, 0)` if one variable is a string add parentheses.

- **NumberFormat2**(Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean)
  `NumberFormat2(12345.67, 0, 3, 3, True)` = 12,345.670

15.9 Timers

A Timer object generates ticks events at specified intervals. Using a timer is a good alternative to a long loop, as it allows the UI thread to handle other events and messages.
Note that the timer events will not fire while the UI thread is busy running other code (unless you call DoEvents keyword).
Timer events will not fire when the activity is paused, or if a blocking dialog (like Msgbox) is visible.
It is also important to disable the timer when the activity is pausing and then enable it when it resumes. This will save CPU and battery.

A timer has:

- Three parameters.
  - **Initialize** Initializes the timer with two parameters, the EventName and the interval.
    Timer1.Initialize(EventName As String, Interval As Long)
    Ex: `Timer1.Initialize("Timer1", 1000)`
  - **Interval** Sets the timer interval in milli-seconds.
    Timer1. Interval = Interval
    Ex: `Timer1.Interval = 1000`, 1 second
  - **Enabled** Enables or disables the timer. **It is False by default.**
    Ex: `Timer1.Enabled = True`
- One Event
  - **Tick** The Tick routine is called every time interval.
    Ex: `Sub Timer1_Tick`

The Timer must be declared in a Process_Global routine.

```
Sub Process_Globals
  Public Timer1 As Timer
```

But it must be initialized in the Activity_Create routine in the module where the timer tick event routine is used.

```
Sub Activity_Create(FirstTime As Boolean)
  If FirstTime = True Then
    Timer1.Initialize("Timer1", 1000)
  End If
```

And the Timer Tick event routine.
This routine will be called every second (1000 milli-seconds) by the operating system.

```
Sub Timer1_Tick
  ' Do something
End Sub
```

You find an example in the RotatingNeedle example program.

15.10 Files

Many applications require access to a persistent storage. The two most common storage types are files and databases.

Android has its own file system. Even on an Emulator a B4A program has no access to files in the Windows system,
It is possible to access Android files, from the Emulator, with the Dalvik Debug Monitor,
look at chapter 6.4 Exchanging files with the PC.
To add files to your project you must add those in the IDE in the Files Tab. These files will be added to the project Files folder.

15.10.1 File object

The predefined object `File` has a number of functions for working with files.

Files locations - There are several important locations where you can read or write files.

File.DirAssets
The assets folder includes the files that were added with the file manager in the IDE.
It's the Files folder in the project folder.
These files are read-only.
You can not create new files in this folder (which is actually located inside the apk file).
If you have a database file in the Dir.Assets folder you need to copy it to another folder before you can use it.

File.DirInternal / File.DirInternalCache
These two folders are stored in the main memory of the device and are private to your application.
Other applications cannot access these files.
The cache folder may get deleted by the OS if it needs more space.

File.DirRootExternal
The storage card root folder.

File.DirDefaultExternal
The default folder for your application in the SD card.
The folder is: <storage card>/Android/data/<package>/files/
It will be created if required.

Note that calling any of the two above properties will add the EXTERNAL_STORAGE permission to your application.

Tip: You can check if there is a storage card and whether it is available with
File.ExternalReadable and **File.ExternalWritable**.

To check if a file already exists use:
File.Exists (Dir As String, FileName As String)
Returns True if the file exists and False if not.

The File object includes several methods for writing to files and reading from files.
To be able to write to a file or to read from a file, it must be opened.

File.OpenOutput (Dir As String, FileName As String, Append As Boolean)
- Opens the given file for output, the Append parameter tells whether the text will be added at the end of the existing file or not. If the file doesn't exist it will be created.

File.OpenInput (Dir As String, FileName As String)
- Opens the file for reading.

File.WriteString (Dir As String, FileName As String, Text As String)
- Writes the given text to a new file.

File.ReadString (Dir As String, FileName As String) As String
- Reads a file and returns its content as a string.

File.WriteList (Dir As String, FileName As String, List As List)
- Writes all values stored in a list to a file. All values are converted to string type if required. Each value will be stored in a separare line.
Note that if a value contains the new line character it will saved over more than one line and when you read it, it will be read as multiple items.

File.ReadList (Dir As String, FileName As String) As List
- Reads a file and stores each line as an item in a list.

File.WriteMap (Dir As String, FileName As String, Map As Map)
- Takes a map object which holds pairs of key and value elements and stores it in a text file. The file format is known as Java Properties file: .properties - Wikipedia, the free encyclopedia
The file format is not too important unless the file is supposed to be edited manually. This format makes it easy to edit it manually.
One common usage of File.WriteMap is to save a map of "settings" to a file.

File.ReadMap (Dir As String, FileName As String) As Map
- Reads a properties file and returns its key/value pairs as a Map object. Note that the order of entries returned might be different than the original order.

Some other useful functions:

File.Copy (DirSource As String, FileSource As String, DirTarget As String, FileTarget As String)
- Copies the source file from the source directory to the target file in the target directory.
Note that it is not possible to copy files to the Assets folder.

File.Delete (Dir As String, FileName As String)
- Deletes the given file from the given directory.

File.ListFiles (Dir As String) As List
- Lists the files and subdirectories in the diven directory.
Example:

```
Private List1 As List
List1 = File.ListFiles(File.DirRootExternal)
```

List1 can be declared in Sub Globals

File.Size (Dir As String, FileName As String)
- Returns the size in bytes of the specified file.
This method does not support files in the assets folder.

15.10.2 Filenames

Android file names allow following characters :
a to **z**, **A** to **Z**, **0** to **9** dot **.** underscore **_** and even following characters **+ - % &**
Spaces and following characters *** ?** are not allowed.

Example : MyFile.txt

Note that Android file names are case sensitive !
`MyFile.txt` is different from `myfile.txt`

15.10.3 Subfolders

You can define subfolders in Android with.

```
File.MakeDir(File.DirInternal, "Pictures")
```

To access the subfolder you should add the subfoldername to the foldername with "/" inbetween.

```
ImageView1.Bitmap = LoadBitmap(File.DirInternal & "/Pictures", "test1.png")
```

Or add the subfoldername before the filename with "/" inbetween.

```
ImageView1.Bitmap = LoadBitmap(File.DirInternal, "Pictures/test1.png")
```

Both possibilities work.

15.10.4 TextWriter

There are two other useful functions for text files: **TextWriter** and TextReader:

TextWriter.Initialize (OutputStream As OutputStream)
- Initializes a TextWriter object as an output stream.

Example:

```
Private Writer As TextWriter
Writer.Initialize(File.OpenOutput(File.DirRootExternal, "Test.txt" , False))
```

Writer could be declared in Sub Globals.

TextWriter.Initialize2 (OutputStream As OutputStream , Encoding As String)
- Initializes a TextWriter object as as output stream.
- Encoding indicates the CodePage (also called CharacterSet), the text encoding (see next chapter).

Example:

```
Private Writer As TextWriter
Writer.Initialize2(File.OpenOutput(File.DirRootExternal,"Test.txt" ,False)," ISO-8859-1")
```

Writer could be declared in Sub Globals.
See : Text encoding

TextWriter.Write (Text As String)
- Writes the given Text to the stream.

TextWriter.WriteLine (Text As String)
- Writes the given Text to the stream followed by a new line character LF Chr(10).

TextWriter.WriteList (List As List)
- Writes each item in the list as a single line.
Note that a value containing CRLF will be saved as two lines (which will return two items when reading with ReadList).
All values will be converted to strings.

TextWriter.Close
- Closes the stream.

Example:

```
Private Writer As TextWriter
Writer.Initialize(File.OpenOutput(File.DirDefaultExternal, "Text.txt", False))
Writer.WriteLine("This is the first line")
Writer.WriteLine("This is the second line")
Writer.Close
```

15.10.5 TextReader

There are two other useful functions for text files: TextWriter and **TextReader**:

TextReader.Initialize (InputStream As InputStream)
- Initializes a TextReader as an input stream.

Example:

```
Private Reader  TextReader
Reader.Initialize(File.InputOutput(File.DirRootExternal, "Test.txt"))
```

Reader could be declared in Sub Globals.

TextReader.Initialize2 (InputStream As InputStream, Encoding As String)
- Initializes a TextReader as an input stream.
- Encoding indicates the CodePage (also called CharacterSet), the text encoding.

Example:

```
Private Reader  TextReader
Reader.Initialize2(File.OpenInput(File.DirRootExternal, "Test.txt", "ISO-8859-1")
```

Reader could be declared in Sub Globals.
See : Text encoding

TextReader.ReadAll As String
- Reads all of the remaining text and closes the stream.

Example:

```
txt = Reader.ReadAll
```

TextReader.ReadLine As String
- Reads the next line from the stream.
The new line characters are not returned.
Returns Null if there are no more characters to read.

Example:

```
Private Reader As TextReader
Reader.Initialize(File.OpenInput(File.DirDefaultExternal, "Text.txt"))
Private line As String
line = Reader.ReadLine
Do While line <> Null
   Log(line)
   line = Reader.ReadLine
Loop
Reader.Close
```

TextReader.ReadList As List
- Reads the remaining text and returns a List object filled with the lines.
Closes the stream when done.

Example:

```
List1 = Reader.ReadList
```

15.10.6 Text encoding

Text encoding or character encoding consists of a code that pairs each character from a given repertoire with something else. Other terms like character set (charset), and sometimes character map or code page are used almost interchangeably (source Wikipedia).

The default character set in Android is Unicode UTF-8.

In Windows the most common character sets are ASCII and ANSI.

- ASCII includes definitions for 128 characters, 33 are non-printing control characters (now mostly obsolete) that affect how text and space is processed.
- ANSI, Windows-1252 or CP-1252 is a character encoding of the Latin alphabet, used by default in the legacy components of Microsoft Windows in English and some other Western languages with 256 definitions (one byte). The first 128 characters are the same as in the ASCII encoding.

Many files generated by Windows programs are encoded with the ANSI character-set in western countries. For example: Excel csv files, Notepad files by default.
But with Notepad, files can be saved with *UTF-8* encoding.

Android can use following character sets:

• UTF-8	default character-set
• UTF -16	
• UTF - 16 BE	
• UTF - LE	
• US-ASCII	ASCII character set
• ISO-8859-1	almost equivalent to the ANSI character-set
• Windows-1251	cyrillic characters
• Windows-1252	latin alphabet

To read Windows files encoded with ANSI you should use the *Windows-1252* character-set.
If you need to write files for use with Windows you should also use the *Windows-1252* character-set.

Another difference between Windows and Android is the end of line character:

- Android, only the LF (Line Feed) character Chr(10) is added at the end of a line.
- Windows, two characters CR (Carriage Return Chr(13)) and LF Chr(10) are added at the end of a line. If you need to write files for Windows you must add CR yourself.

The symbol for the end of line is :

• B4A	CRLF	Chr(10)
• Basic4PPC	CRLF	Chr(13) & Chr(10)

To read or write files with a different encoding you must use the TextReader or TextWriter objects with the Initialize2 methods. Even for reading csv files.

Tip for reading Excel csv files:
You can either:

- On the desktop, load the csv file in a text editor like *NotePad* or *Notepad++*
- Save the file with *UTF-8* encoding
  With *Notepad++* use Encode in UTF-8 without BOM, see below.

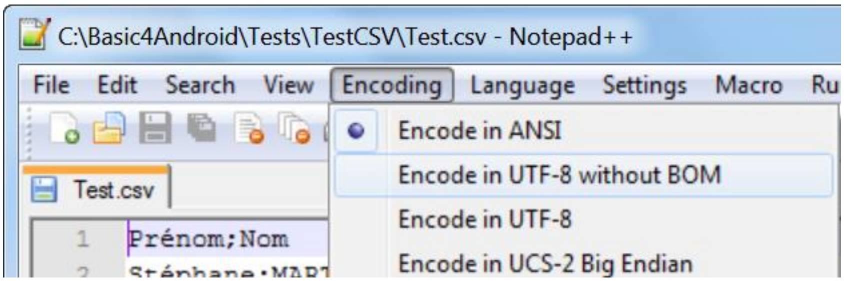
Or

- Read the whole file with TextReader.Initialize2 and "Windows-1252" encoding.
- Save it back with TextWriter.Initialize with the standard Android encoding.
- Read the file with LoadCSV or LoadCSV2 from the StringUtils library.

```
Private txt As String
Private tr As TextReader
tr.Initialize2(File.OpenInput(File.DirAssets, "TestCSV1_W.csv"), "Windows-1252")
txt = tr.ReadAll
tr.Close

Private tw As TextWriter
tw.Initialize(File.OpenOutput(File.DirInternal, "TestCSV1_W.csv", False))
tw.Write(txt)
tw.Close

lstTest = StrUtil.LoadCSV2(File.DirInternal, "TestCSV1_W.csv", ";", lstHead)
```

When you save a file with NotePad three additional bytes are added .
These bytes are called BOM characters (Byte Order Mark).
In *UTF-8* they are represented by this byte sequence : `0xEF,0xBB,0xBF`.
A text editor or web browser interpreting the text as *Windows-1252* will display the characters `ï»¿`.

To avoid this you can use *Notepad++* instead of *NotePad* and use Encode in *UTF-8* without BOM.


C:\Basic4Android\Tests\TestCSV\Test.csv - Notepad++
File
Edit
Search
View
Encoding
Language
Settings
Macro
Ru
Test.csv
Encode in ANSI
Encode in UTF-8 without BOM
Encode in UTF-8
Encode in UCS-2 Big Endian

```
1 Prénom;Nom
2 Stéphane;MART
```

Another possibility to change a text from *Windows-1252* to *UTF-8* is to use the code below.

```
Private var, result As String
var = "Gestió"
Private arrByte() As Byte
arrByte = var.GetBytes("Windows-1252")
result = BytesToString(arrByte, 0, arrByte.Length, "UTF8")
```

15.11 Lists

Lists are similar to dynamic arrays, detailed descriptions of all functions are in chapter List.

Lists are often used and many examples can be found in code examples:

- StringUtils LoadCSV, SaveCSV
- DBUtils module InsertMaps, UpdateRecord, ExecuteMemoryTable, ExecuteSpinner, ExecuteListView, ExecuteHtml, ExecuteJSON
- Charts module to hold different variables.

A list must be initialized before it can be used.

- Initialize Initializes an empty List.
  ```
  Private List1 As List
  List1.Initialize
  List1.AddAll(Array As Int(1, 2, 3, 4, 5))
  ```

- Initialize2 (SomeArray)
  Initializes a list with the given values. This method should be used to convert arrays to lists.
  Note that if you pass a list to this method then both objects will share the same list, and if you pass an array the list will be of a fixed size.
  Meaning that you cannot later add or remove items.
  Example 1:
  ```
  Private List1 As List
  List1.Initialize2(Array As Int(1, 2, 3, 4, 5))
  ```
  Example 2:
  ```
  Private List1 As List
  Private SomeArray(10) As String
  ' Fill the array
  List1.Initialize2(SomeArray)
  ```

You can add and remove items from a list and it will change its size accordingly.
With either:

- Add (item As Object)
  Adds a value at the end of the list.
  ```
  List1.Add(Value)
  ```

- AddAll (Array As String("value1", "value2"))
  Adds all elements of an array at the end of the list.
  ```
  List1.AddAll(List2)
  List1.AddAll(Array As Int(1, 2, 3, 4, 5))
  ```

- AddAllAt (Index As Int, List As List)
  Inserts all elements of an array in the list starting at the given position.
  ```
  List1.AddAll(12, List2)
  List1.AddAllAt(12, Array As Int(1, 2, 3, 4, 5))
  ```

- InsertAt (Index As Int, Item As Object)
  Inserts the specified element in the specified index.
  As a result all items with index larger than or equal to the specified index are shifted.
  ```
  List1.InsertAt(12, Value)
  ```

- RemoveAt (Index As Int)
  Removes the specified element at the given position from the list.
  ```
  List1.RemoveAt(12)
  ```

A list can hold any type of object. However if a list is declared as a process global object it cannot hold activity objects (like views).
B4A automatically converts regular arrays to lists. So when a List parameter is expected you can pass an array instead.

Get the size of a List:

- `List1.Size`

Use the Get method to get an item from the list with (List indexes are 0 based):
To get the first item use Get(0).
To get the last item use Get(List1.Size - 1).

- Get(Index As Int)
  ```
  number = List1.Get(i)
  ```

  You can use a For loop to iterate over all the values:
  ```
  For i = 0 To List1.Size - 1
     Private number As Int
     number = List1.Get(i)
     ...
  Next
  ```

Lists can be saved and loaded from files with:

- File.WriteList(Dir As String, FileName As String, List As List)
  ```
  File.WriteList(File.DirRootExternal, "Test.txt", List1)
  ```
- File.ReadList (Dir As String, FileName As String)
  ```
  List1 = File.ReadList(File.DirRootExternal, "Test.txt")
  ```

A single item can be changed with :

- List1. Set(Index As Int, Item As Object)
  ```
  List1.Set(12, Value)
  ```

A List can be sorted (the items must all be numbers or strings) with :

- Sort(Ascending As Boolean)
  `List1.Sort(True)` sort ascending
  `List1.Sort(False)` sort descending
- SortCaseInsensitive(Ascending As Boolean)

Clear a List with :

- `List1.Clear`

15.12 Maps

A Map is a collection that holds pairs of keys and values, detailed descriptions of all functions are in chapter Map.

The keys are unique. Which means that if you add a key/value pair (entry) and the collection already holds an entry with the same key, the previous entry will be removed from the map.

The key should be a string or a number. The value can be any type of object.

Similar to a list, a map can hold any object, however if it is a process global variable then it cannot hold activity objects (like views).

Maps are very useful for storing applications settings.

Maps are used in these example codes:

- DBUtils module
  used for database entries, keys are the column names and values the column values.
- StateManager module used for settings

A list must be initialized before it can be used.

- Initialize Initializes an empty Map.

```
Private Map1 As Map
Map1.Initialize
```

Add a new entry :

- Put(Key As Object, Value As Object)

```
Map1.Put("Language", "English")
```

Get an entry :

- Get(Key As Object)

```
Language = Map1.Get("Language")
```

Get a key or a value at a given index :

Returns the value of the item at the given index.
GetKeyAt and GetValueAt should be used to iterate over all the items.
These methods are optimized for iterating over the items in ascending order.

- GetKeyAt(Index As Int)

```
Key = Map1.GetKeyAt(12)
```

Get a value at a given index :

- GetValueAt(Index As Int)

```
Value = Map1.GetValueAt(12)
```

Check if a Map contains an entry, tests whether there is an entry with the given key :

- ContainsKey(Key As Object)

```
If Map1.ContainsKey("Language") Then
   Msgbox("There is already an entry with this key !", "ATTENTION")
   Return
End If
```

Remove an entry :

- Remove(Key As Object)

  ```
  Map1.Remove("Language")
  ```

Clear an entry, clears all items from the map :

- Clear

  ```
  Map1.Clear
  ```

Maps can be saved and loaded with :

- File.WriteMap(Dir As String, FileName As String, Map As Map)

  ```
  File.WriteMap(File.DirInternal, "settings.txt", mapSettings)
  ```

- ReadMap(Dir As String, FileName As String)
  Reads the file and parses each line as a key-value pair (of strings).
  Note that the order of items in the map may not be the same as the order in the file.

  ```
  mapSettings = File.ReadMap(File.DirInternal, "settings.txt")
  ```

- File.ReadMap2(Dir As String, FileName As String, Map As Map)
  Similar to ReadMap. ReadMap2 adds the items to the given Map.
  By using ReadMap2 with a populated map you can force the items order as needed.

  ```
  mapSettings = File.ReadMap2(File.DirInternal, "settings1.txt", mapSettings)
  ```

16 Graphics / Drawing

16.1 Overview

To draw graphics we need to use a Canvas object.
Explanations from the help file.
A Canvas is an object that draws on other views or (mutable) bitmaps.
When the canvas is initialized and set to draw on a view, a new mutable bitmap is created for that view background, the current view's background is copied to the new bitmap and the canvas is set to draw on the new bitmap.
The canvas drawings are not immediately updated on the screen. You should call the target view Invalidate method to make it refresh the view.
This is useful as it allows you to make several drawings and only then refresh the display.
The canvas can be temporary limited to a specific region (and thus only affect this region). This is done by calling ClipPath. Removing the clipping is done by calling RemoveClip.
You can get the bitmap that the canvas draws on with the Bitmap property.
This is an 'Activity Object', it cannot be declared under Sub Process_Globals.

It is possible to draw onto the following views:

- Activity
- ImageView
- Panel
- Bitmap (mutable)

In the following functions you will find a number of common parameters.

Bitmap1 as Bitmap	an Android bitmap
x, y. x1, y1, x2, y2 As Float	are coordinates, Float variables.
Color as Int	are color variables. Int variables
SrcRect, DestRact, Rect1 As Rect	are rectangles, Rect objects
Filled As Boolean	flag if the surface is filled (True) or not (False)

The most common drawing functions are:

- **DrawBitmap** (Bitmap1 As Bitmap, SrcRect As Rect, DestRect As Rect)
  Draws the given bitmap or only a part of it..
  SrcRect = source rectangle, can be only a part of the original bitmap.
  DestRect = destination rectangle, can be any size.
  To draw with the same size both rectangles must have same width and same height.
  If DestRect is different size than SrcRect the destination drawing is stretched or shrinked depending on the size ratios between the two rectangles.

- **Draw BitmapRotated** (Bitmap1 As Bitmap, SrcRect As Rect, DestRect As Rect, Degrees As Float)
  Same function as DrawBitmap, but with a rotation of the given Degrees angle around the centre of the bitmap.

- **DrawCircle** (x As Float, y As Float, Radius As Float, Color as Int, Filled As Boolean, StrokeWidth As Float)
  Draws a circle.
  x an y are the centre coordinates of the circle and Radius the circles radius.

- **DrawColor** (Color As Int)
  Fills the whole view with the given color.
  The color can be Colors.Transparent making the whole view transparent.

- **DrawLine** (x1 As Float, y1 As Float, x2 As Float, y2 As Float, Color as Int, StrokeWidth As Float)
  Draws a straight line.

- **DrawRect** (Rect1 As Rect, Color As Int, Filled As Boolean, StrokeWidth as Float)
  Draws a rectangle with given size, color, filled or not and line width.

- **DrawRectRotated** (Rect1 As Rect, Color As Int, Filled As Boolean, StrokeWidth As Float, Degrees As Float)
  Same as DrawRect but rotated by the given angle

- **DrawText** (Text As String, x As Float, y As Float, Typeface1 As TypeFace, TestSize As Float, Color As Int Align1 As Align)

- **DrawTextRotated** (Text As String, x As Float, y As Float, Typeface1 As TypeFace, TestSize As Float, Color As Int Align1 As Align, Degrees As Float)

16.2 Drawing test programs

16.2.1 First steps

The project is in: SourceCode\Graphics\GraphicsFirstSteps.b4a

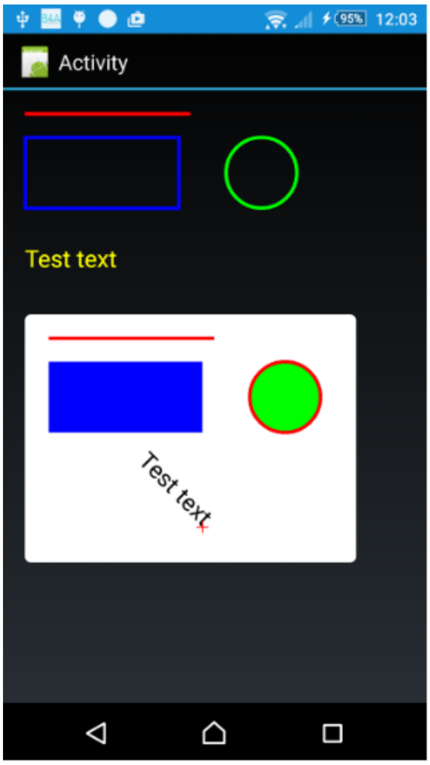
To draw something we need a Canvas object which is simply a drawing tool.
The Canvas draws onto a Bitmap. This Bitmap can be the background bitmap of views.

The most common views to draw on are: Activity, Panel, ImageView or a Bitmap.
The Canvas must be 'connected' to a bitmap or a view background image in the Initialize method.

- Initialize(Target View)
- Initialize2(Target Bitmap)

If we want to draw on different views at the same time we need one Canvas for each view.

In the example program we'll use several drawing functions and draw onto the Activity and onto a Panel `pnlGraph` defined in the 'main' layout file. Here we need two canvases.



16.2.1.1 Start and Initialisation

First we must declare the different views and objects:
We have:

- the Panel `pnlGraph`
- the Canvas `cvsActivity` for the Activity
- the Canvas `cvsPanel` for the Panel

```
Sub Globals
   Private pnlGraph As Panel
   Private cvsActivity, cvsGraph As Canvas
End Sub
```

Then we must load the layout file and initialize the two Canvases:

```
Sub Activity_Create(FirstTime As Boolean)
   ' load the layout file
   Activity.LoadLayout("main")

   ' initialize the Canvas for the activity
   cvsActivity.Initialize(Activity)

   ' initialize the Canvas for the pnlGraph panel
   cvsGraph.Initialize(pnlGraph)
End Sub
```

16.2.1.2 Draw a line

Then in Activity_Resume we draw a horizonzal line onto the Activity :
The function is :
DrawLine (x1 As Float, y1 As Float, x2 As Float, y2 As Float, Color as Int, StrokeWidth As Float)
Where :

- x1, y1 are the coordinates of the start point in pixels
- x2, y2 are the coordinates of the end point in pixels
- Color is the line color
- StrokeWidth the line thickness in pixels

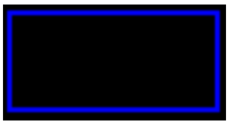
And the code :

```
   ' draw a horizontal line onto the Activity
   cvsActivity.DrawLine(20dip, 20dip, 160dip, 20dip, Colors.Red, 3dip
```

Then we draw a horizonzal line onto pnlGraph with the same coordinates :
The coordinates are relative to the upper left corner of the view we draw on, the Panel `pnlGraph` in this case.

```
   ' draw a horizontal line onto pnlGraph
   cvsGraph.DrawLine(20dip, 20dip, 160dip, 20dip, Colors.Red, 3dip)
```

16.2.1.3 Draw a rectangle

Then we draw an empty rectangle onto the Activity : 
The function is :
DrawRect (Rect1 As Rect, Color As Int, Filled As Boolean, StrokeWidth as Float)
Where :

- Rect1 is a rectangle object
- Color is the border or rectangle color
- Filled: False = only the border is drawn True = the rectangle is filled
- StrokeWidth is the line thickness of the border, not relevant when Filled = True

To draw a rectangle we need a Rect object.
We :

- Define it with the name `rect1`.
- Initialize it with the coordinates of the upper left corner and the coordinates of the lower right corner.
- Draw it

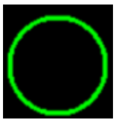
```
' draw an empty rectangle onto the Activity
Private rect1 As Rect
rect1.Initialize(20dip, 40dip, 150dip, 100dip)
cvsActivity.DrawRect(rect1, Colors.Blue, False, 3dip)
```

Then we draw a filled rectangle onto pnlGraph with the same coordinates :

We don't need to define nor initialize a new rectangle because the coordinates are the same so we use the same Rect object.

```
' draw a filled rectangle onto pnlGraph
cvsGraph.DrawRect(rect1, Colors.Blue, True, 3dip)
```

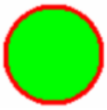
16.2.1.4 Draw a circle

Then we draw an empty circle onto the Activity : 
The function is :
DrawCircle (x As Float, y As Float, Radius As Float, Color as Int, Filled As Boolean, StrokeWidth As Float)
Where :

- x, y are the coordinates of the center in pixels.
- Radius is the radius in pixels.
- Color is the border or circle color
- Filled: False = only the border is drawn True = the circle is filled
- StrokeWidth is the line thickness of the border, not relevant when Filled = True

And the code:

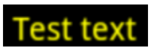
```
' draw an empty circle onto the Activity
cvsActivity.DrawCircle(220dip, 70dip, 30dip, Colors.Green, False, 3dip)
```

Then we draw a filled circle with a border with a different color on the panel. 

There is no direct function to draw a filled circle with a border with a different colors.
So we first draw the filled circle and then the circle border in two steps.
Instead of using fixed values like 220dip we can also use variables like in the code below.
When a same value is used several times it's better to use variables because if you need to change the value you change it only once the value of the variable all the rest is changed automatically by the variable.

```
' draw a filled circle with a boarder onto pnlGraph
Private centerX, centerY, radius As Float
centerX = 220dip
centerY = 70dip
radius = 30dip
cvsGraph.DrawCircle(centerX, centerY, radius, Colors.Green, True, 3dip)
cvsGraph.DrawCircle(centerX, centerY, radius, Colors.Red, False, 3dip)
```

16.2.1.5 Draw a text

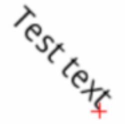
Then we draw a text onto the Activity. 
Test text

The function is:
DrawText (Text As String, x As Float, y As Float, Typeface1 As TypeFace, TestSize As Float, Color As Int Align1 As Align)
Where:

- Text is the text to draw
- x, y are the coordinates of the reference point (depending on the Align1 value) in pixels.
  The reference point is on the texts baseline.
- TypeFace1 is the text style
- TextSize is the text size in a typographic unit called 'point'.
  The text size is independant of the screen density !
  Don't use dip values !
- Color is the text color
- Align1 is the alignement of the text according to the refence point.
  Possible values : "LEFT", "CENTER", "RIGHT".

And the code:

```
' draw a text onto the Activity
cvsActivity.DrawText("Test text", 20dip, 150dip, Typeface.DEFAULT, 20, _
Colors.Yellow, "LEFT")
```


Test text

Then we draw a rotated text onto pnlGraph.

And we draw a cross on the reference point to show where it is and how the align does work.
The function is DrawTextRotated, it's the same as DrawText but with an additional parameter Degrees, the rotation angle.
Instead of using fiexd dip values in the routine we define three variables:

refX and refY	the coordinates of the reference point
hl	the half of the cross line length

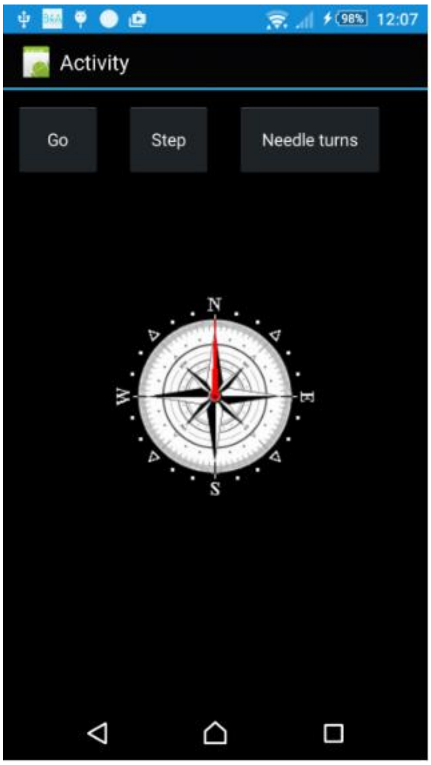
```
Private refX, refY, hl As Float
refX = 150dip
refY = 180dip
hl = 5dip
' draw a rotated text onto pnlGraph
cvsGraph.DrawTextRotated("Test text", refX, refY, Typeface.DEFAULT, _
20, Colors.Black, "RIGHT", 45)

' draw a cross on the reference point
cvsGraph.DrawLine(refX - hl, refY, refX + hl, refY, Colors.Red, 1dip)
cvsGraph.DrawLine(refX, refY - hl, refX, refY + hl, Colors.Red, 1dip)
```

16.2.2 Drawing rotating bitmaps / RotatingNeedle

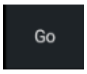
The project is in: SourceCode\Graphics\RotatingNeedle\RotatingNeedle.b4a

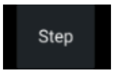
In the second test program we will demonstrate the DrawBitmapRotated function.
The program has two modes:

- A rotating needle with a static compass
- A rotating compass with a static needle



We have in the layout:

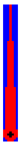
- 3 buttons
  - ○ 

starts rotating
  - ○ 

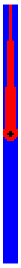
step by step moving
  - ○

we can let turn either the needle or the compass.
- 2 bitmap files
  - ○ compass.png
  - ○ needle.png

In the DrawBitmapRotated function the bitmap rotates around the bitmaps centre.



If we had a needle image like this one, we would need to do some calculations to make sure that it turns around the needle centre.



To avoid these calculations, the needle bitmap looks like this one. We added the lower part so that the needle centre is at the bitmap's centre.

The blue pixels are, in reality, transparent pixels.

Let us have a look at the code.

```
Sub Process_Globals
   Public AngleStep = 6 As Float
   Public Angle = -AngleStep As Float
   Public Mode = True As Boolean
   Public Timer1 As Timer
End Sub
```

Here we define three global variables with their values.

- AngleStep   step in degrees for the angle variations from one step to the next.
- Angle   current angle of the needle
- Mode   program mode
  True = needle turns
  False = compass turns

```
Sub Globals
   Private btnGoStop, btnStep, btnMode As Button
   Private cvsCompass, cvsNeedle As Canvas
   Private bmpCompass, bmpNeedle As Bitmap
   Private imvCompass, imvNeedle As ImageView
   Private RectCompass, SRectNeedle, DRectNeedle As Rect
End Sub
```

Then we define the different objects used by the program.

- The three buttons from the layout file.
- Two Canvas views, one for the compass and one for the needle.
- Two Bitmaps, one for the compass and one for the needle.
- Two ImageViews, one for the compass and one for the needle.
- Three rectangles, one for the compass, two for the needle source and destination.
- One Timer, it is used to move dynamically the needle or the compass.

In the Activity_Create routine we:

```
Sub Activity_Create(FirstTime As Boolean)
   Private x, y As Float

   Activity.LoadLayout("rotatingneedle")
```

- Define two variables used for calculations
- Load the layout file to the Activity

```
   bmpCompass.Initialize(File.DirAssets,"compass.png")
   bmpNeedle.Initialize(File.DirAssets,"needle.png")
```

- Initialize the compass bitmap
- Initialize the needle bitmap

```
    imvCompass.Initialize("")
    imvCompass.Bitmap = bmpCompass
    imvNeedle.Initialize("")
    imvNeedle.Color=Colors.Transparent
```

- Initialize the compass ImageView.
- Set the compass bitmap to the compass ImageView bitmap.
- Initialize the needle ImageView.
- Set the needle ImageView color to transparent.

```
    x = (100%x - bmpCompass.Width) / 2
    y = (100%y - bmpCompass.Height) / 2
    Activity.AddView(imvCompass, x, y, bmpCompass.Width, bmpCompass.Height)
    Activity.AddView(imvNeedle, x, y, bmpCompass.Width, bmpCompass.Height)
    cvsCompass.Initialize(imvCompass)
    RectCompass.Initialize(0, 0, bmpCompass.Width, bmpCompass.Height)
```

- Calculate the Left and Top coordinates of the compass ImageView.
- Add the compass ImageView to the Activity.
- Add the needle ImageView to the Activity
  with the same dimensions as the compass ImageView.
- Initialize the compass Canvas and connect it to the compass ImageView.
- Initialize the compass rectangle.

```
    csvNeedle.Initialize(imvNeedle)
    x = (bmpCompass.Width - bmpNeedle.Width)/2
    y = (bmpCompass.Height - bmpNeedle.Height)/2
    SRectNeedle.Initialize(0, 0, bmpNeedle.Width, bmpNeedle.Height)
    DRectNeedle.Initialize(x, y, x + bmpNeedle.Width, y + bmpNeedle.Height)
```

- Initialize the needle Canvas and connect it to the needle ImageView.
- Calculate the Left and Top coordinates of the needle ImageView.
- Initialize the needle source and destination rectangles.

```
    Timer1.Initialize("Timer1",200)
    Timer1_Tick
End Sub
```

- Initialize the timer, set the Interval to 200 ms.
- Call the Timer1_Tick routine to draw the needle

In the Timer1_Tick routine we:

```
Sub Timer1_Tick
  Private Angle1 As Float

  Angle1 = Angle
  Angle = (Angle+AngleStep) Mod 360
  If Mode = True Then
    cvsNeedle.DrawRectRotated(DRectNeedle,Colors.Transparent,True,1,Angle1)
    cvsNeedle.DrawBitmapRotated(bmpNeedle,SRectNeedle,DRectNeedle,Angle)
    imvNeedle.Invalidate2(RectCompass)
  Else
    cvsCompass.DrawBitmapRotated(bmpCompass,RectCompass,RectCompass,-Angle)
    imvCompass.Invalidate2(RectCompass)
  End If
End Sub
```

- Define a local variable representing the current Angle
- Calculate the new Angle using the Mod operator
- If Mode = True, rotating needle mode we:
  - Draw a rotated transparent rectangle to erase the current needle.
  - Draw the needle with the new angle.
  - Invalidate the needle ImageView to update it.
- If Mode = False, rotating compass mode we:
  - Draw the compass with the new angle, in our case the source and destination rectangle are the same.
  - Invalidate the compass ImageView to update it.

In the btnStep_Click routine we:

```
Sub btnStep_Click
  Timer1_Tick
End Sub
```

- Call the Timer1_Tick routine to draw a new step.

In the btnGoStop_Click routine we:

```
Sub btnGoStop_Click
   If Timer1.Enabled = True Then
      Timer1.Enabled = False
      btnGoStop.Text = "Go"
      btnStep.Visible = True
   Else
      Timer1.Enabled = True
      btnGoStop.TExt = "Stop"
      btnStep.Visible = False
   End If
End Sub
```

- If Timer1 = True, the timer is running.
  - We set the Timer1.Enabled property to False to stop it.
  - Set the btnGoStop button text to "Go".
  - Set the btnStep button to visible.
- If Timer1 = False, the timer is stopped
  - We set the Timer1.Enabled property to True to let it run.
  - Set the btnGoStop button text to "Stop".
  - Hide the btnStep button.

In the btnMode_Click routine we:

```
Sub btnMode_Click
   Mode = Not(Mode)
   If Mode = True Then
      btnMode.Text = "Needle turns"
      cvsNeedle.DrawRect(DRectNeedle, Colors.Transparent, True, 1)
      cvsNeedle.DrawBitmapRotated(bmpNeedle,SRectNeedle,DRectNeedle,Angle)
      cvsCompass.DrawBitmap(bmpCompass, RectCompass, RectCompass)
   Else
      btnMode.Text = "Compass turns"
      cvsNeedle.DrawRectRotated(DRectNeedle,Colors.Transparent,True,1,Angle)
      cvsNeedle.DrawBitmap(bmpNeedle, SRectNeedle, DRectNeedle)
   End If
   Angle = Angle - AngleStep
   Timer1_Tick
End Sub
```

- We change the Mode variable from True to False or from False to True with the Not keyword.
- If Mode = True, rotating needle, we:
  - Set the button text to "Needle turns".
  - Draw a transparent rectangle to erase the current needle.
  - Draw the needle at the new position.
  - Draw the default compass.
- If Mode = False, rotating compass, we:
  - Set the button text to "Compass turns".
  - Erase the current needle
  - Draw the new needle.

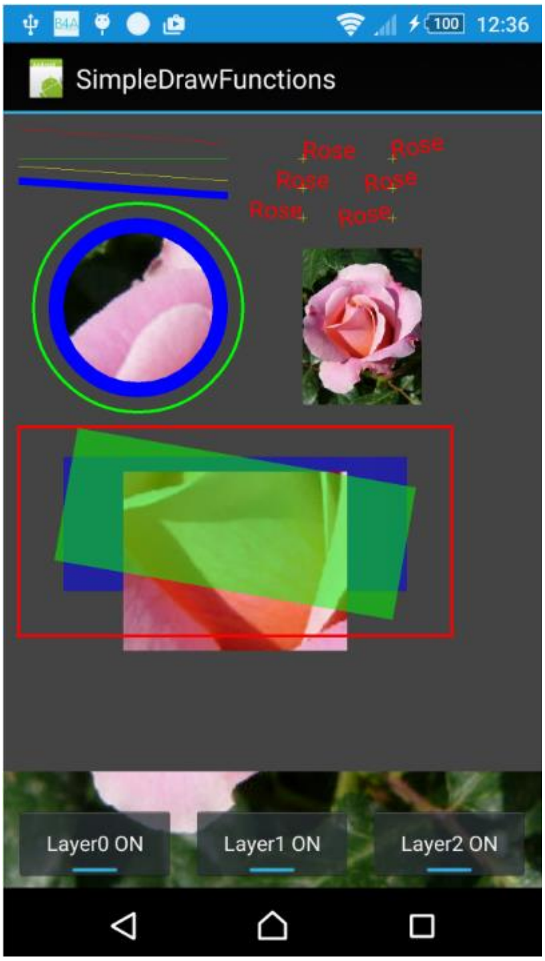
16.2.3 Simple draw functions

The project is in : SourceCode\Graphics\SimpleDrawFunctions\SimpleDrawFunctions.b4a

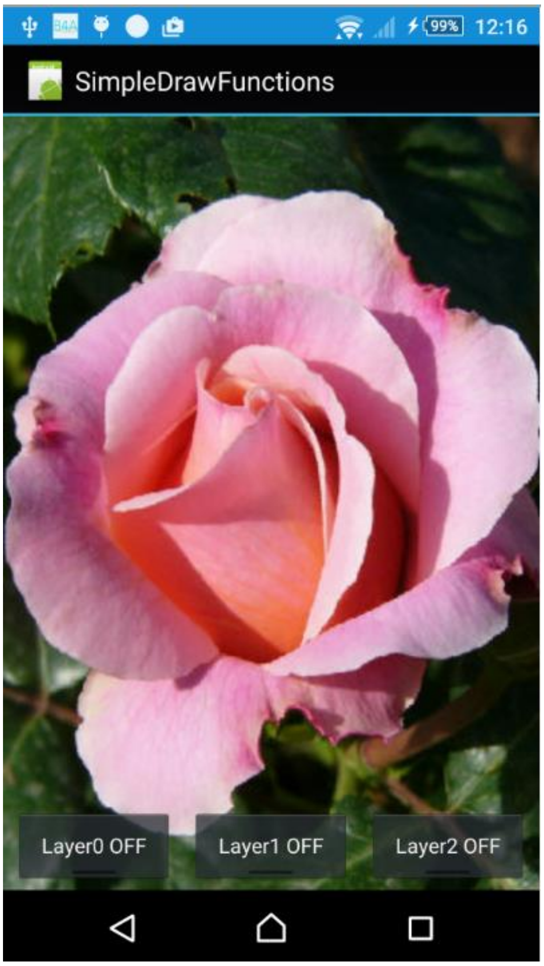
In the third drawing program, SimpleDrawFunctions, we use the other common drawing functions.

The program has no other purpose than to show what can be done with drawings.

The program has three Panels which we use as layers and three ToggleButtons buttons allowing us to show or hide each layer.
Layer(0) has a grey background and the two other layers have a transparent background.

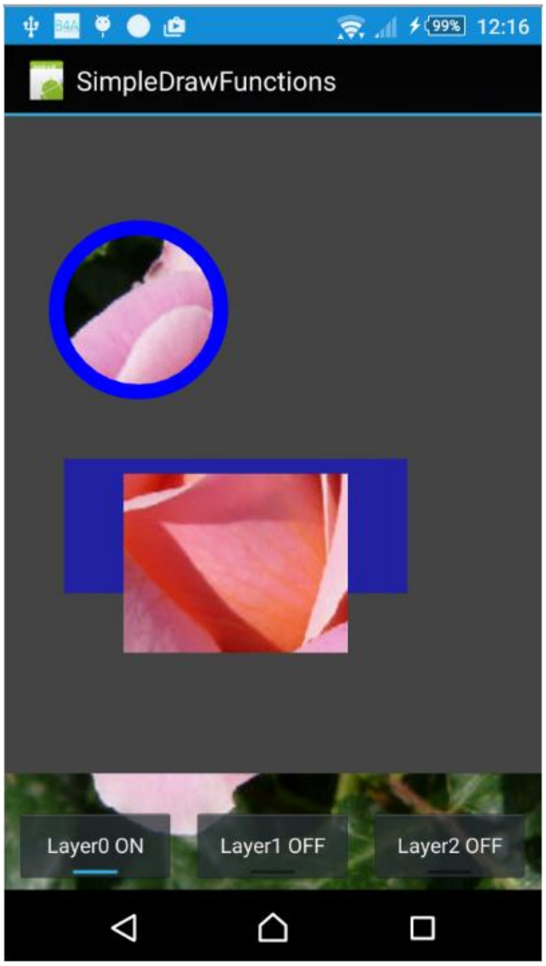


You can play with the buttons to observe the different combinations of visible and hidden layers.



In this screenshot we solely see the background image of the activity.

We use the ToggleButtons to either show or hide the different layers.

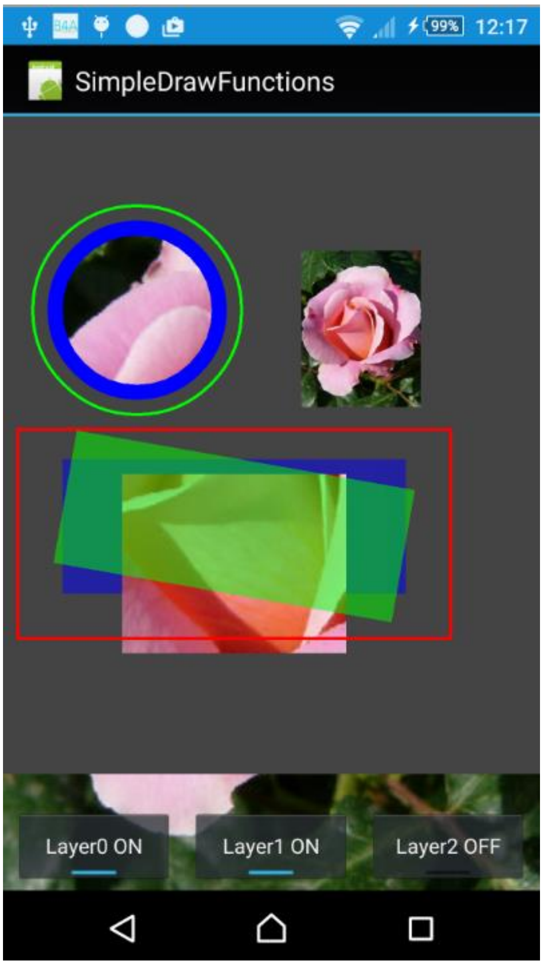


Here we show only layer(0).

The panel has a dark gray background with:

- a blue circle.
- a transparent circle, the activity's background is inside this circle.
- a blue rectangle
- a transparent rectangle, the activity's background is inside this rectangle.

Touching the screen and moving the finger moves the blue and transparent circles on layer(0).

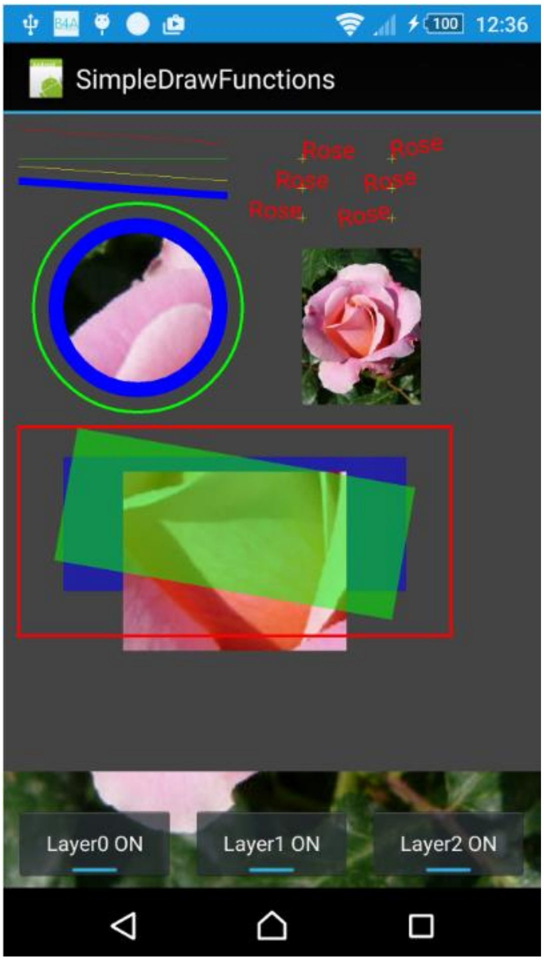


Here we show layer(0) plus layer(1).

The panel has a transparent background with:

- a green circle.
- a small copy of the activity's background image.
- a green, rotated semi-transparent rectangle.

We see that the rectangle covers the activity's background because layer 1 is in front of layer 0.

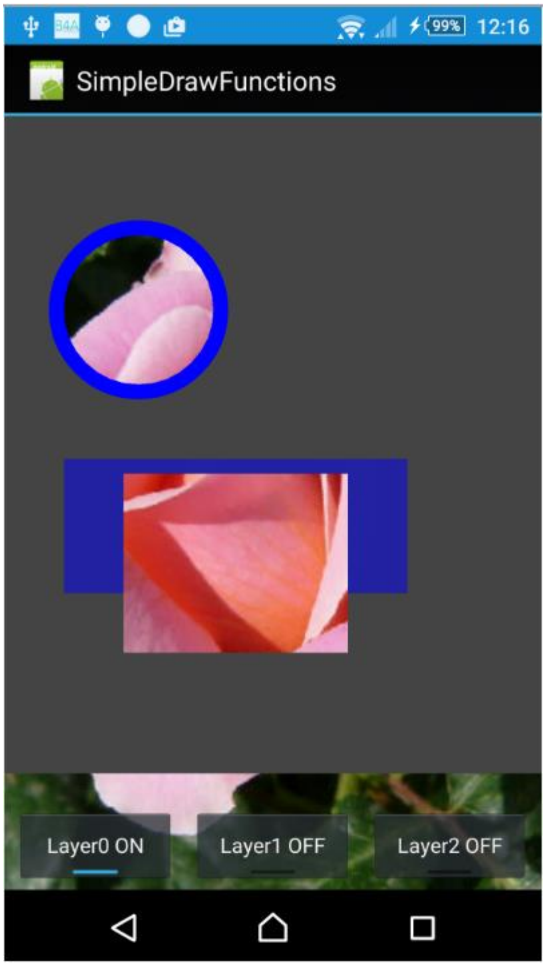


Here we show all three layers.

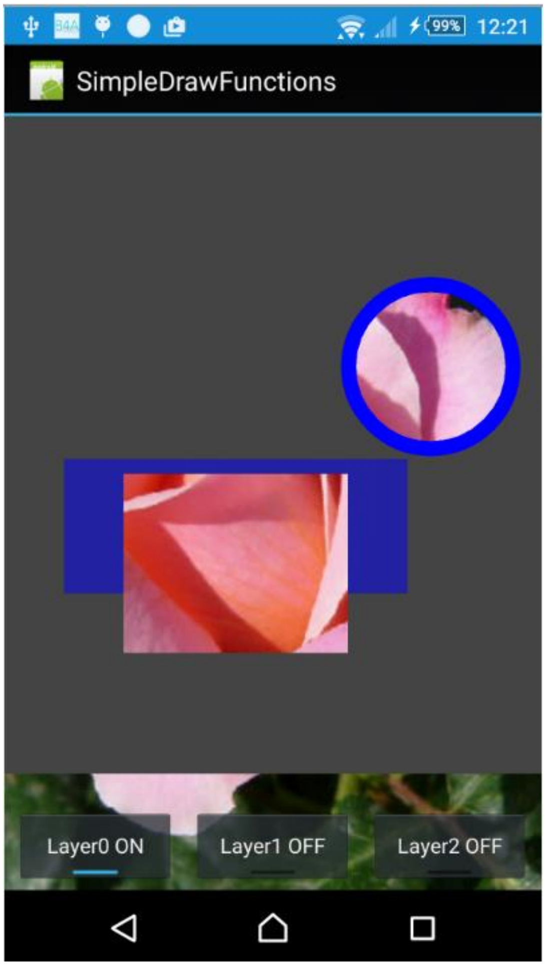
The panel has a transparent background with:

- 4 lines on top.
- 3 horizontal texts with the three different alignments.
- 3 rotated texts with the three different alignments.
- a point for each text showing the position of the reference point.

You can play with the buttons to show the different combinations of visible and hidden layers.



Touching the screen with the finger and moving it, moves the blue and transparent circles.



On each move, the backgound image of the activity appears.

Analysis of the code:

There is no layout file, all views are added by code.

In the Process_Globals routine we declare the bitmap.

```
Sub Process_Globals
   Private bmpBackground As Bitmap
End Sub
```

In the Sub Globals routine we declare the different views and variables:

```
Sub Globals
   Private pnlLayer(3) As Panel
   Private cvsLayer(3) As Canvas
   Private btnLayer(3) As ToggleButton
   Private rect1 As Rect
   Private bdwBackground As BitmapDrawable
   Private xc, yc, x1, y1, x2, y2, r1, r2, h, w As Float
End Sub
```

We have:

- 3 Panels
- 3 Canvases
- 3 ToggleButtons
- 1 Rect, rectangle used to draw rectangles
- 1 Bitmap, holding the activity's background image
- 1 BitmapDrawable, holds the activity's background
- different variables used for the drawing.

Note that we use arrays of views for the three panels, canvases and togglebuttons.
`Private pnlLayer(3) As Panel` instead of `Private pnlLayer0, pnlLayer1, pnlLayer2 As Panel`.

In the Sub Activity_Create routine we initialize the different views and add them to the activity:

```
Sub Activity_Create(FirstTime As Boolean)
   Private i As Int

   If FirstTime Then
      bmpBackground.Initialize(File.DirAssets,"Rose2.jpg")
   End If
   bdwBackground.Initialize(bmpBackground)
   Activity.Background = bdwBackground
```

We:

- initialize the views only if FirstTime = True.
- load the `Rose2.jpg` image file into the bitmap.
- initialize the background image of the activity.
- set the activity's background image

```
      x1 = 2%x
      w = 30%x
      y1 = 100%y - 55dip
      h = 50dip
```

- initialize some variables.

```
        For i = 0 To 2
            pnlLayer(i).Initialize("pnlLayer" & i)
            Activity.AddView(pnlLayer(i), 0, 0, 100%x, 85%y)
            cvsLayer(i).Initialize(pnlLayer(i))
            pnlLayer(i).Tag = i

            btnLayer(i).Initialize("btnLayer")
            x2 = x1 + i * 33%x
            Activity.AddView(btnLayer(i), x2, y1, w, h)
            btnLayer(i).TextOn = "Layer" & i & " ON"
            btnLayer(i).TextOff = "Layer" & i & " OFF"
            btnLayer(i).Checked = True
            btnLayer(i).Tag = i
        Next
    End If
End Sub
```

In a loop we:

- initialize the layer Panels.
  we define an individual EventName for each of the three Panels
  we use only the event for pnlLayer0.
- add the panels to the activity.
- initialize the layer Canvases.
- set the Panels Tag property to the index.

- initialize the layer ToggleButtons.
  we define a single EventName for all three ToggleButtons.
  we manage the showing and hiding of the Panels in one single event routine.
- calculate the left coordinate for each ToggleButton.
- set the texts for the two states.
- set the Checked property to True.
- set the Tag property to the index.

In the Sub Activity_Resume routine we call the Drawing routine.

```
Sub Activity_Resume
    Drawing
End Sub
```

In the Sub Drawing routine we:

```
Sub Drawing
  cvsLayer(0).DrawColor(Colors.DarkGray)
  cvsLayer(1).DrawColor(Colors.Transparent)
  cvsLayer(2).DrawColor(Colors.Transparent)
```

- draw the layout(0) background dark gray.
- draw the layout(1) and layout(2) background transparent.

```
  x1 = 10dip
  y1 = 10dip
  x2 = 150dip
  y2 = 20dip
  cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.Red, 0)
  y1 = 30dip
  y2 = 30dip
  cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.Green, 0.99dip)
  y1 = 35dip
  y2 = 45dip
  cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.Yellow, 0.99dip)
  y1 = 45dip
  y2 = 55dip
  cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.Blue, 5dip)
```

- draw four lines onto layer(2)
  `cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.Red, 0)`
  the last StrokeWidth parameter is '0', this means hairline mode, the width is one pixel.
  `cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.Green, 0.99dip)`
  here we use 0.99dip instead of 1dip because in some cases no line or only parts of it are drawn. This is a known bug in Android with a StrokeWidth of '1'.

```
  xc = 90dip
  yc = 130dip
  r1 = 70dip
  cvsLayer(1).DrawCircle(xc, yc, r1, Colors.Green, False, 2dip)
  r1 = 60dip
  cvsLayer(0).DrawCircle(xc, yc, r1, Colors.Blue, True, 3dip)
  r2 = 50dip
  cvsLayer(0).DrawCircle(xc, yc, r2, Colors.Transparent, True, 1dip)
```

- draw a green circle line on layer(1).
- draw a filled blue circle on layer(0).
- draw a filled transparent circle on layer(0).

```
   rect1.Initialize(10dip, 210dip, 300dip, 350dip)
   cvsLayer(1).DrawRect(rect1, Colors.Red, False, 2dip)
   rect1.Initialize(40dip, 230dip, 270dip, 320dip)
   cvsLayer(0).DrawRect(rect1, Colors.ARGB(128, 0, 0, 255), True, 2dip)
   cvsLayer(1).DrawRectRotated(rect1, Colors.ARGB(128, 0, 255, 0),True, 2dip,10)
   rect1.Initialize(80dip, 240dip, 230dip, 360dip)
   cvsLayer(0).DrawRect(rect1, Colors.Transparent, True, 2dip)
```

- define the coordinates of a rectangle.
- draw a red rectangle on layer(1).
- define the coordinates of a rectangle.
- draw a semi-transparent blue rectangle on layer(0).
- draw a semi-transparent green rotated rectangle on layer(1).
- define the coordinates of a rectangle.
- draw a transparent rectangle on layer(0).
- define the coordinates of a rectangle.
- draw a red rectangle on layer(1).

```
   rect1.Initialize(200dip, 90dip, 280dip, 195dip)
   cvsLayer(1).DrawBitmap(bmpBackground,Null,rect1)
```

Note: `Null` as the source rectangle means the whole bitmap.

- define the coordinates of a rectangle.
- draw the activity's background image in a smaller rectangle on layer(1)

```
   x1 = 200dip
   y1 = 30dip
   cvsLayer(2).DrawText("Rose", x1, y1, Typeface.DEFAULT,16,Colors.Red,"LEFT")
   DrawCross(x1, y1, Colors.Yellow)
   y1 = 50dip
   cvsLayer(2).DrawText("Rose", x1, y1, Typeface.DEFAULT,16,Colors.Red,"CENTER")
   DrawCross(x1, y1, Colors.Yellow)
   y1 = 70dip
   cvsLayer(2).DrawText("Rose", x1, y1, Typeface.DEFAULT,16,Colors.Red,"RIGHT")
   DrawCross(x1, y1, Colors.Yellow)
```

- draw the text "Rose" with the three different possible alignments.
- draw the reference point for each text.

```
   x1 = 260dip
   y1 = 30dip
   cvsLayer(2).DrawTextRotated("Rose", x1,y1,Typeface.DEFAULT,16,Colors.Red,"LEFT",-10)
   DrawCross(x1, y1, Colors.Yellow)
   y1 = 50dip
   cvsLayer(2).DrawTextRotated("Rose", x1,y1,Typeface.DEFAULT,16,Colors.Red,"CENTER",-10)
   DrawCross(x1, y1, Colors.Yellow)
   y1 = 70dip
   cvsLayer(2).DrawTextRotated("Rose", x1,y1,Typeface.DEFAULT,16,Colors.Red,"RIGHT",-10)
   DrawCross(x1, y1, Colors.Yellow)
End Sub
```

- same as above but rotated texts.

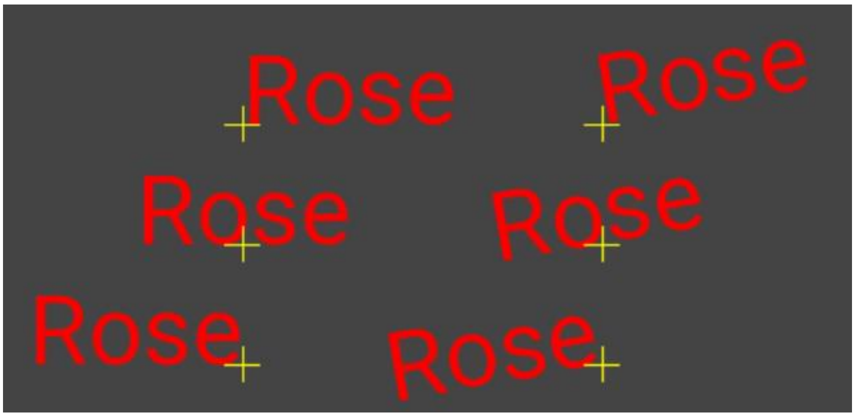
The DrawCross routine:

```
   Sub DrawCross(x As Int, y As Int, color As Int)
      Private d = 3dip As Int

      cvsLayer(2).DrawLine(x - d, y, x + d, y, color, 1)
      cvsLayer(2).DrawLine(x, y - d, x, y + d, color, 1)
   End Sub
```

Looking closer on the displayed texts we see the reference point for each text.

```
   cvsLayer(2).DrawText("Rose", x1, y1, Typeface.DEFAULT,16,Colors.Red,"LEFT")
   DrawCross(x1, y1, Colors.Yellow)
```

These are the x1 and y1 coordinates used to display the texts.



LEFT alignment.

CENTER alignment.

RIGHT alignment.

In the Sub btnLayer_Checked routine we:

```
Sub btnLayer_CheckedChange(Checked As Boolean)
   Private Send As Button

   Send = Sender
   pnlLayer(Send.Tag).Visible = Not(pnlLayer(Send.Tag).Visible)
End Sub
```

- declare a local Button to get the view that raised the event.
- set Send to the Sender view
- change the Visible property from True to False or from False to True.

In the Sub pnlLayer0_Checked routine we:

```
Sub pnlLayer0_Touch (Action As Int, X As Float, Y As Float)
   cvsLayer(0).DrawCircle(xc, yc, r1, Colors.DarkGray, True, 3dip)
   xc = X
   yc = Y
   cvsLayer(0).DrawCircle(xc, yc, r1, Colors.Blue, True, 3dip)
   cvsLayer(0).DrawCircle(xc, yc, r2, Colors.Transparent, True, 1dip)
   pnlLayer(0).Invalidate
End Sub
```

- draw a dark gray circle to erase the previous blue and transparent circle.
- set and yc to the new coordinates of the circle centers.
- draw a blue and transparent circle on layer(1).
- invalidate pnlLayout(1) to force the update of the drawing.

17 VB6 versus B4A

Written by : nfordbscndrd
http://www.b4x.com/android/forum/threads/converting-vb6-to-b4a.9347/#contentcontent

```
  VB6                  B4A
  ===                  ===
controls            Views  (button, edittext, label, etc.)
In the VB6 code window, the top left drop-down list contains all
the controls you have placed in the current form and the right list
contains all the events for each control.  The equivalent in B4A can
be found by clicking on Designer - Tools - Generate Members. Once
you have created Subs in the program coding window, the tab "Modules"
on the right side will list each of the Subs.

In B4A, you start by typing "Sub [viewName]" followed by a space and
follow the prompts, pressing Enter after each selection until B4A
ends with "EventName" highlighted. This is where you would type in
the name of the Sub.

Dim/ReDim:
---------
Dim Array(n)        Dim Array(n+1)
While "n" is the last index number in VB6, it indicates the number
of array elements when used in B4A. For example, to Dim an array
with 0-32 elements in VB6, you would say Dim A(32), while to convert
this to B4A, you need to change it to Dim A(33), yet index #33 is
never used (doing so would cause an out-of-range error).

ReDim Array()       Dim Array(n+1) -- to clear an array, just Dim it again.

[Dim a Int: Dim b as Boolean]
If Not b Then...    If Not(b) Then...
If b Then...        same
If b = True Then    same
If a Then...        If a > 0 Then...
                      B4A does not treat any non-zero value as True like VB6.
a = a + b           If b = True Then a = a - 1
                       Boolean's value cannot be used in a math function in B4A.

Global Const x=1    B4A does not have a Global Const function.
                       In Sub Globals, you can say   Dim x as Int: x = 1
                       but x is not a constant (its value can be changed).

Loops, If-Then, Select Case:
---------------------------
Do [Until/While]    same
Loop [Until/While]  Loop  [Until/While not allowed.]
For - Next          same
For i... - Next i   The loop variable (i) is not allowed with Next.
Exit Do/For         Exit
If - Then - Else    same, except VB's ElseIf is "Else If" in B4A; ditto EndIf

   ---              Continue [Skips to Next in For-Next loop]
For i = 1 to 6        For i = 1 to 6
  If i = 4 Then         If i = 4 Then Continue
    ...code...          ...code...
  End If                ...
Next                  Next

Select Case [expr]  Select [value]
```

```
Colors:
------
L1.BackColor =      L1.Color = Colors.Red
    vbRed
L1.ForeColor =      L1.TextColor = Colors.Black
    vbBlack

Calling a sub:
-------------
SubName x, y       SubName(x, y)

Sub SubName()       Sub SubName() As Int/String/etc.

Function FName()    Sub FName() As [var.type]
   As [var.type]       In B4A, any Sub can be used like a Function by adding a
                       variable type such as
                          Sub CheckX(x As Int) As Boolean
                             ...optional code...
                             If x = [desired value] Then Return True
                             ...optional code...
                          End Sub
                       If no Return is given, then zero/False/"" is returned.
                       The calling code does not have to reference the returned
                       value, so that while "If CheckX(x) = True..." is valid,
                       so is just "CheckX(x)"
Exit Sub            Return
Exit Function       Return [value]

General:
-------
DoEvents            same, except that Erel says:
                    "Calling DoEvents in a loop consumes a lot of resources and
                    it doesn't allow the system to process all waiting messages
                    properly." This was in response to my pointing out that
                    while in a Do Loop with DoEvents in it, WebView could not
                    be loaded or if loaded, would not process a hyperlink click.
                    And Agraham says: "Looping is bad practice on mobile
                    devices. The CPU will be constantly executing code and using
                    battery power as the code will never get back to the OS idle
                    loop where the hardware power saving measures are invoked."

Format()            NumberFormat & NumberFormat2 [see documentation]

InputBox($)         InputList(Items as List, Title, CheckedItem as Int) as Int
                        Shows list of choices with radio buttons. Returns index.
                        CheckedItem is the default.
                    InputMultiList(Items as List, Title) As List
                        Usere can select multiple items via checkboxes.
                        Returns list with the indexes of boxes checked.

MsgBox "text"       MsgBox("text", "title")
i=MsgBox()          MsgBox2(Message, Title, Positive, Cancel, Negative, Icon) as
Int
                        Displays three buttons with text to display for buttons
                           (Positive, Cancel, Negative)
                        Icon is displayed near the title and is specified like:
                           LoadBitmap(File.DirAssets, "[filename].gif")
   ---              ToastMessageShow(text, b) [where b=True for long duration]

Rnd is < 1          Rnd(min, max) is integer >= min to < max

Round(n)            same, or Round2(n, x) where x=number of decimal places


i = Val(string)     If IsNumber(string) Then i = string Else i = 0 --
```

```
                    An attempt to use i=string "throws an exception" if the
string is
                    not numbers.

control.SetFocus    view.RequestFocus

n / 0 : error       n / 0 = 2147483647 -- B4A does not "throw an exception" for
                       division by 0, but it does return 2147483647 no matter
                       what the value of "n" is.

x = Shell("...")    See "Intent". This is not a complete replacement, but allows
                       code such as the following from the B4A forum (by Erel):
                    Dim pi As PhoneIntents
                    StartActivity
(pi.OpenBrowser("file:///sdcard/yourfile.html"))

t = Timer           t = DateTime.Now ' Ticks are number of milliseconds since 1-
1-70

TabIndex:
--------
In VB6, TabIndex can be set to control the order in which controls get focus
when Tab is pressed. According to Erel, in B4A:
   "Android handles the sequence according to their position. You can set
    EditText.ForceDone = True in all your EditTexts. Then catch the
    EditText_EnterPressed event and explicitly set the focus to the next
    view (with EditText.RequestFocus)."

Setting Label Transparency:
--------------------------
Properties - Back Style        Designer - Drawable - Alpha

Constants:
---------
""                  Quote = Chr$(34)
vbCr                CRLF = Chr$(13)
vbCrLf              none

String "Members":
----------------
VB6 uses a character position pointer starting with 1.
B4A uses a character Index pointer starting with 0.

        VB6                        B4A
Mid$("abcde", 1, 1) = "a" = letter array index 0 -- "a" = "abcde".CharAt(0)
Mid$("abcde", 2, 1) = "b" = letter array index 1
Mid$("abcde", 3, 1) = "c" = letter array index 2
Mid$("abcde", 4, 1) = "d" = letter array index 3
Mid$("abcde", 5, 1) = "e" = letter array index 4

     VB6                                B4A
     ===                                ===
Mid$(text, n, 1)                    text.CharAt(n-1)
Mid$(text, n)                       text.SubString(n-1)
Mid$(text, n, x) [x=length wanted]  text.SubString2(n-1, n+x-1) [n+x-1=end
position]
Mid$(text, n, x) = text2            text = text.SubString2(0, n-2) & _
                                           text2.SubString2(0, x-1) & _
                                           text.SubString(n-1 + z)  where...
                                             z = Min(x, text2.length)
Left$(text, n)  [n=num.of chars.]   text.SubString2(0, n)
Right$(text, n)                     text.SubString(text.Length - n + 1)
If a$ = b$...                       If a.CompareTo(b)...
If Right$(text, n) = text2...       If text.EndsWith(text2)...
If Left$(text, n) = text2...        If text.StartsWith(text2)...
If Lcase$(text) = Lcase$(text2)...  If text.EqualsIgnoreCase(text2)...
x = Len(text)                       x = text.Length
text = Replace(text, str, str2)     text.Replace(str, str2)
Lcase(text)                         text.ToLowerCase
Ucase(text)                         text.ToUpperCase
Trim(text)                          text.Trim
  (no LTrim or RTrim in B4A)
Instr(text, string)                 text.IndexOf(string)
Instr(int, text, string)            text.IndexOf2(string, int)
                                       Returns -1 if not found.
                                       Returns char. index, not position.
                                       Starts search at "int".
If Lcase$(x) = Lcase$(y)...         If x.EqualsIgnoreCase(y)...
text = Left$(text, n) & s &         text.Insert(n, s)
          Right$(Text, y)
Asc(s) [where s = a character]      same

Error Trapping:
--------------
VB6:
===
Sub SomeSub
   On [Local] Error GoTo ErrorTrap
      ...some code...
   On Error GoTo 0 [optional end to error trapping]
   ...optional additional code...
   Exit Sub [to avoid executing ErrorTrap code]
ErrorTrap:
   ...optional code for error correction...
   Resume [optional: "Resume Next" or "Resume [line label]".
End Sub

B4A:
===
Sub SomeSub
   Try
      ...some code...
   Catch [only executes if error above]
      Log(LastException) [optional]
      ...optional code for error correction...
   End Try
   ...optional additional code...
End Sub

WIth B4A, if you get an error caught in the middle of a large subroutine, you
can
NOT make a correction and resume within the code you were executing. Only the
code
in "Catch" gets executed. That would seem to make Try-Catch-End Try of use
mainly
during development.

Try-Catch in place of GoTo:
--------------------------
Try-Catch can be used as a substitute for GoTo [line label] for forward, but not
backward, jumps. It cannot be used to replace GoSub, for which B4A has no
equivalent.

Start the code with "Try" and replace the [line label] with "Catch".
Replace "GoTo [line label]" with code which will create an exception, which
causes
a jump to "Catch", such as OpenInput("bad path", "bad filename").

"Immediate Window" vs "Logs" Tab
--------------------------------
```

Comments, variable values, etc., can be displayed in VB6's Immediate Window by entering into the code "Debug.Print ...".

In the B4A environment, the Logs tab on the right side of the IDE is a way to show the values of variables, etc., while the code is running.

Both VB6 and (now) B4A allow single-stepping through the code while it is running and viewing the values of variables. VB6 also allows changing the value of variables, changing the code, jumping to other lines from the current line, etc. Because B4A runs on a PC while the app runs on a separate device, B4A is currently unable to duplicate all of these VB6 debug features.

18 FAQ

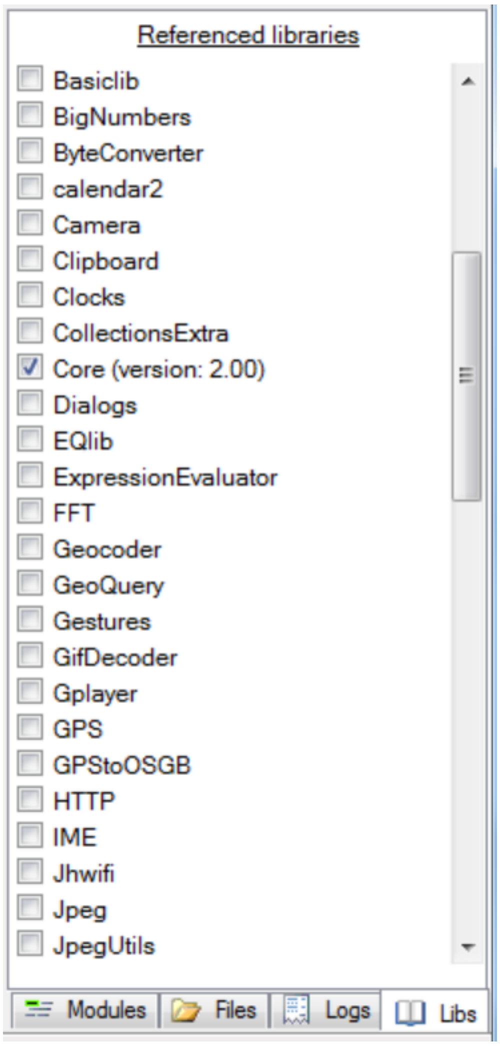
Some of the chapters below have been picked up from the forum.

18.1 "Please save project first" message

When I try to compile or open the Designer I see a message saying: "Please save source code first."
A new project doesn't have a containing folder until it is first saved.
Save your project and this error will go away.

18.2 "Are you missing a library reference" message

Compiler says: "Are you missing a library reference?".

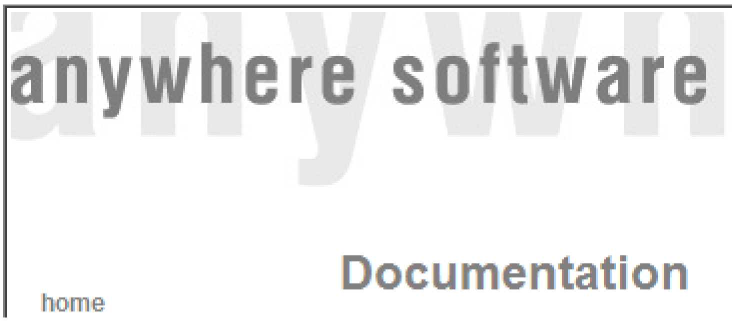


Referenced libraries

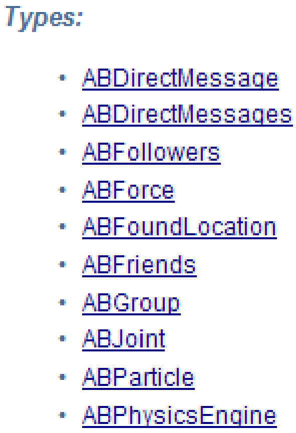
- □ Basiclib
- □ BigNumbers
- □ ByteConverter
- □ calendar2
- □ Camera
- □ Clipboard
- □ Clocks
- □ CollectionsExtra
- ☑ Core (version: 2.00)
- □ Dialogs
- □ EQlib
- □ ExpressionEvaluator
- □ FFT
- □ Geocoder
- □ GeoQuery
- □ Gestures
- □ GifDecoder
- □ Gplayer
- □ GPS
- □ GPStoOSGB
- □ HTTP
- □ IME
- □ Jhwifi
- □ Jpeg
- □ JpegUtils

Go to the Libraries tab in the right pane and check the required libraries.

If you do not know which library a specific object type belongs to, you can go to the documentation page or to the list of additional libraries.



At the bottom of this page there is a long list with all the objects types.



Types:

- ABDirectMessage
- ABDirectMessages
- ABFollowers
- ABForce
- ABFoundLocation
- ABFriends
- ABGroup
- ABJoint
- ABParticle
- ABPhysicsEngine

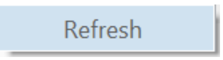
Pressing on any type will take you to the right library.
Note that the trial version doesn't support libraries. Only the full version.

18.3 How loading / updating a library

See the Libraries chapter in the guide.

A list of the official and additional libraries with links to the relevant help documentation can be found on the B4x site in the B4A Documentation page: List of Libraries
To find the library files use a query like http://www.b4x.com/search?query=betterdialogs+library in your internet browser.

To load or update a library follow the steps below:

- Download the library zip file somewhere.
- Unzip it.
- Copy the xxx.jar and xxx.xml files to the
  - B4A Library folder for a standard B4A library
  - Additional libraries folder for an additional library.
- Right click in the libraries list in the Lib Tab and click on  and select the library.

18.4 When do we need to 'Initialize' and when not

View.

For ALL Views:

- To be able to have access to any View by its name you must declare it in the Sub Globals routine.

Views added

- in the Designer in a layout file **MUST NOT** be initialized !
  - Just declare the View in the Sub Globals routine.

    ```
    Sub Globals
      Private lblTitle As Label
    ```

    and nothing else.

- by code it **MUST** be initialized.
  - Declare the View in the Sub Globals routine.

    ```
    Sub Globals
      Private lblTitle As Label
    ```

  - Initialize it and add it to the Activity (or a Panel) in the Activity_Create routine.

    ```
    Sub Activity_Create(FirstTime As Boolean)
      If FirstTime Then
        lblTitle.Initialize("")
        Activity.AddView(lblTitle, 10dip, 10dip, 200dip, 50dip)
      End If
    ```

List / Map. List and Map objects must be initialized before they can be used.

18.5 Split a long line into two or more lines

To split a long line into two or more lines put an underscore character, seperated by a blank character, at the end of the line.

```
Answ = Msgbox2("Do you want to quit the program", "A T T E N T I O N", "Yes", "", "No", Null)
```

Becomes:

```
Answ = Msgbox2("Do you want to quit the program", _
"A T T E N T I O N", "Yes", "", "No", Null)
```

18.6 Avoid closing an application / capture keycodes like Back / Menu

This can be done by intercepting the Activity_KeyPress event.

```
Sub Activity_KeyPress (KeyCode As Int) As Boolean 'Return True to consume the event
  Private Answ As Int

  If KeyCode = KeyCodes.KEYCODE_BACK Then
    Answ = Msgbox2("Do you want to quit the program ?", _
    "A T T E N T I O N", "Yes", "", "No", Null)
    If Answ = DialogResponse.NEGATIVE Then
      Return True
    End If
  End If
  Return False
End Sub
```

- We check if the the KeyCode equals the Back key.
- If yes, we ask the user if he really wants to quit the program.
    - If 'No' we return True to consume the event.
    - Otherwise we return False to transmit the event to the OS.

Just as a reminder, the underscore at the end in the 5th line
`Answ = Msgbox2("Do you want to quit the program ?", _`
means split the line and put the rest on the next line.

ATTENTION :
The Home key cannot be trapped in the `Activity_KeyPress` event routine !

18.7 Unwanted events like Click, Touch or others

Proposed by alfcen:

Suppose you have an Activity containing several buttons with Click events.
Now, you add a Panel onto the Activity, thus covering buttons. As you tap on the panel you will see that a click event was fired on a button on the Activity.
This is NOT a B4A bug, on the contrary, it might be quite useful.
However, if this is not wanted, just add:

```
Sub Panel1_Click
   ' do nothing here or place code to be executed upon tapping on the panel
End Sub
```

18.8 Adding a Menu item

You should also have a look at Example programs / User interfaces.

This is done with the AddMenuItem or AddMenuItem2 methods.
Once a menu item is added you can neither modify it nor remove or disable it.

```
Activity.AddMenuItem("Title", "EventName")
Activity.AddMenuItem("Title", "EventName", image)
```

Examples:

```
Activity.AddMenuItem("Load", "mnuLoad")
Activity.AddMenuItem("Save", "mnuSave", image)
```

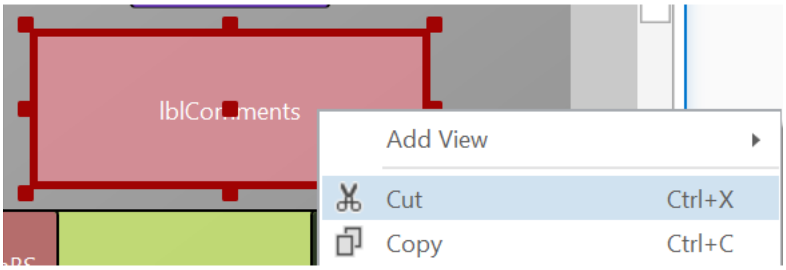
or

```
Activity.AddMenuItem("Load", "mnuLoad", LoadBitmap(File.DirAssets, "Load.png"))
Activity.AddMenuItem("Save", "mnuSave", LoadBitmap(File.DirAssets, "Save.png"))
```

18.9 How do I remove a View with the Designer

To remove a View with the Designer you must:

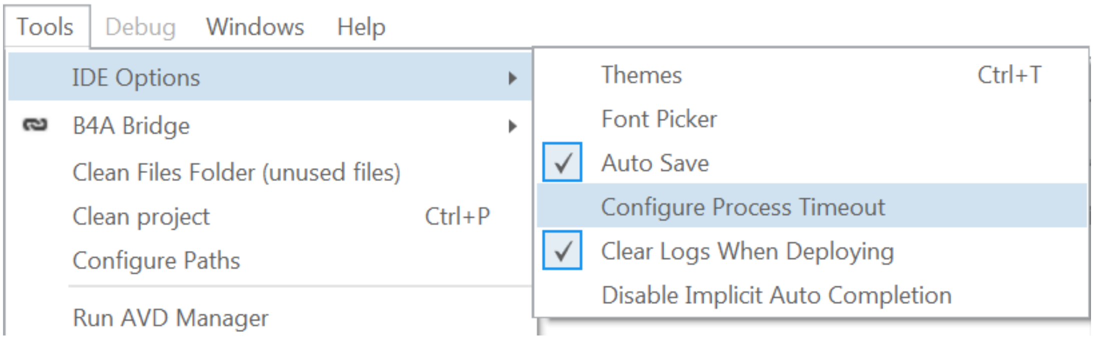
- Select the View to remove either on the device, the Emulator or in the Designer.
- Remove it, right click on the view and in the Popup menu click on Cut.



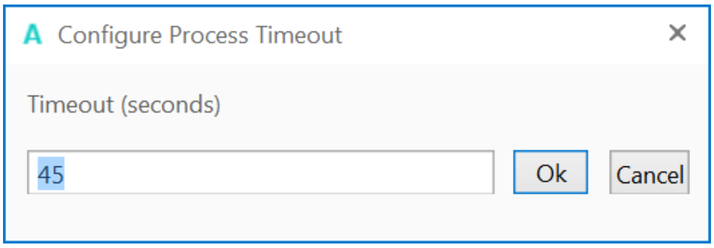
18.10 "Process has timeout" message

If you often get this message "Process has timeout" you can change its value:

- In the IDE menu Tools / IDE Options click on Configure Process Timeout.



- And change the value:


Configure Process Timeout

Timeout (seconds)

45

18.11 Getting a picture from the gallery

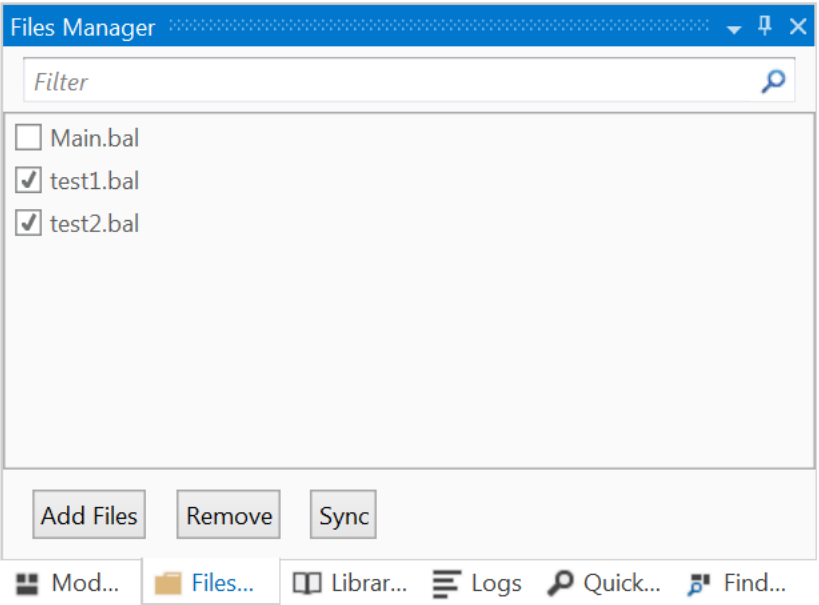
Following code allows you to load a picture from the gallery.

```
Sub Process_Globals
   Private chooser As ContentChooser
End Sub

Sub Globals

End Sub

Sub Activity_Create(FirstTime As Boolean)
   If FirstTime Then
      chooser.Initialize("chooser")
   End If
   chooser.Show("image/*", "Choose image")
End Sub

Sub chooser_Result(Success As Boolean, Dir As String, FileName As String)
   If Success Then
      Private bmp As Bitmap
      bmp.Initialize(Dir, FileName)
      Activity.SetBackgroundImage(bmp)
   Else
      ToastMessageShow("No image selected", True)
   End If
End Sub
```

18.12 How to delete x.bal files or other files from a project

To delete files from the project you must use the Files Tab in the lower right corner of the IDE.

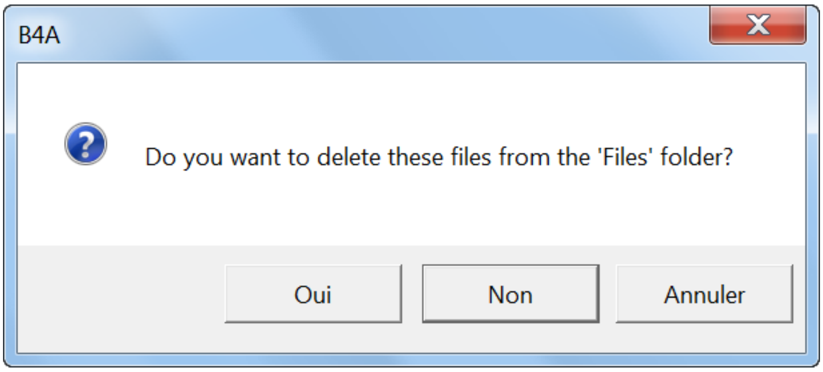
- Select the files you want to delete.



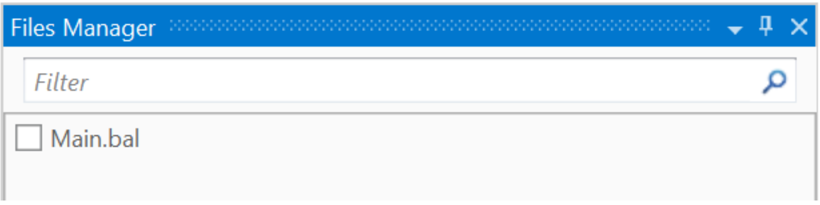
□ Main.bal
☑ test1.bal
☑ test2.bal

- Click on Remove and confirm to delete the files.
  If you delete the files only in the folder, you will get a message for missing files the next time you start the project.

- You will be asked if you want to remove the files from the Files folder.
  Oui > Yes Non > No Annuler > Cancel
  - **Yes** Removes the selected files from the Files folder, be sure that you have backup files somewhere else if you need them afterwards.
  - **No** Removes the files from the project but leaves them in the Files folder.
  - **Cancel** Aborts the function.



Do you want to delete these files from the 'Files' folder?



□ Main.bal

The files are removed.

18.13 Block a screen orientation

To block the orientation either to Portrait or to Landscape.
This is valid for the whole project.
To define different screen orientations for different activities you must do in the code see below.

```
#Region  Project Attributes
   #ApplicationLabel: MyFirstProgram
   #VersionCode: 1
   #VersionName:
   'SupportedOrientations possible values: unspecified, landscape or portrait.
   #SupportedOrientations: unspecified
   #CanInstallToExternalStorage: False
#End Region
```

On top of the Main module in the Project Attributes you can define the supported orientations.

You can define screen orientations in the code with `SetScreenOrientation` from the Phone library:

- Landscape
  `Phone1.SetScreenOrientation(0)`

- Portrait
  `Phone1.SetScreenOrientation(1)`

- Both
  `Phone1.SetScreenOrientation(-1)`

18.14 Close second Activity

From the forum:
Referring to the 'twoactivities' tutorial by Erel, I noticed that when back button was pressed from the main Activity, Activity2 was then shown again.

In the code of Activity2 after `StartActivity(Main)` add `Activity.Finish`.

```
   StartActivity(Main)
   Activity.Finish
```

18.15 Taking a screenshot programaticaly

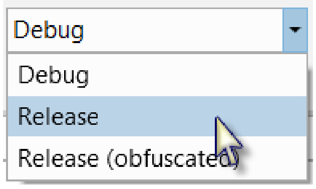
You can take a screenshot of the device or the Emulator with following code:
Needs the Reflection library.

```
Sub btnScrShot_LongClick
   ' Take a screenshot.
   Private Obj1, Obj2 As Reflector
   Private bmp As Bitmap
   Private c As Canvas
   Private now, i As Long
   Private dt As String
   DateTime.DateFormat = "yyMMddHHmmss"
   now = DateTime.now
   dt = DateTime.Date(now) ' e.g.: "110812150355" is Aug.12, 2011, 3:03:55 p.m.
   Obj1.Target = Obj1.GetActivityBA
   Obj1.Target = Obj1.GetField("vg")
   bmp.InitializeMutable(Activity.Width, Activity.Height)
   c.Initialize2(bmp)
   Private args(1) As Object
   Private types(1) As String
   Obj2.Target = c
   Obj2.Target = Obj2.GetField("canvas")
   args(0) = Obj2.Target
   types(0) = "android.graphics.Canvas"
   Obj1.RunMethod4("draw", args, types)
   Private Out As OutputStream
   Out = File.OpenOutput(File.DirRootExternal, dt & ".png", False)
   bmp.WriteToStream(Out, 100, "PNG")
   Out.Close
End Sub
```

For a screenshot of a Panel, even a ScrollView.Panel replace
`Obj1.Target = Obj1.GetField("vg")` by
`Obj1.Target = Panel1` or `Obj1.Target = ScrollView1.Panel`

18.16 After compiling, where are the files

The compiler generates an *.apk file which is located in the 'Objects' folder of your project. When the IDE is connected to a device or to the Emulator the apk file is automaticaly uploaded to it.
The name of the apk file is the 'Application Label' you entered when you defined the project.
Example: GPSExample.apk
If you have compiled in Debug mode you will get an apk file with the _DEBUG suffix.
Example: GPSExample_DEBUG.apk
If you want to distribute your application you should select Release or Release (obfuscated).



18.17 Run an application from another one

Erels' answer to the question:
You can start any application by sending the correct Intent.
The easiest way to see the required Intent is to look at the unfiltered logs while manually starting the application.

The code below shows how to run an application from another one.
The PackageManager is an object in the Phone library.
The exact package name is needed !

```
Private pm As PackageManager
Private in As Intent

in.Initialize("", "")
in = pm.GetApplicationIntent

If in.IsInitialized Then
   StartActivity(in)
End If
```

18.18 How to pass an Array to a Sub

It is possible to pass Arrays, also multidimensional Arrays, to a sub.

Code example.

```
   Private one(1), two(1,2), three(1,2,3) As String

Sub Test(a() As String, b(,) As String, c(,,) As String) As String(,)
   ...
End Sub
'
'
   Test(one, two, three)
```

You need to specify the rank (number of dimensions) in the Sub definition with ',' .
If you want the Sub to return an array you must also speccify it.

```
Sub Test(a() As String, b(,) As String, c(,,) As String) As String
```

Returns a single string.

```
Sub Test(a() As String, b(,) As String, c(,,) As String) As String()
```

Returns a one rank string array.

```
Sub Test(a() As String, b(,) As String, c(,,) As String) As String(,)
```

Returns a two rank string array.

18.19 Getting language and country from device

You can get the current language and country from a device with the following code.

```
Sub Activity_Create(FirstTime As Boolean)
   Log(GetDefaultLanguage)
End Sub

Sub GetDefaultLanguage
   Private r As Reflector
   r.Target = r.RunStaticMethod("java.util.Locale", "getDefault", Null, Null)
   Return r.RunMethod("getDisplayName")
End Sub
```

`GetDefaultLanguage` returns a string with the language and the country.
Note: `getDisplayName` is case sensitive!

Needs the Reflection library (available only for users who bought B4A) !

Examples:

- English (United States)
- Deutsch (Österreich)
- français (Suisse)

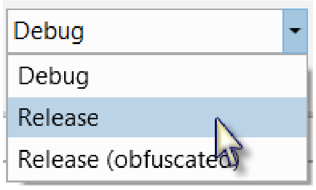
18.20 Where is the apk file

Where is the apk file:
The apk file is located in the Objects folder of your project.

18.21 Why is my apk filename result.apk

The filename is the same as the main project filename but instead of the .b4a suffix it has the .apk suffix.
If you enter non authorized characters, like a space, the apk filename will be **result.apk**.
The apk name has no importance, the displayed name is the Label name you gave when you created the project.

18.22 Why is my apk filename xxx_DEBUG.apk



To distribute a program you must select in the IDE, in the Dropdown list, Release or Release (obfuscated).

18.23 Select True / Case trick

The question : It would be nice to be able to use Select Case using the 'greater than' and 'less than' operators <>. It makes for cleaner code than 'if' 'else' and 'end if' etc.

This trick does it:

```
i = 10
Select True
Case (i < 9)
   Log("False")
Case (i = 10)
   Log("True")
End Select
```

18.24 Fill an array with random numbers without repetition

This code snippet from Erel is based on the Fisher-Yates shuffle algorithm.

```
Sub Globals
   Private numbers(10) As Int
End Sub

Sub Activity_Create(FirstTime As Boolean)
   'put numbers 1 - 10 in the array
   For i = 0 To 9
      numbers(i) = i + 1
   Next
   ShuffleArray(numbers)
   For i = 0 To 9
      Log(numbers(i)) 'print the numbers to the log
   Next
End Sub

Sub ShuffleArray(arr() As Int)
   Private i As Int

   For i = arr.Length - 1 To 0 Step -1
      Private j, k As Int
      j = Rnd(0, i + 1)
      k = arr(j)
      arr(j) = arr(i)
      arr(i) = k
   Next
End Sub
```

18.25 Detect screen orientation

The code below detects the screen orientaion, comparing if the activity width is greater than the activity height then the orientation is "landscape" otherwise it's "portrait".

```
Sub Globals
   Private Orientation As String
End Sub

Sub Activity_Create(FirstTime As Boolean)
   If Activity.Width > Activity.Height Then
      Orientation = "Landscape"
   Else
      Orientation = "Portait"
   End If
End Sub
```

18.26 Some functions don't work in Activity_Pause

Any function that stops the program is not allowed in the Activity_Pause routine.
Like :

Message boxes	MsgBox
Modal dialogs	InputDialog, FileDialog etc.
Custom dialogs	
Breakpoints	

Log and ToastMessage are allowed.

18.27 Calling the internal Calculator

The subroutine below calls the internal calculator.

```
Sub Calculator
   Private i As Intent
   i.Initialize("", "")
   i.SetComponent("com.android.calculator2/.Calculator")
   Try
      StartActivity(i)
   Catch
      ToastMessageShow("Calculator app not found.", True)
   End Try
End Sub
```

Note that `"com.android.calculator2/.Calculator"` is case sensitive !
This would throw an error: `"com.android.calculator2/.calculator"`

Some manufacturers change the name of the internal calculator.
The code below overcomes this problem.
Code provided by dxxxyyyzzz on the forum.
Needs the Phone Library.

```
Private Pm As PackageManager
Private Inte As Intent
Private Packages As List
Private st As String

Packages = Pm.GetInstalledPackages

For i = 0 To Packages.size - 1
   st = Packages.Get(i)
   If st.Contains("calc") = True Then
      Inte=Pm.GetApplicationIntent(st)
      If Inte.IsInitialized Then
         StartActivity(Inte)
         Exit
      End If
   End If
Next
```

18.28 Get the Alpha / Red / Green / Blue values

```
Sub Activity_Create(FirstTime As Boolean)
   Private argb() As Int
   argb = GetARGB(Colors.Transparent)
   Log("A = " & argb(0))
   Log("R = " & argb(1))
   Log("G = " & argb(2))
   Log("B = " & argb(3))
End Sub

Sub GetARGB(Color As Int) As Int()
   Private res(4) As Int
   res(0) = Bit.UnsignedShiftRight(Bit.And(Color, 0xff000000), 24)
   res(1) = Bit.UnsignedShiftRight(Bit.And(Color, 0xff0000), 16)
   res(2) = Bit.UnsignedShiftRight(Bit.And(Color, 0xff00), 8)
   res(3) = Bit.And(Color, 0xff)
   Return res
End Sub
```

In line `Sub GetARGB(Color As Int) As Int()` the `()` after `Int` are necessary because the return value is an array.

18.29 Get device type

```
Sub Activity_Create(FirstTime As Boolean)
   If GetDevicePhysicalSize > 6 Then
      '7'' or 10'' tablet
   Else
      'phone
   End If
End Sub

Sub GetDevicePhysicalSize As Float
   Private lv As LayoutValues
   lv = GetDeviceLayoutValues
   Return Sqrt(Power(lv.Height, 2) + Power(lv.Width, 2)) / lv.Scale / 160
End Sub
```

18.30 Generate a Click event

```
Sub Globals
   Private sp As Spinner
End Sub

Sub Activity_Create(FirstTime As Boolean)
   sp.Initialize("sp")
   sp.AddAll(Array As String("a", "b", "c", "d"))
   Activity.AddView(sp, 10dip, 10dip, 200dip, 50dip)
End Sub

Sub Activity_Click
   OpenSpinner(sp)
End Sub

Sub OpenSpinner(s As Spinner)
   Private r As Reflector
   r.Target = s
   r.RunMethod("performClick")
End Sub
```

18.31 "Out of memory" Error / Bitmaps

Under certain circumstances the program stops with "Out of memory" message.
This can happen when there are big bitmaps or many bitmaps.

The code below disposes the bitmap and frees the memory (code provided by agraham here)

```
Private Obj1 As Reflector
Obj1.Target = bmp ' bmp is the unwanted Bitmap
Obj1.RunMethod("recycle")
```

For a bitmaps of a canvas:

```
Obj1.Target = canv
Obj1.Target = Obj1.GetField("bw")
Obj1.Target = Obj1.RunMethod("getObject")
Obj1.RunMethod("recycle")
```

18.32 Get consumed memory

The routines below get different memory values.
Needs the Reflection library

Get the max memory for the application.

```
Private Sub GetMaxMemory As Double
  Private r As Reflector
  r.Target = r.RunStaticMethod("java.lang.Runtime", "getRuntime", Null, Null)
  Return (r.RunMethod("maxMemory") / (1024*1024))
End Sub
```

Get the currently total consumed memory.

```
Private Sub GetTotalMemory As Double
  Private r As Reflector
  r.Target = r.RunStaticMethod("java.lang.Runtime", "getRuntime", Null, Null)
  Return (r.RunMethod("totalMemory") / (1024*1024))
End Sub
```

Get the currently available free memory.

```
Private Sub GetFreeMemory As Double
  Private r As Reflector
  r.Target = r.RunStaticMethod("java.lang.Runtime", "getRuntime", Null, Null)
  Return (r.RunMethod("maxMemory") - r.RunMethod("totalMemory")) / (1024*1024)
End Sub
```

18.33 Remove the scrollbar from a ScrollView

The code needs the Reflection library.

```
Private r As Reflector
r.Target = ScrollView1
r.RunMethod2("setVerticalScrollBarEnabled", False, "java.lang.boolean")
```

18.34 Check if directory exists

```
   Private MyDirctory As String

   MyDirctory = File.DirRootExternal & "/Images"
   If File.Exists(MyDirctory, "") = False Then
      File.MakeDir(File.DirRootExternal, "Images")
   End If
```

18.35 Set Full Screen in code

```
Sub FullScreen(Active As Boolean, ActivityName As String)
   Private obj1 As Reflector
   Private i As Int

   i = 1024  'FLAG_FULLSCREEN
   obj1.Target = obj1.GetMostCurrent(ActivityName)
   obj1.Target = obj1.RunMethod("getWindow")
   If Active Then
      obj1.RunMethod2("addFlags",i,"java.lang.int")
   Else
      obj1.RunMethod2("clearFlags",i,"java.lang.int")
   End If
End Sub
```

`Active = True`	sets full screen.
`Active = False`	set back to show the status bar.

18.36 Change EditText input modes

The EditText view has several default input modes.

`INPUT_TYPE_NONE`	No input allowed.
`INPUT_TYPE_NUMBERS`	Allows only integer numbers.
`INPUT_TYPE_DECIMAL_NUMBERS`	Allows only decimal numbers.
`INPUT_TYPE_TEXT`	Allows any type of text.
`INPUT_TYPE_PHONE`	Allows phone numbers.

Example:

```
EditText1.InputType = EditText1.INPUT_TYPE_TEXT
```

Other flags are available: A complete list can be found here.

INPUT_TYPE_TEXT flag combinations:

- TYPE_TEXT_FLAG_CAP_CHARACTERS Constant Value: 4096 (0x00001000)
  Sets to upper case characters.
  ```
  EditText1.InputType = Bit.Or(EditText1.INPUT_TYPE_TEXT, 4096)
  ```

- TYPE_TEXT_FLAG_CAP_SENTENCES Constant Value: 16384 (0x00004000)
  Sets the first character of a sentence to upper case.
  ```
  EditText1.InputType = Bit.Or(EditText1.INPUT_TYPE_TEXT, 16384)
  ```

- TYPE_TEXT_FLAG_CAP_WORDS Constant Value: 8192 (0x00002000)
  Sets the first character of all words to upper case.
  ```
  EditText1.InputType = Bit.Or(EditText1.INPUT_TYPE_TEXT, 8192)
  ```

- TYPE_TEXT_FLAG_NO_SUGGESTION Constant Value: 524288 (0x00080000)
  Sets to no suggestion.
  ```
  EditText1.InputType = Bit.Or(EditText1.INPUT_TYPE_TEXT, 524288)
  ```

- TYPE_TEXT_FLAG_AUTO_COMPLETE Constant Value: 65536 (0x00010000)
  Sets auto complete.
  ```
  EditText1.InputType = Bit.Or(EditText1.INPUT_TYPE_TEXT, 65536)
  ```

- TYPE_TEXT_FLAG_AUTO_CORRECT Constant Value: 32768 (0x00008000)
  Sets auto correct.
  ```
  EditText1.InputType = Bit.Or(EditText1.INPUT_TYPE_TEXT, 32768)
  ```

- TYPE_TEXT_VARIATION_EMAIL_ADDRESS Constant Value: 32 (0x00000020)
  Sets for e-mail address.
  ```
  EditText1.InputType = Bit.Or(EditText1.INPUT_TYPE_TEXT, 8192)
  ```

- TYPE_TEXT_VARIATION_PASSWORD Constant Value: 128 (0x00000080)
  Sets password mode.
  ```
  EditText1.InputType = Bit.Or(EditText1.INPUT_TYPE_TEXT, 128)
  ```

INPUT_TYPE_NUMBERS flag combinations:

- TYPE_NUMBER_SIGNED Constant Value: 4096 (0x00001000)
  Allows signed integer numbers
  ```
  EditText1.InputType = Bit.Or(EditText1.INPUT_TYPE_NUMBERS, 4096)
  ```

18.37 Sorting a file list according to last modified time

Code supplied by Erel in the forum.

```
Sub Process_Globals
   Type FileAndTime(Name As String, Time As Long)
End Sub

Sub Globals

End Sub

Sub Activity_Create(FirstTime As Boolean)
   Private files As List
   files = ListFilesByDate(File.DirRootExternal)
   For i = 0 To files.Size - 1
       Private fs As FileAndTime
       fs = files.Get(i)
       Log(fs.Name & ": " & DateTime.Date(fs.Time))
   Next
End Sub

Sub ListFilesByDate(Folder As String) As List
   Private files As List
   files = File.ListFiles(Folder)
   Private sortedFiles As List
   sortedFiles.Initialize
   For i = 0 To files.Size - 1
       Private fs As FileAndTime
       fs.Name = files.Get(i)
       fs.Time = File.LastModified(Folder, fs.Name)
       sortedFiles.Add(fs)
   Next
   sortedFiles.SortType("Time", False)
   Return sortedFiles
End Sub
```

18.38 Get the dpi values of the device (dots per inch)

Needs the Reflection library.

```
Private Xdpi,Ydpi As Float
Private r As Reflector
r.Target = r.GetContext
r.Target = r.RunMethod("getResources")
r.Target = r.RunMethod("getDisplayMetrics")
Xdpi = r.GetField("xdpi")
Ydpi = r.GetField("ydpi")
```

The different fields are :

- density The logical density of the display.
- densityDpi The screen density expressed as dots-per-inch.
- heightPixels The absolute height of the display in pixels.
- widthPixels The absolute width of the display in pixels.
- scaledDisplay A scaling factor for fonts displayed on the display.
- xdpi The exact physical pixels per inch of the screen in the X dimension.
- ydpt The exact physical pixels per inch of the screen in the Y dimension.

18.39 Finding java program lines

Sometimes a program raises java execution error messages with the subname and a java line number.

To find the given line :

- look at the Objects\src\packagename for the activity.java file.
- open it in a text editor showing line numbers (like notepad++)
- look at the given line number and you find the offending code.

Advice given by warwound (Martin Pearman).

19 Glossary

Android Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Google Inc. purchased, in 2005, Android Inc. the company that initially developed the software.

Java Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities.

Activity An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI.

View Provides classes that expose basic user interface classes that handle screen layout and interaction with the user. Examples: Label, Panel, Button, EditText etc.

20 Index

Will be added it he next update.