**Conocimiento: Polimorfismo**

Polimorfismo significa en gramática múltiples formas.

En programación orientada al objeto hay varias formas de polimorfismo:

1. Cuando invocamos el mismo nombre de método sobre instancias de distinta clase
2. Cuando creamos múltiples constructores
3. Cuando vía subtipo asignamos una instancia de una subclase a una referencia a la clase base.
4. Cuando creamos una clase derivada, gracias a la relación es-un podemos utilizar instancias de la clase derivada donde se esperaba una instancia de la clase base. También se conoce como principio de sustitución.

El objetivo del polimorfismo es la reutilización y flexibilidad que se puede obtener de las diferentes clases. Además de la facilidad para adaptarse a cambios

Permite código independiente del tipo, y es fácil de escribir y entender.

Al añadir nuevos tipos, tenemos la ventaja de que:

* No hay que reescribir código
* Los programas son extensibles

El polimorfismo se consigue a través de:

* + - Interfaces
    - Clases abstractas
    - Métodos polimórficos (sobrescritura y sobrecarga)
    - Implementación de Colecciones genéricas
    - Implementación de Métodos genéricos.

Por ejemplo:

Sea:   
public class Empleado { ..... }  
public class Manager extends Empleado { .... }

* Empleado e;  
  e=new Empleado (...); // instancia,   
  e=new Manager (...); // OK. Sustitución

Empleado

e

nombre

sueldo

diaIngreso

Empleado

Empleado

e

Manager

nombre

sueldo

diaIngreso

bono

Empleado

Sólo tenemos acceso a atributos de Empleado, pero con la implementación de Manager

**Enlace dinámico**

Es importante entender qué método es usado al invocar a un nombre que se puede referir a instancias de distinta clase. Al momento de la compilación, el compilador intenta resolver el método que corresponde según su nombre y parámetros. Si la superclase y la clase base tienen definido el mismo método, el problema es a cuál llama.

Si el método en la clase declarada para la variable no es privado, static, o final, se invocará en forma dinámica. Esto es, se invocará el método definido según el objeto referenciado por el nombre y no según la declaración del nombre.

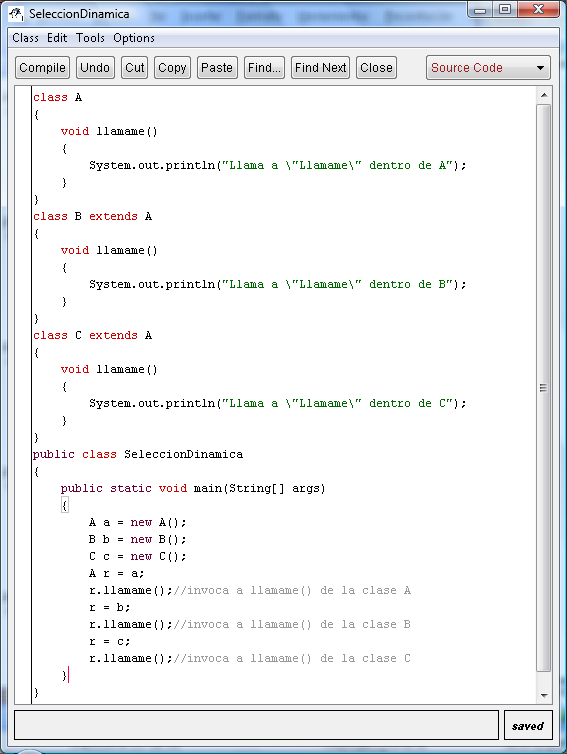
Por ello, si una clase derivada redefine el mismo método, éste será invocado para sus instancias.

El ligado o enlace dinámico se resuelve a tiempo de ejecución.

**Ventajas del enlace dinámico**:

1. La sobrescritura de métodos permite a Java admitir el polimorfismo en tiempo de ejecución.
2. El polimorfismo es esencial en la programación orientada al objeto ya que permite que una clase general especifique métodos que serán comunes a todas las clases derivadas, permitiendo a éstas definir la implementación específica de alguno de estos métodos.
3. Esta forma de polimorfismo dinámico durante la ejecución es uno de los mecanismos más poderosos que ofrece la orientación a objeto para soportar reutilización de código y robustez.
4. La capacidad que tienen las bibliotecas de código existentes de llamar métodos de instancias nuevas de clases sin volver a compilarlos, pero manteniendo a vez la interfaz abstracta y limpia, es una herramienta poderosa.

Por ejemplo: veamos el siguiente código, fijarse en las asignaciones que son las que la JVM debe revisar al momento de invocar el método adecuado:



El resultado de la ejecución del método **main** es:

