**Interfaz Collection**

La interfaz más importante es Collection. Una Collection es todo aquello que se puede recorrer (o "iterar") y de lo que se puede saber el tamaño. Muchas otras clases extenderán de Collection imponiendo más restricciones y dando más funcionalidades. Es de notar que el requisito de "que se sepa el tamaño" hace inconveniente utilizar estas clases con colecciones de objetos de las que no se sepa "a priori" la cantidad (esto podría considerarse una limitación de este framework).

Jerarquía de Clases



Las operaciones básicas de una collection son:

* **add (T):** este método añade un elemento a la colección.
* **iterator ():** este método obtiene un "iterador" que permite recorrer la colección visitando cada elemento una vez.
* **size ():** este método permite obtener la cantidad de elementos que esta colección almacena.
* **contains (t):** este método permite verificar si el elemento t ya está dentro de la colección.

**Interfaz Set**

Esta interfaz define un conjunto. Set extiende de Collection y declara el comportamiento de una colección que no permite elementos duplicados. Por tanto el método add() devuelve falso si se intenta agregar un elemento duplicado a un conjunto. No define ningún método adicional propio.

Su orden de iteración depende de la implementación que los elementos hagan de la interfaz java.lang.Comparable, a través del método int compareTo(Object o)

Existen distintas implementaciones de esta interfaz, que constituyen clases en java:

* HashSet
* LinkedHashset

**Interfaz SortedSet**

Esta interfaz extiende de Set y declara el comportamiento de un conjunto ordenado en orden ascendente.

Esta interfaz agrega otros métodos tales como:

* **E first ():** este método retorna el primer elemento de la colección ordenada.
* **E last ():** este método retorna el último elemento de la colección ordenada.
* **SortedSet subset (E inicio, E fin):** este método obtiene un subconjunto de la colección ordenada, especificando el primer y último elemento
* **SortedSet headSet(E fin):** este método permite obtener un subconjunto desde el comienzo de la colección hasta el elemento especificado por fin
* **SortedSet tailSet(E inicio):** este método permite obtener un subconjunto desde el elemento inicio hasta el último elemento de la colección.

En java la clase TreeSet implementa esta interfaz.

**Interfaz List**

El interfaz List hereda del interfaz Collection. Representa colecciones con elementos en secuencia, es decir, con orden. Además permite tener duplicados.

Es accesible mediante índice, de manera que se puede tanto acceder a un elemento concreto de una posición, o insertar un elemento en una posición concreta.

Los métodos que añade de este interfaz son basados en acceso posicional:

* **Object get (int indice)**: este método devuelve el elemento de esa posición.
* **Object set (int indice, Object elemento)**: este método permite reemplazar el elemento de esa posición con ese elemento.
* **void add (int indice, Object elemento):** este método permite insertar ese elemento en esa posición.
* **Object remove (int indice):** este método permite eliminar el elemento de esa posición.

Existen distintas implementaciones de esta interfaz, que constituyen clases en java:

* ArrayList
* LinkedList
* Vector

**Interfaz Queue**

Esta interfaz se incorporó en la versión 5.0 del JDK. Extiende de Collection y declara el comportamiento de una lista en la que a menudo el primer elemento en entrar es el primero en salir.

Define los siguientes métodos:

* **E element ():** este método permite obtener el primer elemento de la lista, sin eliminarlo.
* **boolean offer (E obj):** este método intenta agregar un elemento a la lista.
* **E peek ():** este método devuelve el elemento del principio de la lista, este no se elimina.
* **E poll ():** este método devuelve el elemento del principio de la lista eliminándolo de la colección
* **E remove():** este método devuelve el elemento del principio de la lista sin eliminarlo de la colección

La clase PriorityQueue implementa esta interfaz.