Colecciones en Java

ÍNDICE

1. Colecciones en Java

El API de Java nos proporciona el *framework* para el trabajo con colecciones de datos (conjunto de datos), el cual nos permite utilizar diferentes estructuras de datos para almacenar y recuperar datos de cualquier tipo. Dichas colecciones están definidas en el paquete `java.util`. Para crear una colección usaremos la siguiente estructura:

```
Coleccion<TipoDato> nombreVariable = new Coleccion<TipoDato>();
```

En el anterior código, "***Coleccion***" hace referencia a la clase del *framework* que queramos utilizar según la estructura de almacenamiento que nos interese. Por ejemplo, para crear una lista ordenada de objetos de la clase `String` implementaríamos lo siguiente:

```
List<String> listaNombres =  new ArrayList<String>();
```

Vamos a utilizar los siguientes dos tipos de colecciones, cada una con una interfaz común y diferentes implementaciones. Las diferentes implementaciones de una misma interfaz realizan la misma tarea aunque la diferencia está en que unas implementaciones son más rápidas en algunas operaciones y más lentas en otras. Tenemos las siguientes:

- Listas
- Conjuntos

Además, aprenderemos a manejar otra estructura de datos dinámica llamada diccionario (en Java también llamados mapa) que nos será muy útil.

¿Para qué vamos a utilizar las colecciones? Cuando deseemos almacenar un conjunto de datos pero no sabemos el número de ellos que vamos a tener. "*El concepto es similar a los arrays pero sin la limitación de únicamente poder tener N elementos como máximo*".

2. Listas

Las listas son estructuras secuenciales (un elemento va detrás de otro), donde cada elemento tiene un índice o posición. Se utiliza la interfaz **`List<E>`**, donde E es la clase o tipo de dato. Podemos utilizar las siguientes implementaciones:

- **ArrayList**<E>: tienen un acceso rápido.
- **LinkedList**<E>: proveen inserciones y borrados rápidos.
- **Stack**<E>: permiten implementar pilas.
- **Vector**<E>: obsoleto.

3. Conjuntos

Son colecciones en las que los elementos de la colección no tienen un orden y no se permiten duplicados. Se define la interfaz **`Set<E>`**. Podemos utilizar las siguientes implementaciones:

- **HashSet**<E>: implementación con tabla *hash*.
- **LinkedHashSet**<E>: tabla hash + doble lista enlazada.
- **TreeSet**<E>: implementación con árbol.

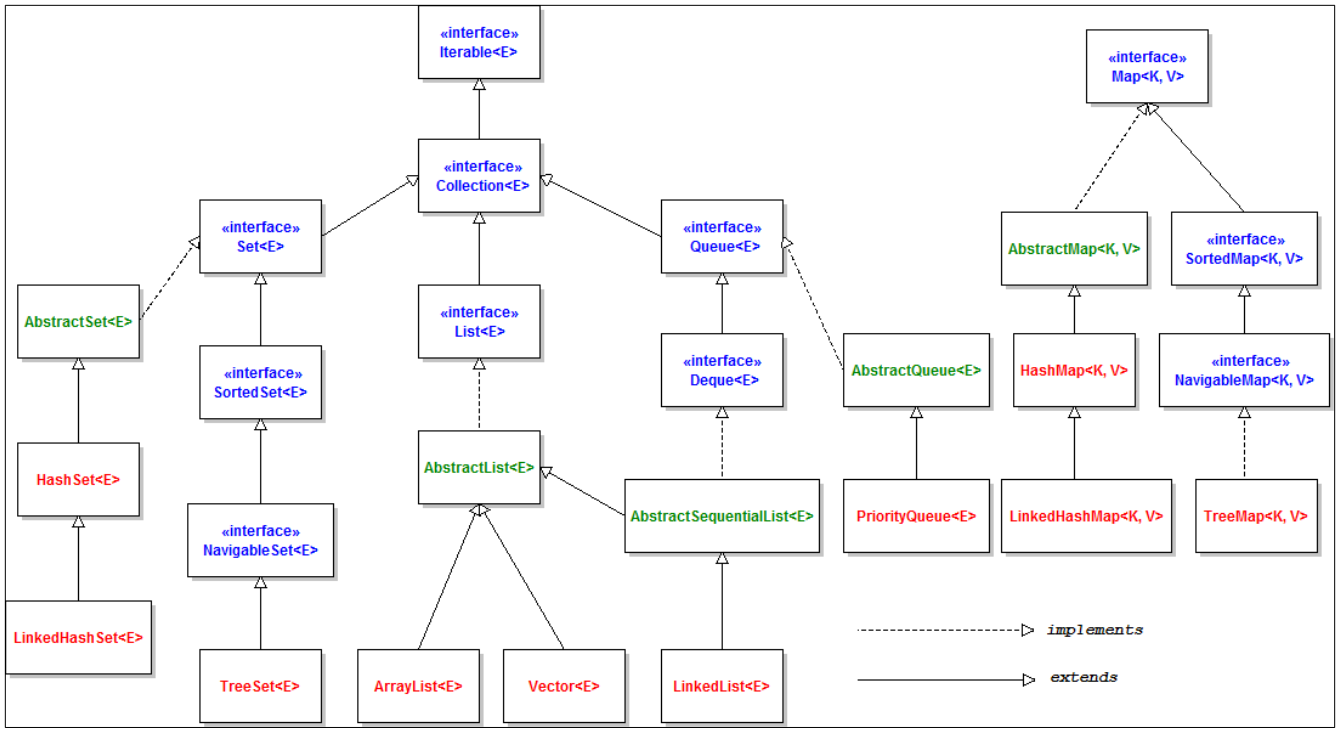
4. Diccionarios

Los diccionarios o matrices asociativas tienen, por cada elemento, una clave que usaremos para recuperarlo. Se utiliza la interfaz **`Map<K,V>`**, donde K es la clave y V el valor del elemento. Podemos utilizar las siguientes implementaciones:

- **HashMap**<K,V>
- **TreeMap**<K,V>
- **LinkedHashMap**<K,V>

5. Visión general del marco de colecciones

A continuación se muestra el *collections framework overview*, para ubicar cada colección en el marco general.



6. ¿Cuándo usar cada colección?

A continuación se muestra un esquema que nos ayuda a decidir qué colección usar en función de lo que deseemos o las restricciones que tengamos.