COLECCIONES:

Las colecciones se pueden separar en dos grandes tipos las Iterables y las Map.

Una colección es un contenedor variable para almacenar un conjunto de objetos o elementos.

En Java, se emplea la interfaz genérica “Collection” para este propósito. Gracias a esta interfaz, podemos almacenar cualquier tipo de objeto y podemos usar una serie de métodos comunes, como pueden ser: añadir, eliminar, obtener el tamaño de la colección etc.

Tipos de colecciones:

Interfaz Iterables<E>

- Nos permite utilizar el bucle for each y método forEach.
- Nos permite recorrer y eliminar los elemento de la colección.

Interfaz Collection<E>

- Esta interfaz extiende de Iterable<E>.
- Representa un grupo de elementos.
- Se utiliza para manipular colecciones (eliminar, añadir, tamaño de la colección etc.)

Operaciones de Collection:

- Tamaño de la colección: size y isEmpty.
- Comprobación: contains.
- Iterar: Iterator.
- Operaciones: containsAll, addAll, removeall, removeIf, retainAll etc.
- Eliminar y añadir: add, remove.
- Para pasar a array: toArray.
- Strams: stream, parallelStream.

Interfaz Set <E>

- Hereda de Collection<E>.
- No permite duplicados.
- No hay acceso posicional.
- Mejora la implementación de equals y hashCode.

Implementación de Set

- HashSet<E>: Es la que tiene major rendiminto pero no tiene un orden en los elemento, nos permite introducir valores null .
- LinkedHashSet<E>: Nos permite valores null y al contrario de la anterior si tiene encueta el orden de los elementos de la colección.
- TreeSet<E>: Mantiene el orden vasado en sus valores a diferencia del anterior y no permite valores null.

Interfaz List <E>

- Hereda de Collection <E>.
- Permite valores duplicados.
- Acceso posicional (se puede acceder a los elementos por su posición).
- Búsqueda.
- Iteración Extendida.
- Operaciones sobre un rango de elementos de la lista.

Implementación de List <E>

- ArrayList <E>: Tiene accesos por índices y suele ser la más utilizada.
- LinkedList <E>: Se accede por índices pero tiene un peor rendimiento que la anterior.
- Queue <E>: Funciona como una cola FIFO (El primero en entrar es el primero en salir).
- Deque <E>: Se podría decir que es una mejora del anterior, funciona como una cola FIFO o como una cola LIFO (El último en entrar es el primero en salir).

Interfaz Map <K,V>

- No hereda de Collection <E>.
- Almacena y manejar pares de claves (una clave y su valor).
- Se cumple lo siguiente: 1clave = 1 valor.
- No puede almacenar tipos primitivos.

Operaciones de Map <K,V>

- Meter nuevo valor junto con su clave: put(ky, value).
- Obtener la clave del valor: getKey(value).
- Obtener el valor de la clave: getValue(key).
- Comprobar si existe esa clave o valor: containsKey(key) .containsValue(value).
- Borrar clave: remove(key).

Implementación de Map <E>

- HashMap <K, V>: Es la que más se utiliza y no es sincronizada.
- LinkedHashMap <K,V> Su rendimiento es peor que HashMap y ordena los valores según el orden en el que se añaden.
- TreeMap <K,V> Tiene un peor rendimiento que las anteriores.

Colecciones no modificables:

Como su nombre indica, son colecciones que una vez creadas no se podrán modificar. En el caso de que se intente se mostrará un error

Colecciones Sincronizadas:

Se utilizan cunado tememos diferentes hilos de ejecución

Librerías de colecciones de terceros:

- Eclipse Collections.
- Apache commons Collections.
- Guava de Google.