TEMA 3 COLECCIONES

4.- Conjuntos de pares clave/valor. Map

¿Cómo almacenarías los datos de un diccionario? Tenemos por un lado cada palabra y por otro su significado. Para resolver este problema existen precisamente los arrays asociativos. Un tipo de array asociativo son los mapas o diccionarios, que permiten almacenar pares de valores conocidos como clave y valor. La clave se utiliza para acceder al valor, como una entrada de un diccionario permite acceder a su definición.

En Java existe la interfaz **`java.util.Map`** que define los métodos que deben tener los mapas, y existen tres implementaciones principales de dicha interfaz:

- **`java.util.HashMap`**,
- **`java.util.TreeMap`** y
- **`java.util.LinkedHashMap`**.

Los **mapas utilizan clases genéricas** para dar extensibilidad y flexibilidad, y permiten definir un tipo base para la clave, y otro tipo diferente para el valor. Veamos un ejemplo de cómo crear un mapa, que es extensible a los otros dos tipos de mapas:

```
HashMap<String,Integer> t = new HashMap<String,Integer>();
HashMap<String,Integer> t = new HashMap<>();  //OTRA FORMA DE DECLARACIÓN
```

El mapa anterior permite usar cadenas como llaves y almacenar de forma asociada a cada llave, un número entero. Veamos los métodos principales de la interfaz **`Map`**, disponibles en todas las implementaciones. En los ejemplos, **`V`** es el tipo base usado para el valor y **`K`** el tipo base usado para la llave:

Métodos principales de los mapas.

Método.	Descripción.
`V put(K key, V value);`	Inserta un par de objetos llave (`key`) y valor (`value`) en el mapa. Si la llave ya existe en el mapa devuelve el valor asociado que tenía antes, si la llave no existía, entonces devolverá `null`.
`V get(Object key);`	Obtiene el valor asociado a una llave ya almacenada en el mapa. Si no existe la llave, retornará `null`.
`V remove(Object key);`	Elimina la llave y el valor asociado. Retorna el valor asociado a la llave, por si lo queremos utilizar para algo, o `null`, si la llave no existe.
`boolean containsKey(Object key);`	Retornará `true` si el mapa tiene almacenada la llave pasada por parámetro, `false` en cualquier otro caso.
`boolean containsValue(Object value);`	Retornará `true` si el mapa tiene almacenado el valor pasado por parámetro, `false` en cualquier otro caso.

Métodos principales de los mapas.	
Método.	**Descripción.**
`int size();`	Retornará el número de pares llave y valor almacenado en el mapa.
`boolean isEmpty();`	Retornará `true` si el mapa está vacío, `false` en cualquier otro caso.
`void clear();`	Vacía el mapa.

FORMAS DE RECORRER UN MAP

Dado el mapa: `Map<String,Integer> agenda= new HashMap<>();`

Usando un **bucle for-each**

```
Integer tlf;
for(String  nombre: agenda.keySet() ){
    tlf= agenda.get(nombre);
    System.out.println(nombre + "-" + tlf);
}
```

Para recorrer un HashMap se usa un bucle con iterador

```
String nombre;
Integer tlf;
Iterator<String> it = agenda.keySet().iterator()
while( it.hasNext()) {
   nombre= it.next(); //devuelve una clave
   tlf = agenda.get(nombre);
   System.out.println(nombre + "-" + tlf);
}
```