76 - Colecciones: HashMap, TreeMap y LinkedHashMap

Listado completo de tutoriales

Esta clases: HashMap, TreeMap y LinkedHashMap nos permite almacenar elementos asociando a cada clave un valor.

Para cada clave tenemos un valor asociado. Podemos después buscar fácilmente un valor para una determinada clave. Las claves en el diccionario no pueden repetirse.

Algunos ejemplos donde podríamos usar un Mapa:

- Guardar en la clave las extensiones de archivos y en el valor los nombres de archivos que lo pueden abrir
- En una agenda podemos guardar como 'clave' la fecha y hora y las actividades en el 'valor'.

Problema

Almacenar un diccionario las palabras en castellano como 'clave' y las traducciones de las mismas en el 'valor'. Probar los métodos más significativos de la clase HashMap.

Programa:

Ver video

```
import java.util.HashMap;
import java.util.Map;

public class PruebaHashMap {

    public static void main(String[] args) {
        Map<String, String> mapa1 = new HashMap<String, String>();
        mapa1.put("rojo", "red");
        mapa1.put("verde", "green");
        mapa1.put("azul", "blue");
        mapa1.put("blanco", "white");
        System.out.println("Listado completo de valores");
        for (String valor : mapa1.values())
            System.out.print(valor + "-");
        System.out.println();
        System.out.println("Listado completo de claves");
        for (String clave : mapa1.keySet())
```

```
            System.out.print(clave + "-");
        System.out.println();
        System.out.println("La traducción de 'rojo' es:" + mapa1.get
        if (mapa1.containsKey("negro"))
            System.out.println("No existe la clave 'negro'");
        System.out.println("La traducción de 'marron' es:" + mapa1.g
        mapa1.remove("rojo");
        System.out.println(mapa1);
    }
}
```

La interfaz Map deben implementar estas clases, luego si creamos un objeto de la clase HashMap debemos hacerlo con la siguiente sintaxis:

```
Map<String, String> mapa1 = new HashMap<String, String>();
```

La clase HashMap utiliza datos genéricos tanto para la clave como para el valor, en este ejemplo la clave y el valor son datos de tipo String.

Mediante el método put agregamos un elemento a la colección de tipo HashMap:

```
mapa1.put("rojo", "red");
mapa1.put("verde", "green");
mapa1.put("azul", "blue");
mapa1.put("blanco", "white");
```

Para imprimir todos los valores del mapa lo recorremos mediante un for:

```
for (String valor : mapa1.values())
    System.out.print(valor + "-");
```

De forma similar si queremos recorrer todas las claves del mapa:

```
for (String clave : mapa1.keySet())
    System.out.print(clave + "-");
```

Para recuperar un valor para una determinada clave llamamos al método 'get' y le pasamos la clave a buscar, si dicha clave no existe en el mapa se nos retorna el valor 'null':

```
System.out.println("La traducción de 'rojo' es:" + mapa1.get("
```

Si queremos verificar si una determinada clave existe en el mapa lo hacemos mediante el método 'containsKey':

```
if (mapa1.containsKey("negro"))
    System.out.println("No existe la clave 'negro'");
```

Una variante del método 'get' es 'getOrDefault' que nos retorna el segundo parámetro si no encuentra la clave en el mapa:

```
System.out.println("La traducción de 'marron' es:" +
                    mapa1.getOrDefault("marrón", "No existe la
```

Para eliminar un elemento de la colección debemos hacer uso del método 'remove', pasamos una clave del mapa:

```
mapa1.remove("rojo");
```

Para imprimir el mapa completo en la Consola podemos hacer uso del método 'println':

```
System.out.println(mapa1);
```

Hemos utilizado la clase HashMap para resolver el problema. La clase TreeMap es idéntica a HashMap con la salvedad que mantiene ordenado los datos por la clave.

Finalmente la clase LinkedHashMap mantiene ordenado los elementos del mapa según el orden de inserción.

Retornar