22. Colecciones (III)

22.1 Set. HashSet.

La interface Set es la encargada del tratamiento de conjuntos en el API de Java. Un conjunto en Java es una colección de elementos que, como el conjunto en matemáticas, no permite elementos duplicados dentro de ella y no tiene orden entre sus elementos. Más formalmente, no permite elementos e1, e2 tales que: e1.equals(e2) sea true. Además nos obliga a implementar el método hashCode().

Un HashSet es una colección de objetos de la misma clase, que no pueden repetirse y sin ningún orden en particular.

```
Set<Integer> conjunto1 = new HashSet<Integer>(List.of(1,2,3,4,5,6,7,8));
Set<Integer> conjunto2 = new HashSet<Integer>(List.of(1,3,6,10,12));

System.out.println(conjunto1);
//[1, 2, 3, 4, 5, 6, 7, 8]

// Add no permite añadir duplicados, cuando se intenta
// insertar un elemento duplicado devuelve false y no añade nada.
conjunto1.add(2);

//Unión de conjuntos
conjunto1.addAll(conjunto2);
System.out.println(conjunto1);
//[1, 2, 3, 4, 5, 6, 7, 8, 10, 12]

//Intersección de conjuntos
conjunto1.retainAll(List.of(1,3,6,10,12,15,18));
System.out.println(conjunto1);
//[1, 3, 6, 10, 12]

//Resta
conjunto2.removeAll(List.of(1, 3, 6, 19));
System.out.println(conjunto2);
// [10, 12]
```

22.2 Map. HashMap.

La interface Map, representa un objeto que sirve para ligar (“hacer un mapeo”) un valor clave (key en inglés) y un valor u objeto (value en inglés). Así podemos pensar que un mapa es una especie de diccionario donde a un objeto clave le asignamos un objeto valor. Como ejemplo podríamos pensar en que una clave puede ser un número de pasaporte de una persona y como valor el objeto Persona que contiene toda la información y métodos de una persona en concreto.

Un mapa no puede tener claves duplicadas. La clave funciona como un identificador único. Para entender esto, considera por ejemplo que esta restricción sería igual que decir que dos personas no pueden tener igual número de pasaporte. También hay que tener en cuenta que un map proporciona tres vistas. Podemos ver un mapa como un conjunto de claves (por ejemplo todos los números de pasaportes en el sistema), colecciones de valores (por ejemplo todas las personas), o un conjunto de pares claves-valor mapeados (todos los números de pasaporte vinculados cada uno a su persona correspondiente).

Hay que tener especial cuidado con los métodos equals y hashCode, ya que los hay que sobrescribir para que al utilizar clases que programamos tengan un funcionamiento acorde a nuestras necesidades.

```java
Map<Integer,String> mapa=new HashMap<>();

// ¡Ojo!, es put, no add.
mapa.put(1, "uno");
mapa.put(2, "dos");
mapa.put(3, "tres");
// Si repetimos una clave se pondrá el último valor.
mapa.put(4, "cuadro");
mapa.put(4, "cuatro");

System.out.println(mapa);
//{1=uno, 2=dos, 3=tres, 4=cuatro}

//El uso más común es get, recogiendo el valor asociado a la clave.

String valor = mapa.get(2);
System.out.println(valor);
//dos

// Podemos obtener un conjunto de claves y una colección de valores.

Set<Integer> claves = mapa.keySet();
System.out.println(claves);
// [1, 2, 3, 4]

Collection<String> valores = mapa.values();
System.out.println(valores);
//[uno, dos, tres, cuatro]


// Podemos saber si una clave o valor se encuentran en el Map

boolean estaClave = mapa.containsKey(2);
boolean estaValor = mapa.containsValue("tres");

System.out.println("Clave encontrada "+estaClave);
System.out.println("Valor encontrado "+estaValor);
//Clave encontrada true
//Valor encontrado true
```

Ejercicio 22.1

Escribe un programa que utilizando un Set HashSet permita, a través de un menú, gestionar una lista de palabras de número ilimitado donde no se permitirán palabras repetidas y siendo el orden de las mismas totalmente irrelevante:

Opciones

```
1.-Introducir palabra

2.-Listar palabras

3.-Eliminar palabra

4.-Borrar todas

5.-Mostrar tamaño

6.-Buscar

7.-Salir
```

Ejercicio 22.2

Crea una clase Empleado con los siguientes atributos y constructor:

```
private String nombre;

private double sueldo;

public Empleado(String nombre, double sueldo)
```

• Además tendrá los siguientes métodos:

```
public String getNombre()

public double getSueldo()
```

Escribe un programa que utilizando una colección HashSet permita, a través de un menú, gestionar una lista de empleados de número ilimitado donde no se permitirán nombres de empleados repetidos y siendo el orden de los mismos totalmente irrelevante:

Opciones

```
1.-Introducir empleado

2.-Listar empleados

3.-Eliminar empleado

4.-Borrar todos

5.-Mostrar número de empleados

6.-Buscar empleado

7.-Salir
```

• Para introducir un empleado hará falta también introducir su sueldo.

• Al listar los empleados se mostrará su nombre y sueldo.

• Para eliminar un empleado será necesario introducir su nombre.

• El programa no será sensible a mayúsculas.

Ejercicio 22.3

Realiza un programa que sepa decir la capital de un país (en caso de conocerla respuesta) y que, además, sea capaz de aprender nuevas capitales. En principio, el programa solo conoce las capitales de España, Portugal y Francia. Estos datos deberán estar almacenados en un diccionario. Los datos sobre capitales que vaya aprendiendo el programa se deben almacenar en el mismo diccionario.

Ejercicio 22.4

Un supermercado de productos ecológicos nos ha pedido hacer un programa para vender su mercancía. Se tendrán en cuenta los productos que se indican en la tabla junto con su precio. Los productos se venden en bote, brick, etc. Cuando se realiza la compra, hay que indicar el producto y el número de unidades que se compran, por ejemplo “guisantes” si se quiere comprar un bote de guisantes y la cantidad, por ejemplo “3” si se quieren comprar 3 botes. La compra se termina con la palabra“fin. Suponemos que el usuario no va a intentar comprar un producto que no existe.

Utiliza un diccionario para almacenar los nombres y precios de los productos y una o varias listas para almacenar la compra que realiza el usuario.

A continuación se muestra una tabla con los productos disponibles y sus respectivos precios:

```
avena garbanzos tomate jengibre quinoa guisantes

2,21   2,39     1,59    3,13    4,50    1,60
```

```
Producto: tomate

Cantidad: 1

Producto: quinoa

Cantidad: 2

Producto: avena

Cantidad: 1

Producto: quinoa
```

```
Cantidad: 2

Producto: tomate

Cantidad: 2

Producto: fin



Producto       Precio  Cantidad       Subtotal

-----------------------------------------

tomate         1,59    3              4,77

quinoa         4,50    4              18,00

avena          2,21    1              2,21

-----------------------------------------

TOTAL: 24,98
```

Ejercicio 22.5

Crea una función que a partir de un String devuelva cual es el carácter que más se repite.

Ejercicio 21.5

Mejora el MiniRPG de forma que:

- Además del guerrero existirán tres personajes más, Arquera, Maga y Nigromante, cada uno de ellos tendrá un método atacar distinto.

- Todos los personajes pueden portar un **Arma**, el Arma tendrá métodos int minDanho() e int maxDanho(), que proporcionarán un daño extra al atacar. Existen tres tipos de Arma: Espada, Arco, y Varita. Sólo las Magas y los Nigromantes podrán portar varitas, además del resto de Armas.

Las Armas tendrán atributos destrezaMinima() y fuerzaMinima(), los Personajes no podrán **portar** un Arma si no cumplen en sus atributos.

Las Magas pueden lanza **Hechizos**, los Hechizos los tienen que **aprender**, las magas usan su método **hechizar(Personaje, Hechizo h)** para hechizar a sus oponentes, quitándole un daño extra, siempre y cuando el Hechizo haya sido aprendido. Los hechizos tendrán un método **hechizar(Personaje)**, que infringirá el daño mágico.

Los Nigromantes tienen un método **sanar(Personaje p)**, que recupera la vida otro Personaje al máximo.

Una vez acometidos los requerimientos básicos, se puede mejorar libremente la jerarquía de **Armas** y **Hechizos**, además de clasificar los Hechizos e implementar resistencias a ellos.