Programación Orientada a Objetos
Curso 2023/2024

Sesión 8

Mapas: HashMap

Para la implementación de los ejercicios se debe hacer uso de la clase `java.util.HashMap`, que implementa una estructura de datos que asocia pares <clave, valor>. El mapa se gestiona como un conjunto de claves, por tanto, no puede contener claves repetidas.

La documentación de la clase `java.util.HashMap` está disponible en https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html o en línea en Eclipse.

A continuación, se muestra un fragmento de código con las operaciones básicas para el manejo de los mapas.

```
// > Construye un mapa que asocia cadenas (clave) con números (valor)
// Los pares que se registran son <alumno, nota>
HashMap<String, Double> calificaciones = new HashMap<String, Double>();

// > Insertar un par <alumno, nota> en el mapa
calificaciones.put("Candela", 9.5);
calificaciones.put("Antonio", 6.0);

// > Consultar el número de entradas registradas en el mapa
System.out.println("Presentados = " + calificaciones.size());

// > Consultar si existe una clave
boolean resultado = calificaciones.containsKey("Candela");

// > Obtener el valor asociado a una clave.
// get devuelve null si la clave no existe
if (resultado) {
   double nota = calificaciones.get("Candela");
   System.out.println("Candela ha sacado un " + nota);
}

// > No se puede recorrer el mapa pero si las claves o los valores
// > Recorremos las claves para imprimir el nombre de los alumnos
for (String alumno : calificaciones.keySet())
      System.out.println(alumno);

// > Recorremos los valores para imprimir la mejor calificación
double mejorNota = 0;
for (Double nota : calificaciones.values()){
      if (nota > mejorNota)
            mejorNota = nota;
}
System.out.println("Mejor calificación = " + mejorNota);
```

Conjuntos: HashSet

La documentación online está disponible en
http://docs.oracle.com/javase/8/docs/api/java/util/HashSet.html

```
// > Creación de un conjunto vacío

HashSet<String> conjunto = new HashSet<String>();

// > Añadir elementos

conjunto.add("uno");
conjunto.add("tres");
conjunto.add("dos");
conjunto.add("cinco");

// El método add retorna un booleano indicando si el elemento
// es aceptado en el conjunto

// Un conjunto EVITA REPETIDOS

boolean insertado = conjunto.add("dos");
System.out.println("¿Insertado? " + insertado);

// > Consultar el tamaño

System.out.println("Tamaño: " + conjunto.size());
System.out.println("¿Es vacío? " + conjunto.isEmpty());

// > Consultar si contiene un elemento

System.out.println("¿Contiene 'cuatro'? " + conjunto.contains("cuatro"));

// > Borrado

// El método remove intenta borrar el elemento

boolean borrado = conjunto.remove("uno");
System.out.println("¿Borrado? " + borrado);

System.out.println("¿Borrado? " + conjunto.remove("seis"));

// > Recorridos
// No podemos recorrer una colección por índice
// Debemos recorrer con "for each" (iterador)

for (String elemento : conjunto) {
      System.out.println(elemento);
}

// IMPORTANTE: observa que el orden de los elementos no corresponde con la
inserción

// > Copia

HashSet<String> copia = new HashSet<String>(conjunto);

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

Ejercicio

Queremos desarrollar una aplicación para gestionar apuestas en eventos deportivos.

Un **evento** contiene la descripción de un encuentro deportivo y la información sobre las apuestas que se realizan para ese encuentro. Las propiedades de los eventos son:

- Nombre del evento deportivo. Por ejemplo, "Real Madrid - F.C. Barcelona".
- El precio de la apuesta (0,50 euros, 2,50 euros, etc.). Este precio es fijo para todas las apuestas que se hagan para el evento.
- **Mapa de las apuestas** realizadas. El mapa asocia el nombre del usuario que realiza la apuesta con el marcador por el que ha apostado. Dado que en un mapa no puede haber claves repetidas, sólo se permite una apuesta por usuario.
- Número de apuestas que se han realizado.
- Recaudación: cantidad de dinero recaudada por el evento. Equivale al número de apuestas por el precio de la apuesta.

Un **marcador** representa el tanteo de un encuentro deportivo. Se caracteriza por dos propiedades de tipo entero: tanteo local y tanteo visitante. Estas propiedades son fijas y no varían una vez establecidas. Implementa los métodos `equals/hashCode` en esta clase. Dos marcadores son iguales si tienen los mismos tanteos.

La funcionalidad principal de un evento es *apostar*. Esta operación acepta como parámetros un nombre de usuario (cadena) y un marcador. Retorna un valor booleano indicando si la apuesta ha sido aceptada. Es importante destacar que no todos los marcadores son aceptables como apuesta, por lo que la aceptación de una apuesta depende del tipo de evento. Por tanto, el algoritmo que implementa esta operación se define como un **método plantilla**:

- Si el marcador *es aceptable* como apuesta, se registra el par <usuario-marcador> en el mapa de las apuestas realizadas. La operación finaliza devolviendo un valor verdadero indicando que la apuesta ha sido aceptada.
- En caso contrario, retorna un valor falso indicando que no se acepta la apuesta.

Tipos de eventos.

Un **evento libre** es un evento que permite apostar por cualquier marcador, siempre que no se haya realizado una apuesta con ese marcador previamente. Por tanto, un marcador *es aceptable* si no es igual al marcador de alguna de las apuestas que se hayan realizado en el evento.

Un **evento restringido** es un evento que sólo permite apostar por un **conjunto** fijo de marcadores que denominamos *opciones*. Estas opciones son una propiedad consultable del evento, que se establece en el constructor y no varía. Por tanto, un marcador *es aceptable* si está contenido en el conjunto de las *opciones*.

Una funcionalidad adicional que aportan los eventos restringidos es la consulta del número de apuestas que se han realizado para un marcador.

El ejercicio consiste en:

- Programar los tipos de datos necesarios para implementar esta aplicación.
- Implementar un **programa** que incluya la siguiente funcionalidad:
    - Declara y construye el siguiente evento libre: "Real Madrid - F.C. Barcelona", precio apuesta 1 euro.
    - El usuario "juan" apuesta por el marcador (5, 0).
    - El usuario "pepe" apuesta por (1, 3).
    - Declara y construye el evento restringido: "Alcaraz vs Djokovic", 3 euros por apuesta y los marcadores posibles serían (2, 0), (2, 1), (0, 2) y (1, 2).
    - El usuario "juan" apuesta por (2, 0)
    - El usuario “pedro” apuesta por (2, 1).
    - El usuario "pepe" apuesta por (2, 0).
    - Declara y construye una lista para almacenar eventos. Añade los eventos a esa lista.
    - Recorre todos los eventos: el usuario “enrique” apuesta por (5, 0) en cada evento. Observa que esta apuesta no se admite en ninguno de los dos eventos. En el libre porque ya hay una apuesta para ese marcador, y en el restringido porque no está incluido en el conjunto de marcadores válidos (opciones).
    - Recorre todos los eventos: muestra el nombre, el número de apuestas y la recaudación.
    - Recorre todos los eventos: si el evento es restringido, muestra el nombre y el número de apuestas para el marcador (2, 0).