EJERCICIO 25: Bag

Primera parte:

Un **Map** en Java es una colección que asocia objetos que actúan cómo **claves** (keys), a otros objetos que actúan cómo **valores** (values).
En otros lenguajes también se llaman **Diccionario**.
Cada clave está vinculada a un único valor; ***no pueden existir claves duplicadas en un mapa***, aunque ***sí podrían haber valores repetidos***.
A los pares clave-valor se los denomina entradas (entries).
Para definir un Map, es necesario indicar el tipo que tendrán sus claves y valores:
por ejemplo, una variable de tipo Map <String , Integer> define claves String, y valores Integer.
Observe que Map <K,V> es una interfaz.

Tareas:

A) Lea la documentación de Map en Java y responda:

1) ¿Qué implementaciones provee Java para utilizar un Map? ¿Cuáles de ellas son destinadas a uso general?
2) Investigue cómo consultar sí un mapa contiene una determinada clave (key). Explique qué métodos deben implementar las claves para que esto funcione correctamente.
3) ¿Con qué método se puede recuperar el objeto asociado a una clave? ¿Qué pasa sí la clave no existe en el mapa?
4) Investigue cómo agregar claves y valores a un mapa. ¿Qué pasa sí la clave ya se encontraba en el mapa? ¿Permite agregar claves y/o valores nulos?
5) Determine cómo se pueden eliminar claves y valores de un mapa. ¿Es necesario controlar la presencia de alguno de ellos?
6) Investigue cómo reemplazar un valor en un mapa.
7) Teniendo en cuenta los métodos keySet(), values() y entrySet(), explique de de qué formas se puede iterar un mapa ¿Es posible utilizar streams?

B. Para practicar los mensajes investigados anteriormente, escriba un test de unidad que contenga lo siguiente:

1) Cree un map un Map, y agregue las tuplas (“Lionel Messi”, 111), (“Gabriel Batistuta”, 56), (“Kun Agüero”, 42) .
2) Elimine la entrada con clave “Kun Agüero”.
3) Messi hizo 112 goles a día de la fecha; actualice la cantidad de goles.
4) Intente repetir la clave “Gabriel Batistuta” y verifique que no es posible.
5) Obtenga la cantidad total de goles.

C) Como se mencionó, cualquier objeto puede actuar como clave.
Es decir, pueden ser instancias de clases definidas por el programador.
Modele e implemente la clase Jugador con apellido y nombre.
Escriba otro test de unidad similar al de la tarea 2, pero utilizando Map.

Segunda parte:

Un **Bag** (bolsa) es una colección que permite almacenar elementos sin ningún orden específico y admite elementos repetidos.
Este objeto requiere un buen tiempo de respuesta para conocer la cardinalidad de sus elementos, y por esa razón almacena la cardinalidad de cada elemento (cantidad de veces que fue agregado en la bolsa).
Por ejemplo, sí agregamos 3 veces un objeto en la bolsa, y luego eliminamos 1 referencia, la cardinalidad de ese objeto en la bolsa es 2.

El protocolo de la interface Bag<T> es:

```
public interface Bag<T> extends Collection<T> {
    /**
     * Agrega un elemento al Bag, incrementando en 1 su cardinalidad.
     */
    @Override
    boolean add(T element);

    /**
     * Devuelve la cardinalidad del elemento. Sí el elemento no está en el Bag,
     * devuelve 0.
     */
    int occurrencesOf(T element);

    /**
     * Elimina una referencia del elemento del Bag. Sí el elemento no está en
     * el Bag, no hace nada.
     */
    void removeOccurrence(T element);

    /**
     * Elimina el elemento del Bag. Sí el elemento no está en el Bag, no hace
     * nada
     */
    void removeAll(T element);

    /**
     * Devuelve el número total de elementos en el Bag, es decir, la suma de
     * todas las cardinalidades de todos sus elementos.
     */
    @Override
    int size();
}
```

Tareas:

1) Liste los métodos que debe contener una clase que implementa la interface Bag<T>.

2) Explique cómo implementaría un **Bag<T>** usando composición con un **Map<K,V>.** ¿De qué tipo tendrían que ser las claves y valores del Map?

3) Implemente la interfaz Bag, utilizando **AbstractCollection<T>** como superclase, y componga con un **Map<K,V>.** Para simplificar la implementación utilice la clase BagImpl que se encuentra en este link.

4) *Discuta con un ayudante:*

a) ¿Cuáles son los beneficios de utilizar AbstractCollection como superclase para implementar el Bag?

b) ¿Qué ventajas tiene componer con un Map para implementar el Bag?

c) En lugar de componer con un Map, ¿es posible extenderlo para poder implementar el Bag? ¿Qué diferencias tendría esa solución con respecto a la planteada en este ejercicio?

d) ¿Para qué cree que podría ser útil un objeto Bag?