UF3.2.3
Set - Map

Pedro J. Camacho

U-tad
CENTRO UNIVERSITARIO
DE TECNOLOGÍA Y ARTE DIGITAL

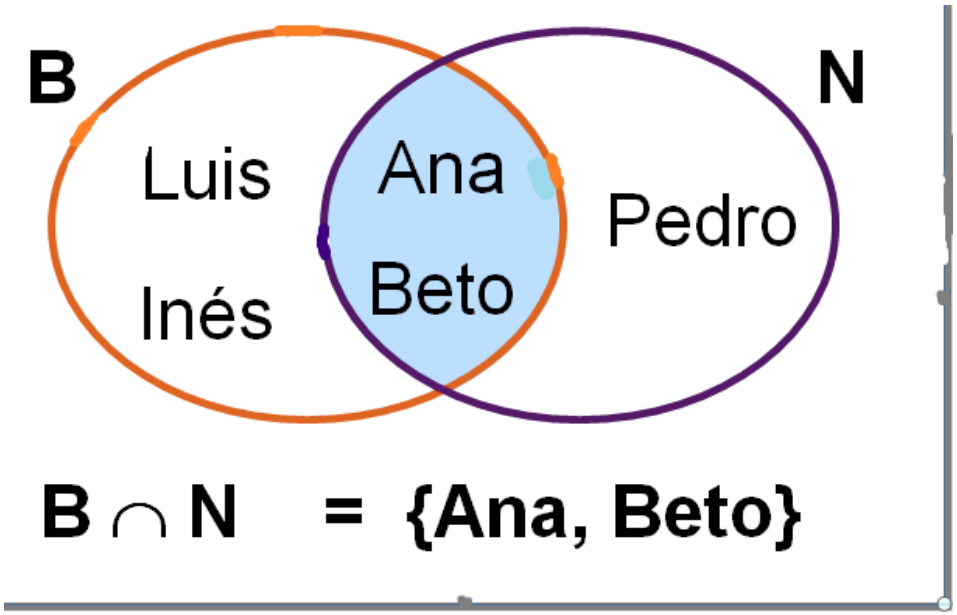
CONTENIDOS

1. Set - Introducción
   - HashSet
   - TreeSet
   - LinkedHashSet
2. Map - Introducción
   - HashMap

SET

Introducción

- Un **Set** es una Collection de tipo “conjunto” que tiene dos características muy importantes:
  - No hay repetidos.
  - No importa el orden (aunque veremos alguna excepción)



- La ventaja principal de utilizar Sets es que preguntar si un elemento ya está contenido mediante “**contains**()” suele ser muy eficiente.

HASHSET

Definición

- La implementación más común de un conjunto es **HashSet**.
- Observa el siguiente código y comenta qué opinas sobre el orden y los repetidos:

```
import java.util.HashSet;

public class Main {
    public static void main(String[] args) {
        HashSet<String> conjuntoB = new HashSet <String>();
        HashSet<String> conjuntoN = new HashSet <String>();
        conjuntoB.add("Luis");
        conjuntoB.add("Inés");
        conjuntoB.add("Ana");
        conjuntoB.add("Beto");
        conjuntoN.add("Ana");
        conjuntoN.add("Beto");
        conjuntoN.add("Pedro");
        conjuntoN.add("Pedro");

        System.out.print("Conjunto B: ");
        for (String elto : conjuntoB) {
            System.out.print(elto + " ");
        }
        System.out.println();
        System.out.print("Conjunto N: ");
        for (String elto : conjuntoN) {
            System.out.print(elto + " ");
        }
    }
}
```

HASHSET

Ejercicio 1/4

- Hay dos métodos interesantes cuando se trabajan con conjuntos:
  - **removeAll** (conjunto): Elimina todos los elementos que tiene el conjunto.
  - **retainAll** (conjunto): Hace lo contrario que la anterior, mantiene los elementos de un conjunto, eliminando el resto.

- Utilizando el siguiente código obtén B ∩ N = {Ana, Beto}

```
import java.util.HashSet;

public class Main {
     public static void main(String[] args) {
          HashSet<String> conjuntoB = new HashSet <String>();
          HashSet<String> conjuntoN = new HashSet <String>();
          conjuntoB.addAll(Arrays.asList("Luis Inés Ana Beto".split(" ")));
          conjuntoN.addAll(Arrays.asList("Ana Beto Pedro".split(" ")));
          //Por aquí tu instrucción

          //Conjunto resultante
          System.out.print("Conjunto Intersección:");

     }
}
```

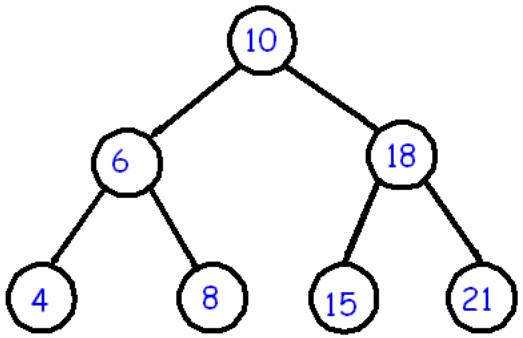
TREESET

Ejercicio 2/4

- Cuando necesitamos que los elementos del conjunto estén ordenados, debemos utilizar **TreeSet.**
- TreeSet construye un árbol binario ordenado con los objetos que se van agregando al conjunto. Esto tiene un coste, ya que es más lento que HashSet.

- Ejecuta el siguiente código para ver el orden que tiene HashSet y cámbialo a TreeSet para ver la diferencia:

```
import java.util.HashSet;

public class Main {
    public static void main(String[] args) {
        HashSet<Integer> numeros = new HashSet <Integer>();
        numeros.add(4);
        numeros.add(6);
        numeros.add(8);
        numeros.add(10);
        numeros.add(15);
        numeros.add(18);
        numeros.add(21);

        //Conjunto resultante
        for (Integer elto : numeros) {
            System.out.println(elto);
        }
    }
}
```



LINKEDHASHSET

Ejercicio 3/4

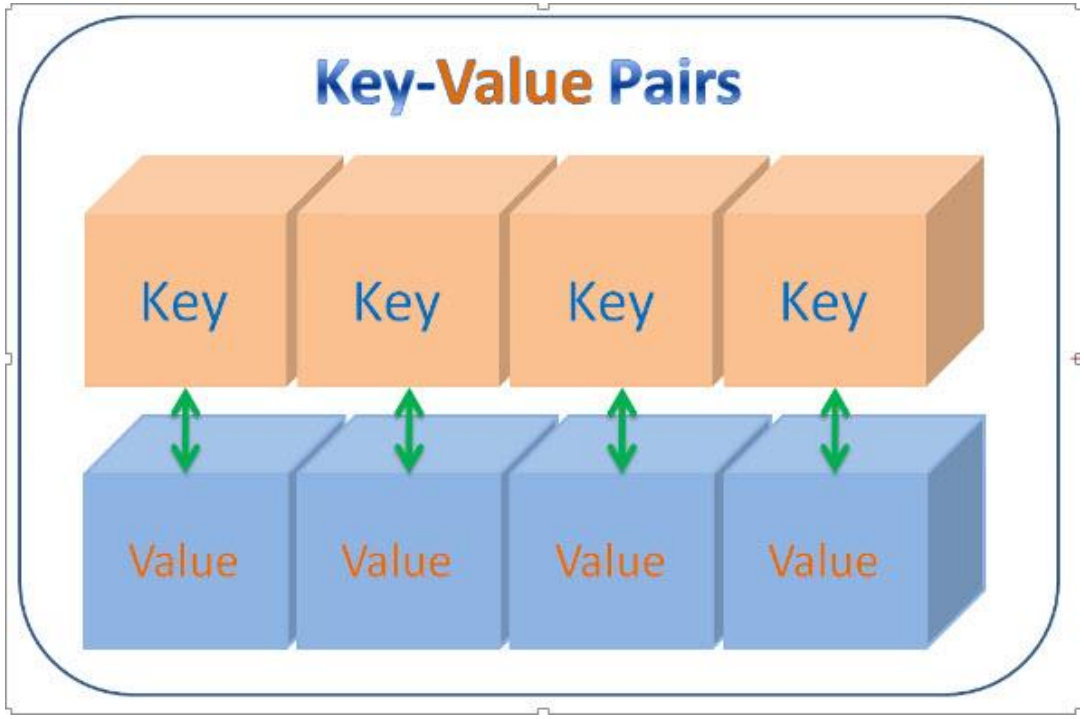
- Cuando necesitamos que los elementos del conjunto mantengan el orden de insertado, podemos utilizar **LinkedHashSet.**
- Observa esta propiedad en el siguiente código y luego cambia el tipo de conjunto a HashSet y TreeSet respectivamente, ¿qué observas?

```
import java.util.LinkedHashSet;

public class Main {
    public static void main(String[] args) {
        LinkedHashSet<String> numeros = new LinkedHashSet <String>();
        numeros.addAll(Arrays.asList("Uno","Dos","Tres","Cuatro"));

        //Conjunto resultante
        for (String elto : numeros) {
            System.out.println(elto);
        }
    }
}
```

MAP

Introducción

- Un **Map** representa lo que en otros lenguajes se conoce como **diccionario** y que se suele asociar a la idea de “tabla hash”.
- Un Map puede asemejarse a una tabla de base de datos con dos columnas, la primera sería la **clave** o “key” y la segunda el **valor** o “value”.
- Las claves forman un conjunto en el sentido Java: Son un “**Set**”, no puede haber duplicados.



HASHMAP

Definición

La clase que debemos conocer es **HashMap** y éstos son algunos de los métodos más importantes :

get(K clave)

Obtiene el valor correspondiente a una clave. Devuelve null si no existe esa clave en el map.

put(K clave, V valor)

Añade un par clave-valor al map. Si ya había un valor para esa clave se lo reemplaza.

keySet()

Devuelve todas las claves (devuelve un Set, es decir, sin duplicados).

values()

Devuelve todos los valores (los valores sí pueden estar duplicados, por lo tanto esta función devuelve una Collection).

entrySet()

Devuelve todos los pares clave-valor (devuelve un conjunto de objetos Map.Entry, cada uno delos cuales devuelve la clave y el valor con los métodos getKey() y getValue() respectivamente).

HASHMAP

Ejercicio 4/4

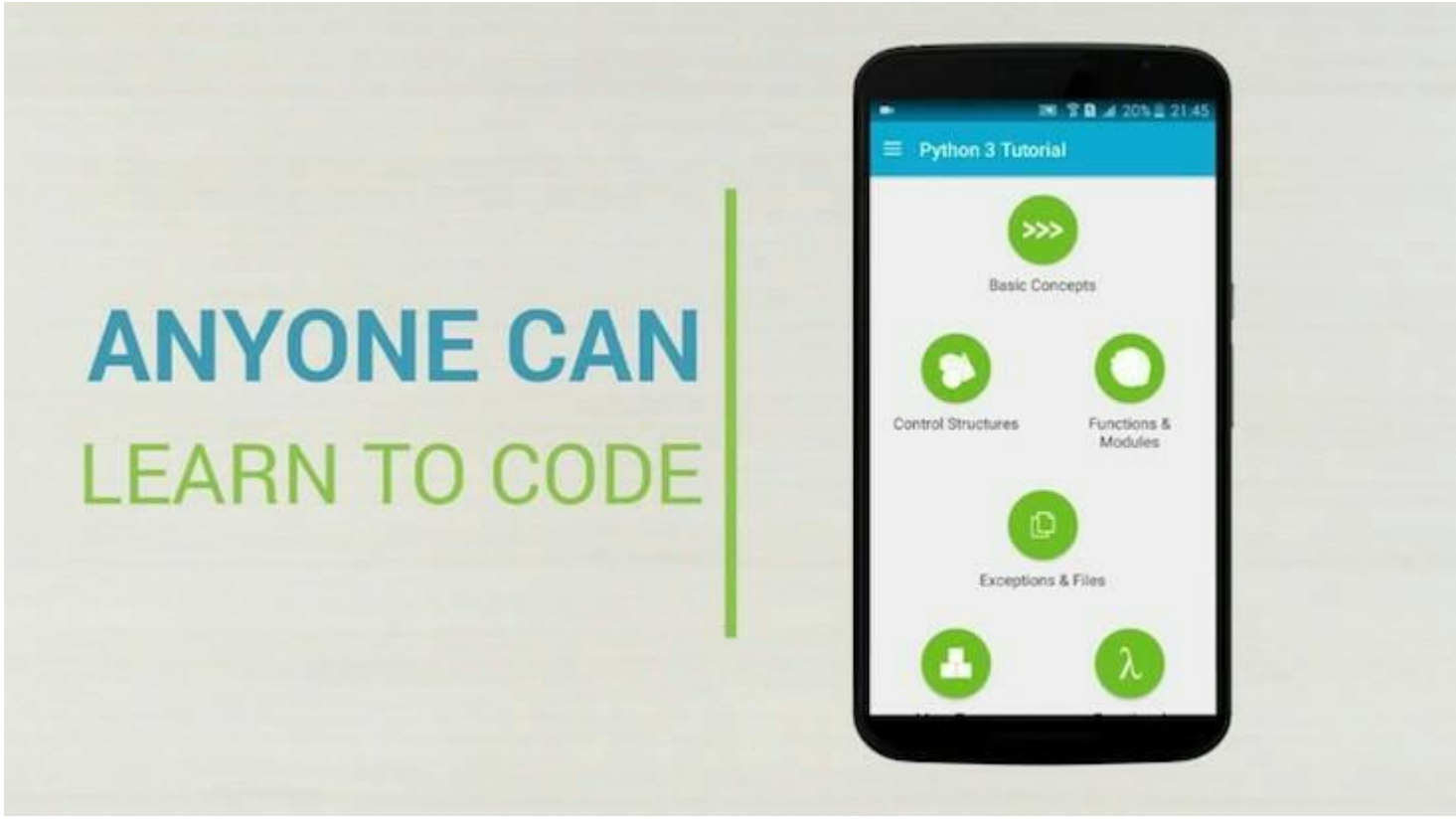
Utilizando los métodos anteriores, resuelve las siguientes cuestiones de este código:

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

public class Main {
    public static void main(String[] args) {
        HashMap<String, Integer> notas = new HashMap<String, Integer>();
        notas.put("Antonio", 7);
        notas.put("Pedro", 9);
        notas.put("Luis", 10);
        notas.put("Marco", 7);
        notas.put("María", 4);
        notas.put("Ana", 10);

        // Obtener la nota de Pedro con get

        // Preguntar si el alumno María está en lista

        // Sacar la nota media de la clase

        // Listado de todos los alumnos -> Alumno: Nota
    }
}
```

APRENDE A PROGRAMAR

…desde el móvil

Pulsa en la imagen:



CENTRO PROFESIONAL
DE TECNOLOGÍA Y ARTE DIGITAL