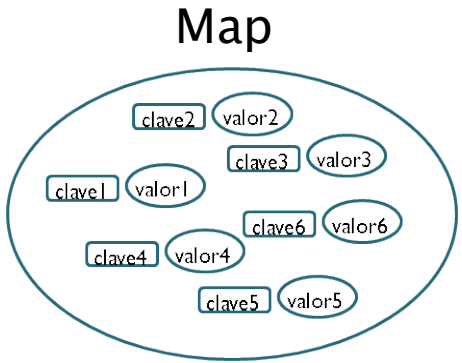
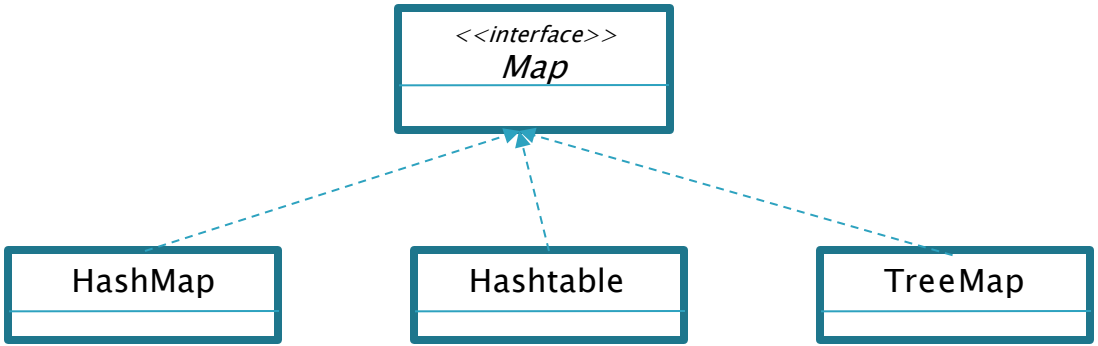
Colecciones de tipo tabla (Map)

Características

- Cada elemento tiene asociada una clave única
- No hay un orden o posición
- Las tablas implementan la interfaz Map
- Tanto el tipo de la clave como del valor son genéricos
- La principal clase de colección es HashMap



Clases e interfaces de tablas



Creación de tablas

Como instancias de HashMap:

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

Mediante método de factoría de Map:

```
Map<Integer, String> contactos=Map.ofEntries(
                        Map.entry(123, "Luis"),
                        Map.entry(300, "Ana"),
                        Map.entry(500, "Maria"));
```

INMUTABLES, no admiten la eliminación, modificación e inserción de elementos.
No admiten ni claves ni valores null

Mediante el método copyOf de Map:

```
Map<Integer, String> nuevos=Map.copyOf(contactos);
```

Métodos HashMap I

- T put(K clave, T dato). Añade el dato a la colección y le asocia la clave indicada como primer parámetro. Si ya existiera esa clave, el valor existente será sustituido por el nuevo:

```
HashMap<Integer,String> tabla=new HashMap<>();
tabla.put(200, “dato1”);
tabla.put(400, “dato2”);
tabla.put(200,”dato3”); //dato1 es sustituido por dato3
```

put admite claves null

- T putIfAbsent(K clave, T dato). Añade la entrada en caso de que la clave no exista o tenga asociado el valor null

- T get(K clave). Devuelve el dato que tenga asociada dicha clave. Si no hay ninguno, devolverá null.

Revisión conceptos

Indica que sucederá al ejecutar el siguiente código

```
HashMap<Integer,String> tabla=new HashMap<>();
tabla.put(25,"hello");
tabla.put(null, "by");
System.out.println(tabla.get(null));
```

a. Se imprime by
b. Se imprime hello
c. Se imprime cadena vacía
d. Se produce una excepción

Respuesta

La respuesta correcta es la a. Como admite clave null, el valor asociado con dicha clave es "by"

Métodos HashMap II

- int size(). Devuelve el tamaño de la colección

- T remove(Object clave). Elimina el dato que tenga dicha clave asociada y devuelve el elemento eliminado.

```
System.out.println(tabla.remove(400)); //muestra dato2
```

- boolean containsKey(K clave). Indica si hay algún elemento en la colección con dicha clave asociada.

- boolean containsValue(T valor). Indica si el elemento está presente en la colección

Recorrido de una tabla

- Se pueden utilizar los siguientes métodos:
  - Collection<T> values(). Devuelve una colección solo con los valores. Puede utilizarse para recorrer el conjunto de valores con un for each:

```
Collection<String> datos=tabla.values();
for(String s:datos){
        System.out.println(s); //muestra cada elemento
}
```

  - Set<K> keySet(). Devuelve el conjunto de claves de la tabla. Se puede recorrer con un for each.
  - Método forEach (se estudia en lambdas)

Revisión conceptos

Indica cual es la instrucción que falta en la línea 1

```
//linea 1
for(List<String> lt:datos.values()) {
    for(String s:lt){
        System.out.println(s);
    }
}
```

Respuesta

La instrucción que falta en línea 1 podría ser:
Map<String, List<String>> datos=new HashMap<>();
De hecho, la clave podría ser de cualquier tipo, lo importante es que sea un Map de valores lista de String

TreeMap

- Similar a HashMap, con la diferencia de que los objetos están ordenados por la clave:

```
TreeMap<Integer,String> tabla=new TreeMap<>();
tabla.put(200, "dato1 ");
tabla.put(400, "dato2 ");
tabla.put(100,"dato3 ");
for(String val:tabla.values()){
    System.out.print(val);   // dato3 dato1 dato2
}
```