Román Ramos María Fernanda

# Avance Práctica 3 - Algoritmos de Búsqueda Parte 1

## Parte 1: Tablas Hash En Java

Crea una nuevo proyecto o programa y agrega la siguiente clase principal y responde las preguntas:

1. Agrega 3 elementos más a la tabla hash. Además de put, ¿con qué otro método se pueden agregar elementos?

```
//agregar 3 elem con funciones diferentees a put
personas.putIfAbsent(key:3011, value:"Carlos Quinto"); //solo la inserta si no existe la clave, si existe, no
personas.putIfAbsent(key:3014, value:"Karla Santamaria"); //agregar primero: no existe la clave, entonces crea
System.out.println("HashMap después de putIfAbsent: \n" + personas);
personas.merge(key:2134, value:" Castillo", (valorExistente, nuevoValor) -> valorExistente + nuevoValor); //le
personas.merge(key:5555, value:"Valencia", (valorExistente, nuevoValor) -> valorExistente + nuevoValor); //agr
System.out.println("HashMap después de merge: \n" + personas);
```

   - ★ putIfAbsent(): agrrega solo si la clave no existe
   - ★ merge(): combina los valores existentes o agrega nuevos si no existe

*fuentes:* *Java HashMap putIfAbsent() Method*, *Java HashMap merge() Method*

2. ¿Cuál es la diferencia entre los métodos containsKey y get?
   - ★ *map.**containsKey(clave)***: verifica si existe una ***clave***, y devuelve un booleano, true si existe y false si no está
   - ★ *map.**get(clave)***: devuelve el **valor** asociado a la **clave**
3. ¿Qué otros métodos hay para buscar elementos?

```
//Otros métodos para buscar elementos:
System.out.println(personas.getOrDefault(key:3011, defaultValue:"no se encontró")); //clave que si hay
System.out.println(personas.getOrDefault(key:3015, defaultValue:"no se encontró")); //clave que no está
System.out.println("¿Se encuentra Juan Pérez?"+personas.containsValue(value:"Juan Pérez")); // SI ESTÁ devuelve true
System.out.println("¿Se encuentra Juan Correa?"+personas.containsValue(value:"Juan Correa")); // NO ESTÁ devuelve false
```

   - ★ map.**getOrDefault(clave, valorPorDefecto)** : si existe la **clave**, devuelve el valor asociado a ella, o si no existe, el **valor por defecto**, que puede ser un valor numérico, string
     - *fuente:* *Java HashMap getOrDefault() Method*
   - ★ map.**containsValue(valor )**: verifica si existe valor, devuelve un booleano
     - *fuente:* *Java HashMap containsValue() Method*
4. Investiga para qué sirve el método size, agrega algún ejemplo de uso de ese método al código

```
//USO DE SIZE
System.out.println("El HashMap tiene: " + personas.size() + " elementos\n");
```

   - ★ **size()** devuelve el número de pares clave-valor en el HashMap. Sirve para conocer la cantidad de elementos en el HashMap.

5. Agrega las instrucciones para iterar sobre la tabla hash y mostrar los elementos que tiene la tabla

```
//iteraciones
for (Map.Entry<Integer, String> entry : personas.entrySet()) {
    System.out.println("Clave: " + entry.getKey() + ", Valor: " + entry.getValue());
}
```

   ★ for (**Map.Entry**<*Integer*, *String*> entry : **personas**.***entrySet()***) { System.out.println("Clave: " + entry.***getKey()*** + ", Valor: " + entry.***getValue()***);
   }:
      - **personas**.***entrySet()***: devuelve un Set<Map.Entry<String, Integer>> que contiene todas las entradas del HashMap
      - Map.Entry<String, Integer> entry
         - Map.Entry es una interfaz que representa un par clave-valor
         - <*Integer*, *String*> indica que:
            - La clave es de tipo *Integer*
            - El valor es de tipo *String*
      - entry.***getKey()***: *obtiene la llave actual*
      - entry.***getValue()***: *obtiene el valor actual*

6. Investiga cómo funciona el método remove y elimina algún elemento de la tabla hash (solicita al usuario el elemento que se va a eliminar)

```
//remove
Scanner scanner = new Scanner(System.in);
System.out.println("Contenido del HashMap: " + personas);
System.out.print(s:"Ingrese la clave de la persona que quiere eliminar: ");
int elim = scanner.nextInt();

if (personas.containsKey(elim)) { //primero verificar si está la clave
    String persElim = personas.remove(elim); //puede almacenar el que se estpa quitando (su valor)
    System.out.println("se eliminó a: " + persElim);
    System.out.println("después de eliminar: " + personas + "\n");
} else {
    System.out.println("la clave" + elim + " no se encontró.\n");
}
scanner.close();
```

   ★ map.remove(clave) elimina el valor correspondiente a la clave
      - *fuente: Java HashMap remove() Method*

7. Qué otros métodos relevantes hay en la clase HashMap

   ★
```
HashMap<Integer, String> copiaPersonas = new HashMap<>(personas);
```

      - copia el hashmap que se le pasa como argumento en su constructor

★
```java
// clear() - Limpia el HashMap
HashMap<Integer, String> copiaPersonas = new HashMap<>(personas);
System.out.println("copia: " + copiaPersonas);
copiaPersonas.clear();
System.out.println("copia después de clear: " + copiaPersonas);
```

- map.**clear()** vacía el HashMap

★
```java
// isEmpty()
System.out.println("El HashMap está vacío? " + personas.isEmpty());
```

- map.**isEmpty()**: verifica si está vacío, devuelve un booleano

8. Investiga las diferencias de HashMap con TreeMap

*HashMap es una implementación basada en tablas hash que está ordenada, y TreeMap mantiene los elementos ordenados según su clave*

*fuente:*
*https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html*
*https://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html*

¿Qué métodos hay en TreeMap que no haya en HashMap?

★ *firstKey() y lastKey(): devuelven la primera y última clave según el orden.*

★ *lowerKey(), floorKey(), ceilingKey(), higherKey(): métodos que devuelven claves según el orden.*

★ *pollFirstEntry() y pollLastEntry(): quitany devuelven la primera/última entrada.*

★ *descendingMap(): devuelve una vista del mapa en orden inverso.*

★ *subMap(), headMap(), tailMap(): devuelven vistas de subconjuntos del mapa con rangos de claves.*