

Set, Map y Priority Queue

Profesor: Yerko Ortiz

Ayudante Lab: Diego Banda

Set

```
System.out.println("-----hashSet-----");
Set<Integer> hashSet = new HashSet<>();
hashSet.add(5);
hashSet.add(1);
hashSet.add(5);
System.out.println("HashSet: " + hashSet);
```

```
System.out.println("-----linkedHashSet-----");
Set<Integer> linkedHashSet = new LinkedHashSet<>();
linkedHashSet.add(5);
linkedHashSet.add(1);
linkedHashSet.add(5);
System.out.println("LinkedHashSet: " + linkedHashSet);
```

```
System.out.println("-----TreeSet-----");
Set<Integer> treeSet = new TreeSet<>();
treeSet.add(5);
treeSet.add(1);
treeSet.add(5);
System.out.println("TreeSet: " + treeSet); // Ordenado
```

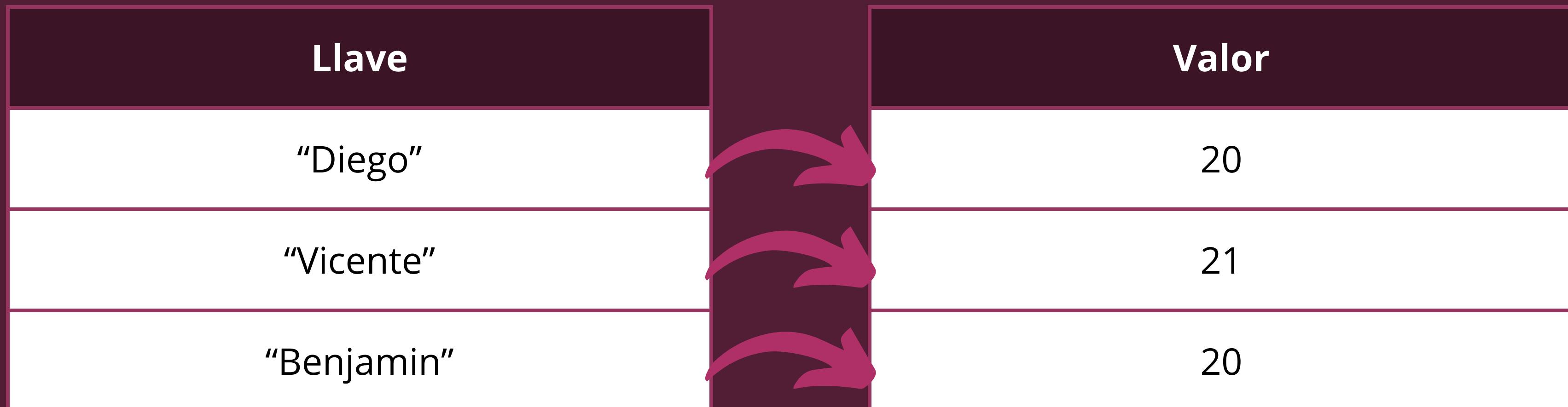
“Set” es una colección de datos que no permite duplicados.

Los elementos de un Set no están ordenados.

Varias implementaciones de set en Java: hashSet, linkedHashSet y TreeSet

Map

- “Map” es una colección de datos del tipo llave-valor
- Cada llave es única y tiene un solo valor
- Varias implementaciones de Map en Java: hashMap, linkedHashMap y TreeMap

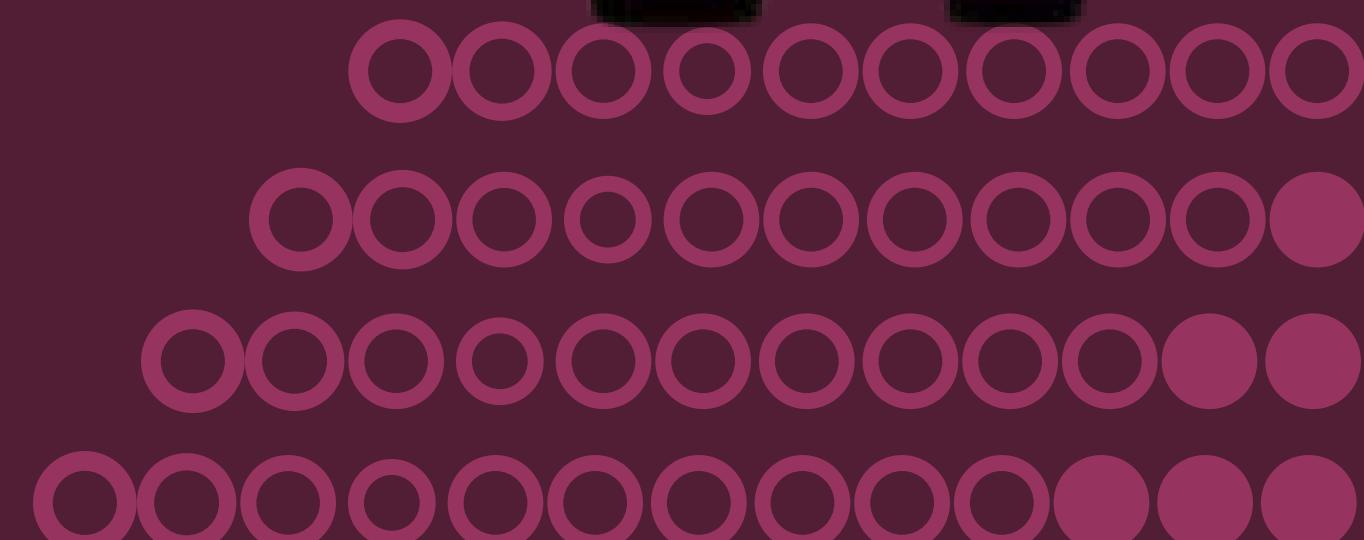


Map

```
System.out.println("-----hashMap-----");
Map<String, Integer> hashMap = new HashMap<>();
hashMap.put("Diego", 20);
hashMap.put("Vicente", 21);
hashMap.put("Benjamin", 20);
System.out.println("HashMap: " + hashMap);

System.out.println("-----linkedHashMap-----");
Map<String, Integer> linkedHashMap = new LinkedHashMap<>();
linkedHashMap.put("Diego", 20);
linkedHashMap.put("Vicente", 21);
linkedHashMap.put("Benjamin", 20);
System.out.println("LinkedHashMap: " + linkedHashMap);

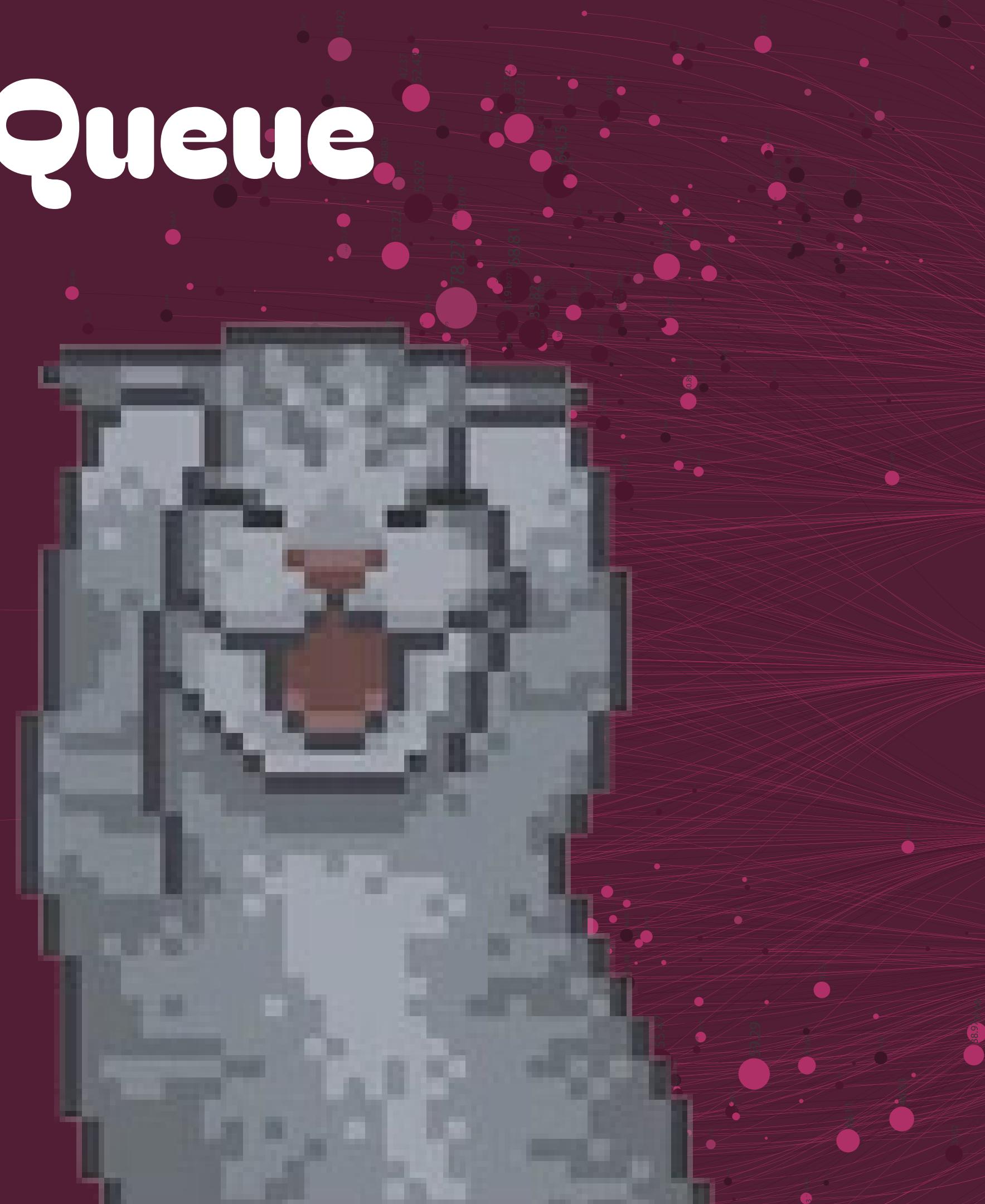
System.out.println("-----TreeMap-----");
Map<String, Integer> treeMap = new TreeMap<>();
treeMap.put("Diego", 20);
treeMap.put("Vicente", 21);
treeMap.put("Benjamin", 20);
System.out.println("TreeMap: " + treeMap); // Ordenado por llave
```



Priority Queue

“Priority Queue” es una cola que ordena sus elementos a la hora de insertarse.

Se ordenan a partir de un “comparator” entregado.



Problema 1: Playlist

Recibes una playlist de una estación de radio, esta tiene un total de **n** canciones. ¿Cuál es la máxima secuencia de canciones que se pueden escuchar sin repetir ninguna?

Input: La primera linea recibe un entero **n** con la cantidad de canciones. La siguiente linea tiene **n** enteros con el id de cada canción

Output: Imprime el largo de la máxima secuencia que se puede escuchar sin repetir ninguna canción

Ejemplo:
8
1 2 1 3 2 7 4 2

Ejemplo:
5



Problema 2: Torres

Te dan **n** cubos con un tamaño y en un orden determinado y debes crear torres con ellos. El cubo de arriba siempre debe ser de tamaño inferior que el de abajo, debes procesar los cubos en el orden entregado, puedes colocar el cubo en una torre existente o crear una nueva ¿Cuál es el numero mínimo de torres a crear?

Input: La primera linea contiene un entero **n** con la cantidad de cubos. La siguiente linea tiene n enteros con el tamaño de los cubos

Ejemplo:

5

3 8 2 1 5

Output: Cantidad mínima de torres que se pueden crear

Ejemplo:

2

1

2

3

5

8

Problema 3: Habitaciones de Hotel

Usted maneja un hotel el cual recibe muchas personas y quiere crear un programa para automatizar el uso de las habitaciones. Una cantidad n de clientes llegará pronto y cada uno quiere una habitación solo, sabes el día de llegada y de ida de cada cliente, un cliente puede usar una habitación usada por otro cliente si su día de llegada es después del día de ida del cliente anterior. ¿Cuál es el número mínimo de habitaciones a usar? ¿Y cómo pueden ser usadas las habitaciones?

Input	
Descripción	Ejemplo
n : número de clientes que llegarán (Integer)	3
a, b : a es el día de llegada y b el día de ida de cada cliente	1 2 2 4 4 4

Output	
Descripción	Ejemplo
Cantidad de habitaciones usadas	2
Número de habitación usada por cada cliente	1 2 1