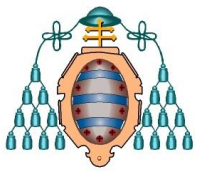
ESTRUCTURAS JERARQUICAS SESIÓN 3

Estructura de Datos

2021-2022

María del Rosario Suárez Fernández



Departamento de Informática
UNIVERSIDAD DE OVIEDO

Proyecto3 - Árboles

- Abrir el proyecto llamado **Arboles**
- Crear un nuevo paquete llamado **ColasPrioridad**
- Dentro del paquete crear:
  - Interface **PriorityQueue** → define los métodos cuya implementación dependerá del tipo de filosofía aplicada para la gestión de Colas de prioridad

```
public interface PriorityQueue <T extends Comparable <T>>
```

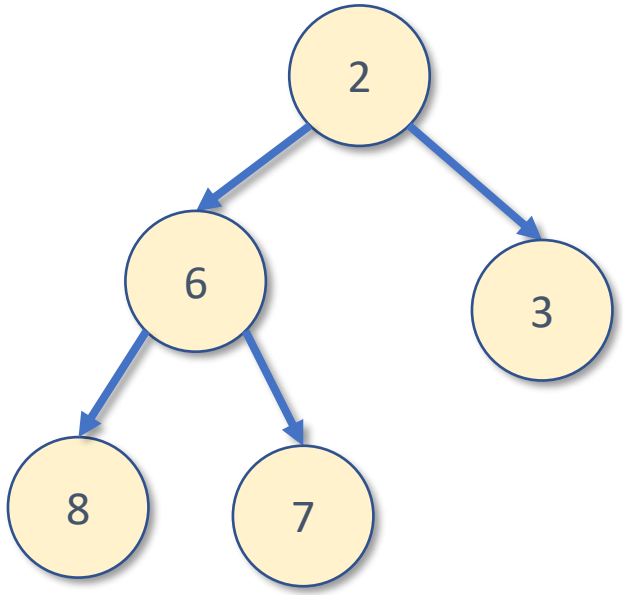
Interface - Métodos

- `public int add(T elemento)`
- `public T sacar()`
- `public int remove (T elemento)`
- `public boolean isEmpty()`
- `public void clear()`
- `public int cambiarPrioridad(int pos, T elemento)`
- `public String toString()`

Proyecto3 - Árboles

- Dentro del paquete **ColasPrioridad** crear:
  - Clase **BinaryHeapMin** → implementa una cola de prioridad mediante un montículo de mínimos donde se admiten **claves repetidas**

```
public class BinaryHeapMin <T extends Comparable <T>> implements PriorityQueue<T>
```



Montículo de mínimos

2	6	3	8	7	

Clase BinaryHeap - Propiedades

Propiedad	Descripción
`private T[] monticulo`	Vector de elementos de un determinado tamaño
`private int numElementos`	Número de elementos insertados en el vector

Montículo de mínimos

Clase BinaryHeap - Métodos

- `public BinaryHeap(int n)`
- `public int add(T elemento)`
- `public T sacar()`
- `public int remove (T elemento)`
- `public boolean isEmpty()`
- `public void clear()`
- `public int cambiarPrioridad(int pos, T elemento)`
- `public String toString()`
  - Elementos separados por tabuladores. No hay tabulador al final.

Clase BinaryHeap – Métodos opcionales

- `private void filtradoAscendente(int pos)`
  - Filtrado ascendente a partir de la posición pasada como parámetro. Es necesario realizarlo cada vez que se inserta un nuevo elemento
- `private void filtradoDescendente(int pos)`
  - Filtrado descendente a partir de la posición pasada como parámetro. Es necesario realizarlo cada vez que se borra un elemento o se invoca al método ***sacar***

Tareas para casa (Evaluación continua)

- Terminar los métodos de la clase BinaryHeapMin funcionando
- Elaborar un conjunto de baterías de test que prueben:
  - El método añadir con todos los casos posibles de error
  - El método borrar con todos los casos posibles de error
  - El método poll con todos los casos posibles de error
- Documentar las clases con JavaDoc