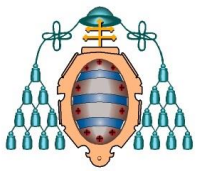
ESTRUCTURAS JERARQUICAS SESIÓN 1

Estructura de Datos

2021-2022

María del Rosario Suárez Fernández

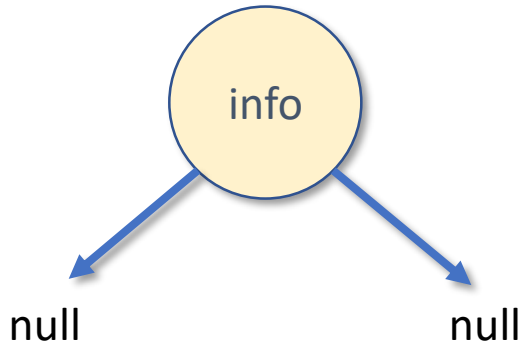


Departamento de Informática
UNIVERSIDAD DE OVIEDO

Proyecto3 - Árboles

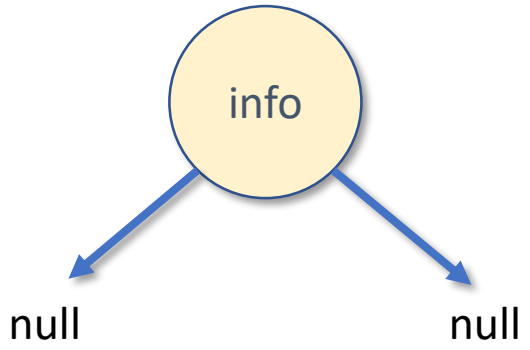
- Crear un nuevo proyecto llamado **Arboles**
- Crear un nuevo paquete llamado **BST**
- Dentro del paquete crear:
  - Clase **BSTNode** → implementa el nodo de un árbol BST

```
public class BSTNode <T extends Comparable <T>>
```



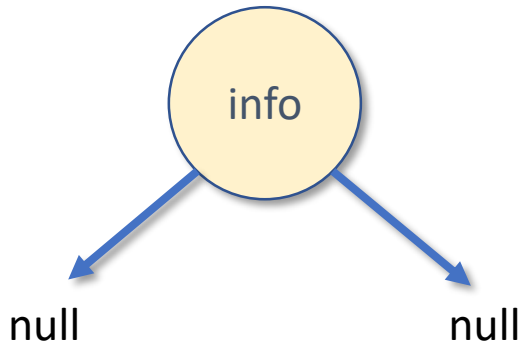
Clase BSTNode - Propiedades

Propiedad	Descripción
`private T info;`	En contenido del nodo que será de tipo genérico
`private BSTNode<T> left`	Nodo hijo izquierdo
`private BSTNode<T> right`	Nodo hijo derecho



Clase BSTNode - Métodos

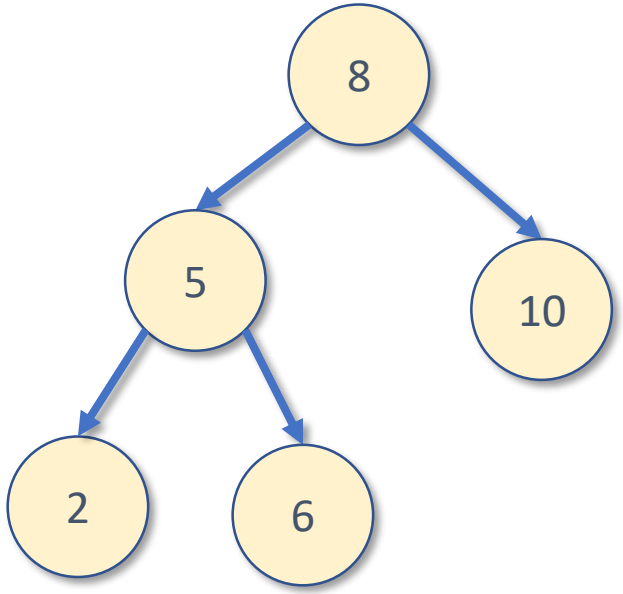
- `public BSTNode(T clave)`
- `public void setInfo(T clave)`
- `public T getInfo()`
- `public void setLeft(BSTNode<T> nodo)`
- `public void setRight(BSTNode<T> nodo)`
- `public BSTNode<T> getLeft()`
- `public BSTNode<T> getRight()`
-
```
public String toString() {
    return info.toString();
}
```



Proyecto3 - Árboles

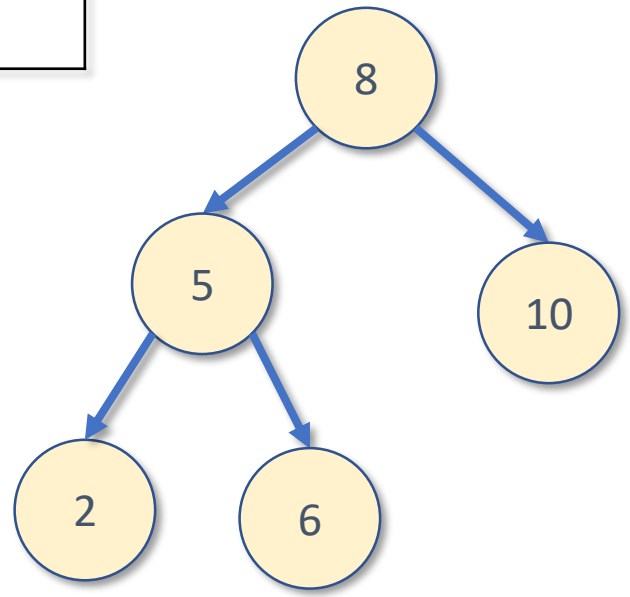
- Dentro del paquete **BST** crear:
  - Clase **BSTree** → implementa un árbol BST

```
public class BSTree<T extends Comparable<T>>
```



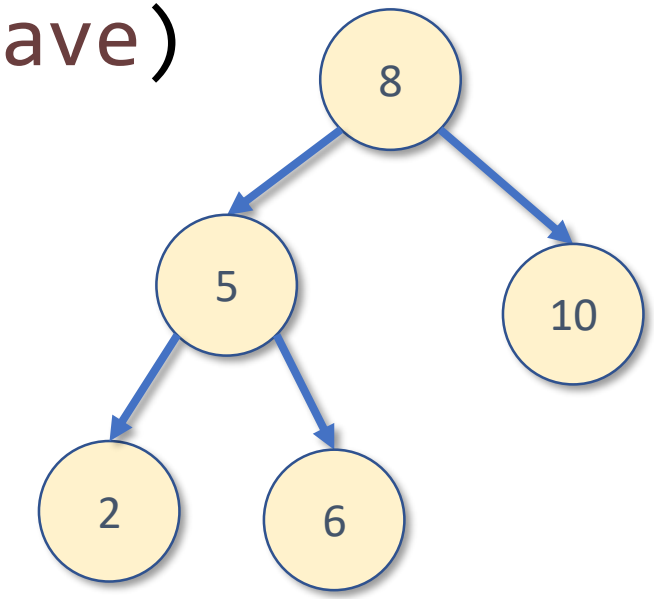
Clase BSTree - Propiedades

Propiedad	Descripción
`private BSTNode<T> raiz`	Nodo raíz del árbol BST

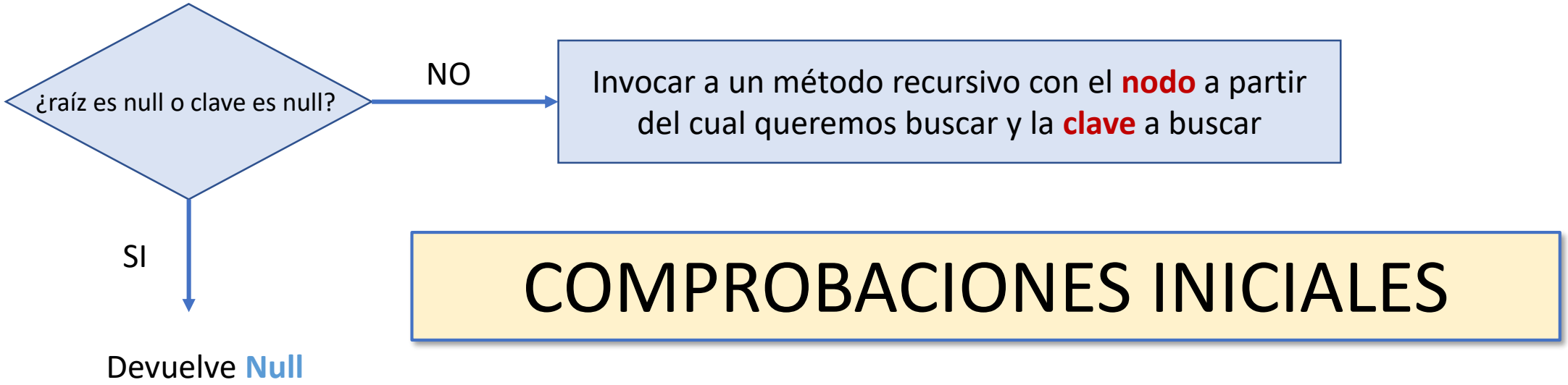


Clase BSTree - Métodos

- `public BSTree()`
- `public BSTNode<T> searchNode(T clave)`
- `public int addNode(T clave)`
- `public String preOrder()`
- `public String postOrder()`
- `public String inOrder()`
- `public int removeNode(T clave)`

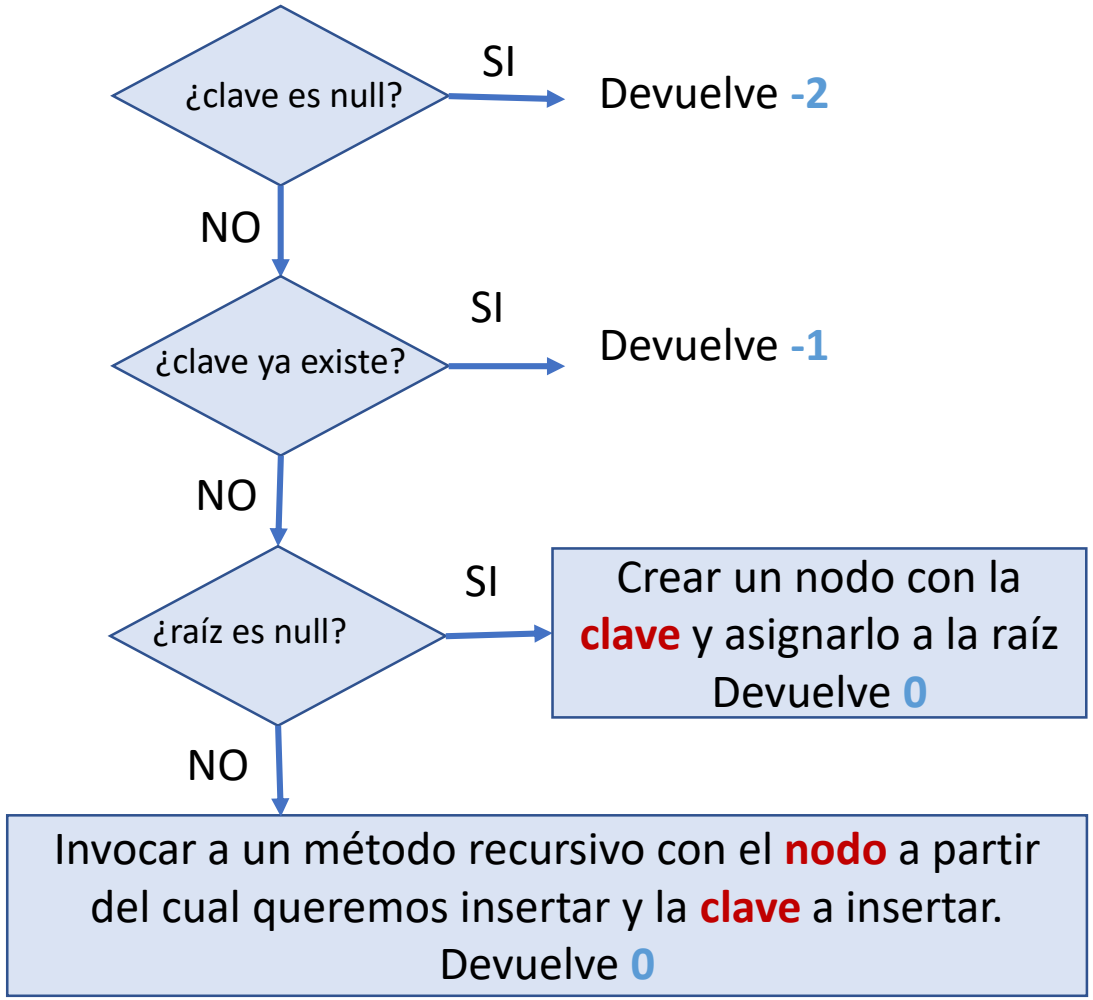


Clase BSTree - searchNode



El **método recursivo**, buscar la **clave** por la derecha o por la izquierda, dependiendo de si esta es mayor o menor que el **nodo** a partir del cual se busca. Devuelve el **nodo completo** encontrado o **Null** si no lo encuentra

Clase BSTree - addNode



COMPROBACIONES INICIALES

El **método recursivo**, busca un hueco por la derecha o por la izquierda, dependiendo de si la **clave** a insertar es mayor o menor que el **nodo** a partir del cual se quiere insertar. Cuando lo encuentra, crea un nuevo nodo con la **clave** y lo asigna por la derecha o por la izquierda.
Devuelve **0**

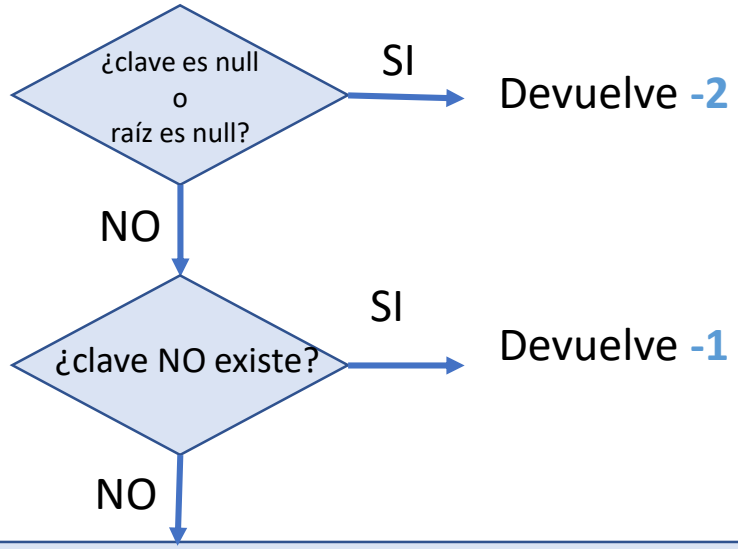
Clase BSTree - Recorridos

- Devolver una cadena con el recorrido. Elementos separados por tabuladores, eliminando el último tabulador si lo hubiese
- Algoritmo general para cualquier recorrido:

```
public String recorrido() {
    String cadena=recorridoR(raiz);
    return cadena.substring(0,cadena.length()-1);
}
```

- Recorridos
  - preOrder() → raíz, izquierda, derecha
  - postOrder() → izquierda, derecha, raíz
  - inOrder() → izquierda, raíz, derecha

Clase BSTree - removeNode



Invocar a un método recursivo con el **nodo** a partir del cual queremos borrar y la **clave** a borrar.
Asignar a la raíz lo que devuelve el método recursivo
Devuelve **0**

COMPROBACIONES INICIALES

El **método recursivo**, busca la clave por la derecha o por la izquierda, dependiendo de si la **clave** a borrar es mayor o menor que el **nodo** a partir del cual se quiere borrar. Cuando la encuentra (puede no tener hijos, un hijo o dos) la borra con los mecanismos vistos en clase de teoría.
Devuelve el **nodo actualizado**

Tareas para casa (Evaluación continua)

- Terminar los métodos de la clase BSTree 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
  - Los recorridos
- Documentar las clases con JavaDoc