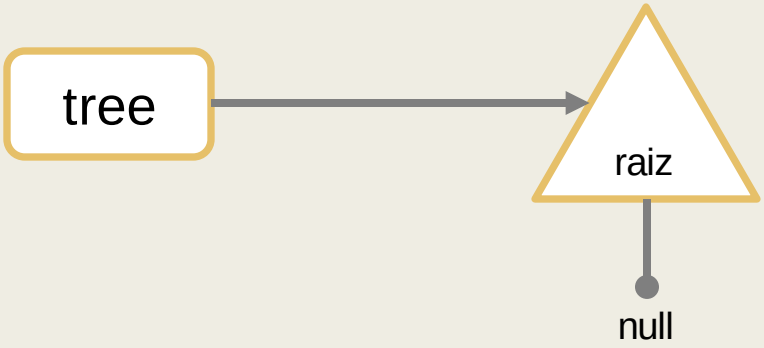
PRÁCTICO 2 - ÁRBOLES

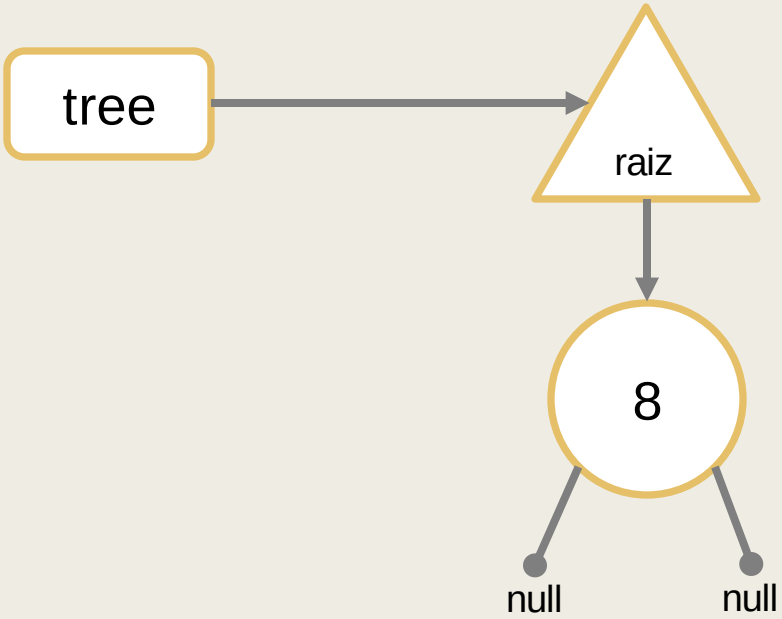
Implementación propuesta

Árbol con nodos



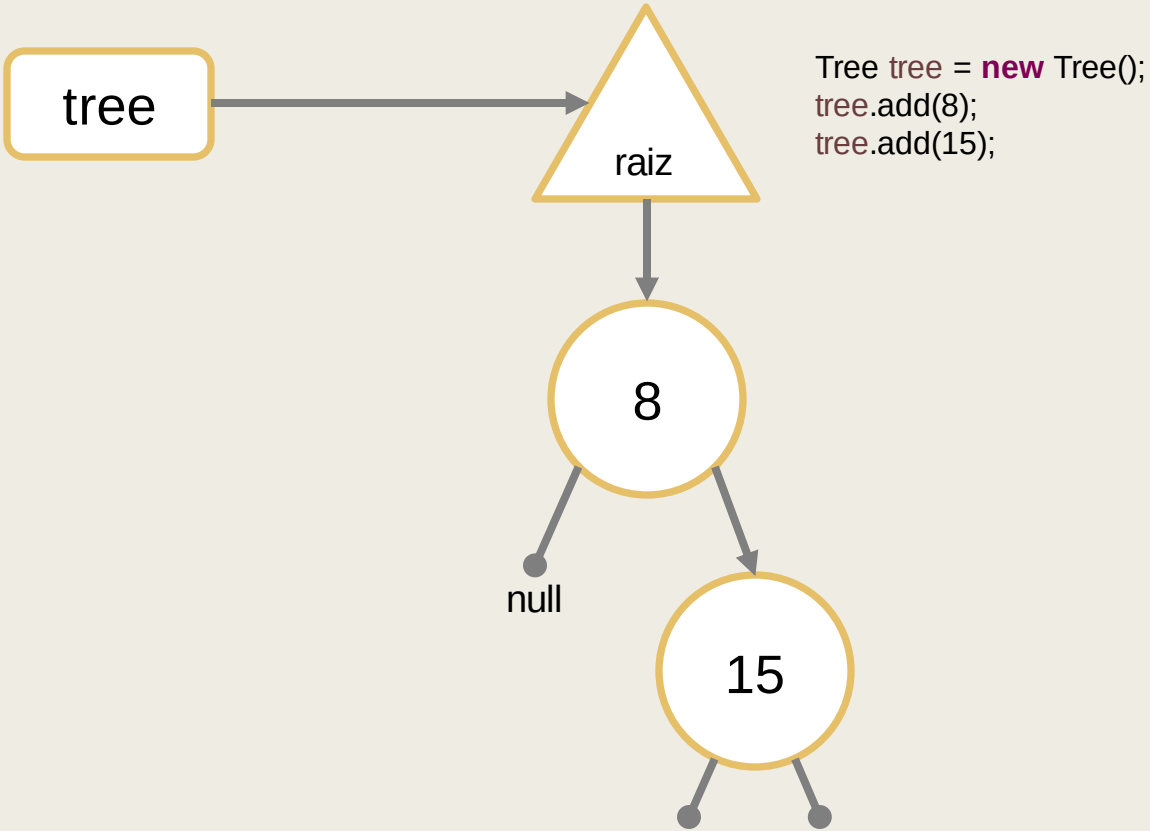

```
Tree tree = new Tree();
```

Árbol con nodos



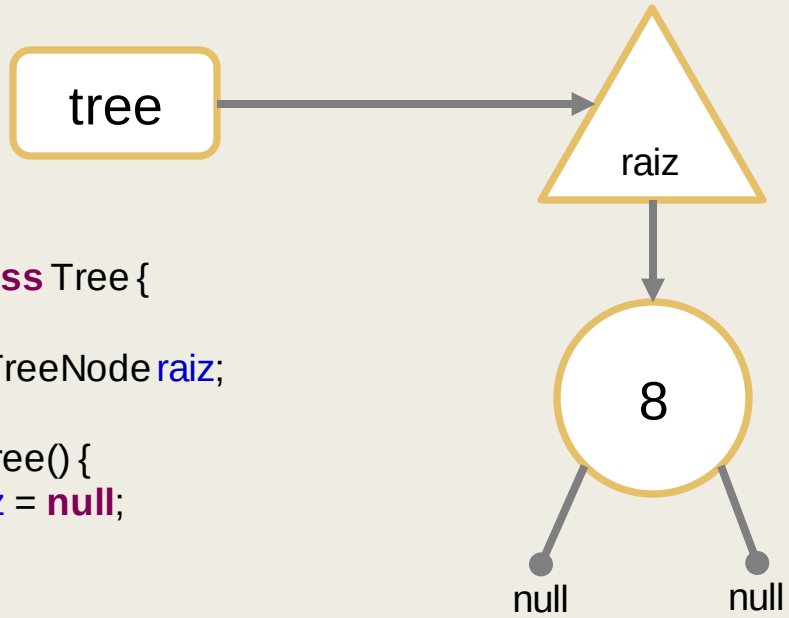

```
Tree tree = new Tree();
tree.add(8);
```

Árbol con nodos



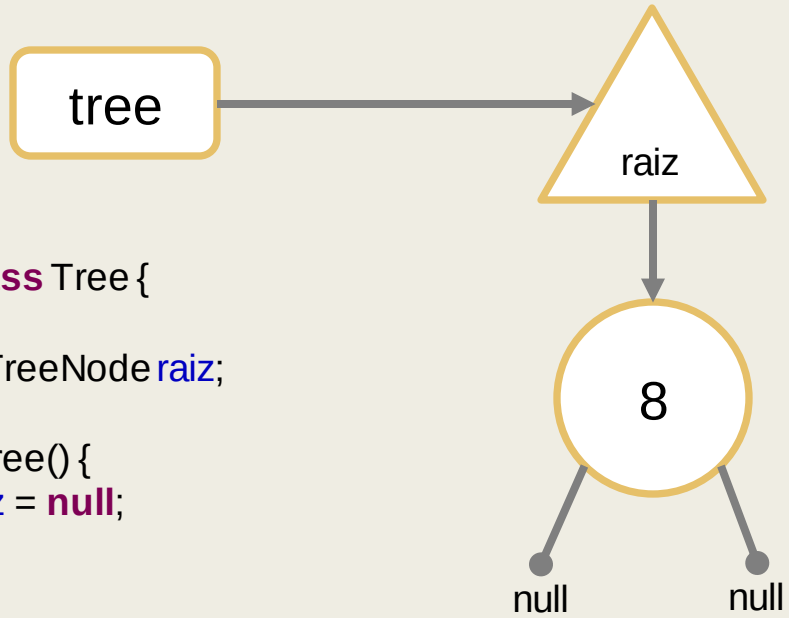

```
Tree tree = new Tree();
tree.add(8);
tree.add(15);
```

Implementación




```
public class Tree {

  private TreeNode raiz;

  public Tree() {
    this.raiz = null;
  }

  …
}
```

Implementación

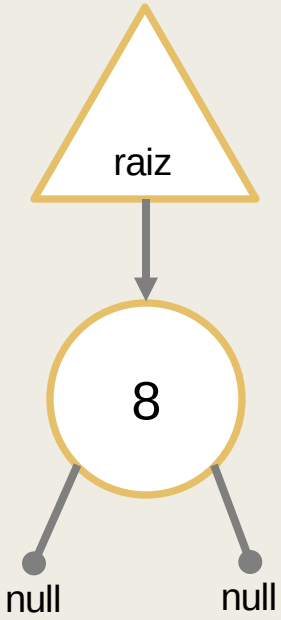



```
public class Tree {

  private TreeNode raiz;

  public Tree() {
    this.raiz = null;
  }

  …
}
```

```
public class TreeNode {

  private int valor;
  private TreeNode izquierda;
  private TreeNode derecha;

  public TreeNode(int value) {
    this.valor = value;
    this.izquierda = null;
    this.derecha = null;
  }

  …
}
```

Implementación

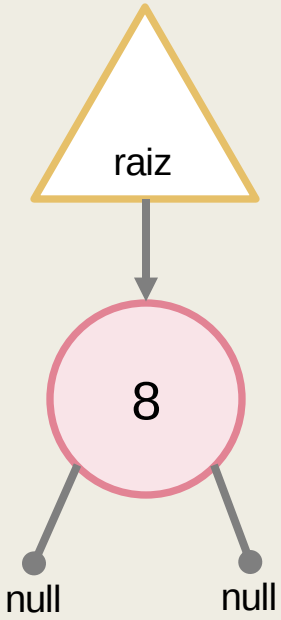
¿Cómo afecta a la hora de implementar los métodos (add, delete, etc.)?




```
public void add(TreeNode nodo, int valor) {
  if (nodo.getValor() > valor) {
    if (nodo.getIzq() == null) {
      TreeNode temp = new TreeNode(valor);
      nodo.setIzq(temp);
    } else {
      add(nodo.getIzq(),valor);
    }
  } else if (nodo.getValor() < valor) {
    if (nodo.getDer() == null) {
      TreeNode temp = new TreeNode(valor);
      nodo.setDer(temp);
    } else {
      add(nodo.getDer(),valor);
    }
  }
}
```

Implementación

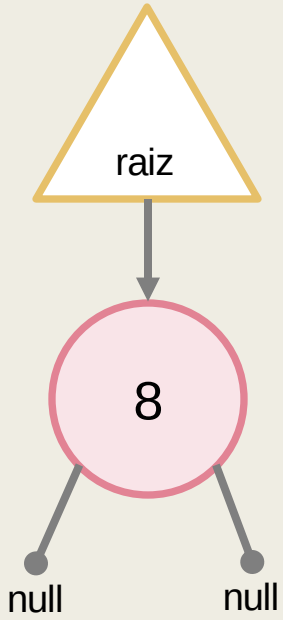
¿Cómo afecta a la hora de implementar los métodos (add, delete, etc.)?




```
public void add(TreeNode nodo, int valor) {
  if (nodo.getValor() > valor) {
    if (nodo.getIzq() == null) {
      TreeNode temp = new TreeNode(valor);
      nodo.setIzq(temp);
    } else {
      add(nodo.getIzq(),valor);
    }
  } else if (nodo.getValor() < valor) {
    if (nodo.getDer() == null) {
      TreeNode temp = new TreeNode(valor);
      nodo.setDer(temp);
    } else {
      add(nodo.getDer(),valor);
    }
  }
}
```

Implementación

¿Cómo afecta a la hora de implementar los métodos (add, delete, etc.)?




```
public void add(TreeNode nodo, int valor) {
  if (nodo.getValor() > valor) {
    if (nodo.getIzq() == null) {
      TreeNode temp = new TreeNode(valor);
      nodo.setIzq(temp);
    } else {
      add(nodo.getIzq(),valor);
    }
  } else if (nodo.getValor() < valor) {
    if (nodo.getDer() == null) {
      TreeNode temp = new TreeNode(valor);
      nodo.setDer(temp);
    } else {
      add(nodo.getDer(),valor);
    }
  }
}
```

¡Rompo encapsulamiento!

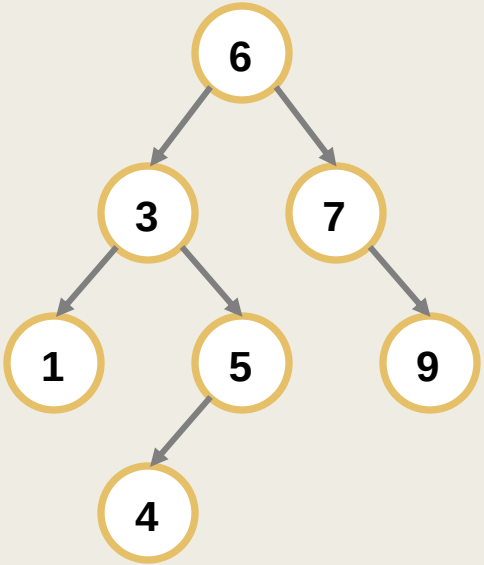
Implementación

¿Cómo afecta a la hora de implementar los métodos (add, delete, etc.)?

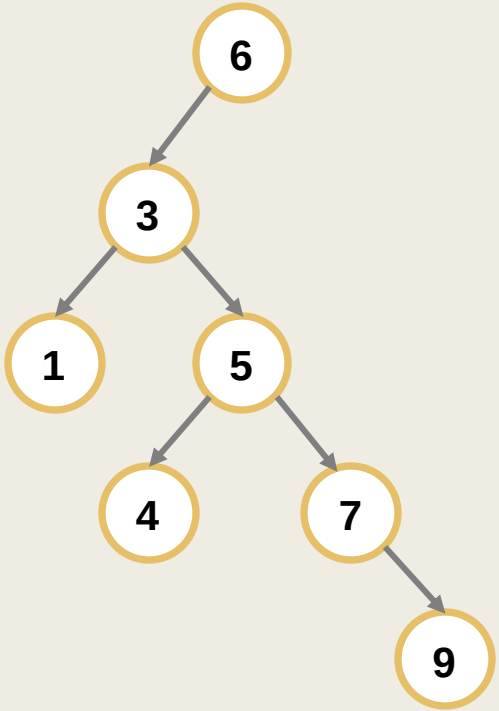
```
public void add(int valor) {
  if (this.raiz == null)
    this.raiz = new TreeNode(valor);
  else
    this.add(this.raiz,valor);
}
```

```
private void add(TreeNode nodo, int valor) {
  if (nodo.getValor() > valor) {
    if (nodo.getIzq() == null) {
      TreeNode temp = new TreeNode(valor);
      nodo.setIzq(temp);
    } else {
      add(nodo.getIzq(),valor);
    }
  } else if (nodo.getValor() < valor) {
    if (nodo.getDer() == null) {
      TreeNode temp = new TreeNode(valor);
      nodo.setDer(temp);
    } else {
      add(nodo.getDer(),valor);
    }
  }
}
```

Recorrido pre-orden en un árbol binario

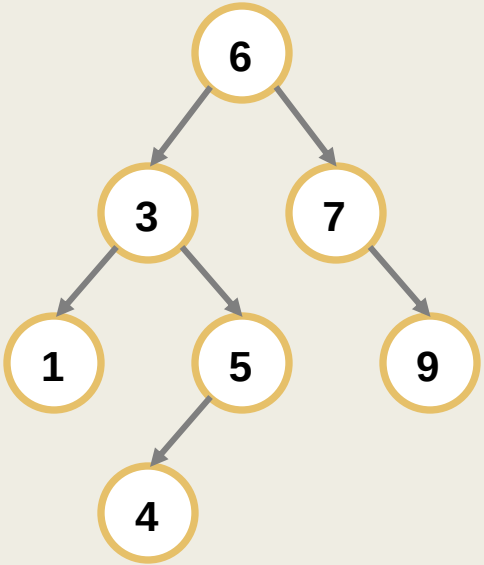


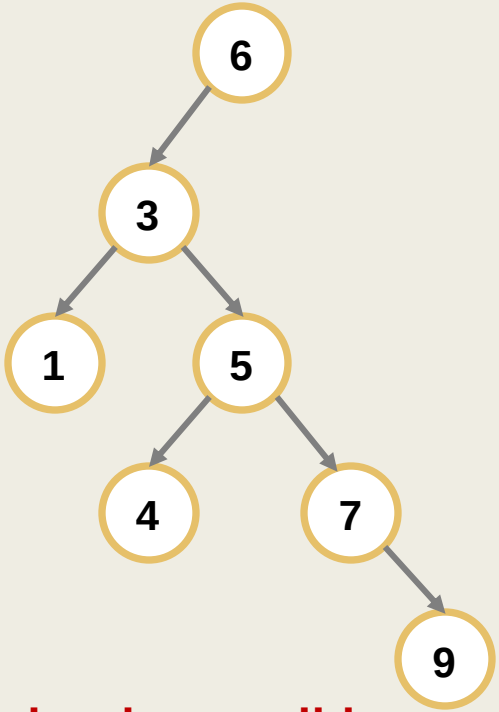
Salida: 6 3 1 5 4 7 9



Salida: 6 3 1 5 4 7 9

Recorrido pre-orden en un árbol binario



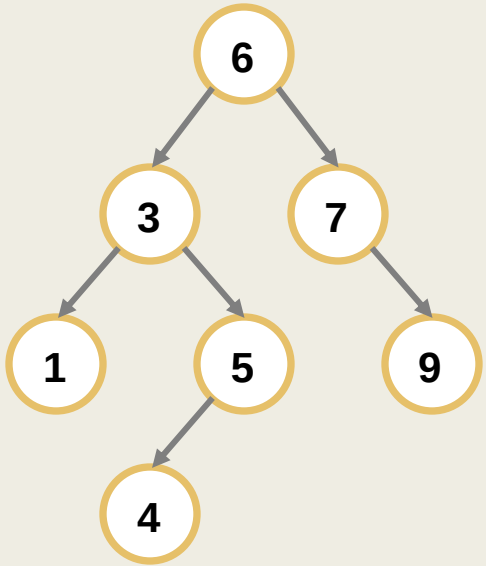


Dos árboles distintos pueden dar la misma salida

Salida: 6 3 1 5 4 7 9

Salida: 6 3 1 5 4 7 9

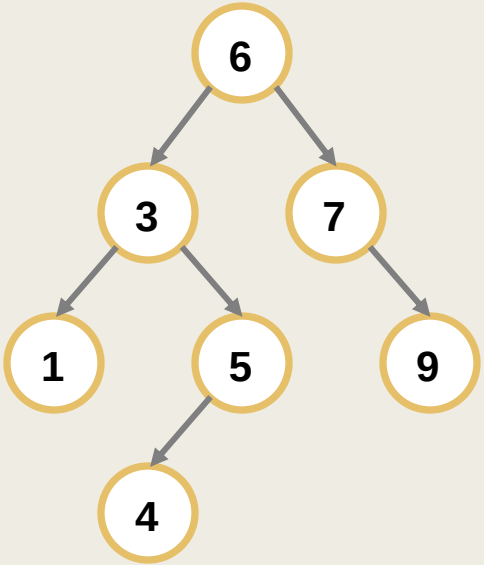
Recorrido pre-orden en un árbol binario



Modificamos el recorrido pre-orden para que nos devuelva más información:

- Hacemos el recorrido normalmente mostrando los valores de los nodos.
- Cuando nos encontramos con un nodo vacío (es decir, un izquierda o derecha en null) mostramos un caracter especial (“-”).

Recorrido pre-orden en un árbol binario



Modificamos el recorrido pre-orden para que nos devuelva más información:

- Hacemos el recorrido normalmente mostrando los valores de los nodos.
- Cuando nos encontramos con un nodo vacío (es decir, un izquierda o derecha en null) mostramos un caracter especial (“-”).

Salida: 6 3 1 - - 5 4 - - - 7 - 9 - -