# Java Week 7

**public** **class** BinaryTree<T **extends** Comparable<? **super** T>> **implements** BTree<T>

{

TreeNode<T> root;

@Override

**public** **void** insert(T value)

{

//if the rool is empty

**if**(root==**null**)

{

//Add the new node to the root

root = **new** TreeNode<T>(value);

}

//if the value compared to the root is less than zero (it's smaller than the root)

**else** **if**(value.compareTo(value())<0)

{

//move to the left node, try insert again

root.left().insert(value);

}

**else**

{

//move to the right node, try insert again

root.right().insert(value);

}

}

//Get root value

@Override

**public** T value()

{

**return** root.value;

}

//Get value of the left node

@Override

**public** BTree<T> left()

{

**return** root.left;

}

//Get value of the right node

@Override

**public** BTree<T> right()

{

**return** root.right;

}

**public** **static** **void** main(String args[])

{

{

BinaryTree<Integer> tree = **new** BinaryTree<>();

tree.insert(1);

tree.insert(2);

tree.insert(3);

tree.insert(-1);

Integer leftV = tree.left().value();

Integer rightV = tree.right().value();

Integer rightRightV = tree.right().right().value();

System.***out***.println(tree.value());

System.***out***.println(leftV);

System.***out***.println(rightV);

System.***out***.println(rightRightV);

}

}

Results:

1

-1

2

3