Binary Tree Implementation

For the implementation, there's an auxiliary *Node* class that will store *int* values and keeps a reference to each child. The first step is to find the place where we want to add a new node in order to keep the tree sorted. We'll follow these rules starting from the root node:

- **if the new node's value is lower than the current node's, go to the left child**
- **if the new node's value is greater than the current node's, go to the right child**
- **when the current node is *null*, we've reached a leaf node, we insert the new node in that position**

```
public class Tree {
        static class Node {
        int value;
        Node left, right;
        Node(int value){
            this.value = value;
            left = null;
            right = null;
           }
        }
        public void insert(Node node, int value)
        {
        if (value < node.value) {
                if (node.left != null) {
                   insert(node.left, value);
                   }
                   else
                   {
                    System.out.println(" Inserted " + value + " to left of " + node.value);
                     node.left = new Node(value);
                   }
                   } else if (value > node.value)
                   {
                   if (node.right != null)
                   {
                  insert(node.right, value);
                   }
                   else
                   {
                    System.out.println("Inserted "+ value+" to right of "+ node.value);
                   node.right = new Node(value);
               }
              }
        }
   public void traverseInOrder(Node node) {
     if (node != null) {
        traverseInOrder(node.left);
        System.out.print(" " + node.value);
        traverseInOrder(node.right);
     }
   }
```

```
    public static void main(String args[])
    {
    Tree tree = new Tree();
            Node root = new Node(5);
            System.out.println("Binary Tree Example");
            System.out.println("Building tree with root value " + root.value);
            tree.insert(root, 2);
            tree.insert(root, 4);
            tree.insert(root, 8);
            tree.insert(root, 6);
            tree.insert(root, 7);
            tree.insert(root, 3);
            tree.insert(root, 9);
            System.out.println("Traversing tree in order");
            tree.traverseInOrder(root);
          }
}
```