# LAB REPORT 10

Course:CSE 207

**Sec:3**

Submitted By:Nuran Farhana Prova

Student Id: 2023-1-60-075

Program Name: Object-Oriented Programming(Data Structure)

Date: 5/6/2024

**Submitted to:**

<u>**Dr. Anup Kumar Paul**</u>

Associate professor,

Department of Computer Science and Engineering

## Main Class:

```
package BST;
import java.util.Scanner;
public class Main {
	public static void main(String[] args) {
	    Scanner scanner = new Scanner(System.in);
	    // Manual input of tree data
	    System.out.println("Enter the root value:");
	    int rootValue = scanner.nextInt();
	    Node root = new Node(rootValue);
	    System.out.println("Enter the left child value of the root (or -1 if 
none):");
	    int leftValue = scanner.nextInt();
	    if (leftValue != -1) {
	        root.lchild = new Node(leftValue);
	    }
	    System.out.println("Enter the right child value of the root (or -1 if 
none):");
	    int rightValue = scanner.nextInt();
	    if (rightValue != -1) {
	        root.rchild = new Node(rightValue);
	    }
	    // Inorder Traversal
	    System.out.println("Inorder Traversal:");
	    TreeTraversal.inorder(root);
	    System.out.println();
	    // Preorder Traversal
	    System.out.println("Preorder Traversal:");
	    TreeTraversal.preorder(root);
	    System.out.println();
	    // Postorder Traversal
	    System.out.println("Postorder Traversal:");
	    TreeTraversal.postorder(root);
	    System.out.println();
	    scanner.close();
	}
    }
```

## Node :

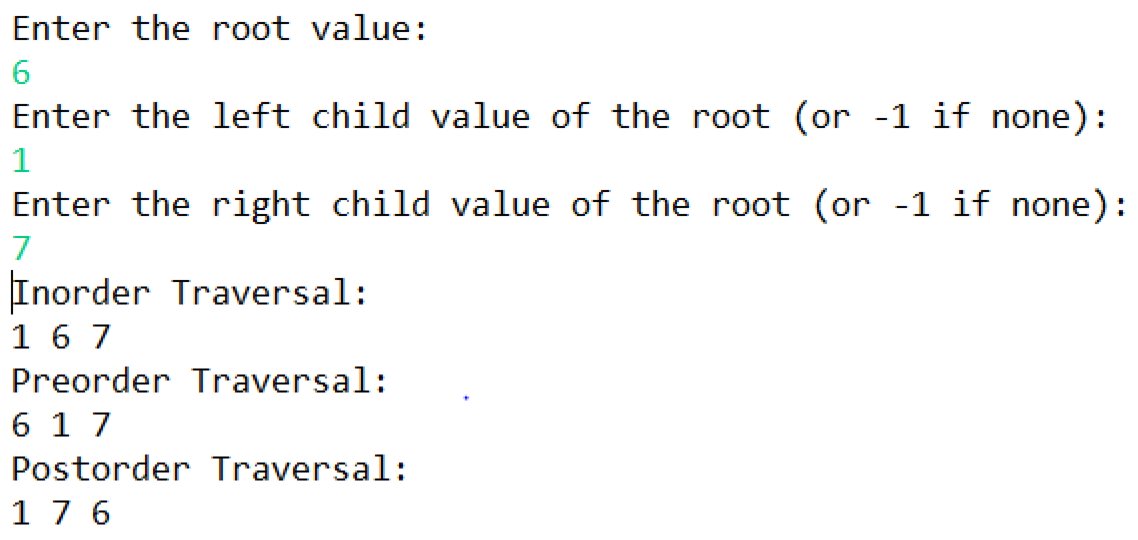
```
package BST;
public class Node {
   int data;
   Node lchild;
   Node rchild;
   public Node(int data) {
       this.data = data;
       this.lchild = null;
       this.rchild = null;
   }
}
```

## Source Code:

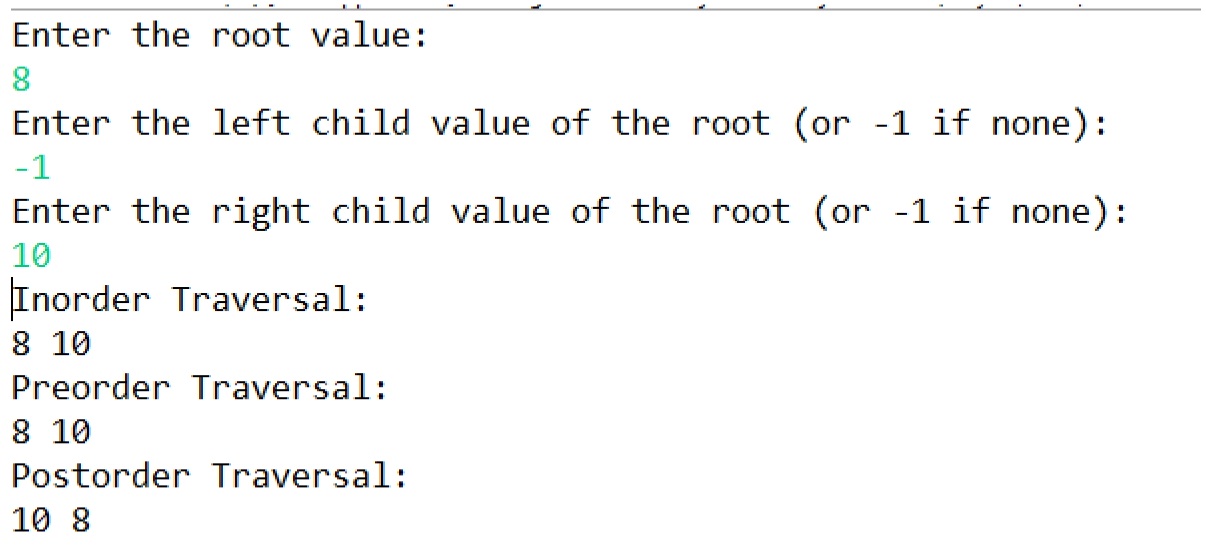
```
package BST;
public class TreeTraversal {
   // Inorder Traversal
   public static void inorder(Node root) {
       if (root != null) {
           inorder(root.lchild);
           System.out.print(root.data + " ");
           inorder(root.rchild);
       }
   }
   // Preorder Traversal
   public static void preorder(Node root) {
       if (root != null) {
           System.out.print(root.data + " ");
           preorder(root.lchild);
           preorder(root.rchild);
       }
   }
   // Postorder Traversal
   public static void postorder(Node root) {
       if (root != null) {
           postorder(root.lchild);
           postorder(root.rchild);
```

```
            System.out.print(root.data + " ");
        }
    }
}
```

## Output:



```
Enter the root value:
6
Enter the left child value of the root (or -1 if none):
1
Enter the right child value of the root (or -1 if none):
7
Inorder Traversal:
1 6 7
Preorder Traversal:
6 1 7
Postorder Traversal:
1 7 6
```



```
Enter the root value:
8
Enter the left child value of the root (or -1 if none):
-1
Enter the right child value of the root (or -1 if none):
10
Inorder Traversal:
8 10
Preorder Traversal:
8 10
Postorder Traversal:
10 8
```