

**Assignment Group: 14**

**Program:** Automotive Software Engineering (Master)

**Subject:** Software Engineering and Programming Basics W18/19

**Date:** 07/01/2019

Group Members	
Name	Immatrlikulation Number
Majid Ali Khan	510868
Ishfaqe Ahmed	553225
Ammad Jamil Chaudhry	511951
Muhammad Umair Sabir	487265
Abdul Rehman Sweidat	513012
Murtaza Ali	553212
Faheem Azfar Bhatti	554558
Aashar Azeem	562784
Waseem Khan	481388

## Code:

## Main Class:

---

```
public class Main {
    public static void main(String[] args){
        //Order of insertion - [5,7,8,15,20,21,13,18,11,42]

        //Inserted first five values 5,7,8,15,20
        Node ob = new Node();
        Node firstNode = ob.createNode(5);
        Node secondNode = ob.createNode(7);
        Node thirdNode = ob.createNode(8, firstNode, secondNode);
        Node fourthNode = ob.createNode(15);
        Node fifthNode = ob.createNode(20, thirdNode, fourthNode);

        //Inserted second five values 21,13,18,11,42
        Node sixthNode = ob.createNode(21);
        Node seventhNode = ob.createNode(13);
        Node eighthNode = ob.createNode(18, sixthNode, seventhNode);
        Node ninthNode = ob.createNode(11, eighthNode, null);

        //Root
        Node root = ob.createNode(42, fifthNode, ninthNode);

        //Setting parents of the node manually
        firstNode.setParent(thirdNode);
        secondNode.setParent(thirdNode);
        thirdNode.setParent(fifthNode);
        fourthNode.setParent(fifthNode);
        sixthNode.setParent(eighthNode);
        seventhNode.setParent(eighthNode);
        eighthNode.setParent(ninthNode);
        ninthNode.setParent(root);
        fifthNode.setParent(root);

        Visitor visitor = new Visitor();
        //Providing any node in the tree we find the root and print its value
        System.out.println("Root is --> " + visitor.findRoot(firstNode).getNodeValue());
        //Providing public node we find sum of all the leaves
        System.out.println("Sum of all the leaves is --> " + visitor.sumOfLeaves(visitor.findRoot(firstNode)));
        //Providing public node we find sum of all the nodes
        System.out.println("Sum of all the nodes is --> " + visitor.sumOfNodes(visitor.findRoot(firstNode)));
    }
}
```

## Visitable Interface:

---

```
//This interface is used to demonstrate the visitor design pattern
public interface Visitable<T>
{
    Node findRoot(Node node);
    int sumOfLeaves(Node node);
    int sumOfNodes(Node node);
}
```

## Node Class:

---

```
public class Node //Generic to make it accept Integer, Character, Double, Anything
{
    private int nodeValue; // node's value
    private Node left, right, parent; // subtree reference
    // create instance with a value and null subtrees
    public Node(int item)
    {
        nodeValue = item;
        left = right = parent = null;
    }

    public Node(){ //for initial case
    }

    // initialize the value and the subtrees
    public Node (int item, Node left, Node right)
    {
        nodeValue = item;
        this.left = left;
        this.right = right;
    }

    //sugar for creating nodes rather than following default constructor, saves writing new multiple times
    public Node createNode(int item, Node left, Node right){
        return new Node(item, left, right);
    }
    //overloading for single parameter
    public Node createNode(int item){
        return new Node(item, left, right);
    }

    public int getNodeValue() {
        return nodeValue;
    }

    public void setNodeValue(int nodeValue) {
        this.nodeValue = nodeValue;
    }

    public Node getLeft() {
        return left;
    }

    public void setLeft(Node left) {
        this.left = left;
    }

    public Node getRight() {
        return right;
    }

    public void setRight(Node right) {
        this.right = right;
    }

    public Node getParent() {
        return parent;
    }

    public void setParent(Node parent) {
        this.parent = parent;
    }
}
```

---

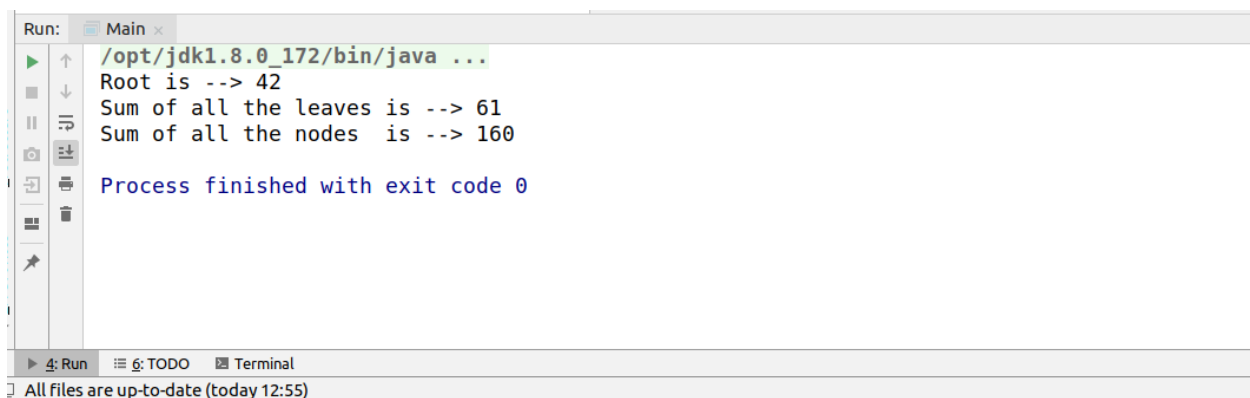
## Visitable implementation, Visitor Class:

```
//This class implements the functionality of the visitorInterface
public class Visitor<T> implements Visitable<T>
{
    //Method to find the root of the tree, from any given node
    public Node findRoot(Node node) {
        while (true) {
            if (node != null && node.getParent() != null) {
                node = findRoot(node.getParent());
            } else {
                break;
            }
        }
        return node;
    }

    //Method to find sum of all the leaves of the tree
    public int sumOfLeaves(Node node) {
        if(node != null) {
            if(node.getLeft() != null || node.getRight() != null) {
                return sumOfLeaves(node.getLeft()) + sumOfLeaves(node.getRight());
            }
            else {
                return node.getNodeValue();
            }
        }
        else {
            return 0;
        }
    }

    //Method to find sum of all the nodes of the tree
    public int sumOfNodes(Node node){
        if(node != null)
            return node.getNodeValue() + sumOfNodes(node.getLeft()) + sumOfNodes(node.getRight());
        return 0;
    }
}
```

## Output:

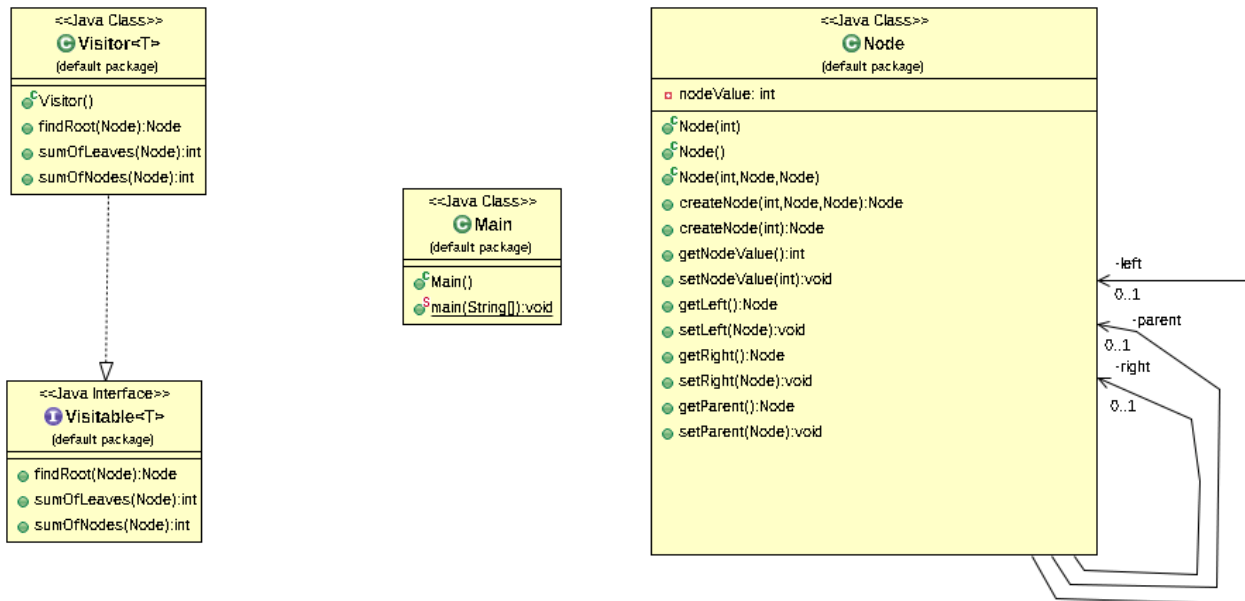


```
Run: Main x
/opt/jdk1.8.0_172/bin/java ...
Root is --> 42
Sum of all the leaves is --> 61
Sum of all the nodes is --> 160
Process finished with exit code 0
```

4: Run    TODO    Terminal

All files are up-to-date (today 12:55)

## CLASS DIAGRAM:



## After Five Entries Graph Only Horizontal Known Links:

(20)

(8) → (15)

(5) → (7)

## After Ten Entries Graph Only Horizontal Known Links:

(42)

(20) → (11)

(8) → (15) → (18)

(5) → (7) → (21) → (13)