```java
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package binarytree;

public class BinaryTree {
   int[] branch = {51, 54, 69, 79, 88, 99, 102, 113, 116, 124, 127, 132, 135};
   int[] counter = new int[200];
   public static int getLineNumber() { return
Thread.currentThread().getStackTrace()[2].getLineNumber();} //from
https://stackoverflow.com/questions/115008/how-can-we-print-line-numbers-to-
the-log-in-java
   public static float getbranch(float x, float y) { return x/y; }
   private Node root;

   public BinaryTree(){

      root = null;
    }

    public void insert(int val){
        root=insert(root, val);
    }

    public void printPostOrder(){
            printPostOrder(root);
    }

    public void printInOrder(){
            printInOrder(root);
    }

    public void printPreOrder(){
            printPreOrder(root);
    }

    public int size(){
            return size(root);
    }

    public int maxDepth(){
            return maxDepth(root);
    }

    public boolean isBST(){
            return isBST(root, Integer.MIN_VALUE, Integer.MAX_VALUE);
    }

    private  boolean isBST(Node node, int min, int max){

      counter[getLineNumber()]++; if(node == null)
```

```
        {    counter[getLineNumber()]++;return true;}

       counter[getLineNumber()]++;if(node.data<min || node.data>max)
        {   return false;}

       counter[getLineNumber()]++;return isBST(node.left, min, node.data) &&
                    isBST(node.right, node.data, max);
    }
    private Node NewNode(int val){
        Node root = new Node();
        root.data = val;
        root.left = null;
        root.right = null;
        return root;
       }
    private void printInOrder(Node node){

       counter[getLineNumber()]++; if(node == null) {
             counter[getLineNumber()]++; return;
        }
        printInOrder(node.left);
        System.out.println(node.data);
        printInOrder(node.right);
       }

    private void printPostOrder(Node node){

       counter[getLineNumber()]++; if(node == null){
            return;
        }
       counter[getLineNumber()]++;printInOrder(node.left);
        printInOrder(node.right);
        System.out.println ( node.data );
       }
    private void printPreOrder(Node node) {

       counter[getLineNumber()]++;if(node==null){
             counter[getLineNumber()]++;return;
        }
        System.out.println(node.data);
        printPreOrder(node.left);
        printPreOrder(node.right);
       }


    private int size(Node node){

       counter[getLineNumber()]++;if(node==null){
             counter[getLineNumber()]++;return 0;
        }
        else {
             counter[getLineNumber()]++;;return (size(node.left) + 1 +
size(node.right));
        }
    }
```

```
    private Node insert(Node node, int val){

       counter[getLineNumber()]++;if(node==null){
             counter[getLineNumber()]++;return NewNode(val);
        }

       counter[getLineNumber()]++;if(val <= node.data ){
             counter[getLineNumber()]++;node.left = insert(node.left, val);
        }
        else {
             counter[getLineNumber()]++;node.right = insert(node.right, val);
        }
       counter[getLineNumber()]++;
        return node;
       }
    private int maxDepth(Node node){

       counter[getLineNumber()]++;if(node==null){
             counter[getLineNumber()]++;return 0;
        }
        else
        {   counter[getLineNumber()]++;
            int leftDepth = maxDepth(node.left);
            int rightDepth = maxDepth(node.right);
            counter[getLineNumber()]++;
            if(leftDepth>rightDepth){
             counter[getLineNumber()]++;return leftDepth+1;
            }
            else {
             counter[getLineNumber()]++;return rightDepth+1;
            }
        }
    }

    public void printnumber() {
        for(int i = 0; i < counter.length; i++) {
             if(counter[i] != 0) {
                   System.out.println("Line "+ i + " executed " + counter[i]
+ " times");
             }
        }
    }
    public void printbranch() {
       for(int i = 0; i < branch.length; i++) {
             for(int j = 0; j < counter.length; j++) {
                   if(j == branch[i]) {
                         if(j == 102) System.out.println("Branch at line " +
branch[i] + " taken " + getbranch((float)counter[j+1], (float)counter[j-3]) +
"%");
                         else if(j == 116) System.out.println("Branch at line
" + branch[i] + " taken " + getbranch((float)counter[j+1], (float)counter[j-
3]) + "%");
```

```
                        else if(j == 127) System.out.println("Branch at line
" + branch[i] + " taken " + getbranch((float)counter[j+1], (float)counter[j-
3]) + "%");
                        else if(j == 132) System.out.println("Branch at line
" + branch[i] + " taken " + getbranch((float)counter[j+1], (float)counter[j-
1]) + "%");
                        else if(j == 135) System.out.println("Branch at line
" + branch[i] + " taken " + getbranch((float)counter[j+1], (float)counter[j-
4]) + "%");
                        else System.out.println("Branch at line " +
branch[i] + " taken " + getbranch((float)counter[j+1], (float)counter[j]) +
"%");

                    }
                }
            }
        }
    }
```