**Tree traversals:**  
class Main {

public static void main(String[] args) {

Main tree = new Main();

tree.root = new Node(1);

tree.root.left = new Node(2);

tree.root.right = new Node(3);

tree.root.left.left = new Node(4);

tree.root.left.right = new Node(5);

System.out.print("Inorder: ");

tree.inorder(tree.root);

System.out.print("\nPreorder: ");

tree.preorder(tree.root);

System.out.print("\nPostorder: ");

tree.postorder(tree.root);

}

Node root;

void inorder(Node node) {

if (node == null) return;

inorder(node.left);

System.out.print(node.data + " ");

inorder(node.right);

}

void preorder(Node node) {

if (node == null) return;

System.out.print(node.data + " ");

preorder(node.left);

preorder(node.right);

}

void postorder(Node node) {

if (node == null) return;

postorder(node.left);

postorder(node.right);

System.out.print(node.data + " ");

}

}

class Node {

int data;

Node left, right;

public Node(int value) {

data = value;

left = right = null;

}

}

**Count leaf nodes**

int countLeaf(Node node)

{

if(node == null) {

return 0;

}

if (node.left == null && node.right == null) {

return 1;

}

else {

return countLeaf(node.left) + countLeaf(node.right);

}

}  
**sum of ndoes**

int sum(Node node)

{

if(node==null)

{

return 0;

}

return node.data+sum(node.left)+sum(node.right);

}  
  
**height of tree**

int height(Node node) {

if (node == null) return -1;

int leftHeight = height(node.left);

int rightHeight = height(node.right);

return 1 + Math.max(leftHeight, rightHeight);

}  
  
**Maximum in the given nodes**

int maxnode(Node node)

{

if(node==null)

{

return 0;

}

int leftHeight=maxnode(node.left);

int rightHeight=maxnode(node.right);

int root=node.data;

return Math.max(root,Math.max(leftHeight,rightHeight));

}