**6. Detailed codes with Relevant comments:->**

package tree;

import java.util.Scanner;

public class binarytree {

// class node

public class node {

int data;

node left;

node right;

// constructor for creating node with given values

node(int data) {

this.data = data;

this.left = null;

this.right = null;

}

}

public static node root = null;

// function to perform a inOrder traversal of the tree

public void inOrder(node root) {

if (root == null) {

return;

}

inOrder(root.left);

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

inOrder(root.right);

}

// function to perform a preOrder traversal of the tree

public void preOrder(node root) {

if (root == null) {

return;

}

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

preOrder(root.left);

preOrder(root.right);

}

// function to perform a postOrder traversal of the tree

public void postOrder(node root) {

if (root == null) {

return;

}

postOrder(root.left);

postOrder(root.right);

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

}

// function to count no of nodes in tree

public int cutnodes(node root) {

if (root == null) {

return 0;

}

int lcout = cutnodes(root.left);

int rcout = cutnodes(root.right);

return 1 + lcout + rcout;

}

// function to calculate sum of all nodes values in tree

public int sum(node root) {

if (root == null) {

return 0;

}

int lsum = sum(root.left);

int rsum = sum(root.right);

return lsum + rsum + root.data;

}

// function to calculate height of the tree

public int height(node root) {

if (root == null) {

return 0;

}

int lheight = height(root.left);

int rheight = height(root.right);

int height = Math.max(lheight, rheight) + 1;

return height;

}

// function to calculate sum of Left leaf sum

public int rleavesum(node root) {

if (root == null) {

return 0;

} else if (root.right != null && root.right.left == null && root.right.right == null) {

return rleavesum(root.left) + root.right.data;

} else {

return rleavesum(root.left) + rleavesum(root.right);

}

}

// function to calculate sum of Right leaf sum

public int leftleavesum(node root) {

if (root == null) {

return 0;

} else if (root.left != null && root.left.left == null && root.left.right == null) {

return root.left.data + leftleavesum(root.right);

} else {

return leftleavesum(root.left) + leftleavesum(root.right);

}

}

// function to create Binary tree

public node create() {

int data;

Scanner sc = new Scanner(System.in);

node root = null;

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

data = sc.nextInt();

if (data == -1) {

return null;

}

root = new node(data);

System.out.print("Enter left child of " + root.data);

root.left = create();

System.out.print("Enter right child of " + root.data);

root.right = create();

// System.out.println("Root data is = " + root.data);

return root;

}

// Main function for choise of operation

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

binarytree tree = new tree.binarytree();

int ch;

root = null;

// for a while loop

while (true) {

// List of choise

System.out.println("1.create tree ");

System.out.println("2.inorder");

System.out.println("3.preorder");

System.out.println("4.postorder");

System.out.println("5.count no of nodes");

System.out.println("6.Sum of no of nodes");

System.out.println("7.Height of tree");

System.out.println("8.sum of left leave node");

System.out.println("9.sum of right leave node");

System.out.println("0.exit");

System.out.println("Enter choice ");

ch = sc.nextInt();

switch (ch) {

case 1:

root = tree.create();

break;

case 2:

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

tree.inOrder(root);

break;

case 3:

System.out.print("Preorder = ");

tree.preOrder(root);

break;

case 4:

System.out.print("Postorder = ");

tree.postOrder(root);

break;

case 5:

int count = tree.cutnodes(root);

System.out.println("Count is = " + count);

break;

case 6:

int sum = tree.sum(root);

System.out.println("Sum is = " + sum);

break;

case 7:

int heightt = tree.height(root);

System.out.println("Height is = " + heightt);

break;

case 8:

int lsum = tree.leftleavesum(root);

System.out.println("Left leaf sum is " + lsum);

break;

case 9:

int rsum = tree.rleavesum(root);

System.out.println("Right leaf sum is " + rsum);

break;

case 0:

System.exit(0);

break;

} } } }

**7.Results:->**

1.create tree

2.inorder

3.preorder

4.postorder

5.count no of nodes

6.Sum of no of nodes

7.Height of tree

8.sum of left leave node

9.sum of right leave node

0.exit

Enter choice

1

Enter data 5

Enter left child of 5 Enter data 4

Enter left child of 4 Enter data 11

Enter left child of 11 Enter data -1

Enter right child of 11 Enter data -1

Enter right child of 4 Enter data 10

Enter left child of 10 Enter data -1

Enter right child of 10 Enter data -1

Enter right child of 5 Enter data 6

Enter left child of 6 Enter data -1

Enter right child of 6 Enter data 25

Enter left child of 25 Enter data -1

Enter right child of 25 Enter data -1

Inorder = 11 4 10 5 6 25

Preorder = 5 4 11 10 6 25

Postorder = 11 10 4 25 6 5

Count is = 6

Sum is = 61

Height is = 3

Left leaf sum is 11

Right leaf sum is 35

Process finished with exit code 0