

The screenshot shows the NetBeans IDE with a single file named `BinaryTreeTraversals.java` open. The code defines a `Node` class and a `BinaryTree` class. The `BinaryTree` class contains three methods: `inOrder`, `preOrder`, and `postOrder`, each taking a `Node` and a `List<Integer>` as parameters. The `inOrder` method calls `inOrder(n.left, out)`, `out.add(n.data)`, and `inOrder(n.right, out)`. The `preOrder` method calls `out.add(n.data)`, `preOrder(n.left, out)`, and `preOrder(n.right, out)`. The `postOrder` method calls `postOrder(n.left, out)`, `postOrder(n.right, out)`, and `out.add(n.data)`. The IDE interface includes a menu bar, a toolbar, and a status bar at the bottom showing the temperature as 29°C and the time as 6:39 am on 26/09/2023.

```
1 package binarytree.traversals;
2 import java.util.*;
3
4 class Node {
5     int data;
6     Node left;
7     Node right;
8
9     public Node(int data) {
10         this.data = data;
11     }
12 }
13
14 class BinaryTree {
15     Node root;
16
17     public void inOrder(Node n, List<Integer> out) {
18         if (n == null)
19             return;
20         inOrder(n.left, out);
21         out.add(n.data);
22         inOrder(n.right, out);
23     }
24
25     public void preOrder(Node n, List<Integer> out) {
26         if (n == null)
27             return;
28         out.add(n.data);
29         preOrder(n.left, out);
30         preOrder(n.right, out);
31     }
32
33     public void postOrder(Node n, List<Integer> out) {
34         if (n == null)
35             return;
36         postOrder(n.left, out);
37         postOrder(n.right, out);
38         out.add(n.data);
39     }
40
41     public List<Integer> levelOrder(Node n) {
42         List<Integer> out = new ArrayList<>();
43
44         if (n == null)
45             return out;
46
47         Queue<Node> q = new ArrayDeque<>();
48         q.add(n);
49
50         while (!q.isEmpty()) {
51             Node cur = q.remove();
52             out.add(cur.data);
53             if (cur.left != null)
54                 q.add(cur.left);
55             if (cur.right != null)
56                 q.add(cur.right);
57         }
58
59         return out;
60     }
61 }
62
63 public class BinaryTreeTraversals {
64     public static void main(String[] args) {
65         BinaryTree t = new BinaryTree();
66
67         t.root = new Node(10);
68         t.root.left = new Node(5);
69     }
70 }
```

The screenshot shows the NetBeans IDE with the same file `BinaryTreeTraversals.java` open. The code continues from the previous screenshot, adding a `levelOrder` method and a `main` method. The `levelOrder` method uses a queue to traverse the tree level by level. The `main` method creates a `BinaryTree` object, sets its root to a node with value 10, and sets the left child to a node with value 5. The IDE interface includes a menu bar, a toolbar, and a status bar at the bottom showing a flood warning and the time as 6:39 am on 26/09/2023.

```
37     postOrder(n.left, out);
38     postOrder(n.right, out);
39     out.add(n.data);
40 }
41
42 public List<Integer> levelOrder(Node n) {
43     List<Integer> out = new ArrayList<>();
44
45     if (n == null)
46         return out;
47
48     Queue<Node> q = new ArrayDeque<>();
49     q.add(n);
50
51     while (!q.isEmpty()) {
52         Node cur = q.remove();
53         out.add(cur.data);
54         if (cur.left != null)
55             q.add(cur.left);
56         if (cur.right != null)
57             q.add(cur.right);
58     }
59
60     return out;
61 }
62
63 public class BinaryTreeTraversals {
64     public static void main(String[] args) {
65         BinaryTree t = new BinaryTree();
66
67         t.root = new Node(10);
68         t.root.left = new Node(5);
69     }
70 }
```

