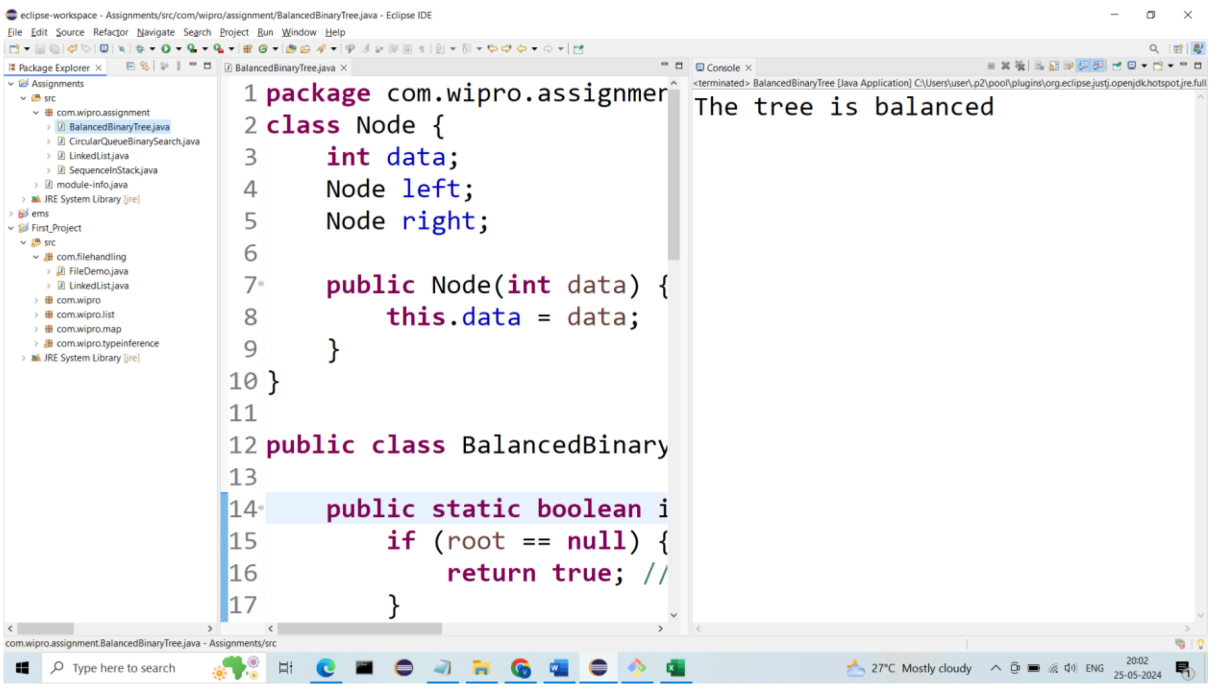
Task 1: Balanced Binary Tree Check

Write a function to check if a given binary tree is balanced. A balanced tree is one where the height of two subtrees of any node never differs by more than one.

```
package com.wipro.assignment;
class Node {
    int data;
    Node left;
    Node right;

    public Node(int data) {
        this.data = data;
    }
}

public class BalancedBinaryTree {

    public static boolean isBalanced(Node 
root) {
        if (root == null) {
            return true; // Empty tree is 
considered balanced
        }
        int leftHeight = 
getHeight(root.left);
        int rightHeight = 
getHeight(root.right);

        int heightDiff = 
Math.abs(leftHeight - rightHeight);

        return heightDiff <= 1 && 
isBalanced(root.left) && 
isBalanced(root.right);
    }

    private static int getHeight(Node 
root) {
        if (root == null) {
            return 0; // Height of an empty 
tree is 0
        }

        int leftHeight = 
getHeight(root.left);
        int rightHeight = 
getHeight(root.right);

        return Math.max(leftHeight, 
rightHeight) + 1; // Height is 1 more 
than the max of left and right subtrees
    }
    public static void main(String[]
args) {
        Node root = new Node(1);
        root.left = new Node(2);
        root.right = new Node(3);
        root.left.left = new Node(4);
        root.right.right = new Node(5);

        if (isBalanced(root)) {
            System.out.println("The tree is
balanced");
        } else {
            System.out.println("The tree is
not balanced");
        }
    }
}

Output: -
```



```java
package com.wipro.assignmen
class Node {
    int data;
    Node left;
    Node right;

    public Node(int data) {
        this.data = data;
    }
}

public class BalancedBinary

    public static boolean i
        if (root == null) {
            return true; //
        }
```

```
<terminated> BalancedBinaryTree [Java Application] C:\Users\user\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full
The tree is balanced
```

Explanation:

1. **Node Class:** Defines the basic structure of a node in the binary tree with data and pointers to left and right children.
2. **isBalanced Function:**
   - Takes the root node of the tree as input.
   - Base case: If the tree is empty (root is null), it's considered balanced.
   - Calculates the heights of the left and right subtrees using the getHeight function.
   - Calculates the absolute difference in heights.
   - Returns true if the height difference is less than or equal to 1 and both left and right subtrees are balanced (recursive calls).
3. **getHeight Function:**
   - Takes a node as input.

   - Base case: If the node is null, its height is 0 (empty subtree).
   - Recursively calculates the heights of the left and right subtrees.
   - Returns the maximum of the left and right subtree heights plus 1 (current node's height).
4. **Main Method:**
   - Creates a sample binary tree.
   - Calls the isBalanced function to check if the tree is balanced.
   - Prints the result.