Program:

```
class TreeNode {
    int val;
    TreeNode left, right;
    TreeNode(int x) { val = x; }
}
public class LongestUnivaluePath {
    int max = 0;
    public int longestUnivaluePath(TreeNode root) {
        dfs(root);
        return max;
    }
    private int dfs(TreeNode node) {
        if (node == null) return 0;
        int left = dfs(node.left);
        int right = dfs(node.right);
        int leftPath = 0, rightPath = 0;
        if (node.left != null && node.left.val == node.val)
            leftPath = left + 1;
        if (node.right != null && node.right.val == node.val)
            rightPath = right + 1;
        max = Math.max(max, leftPath + rightPath);
        return Math.max(leftPath, rightPath);
    }
    public static void main(String[] args) {
        TreeNode root = new TreeNode(5);
        root.left = new TreeNode(4);
        root.right = new TreeNode(5);
        root.left.left = new TreeNode(1);
        root.left.right = new TreeNode(1);
        root.right.right = new TreeNode(5);

        LongestUnivaluePath obj = new LongestUnivaluePath();
        System.out.println("Longest Univalue Path: " + obj.longestUnivaluePath(root));
    }
}
```

Output:

```
Longest Univalue Path: 2
```

Program:

```
class BinaryTreePaths {
    static class Node {
        int data;
        Node left, right;
        Node(int item) { data = item; }
    }
    Node root;
    int countPaths(Node node) {
        if (node == null) return 0;
        if (node.left == null && node.right == null) return 1;
        return countPaths(node.left) + countPaths(node.right);
    }
    public static void main(String[] args) {
        BinaryTreePaths tree = new BinaryTreePaths();
        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.println("Number of paths: " + tree.countPaths(tree.root));
    }
}
```

Output:

```
Number of paths: 3
```

Program:

```
import java.util.*;
class LevelOrderTraversal {
   static class Node {
      int data;
      Node left, right;
      Node(int item) { data = item; }
   }
   Node root;
   void levelOrder(Node node) {
      if (node == null) return;
      Queue<Node> queue = new LinkedList<>();
      queue.add(node);
      while (!queue.isEmpty()) {
         Node temp = queue.poll();
         System.out.print(temp.data + " ");
         if (temp.left != null)
            queue.add(temp.left);
         if (temp.right != null)
            queue.add(temp.right);
      }
   }
   public static void main(String[] args) {
      LevelOrderTraversal tree = new LevelOrderTraversal();
      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("Level Order Traversal: ");
        tree.levelOrder(tree.root);
    }
}
```

Output:

```
Level Order Traversal: 1 2 3 4 5
```