```java
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

import java.util.*;

class Program {
  // Average case: when the tree is balanced
  // O(nlog(n)) time | O(h) space - where n is the number of nodes in
  // the Binary Tree and h is the height of the Binary Tree
  public static int allKindsOfNodeDepths(BinaryTree root) {
    int sumOfAllDepths = 0;
    List<BinaryTree> stack = new ArrayList<BinaryTree>();
    stack.add(root);
    while (stack.size() > 0) {
      BinaryTree node = stack.remove(stack.size() - 1);
      if (node == null) continue;
      sumOfAllDepths += nodeDepths(node, 0);
      stack.add(node.left);
      stack.add(node.right);
    }
    return sumOfAllDepths;
  }

  public static int nodeDepths(BinaryTree node, int depth) {
    if (node == null) return 0;
    return depth + nodeDepths(node.left, depth + 1) + nodeDepths(node.right, depth + 1);
  }

  static class BinaryTree {
    int value;
    BinaryTree left;
    BinaryTree right;

    public BinaryTree(int value) {
      this.value = value;
      left = null;
      right = null;
    }
  }
}

```