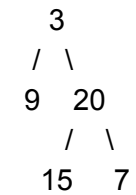
1**. Balanced Binary Tree**

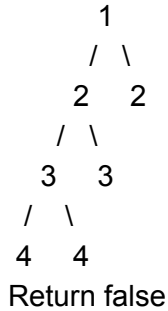
Given a binary tree, determine if it is height-balanced. For this problem, a height-balanced binary tree is defined as: a binary tree in which the depth of the two subtrees of every node never differ by more than 1.

Example 1: Given the following tree [3,9,20,null,null,15,7]:




Return true.

Example 2: Given the following tree [1,2,2,3,3,null,null,4,4]:




Return false.

Implement and test your code in the BinaryTree.java as flowing:

```
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class BinaryTree {
	public Node root;

	public static class Node {
		public int key;
		public Node left, right;

		public Node(int item) {
			key = item;
			left = right = null;
		}
	}
	public boolean isBalanced(Node root) {
		// place your code here
```

```
            return true;
      }
      public boolean isBalanced() {
            return isBalanced(root);
      }
      public static void main(String args[]) {
            BinaryTree tree1 = new BinaryTree();
            tree1.root = new Node(3);
            tree1.root.left = new Node(9);
            tree1.root.right = new Node(20);
            tree1.root.right.left = new Node(15);
            tree1.root.right.right = new Node(7);

            BinaryTree tree2 = new BinaryTree();
            tree2.root = new Node(1);
            tree2.root.left = new Node(2);
            tree2.root.right = new Node(2);
            tree2.root.left.left = new Node(3);
            tree2.root.left.right = new Node(3);
            tree2.root.left.left.left = new Node(4);
            tree2.root.left.left.right = new Node(4);

            System.out.println(tree1.isBalanced());
            System.out.println(tree2.isBalanced());
      }
}
```

The expected output of the code is as follows:

```
true
false
```