```java
/*Author: Bochen (mddboc@foxmail.com)
Last Modified: Tue Apr 10 22:28:44 CST 2018*/

/*Given a binary tree, check whether it is a mirror of itself (ie, symmetric around its center).

        For example, this binary tree [1,2,2,3,4,4,3] is symmetric:

        1
        / \
        2   2
        / \ / \
        3  4 4  3
        But the following [1,2,2,null,3,null,3] is not:
        1
        / \
        2   2
        \   \
        3    3
        Note:
        Bonus points if you could solve it both recursively and iteratively.*/


import java.util.*;
import java.lang.Math;
import java.lang.System;
import java.lang.Integer;


public class Main {

    public static void main(String[] args) throws ArithmeticException {

        TreeNode root = new TreeNode(1);
        root.left = new TreeNode(2);
        root.right = new TreeNode(2);
        root.left.left = new TreeNode(3);
        root.left.right = new TreeNode(4);
        root.right.left = new TreeNode(4);
        root.right.right = new TreeNode(3);

        boolean result = new Solution().isSymmetric(root);

        System.out.println(result);
    }

}


class ListNode {
    int val;
    ListNode next;

    ListNode(int x) {
        val = x;
    }
}


class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int x) {
        val = x;
    }
}


class Solution {
    public boolean isSymmetric(TreeNode root) {

        if (root == null) {
            return true;
        }

        return isSymmetricHelper(root.left, root.right);
    }

    private boolean isSymmetricHelper(TreeNode left, TreeNode right) {

        if (left == null || right == null) {
            return left == right;
        }

        if (left.val != right.val) {
            return false;
        }

        return isSymmetricHelper(left.left, right.right) && 
        isSymmetricHelper(left.right,right.left);
    }
}
```