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

/* Design a stack that supports push, pop, top, and retrieving the minimum element
in constant time.

push(x) -- Push element x onto stack.
pop() -- Removes the element on top of the stack.
top() -- Get the top element.
getMin() -- Retrieve the minimum element in the stack.
Example:
MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin();   --> Returns -3.
minStack.pop();
minStack.top();      --> Returns 0.
minStack.getMin();   --> Returns -2. */

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 MinStack {

    /**
     * initialize your data structure here.
     */
    private List<Integer> stack = new ArrayList<>();
    private List<Integer> minValue = new ArrayList<>();
    private int stackIndex = -1;
    private int minIndex = -1;


    public MinStack() {

    }

    public void push(int x) {
        if ( stackIndex == -1 ) {
            minValue.add(x);
            minIndex++;
        } else if (x <= minValue.get(minIndex)) {
            minValue.add(x);
            minIndex++;
        }

        stack.add(x);
        stackIndex++;
    }

    public void pop() {
        int temp = stack.get(stackIndex);
        if (minValue.get(minIndex) == temp) {
            minValue.remove(minIndex);
            minIndex--;
        }
        stack.remove(stackIndex);
        stackIndex--;
    }

    public int top() {
        return stack.get(stackIndex);
    }

    public int getMin() {
        return minValue.get(minIndex);
    }
}
```