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

/*
Given a binary tree, return the bottom-up level order traversal of its nodes' values. (ie, from left to right, level by level from leaf to root).

For example:
Given binary tree [3,9,20,null,null,15,7],
    3
   / \
  9  20
    /  \
   15   7
return its bottom-up level order traversal as:
[
  [15,7],
  [9,20],
  [3]
] */


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


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

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

public class Test {
    public static void main(String[] args) {

        TreeNode root = new TreeNode(3);
        root.left = new TreeNode(4);
        root.right = new TreeNode(5);

        new Solution().levelOrderBottom(root);
    }
}

class Solution {
    public List<List<Integer>> levelOrderBottom(TreeNode root) {

        LinkedList<List<Integer>> result = new LinkedList<>();

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

        Queue<TreeNode> queue = new LinkedList<>();

        queue.add(root);

        while (!queue.isEmpty()) {
            List<Integer> tempResult = new LinkedList<>();
            int size = queue.size();
            for (int i = 0; i < size; i++) {
                TreeNode currentNode = queue.poll();
                tempResult.add(currentNode.val);
                if (currentNode.left != null) {
                    queue.add(currentNode.left);
                }
                if (currentNode.right != null) {
                    queue.add(currentNode.right);
```

```
73                  }
74              }
75  ->  ->  ->result.addFirst(tempResult);
76          }
77
78  ->  ->return result;
79      }
80  }
```