67. Binary Tree Inorder Traversal

* [Description](http://lintcode.com/en/problem/binary-tree-inorder-traversal/" \l "description)
* [Notes](http://lintcode.com/en/problem/binary-tree-inorder-traversal/#note)
* [Testcase](http://lintcode.com/en/problem/binary-tree-inorder-traversal/#testcase)
* [Judge](http://lintcode.com/en/problem/binary-tree-inorder-traversal/#judge)

Given a binary tree, return the *inorder* traversal of its nodes' values.

Have you met this question in a real interview?

Yes

**Example**

Given binary tree {1,#,2,3},

1

\

2

/

3

return [1,3,2].

<http://lintcode.com/en/problem/binary-tree-inorder-traversal/#>

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package javaapplication1;

import java.util.\*;

/\*\*

\*

\* @author Usuario

\*/

class TreeNode {

public int val;

public TreeNode left, right;

public TreeNode(int val) {

this.val = val;

this.left = this.right = null;

}

}

public class JavaApplication1 {

/\*\*

\*

\* @param root

\* @return

\*/

static List<Integer> inorderTraversal(TreeNode root) {

// write your code here

List<Integer> lista = new ArrayList();

if (root == null) {

return new ArrayList();

}

//keep the nodes in the path that are waiting to be visited

Stack<TreeNode> stack = new Stack();

TreeNode node = root;

//first node to be visited will be the left one

while (node != null) {

stack.push(node);

node = node.left;

}

// traverse the tree

while (stack.size() > 0) {

// visit the top node

node = stack.pop();

//System.out.print(node.val + " ");

lista.add(node.val);

if (node.right != null) {

node = node.right;

// the next node to be visited is the leftmost

while (node != null) {

stack.push(node);

node = node.left;

}

}

}

return lista;

}

public static void main(String[] args) {

// TODO code application logic here

TreeNode tree = new TreeNode(1);

tree = new TreeNode(1);

tree.left = new TreeNode(2);

tree.right = new TreeNode(3);

tree.left.left = new TreeNode(4);

tree.left.right = new TreeNode(5);

for(int elem : inorderTraversal(tree)) {

System.out.print(elem + " ");

}

}

}