No Fancy Algorithm, just Simple and Powerful In-Order Traversal

This question appeared difficult to me but it is really just a simple in-order traversal! I got really frustrated when other people are showing off Morris Traversal which is totally not necessary here.

Let's start by writing the in order traversal:

**private** **void** **traverse** (TreeNode root) {

**if** (root == **null**)

**return**;

traverse(root.left);

// Do some business

traverse(root.right);

}

So when we need to print the node values in order, we insert System.out.println(root.val) in the place of "Do some business".

What is the business we are doing here?  
We need to find the first and second elements that are not in order right?

How do we find these two elements? For example, we have the following tree that is printed as in order traversal:

6, 3, 4, 5, 2

We compare each node with its next one and we can find out that 6 is the first element to swap because 6 > 3 and 2 is the second element to swap because 2 < 5.

Really, what we are comparing is the current node and its previous node in the "in order traversal".

Let us define three variables, firstElement, secondElement, and prevElement. Now we just need to build the "do some business" logic as finding the two elements. See the code below:

**public** **class** Solution {

TreeNode firstElement = **null**;

TreeNode secondElement = **null**;

// The reason for this initialization is to avoid null pointer exception in the first comparison when prevElement has not been initialized

TreeNode prevElement = **new** TreeNode(Integer.MIN\_VALUE);

**public** **void** recoverTree(TreeNode root) {

// In order traversal to find the two elements

traverse(root);

// Swap the values of the two nodes

int temp = firstElement.**val**;

firstElement.**val** = secondElement.**val**;

secondElement.**val** = temp;

}

**private** **void** traverse(TreeNode root) {

**if** (root == **null**)

**return**;

traverse(root.left);

// Start of "do some business",

// If first element has not been found, assign it to prevElement (refer to 6 in the example above)

**if** (firstElement == **null** && prevElement.**val** >= root.**val**) {

firstElement = prevElement;

}

// If first element is found, assign the second element to the root (refer to 2 in the example above)

**if** (firstElement != **null** && prevElement.**val** >= root.**val**) {

secondElement = root;

}

prevElement = root;

// End of "do some business"

traverse(root.right);

}

And we are done, it is just that easy!