AVL Tree

You are given classes for AVL tree. With the following method in class AVLTree that can help you add new data to the tree without rotation (class AVLNode, AVLTreeIterator, BTreePrinter are also available).

`public AVLNode insertNoBalance(int v)`

This method inserts data into AVL tree just like in ordinary binary search tree (no rotation). The method updates the height of all the nodes and the size of the tree correctly.

Write the following methods for class AVLTree

`public static boolean` `same(AVLTree t1, AVLTree t2):` this utility method compares two trees, t1 and t2. It returns true if both trees have exactly the same structure and all data in both trees are the same too (data are stored in exactly the same way). This method must work even though the trees do not really have the height properties of AVL.

`public boolean` `isAVL():` this method returns true if our tree is really an AVL tree (assume that it is already a binary search tree). That means, if root is null, then it is true. If the root is not null, its tilt degree must be between -1 and 1, and its left and right subtrees must also be AVL.

`public void` `makeAVL()` **`throws`** `Exception:` this method changes our tree so that isAVL() returns true.

JUnit test cases are provided for each method.

How to submit:

Submit the jar file of your project (the jar file must include all your java files and test cases) to Courseville, name it **YourID_Lab09_AVL** where YourID is your student ID).