Homework Turnin

Account: 6G_06 (rgalanos@fcps.edu)
Section: 6G
Course: TJHSST APCS 2016-17
Assignment: 07-21

Receipt ID: 128160f7d109797d405df69582edb975

Turnin Successful!

The following file(s) were received:

CL_TreeToString.java (3836 bytes)

```java
//name: Kazuya Chue  date: 3/3/17

/* Write a method treeToString. The method returns "empty" for an empty
tree. For a leaf node, it should return the data in the node as a
String. For a branch node, it should return a parenthesized String
containing (data, its left child, its right child).  For example, a
binary search tree made from the letters "BACD" would return
(B, A, (C, empty, D)).   For large trees, the parenthesized strings
will be nested.

The expected output for two given strings and an empty tree has been
pasted at the bottom of this code.
*/
public class CL_TreeToString
{
   public static String a = "BCDA";
   public static String c = "COMPUTER";

   public static void main(String[] args)
   {
      TreeNode root = null;
      root = buildTree( root, a );
      System.out.print( display( root, 0) );
      System.out.println("\n" + treeToString(root));
      System.out.println("-----------------------------");
      root = null;    //start all over
      root = buildTree( root, c );
      System.out.print( display( root, 0) );
      System.out.println("\n" + treeToString(root));
      System.out.println("-----------------------------");
      root = null;    //start all over
      System.out.print( display( root, 0) );
      System.out.println("\n" + treeToString(root));
   }
   public static TreeNode buildTree(TreeNode t, String s)
   {
      for(int k = 0; k < s.length(); k++)
         t = insert(t, "" + s.charAt(k));
      return t;
   }
   public static TreeNode insert(TreeNode t, String s)
   {
      if(t == null)
         return new TreeNode(s, null, null);
      else
      {
         if(s.compareTo(""+t.getValue()) <= 0)
            t.setLeft(insert(t.getLeft(), s));
         else
            t.setRight(insert(t.getRight(), s));
      }
      return t;
   }

   public static String display(TreeNode t, int level)
   {
      String toRet = "";
      if(t == null)
         return "";
      toRet += display(t.getRight(), level + 1);
      for(int k = 0; k < level; k++)
         toRet += "\t";
      toRet += t.getValue() + "\n";
      toRet += display(t.getLeft(), level + 1);
      return toRet;
   }

   private static String treeToString(TreeNode root)
   {
      if(root == null)
         return "empty";
      else if(root.getLeft()==null&&root.getRight()==null)
         return root.getValue()+"";
      else
      {
         return "(" + root.getValue() + ", " + treeToString(root.getLeft()) + ", " + treeToString(root.getRight(
      }
   }

}
/**************************************************

    ----jGRASP exec: java CL_TreeToString_teacher

      D
   C
 B
   A

 (B, A, (C, empty, D))
 -----------------------------
         U
            T
               R
      P
   O
      M
         E
 C

 (C, empty, (O, (M, E, empty), (P, empty, (U, (T, R, empty), empty))))
 -----------------------------

 empty

    ----jGRASP: operation complete.
  ******************************************************/

class TreeNode
{
   private Object value;
   private TreeNode left, right;

   public TreeNode(Object initValue)
   {
      value = initValue;
      left = null;
      right = null;
   }

   public TreeNode(Object initValue, TreeNode initLeft, TreeNode initRight)
   {
      value = initValue;
      left = initLeft;
      right = initRight;
   }

   public Object getValue()
   {
      return value;
   }

   public TreeNode getLeft()
   {
      return left;
   }

   public TreeNode getRight()
   {
      return right;
   }

   public void setValue(Object theNewValue)
   {
      value = theNewValue;
   }

   public void setLeft(TreeNode theNewLeft)
   {
      left = theNewLeft;
   }

   public void setRight(TreeNode theNewRight)
   {
      right = theNewRight;
   }
}
```