CS1027A - Assignment 4
Due: Sunday December 3rd 11:55pm
Weight: 8%

Overview

This assignment will take a hierarchical source of data (Data class) which represents files and folders on your computer, and create a tree data structure to be printed out.

You will use recursion, iterators, comparators, ordered lists, and a generic tree structure.

Follow the guided instructions carefully and test each phase using a test harness or other debugging means!

You will be marked on functionality, good coding style, and implementation of the required elements.

You are encouraged to use any lecture, lab or sample code from the course site.

Do NOT wait to start until the last minute! You know enough to do parts 1 and 2 now, and we will learn about trees this week.

Part 1. Familiarize yourself with the Data.java class.

Run the test harness. This will allow you to interact with the program through the console. When you are running the harnesss through eclipse, click on the console window to enter your choice. Q will exit the program, or else enter the number of the file or folder display to navigate to it. You will use this class to get data for your application. You can set the constants found in this class to limit the amount of data you will be working with. Visually compare the output of this program with your program to unsure yours is working correctly - some systems do not display hidden files/directories, so you must use this harness to verify your output.

Part 2. OrderedList and Iterators.

Create a simplified OrderedList. It can be Linked or Array based. You do NOT need to implement an ADT, but you must make the class templated. You may add any methods, BUT you will only NEED the following methods to be implemented:

Constructor

```
public boolean isEmpty()
public void insert(T element)
public Iterator<T> iteratorAscending()
public Iterator<T> iteratorDescending()
```

Create the 2 required iterator classes.

Test your OrderedList functionality with any class type that implements the Comparable<T> interface!!!

Part 3. The Tree.

We will not be creating a Tree wrapper class, instead we will consider our tree to be a reference to the root node: similar to how a Linked List may simply be a refernce to the head node. Each node (TreeNode<T>) of the tree will have a reference to it's parent (null in the case of the root node). It will have an OrderedList of child nodes. Since our child nodes will be stored in an OrderedList, we must make sure to implement the Comparable<T> interface and implement the compareTo function. It will also have a name (String type). Our tree node will have a templated data type in which we can store information associated with the node.

The following functions are sufficient for the assignment, but you may implement more at your own risk:

```
public TreeNode(TreeNode<T> parent, T data, String name)
public TreeNode<T> getParent()
public T getData()
public String getName()
public addChild(TreeNode<T> child)
* Some form of print function to output the required assignment data.
      recommend: public void print(int depth)
```

4. SystemPrint class which contains 2 functions, a main and a recursive function.

4a. main function algorithm

```
      * Create source Data object
      * Create an OrderedList of names
      * For each ChildData item
            * Add the ChildData name to the OrderedList of names
      * Create the root node of your tree with the previous information
      * Call the recurse function with the Data object and the root node
      * Print the data
            * The depth of the data element should equate to the number of spaces
                displayed before the item symbol
            * Individual data elements (files) should be preceded with the # symbol
            * Leaf node names should be preceded with the - symbol
            * Internal node should be preceded with the + symbol
            * Data elements (files) must be listed in ascending order
                (String.compareTo)
            * Node elements (directories) must be listed in descending name order
```

4b. recurse function algorithms

```
      Input: Data container item (dir) and corresponding node (dirNode)
      * For each ChildContainer item in dir
            * Create an OrderedList of names
            * For each ChildData item of the ChildContainer
                * Add the ChildData name to the OrderedList of names
            * Create the a node of your tree with the previous information
            * Add the node to it's parent (dirNode)
            * Call the recurse function with the ChildContainer and newly created
                node
```

Sample output:

```
+Root

 +C:\

  #hiberfil.sys

  #bootmgr

  #BOOTNXT

  +$GetCurrent

   -Logs

    #oobe_2016_09_10_02_10_18_844.log

    #downlevel_2016_09_09_20_47_54_620.log

    #PartnerSetupCompleteResult.log
```

```
  -SafeOS
   #PartnerSetupComplete.cmd
   #GetCurrentRollback.ini
   #GetCurrentOOBE.dll
 +$Recycle.Bin
  -S-1-5-18
  -S-1-5-21-1378745871-3090889480-289890799-500
  -S-1-5-21-3295299239-1531540638-1015378035-1002
   #$I204CFN.zip
   #$I1UCMPN.csv
   #$I09Q7WN.cpp
 +$Windows.~WS
  -Sources
+D:\
 #bootmgr.efi
 #bootmgr
 #RP.ini
 +$RECYCLE.BIN
  -S-1-5-21-3295299239-1531540638-1015378035-1002
   #desktop.ini
 +EFI
  #Desktop.ini
  -Boot
   #bootx64.efi
  -Microsoft
 +boot
  #boot.sdi
  #bcd.LOG
  #BCD
  -bg-bg
   #bootmgr.exe.mui
  -cs-cz
   #memtest.exe.mui
   #bootmgr.exe.mui
  -da-dk
```

```
#memtest.exe.mui

#bootmgr.exe.mui
```

Code similarities are both violations of Academic Dishonesty for those sharing their code as for those using other's code. As was explained in class, never use or share code! If you wish to help another student, explain concepts in English, draw diagrams, or refer the student to specific reference material.

Similarity detection software will be used on all assignments. Any students who achieve a high score for improbable code similarities will receive a mark of 0. In addition, further Academic Dishonesty reporting may be performed.

Classes to Submit: **Do NOT zip your files, or submit any additional files! You will lose marks!**

- `SystemPrint.java`
- `OrderedList.java (Array or Linked)`
- `TreeNode.java`
- `Both of your Iterator.java classes`