COP 3540 – Data Structures with OOP

Project 4 – Spring 2007

Due: Wednesday 4 April 2007

Binary Trees

Using NetBeans 5.5, you are to write a Java program using OOP principles to accommodate the following functionality

Assignment #4

Objectives:

Provide student with additional experiences with file input output.
Provide student with expanded exercises in UML
Provide student with exercises in Javadoc and its various formats
Provide student with exercises in building a binary tree with object nodes
Provide student opportunity to display binary trees in NLR and LNR traversals
Provide student with experience in inserting, deleting, and changing node contents of binary trees.
Provide student the use of iterators, **which must be used to facilitate good program design.**.

Functionality:

Task 1: Build an array of State objects.

Given a sequential file, States.Spring2007, on my web page, you are to build an array of objects only from input strings with region numbers 5 and 6. As we build the binary trees ahead, we will not distinguish between states from each of these regions. I am only getting you a 'critical mass' of state objects. **Design: You are to have a State class and a State container class that contains an array of States. This State container class (you may call it what you wish) is to contain the methods that operate on the State objects. You may keep your I/O separate from the Container class, if you wish. I recommend (not a mandate) that you put all of your I/O in a separate class (opening the file, reading the strings, converting to objects, and passing objects to the array of States, etc. But this choice is yours.**

Be certain to limit your use of static methods. I do not want to see more than two or three in this entire program, and they should be in a Main class. Yes, you are to have a Main class with a main() method within it.

Please note that I am dictating a design. Most of you already do this, but a few do not. So it is time to get on-board.

Task 2: Build a binary tree from the array of State objects.

Each node, however, is to contain not only the state name, but the population of that state. None of the other attributes are needed in the nodes of the binary tree. The binary tree is to be build in the same order as the states appear in your array.

Task 3: Display the Binary Tree. You are to display the tree using an NLR iterative scan.

You are to display the binary tree. The format is to contain a header, left justified on the print line that contains the entries: Iterative NLR Scan followed by a second header containing: Node # State Name Population with ascending node numbers, the state name and its population underneath this header. There is to be a blank line prior to the first header and all subsequent print lines are to be single spaced hereafter.

Task 4: Display the Binary Tree. You are to display the tree using a recursive NLR scan.

Same formats as above. But your first header says: Recursive NRL Scan.

Task 5: Using the original input file, States.Spring2007, you are to create additional state objects from region 1. These six additional state objects are to **extend** the original array you built in Task 1.

Task 6: Re-Build the binary tree from the extended array of States. (You may build a separate tree…)

Using appropriate headers for each of the lists below, you are to display the resulting binary tree using either iterative techniques or recursive techniques. Be certain your header label describes the scan approach you have elected. Make your outputs clearly distinguishable via the headers as needed.

Task 7: NLR format

Task 8: LNR format

Task 9: RNL format

Deliverable: Your zipped folder to me MUST include copies of your input file. I will need these to run your program and to test it. Do not provide me with output your program generates. I will get your program to generate your outputs. As noted, your outputs will be displayed on the screen.

You are to zip all files in your P4 as expected and Send them to be via Digital Dropbox using the same naming conventions as in P0. Your zip file is NOT to include your N-number. Rather, it must be project4.youruserid, as project4broggio NOT project4n00010109. **Be certain your zipped file runs! Unzip it and run it locally before you send it to me. If it does not work, I cannot grade it.**

Grading

Source Code – 10 points

Indentation
Internal comments
Scope terminators

Program Design – 20 points

This is hot for this deliverable. Be certain to have a good, solid, defensible design. We have talked over and over about this. See design guidance above. I will be looking for a Main class and a state container class containing the state array and its associated data manipulating methods. The scans **must appear in this container class!**

Javadoc – 10 points - no excuse to not have this wonderful this time!

Appropriateness and completeness of comments
ALL methods must have Javadoc comments up front that are meaningful, please.
A couple of you still do not even turn this in!

UML – 10 points – no excuse to not have this wonderful at this time too.

You were presented examples of 'the right stuff.'
Correctness, associations, completeness. This means that the classes you identify are correct, that associations are indicated, and that the attributes and methods are documented within the classes. Get the visibility indicators right. If no visibility indicators are shown, then you will receive **no points** for your UML efforts.

Outputs – 50 points

Accuracy and **Format. All functionality present!**
Skip lines in between displayed outputs as described above. .
Include headers / descriptors as described.

➔ If your program does not contain all the functionality required, you cannot expect a grade above 60, that is a 'passing grade.'
Please note that this will be rigidly enforced. Program must be on time for full credit; one class late results in 25 point reduction in your grade. Over one class late results in a zero. You must, however, submit all programs that run to ultimately pass this course.

Start early and do this a little at a time.
This is a fun program if you do it a little bit at a time.
Remember to iterate! Good luck and have fun!!!