COP 3540 – Data Structures with OOP

Project 4 – Spring 2010

Due: Tuesday, April 6th, 2010

Binary Trees

Using NetBeans 6.7.15, 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.
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.

Functionality:

Task 1: Build an array of Country objects.

Given a sequential file, Countries.txt, on my web page, you are to build an array of objects only from input strings with region numbers 1 and 2. As we build the binary trees ahead, we will not distinguish between countries from each of these regions, but you are to build the array and binary tree in the order that these strings are read in from the input file. **Design:** You are to have a country class and a some kind of container or collection class that contains an array of countries (an aggregate for your UML). This class (you may call it what you wish) is to contain the methods that operate on the country objects. You should include your file I/O in a separate class (although I will somewhat reluctantly accept static methods in Main() for file access). I suggest: public class FileInterface. The code in an object of this class should deal with opening the file, reading the input strings, passing strings to the object that builds the array of country objects. So, I am restricting your design such that your pass input strings from the input file to the Constructor in your country class for parsing and initializing the attributes for each object. You have likely been doing all this. **Please ensure that you do this as described.** Please note that I am dictating a design.

Task 2: Build a Binary Tree from the Array of Country Objects.

Each node in the tree is to contain only country name, country capital, capital's population, and region number of that country. None of the other attributes are needed in the nodes of the tree – at this time. The binary tree is to be built in the same order as the countries appear in your country array and it is to be built based on country name.

Task 3: Display the Binary Tree. 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: **Task3: Iterative NLR Scan** followed by a second header containing: **Country Name Capital's Population** suitably spaced out with ascending node numbers (nodes on the tree), the country name and its capital's 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. Display the tree using an iterative LNR scan.

Same as above. But your first header says: **Task 4: Iterative LNR Scan.**

Task 5: Display the Binary Tree: Display the tree using an iterative LRN scan.

Same as above, but your first header says: Task5: Iterative LRN scan.

Tasks 6-8: Using appropriate headers for each of the lists below, you are to display the resulting binary tree using scans indicated. Be certain your header label describes the scan approach used. Make your outputs clearly distinguishable via the headers as needed.

Task 6: Task 5: NLR format – recursive scan.

Task 7: Task 6: LRN format – recursive scan.

Task 8: Task 7: LNR format - recursive scan.

Task 9: Delete the following nodes from the binary tree in the order given: Delete the nodes, if present, from region 1. Your output should say: Task 9: Country <country name> deleted from the binary tree.

Task 10: Display the binary tree using an LNR recursive scan – appropriately tagged.

Task 10: LNR Recursive Scan after Update.

Zip all files in your project as you have done. Submit via Blackboard in our accustomed way. Be certain your zip files are correct. **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.**

Grade Sheet – Program 4
COP 3540 – Spring 2010
Due 6 April 2010 – start of class
Drop dead date: 8 April – start of class.

Name: _____ **Score:** _____**/110**_____

Source Code – 20 points _____

Indentation; Internal comments; Scope terminators; lack of static methods…

Program Design – 20 points _____

Be certain to accommodate the I/O as prescribed in the specifications. This is very important and I will look for it, please. Also, ensure your main class is entitled Main. It will help me grade your code.

Javadoc – 10 points _____

Appropriateness and completeness of comments
ALL methods must have Javadoc comments up front that are meaningful, please.
Include @params and @returns as appropriate.

UML – 20 points _____

Standard UML conventions **must be subscribed to this time.** Please ensure that you follow the rules. All classes are connected; use correct method formats and correct connections and visibility indicators.

Outputs – 40 points

Accuracy and **Format. All functionality present!**
Clearly the deletes constitute most of these points. The search option will be added **if all goes well** with the deletes and all design components are included in your deliverable.

➔ If your program does not contain all the functionality required, you cannot expect a grade above 60%; that is a 'passing grade.' 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. Even if you should receive a zero, you still must submit all programs to ultimately pass this course.

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