COP 3540 – Data Structures with OOP

Project 3 – Spring 2011

Due: 11 March (Friday) late date: 14 March 2011 (Monday) 2pm

Doubly-Linked Lists, Stacks, and More

Using NetBeans 6.9.1 or later version, you are to write a Java program using OOP principles to accommodate the following functionality

Assignment #3

Objectives:

Provide student additional experiences with file input.
Provide student exercises in learning UML (architectural design)
Provide student exercises in developing detail design pseudo-code
Provide student exercises in Javadoc and its various formats
Provide student exercises in building a doubly-linked list of Country objects.
Provide student with experience in inserting, deleting, and searching the doubly-linked list
Provide student with exercises in updating a linear linked list

Overview:

Given the sequential file, ***Euro-Countries.2.txt,*** you are to create a doubly linked list of objects from countries in Western Europe only. Then you are to create two stacks implemented via linked lists – one for 'adds' and one for 'deletes' from the doubly-linked list. Perform this maintenance, and then display the updated linked-list backwards.

Functions:

Task 1: Create an array of Euro-Country objects from the input file (Western Europe only). You have basically already done this. You must use an Arraylist.

Task 2. Sort the file using a **bubble sort**, sorting on Country name.

Task 3. With an appropriate header describing the list of sorted countries below, display the array of countries from Western Europe. Be certain that your product looks professional!

Task 4: Create a doubly-linked list of ordered Euro-Country objects based on country name. You can do this directly from your sorted array.

Task 5: Using a display() method (do not use toString(), display the ordered doubly-linked list – one object per line two ways: one using the forward pointers, one using the backward pointers. Be certain to precede this display with an appropriate header naming

your outputs, such as Doubly-Linked List Displayed Using Forward Pointers followed by a column header the names the attributes of the country objects that follow. These entries will be in sorted, ascending and descending order.

Task 6: Using the input file, **LinkedListUpdateTrans.Spring2011.txt**, you are to create two stacks of country objects with each stack implemented via a linked list. One stack is to contain the Add transactions, while the other stack is to contain the Delete transactions. You are to open() this file and create the stacks with a single pass of the input file. Read and process these inputs until EOF. Close the input file explicitly. The 'A' stands for Add and the 'D' stands for Delete.

Task 7. Using appropriate professional headers that indicate the data beneath, you are to then display each stack separately. You are to use a toString() method to display these. Tag each one so that it is clear which stack contains the Adds and which one contains the Deletes.

Task 8: Update the doubly-linked list using the stack of delete transactions. For each transaction, indicate the success of the Delete attempt (Found) or the failure of the Delete attempt (Not Found). Found and Not Found should follow the displaying of the string itself on the same line. For each successful attempt, also provide the number of links inspected before **and including** getting a good hit on the same line. Line these outputs up using a %s field descriptor. At the end of processing the deletes, provide a count of successful attempts and unsuccessful attempts at the end. The output must have an appropriate header, such as Deletions from Linked List

Task 9: Same as Task 8 –just about. Update the doubly linked list using the Add stack as inputs. For each transaction, indicate the success of the Add attempt (Success) or the failure (Dupe Add Attempted) of the Add attempt. Ensure the add node is in its proper ordered position in the linked list. Provide a count of successful attempts and unsuccessful attempts after the flag, Success or Dupe Add Attempted on the same line. Provide a count of successful attempts and unsuccessful attempts at the end. For each successful Add attempt, also provide the number of links inspected before getting to the correct place in the linked list.

Task 10: Display the updated doubly-linked list backward using rear pointers. Naturally, as always, include appropriate headers.

You are done!

This assignment looks long. But it really isn't. Take it one step at a time, as usual. You have PLENTY of time if you start right away and work slowly and methodically.

Procedure:

I urge you to tackle this problem incrementally. Using a small sample of the input file to test your procedure. Then build the entire linked list. Verify as you go. Use the

toString method or other display method to your advantage and VERIFY that you are in fact building the doubly-linked list correctly!

Deliverable: Your zipped folder to me MUST include copies of your data files. 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. **I very well may, however, substitute a different LinkedListUpdateTrans.Spring2011 file for testing.**

You are to zip all files in your COP3540 folder being certain to have a Main class and your project named, project3yourname as expected. Zip your project file and submit it to me via Blackboard Assignment links as you have done in the past. **Check** out your work prior to submitting it to me and print out a copy of your submission. Cover yourself!

Grading – Project 3 – COP 3540
Spring 2011 Program is worth 100 points

Source Code – 20 points
Looked at closely:

Indentation – consistent and reasonable. Suggest four spaces; Internal comments; Scope terminators; Overall program documentation and readability.

Architectural Design (using UML) – 15 points

Your design must reflect good object-oriented design as we have discussed repeatedly. Objects are to contain the methods that operate on the data they contain as much as possible. Ensure your UML design reflects these classes and their methods with appropriate signatures. 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. All classes are to be connected. Be certain to note that UML entries for a method are NOT the same as a Java method entry.

Detail Design (using pseudo-code) – 15 points

You must provide your pseudo-code for your Euro country collection class only (or whatever you call it. This is the one that will contain the pointer to the start of the linked list. It really is your most significant class, and should have the most intelligence in it. This needs to be the class that actually has the methods that does the real work regarding the linked lists: building updating, displaying, validating, etc. Examples may be found on my web page. Recognize that your pseudo-code should be done prior to your coding in Java. **If you have any questions, please ask early!**

Javadoc – 10 points.

Appropriateness and completeness of comments

ALL methods must have Javadoc comments up front that are meaningful, please.

Use recent email guidance on @params and @returns.