COP 3540 – Data Structures with OOP

Project 2 - Spring 2007

Due: February 19th, 2007 (Monday)

Priority Queues

Using NetBeans 5.0, you are to write a Java program using OOP principles to accommodate the following functionality

Assignment #2

Objectives:

Provide student with additional experiences with file input output.
Provide student with exercises in learning UML
Provide student with exercises in Javadoc and its various formats
Provide student with exercises in building an array of objects to support priority queuing.
Provide student with experience in inserting, deleting, and searching priority queue of objects.

Functionality:

Given a sequential file, States.Spring2007 on my web page, you are to build a series of priority queues discussed ahead, and one single input queue (not a priority queue – just a regular queue). You are specifically required to build six priority queues one for each Region based on Nbr of Region to group the states in each Region.

Please note that there is considerable additional data in the file that you may not need for this program. Do NOT eliminate it. Just don't process it.

The input data is not sorted in any way. As you read an input record from this file, create an object of type (class) State. Add this object to one of the six queues, depending on the NbrofRegion. All other inputs are to be ignored. As you insert() an object into its respective priority queue, the objects are to be **in order** by state name. Your insert() routine must reflect this. (this means that the states in each of the resulting priority queues will be ordered by state name.)

You will note that at the conclusion of building your priority queues, each priority queue may well have a different number of objects within it.

Once all six priority queues have been built, you are to display each of the queues with an appropriate centered header on your screen, such as

New_England
<skip a line> <note also, this is Region #1>
Connecticut
Maine
Massachusetts
New Hampshire
Rhode Island
Vermont

Middle_Atlantic

Delaware
Maryland
New Jersey
New York
Pennsylvania
Virginia
West Virginia

Etc.

Once you have displayed the six priority queues, you are to open a transaction file called ***newStates.txt.*** You are to read each of these input streams and build an array of objects. (This will be a small array – fewer than ten records). The format of the objects, however, should be the same as the format of the objects in the priority queues. **Ensure your program is not dependent upon exact number of input records.**

Once this non-priority queue is built, you are to display the queue in a professional manner – simply a text listing, centered and titled will be sufficient.

Then, taking each of these entries, insert them into the appropriate priority queues using your same insert() algorithm you used when the priority queues were initially built.

Display the updated priority queues in the same order as you did the original set.

Bingo! You are done.

Your zipped folder to me MUST include copies of the input 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.

UML

You are to include a UML class diagram. You may use Word or Power Point. Drag your UML design file into your P1 subfolder within your COP3540 desktop folder. It will be included in the zip file to me.

Javadoc

All programming is to be accompanied by appropriate Javadoc. Generate your Javadoc files and include the **xxxxxxx** file generated.

Get it right this time, okay?

You are to zip all files in your P2 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 project2.youruserid, as project2broggio NOT project2n00010109.

Grading

Source Code – 30 points

Indentation
Internal comments
Scope terminators
Overall program structure

Program Design – 20 points

Appropriateness of the objects and their services provided
Interface to objects
Attribute and method visibility

Javadoc – 10 points

Appropriateness and completeness of comments
ALL methods must have Javadoc comments up front that are meaningful, please.

UML – 10 points

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.

Outputs – 30 points

Accuracy and Format
Skip lines in between displayed numbers for readability.
Include headers / descriptors as you may feel appropriate.