COP 3540 – Data Structures with OOP

Project 2 – Spring 2011

Due: Wednesday, 21 February, 2pm

Drop dead date/time: 21 Feb, 2pm

Circular Priority Queues

Using NetBeans 6.7.1 or later version, 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 and circular queuing.
Provide student with experience in inserting, deleting, and searching priority queue of objects.

Functionality:

Task 1: Build ArrayLists: Given a sequential file, ***Euro-Countries.2.txt*** on my web page, you are to build a series of circular priority queues discussed ahead, and one single input stack.

Here in Step 1, you are to build four single dimensional arrays **using ArrayList().** Thus you will not declare an input size restriction on the array. The input data is not sorted in any way. As you read an input record from this file, create an object and write the object to the appropriate ArrayList. There should be one array for each region as we have done. You are only to build ArrayLists for those records whose region-code is 1-4. When you have fully processed the input file, you will have four arrays of objects of differing lengths- something you are aware of.

Do not sort the original array of strings that you read as inputs. The order / contents is changed from program 1.

Task 2. Exception Processing. Using Exception Processing Procedures (try…catch), your program is to diagnose input fields (attributes) in error. You need to check for numeric fields in the region number (permissible values are 1 through 4) and the numeric population field that represents the population of the capital of the particular country. (Input attribute must be able to be converted to an integer)

For those input 'records' that have problems, you are to **not** build an object for them and they are not to be entered into the array of Euro-Country objects. Rather, these bad records are to be written to an output sequential file entitled, ***ErrorFile.txt.*** As you encounter a bad input, your catch code is to simply write the entire record unchanged to the errorfile. The next 'write' is to include: BAD INPUT (left justified) followed by the appropriate messages (you may have one or two messages – one per line): "Bad / Missing Input Region Number followed by a single space followed by the number and/or Invalid Population Value followed by a space followed by the offending input value. (Again, you may have one or two violations for each of these bad records). At the end of building the circular priority queues, you may close() this file. Be certain to drag it into your project folder so I can look at it later.

Task 3: Using each array in turn, you are to build a **circular priority queue** for each array. The circular queue for each queue is to be of size 10. **The priority used in inserting objects into the circular priority queue is ascending country population.** So your insertpq() algorithm will need this information.

You may define each queue with the length of 10 if you wish. As you build each circular priority queue from its respective unsorted sorted array, you will note that there are more entries in a couple of arrays than there is room in the circular priority queue for that array. Thus you have a problem. 'No Room in the Inn."

Task 4: Continuation of Step 3 As you attempt to add to the circular queues and you discover that the queue is full, you are to display a message citing the country name of the object you wish to add to the queue, display on the next line the country name of the object you are removing from the queue to make room for the new object, add this new object into the priority queue, and then print "success" on the third line indicating that the priority queue now has the new item in it and has removed() one item. (Lots of fun!!!)

Task 5: Upon completion of building the circular priority queues (note, you have not done anything circular yet…), you are to print out **nicely formatted** each queue in column order such as:

Priority Queue for Northern Europe
XX
XX
XX
…
XX

Priority Queue for Central Europe
XX
XX
XX
…

All of your answers should be the same. So check with your classmates!! Also, take advantage of the discussion board on Blackboard.

Task 6: Remove four objects from Eastern Europe and from Western Europe.

Task 7: Clearly label and print out the resulting circular priority queues.

DONE!!!!

UML

You are to include a UML class diagram. You may use Word or Power Point. You may supply a .pdf file or use SmartDraw. Drag your UML design file into your p2 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 using the procedure on my web page and include these in your submission to me.

You are to zip all files in your P2 as expected and transmit them to me via Blackboard using the same naming conventions as in P0 and P1. Your zip file is NOT to include your N-number. Rather, it must be project2.youruserid, as project2broggio NOT project2n00010109.

If you have questions, please ask early! As of today, 31 Jan, you have all you need to do this program.