Assignment Prefix: Lab105

Due Date: Friday, Feb 22nd @ 11:59pm

Points: 100

This is an individual assignment.

**Restrictions:**

**You cannot use any predefined Java classes in writing this lab.**

**You CAN import the java.util.Random class.**

Create a NetBeans project using the standard naming convention, i.e. Lab05-LastFM and save it to a location like the desktop or your flash drive. In the project you will do the following:

Implement the following data structures as described in the textbook:

* Stack interface, Code Fragment 6.1
  + Modify the interface so that the push is required to throw a Stack Full exception if called on a full stack.
* ArrayStack, Code Fragment 6.2
* LinkedStack, Code Fragment 6.4
* Queue interface, Code Fragment 6.9
  + Modify the interface so that enqueue is required to throw a Queue Full exception if called on a full queue.
* ArrayQueue, Code Fragment 6.10
* LinkedQueue, Code Fragment 6.11
* List interface, Code Fragment 7.1
* ArrayList, Code Fragments 7.2, 7.3, 7.4, 7.5
* You may find it handy to add a toString method to each of the data structures (not required)

Your code for each of the data structures must be fully commented:

* Each structure must include a header block similar to the example provided below.
* The comments provided by the textbook authors are sufficient for each of these data structures and must be included in your code.
* You can add additional comments for your benefit (not required).

Note – to implement the above data structures you may need to use some of the textbook data structures from earlier chapters. If you implemented these needed data structures in an earlier assignment you may copy their class files into this assignment. If you did not implement these needed data structures in an earlier assignment, then you will need to implement them as part of this assignment. Any copied or newly implemented data structures must be commented as indicated above.

**You will be using these data structures in future assignment. Be sure that you understand how each of the structures works. Be sure that each data structure is fully tested. Testing beyond what is described for the Client is highly recommended but not required.**

**Each of these data structures may be used as quiz and/or exam questions. Be sure that you can write the Java code for any of these data structures in full or in part.**

Write a Client class with a main method that tests the data structures as follows:

* For the ArrayStack, LinkedStack, ArrayQueue and LinkedQueue:
  + Perform a timing test for each of these data structures.
  + Each timing test should measure **in nanoseconds** how long it takes to add N Integers to the structure and how long it takes to remove N Integers from the structure.
  + N should vary from 10 to 100,000,000 increasing N by a factor of 10 for each test.
  + Depending on your system you may run out of memory before you reach the maximum value of N. If you run out of memory, your program should “gracefully” stop the test and move on to the next test.
  + Test results must be displayed in a nicely formatted ASCII table similar to the examples provided at the end of the assignment.
  + In the ASCII table:
    - Values in each cell are padded by 2 blank spaces
    - Each column is just wide enough to display the widest entry in that column including the cell padding. Your program must automatically adjust the width of each column based on the values that it needs to print.
    - It is strongly suggested that you create a method that generates the ASCII table. You could pass this method a 2-dimensional array of values that are to be printed.
    - Future assignments may require that you print out results in a similar ASCII table.
  + **Note that in the example tables the times are measured in milliseconds. Your project must measure the time in nanoseconds.**
* For the ArrayList:
  + Use Example 7.1 in the textbook as a guide for testing this data structure.
  + You might consider using your ASCII table method to generate a table similar to Example 7.1.

**Things to turn in:**

* Open a Microsoft Word document named using the standard naming convention, i.e. Lab105LastFM.docx
* Copy and Paste the source code of all of the classes used in this assignment.
  + put the Client class first.
* Copy and paste the output of the client program (i.e. the contents of the Console window after running your project.
* When copying the contents of a window use CTRL-A to insure that you get all of the contents.
* **Set your Windows font to a non-proportional font, e.g. Courier New, for the ASCII tables, and adjust the font size to avoid line wrap.**
* Next, zip the Project folder.
* Finally, on Blackboard, submit both your Word document and project zipped file.

**Example of header block for textbook code**

/\*\*

\* Data Structures & Algorithms 6th Edition

\* Goodrick, Tamassia, Goldwasser

\* Code Fragments 7.2, 7.3, 7.4 and 7.5

\*

\* An implementation of a simple ArrayList class.

\* \*/

**Example output of ArrayStack Timing test printed as ASCII Table:**

**Note: the examples are in milliseconds (msec); your project should use nanoseconds (nsec).**

**+-----------------------------------+**

**| ArrayStack Test |**

**|---------------+----------+--------+**

**| N | push | pop |**

**| | (msec) | (msec) |**

**+---------------+----------+--------+**

**| 10 | 0 | 0 |**

**+---------------+----------+--------+**

**| 100 | 0 | 0 |**

**+---------------+----------+--------+**

**| 1,000 | 0 | 0 |**

**+---------------+----------+--------+**

**| 10,000 | 0 | 0 |**

**+---------------+----------+--------+**

**| 100,000 | 0 | 0 |**

**+---------------+----------+--------+**

**| 1,000,000 | 16 | 0 |**

**+---------------+----------+--------+**

**| 10,000,000 | 235 | 0 |**

**+---------------+----------+--------+**

**| 100,000,000 | 31,748 | 62 |**

**+---------------+----------+--------+**

**|---------------+----------+--------+**

**| LinkedStack Test |**

**|---------------+----------+--------+**

**| N | push | pop |**

**| | (msec) | (msec) |**

**+---------------+----------+--------+**

**| 10 | 0 | 0 |**

**+---------------+----------+--------+**

**| 100 | 0 | 0 |**

**+---------------+----------+--------+**

**| 1,000 | 0 | 0 |**

**+---------------+----------+--------+**

**| 10,000 | 0 | 0 |**

**+---------------+----------+--------+**

**| 100,000 | 16 | 0 |**

**+---------------+----------+--------+**

**| 1,000,000 | 15 | 0 |**

**+---------------+----------+--------+**

**| 10,000,000 | 2,125 | 31 |**

**+---------------+----------+--------+**

**| 100,000,000 | 60,152 | 437 |**

**+---------------+----------+--------+**