Assignment Prefix: Lab05

Due Date: Tuesday, Oct, 2nd @ 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 named Lab05 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
- ArrayStack, Code Fragment 6.2
- LinkedStack, Code Fragment 6.4
- Queue interface, Code Fragment 6.9
- 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.

Each of these data structures has been 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 (you may break these test up into separate methods if you wish):

- For the ArrayStack and LinkedStack
  - Use Code Fragment 6.3 as a guide to test the stacks.
  - be sure to test the ArrayStack to see what happens when you push onto a full stack
  - pop all of the items off the stack
  - be sure to test to see what happens when you pop from an empty stack
  - make sure that the popped items come off the stack in the correct (LIFO) order.
- For the ArrayQueue and LinkedQueue
  - Use example 6.4 to as a guide to test the queues.
  - Be sure to test the ArrayQueue to see what happens when you enqueue into a full queue
  - Dequeue all of the items from the queue
  - Be sure to test to see what happens when you dequeue from an empty queue
  - make sure that the dequeued items come out of the queue in correct (FIFO) order.
- For the ArrayList
  - Run a timing test to see how long it takes to add items to the ArrayList just before, at, and just after the ArrayList has to double it data array size (see example below).
  - Start you test at n slightly less than 2^27 (i.e. 134,217,728) and then for each additional test double the starting value for n.
  - Keep increasing the starting value of n until you run out of memory.
  - Write your test so your program does not crash when it runs out of memory.
  - Use the nanosecond timer instead of the millisecond timer.
  - Use an ArrayList of Booleans to minimize memory requirements.

Things to turn in:

- Open a Microsoft Word document
- 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 copy the contents of a window use CTRL-A to insure that you get all of the contents.
- 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 of ArrayList Timing test:

```
run:

=====  Test adding  134,217,728 items to ArrayList  ======

Size =  134,217,726   Time =           7,128 nsec

Size =  134,217,727   Time =           1,140 nsec

Size =  134,217,728   Time =             855 nsec

Size =  134,217,729   Time =     443,055,016 nsec

Size =  134,217,730   Time =           1,996 nsec

Size =  134,217,731   Time =             570 nsec

Size =  134,217,732   Time =             855 nsec

=====  Test adding  268,435,456 items to ArrayList  ======

Size =  268,435,454   Time =             570 nsec

Size =  268,435,455   Time =             285 nsec

Size =  268,435,456   Time =               0 nsec

Size =  268,435,457   Time =  23,605,535,482 nsec

Size =  268,435,458   Time =           1,996 nsec

Size =  268,435,459   Time =             570 nsec

Size =  268,435,460   Time =           1,141 nsec

=====  Test adding  536,870,912 items to ArrayList  ======

Size =  536,870,910   Time =             285 nsec

Size =  536,870,911   Time =             285 nsec

Size =  536,870,912   Time =             285 nsec

Out of memory at n =  536,870,912


BUILD SUCCESSFUL (total time: 1 minute 30 seconds)
```