COMP 2150 – Spring 2014
Homework 8: Stacks and Queues
(Posted Apr. 21 – Due Apr. 28 by 12:40 pm)

Remember that as stated on the syllabus, this assignment should be an individual effort (contact me or visit the Computer Science Learning Center, www.cs.memphis.edu/cslc, if you need help).

Submission: Please zip your source code into a single file (you can zip the entire project folder if you're using BlueJ) and upload it to the proper folder in the eCourseware dropbox at https://elearn.memphis.edu. The dropbox will cut off all submissions after the indicated deadline, so don't wait until the very last minute to submit your work!

Grading: This assignment will be graded by the TA Rahul Vemuri (rvemuri@memphis.edu). If you have questions or concerns about your grade, please contact him first. I'll be happy to look over your assignment myself if he's not able to resolve the situation to your satisfaction. Also remember that as stated on the syllabus, all submissions MUST compile and run to receive credit. The TA does not have time to find and correct your syntax errors!

Coding Style: Be sure to follow the good coding practices that were discussed in COMP 1900:

- Use descriptive variable and method names. Avoid using single-character names unless it's very obvious what the variable's being used for (like a loop counter).
- Follow standard Java programming conventions for **variableAndMethodNames**, **ClassNames**, **CONSTANT_NAMES**.
- Consistently indent your code.
- Use comments judiciously. Every source code file should include a comment block at the top that lists your name and the assignment number. Every non-trivial method should have a comment preceding it that specifies what the method does, what its parameters are, and what it returns. For simple methods like accessors or mutators this isn't necessary, but it doesn't hurt to put in a simple comment like `// accessor methods`. I also expect to see comments throughout your code explaining what actions are being taken!

This assignment will give you a concrete idea of why we made some of the decisions we did in the linked list implementation of a stack and the array implementation of a queue.

1. **(10 pts)** Write a **NaughtyLinkedListStack<E>** class. Your class should implement a stack using a linked list, but it should perform the push and pop operations from the tail of the list instead of the head. Furthermore, do not maintain a reference to the list's tail node.

2. **(10 pts)** Write a **NaughtyArrayQueue<E>** class. Your class should implement a queue using an array. However, it should fix the front of the queue at index 0 of the array, meaning that when a dequeue is performed, all remaining array elements need to be shifted down 1 index to the left.

3. **(5 pts)** Write a client program that performs a large number* of push/pop/dequeue operations on these "bad" implementations, displaying the time required to do so. Compare this with the time required for the same operations on the "good" implementations we wrote in class.

   Remember that you can use the built-in method **System.currentTimeMillis()** to get the current system time in milliseconds. Use this as a "stopwatch" – call **currentTimeMillis** once to get the starting time, execute the code you want to time, and then call **currentTimeMillis** again to get the ending time. The elapsed time is simply the difference between the start and end times.

* The precise definition of "a large number" will vary depending on your computer hardware. You want the number to be large enough to see a noticeable difference between the different implementations, but you don't want it to be so large that you have to sit and wait for a few minutes every time your code runs! If you're running this on a low-power netbook, make it on the smaller side… if you're on an overclocked Core i7, knock yourself out with a gazillion operations.