CS 1316 – Homework 10 – Quack / Ring Buffer

Due: Friday April 9th, 2010 before 6pm.

Out of 100 points

This is a **pair programming** problem! You are expected to work with the person you have been pared with in class, and you are both responsible for submitting the exact same code to T-Square. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. You should not exchange code or write code for others. For pair programming assignments, you and your partner should turn in identical assignments. Collaboration at a reasonable level will not result in substantially similar code (with other groups).

Implementing a Quack using a RingBuffer

For this assignment, you will be completing several methods in the (currently incomplete) RingQuack.java class file. Specifically, you will fill in the implementation of:

- void enqueue(E element)
- void push(E element)
- E pop()
- E dequeue()

The specifications and behavior for each method in this assignment are located in the Javadoc comments inside the incomplete RingQuack.java class file. There are several completed methods in addition to the ones that you need to implement. It is your task to implement each of the missing methods. Make sure that your code makes the same assumptions as the existing code with respect to the head/tail indexes!

While implementing the above methods, you may not use any methods from java.util or any other Java libraries.

NOTE: The Quack.java interface is using Java Generics, so that your Quack can contain objects of any type. The main() method in the RingQuack.java file instantiates an object of type RingQuack that can hold Strings, but your code should work with any objects stored within the list.

WARNING: If you are implementing your code correctly, you will have to cast Objects stored in the array to type E (the generic type). This will cause a compiler warning. Although you do NOT have to suppress the compiler warnings, you should understand why they are appearing. The peek() method has an example of how to suppress the compiler warning if you wish.

There is a main method in the RingQuack.java file that you can use to test your code. As you work on each method, you can test it by commenting out sections of the main method that use the methods you have not yet written. Note that the provided RingQuack.java file uses an array size of 3 for testing. Your code must work for ANY non-zero array size, using the MAX_SIZE constant instead of hard coding the number 3.

Grading Criteria 100 points total

Correct operation:

- enqueue(item) ……..…….…… 35 pts
  - Checks for full array 5 pts
  - Updates numStored 5 pts
  - Updates tail index correctly 10 pts
  - ....corrects for wraparound 10 pts
  - Places item in store 5 pts
- push(item) …..…….. 35 pts
  - Checks for full array 5 pts
  - Updates numStored 5 pts
  - Updates head index correctly 10 pts
  - ....corrects for wraparound 10 pts
  - Places item in store 5 pts
- pop( ) & dequeue() ……...……...… 30 pts
  - both metods return the correct data 10 pts
  - both methods update the head index correctly 5 pts
  - ...corrects for wraparound 10 pts
  - Both have the exact same behavior. 5 pts