CPSC 2151

Lab 4 – Interfaces and Interface Specifications

Assigned: Tuesday 9-26 && Thursday 9-28

Due: Tuesday 10-10 && Thursday 10-12 (**Before lab starts**)

In this lab, you'll be provided with some starter code that contains 4 files:

- IDoubleQueue.java – An unfinished interface for a Double queue.
- ArrayDoubleQueue.java – An unfinished implementation of a Double queue with an array backend structure.
- ListDoubleQueue.java – An unfinished implementation of a Double queue with an arrayList backend structure.
- DoubleQueueDriver.java – A main method file. Mostly used for verifying your solution.

A queue is a data structure where the first item added to the structure is the first item that is removed. FIFO – First in, first out. This specific Queue will hold **Doubles**. Note that a double in Java is not the same thing as a Double. We'll talk about this later… For now, know that they behave the same way: numbers with decimals.

Instructions:

The first three of the above listed files are unfinished. Your job is to finish them.

1. **IDoubleQueue.java** contains the method declarations but doesn't contain any of the contacts for its specification. Write the specification for this interface.
    a. Don't forget to do both the interface contract (with initialization ensures, defines, constraints) and the method contracts (pre and postconditions).
2. **ArrayDoubleQueue.java** contains all the contracts but is missing implementation for its functions. Take the provided contracts and implement the functions based on all the contracts.
    a. Remember to read all your contracts before writing your implementation.
    b. You can test your implementation via the **DoubleQueueDriver**.java code. Without modifying the driver, both implementations will produce basically the same output (the array implementation will have some [null] fields because

arrays are not resizable in java. The driver will set the max size of both to 10 but will only enqueue 8 times).

3. **ListDoubleQueue.java** contains the implementation for all the functions, but no contracts. Write the contracts based on the implementation.
    a. Don't forget to do both the class contract (with the invariant (if there is one) and corresponds) and the method contracts (pre and postconditions).

Contracts and Formatting:

Contracts should be formally stated when possible. We have not covered much formal notation, but if a parameter named row needs to be greater than 0, then the precondition should say "x > 0" not "[x must be greater than 0]."

Everything that has been mentioned as a "best practice" in our lectures should be followed, or you risk losing points on this portion of the lab assignment.

Partners:

You must work in teams of 3-4 students. You do not need to team with the same students that you have teamed with before in previous labs. **Remember that working with a partner means working with a partner, not dividing up the work.** You don't need to run a driver-navigator set-up for this kind of assignment, but you should still make sure that everyone works on contracts to some degree, and everyone works on implementation to some degree…

You will need this code for a later lab, so make sure both partners have a copy of it.

Tips:

Read the starter code's README :^)

Submitting to Gradescope:

Also in the starter code's README :^)