RArrayQueue Completion
Due: Day of lab at 11:59PM

1 RArrayQueue Completion Specification

1.1 Available Resources

- Lecture slides
- Other sections of the provided code
- me
- The textbook
- **DO NOT** refer to or use online implementations

1.2 Lab Instructions

- This is an *individual* lab.
- Make sure to read through all of the specifications so your submission is complete.
- Follow all the submission steps in the Setup document by the lab deadline.

1.3 Lab Link

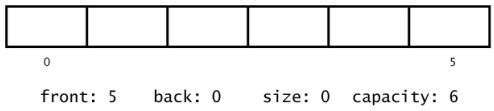
The skeleton code for the lab is available at `https://classroom.github.com/a/MNScd6H6`.

1.4 Implementation

This lab introduces a `RArrayQueue` class to implement a queue. The class is based on the `ArrayQueue` class with one major difference – the array fills in from the end of the array to the beginning.

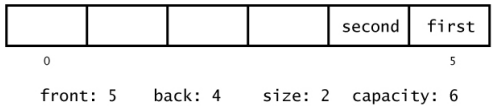
I have provided the class code in the **edu.wit.cs.comp2000** package. You will complete the remaining two methods – `offer` and `poll`. Descriptions of all of the method's expected behaviors are included in the lecture slides. In addition, Javadoc comments for them are included.

For your implementation, consider what effect the methods should have on all the data fields of the queue data structure. Filling in the array from end to beginning means that the front should be at a larger index than the back (unless the queue is currently looping around the ends of the array). Before writing code for a method, try drawing a diagram on paper that shows the changes that you want. Identify the data fields that need to change. As you write your code, verify that you change those data fields appropriately in code.

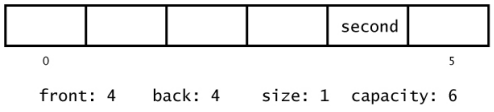


1. An empty `RArrayQueue`

2. After an `offer` call



3. After another `offer` call



4. After a `poll` call

1.5 Testing

In addition to the `RArrayQueue` code, JUnit tests are provided in the **edu.wit.cs.comp2000.tests** package. You can run these tests to see if the `RArrayQueue` implementation is performing correctly. The tests that I have provided check that some of the operations are behaving as expected. They also provide examples of the JUnit methods **assertTrue/assertEquals/assertFalse/assertNull**.

For the rest of the provided test methods (the ones that fail as not implemented), implement tests that check if that method works with your queue implementation. You can follow the steps that other test methods take for each test – create a queue, modify the queue, and then assert the results are what you expect. Make sure that you actually call the method that you are testing.

1.6 Considerations For Each Method

Be considerate in testing your ADT. Your goal is to test all of the possible cases for each `RArrayList` method. Each test method may have several assert method calls depending on how many possibilities you test for.

1.6.1 Common Edge Cases

How might our queue be structured when we run an operation? Consider all the different possibilities for each operation. For each case that applies, we will want to write some code that sets up a queue, followed by at least one assert statement.

2 Double Check:

- Have you implemented the `offer` and `poll` methods?
- Have you written two JUnit tests?
- Have you committed/pushed your code from the two files?

3 Grading

Each of the 4 **TODO** sections is worth $\frac{1}{4}$ of the lab grade.

Grades and any comments for the lab will be posted to your project on github. Grades will also be posted to Brightspace, eventually.