SE2832 Lab 1: Testing a Circular Queue with JUnit

1. Introduction

This is your first verification lab. Its purpose is to help you to start thinking about how to test existing software. From doing this, you will review the concepts of JUnit as well as begin to see some of the patterns necessary to ensure correct operation of software.

2. Lab Objectives

- Review the Java queue data structure
- Develop JUnit test cases to properly verify the operation of a basic Java data structure.
- Use independent learning to understand the operation of a circular queue.

3. Problem Overview

One commonly used low level data structure is the circular queue or circular buffer. It is often used in systems which requires fast adds and removes, as unlike other forms of queues, additions and removals can be designed to be atomic without the usage of blocks.

For this lab, you have been provided with a full implementation for a Java circular queue class. However, this code has not been tested and will most likely contain errors. You are responsible for developing JUnit tests to ensure that this class operates properly. If there are problems with the implementation, you will need to go into the source code and perform the appropriate fixes.

It is possible that there are some methods which are not implemented in the source code. If the methods are not implemented, it is acceptable for the code to simply throw an UnsupportedOperationException.

The queue, as it is structured, is perfectly capable of holding null references. Therefore, you should make certain that null can be added to the queue.

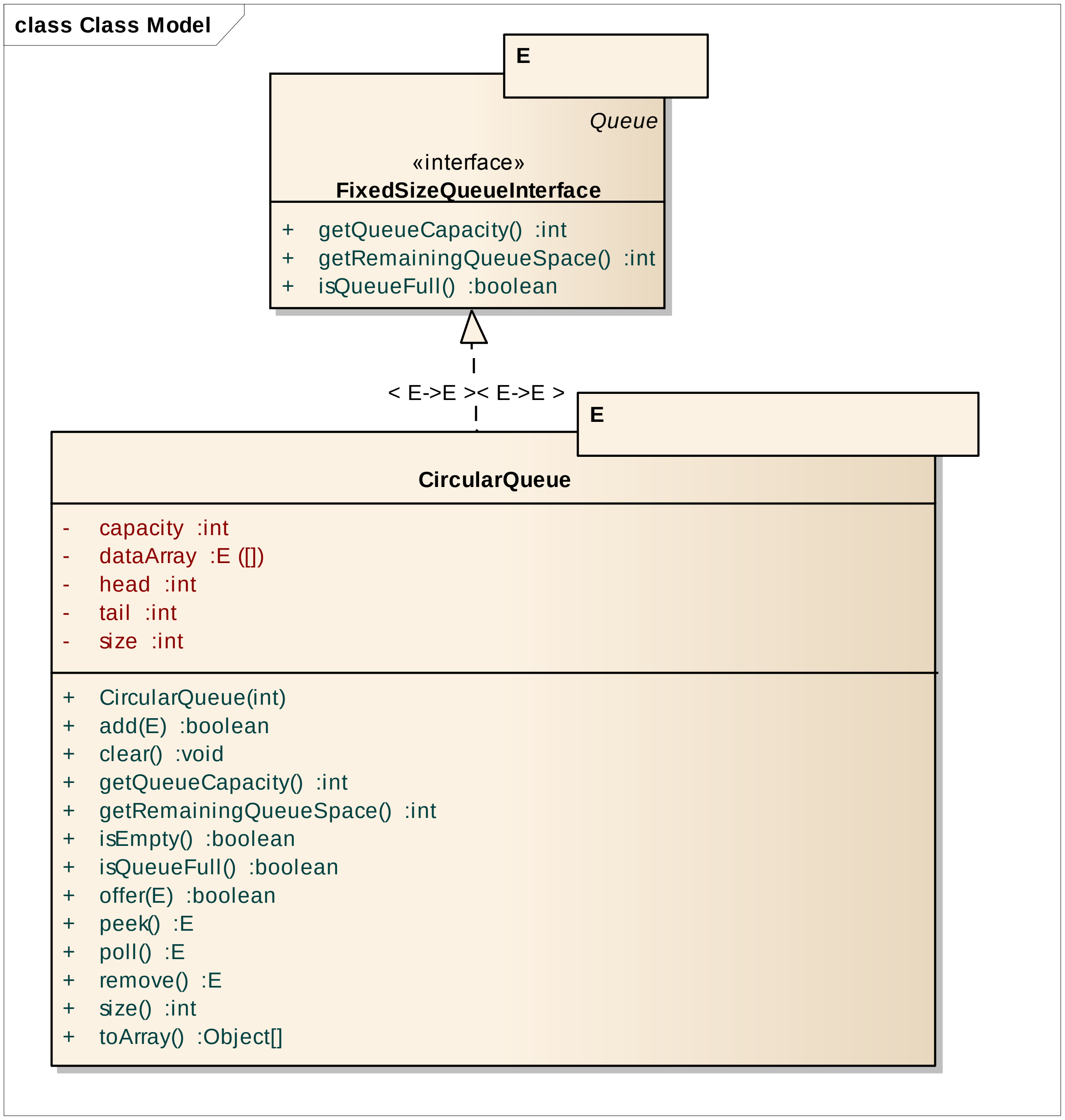

Figure 1 UML Class diagram for the project.

4. Deliverables / Submission

Each person will be responsible for submitting one report with the following contents:

1. Introduction -> What are you trying to accomplish with this lab? This section shall be written IN YOUR OWN WORDS. DO NOT copy directly from the assignment.
2. Testing Strategy -> What strategy did you use to create your test cases? How did you go about coming up with stock values to use for testing?
3. Fault Locations -> Where did you find problems in testing? What fixes resolved the problems? For this lab, this can be a simple table with line numbers and descriptions of the bugs and the fixes performed.
4. Things gone right / Things gone wrong -> This section shall discuss the things which went correctly with this experiment as well as the things which posed problems during this lab.
5. Conclusions -> What have you learned with this experience? What improvements can be made in this experience in the future?
6. Appendix -> JUnit test code. Submit your JUnit test code. You JUnit test code must be fully commented and meet the accepted practices for good documentation.

This material should be submitted as a single pdf file.

Additionally, your complete development package (corrected code, JUnit test suite, etc.) should be uploaded to blackboard.

If you have any questions, consult your instructor.

5. Preliminary Grading Rubric

	Weight Factor	**Rubric Score**	
Test Cases	2		Developed Test Cases - Test cases are thorough and complete. - Test cases properly simulate all possible behaviors based upon the potential sequences. - Test cases fully verify appropriate aspects of the component.
	1		Test Case Comments - Test cases contain comments and other documentation explaining the reason for their existence. - Test cases commented following appropriate MSOE commenting practices for Java.
Bug Fixes	2		Bug fixes - Only significant changes made in bug locations. No extra changes made in the code. - Bugs fixed efficiently. No excessive overhead introduced with bug fix. - Bug fixes pass the “master” test set from the instructor.
	1		Bug Fix Comments - Each bug fix commented with brief line explaining the correction made.
Submission	1		Introduction - Introduction fully describes what was intended for the lab - Introduction written in words separate from the lab assignment - Introduction written in multiple sentences
	1		Testing Strategy - Strategy for approaching test design described - Explanation of stock values used for testing provided - Multiple, complete sentences provided.
	1		Bug Log - Bug log details the bugs that were uncovered - Bug log details location of bugs in source code - Bug log details the fixes applied to the source code
	2		Reflection: Things gone right and things gone wrong - Things which went right with the lab fully described. - Things which went wrong with the lab fully described.
	1		Conclusions - What was learned from the lab described - Written in multiple sentences
	1		Non-Lab Material Submission - JUnit test cases submitted both in report and electronically - Corrected source code submitted as well