SM (M) 2017-2018 Lab 7: JUnit Testing

The Moodle webpage has a workspace for Lab 7. Download this an import it into your workspace.

To create a JUnit test case, click "New" in the Eclipse Menu, and select JUnit Test Case (Java/JUnit). This will create your basic test case (for example, see SimpleCalcTest.java in the workspace.)

Queue

This lab will involve writing a set of test cases for a Queue class. The description of the functional requirements is below. For each method described below, create a test function in the workspace that gives a full test coverage of that method. You can organize your test cases by function or group them based on the sample data.

The run the tests, you'll need to design the sample data and expected results. Your test methods need to cover each of the functions below. Use the @Before and @BeforeClass annotations to setup your sample data. The goal of these tests will be to identify faults (and you should find several). As you develop the test cases, try to correct these faults for a successful set of test cases.

Queue API

This impleme is in Queue.java. d be the first item that is returned (using dequeue). Y e of this running in Que

```
Queue queue = new Queue();

queue.enqueue("Frodo");
queue.enqueue("Samwise");
queue.enqueue("Gandalf");

queue.dequeue(); // Frodo
queue.dequeue(); // Samwise
queue.dequeue(); // Gandalf
queue.dequeue(); // No Such Element Exception
```

`isEmpty()` – Returns True if the queue is Empty, return False otherwise
`size()` – Returns the size of the Queue
`peek()` – Shows the oldest item in the Queue (the one that would be returned by dequeue)
`enqueue(Item item)` – Add an item to the Queue
`dequeue()` – Return the oldest items from Queue
`toString()` – Prints to System.out.print the Queue in FIFO order.