Unit Testing of the OpenMRS Thread-Safe Circular First-In-First-Out (FIFO) Queue

By: Megan Simpson, George Lutas, and Christian Ellwood

CSCI 362-01: Dr. Bowring

College of Charleston

Table of Contents:

Introduction:

This report will serve as a comprehensive recounting of the experiences of Team-6 as we developed an automatic testing framework of the Thread-Safe Circular First-In-First-Out (FIFO) Queue of the OpenMRS Open-Source Project. This framework has been developed to work with any system running with a Bash terminal. OpenMRS is a project developed in Java as a Medical Record Keeping Software and as such, we chose to work with this software with respect to the current Coronavirus-19 pandemic sweeping the world as well as the respectable efforts to help developing nations secure medical data of patients suffering from very serious diseases, such as Acquired ImmunoDeficiency Syndrome (AIDS), Tuberculosis, or Human Immunodeficiency Virus (HIV).

The goals of this system were to gain confidence with automatic testing and the Bash scripting language in order to achieve a comprehensive testing system that can be extended to support an indeterminate number of test cases, as long as the supplied test case follows the format that has been established and any necessary test drivers are supplied. However, in doing so, we uncovered a bug in the system that, while not entirely crippling to the system, could potentially cause issues in the operation of OpenMRS. As of the writing this report, we have not reported this bug, nor do we know the potential severity of this vulnerability.

Explanation, Instructions, and Examples of the Testing Framework

This testing framework is written exclusively in Bash, and as such, provided that there is a Bash terminal accessible to the user at time of running this script, the script should run as expected. The first operation that the script begins to do is to retrieve dependencies that the framework needs to operate. As of now, the dependencies that have been noted and are automatically installed at the running of the script is the Java Runtime Environment (JRE) and the Java Development Kit (JDK). The only input required from the user at this time is to provide the username and password to the system to install these dependencies as well as the credentials for an associated GitHub account. Once the system has installed these dependencies, the terminal is cleared, and the testing begins.

After this point, the OpenMRS code base is retrieved from the GitHub repository, pending authentication to a GitHub account, and is placed into the projects directory, where it will stay until the Team-6 repository is deleted. The script then proceeds to look in the testCases directory and counts the number of files in this directory. As it currently stands, the script assumes that any files in the testCases directory is a test case that contains the same information that the test cases currently in the directory contain. The order is irrelevant for correct operation.

The script parses each file for the test case driver, the test case ID, method being tested, any inputs to the method, and expected value from the test case text file and this is printed to the HTML file that contains the output for each test case. The file being tested is copied into the testCasesExecutables directory and compiled with the test driver that was parsed from the test case file. In order to ensure that all of the pertinent .class files are in the correct locations; the file being tested is compiled with both the -cp and -d flag with both options pointing to the current directory. The command is then joined with the command compiling the driver in the same location with the -cp flag pointing to the current directory. Upon successful compilation, the test driver runs, and the output is captured by the script and then redirected to the HTML file for output. This process repeats until the directory has no more test cases to traverse through, and, at the end of this, the script opens the default browser for the user and displays the outcome for all test cases that were provided at runtime.

For the ease of the viewer, the script also outputs a message for each test case, indicating whether the test case passed or failed. In addition, the script colors the word “PASSED” in green, and “FAILED” in red to increase readability. A sample of the output with both of these events occurring follows below.

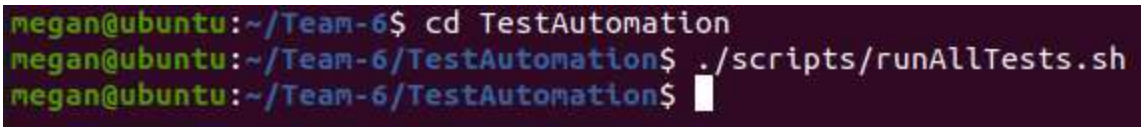

Figure 2.1: Running the script from the TestAutomation directory

Date and Time Test Ran: 02/12/2020 05:21:40 EST
Test Case ID: TC006
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: add
Test Arguments: {10, 50, 20} 25
Expected Result: [50, 20, 25]
Test Result: [50, 20, 25]
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:41 EST
Test Case ID: TC007
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: add
Test Arguments: {14} 10
Expected Result: [10]
Test Result: []
Test case FAILED. Result of the test is not the expected result.

Date and Time Test Ran: 02/12/2020 05:21:42 EST
Test Case ID: TC008
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: add
Test Arguments: {15, 23} 14
Expected Result: [23, 14]
Test Result: [23, 14]
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:44 EST
Test Case ID: TC009
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: add
Test Arguments: {2147483647, 0, -1, 2147483647} -2147483648
Expected Result: [0, -1, 2147483647, -2147483648]
Test Result: [0, -1, 2147483647, -2147483648]
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:45 EST
Test Case ID: TC010
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: add
Test Arguments: {-2147483648, -5, 0} -2147483648
Expected Result: [-5, 0, -2147483648]
Test Result: [-5, 0, -2147483648]
Test case PASSED.

Figure 2.2: Example output from the testing framework with both Pass and Fail

Building OpenMRS

Overall, the documentation made building the project less difficult, but not easy. Once the prerequisite software was installed, the documentation was followed closely, which led to the group being mostly able to build effectively. However, the documentation for building the project was incomplete, as there were missing dependencies, such as Jetty and MySQL, which gave the group some initial troubles. There were also some issues getting Jetty to work with the OpenMRS software. The solution to this was already present in the OpenMRS build commands, as the clean package command installs a Jetty instance for deployment.

There were a number of issues that were faced after the localhost website was running. It required a connection to a MySQL database, which presented a few challenges. First, the OpenMRS software requires access to a root account on the MySQL server. Given that the initial settings on the root account for MySQL by default authenticates on the user trying to access it, the first issue was to change the settings to a password-based authentication and a password for the account. Once that was finished, the OpenMRS building was unable to connect to the server, which presented additional problems. Since all of the software was running on localhost, there was no issue with connection errors. Upon looking through the error logs for the terminal hosting the website, it was found that the time zone on the MySQL server was set to "SYSTEM" time zone, which translated to "EDT," and was unrecognizable, thus the connection was terminated. After changing the time zone to UTC ("+0:00"), the website connected properly and worked as intended. Prop data was set up by OpenMRS to emulate proper operation.

All tests were run at the cleaning of the project by the Maven module. As a result, no work was required by the team to run any additional tests to ensure the proper operation of the OpenMRS software.

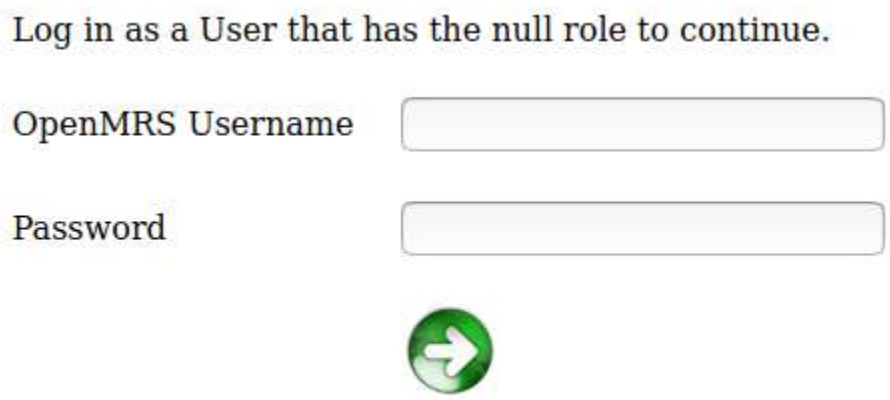

Figure 3.1: OpenMRS log-in screen

Developing A Test Plan

After building the project, the team's attention went directly to finding places to test. In the beginning of the project, the immense size of the code base made it very difficult to find places to test effectively, and once the User.java file was found, methods were determined as viable methods to employ our testing framework on.

We originally decided on the hasRole method from within the User.java file in order to generate the first five test cases of the project. Such reasons for this decision include the following reasons:

- hasRole is neither a getter nor a setter method
- hasRole is not a constructor for any object in the OpenMRS framework
- hasRole is a public method that returns a value at the end of the method
- hasRole calls other methods, which, in turn, identified multiple possibilities for testing

However, as discussed in the next chapter, a significant number of dependencies, including several external dependencies, made compilation next to impossible and development of the drivers was unable to proceed. At the time of selecting the hasRole method as the first method, the team was unaware of these dependencies as the OpenMRS software does not document these in the wiki or otherwise. However, once the team settled on hasRole, we generated five test cases shown below, as well as an example of a finalized test case, which is also shown below.

Test Case ID: 1
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/User.java
Method Being Tested: hasRole
Test Case Driver: ../testCasesExecutables/testCase1Driver.java
Test Data: Doctor false
Expected Result: true

Test Case ID: 2
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/User.java
Method Being Tested: hasRole
Test Case Driver: ../testCaseExecutables/testCase2Driver.java
Test Data: Doctor true
Expected Result: true

Test Case ID: 3
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/User.java
Method Being Tested: hasRole
Test Case Driver: ../testCaseExecutables/testCase3Driver.java
Test Data: Doctor false
Expected Result: false

Test Case ID: 4
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/User.java
Method Being Tested: hasRole
Test Case Driver: ../testCasesExecutables/testCase4Driver.java
Test Data: Nurse true
Expected Result: false

Test Case ID: 5
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/User.java
Method Being Tested: hasRole

Test Case Driver: ../testCasesExecutables/testCase5Driver.java
Test Data: Nurse true
Expected Result: true

Figure 4.1: All 5 test cases from the User.java file. Note the extreme differences between the original test cases and the test cases contained in the final framework.

```
Test Case ID: TC001
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: contains
Requirement: Determine elements in a queue to ensure security and reliability.
Test Case Driver: ../testCasesExecutables/containsTestDriver.java
Test Data: {3, 1234, 4321, 25} null
Expected Result: false
```

Figure 4.2: Example of a finalized test case in the framework.

Looking at the original test cases, it is very easy to see that there are some test cases that return different results for the same input data. This was due to the assumption that we could change the system configuration and grant or revoke super user status at run time, which would potentially return different results, based on the permissions granted to each role. Having the experience that we currently have; we understand the fundamental misunderstanding behind this and are all in agreement that this was a poor strategy to employ. The subsequent test cases in the final project do not employ this in the new test cases, as testing of this nature should return one value for each input, and not employ non-deterministic measures based around system configuration.

Designing and Building the Framework

Once the test plan was finalized and the methods selected, the team set on to develop the framework to test the methods selected in the User.java program. As none of us had any significant experience in Bash, the team elected to use Bash to develop the testing framework. Another significant reason comes at the fact that Bash much more easily interfaces with Git as well as the file structure of a Linux machine. While we evaluated Python to be able to handle these same constructs, in order to have a much easier time in directory traversal, the team found Bash to be much more potent in solving these problems. The team took this as motivation to become much more proficient in Bash, which yielded significant rewards as we were able to gain proficiency with Bash, as well as find resources to be more effective with Bash scripting.

However, as development continued, significant issues plagued the development, primarily the requirement of external dependencies to compile the User.java program. Some external dependencies that the User.java file required were the Apache Lang3 package as well as the Simple Logging Facade for Java (SLF4J) package. The team began attempts to gather the necessary dependencies, however, as a result of the immense scope of this task, the team decided to switch to the Thread-Safe Circular FIFO Queue program. This program had no external dependencies, nor any other dependencies in the OpenMRS codebase. This made development of the testing framework significantly easier, as the script was able to locate the file in the project directory and copy into the directory that contains the driver. All of the pertinent information was parsed directly from the test case file that the driver was reading. As a result, we were able to achieve our goals of a functional testing framework much more effectively and development chugged along much smoother.

Another significant challenge that the team faced with regards to developing the framework involved a unique challenge as a result of both Bash and the text editors used to create the test case files. In the parsing of the files, carriage returns at the end of the line was presenting an issue with Bash overwriting the variable value due to the carriage return. Because of this, after the first test case, the framework would not be able to find pertinent files. Through hours of debugging and outputting values with the terminal, the team was able to find the error and correct it, though at a significant time cost to the group.

The most potent challenge that the team faced, however, was the challenge of working virtually due to COVID-19 restrictions. Prior to this semester, the team members had felt very comfortable being able to collaborate in real-time. In addition to having to delegate significant tasks through text messages primarily, the challenge in having to be precise and concise with documentation made it much harder to collaborate, particularly as prior classes that had had any collaborative elements were able to work together in person, and the team members had significant trouble adjusting to this new dynamic. One solution that the team elected to employ was delegation of work between team members to work on separate parts of the project to ease the strain of collabortative work. This also presented some severe issues in that every person was responsible for their part and each person had to take the initiative to ask for assistance if there was a blocker. As communication got much better, this issue began to resolve itself.

Extending the Framework

Our experience with this deliverable was somewhat unexpected, as our group had a more difficult time in creating all twenty-five test cases than we had originally thought. Because our first five test cases were already up and running successfully, we assumed that creating twenty more in a similar fashion would be much easier. However, as with any task, things did not always go as planned and we learned very quickly that our assumption was made naively and prematurely. One small challenge that the group ran into was diligently looking through all code documentation to ensure that the correct datatypes were used for test case input and expected output, as several of our test cases involve abstract data types.

Additionally, we faced a problem specific to just one method, toArray, and its test cases despite all other test cases for the other tested methods functioning as intended. Although the toArray expected outputs and actual outputs of the test cases matched and should have caused all test cases to pass, the results of the test run indicated that the cases had failed. After some research, we found that the problem was likely due to regex-matching within the script, and placing quotations around the variable name, $expected, allowed for a simple fix, which allowed the variable to be treated as a string literal, another idiosyncrasy of the Bash scripting language.

However, the positive side to deliverable #4 is that the group continued learning new things and gaining valuable experience. Furthermore, the work progressed at a faster rate than for past deliverables, allowing the group time to fine-tune past deliverables. This included adding a time and date stamp to the testing output file for each time that the runAllTests.sh script is run. Our group also worked to better format the output of the test case results so that it is in a neater format that is easier to read and more detailed information is presented for each test case, including the expected output for each test case, the actual output, method being tested, etc. One way that the output is more neatly formatted is the use of color; “PASSED” is colored in green for each test case if the test case output matches the expected output, while “FAILED” is colored red if test case output does not match expected output for the tested method.

In addition, all methods are located within the ThreadSafeCircularFifoQueue.java file and are as follows: toArray, add, addAll, contains, and containsAll.

Each method contains five test cases each, which may be found in the testCases directory of the team repository. Each tested method has one driver each, called by the Bash script, runAllTests.sh, and the script automatically uses all twenty-five test cases within the testing framework. Like in past deliverables, output from the testing is written into the file out.html, which is automatically opened in the default web browser upon completion of running the Bash script.

Date and Time Test Ran: 02/12/2020 05:21:32 EST
Test Case ID: TC001
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: contains
Test Arguments: {3, 1234, 4321, 25} null
Expected Result: false
Test Result: false
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:34 EST
Test Case ID: TC002
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: contains
Test Arguments: {3, 1234, 4321, 25} 43
Expected Result: false
Test Result: false
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:35 EST
Test Case ID: TC003
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: contains
Test Arguments: {3, 1234, 4321, 25} 2344
Expected Result: false
Test Result: false
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:37 EST
Test Case ID: TC004
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: contains
Test Arguments: {3, 1234, 4321, 25} foo
Expected Result: false
Test Result: false
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:38 EST
Test Case ID: TC005
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: contains
Test Arguments: {3, 1234, 4321, 25} 1234
Expected Result: true
Test Result: true
Test case PASSED.

Figure 6.1: Test Results of contains method

Date and Time Test Ran: 02/12/2020 05:21:40 EST
Test Case ID: TC006
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: add
Test Arguments: {10, 50, 20} 25
Expected Result: [50, 20, 25]
Test Result: [50, 20, 25]
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:41 EST
Test Case ID: TC007
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: add
Test Arguments: {14} 10
Expected Result: [10]
Test Result: []
Test case FAILED. Result of the test is not the expected result.

Date and Time Test Ran: 02/12/2020 05:21:42 EST
Test Case ID: TC008
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: add
Test Arguments: {15, 23} 14
Expected Result: [23, 14]
Test Result: [23, 14]
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:44 EST
Test Case ID: TC009
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: add
Test Arguments: {2147483647, 0, -1, 2147483647} -2147483648
Expected Result: [0, -1, 2147483647, -2147483648]
Test Result: [0, -1, 2147483647, -2147483648]
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:45 EST
Test Case ID: TC010
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: add
Test Arguments: {-2147483648, -5, 0} -2147483648
Expected Result: [-5, 0, -2147483648]
Test Result: [-5, 0, -2147483648]
Test case PASSED.

Figure 6.2: Test Results of add method

Date and Time Test Ran: 02/12/2020 05:21:47 EST
Test Case ID: TC011
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: toArray
Test Arguments: 48, 20, 57
Expected Result: {48, 20, 57}
Test Result: {48, 20, 57}
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:48 EST
Test Case ID: TC012
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: toArray
Test Arguments: 22, 47, 20, 57
Expected Result: {22, 47, 20, 57}
Test Result: {22, 47, 20, 57}
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:50 EST
Test Case ID: TC013
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: toArray
Test Arguments: 48, 99, 57
Expected Result: {48, 99, 57}
Test Result: {48, 99, 57}
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:51 EST
Test Case ID: TC014
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: toArray
Test Arguments: 8, 11
Expected Result: {8, 11}
Test Result: {8, 11}
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:53 EST
Test Case ID: TC015
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: toArray
Test Arguments: 48, 20, 57, 77, 30
Expected Result: {48, 20, 57, 77, 30}
Test Result: {48, 20, 57, 77, 30}
Test case PASSED.

Figure 6.3: Test Results of toArray method

Date and Time Test Ran: 02/12/2020 05:21:54 EST
Test Case ID: TC016
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: addAll
Test Arguments: {930, 37, 109}
Expected Result: true
Test Result: true
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:56 EST
Test Case ID: TC017
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: addAll
Test Arguments: null
Expected Result: true
Test Result: true
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:58 EST
Test Case ID: TC018
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: addAll
Test Arguments: {39, 15, 20}
Expected Result: true
Test Result: true
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:21:59 EST
Test Case ID: TC019
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: addAll
Test Arguments: {499, 105, 175, 200}
Expected Result: true
Test Result: true
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:22:01 EST
Test Case ID: TC020
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: addAll
Test Arguments: {48}
Expected Result: true
Test Result: true
Test case PASSED.

Figure 6.4: Test Results of addAll method

Date and Time Test Ran: 02/12/2020 05:22:03 EST
Test Case ID: TC021
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: {10, 32, 15} {10, 15}
Expected Result: true
Test Result: true
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:22:04 EST
Test Case ID: TC022
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: null {10}
Expected Result: false
Test Result: false
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:22:06 EST
Test Case ID: TC023
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: {390, 4710, 3094} {390, 4709}
Expected Result: false
Test Result: false
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:22:08 EST
Test Case ID: TC024
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: {390, 10} null
Expected Result: true
Test Result: true
Test case PASSED.

Date and Time Test Ran: 02/12/2020 05:22:09 EST
Test Case ID: TC025
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: {390, 40, 10} {390, 40, 10}
Expected Result: true
Test Result: true
Test case PASSED.

Figure 6.5: Test Results of containsAll method

Fault Injection Results

After implementing the necessary changes from Deliverable 4, the team set to work finding potential faults in the ThreadSafeCircularFifoQueue.java program. Throughout all of the methods being tested and the associated internal methods that are called by these methods, the following changes to the code will produce the following changes in the test cases, with pictures of the output when that change is triggered.

Change 1 - In line 353, in the toArray method, changing the conditional from "if (result.length > size) {" to "if (result.length >= size) {" will trigger an off-by-one error and populate the final space with a null value in all toArray test cases, causing them all to fail.

Date and Time Test Ran: 23/11/2020 22:59:36 EST
Test Case ID: TC011
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: toArray
Test Arguments: 48, 20, 57
Expected Result: {48, 20, 57}
Test Result: {48, 20, null}
Test case FAILED. Result of the test is not the expected result.

Date and Time Test Ran: 23/11/2020 22:59:41 EST
Test Case ID: TC012
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: toArray
Test Arguments: 22, 47, 20, 57
Expected Result: {22, 47, 20, 57}
Test Result: {22, 47, 20, null}
Test case FAILED. Result of the test is not the expected result.

Date and Time Test Ran: 23/11/2020 22:59:46 EST
Test Case ID: TC013
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: toArray
Test Arguments: 48, 99, 57
Expected Result: {48, 99, 57}
Test Result: {48, 99, null}
Test case FAILED. Result of the test is not the expected result.

Date and Time Test Ran: 23/11/2020 22:59:51 EST
Test Case ID: TC014
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: toArray
Test Arguments: 8, 11
Expected Result: {8, 11}
Test Result: {8, null}
Test case FAILED. Result of the test is not the expected result.

Date and Time Test Ran: 23/11/2020 22:59:56 EST
Test Case ID: TC015
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: toArray
Test Arguments: 48, 20, 57, 77, 30
Expected Result: {48, 20, 57, 77, 30}
Test Result: {48, 20, 57, 77, null}
Test case FAILED. Result of the test is not the expected result.

Figure 7.1: Test Results of the first Fault Injection

Change 2 - In line 160, in the containsAll method, changing the logical operator from "||" to "&&" in the conditional will trigger a failed null check in test case 24, which results in a failure for this test case.

Date and Time Test Ran: 23/11/2020 23:11:19 EST
Test Case ID: TC023
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: {390, 4710, 3094} {390, 4709}
Expected Result: false
Test Result: false
Test case PASSED.

Date and Time Test Ran: 23/11/2020 23:11:24 EST
Test Case ID: TC024
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: {390, 10} null
Expected Result: true
Test Result:
Test case FAILED. Result of the test is not the expected result.

Date and Time Test Ran: 23/11/2020 23:11:29 EST
Test Case ID: TC025
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: {390, 40, 10} {390, 40, 10}
Expected Result: true
Test Result: true
Test case PASSED.

Figure 7.2: Test Results of the second Fault Injection

Change 3 - In line 422, in the internalContains method, which is called by both "contains" and "containsAll," changing the ".equals" comparator to the "==" triggers a fault in test cases 5 and 25 only.

Date and Time Test Ran: 30/11/2020 14:55:26 EST
Test Case ID: TC004
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: contains
Test Arguments: {3, 1234, 4321, 25} foo
Expected Result: false
Test Result: false
Test case PASSED.

Date and Time Test Ran: 30/11/2020 14:55:30 EST
Test Case ID: TC005
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: contains
Test Arguments: {3, 1234, 4321, 25} 1234
Expected Result: true
Test Result: false
Test case FAILED. Result of the test is not the expected result.

Date and Time Test Ran: 23/11/2020 23:19:36 EST
Test Case ID: TC023
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: {390, 4710, 3094} {390, 4709}
Expected Result: false
Test Result: false
Test case PASSED.

Date and Time Test Ran: 23/11/2020 23:19:41 EST
Test Case ID: TC024
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: {390, 10} null
Expected Result: true
Test Result: true
Test case PASSED.

Date and Time Test Ran: 23/11/2020 23:19:46 EST
Test Case ID: TC025
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: {390, 40, 10} {390, 40, 10}
Expected Result: true
Test Result: false
Test case FAILED. Result of the test is not the expected result.

Figure 7.3: Test Results of the third Fault Injection

Change 4 - In line 168, removing the negation on the internalContains() method call will cause all containsAll test cases to fail that do not use the null check to bypass checking the array.

Date and Time Test Ran: 23/11/2020 23:29:05 EST
Test Case ID: TC021
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: {10, 32, 15} {10, 15}
Expected Result: true
Test Result: false
Test case FAILED. Result of the test is not the expected result.

Date and Time Test Ran: 23/11/2020 23:29:10 EST
Test Case ID: TC022
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: null {10}
Expected Result: false
Test Result: true
Test case FAILED. Result of the test is not the expected result.

Date and Time Test Ran: 23/11/2020 23:29:15 EST
Test Case ID: TC023
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: {390, 4710, 3094} {390, 4709}
Expected Result: false
Test Result: false
Test case PASSED.

Date and Time Test Ran: 23/11/2020 23:29:20 EST
Test Case ID: TC024
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: {390, 10} null
Expected Result: true
Test Result: true
Test case PASSED.

Date and Time Test Ran: 23/11/2020 23:29:24 EST
Test Case ID: TC025
File Being Tested: ../project/openmrs-core/api/src/main/java/org/openmrs/util/ThreadSafeCircularFifoQueue.java
Method Being Tested: containsAll
Test Arguments: {390, 40, 10} {390, 40, 10}
Expected Result: true
Test Result: false
Test case FAILED. Result of the test is not the expected result.

Figure 7.4: Test Results of the fourth Fault Injection

Change 5 - In line 160, changing the "==" equality operator in "c == null" check to a "!=" equality operator, this causes 3 out of the 5 test cases to fail, interestingly enough, one test case that does not rely on null at all, but passes under normal circumstances.

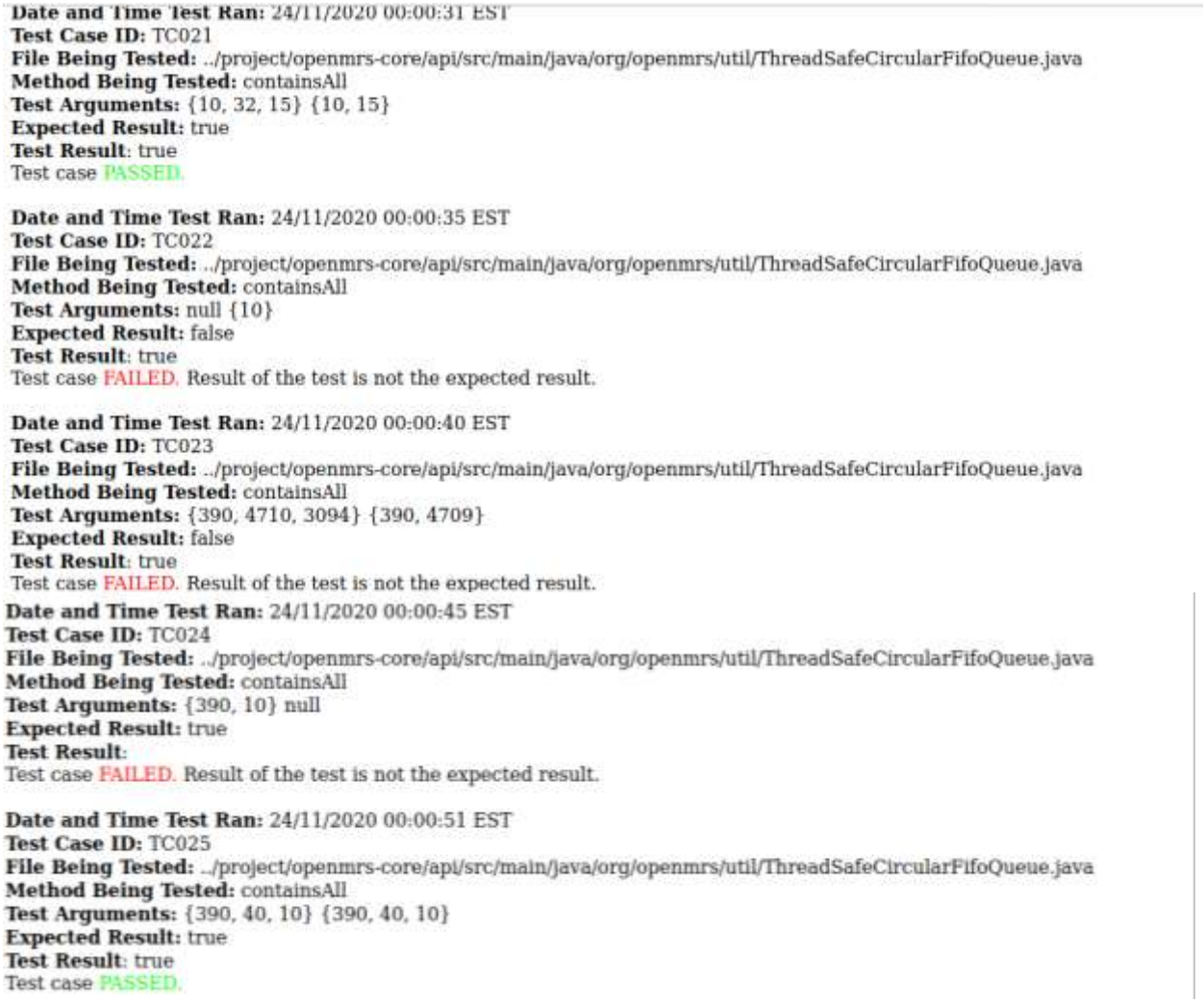

Figure 7.5: Test Results of the fifth Fault Injection

Overall, this learning experience was one of patience and determination. The codebase that we were working with tended to be scarce at certain points, which made finding potential faults much more difficult for certain methods. As a result, we had to work around those and try to capitalize on certain methods that were a little more substantial. We also tried to emphasize minimal changes that would be common for a programmer who is unfamiliar with the problem or unfamiliar with Java would likely make to determine whether the program was sufficiently redundant to protect against these. Off-By-One errors are very common and in the case highlighted, could potentially cause errors with the list. As well, the switch from "||" to "&&" is a very common mistake, especially with complex data structures, it can be very easy to mix the two up and make an error of this kind, which, as we saw, only triggered one error in one case, which would be significantly hard to detect.

Lessons and Experiences

Our overall experiences were relatively positive. From the outside, one would be forgiven for thinking that a project like this is a linear path through fixed and known obstacles. Eventually, our experiences taught us that a project like this is anything but smooth sailing. We had experienced bursts of massive progress followed by immense blockers that stopped progress for days at a time.

We learned that hidden problems that seem small can be truly massive in the amount of time it takes to fix them, and that communication is a reliable way to avoid making mistakes. Aspects of the project may end up being incorrect or unassigned, and without effective communication of these aspects, it is extremely likely that those things will slip through the group's notice.

Further, development of a constructed environment was an experience that each of us individually learned and underwent. The development environment of a computer scientist is varied between different machines, and each individual has to set up and maintain their environment. However, in the instance that the environment crashes or becomes corrupted, the developer is responsible for restoring their copy. The major lessons learned were to keep backups and never work off the master copy, save updates.

However, not all skills needed for this project were technical. Having the skillset to articulate your thoughts sufficiently to show our progress was a key skill that is a necessary auxillary skill to computer science and software development. As is the nature with client-driven work, being able to not only communicate your own code and progress to your co-workers, who usually have a very similar level of knowledge to your own, but also to clients, who may be less knowledgeable about technical matters, is a requirement that is sadly understated in the current curriculum until this class.

Besides the general project lessons learned, there was project-specific skills gained in terms of Bash scripting, unit testing, HTML, virtual machine setups. The truth is that the field of computer science encompasses a large field of technologies, and as such, learning to employ the technology that was originally researched has the potential to cascade into much more learning. It would be difficult to describe exactly what was learned for this reason, not to mention the individual experiences of each team member. As such, this serves as merely a highlight of the important lessons, not as an exhaustive list of every lesson learned by each individual.

Evaluation of Performance

The group's performance was, by far, exceptional in multiple areas. Having the years of experience with working in Java as well as the confidence that we all had going into the project, we felt as though the project would be time-consuming, but not necessarily difficult. While the project was time-consuming, we did encounter a series of difficulties that were eventually overcome. However, there were some significant issues that came up throughout the semester that definitely gave some troubles to us as we developed the testing framework, albeit not to the point of failure.

The most potent of the problems that we encountered was the delegation of work between group members and overcoming our own personal preferences to work well in the group dynamic. For example, each group member has their own specialties, however, we have been trained to work as effectively as possible on any topic that we have in front of us. We also have been trained to solve any problem in front of us, no matter how long it takes to solve it, even if there is another person who may have already encountered the problem or knows the fix to make it work. This issue was prevalent through all of the group members at some point or another, although not nearly to the extent that we were close to not finishing anything on time. When it came down to the final days before any deliverable was due, we all came together and re-delegated work as necessary to make sure that the required materials were present and finished.

Another signifcant issue faced involved communication in a purely digital medium. As it was previously stated, COVID-19 restrictions limited our ability to work collaboratively in a physical setting. In order to solve this issue, we employed texting, group chats, emails, and other forms of communication to ensure that any necessary communications were seen by the entire group and that other group members could check this and confirm these messages. Despite this, communication broke down a few times throughout the semester. The strain of other classes in addition to having to check text messages or other forms of communication for any sort of actual communication with the group meant that sometimes, messages went unread and other group members had to confirm that the message was sent or, in the worst case, pick up the work that was delegated elsewhere, just because there was silence on the other end. It did not cause significant disruptions, as these measures were put into place sooner rather than later, but this issue was a problem that the team had to overcome, which, as shown by the final product, the group did.

To end the evaluation, instead of focusing on the negatives, we would like to highlight the positives. There were a few times throughout the semester where the product we delivered at any given deliverable was not what we needed, and we had to make major course corrections. Those times were the times when the team rallied and put even more effort than before into the framework and spent hours fixing those mistakes to make sure that the next presentation solved those issues and introduced as few issues as possible in the next. Whenever a deliverable was not up to par, we regrouped within the hour of the presentation and jumped right in, fixing everything we could as soon as we could. Because of that, we believe that our group really took criticism in stride, which made our final efforts much more comprehensive.

Another positive note that the team would like to highlight is the good chemistry present within the group. In group meetings and communications, we were really able to have effective and potent communication to the extent that we were able to communicate. While the frequency, especially at first, was low, the effectiveness was high, and work was able to be delegated initially very quickly. In addition to this, the team was generally able to stay in high spirits. This really paved the way for high efficiency and high throughput with work. This positive attitude also helped keep meetings light and very effective.

As for the delivery of the class, an online format honestly helped more than hurt the learning aspect. There were far more effective lessons learned in the setting of the project that we feel would have been lost without the virtual format, such as the importance of documentation and commenting code, as previously mentioned. While the discussion over the homework topics was, we feel, lighter, the lessons that the project taught was extremely valuable and much more pertinent to software engineering.

One significant issue that we did feel affect our progress in developing the framework was particularly a lack of positive examples to generate our project outline as well as each deliverable. The freedom to develop the framework was definitely a boon, especially as every individual encountered different problems as well as approaching the problem from different angles, however, it felt as if, despite this freedom, many of the deliverables were not as open-ended as they could have been. This contrast left the group confused at times about what specifically was required for each deliverable.

One specific aspect of the class that we did specifically enjoy was the inclusion of pertinent readings that emphasized the importance of the lessons we were learning, such as the Therac-25 machine glitches underlining how important it is to test comprehensively. As visceral as it sounds, having those examples of the loss of life that can occur from only a software glitch imprints the importance of testing in a way that previous classes had not really spent time explaining in detail. As most of the group is preparing to graduate and develop software professionally, it is a sobering reminder that human lives are in our hands when we write code.

The inclusion of breakout rooms and a secluded spot for the team to work with almost no background noise also helped to focus the team on our goals. This was further helped by the early team-building exercises that we were assigned to get to know each other better and get a grasp on how each person thinks. These factors combined perfectly to generate an efficient team that worked at their goal tirelessly and presented a functional framework, complete with the deliverables requested as a result of the project.