CS320 Summary and Reflections Report

Matthew L. Lewis

Southern New Hampshire University

Table of Contents

[Table of Contents 2](#_Toc117338013)

[1. Summary 3](#_Toc117338014)

[1a. Describe your unit testing approach for each of the three features. 3](#_Toc117338015)

[1b. Describe your experience writing the JUnit tests. 4](#_Toc117338016)

[2. Reflection 4](#_Toc117338017)

[2a. Testing Techniques 4](#_Toc117338018)

[2b. Mindset 5](#_Toc117338019)

[References 6](#_Toc117338020)

CS320 Summary and Reflections Report

# 1. Summary

## 1a. Describe your unit testing approach for each of the three features.

### Contact Service.

#### To what extent was your approach aligned with the software requirements?

To the fullest extent possible, I would say. Before writing any of the code, I analyzed the software requirements and wrote code specifically to perform those requirements. At this point, I carefully wrote JUnit tests to verify the correct functionality of every piece of code using valid input, invalid input, and null input. I had to ensure all exceptions coded would be properly thrown in addition to all the correct variables being stored. At this point, I was able to identify and fix all errors in the code until all tests passed and the coverage was 100%.

For example, look at the following code…

Text

Description automatically generated

The code pictured above shows update methods that were written specifically to find an entered ID within the list, throw an exception if it isn’t in the list, and then update its relevant variable based on the input of the method. There is one update method written for each of the variables that the software requirements dictated should be able to be updated.

#### Defend the quality of your JUnit tests.

My JUnit tests, as stated in the previous response, tested valid inputs into every method as well as invalid inputs and null inputs. The tests asserted that the correct variables were properly stored and all coded exceptions were properly thrown. I ensured that all tests passed before finalizing the code and verified 100% code coverage as well. With this in mind, I am absolutely positive that the quality of the tests and thus the code itself is exceptional.

For example, look at the following code…

Graphical user interface, text, application

Description automatically generated

The cited code above shows where I tested the previously cited update methods. As can be seen, the methods were tested with assertions showing that they correctly updated the variables of the objects within the list that they were meant to update. Furthermore, the exceptions for invalid ID input were also tested to assert that they were properly thrown.

### Task Service.

#### To what extent was your approach aligned with the software requirements?

The approach here was very much the same as the approach for the Contact Service. The two were, in fact, based on extremely similar requirements. One distinct difference was that far fewer variables would need to be stored and only two update methods would be needed, as can be seen below…

Graphical user interface, text, application, email

Description automatically generated

#### Defend the quality of your JUnit tests.

My tests, again, achieved 100% coverage, testing for all forms of valid, invalid, and null input. As an example, I will show the tests for the previously cited code…

Graphical user interface, text, application, email

Description automatically generated

As can be seen above, the previously cited update methods were thoroughly tested using valid inputs and assertions to assert the correct variables were correctly updated. Invalid ID searches were also tested to assert that exceptions were properly thrown.

### Appointment Service.

#### To what extent was your approach aligned with the software requirements?

Finally, the appointment service was, again, very similar, and thus my approach was very much the same. One notable difference was its usage of dates which had to be coded a bit differently, as well as the distinct lack of update methods needed, as seen in the example below…

Graphical user interface, text, application, email

Description automatically generated

Pictured above is actually the entirety of the very small Appointment Service class. There are no update methods, and you can see that the dates are of date type, which is a departure from the previously exclusive usage of strings and integers.

#### Defend the quality of your JUnit tests.

Finally, I tested this just as thoroughly as any other script I’d written for the project, verifying that every object within the list could be created and removed with valid input as well as testing for invalid input and asserting that exceptions were properly thrown, as pictured below.

Graphical user interface, text, application, email

Description automatically generated

## 1b. Describe your experience writing the JUnit tests.

### How did you ensure that your code was technically sound?

### Technically sound code can be ensured by following standards such as proper usage of whitespace and formatting for readability purposes. In addition, ensuring that code throws exceptions for invalid input while performing the necessary functions with valid input helps to ensure that the code can actually be adequately tested. For example, this snippet of code I wrote here…

### 

### The above code is a method which has been carefully commented on to state its purpose. It first verifies that the String input actually exists in the HashMap and throws an exception if it does not. Afterward, it then puts the found String into a local Contact object, adequately named “foundContact” and returns it. By doing this, both valid and invalid input could be tested such as in the following code…

### 

### 

### As can be seen, both valid inputs and invalid inputs for the method in question were tested. Doing so helps to ensure technically sound code.

### How did you ensure that your test code was efficient?

In this case, code efficiency was simple to ensure. Since no loops were used in any of the code or otherwise memory-intensive operations, there wasn’t much inefficiency to worry about. In a case where a loop is used to find an object in a list, for example, it is useful to break from the loop once it is found so that memory isn’t wasted searching the rest of the list after finding the object in question. One thing I did for the sake of efficiency, after receiving advice from my instructor, was to use Hash Maps for the lists so that they would be in ID/pair combinations. This resulted in not having to use loops to search for objects by IDs. In addition, validating input and throwing exceptions before running the code of a method helps to ensure efficiency as well.

Look at the following code, for example.

Graphical user interface, text, application, email

Description automatically generated

In the above code, a class and several of its methods are shown. By using the Hash Map, no loops were necessary to perform the required operations. For each method, input is validated before performing any operations. The code is readable, and everything uses industry-standard naming conventions. All of this helps to ensure efficiency, not only when the code is run, but also if it needs to be changed at a later date. It will be quick and simple for anyone on the team to open the code, understand all its functionality and make any needed edits.

# 2. Reflection

## 2a. Testing Techniques

### What were the software testing techniques that you employed in this project?

I used assertions, exclusively. Specifically, I instantiated class objects that needed to be tested and used their methods that needed to be tested. I then wrote “assertTrue” statements to verify that all of the correct parts were updated to what they should be. Furthermore, I used “assertThrows” to verify every case of thrown exceptions through invalid or null input would be properly thrown. The previously cited test code shows these statements and how I used them.

### What are the other software testing techniques that you did not use for this project?

There are many other forms of valuable assertions such as “assertArrayEquals” which was obviously not needed due to my choice of Hash Maps over array lists. I also did not use “assertNotSame” for example, because it was unnecessary for my project. I only needed to test equality, showing correctly used update methods and of course, that exceptions were properly thrown. Therefore, the assertions used were all that were needed.

### For each technique you discussed, explain their practical uses and implications for different software projects and situations.

The “assertArrayEquals” assertion would be very helpful when ArrayLists are called for. For example, if a project needed a list of every class that could be taken by a student, I might simply make an array that holds the class type in it with no associated values. In this case, this assertion might be called for.

“assertNotSame” would be useful if you had two variables of an object that should never be equal such as if you had a Hash Map that kept track of a lowest amount statistic and a highest amount statistic of specified objects and their values were their amounts. You might want to ensure that the two do not have equal values.

## 2b. Mindset

### Assess the mindset you adopted working on this project.

I employed caution in how I approached the functionality of the code itself. Originally, I rushed a little bit through it and overcomplicated it with Arrays as opposed to Hash Maps and failed to even break out of loops. Afterward, I kept efficiency at the forefront of my mid and tried to code every operation the requirements called for in the simplest, efficient way I could think of. In doing so, I couldn’t help but think about the complexity and interrelationships that could come from deeper, more intricate projects. In this project, the interrelationships of the object class and the list class for that object was crucial to keep in mind. For example, verifying the inputs in the constructor method of the object itself helped to ensure the smoother functionality of the methods and exceptions that would be needed in the list class.

### Assess the ways you tried to limit bias in your review of the code.

Bias would absolutely be a concern. If you don’t feel confident in your own abilities, you will struggle to produce good results and probably spend an exorbitant amount of time testing, making it difficult to meet deadlines and good levels of productivity.

On the flipside, if you are overconfident in your abilities, you may fail to fully test your own code, rushing through the tests as more of a formality than anything else, and thus unnecessarily introducing errors into the overall project that could cause larger problems down the road.

### Finally, evaluate the importance of being disciplined in your commitment to quality as a software engineering professional.

It is crucial not to cut corners as a developer or as a tester. As a developer, you need to ensure that your code not only satisfies the requirements but also, that it does so in an efficient and optimized way. You also need to make sure it is well written and commented on so that any edits that are required can be done so without struggle.

As a tester, you need to ensure that you correctly verify the functionality of the code not only with valid inputs to every method but also invalid or null inputs. In each case, you need to test to ensure that the code is functioning as it was intended based on the software requirements, and you need to be vigilant to ensure that a high percentage (100% when possible) of coverage is attained by all of your tests.

References

Last Name, F. M. (Year). Article Title. *Journal Title*, Pages From - To.

Last Name, F. M. (Year). *Book Title.* City Name: Publisher Name.