CS320 Summary and Reflections Report

Jeffrey S. Carlson

Southern New Hampshire University

Table of Contents

[Table of Contents 2](#_Toc121853840)

[1. Summary 3](#_Toc121853841)

[1a. Describe your unit testing approach for each of the three features. 3](#_Toc121853842)

[1b. Describe your experience writing the JUnit tests. 4](#_Toc121853843)

[2. Reflection 4](#_Toc121853844)

[2a. Testing Techniques 4](#_Toc121853845)

[2b. Mindset 5](#_Toc121853846)

[References 6](#_Toc121853847)

CS320 Summary and Reflections Report

# 1. Summary

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

### Contact Entity and Contact Service.

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

When I was writing the code for Contact Entity and Contact service I would have the IDE on one screen and the requirements on the other screen. This way I would see one requirement and I would translate it to code. This can be seen with lines 17 to 46 of Contact.java and line 7 to 34 of ContactSerivce.java. If I would get confused by the requirements I would go through them slowly and write it down on the paper in a very straightforward way so it wasn’t wordy like the requirements were. Another thing that helped me get the requirements accurate was the feedback given by the teacher so I could see how I could improve the code or what I was missing. I also wrote the code for each assignment a little different and organized it differently each week. This way when It came time for the project I took the organization that I liked and rewrote the other two in the same way. This can be seen comparing the first submission compared to the final submission.

[Explain how your approach to testing the Contact Service was aligned with the software requirements. Support your claims with specific examples and evidence.]

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

For the JUnit tests for contact and contact service I tried to go through them line by line and find each line of logic that needed to be tested. I would write one passing test, and then I would write a failing test for the length, or a failing test for null. This helped me correct some logic by getting unexpected results. Sometimes I forgot to negate the Boolean value or I miswrote the logic so it was greater than 10 instead of 10 or greater.

### Task Entity and Task Service.

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

When I was writing the code for Task Entity and Task service I would have the IDE on one screen and the requirements on the other screen. This way I would see one requirement and I would translate it to code. This can be seen with lines 12 to 33 of Task.java and line 6 to 33 of TaskSerivce.java. If I would get confused by the requirements I would go through them slowly and write it down on the paper in a very straightforward way so it wasn’t wordy like the requirements were. Another thing that helped me get the requirements accurate was the feedback given by the teacher so I could see how I could improve the code or what I was missing. I also wrote the code for each assignment a little different and organized it differently each week. This way when It came time for the project I took the organization that I liked and rewrote the other two in the same way. This can be seen comparing the first submission compared to the final submission.

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

For the JUnit tests for contact and contact service I tried to go through them line by line and find each line of logic that needed to be tested. I would write one passing test, and then I would write a failing test for the length, or a failing test for null. This helped me correct some logic by getting unexpected results. Sometimes I forgot to negate the Boolean value or I miswrote the logic so it was greater than 10 instead of 10 or greater.

### Appointment Entity and Appointment Service.

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

[Explain how your approach to testing the Appointment Service was aligned with the software requirements. Support your claims with specific examples and evidence.]

When I was writing the code for Appointment Entity and Appointment service I would have the IDE on one screen and the requirements on the other screen. This way I would see one requirement and I would translate it to code. This can be seen with lines 8 to 49 of Appointment.java and line 5 to 25 of AppointmentSerivce.java. If I would get confused by the requirements I would go through them slowly and write it down on the paper in a very straightforward way so it wasn’t wordy like the requirements were. Another thing that helped me get the requirements accurate was the feedback given by the teacher so I could see how I could improve the code or what I was missing. I also wrote the code for each assignment a little different and organized it differently each week. This way when It came time for the project I took the organization that I liked and rewrote the other two in the same way. I formatted the other files based off of what I wrote for this initial submission. I like the way it looked and it seemed more organized and put together compared to the other two groups of files.

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

For the JUnit tests for contact and contact service I tried to go through them line by line and find each line of logic that needed to be tested. I would write one passing test, and then I would write a failing test for the length, or a failing test for null. This helped me correct some logic by getting unexpected results. Sometimes I forgot to negate the Boolean value or I miswrote the logic so it was greater than 10 instead of 10 or greater.

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

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

[Explain how you ensured your application’s source code was sound. Include examples of your unit tests to illustrate your points.]

I went through each file one at a time, line by line to write JUnit tests to ensure that most of the logic was covered by the tests. Each line of logic would have about three tests, one test for passing, one test for failing the length, and one for failing being null. There was one that had to be in the future which was AppointmentDate.

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

I ensured that my test code was efficient because I used the same format for most of the test code and only tested one variable at a time so if there was a failing point it was clear and obvious where it failed.

# 2. Reflection

## 2a. Testing Techniques

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

[Describe *each* of the software testing techniques you used. Describe their characteristics, explain why you employed these techniques, and use *specific examples and details.*]

I tried to cover most scenarios for the logic. This made sure that the logic worked as expected and didn’t randomly fail in deployment. I had the code for the tests and the code being tested split on my screen and I went through code being tested file line by line and then wrote the passing/failing tests to be sure that the code worked as expected. This took quite a bit of time because for every line of code being tested there was three times or more test code being written. It could take 30 lines of code that was being tested and grow the testing code to 100 lines or more.

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

[Describe these other techniques you did not use. Explain why you did not use them. Describe their characteristics using specific details.]

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

[Describe use cases where these techniques are applicable. Explain what you gain (or lose) by using each use case. Use hypothetical problems as case studies for each technique might help.]

## 2b. Mindset

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

[In acting as a software tester, to what extent did you employ caution? Why was it important to appreciate the complexity and interrelationships of the code you were testing? Provide specific examples to illustrate your claims.]

The mindset I had was to try and test all the logic. I used caution to try and not be redundant and not have my ego get in the way and just delete test code if the test kept failing and I couldn’t figure out why. It’s important to appreciate the complexity and interrelationships of the code I was testing so that I could understand what the logic was doing and all the scenarios that had to be tested. The bigger the tree of logic the amount of code grows exponentially.

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

[On the software developer side, can you imagine that bias would be a concern if you were responsible for testing your code? Provide specific examples to illustrate your claims.]

I could see bias in the code being a concern because of egos of developers. I feel that I didn’t have that in my testing because of how I went through line by line for writing the testing code, this way there wasn’t really any logic that I skipped out on.

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

[Why is it important not to cut corners when it comes to writing or testing code? How do you plan to avoid technical debt as a practitioner in the field? Provide specific examples to illustrate your claims.]

This is important because when corners are cut in any project it could cause anywhere from a minor inconvenience to death in major projects related to infrastructure. That’s why it’s important to test because one missing zero or comma could lead to a massively different outcome than what was expected.

References

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

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