Adrienne Sturgeon

CS-320

October 20, 2024

Project Two

1. **Summary**
   1. Describe your unit testing approach for each of the three features.
      1. To what extent was your approach **aligned to the software requirements**? Support your claims with specific evidence.

This project provided a great chance to utilize a test-driven approach with development. I made sure that my testing followed software requirements by making code that specifically tested fields of each object so that the field met Grand Strand’s guidelines. For example, the name string could not be longer than 10 characters and had to be not null. I created tests to ensure that exceptions were thrown when a field did not meet its requirements. Based on the name field, I created code such as the following:

A computer code with colorful text

Description automatically generated

Next, I did Junit tests to make sure that the field was behaving correctly when tested with invalid inputs:

A computer code with text

Description automatically generated

I did similar tests for the add, delete, and update methods of each service class as well. An example is shown here:

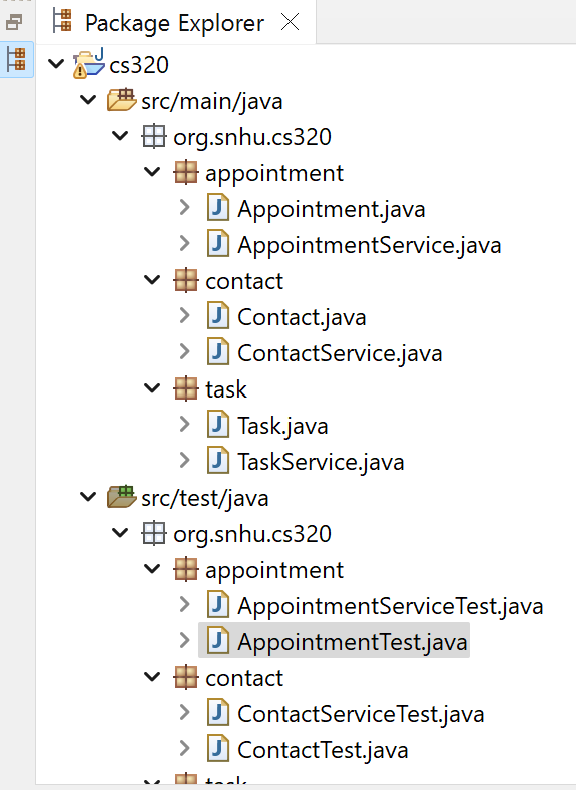
A screen shot of a computer code

Description automatically generated

I completed my methods and tests for every class and service in this same fashion, which made sure that code accurately followed the guidelines and software requirements that Grand Strand gave me on my rubrics. This was a wonderful way to anticipate the ways that every line of code might fail.

* + 1. Defend the overall quality of your JUnit tests. In other words, how do you know your JUnit tests were **effective** based on the coverage percentage?

I was able to make my Junit tests high quality by first creating a structure for the tests to be conducted in. This looked like the following:



This was a terrific way to make sure that my code was correctly organized and had the best dependency structure for Junit tests to be conducted easily. I also anticipated the majority of ways that code could malfunction based on incorrect field values. To do this, I used csv resources and parameterized tests to iterate over each entry and confirm the expected failure. This is an example:

A screenshot of a computer program

Description automatically generated

By conducting tests to ensure correct failures throughout the classes and services, including the add, delete, and update methods, I guaranteed high test coverage. Another example is shown here:

A computer code with text

Description automatically generated with medium confidence

Test success and failure in my code was part of how I confirmed over 80% test coverage for every line of code.

* 1. Describe your experience writing the JUnit tests.
     1. How did you ensure that your code was **technically sound**? Cite specific lines of code from your tests to illustrate.

I was able to create technically sound code by following the structure that I previously defined and following best practices when conducting Junit tests. I used an agile method of iterative testing to make sure that as my project progressed, I continued to meet the expected parameters. At each step in development, I completed Junit tests and was not satisfied until all tests passed, such as in this example:

A screenshot of a computer error message

Description automatically generated

I also used the singleton pattern to make sure that there was only one instance for each service class so that I could guarantee validation of all processes and fields. An example is shown here:

A screenshot of a computer program

Description automatically generated

By using the singleton pattern, I could easily ensure that code followed expected behavior based on Grand Strand’s prompted needs. This pattern can be seen in the first 30 lines of code of every service class in the project. Making dedicated classes for testing also ensured that I adhered to the concepts of object-oriented programming such as encapsulation, which guaranteed code that is both usable, easy to understand, and portable.

* + 1. How did you ensure that your code was **efficient**? Cite specific lines of code from your tests to illustrate.

The main way that I made sure my code was efficient was by using my existing code to prevent as little repetition as possible. For example, I used the thrown exceptions of my setters within my constructor so that I did not have to repeat the exceptions separately for both. Where I could, I used the setter methods within my constructors so that the thrown exception within each could be used by the constructor without having to retype it. This is shown here:

A computer code with text

Description automatically generated

The thrown exception code:

A computer code with text

Description automatically generated

The code shown above is a great demonstration of how the thrown exception is used within the constructor by borrowing from the setter’s code. If I had not done this, I would have to repeat the same thrown exception within the constructor itself, which is inefficient. Practices like this are part of my strategy in meeting expectations of the code in the most efficient way possible.

1. **Reflection**
   1. Testing Techniques
      1. What were the **software testing techniques** that you employed in this project? Describe their characteristics using specific details.

The software techniques that I used were derived from object-oriented programming and test-driven development. The first technique that I used was the singleton pattern. This made sure that only one instance of each service could exist within any given moment of the program running. This is important because data validation can be made rock solid when every instance is controlled and limited in a program. My test-driven development was based on the techniques of Junit testing. I did this by creating a test class for each functional class in the application, and then I completed data validation and process validation for all foreseen uses of the program. Each test class has the correct Junit annotation to specify the type of test being used, such as the parameterized tests and csv sources I have shown.

* + 1. What are the **other software testing techniques** that you did not use for this project? Describe their characteristics using specific details.

Additional software testing techniques that were not used in the project included integration tests, functional tests, and automated testing. This is because the code that I produced at this point is in an early form of development, so it lacks the complexity necessary for efficient use of more advanced testing techniques. Until this project reaches higher levels of complexity, simple unit tests are the best fit. As development progresses and high-level needs arise, I will implement the appropriate tests.

* + 1. For each of the techniques you discussed, explain the **practical uses and implications** for different software development projects and situations.

The techniques of object-oriented programming are very practical because they make sure that code creation is very consistent in terms of reusability. By making designated object classes and mirroring them with testing classes, it is easy for me to apply the same coding strategies across the entire application. Test-driven methods also benefit the development of code because they utilize the benefits of Agile methodology, an iterative approach to coding. By following this, I test the code as I create it, which means that every step of development can be tested to make sure it perfectly meets Grand Strand’s needs and is functionally correct.

* 1. Mindset
     1. Assess the mindset that 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 that I adopted while working on this project was both cautious and stepwise. I appreciated the complexity and interrelationships or the code I was testing by being careful to be perfectly consistent with each and every class. Any tests that were done within one class were also done in every other appropriate class, without exception. For example, I used the exact same testing strategy in my contact, task, and appointment classes when testing setters, as is shown here:

**A computer screen shot of text

Description automatically generated**

A computer code on a white background

Description automatically generated

A computer screen shot of text

Description automatically generatedThis is a fantastic way to show that the code is similar for every class, which highlights my attention to detail and consistency. Consistency is a perfect way to make code efficient because every testing strategy is used within every class.

* + 1. 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 own code? Provide specific examples to illustrate your claims.

I was able to limit bias in my review of code by being careful to follow the results of my Junit tests. If the test failed, I knew that additional work was necessary to make sure the appropriate standards were used throughout the project. Every Junit test must pass within my work before I consider it a success. This pass/fail strategy was perfect to make sure that every line of code got the attention and results it required. With every line tested, and every test passed, I guaranteed my test coverage was above the 80% expectation that Grand Strand set. An example of every test passing is shown here:

**A screenshot of a computer

Description automatically generated**

This strict approach to following the results of tests made sure that I did not test based on my own feelings regarding the success of my programming. Instead, I listened to the results of my tests and corrected where necessary.

* + 1. 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.

The rigorous approach to testing that I followed highlights the discipline with which I created my code as a software engineering professional. In order to apply tests evenly across every class and to make sure that they passed, it meant that I could not cut corners. For example, to make sure that the date field was tested appropriately, I had to adapt my testing strategy to test individual fail cases. This required more work on my part, but I was determined to make sure that I came as close to 100% coverage as possible. I show this in the following:

**A computer code with text

Description automatically generated**

Following this I tested each fail case:

A computer code with many colored text

Description automatically generated with medium confidence

If I had cut corners with this more difficult testing scenario, I would have left a hole in my testing approach. This hole would become a likely source of the application’s failure, or a bug that prevented the app from meeting Grand Strand’s needs. My commitment to the success of this project means that I could not accept any such failure. I appreciate you choosing me to ensure the success of your application.

Resources:

Chandel, M. (2019, May 23). *What are four basic principles of Object Oriented Programming?*. Medium. https://medium.com/@cancerian0684/what-are-four-basic-principles-of-object-oriented-programming-645af8b43727

*Dynamic Testing - Software Testing*. GeeksforGeeks. (2023, December 27). https://www.geeksforgeeks.org/software-testing-dynamic-testing/