Project Two Submission

Justin J Perez

SNHU

Course Number: CS-320

Desmond Springer

6/23/2025

Abstract

*Unit testing is an important part of building strong and dependable software. I have used unit testing before in Unreal Engine while developing my own video games, where it helped me test things more quickly and spot problems early. In this CS 320 project, I applied unit testing in a new way using Java, which I wasn’t very familiar with at first. Even though it was different from what I was used to, I was interested in learning more about it. Writing JUnit tests helped me make sure that the Contact, Task, and Appointment features worked as expected and followed the project requirements. This experience helped me grow as a developer by showing me the value of careful and consistent testing in any programming environment.*

# Summary

1. **Summary**
   1. Describe your unit testing approach for each of the three features.

*My testing approach for each of the three features was to test valid responses as well as invalid responses for each test. I did this by testing the best-case scenario such as the user typing everything correctly while also testing if the user types everything incorrectly. These tests and features are also modular so they can be quickly removed and swapped if necessary.*

* + 1. To what extent was your approach **aligned to the software requirements**? Support your claims with specific evidence.
       1. *My plan for approaching was to do exactly what the requirements stated at a bare minimum as well as improve it with my own ideas. For example, in the task test I am validating an exact requirement like ensuring the name is valid.*   
          A computer screen shot of a computer code

          AI-generated content may be incorrect.
    2. 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?
       1. *While a coverage percentage test wasn’t used my testing was very exact and decisive. The areas where I tested were some of the only places where an error could even occur.*
  1. Describe your experience writing the JUnit tests.

*While I always understood why it was important to do Junit test and immediately see the value as I have done something similar on Unreal engine. Junit testing to me at the start was extremely hard. I didn’t understand how to test at all, and it felt like there was a struggle to find information that would help with what I wanted to do. To be specific I was confused how to test user information without millions of different imports. I wanted to keep imports to a minimum for this assignment and try to do it as the assignment intended but it just wasn’t working. However, with your help I realized that I was missing understanding the assignment and realized I made it a lot harder for myself than I needed too. I then scrapped a lot of features that I mistakenly thought were required for this assignment. After that it became a lot easier. Like a lot of coding the hardest part is the beginning. When you start understanding with a small bit of repetition it becomes a lot easier. By week 5 I was able to implement it fast and effectively as most tests were starting to feel the same.*

* + 1. How did you ensure that your code was **technically sound**? Cite specific lines of code from your tests to illustrate.
       1. *I ensured my code was technically sound by testing the best-case scenario as well as the worst-case scenario. This will ensure no matter what the user does there will be code in place to assist the computer. In my task test file, you can see I test every possible scenario of what could happen. The user passes information correctly, the user passes information incorrectly, and the user passes information correctly but user errors happened so they would like to fix it. This code is a good example of how it is technically sound.*

A computer screen shot of code

AI-generated content may be incorrect.

* + 1. How did you ensure that your code was **efficient**? Cite specific lines of code from your tests to illustrate.
       1. *Ensuring my code is efficient is a little bit harder but I did this by using helper methods and reusing objects often. I also limited my code best I can while keeping everything modular so it’s not reliant on multiple processes that will do nothing for the program to work.*   
          A screen shot of a computer code

          AI-generated content may be incorrect.

# Reflection

1. **Reflection**
   1. Testing Techniques
      1. What were the **software testing techniques** that you employed in this project? Describe their characteristics using specific details.
         1. *The testing techniques I used in this project were:* 
            1. *Unit testing – Testing each method using Junit Testing*
            2. *Boundary value analysis – Testing input fields and limits 1 by 1 using 10 characters names*
            3. *Equivalence partitioning – I tested 1 valid vase, one invalid case, and 1 edge case for each field.*
      2. What are the **other software testing techniques** that you did not use for this project? Describe their characteristics using specific details.
         1. *The testing techniques I didn’t use in this project were:* 
            1. *System testing – Test the entire application as a complete system in 1 button*
            2. *User Acceptance Testing (UAT) – Test the software from the user’s perspective*

*While these would have been useful to test with the time permitted it would be a lot of extra work for something that isn’t required though they would be extremely useful.*

* + 1. For each of the techniques you discussed, explain the **practical uses and implications** for different software development projects and situations.
       1. *Unit testing: Unit testing is used in development the most often to catch bugs early that happen in development.*
       2. *Boundary value testing: Boundary value testing is used to test the size of a value. This is extremely helpful when working with numbers and you can avoid overflows in integers.*
       3. *Equivalence partitioning: Equivalence partitioning is almost always used with unit testing as it tests if there is a scenario for each scenario.*
       4. *System testing: System testing is useful when you want to ensure everything works in a deployed environment. This is another test that is commonly used in development all the time*
       5. *User acceptance testing (UAT): User acceptance testing is always used to ensure the program meets the clients’ requirements and gets their approval.*

# Mindset

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.
        1. *The mindset I adopted while working on this project would be a strategic one. While writing these tests I would always have to think ahead and think of every possible scenario before they happened. I would think about passing scenarios, failing scenarios, human error scenarios. I would also attempt to stop any injections that could happen with my code by stopping at a character limit even when not required.*A computer screen with colorful text

           AI-generated content may be incorrect.
     2. 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.
        1. *While I do think developer bias is unavoidable, especially when you write the code, I attempted my best to limit it by leaving comments everywhere. This will allow the professor to effectively grade and understand my code the recommend helpful improvements. I also left a code that will test if the program doesn’t work at all which covers every scenario.*   
           A screen shot of a computer program

           AI-generated content may be incorrect.
     3. 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.
        1. *Discipline is essential when developing software. Not having it means your software might be buggy or unreadable. While it might require a lot more thought and preparation it helps a lot when delivering a fast neat product. Cutting corners has a direct correlation into bugs and development being ruined which is why I would much rather have good code and ask for an extension like I have then deliver software that is bad. Toa void “technical debt” there is a couple of things you can do:*
           1. *Be ok to ask someone for help. This will get another person’s opinion on a product which might be all you need before you’re able to complete it.*
           2. *Study on implementation if unsure how to make something. Programming has been around for years now and there are millions of documentations. Every issue you had hundreds of others might have had the same issue. Spending time searching the internet for ways to properly complete your task. If you can’t find any information on the subject, you are most likely typing it incorrectly.*
           3. *Refactor, when necessary, even if it comes with a time loss. This is something that happened with this product exactly. My first attempt with this project on week 3 was awful, however doing that refactor even if it meant losing a lot of progress, I did make it much easier to make progress on other assignments as I now knew how to do it properly.*
           4. *Following naming standards. While this is one that is often overlooked it’s very necessary when in this field. Breaking away from naming standards causes multiple issues if you ever need to search for help or have other people view your code.*

*An example in my project is how I name everything in accordance with proper naming standards as well as refactored it after studying online how to implement better. I also asked my professor for help with this project which gave me a lot of insight into programming.*

A screen shot of a computer program

AI-generated content may be incorrect.

# Conclusion

*Working on this project helped me understand how important testing is when writing software. I learned that testing isn’t just about seeing if something works, it’s also about finding possible problems before users do. I made sure my tests matched the app’s requirements and thought carefully about how the code worked. I tried to think like both a developer and a tester to catch mistakes. This experience taught me to be more careful, to avoid bias, and to take testing seriously. These are skills I will use in the future to write better code and avoid problems later.*

References

***The 5th major version of the programmer-friendly testing framework for Java and the JVM*. JUnit 5. (n.d.).** https://junit.org/junit5/

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

**GeeksforGeeks. (2024a, June 19). *Static testing - software testing*.** https://www.geeksforgeeks.org/software-testing-static-testing/

**GeeksforGeeks. (2024b, September 19). *Introduction of junit*.** https://www.geeksforgeeks.org/advance-java/introduction-of-junit/

***Junit Test Framework*. Tutorialspoint. (n.d.).** https://www.tutorialspoint.com/junit/junit\_test\_framework.htm