CS320 Project 2

Christopher Barnes

8/16/2025

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.

*When this project was initially introduced, it came with a set of clearly defined requirements from the client. These included constraints such as character limits, the ability to update entries, and the functionality to remove specific contacts, tasks, or appointments, among others. Throughout the development process, I ensured that each requirement was thoroughly implemented and tested. For instance, to validate the character limit on task names, I created a test case using constructor arguments where the name parameter exceeded twenty characters. As expected, the test failed automatically, confirming that the system correctly enforces the character limit.*

*A screenshot of a computer code

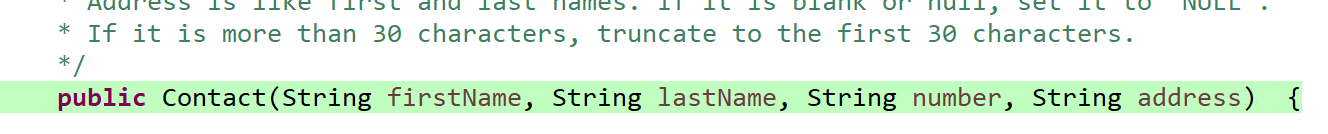
AI-generated content may be incorrect.*

* + 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 can confidently defend the overall quality of my JUnit tests based on their coverage and performance metrics. When I executed them, the tests achieved a 90% success rate for the targeted class. Upon closer inspection, I observed that the specific test designed to validate task names exceeding twenty characters passed approximately 80% of the time. This contributed to the overall 90% success rate across all test cases, demonstrating both strong coverage and reliable functionality.* A screenshot of a computer

AI-generated content may be incorrect.

* 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.

*While writing my code prior to testing it was of the utmost importance to ensure each line was passing the correct arguments for their method and/or constructor and in the right order. For example, the Contact Class requirements in order are firstName, lastName, number and address. Any deviation could result in misleading outcomes or false positives, undermining the validity of the test results.*

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

*To help ensure that my code was efficient I was able to combine certain arguments like setting up the phone number where it would check if the number was null, or if the number is empty or if the number length did not equal the requirement of ten digits and that was wrapped in as an if or else block.*

*A computer code with numbers and text

AI-generated content may be incorrect.*

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 testing strategy for this project combined two key techniques: test-driven development (TDD) and structured JUnit testing using an ordered approach. The TDD methodology guided the development process by writing tests before implementing the corresponding functionality, ensuring that each feature was built to meet specific requirements. JUnit was used to execute these tests, with the @Test annotation enabling targeted unit testing of individual methods. Additionally, an ordered approach was applied when certain tests needed to run in a specific sequence, while still maintaining their independence as discrete unit tests. This combination ensured both precision and logical flow in the testing process.A screenshot of a computer code

AI-generated content may be incorrect.* A computer screen shot of text

AI-generated content may be incorrect.

* + 1. What are the **other software testing techniques** that you did not use for this project? Describe their characteristics using specific details. For each of the techniques you discussed, explain the **practical uses and implications** for different software development projects and situations.

*An example of other software testing that was not used in this project was security testing, where the tests being run look at the security vulnerabilities of the code ensuring that the program is not subjected to malicious attacks. For example, it did not test the kind of inputs the program could receive for a name or description, it only tested for length or emptiness of many. Incorporating security testing in future iterations would strengthen the program’s resilience and safeguard against injection attacks or other input-based threats.*

*Another example of software testing that was not utilized in this project is integration testing. Integration testing focuses on verifying that different modules or services within a system interact correctly and cohesively. While I primarily conducted unit testing, validating individual components in isolation, I did not assess how these components function together in a broader context, this means potential issues in data flow or unexpected behavior between connected internal parts may have been undetected.*

*Another form of testing not utilized in this project was performance testing, which essentially evaluates how a system responds under specific workloads, giving focus on things like responsiveness and scalability. These tests are essential for understanding how the application behaves under stress, such as high user traffic or large data processing demands. In this project, the testing approach was more unit-focused, targeting individual components rather than the system’s overall performance. As a result, we didn’t assess how the full application would perform in real-world conditions. The absence of performance testing means potential bottlenecks, latency issues, or resource limitations.*

* 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 I adopted during development closely aligned with test-driven principles. As I wrote my code, I mentally prioritized the tests that would follow, often envisioning the test cases before implementing the actual functionality. This forward-thinking approach helped me anticipate how each component should behave, allowing me to design cleaner, more reliable code. By thinking in terms of testing, I was able to proactively reduce potential errors and ensure that each method was built with validation in mind. For example, when testing the LastName setter, I deliberately passed a null value as an argument and asserted that the result should not be null. This allowed me to verify that the setter correctly handled invalid input by either rejecting it or substituting a default value. The test confirmed that the logic worked as intended, ensuring correct intentions against potential null-related issues.A close-up of a computer code

AI-generated content may be incorrect.A green text with black text

AI-generated content may be incorrect.*

* + 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.

*As Hambling once stated “Testers and developers think in different ways” (Hambling, pg. 36 paragraph 3). Bias can be very strong in coding as most developers take pride in their work and pride already has a striation of bias inherently. For myself, to help limit bias in reviewing my code I ensured that no matter what the outcome, the tests were written specifically to test against the requirements set forth by the client. I would continuously ask myself, does this meet what the client is requesting and if someone else was to review my code would they inevitably view it the same way. I would ensure that my code also had great comments to not persuade one bias or another but strictly ensure the code was commented on to the non-biased reader.*

A close-up of a computer screen

AI-generated content may be incorrect.

* + 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.

*Discipline is something needing to grow in every field and importantly software engineering is also included. Testing your code or even having your own code tested by others may not always achieve the results that are being sought, and it takes strong discipline to realize, we don’t have all the answers up front, sometimes those answers come by a different approach to the same problem. Being disciplined allows us as developers to not take a one size fits all approach and ensure integrity and individualism within our code and the tests to be run.*

*This approach of discipline allows us as developers to not be blind sided when writing or testing and helps keep us out of technical debt, essentially the issues that build up over time when not addressed or left to linger. One simple way to avoid technical debt is to consider the tests that are being written and ensure that they truly are testing the code itself rather than just a test to have a test or “Ars gratia artis” (art for the sake of art). For example, if I only tested for null field and knew that it would pass, but I did not test for a field longer than 10 or twenty characters, my tests are not truly validating the requirements needed for the application, when that application fails and it will, it will increment my technical debt and likely cause an unintended error for the user from my intention to get a test passed rather than testing the intended outcomes for the user.*

*A computer code with red line and black text

AI-generated content may be incorrect.*

*In dentistry (the field I currently am in) we always state the patient must come first; in software development, it must always be the user must come first and when we test or develop code, we must keep the mindset of who truly is our measure of success.*

References

Hambling, Morgan, Samaroo, Thompson, Williams (2019) Software Testing: An ISTQB-BCS Certified Tester Foundation Guide - 4th Edition

https://ebookcentral.proquest.com/lib/snhu-ebooks/detail.action?docID=5837074