**Module 7-2 Project Two**

Mitchel Harmon

Online Campus, Southern New Hampshire University

CS-320-H7029 Software Test Automation& QA

Instructor Jonathan Norman

December 09, 2023

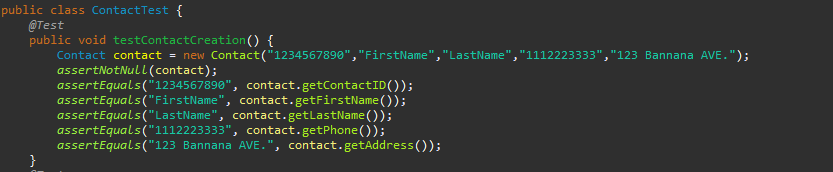
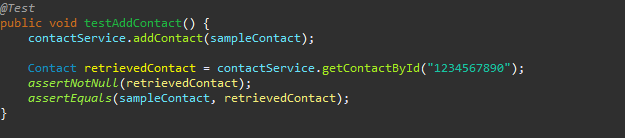
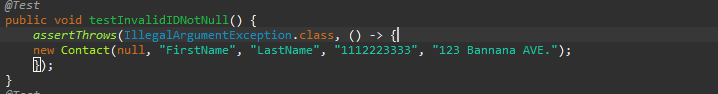
**Module 7-2 Project Two**

**Summary**

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.
   1. My approach met every software requirement such as with the Appointment Class requirements, the appointment object was required to have a unique appointment ID string that could not be longer than 10 characters. The appointment ID also could not be null and could not be updatable. Another example is with the Contact Class requirements, the contact object required to have a unique contact ID string that could not be longer than 10 characters. Also, the contact ID could not be null and could not be updatable.
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. My JUnit test covered every requirement for each service by aligning the tests to the requirements and ensuring each method performed as expected on each service model.

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.
   1. I ensured technical soundness by adhering to best practices, such as descriptive test method names.
   2. I also ensured comprehensive coverage, meaning that the tests cover various scenarios, including input validations and expected behaviors.
2. How did you ensure that your code was efficient? Cite specific lines of code from your tests to illustrate.
   1. I ensured efficiency by having each test method be self-contained and focused on testing a specific scenario. Also, there is no excessive duplication or unnecessary complexity in the tests that could impact their efficiency.



**Reflection**

Testing Techniques

1. What were the software testing techniques that you employed in this project? Describe their characteristics using specific details.
   1. I employed Unit Testing as my software testing technique. Unit testing is a software testing technique where individual units or components of a software application are tested in isolation to ensure that they function as expected.
2. What are the other software testing techniques that you did not use for this project? Describe their characteristics using specific details.
   1. A software testing technique I did not use was system testing. System testing involves testing the entire software system as a whole. It verifies that the entire application, including all integrated components, meets the specified requirements.
   2. Another software testing technique I did not use was integration testing. Integration testing focuses on verifying the interactions and interfaces between different components or modules within a system. It ensures that integrated components work together as expected.
3. For each of the techniques you discussed, explain the practical uses and implications for different software development projects and situations.
   1. Unit testing is ideal for testing individual units in isolation. It helps in catching bugs early, facilitates refactoring, and ensures that each unit of code works as intended.
   2. System testing verifies the entire system's functionality against defined requirements. It includes functional, performance, security, and usability testing.
   3. Integration testing validates interactions between integrated components/modules. It ensures that various parts of the software work together seamlessly.

**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. While working on the project I paid meticulous attention to every aspect of the software, ensuring that all functionalities and scenarios were thoroughly tested. It is important to appreciate the complexity and the interrelationships of the code because complex code might hide potential defects or unexpected behaviors and a change in one part might affect other parts, leading to unintended consequences.
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. Following industry best practices and coding standards helped to minimize personal biases. By reviewing the code against established conventions, consistency and reduced biases. Confirmation bias is when developers might unconsciously seek to confirm their belief that their code works as intended. This could lead to overlooking potential issues or flaws in their implementation.
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. Being disciplined in the commitment to quality ensures reliability, reducing the likelihood of bugs or unexpected behaviors. This reliability directly impacts user satisfaction, leading to a positive user experience. I plan to avoid technical debt by following established coding standards and best practices ensuring consistency and readability, making the codebase more maintainable.