**Project Two, Summary and Reflections Report – Module 7**

Jime Balvin

Southern New Hampshire University

CS-320: Software Test Automation QA

Federico Bermudez

February 23rd, 2025

1. **Summary**

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

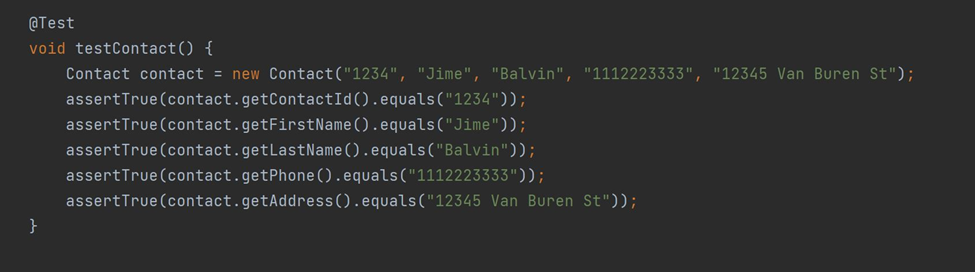
* 1. To what extent was your approach aligned to the software requirements? Support your claims with specific evidence.
  2. Defend the overall quality of your JUnit tests. In other words, how do you know your JUnit tests were effective based on the coverage percentage?

The unit testing approach implemented verifies that each feature behaves as expected following the given constrains and maintains functionality under normal and edge-case scenarios. For example, one common requirement was that fields containing contact, task and appointment IDs were not to exceed 10 characters and could not be null; the unit testing in place validates that any input that doe not meet these specific constrains will be rejected. All major functionalities were tested against the requirements and it is evident on the coverage percentage obtained after the coverage analysis. This minimized the risk of undetected defects as most of the codebase was exercised during testing. I addition, the tests were implemented keeping in mind the need for rapid execution, maintainability and clarity to achieve high test coverage.

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

Including tests that checked validation for valid inputs and edge cases ensured that the code was technically sound. This confirms once again that the system is behaving as expected. For example, with the use of assertions, the tests ensures that the Contact object is properly created and its fields are set and retrieved as expected:



To ensure the code was efficient, the tests were focused on the key behaviors without redundant checks and reusing variables across the them. For example, the apptDate object was initialized once in the constructor ensuring the instance could be used in every test avoiding repetitive code:



1. **Reflection**

**Testing Techniques**

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

Unit testing was the main technique used; it focused on testing the classes independently to ensure they worked as expected. This helps prevent regressions and ensures correctness of individual components, catches issues before they become too complex and ensures any changes to code don’t introduce new issues which improves maintainability of the code. Boundary testing was applied to all. This testing techniques allows to check how the system behaves with inputs that are on the boundary of acceptance values. It is essential in systems that rely on strict data validation or where the field have defined limits; it ensures that the software behaves correctly when encountering maximum, minimum or out of range inputs and helps maintain the application’s stability under different input conditions. Exception handling testing was another technique applied in order to handle invalid or erroneous input without producing unpredicted and unsafe results. It verifies that appropriate error messages are raised when the input doesn’t meet the expected format and avoid the system to fail silently. The technique provides clear messages that guide troubleshooting and help maintain system stability and data integrity. Integration testing was implemented in milestone two and three. The technique was applied in order to verify that the components worked together and communicated as expected. It is essential in systems where different components relay on each other to perform tasks. Some of the tests that were not implemented: User acceptance testing involves testing the application with real users to ensure it meets their expectations and aligns with business needs. It is usually done after functional testing has been implemented. It is useful to test that the application solves real-world problems and doesn’t cause confusion while interacting with it. Performance testing shows how the system performs under normal and extreme conditions. It ensures that system remains responsive as it scales. Finally, Security testing is crucial in this case since the application will be dealing with sensitive data like contact information and appointments. It will ensure the data is properly encrypted and the application is protected from vulnerabilities and attacks. This will help blocking unauthorized users from accessing and handling said data and keeping integrity.

**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.
  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.
  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.

Testing requires attention to detail and organization, which is the mindset I adopted throughout the project. For example, carefully checking that boundary fields were tested to catch any issues that could cause unexpected behavior later on. To limit bias, I made sure to stay focused on the given requirements and did not assume the code would behave as expected; having an open mind and implementing any feedback. The importance of having a strong foundation withdraws the “convenience” of cutting corners. The goal is to avoid propagation of issues that can become bigger and more expensive later on, and cutting corners only sets a difficult path in exchange for an instant gratification. To avoid technical debt, I plan to stay focus on the fact that the priority and commitment is to deliver high quality code by following standards and writing comprehensive tests to prevent defects propagation.

**References**

* Hambling, B., Morgan, P., Samaroo, A., Thompson, G., & Williams, P. (2019). *Software testing : An istqb-bcs certified tester foundation guide - 4th edition*. BCS Learning & Development Limited.
* Boni Garcia. (2017). *Mastering Software Testing with JUnit 5 : Comprehensive Guide to Develop High Quality Java Applications*. Packt Publishing.