The testing strategy was in line with the software requirements since it tested the needed methods that were implemented in the base code. According to the laws, the first and surname names in the contact class may not be more than 10 characters long. ContactTestJUnit developed the assertion "Assertions.assertThrows(IllegalArgumentException.class, () -> new Contact("ABCDEFHIJK", "Jeff", "Roary","8017101127", "1015 Reptile Lane");" to test for overly long input. According to the task class standards, the contactid could not be longer than 10 characters. As a result, the ContactTest JUnit now includes the "void updateContactIdTest" function, which raises an incorrect parameter flag if the task id is larger than 10 characters.

JUnit tests are improving with time. The courses' resources were quite beneficial. The coverage percentage in the contact test was higher than in the task test. However, the coverage rate for both functions was much higher in the service tests. The fact that the tests gave a positive percentage allowed me to determine how effectively the functions were covered. I intend to use them more frequently in order to ensure that each statement is utilized in at least one test case.

The software testing methods I used for the milestones would fall into the black box and white box categories because they were all structure- or specification-based.According to Knovel, black box techniques "derive test cases directly from the specification or from some other kind of model of what the system should do" (Hambling et al., 2015). Black box approaches used to test for both valid and wrong inputs include equivalence partitioning, decision tables, and state transition testing. the present state, outputs produced during testing, use cases created from test cases, and test boundary values. Structure-based testing was extensively used in the coverage tests that investigated the components and if-then statements. It's used to break down tests into testable parts.Structure-based techniques include statement coverage, path coverage, and branch coverage.Structure-based testing techniques "are used to explore system or component structures at several levels" (Hambling et al., 2015).

My approach to this project was analytical, exploratory.. I proceeded with caution, completing extensive study, testing, and trial-and-error runs while watching tutorials. Given how much it influences the product's quality and performance, understanding the code's complexity and links is critical. For example, when I first performed my coverage tests, the coverage% was low. My final product's coverage percentage, which included more tests and covered a bigger chunk of the code, exceeded the minimum 80%. The additional testing ensured that the code was technically solid and that the final product was of the highest standard.

I tested everything repeatedly, regardless of whether I was certain it would function well or not, in an effort to reduce prejudice in my assessment of the code. We all know what occurs when we presume, therefore I attempted to develop hypotheses rather than assumptions. Therefore, if I were in charge of testing my own code, I could see how bias may be a problem. For instance, I may have missed the fact that the ID test wasn't working because of a missing line of code if I simply tested the function that verified that the length of the first name was no greater than 10 characters. Testing both valid and invalid input as opposed to simply one or the other also helps to reduce bias.

It is critical for a software engineer to preserve and "advance the integrity and reputation of the profession" by a disciplined attention to quality (Software Engineering Code, 2018). It is vital to avoid taking shortcuts to avoid affecting the finished product's quality and performance. According to the software engineering code of ethics, "software engineers shall act in a manner that is in the best interests of their client and employer while remaining consistent with the public interest" (Software Engineering Code, 2018). This statement highlights a software engineer's responsibilities. In order to reduce technical debt as a practitioner in the field, I aim to adopt agile development approaches that often test code, always strive for high-quality, demonstrable software production, and maintain open communication between the client and the engineers.

References

Hambling, Brian Morgan, Peter Samaroo, Angelina Thompson, Geoff Williams, Peter. (2015).

Software Testing - An ISTQB-BCS Certified Tester Foundation Guide (3rd Edition) -

4.1.1.1 The Test Development Process (K3).

BCS The Chartered Institute for IT.

Retrieved from

<https://app.knovel.com/hotlink/pdf/id:kt00UC2IS4/software-testing-an-istqb/test->

development-process

Software Engineering Code. ACM Ethics. (2018, December 19).https://ethics.acm.org/code-of-

ethics/software-engineering-code/