Brittany Winters

7-2 Project Two

Southern New Hampshire University

My testing approach in relation to project two was heavily aligned to the software requirements given as I attempted to correctly test all of the requirements listed for each class including Contact, Contact Service, Task, Task Service, Appointment, and Appointment Service.

For example, The Contact Class requirements were to:

1. Ensure that the contact object shall have a required unique contact ID string that cannot be longer than 10 characters. The contact ID shall not be null and shall not be updatable.
2. The contact object shall have a required firstName String field that cannot be longer than 10 characters. The firstName field shall not be null.
3. The contact object shall have a required lastName String field that cannot be longer than 10 characters. The lastName field shall not be null.
4. The contact object shall have a required phone String field that must be exactly 10 digits. The phone field shall not be null.
5. The contact object shall have a required address field that must be no longer than 30 characters. The address field shall not be null.
6. To ensure that these requirements were met, I set exceptions to be thrown if these requirements were challenged.

Referring to the requirements, the address field can be no longer than 30 characters. If a user attempts to enter more than 30 characters, the program will only accept the first 30. This piece of code also prevents the user from leaving the field blank:

if (address == null || address.isBlank()) {

this.address = "NULL";

} else if(address.length() > 30) {

this.address = address.substring(0,30);

} else {

this.address = address; }

When we run a Junit test by coverage we can see the overall success rate of our test by percentage. Having a higher percentage and mostly green bar tells me that the majority of test were successful while leaving me to focus on the red. My personal tests could be refined more in order to achieve a higher success rate.

One of the ways I ensured my code was technically sound was by using an Object Oriented approach. For example, using Encapsulation with the Contact Service and Task Service objects protects them outsiders changing their state.

I ensured my code was efficient by using appropriately named variables and functions. For example, the variable names match their purpose, “private String address” holds an Address. I can practice being more efficient in the future by incorporating more appropriate data structures, and also reducing “if-else” statements.

For the final project two, I have focused almost exclusively on the “Exercise” and “Verify” steps of the test lifecycle. During the exercise stage, the test interacts with the system under testing, and gets some outcome from it as a result. For example, in the Appointment Class, we interact with the system to return variables such as the ID and description. While in the verify stage, we verify that the outcome from the system being tested is the expected value using assertions. Test verdicts are created from the result of the verification stage.

Unit testing is the process of testing small sections also called “units” of code. Unit testing was used in each milestone in order to check that the proper operations are being performed. Test coverage is also a technique that was implemented for each of the milestones. Test coverage is used to find untested parts of the system under testing, it is the rate of code in the system that is exercised for any of their test.

While working on this project, I developed a more detailed oriented mindset. Acting as a software tester, caution was employed during each unit test. It is important to appreciate the complexity and interrelationships of the code being tested because it can make debugging an easier tasks to complete. For example, when testing the Appointment Class, we need to deeply understand all of the functions in order to test them.

As a developer, I attempted to eliminate biases when testing my own code by testing more than once, with different inputs. For example, we know that the Add a Task Service has strict limitations. Instead of just testing the maximum character requirements, I test what happens when the input is beyond what is expected. For example, and input of all numbers, the same character repeated, etc.

It is extremely important to be a disciplined software engineering professional that is committed to quality. Cutting corners when it comes to writing or testing code can be detrimental to an organization and their potential customers. Cost is also a huge consideration when it comes to cutting corners. The cost to fix an error can be extremely high especially if the product is close to being deployed or already live. For this reason, I plan to avoid technical debt by working in a organized and detail oriented manner.

References

García, B. (2017). *Mastering Software Testing with JUnit 5*. Packt Publishing.