CS 320 Project Two

Tyler Pickard

April 12, 2021

I have no prior knowledge to JUnit testing, so this was a new experience for me. I found the provided video helpful, especially for the Contacts and Task class tests. Writing JUnit tests for the first time was challenging, but I plan to learn from any mistakes that were made.

In the Contacts class, the requirement was to establish contact information with a set length limit and the information could not be null. The tests I wrote were able to directly test that this criteria was met. I was able to write a test to make sure that when the guidelines were followed that the contact was created. The next set of tests were written to test the throwback error if one of the information slots exceeded the character count limit. These tests were run for each and they all came back as passed. Then the contact information was tested to return the error message if any of the information was null. These tests passed as well. The Task class was set up in a similar fashion and returned all passed tests. All these tests being passed with coverage show that the code was technically sound.

The Contact Service and Task Service classes were more challenging to test. I did not have the same confidence in my tests for these two classes that I did in the previous two. For these, an array list was set up to hold the contacts and the task information. I set up a test to add an item to the array list and another to remove for editing. I would like to gain more feedback on these tests and learn from the experience.

The tests for all four classes proved effective. I was able to run the JUnit test with coverage to see the detail behind the testing. I can say with the utmost confidence that the Contact class and Task class had very effective and efficient testing. The testing for the Contact Service and Task Service was more difficult, but I do believe the tests were able to cover what was needed. I do feel that these Service class tests could be improved to ensure the highest quality.

Overall, the code was written soundly and efficiently. For example, when writing the phone number requirement for the Contacts class, one if statement was able to contain the null, greater than 10 characters, and less than 10 characters. Combining requirements into one statement allows for a cleaner line of code and limits the amount of lines. An Agile rule I try to follow is to write code to the requirement and do not go far beyond that. Any extra was not asked for and may not provide the value to the reviewer / client that I feel it does. This allows for the code to be written efficiently.

Focusing on software testing techniques allow for better tests to be created. It is important to cover all aspects of a class that you are testing. Using common techniques can allow for more efficient testing that ensures tests are challenging the code written to cover all requirements for an application. Running a sufficient number of tests can be difficult, so following common techniques can allow for better coverage.

The most common technique that is used in my tests is state transition testing. This is a testing method were the tester changes the input to test for different scenarios and outcomes. An example of this is the tests that are run for the Appointments class to test the id variable. I first set up a test that enters an id that is less than 10 characters and is not null. This input is expected to pass. Next I run the test with an id that is longer than 10 characters and expect this to fail and return the error message for invalid id. Finally, this is put to the test by entering null to get the same error message. This is an example of how changing the input can test for different reactions from the code.

Testing the input for each variable could also be an example of error guessing. Error guessing could be relevant in this scenario because we are testing with an invalid input that we are expecting to return an error. For this type of testing, the tester really needs to understand what will cause an error, so previous experience with similar code is important to have. This type of testing is really meant to be directed as trouble areas and make sure that errors are exposed.

Testing the limit for variables such as an id or description character limit is a type of boundary value test. These tests have been set up to make sure that any input of the variable that does not fall within the given boundary returns an error. To improve on boundary value testing, more tests may be needed to push along the boundaries set up. Currently, the tests are set to only test one invalid variable. This may not be sufficient for this technique.

One technique that is not used is a stat transition table. This technique is used for setting up a variable that allows a certain number of tries before another action takes over. An example would be a password to enter an application. After a set number of tries and fails, the application or user would be locked out. To this point in our application building, this feature has not been necessary.

Another technique that has not been used is a decision table. This is used for a “cause and effect” scenario. This would be used to map out all possible scenarios for an input. This type of testing is meant to test out that all scenarios are covered and function properly. An example is the use of a submit button and testing out all scenarios based on the form that has been submitted.

Equivalence class partitioning is another technique that has not been used in the tests so far. The idea here is to divide data into partitions and test the variables in that range. The ranges can then be identified as either valid or invalid. An example would be looking at a set of number and having the range 1-10 and 20-30 as valid, but 11-19 as an invalid set of entries. This type of testing has not been necessary.

Using testing techniques allows the tester to design the best test cases possible for the scenarios provided. With these techniques, tests can be guided to fit any application and control errors to the best of the availability of tests.

Whenever new code is written there is a certain amount of caution that goes into deploying the new code or before moving onto the next section. Testing allows the programmer to implement a safeguard to make sure that the code is running as expected and the errors would also work as expected. When working with large programs, it is not possible to cover all possibilities or to even think of all of them. For a smaller scale project such as this one, we were able to develop tests to make sure the contacts, appointments, or services were able to be created and fit within a certain guideline. The tests were also put in place to make sure that an error message would pop up if user input did not fall within the guideline. For example, if a name of a contact was left null, then an error would appear to let the user know that the entry is not valid. Putting tests in place to cover not only the code functioning properly, but also to return expected errors allows for safeguarding on new code.

I can see how a programmer could be biased if they are in charge of creating tests for their own code. They could set up “easy” tests or tests that they know will return the results they want and avoid more challenging areas. This mentality would likely lead to failure of the project and cause the team to run into issues down the line. To avoid bias, I wrote my tests to the syllabus standards and tested for what was asked. I see the syllabus in a similar way that on an agile team the testers would provide what exactly the software should accomplish. The programmer then writes just enough to accomplish those goals or function. This is the path I took on this project and writing my tests to cover all functions allowed for no bias.

This has been a great exercise to learn about building a program that is able to function and how to build it without cutting corners. For any large project, it is best to break the large problem into many small problems and then frequently test along the way. This was my introduction to using formal testing opposed to just running the program frequently. Building the tests can seem like extra work that may not always be necessary, but it is a nice safeguard to have to avoid running into issues down the road. It is best to fail fast and fail often. Creating tests helps in that regard. It would be easy to skip writing tests, but especially for large programs they serve a very important function. I’m glad to have gotten an introduction in testing that can help me on my path to becoming a software engineer.

Sources:

Rungta, K. (2021b, March 26). *Software Testing Techniques with Test Case Design Examples*. Guru99. https://www.guru99.com/software-testing-techniques.html