Final Project

Adam Benoit

Southern New Hampshire University

CS-320-T3325 Software Test Automation& QA 23EW3

2/19/2023

For this project we were given requirements to implement into to code and to run JUnit tests on what we coded. To create the code, I broke the requirements down into their induvial needs and implemented them. For example, this can be seen in the class Task in module four milestone. Specifically, you can see from line 8 to line 22 that I set each variable to the required length given and also if it was null to throw an illegal argument. I then tested this in TaskTest class from lines 23 to 57, testing each for an incorrect amount of characters and in the case of the variable being null. This is how I proceed with each and every requirement given. After I separated each requirement and wrote it into code I would write a test for it to make sure it was working correctly and would make adjustments before moving onto the next requirement if needed.

I believe my overall quality of JUnit tests to be relatively effective due to both contact service and task service being over 80% covered by the tests. I made sure that my code was following the requirements given in the rubric. By writing test cases based off the requirements I was able to check my code for the client’s required functions and was able to see to what percentage my code followed those requirements. A low percentage would of told me that I was either miss coding something or if I had unneeded code that was just clutter and allowed me to adjust until I had a high enough percentage of quality code.

To make sure that my classes were as technically sound as possible I used the @test for declaring test methods. I also used proper naming convention that can be seen in each of my test cases that before the test it can be read what the test is for. For example, looking at task service test class on line 26 it can be seen that the test name is testTaskServiceDelete telling anyone who reads the code that the test is going to test the delete function of the test service class.

I was able to ensure my codes efficiency through the testing process in a couple of ways. One way was that when the tests were run with coverage, I could see what lines of codes were being tested and which lines were not being reached. I also could tell which test cases were failing or performing how they were supposed to. This would lead me to make adjustments to the code to improve the quality and make the code more efficient. For example, the first time I ran all the tests some of the tests started failing. Then if I removed tests and ran the test individually, they would pass. This lead me to double checking the code and I was able to see red highlight over sections that I had miss coded. I was able to fix this and make sure that when everything ran together it was efficient. One specific example of my tests check the efficiency of the code would be line 15 to 24 in contact service test class. This shows the code was efficient able to add a new contact to the array and effective assign it a new Id number while also taking in the input on line 17 for the other variables needed to create a contact in the array.

For each of the classes for this project I used Junit tests to test my codes functionality. I did this by first writing the code each week for the main classes for each milestone following the requirements given very carefully. After doing this I created Junit tests in a test package for each milestone. The Junit tests would be made up of multiple test cases. The test cases would have predetermined input and place it into the code and depending on what is expected a true or false the test would pass or fail. For example for the first milestone we had to create a contact ID and it had to be no more than 10 characters. I had a test set up to accept an ID number that was less 10 and it passed letting me know that my code was functioning correctly when the conditions were correct. But I also had to check that when more than the limit was put in that the code would properly reject the ID number. So I wrote a similar test with the input for ID number being longer than 10 characters with it set to throw and illegal argument and the test passed. I tried to follow this method closely with all requirements given by testing successful cases but also by testing the code for wrong inputs to see if the code would behave in the manner expected. The testing method I think my approach most matches would be black box and functional testing since I tested that the code functions as intended and from the examples listed above.

I think a couple testing technique that I did not use for the milestones was stress, load, and stability. These tests would have tested the code by seeing how much it could handle at once. It would do these by applying loads to the application/code to see how it functions as it fluctuates and how it performs during the increased stress on it. I think that the milestones are very early parts of an application and wouldn’t greatly benefit from testing this was just yet. I could have also used JBehave instead of Junit testing. I think for the milestones we were testing the Junit testing strategy was a better option than Jbehave. Jbehave seems to be used in larger projects and tests the behavior of code from the end users’ side instead of individual pieces like the Junit testing offers.

For different projects I believe that all the above I talked can be used to help in the testing and QA of code/applications. Junit tests I enjoyed getting to use and test certain aspects of my code. This will allow me in the future to write some code and be able to test what I wrote right away to see how it functions. Rather than waiting to the end trying to run a project and getting a lot of errors. The constantly testing would allow me catch errors earlier and fix as I go along. I believe performance testing will also offer practical use when testing the strength of the service an application or code is set to deliver. This testing can help predict crashes from high loads or to much stress on the system.

The mind set I had to take for this project was different than what I usually had to use in past projects. Since this project had me writing and then testing the code I wanted to do my best to adopt a mindset that separated myself when performing both roles. Meaning when I was writing the code I was writing the code and not testing the lines as I went along with Junit tests. When I was doing the Junit tests and testing the code that was what I was doing. I found that it helped because it helped kind of see it as not my code and allow me to eliminate some bias I would have reading and testing my own code. I applied caution as much as I could as testing would permit with out handcuffing myself. I made sure that the tests were getting and passing a majority of the code. It is very important to appreciate complexity and interrelationships of the code you write or review on many levels. For example when I was testing classes with their respective service class it was important to understand how they connected to each other to be able to properly test them. If I did not understand how they interacted with each other I would not have been able to properly test them and would have gotten a lower percentage.

As I mentioned in the previous paragraph, I attempted to eliminate the bias I would have when reviewing my code by separating my role into two. When I was coding the classes that was what I was doing and when I was writing junit tests and reviewing that the requirements were met that was what I was doing. I would also take a 10 or 15 minute break before switching the role I was performing. This allowed me to separate myself as the coder and then as the QA/tester. I believe that there should be caution and concern when ever on the software developer side that myself or someone else is responsible for testing their own code. Even though it may not be intentional bias would find a way to creep in. Even in this project and with the steps I did to try and eliminate the bias, I still noticed it creep in time to time. For example I did get over 80% coverage but it would be more ideal to have a higher percentage and bias more than likely creeped in and I though it was good as is to submit.

Discipline in any line of work is highly important and cutting corners is never a good business motto. I believe that at some point that it will end up coming back and biting you. For example, in coding software cutting corners may lead to future costs in the future to release bug fixes. Now your paying coders to edit already released code, you have upset clients and the small amount of money you saved at the beginning to cut a corner is now coming back 10x the amount or more. For myself going forward to avoid technical debt my plan is to always to proceed with caution and discipline with everything I do related to my career. For me better quality is better than a better timeline. For example, I would rather tell a client we need to extend the release time by a week or two rather than skipping testing and releasing a product to the client that wasn’t ready and hoping for the best. I also plan on properly testing all functions of code before moving on to the next phase rather than trying to juggle the next phase of coding with the last phase which could possibly now lead to two phase of the development being of lower quality than it would have been if I had just taken some extra time to fix the first one.

References

Garcia, B. (2017). Mastering software testing with junit 5. Packt Publishing.

Hambling, B., Morgan, P., Samaroo, A., Thompson, G., &amp; Williams, P. (2019). Software testing: An Istqb-Bcs certified tester foundation n guide - 4th edition. BCS, The Chartered Institute for IT.