**Grand Strand Systems Summary & Reflection**

**CS-320 Software Automation**

**Justin Hancock**

**10-14-2023**

My approach was aligned with the software requirements of these programs based on the requirements given in the instructions for this assignment. Below are the requirements given for the task service program given back in module 4. Under Task class requirements, I needed to create a unique ID string that is no longer than 10 characters, not empty, and unable to be updated. I also needed to create objects’ name and description with string fields no more than 20 and 50, respectively, and cannot be empty. My task test JUnit test accounts for all these requirements except the ID. I created a test that accounts for the length of the ID, but the unique ID, empty ID, and unable to update are all accounted for within the Task class. I did not need to run a test to validate these pieces of the requirements. The task service requirements listed options to add, delete, and update tasks based on the unique ID. These requirements are all tested and validated in my task service JUnit test.

After reviewing the coverage percentage for all the files, the total coverage percentage is a combined 80.7%. I’m not 100% sure what changed with the coverage percentage from the individual programs as opposed to combining them into one program, but the coverage has decreased for each test file. I do know that my tests are still effective because the requirement for the program was to be above 80%, and although close, I am above 80% total. The tests for contact and contact service run the highest percentages at 100% and 96.8% respectively. Although the overall percentage is low, I feel like I have developed quality tests for each of these test files.

Below is a screenshot from the contact service test that shows an example of how my code is technically sound. By running the coverage JUnit test, you can see below that most of this section of code is covered by the test. This coverage does not mean there are not issues within my code, but it helps to provide confidence that my code is technically sound.

A screenshot of a computer

Description automatically generated

Below is a screenshot from the contact program that shows an example of how my code is efficient. On the left-hand side of the screenshot, you can see the amount of time it took to run the test and provide feedback on the coverage. The times provided tell me that the program is running efficiently; however, the most important feedback from a JUnit test is focusing on accuracy and not efficiency. We can always address performance issues on real performance which this test does not technically provide. We can combine JUnit tests with other testing methods and tools to help us gain a better understanding of the efficiency of the program.

A screen shot of a computer

Description automatically generated

The most common testing technique that I use throughout the program is data validation and functional testing. I think of the impact that errors can cause within a method or a loop whenever I am writing a program. I think that I do a good job of validating information before moving on to other sections of the program. An example of this throughout these programs is making sure that requirements like length of strings are properly validated within the method. I have learned many new techniques now that I have JUnit testing in my pocket of tricks. I think they will become valuable to me throughout my programming career. Regression testing is something that I used whenever I made changes to the code to make sure that I didn’t create any new bugs. As all of us use it on a regular basis, white box testing allows the programmer to look through the code and find logic and other hidden errors.

A few tests that I did not complete with these programs are load, stress, usability, security, compatibility testing. There are so many ways to test a program that one person cannot use them all to test. I guess you could, but the tests work much smoother to have a team working on them collectively. Some of the characteristics of the tests that I mentioned that I didn’t use are testing the systems performance based on workload, subjecting the program to extreme conditions, evaluating user-friendliness, identifying vulnerabilities and weaknesses in the security, and making sure the software works on multiple platforms, devices, or browsers.

Each testing technique has a time and place to be useful, but the testing techniques when used together can be the most effective. Projects can vary in types of projects and size which all require different factors to be considered when developing the project. Each factor within a project like scalability, security, and the different types of projects require certain tests to be run collaboratively to ensure a great product.

I initially had a difficult time changing my mindset from developer to tester, especially being that I was the developer and tester within each project. After writing the code for the program in week 3, I went straight into writing the tests, and I don’t feel like my mind really switched properly to what I was doing. I spent a lot of extra time on the first project to make sure I was covering everything that needed to be covered in the JUnit tests. After reviewing back over these projects for each week, I probably could have added more tests to have further coverage. I didn’t think with as much of an open mind as I should have to properly test everything. My closed-mindedness was focused more on the requirements and making sure they were met and tested than some of the other aspects of the program.

When looking at the relationship between developer and tester, I think it is extremely difficult to do both without some bias within the program because you as the developer want the program to work the way it is intended and to not return any errors. From the testing perspective, you want to find any potential bugs with as many tests as possible to make sure you have a properly working program. The intention is not to single someone or a team out for not doing something correctly, but it can hurt feelings when hard work has been applied to something to have it returned for corrections. I think that I looked over my programs with bias towards myself, and I will have to learn how to make that adjustment if I am ever in a situation where I must develop and test my own code.

It is important not to cut corners and to be disciplined when writing code because you want to have the best product that you can have when the job is said and done. Writing and testing in a disciplined manner become the highest standard to provide such a product to the user. Throughout the stages of the software development lifecycle, testing is completed along side the developmental processes to make sure the program will run without error, protects any data within, and has a secure foundation. Using the testing techniques mentioned throughout this paper and sticking with the coding standards in development any programmer can achieve the creation of a great product.

**Citation**

GeeksforGeeks. (2023). Software testing techniques. *GeeksforGeeks*. https://www.geeksforgeeks.org/software-testing-techniques/