Project 2 Summary CS320

Jordan Carver

October 14th, 2022

**Summary**

With my testing approach my goal was to ensure the three features met the system requirements as close as possible. An example of these was with the ID variables. Specifically the first name and last name having a max character span of 10 based on the system requirements. Therefore when it came to creating the ContactTest file by using an IllegalArgumentException class I was able to specifically check each of these factors. By throwing the illegal exception my test was able to perform the test for the first name and last name to ensure each field could not go a one 10 characters or more. Additionally if the argument was thrown, if either field had hit 10 characters an argument flag would alert me to which field had been triggered.

There are two methods I used to ensure my Unit tests were being effectively with each of the weekly modules. The first method being the overall percentage each JUnit test covered with each module being around 80% effectively covered. As each module progressed the goal was to ensure each component went from working by itself to working as a whole program by compiling the files together and ensuring each component had been throughly tested. With each test showing coverage percentage I was able to build off of this and know exactly what areas I needed to perform continued Unit testing. This was the first time I have used Unit testing and now I plan on implementing it as often as possible in any future builds as it truly shows the progression of the build and the overall effective status before the end of the life cycle production.

ensure my code was technically sound I utilized Array lists built within the Java language to help with data structure and overall layout of the system. Another factor was keeping everything relatively organized by keeping things somewhat familiar with each other. For example incorporating add, equals, length. By using these to help with creating code I it become much easier to understand and organize what I was creating. It also helped with JUNit testing as I could simply just test each of these functions such as testing the Length of taskIdLength, or testing contactService.add(C4));.

Being new to programming projects such as this it was hard to determine code efficiency because I had little previous knowledge writing code to go off of. However, watching tutorials on Unit tests, and reading the weekly resources provided learning what effect code was became much clearer. Rather than writing the full program then performing bug fixes and troubleshooting I took it one step at a time and focused on one segment of code at a time. Each time I wrote a segment I would proof read and test to ensure it was error free before moving on to the next. The more important factor I found in all of this was discovering how case sensitive the Java language truly is. At first I had several errors specifically with the Contact\_ID and Contact\_List not being recognized when running the code.After discovering my mistake of not always using capital or lowercase letters I was able to overcome my issue and correct all mistakes. In order to be efficient and effective it is best to test as many things as possible using Unit tests. This ensures each component will function the way you intend and that it does not throw any errors when the program is ran through debug.

**Reflection**

In this project I primarily used black box and white box testing techniques. Black box techniques focuses on testing software with no prior knowledge of internal workings (*What is Black Box Testing: Techniques & Examples: Imperva* 2020). Additionally, black box testing supports end to end testing of invalid inputs, tables as well as specific test conditions. While as white box testing focuses more on the structure of the program and test data from logic of code. Both methods are equally important to utilize within a testing environment as each supports different major components within a system. In my opinion in order to be a successful programmer we must test our code as much as possibly and utilize and many tools as possibly to ensure our code has a great quality and that it succeeds all expectations. By utilizing black and white box testing a developer will ensure the highest of code quality and be able to meet system and client requirements in a more effective and efficient way.

One other software testing technique that I did not utilize within my modules or final project is experienced based testing. What this form of testing does is utilizes system users and software testers to use the system and to test it in such as way that it seems more or a trial run than anything. The primary focus of this form of test is to collect valuable data from the primary functions and components of using the system as well as navigating the system. This data is then utilized by the testers to correct previously unaware bugs and error such as issues with character limits. This method was not used with this project as it did not match the software requirements as it is a coding project with no real end user scenario to gather feedback from. However, experienced based testing would be a valuable tool for any project with a end user who will have a large amount of foot traffic using the application on a daily basis.

The techniques listed above are black box testing, white box testing and experienced based testing. With black box testing its primary function is to test each component or section of code to ensure it function the way it was intended for. While with white box testing it is used to focus more on the structure and ensures functionality is clearly understood. And with experienced based testing, requires both end users and developers to test the software end to end with average daily usage to identify errors and to ensure all components function without failure. Each of these testing techniques are vital to software development and focus on different aspects that range from the overall structure of the software, to the day to day use of the software. Primarily, I believe using all three of these testing methods would be very beneficial as it will allow a developer to test each segment of code through various degrees of tests while at the same time gathering feedback from users to improve upon and make the software even better than before.

The mindset I had to adapt while working on this project was primarily the mindset of growth. With this being the first major coding project I have partaken in the overall goal for me was to grow my existing experience and knowledge with both coding and testing. This mindset in very important because we learn from mistakes and from analyzing various situations. Therefore, reviewing JUnit test tutorials and performing test trails to ensure particular elements of the software functioned correctly was a big goal of mine when it came to the weekly objectives of writing specific classes. By doing these steps I was able to ensure my product was the highest possibly quality, and that it met all existing project requirements. The mindset of growing was very important as it being a larger application that something such as a “Hello World” program, so as the project progressed learning from errors and mistakes within the code really helped tune my ability to code more effectively with Java. Because of this I feel my professional skills and not just as a software developer but as a software tester have greatly been enhanced and have helped me further my career as a software developer.

Being biased in computer engineering could certainly be a concern in terms of testing your own code vs having a team or supervisor who routinely oversees and tests various developers code. It can lead to you being biased because if we are to skip corners, or have the mindset of thinking we do not make mistakes in our own work. Which would then becomes a huge problem and only allow us to see one side of things and not be able to grow as developers because we are choosing to not accept or grow from mistakes. This will lead to a bad quality in software and we present the client with very poor biased code. In this particular project the way I made sure to limit any bias was to test everything as often as possibly and multiple times rather than testing one time. With every Unit test I would write, I would run it right away to ensure each component performed the way I had intended, then as additionally measure I would review it side by side with the requirements to ensure it matched the clients requirements. This in incredibly important to test often because if I had only tested the length of the address and not that of the first name, last name, or even the full ID itself it would have thrown a significant error when the application was actually being used. At the end of the day it is important to always remember to limit as much bias as possibly when testing our own code. Always test often, and test and many variables as we can to ensure the code is high quality and functions as the requirements need.

It is extremely important to stay disciplined in terms of quality and commitment as a software engineer because we are often not just representing a software engineering company, but also we have to project and maintain our own image as well. What this means is we have to act in the better interests for the project and write clean code that best matches any and all requirements we are given for a project that will best match client demands. The quality of the code we produce will always outweigh the quantity as we are responsible for creating efficient, yet quality code that protects value data, and consumers that are going to use the software we are working on. In my opinion one fo the best methods to stay disciplined and to fulfill the commitment of a project is to follow any guidelines given by the client. Reviewing them as often as possible will help to realign progress made on a project and ensure all standards and requirements are being met to complete an assignment.

References:

*What is Black Box Testing: Techniques & Examples: Imperva*. Learning Center. (2020, September 24). Retrieved October 16, 2022, from https://www.imperva.com/learn/application-security/black-box-testing/