Justin Finch

CS 320

SNHU

April 16, 2023

Project Two

Moving throughout this course my testing approach has changed. I started with a minimal understanding of testing software and learned throughout the course how to test properly. The testing approach we used met the software requirements because we tested the required methods of the general code. As an example we can look at the contact class a requirement we needed to meet was the first and last name could not be longer than 10 characters. As a testing method we then used the ContactTest code of: “JUnit implemented the assertion "Assertions.assertThrows (IllegalArgumentException.class, () -> {new Contact("1234567890", "Mcdouglason", "JenniferMarie", "12345678901","1234 Elm Street, Staffordville, VA 01101");" Using this code we are able to test the method of making sure the value of the first name and last name were no more than 10 characters. to test for input that is too long. Also, the task class requirement was that the task ID could not be longer than the 10 characters as well. As we can see for the test code both names and the ID are longer than 10 characters. Using a specific method of "void testTaskIdTooLong()” combined with the JUnit test the system flagged the entry as invalid. Making the used code examples invalid meaning the system was functioning properly.

Understanding a JUnit test took me some time and it took me longer to get the test to function properly. I had a learning curve of using a Mac even though I did all of my work through Apporto I ran into issues of understanding how to perform the test. Based on my test performances I believe that my Contact test scored better than my Task test based on the outcomes of the test. My Contact test was harder for me to get working as it was one of my first attempts at testing. Using given resources and looking at some other resources such as Youtube and StackOverflow I was able to get the test to function and better than the Task test. Both tests are valid and ranged at a high percentage and the service test was the highest. Seeing the test provide a positive feedback showed how much of the test was properly functioning and indicated the test is valid. I will be using Junit testing in the future as it was a simple and effective testing method for my Java needs.

Before testing each code I needed to make sure my code was accurate and free of syntax and spelling errors. As a example we can look at the Contact class I needed to add this section "private static List<String>” because my Junit test could not function. As I learned on StackOverflow to add this code I was able to move forward with the test, before I added this my code was accurate but not functional for the test. I also took advantage of arrays through my code an example is ”ContactID = new ArrayList<String>();" this code allows strings to be added to the array. In my original attempts I used more classroom methods I had used and ran into syntax errors multiple times, I used class resources to find an easier way to may arrays. As an example my first array attempt within the Contact code was riddled with original errors and syntax issues. I cross referenced with Java syntax and I could not understand my errors. I found some simple errors that were syntax but I could not get the array to work so, I sought more help and was able to find the defined method.

I enjoy the aspects of coding and developing working software projects but I do struggle with making sure there are no errors. As I moved through my codes I had to make sure my syntax was correct and visible or flagged errors I fixed immediately but their were some sections of my code that looked visually functional but when run were not. Using YouTube to help me through the some of my functional errors and showing me how to set the code up for easy reading and it actually helped me with some of my errors walking me through setting up the code. Using the class resources and others like Microsoft’s website I was able to complete the modules and the tests within. An example of an issue I was running into is I did not declare my variables before I initiated them. This error was not flagged and drove me crazy for a while before I watched a Youtube video that showed me my errors. Within the JUnit test I declared the following: "if (!alreadyPresent) { tasks.add(task); System.out.println("Contact added"); return true;" Once I tested this section I would be able to check the test id and see if it was repeating. Another example I would like to highlight is from the contact service test. My test tested the contact validity and if one was already created. The code I used is, "assertEquals(true, contactService.update("12345", "John", "Smith", "1234567890", “53 Delphi Road, Jacksonville, FL 12345”)); assertEquals(true, contactService.update("54321", "Cerina",

"Douglas", "1222222222", “29 Hemlock Ridge, Columbus, CA, 03033”));”. These examples tested valid contact information and made sure it was functional within the system. On the other hand testing an invalid contact looked like, ”assertEquals(false, contactService.update("CID222", "Mike", "Mcdougal", "111111222222", “99 Smokey Lane,” Springfield, MA 11011”));.” This tested for an invalid contact and returned an error meaning the test was valid and functioning properly.

**Reflection:**

The testing techniques and methods I used throughout this course are considered black and white box categories. This is because these methods were considered specification-based or structure-based testing methods. Through my understanding” Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test.(Imperva)” Even though this was not directly true as I generated the code and worked through requirements to get the code to work, I was able to generate an output through testing to help me better understand the code I wrote. As I provided the example in the last paragraph I was testing for valid and invalid inputs and from the Contact we can see I used both methods to see how the code would react. As the code was tested we could see that the valid contacts worked and were accepted by the system and the once designated to fail was rejected properly. I also took advantage of structure based testing, this was used before this course as I would test sections of my code to make sure it was functioning. I did this throughout this course as well and I find it extremely helpful for putting my entire code together. I look at it like building a paragraph my section or sentence needs to make sense on its own and work on its own. Once it is valid I can add it to a paragraph or my project knowing it fits and works. The definition that helped me think about structure based testing in such a manner was,”Structure-based testing techniques use the internal structure of a software to derive test cases. They are commonly called 'white-box' or 'glass-box' techniques. Structure-based techniques can also be used at all levels of testing.(HCL Tech)” This definition allowed me to understand that my testing method is considered a white box technique and helped me to understand both concepts.

There were a few methods of testing that I did not take advantage of throughout this course. These testing methods were more of a public testing method. The methods I am thinking of include publishing my code to users to test. As this method is not necessarily a good option for online classes I was not able to publish my code and have anyone test to make sure it was operating. This type of testing highlights error guessing and exploratory testing not necessarily pin pointing a specific issue to fix directly. The concept of error guessing is where prior experience is used to determine which tests would be best to analyze the code. As I did not have a lot of testing knowledge to begin with I would not have done well with this method. Exploratory testing is used to test areas that are lacking specifications. I also did not use this technique as I Did not have enough experience to pin point lack of specifications and issues. I will be using these methods one day but I am currently not equipped too use these methods

The techniques described above include black box methods, white box methods, and experience based methods. Black box is used for outsource coding and generally are used when code has specific sections. White box techniques are used when the entire system is well understood and the developer knows what they want. These two techniques are both valid in their own way and can be used on many different coding structures it is the developers decision which technique they want to use. My mindset for working on these code was technical, experimental, and critical. I needed to make sure my code was technically sound and free of syntax and spelling errors. Also from a technical standpoint I needed to make sure all of sections were functioning. My mindset was experimental seeing how to produce the code in a easy to read manner and following OOP thought process I wanted to make sure my code was functioning and I could work and make it better. I used some out dated syntax that actually had easy shorthands to use and reduce the bulk of my code. Finally my mindset of being critical also falls onto the last section of this paper making sure my code was unbiased. I made sure to go through my code and fix any errors and taking the time to learn different short hands I wasn’t stuck in the mindset of this is the only way to move forward with my code. I used the the critical eye to reduce redundancies and fix errors before my code suffered. Knowing as the developer I am the one testing my code made me think about how I was coding. In my first attempts I coded as I would and it was bulky and hard to sift through. Using the new shorthand techniques I reduced the bulk and made my code easier to read and once tested easier to fix. Finally, cutting corners in coding is a bad idea. I have run into this issue in the past by using improper shorthand before I truly understood the concept and this caused me to not understand exactly what I was doing. Doing this created a loop in my mind seeing the syntax of the short hand done properly but still getting errors made me go insane. Only once I reverted back to the long and I was able to understand exactly what I was coding and testing for. Reducing short cuts until the concepts of the system and coding are understood is a great way to prevent returning to the code post testing for major errors.

Citations:

*Structure based or whitebox testing techniques*. HCLTech. (n.d.). Retrieved April 16, 2023, from https://www.hcltech.com/blogs/structure-based-or-whitebox-testing-techniques#:~:text=Structure%2Dbased%20testing%20techniques%20use,at%20all%20levels%20of%20testing.

*What is Black Box Testing: Techniques & Examples: Imperva*. Learning Center. (2020, September 24). Retrieved April 16, 2023, from https://www.imperva.com/learn/application-security/black-box-testing/#:~:text=Black%20box%20testing%20involves%20testing,by%20the%20system%20under%20test.