Steven Cognata

Southern New Hampshire University

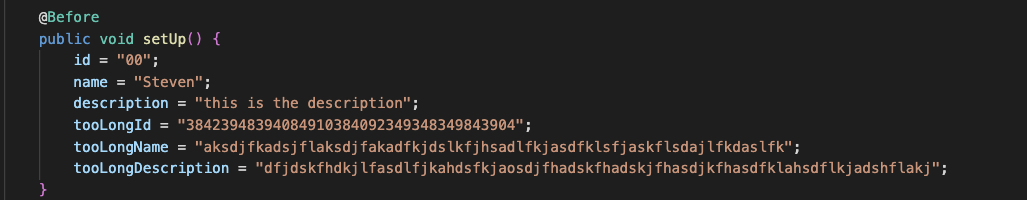
CS305-T1165

Dr. Bolton

October 16, 2022

In order to ensure that all my testing was being aligned with said software requirements I had to create a test per each option. To give an instance of this in the TaskServiceTest.java file I have a test for when the requirements are met, when the requirement is null, and when the requirement is too long. I took this approach because this is how I can ensure every single function is tested, every single outcome that can happen to that input gets tested and properly dealt with. This ensures that the functionality of the program is protected and is working as it should.

The quality of my JUnit tests were all exceptional except for a few. I was able to conduct research on these types of tests to better understand why they were failing. It turns out that the tests were set up correctly; however, I had not done the set up before as I had to initialize every single variable I was using in that document.

I like to follow best practices and industry standards as that is what will help me grow most as a software developer. I created the document in such a way to ensure all my variable names made sense. In this case of the TaskServiceTest.Java I ensured to label all the variables properly as to what their purpose was. As you can see the variable tooLongName is clearly labeled as such because it is testing the name section of the software and what would happen if this many characters were to be used.

I would also leave myself many comments throughout the software files to make sure that when I came back to it or if another developer where to look at it they would understand what each test under the comment was doing without having to read through every single line of code there is on that file. As you can see below the comments indicate what each test is doing before you even take a look at what the test has in it; this was all intentional. 

Looking back at my code I have come to the realization that without knowing it I was using something called white box testing. This technique is used by developers who know the inner workings of the software which I then used to test the functionality of this program.

A technique I did not use at all is something called experience based testing. You need to be an experienced developer and know what you are doing to conduct this type of testing. Hence the name “experience” based testing. You conduct something called error guessing where you apply your knowledge to guess where areas are most prone to errors through your experience as a developer.

I originally thought tests were automatic before this course as I left them all blank my first assignment. I came to the realization that this is not the case and you have to understand what you are doing. After this I applied a lot of caution moving forward as I was scared to mess up again like that due to my inexperience as a student. I have learned that complexity is something not to be afraid of because in due time and experience I will figure it out either by myself conducting my own research and knowledge or with a helping hand of a senior developer.

I do not have a bias whether I wrote the code or not because I do not view programming as such. I feel like it is almost like a math equation per se solely logic based. I do not take it personally testing my own code and finding bugs or vice versa.

Software development is a type of engineering. In short you do not want an architectural engineer cutting corners while constructing a bridge. In programming it is no different, what if the developer cut corners during some security programming, now there is a problem. I learned that there is always going to be someone who views the code I write before it is sent out into the real master branch version so any mistake will be brought to my attention before any real harm is done. However, I will take everything like I said previously with a grain of salt and not personally purely logically.