Branden McCusker

Aug 11 2022

Project Two

In this course, I worked on three projects; The Contact project, The Task project, and the Appointment project. All of these were very similar in functionality, where the biggest differences were specific tweaks to better fit the context. For example, all of the projects had an ID variable, while only the Contact project had a phone number variable. Despite these programs being simple, the final project contained the java.util.Date import that made it much easier to develop.

The reason for these programs being simple is so I can test each method easier. For testing these methods, we used JUnit 5. My approach was aligned to the software requirements because I tested each method to check for errors and incorrect expected data. To test for this, I used the assertTrue method for the bulk of my testing on all programs. I only used assertTrue as opposed to other methods in my JUnit portion of testing because I was able to check for inaccuracies outside of the plugin. For example, if I wanted to test if the program would throw an error due to too many characters being in the description variable, I can test that by running the program and inputting it myself. However, because the methods are very simple, and I am able to implement functional testing and visually inspect the code without running it to confirm it will function as intended. All of my JUnit tests are of good quality, as the coverage percentage was on every method worth testing.

As my JUnit tests were very simplistic, I can confirm that my code is technically sound and efficient by using the assertTrue() function. This method allows me to both test if the programs data matches what I give the assertTrue() function, as well as test if the programs output matches what is expected. For example, in the Task program, two lines of code from my JUnit test are:

assertTrue(taskService.updateDescription("123", "Psychic Arts Student at OUX”));

taskService.displayAllTasks();

The assertTrue() line is calling the updateDescription() method in taskService with inputs I have given it, and the displayAllTasks() method confirms the data to be accurate.

The software testing techniques I employed for each of the milestones including assertTrue from the JUnit 5 plugin, functional testing, and nonfunctional testing. If a team existed for these milestones, Agile testing would’ve been used as well. By using JUnit 5, the program was able to be logically tested. This means that all of the functions were individually tested for errors. Functional testing allowed me to make sure the program worked the way I wanted to without errors, while nonfunctional testing allowed me to ensure the program was fast and efficient.

As already stated, Agile testing would be of great use if a team was involved. The reason for this is because with the more people developing for a program, the more chances there are to catch bugs, logical errors, and security flaws. Continuously looking for bugs adds stress to a workflow, however it ensures a more smooth end product. Dependency checks are also very helpful for larger programs, but this focuses more on security flaws instead of bugs. To give a more detailed version on why these specific testing methods are beneficial in certain scenarios; JUnit is best for ensuring all methods are logically correct, functional testing is best for ensuring the program is free of bugs, and nonfunctional testing is best for ensuring the program is efficient.

As a programmer and a developer, my biggest task when designing any code is implementing proper security measures. Even a simple program that does not communicate over the internet still poses a threat to an end user if it puts their private information at risk in the form of a virus. Because of this, cautious measures must be taken when implementing plugins. For example, the Java.util.Date plugin has been determined to be safe due to the result of dependency checks.

Me not bothering to implement more JUnit methods and solely relying on assertTrue() is a form of bias on my part because I am assuming my code is correct, despite my solid understanding of the code. I tried to limit my bias and follow guidelines by performing tests on these methods in other ways, such as nonfunctional and functional testing as previously stated. However, I do understand in a classroom setting that I should have utilized JUnit more, and I will do so moving forward in my degree.

Being disciplined as a programmer is something that took a while for me to learn, and is something I am still working on. Many programmers go into the field assuming that they can make whatever they want and they will get paid for it. While this is sometimes true, the reality is that you will most likely be programming for somebody else, sticking to guidelines and end results that they set out for you. You may have some creative freedom with someone who doesn’t necessarily know what they want, but if you’re programming for a big company, chances are you will be making what they want you to make. It is also important to not cut corners when developing any program as functionality will always suffer as a result.