Hillary Loyd

Professor Linda Gaston

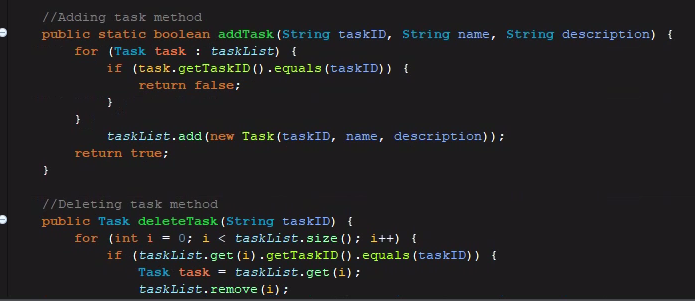
CS 320 Software Test Automation& QA

August 14, 2022

CS 320 Final Project

While developing and testing each of the components that were asked of me, my approach was very much in line with the software requirements that were asked of me. One example of this would be in appointment and appointment services, the description of this could not be over fifty characters in length or could not be null. When testing this component of the system, I had a setter methos throw an illegal argument exception. This would be the case for all the requirements that were asked of me throughout the whole system. The quality of my JUnit testing was effective. While it was not at 100% it was over the 80% that was given as a requirement. While my overall coverage for JUnit testing was 85.5%, each of the individual files has different coverage. The ContactTestServiceTest.java, ContactTestService.java and Task.java all had 100% there were some coverages that only had minimum coverage. Together is what makes the whole coverage of the system.

One of the ways that I made sure that the code that I was developing was technically sound was making sure that the annotations that I created were proper in use. For all the test that I have in my test files I made sure to have the @Test annotation above them. This is to declare that the method that is running is a test and allow me to provide those assertions. As we can see below for the given example:

Text

Description automatically generatedOne of the ways that I endure that my code was efficient was making sure that I was using conventional naming and comments. This technique promotes an efficient environment, which is great if you are working in a team to allow them to see your comments and help them understand the portion of the code that you developed. An example of how I preformed this in my code is shown below in the image:

The software testing technique that I employed in this project was a type of dynamic testing called structure based testing or white box testing. This type of technique is focused on how the code structure, data structure and internal design works and test accordingly. One of the important factors of this type of technique is code coverage. The goal of this is to traverse all statements at least once. This is to find any faults within the code that might exist. For the milestones assignments that would make up the final project this was very important. As it would make sure that all the code was functioning as intended and would also be a requirement for the final to have at least 80% code coverage.

A type of technique that was not used in this class was a static testing technique. This is a type of testing that is done without running the code and the tester trying to find the defect that might be written within the code. Static testing could be done with a peer review, walkthrough, technical review, or an inspection of the code. In a peer review, another developer would look at the code and try to find the faults. Whereas a walkthrough is walking through the code to the shareholders to explain my logic behind the code that was written. Since this was not needed we did not implement these techniques into the process of testing.

While this project is not a difficult project to tackle, the mind set that I had for this project was to go in and effectively execute the requirements that were being asked while still being cautious. This is because of the relationship between the classes, such as getting the appointment, contact, and task classes to talk the service class. If one thing was wrong within the code it could throw off the entire project. This is something that would have to be done with caution and attention to detail to make sure that the variables and methods work accordingly.

Limiting bias was not done in my code all that much. This was because I wrote the code myself and it was not a big concern. While working in a team, however, it is very important to limit bias. It is very possible that when writing with other developers to believe that your code is the one that takes the cake so to speak. This is likely to cause you to miss potential defects within your own code. This can lead to a whole lot of problems within development and is easily avoidable. Putting away an ego so to speak and making sure that the review that you do on your own code is the same that you would do for a peer is a very important skill to have in your tool belt as a developer.

The importance of being disciplined in the commitment of quality is something that is very important. While it may seem like the best thing to do is take short cuts and cut corners, this can lead to a lot of problems. This can impact the quality of the code. If the commitment for testing your software is not there either then it is almost certain that there will be at least one problem within the code. We are not perfect, and we all make mistakes. These mistakes can lead to loss of money and time within a project. Therefore, it is very important to have disciplined and not cut corners or take short cuts on the projects that you would develop.