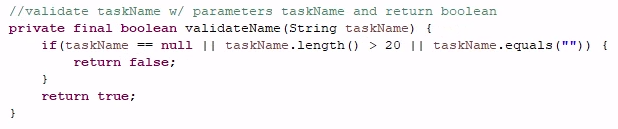
Kirissa Byington

CS320 Project 2

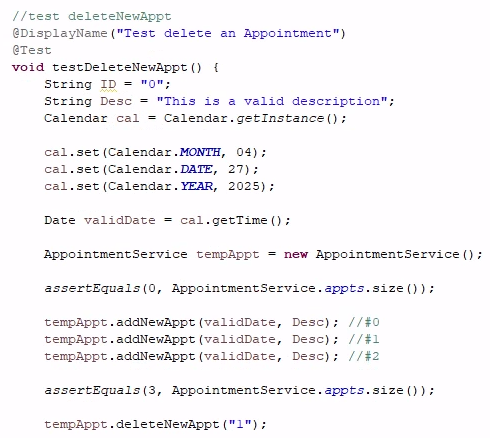
12/13/24

My unit testing approach for this assignment and the three features, Contact Service, Task Service, and Appointment Service, was more of a manual approach. I used unit testing to test the individual components of my code to verify that they worked correctly on their own. My approach was aligned to software requirements by creating code that fulfilled and tested again certain requirements. For example, for the Task service feature there was a requirement that the name cannot be a null value, more than 20 characters long, and is a required value. I successfully set those requirements within the following code.

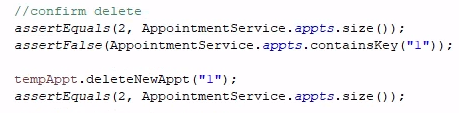


Each requirement for taskName was set and then I tested for a valid name, then a string with a null value, and an empty string to ensure the requirements set are being fulfilled. I was able to ensure that my code was efficient through the use of JUnit testing. I completed several rounds of testing and then reflecting to update code and ensure functionality of my program. Although I wasn’t able to quite make it to 100%, I still came along way from my first JUnit test with a coverage percentage of 3. The testing helped me find errors within my code and syntax to fix these mistakes.

An example of where I was able to ensure my code was efficient can be found in my Appointment Service program where I created a method for deleting an appointment. I was able to test this method by first adding a few appointments to a list and then confirming the list held 3 items as seen below.



Then I was able to delete an item, an appointment, from that list and confirm the list had only 2 items within list, as seen below.



Testing this method with the JUnit tests I proved that my code was efficient in deleting an appointment successfully.



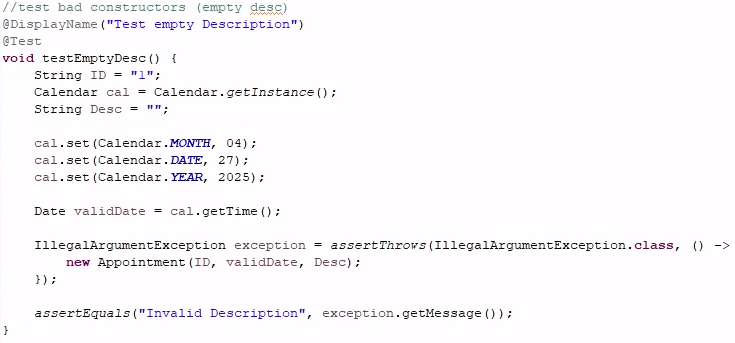
The software testing techniques that I employed in this project would be both static and unit testing. These testing techniques are a form of Whitebox testing which allows a user to verify the inner workings of a program. Specifically, static testing is involved with studying the code and then comparing it with the requirements and specifications to be able to identify bugs and possible errors. On the other hand, unit testing, in this case JUnit testing allowed me to find and fix potential errors that in end ensured a working test case and efficient program. After creating the static code to meet the requirements of the project I then used unit testing to find errors within my code. Since I started out with a test coverage of 3%, I was able to find and fix many ongoing problems. Some of these problems were as simple as improper syntax used or spelling errors. After resolving the minor issues, I would rerun my unit testing to continuously find, fix, and reflect on my work until I was satisfied with the result.

Other software testing techniques that I specifically didn’t call upon withing this project were integration testing and security scanning. Integration testing would test the whole program with all systems together. The security scanning would be useful for projects that have sensitive data within libraries and the like which would need an additional test ensuring privacy and security within the program. I didn’t perform any security scanning withing my project specifically because there was no data being used from libraries or databases or even user input.

Each of these testing techniques is practical and useful for different things and cases. For example, unit testing is useful for finding and resolving small issues before they become a bigger headache later. Static testing is beneficial for detecting bugs and other issues early on in the development process. Integration testing is useful for big projects. Finally, security scanning is useful for data protection and when utilizing user input or public and private databases.

The mindset that I adopted while working on this project was that of a analytical one. I tried to dig deeper into what the requirements of the project were and what components may be needed to fully meet these specifications. For example, within the Appointment Serice project requirements was this specification.

 At first it just seems that a description with less than 50 characters and not null is required. But using analytical thinking you can also deduce that a required description also mean that the description cannot be an empty string as well. Therefore, I also tested to ensure that the description field would not be empty, as seen below.



It is important to appreciate the complexity and interrelationships of the code I was testing because it allows me to create more cohesive and comprehensive test cases. It will also allow me to identify any potential bugs and issues that could develop between each different part of this application.

While completing these assignments I tried to limit my bias by focusing on the functionality of the code itself. As a software developer, I could possibly imagine that bias would be a concern if I was responsible for testing my own code. We tend to be easier on ourselves due to this bias which could open areas for errors and continuous improvements. As such, if it so happened I needed to test another's code, I believe anonymity of the author is important to avoid letting any personal bias of said author cause any issues.

Being disciplined in your commitment to quality as a software engineering professional is essentially important. By holding your quality standards high, you ensure that you only create quality programs as a professional. Sloppy and lazy work will not get you far or impress any prospective clients. Therefore, being diligent in the quality of one’s work is an immediate reflection on developers themselves.

I personally plan to avoid technical debt as a professional within the field by ensuring my code is clean and organized, implement testing to ensure functional software, and by organizing and scheduling refactoring sessions to allow myself time to clean and improve my existing code which will in turn create a better structured overall application.