Thomas Comer

August 17 2025

Desmond K. Springer

CS-320

Project 2

**Summary**

* **To what extent was your approach aligned to the software requirements?**

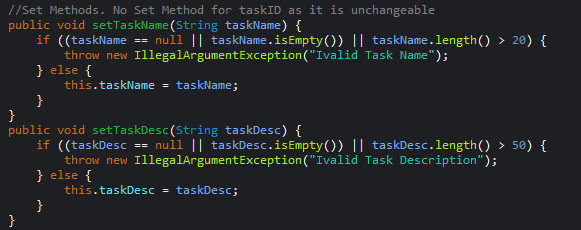
My approach to testing was to ensure that every single requirement that I was given was both met, and tested to ensure functionality. The assignment was to create 3 different primary classes, and 3 different service classes, each for the purpose of creating different identifier lists for contacts, tasks, and appointments for the customer’s mobile application. For example, the requirements for the appointments class were to create a unique string less than 10 characters for the ID, a date that isn’t in the past, and a description less than 50 characters, and ensure none of the items were null. My JUnit testing for this class was to ensure that all of the functionality in the requirements were met, which required 5 unit tests overall, one test for each specific requirement.

* **Defend the overall quality of your JUnit tests.**

My unit testing coverage totaled 67% with 34 unit tests across all my classes. Usually, it is recommended to create coverage that totals 80%, but I feel that my coverage was sufficient despite being 13% lower than the average recommendation. This is because I focused my coverage more specifically on ensuring every single requirement was tested above anything else. For instance, the reason that it isn’t recommended to achieve 100% coverage, and instead only 80% coverage is because that almost always requires creating testing for every single line of code, which is usually unnecessary. For my code, I ensured that every single requirement for every single class was tested, which would ensure the program always runs exactly as is requested by the client. This ensured every necessary test was done, and thus would always meet the requirements of the client.

* **How did you ensure that your code was technically sound?**

The main ways I ensured my code was technically sound was through proper input validation and consistent data handling throughout the program. One of the ways I accomplished this was by ensuring that all requirements are met through throwing IllegalArgumentExceptions if the user inputs an invalid input. The screenshot below shows an example of me using that form of input validation to ensure my code is technically sound and meets the requirements.



This, along with my testing efforts, ensured my code was technically sound and had no errors, only producing a result that met the requirements.

* **How did you ensure that your code was efficient?**

I ensured my code was efficient through utilizing the appropriate data structures for the requirements, as well as utilizing fast methods to ensure the IDs for each test are unique. By storing the distinct items in an arraylist, I ensured that my code is able to follow the operations required for each item in a fast manner, while always ensuring there are no problems with each operation. The other main efficiency was utilizing the AtomicLong library for generating IDs, as that automatically increments a unique ID. This is both thread-safe and efficient because the atomic operation ensures that the task IDs are generated sequentially, ensuring that the operation will always result in a unique ID. The following screenshot shows how the code utilizes this method to ensure an efficient method of creating unique, unchanging IDs.  


All of these efforts guarantee that my code runs fast, efficiently, and with no issues, ensuring that all requirements are met.

**Reflection**

* **What were the software testing techniques that you employed in this project?**

I employed similar techniques for all 3 milestones, since all 3 of the milestones followed a similar format of following requirements and using testing to ensure those requirements were met. The primary method of testing used in this milestone was Unit testing. Unit testing is essentially testing smaller portions of the code, instead of testing the whole code, to ensure specific functions function properly. This was done primarily through the JUnit5 library, and by writing code that tested all possible user inputs for every function to ensure that the requirements are met. Since all 3 milestones follow a similar format the main thing I changed with my testing from each milestone to milestone was just ensuring I was testing the requirements that were requested of me for each milestone. I can see unit testing as something that would be incredibly valuable in testing larger programs that have a large number of independent functions. In a situation like that, testing each variable makes more sense than testing the entire program as each function may be largely separated from other parts of the program, and so testing everything at once slows production down.

* **What are the other software testing techniques that you did not use for this project?**

A testing technique that I did not utilize in any of these milestones was automated testing. Automated testing uses an outside software solution for testing that autonomously tests code to ensure it functions correctly every time. Autonomous testing makes the most sense to utilize in rapidly updating, largescale programs that feature several places of user input that can go wrong if something changes. The scope of the milestones from this course did not meet any kind of scale that would make sense to perform automated testing, so I never attempted using it for anything in these assignments.The scope to each milestone was limited to a handful of exactly worded requirements that really only needed a few testing groups each to ensure that the requirement was met, so I don’t believe there are any major benefits I could have gained by utilizing automated testing here. If the scope of the milestones was dramatically larger, and each one built upon the last one, thus requiring testing of previous features each time, then automated testing would have made more sense to have implemented, but for now, Unit testing was the most practical way to test the program.

* **Assess the mindset that you adopted working on this project, assess the ways you tried to limit bias in your review of the code, and finally, evaluate the importance of being disciplined in your commitment to quality as a software engineering professional.**

The mindset I had was to ensure that all requirements were met, as to ensure the customer receives a product that does exactly as they requested from the requirements and doesn’t fail their expectations. To accomplish this, I made it my priority to ensure all requirements were met and tested to ensure satisfaction. I employed caution in my testing by ensuring all of the requirements were tested, and satisfied. If I missed even a single requirement in my testing, then it could lead to a program that does not do what the client expects. I would attempt to avoid bias by listening to feedback from others and making changes when necessary. I received feedback on the contact and task classes that I was able to implement in my final class and then reimplemented for the final submission that contained all of the classes in one project. Finally, I ensured that I was disciplined in my commitment to quality with my consistent testing to confirm my code followed the requirements. By ensuring that the requirements of the client were met above anything else, I ensured that the program was high quality and did exactly what it needed to do.