**7-2 Project Two Submission**

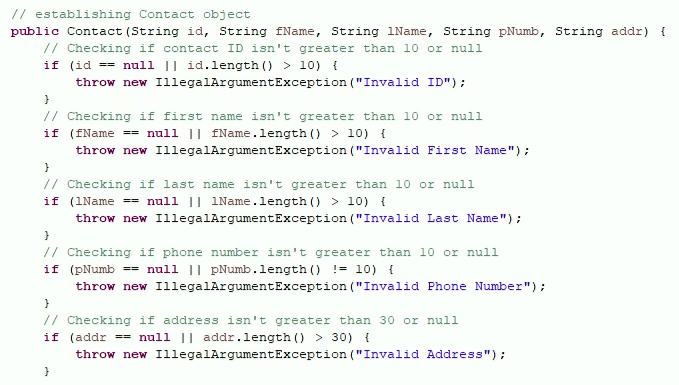
**CS-320**

**Chris Wong**

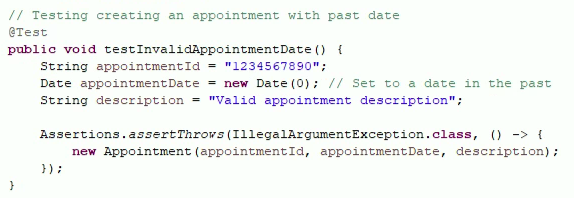
**June 18, 2023**

**Summary**

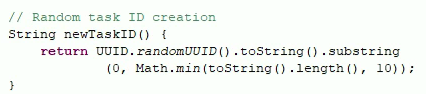
I followed a comprehensive unit testing approach to ensure that all features aligned perfectly with the software's requirements. For example, when dealing with the Contact service, I made sure that the ID, first name, and last name fields were limited to a length of ten characters each. To validate this, I implemented separate unit tests for each item, verifying that any input exceeding the limit would result in an invalid argument exception being thrown.

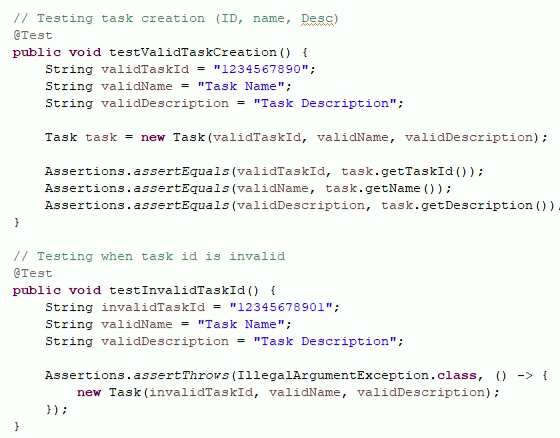


In the case of the Appointment service, it was crucial to ensure that appointment dates were not set in the past. To achieve this, I wrote a unit test that checked if the appointment date was not before the current day, utilizing the date utility in Java.



The Task service required the ability to add tasks with unique IDs. To address this requirement, I utilized a unique ID utility and designed a unit test to confirm that tasks could be added with distinct IDs, along with their respective values.





Once my tests were refined and fully functional, they achieved a high percentage coverage. This high percentage indicates that my tests effectively cover a sizable portion of the application's requirements. Writing JUnit tests was an enjoyable experience for me. Prior to undertaking these projects, I had never ventured into the realm of unit testing, including JUnit tests. Therefore, this opportunity allowed me to acquire new knowledge and skills that will undoubtedly prove beneficial in the future.

**Reflection**

For this project, I used white-box testing, an automated method. White-box testing involves assessing how a system handles specific inputs and produces the expected outputs. Typically, white-box testing is conducted by the developers themselves, as they have comprehensive knowledge of the code's internal structures and the intended purpose of each line. Since I was the sole developer working on this project, I had complete awareness of the inputs and the desired outputs according to the customers' requirements.

I did not incorporate black-box testing into our project, although it does have several suitable applications. Black-box testing involves independent testers who lack knowledge of the internal structure of the code. In such testing, the code's backend may not undergo evaluation. Unfortunately, due to the absence of a dedicated tester, we were unable to employ black-box testing for our project.

Black-box testing is commonly employed in the development of prominent consumer products like mobile or computer operating systems. Since end users are not privy to the inner workings of the product and only interact with the outcome, black-box testing becomes crucial in ensuring its reliability.

On the other hand, white-box testing is well-suited for development teams who conduct testing on their own code. With full access to the internal mechanisms of the code, these teams can effectively implement white-box testing to assess and validate their codebase.

Throughout this project, I embraced a learner's mindset, as I had no prior experience with unit testing or working on projects that required multiple packages or objects. It was important for me to exercise caution and avoid getting carried away with excessive testing. I focused on writing tests specifically for the necessary components. Completing the project allowed me to step back and appreciate its complexity, especially since I carried out it entirely on my own. I successfully coordinated multiple packages, even if some code segments were similar, which was a notable achievement considering my lack of prior exposure to such tasks.

As the sole developer, I took pride in my work, but I also recognized the need to avoid bias. It was crucial to remain committed to the end goal of a fully functional project and not become complacent with the milestones I had already achieved. Regardless of my confidence in the code's flawlessness, I thoroughly tested it to ensure its intended functionality. I believe that bias can become an issue when developers handle testing their own code, sometimes interfering with the goal of delivering a fully functioning project that satisfies the client's needs.

Maintaining discipline is highly significant for a software engineering professional. Without discipline, progress and career advancement can be impeded. Taking shortcuts in code writing and testing is detrimental not only to the company and end users but also to the developer or tester themselves. Cutting corners for time-saving purposes can result in costly consequences later, potentially requiring a complete restart if important security features were overlooked. To avoid accumulating technical debt, I plan to adhere to the best practices I have learned, such as incorporating comments, maintaining proper spacing, organizing code effectively, and utilizing secure data structures. By doing so, I not only safeguard my position as a software engineer but also protect the interests of the business I work for. Ensuring that everything is done correctly serves as a dual benefit.