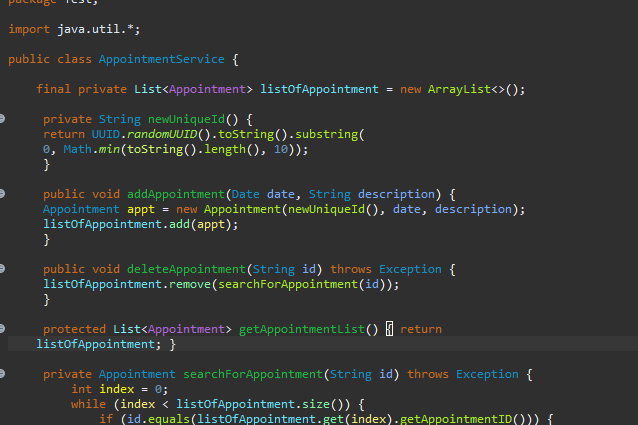
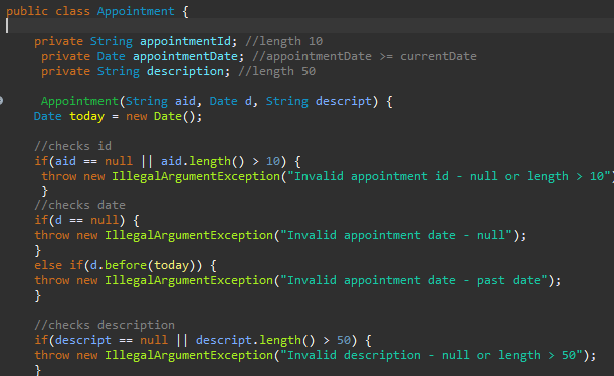
**SUMMARY**

The accuracy, efficiency, and compliance of code development is heavily reliant on effective testing. In order to meet the requirements stated in the User Requirements Specification document, I have created several class files: Contact.java, ContactService.java, Task.java, and TaskServices.java. To facilitate the necessary testing, I have also developed corresponding test class files: ContactTest.java, ContactServiceTest.java, TaskTest.java, and TaskServiceTest.java. This allowed me to break down the requirements and perform the required JUnit testing.

The overall quality of the JUnit testing was maintained at a minimum of 90%. Throughout the development process, I ensured that the code for the classes and test classes adhered to the business requirements by meticulously reviewing the provided rubric item by item. Although I acknowledge that my testing may not have covered every aspect from end to end, I am confident that the code I have written will fulfill the requirements once it is deployed to Production. It is worth mentioning that I faced some challenges with my TaskTest.java, but through that experience, I gained valuable insights and applied them to this application.

To guarantee the technical soundness of my code, I prioritized giving clear and meaningful names to all classes and methods. This practice not only facilitates future development and releases by other developers but also enhances the comprehensibility of each code block. The following screenshots provide examples of this approach.





Ensuring the technical soundness and efficiency of code is a crucial aspect of development. It is essential to review the code, especially when preparing for future releases, to identify potential memory leaks caused by deprecated functionality or poorly programmed logic statements. Although this application is currently in its initial release, I am quite confident in the efficiency of the code I have written. One example of this efficiency can be observed in the provided screenshot, where I have implemented meaningful error messages for end users who enter information that exceeds the specified user requirements. This approach enhances the user experience by providing clear feedback and guidance in completing the electronic form that serves as the basis for this application.

**REFLECTION**

For this project, I exclusively utilized white box testing, which involves testing the "Expected Output" and "Loop Testing" without the need for external involvement(2023). Since there were no dedicated software testing resources available, the software developers themselves conducted the testing. This approach allowed us to create tests based on the business requirements. However, we did not employ any automation or Continuous Integration/Continuous Delivery (CI/CD) processes because there was no specific requirement for automation, and as the sole developer, branching functionality for merging into a repository was not necessary for this initial release.

During the test coding process, I made use of if/then/else and try/catch statements, enabling me to write clean and efficient code that can be easily reviewed and improved by future developers as new releases are requested.

Conversely, I did not utilize Black Box or Performance Testing. Black Box testing is performed without knowledge of the internal structure, design, or code of the system under test(2023). In this case, it would typically be conducted by a dedicated Quality Assurance (QA) team, but it is unclear if such a team was involved in this project. Black Box testing focuses solely on the input and output of test objects. Another type of testing that was not employed is Performance Testing, which includes Load Testing, Stress Testing, Scalability Testing, and Volume Testing. Since there was no guidance provided regarding the number of users expected to use the software, the system architecture, or the database storage, it is difficult to determine the specific requirements for Performance Testing. It is reasonable to assume that this type of application would not store all records locally on each machine, as that would create conflicts with IDs.

White box testing is suitable for small businesses with limited resources or users who may not have technical expertise. However, the drawback of this approach is that it relies solely on the developers creating tests based on the provided requirements, and potential bugs may be discovered by QA testers or end-users. Conversely, Black Box testing is more suitable for medium to large businesses where dedicated teams can be allocated to perform additional testing beyond what the developers have coded(2023). Performance testing can be implemented in businesses of various sizes, independent of the number of available testers. However, it is crucial to have an understanding of the current user base, as well as future growth projections. Additionally, it is important to consider the configuration of the server hosting the application and whether a separate database server is required or if it can be hosted on the same application server.

**MINDSET**

Ensuring high-quality code requires a combination of accurate coding and effective testing practices. Throughout the development process, I strived to achieve comprehensive test coverage, demonstrating my commitment to thoroughly testing the code. However, it is crucial to recognize that self-evaluation can be susceptible to bias. To address this, incorporating peer review becomes invaluable. Having a fresh set of eyes review the code and test cases can uncover potential shortcomings and offer valuable insights, leading to a more robust and reliable end product. By embracing a collaborative approach, developers can work together to identify areas for improvement and ensure that the code meets the specified requirements. Peer review helps to deliver a high-quality product by minimizing the chances of overlooked issues and enhancing overall code quality.

Discipline in coding practices is fundamental to avoiding shortcuts and delivering a reliable solution. Rushing through tasks or neglecting proper coding practices can lead to wasteful resource usage, delayed results for end users, and potential revenue loss. It is important to meticulously review the requirements, write clean and consistent code, and adhere to coding best practices. Additionally, incorporating thorough code reviews, testing, and documentation further enhance the quality of the codebase. By investing time and effort into maintaining disciplined coding practices, developers can create a solid foundation that is easier to maintain, debug, and enhance in the future.

Moreover, having proper documentation, such as User Requirement Specifications, System Architecture Specifications, and Design Configuration Specifications, plays a crucial role in managing projects and reducing technical debt. These documents provide a clear understanding of the initial requirements and system design, allowing for effective communication and decision-making during development and future releases. They serve as a reference point for developers, testers, and stakeholders, ensuring that everyone is aligned with the project goals and expectations. Additionally, fostering open communication with the business and stakeholders facilitates feedback, highlighting areas that need improvement, unresolved issues, and new feature requests. This collaborative approach promotes a shared understanding of the project's objectives and minimizes the accumulation of technical debt over time.

**REFERENCES**

The QA Lead, & Draniceanu, A. (2023, April 19). *Black-box vs white-box testing: Key differences*. The QA Lead. https://theqalead.com/test-management/black-box-vs-white-box-testing/