Allen Jhane Dela Cruz

CS 320 - Software Test Automation & QA

18 Jun 2023

**Project 2**

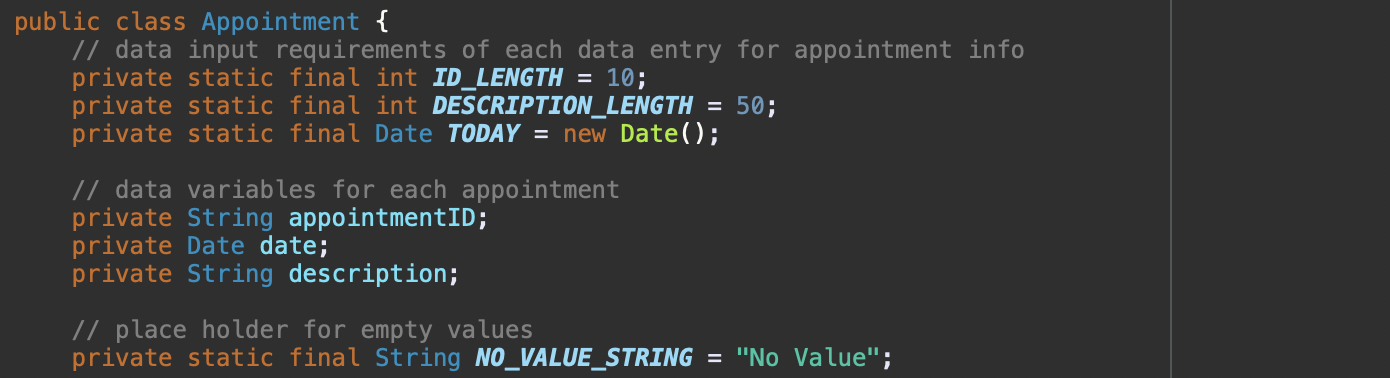
In the past couple of weeks, we have programmed a mobile application for a customer as a software engineer for Grand Strand Systems with three features that include contact, task, and appointment services.

1. **Summary**
   1. Describe your unit testing approach for each of the three features.
      1. To what extent was your approach **aligned to the software requirements**? Support your claims with specific evidence.

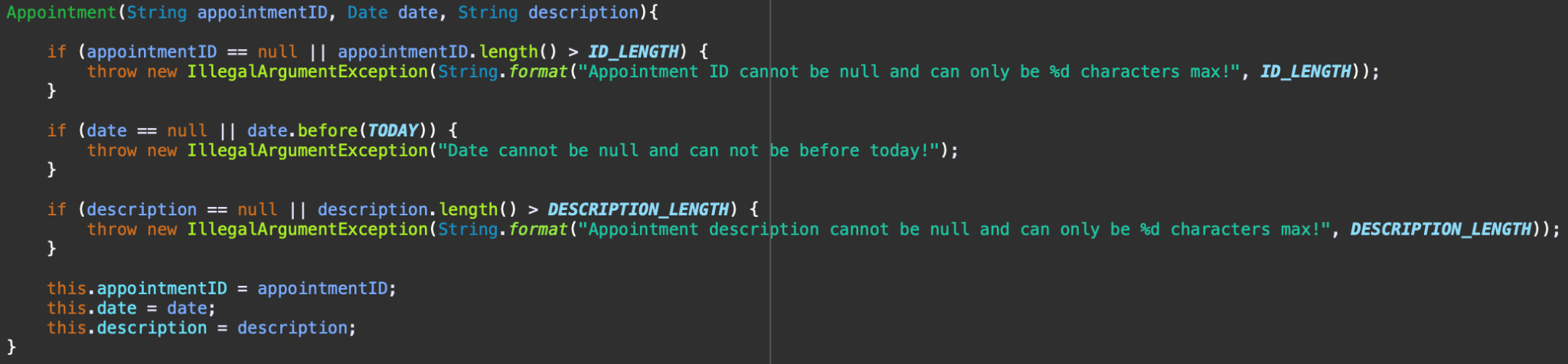
* My approach for each of the three software features (Contact, Task, and Appointment Services) aligned with the software requirements entirely. As I programmed the software, I referenced the requirements of each class and made sure to include all aspects needed in the program. I then checked the variable requirements for each of the required classes through branch statements that would output an error when misaligned with the variable requirement. I did this through each class’s respective test classes and test cases.

Below are examples from the Appointment class that will provide visual examples of what I implemented to complete the requirements of the software:

Variable declarations that are used for identifying the lengths of the variable and initial global variable that will be used throughout the class



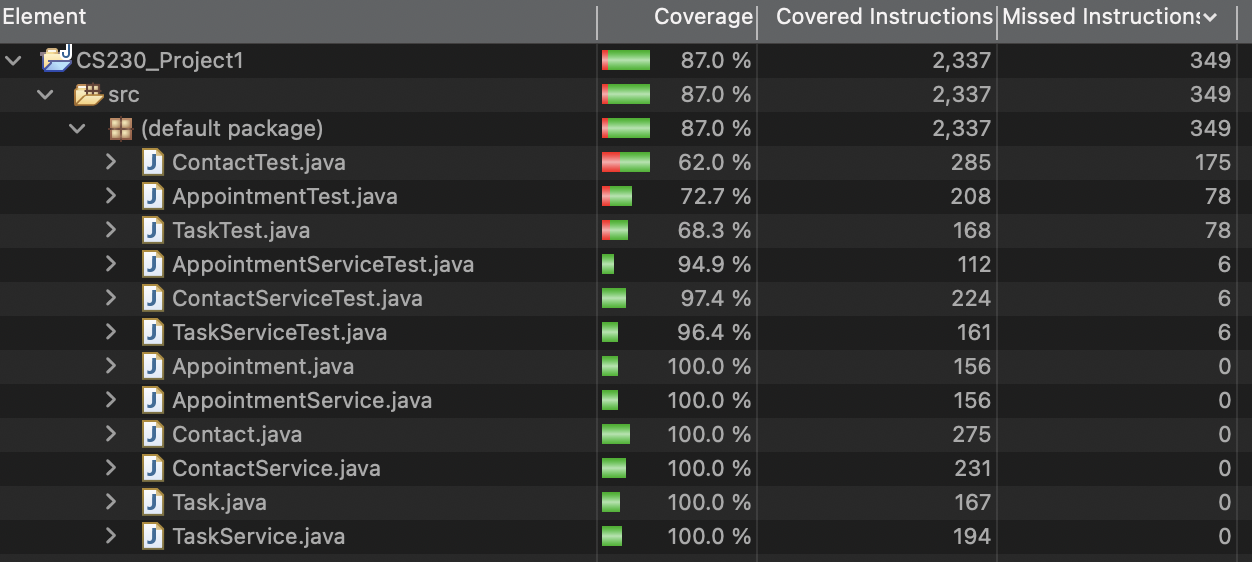
Constructor method that creates an appointment if all requirements are met, else it will throw an error to the console



Test method from AppointmentTest.java class that will test the constructor and its constraints based on the code shown above



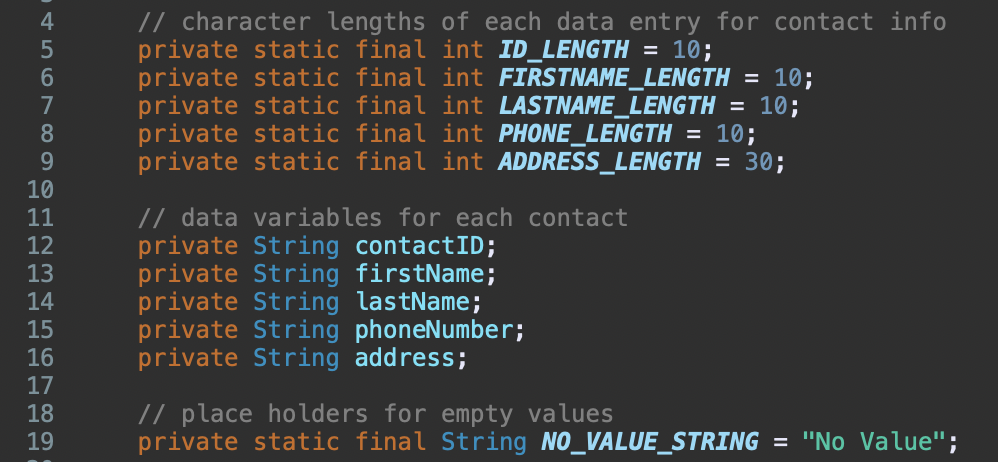
* + 1. Defend the overall quality of your JUnit tests. In other words, how do you know your JUnit tests were **effective** based on the coverage percentage?
* I know that my JUnit tests were effective based on the coverage percentage that I got in all their test classes. I received 100% coverage on all the classes and accounted for all branches that were indicated in my software. Below is a visual of their percentages:



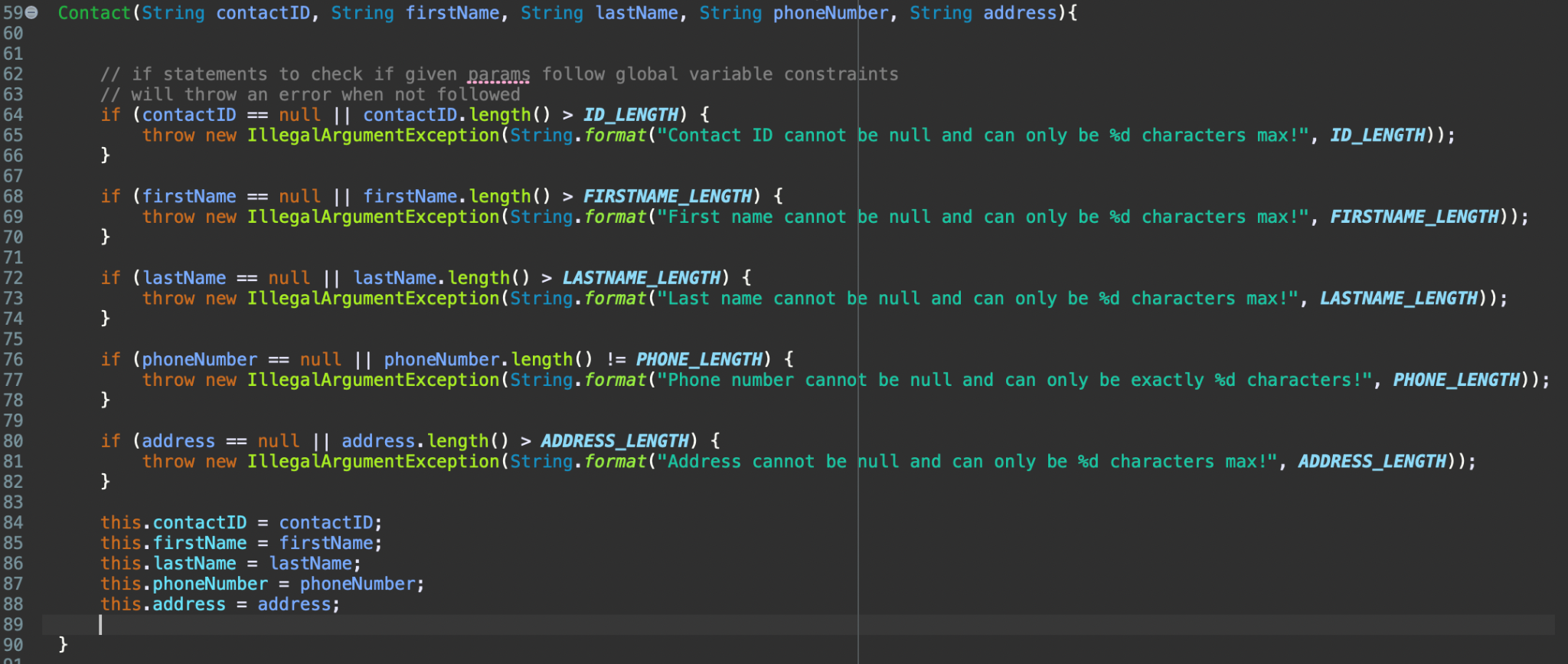
The actual test class percentages do not matter but the actual classes to be tested (the last 6 indicated in the list) do matter since these are the programmed classes that we are testing for full coverage.

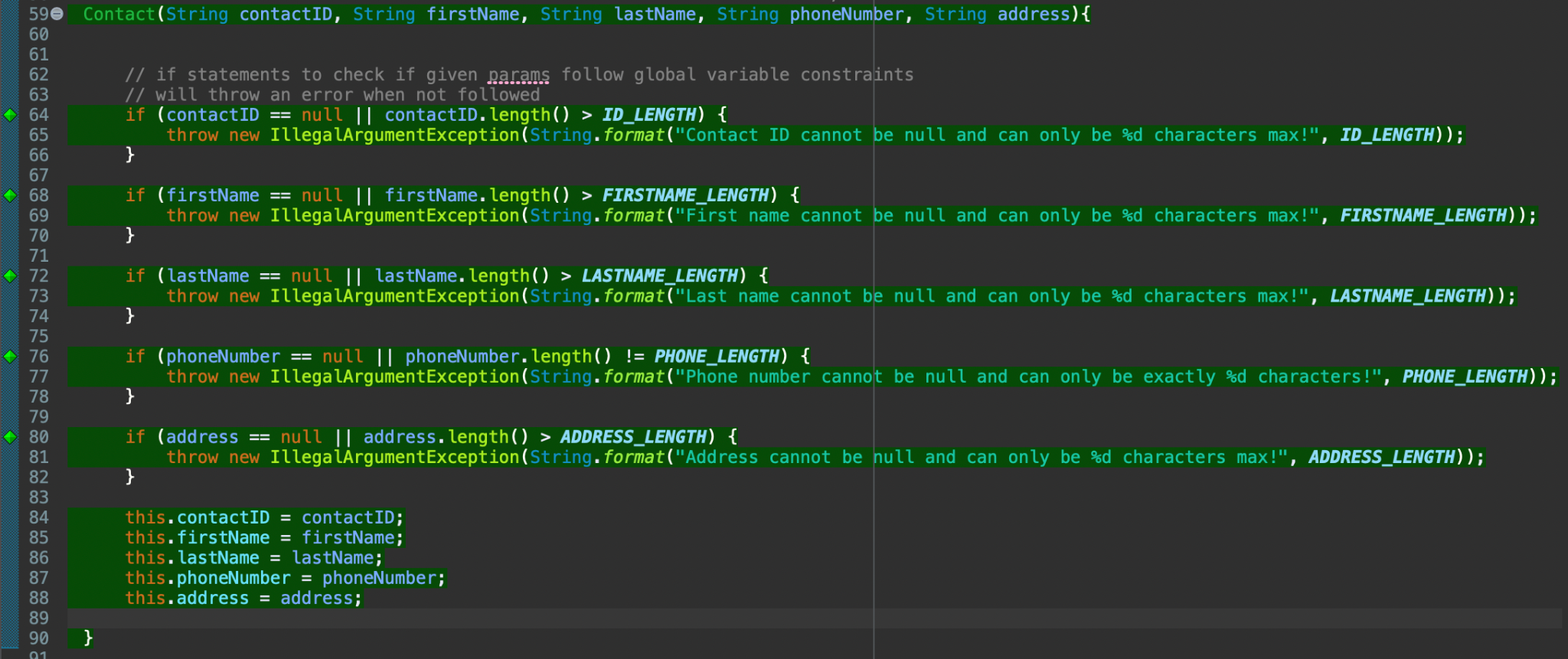
* 1. Describe your experience writing the JUnit tests.
     1. How did you ensure that your code was **technically sound**? Cite specific lines of code from your tests to illustrate.
* To ensure that my code was technically sound, I included error cases whenever a variable constraint was not met. I also used global variables for all numerical value constraints so there would only be one place to change a constraint if need be. These were all tested in JUnit tests to identify if all expected outcomes are met. Below are the lines of code that I have done this on for the Contact class.

Global variables are declared for contact class (lines 4 - 19) variable constraints so they can only be changed in the program once so the software is free from as much human programming error.



Contact constructor class (lines 59 - 90) that will provide errors when constraints from global variable values are not met when creating a contact (1st picture) and an indication that all tests passed and all branches were tested (2nd picture).

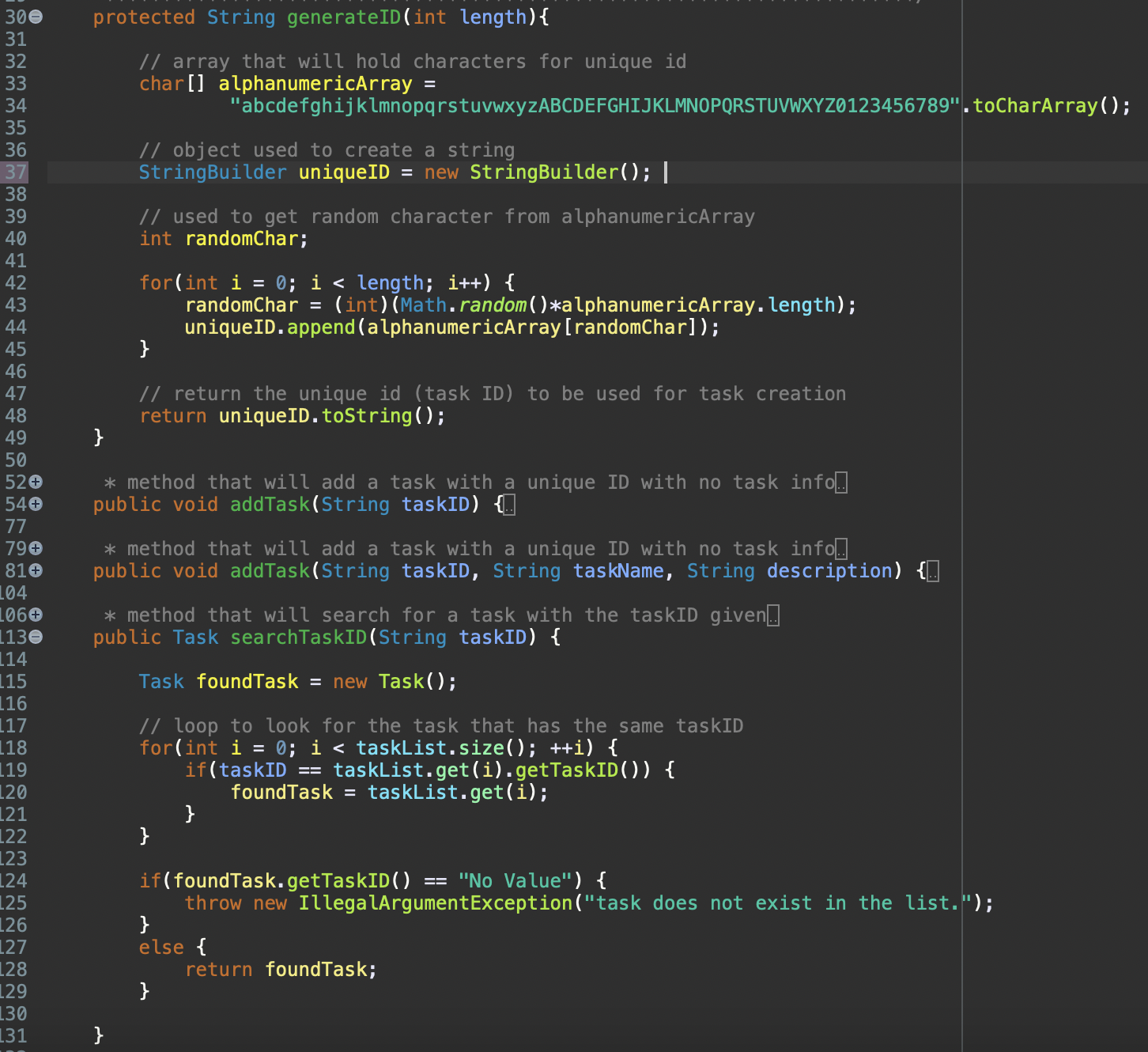




Test (lines 51 - 96 in ContactTest.java) that will test the constructor of creating a contact and if all output required (good and bad) has the expected outcome.



* + 1. How did you ensure that your code was **efficient**? Cite specific lines of code from your tests to illustrate.
* To ensure that my code was efficient, I made sure to create additional methods that will be needed in multiple areas. I did this for generating a unique id and searching for an object in a list for all 3 features in their respective service classes. Below are the examples from the TaskService class:



ID generation (lines 30 - 49) and ID search (lines 113 - 131) methods that are used throughout TaskService, notably the addTask (lines 61 and 88), the removeTask (line 140), updateTaskName (line 154), and updateDescription (line 170) methods.

1. **Reflection**
   1. Testing techniques along with their explanation of their **practical uses and implications** for different software development projects and situations
      1. What were the **software testing techniques** that you employed in this project? Describe their characteristics using specific details.

* The software testing techniques that I employed are automated, functional, unit, and acceptance testing. Throughout the past weeks, I created automated tests that tested the software of its respective class. Automated testing involves the use of software tools to automate the testing process. These tests were automated since we would just need to right-click, run the J-Unit test, and have Eclipse do the rest in a matter of seconds.

While programming test cases, however, I employed functional and unit testing for each of the class requirements and the methods I created for each class. Functional testing is when we test for the functional requirements of the software to ensure that they are met and unit testing is when we test the individual units or components of the software to ensure that they are functioning as intended. I implemented functional and unit testing by creating a test case for each method that needs to be tested referring to the class requirements while developing the classes and their tests. I also implemented acceptance testing which ensures software meets the end user's expectations. This is done through testing if the class variables are null or do not meet software requirements and outputting an error if invalid variables are inputted.

* + 1. What are the **other software testing techniques** that you did not use for this project? Describe their characteristics using specific details.
* The other software testing techniques that I did not use for the milestones are manual, integration, system, performance, and security testing. Manual testing involves manual inspection and testing of software by a human tester. This could be a developer testing input values that for variables in the system and making conclusions based on the console output results. Integration and system testing go hand in hand since they test if system components work together (integration) and if all the software works as a whole (system). This can be done by testing the classes that use multiple objects and the main driving class that implements the system’s entire software. Performance testing determines the performance of a system including speed, scalability, and stability. This is done by seeing how fast a method runs and the consistency that it runs at. Security testing identifies vulnerabilities and ensures that a system meets security requirements. This can be done by conducting a vulnerability and dependency check on the system (i.e. a Maven dependency/vulnerability check).
  1. Mindset
     1. Assess the mindset that you adopted working on this project. In acting as a software tester, to what extent did you employ **caution**? Why was it important to appreciate the complexity and interrelationships of the code you were testing? Provide specific examples to illustrate your claims.
* In acting as a software tester, I believe that I was very cautious when considering the software that I was programming. I believe this because I considered all possible outcomes for a user, developer, and requirements for the program. An example of this is that I used 3 different constructor methods for every feature of the software application (contact, task, appointment) and had 2 different add object methods for their respective services.

It was important for me to appreciate the complexity and interrelationships of the code because, in the end, it is all part of the same program. With the software that needs to be programmed, aspects of the code need to be able to talk with each other, especially with objects and their services in this project. Having the object be able to interact with its respective service will let the application conduct all necessary software requirements. An example of this is when an appointment is added, removed, or invalid in AppointmentService.java. An appointment can be created to be added to the service, removed when the service needs it to and can be invalid if an incorrect date has been inputted into the system. These actions for an appointment object show its relation with its service and how it interacts with the whole software application. Without the interaction between the two classes, the software application will be unable to schedule appointments for users.

* + 1. Assess the ways you tried to limit **bias** in your review of the code. On the software developer side, can you imagine that bias would be a concern if you were responsible for testing your own code? Provide specific examples to illustrate your claims.
* Ways I tried to limit bias in the review of my code is to expect that a user or developer will make an input error (we’re only human) and not assume that everything is working well after compilation. Not making any assumptions about what people or the code will do helps me think of different outcomes, what I should do with those outcomes, and how I should overcome the errors. While I programmed, I set up console outputs as I coded as a precautionary measure to ensure that all my code was working as intended before going further. I also asked my husband (who has no programming knowledge) to always try to break my code. I believe on the software developer side if I was biased about my code being the best and will always intend to work as it should with limited testing, it will have more errors and possible security problems without peer feedback on my work. With peers looking over your work, their feedback can help you notice flaws in your code that you did notice before. They can also have suggestions that can better your code. Without constant testing and feedback, code will be lacking if you were biased and always believed it will always work as intended.
  + 1. Finally, evaluate the importance of being **disciplined** in your commitment to quality as a software engineering professional. Why is it important not to cut corners when it comes to writing or testing code? How do you plan to avoid technical debt as a practitioner in the field? Provide specific examples to illustrate your claims.
* It is important to not cut corners when it comes to writing or testing code because it could cost problems in the future. Though it may be faster to do it one way, it will cost you the quality of the software if best practices are not followed. As a software engineering professional, knowing the best practices for your code will help commit to quality code in your software. With quality code, it can also help others follow your portion of a code base especially by not skipping commentation and explanations within the code. Though portions of code, like comments, or constant testing, can be tedious, it will be more helpful in the long run. To avoid technical debt as a practitioner in the field, I plan to apply best practices for the code base and constantly conduct testing throughout development. I also will try to break the limits, depending on the code I am working on. Some examples of what I will do to break the limits within my code are to test the bandwidth limitations of the code, the processing speeds of the code data, and also the security of the code by testing for possible data leakage. By breaking the limits within my code, as well as following best coding practices, I believe that this will contribute to a more disciplined code that will mitigate any need for technical debt for the software in the long run. This will also help our development team understand our system constraints and make a plan to combat the system’s stresses.

References:

GeeksforGeeks. (2023). Software Testing Techniques. *GeeksforGeeks*. https://www.geeksforgeeks.org/software-testing-techniques/

*Technical Debt*. (2021, September 2). https://www.productplan.com/glossary/technical-debt/