Elorha Newcomb

Instructor: Angelo Luo

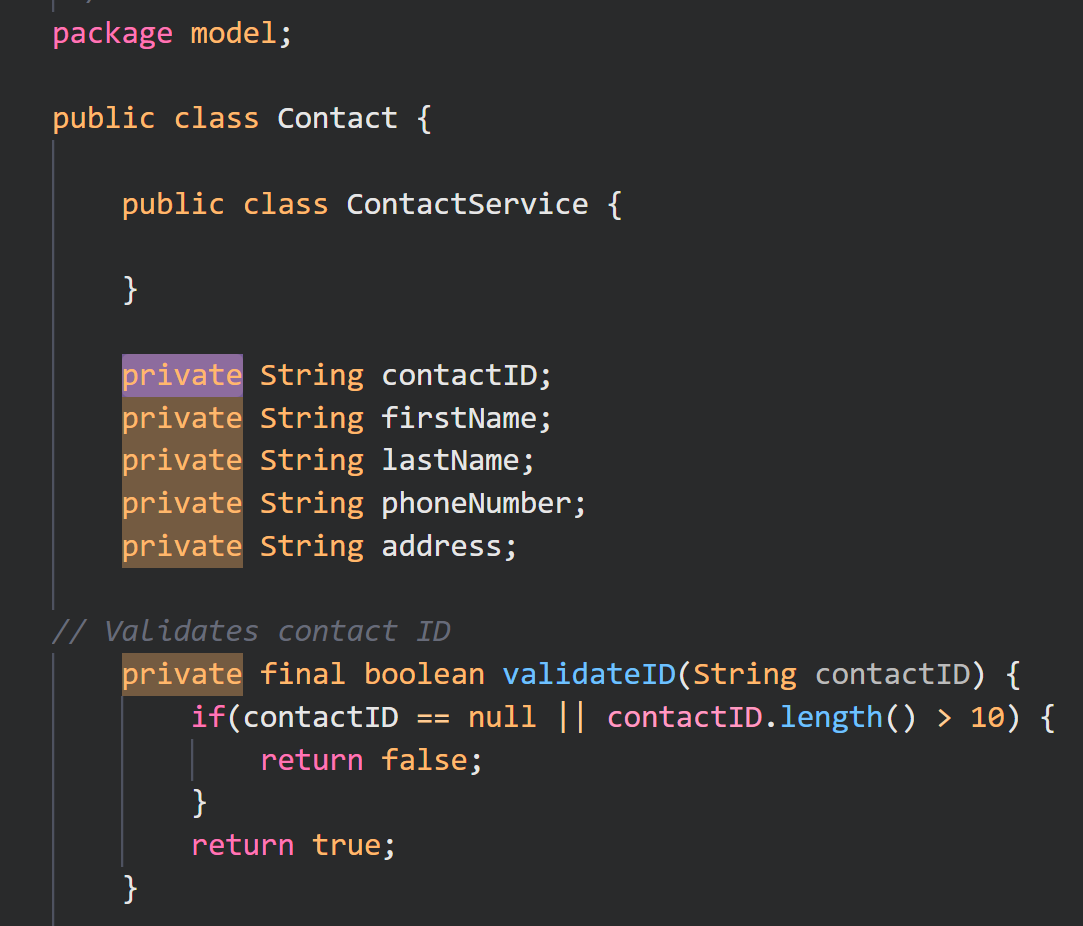
CS-320 – Software Test Automation & QA

21 October 2022

6-1 Project Two

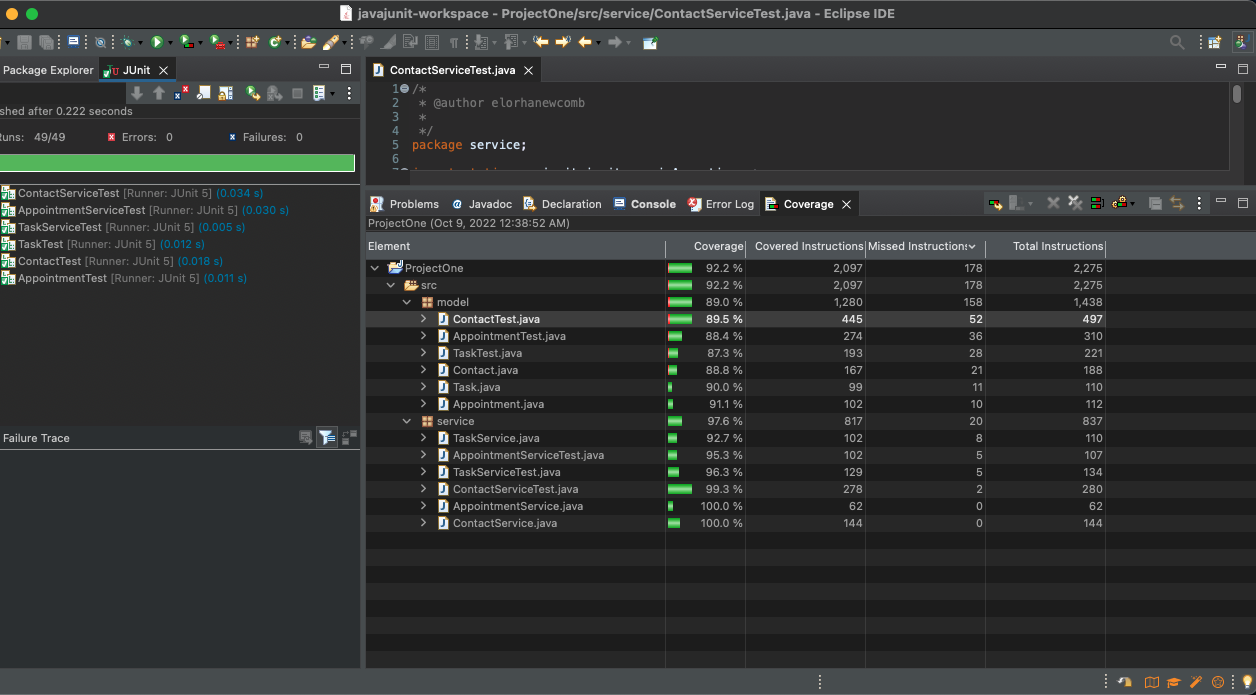
**Describe your unit testing approach for each of the three features.**

* To what extent was your approach aligned to the software requirements? Support your claims with specific evidence.
* 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?



The code I created was relatively straightforward as a result of the requirements provided. Using the assignment Contact Service above as an example, I ensured that my variables were being protected against malicious attacks by employing the principle of encapsulation and providing them with getter and setter methods. The code for adding, deleting, and updating the ContactService.java file was created using the setter methods to ensure the variables were of the correct length and not null.

During my development I made sure to tackle all the requirements given by the project, and experienced some trouble during the creation of the *Appointment* services due to the difference in requirements compared to the previous two for *Contacts* and *Tasks*. However, the biggest challenge for me was definitely starting the codebase for *Contacts* from the ground up. I had never used JUnit testing before, so I did not know what that entailed or what was necessary for me to use it. Even though we had access to a video explaining its usage, it was still not clear whether the JUnit would generate its own files (like a *pom.xml* file, for example) after I had created the testing, or if it should work as a simple processor application. My discovery was something in-between, since you can generate the test files by either creating a whole new test file by clicking with the right-mouse click and choosing a JUnit file or you can paste from the previous objects and adapt the code, as I did for the subsequent JUnit tests for *Task* and *Appointment* after creating the first tests for *Contact*.I believe my tests executed were functional and efficient based on my test coverage of 92%.



**Describe your experience writing the JUnit tests.**

* How did you ensure that your code was technically sound? Cite specific lines of code from your tests to illustrate.
* How did you ensure that your code was efficient? Cite specific lines of code from your tests to illustrate.

I ensured that the tests covered all methods and adhered to the given requirements, that exceptions covered the extensions of errors that could be thrown from invalid inputs. Using the specifications listed in the rubric, I used my constructors to validate the amount of characters, check if they were valid and also not null, and instructed the system to throw an error whenever the runtime didn’t meet any of the required guidelines.

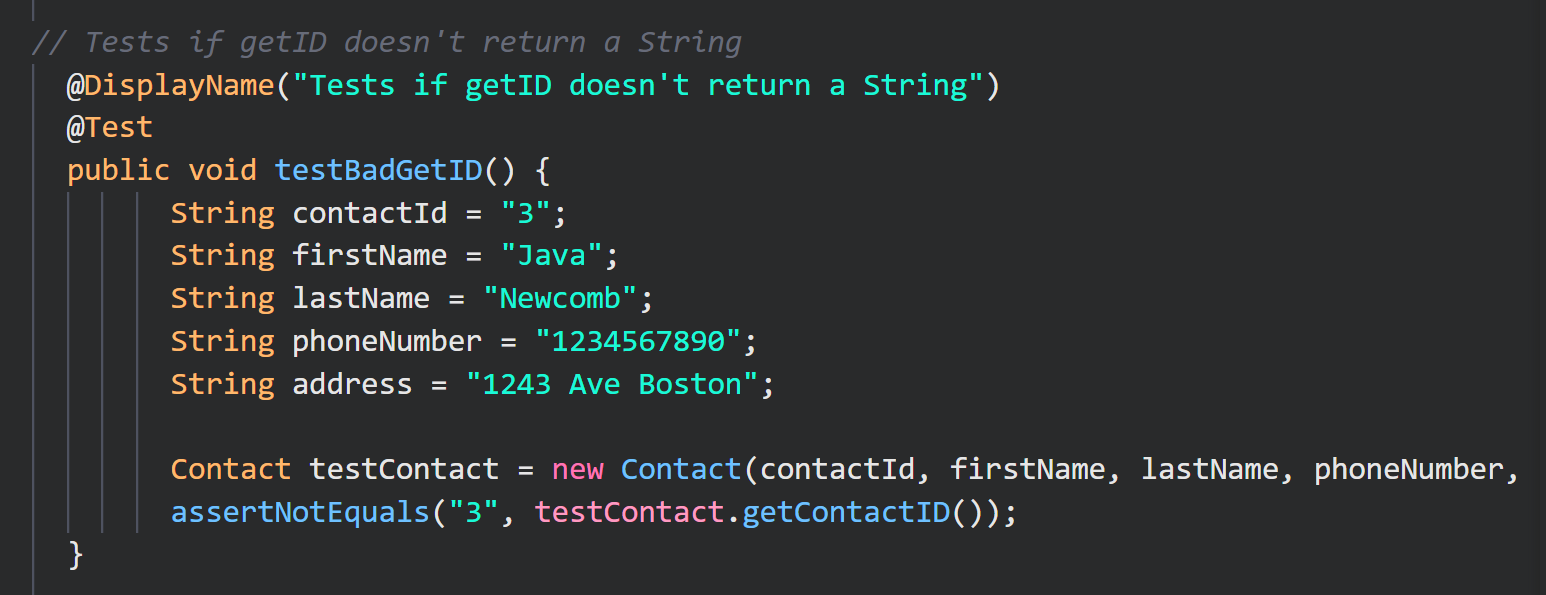


Having never used Junit before, I am unsure how efficient my system actually is. The only reliable source of information I acquired came from the coverage test, which showed satisfactory enough runtime results. I believe that I could rewrite these test cases more efficiently by doing some research and practice on my own time, thereby eliminating a considerable amount of redundant code.

**Testing Techniques**

* What were the software testing techniques that you employed in this project? Describe their characteristics using specific details.
* What are the other software testing techniques that you did not use for this project? Describe their characteristics using specific details.
* For each of the techniques you discussed, explain the practical uses and implications for different software development projects and situations.

The main element in my project was the use of *assertEquals* which is an example of Structure-Based Testing. This technique is one of many software testing techniques employed in software development that explores the structure and design of a system or component at several levels to verify their input and output. Additionally, I have also used black box testing techniques such as decision table testing and drafted flow charts outlining how the program would function under certain circumstances.  
 According to how I had configured Contact Service prior, I developed decision tables that described how a program should handle valid and invalid parameters as a base for reviewing and revising Task and Appointment service classes.



The techniques used for this assignment worked well, but other testing techniques would have worked better for larger programs. This program was so simple, however, that many techniques were dismissable for its application. Integration testing was not comported, for example, since the project contained few components; performance testing also did not make sense since the program was easy to code, thus easy to protect. Therefore, I would have definitely enhanced security if it had been released to the public or I had been required to.

**Mindset**

* 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.
* 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.
* 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.

The mindset I adopted throughout the project was one of curiosity first. This is because I was learning something highly intriguing and beginning to understand how significant the testing automation spectrum is for the successful completion of a project. In order to reduce human error, I kept the code as simple as possible during the steps, and understanding how the code is built and how the classes and services interact is essential to reducing the possibility of bugs. During the development process, I reviewed and tested the code according to well-known testing techniques, JUnit testing was employed for this project in particular. Although I recognize there will always be a bias towards your own work if you are assigned to test it, I have devoted many hours into performing and understanding all the tests for each instance of my code to ensure that the work is objective, clear, and to be always open to changes that optimize its applications.

A reason why cutting corners is a malpractice, among many other reasons, is that the Team, an image representation of the company they work for in the flesh, does not benefit from the use of automated testing if all the parties involved are not trying to prioritize quality over quantity. The scope of projects in the tech field are constantly changing, so are their collaborators, and when a Team is short-sighted enough that they do not create maintainable, clean, easy code that can be accessed and perfected years later, they are setting their own current work for failure. Practices like resorting to manual testing because the team was not able to implement proper automation for example only slows the process down. I also made sure the tests were returning what I really intended for each of them. Oftentimes, we can have code that "works", in that the program won't break right away, but it doesn't perform the actions it was supposed to either, resulting in a double error. It is crucial to sort out these interrelationships when one class calls another class' public method; some functions will fail if the class name is not spelled correctly, for example, a classic grammatical error. Testing for these calls and input variations can help the team reduce post production problems and especially data leaks after the application has reached its user base in its final project. Software manufacturers will be paying attention to the numbers and know exactly what to expect from the team as a result, even if they don’t initially know much about the start of the project themselves. In the event a company makes a faulty product, no one will buy it; a poorly-made product, released with no regard for its quality, will be reviewed low by users and shareholders, which will reduce sales. This is exacerbated by the rise of the internet and social media, which allows easy access to user’s personal opinions.