**Teodoir O’Ceallaigh**

**CS-320**

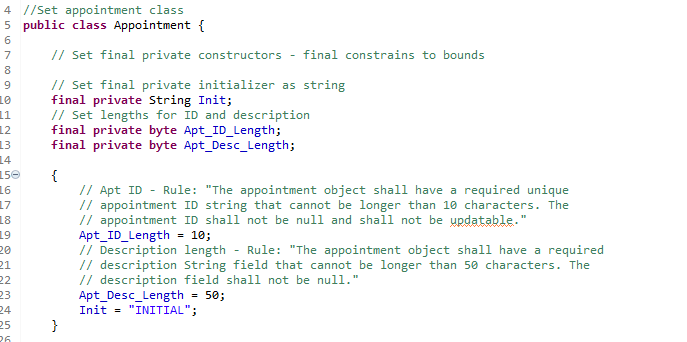
**Southern New Hampshire University**

**April 2022**

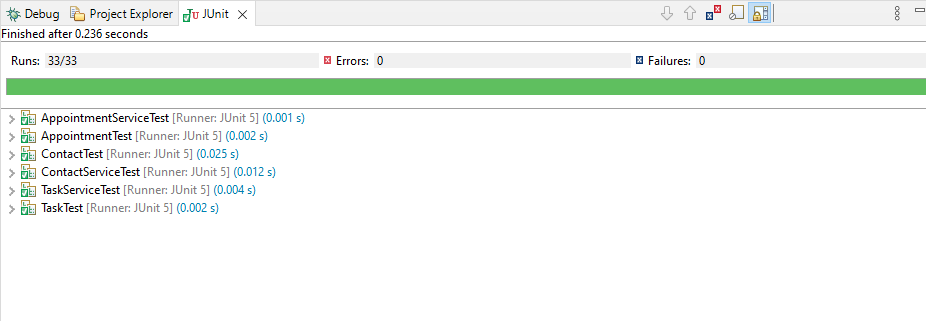
**Project 2**

**Summary**

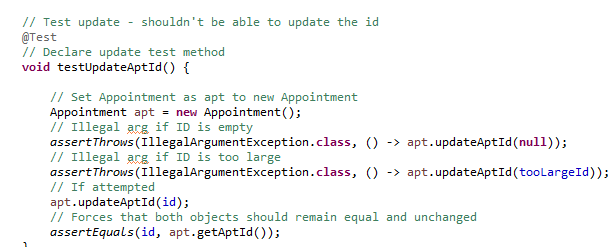
1. **My unit testing approach for each of the three features.**
   1. **To what extent was your approach aligned to the software requirements? Support your claims with specific evidence.**My approach to coding was to take a look at the requirements for each program portion and understand what was needed, what should be called back with an invalid/illegal exception, and what would allow for my testing to catch any errors in my creation process. A great example is when coding the AppointmentService.java I used code noting practice to outline the requirements given for the software and set the bytes in a format that would restrict them to a specific count.



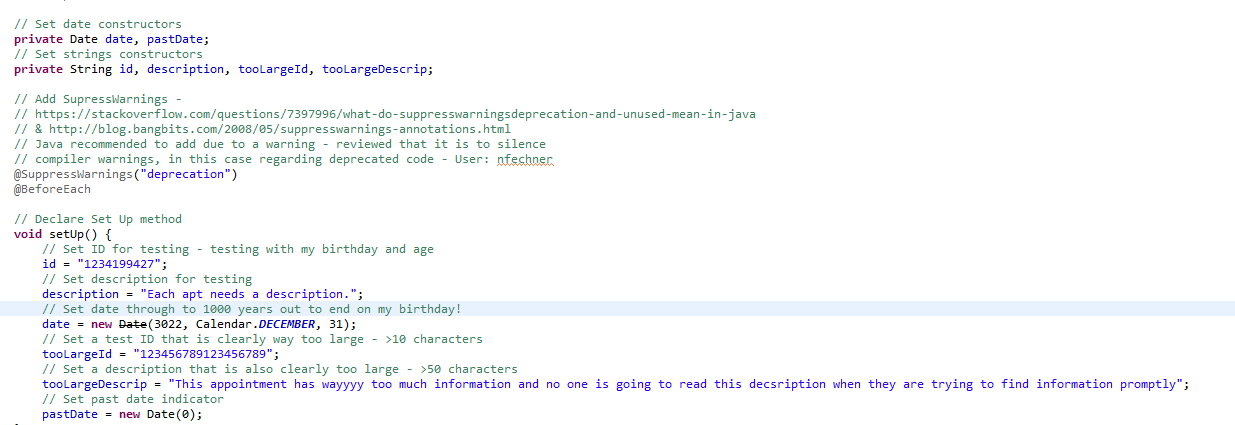
1. **Defend the overall quality of your JUnit tests. In other words, how do you know your JUnit tests were effective based on the coverage percentage?**  
   During the creation of my JUnit tests I also took extra care to understand each portion, what errors I would need to catch to show that they return an invalid argument, and made sure to continue the practice of strong code notation. With my full code running 33/33 with no errors or failures I feel confident in my JUnit’s creation and understanding how it worked with the classes created for each portion. When errors popped up - such as one case where I was getting an error due to copying the code so I didn’t have to manually type out all my notes and hadn’t amended the referencing numbers in the code itself - I was able to diagnose the issue swiftly and resolve the error.



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.**When writing my JUnit tests I tried to follow best practices for structuring, covering every use case expected to fail in the application of the software, and also reviewed potential code usages that would better allow my code to accomplish goals without being overbearing.



* 1. **How did you ensure that your code was efficient? Cite specific lines of code from your tests to illustrate.**While I had fun with my code to stay engaged during writing, I used easily remembered and replicable reference Strings or Integers so that errors would not be thrown for correct use cases when compared to the String and Integers that would not meet requirements.



**Reflection**

1. **Testing Techniques**
   1. **What were the software testing techniques that you employed in this project? Describe their characteristics using specific details.**The techniques I used for this project were Black Box and White Box techniques. Black Box techniques were used by fully understanding the requirements for the software, what it should allow or do, and what it shouldn’t allow or do. By employing this tactic I was able to align with the requirements of the customer and create a software, and JUnit test, that ensured the deliverable would be functional with what they wanted to see. White Box techniques involve structuring items appropriately so with this I worked to find quick formatting tips (ctrl+shift+f) and ensure that each portion of code flowed and was completed in steps alongside the requirements. I worked to make sure that each portion was written where the step during the constructing phase continued throughout the rest of the code so errors in line up did not happen during the JUnit tests.
   2. **What are the other software testing techniques that you did not use for this project? Describe their characteristics using specific details.**

As I have not written many programs nor to a deployable extent at this time, I was unable to use any experience-based techniques to review the code and apply experience-based fixes. There was a bit of googling with sources provided to find fixes or code implementations that would allow me to continue to accomplish a working code or work around a bug that may be simple but was not something I had encountered before.

* 1. **For each of the techniques you discussed, explain the practical uses and implications for different software development projects and situations.**Black Box techniques are great for having an outlined understanding of the code and ensuring that everything that is needed to function does so, and where errors exist, they are fully removed or properly throwing exceptions. This allows for the user to identify areas where there is not some sort of specification being met. For White Box, being structure based, this allows for potentially convoluted code to be dissected, rearranged, and error-ridden portions remedied through comprehensive understanding of syntax and formatting requirements in the programming portion. For Experience-based, this technique is most useful outside of the other two by having trained eyes on the code to find where the code may technically work but is likely to not predict some kind of input or outlier issue and the experienced user can preemptively build in a solution versus waiting to have to find a fix down the line.

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.**It was really important to begin by creating the code in a format that was going to be easy to read and review. It started from the very beginning in the creation stages! Understanding how quickly some code can turn into a ragged mess based on one spelling error is how you can approach the complexity of the creation process while still being able to regularly identify smaller errors, even during testing, that will allow you to avoid a much bigger headache down the line. Because of how everything interacts together, proper naming syntax, spelling, and formatting make the world go round for the creator and the tester.
   2. **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.**I tried to best create the code and test in a straightforward manner. As I don’t have the experience to complete experience based testing practices, I imagine down the line when I do have that experience I may find myself more biased on how code should be formatted, the type of notes that should be used, or what type of code is written to approach a certain problem. Perhaps I will have a never ending hatred of Booleans and think that every code should have conditionals used to provide the right output. While this is a silly example, I don’t think I have the right experience to justify any bias at this time.
   3. **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 importance of correct and secure coding practices is astronomical. Certain code may be intended for one project, turned out as open source, and be responsible for larger systems. Code I create today with no security and minimal software testing could potentially find itself being utilized in a system that contains personal information and would very realistically be open to penetration attacks and other hacking attempts or result in just a full breakdown of the software entirely. By writing thorough code with a full understanding of the specifications and well tested, I can ensure that the code I put out as a developer and tester meets my expectations as someone who wants to create and provide the best in everything I do. I plan to continue working hard to better understand the field, moral and ethical expectations, as well as further my experience so that I can recognize where I can take or create action before code begins to crumble. I very much believe in measuring twice, cutting once and don’t want code I create to ever negatively impact the end user or the client.