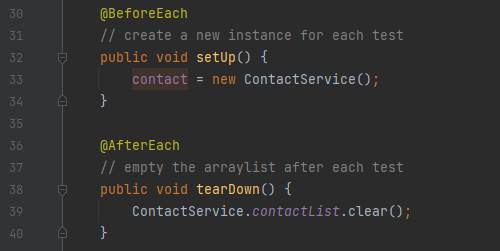
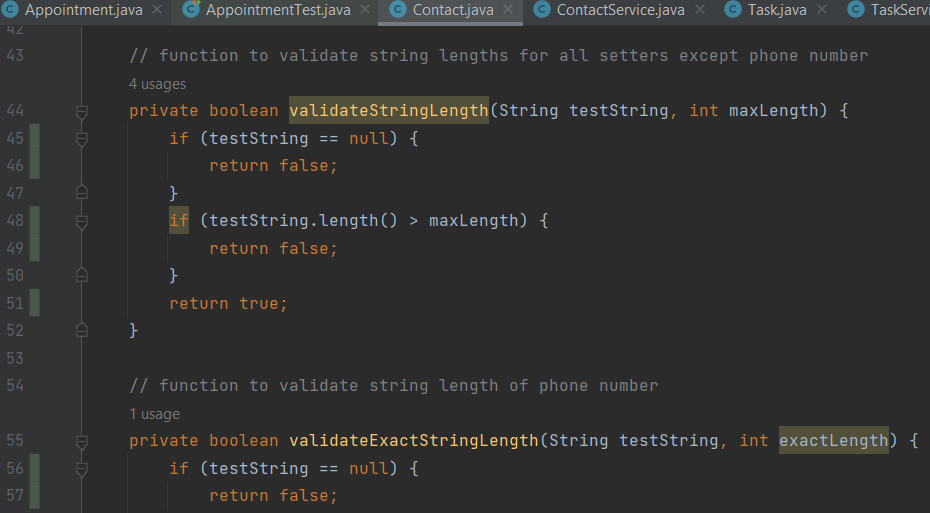
Project Two  
CS-320 Software Test Automation & QA  
Emily Wood  
Ramsey Kraya  
10/03/2022

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.**  
         When I first created the code, I wanted to make sure that the requirements around null entries and length of entries. I created test cases to ensure those entries threw exceptions when an invalid entry was received. I also wanted to ensure there were methods for each requirement, ie. phone number. Each method needed a test case. I used IntelliJ which created empty test cases for me with a single click. I also had to create variables that would pass and fail so all requirements were tested.
      2. **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?**  
         “Code coverage is a metric that can help you understand how much of your source is tested” (Pittet). The higher the number, the more lines of code are coverage by the tests. When I tested my code with coverage, all points (class, method, line) had a 100% rating. I would say this percentage means that my unit tests were well constructed.  
           
         **Graphical user interface, text

         Description automatically generated**
   2. **Describe your experience with 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.**  
         I learned new tips and techniques through my failing tests. For example, my individual test methods were passing with flying colors, however, the entire test class would fail. This was because the linkedLists were not cleared after each test case. I researched @BeforeEach and @AfterEach to implement and the tests passed as a whole.  
           
         ****  
           
         The tests themselves made sure that my code base was technically sound. It’s a little different creating code without a main to execute it. I also had a software engineer peer perform a static review of my code prior to completing the tests to point out any errors.
      2. **How did you ensure that your code was efficient? Cite specific lines of code from your tests to illustrate.**  
         I started by declaring all variables that I’m going to use per the requirements. Then I used IntelliJ’s shortcut to create required setters and getters. For the actual service code, I commented out the requirements and then created stubs. This ensured that I was hitting every requirement and there were no unnecessary functions.  
           
         For example, lines 28 – 38 in ContactService started out as comments, then stubs, then eventually code.  
           
         Text

         Description automatically generated  
           
         As previously mentioned, I had a peer perform a static code review. He always suggests to me to add functions for efficiency and readability. I originally had each method verifying the strings (not NULL and length). I created functions to be called in every method, so verification was efficient.  
           
         
2. **Reflection**
   1. **Testing Techniques**
      1. **What were the software testing techniques that you employed in this project? Describe their characteristics using specific details.**  
         I first used the static testing technique of a Peer Review by having another software engineer informally review my code and tests. I then used the Dynamic Testing technique of Structure-based testing with code coverage. I tested the code coverage by statements (number of statements of code exercised), decision coverage, and conditional coverage. In my test cases I made sure each validation statement was exercised by creating variables that would pass and variables that would throw exceptions.
      2. **What are the other software testing techniques that you did not use for this project? Describe their characteristics using specific details.**I did not use the static testing technique of Inspection. This is the most formal category of review where documents are prepared, and defects are logged. I did not use the Experience-Based technique of Exploratory testing. This testing technique uses a hands-on approach that comes with years of experience that I do not yet have. Test design and execution are carried out simultaneously without documenting the test design steps. Within the scope of these milestones, it was helpful to me to have the tests planned out prior to coding and executing. I also didn’t use any testing technique related to changes. The code base didn’t change and the tests themselves didn’t warrant many changes, so no additional defects were found and required retesting after changes.
      3. **For each of the techniques you discussed, explain the practical uses and implications for different software development projects and situations.**  
           
         Informal Reviews can be implemented in pair programming where two programmers work together in a constant cycle of reviewing each other’s code. Informal reviews are also beneficial in that they can save money and time because formal meetings and multiple personnel are not required. Testing with code coverage can be helpful in a safety-critical system such as a pharmacy system. It is very important to know that nothing happens in the code that you do not want to happen. Using testing with code coverage can exercise all possible paths within the code for the expected outcome. Exploratory testing maximizes the amount of testing that can be achieves within a smaller time frame, thus saving man-hours. This is especially helpful when the person coding has a high level of experience. Maintenance testing is helpful when a system needs to change for a variety of reasons, additional features are added, upgrades, system migrations, or system retirement. Testing when changes are made helps identify new errors.
   2. **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.**  
           
         I wanted to be cautious of all possible outcomes. That’s why I wanted to create test cases for each possible outcome. If the string entered had to not be null and be a certain number of characters, I wanted to test what would happen if someone entered NULL, if someone entered too many characters, and if someone entered the correct number of characters. Since the Service classes interacted with the other classes, it was important to understand how these classes interacted to create unit tests. For example, my tests would not receive 100% code coverage until the service classes were included. Some code in the base classes would never be executed until it was called through a service class.
      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 create tests that would pass and fail. As a developer, it can be difficult to test your own code in that you want it to pass. I also wanted to take on the mindset of the end user and future developers working in my code. As a developer, you can have tunnel vision to make it work at all costs. The end user needs to be kept in mind to be sure that it is meeting all the requirements. A future developer may need to edit my code, so it needs to remain clean.
      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.**As I mentioned before, I wanted to take on the mindset of a future developer who may have to work within my code. I wanted clean methods, comments, and clarity in what I was trying to achieve. Cutting corners can create a lot of future expense for the company you are working for. In our reading, it was explained that the cost of errors increases exponentially the further along in the Software Development Lifecycle. I like the idea of the review process prior to writing code to avoid technical debt. The earlier an error is caught, the less expense to my company, my customer, and ultimately myself is lost.