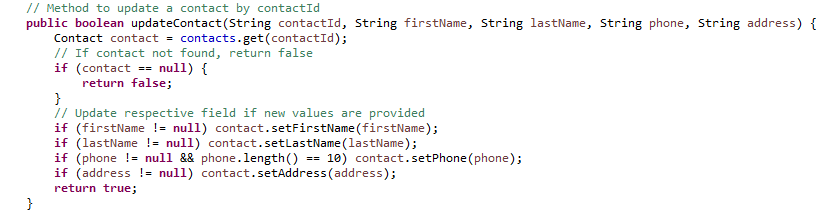
Tyler Doupe 10/19/24

Project Two CS 320

Following the development of the mobile application for Grand Strand Systems, along with writing respective JUnit tests to go along with the various services created for the mobile application, I have learned a great deal about JUnit tests while also further expanding my Java development knowledge. For the contact service, I was asked by Grand Strand Systems to include the features of adding, updating and deleting contacts from the mobile application. I was asked to ensure that each contact had a first name, last name, contact ID, phone number and address, each with their own unique criteria such as the phone number had to be exactly 10 digits and the first and last names could not exceed 10 characters. I was able to capture this and ensure these criteria were met by implementing various input validation strands in the code that would throw an exception if the rules were not met. Along with the contact service, I created a task service and appointment service. The task service and appointment service each had their own unique criteria that were outlined in the modules leading up to Project One as well as their own unique input validations that needed to be addressed. In a similar fashion, I implemented the input validation through exception rules that would be thrown if the input validation criteria was not met. By implementing strict input validation rules into my code for the mobile application, I ensured that the software requirements were met for each service I implemented into the mobile application. With the construction of each service, I built JUnit tests that would test that the software requirements were met. I was able to ensure that my JUnit tests were effective because I could see a 100% coverage percentage following the execution of my JUnit tests which instills confidence that the component executed as intended and met the software requirements that I outlined in the test. My experience with JUnit tests was very enjoyable! Prior to this class, I had never written a JUnit test before and frankly, I had never heard of a JUnit test prior to this class. Through the course’s readings, personal deep dive into JUnit testing, YouTube videos, etc., I was able to grow my knowledge of JUnit tests and ultimately expand on my capabilities as a developer. With all of that being said, in order to ensure that my code was technically sound, I implemented strict input validation checks in my JUnit tests. For example, for the contact class, I ensure that the JUnit test not only had exactly 10 characters, but I ensured that these 10 characters were exactly 10 digits:



As aspiring developers with SNHU, we are constantly told to exercise code efficiency. I am always trying to make my code as efficient as possible while also making it as reader friendly as possible so anyone can understand what I was doing in my code. One example of code efficiency within this mobile application can be seen in the task service with how I wrote the code for updating a task. I developed the code in a way that the update task method would only update the fields that were explicitly provided, thus avoiding unnecessary operations:

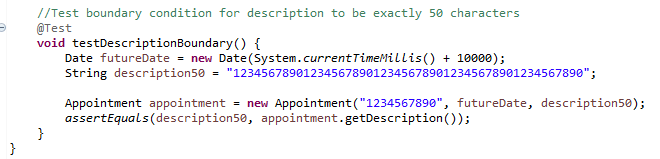


For this project, the main software testing techniques that I applied to my code were unit testing and boundary testing. As part of the development of the mobile application, we were asked to develop various components of the mobile application, along with JUnit tests that would individually tests the methods within these components. By implementing this type of software testing, I was able to test each individual method without any interference from other components or lack of components if they weren’t developed yet. Boundary Testing was the other major software testing technique that I applied to my code. Boundary testing allowed me to test the maximum allowed string length or an invalid input such as a null value in my code. An example of this can be seen in the task test when testing the boundaries of the taskId, name and description:



Unfortunately, with this project I was not able to use integration testing or system testing. Integration testing would have involved integrating the various components that I had built together and seeing how they worked together. The Modules leading up to Project One and ultimately Project One focused on the development of individual components of the overall mobile application thus I did not have the opportunity to test how the various components and methods I had built would integrate with each other and it would be an excellent step to take to further develop this mobile application. I also was unable to perform system testing for the mobile application as Project One did not request for the entire system build. Following the unit testing that was performed, I feel confident that with some work, the individual units could be integrated together with some tweaks applied and ultimately would fit well into the overall system build but at this time I was not able to apply integration and system tests to the mobile application. Overall, each testing type plays a crucial role in the development process of an application. Unit testing is perfect for testing individual methods and components without any interference. It allows us to detect bugs and code flaws early on in the development process. In the development of a larger project, unit tests would allow for us to test particular parts of a system without breaking an entire system to test something. Integration testing is also vital to overall system development as it can help detect where issues may arise when developing a complex system. Its possible that unit tests are all performing accordingly but when put together as a system, they are not working as intended or possibly crashing the system. Integration testing can help identify where the integrations flaws are taking place to ultimately allow for the system build to take place. Lastly, system test are important as not all bugs and flaws will be detected from unit and integration tests. Sometimes the entire system must be in play before you catch a bug or flaw that could be detrimental to the system’s overall performance and thus it is vital that when the system is put together that system tests are applied to the system.

In acting as a software tester on this project, I tried to employ caution through each component and respective test. As an aspiring developer, I wish to be a sponge and take in as much knowledge as possible and in doing so I aim to be very meticulous about the work I produce as I believe in the phrase “you play as you practice”. I believe if I apply caution to my homework assignments/projects in my SNHU courses it will feel more natural when I am performing this type of work for a future employer. Although for this project, the reality is that there are no true dangers to poor code quality or slight overlooks on developing my code when trying to meet the software requirements, but when I am working for an employer, they are trusting that I am the subject matter expert and will be putting out my best work. With that being said, a slight overlook such as allowing a contact to input more than 10 characters or even put in values that are not digits could potentially lead to flaws in the system. It may seem silly to think that something like this could lead to a large consequence but let’s say the phone number is the only way we can access the client and the phone number has no input validation restrictions. It could lead to the system not performing how it needs to or the inability for us to contact the individual. Being both the developer and tester of an application could be a dangerous trap when developing. Being the developer and tester could lead to a great amount of bias in that as the developer, you designed the application based on your understanding of the software requirements and when you say the project is finished, you are under the impression that the work is done correctly, accurately and ideally perfectly. Thus, when it comes to testing this application, you go into it knowing what you did and may not have a questioning attitude or mindset for why the code is written the way that it is or how it may meet the software requirements. As a tester you may have a different idea of what the software requirements are or even how the code should have been written to be more or less specific. With that being said, as the developer and tester of my work for Project One, I tried to be as specific as possible in hopes of being my own harshest assessor of my code. An example of this can be seen when performing a boundary test on the appointment service:



Lastly, it is extremely important as a software engineering professional that we are disciplined in creating the best quality code for our customers. Cutting corners can come at a significant cost as it could not only cost a company financially but it could also come at a reputation cost as well. By not producing the highest quality code and testing the code thoroughly, there could be significant impacts to a company’s reputation as it often only takes one mistake to lose the public’s trust with the company and the work that they produce. In addition to the loss of reputation, the company is often stuck paying for the issues on the backend which hurts the company even more. I plan to write comprehensive and thorough unit tests to ensure the work that I perform is complete and meets specified criteria. In time I hope to learn how to better integrate the respective components I build and ultimately build the entire system so I can perform all three levels of code testing. I plan to test my code earlier on in the development and more frequently to ensure I stay on top of my code rather than waiting until the end of a project to then seek out bugs and flaws in my code.

References:

GeeksforGeeks. Software testing techniques. https://www.geeksforgeeks.org/software-testing-techniques/

LambdaTest. (2022, July 21). 14 types of software testing techniques: A detailed guide. https://www.lambdatest.com/blog/testing-techniques/