Project 2

Rebekah Jackson

Summary:

For the project, I wrote several classes and Junit test cases to test the functionality of those classes. My testing approach was inline with the software requirements because for each requirement listed, there is at least one unit test. I tried to make sure that I tested every possible scenario. One example of this is that I added tests to check if what happens if I try to add a contact with a phone number that was both too long and too short. It is important to try and test every scenario to ensure that nothing gets missed.

I think the overall quality of my tests was very good. The Junit code coverage percentage for my test classes was around 72%, which I think is fairly good. For each requirement, I made sure to thoroughly test the corresponding piece of software. For example, I created additional constructors where an object could be added to the list by either creating the object and then passing it in or creating the object in line. This was not a specified requirement, but I thought it improved the quality of the code, so I implemented it and made sure to test it completely.

I made sure my code was technically sound by making sure that it ran as expected. For example, one of the requirements that I was given was to be able to add and remove a Contact object with the ContactService class. In order to test that, I wrote tests that added a contact, verified that the contact was added, removed the contact and then verified that the contact was removed. This test is line 39-47 of the ContactServiceTest.

I made sure that my code was efficient by ensuring that the processes all ran in the best way possible. For example, in both the ContactService and TaskService contain a list of objects that must contain a unique ID. In order to test that the code is working as expected, in lines 28-35 there is a Junit test that verifies that an object cannot be added unless it has a unique ID.

Reflection

Over the course of the project, I used several different testing techniques. The first technique that I used was unit tests. There were 3 object classes with 3 companion service classes. I was able to unit test the service classes by adding, removing, and updating objects within that service. I was able to write tests so that every piece of code was run. Unit testing is important to software development because it ensures that each small piece of code works the way that it is supposed to. Another method of testing that I did was white box testing. Because I knew the inner workings of each object, I was able to test it thoroughly. For example I was able to add in an extra test to see if a user entered a phone number that was less than 10 numbers. This kind of thorough testing will help find bugs that someone who isn’t completely familiar with your code might find.

There are lots of software testing techniques that I did not use in the project. One method is integration testing. Integration testing is when a developer tests that their code is able to successfully connect to and work with the other systems it is supposed to. There weren’t any systems to integrate with during these coding assignments. If I was storing the objects in a database or writing data to a file in a system somewhere, integration testing would have been crucial to the success of the project. Another method of testing that wasn’t used during the project is regression testing. Regression testing is when a developer makes sure that the new changes don’t affect any previously working code. Each week, I was writing new code and new tests, so I wasn’t really changing any old code. Therefore, I did not go back to make sure that the old code was still working. If I had made changes to the project that could have affected the other objects in any way, regression testing might have been beneficial.

I employed caution when writing the code and the tests by making sure to not try and get too complicated. I sometimes tend to chain too many methods together and over encapsulate objects. I was cautious not to over complicate the code so that it is easy to run, read, and understand. I tried to limit the bias I had in my code review by pretending like I was testing somebody else’s code. If I am able to critically look at my code and actively look for it’s flaws, I will be able to improve it. Finding bugs is not a bad thing, it is how we improve code. Finally, I was disciplined in my commitment to quality by ensuring that I tested every possible scenario for each of the classes. I had created some extra constructors for the AppointmentService class, and I made sure that I added extra tests in my unit testing so that I tested both of the constructors. It is important to always develop clean code and thoroughly test it.