When developing the Mobile Application project, I made it a point to stick to the requirements has tightly as possible. The Contact object proved to be the most complicated section of this application for me, since I was both just getting a feel for JUnit testing and refreshing my memory of java coding having only used it once nearly a year ago. I focused heavily on getting the core objects setup with their required variables and set/get methods. Once the basic object was created, I researched simple lists that would be easy to maneuver while testing. You can see in ContactService.java line 10, that I decided to go with an ArrayList due to its easy object handling and immediate add and remove methods. The creation of the methods was rather easy, however making the correct tests proved to be difficult the first time around. The first time I worked on this section I bundled several of my error tests together, which made finding the one that went wrong a little harder than it needed to be. If you look at my ContactTest.java file you can see how each statement that changes, creates, or receives information is tested to ensure the statement went through alright. Specifically, if you look at lines 47 – 125 you can see how each test focuses on a single piece of the whole object. The coverage for all of these tests comes out to about 80%, however the lack of coverage mainly comes from the throw tests I created. The creation lines in my throw tests don’t register as covered so the coverage rating takes a small decrease compared to what it covers.

Creating the Task Object and service was a much simpler task after practicing with the contact object previously. I continued to focus on the individual requirements given to us, along with adding a couple more tests to ensure more base functions of the objects created were working properly. The methods in the TaskServiceTest.java are more focused from the beginning making sure that no task is manipulated without the correct information. In the TaskService class I add lines 36-47 to allow the extraction of a single task from the list created. This is tested on lines 25-38 of the TaskServiceTest class using a couple of task objects. The coverage for this section of the application improved quite a bit after refining some of the tests a bit. The new coverage became 90%, but we still run into the error of not counting creation statements in my throw tests. All other sections of the code run through with no issues and each test focuses on a single error point possible in the task section as shown in my TaskTest class.

The final section of the application was the appointment handling service. The creation of the object and service proved to be slightly tricky, due to the old line of code we were advised to use. This code created some odd errors when trying to test this section of code but was easy to overcome once the issue was identified. One thing I must note is that the requirements for this section was shorter and less inclusive compared to the previous two. The date variable had no limit going forward but couldn’t be in the past, and the update method of the AppointmentService was no longer required. I created an update method regardless, to be consistent with the other sections of the application, as well as to make sure my checks were working on the different variables. This method was tested heavily between lines 68-143 of the AppointmentServiceTest class to ensure it wouldn’t introduce any new issues including changing the ID. I also created a PrintList method as opposed to the individual get methods that required the ID beforehand. The overall coverage didn’t improve any compared to the task section, however I did end up using a few more of the throw tests creating more coverage gaps in the test classes. I still ensure that each test covers one of the individual error possibilities that could arise from each method and labeled them accordingly for easy reading.

Testing techniques when developing this application were centered around the movement through the code and ensuring that each function only accepted logical inputs. The specifications for this application were rudimentary and only focused on ensuring input length was checked and no blanks were left. This wasn’t a difficult task however it left only a few tests to run. The majority of the test were focused around making sure all possible errors were handled and captured. This will pave the road for later should the program need to be added on to for other purposes.

I found it unnecessary to test for extreme inputs like command lines and ping requests, due to the lack of user input required for this application thus far. These tests are more meant to see if the program can catch a user trying to run commands through one of the input sections of the application. On top of looking for command lines, the tests look for inputs that would register as a halt in the code or possible commenting out a large section to avoid security. We can also write tests to test the connection between a server and a client or how the data moves between the two. This can prove incredibly useful for a company or website looking to ensure security; however, the current application doesn’t require any of that information. These tests are made for more complex code that is handling more data and possibly many more connections. Simple error testing is the best candidate for our mobile application for the time being.

When I made my tests, I had the mindset of making sure the functions were able to run successfully and they caught the errors outlined in the requirements. I wasn’t exactly aiming to break my code by throwing in random symbols and possible command lines, however I did want to make sure the requirements were touched on thoroughly. The service classes were the biggest worries due to how many different libraries and classes they accessed. It wouldn’t be that hard for information to get manipulated or misinterpreted between class calls.

Now as everyone knows we as people show a bit of bias in things that we have a hand in. I am no exception to that rule. People don’t want to see fault in their own work and when it comes to school even less so. I tried to keep myself to a cycle of testing that went through each section of the class to ensure I touched on all methods and functions. My small level of discipline could be speculated to that of a larger company using more complex code. Developers on their end could be tired of looking at code or wanting to ensure a deadline is met and overlook possible issues or avoid certain tests. Even overconfident developers will ignore tests and overlook small errors believing in their own ability. This could cause you to release a broken program that could potentially cost tens of thousands to fix if that proves possible after the public backlash.

To top off everything it remains incredibly important to remain disciplined when it comes to code quality. Sloppy code can lead to a load of weird errors that will end up pointing to other places. This can also stand true with the programs and libraries you write your code with. If you write out a program on an old system with libraries that have gone out of use, then anyone who goes to work on or run your program on any other system will encounter numerous errors. This just creates more work for the next person and could possibly lead to your code being scrapped. I plan to avoid these possibilities by keeping my libraries and program up to date so I have all of the working libraries. As another safety measure I will be trying to run all of my programs on multiple computers to ensure that the application is not limited to my system of development. I’ve come to learn that no errors for one person does not mean there will be no errors for someone else with potentially the same setup.