Over the past three modules I have successfully designed and tested 3 separate applications designed to do different things (although they are all very similar). For each, I employed JUnit methods and Eclipse syntax highlighting to ensure all of my own methods were covered to almost 100%.

As I mentioned in my last journal, my favorite JUnit method is “assertAll”. This allows for running multiple assertions at once which allows for organization as well as the lifting up of repetitive code. I used several other assertions as well such as “assertTrue” or “assertFalse, “assertNotNull”, and “assertEquals”. Out of all these assertions, I found that while “assertEquals” was probably the most versatile because you could check if your returns were equal to anything, most of my methods could be boiled down to one of the other assertions which were more efficient and required less code.

There were plenty of software development techniques that I did not use in my code as well. For example, having followed the requirements document in Module 5 as closely as possible, while I needed to be able to construct an Appointment with a date and a description, I did not need to edit them or display them with the service, so I did not write getters and setters for those fields because they were not needed. In a real-world application they would probably at the very least need to be displayed and therefore I would need getters for these private fields.

So far, I feel like I am getting a decent grasp on JUnit and testing. I was surprised at how simple it was to understand and how much JUnit does with so little. As an example, I noticed that JUnit does not have as many assertions as I thought it would, but if one thinks about it, more assertions are not really needed! Java is strongly typed, and methods can only return what is specifically defined within them. Working with only this simple paradigm means JUnit can be just as unforgiving, which is good for testing. Using JUnit on larger, more complex applications would be a boon to developers as such programs can quickly become unwieldy and yield unwanted results.