Project Two

Shayne Greene

Aug 21, 2025

My approach to unit testing grew with each program as my understanding of each Junit test became clearer. Each of the programs had similarities in their requirements, I was able to incorporate the development of the unit tests from one to the next. To be specific, each program had both a requirement to be both not null and under a certain character length. Once I was able to determine solid testing methods, I use that structure for each of the tests. This approach helped keep the software requirements at the forefront. Making sure that each of the main programs had a test to mirror the criteria. Taking the Appointment as an example, for each of the requirements, there is a unit testing method to match or methods in the main program that aligned to the requirements. In the screen shot below taken from a method in Appointment.java, I am using a private method to set the appointment ID. Being private allows only the constructor to set the ID value. In this case, an instance of the Appointment has no visibility to the setID method.

A screen shot of a computer code

AI-generated content may be incorrect.

In order to measure the effectiveness of each JUnit tests I made sure a coverage test was completed with no less than 100% for each of the program’s tests. If I found that there were gaps or coverage less than 100% I would find the criteria that was missed and create a testing method to test that criteria. Then run the coverage test again to assure I captured 100%.

To ensure my code was technically sound I generated each test to align with the methods in the main program. For instance, I would always start by testing the constructor when creating a new instance of the class. Below, I am testing that when creating a new instance of the Appointment class, it is in fact returning the values that were used to create the object. In turn, this also tests the getters for the class.

A screen shot of a computer code

AI-generated content may be incorrect.

To ensure that the code is efficient, I used the setter methods to initialize the variables. This ended up limiting the amount of repeated code, specifically the tests for null and length of the value. Also when I test the creation of a new instance, the setters functionality are also being tested.

A screen shot of a computer code

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

The testing techniques I applied in this project were focused on function testing. The first and likely the most important is unit testing. After I developed a method, I would also develop the JUnit test to coincide with that method. This helped me verify that I was upholding all the criteria for that method. For instance, when I developed the constructor, a method that will allow the user to set all fields, I created the test to ensure that each value was being set correctly. Another technique I used was integration testing (Testsigma.).. For this technique, I was utilizing more than one object in a single test. In my ContactServiceTest class, I was testing a method in the ContactService class: addNewContact. Which calls the Contact class constructor to initialize a Contact object.

A screen shot of a computer code

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

There are also testing techniques that I did not use for this project. Those would have included acceptance testing. This would be a test conducted by the end user to validate that the functions of the program met the criteria initially set. This is one of the most valuable testing techniques I used in my professional career. Once the user can play with a developed piece of software, they can either pass it back with notes, or accept the software for release. A non-functional test that I did not employ in the project were performance testing. I could have used a semi-automated program from Apache called JMeter to simulate multithreaded users to initialize thousands of objects in a certain amount of time. This would not only test the speed of the program, but also the memory usage when that many objects are created. Not to go too off topic, but I had a recent experience with a JVM memory issue. When a user loads parts for this customer, they happen to have over 10 thousand parts. Typically, not an issue. The problem was the program was using a depreciated function that when clearing the grid, did not handle garbage cleanup as it should have. So, when the user loaded another customer, the program froze and shutdown unexpectedly. The fix was to replace the deprecated method with the current one. This type of testing is important not only for new development, but also to periodically test older programs that may have old land mines just waiting to get stepped on.

Setting up the JUnits tests did make me appreciate the importance of making sure the programs met all aspects of the requirements. There is a certain amount of caution that should be observed when writing software. If you decide to not test your programs in chunks, you have the potential to eat up a lot of time. Say I tried to write everything all at once without a single test. Yes, I could go over the program line by line and everything looks great. But when you run your tests, you find that a single mistake, one that you happen to reuse in all your methods, breaks the program and nothing runs…(speaking from experience). If you took the time to test each method as you went, the mistake would have been discovered in the beginning, and there would be no need to rewrite most of the program.

The beauty of using the JUnit testing is that it makes it difficult to induce your bias in your code. I suppose if you were also bias in your testing programs you could test around your mistakes. Testing your own code is difficult because it is tough to sit outside yourself and look at your code from a different perspective. Having a second person look at your code, they may see something that you would not have thought of. From more experience, of just plain having a new set of eyes on it.

Having a commitment to quality is and should be, the forefront of good development. At the end of the day, cutting corners will do nothing in the software development world except for destroying reputations. Users have a level of expectations from developers to be thoughtful in their approach to giving them what they are asking for. Cutting corners will only hurt you in the long run. Yes, your software may be completed earlier, or ahead of schedule. But the number of reworks will outweigh the benefits of being out early. This is sometimes seen with gaming releases. Developers are given these solid deadlines to release games. And there have been a number of times that a game was clearly not finished, or cut certain corners to meet these deadlines.

**References**

Testsigma. (n.d.). *Functional testing: Complete guide – Types, process and best practices*. Testsigma. Retrieved August 23, 2025, from <https://testsigma.com/guides/functional-testing/>