7-2 Project Two

Michael Wood

SNHU

Course Number: CS-320 Software Test, Automation

Instructor: Tanisha Jacks

Date: 04/18/2025

7-2 Project Two

I entered this assignment not understanding what to expect. I have never utilized any testing software or attempted to develop the tests required to ensure the quality of my code or application. First and foremost, my coding skills are not as advanced as I would want, therefore much of my code is trial and error. This becomes test one. Because my coding skills are limited and I utilize a trial-and-error approach to coding, I am continuously testing my code to see if it works or runs at all.

For this project I took the software requirements; for example, the Contact Class requirements were:

* The task object shall have a required unique task ID String that cannot be longer than 10 characters. The task ID shall not be null and shall not be updatable.
* The task object shall have a required name String field that cannot be longer than 20 characters. The name field shall not be null.
* The task object shall have a required description String field that cannot be longer than 50 characters. The description field shall not be null.

Mind you, before I started using Junit testing, I wrote code, ran it, and used my IDE to find and solve any bugs that arose. This initial step was repeated several times to ensure that the code worked properly and that I received the desired outcomes for the client. I believe that many flaws in our designs are the result of human error.

These problems can range from simple syntax (which is not always easy to spot even when you are shown the line of code with the issue) to forgetting to include a command such as "import java.util.ArrayList" although you know you have coded arrays into your current code. It will not function properly. How do I know? Quite simply, by trial and error. Write some code and then test some of it. I have discovered that handling tiny portions of code rather than vast amounts of code allows me to spot more bugs and correct them more easily. That is the beauty of classes: all your code does not have to be in one place, and you do not have to search through thousands upon thousands of lines of code to uncover problems or bugs, depending on the scale of the program or application being designed. JUnit testing is so effective because it allows you to work through each class individually.

I had never heard of Junit testing before this class. I was always under the assumption that we (humans) ran the application and tested the application manually. This class has opened my eyes to a whole other world of testing. I did my best to create the proper tests that would test my code properly and in some cases I think that I did very well and in others not so well.

So, you ask how do I know that my code was technically sound? Simply put, because it works, and the outcome is as predicted or as per the client’s requirements. Now, that is the simple answer. The not so simple answer is through testing. Much of the testing that I did was unit and boundary testing. Due to the client’s requirements, I thought that this was the best approach.

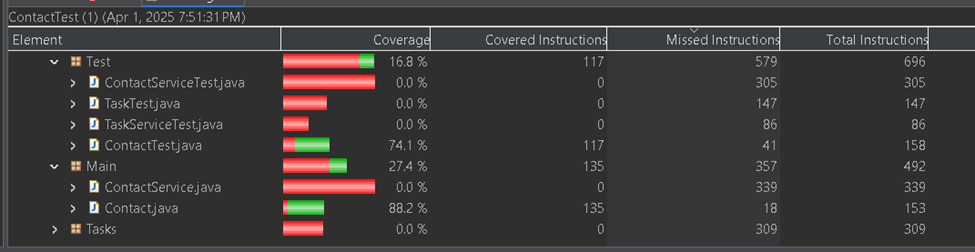
Unit testing was also carried out by testing individual components across the code. Unit testing is faster and more efficient than testing all the code at once. I encouraged unit testing that included boundary checks and error handling. The primary benefit of unit testing is early detection of errors, higher code quality, and increased confidence for developers and coders because they can validate that their code or product is functioning as intended.

My research indicates that boundary testing is highly effective for swiftly detecting flaws. This form of testing ensures reliability and quality. This type of testing reduces overall effort and time by evaluating a broad spectrum of input values with fewer test cases. When executed correctly, boundary testing guarantees that the product responds as anticipated.

Boundary testing is done using the function input ranges. I did not bother to test the bare minimum because I did not think it mattered for these assignments. evaluating the upper and lower bounds. It would be necessary to input some type of character for the system to work correctly, as none of the inputs could be NULL in these cases. The next step was to run the tests on the typical values, which were slightly higher than the maximum allowed values.

I believe that this type of testing was effective as shown below:





Contact Service test covered 64.6% of the instructions. The Task Test covered 28.6% of the instructions. The Task Service test covered 74.4% of the instructions. By checking the coverage, it lets me know how much of my code is being tested.

When it comes to Junit testing, I realized I need more education, experience, and practice. Testing was not thorough enough when I was able to get it to work because I did not receive the coverage, I had hoped for with my Junit testing. It would occasionally fail to import the libraries required to run the Junit tests correctly, for whatever reason. This means I should focus on checking my libraries and configuration for errors.

Although this paper falls short of the assignment's expectations, it represents my best effort given the resources I have available. I feel like I have barely scratched the surface of what this course has to offer. To refine my abilities and make my Junit designs and tests better, I know I need to practice and use what I have learnt.