Jordan Colbert

CS 499

June 19, 2022

Southern New Hampshire University

Milestone Two Narrative

The project included as my software design artifact was a program created in a course taken here at SNHU, CS 320. It was an assignment that allowed me to demonstrate my ability to create tests. The program takes fields such as name, ID, and address, and adds them to a hash map. The information can then be read, updated, and deleted after creation. It works in tandem with another file in the package that throws exceptions based on the entries falling within the specified parameters. It retrieves information based entirely on the user feeding it to the program. Therefore, testing is done by using sample names fed to each method, with the tests checking if the hash map populates correctly, if the delete and update functions work, and if the entries can be retrieved. The program was originally created in 2021, towards the end of my computer science program.

I included it as an example of my software design abilities, as it is a very simple, yet effective program. It sets out to easily create a CRUD function for user-generated inputs that works as it should. Additionally, I designed each test to be rather rigorous, making sure that any possible issue was caught in one way or another. This even meant including throwing assertions in the second Java class that was not even a test file. Naturally, this was due to the course being focused on testing and catching errors, but the design of the program is well thought out and logical. For example, the Contact Test file double checks that the length of each entry is correct in short, simple code. I simply input data that should trigger the assertion I created previously to make sure that it works correctly. This demonstrates that my software design ability goes beyond making simple, effective code, but code that is well-tested with multiple points of confirmation.

I enhanced this program by modifying one of the tests. Previously, I did not know how to test for exact results in my entries due to not having enough time to view how Java handled data being fed to it. Once I got a chance to run a “print” function, I figured out how the text was formatted. Previously, I used a “.contains” command to determine if the entry was or was not what I was testing for. This time, however, I used “.equals” to ensure that the correct data is entered into the hash map. All I truly need was a bit more time to test how the text really looked to make sure that the example I created matched what Java was being fed after formatting. The process was the simplest out of the artifacts I worked on for this project, with just a bit of experimentation needed. The experience proved that just a bit of time and exploration can make all the difference when designing a program. Sometimes, small, simple changes can ensure that a project is functioning as intended.