Summary and Reflections Report

Due to the similarity of each project specification, my unit tests also followed a similar format. Each test analyzes the function’s ability to handle invalid input and validates that the function performs its purpose. Each instance that accepts user input is passed through exception handling which ensures the value matches requirements before being stored. Unit tests are set up to test that exceptions are thrown and caught when invalid input is submitted. Other tests deal with functionality and ensure that add, modify, and delete processes have the correct outcome.

My code is technically sound in its implementation of abstraction and private variables for data protection. It is also maintainable, with specific, easy to understand, methods for each process. The basic validateInput method within each driver class can be updated easily to accommodate for different input specifications simply by changing one or two values within the try/catch block. The efficiency in my program can be seen in my use of methods for each process to avoid cluttering the main driver with code. Each method involving user input utilizes other methods whose purpose is to validate that input and only exits the validation process when correct input is submitted. These validation methods free up space and provide a layout that is easy to understand and navigate.

The software testing techniques that I employed in this project include unit testing, functional testing, and usability testing. My unit tests are designed to test individual units of the program in isolation. Their purpose is to verify the correctness of each function of the code. The functional testing, I performed manually to verify the program behaves as expected with different input scenarios. For usability testing I focused on the user experience and tested for areas that could cause confusion. I did not include user interviews, but I reviewed each menu option as a user and checked the language for areas that could cause confusion.

The tests that I did not perform include integration testing, performance testing, and security testing. Since my program has no external aspects (such as a database or web server) I did not perform any integration tests to check the interactions between different components. I also did not perform any performance tests. My program has low scalability without implementing an external database. In that case I did not test for different workloads, specifically large volumes of data. Lastly, I did not perform any security tests to identify weaknesses that could be exploited. If I were to include a database and utilize my program with a web server I would implement more abstraction and protected components.

I approached the project with the perspective that the user will always enter invalid data. With that in mind I utilized exception handling within each section that included user input. After writing my first test class it became habitual to format my code for ease of testing. This also aids in the understandability of my code. The requirement whose creation benefitted from this mindset the most was the validateDate method. I approached the method by coding a portion of it first, then worked on the unit test to identify how it might fail before continuing to code the rest of the method. I’m uncertain if I avoided bias, but I did my best to formulate my program with the mentality that each function would fail. That is why I included exception handling and input validation at each point of possible weakness.

One easily identifiable result of disciplined programming is gaining the appreciation of peers and supervisors. It becomes quickly apparent which employees have a habit of cutting corners and this impacts the whole team. There is a high likelihood that sloppy code will also come back to haunt the programmer when they must update their project or even redo it. In each company programmers come and go, but legacy code will always be there to cause tears. If each programmer took care with their work and testing, life would be much easier for those who follow. Those who write sloppy code will one day inevitably have to deal with someone else’s lack of discipline.