The program produced had been tested according to all provided requirements. Since the familiarity with the Junit library was limited, writing the service and main classes were not initially test driven. While an idea of the tests had been maintained, the program was developed in a 50/50 aspect. The approach taken was to write the program and its structures, then writing the test code to make sure the correct references were established. For these assignments a standard was given & those standards were repetitive throughout the course.

Those set of standards identified criteria for the String ID, String title-parameters {names, description} & a single occurrence of Date() which required the implementation of the Date() class. Those parameters required a few conditionals which were made modular and efficient by enumerating the set of conditionals to a separate method. Those methods would take one or more constructor variables, to define or redefine that variable to meet the specification.

For example, the String ids in the programs were to not exceed 10 characters long & not be null or empty. This method checks for nullity or an empty string, if either of those conditions fail, the method establishes a new user id ‘tempID’.Text

Description automatically generated

The effectiveness of the Junit tests was through according to provided requirements; however, had this been an industry standard set of tests (assuming that Junit tests run in an organization catch as many deviances or variances possible) this Junit test provides a relatively limited scope of testing. With time familiarity of vulnerabilities, errors, occurrences the capacity to provide a greater testing scope will begin to match and potentially exceed industry standards. This practice may give way to developing a Junit automated testing library.

Ensuring code is technically sound requires many levels of testing and various approaches in debugging and breaking code. While this code does not have any obvious flaws it could benefit from further optimization and refactoring to create an even more technically sound and more efficient program.

By ensuring code was not redundant and lines of code were removed if they did not serve a purpose ie. unnecessary accessor variables. Modular methods which would accept a variety of constructor variables keep performance optimized and caching data easier. For example providing a method which accepted various variables and made assessments based on attributes passed with that particular variable. Those attributes included Boolean and String data types which would identify incoming variables as one or another type of data.

Text

Description automatically generated

According to the definition of software testing detailed in the textbook, tests conducted were a style of acceptance, black-box & system tests. This is because the tests written were derived from specifications & not experience {black-box testing}. The tests conducted were on the program to ensure variables functioned properly {systems testing} & the tests also made sure those tests were passed through a set of defined standards {acceptance}.Graphical user interface, text, application

Description automatically generated

There were quite a few testing approaches not put in to use for these programs a few to mention would be beta testing, & integration testing. Since testing the programs individually did not necessarily mandate that the programs worked together as one it was not integration. Graphical user interface, text

Description automatically generated

In order to provide an impartial review of programs written reading code produced by professional developers is a practice considered having some sort of realistic idea of program implementation provides a source of comparison. There are many places to watch live coding session, as well as learn good coding habits. Mindsets that can be dangerous pits falls might include imposter syndrome, “if I keep changing something this program should work eventually”, “that’s supposed to do that”, “Im just starting programming I don’t need to know the importance of bitrate and how long my program takes to compile and execute.” Knowing the standards of a company also helps. Since there are infinite ways to tackle different issues in coding going with the industry defined standard is typically a good practice to make habit. Things like naming convention, and commenting can help guide ourselves as programmers, later in the program development. This is typically as cautious as programs like these will require right now. Memorizing typical patterns is a good idea, such as a singleton pattern & then searching algorithms are a great way to skip over the mundane low-level problem-solving situations that arise.

Text

Description automatically generated

Becoming disciplined as certain individuals has been proven throughout history, can mean the difference of, retrieving rock samples from asteroids {Osiris-Rx}, orbiting the sun at a million plus miles / hour {Parker Probe}. Undisciplined individuals may endure troubles in committing file in / out libraries to memory or having to research code that could easily be committed to memory such as basic search algorithms. Cutting corners could be a dangerous phrase to use with programmers and coders that is how we optimize. If cutting a corner in a sequence could save 3 bytes in memory that would be a good thing. However, cutting corners in programs via security, could mean a high-school bug hunter could make a million dollars from someone’s laziness. Avoiding technical debt would be easily remedied by staying up to date with industry practices and the latest open source contribution, or even better developing a library that helps solve one of programming’s technical deficits is a great way to maintain mental acuity.

Resources:

Hambling, Brian Morgan, Peter Samaroo, Angelina Thompson, Geoff Williams, Peter. (2015). *Software Testing - An ISTQB-BCS Certified Tester Foundation Guide (3rd Edition).* BCS The Chartered Institute for IT. Retrieved from   
https://app.knovel.com/hotlink/toc/id:kpSTAIST01/software-testing-an-istqb/software-testing-an-istqb