David Droege

SNHU CS 320

June 22, 2025

# Project Two

- **Summary**
  - The general answer to this question is that JUnit tests have been used in each of the three features. In each of the features' unit tests were used to ensure that adding, deleting, and editing data works properly. This was done by inputting valid and invalid data values to ensure that valid inputs are cleared and invalid inputs are caught. In the Appointment code we also used a date which required using a tool that pulls the current date. A test was then run to ensure no invalid dates were inputted. When running the final product as a JUnit test, we get the result that 100% of tests passed. 100% is a very good sign, but a review is required to ensure the validity of that number. The next section will review the thoroughness of the tests employed and will display that the 100% is accurate. Proving that the tests were effective and of good quality.
  - For code to be technically sound, it needs to properly use functions and commands. For the tests and base code employed, the proper tools were imported which can be seen in the top of each java file for this project. Importing the proper files and tools ensures that there will be no error of unrecognized commands.
  - Efficiency should be a biproduct of technically sound code. As mentioned in the previous paragraph, importing tools ensure technical soundness and efficiency. By importing tools, we save the work, time, and redundant code of defining the

functions and variables provided in those tool sets. Again, as can be seen in the first few lines of code of each file, the proper tools and files are imported.

- **Reflection**
    - **Testing Techniques**
        - The techniques employed in this project were mainly trial and error tests. In each of the tests done, the approach was to input a sample input that a user could enter. Both correct and incorrect samples were entered. The goal was to ensure that the valid inputs were cleared, and the invalid ones were stopped. By using JUnit tests, we ensure that the tests properly clear and catch valid and invalid responses. Another testing method that was not employed was physical testing. Sending the code to a tester and having them run the program to test for errors. As physical testing is good practice, using JUnit tests saves time and money. Essentially what a tester would tell us is being told to us by JUnit tests quicker. Having a person test code requires their time to go through the program and try every feature, work that can be done by a computer in milliseconds. It is more practical to employ testers when the program relies on user experience and the user interface. A JUnit test is unable to tell you how aesthetically pleasing a visual is, but a physical tester is. For programs more focused on security and code functionality, JUnit tests are better as they cover the most important details.
    - **Mindset**
        - I learned that the best employment of caution is to be thorough. It is important to know that no stone was left unturned when testing. Leaving

even the smallest function untested can be a tool used to break the program and leak information. Security is a large player in the coding world and exposure to malpractice can be very damaging. On that same note of security, it helps build appreciation for good code. For example, if you were to store your precious diamonds in a vault, you would be satisfied to see a vault that is heavily fortified and secure. Good code follows the same principle. Testing and reviewing code that is clean, complex, and well secured scratches an itch for a tester, leaving satisfaction that the users are safe. Like any piece of work, it is important to be able to step back and have an open mind. Having bias for your own code could lead to you leaving vulnerable features, due to your desire to keep that personal touch. From an unbiased perspective its easy to critique complicated code, but from a personal perspective its hard to be harsh on your own work. Lack of bias and discipline go hand in hand. You have to be disciplined in order to lack bias. Maybe you have a personal technique to execute or define a certain function, but in the grand scheme of efficiency and quality, it is better to use a more modern technique. Discipline requires that you set your personal bias aside and do what is best. The best being the most secure and efficient. When testing, the same applies. Discipline requires that you be thorough and test every possible outcome. Doing a general test that covers the general bases will not ensure the same amount of security that a scrutinous test would. It is important to take the time and ensure security, rather than skip steps to save time, leaving potential risks in the code.