Through this class, I have learned several techniques and systems that have improved my programming and changed how I approach problem-solving. From teaching me to branch out and try different libraries before settling on one to truly learning how to build and use objects in OOP and even how to judge what ADT to use for a given situation, I have enjoyed finding ways to improve my capabilities and refactoring my programming style to more efficient and elegant styles.

Working with the GUI libraries and systems, these modules taught me that many different libraries could exist, and while they may solve the same problems, the approach they take can impact the program you are writing. Suppose one module can solve your problem, but it takes a little hacking and changing to make it work. In that case, there may be a library or module that does it without having to make those changes or even one that, while it may need refactoring, in the end, makes it easier to integrate the system you are making into your overall program. A JPane might work just as well, but using a JavaFX Pane might have the functionality you need already built into it instead of writing the functionality for the JPane and using your method. I have started to see this in my work in programs for NextJS. Where before, I would find a library or module that could solve my problem and fight with it to shoehorn it into my system; depending on what the problem is and where the system fits into the overall program, there may be a better module out there that already fits without the time-consuming refactoring the original one might. I’ve seen a huge change in my performance on my development team just by trying several systems to see what works best right out the gate instead of spending hours trying to make one work just because I found it on StackOverflow or whatever the case may be.

Working with Classes, Polymorphism, and Objects has also helped immensely. Most of my programming experience is self-taught through hobby projects and the like, I have never truly branched out into OOP and using encapsulation to design a system, and the code I wrote reflected this entirely. When creating a class or element of a system, my code would almost always be extremely long and convoluted, going back and forth to create the same functionality a simple class-based system would make. Transitioning from function-oriented programming to object-oriented has helped with this greatly. Truly learning how to utilize different classes and splitting them up into different files has greatly increased my code readability, according to my development teammates, as well as help with the overall flow of its elements, making it easier to see when redundant systems are called or when data is needlessly duplicated and overutilized, making my systems easier to debug and more efficient overall.

Previous to taking this class, I had a difficult time understanding the types of ADT’s out there, as well as different storing and searching functions. Working through problems on my own projects or professional projects lead me to simply finding a solution to the issue and using whatever structure or algorithm I found on Google or StackOverflow. Learning how these structures are built and how they work within a system has helped me understand why certain choices are made when planning and coding a program. Due to how I scheduled my classes, in Week 4, I started Programming III along with Data Structures and Algorithms. It was perfect timing when we hit week six and worked on sorting, searching, array lists, hash maps, and linked lists. I had a basic understanding of these concepts but never branched out and used them in my professional work. I had learned some of it through tutorials or StackOverflow entries to solve some specific problem, but the actual workings of it never truly clicked. Going over these concepts in this class has filled in broad gaps in my experience with them, making my other two classes much easier to work through. I can now judge what kind of sort or ADT to use based on the problem I am solving without having to hack away at it and try several to see which one works.