Daniel Gutierrez

"Enter a number to select a sorting method:\n"

The homework assignment I was dreading the most turned out to be the one that greatly opened my programming perspective. I went down some interesting rabbit holes during my quest to complete the program. I came across Object-Oriented Programming concepts as I was creating the sorting class. The internet reminded me on the importance of code organization, reusability, maintainability and testability. My skill in construction of classes definitely leveled up, I also learned a good understanding of dependency injection. Additionally, I was able to continue practicing my programming skills by developing a program to sort a user-selected method.

Reversing the algorithms was not as intuitive for me as the sick if-else statement I coded in the Program.java file. Most of the algorithm building was done in class, so it saved me a headache and a lot of time. Integrating the scanner was challenging, luckily, Microsoft Visual Studio (MSV) provides a LOT of helpful information about data types, classes, objects, arrays, Java, or anything coding related in general. I instantiated the scanner but I was having trouble receiving user input. I was able to read the blurb that appears when you hover over key words in MSV. I read that a int selection = scanner.nextInt() to receive the input as an int was my missing key. I learned a little bit about how Java compiles code and that there is a certain order to coding. I ran into this issue when my print statements, scanner, and arrays were not being read, compiled, or output properly. When it came to setting up the array it was pretty intuitive to opt for choosing Student[] for the array. Not until I began working with the algorithms portion did I realized it was a different beast than the int[] arrays we’d been working with. There were a fair number of squiggly lines as I tried to figure out the Student array of objects.

Overall, I learned the more robust and clear your classes and methods are, the smoother the coding experience. I got a good handle on accessing classes and invoking methods through dot notation. Basically, a class may essentially hold unlimited functionality if well designed. I surprised myself when it came time to construct the if-else statement logic for my user input. I chose the if-else statement because it made the most sense to check if the user entry was equal to a number and, if so, then execute. I had a nice tour around the internet during this assignment and picked up a few tricks. I also began thinking about my code structure and what I wanted that to look like.