**Project III -: Project Summary \*Constant Factor Data Table is in Calculation Sheet.**

**Requirements -:** What did this project ask you to do? What was the input? How was the input processed? What was the output?

* Project required us to use different type of sorting to compute complexity times, number of comparisons and moves made in each method to sort an array of numbers. We were required to use different classes for each sorting method and connect those classes with an interface which was initialized in the main method. Main method held the user interface which asked the user to select a sorting method and enter a number, which was raised to the power of 10 and a random number array was created with 10^n integer. As output, number of time microseconds, moves and comparison was printed out for each array as it was sorted.

**Method -:** Describe the algorithm and data structures you used.

* In this project, an interface was used to give each sorting class, a method to implement. This method was called based on specifics given by user in the main method. Main method contained a scanner to read a string and an integer. String was processed and passed onto switch statement which decided the method of sorting based on user input. Integer was passed on the randomize method, which contained a loop going from 0 to n (number). This loop raised 10 to power, and passed it into random generator which created a random array with that number. Loop kept processing numbers until it reaches the number n which was entered by the user. As result random arrays were created with 10, 100, 100, 1000 and so on numbers, providing wide set of data to test.

**Implementation -:** Describe the structure of your code and the packages used.

* My code structure included a main method which processed user input and connected the data with other classes. Other classes included “Selection Sort, Bubble Sort, Insertion Sort, Heap Sort, Merge Sort and Quick Sort. Even though specification of the project mentioned only 4 sorting methods, I tried all of the sorting methods (selection and bubble sort). My program also included Sort Timer class which kept track of comparisons, moves and time for any sorting method. Program also included an interface which connected all the sorting classes, sort timer and main method

**Testing -:** How did you make sure your implementation is correct?

* To start with, I tested each sorting method with print method, to verify that each method is working properly. Later, when I merged all the methods in one program together, I created a verified method, which verified if the return array from the methods is in ascending order. These tests confirmed the efficiency of my sorting method but to test my sort timer, I put whole program together and ran it with each method. This gave me data for each method and assured everything is working well but graphs of each method later confirmed my results.

**Findings -:** What did you learn from this project? Graph and analysis that might be required for this section.

* I learned many things from this project. It was my first program to use interface to connect so many classes together with main method. I learned about the comparisons and moves it takes to sort an array. I also learned about System.nanoTime(), which can give me the running time of each method. I learned about heap sorting method, this gave me introduction on heap. This program taught me a lot and taught me unique ways to use data structures and interfaces.