ESOF 322: Software Engineering I **DUE Date: October 7, 2021**
30 pts

Instructions:

- Do all exercises
- No hand-written answers allowed.
- Absolutely no late assignments.
- Your assignment should be turned in to D2L by all students

Exercise

In this assignment, you are to design, then implement a program in Java (or your OO language of choice) that uses the Strategy pattern to solve the following problem.

You would like to provide a system for your customer that allows them to choose any one of three types of sorting algorithms (bubbleSort, mergeSort, insertionSort, quickSort, etc.). Each of these sort algorithms provides a function (*sort()*) to sort information. You would like to allow your customer to select any one of these sorting algorithms and allow them to change the default algorithm dynamically. The customers of the sorting behaviors are Inventory modules. There are potentially many Inventory modules, and each may need to sort items using a different algorithm.

A) Design a UML **class diagram** to solve this problem. Use the Strategy Pattern.
B) Write code in your favorite OO language to implement your design. The various sort() methods can be implemented as dummy methods. That is, just display a statement that says what kind of sort you are using. The emphasis of this homework is not on sorting techniques, rather on the Strategy pattern. The Inventory modules can be names Inventory1, Inventory2, etc.
C) Create a UML sequence diagram that clearly exemplifies the creation/usage of any Inventory product, its sorting execution, the runtime switching of algorithms, and then a *sort( )* call again.

You will test your code by creating a client program (i.e. a main function) where you simulate the interaction between a client and the selection of an Inventory module. The main function will request your selection of sorting then call *sort()* to execute the default strategy to sort information. Then call *setSortStrategy()* on your Inventory object to dynamically switch strategies to one of the other sorting alternatives and execute it.

Hand in to D2L as a **zip file**:

A UML class diagram of your design (part A) (**11pts**)

A file of your code in text format (part B). For example, hand in a .java or a .cpp file that I could compile and run. Add a comment in your code that states how to compile your code. (**11pts**)

The output of running your program that clearly shows the main function executing the default strategy and the dynamic switching to another strategy. Include print statements on every method to help instrument your code. Your code should be well documented (but not excessively). (**4pts**)

A UML sequence diagram (part C) (**4pts**)