WESTERN UNIVERSITY - CANADA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

SE2203b –Software Design

Assignment 1

Due Date: February 10th , 2023 - 11:55PM

This assignment is to be done individually. Cheating is prohibited in this assignment, that is not to show your code to any other student, do not look at any other student's code, and do not put your code in any public domain. However, getting help is easy and available, just ask the instructor or the TAs. At the same time, you are encouraged to discuss your problems (if any) with other students in general terms, but the actual program design and code must be your own.
Please note that the consequences of plagiarism are much, much worse than getting a low mark for that item of work. The very best case is that you get a zero for the whole thing. Keep in mind that turning in your own work would have at least gotten you a few marks, better than a zero.

1 Goals

- In this assignment you will implement the skills you have gained from lab1 and 2 exercises and develop a JavaFX based application. This application simulates the efficiency of various sorting algorithms. You will also gain new skills in JavaFX GUI applications.

2 Introduction

Graphical demonstrations of various sorting algorithms are instructive, as they provide insight into how an algorithm behaves. In this assignment you are required to develop a sorting engine (Hub) application, where the Strategy design pattern is utilized. JavaFX and Scene Builder will be your programming language and the tools to build your application. Please use IntelliJ Idea IDE to develop your application.

The sorting hub application is responsible for displaying a collection of vertical rectangles (using `javafx.scene.shape.Rectangle` objects) of varying lengths, such as the ones shown in Figure 1, and

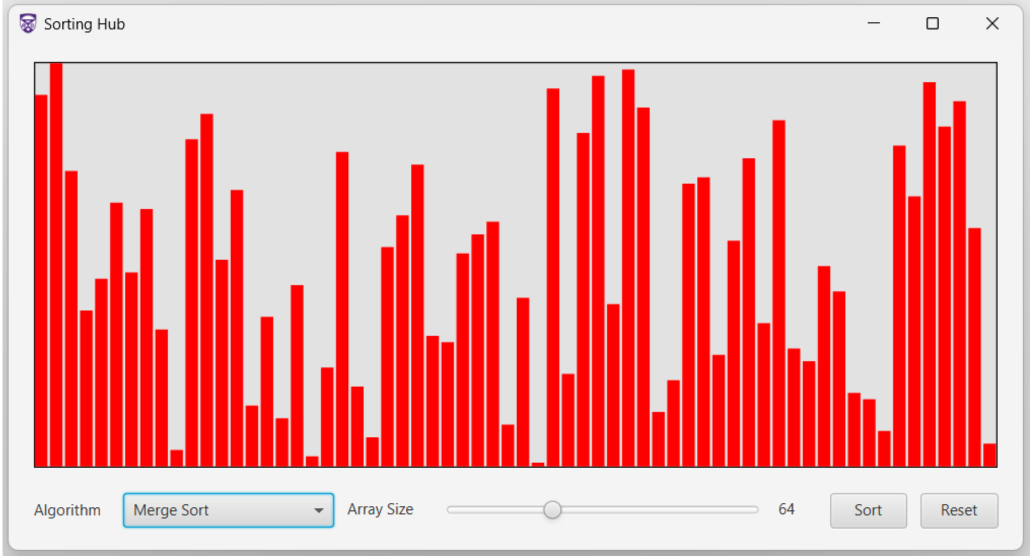

Figure 1

for creating a sorting demonstration that sorts the rectangles by length, as shown in Figure 2. This can be done by drawing a set of rectangles after every swap or move that is performed by the sorting algorithm. If you delay execution very briefly after each redraw, the result will be an animation of the sorting process.

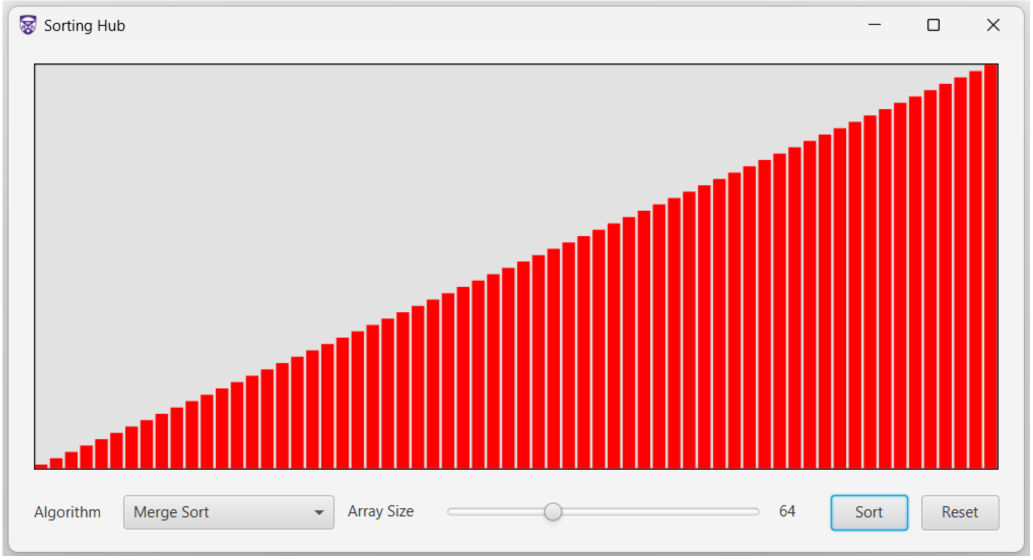

Figure 2

3 Requirements

In this assignment you required to use JavaFX libraries to simulate the efficiency of **two** sorting algorithms. This should include "Merge sort" algorithm and any one of "Selection sort", Insertion sort", or "Quick sort" algorithms.

3.1 Initial configuration

The program will start by generating unsorted array of integers. The generated integers are distinct (no duplications) and range from 1 to the array size. The array size will be determined by the Slider control value that has the minimum value 32 and the maximum value 128. Please consider the following general requirements:

1. JavaFX is the GUI library you should use.
2. The window stage size is 800 x 400 pixels.
3. Use the provided Western logo image to setup the icon of the main window.
4. The program starts by setting the initial value of the array size by 64.
5. Each time the Reset button clicked a new shuffled bars (equal to 64) are displayed, the algorithm type is set to "Merge Sort", and the slider is set to 64.
6. Please watch the provided demo video to learn more about the requirements of this assignment.

3.2 The UI component

This component is responsible for displaying the application data into a form suitable for interaction, typically a user interface element. You are required to build this interface that include all the UI controls demonstrated in the provided demo video, these include:

1. the "Sort" and "Reset" buttons,
2. the Slider control, titled "Array Size", that ranges from 32 to 128 integer values,
3. a Label control to hold the slider value,
4. a ComboBox control, titled "Algorithm", to select the sorting algorithm name,

5. the height of the red bars should represent the array values and scale proportionally to the stage height.

3.3 Controller component

In this assignment you are required to build the controller component that follow the structure design shown in Figure 3. Four classes are needed for this component, SortingStrategy interface, SortingHubController, MegreSort, and one of the SelectionSort, InsertionSort, or QuickSort. The SortingsStrategy interface extends the Runnable Java interface.

The SortingHubController class uses this SortingStrategy interface to define the sortingMethod attribute and to call the sorting algorithm defined by the interface subclasses.

The sorting classes should be developed such that they take unsorted array of integers and return a sorted version of it. They also maintain a reference object of type SortingHubController to be able to update the screen during its sorting process.

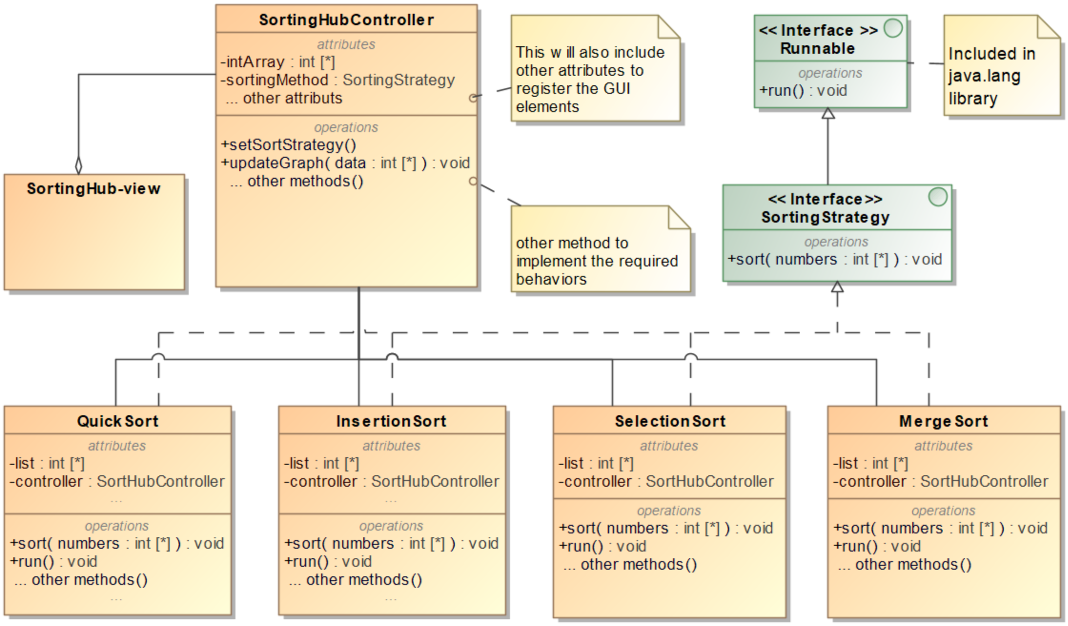
SortingHubController
attributes
-intArray : int [*]
-sortingMethod : SortingStrategy
... other attributs
operations
+setSortStrategy()
+updateGraph(data : int [*]) : void
... other methods()
This w ill also include other attributes to register the GUI elements
other method to implement the required behaviors
<< Interface >> Runnable
operations
+run() : void
Included in java.lang library
<< Interface>> SortingStrategy
operations
+sort(numbers : int [*]) : void
SortingHub-view
QuickSort
attributes
-list : int [*]
-controller : SortHubController
operations
+sort(numbers : int [*]) : void
+run() : void
... other methods()
InsertionSort
attributes
-list : int [*]
-controller : SortHubController
operations
+sort(numbers : int [*]) : void
+run() : void
... other methods()
SelectionSort
attributes
-list : int [*]
-controller : SortHubController
operations
+sort(numbers : int [*]) : void
+run() : void
... other methods()
MergeSort
attributes
-list : int [*]
-controller : SortHubController
operations
+sort(numbers : int [*]) : void
+run() : void
... other methods()

Figure 3

On the top of these functionalities the controller should be able to handle all the events raised by the UI controls due to the user interactions mentioned above.

Notice: If you have any question or if you need to make sure you are doing the right answers, please talk to your TAs during their announced office hours (the lab hours).

4 Hand In

- From the IntelliJ main menu, select File → Export to Zip File…, the Save as pop-up window appears, click OK to save your project as *yourUwoId*_Assignment1.zip file.
- Submit your final zip file through OWL on the due date mentioned above.