Daniel Ilie
215528094
EECS 3311 A
Shape Sorting Lab 01
Kanathipan Kajanan (kana0603@my.yorku.ca) (**eecs3311tafall2021@gmail.com**)

https://github.com/daniilie/EECS3311-Lab01-215528094

PART I : INTRODUCTION

The software project is a UI designed to display various randomly generated shapes (circles,rectangles and squares) in a random order and then allow the user to shuffle through the shapes until satisfied. After that the user can sort the shapes based on their areas by pressing a button. The software's goal is to organize randomly generated shapes based on surface area.

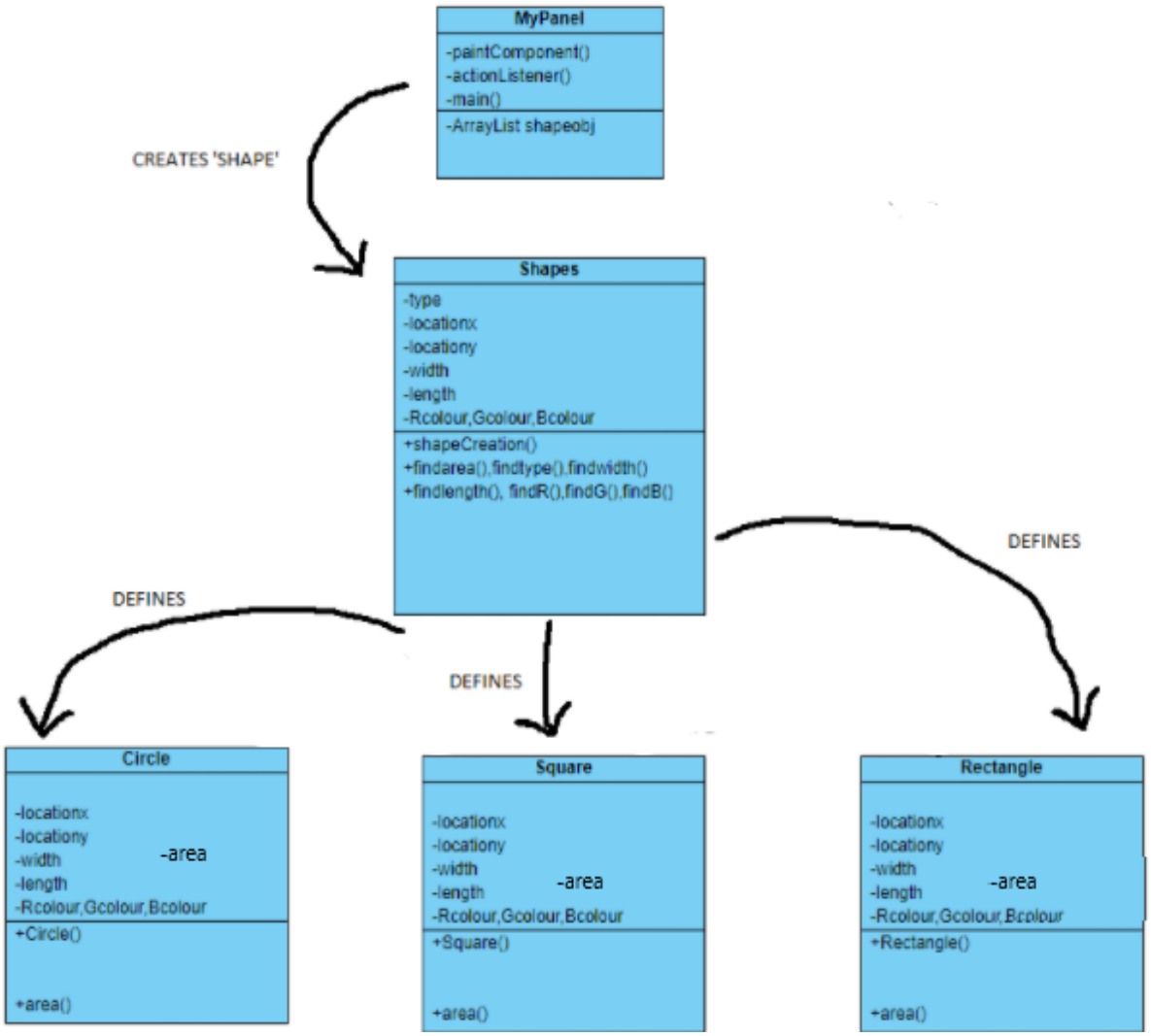
The biggest challenge of the software project was having to quickly pick up new documentation and put it to use (JFrame). At first playing around with JFrame made it seem pretty complicated but once the sections were broken down it was easy to manipulate in order to meet the software design requirements. Specifically the most challenging part of the implementation was having the screen be blank when the UI first loaded up, it took some deep understanding of how JFrame worked. All in all, once past these challenges it was not too difficult to put together the software to meet the requirements.

The software uses a couple of concepts to complete the goal of sorting shapes. The first and most important concept was OOD (Object Oriented Design), this was required to keep track of the shapes and their respective dimensions, colour, where they were located and which type of shape they were. This also made it easier to calculate their areas and sort the shapes afterwards. Another "concept" used was merge sort, this made it easy to take the areas of the shapes and order them correctly while using a low running time just in case the project was ever scaled up to more shapes. Another concept used was the manipulation of the method paintComponent to understand when the shapes were to be printed and when they were to be sorted, this was done by creating a public variable that was used to keep track of which button the user had pressed. And the final concept used was the manipulation of action listeners to carry out specific paths in the code when either of the buttons was pressed and able to reset the shapes if the user wanted to load shapes after sorting a set of shapes already.

The report will be structured by displaying a UML Class Diagram which will break down the concepts used and how they were used. After that will be a display of the classes used in the development of the project in a class diagram. After that will be a breakdown of the algorithms used and their functions along with a

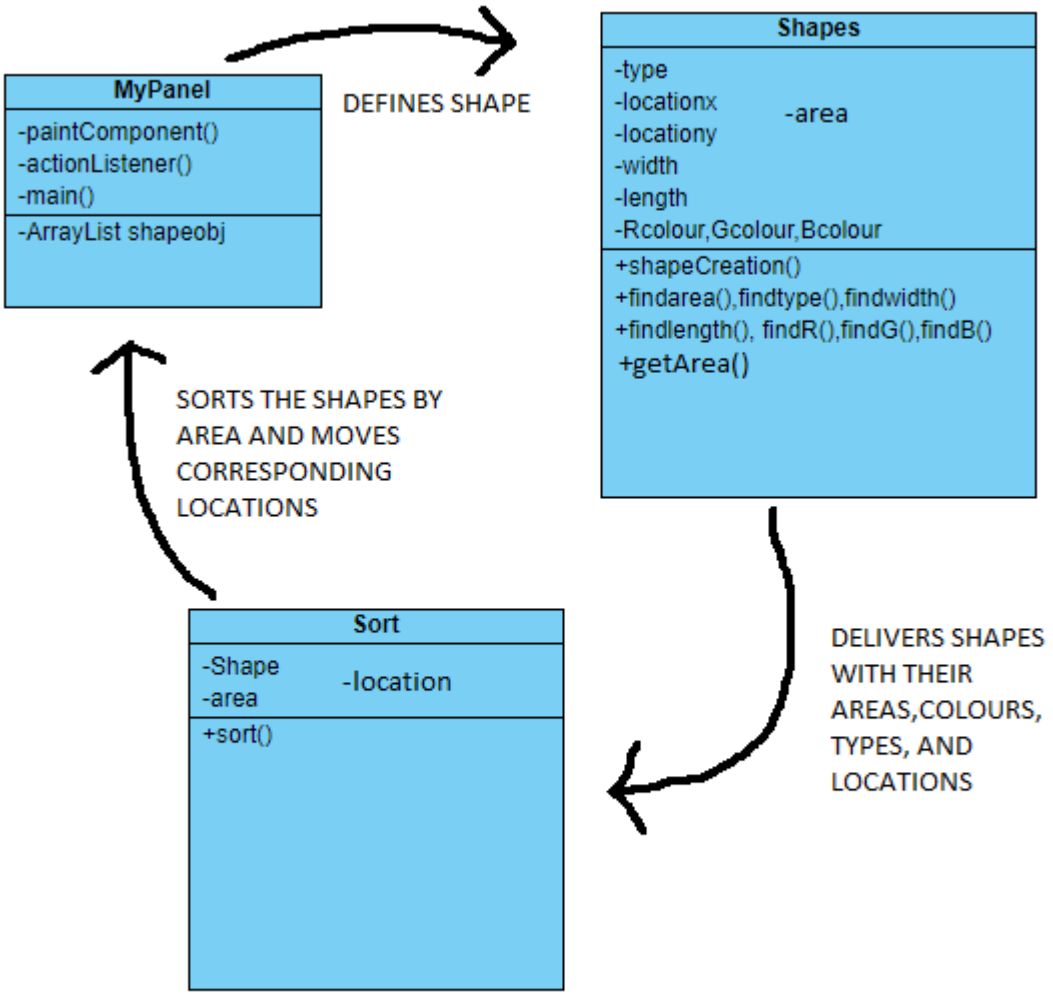
demo video of the software working from the user's perspective. Finally, a conclusion highlighting everything that crossed my mind while working on the software.

PART II : Design of the solution



At first a shape is just a set of variables defining it in class MyPanel, these variables are then passed through to the class 'Shapes' and the object Shape is created with shapeCreation(). From there the object is further defined based on the type of shape it is, being further passed down to the class that defines the type of shape it is (either Circle,Square or Rectangle) each area is now calculated using area() from there the Shape is now defined as a specific type of shape and passed back up to Shapes. Shapes then passes the information back to the main class MyPanel where the class can now sort the shapes based on surface area and reprint them onto the UI using paintComponent() and main().

Alternative UML Class Diagram



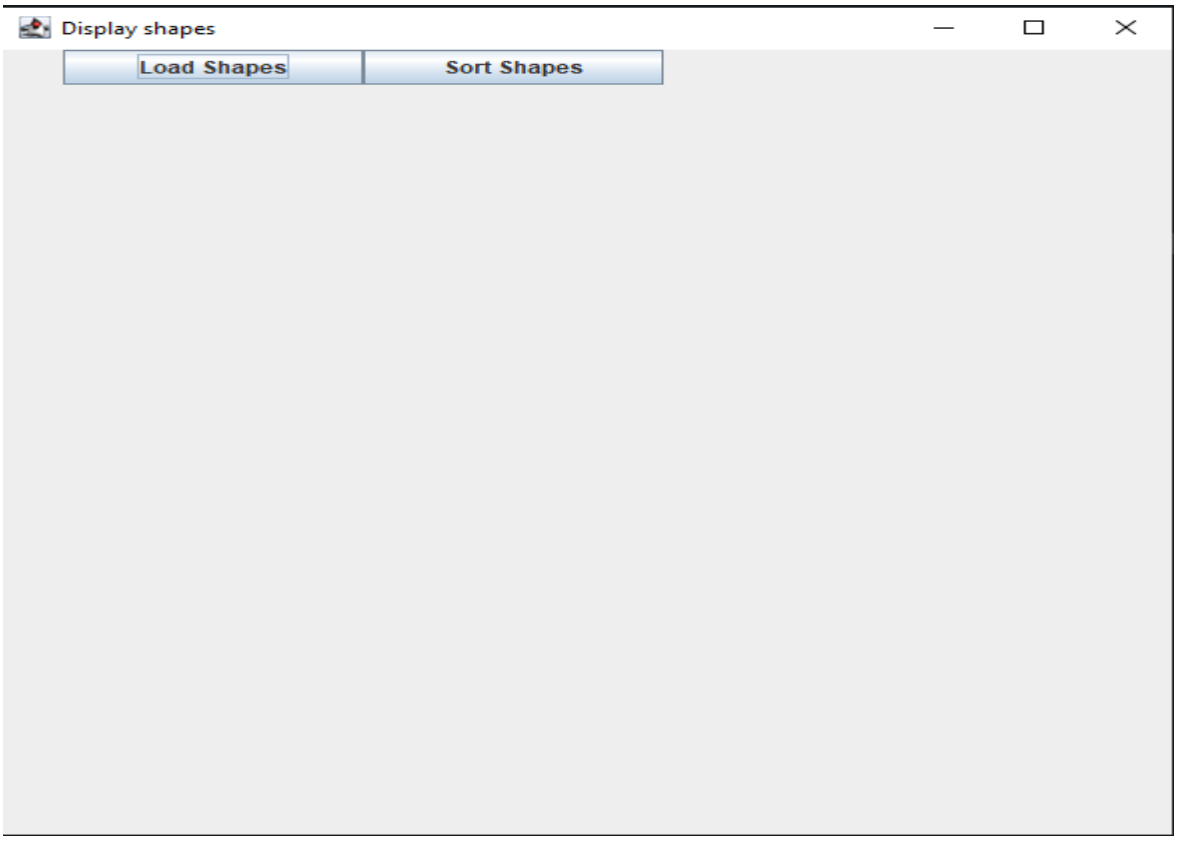
This UML Diagram shows MyPanel creating a Shape object with its corresponding variables (type,locationx /locationy, area, width, length, RGB colours) and sends them to class Shape where the object is defined and its area is calculated based on the type of shape. All shapes are then stored in an arraylist and passed onto sort which takes the list of shapes and sorts them based on area and reorganizes the shapes location from greatest surface area to least. This is then passed back to MyPanel and the shapes are printed using paintComponent().

PART III : Implementation of the solution

The algorithm used to sort the shapes based on areas is mergesort. The algorithm uses two components, sort and merge. The merge component uses recursion to get its job done and the total running time of sort is O(nlogn). How it works is the set of shapes is first divided into two sub sets, then it is divided again and the two elements in each sub set is compared and switched if needed. The sorted subset is passed back up and now the new subset is compared and sorted. Then it is passed back up once again and sorted once again creating a sorted list of areas.

I prefer to have used the second class diagram, for the reason that it removes redundancies. The second class diagram basically has all the attributes that the Circle,Rectangle and Square class would have but more condensed. I prefer to use this way because since the software is currently small scale (6 shapes) there is really no need to define each shape specifically but just have the software understand that it is a shape and each shape type's area is calculated differently. How I implemented the class diagrams into my solution was I had myPanel create random shapes and store them in an arraylist these shapes were then passed onto Shape to be defined as a shape and have their areas calculated, from there it was passed into the sorting method and based on their surface areas were organized from biggest to smallest and then passed back to myPanel for the paintComponent() method to print out the sorted shapes.

To write and read this code I used Eclispe 2020-2019 IDE to work on the project using Java because personally I think it is the best OOD IDE and compiler. I used JDK build 1.8.0_271-b09.



The UI is exactly how it was asked for in the software development requirements for this project. It has a title ‘Display shapes’ and two buttons ‘Load shapes’ and ‘Sort shapes’.

Part IV : Conclusion

What went well in this software project was the logic of the whole design. It was easy to define and program that took a set of variable (Object Shape) and organize it by a variable and then change another variable based on the order of the list.

What went wrong in this software project was mostly getting the hang of JFrame. The paintComponent method runs automatically so being able to clear the UI of shapes on startup was a problem and organizing the shapes based on x and y coordinates was complicated and time consuming (lots of guessing and testing).

From this software project I learned that defining something as an object and creating a class diagram to organize the components makes the project much easier and more organized. I can’t believe I haven’t done this before.

My recommendations to design the software easier is take each component step by step. For example take the design one button at a time and break down the

logic before you begin coding. Another suggestion for easier development is take the time to learn JFrame instead of playing around with the built in methods and burning your own time. The most important recommendation I have is always keep organized and a clear head, this makes putting multiple modules together a much easier task.