Parsa Bahmanikia, 215141104, EECS3311 section B, first software project, Name of the TA: Naeiji Alireza

Intro

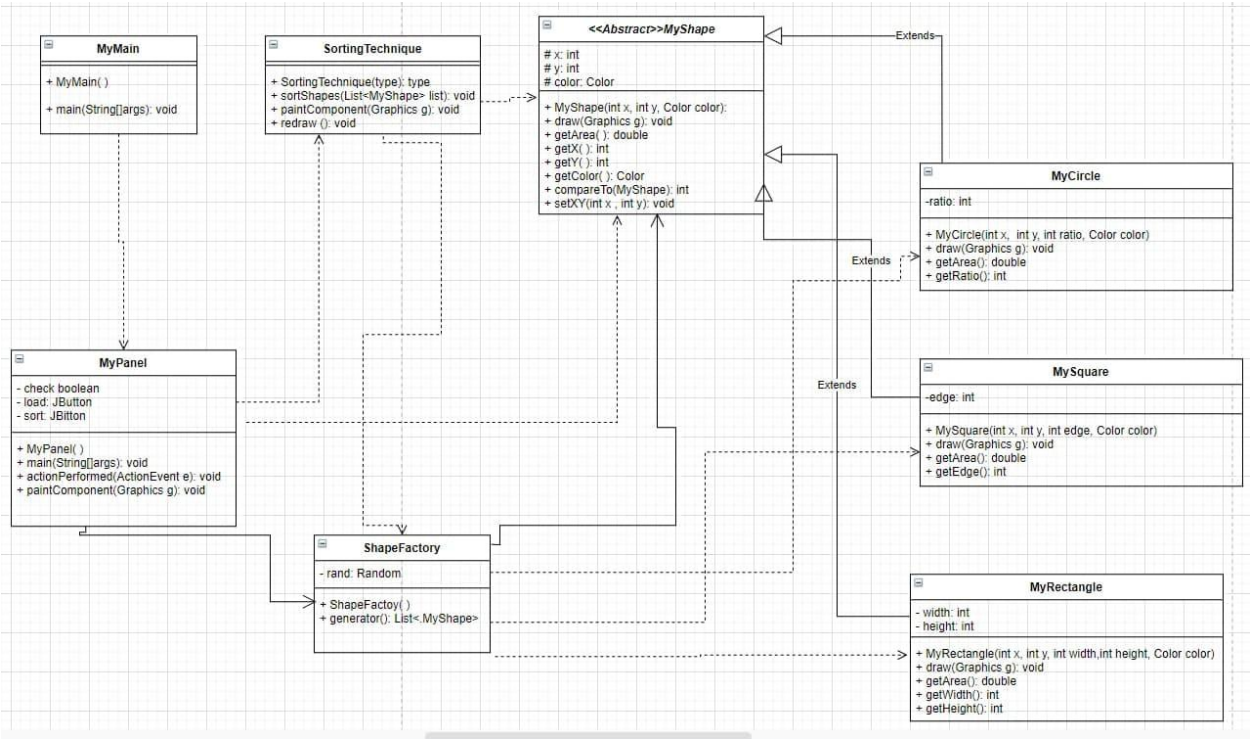
This project is about creating an application which displays an interface with two buttons . The first button in the respected interface is the load shapes button. By clicking on this button every time six random shapes including rectangle, square or circle with different dimensions and colors will be instantiated and created. The second button is the sort shapes button. By clicking on this button the loaded shapes on the interface will be sorted with respect to their areas.

There are several challenges associated with this software project. One of the challenges that could be mentioned is working with classes and methods that were related to Java graphical user Interface. Another challenge that can be mentioned is the logical part of the application for generating random shapes as well as declaring a class to embody a generic concept of shape.

There are several Object Oriented Design (OOD) principles and design patterns which will be used in order to carry out the respected software projects including abstraction, polymorphism , inheritance and encapsulation .

The report will be divided into two main parts that will be focused on the design and implementation process of the project. The design part will be forced on UML class diagrams and it will illustrate the OOD principles and design patterns which are used in the design process. The implementation part of the report will be abou implementation of different classes and algorithms that are used for implementation such as algorithms for sorting the loaded shapes. This part will also include an explanation about the different tools that are being used during implementation such as eclipse.

Design



Commenting elements of the above UML class diagram :

MyMain class which has a main method

MyShape class which has attributes(x and y of type integer and color attribute of type Color) and methods (abstract methods draw and get area and getX, getY, getColor and setXY and compareTo methods)

MyCircle class which has an attribute ratio of type integer and method getRatio as well as implementation of abstract methods of MyShape class.

MySquare class which has an attribute edge of type integer and method getEdge as well as implementation of abstract methods of MyShape class.

MyRectangle class which has attributes width and height of type integer and methods getWidth and getHeight as well as implementation of abstract methods of MyShape class.

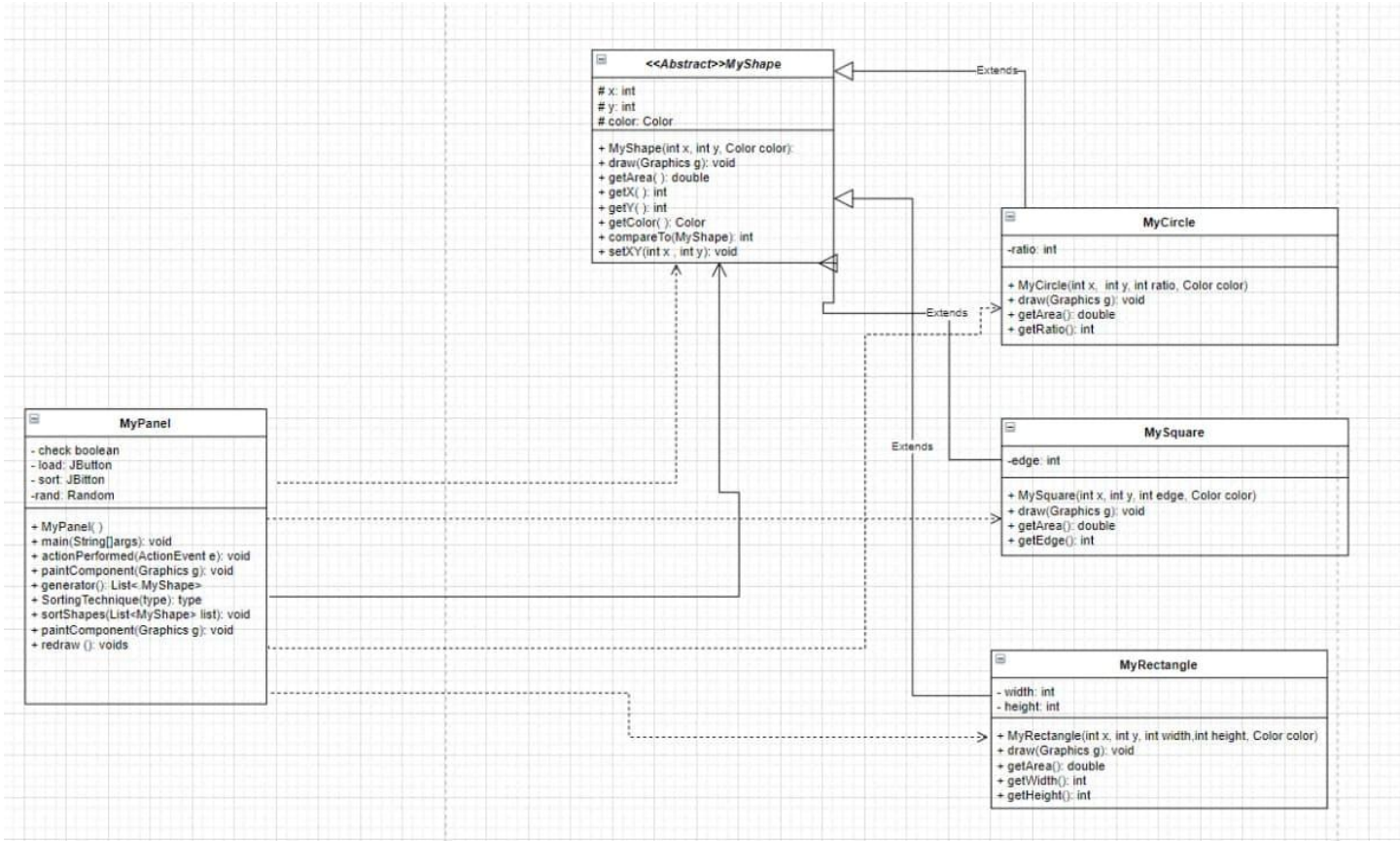
ShapeFactor class with field rand of type random and generator method which generates a list of MyShape objects

SortingTechnique class consists of three methods including sortShapes and paintComponent and redraw methods.

MyPanel class has a field named check of boolean type and two fields of type JButton named load and sort. This class also has two methods named ActionPerformed and PaintComponent.

As it can be observed in the class diagram several OO design principles are used such as inheritance, abstraction and polymorphism. Use of inheritance can be seen in the part that MyCircle class,MyRectangle and MySquare class that all are extended from MyShape class. Use of abstraction in diagrams can be observed in MyShape class since MyShape class is an abstract class with abstract methods. Polymorphism design can be seen in the part of the diagram where each of MyCircle class, MyRectangle and MySquare class override their own version the draw and getArea method so these methods definition will be determined at the run time.

An alternative design:



In my opinion the first design diagram yields better design than alternative design diagram since it makes separate classes for generating random shapes as well as sorting generated shapes and it follows more design patterns compared with the second diagram which only has one class MyPanel to generate the shapes as well as sorting them.

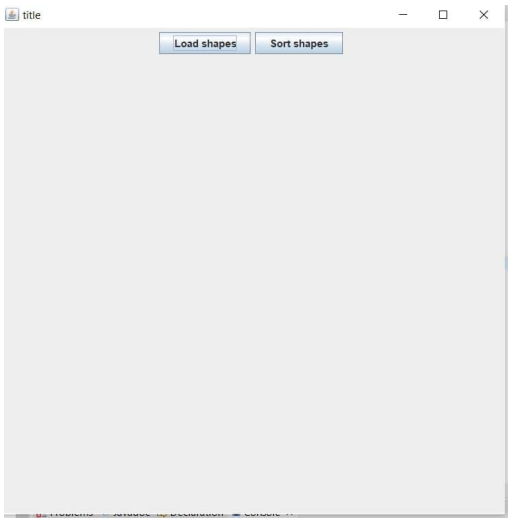
Implementation

The algorithm that has been used in SortingTechnique class in order to sort the shapes is selection sort.This algorithm sorts a list of shapes based on their surface area by repeatedly finding the minimum element from the unsorted part and putting it at the beginning. This algorithm maintains two sublists, one sublist is already sorted and the remaining sublist is unsorted. In every iteration of selection sort algorithm, the minimum amount of unsorted sublist is picked and moved to the sorted sublist. In the SortingTechnique class the sort algorithm takes a list of shapes as parameters and then sorts the list and the PaintComponent method would be used to redraw the generated shapes in a sorted manner.

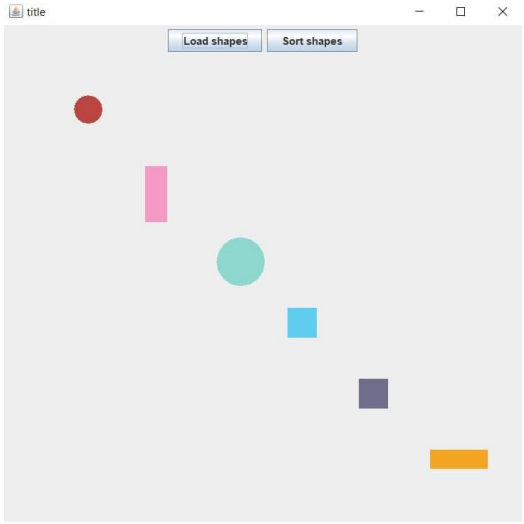
I begin the implementation part of the project based on the first class diagram provided in the design section of the report. Based on this class diagram first I implemented MyShape class which is an abstract class and embodies a generic concept of shape and has three fields that will be inherited by its sub-classes. Then I implemented the sub-classes of MyShape class which are MyCircle class, MySquare class, MyRectangle class. These classes need to implement abstract methods of MyShape class which are the draw method and getArea method. After this part, I begin the implementation of ShapeFactory class and the purpose of this class is to generate a random list of shapes by generator method. The Shapefactory class also has a field in which its type is random and will be used to generate random numbers. After this part, SortingTechnique class which has a sort method and a paintComponent method can be implemented in order to be called for sorting the generated shapes. The method sortShapes has a list of MyShape objects as parameter and use selection sort to sort the list based on the area of the shapes in the list then MyPanel class is implemented which extends JPanel class to provide a panel that implements ActionListener interface to make the buttons in the panel interactive. This panel also has two JButtons fields and these buttons will be instantiated at the constructor of this class. The MyPanel class also has a paintComponent method to draw generated shapes on the interface when pressing the load shapes button. At the last step, MyMain class will be implemented which has the main method. In the main method an instance of MyPanel class will be instantiated and will be added to an object of JFrame class. In order for the software project to work properly by showing the needed interface.

The tools where i have used during the implementation of this software project was eclipse IDE version 2021-09 (4.21.0) and also Java development kit(JDK) version 14.0.2

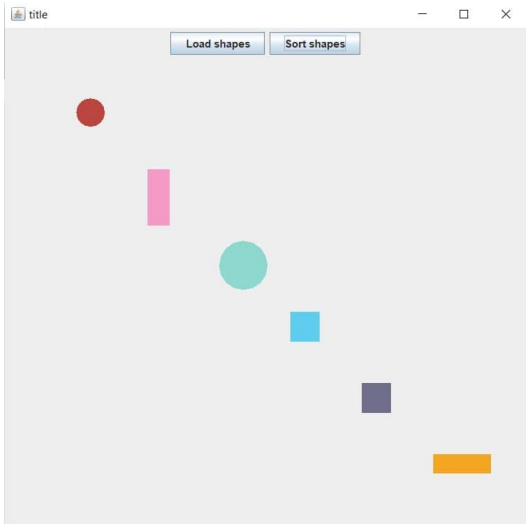
Three snapshots of the program are provided, the first one shows the interface before pressing any button and the second one shows interface after pressing the load shapes button and it can be observed that shapes are generated in a random way and every time with pressing the load shapes button a new random set of shapes will be generated. The third one shows the interface after pressing the sort button. By pressing the sort shapes button, the interface is supposed to sort the generated shapes however it does not meet this requirement completely.



First snapshot of the program before pressing any button.



Second snapshot of the program after pressing load shapes button.



Third snapshot of the program after pressing the sort shapes button.

youtube link for the video that explains how to run the application:
https://youtu.be/lehzYinxEMQ

Conclusion(dones)

For this software project, The part for designing and implementing the MyShape class and its subclasses really went well since it can be seen that the MyShape class embodies the generic concept of shape and each of its subclasses embodies and implements a concept of a particular shape. Moreover, the method generator of the ShapeFactory class which uses a variable name rand in order to generate six random shapes properly was one of the parts of the projects that went exactly as planned.

From my perspective, One of the challenging parts of this project was to define the action listener for the sort button in a way to call the sort method and invoke the repaint method on the same collection of shapes in order to redraw the same set of the shapes in a sorted manner. The challenging part was to invoke the paintComponent method indirectly in such a way that it just redraws the same set of shapes not to draw a different set of random shapes on the interface.

It can be seen that different concepts can be learned by designing and implementing this software project such as classes and methods associated with graphical user interface in

java such as JFrame, Jpanel, Jcomponent classes and paintComponent and repaint methods. Moreover, this project shed light on the importance of different OOD principles design patterns for designing and implementing software projects.

My top three recommendations to ease and facilitate completion of the projects are as follows:
1- Getting familiar with java graphical user interface classes and methods by reading and learning from articles and videos on the internet.
2- Study and review Object Oriented Design principles in order to better understand the relationship between different classes and achieve a better design and implementation for the software project.
3- Study and learn about different design patterns since knowing different design patterns can be helpful and facilitate the design process as well as implementation.