EECS 3311 Project1

Section B

Chang Liu

Student Number: 216882813

TA's Name: Naeiji Alireza

Introduction

This software project is to complete an application that can load and sort three different shapes by using Java Swing. The goals are loading, displaying and sorting the six shapes successfully. The application displays an interface with two buttons "Load Shapes" and "Sort Shapes". Initially, when the user launches the application, the interface is empty, each time the user clicks on the "Load Shapes" button, six shapes that can be circles, rectangles or squares will be displayed, their shapes, sizes and colors are all generated randomly. Each time the user clicks on the "Sort Shapes" button, the six shapes that have been loaded will be sorted and displayed based on their surface area.

There are many challenges associated with this software project. For example, set up the panel and buttons, display shapes on the panel, generate different shapes, sizes and colors randomly, design the compare method to comparing the shapes surfaces and the sorting technique to sort the six shapes based on their surfaces. I'll explain all of them in detail as well as how to overcome them in the following paragraphs.

In this software project, I will be using many concepts from OOD, OOD principles and design patterns to carry out the project. Such as Object-oriented analysis and design workflow, Encapsulation, Abstraction, Inheritance, Polymorphism, Factory pattern and Singleton pattern.

This report including four main parts including introduction, design, implementation and conclusion, I will structure it strictly followed by the project requirement, all the mentioned details from the document will be explained. The report staring with a brief introduction, after that, I will explain my design in detail and followed by two different design solutions with their UML class diagrams. Third, I will be presenting the details of my implementation, a short video presentation will be provided. At last, I will give an overall conclusion to this project, explaining what went well, what went wrong and what I have learned from this project as well as my recommendations to ease the completion of this project.

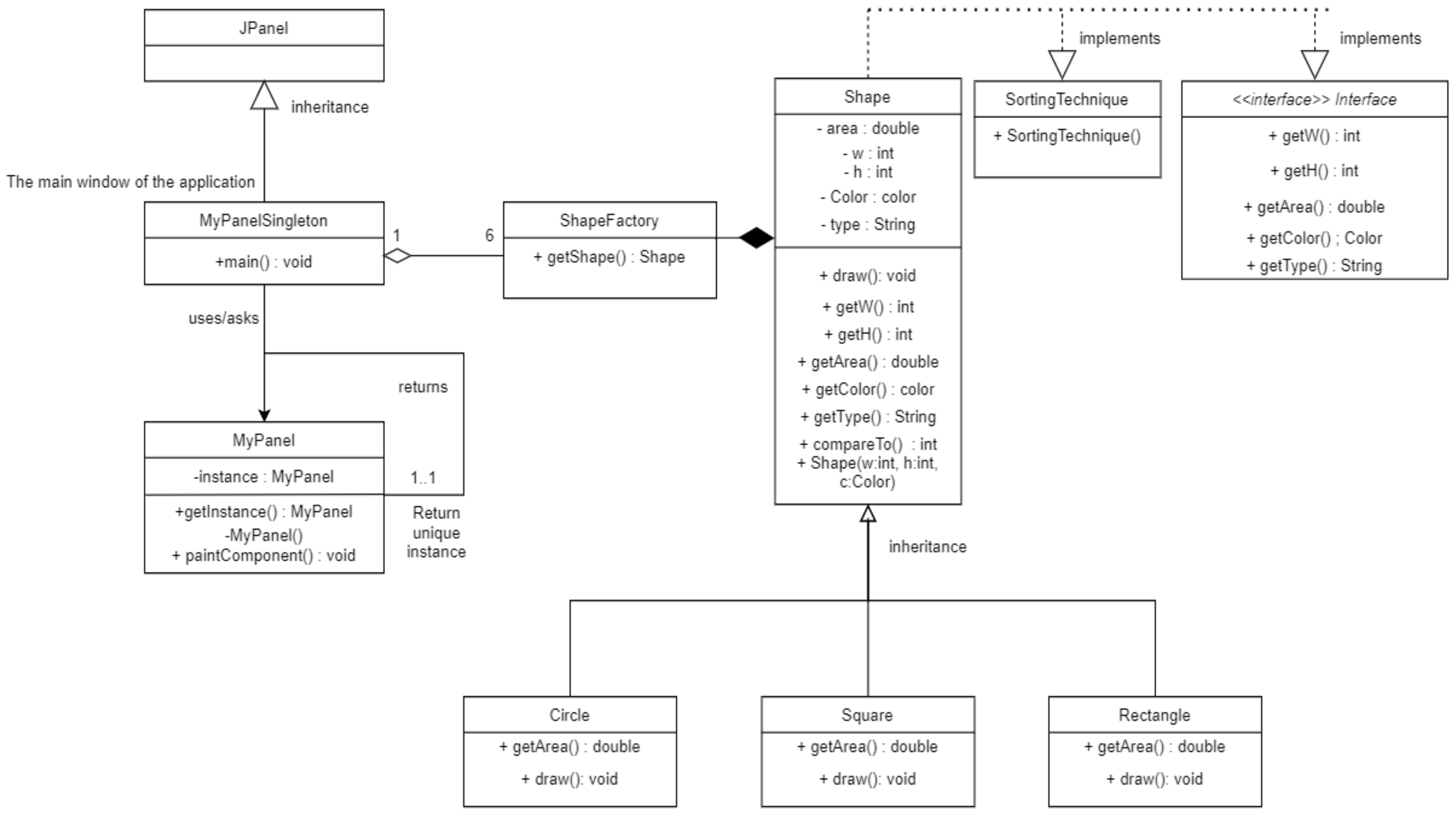
Design

Considering the software process model in this project, the requirements are fixed and clearly specified, the project itself is not too complicated or big and can be completed quickly. As a result, I decided to use the Waterfall model for this project, which is requirement analysis, design, coding, testing, deployment and maintenance.

In the project, my design is strictly followed by the four-step workflow to analyze and design the project. The first step is to define use cases that are creating UML use case diagrams. Step 2 is to define a domain model which means the set of class diagrams where no operations are specified. After identified objects, step 3 is to assign responsibilities to objects and design interaction diagrams which is creating UML sequence diagrams. The last step 4 is to design class diagrams. With the help of preliminary steps 1, 2 and 3, I have created the class diagram by using Diagrams.Net. I will be using three design patterns for this project which are the Singleton design pattern, Builder design pattern and Factory design pattern.

EECS 3311 Section B
Chang Liu
216882813 TA: Naeiji Alireza

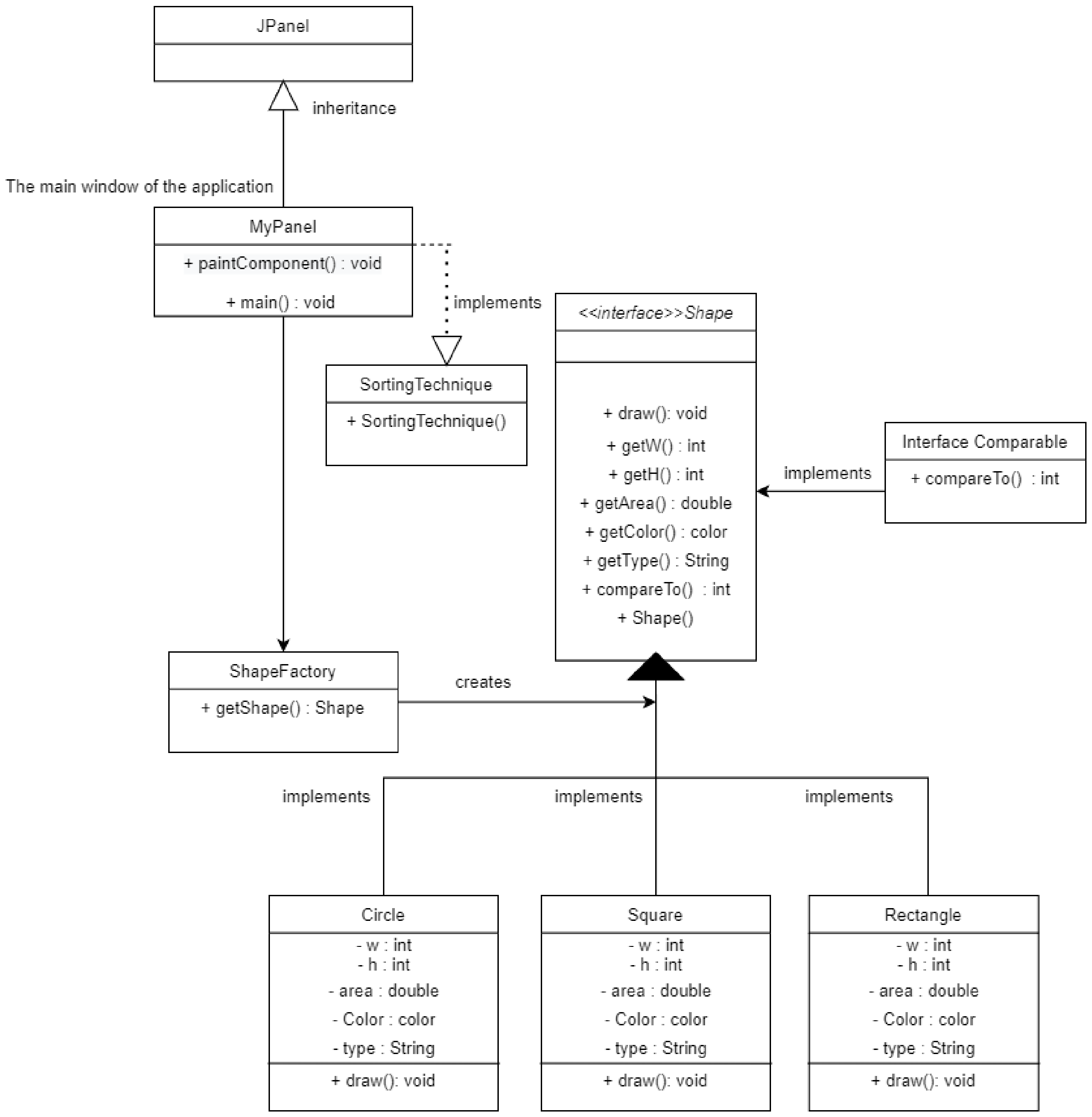
The first design pattern I have used are Singleton pattern and Builder Pattern. Singleton pattern is one of the simplest design patterns in Java, it comes under creational pattern as this pattern provides one of the best ways to create an object. Singleton pattern involves a single class that is responsible to create an object while making sure that only a single object gets created. This class provides a way to access its only object which can be accessed directly without the need to instantiate the object of the class. Builder pattern builds a complex object using simple objects by using a step-by-step approach, this type of design pattern is also coming under creational pattern as this pattern provides one of the best ways to create an object. A Builder class builds the final object step by step. This builder is independent of other objects. I have added the UML diagram figure in the following paragraph with the comment on its elements, and I will explain how do I use these two design patterns in this project.



From this design, by applying design patterns and Object-oriented design principles, we can see that MyPanel class has its constructor as private and have a static instance of itself, this class provides a static method to get its static instance to the outside world. MyPanelSingleton class will use MyPanel class to get a MyPanel object. Shape class implements Interface and SortingTechnique. Shape is the parent class of Circle, Square and Rectangle class. These 3 classes inherit the properties from the Shape class. ShapeFactory class is the builder class for Builder pattern, it helps to build different types of shape objects by combining each different shape. The main class will use ShapeFactory to build a shape. Polymorphism is also used in this project; it is done through inheritance. Since the way to calculate area is different between circle, rectangle and square, in each subclass, the getArea() method has been overridden of its superclass to meet the correct formulas. Encapsulation is used in Shape class. The area, w, h, type and color variables in this class are private, public setter and getter methods that have been created to modify and view these variables values.

I will be using Factory Pattern as an alternative design to create the second UML class diagram. Factory pattern is one of the most used design patterns in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object. In a Factory pattern, the object is created without exposing the creation logic to the client and refers to a newly created object using a common interface. I have added the UML diagram figure on the following page with the comment on its elements.

EECS 3311 Section B
Chang Liu
216882813 TA: Naeiji Alireza



From this design, by applying design patterns and Object-oriented design principles, we can see that the design idea is quite different from the first one. We have created a Shape interface and concrete classes implementing the Shape interface. A factory class ShapeFactory is also defined. The MyPanel class will use ShapeFactory to get a Shape object. It will pass information from Circle class, Square class and Rectangle class to ShapeFactory to get the type of object it needs. Another important concept of the Object-oriented principle that has been used here is Abstraction. Since we have the Shape interface, an interface is a completely "abstract class" that is used to group related methods with empty bodies, which means only the necessary characteristics are exposed to the users.

Comparing the two class diagrams, the second design seems like a better design than the first class diagram. The singleton and builder pattern are very simple and straightforward, it ensures that we can always get back the same instance of whatever type we are retrieving and the builder class helps to build it. However, the factory pattern can give a different instance of each type. Since we want to get six random shapes, sizes and colors each time, so the factory pattern is definitely a better way to help to create and return these new instances each time.

Implementation

The algorithm of the sorting technique I have used to sort the shapes is bubble sort. This simple algorithm performs poorly in real-world use; however, this is a school project, bubble sort is easy to use and understand in this case. It is a simple sorting algorithm that basically just repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order, and it passes through the list until the list is sorted. In our case, we have the shapes into an ArrayList and do the sorting process, since I have created a new Comparable interface method which is compareTo, that allows shapes to compare their surface area, if bigger then return 1, if smaller then return -1, else return 0. So, when comparing the area of the shape in the bubble sort, we compare the area of the current index with the next index by using the compareTo method to see if this is equal to 1, if the condition is met then swap, and pass through the whole list until it is sorted.

I have implemented and compiled all the classes of the first class diagram in Java. By applying design patterns and Object-oriented design principles, I have created the MyPanel class as the main class, it extends from JPanel and contains the JFrame and JButton in it, with the help of the paintComponent method, we can paint out the shapes each time. Moreover, addActionListener lets the button perform load and sort actions each time the user clicks on buttons. From MyPanel class, we can see that it has its constructor as private and have a static instance of itself, this class provides a static method to get its static instance to the outside world. MyPanelSingleton class will use MyPanel class to get a MyPanel object. Shape class implements Interface and SortingTechnique. SortingTechnique Class uses bubble sort to apply the sort shape function of the program as explained above. I have created the Shape class as the parent class of Circle, Square and Rectangle class, since they are the shapes that we want to generate, these 3 classes inherit the properties from the Shape class. ShapeFactory is a class that can help to get and create shapes. Polymorphism is also done through inheritance, the getArea() method has been overridden of its superclass to meet the correct formulas. Encapsulation is used in Shape class. The area, w, h, type and color variables in this class are private, public setter and getter methods that have been created to modify and view these variables values.

The tool I have used during the implementation: Eclipse IDE, Version: 2021-06 (4.20.0), version of JDK: 1.8.

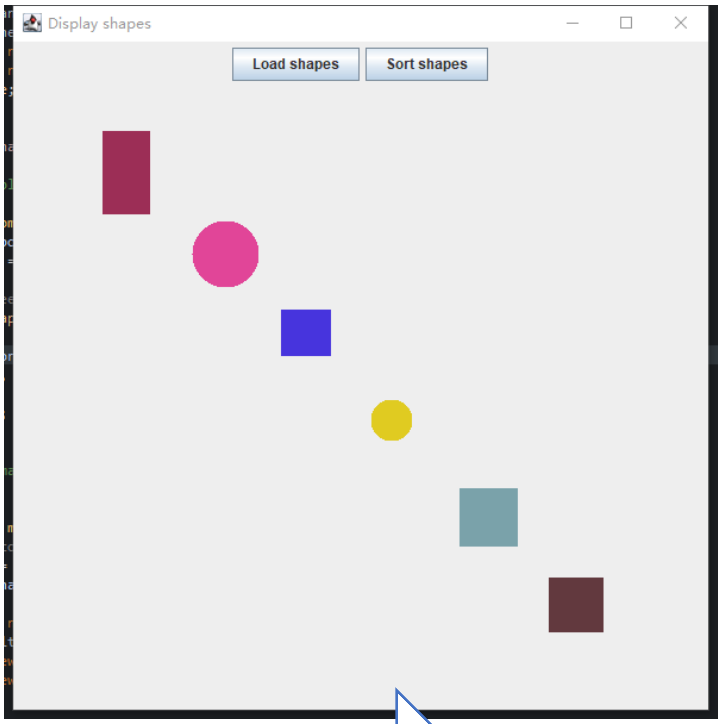
I have added snapshots of the execution of the interface on the following page. We can see that after running the program, the interface is empty with only two buttons on it, each time the user clicks on the Load shapes button, six random shapes will be displayed randomly. Each time the user clicks on the Sort shapes, the six shapes will be sorted and displayed.

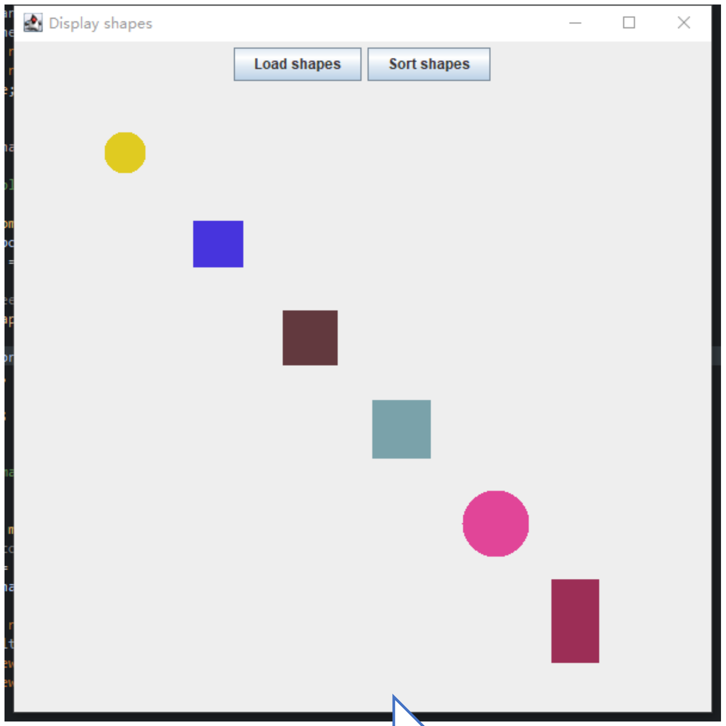
A short video is also created with the link below that showing how to launch the application and how to run it.

https://youtu.be/gGQbt3k3Ecc

EECS 3311 Section B
Chang Liu
216882813

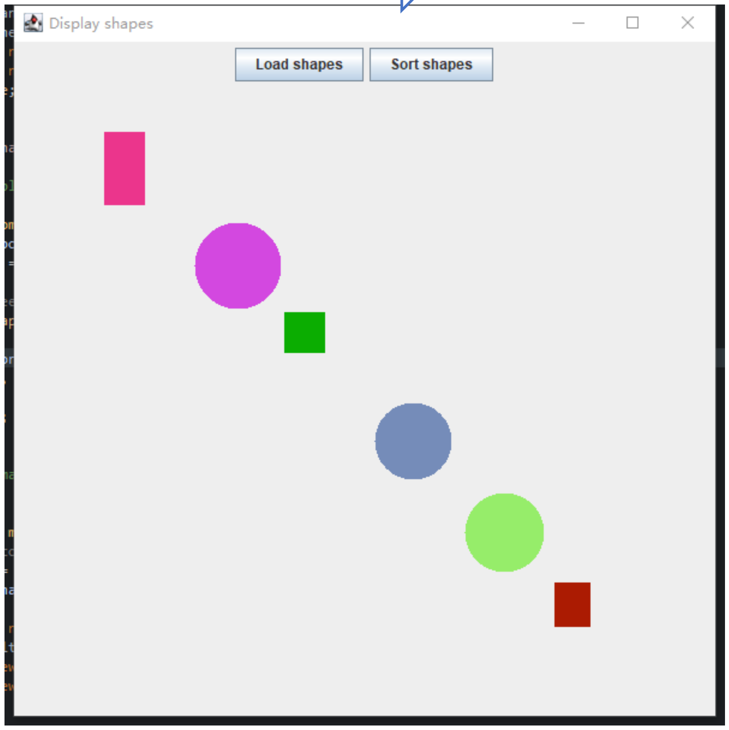
TA: Naeiji Alireza

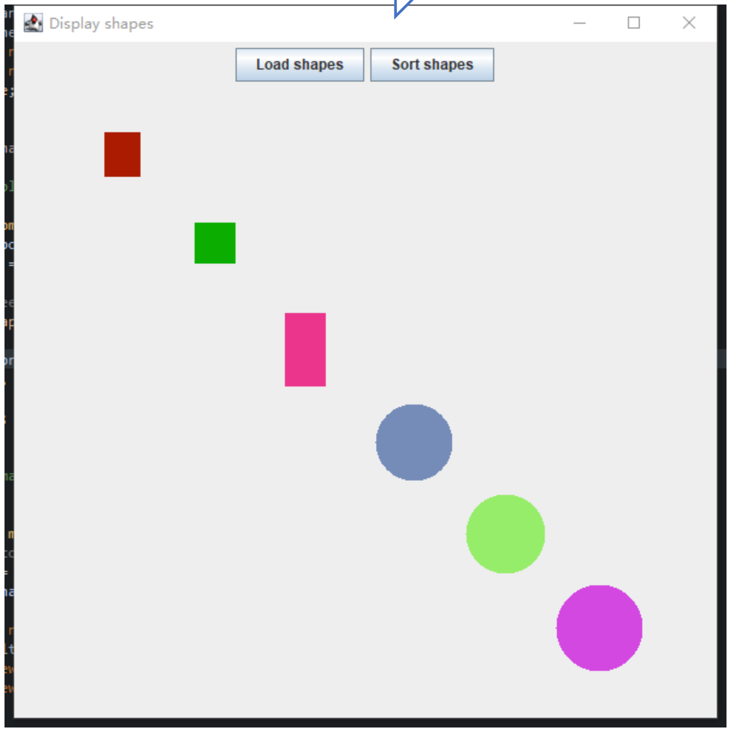




After click on Load shapes

After click on Sort shapes





Conclusion

Overall, the project went pretty well, I worked quickly to make things testable, and finished all requirements on time. After I finished the design part, the implementation part just follows the design of my class diagram, it clearly shows the connection between classes and how the methods will be implemented. The only thing that went wrong was when I first started the UML diagram, I had trouble figuring out how to apply design patterns to this project. However, after the TA gives clarifications during the lab session, everything is clear, and I completed it in a short time. I have learned a lot from this software project, the most important thing is how to apply design patterns in the project and that let me know the importance of doing a proper design process instead of just writing the code, I have never done this before, now I have found out with these design techniques, working on a software project can be very clear and straightforward. In the end, to ease the completion of the software project, the top three recommendations that I will give are: provide a reference version of the class diagram, set goals within realistic timelines and provide more lab sessions to clarify and solve problems.