Name: Wenjun Junlong Jiang Huang

Student Id: 216572281

Course: EECS3311 Section B

Software project name:LAB 3 first software project

Sofware Project 1 Report

**Part 1 Introduction**

In this software project the student, through the abstract design patterns, inheritance, JAVA swing and software design principles learnt during the lectures, will be applying these concepts into practice into this lab, which consists of an application that displays an interface and will sort 6 different shapes. Through the development of the project, the student will create a graphic window with two buttons: Load Shapes and Sort Shapes. While the former will automatically create 6 random shapes painted with different colors, the latter will sort these shapes based on their surface area. All of these aspects consist on the goal of this software project.

The main challenge associated with the software project is implementing an interactive application that is capable of responding to events and outputting graphical responses by using an unfamiliar tool called Java JFrame, while simultaneously applying software design concepts such as design patterns in order to create a sorting algorithm

OOD principles

* Encapsulation: This concept will be used in every created class to keep the attributes and methods within the class. For instance, the MyCircle class has a radius property and a get area method that is uniquely encapsulated within that class.
* Inheritance: In this Project, we have created a parent abstract class denominated as Shape which has 2 subclass/child classes (Rectangle and MyCircle) which inherit the respective methods and attributes from its parent. Note that MySquare Class is the child of rectangle as a square is a rectangle of equal sides.
* Abstraction: Certain variables have been declared as private or protected in order to ease the interactions between them. For instance the MyCircle class has a private attribute (radius) that is hidden to other classes as it is of no interest to them.
* Polymorphism: In this case, the Shape Class has a abstract method getArea() that transforms depending on the subclass that inherits this method.

Design Pattern

In this project a Factory design creational pattern has been carried out through the implementation of a Shape Factory class which allows the application to dynamically create any Shape.

**Part 2: Design of the solution**

Diagram

Description automatically generated

In this UML class diagram a factory pattern has been adopted in order to illustrate the relationship between classes. In this case, we can observe that the OO design principle Inheritance has been used; Both Rectangle and MyCircle classes are subclasses of the parent class Shape, therefore inheriting the parent’s attributes and methods and changing them accordingly (polymorphism).

**Part 3: Implementation of the solution**

In this project, a BubbleSort algorithm has been used to sort the shapes according to their areas. For a given list of Shape objects, two adjacent shapes are compared in terms of their area and swapped accordingly in an ascending order.

**Shape Class**

This parent class, which implements a comparable interface, contains protected attributes and abstract methods that will be inherited by their children classes (Rectangle and MyCircle) which will override them by their own private methods. The constructor method Shape(Color color), initializes a shape with a given color and coordinates (upperX and upperY). The abstracts methods getArea() and paint(Graphics g) will return the surface area of a given shape and paint a specific shape depending on the class respectively. The compareTo(Shape other) method compares the area of two different shapes and returns a difference value which will be later used by the bubble sort algorithm.

**MyCircle Class**

The MyCircle subclass will inherit all the methods and properties from its parent class Shape. It has a constructor MyCircle(int upperX,int upperY,int radius,Color shapeColor) which instantiates a circle object based on the color of the parent’s shape color. The inherited methods getArea() and paint(Graphics g) will be overridden to adapt the MyCircle’s class requirements.

**Rectangle Class**

The Rectangle subclass will inherit all the methods and properties from its parent class Shape. It has a constructor Rectangle (int upperX, int upperY,int width, int height, Color shapeColor) which instantiates a rectangle object based on the color of the parent’s shape color. The inherited methods getArea() and paint(Graphics g) will be overridden to adapt the Rectangle’s class requirements.

**MySquare Class**

The Rectangle subclass will inherit all the methods and properties from its parent class Rectangle. It has a constructor MySquare(int upperX,int upperY,int width,int height,Color color) which instantiates a rectangle object based on the height, width and color of its direct parent’s rectangle color. This is due to the fact that a square is a rectangle of same height and width (sides are equal).

**ShapeFactory Class**

This class has a createRandomShape() which randomly sets a random shape with a random color.

**BubbleSort Class**

In this class a bubble sort algorithm has been implemented through the interaction between the constructor BubbleSort(), which creates a list of six random shapes, and the sorting() method, which sorts them according to their surface area in an ascending order.

**MyPanel Class**

This main class is where all the User Interface and graphical shapes will be implemented. By using Jpanel and JFrame, the class is able to create two interactive buttons ‘Load shapes’ and ‘Sort shapes’ through the MyPanel() constructor method. Then, an action listener is implemented which will allow the buttons to respond to an event through the actionPerformed(ActionEvent e) method. The load button will wait for the user to trigger an event(clicking the button) and create six random shapes with six distinctive random colors. On the other hand, the sort button will sort these shapes using the sorting() method from the BubbleSort class, and portray them in a diagonal line in increasing area.

The tools used in this project were mainly Eclipse and Java.

Graphical user interface, application, Word

Description automatically generatedWhen running the application initally

When Clicking on the ‘Load Shapes’ button.

Chart, scatter chart

Description automatically generated

Chart, scatter chart

Description automatically generatedWhen clicking the ‘Sort shapes’ button

**Part 4: Conclusion**

While the application managed to successfully initialize and portray 6 random shapes with different colors and sort them based on their surface area, I did encounter some minor problems that did delay my steady progress of the project. These were primarily related to JFrame and JPanel as I am completely unfamiliar with the use of these tools, so there were moments were the buttons were not displayed properly, or the shapes would display out of the frame.

Throughout this software project, not only have I learnt these new graphical design tools (implementing responsive buttons), but also apply the theory learnt in class such as OOD principles (Inheritance, abstraction, polymorphism) and design patterns into a project.

My top three recommendations to ease the completion of the software project is to first draw the UML class diagram so that you have a general idea on how the classes will interact with each other before actually trying to implement them. Then, using the design principles (inheritance and polymorphism), start by implementing the parent classes, followed by their respective subclasses/child’s classes. Finally finish implementing the graphical interface at last once the previous steps have been done. All of these will allow a much less disruptive flow of the project.