Java Implementation of Shape Sorter

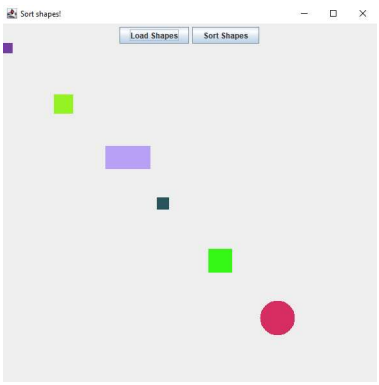
EECS 3311: Project 1

Laura Toro
216650236

Due: Oct 8th, 2021

Structure of Report

1 INTRODUCTION



1.1 GOALS

Shape Sorter is initialized with two buttons at the top center of interface: Load and Sort. Load generates a new set of random shapes displayed on the screen in a diagonal format and Sort sorts them by area in ascending order. Shapes cannot overlap.

The goal of Shape Sorter is to practice and familiarize students with implementing solutions using software design patterns and object-oriented design principles.

1.2 Challenges

i. Threads

Because Shape Sorter implicitly uses threads, there are bugs in implementation that cannot be easily recreated.

ii. Help and Tutorials

As there are many methods for implementation of *Shape Sorter*, there are limited resources available online for assistance when your code experiences bugs.

1.3 Concepts

i. Factory Pattern

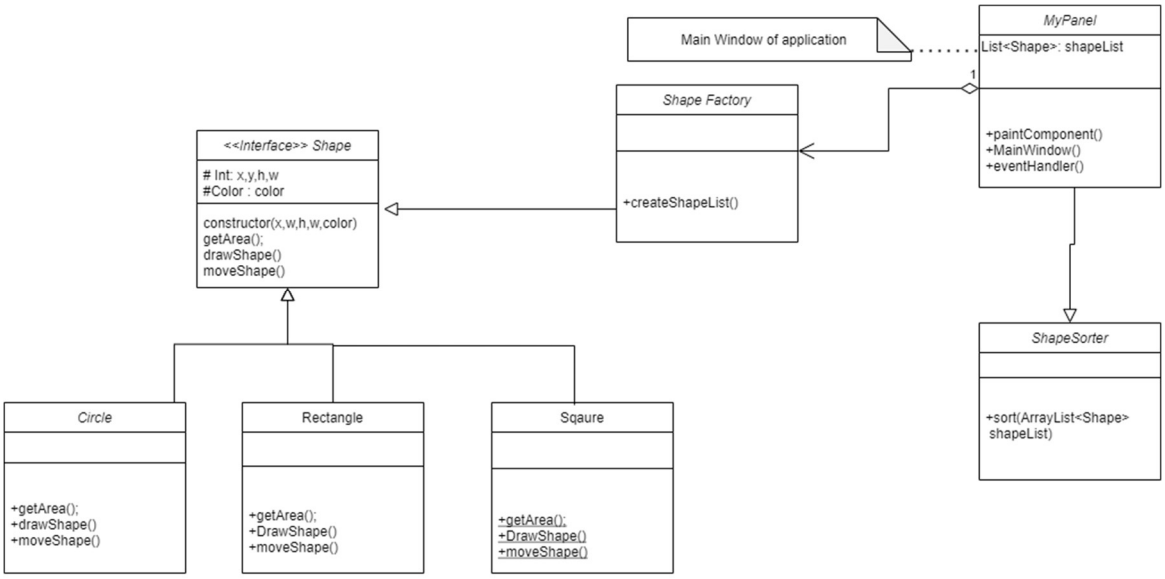
The factory pattern is one of the Creational patterns in software design patterns. With factory pattern, objects are created without exposing logic to the client.

ii. Object Oriented Design Principles

Encapsulation, abstraction, inheritance and polymorphism.

2. DESIGN

2.1 UML Class Diagram



2.2 OO Design Principles

i. Encapsulation

Private variables are maintained within the Shape class.

ii. Inheritance

The shapes (rectangle, square, and circle) all inherit from Shape.

i. Polymorphism

Each child of Shape has separate implementations for getArea() methods.

ii. Abstraction

Only relevant information is shown, such as x and y coordinates in the SortShape method.

3 IMPLENTATION

3.1 Sorting Technique

```
public void bubbleSortArrayList(List<Element> list) {
    Element temp;
    boolean sorted = false;

    while (!sorted) {
        sorted = true;
        for (int i = 0; i < list.size()-1; i++) {
            if (list.get(i).compareTo(list.get(i + 1)) > 0) {
                temp = list.get(i);
                list.set(i, list.get(i + 1));
                list.set(i + 1, temp);
                sorted = false;
            }
        }
    }
}
```

Figure 1 https://stackabuse.com/bubble-sort-in-java/

Bubble Sort was used to sort shapes by their area in the **SortShapes** class. The CompareTo() method was overridden in the **Shape** class to compare areas with the getArea() method. Every time the position of two shapes is switched, their coordinates are also swapped so that they can be displayed on the GUI accordingly.

3.2 Implementation

Shape.java

Packages: java.awt.Color

java.awt.Graphics

Abstract Class: Shape implements Comparable<Shape>

Methods: getArea();

drawShape();

move(int x ,int y);

CompareTo()

Circle.java: -Class Circle implements Shape

-getArea() method overridden, area = (PI * h/2 *h/2)

Rectangle.java: -Class Rectangle implements Shape

-getArea() method overridden, area = (h*w)

Square.java: -Class Square implements Shape

-getArea() method overridden, area = (h*w)

The CompareTo() method in Shape is overridden in order to sort shapes by area.

```
@Override
public int compareTo(Shape s) {
    return Integer.compare(this.getArea(), s.getArea());
}
```

Figure 2 CompareTo() method

myPanel.java ← main window

Packages: import java.awt.event.;*

java.awt.Graphics;

java.awt.Graphics2D;

java.util.List;

javax.swing.JButton,

JFrame, JPanel

Class: Shape extends JPanel
implements ActionListener

Variables: ArrayList of shapes:

shapeList

Methods: actionPerformed();

paintComponent();

createWindow();

ShapeFactory.java:

Packages: java.awt.Color; java.security.SecureRandom;

java.util.ArrayList; java.util.List;

Algorithm: create Shape ArrayList: shapeList

Instantiate SC random number generator

for i…0 do

generate random color c, x,y, h,w

generate number between 1 and 3

if 1 new Rectangle(x,y,w,h,c) add to shapeList

if 2 new Circle(x,y,w,h,c), add to shapeList

if 3 new Square(x,y,w,h,c), add to shapeList

return ShapeList

actionPerformed() listens for events, in this case, when either loadButton or sortButton is clicked.

If loadButton is clicked, call static method createShapes() from ShapeFactory and save to shapeList. This generates a new shape list every time load button is selected.

If sortButton is clicked, call static method Sort(shapeList) from SortShapes class. If shapeList is null (loadButton has not been clicked yet) then it will print a message to the console.

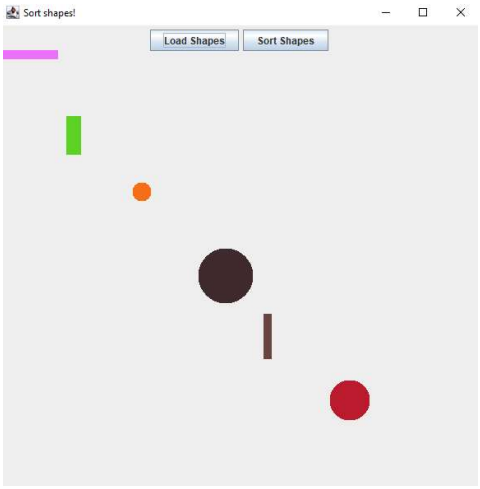


Figure 3 Load Shapes is clicked

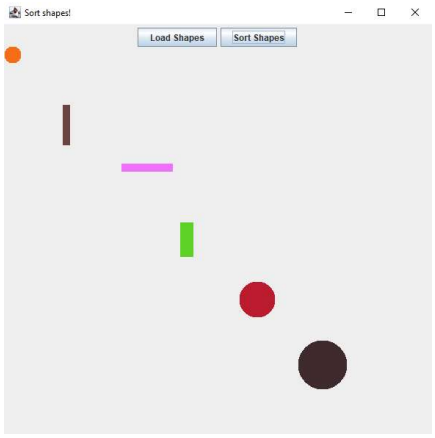


Figure 4 Sort Shapes is clicked

3.3 Tools Used

Tools used:

- Eclipse IDE Version: 2021-09 (4.21.0)
- Java JDK 17

4 CONCLUSION

4.1 What went well?

During this assignment, the creation of shapes was simple. I was able to successfully generate random colors and sizes and display them without overlapping. I was able to successfully load new shapes, and sort them after debugging.

4.2 What went wrong?

I had a few logic errors with my code, specifically with my ArrayList not updating properly. Making the shapeList static fixed the error.

4.3 What have I learned?

This was my first experience with Java Swing GUI, I learned a lot about how to display shapes onto the Frame. I have also learned to implement the abstract design pattern. This was also an opportunity to practice working with sorting and objects.

4.4 Recommendations to complete this project

I would recommend anyone who would like to complete this to watch and read tutorials available online. JPanel can be a bit difficult for first time learners.

Three resources I found useful: Youtube, StackOverflow and TutorialsPoint.