Paint app

Programming 2 lab final

Logo, company name

Description automatically generated

Code outline.

This Java code is used to build user friendly app that allows the user to make digital art by the use of different shapes and colors. This Program is made up of two packages each containing multiple classes:

1. Shape package: contains all classes to do with shapes and colors
2. GUI package: contains all classes to do with the GUI and functions used in the program.

Shape Package

This package contains 6 classes, each shape has a class making a total of five and the sixth class is a class called “Shape” and is in charge of the shapes color.

1. Shape class: This class is in charge of the color of the shapes and since color is a common variable between all shapes inheritance was used and the shape class is used as a parent class.
2. Circle class: This class is used to set the color and set dimensions for a circle shape using its radius, and it extends the shape class meaning it’s a child class.
3. Rectangle class: This class is used to set the color and set dimensions for a rectangle shape using its length and width, and it extends the shape class meaning it’s a child class.
4. Square class: This class is used to set the color and set dimensions for a square shape using its length and width, and it extends the shape class meaning it’s a child class.
5. Triangle class: This class is used to set the color and set dimensions for a triangle shape using its base and length, and it extends the shape class meaning it’s a child class.
6. Line class: This class is used to set the color and set dimensions for a line using its length, and it extends the shape class meaning it’s a child class.

7- ShapeDecorator: A design pattern for adding the fill option to the shapes.

8- ShapeFactory: A design pattern for creating Instances of other objects like Rectangle,Triangle….etc.

GUI package

This package contains 2 classes. First class is “Painting” and the second is “Board”, this package is generally in charge of the user interface and the functions the user can use while designing.

1. Painting class: This is the GUI class that is used to design the window including the drawing area and buttons used to activate certain tools. This class is a child class to javax.swing.JFrame.
2. Board class: This class is responsible of taking the user input from the GUI (using association) and executing. There are four main methods used in this class.
3. Mouse clicked: This method is used record the user’s selection and identify what they want to do.
4. Mouse pressed: This method is used record where the user had pressed and held the mouse showing the starting point to draw or to use another function.
5. Mouse dragged: This method is used to draw the shape to the right size or move it using relocating or adjust the size of an existing shape using resize.
6. Paint component: This method is used to set the colors of the shapes.

Algorithm used

The main idea was to make a board class that extends Jpanel so that the mouse input could be detected by adding mouse listener and mouse motion listener and handling every mouse event possible across the different modes like(copy, resize, create…. etc.)

And getting the required x and y coordinates from the mouse and using them to do different functionalities.

The out is a list that contains shapes and is printed so any shape that is added or modified will be change the list thus changing the output.

Design patterns used

Singleton

Iterator

Factory

Prototype

Template

Decorator

Diagram

Description automatically generated

**SOLID PRINCPILES:**

In order to reach accessibility, ease of refactoring, extensibility, debugging, and readability, SOLID principles are used in our program.

**Liskov Substitution Principle:**

Using inheritance principle, a child must always be able to perform everything in its parent class .The child can replace the base class and must keep the program behavior unchanged. It was implemented in our code by having every shape that extends the abstract class Shape must be able to do every abstract method in this class. None of the child shapes has an incapability of doing a certain action.

**Open Closed Principle:**

A software must be easily accessible with new features without having to modify the whole code.

**Single Responsibility Principle:**

By having many classes the risk of having errors in our code might increase so we need to separate behaviors so that we reduce bugs in the code. Each class should have a specific function in the program to reduce this risk.

**Interface Segregation Principle:**

Classes should not be required to do tasks that are not needed by the program so in order to do this we did not implement any interfaces that won’t be needed in our code.

**Dependency Inversion Principle:**

The idea behind this principle is to reduce the dependency between classes but to link them using interfaces and abstract classes.

Anas Emad group 1 section 1 7048

Hazem Abdelsalam group 3 section 1 6921

Abdelrahman sewelam group 3 section 1 6962

Seif Sherif Group 3 Sec 2 7006