DETAILED DESCRIPTION OF FEATURES IMPLEMENTED IN THE OBJECT-ORIENTED PAINTING APPLICATION

by

ADHAM MOHAMED ALY 6744

MOHAMED FARID ABDELAZIZ 6905

YOUSSEF AHMED ABOUEID 6883

YOUSSEF ASHRAF HASSAN KOTP 7140

Submitted in Partial Fulfillment of the
Requirements of the Final Project of

REPORT

Programming II

CC272 Course

Prof. Dr. Laila Abou Hadid

Dr. Mohamed Kholeif

Table of Contents

I. Introduction

In a program in which we focused heavily on providing a highly user-friendly interface, we also maintained a high level of readability to the source code by implementing the S.O.L.I.D principles as much as possible. We were also successful at implementing 6 different design patterns within our program and putting them to use.

These are the design patterns used:

1. Singleton design pattern
2. Observer design pattern
3. Façade design pattern
4. Factory design pattern
5. Prototype design pattern
6. Iterator design pattern

Details of how the design patterns were implemented will be discussed in their respective section. The project was also written across twenty different classes. The program's GUI consists of two windows (2 JFrames) designed using javax.swing as required.

Efforts in coding and debugging were evenly distributed among the four participating students which facilitated in adding additional features which are not explicitly required in the assignment description.

The following table represents a list of **all** features implemented within the app with a brief description for each one.

Feature	Description
1. Draw line	Draws a line segment from one point to another set by dragging the mouse
2. Draw rectangle	Draws a rectangle resized by pressing and dragging the mouse cursor
3. Draw square	Draws a square resized by pressing and dragging the mouse cursor
4. Draw circle	Draws a circle resized by pressing and dragging the mouse cursor
5. Draw triangle	Draws a triangle resized by pressing and dragging the mouse cursor
6. Free Paint	The user can draw freely by pressing and dragging the mouse in any direction
7. Erase	Similar to a real eraser, the user can erase certain parts freely by pressing and dragging
8. Clear screen	Clears the paint board completely
9. Undo	Undo actions latest to oldest

10. Redo	Redo undone actions latest to oldest
11. Choose paint color	The user can choose the paint color from quick buttons or from a wide-range palette and can even colors by RGB
12. Fill shape	The user can fill shape any drawn shape with the current paint color by clicking on it.
13. Move shape	The user can press and hold on a certain shape to choose and move it around the paint board.
14. Resize shape	The user can press and hold on a certain shape to choose and resize it to the size of their will.
15. Copy shape	The user can copy a shape by pressing the 'copy' button then clicking on the shape.
16. Paste shape	The user can paste a copied shape any where and as many times as they want by pressing the 'paste' button and clicking any where on the paint board.
17. Set brush thickness	The user can set the stroke of the brush. This affects shape borders, free paint and the eraser.
18. Return to start page	User can return to the start page by pressing on the app logo in the top center of the window.

Figure 1.1 Table of features

II. Layout

As mentioned above the program has a total of 20 classes that are specifically designed for this program and all classes are named accurately according to their role in the program, the program perfectly applies the S.O.L.I.D principle.

First of all the main classes that were implemented at first were GeometricShape, Square, Rectangle, Circle, LineSegment, Triangle, PaintBoard and Point classes. Afterwards more classes were created to add more features and apply important design patterns principles as ScreenShotter, FreePaint, Eraser, Observer, ShapeFactory, ShapeMakerFacade, ClearedScreen and Finally two JFrames for GUI. Square, Rectangle, Circle, LineSegment, Triangle classes all applies Inheritance as they all (extends) from the super class GeometricShape. Each class contains private variables, constructors and several methods and may contain setters and getters.

Let's have a detailed look at the 2 JFrame of the GUI the StartPage and Paint, the StartPage is the first interface you see when you start the program, this window contains two buttons as in fig-2.1 which are project details that contains names and ID's of the students that participated in thinking and coding this program.

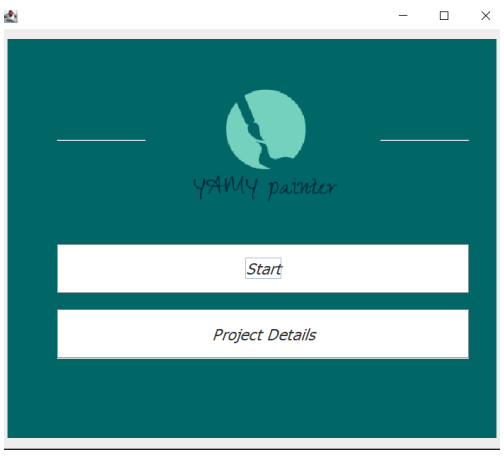


Figure 2.1 Start page

The second button is start that leads the user to the second JFrame which is the Paint here all the magic happens. When the window opens you see in the center of the tool bar our specially designed logo for this project which also works as a button to go back to the start page, then on the left hand side there are the color selector panel, free paint button that has a brush icon on it, eraser button that is used to erase unwanted drawn shapes or even with free paint and finally a slider to select the stroke thickness of shapes, free paint and the eraser thickness too. On the right hand side there is the copy and paste icons then the shapes with a clear icon on each button and the fill button as well. Finally, there are the move, resize, undo, redo, clear buttons and an extra feature that can facilitate saving of your drawing which is screenshot button as shown in fig-2.2

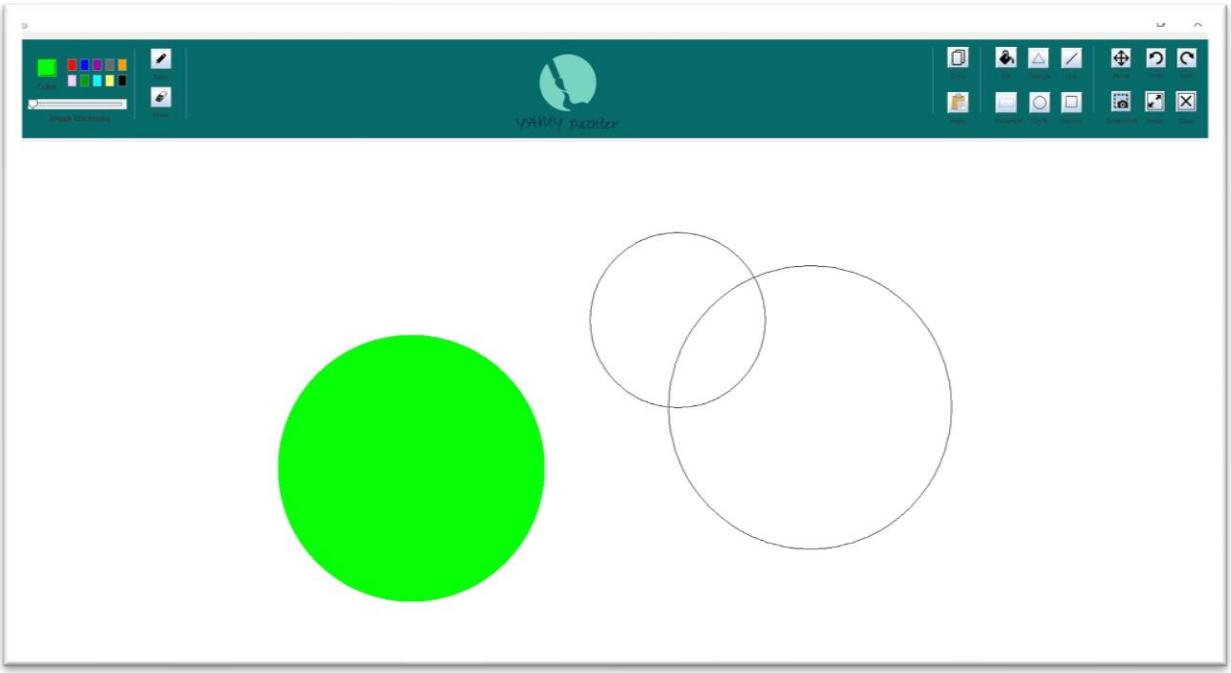


Figure 2.2 Main app interface (Taken using our in-app screenshot feature)

III. Class Details

1) GeometricShape

This class was created as super class to be extended later by the other shapes classes and it contains main attributes, four constructors, setters and getters and two abstract methods that should be implemented in subclasses these methods are draw() and select(). This class applies cloneable design pattern.

2) LineSegment

This is one of the subclasses that extends from the super class in this class therefore it overrides the abstract methods draw, select and clone method and also four constructors.

3) Rectangle

This is the second subclass that extends from GeometricShape super class so it overrides the abstract methods implemented in the super class draw, select, clone methods and as other classes it has four constructors. In the draw method we handled an important part which is drawing in different quadrants this made the program more reliable and easy to use as you can drag the point in any direction you want. The select() method uses an integrated method in Java which is Rectangle that takes two integer parameters that indicate the location of the mouse and it checks if the sent parameters are inside the rectangle if yes it selects it if they does not match it does not work.

4) Square

This class is a subclass that extends from super class mentioned above and it is similar to the Rectangle class and has four constructors. We used the same method used to draw the rectangle but with length equals width as the square is a special case of rectangle. Moreover, it implements the abstract methods from super class draw, select and clone methods, the draw method also handles the drawing in four quadrants.

The select method uses an integrated method in Java which is Rectangle that takes two integer parameters that indicate the location of the mouse and it checks if the sent parameters are inside the rectangle if yes it selects it if they does not match it does not work.

5) Circle

This class is also a subclass that extends super class's abstract methods that are draw, select and clone and has four contractors inside it. The circle is a special case of the oval with width equals height so we used the method of drawing the oval while applying this special case and as the other classes we can draw the circle in any desired quadrant.

The select method uses an integrated method in Java which is Rectangle that takes two integer parameters that indicate the location of the mouse and it checks if the sent parameters are inside the rectangle if yes it selects it if they does not match it does not work.

6) Triangle

This is the last subclass that extends super class all abstract methods draw, select and clone and has two constructors as well. This class is a

different from others as it has another attribute and a getter and setter for this attribute.
The draw method calls the polygon drawing method and the added attribute. The select method uses an integrated method in Java which is Rectangle that takes two integer parameters that indicate the location of the mouse and it checks if the sent parameters are inside the rectangle if yes it selects it if they does not match it does not work.

7) Point

This class contains two private attributes x and y, 2 getters and setters for x and y, a constructor and it implements cloneable interface and it overrides a method from cloneable.

8) Free Paint

This class adds the feature of free drawing with no need to select certain shape, it has 2 attributes: X-coordinates & Y-coordinates, those attributes will be used later in the function drawPolyline() which takes 3 parameters: 2 arrays of integers and array size.
This class also extends GeometricShape class so both functions draw() & select were overridden to perform the required task.

9) Eraser

This Class enables us to delete specific location inside the whole board, this class has the same concept as FreePaint class but it has white color instead, it also extends the GeometricShape class.

10) Cleared Screen

With the help of this class, the program can clear the whole screen while maintaining the undo & redo functionality, it draws a rectangle filled with white colors which makes the screen look like cleared one but it is not.
This class also extends the GeometricShape class.

11) ScreenShotter

This class is one of the added features to make the program more reliable and easy to be used, creates new object of type robot and then a rectangle is created which takes dimensions and the robot method takes these dimensions and calls the screenshot. This class applies two important which are Singleton and observer, this application will be illustrated later in the report.

12) PaintBoard

This class extends JPanel and implements MouseListener and ActionMouseListner, we declared global static variables to be used later. One of the most important variables is shapes array lists of geometric shapes.
The method paintComponent was overridden to print what is inside this array list, a variable named currentShape was used which indicates the working mode (resize, draw, etc.).
The currentShape is responsible for handling the process of mouse listening for example if the currentShape equals 1 this indicates that the program should draw a rectangle.

13) ClearedScreen

This class extends from the super class GeometricShape as a result it implements the abstract methods which are draw and select it also has a constructor.

14) ShapeFactory

This class was created to apply design pattern principle, this class contains a method that takes integer parameter named mode according this function returns shape according to the entered parameter (mode).

15) Observer

This class was created to apply design pattern principle, and a void method that updates the screenshots taken by user.

16)ShapeMakerFacade

This class was created to apply design pattern principle, this class has private attributes, constructor for each attribute and five methods each one for drawing one of the shapes.

17) MainMethod

This class contains two important methods which are draw and undo

18) FinalProjectPaint_V1

This class contains the main method to run the program.

IV. Design Patterns

Six different design patterns were used in this program which are as follows:

1-Factory design Pattern:

(Creational design Pattern)

We have used Factory design patter in order to create new Objects without need of construction and to avoid coupling and dependency between Classes. So we constructed new instances of classes through this factory and returning it in needed methods or classes.

2-Prototype design pattern:

(Creational design Pattern)

We have used Prototype design pattern by implementing “Cloneable” interface and implementing “clone()”method in each and every Shape class and also Cloning instances of “Point” class.It helped us making copies of our Objects using values not references that helped us in copy method.

3-Singleton design pattern:

(Creational design Pattern)

We have used Singleton design pattern to assure the creation of only one object of “ScreenShotter” class which has main aim to take screenshot of the painting board and it is the only instance that is allowed to do such task.

4-Façade design pattern:

(Structural design Pattern)

We have used façade design pattern as it offers a simple interface to more complex underlying objects. So we could use draw methods using objects of “ShapeMakerFacade” class and not by accessing the shapes classes itself.

5-Iterator design pattern:

(Behavioral design Pattern)

Iterator design Pattern facilitates looping across the array list of Geometric shapes needed to be drawn by only using 2 basic methods which are : “hasNext()” and “next()” avoiding looping with varying conditions.

6-Observer design Pattern:

(Behavioral design Pattern)

Observer design pattern helped us with automatic screenshotting(update method) while drawing (auto-documentation) as it takes screenshot each and every time we release the mouse by only changing the state in the PaintBoard class.

V. Custom Algorithm

Upon implementing the undo and redo features, we found it necessary to come up with a new algorithm to handle the undo operation universally across all actions. Thus, we invented an algorithm and called it “The Domzing Algorithm”.

The algorithm depended on 2 stacks, ‘done’ and ‘redo’ in addition to the main painting array list ‘shapes’ . As well as an attribute in the parent class ‘GeometricShape’ called nextShape which is a private GeometricShape object.

Whenever a shape is painted, its nextShape is set to null. Then, it is added to the shapes array list and pushed into the ‘done’ stack.

If the shape is moved, a new cloned instance of the shape is created and added to the ‘shapes’ array list and the old shape is removed from it. The nextShape of the old shape is set to be the new instance. Then, the old shape is pushed into the ‘done’ stack.

Assume we draw a shape called 1. Then its nextShape is set to null. Then it is moved to become shape 2. Therefore, 1 is pushed again with nextShape = shape 2. Then shape 2 is resized to shape 3 and the same procedures are done. As shown in figure 5.1.

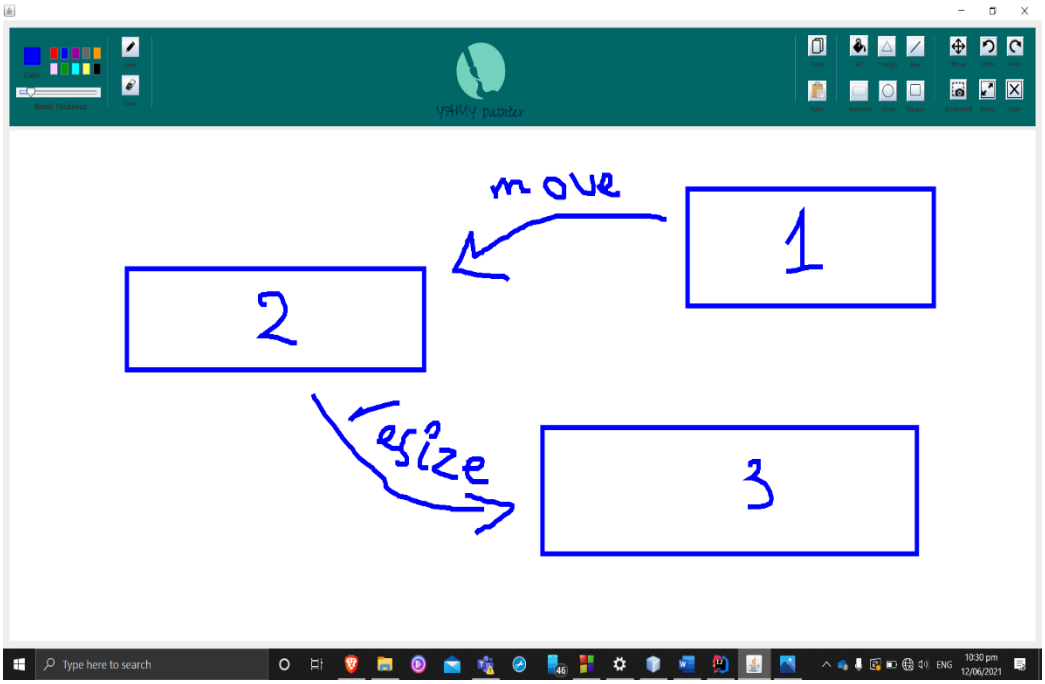


Figure 5.1 Example procedure

When we press undo, the topShape in the done stack in popped. The 'next' shape is searched for in the shapes array list and removed. The topShape is added to take its place in the array list. Then, the topShape is pushed into the redo stack. As shown in figure 5.2.

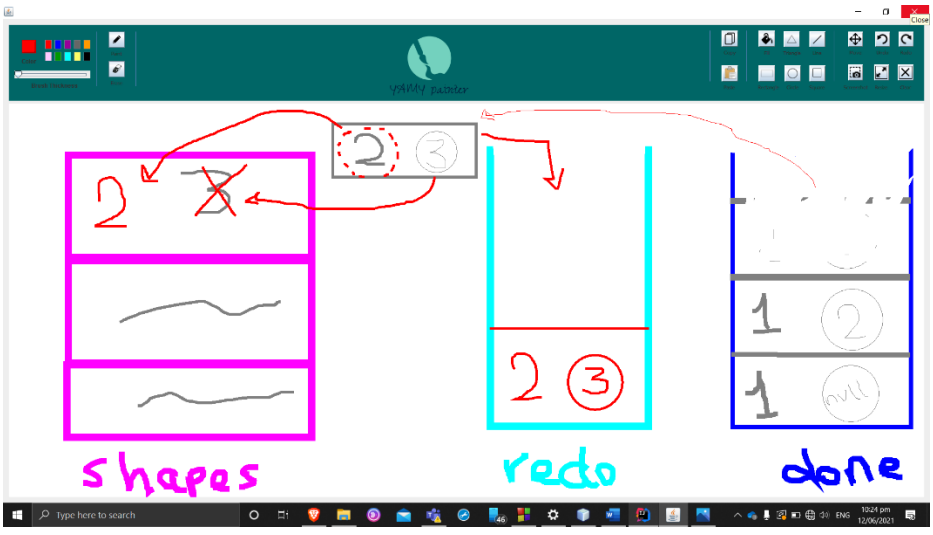


Figure 5.2 Undo procedure

When we press redo, the topShape in the redo stack in popped. The ‘topShape’ is searched for in the shapes array list and removed. The next is then added to take its place in the array list. Then, the topShape is pushed into the done stack. As shown in figure 5.3.

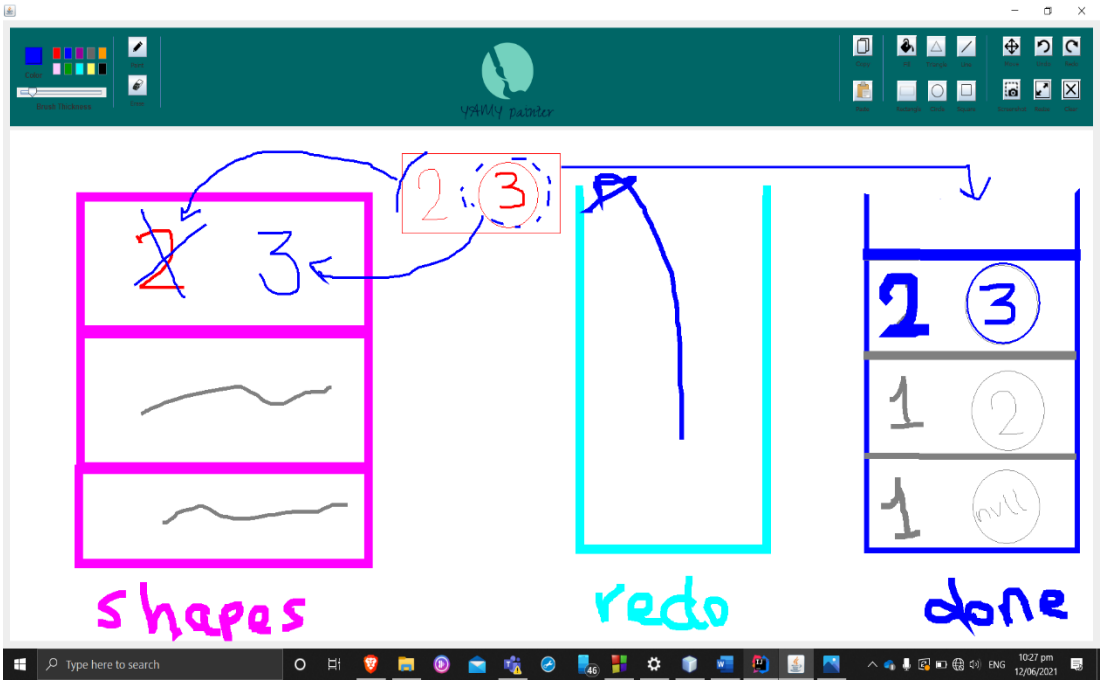
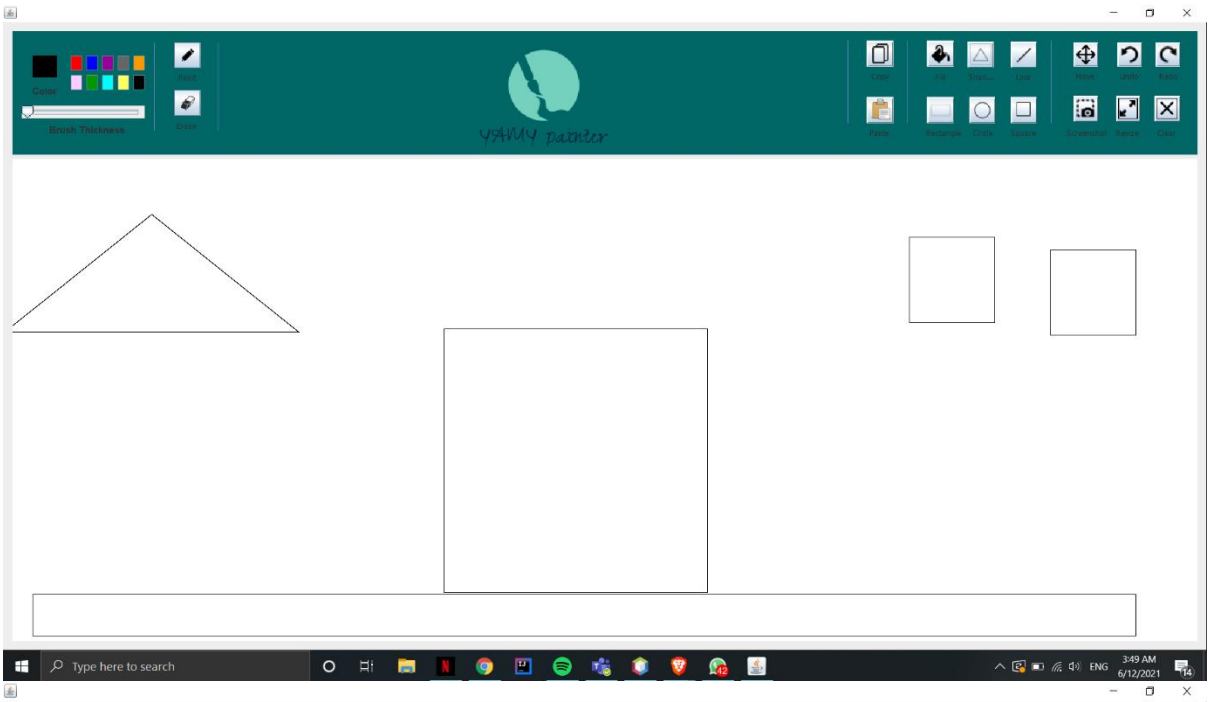


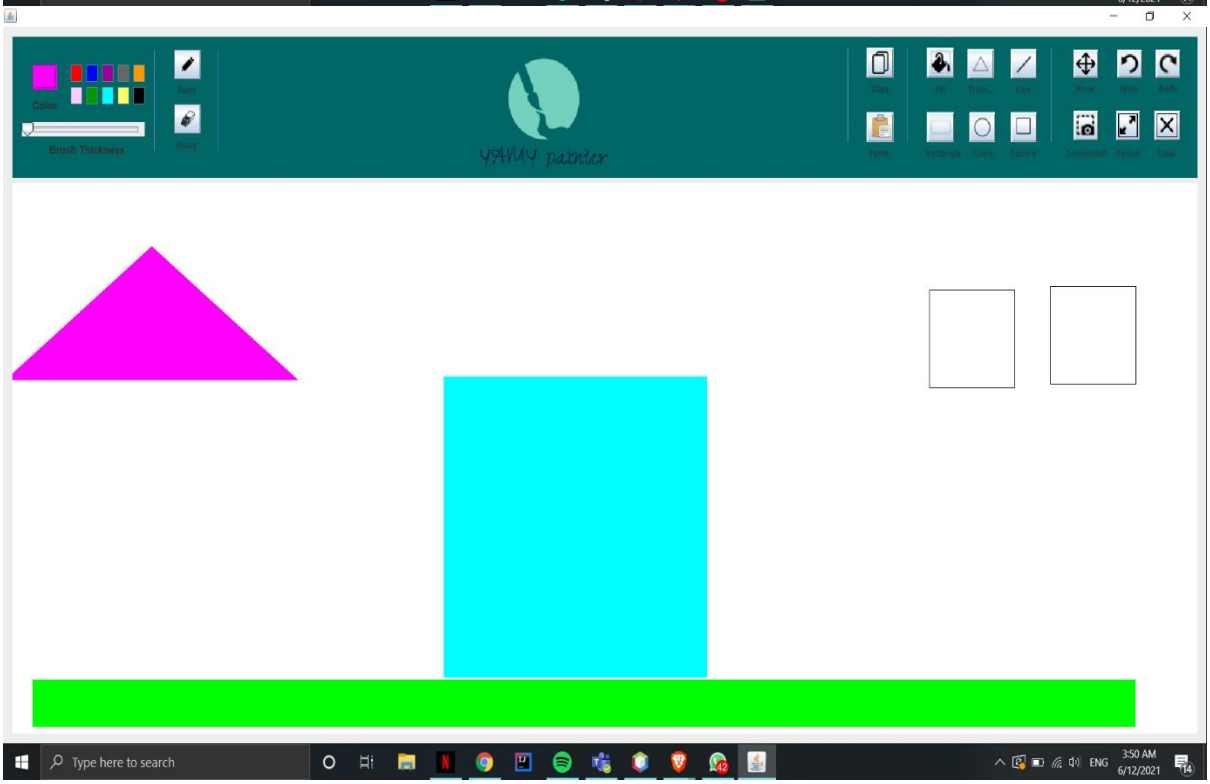
Figure 5.3 Redo procedure

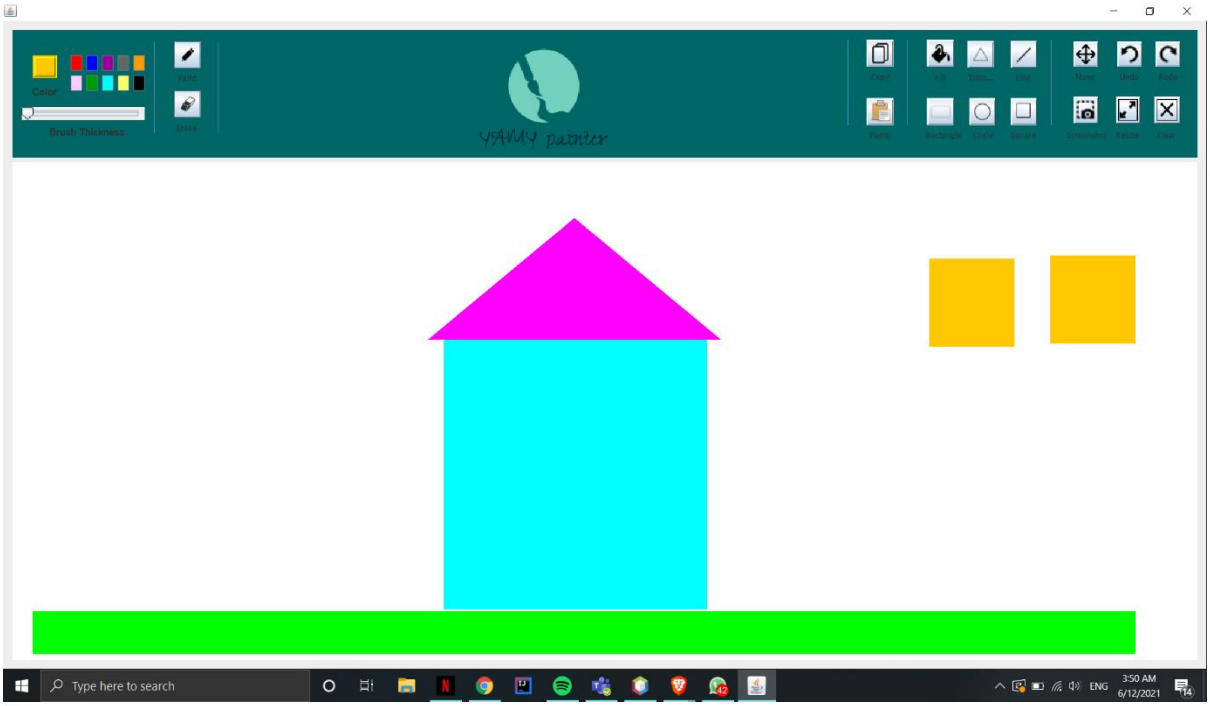
In case the topShape (popped) in the ‘done’ stack has nextShape = null, the topShape is searched in the shapes array list and removed. Then it is added to the redo stack.

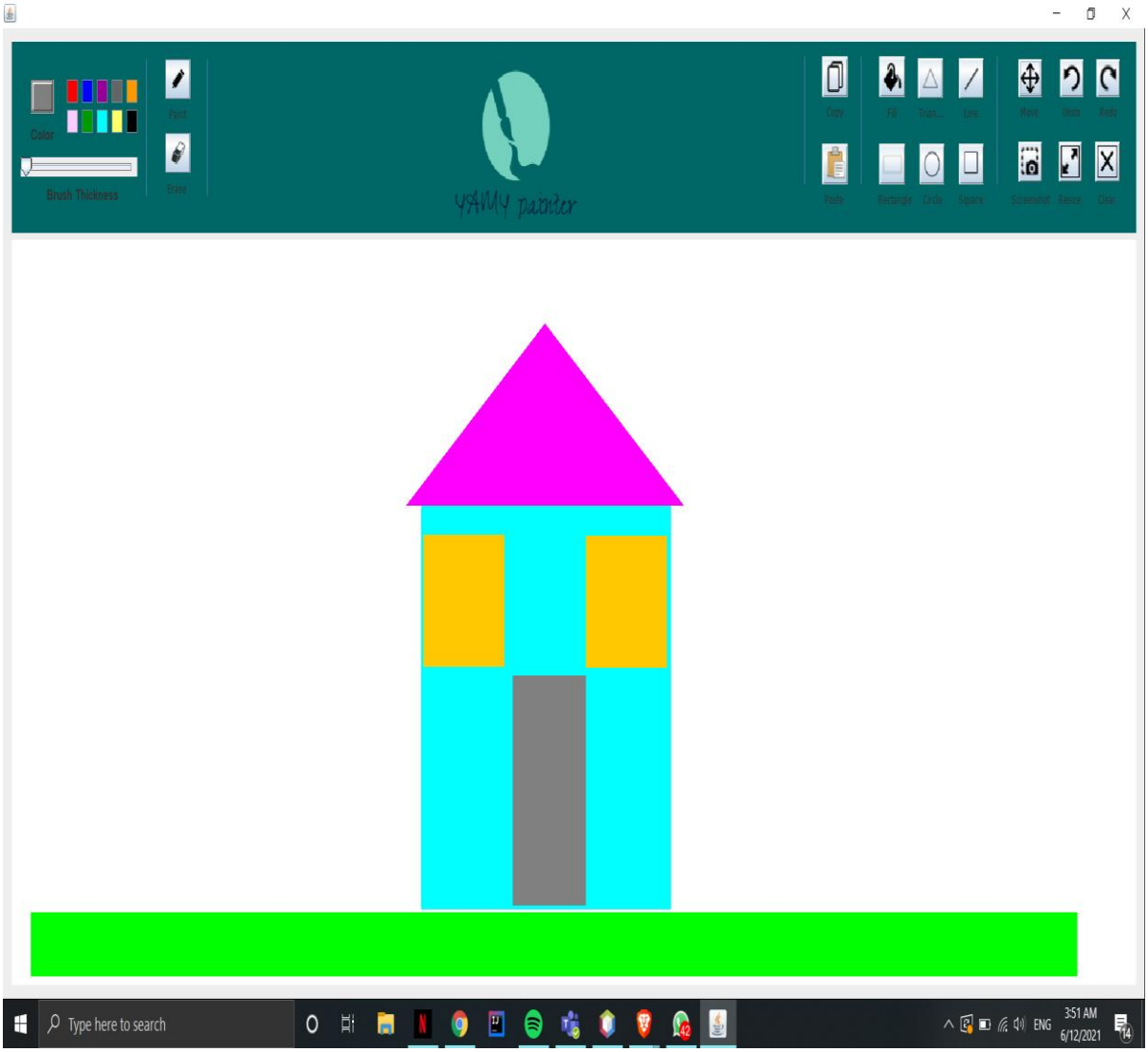
If the done stack is empty, undo does nothing. Same for redo and redo stack.

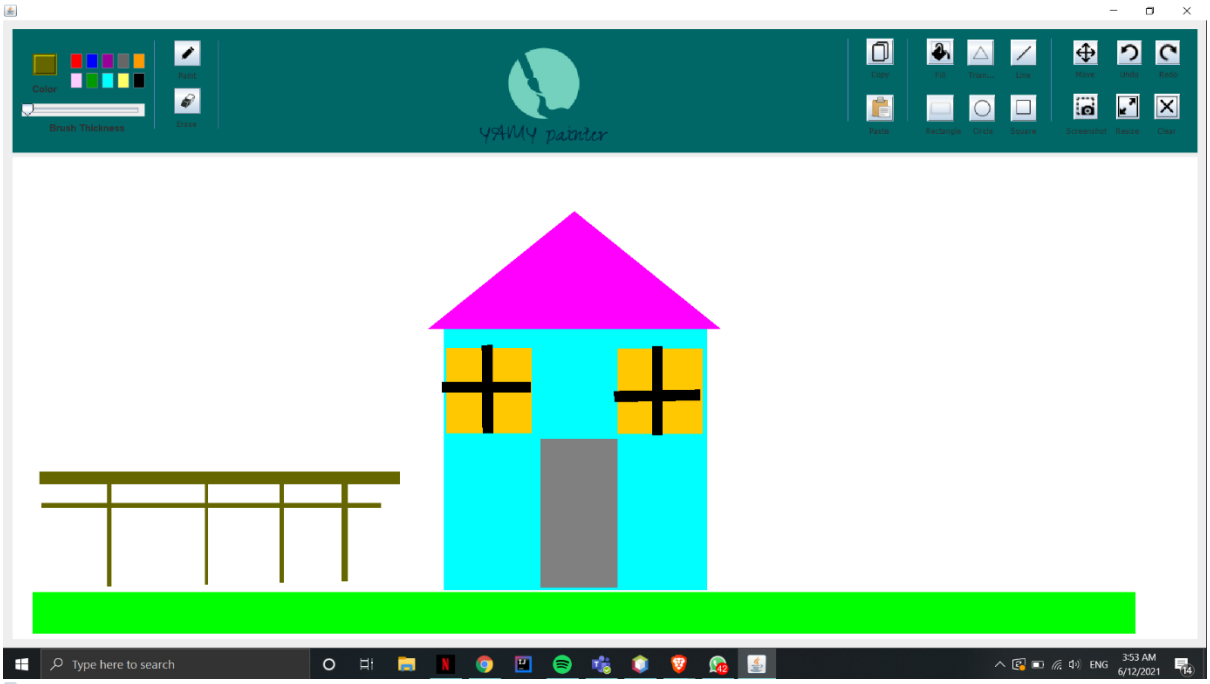
Sample Run

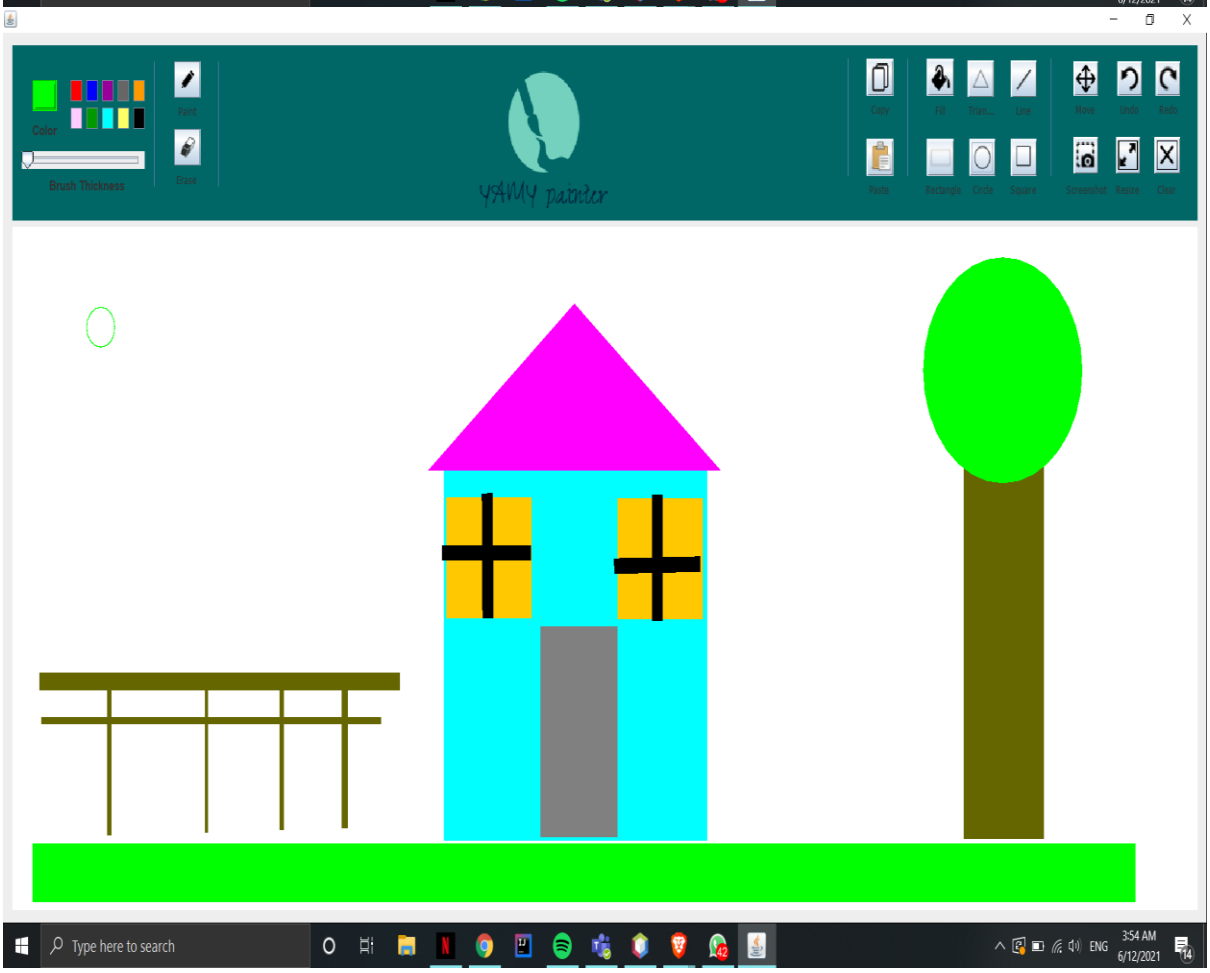

Color
Brush Thickness
YAMY painter
Type here to search
ENG
3:49 AM
6/12/2021

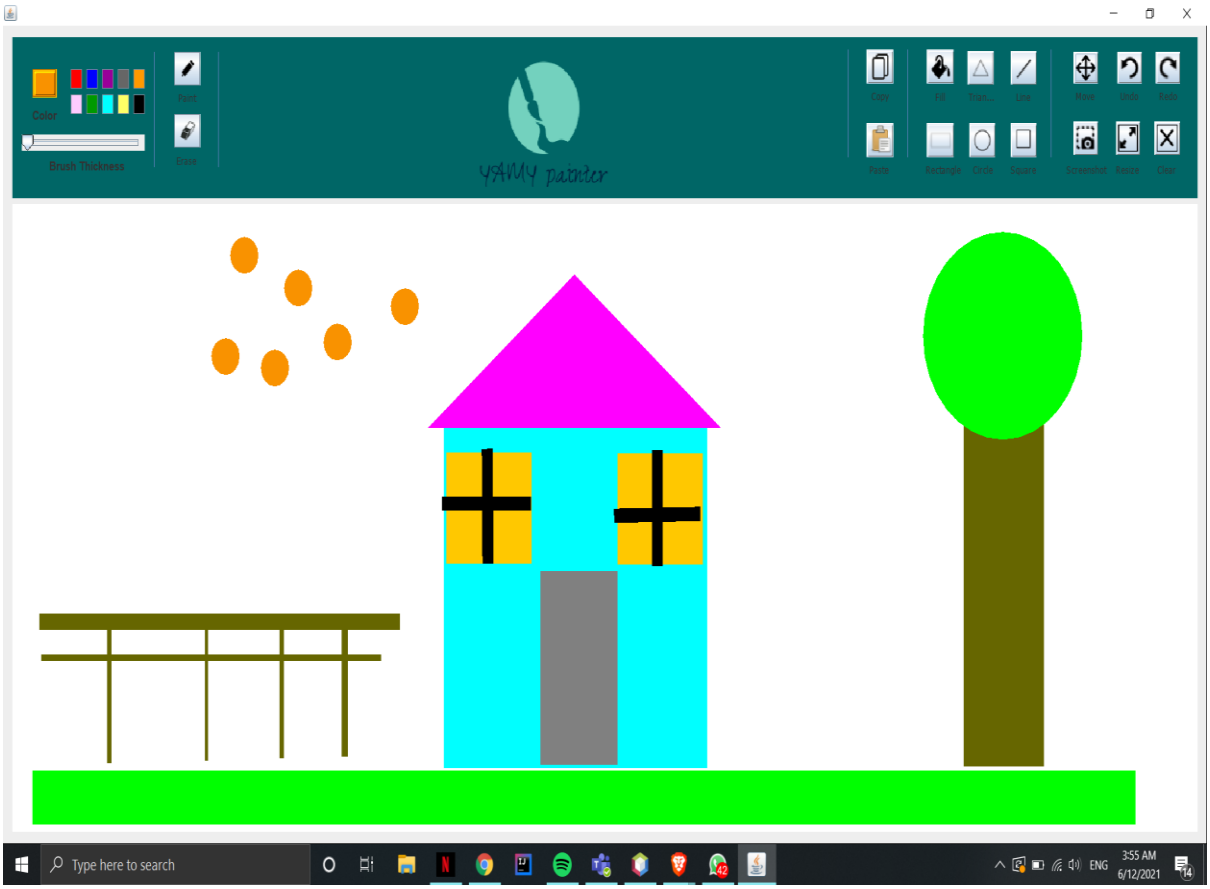

Color
Brush Thickness
YAMY painter
Type here to search
ENG
3:50 AM
6/12/2021

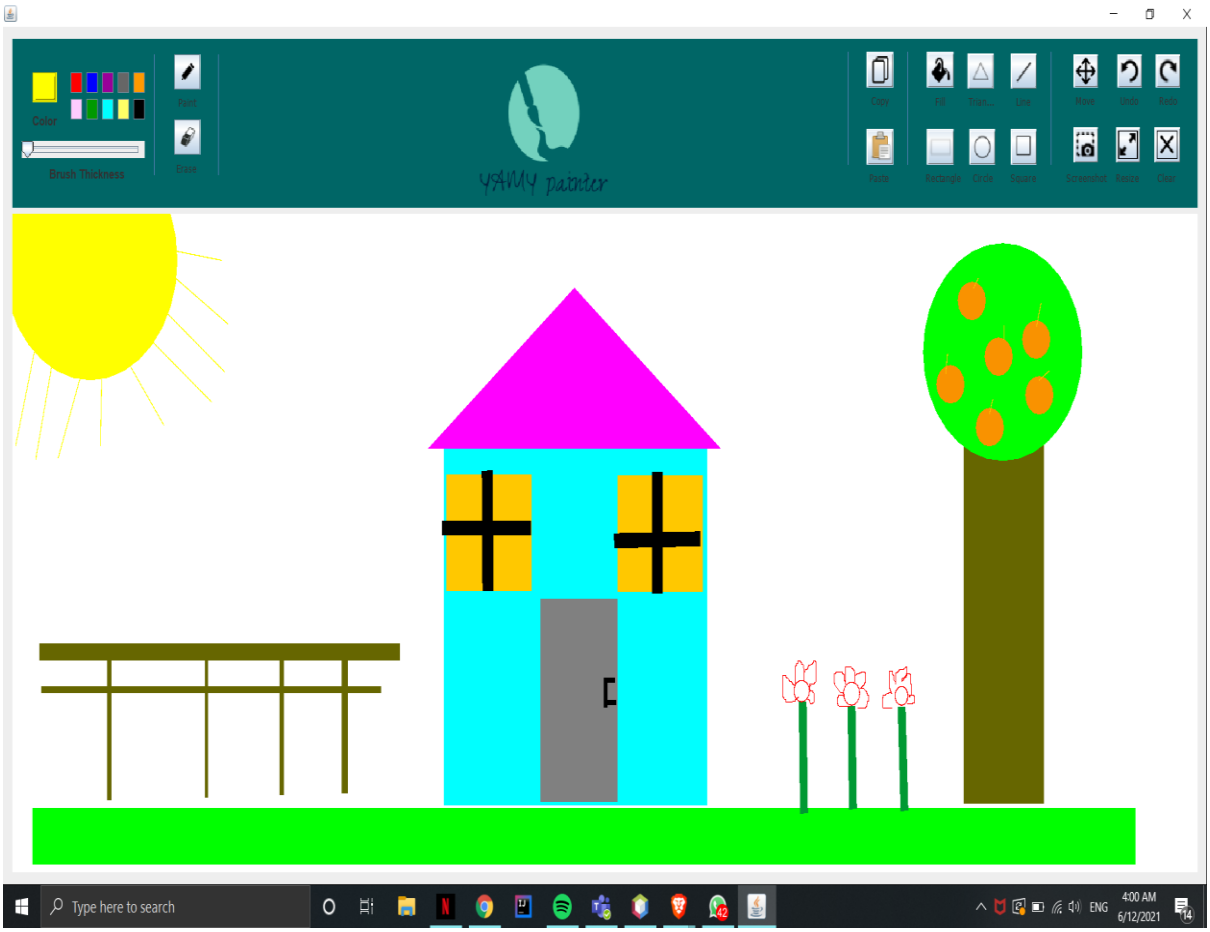

Color
Brush Thickness
YAMY painter
Type here to search
ENG
3:50 AM
6/12/2021


Color
Brush Thickness
YAMY painter
Type here to search
ENG
3:51 AM
6/12/2021

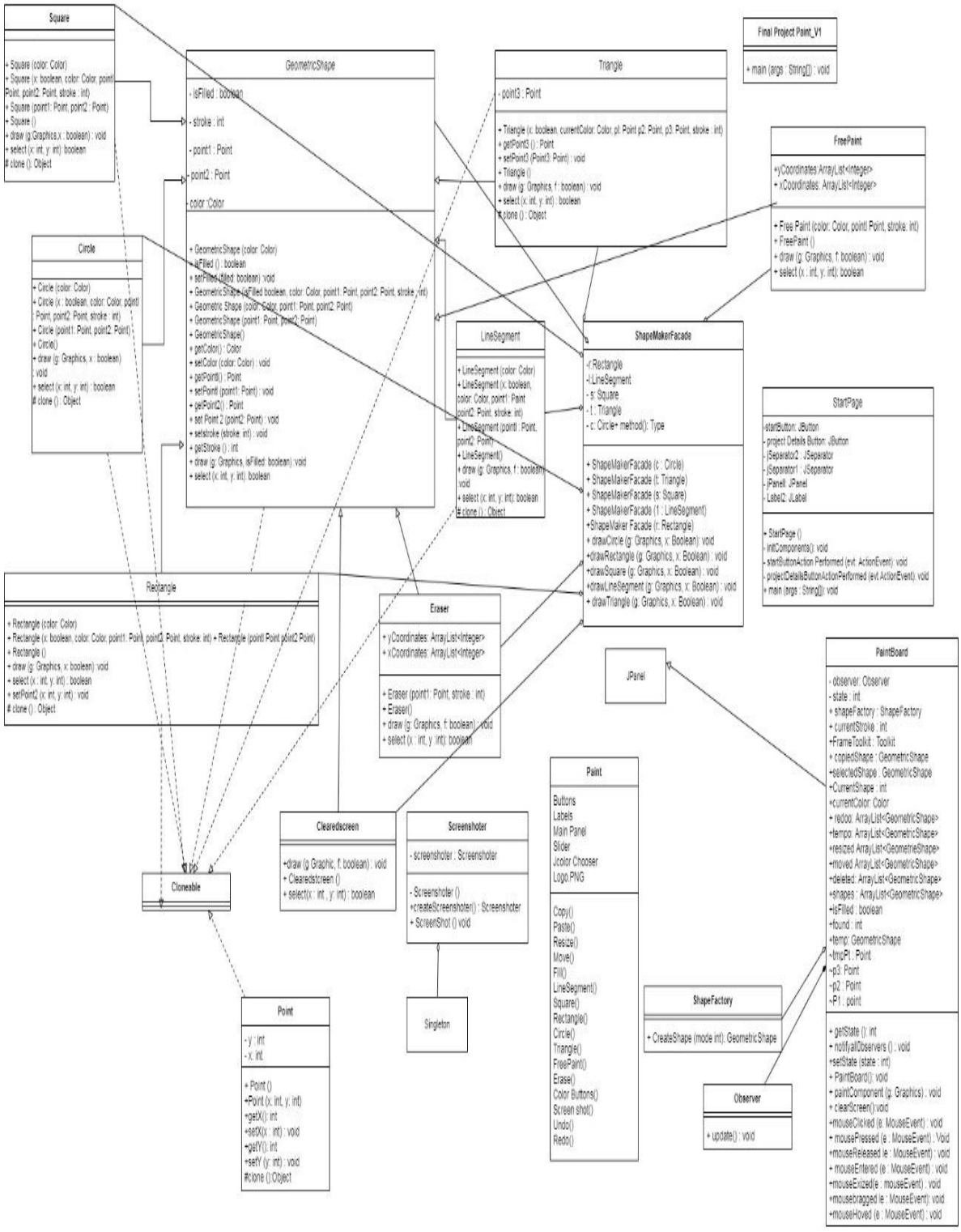

YAMY painter
Type here to search
ENG
3:53 AM
6/12/2021


YAMY painter
Type here to search
ENG
3:54 AM
6/12/2021


YAMY painter
Color
Brush Thickness
Type here to search
ENG
3:55 AM
6/12/2021


YAMY painter
Color
Brush Thickness
Type here to search
ENG
4:00 AM
6/12/2021

ClassDiagram



Use-CaseDiagram