# Drawing and Painting Application

## Assignment 5

- Asala Ahmed ID: 6916
- Jera Gamal ID: 6914
- Mohab Ayman ID: 7127

## Introduction:

Painting and Drawing programs are very popular where many users need it. It offer large number of features drawing many shapes and edit on them by coloring, resizing, deleting and also use set of operations as copying shapes and undo or redo actions.

## Implementation:

The code contains two source packages the 1st is classes (contains 11 classes) and the 2nd is paint (contains 1 frame and 1 class).

## 1 .1st package (classes)

### *Part 1 implementation of shapes*

#### *Class1 Points:*

Contains the 2 main points that we are going to build the shapes with, also contains getters and setters to them and their constructor .

#### *Class2 Shape:*

```
public abstract class Shape extends Points implements Cloneable.
```

It contains all data of shapes were other shapes extends it.

- Constructor contains the 2 main points.

- Color of the shape, getter and setter for it.

- public abstract void draw.

Method used to draw the shape using methods of Graphics class, implemented in every shape class.

- public abstract boolean contains.

Return a boolean identify weather the shape contains point(x,y)

- public Object clone()

This method override clone() from object class, creates a copy of object of shape.

- public abstract void move

Implemented in every shape subclass .

***Class3 Line:*** public class Line extends Shape

-contains constructor initialize starting position (x1,y1) , ending position (x2,y2) and shape color.

-public void draw(Graphics g).

Uses drawline(), get and set color methods from Graphics class to draw a line from point (x1,y1) to (x2,y2).

***Class4 Triangle:*** public class Triangle extends Shape

-declare 3rd point (x3,y3) and make get and set for it.

-constructor initialize 3 points of triangle and its color.

-public void draw(Graphics g)

Uses drawpolygon() from Graphics class which takes array of x coordinates, array of y coordinates and number of points, and uses getcolor and setcolor to apply the color chosen by user.

-public boolean contains(int x, int y)

Uses method polygon(),contains() to determine whether the point (x,y) Inside the triangle or not.

***Class5 Circle:*** public class Circle extends Shape

-constructor initialize 2 coordinates of circle and its color.

- public void draw(Graphics g)

Uses drawoval() method from Graphics class that takes the 1st point (x,y), the width and the height as input to draw the circle .

-public boolean contains(int x, int y)

Uses Ellipse2D.Float class to check whether the coordinate (x,y) is inside the circle or not.

***Class6 Square:*** public class Square extends Shape

-constructor initialize 2 coordinates of square and its color.

- public void draw(Graphics g)

Uses drawrect() method from Graphics class that takes the 1st point (x,y), the width and the height as input to draw the square .

-public boolean contains(int x, int y)

Uses java.awt.Rectangle class to check whether the coordinate (x,y) is inside the square or not.

## *Class7 Rectangle:* public class Rectangle extends Shape

-constructor initialize 2 coordinates of rectangle and its color.

public void draw(Graphics g)

Uses drawrect() method from Graphics class that takes the 1st point (x,y), the width and the height as input to draw the rectangle .

-public boolean contains(int x, int y)

Uses java.awt.Rectangle class to check whether the coordinate (x,y) is inside the rectangle or not.

## *Class8 Filledtriangle:* public class FilledTriangle extends Shape

-declare 3rd point (x3,y3) and make get and set for it.

-constructor initialize 3 points of triangle and its color.

-public void draw(Graphics g)

Uses fillpolygon() from Graphics class which takes array of x coordinates, array of y coordinates and number of points, and uses getcolor and setcolor to apply the color chosen by user.

-public boolean contains(int x, int y)

Uses method polygon(),contains() to determine whether the point (x,y) Inside the triangle or not.

## *Class9 Filledcircle*: public class FilledCircle extends Shape

-constructor initialize 2 coordinates of circle and its color.

- public void draw(Graphics g)

Uses filloval() method from Graphics class that takes the 1st point (x,y), the width and the height as input to draw the circle .

-public boolean contains(int x, int y)

Uses Ellipse2D.Float class to check whether the coordinate (x,y) is inside the circle or not.

## *Class10 Filledsquare:* public class FilledSquare extends Shape

-constructor initialize 2 coordinates of square and its color.

- public void draw(Graphics g)

Uses fillRect() method from Graphics class that takes the 1st point (x,y), the width and the height as input to draw the square .

-public boolean contains(int x, int y)

Uses java.awt.Rectangle class to check whether the coordinate (x,y) is inside the square or not.

## *Class11 Filledrectangle:* public class FilledRectangle extends Shape

-constructor initialize 2 coordinates of rectangle and its color.

public void draw(Graphics g)

Uses fillRect() method from Graphics class that takes the 1st point (x,y), the width and the height as input to draw the rectangle .

-public boolean contains(int x, int y)

Uses java.awt.Rectangle class to check whether the coordinate (x,y) is inside the rectangle or not.

## *2.2nd package(Paint)*

### *Part 2 implementing  drawing and painting tools*

### *1.class Board:*

public class Board extends JPanel implements MouseListener, MouseMotionListener.

-First we import all shapes classes implemented in part 1.

-Declare some attributes that will be used in this class:

-instance of shape(selectedShape)- object of class Board (Board1)

-Arraylist of shapes(shapes)- points(x1,x2,x3,y1,y2,y3)

-boolean (before,after)-int(point1,point2)-color(currentcolor)

-stack(arraylist(shape))(undo,redo).

-constructor initializes arraylist(shapes), addMouseListener() and addMouseMotionListener().

-public void undo()

If undo stack contains more than one action then we pop the last action from undo stack and push it to the redo stack, let arraylist shapes = last action (after pop) in undo stack by using peek(), call repaint() function to apply the graphical changes made.

`-public void redo()`

If redo stack contains atleast one action then we pop the last action from redo stack and push it to the undo stack, let arraylist shapes = last action (after pop) in undo stack by using peek(), call repaint() function to apply the graphical changes made.

`- private ArrayList<Shape> copy(ArrayList<Shape> shapes)`

This function creates a copy of the arraylist shapes (co) and returns the cloned arraylist (co).

`-public static Board getInstance()`

This method creates an instance of class Board if it is the first time to create it and returns the new board.

`-protected void paintComponent(Graphics g)`

This method overrides the super class jPanel, by invoking super.paintComponent(g) we allow drawing on the the jPanel, make new Iterator of arraylist shapes to draw the shapes in It if the Iterator still have shapes (it.hasNext() is true).

`-public void mouseClicked(MouseEvent me)`

Implement the method of interface MouseListener which is invoked when the mouse button clicked(press then release) on a component.

-public void mousePressed(MouseEvent me)

Implement the method of interface MouseListener which is invoked when the mouse button pressed on a component.

Firstly we initialize before bool to false, then get coordinate (x1,y1) were mouse is pressed (start point).we use switch case to identify the shape drawing in its first phase.

## *Case 0:(Line)*

Make new line with the color and the point (x1,y1) as the starting point as input, then add it to the arraylist shapes and repaint().

## *Case 1:(triangle)*

If arraylist shapes contain elements, make a shape(tri) = last shape in shapes ,if tri is instance of triangle then declare new triangle (temp)=tri, if x1 (x of 1st coordinate) is equal to x2 (x of 2nd coordinate) then make 1st coordinate equal to 2nd coordinate and then set before to true. If before is false then initialize new triangle with (x1,y1) as three coordinates , the color as inputs and add it to shapes. At the end call repaint().

## Case 2: (circle)

Make new circle takes 1st and 2nd coordinates as point (x1,y1) , color as input then add it to the arraylist shapes and repaint().

## Case 3: (rectangle)

Make new rectangle takes 1st and 2nd coordinates as point (x1,y1) , color as input then add it to the arraylist shapes and repaint().

## Case 4: (square)

Make new square takes 1st and 2nd coordinates as point (x1,y1) and (x2,y2), color as input then add it to the arraylist shapes and repaint().

## Case 5: (filled triangle)

If arraylist shapes contain elements, make a shape(tri) = last shape in shapes ,if tri is instance of filledtriangle then declare new filledtriangle (temp)=tri, if x1 (x of 1st coordinate) is equal to x2 (x of 2nd coordinate) then make 1st coordinate equal to 2nd coordinate and then set before to true. If before is false then initialize new filledtriangle with (x1,y1) as three coordinates , the color as inputs and add it to shapes. At the end call repaint().

## Case 6: (filled circle)

Make new filledcircle takes 1st and 2nd coordinates as point (x1,y1) , color as input then add it to the arraylist shapes and repaint().

## Case 7: (filled square)

Make new filledsquare takes 1st and 2nd coordinates as point (x1,y1) and (x2,y2), color as input then add it to the arraylist shapes and repaint().

## Case 8: (filled rectangle)

Make new filled rectangle takes 1st and 2nd coordinates as point (x1,y1) , color as input then add it to the arraylist shapes and repaint().

## Case 9: (select)

Check iteratively whether the shape at current index contain the coordinate (x1,y1) ,if yes let selectedShape (declared above) equals the shape at the current index, the print “selected” and break out of the loop.

## Case 11: (copy)

First check if there is a selected shape, if yes declare new shape(sh) and clone the selected shape to it . relay the generated copy and then add it to the arraylist shapes then repaint.

## Case 12: (Move)

When clicking on the shape print click.

## Case 13: (undo)

Call undo method (previously declared) and print undo.

## Case 14: (redo)

Call redo method (previously declared) and print redo.

## Case 15: (delete)

Call method remove() to remove the selected shape from arraylist shapes then repaint().

- public void mouseReleased(MouseEvent me)

Implement the method of interface MouseListener .Invoked when a mouse button has been released on a component. At this point we push a copy of arraylist shapes to undo stack.

-public void mouseEntered(MouseEvent me)

Implement the method of interface MouseListener .Invoked when a mouse enters a component.

- public void mouseExited(MouseEvent me)

Implement the method of interface MouseListener .Invoked when a mouse exits a component.

- public void mouseDragged(MouseEvent me)

Implement the method of interface MouseListener Firstly if the arraylist is not empty, get the coordinate of the second point (x2,y2).

Declare new shape(s) and let equal last shape in arraylist, then we use switch case to continue drawing shapes.

## *Case 0:(Line)*

Check if s is a line then create line and set its 2nd coordinate(x2,y2) to x2 and y2, then repaint and set first to false.

## *Case 1:(triangle)*

First get the 3rd coordinate (x3,y3) Check if s is a triangle then create triangle and set its 3rd coordinate(x3,y3) to (x2,y2) , then repaint and set first to false.

## Case 2: (circle)

Check if s is a circle then create circle and set its 2nd coordinate(x2,y2) to x2 and y2, then repaint and set first to false.

## Case 3: (rectangle)

Check if s is a rectangle then create rectangle and set its 2nd coordinate(x2,y2) to x2 and y2, then repaint and set first to false.

## Case 4: (square)

Check if s is a square then create square and set its 2nd coordinate(x2,y2) to x2 and y2, then repaint and set first to false.

## Case 5: (filled triangle)

First get the 3rd coordinate (x3,y3) Check if s is a filledtriangle then create filledtriangle and set its 3rd coordinate(x3,y3) to (x2,y2) , then repaint and set first to false.

## Case 6: (filled circle)

Check if s is a filledcircle then create filledcircle and set its 2nd coordinate(x2,y2) to x2 and y2, then repaint and set first to false.

## Case 7: (filled square)

Check if s is a filledsquare then create filledsquare and set its 2nd coordinate(x2,y2) to x2 and y2, then repaint and set first to false.

## Case 8: (filled rectangle)

Check if s is a filledrectangle then create filledrectangle and set its 2nd coordinate(x2,y2) to x2 and y2, then repaint and set first to false.

## Case 10: (resize)

First get the 3rd coordinate (x3,y3) let (s) equal selectedshape, set 2nd coordinate of (s) (x2,y2) to (x3,y3)  , point x2=x3 and point y2=y3, then repaint.

<u>Case 12: (Move)</u>

Consist on 2 parts move for triangle and for other shapes.

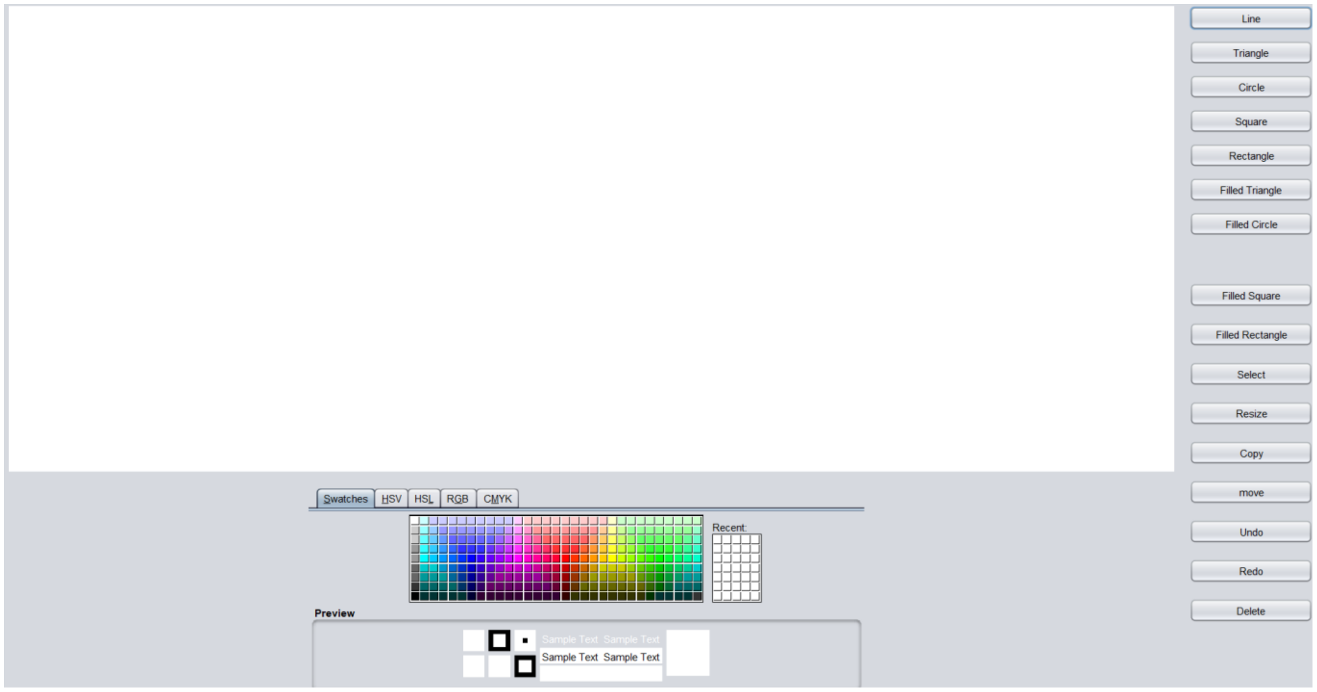
First declare shape s1 = selected shape.

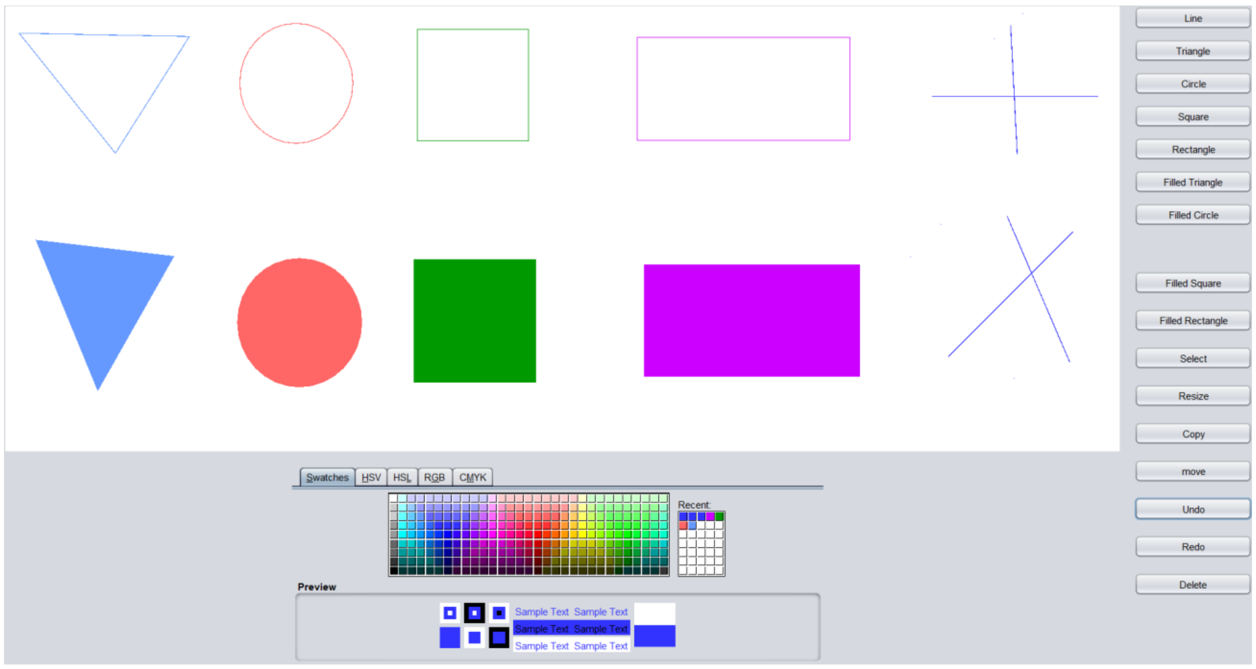
If s1 is triangle, create a triangle (t) ,set s1.(x1,y1)and s1.(x2,y2) and t.(x3,y3) to the new positions. Let point x1=x2 and y1=y2 then repaint.

For other shapes, set s1.(x1,y1)and s1.(x2,y2) to the new positions. Let point x1=x2 and y1=y2 then repaint.

## *<u>Home frame (GUI):</u>*

It contains main method and link the actions performed by buttons with Board class .


Line
Triangle
Circle
Square
Rectangle
Filled Triangle
Filled Circle
Filled Square
Filled Rectangle
Select
Resize
Copy
move
Undo
Redo
Delete
Swatches
HSV
HSL
RGB
CMYK
Recent:
Preview
Sample Text Sample Text
Sample Text Sample Text


Line
Triangle
Circle
Square
Rectangle
Filled Triangle
Filled Circle
Filled Square
Filled Rectangle
Select
Resize
Copy
move
Undo
Redo
Delete
Swatches
HSV
HSL
RGB
CMYK
Recent:
Preview
Sample Text Sample Text
Sample Text Sample Text
Sample Text Sample Text

## -Design patterns implemented

### 1)singleton pattern:

In class Board we need only one instance of it (Board1)

So we create method getinstance to return Board1.

```
public static Board getInstance() {
    if (Board1 == null) {
        Board1 = new Board();
    }
    return Board1;
}
```

### 2)prototype pattern:

public abstract class Shape extends Points implements Cloneable.

```
public Object clone() throws CloneNotSupportedException{
    Shape s= (Shape)super.clone();
    s.setColor(this.color);
    return s;
}
```

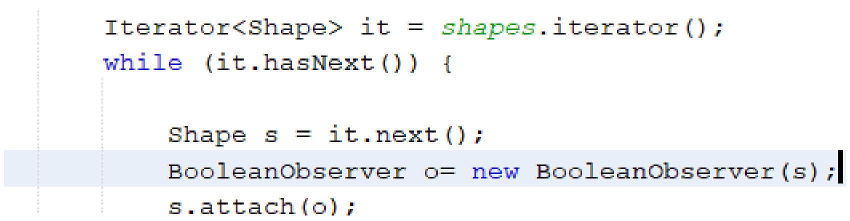
This design pattern used to make a copy of shape and return the copy.

```
Shape sh = (Shape) selectedShape.clone();
```

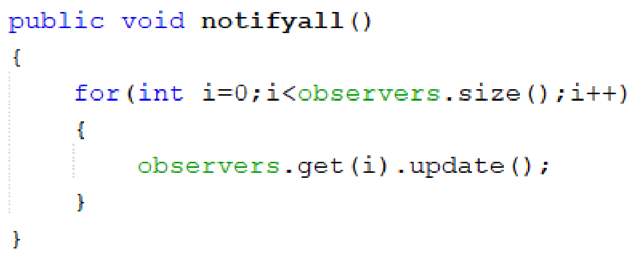
### 3)observer pattern:

Observer pattern takes an action when the observable change.

Observable is the shapes as when the color is changed (setcolor()) it notify the observer (notifyall()) then the observer call update() to print the observer + current color .



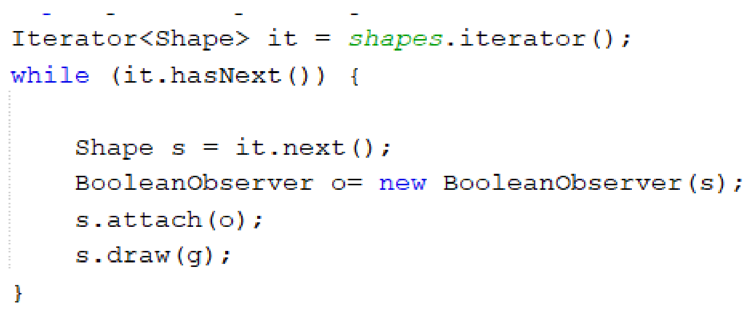
```
    Iterator<Shape> it = shapes.iterator();
    while (it.hasNext()) {

        Shape s = it.next();
        BooleanObserver o= new BooleanObserver(s);
        s.attach(o);
```



```
public void notifyall()
{
    for(int i=0;i<observers.size();i++)
    {
        observers.get(i).update();
    }
}
```

## 4)Iterator pattern

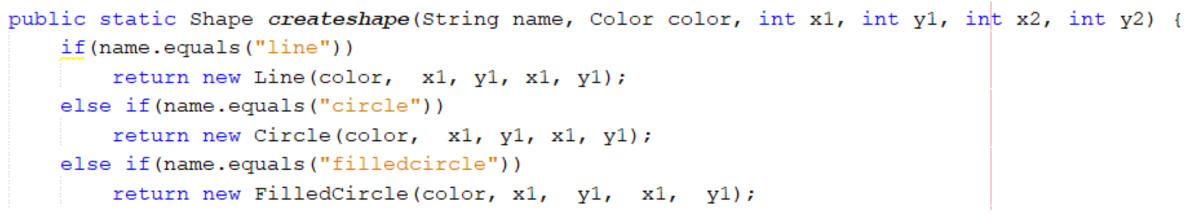
Iterator pattern affords loop around arraylist shapes while the list have next(havenext()) shape to perform certain action (drawing the shape s.draw()).



```
Iterator<Shape> it = shapes.iterator();
while (it.hasNext()) {

    Shape s = it.next();
    BooleanObserver o= new BooleanObserver(s);
    s.attach(o);
    s.draw(g);
}
```

## 5)Factory pattern

Factory design pattern used to create object of a shape according to its type (circle,square,……) and return the new object.



```
public static Shape createshape(String name, Color color, int x1, int y1, int x2, int y2) {
    if(name.equals("line"))
        return new Line(color,  x1, y1, x1, y1);
    else if(name.equals("circle"))
        return new Circle(color,  x1, y1, x1, y1);
    else if(name.equals("filledcircle"))
        return new FilledCircle(color, x1,  y1,  x1,  y1);
```

```
Line l = (Line) Classes.Shapefactory.createshape("line",currentColor, x1, y1, x1, y1);
```

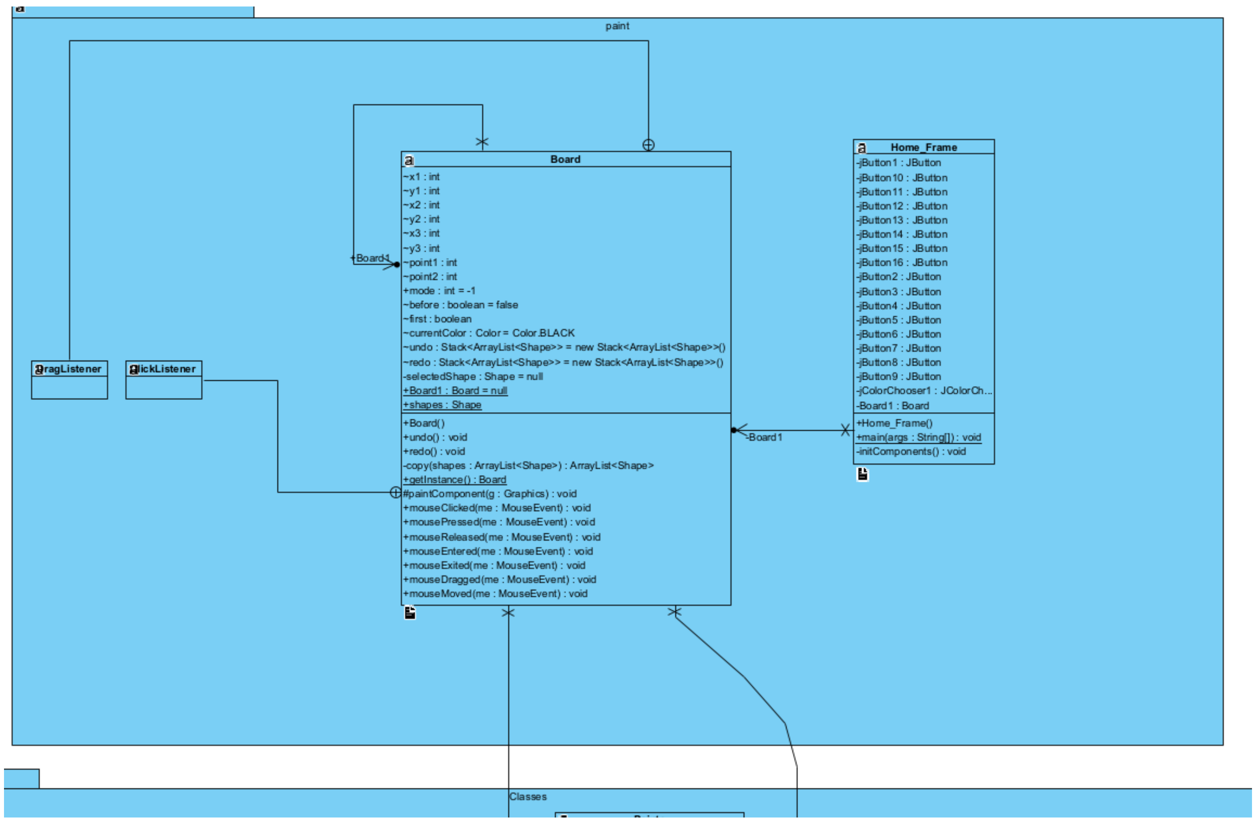
## 6) façade pattern:

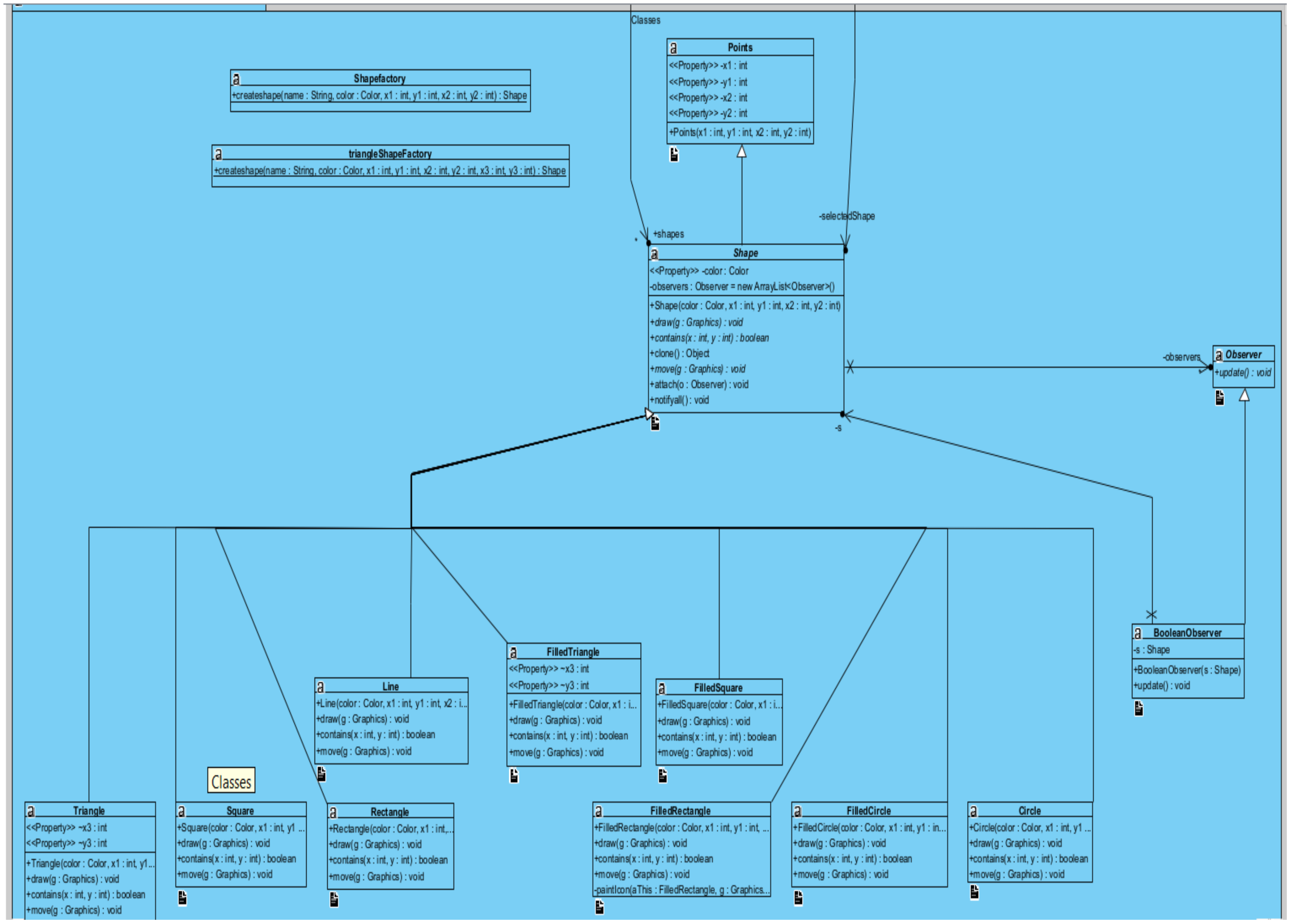
Facade pattern hides the complexities of the system. As it involves a single class (Board) to call methods, instances from existing system classes(package classes).

```
s.attach(o)    temp.setX2(x1);   x1 = me.getX();
s.draw(g);     temp.setY2(y1);   y1 = me.getY();
```

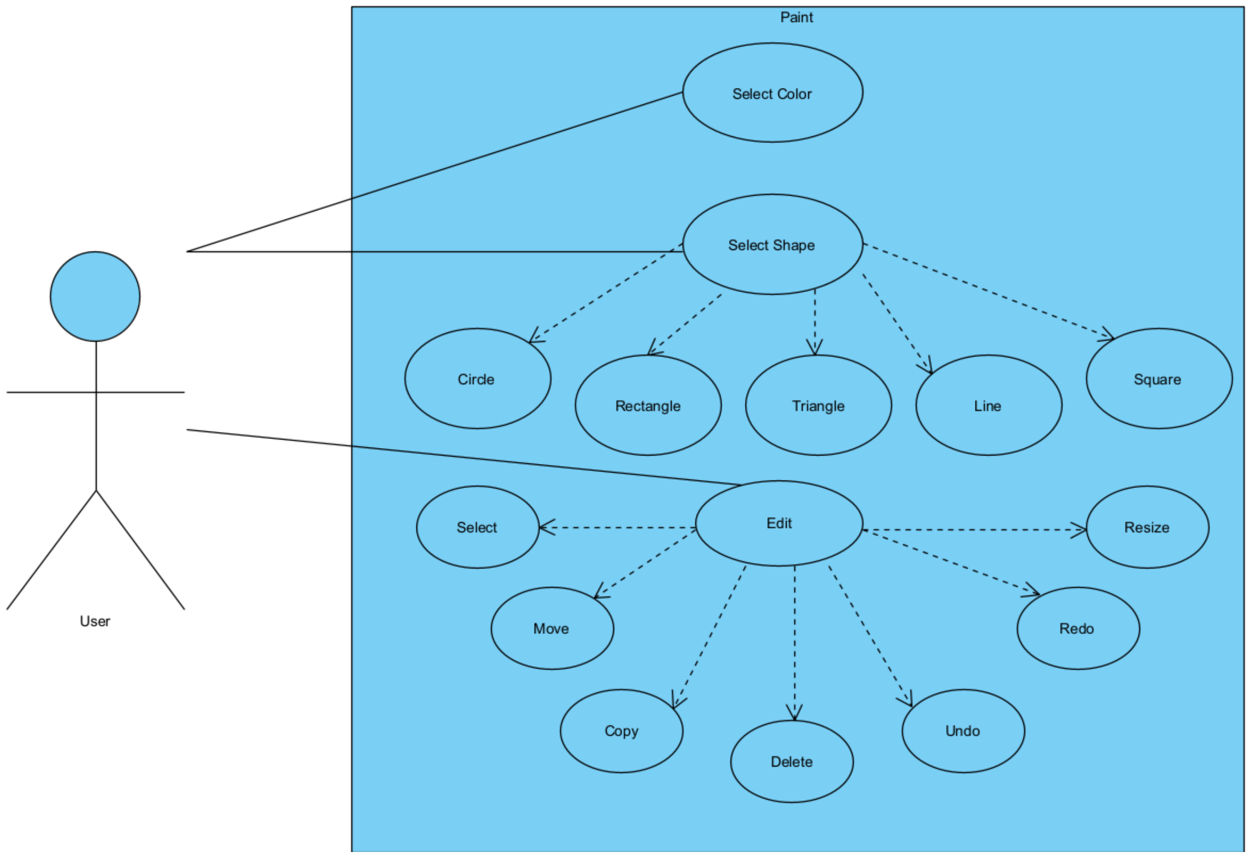
## *UML diagrams:*

### 1)class diagram




Classes
Shapefactory
+createshape(name : String, color : Color, x1 : int, y1 : int, x2 : int, y2 : int) : Shape
triangleShapeFactory
+createshape(name : String, color : Color, x1 : int, y1 : int, x2 : int, y2 : int, x3 : int, y3 : int) : Shape
Points
<<Property>> -x1 : int
<<Property>> -y1 : int
<<Property>> -x2 : int
<<Property>> -y2 : int
+Points(x1 : int, y1 : int, x2 : int, y2 : int)
+shapes
-selectedShape
Shape
<<Property>> -color : Color
-observers : Observer = new ArrayList<Observer>()
+Shape(color : Color, x1 : int, y1 : int, x2 : int, y2 : int)
+draw(g : Graphics) : void
+contains(x : int, y : int) : boolean
+clone() : Object
+move(g : Graphics) : void
+attach(o : Observer) : void
+notifyall() : void
-observers
Observer
+update() : void
-s
BooleanObserver
-s : Shape
+BooleanObserver(s : Shape)
+update() : void
Line
+Line(color : Color, x1 : int, y1 : int, x2 : i...
+draw(g : Graphics) : void
+contains(x : int, y : int) : boolean
+move(g : Graphics) : void
FilledTriangle
<<Property>> ~x3 : int
<<Property>> ~y3 : int
+FilledTriangle(color : Color, x1 : i...
+draw(g : Graphics) : void
+contains(x : int, y : int) : boolean
+move(g : Graphics) : void
FilledSquare
+FilledSquare(color : Color, x1 : i...
+draw(g : Graphics) : void
+contains(x : int, y : int) : boolean
+move(g : Graphics) : void
Classes
Triangle
<<Property>> ~x3 : int
<<Property>> ~y3 : int
+Triangle(color : Color, x1 : int, y1...
+draw(g : Graphics) : void
+contains(x : int, y : int) : boolean
+move(g : Graphics) : void
Square
+Square(color : Color, x1 : int, y1 ...
+draw(g : Graphics) : void
+contains(x : int, y : int) : boolean
+move(g : Graphics) : void
Rectangle
+Rectangle(color : Color, x1 : int,...
+draw(g : Graphics) : void
+contains(x : int, y : int) : boolean
+move(g : Graphics) : void
FilledRectangle
+FilledRectangle(color : Color, x1 : int, y1 : int, ...
+draw(g : Graphics) : void
+contains(x : int, y : int) : boolean
+move(g : Graphics) : void
-paintIcon(aThis : FilledRectangle, g : Graphics...
FilledCircle
+FilledCircle(color : Color, x1 : int, y1 : in...
+draw(g : Graphics) : void
+contains(x : int, y : int) : boolean
+move(g : Graphics) : void
Circle
+Circle(color : Color, x1 : int, y1 ...
+draw(g : Graphics) : void
+contains(x : int, y : int) : boolean
+move(g : Graphics) : void

## 2)use case diagram:


Paint
Select Color
Select Shape
Circle
Rectangle
Triangle
Line
Square
User
Select
Edit
Resize
Move
Redo
Copy
Delete
Undo