

VECTOR BASED DRAWING APPLICATION REPORT

Omar Shawky Anwar -Dahlia Chehata Mahmoud | OOP | November 3, 2016

Overview:

This paint application is an implementation of an object oriented designed model that covers different geometric shapes such as: Line Segment, Circle, Ellipse, Triangle, Rectangle and Square.

It follows OOP principles and the concepts of inheritance and polymorphism.

Our program offers a big number of features that includes: Drawing, Coloring, and Resizing. It also includes a number of built in, and extensible set of geometric shapes, and it allows the user to undo or redo any instructions performed so as to make the application more usable.

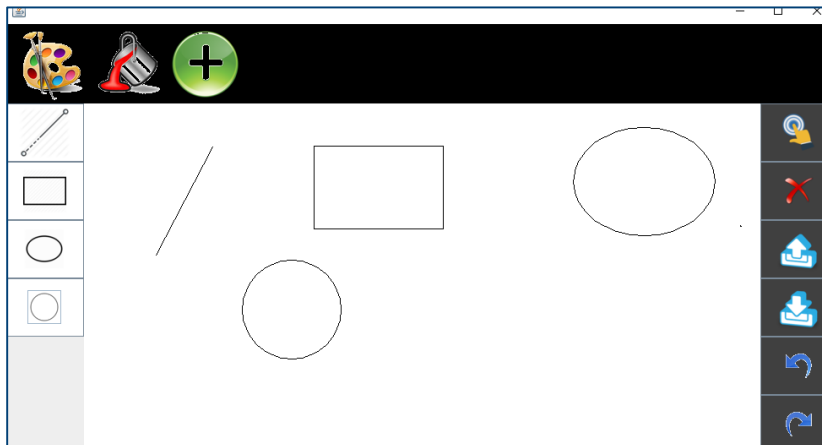
It offers many more features such as saving and loading from Json and XML, shape moving, deletion, dynamic class loading, selection which will be discussed separately.

Design:

The graphical user interface GUI design is simple and close to any paint application with a shapes panel ,actions panel and another one serving coloring purposes.

As usual, by clicking on any shape, the user is able to draw it by moving the cursor on the drawing frame

Screenshot of the program:

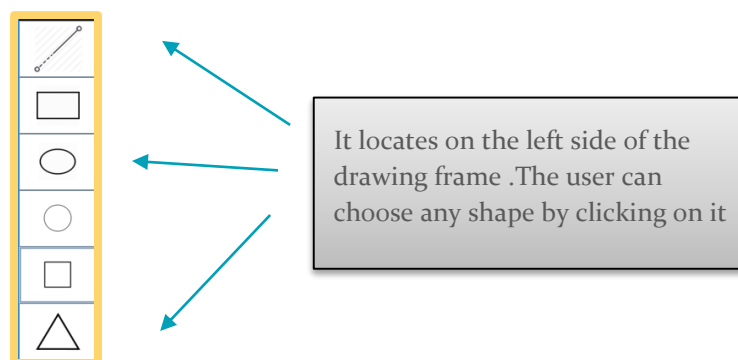


User Guide



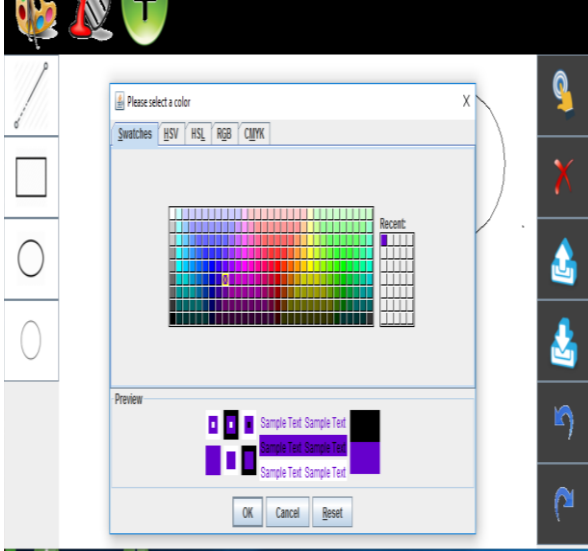
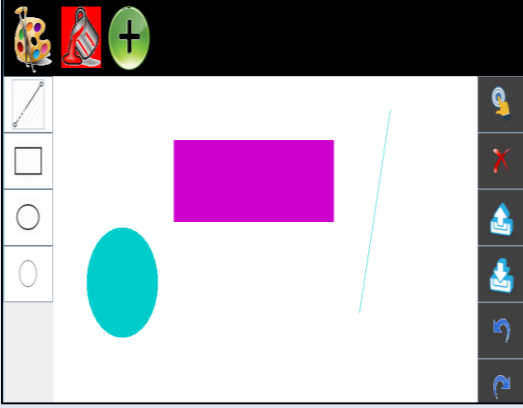
- Tasks and actions details:

	Select	The user is able to select any shape by clicking on it , and the shape borders are thickened immediately with a little square on the upper left and lower right corners
	Delete	After the selection, by clicking this icon the shape is erased and disappears from drawing frame
	Save	An option message dialog appears on click and the user is asked where to save his work in Json file or in XML
	Load	An option message dialog appears on click and the user is asked to choose to load his work from Json file or in XML
	Undo	The user can undo his last performed action and that includes drawing and coloring any of the Shapes As any paint program ,the move, delete , resize actions can't be undone
	Redo	The user can redo his last performed action and that includes drawing and coloring any of the Shapes As any paint program ,the move, delete , resize actions aren't supported in the redo mode

- Shapes panel:



- Coloring:

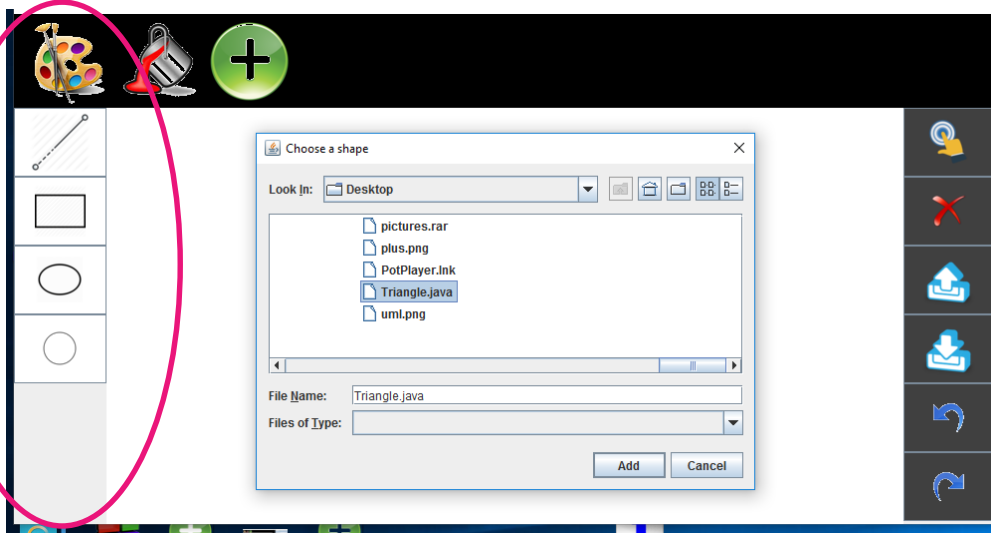
Color Chooser	filling
 <p>The user can choose the color for drawing and filling .By clicking on the above icon, a color chooser pane appears and the user select the required color and press ok.</p> <p>Then by default the stroke color and the fill color are set to the chosen color</p>	 <p>The selected shapes can be filled with any color. The user has to select the shape first ,choose the color and then press the icon and then the fill icon changes to</p>
	

- Loading an external Shape :

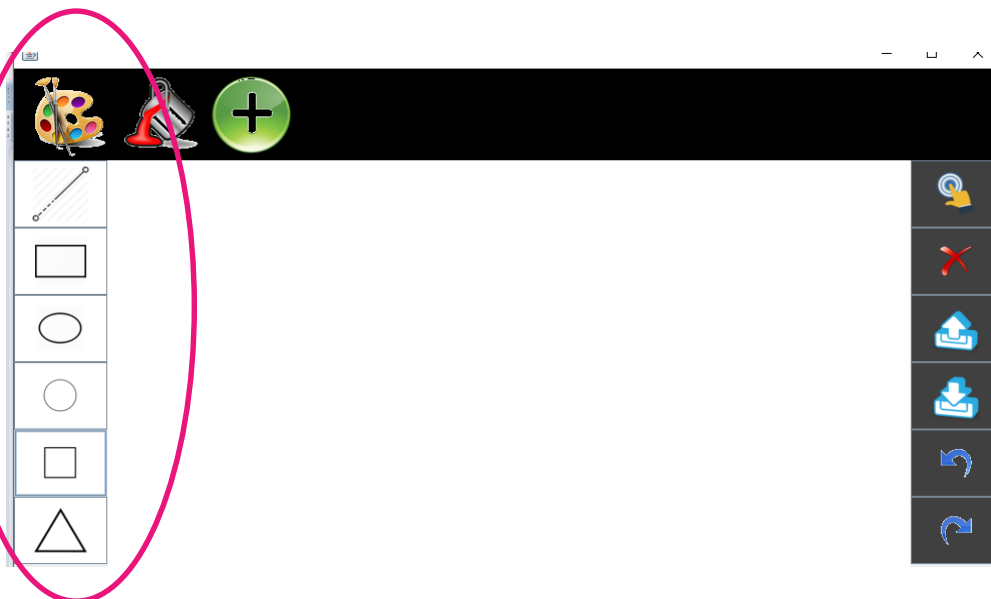


The user can load an external shape by clicking this icon which is known by the concept of dynamic class loading .An extension of the added shape is already appended to the list of shapes .

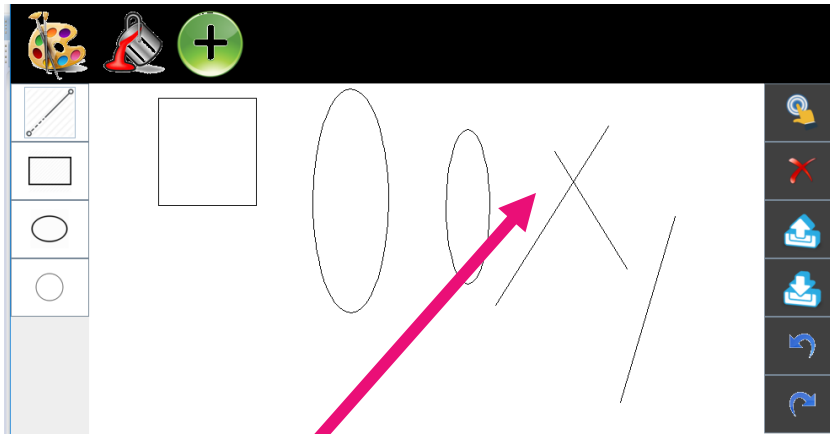
Before loading an external shape:



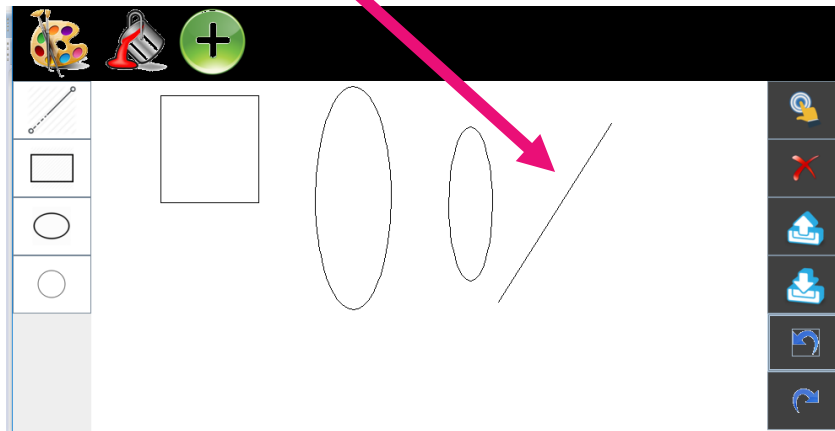
After loading the triangle and square:



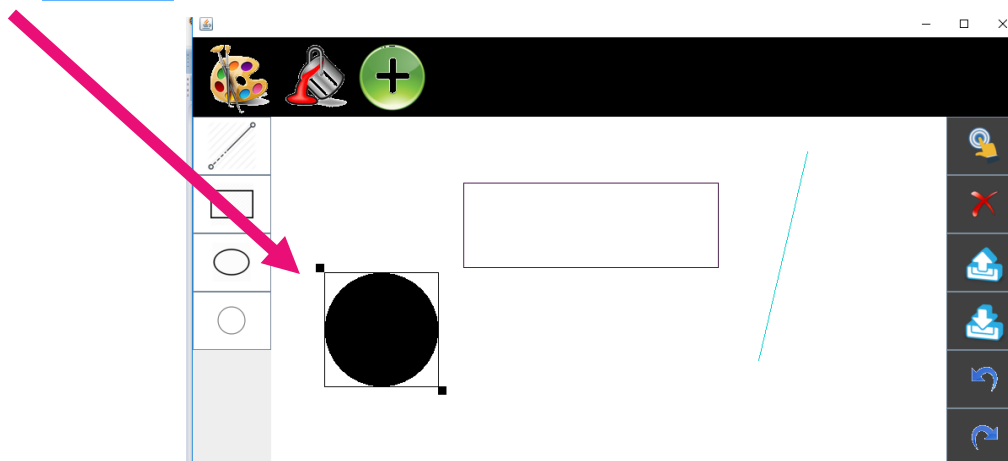
- Screenshots demonstrating some of the above features:



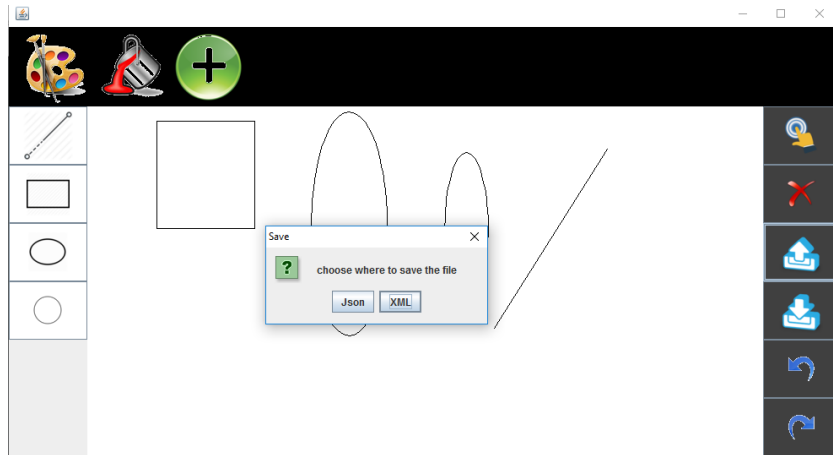
Undo twice example:



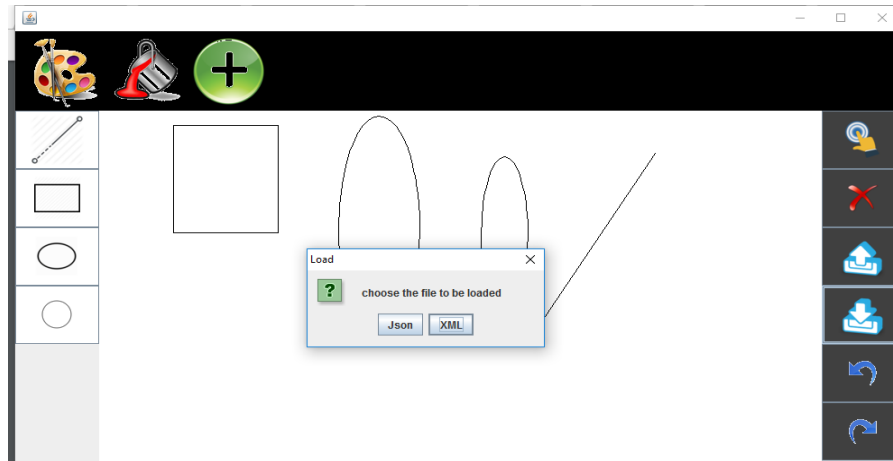
Selection:



Save:



Load:

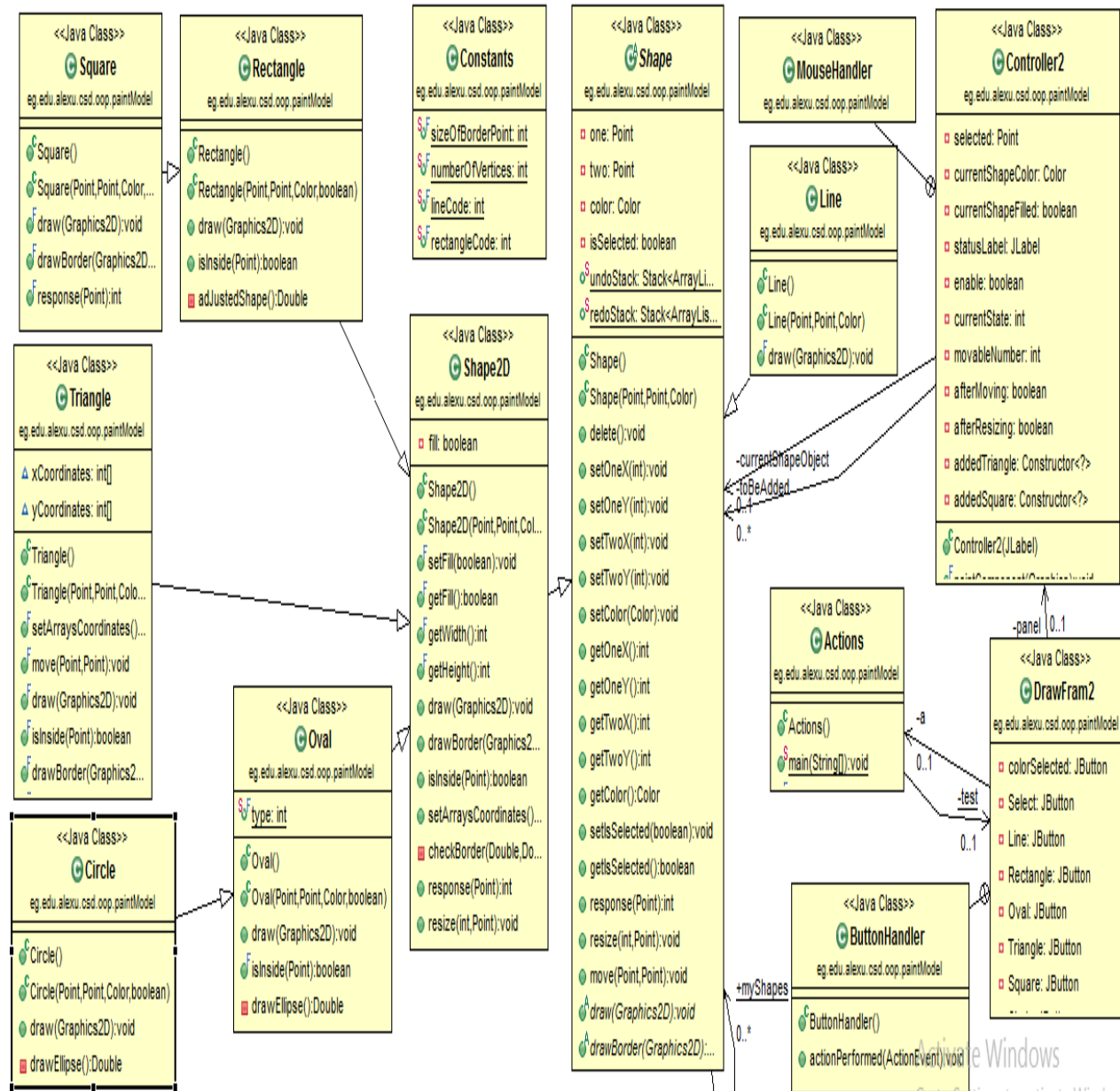


****The move and resize features are done by dragging the required shape through the drawing frame*

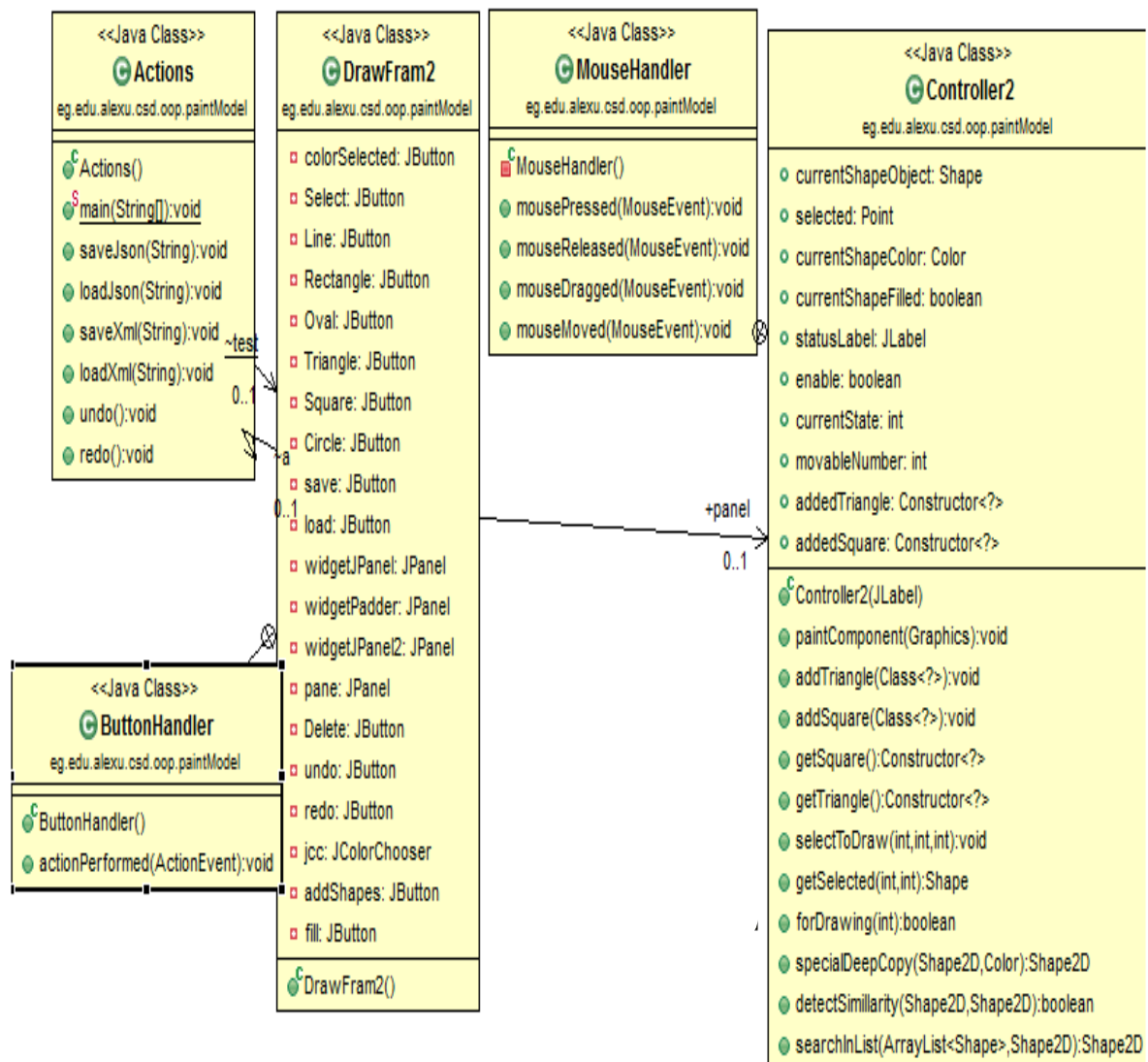
UML

The UML class diagram was created using OBJECT_AID plug-in in eclipse. Due to the number of methods and attributes in each class. The classes were minimized to be able to see the relations between classes. And for a detailed view for each class separately, the UML was divided in 2 Files

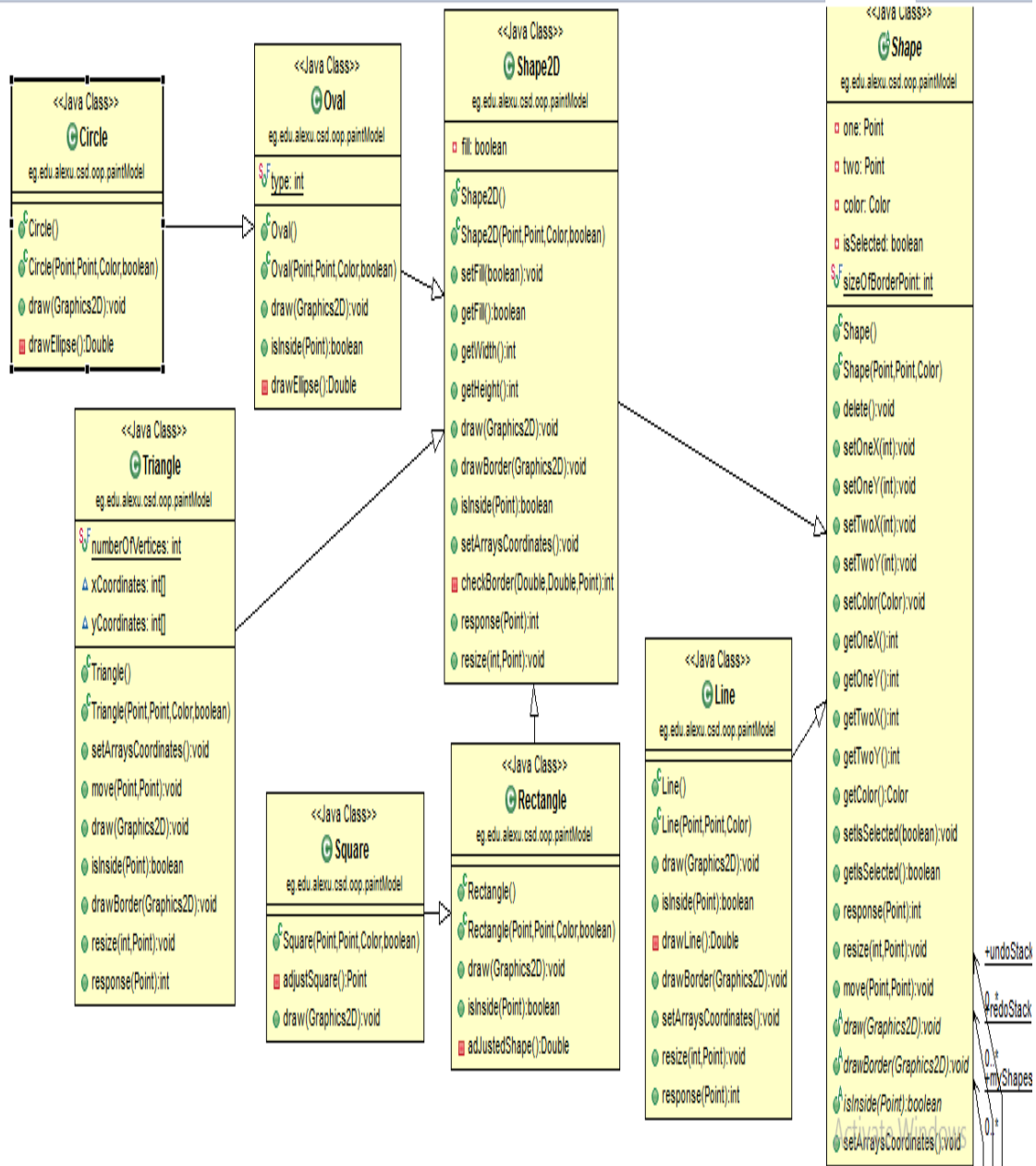
- Relations between all classes:



• View-Control relations:



- Shapes relations:



Technical details and coding

The program follows MVC design pattern, it is divided into 3 main modules: model, view and controller .Each can contain one or more sub modules of course.

The main abstract class Shape is extended by a group of concrete classes such as line, Shape2D which is also extended by (oval, circle, square, triangle, rectangle...)

The "DrawFrame" class contains the GUI, the buttons actions and the controller functions calls.

The Controller includes all the functions responsible for tasks and actions such as undo, redo, paintComponent ...when the mouse is moved or released

[Screenshots of the code:](#)

```
13 abstract class Shape {
14
15
16     private Point one,two;
17     private Color color;
18     private boolean isSelected;
19
20
21     public static final int sizeOfBorderPoint = 10;
22     public static ArrayList<Shape> myShapes = new ArrayList<Shape>();;
23     public static Stack<Shape> undoStack = new Stack<Shape>();
24     public static Stack<Shape> redoStack = new Stack<Shape>();
25
26
27
28     public Shape () {
29         one.x=0;
30         one.y=0;
31         two.x=0;
32         two.y=0;
33         color=Color.BLACK;
34         this.isSelected = false;
35     }
36
37
38     public Shape (Point one,Point two,Color color)
39     {
40         this.one=one;
41         this.two = two ;
42         this.color =color;
43         this.isSelected = false;
44     }
```

```

62
63 public void loadJson (String src) throws Exception {
64     Shape.myShapes.clear();
65     JSONParser parser = new JSONParser();
66     try{
67         JSONArray jsonArray = new JSONArray();
68         FileReader f =new FileReader(src);
69         Object obj = parser.parse(f);
70         JSONObject object = (JSONObject) obj;
71         jsonArray = (JSONArray)object.get("list");
72         long X11, Y11, X22, Y22;
73         long r,b,g;
74         Color stkCol;
75         String type;
76         boolean Fill=false;
77         for (int i = 0; i < jsonArray.size(); i++) {
78             JSONObject jsonObject;
79             jsonObject = (JSONObject)jsonArray.get(i);
80             type = (String) jsonObject.get("type");
81
82             Shape newObj=null;
83
84             X11=(long)jsonObject.get("X1");
85             Y11 = (long) jsonObject.get("Y1");
86             X22 = (long) jsonObject.get("X2");
87             Y22 = (long) jsonObject.get("Y2");
88             if (jsonObject.get("fillCol")!=null)
89                 Fill = (boolean) jsonObject.get("fillCol");
90
91             b= (long) jsonObject.get("StrokeCol1");
92             g= (long) jsonObject.get("StrokeCol2");
93             r= (long) jsonObject.get("StrokeCol3");

```

```

62 public void paintComponent(Graphics g) {
63     super.paintComponent(g);
64     Graphics2D g2 = (Graphics2D) g;
65
66     RenderingHints rh = new RenderingHints(RenderingHints.KEY_TEXT_ANTIALIASING,
67         RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
68     g2.setRenderingHints(rh);
69     System.out.println(Shape.myShapes.size());
70     for (int counter = Shape.myShapes.size() - 1; counter >= 0; counter--) {
71         Shape.myShapes.get(counter).draw(g2);
72         boolean check = Shape.myShapes.get(counter).getIsSelected();
73         if (check) {
74             Shape.myShapes.get(counter).drawBorder(g2);
75         }
76     }
77     if (currentShapeObject != null)
78         currentShapeObject.draw(g2);
79 }
80 public void addTriangle(Class<?> loadedClass) {
81     addedTriangle = loadedClass.getConstructors()[0];
82 }
83
84 public void addSquare(Class<?> loadedClass) {
85     addedSquare = loadedClass.getConstructors()[0];
86     System.out.println("da51 fl added");
87 }
88
89 public Constructor<?> getSquare() {
90     return addedSquare;
91 }

```

```

JLabel statusLabel = new JLabel("");

colorSelected=new JButton();
colorSelected.setBackground(Color.BLACK);
colorSelected.setIcon(new ImageIcon("pictures//palette.png"));
jcc =new JColorChooser();
panel = new Controller2(statusLabel);
pane= new JPanel();
fill = new JButton("");
fill.setBackground(Color.black);
fill.setIcon(new ImageIcon("pictures//fill.png"));
addShapes = new JButton();
addShapes.setBackground(Color.black);
addShapes.setIcon(new ImageIcon("pictures//plus.png"));
Select = new JButton();
Select.setBackground(Color.darkGray);
Select.setIcon(new ImageIcon("pictures//select.png"));
Line = new JButton();
Line.setBackground(Color.white);
Line.setIcon(new ImageIcon("pictures//line.png"));
Rectangle = new JButton();
Rectangle.setBackground(Color.white);
Rectangle.setIcon(new ImageIcon("pictures//rectangle.png"));
Circle = new JButton();
Circle.setBackground(Color.white);
Circle.setIcon(new ImageIcon("pictures//circle.png"));
Square = new JButton();
Square.setBackground(Color.white);

```

```

}
else if(event.getSource()==save)
{
    String[] options = {"Json", "XML"};
    int code = JOptionPane.showOptionDialog(
        panel, " choose where to save the file",
        "Save",
        JOptionPane.YES_NO_CANCEL_OPTION,
        JOptionPane.QUESTION_MESSAGE,
        null,
        options,
        options[1]);
    if (code==0)
    try {
        a.saveJson("jsonFile");
    } catch (FileNotFoundException | UnsupportedEncodingException e) {
        // TODO Auto-generated catch block
        ((Throwable) e).printStackTrace();
    }
    else a.saveXml("xmlFile");
}
else if (event.getSource()==load)
{
    String[] options = {"Json", "XML"};
    int code = JOptionPane.showOptionDialog(
        panel, " choose the file to be loaded",
        "Load",

```

```

@Override
public void setArraysCoordinates() {
    // TODO Auto-generated method stub

}

public void resize(int movablePoint, Point moved)
{
    if(movablePoint==1)
    {
        setOneX(moved.x);
        setOneY(moved.y);
    }
    else if(movablePoint == 2)
    {
        setTwoX(moved.x);
        setTwoY(moved.y);
    }
}

public int response(Point pressed)
{
    Rectangle2D.Double check = new Rectangle2D.Double(getOneX()-sizeOfBorderP
    if(check.contains(pressed.x,pressed.y)) return 1;
    check =new Rectangle2D.Double(getTwoX(),getTwoY(),sizeOfBorderPoint,sizeOf
    if(check.contains(pressed.x,pressed.y)) return 2;
    return -1;
}
}

```

-----END-----