­­

*Alexandria University Faculty of Engineering Computer and Systems Engineering Dept. CS221: Object Oriented Programming*

**Assignment 2**

**Team members:**

1. **Mohamed Ayman Farag (61)**
2. **Moamen ElBaroudy (51)**
3. **Introduction:**
   1. **Purpose of the Project:**

Design and Implement an OOP model that supports Vector Based drawing application where the user can draw using different geometric shapes (ex: Elliptical Shapes, Polygons …etc.). In the meantime these shapes should support various editing modes from resizing one shape to have different size constraints to moving it or changing its style properties whether it’s a stroke color, fill color or stroke thickness. It doesn’t end here but also the user can save the current drawing to be loaded at any time for later usage supporting two models, Both JSON and XML.

Of course the project is implemented in a dynamic way that supports adding new Shapes using dynamic loading feature in Java where upon following certain conventions set by the developers, you can create a new shape which will be supported fully by the Program.

* 1. **Overview of the document:**
     1. **Software Design :**

Illustrates the design of the project and the relation between the different classes of the project and because a diagram is worth a thousand of word. UML design is used to show the relation between multiple classes of the project. It is used to ease the understanding of the design and architecture of the project.

* + 1. **User Guide:**

Guides the user how to use the project how to use the various features of the application, and how to be familiar with it. It also illustrates the features of the application.

* + 1. **Design Decisions :**
       1. Illustrates the reason why we have chosen that design for project.
       2. Illustrates the benefits of the current design of the project.
       3. Difficult decisions while planning for the design of the project.

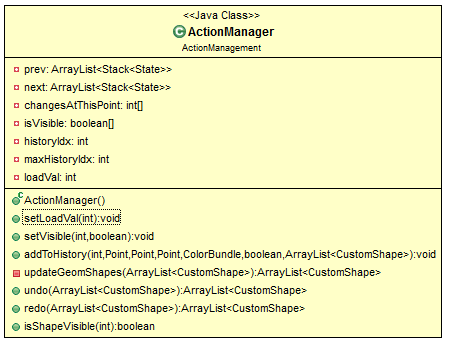
1. **Software Design:**

The project is divided into 7 main packages:

1. **actionManagement:**
   1. **ActionManger class:**

Responsible for handling both the redo and undo actions using two arrays of stacks for each shape where only the needed information are saved when a shape is modified the next state is saved to avoid duplication of saving information that isn’t needed.

One of the most important decision made was to separate the action management from the GUI and the drawing area to have its own class also to use the state class to lower memory usage in case of undo and redo from O(­n­2) to O(n), where ‘n’ is the number of actions done. The reduction of memory usage mightn’t be useful at the current state of this project but if need to add a free drawing mode it’ll be efficient using this way.

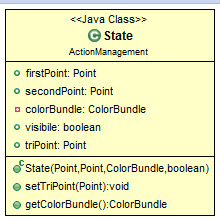


Also it handles the try to undo before the load operations.

* 1. **State class:**

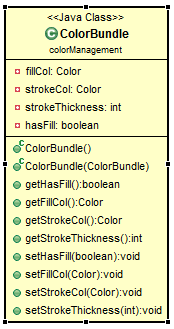
Responsible for describing the current state of an object containing all its attributes. Where it’s used in the action manager to handle undo and redo.

The State class was necessary to keep up with the modularity of the project.



1. **colorManagement:**
   1. **ColoBundle class:**

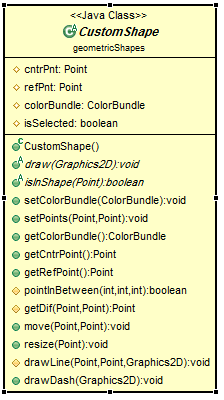
Responsible for saving the style attributes of a shape where it saves the fill color, stroke color and the stroke thickness.



For modularity of the project we used the ColorBundle to keep the different states of stroke color, fill color, and stroke thickness and avoid having multiple fields which would disrupt the readability of the code.

1. **geometricShapes:**
   1. **Shape class:**

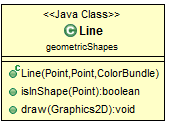
Responsible for representation of data of all shapes where all the other shapes extend it and it contains the implementation of the methods which behaves the same independent of the current shape.



We applied inheritance and polymorphism on the CustomShape class where the methods that are independent of the shape such as move, or resize are implement while methods as draw and inShape are left while some methods which are implemented in some shapes are overridden in others which extend them.

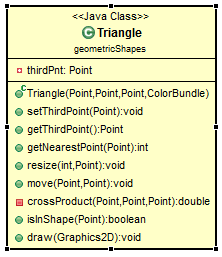
* 1. **Line class:**

Responsible for both the visual and data representation of the line supporting all the actions related to it.



* 1. **Triangle class:**

Responsible for the representation of the Triangle shape using three points to represent a triangle where the triangle can take any position.

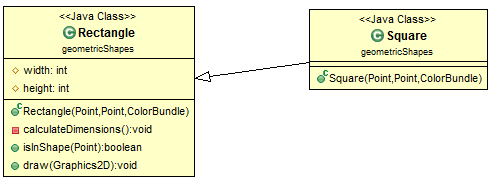


* 1. **Rectangle class:**

Responsible for the representation of a rectangle using 2 points the upper left and the lower right.

* 1. **Square class:**

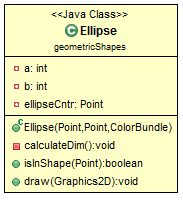
Since the Square is a special case of the rectangle so there’s no code in the file but only a constructor to differentiate the two shapes and the Square extends the Rectangle and its methods.



Square object is treated as Rectangle with equal length and width which is taken as an assumption as the maximum between both the length and the width.

* 1. **Ellipse class:**

Responsible for the representation of Ellipse using two points where the ellipse is contained with the Rectangle specified using those two points.



1. **graphiclInterface:**
   1. **DrawingArea class:**

Responsible for drawing over the screen and act as a connecting part between the GUI class and the ActionManager and LoadingClass.

* 1. **GUIPanel class:**

Responsible for displaying the GUI for the user and handling all the requests and input of the user by directing it to the other classes where it acts like the request manager. It uses two instance one of the UpperPanelBar and one of the LeftBar.

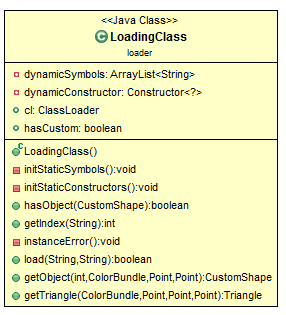
* + 1. **UpperPanelBar:**

Controls the buttons used to move, resize, delete or change the default colors.

* + 1. LeftPanelBar:

Controls the shape drawing buttons.

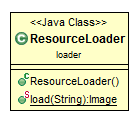
1. **Loader:**
   1. **LoadingClass class:**

Responsible for loading classes dynamically as a plugin. It also controls returning objects of different shapes by using their constructor. 

To ease the usage of different shapes and also in case of adding new shapes dynamically even if the number of added shapes isn’t limited by a certain parameter, we match each button with values ranging from zero to the current number of shapes that is loaded in the runtime and using array of constructors where the value of the shape refers to its constructor in the array which allows the program to check for duplication in dynamic loading and help the developer avoid hard coded methods to match buttons with constructors.

* 1. **ResourceLoader class:**

Responsible for converting icons to an input stream so that we can show these icons in the .jar file so that the graphical interface remains as it is in the .jar file.



1. **parser:** 
   1. **Parser class:**

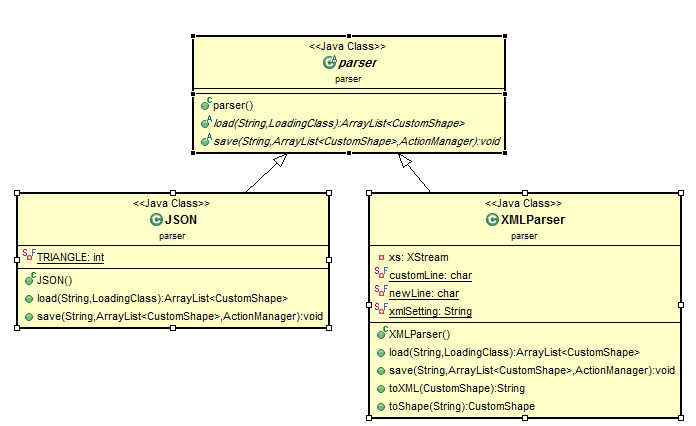
It’s an abstract class containing two methods which every parser should implement Load and save where the load method takes a string which is the path of the file to be loaded and returns an array List of the parsed data, and the save methods takes two arguments the path of the new file and the array List to be parsed into either JSON or XML.

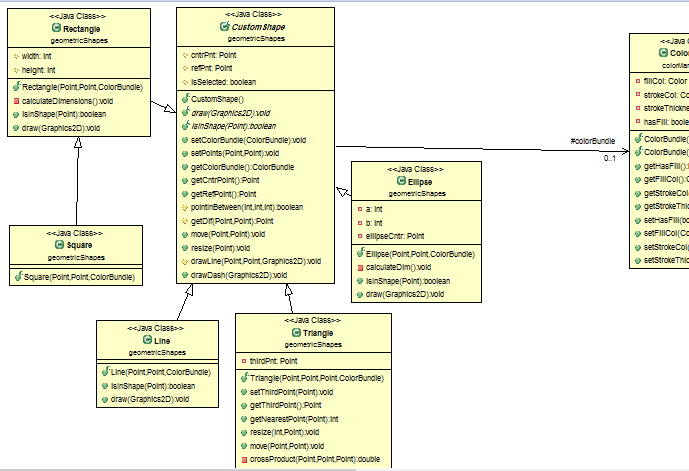
* 1. **XMLParser class:**

Implements the parser class where it can parse objects to XML or parse XML to objects.

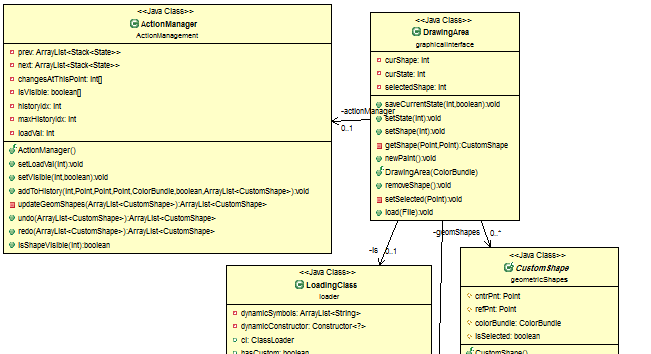
* 1. **JSONParser class:**

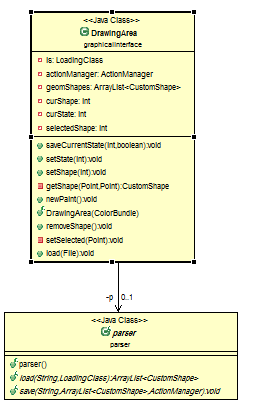
Implements the parser class where it can parse objects to XML or parse XML to objects.

**Full UML Diagram of Shapes relation.**



**Other important UMLs**

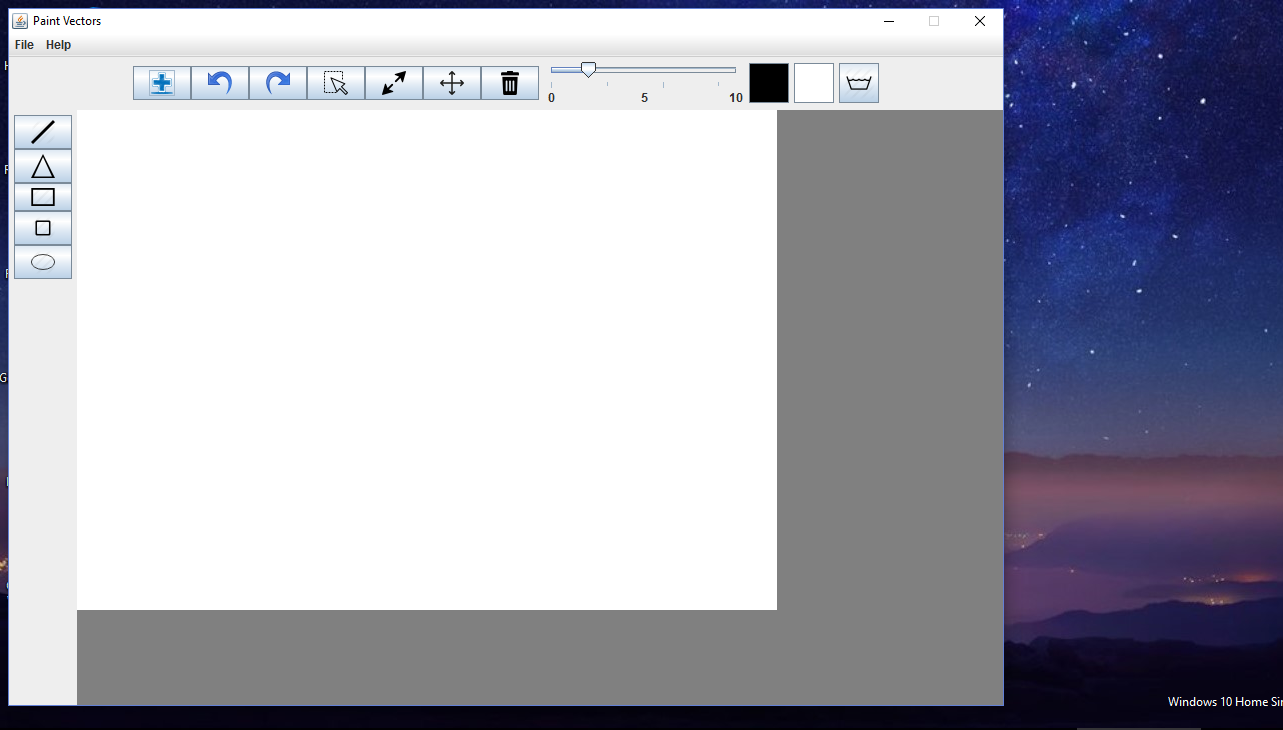




1. **User Guide:**

A resume to enable the user to be familiar with that application. Shows the user how to draw all the shapes, to edit the drawn shapes, to undo a performed action, to redo a performed action, to save a current drawing and to load a previously saved one.

Over view of the program:



This image shows the main design of the project where the buttons are divided into two section:

* 1. Buttons at the top of the application. And they are considered as controller buttons so the user can modify a currently selected shape , undo , redo , select a shape and load a custom plugin shape.
  2. Buttons at the left of the applications. And they are used to draw different shape.

**How to use the program:**

1. Drawing:
2. Rectangle : to draw a rectangle you should :

* Select the shape rectangle from the left.
* Click on the first needed point then drag till the other Point.

1. Square: to draw square you should :

* Select the shape square from the left.
* Click on the first needed point and drag to specify the side length of the square

1. Line: to draw a line you should :

* Select the shape line from the left.
* Click to on the first needed point and drag to the end point.

1. Triangle: to draw a triangle you should:

* Select the shape triangle from the left.
* Click to specify the first point then drag to specify the second one, finally you should release the click and move to specify the last point through a click.

1. Circle: to draw a circle you should:

* Select the shape circle from the left.
* Click to specify the center then drag to draw the circle.

1. Ellipse: to draw an ellipse you should:

* Select the shape ellipse from the left.
* Click and drag to draw the rectangle where the ellipse is subscribed.

1. **Undo:**

To undo you should select this icon.

1. **Redo:**

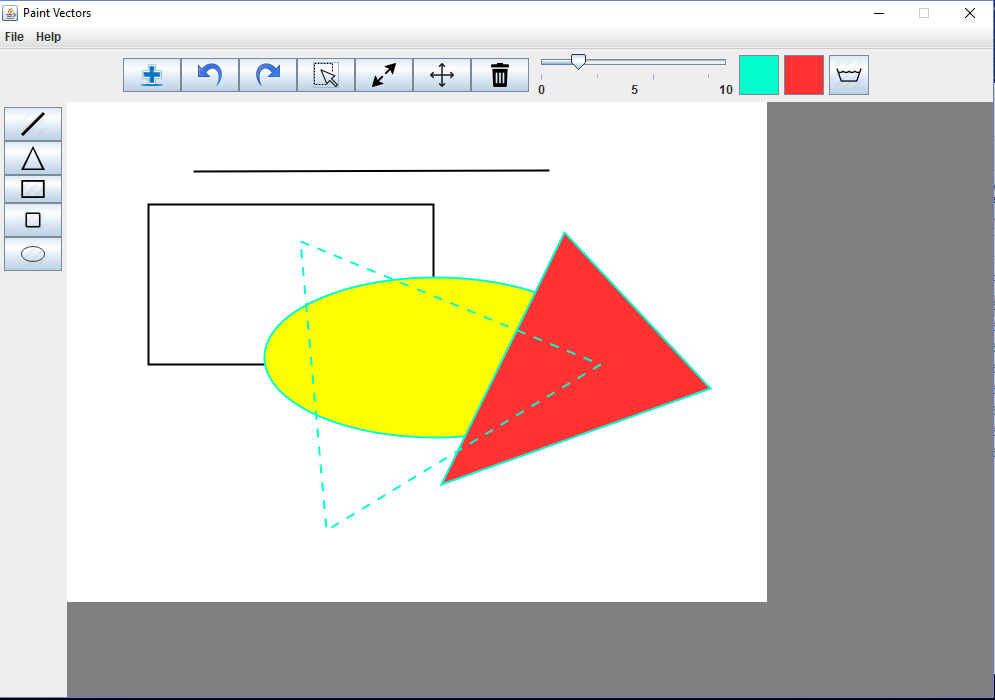
To redo you should select this icon.

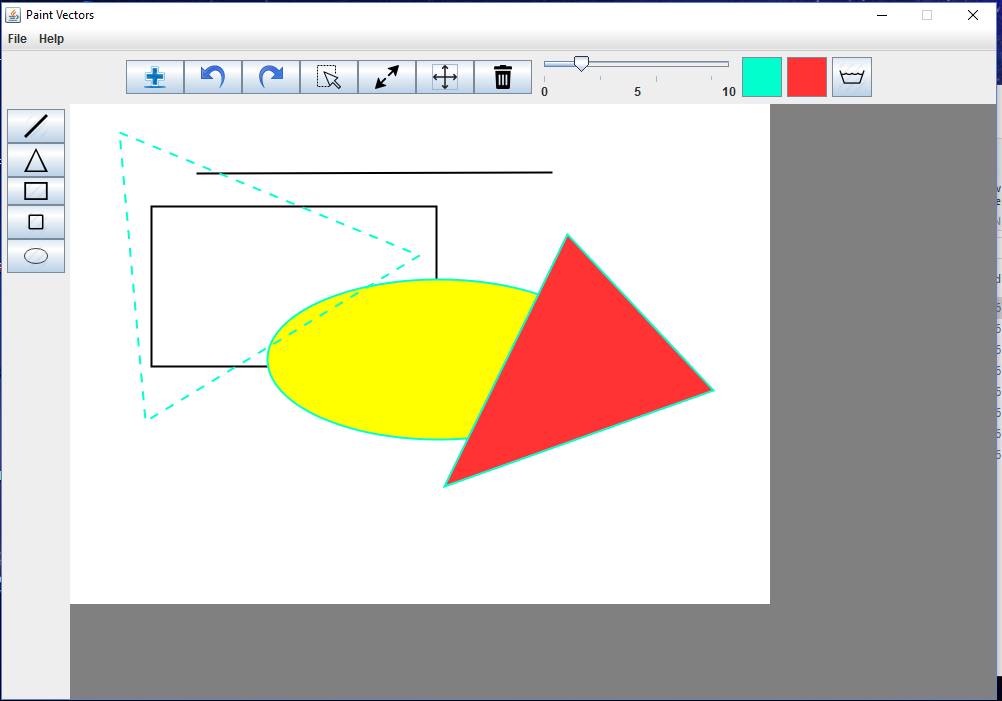
1. **Select:**

To select a shape you should select this icon.

1. **Move:**

To move a shape you should select it first then select this icon finally you can click and drag to move it.



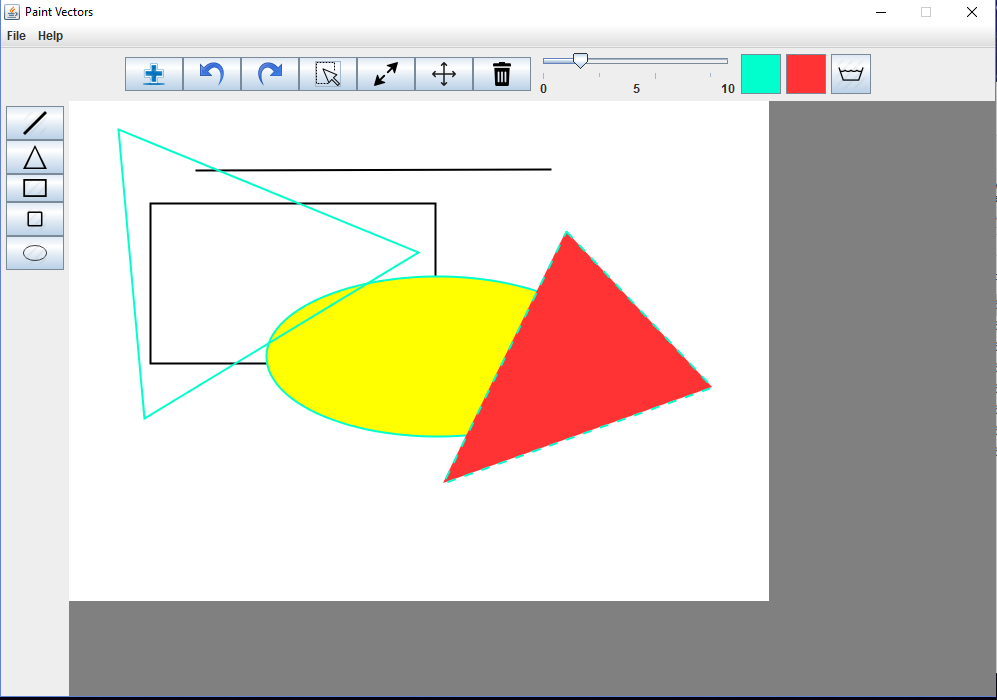
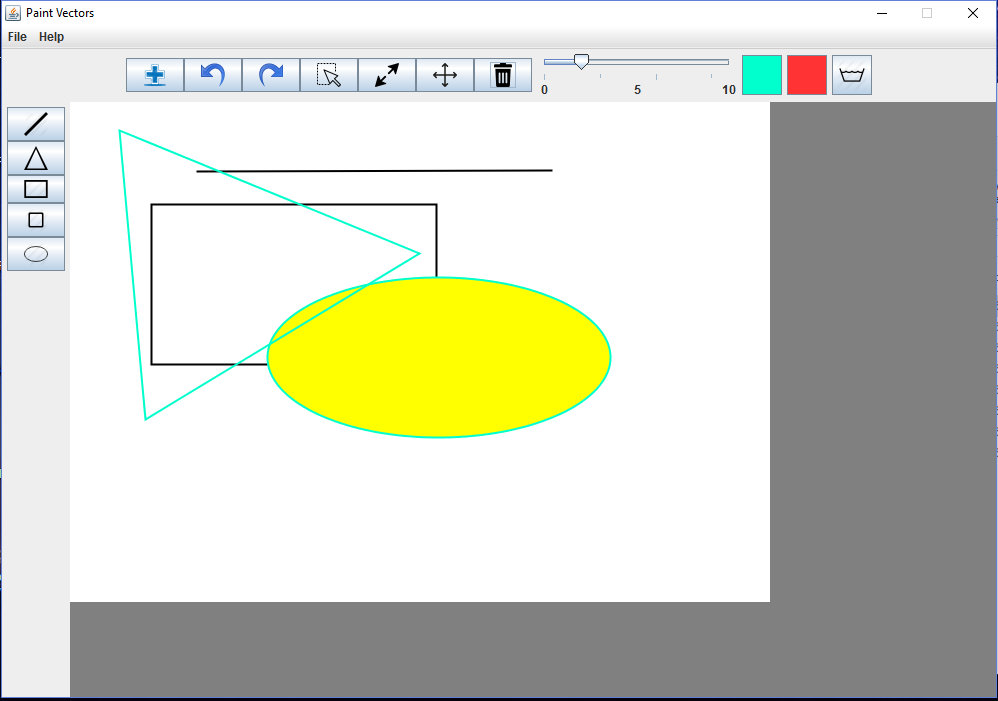


1. **Resize:**

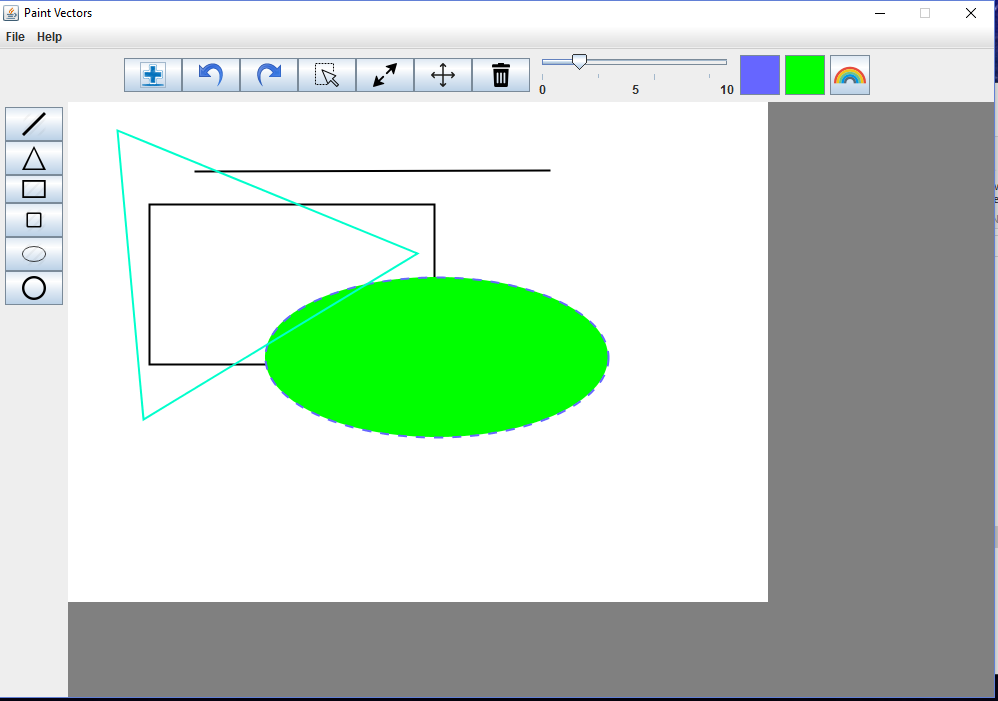
To resize you should select it first then select the icon finally you can click and drag to resize it.

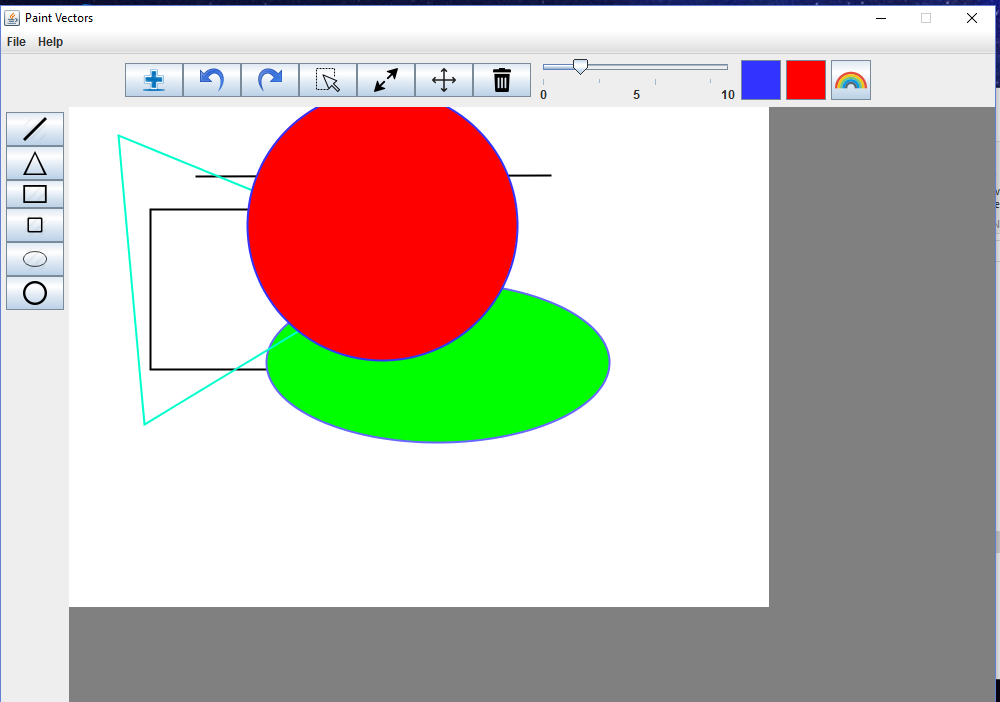
1. **Delete:**

To delete you should select the shape the select this icon.

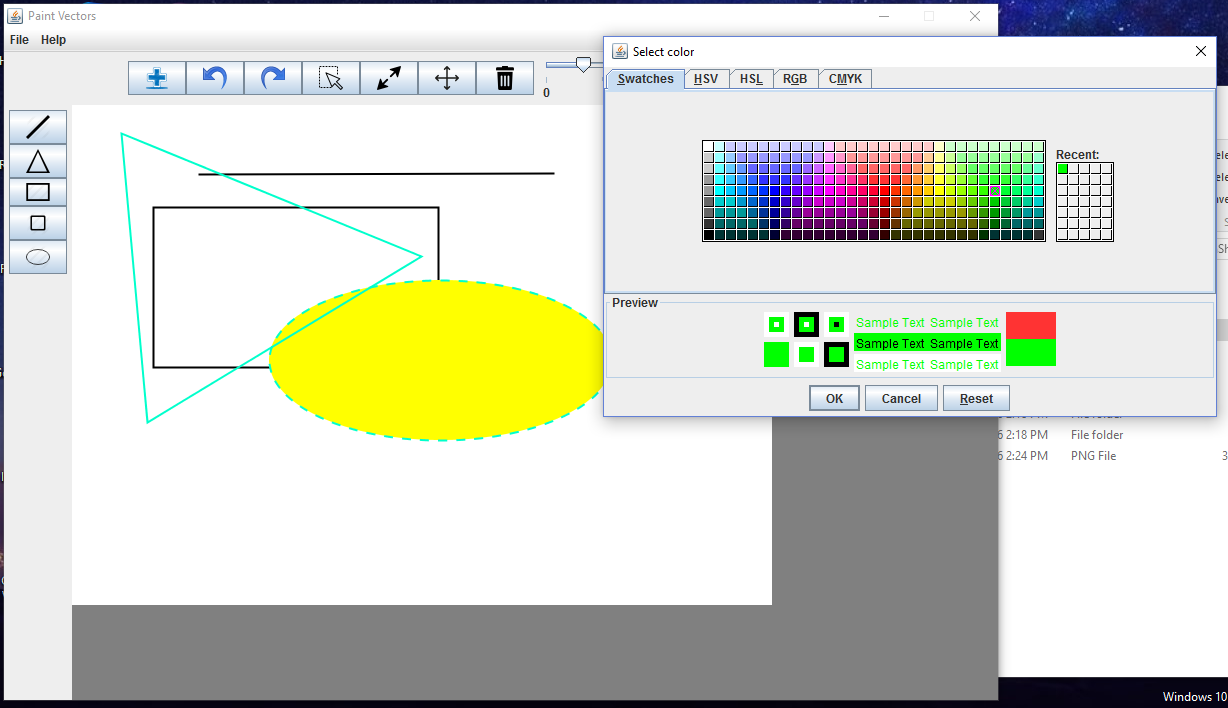
1. **Dynamic loading :**

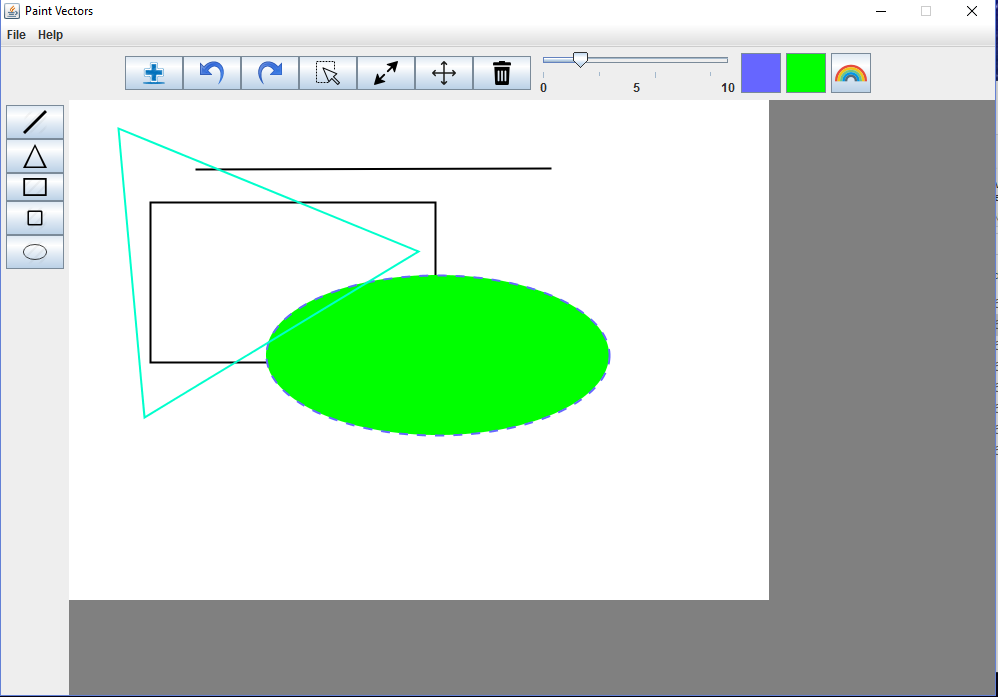
To load a plugin you should select the icon then choose the plugin. 



1. **Change color:**

To change the color should select the icons for colors at the top.





1. **Change stroke thickness:**

You should select the thickness through the slider

.

1. **Save and load:**

to save and load you should follow that image:

