Programming II

Paint Project

Description:

We Designed an object-oriented model for representing (drawing) geometric shapes that applies OOP concepts of inheritance and polymorphism using Design Patterns such as :

- Singleton
- MVC
- Factory
- Command
- Memento
- Strategy
- Dynamic Linkage
- State

The application represents shapes such as :

1. Circle
2. Ellipse
3. Square
4. Rectangle
5. Line
6. Rounded Rectangle
7. Triangle

And it is capable of drawing, colouring, moving, copying, resizing and deleting the shapes. At anytime you can undo/redo the last action.

You can save/open projects easily as well.

Any additional shape can be easily added.

Features:

We provide a paint project which can draw many shapes with dragging such as :

- Rectangle
- Circle
- Triangle
- Ellipse
- Square
- Rounded rectangle
- Line

And also the user can add any shape if he/she wants.

The user can select a shape or multiple shapes and make any of the following operations on it/them such as

-Resize

-Change the fill color

-Change the stroke color

-Change the stroke width

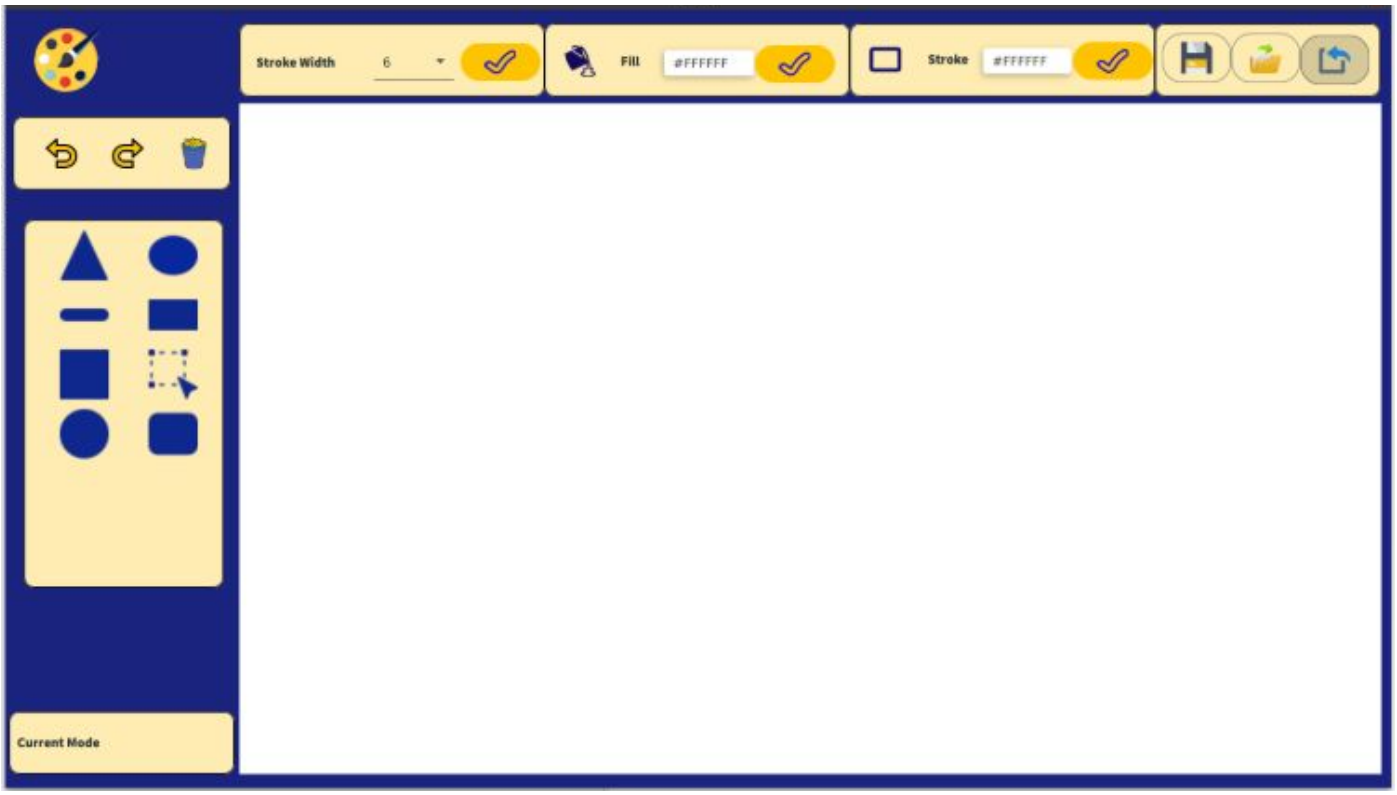
-Delete

-Move

The user can undo/redo any action performed

The user can also save the drawing on JSON files or XML files in the place he/she wants and can load it again.

Design Overview:



Drawable Shapes are represented on the left of the application.

At the bottom left corner there is a Current Mode Display (Which Displays Circle,Selector,etc).

Undo/Redo and Delete Buttons are at the top left of the screen

Stroke Width, Colour and Fill Colour are displayed on the screen top.

Save/Open and Import plugin buttons are at the top right corner.

Assumptions:

The undo action can only undo the last 20 actions only .

To load a saved drawing the user must know its location on the computer.

Team Work:

Ahmed Al-Ansary :

- The Saving/Loading (JSON files/XML files)
- The Selector - Resizable Rectangle.
- The properties of all Shapes.
- GUI of the project.
- Animation Manager.
- The Dynamic Linkage.
- Connecting to real time database using Firebase database.

Abdallah Mohamed:

- Triangle shape.
- Ellipse shape.
- Round rectangle shape.
- Class diagram and Sequence diagram.

Mohammed Badawi:

- Circle shape.
- Square shape.
- Rectangle shape
- File Chooser to load and save.
- Report.

Ahmed El-Shoubashy:

- Undo/Redo.
- Line shape.
- Use Case Diagram.
- File Chooser to import plugins.

Data Structures used in the project :

- ArrayList (to hold the classes of the supported shapes that can be drawn- to hold the shapes in the canvas-to hold the memento objects for the undo/redo)
- Map (to hold the properties of shapes).

Description of Important functions/classes:

ResizableRectangle Class:

A rectangle with 9 small circles (for Control) on the corners and midpoints of the rectangles' sides. The small circles are binded to the rectangle, each of which has a listener that changes the shape of the whole rectangle.

When any shape is selected, this resizable rectangle appears and the selected shape is binded to it.

The shape is binded to the resizable rectangle using mathematical equations that depend on the shape itself, in the function that is implemented in the shapes "**bindToResizableRectangle**".

PluginManager Class:

Opens the directory where the plugins are, loads each jar file and uses its manifest (Json format) to obtain required data about the shape using **pluginManifest** class, the data is then sent to **CanvasController** Class in order to display the data on the GUI.

bindToResizableRectangle [Rectangle ,Triangle ,Ellipse,....] : to Select The Shape and surround it with resizable rectangle and bind the dimension of the shape with the dimension of resizable rectangle to resize any shape easily.

Draw : Adds shape to canvas children.

Clone : to clone the properties of the shape and use them in undo/redo methods.

Get JSON String : to save the shape in json file through the creation of json object and putting the properties of the shape into it and returning a string of this object to save through the method **save** in **saveJSON** class in package **paint.model**.

loadJSON : take json object of the shape properties and get these properties through their keys to load them on the canvas.

getXMLNode : to save the shape in xml file through create element for each property of the shape and when create element for each property ,append it to element that represent the all properties of shape and return this element to save through the method **save** in **saveXML** class in package **paint.model**.

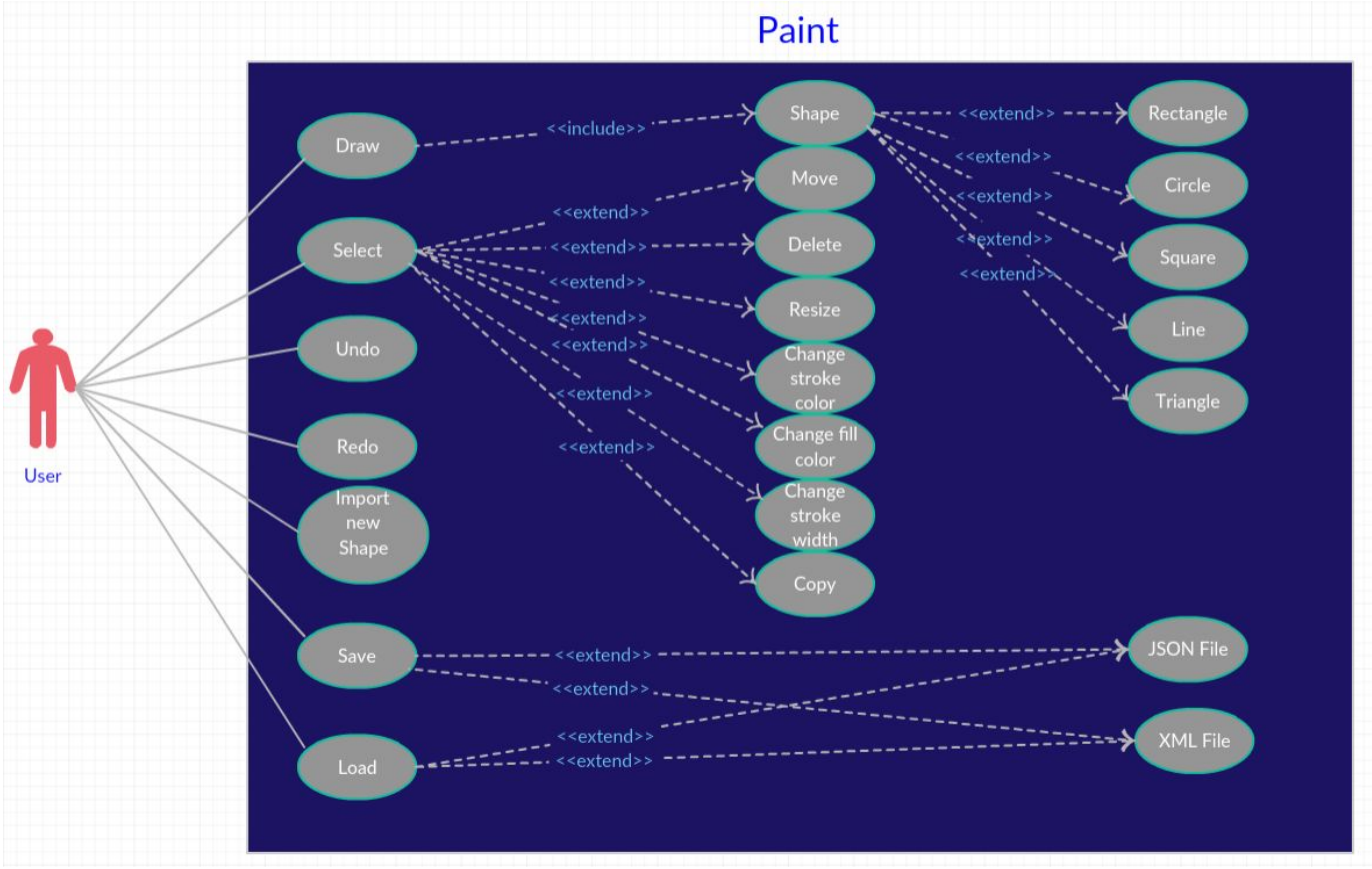
loadXML : take element of the shape properties and get these properties through their keys to load them on the canvas.

Undo: Save the current shapes on the screen in the **Caretaker** if they are not saved before and then remove all the shapes on the canvas and draw the shapes before the last change which the user has done , the undo can get the last 20 actions only.

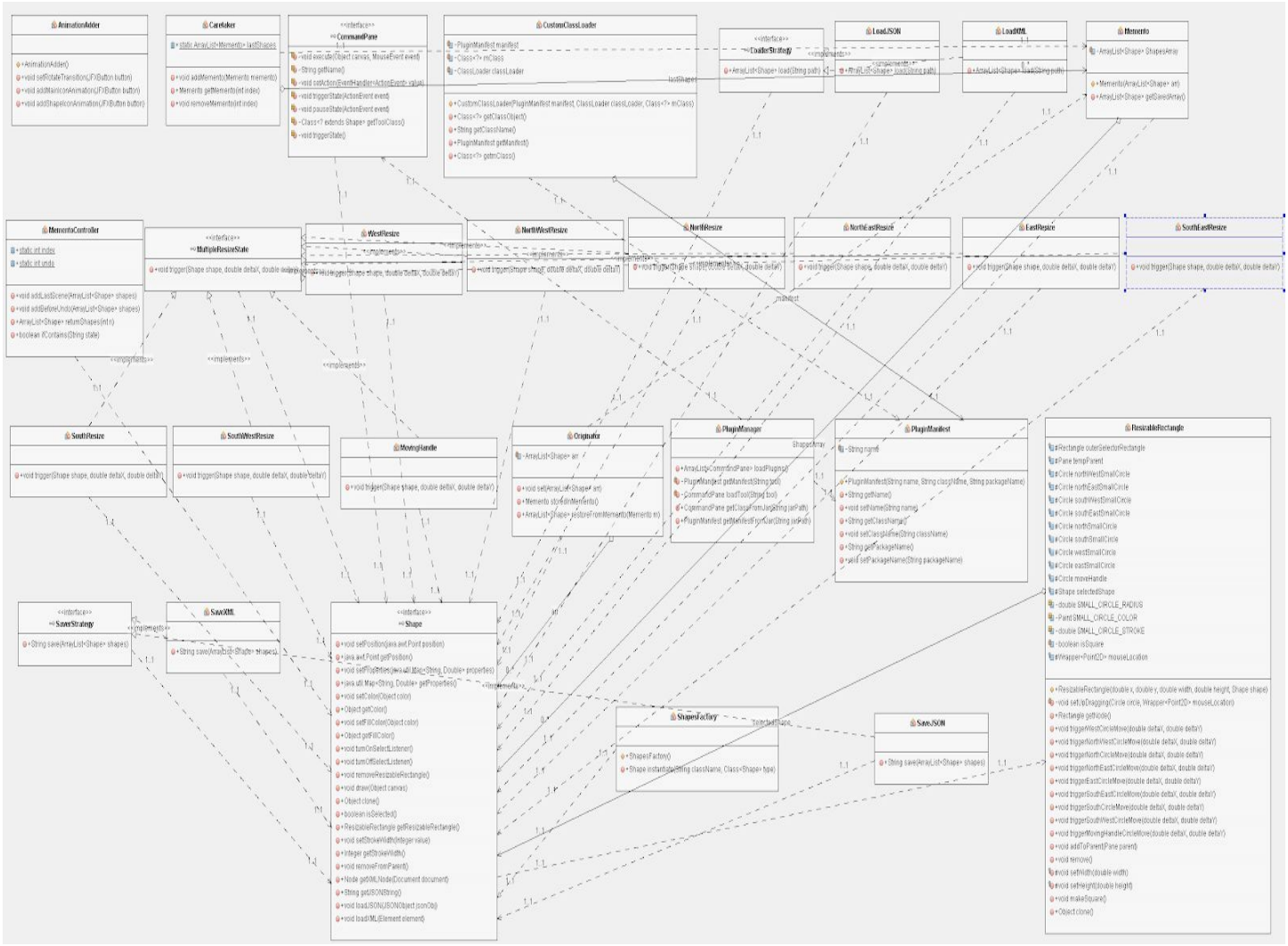
Redo: Remove all the shapes from the canvas and then get all the shapes before the user pressed the undo button from the **Caretaker** class .

UML Diagrams:

*Use Case:

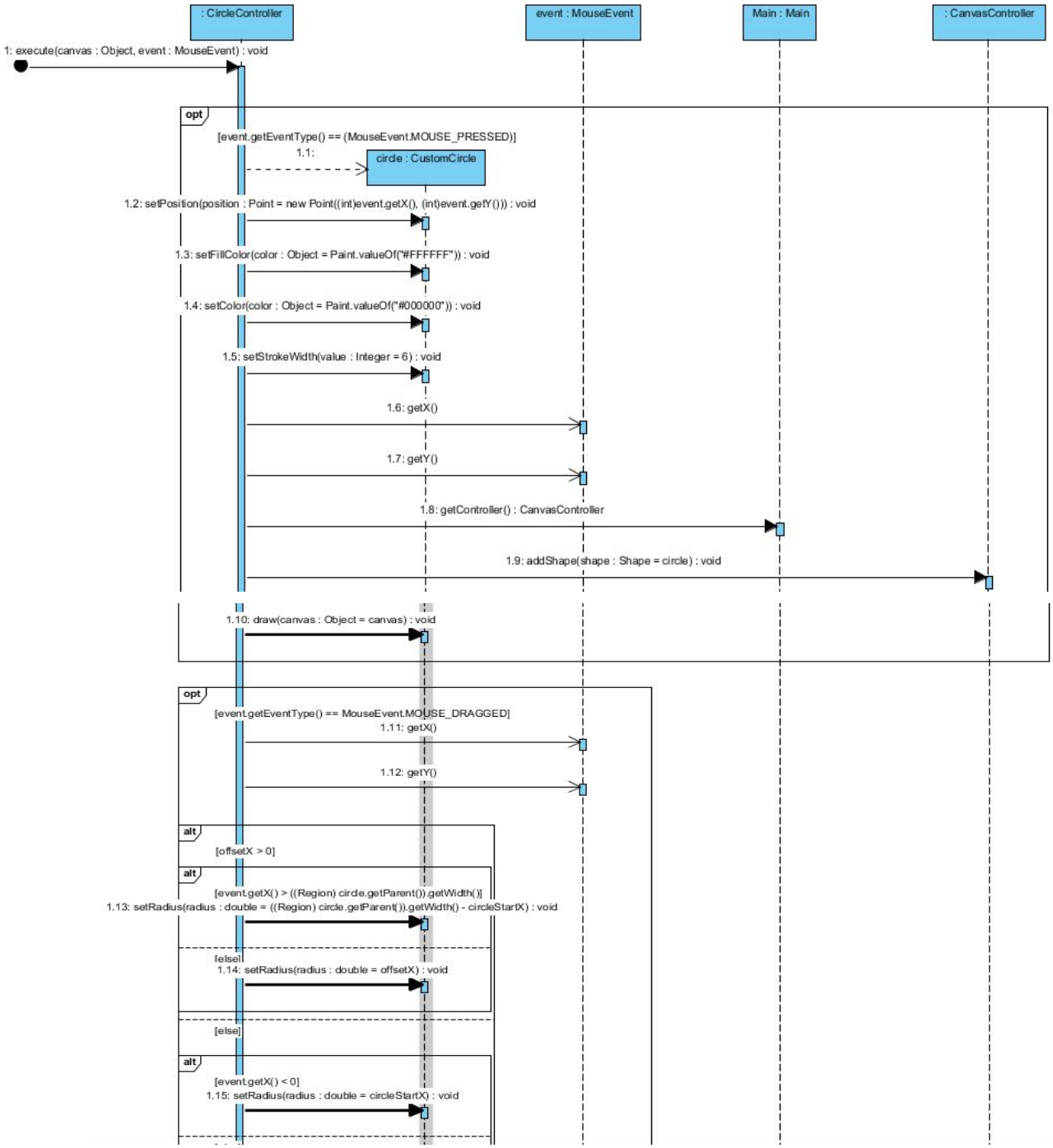


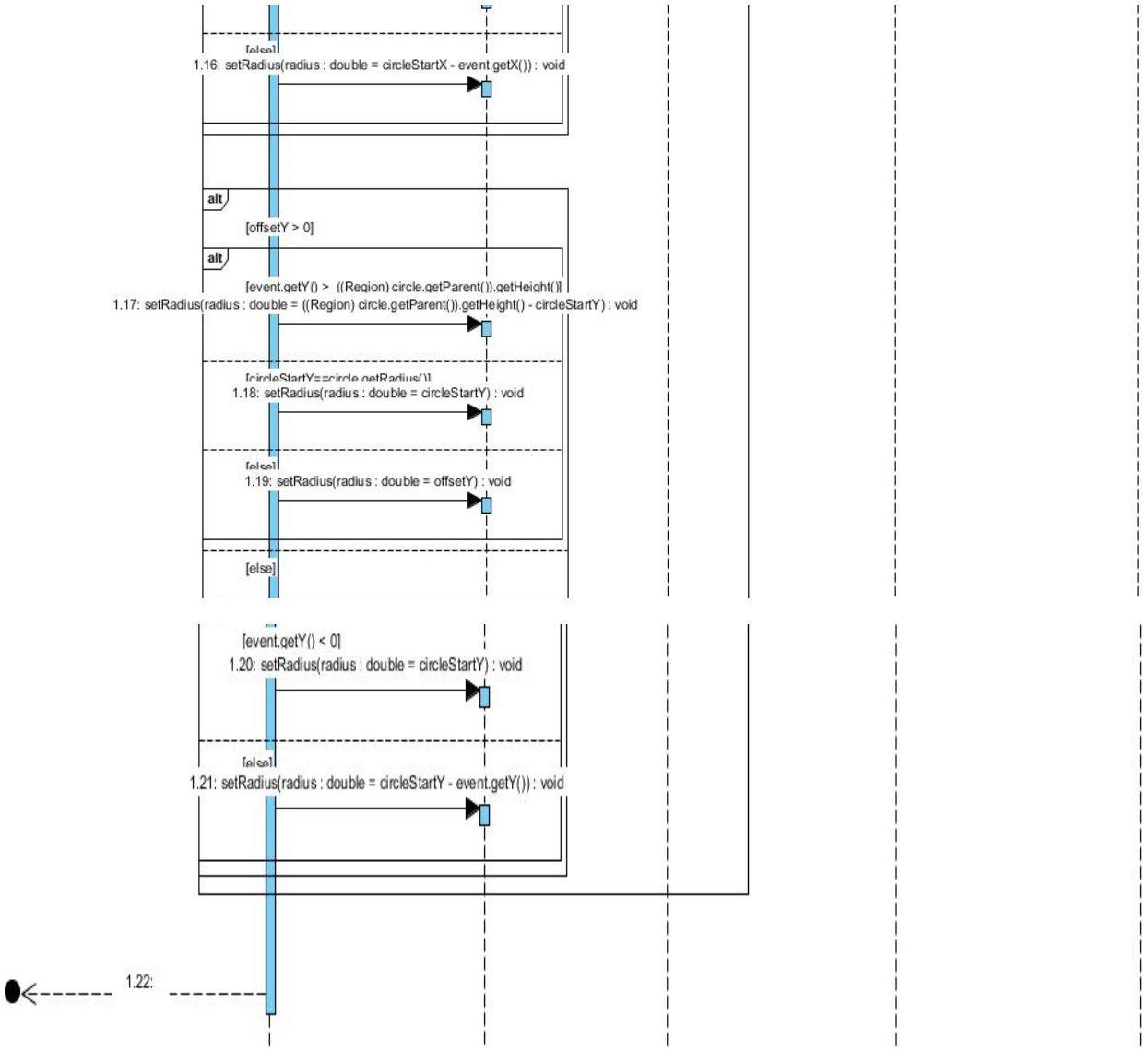
*Class Diagram:



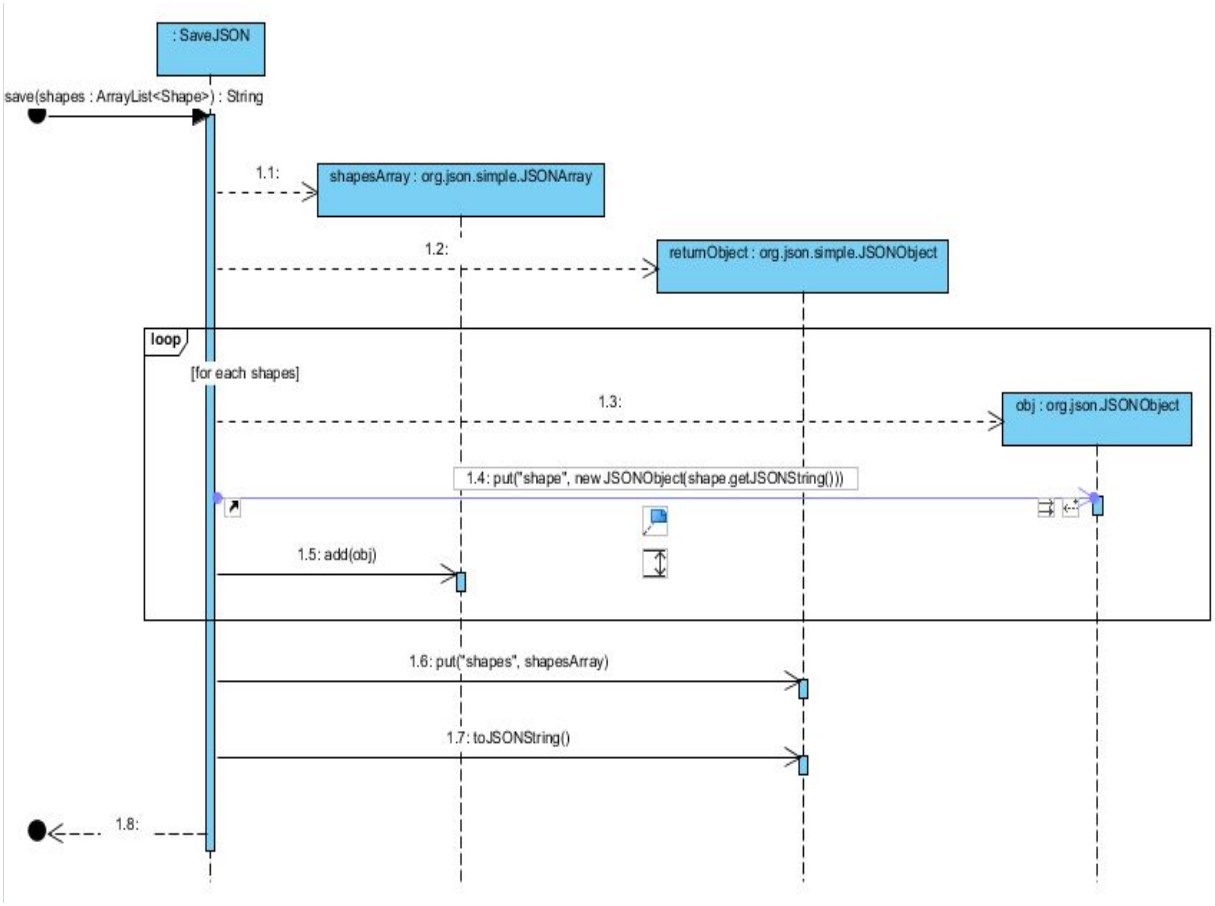
*Sequence Diagram:

Drawing Circle:

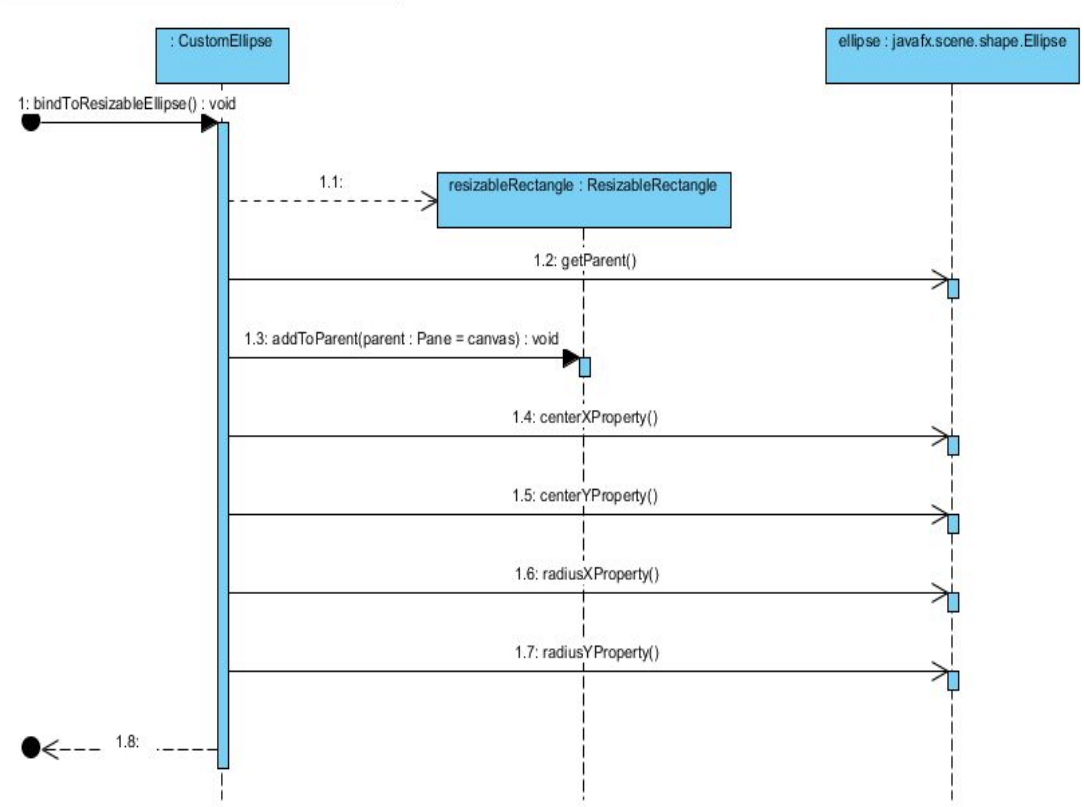



[else]
1.16: setRadius(radius : double = circleStartX - event.getX()) : void
alt
[offsetY > 0]
alt
[event.getY() > ((Region) circle.getParent()).getHeight()]
1.17: setRadius(radius : double = ((Region) circle.getParent()).getHeight() - circleStartY) : void
[circleStartY==circle.getRadius()]
1.18: setRadius(radius : double = circleStartY) : void
[else]
1.19: setRadius(radius : double = offsetY) : void
[else]
[event.getY() < 0]
1.20: setRadius(radius : double = circleStartY) : void
[else]
1.21: setRadius(radius : double = circleStartY - event.getY()) : void
1.22:

Save in Json:

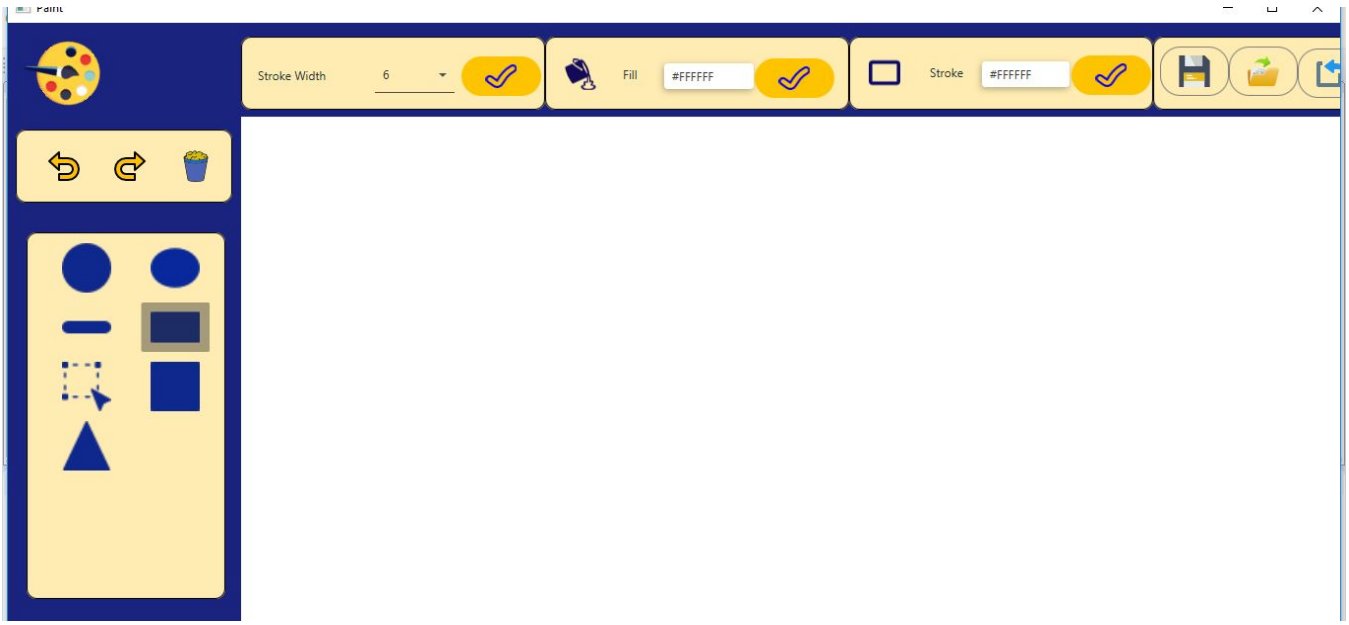

: SaveJSON
save(shapes : ArrayList<Shape>) : String
1.1:
shapesArray : org.json.simple.JSONArray
1.2:
returnObject : org.json.simple.JSONObject
loop
[for each shapes]
1.3:
obj : org.json.JSONObject
1.4: put("shape", new JSONObject(shape.getJSONString()))
1.5: add(obj)
1.6: put("shapes", shapesArray)
1.7: toJSONString()
1.8:

Editing Ellipse:



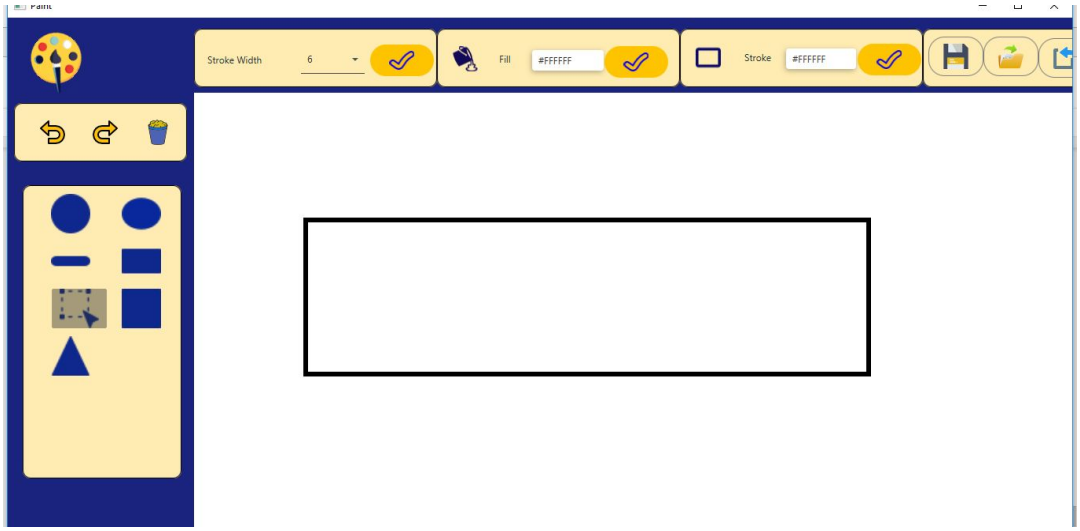
User Manual:

1) To draw any shape, you should just click on its icon

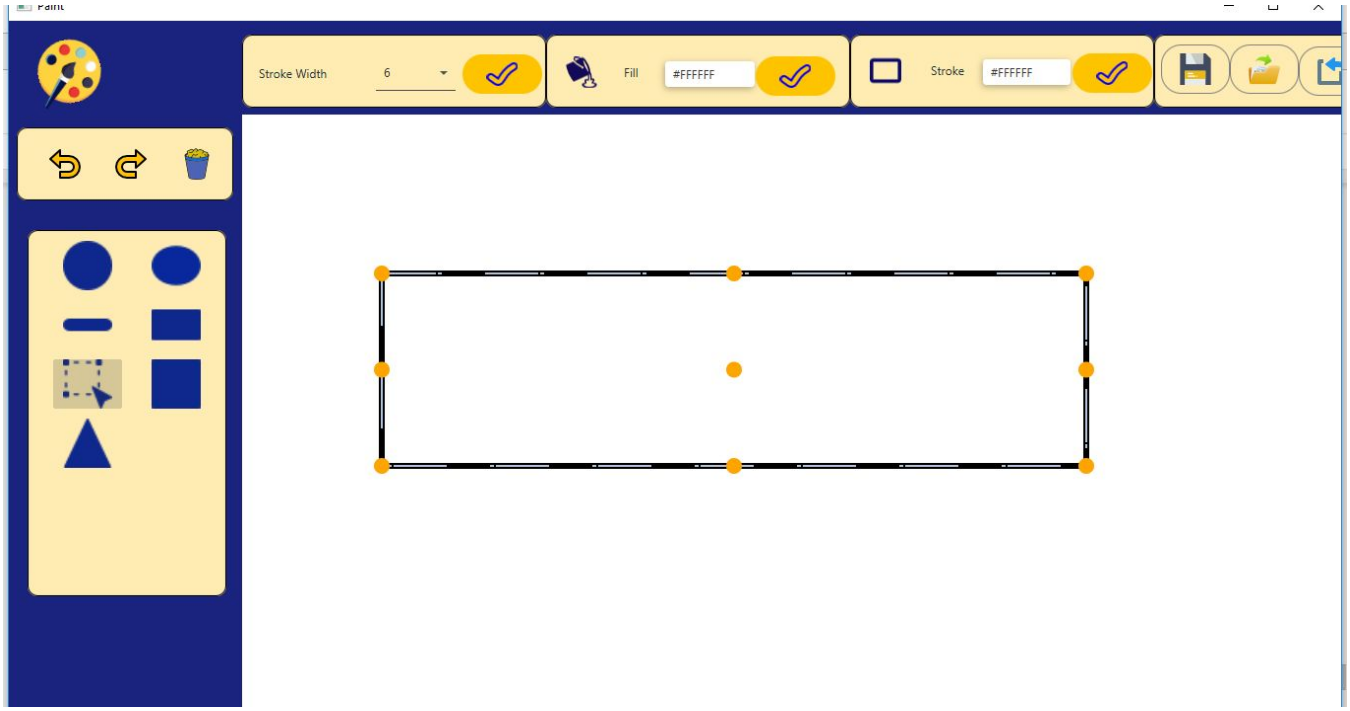


2) To edit any shape :

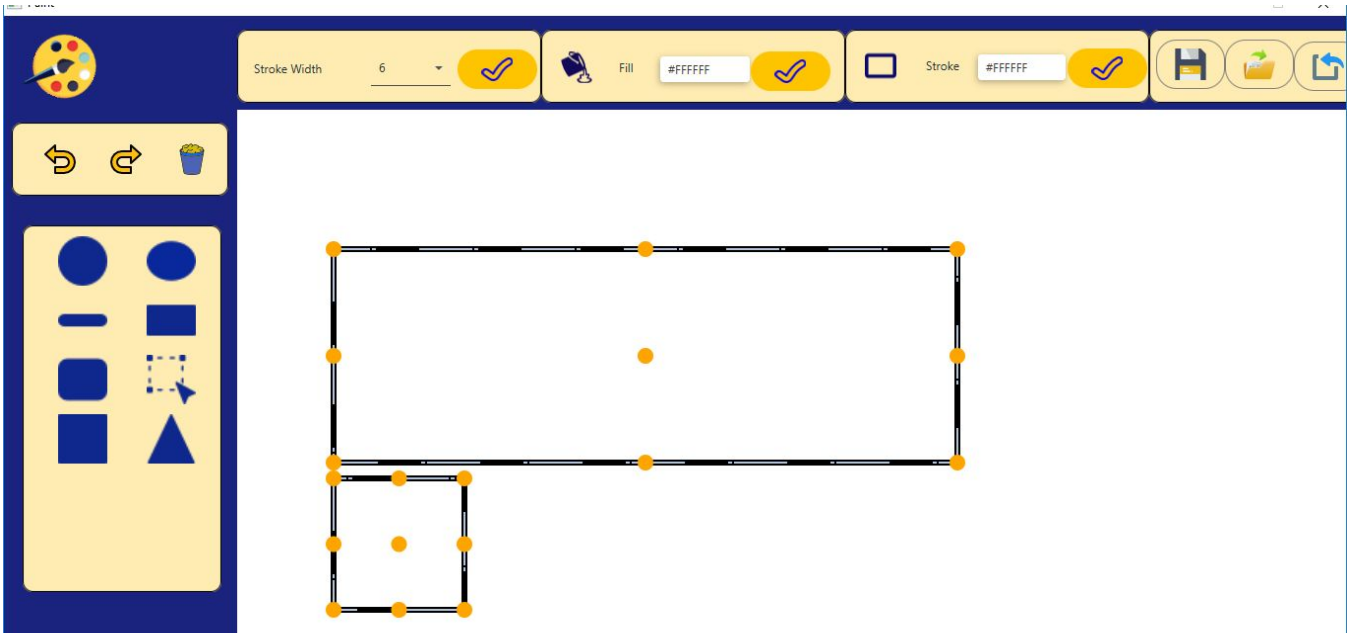
a) Choose the selector:



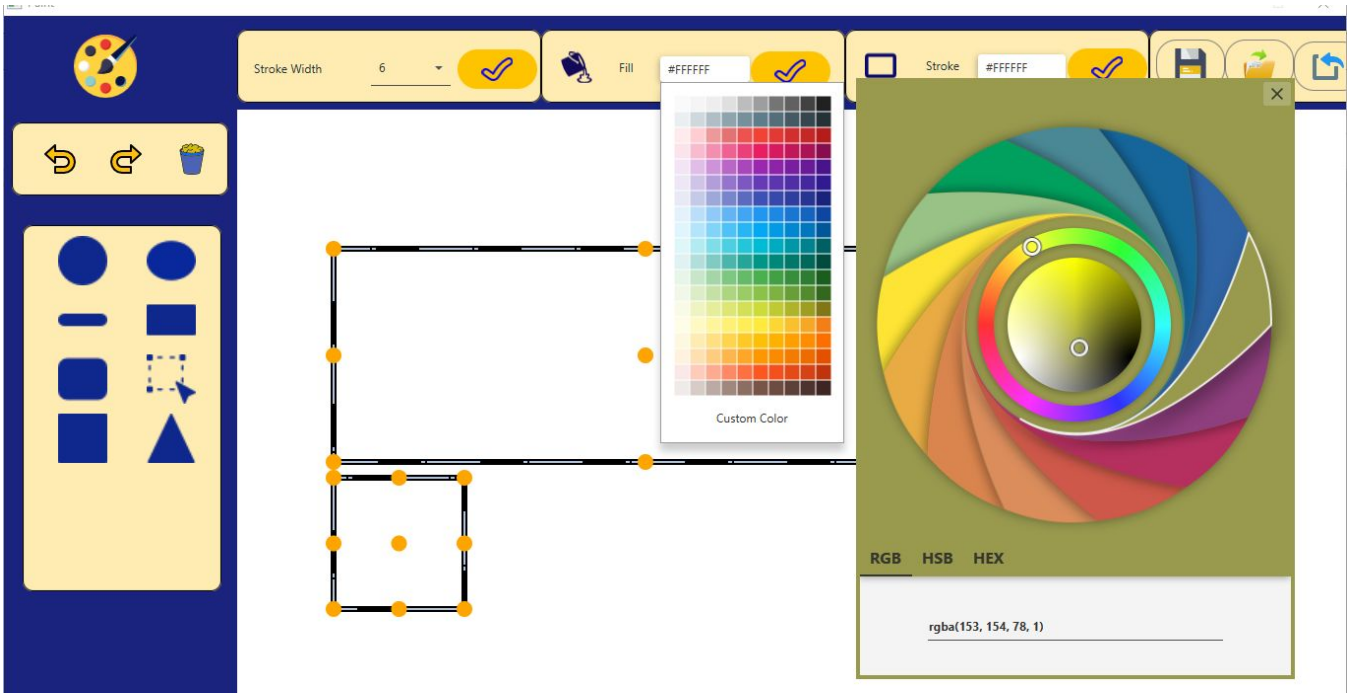
b) Then click on the shape you want to edit



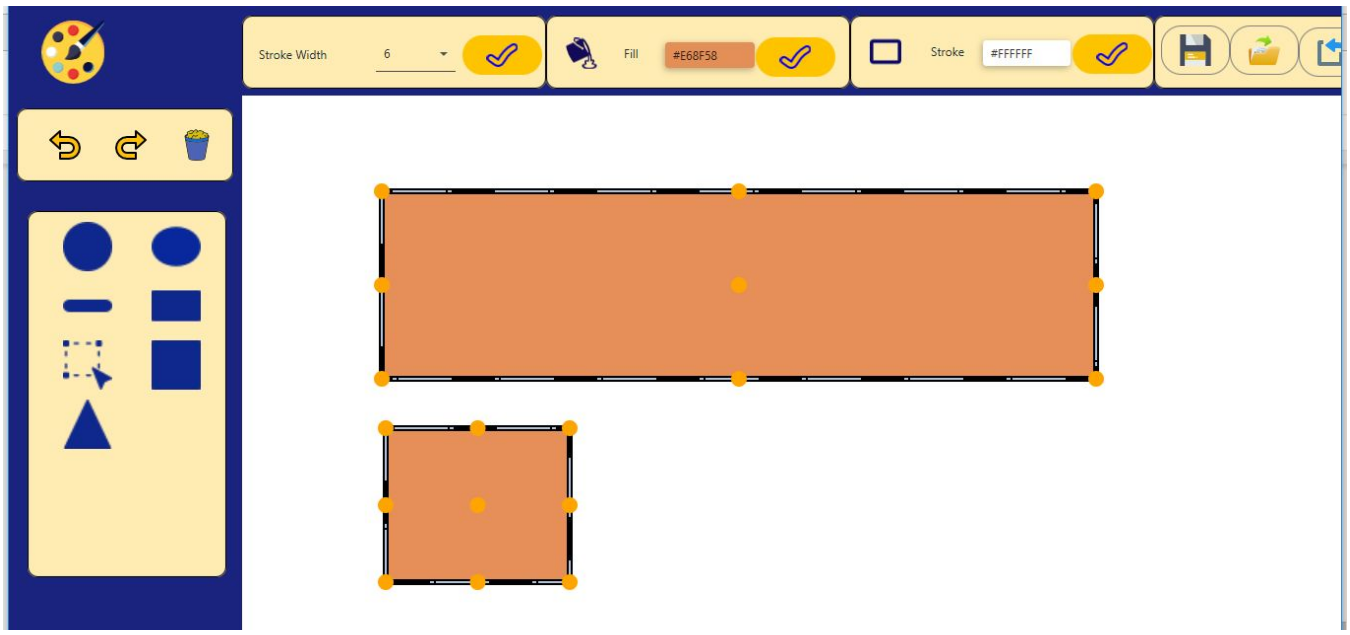
(You can select multiple shapes too)



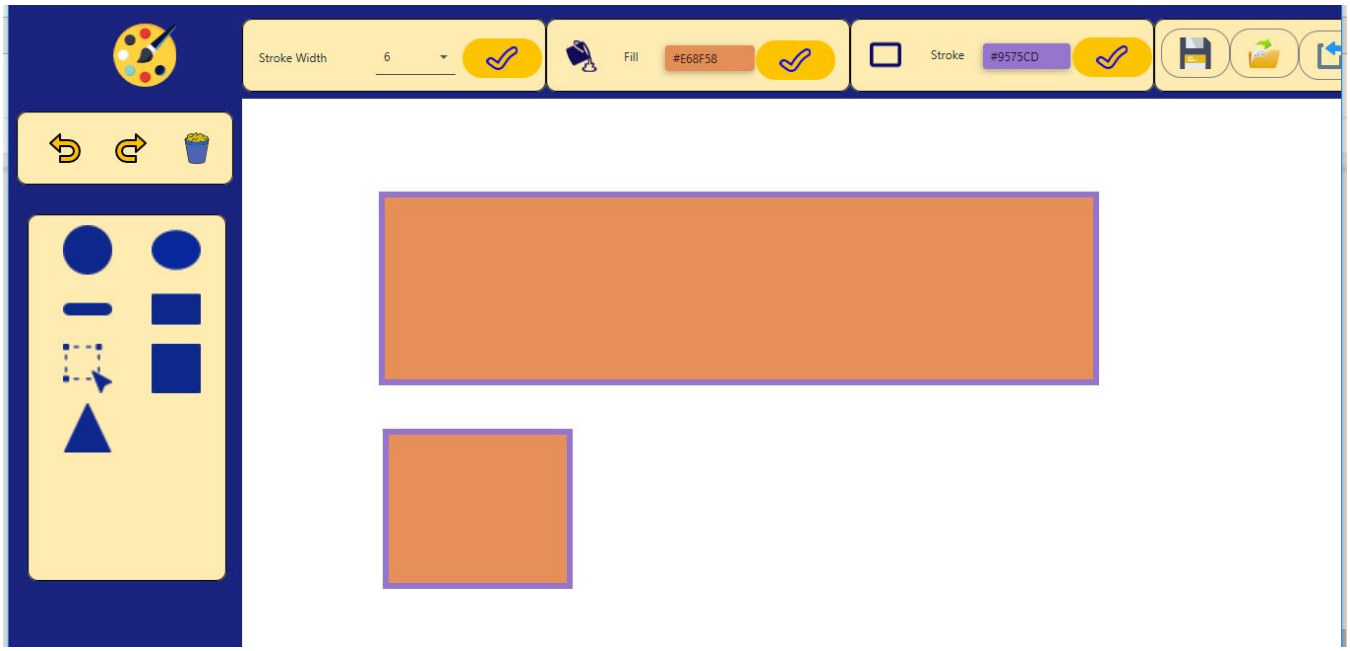
3)To change Stroke colour or Shape fill colour, Choose the preferred colour and click the tick button.



Then click the tick button!



Stroke colour can be changed in the same way

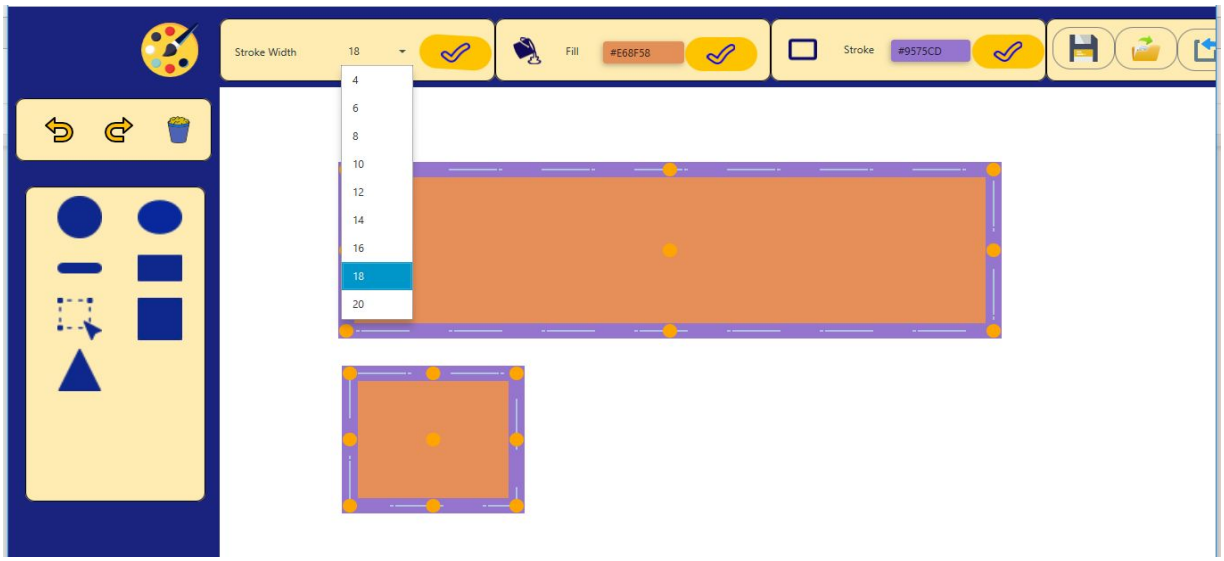


4)Stroke Width

a) Select Shapes as shown above

b) Click on stroke width combo Box and choose desired size

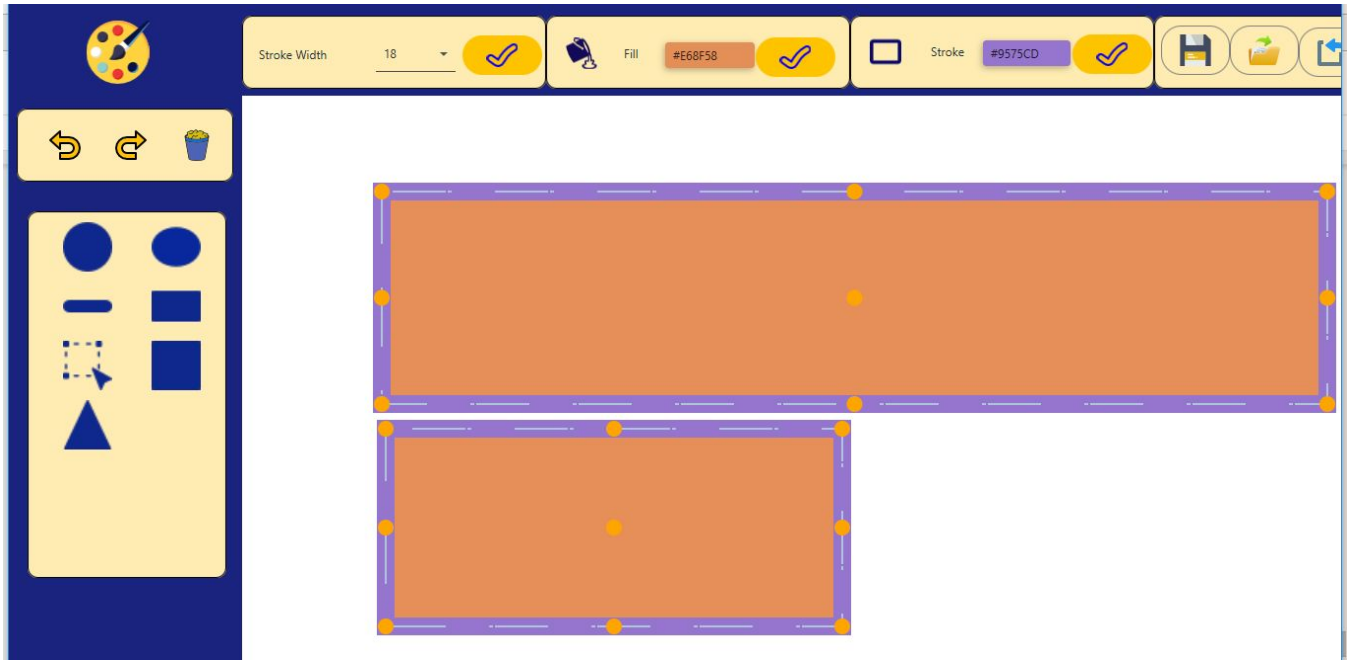
c) Click on the tick button!



4) Resize shapes (This will work on multiple shapes as well)

a) Select Shapes to resize as described above

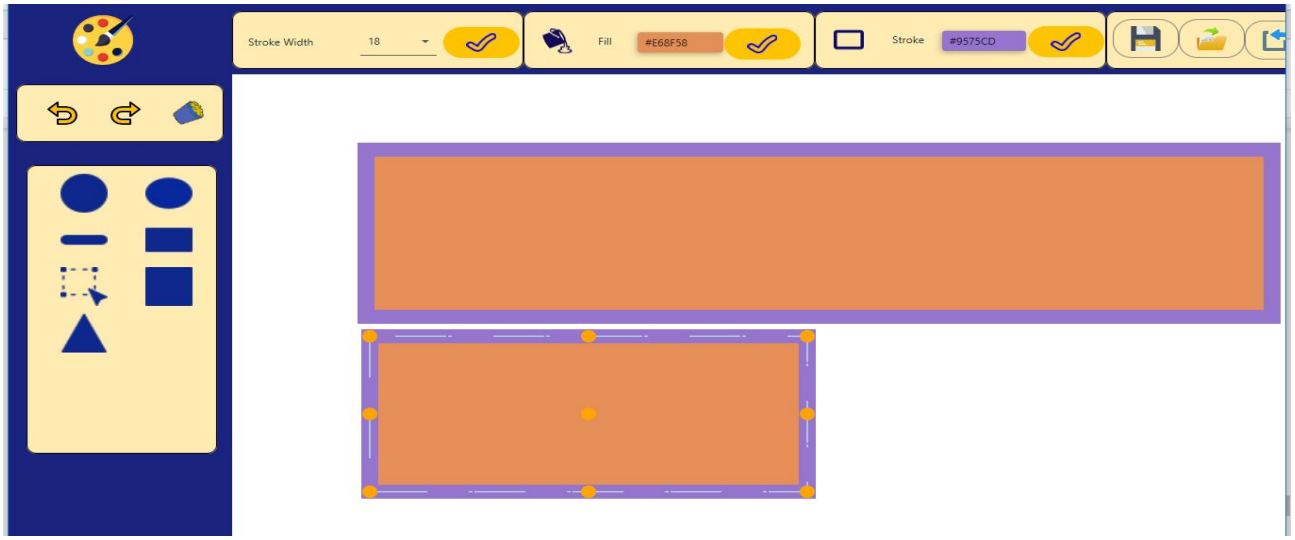
b) Just Drag them in the desired direction!

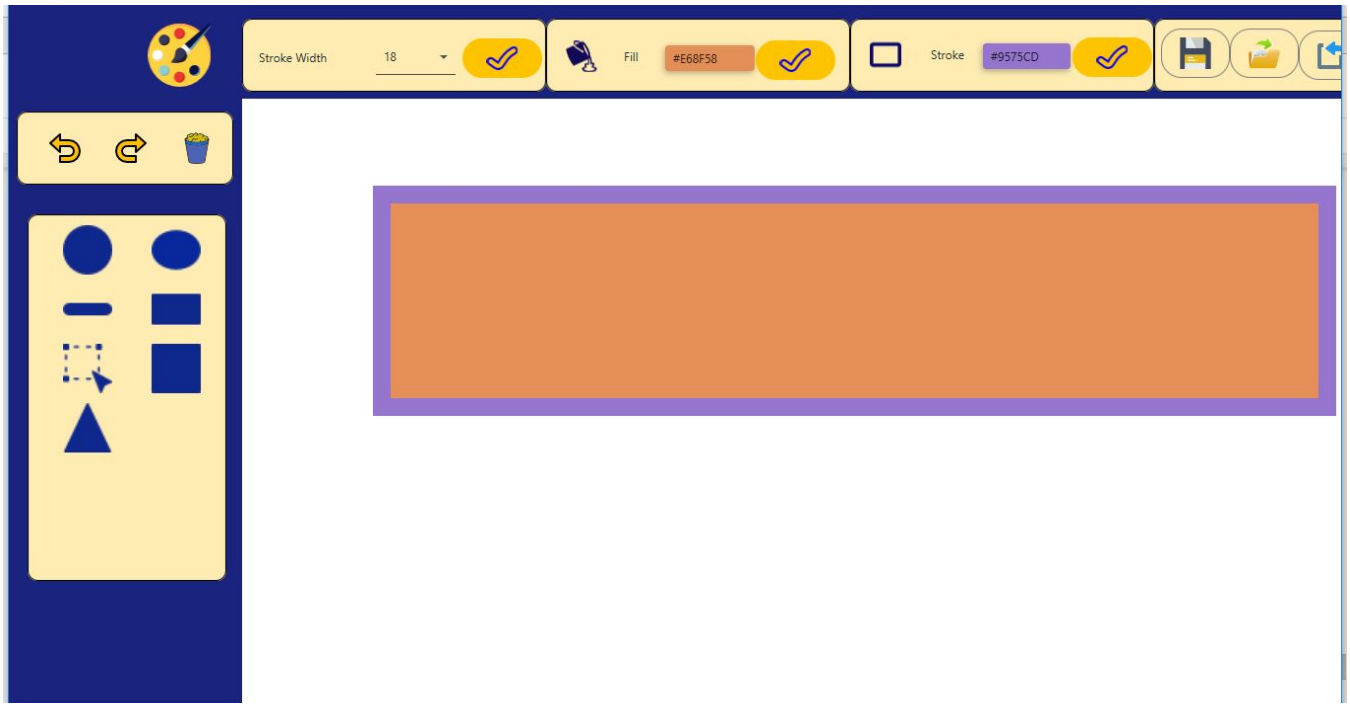


5) Delete Shape

a) Select shape to delete as described above

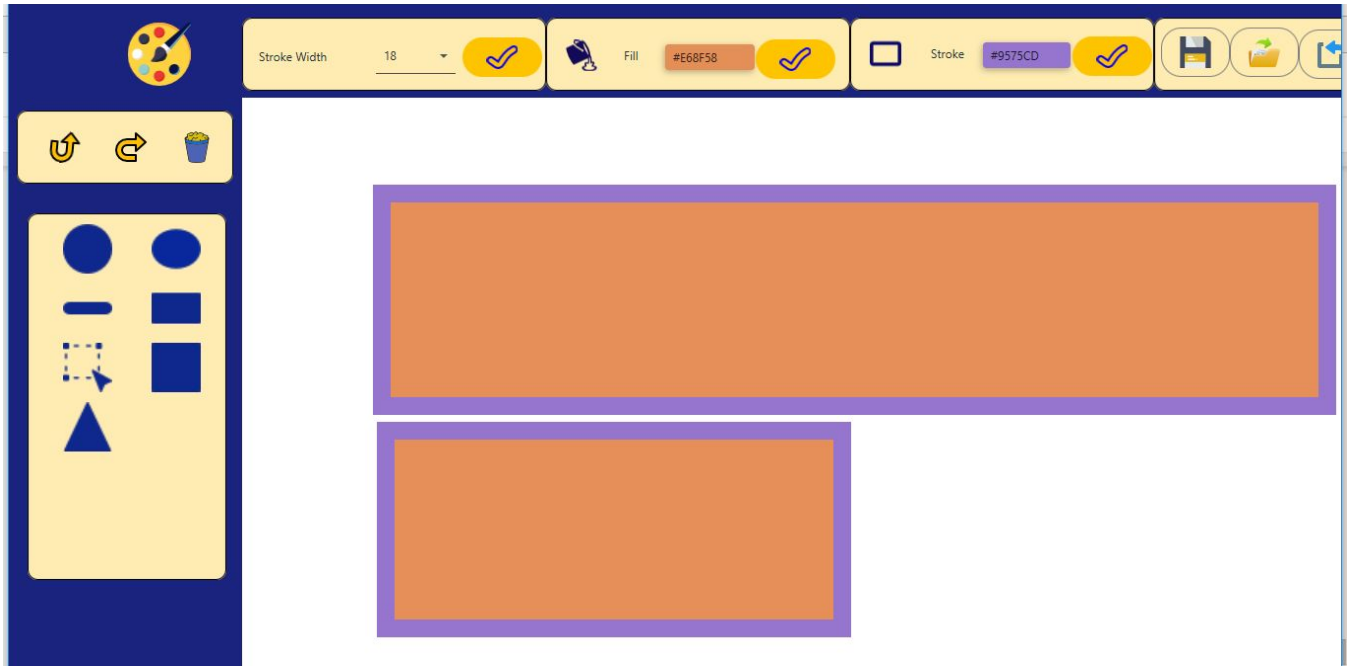
b) Click on the Trash Icon!



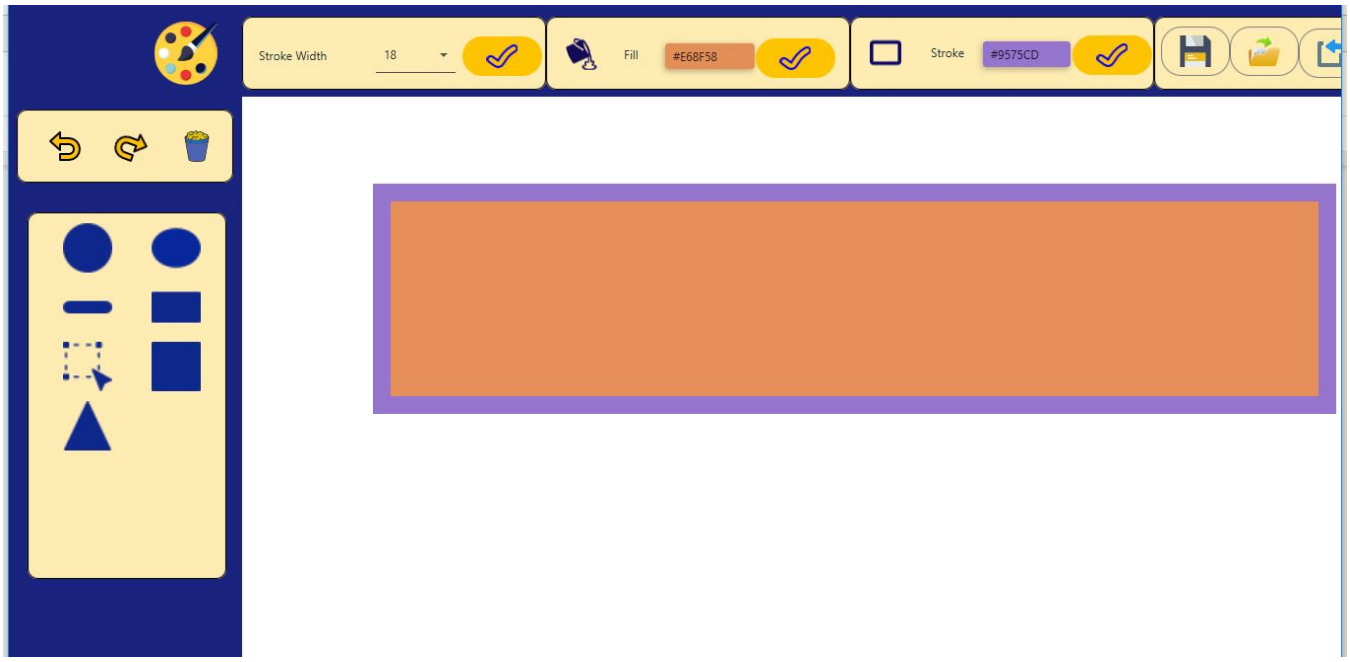


6)Undo/Redo

Just click Undo button to undo last Action (Up to 20 Actions)



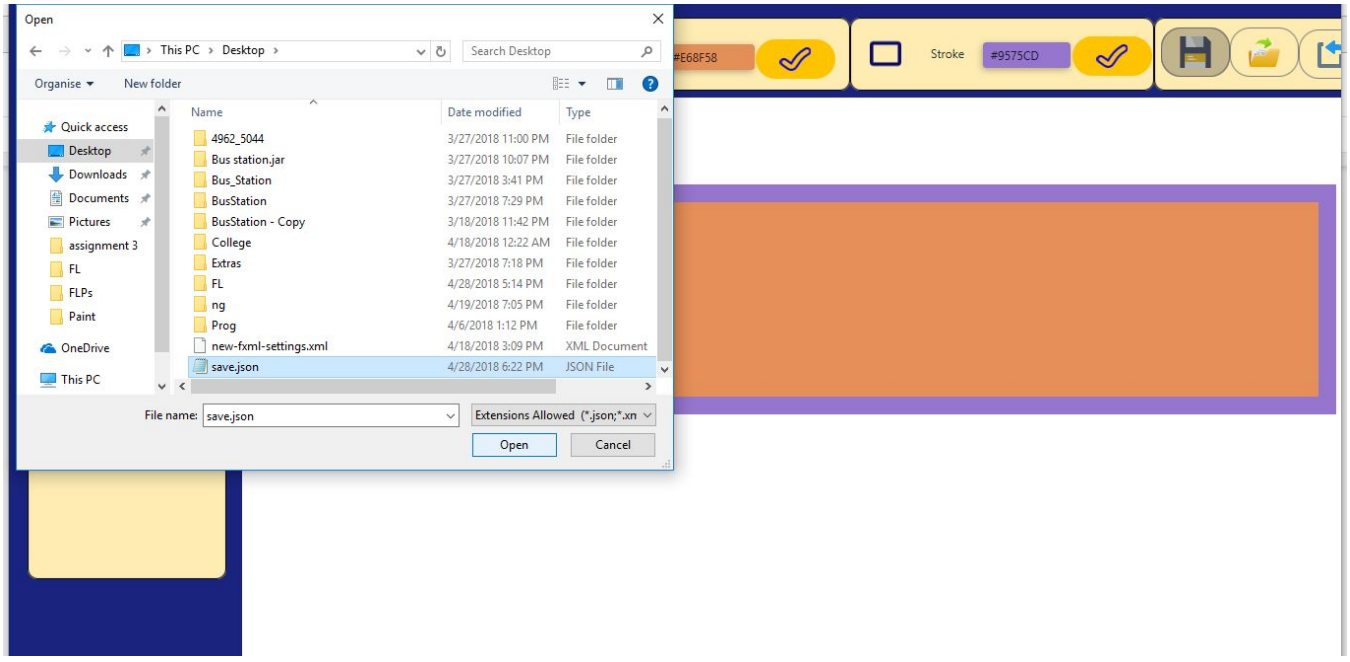
And Click Re-do to re-do the last Action (Up to 20 Actions)



7) To Save a project (Drawing)

a) Click the Save button at the top-right

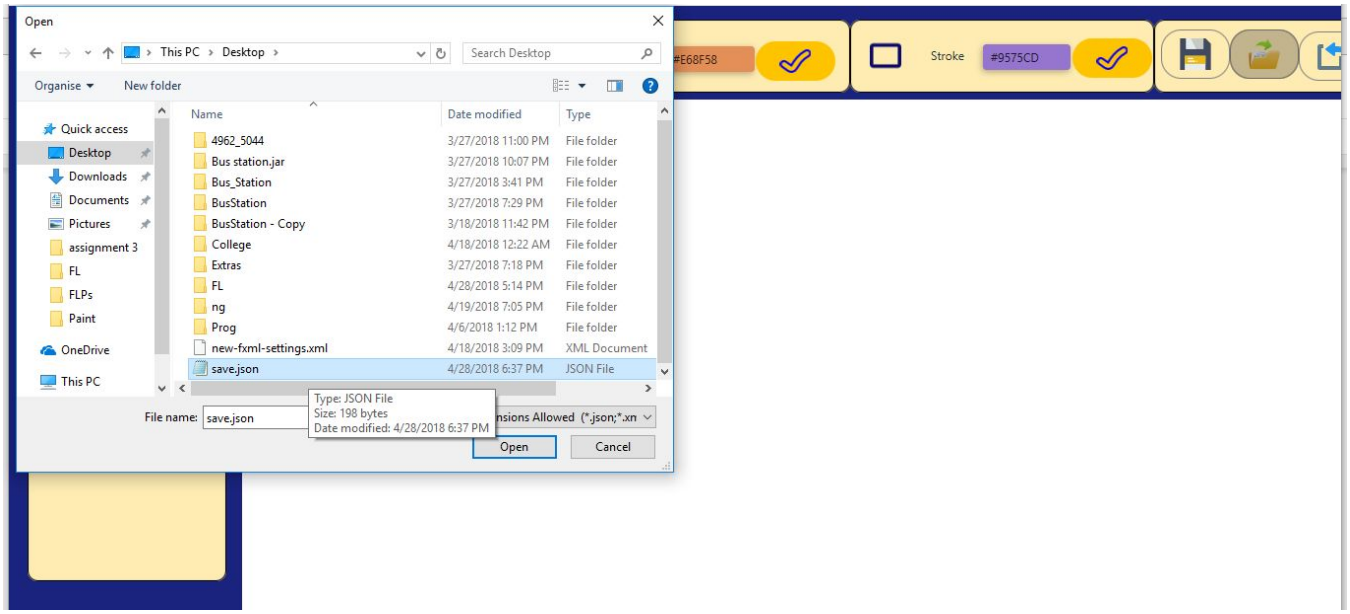
b) Select a json/xml file to save in

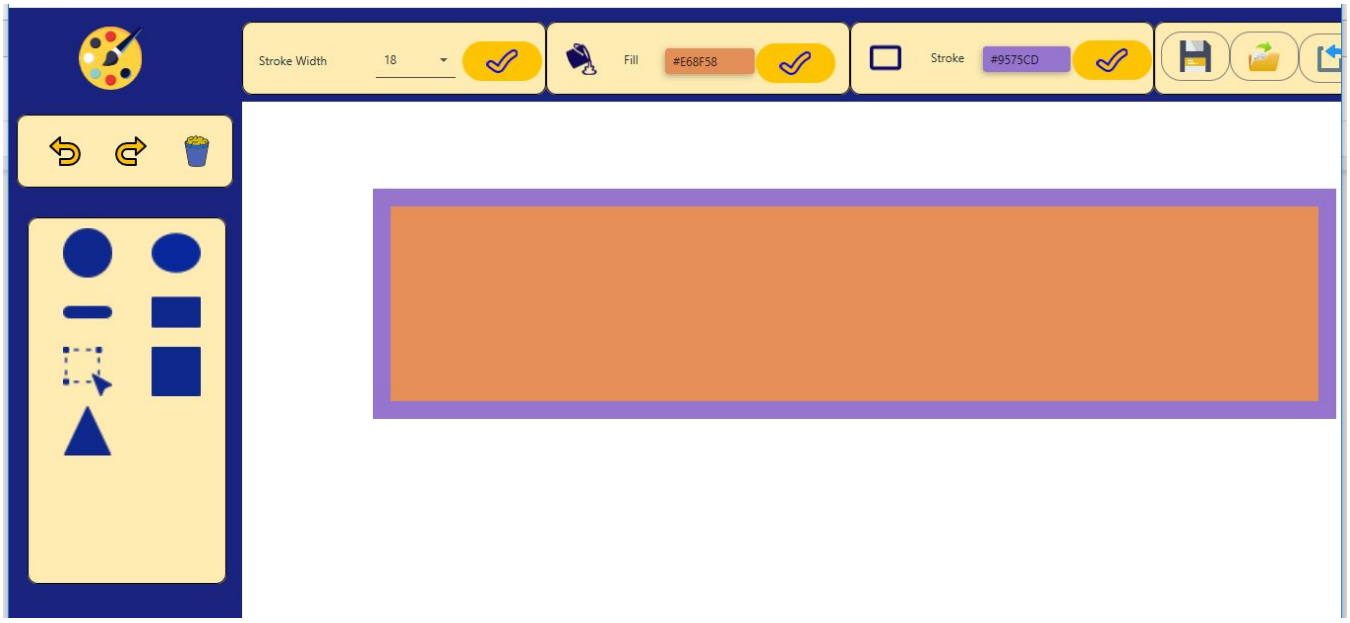


8) To open/load a saved project

a) Click the open icon in the top right of the screen

b) Choose the .json/.xml file where you have your drawing saved

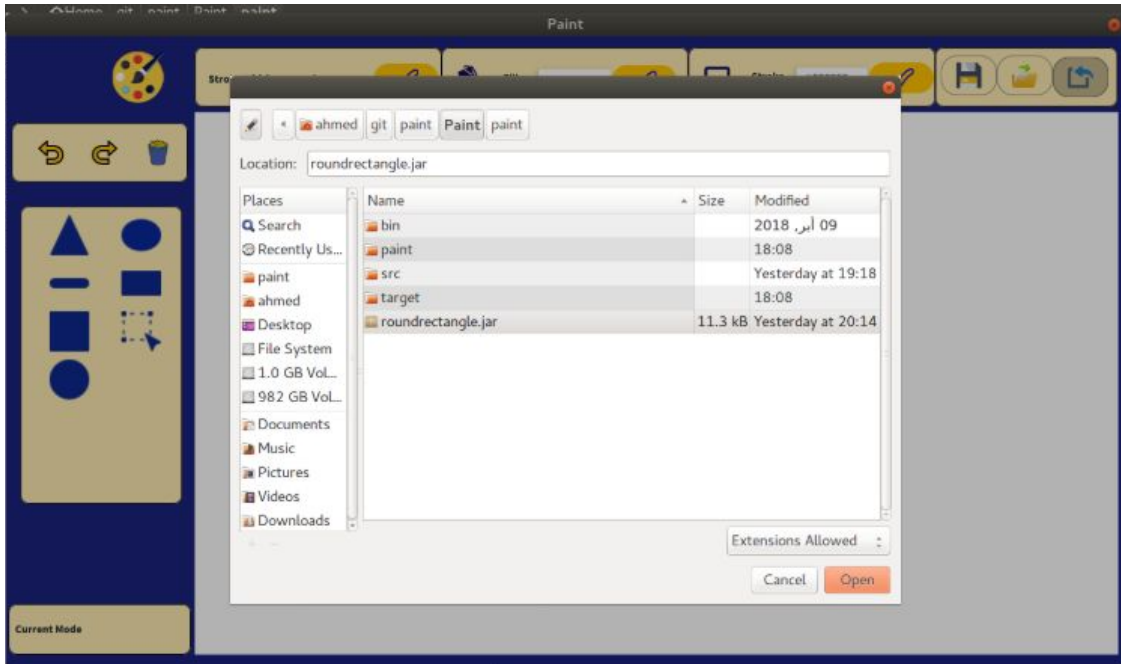


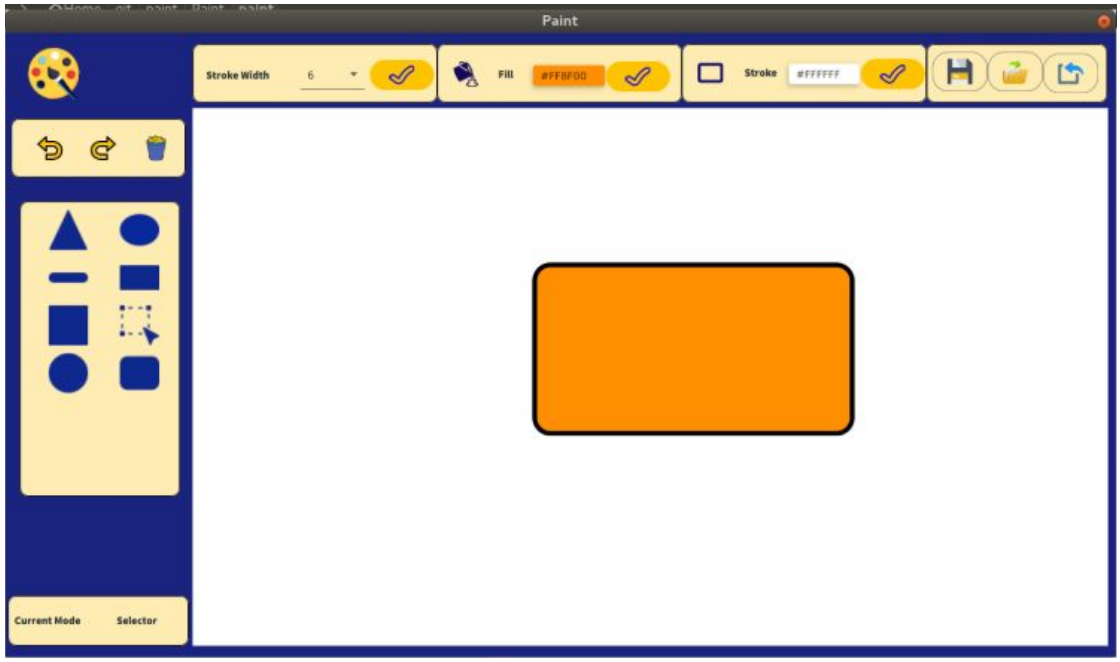


9) Add new plugin (Round rectangle in this example)

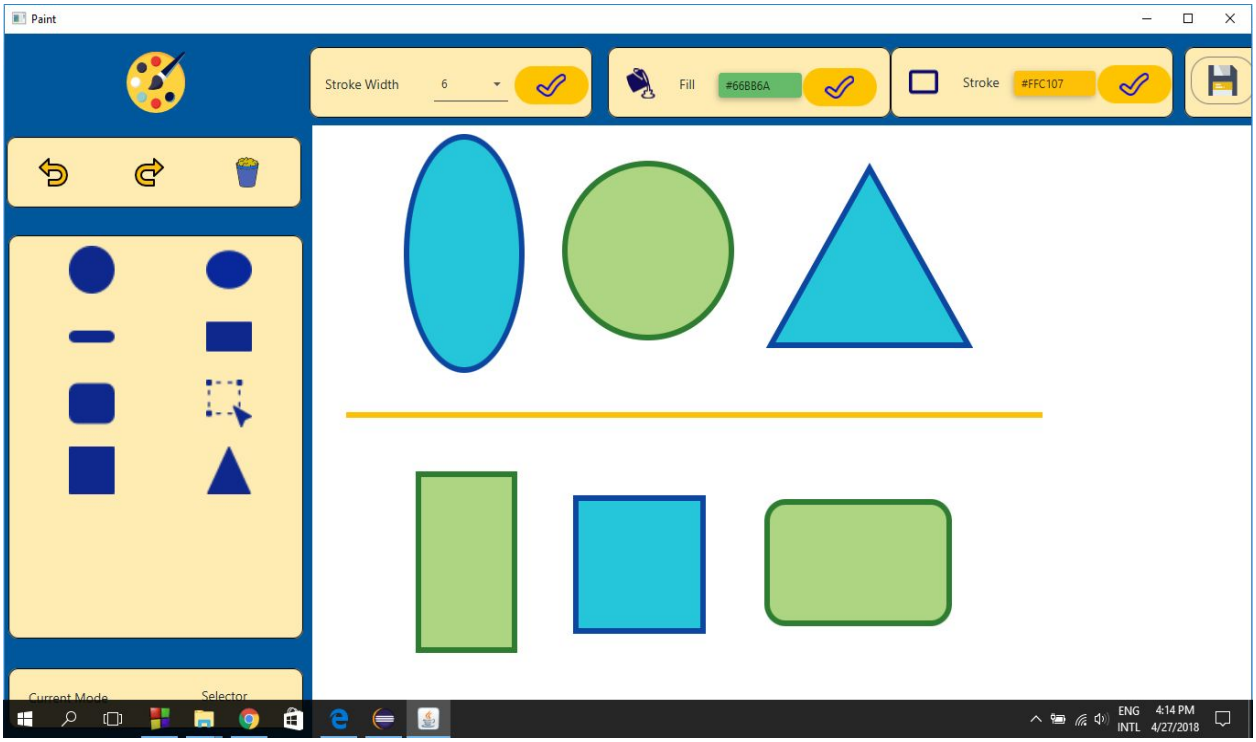
a) Click on the import icon on the top right corner

b) Choose the jar file





Sample Runs:



This sample run shows all the available drawable shapes.

References:

1- Json library : json-simple library

2- GUI library : JFoenix