24_10 Paint

A report

@hossamElDinElKordy
@ahmedYasserAhmed
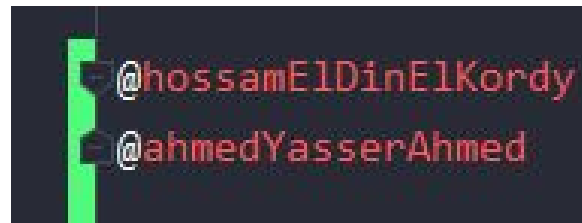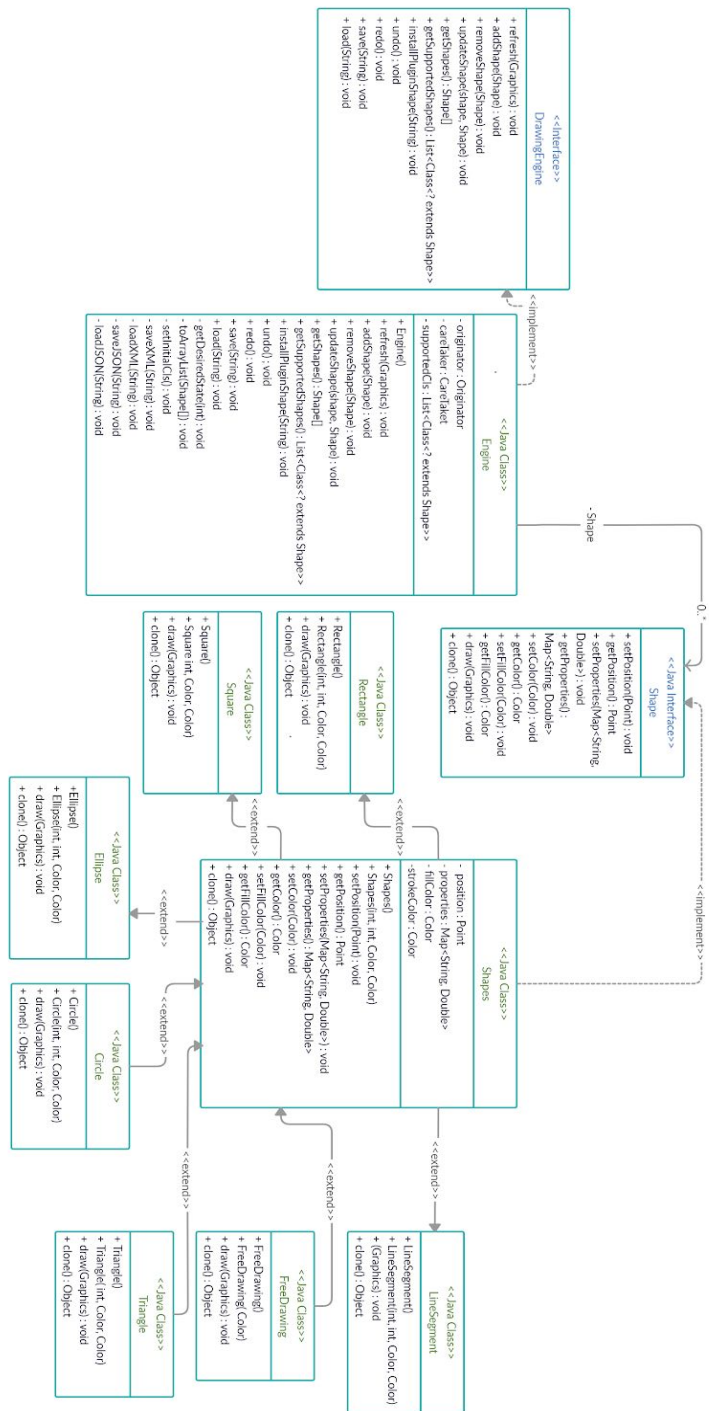
Problem Statement

It was required to develop a paint-like application in java using the object oriented features, the application should be able to draw basic shapes rectangle, square, circle, line segment , ….etc and should be able to save and load in two file extensions xml and json and should be able to dynamically load classes.

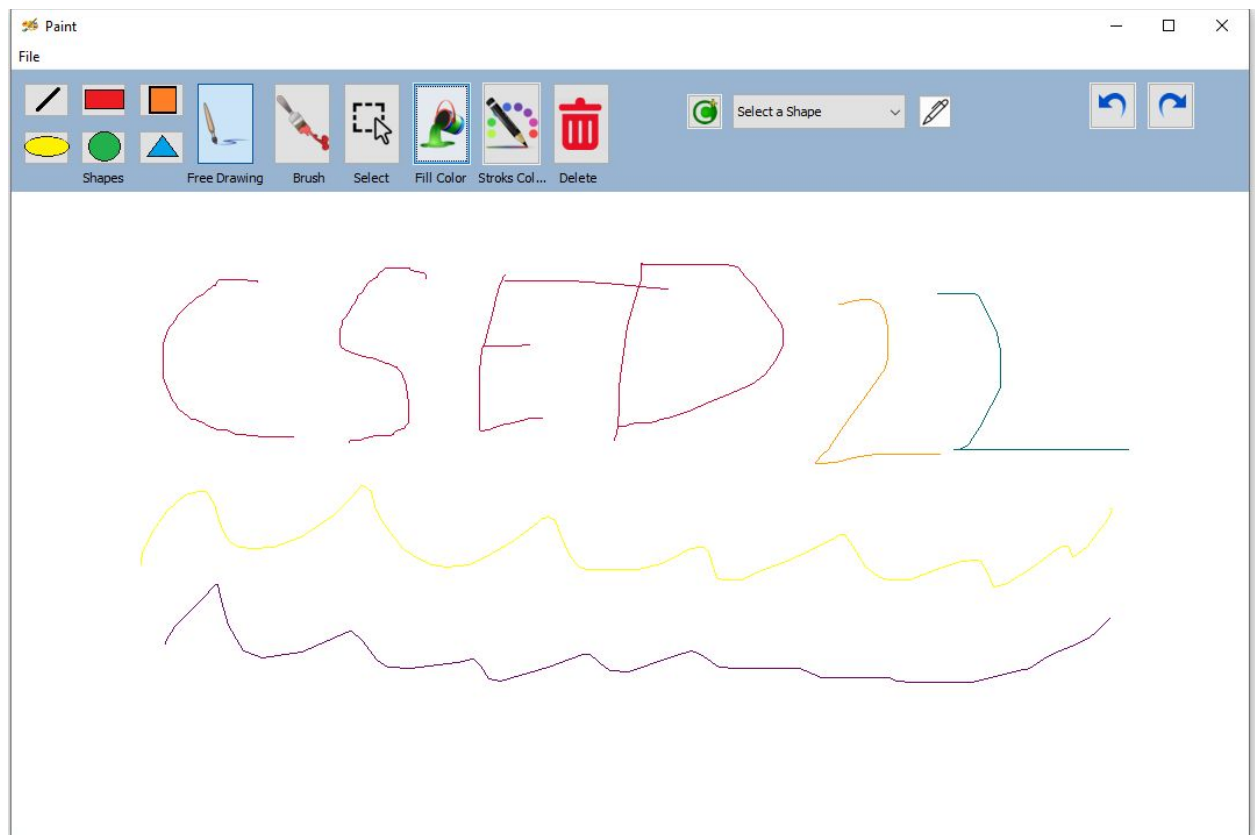## Part 1:Geometric Shapes Data Model

We created an abstract class called shapes that implements the interface Shape and all the shapes inherit from this class, one advantage of this approach is that the simple repetitive functions in each shape will be written only once.
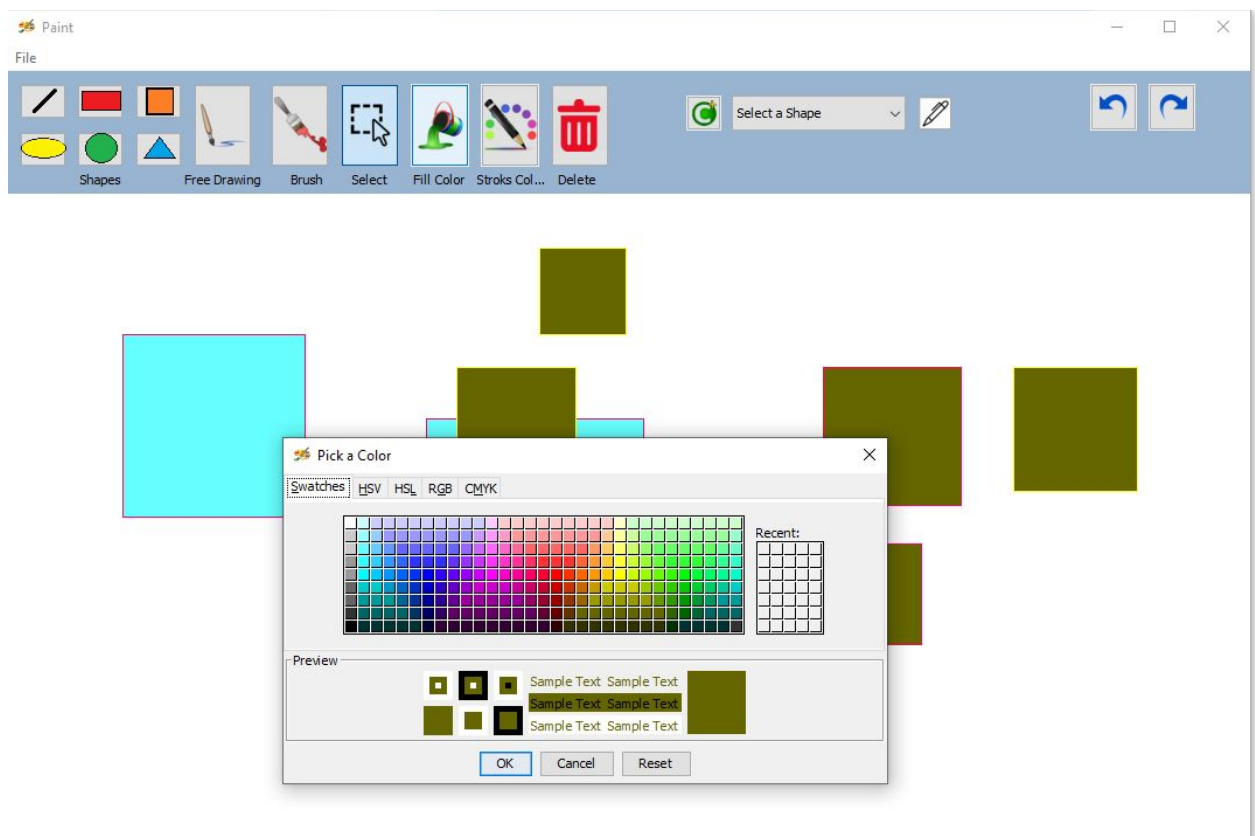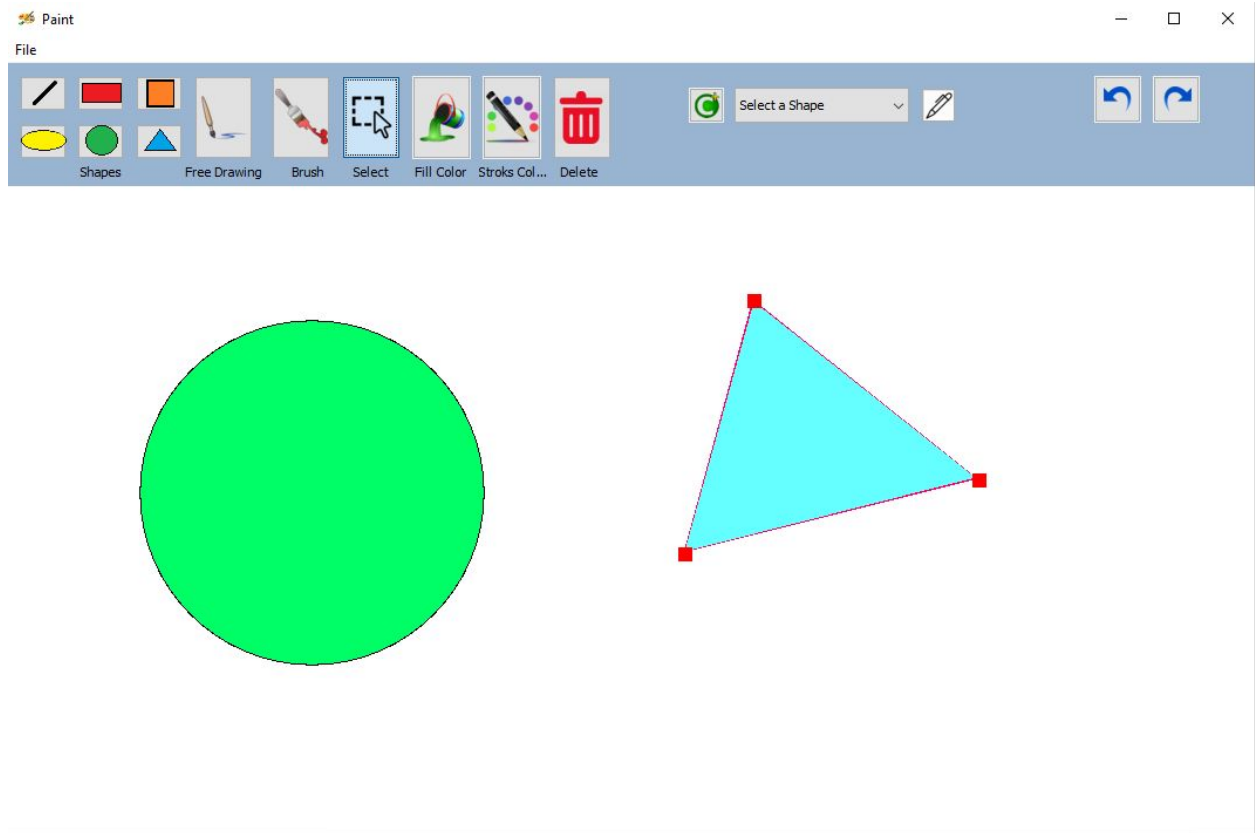
The following page contains the uml diagram.

(You can find a higher quality in the repository)

<<Interface>>
DrawingEngine

+ refresh(Graphics): void
+ addShape(Shape): void
+ removeShape(Shape): void
+ updateShape(Shape, Shape): void
+ getShapes(): Shape[]
+ getSupportedShapes(): List<Class? extends Shape>>
+ installPluginShape(String): void
+ undo(): void
+ redo(): void
+ save(String): void
+ load(String): void

<<implement>>

<<Java Class>>
Engine

- originator : Originator
- careTaker : CareTaker
- supportedCls : List<Class? extends Shape>

+ Engine()
+ refresh(Graphics): void
+ addShape(Shape): void
+ removeShape(Shape): void
+ updateShape(Shape, Shape): void
+ getShapes(): Shape[]
+ getSupportedShapes(): List<Class? extends Shape>>
+ installPluginShape(String): void
+ undo(): void
+ redo(): void
+ save(String): void
+ load(String): void
+ getDesiredState(int): void
- setInitialCls(): void
- toArrayList(Shape[]): void
- saveXML(String): void
- saveJSON(String): void
- loadXML(String): void
- loadJSON(String): void

- Shape
0..*

<<Java Interface>>
Shape

+ setPosition(Point): void
+ getPosition(): Point
+ setProperties(Map<String, Double>): void
+ getProperties(): Map<String, Double>
+ setColor(Color): void
+ getColor(): Color
+ setFillColor(Color): void
+ getFillColor(): Color
+ draw(Graphics): void
+ clone(): Object

<<implement>>

<<Java Class>>
Shapes

- position : Point
- properties : Map<String, Double>
- fillColor : Color
- strokeColor : Color

+ Shapes()
+ Shapes(int, int, Color, Color)
+ setPosition(Point): void
+ setProperties(Map<String, Double>): void
+ getProperties(): Map<String, Double>
+ setColor(Color): void
+ getColor(): Color
+ setFillColor(Color): void
+ getFillColor(): Color
+ draw(Graphics): void
+ clone(): Object

<<Java Class>>
Rectangle

+ Rectangle()
+ Rectangle(int, int, Color, Color)
+ draw(Graphics): void
+ clone(): Object

<<Java Class>>
Square

+ Square()
+ Square int, Color, Color)
+ draw(Graphics): void
+ clone(): Object

<<Java Class>>
Ellipse

+ Ellipse()
+ Ellipse(int, int, Color, Color)
+ draw(Graphics): void
+ clone(): Object

<<Java Class>>
Circle

+ Circle()
+ Circle(int, int, Color, Color)
+ draw(Graphics): void
+ clone(): Object

<<Java Class>>
Triangle

+ Triangle()
+ Triangle(int, Color, Color)
+ draw(Graphics): void
+ clone(): Object

<<Java Class>>
FreeDrawing

+ FreeDrawing()
+ FreeDrawing(Color)
+ draw(Graphics): void
+ clone(): Object

<<Java Class>>
LineSegment

+ LineSegment()
+ LineSegment(int, int, Color, Color)
+ (Graphics): void
+ clone(): Object

<<extend>>

Part 2: Drawing and Painting Application

The application uses java swing gui library here are some snapshots

The application contains a free drawing feature, you can draw whatever you want using the mouse as demonstrated in snapshot 1&2.

The application supports interactive move and resize; you can use the mouse to move a shape making sure the cursor is inside it and dragging it, resizing can be done by dragging the red squares that appear upon selecting the shape.

The application supports undoing and redoing the steps you have done, this was implemented using the memento design pattern.

Part 3: Dynamic Application Extensions and plug-ins

Our application can dynamically load extension jars at runtime and draw them just like predefined shapes
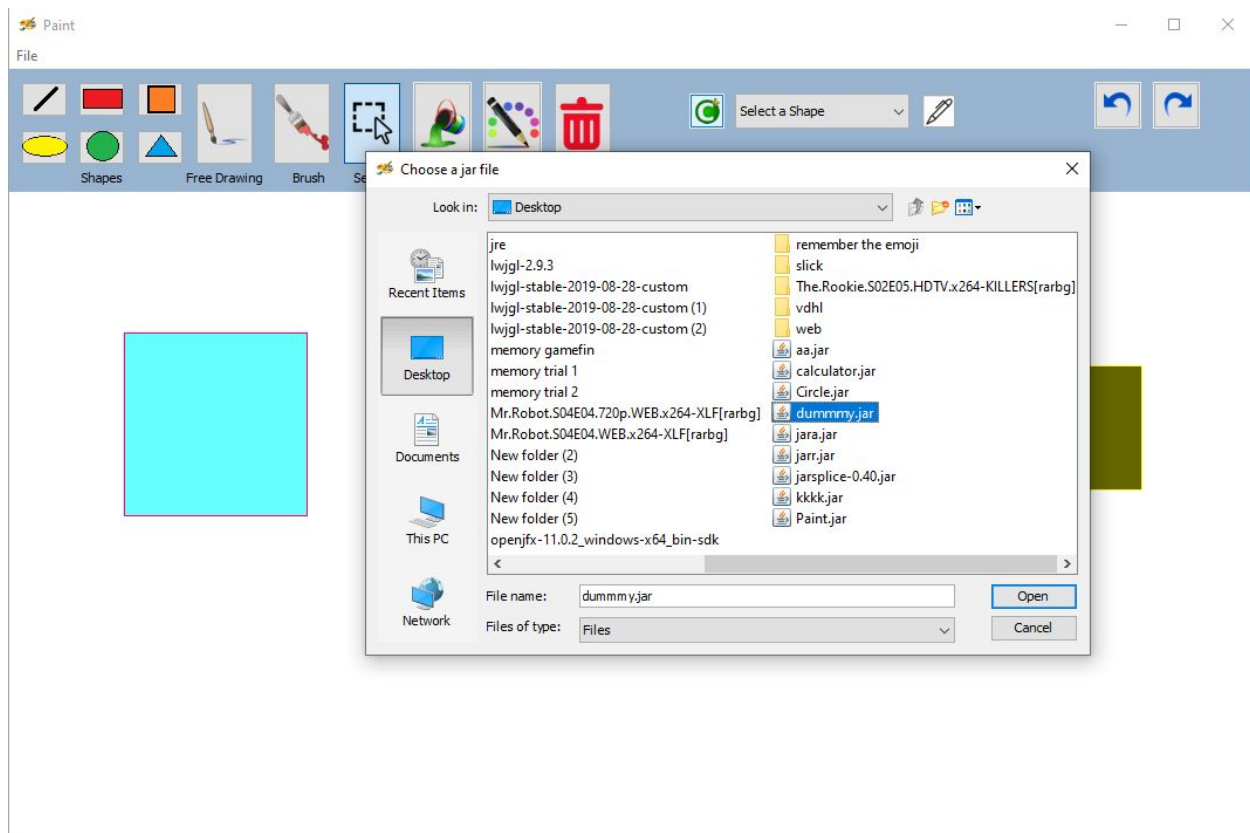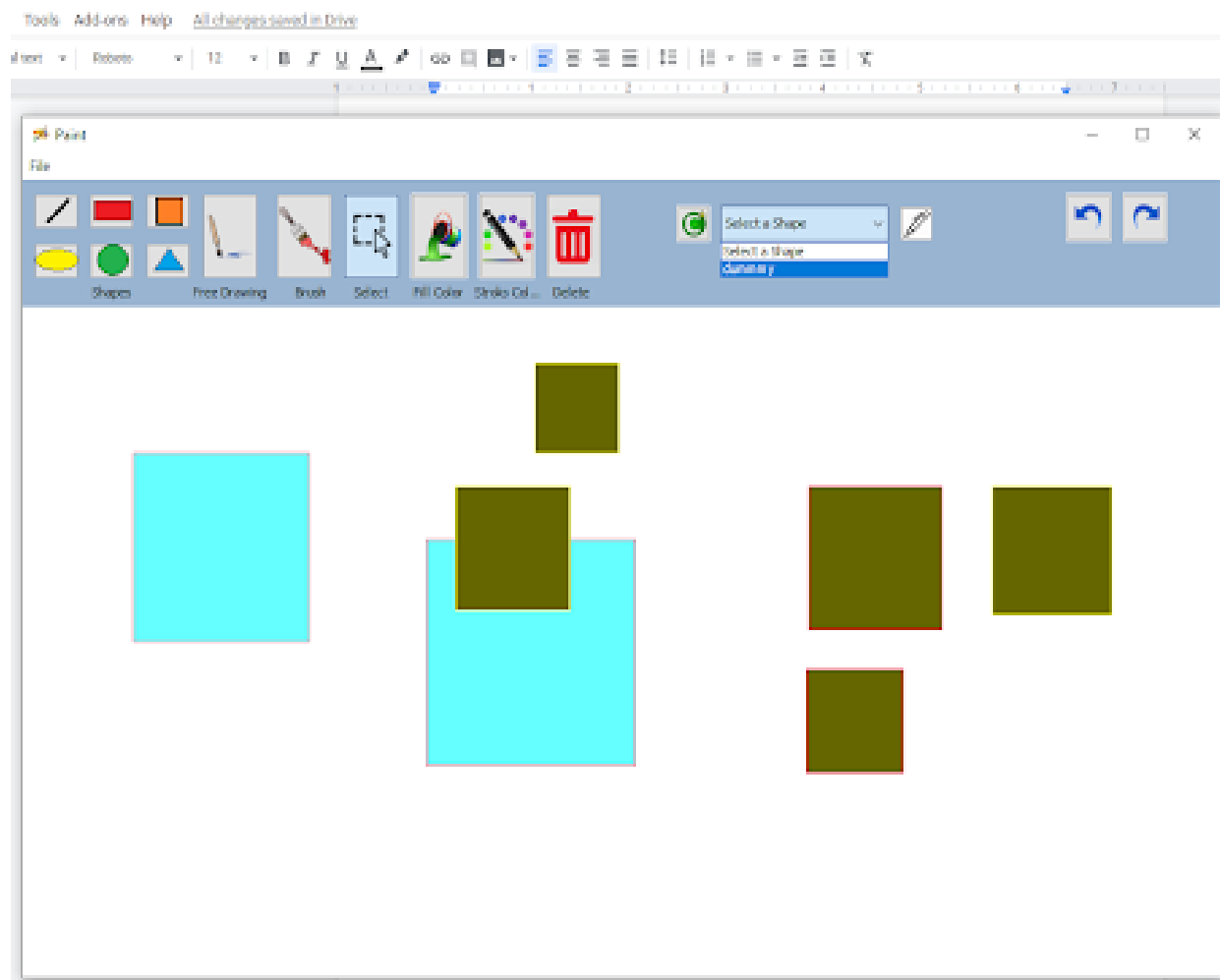
Part 4: Save and load

Our application saves and loads in two formats xml and json; for xml we used the jdk xml library, for the json we created our own writer and parser from scratch
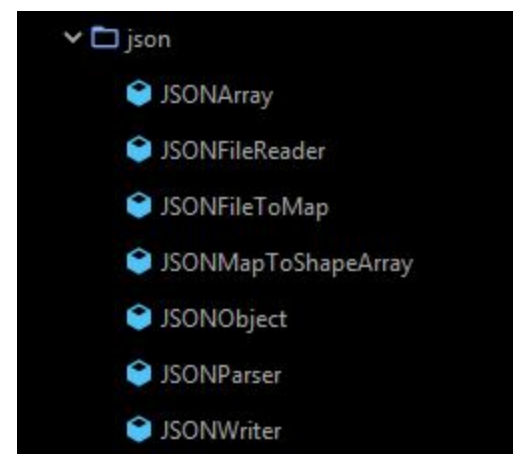
User manual

1)creating shapes

You can select any shape from the upper panel and create it by dragging on the screen using the mouse

You can use the mouse to draw freehand shapes using the free drawing tool.
You can choose the fill and stroke color before drawing or yocan edit them later using the brush tool

You can load dynamic extensions by clicking the loadclass icon (the green one with c on it in the upper mid section of the application) and you can select one of the loaded classes and you can press the icon with the pen on it next to the combo box to draw it

2)moving and resizing shapes

To move and resize shapes select the select tool (upper mid section) and to move drag the shape you want to move and it will move.

To resize click the shape once rectangles will appear around the shape the red ones can be dragged to resize the shape

3)undoing and redoing

To undo a certain change simply click the undo backward arrow in the upper right part of the application and to redo click the other arrow

4)deleting shapes

You can delete a shape by clicking the red delete bin in the upper mid section of the application and then clicking inside the desired shape once

5)saving a file

To save a file click file in the upper menu and then save and choose your desired path and write the file name followed by .json or .xml

6)loading a file

To load a file click file in the upper menu and then open and choose your desired file