Peinto

Description

The application is a paint app called *Peinto.* It's implemented in Java and the GUI is made using SWING in which we made use of its Graphics class which is the abstract super class for all graphics contexts which allow an application to draw onto components. The shapes are drawn on Canvas using Graphics2D.

Design Overview

The code's architecture follows MVC (Model View Controller) architectural pattern in which the Model is responsible for managing the app's drawing data, plus saving and loading them. On the other hand, the view is the user's interface, the paint window and it doesn't talk directly to the model but through the controller. The controller commands the View's actions and command what to display. The controller takes command and information from the user through multiple action and mouse listeners and motion, process the information and talks to the Model to retrieve or send data. It also takes all compiled information and command the View to how represent it to the user.
The code implements some important design patterns such as:

<u>Singleton DP:</u> used to create classes from which we can create only one object. The Canvas class is used as singleton because only one Canvas is needed per paint project.

<u>Strategy DP:</u> used in saving and loading in XML and JSON. XML and JSON classes both implement an interface called SaveAndLoad and override its two functions: save() and load().

<u>Factory DP:</u> used in constructing new shapes to draw on Canvas by taking String parameter specifying the required shape to draw and returning the new shape created.

<u>Memento DP:</u> used in redo/undo functionalities, consists of 3 classes: *Memento*, *Originator* and *CareTaker*. The *Memento* is the project's state at a specific point in time, it saves the ArrayList of shapes. The *Originator* takes the state of the project as the ArrayList shapes and save it to a Memento object and vice versa. *CareTaker* contains 2 LinkedLists: undo and redo which contain the project's states/Mementos.

Command DP: used in redo/undo functionalities, consists of an interface *Command* which has one function *execute*, *UndoCommand* class and *RedoCommand* class which implement *Command*, an *Invoker* class which takes and executes commands and the *ControlDrawingEngine's* redo and undo functions request invoking *UndoCommand* and *RedoCommand* from *Invoker*.
Invoker object uses command pattern to identify which object will execute which command based on the type of command.

The features of the application

- Drawing shapes (line segment, circle, ellipse, square, rectangle, and triangle) and editing them using mouse dragging.
- Selecting single or multiple shapes one which different operations can be performed such as: resize, delete, move, change fill and stroke color.
- Undo/Redo on all operations.
- Save shapes in XML or JSON.
- Load previous drawings and modify them.
- Choose where to save the file.

Assumptions

- Every time a user wants to draw a shape he must click on its corresponding label then drag and drop the mouse on Canvas.
- If user wants to select a shape, he must click on select label then click on the shape.
- If user wants to deselect a shape, he must click on deselect then click on the shape he wants to deselect.
- If user wants to deselect all shapes he can just click on deselect all button
- When more than one shape is selected, operations like copy, delete, move and resize will be performed on all selected shapes.
- When selecting a shape it's stoke will be dashed.
- If the user wants to do any operation on the shape like coloring the shape, changing the stroke color, resizing the shape… he must select it first.
- The default fill color is white and the default stroke color is black.

- Undo and redo can be done over 20 operations maximum.
- Resizing and moving can be done anywhere in the Canvas.

Team work

Mayssara Omar

- Code design.
- Implemented MVC AP.
- Canvas (Singleton DP) and its Mouse Listeners and Adapters.
- Redo/Undo functionalities (Memento and Command DP).

Rowan Khaled

The saving and loading feature:

- JSON, XML files and file chooser.
- Following strategy DP.

Fatma Sherif

- Implemented Factory DP.
- Functions in ControlDrawingEngine.
- GUI design.

Data Structures

The code has 3 main data structures HashMap, Arraylist and LinkedList. The HashMaps were used to store each shape properties. The shapes currently drawn on the Canvas were stored in an ArrayList of type Shape. And finally, in Memento design pattern, two LinkedLists were used in CareTaker class, one to store the Canvas undo states (Mementos) and the other to store the Canvas redo states (Mementos).

Description of the important functions/modules

Canvas Class

Canvas class is a class that extends JPanel which is used to draw on. It displays the arrayList of Shapes and it contains important methods like: PaintComponent method where all of the painting code should be placed. The method will be invoked when it is time to paint. Singleton DP is also implemented in this class because we only need one object for the canvas.

ControlDrawingEngine Class:

ControlDrawingEngine class implements the interface “DrawingEngine”. This class has all the main methods and operations done on shapes (select, deselect, copy, move, delete, clear, resize, redo, undo, save, load). It also has the method drawCurrentShape which calls the draw method for the current shape at that moment.

<u>Refresh method:</u> redraws all the shapes in the arrayList on the canvas, it’s called in the paintComponent method which is in the canvas class.

<u>DrawCurrentShape:</u> it draws the current shape which is held in currentShape object by setting all its properties then calling the draw method in the shape’s class.

<u>Copy:</u> It saves the state of the canvas only if there are selected shapes, then it clones the selected shapes (deep copying) and adds them to the arrayList and draws them on the canvas.

<u>Delete:</u> delete methods saves the state of the canvas only if there are selected shapes, then it deletes all the shapes selected by removing them from the arrayList then repainting the Canvas.

<u>DeleteAll:</u> it clears the Canvas by deleting all the shapes.

<u>Resize:</u> it resizes all the selected shapes by changing their end or start positions by dragging the mouse.

<u>Select:</u> loops through the arrayList and sets the first shape that the mouse is on to be selected by changed its stroke to a dashed line and setting the selected

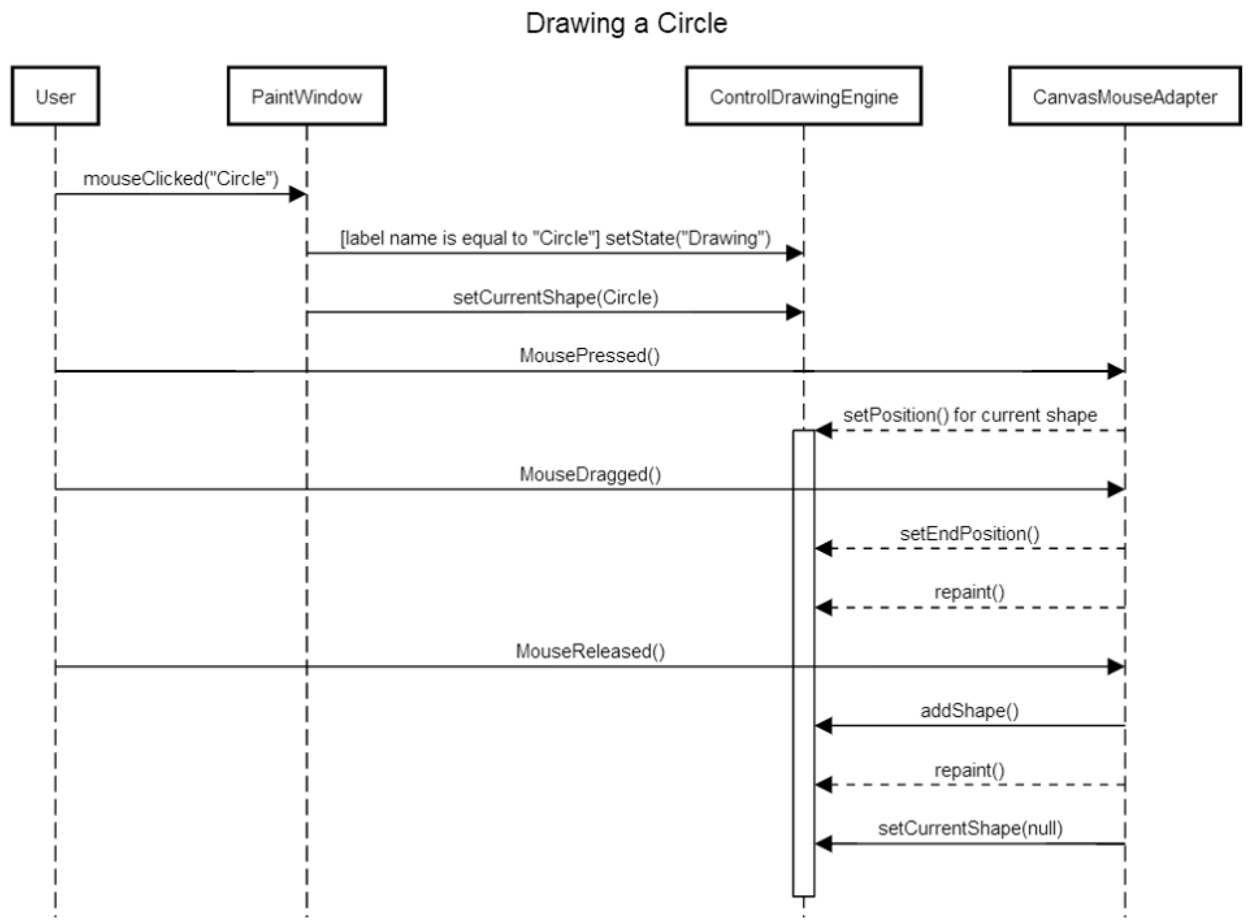
property to 1.0 , it also makes the selected shape appear on the top of another shapes if there is shape overlapping.

Move: it gets the difference between the first mouse click and then the dragged distance and adds this difference to the start and end points of the shape. If multiple shapes are selected they are all moved together from any point.

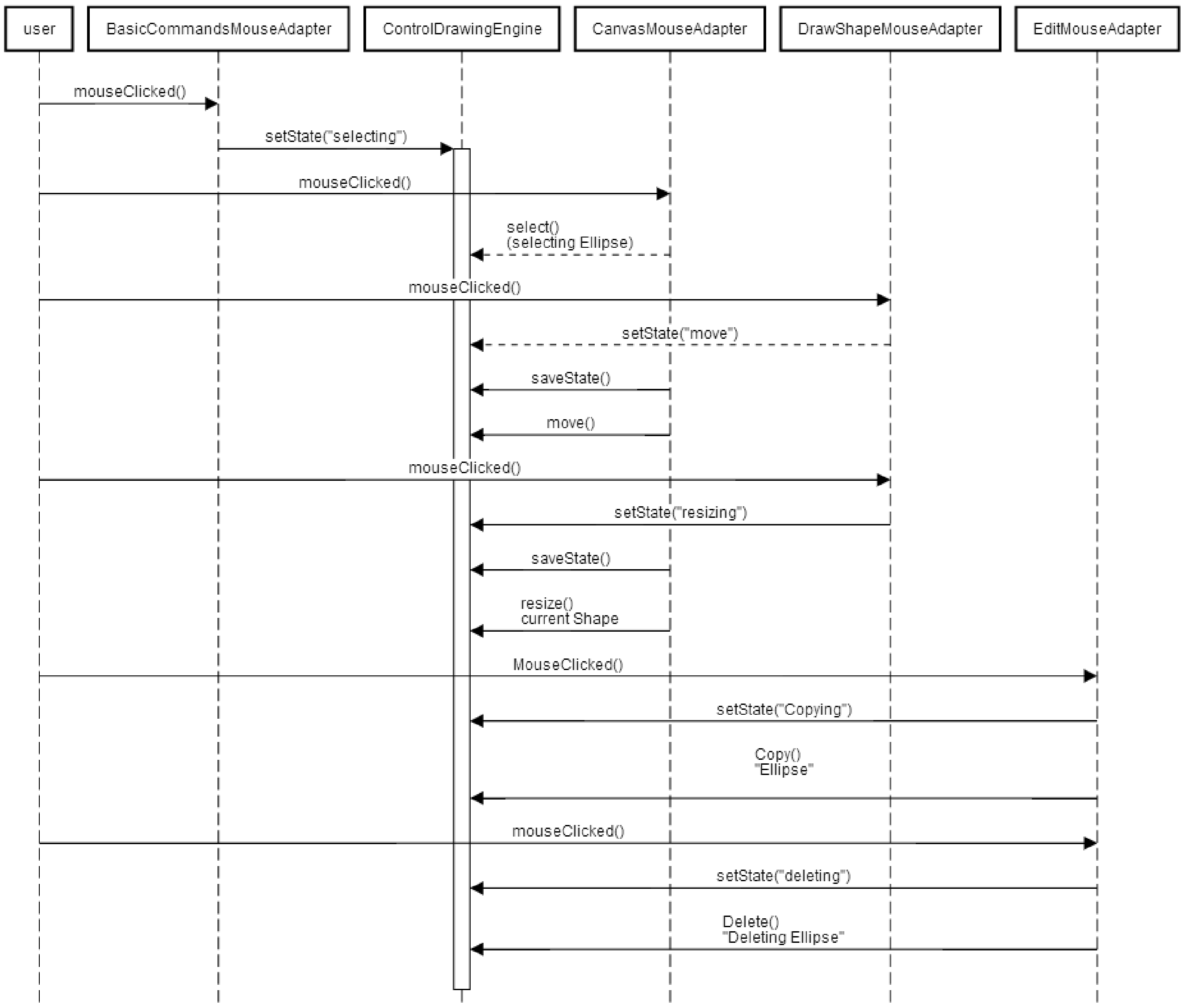
Saving and loading functions in XML and JSON classes:

The function takes two parameters: path, ArrayList of type Shape.
As the user choose the type of file (JSON or XML) the corresponding function will be called.
XML saving function:
The encoder will write the ArrayList of shapes in the file following XML format.
XML loading function:
The decoder will read the ArrayList from the XML file and the previous shapes will display on the Canvas.
JSON saving function: Using “json-simple 1.1 library”.
The function takes every detail in the ArrayList and save it in a HashMap.
JSON loading function: Using “json-simple 1.1 library.”
Using parser in the reading, it will read the HashMap from the file then display the previous shapes on the Canvas.

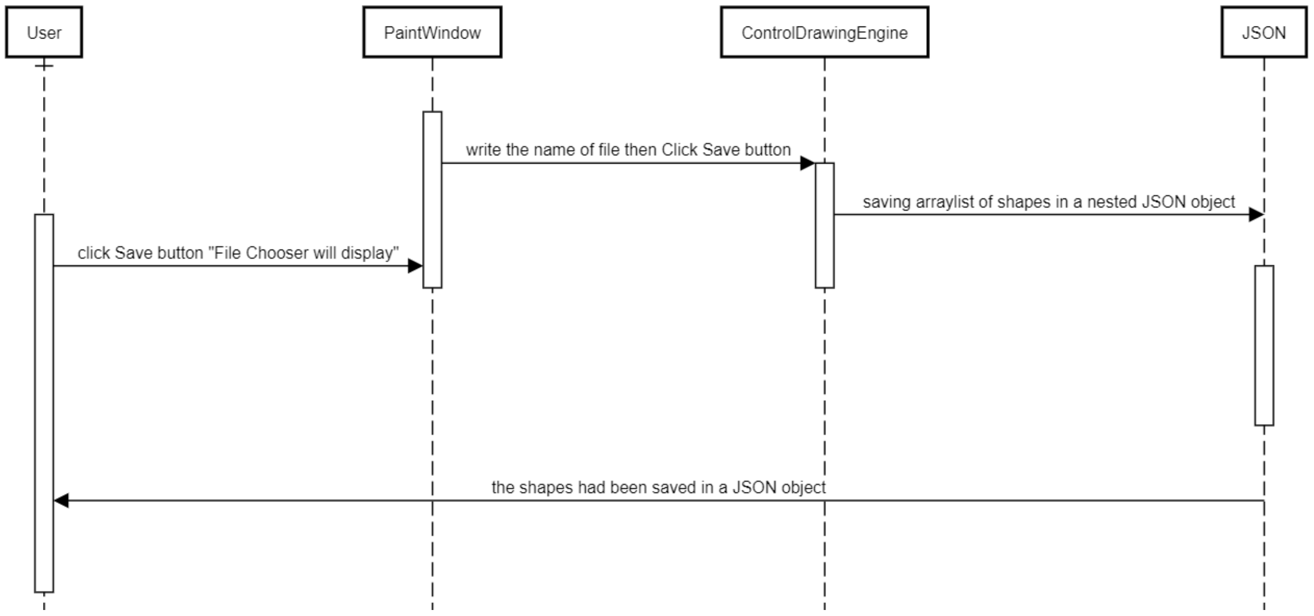
Sequence diagrams



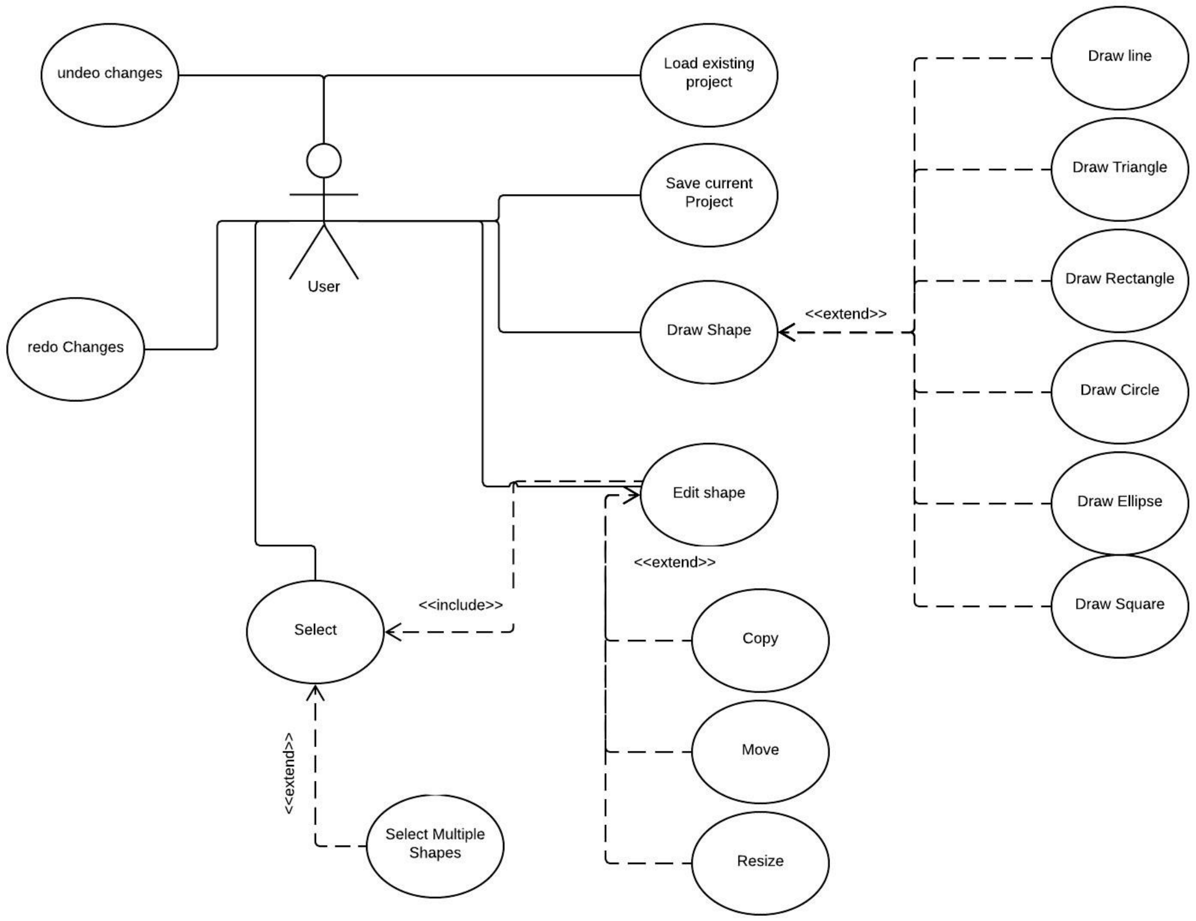
Editing an Ellipse



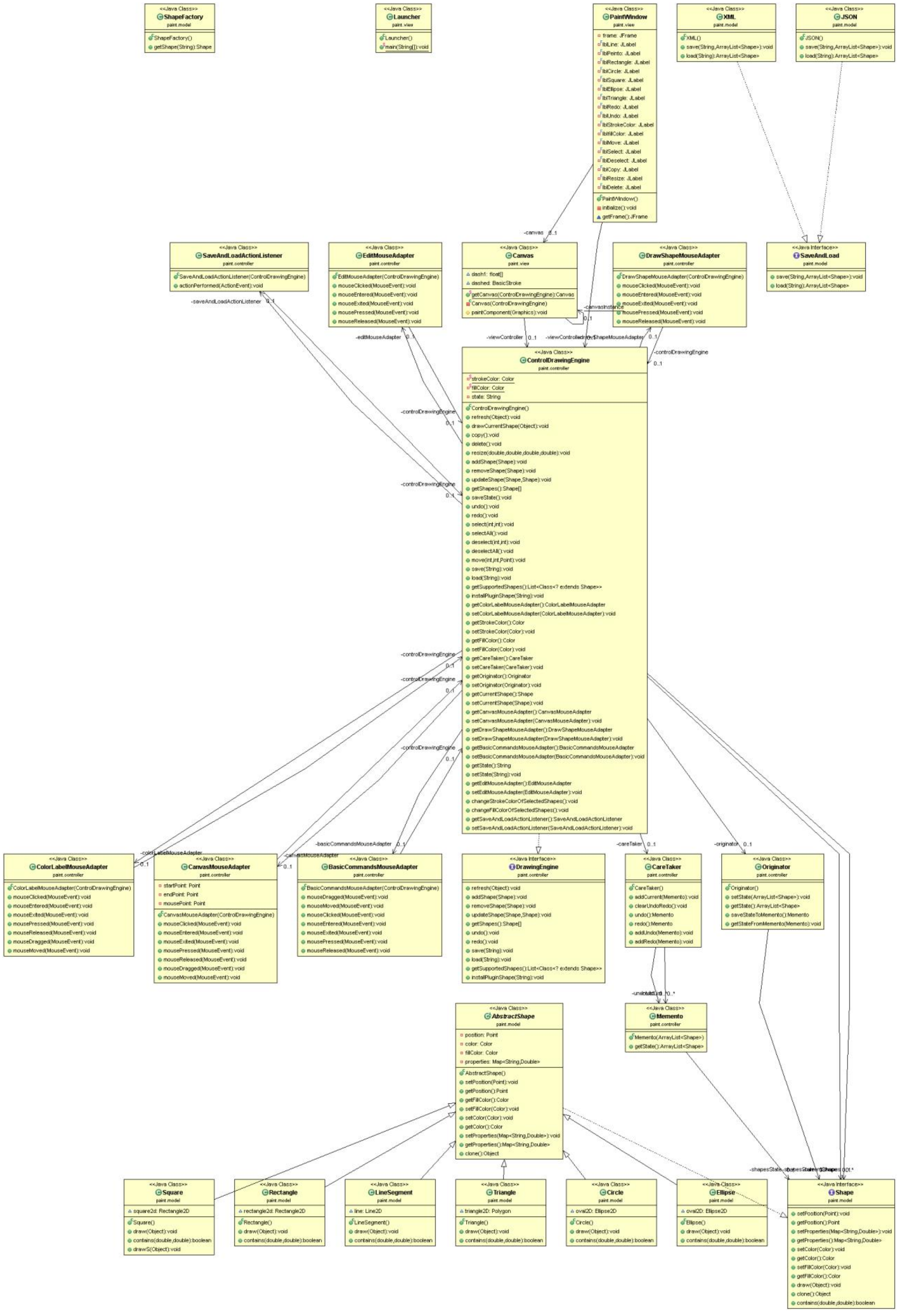
Saving in JSON file


User
PaintWindow
ControlDrawingEngine
JSON
write the name of file then Click Save button
saving arraylist of shapes in a nested JSON object
click Save button "File Chooser will display"
the shapes had been saved in a JSON object

Use case diagram



UML



<<Java Class>> ShapeFactory paint.model
ShapeFactory()
getShape(String):Shape
<<Java Class>> Launcher paint.view
Launcher()
main(String[]):void
<<Java Class>> PaintWindow paint.view
frame: JFrame
lblLine: JLabel
lblPeinto: JLabel
lblRectangle: JLabel
lblCircle: JLabel
lblSquare: JLabel
lblEllipse: JLabel
lblTriangle: JLabel
lblRedo: JLabel
lblUndo: JLabel
lblStrokeColor: JLabel
lblFillColor: JLabel
lblMove: JLabel
lblSelect: JLabel
lblDeselect: JLabel
lblCopy: JLabel
lblResize: JLabel
lblDelete: JLabel
PaintWindow()
initialize():void
getFrame():JFrame
<<Java Class>> XML paint.model
XML()
save(String,ArrayList<Shape>):void
load(String):ArrayList<Shape>
<<Java Class>> JSON paint.model
JSON()
save(String,ArrayList<Shape>):void
load(String):ArrayList<Shape>
-canvas 0..1
<<Java Class>> SaveAndLoadActionListener paint.controller
SaveAndLoadActionListener(ControlDrawingEngine)
actionPerformed(ActionEvent):void
<<Java Class>> EditMouseAdapter paint.controller
EditMouseAdapter(ControlDrawingEngine)
mouseClicked(MouseEvent):void
mouseEntered(MouseEvent):void
mouseExited(MouseEvent):void
mousePressed(MouseEvent):void
mouseReleased(MouseEvent):void
<<Java Class>> Canvas paint.view
dash1: float[]
dashed: BasicStroke
getCanvas(ControlDrawingEngine):Canvas
Canvas(ControlDrawingEngine)
paintComponent(Graphics):void
<<Java Class>> DrawShapeMouseAdapter paint.controller
DrawShapeMouseAdapter(ControlDrawingEngine)
mouseClicked(MouseEvent):void
mouseEntered(MouseEvent):void
mouseExited(MouseEvent):void
mousePressed(MouseEvent):void
mouseReleased(MouseEvent):void
<<Java Interface>> SaveAndLoad paint.model
save(String,ArrayList<Shape>):void
load(String):ArrayList<Shape>
-saveAndLoadActionListener 0..1
-canvasInstance 0..1
-editMouseAdapter 0..1
-viewController 0..1
-viewController
-drawShapeMouseAdapter 0..1
-controlDrawingEngine 0..1
<<Java Class>> ControlDrawingEngine paint.controller
strokeColor: Color
fillColor: Color
state: String
ControlDrawingEngine()
refresh(Object):void
drawCurrentShape(Object):void
copy():void
delete():void
resize(double,double,double,double):void
addShape(Shape):void
removeShape(Shape):void
updateShape(Shape,Shape):void
getShapes():Shape[]
saveState():void
undo():void
redo():void
select(int,int):void
selectAll():void
deselect(int,int):void
deselectAll():void
move(int,int,Point):void
save(String):void
load(String):void
getSupportedShapes():List<Class<? extends Shape>>
installPluginShape(String):void
getColorLabelMouseAdapter():ColorLabelMouseAdapter
setColorLabelMouseAdapter(ColorLabelMouseAdapter):void
getStrokeColor():Color
setStrokeColor(Color):void
getFillColor():Color
setFillColor(Color):void
getCareTaker():CareTaker
setCareTaker(CareTaker):void
getOriginator():Originator
setOriginator(Originator):void
getCurrentShape():Shape
setCurrentShape(Shape):void
getCanvasMouseAdapter():CanvasMouseAdapter
setCanvasMouseAdapter(CanvasMouseAdapter):void
getDrawShapeMouseAdapter():DrawShapeMouseAdapter
setDrawShapeMouseAdapter(DrawShapeMouseAdapter):void
getBasicCommandsMouseAdapter():BasicCommandsMouseAdapter
setBasicCommandsMouseAdapter(BasicCommandsMouseAdapter):void
getState():String
setState(String):void
getEditMouseAdapter():EditMouseAdapter
setEditMouseAdapter(EditMouseAdapter):void
changeStrokeColorOfSelectedShapes():void
changeFillColorOfSelectedShapes():void
getSaveAndLoadActionListener():SaveAndLoadActionListener
setSaveAndLoadActionListener(SaveAndLoadActionListener):void
-controlDrawingEngine 0..1
-colorLabelMouseAdapter 0..1
-canvasMouseAdapter 0..1
-basicCommandsMouseAdapter 0..1
-careTaker 0..1
-originator 0..1
<<Java Class>> ColorLabelMouseAdapter paint.controller
ColorLabelMouseAdapter(ControlDrawingEngine)
mouseClicked(MouseEvent):void
mouseEntered(MouseEvent):void
mouseExited(MouseEvent):void
mousePressed(MouseEvent):void
mouseReleased(MouseEvent):void
mouseDragged(MouseEvent):void
mouseMoved(MouseEvent):void
<<Java Class>> CanvasMouseAdapter paint.controller
startPoint: Point
endPoint: Point
mousePoint: Point
CanvasMouseAdapter(ControlDrawingEngine)
mouseClicked(MouseEvent):void
mouseEntered(MouseEvent):void
mouseExited(MouseEvent):void
mousePressed(MouseEvent):void
mouseReleased(MouseEvent):void
mouseDragged(MouseEvent):void
mouseMoved(MouseEvent):void
<<Java Class>> BasicCommandsMouseAdapter paint.controller
BasicCommandsMouseAdapter(ControlDrawingEngine)
mouseDragged(MouseEvent):void
mouseMoved(MouseEvent):void
mouseClicked(MouseEvent):void
mouseEntered(MouseEvent):void
mouseExited(MouseEvent):void
mousePressed(MouseEvent):void
mouseReleased(MouseEvent):void
<<Java Interface>> DrawingEngine paint.controller
refresh(Object):void
addShape(Shape):void
removeShape(Shape):void
updateShape(Shape,Shape):void
getShapes():Shape[]
undo():void
redo():void
save(String):void
load(String):void
getSupportedShapes():List<Class<? extends Shape>>
installPluginShape(String):void
<<Java Class>> CareTaker paint.controller
CareTaker()
addCurrent(Memento):void
clearUndoRedo():void
undo():Memento
redo():Memento
addUndo(Memento):void
addRedo(Memento):void
<<Java Class>> Originator paint.controller
Originator()
setState(ArrayList<Shape>):void
getState():ArrayList<Shape>
saveStateToMemento():Memento
getStateFromMemento(Memento):void
0..*
<<Java Class>> Memento paint.controller
Memento(ArrayList<Shape>)
getState():ArrayList<Shape>
<<Java Class>> AbstractShape paint.model
position: Point
color: Color
fillColor: Color
properties: Map<String,Double>
AbstractShape()
setPosition(Point):void
getPosition():Point
getFillColor():Color
setFillColor(Color):void
setColor(Color):void
getColor():Color
setProperties(Map<String,Double>):void
getProperties():Map<String,Double>
clone():Object
-shapesState 0..*
<<Java Class>> Square paint.model
square2d: Rectangle2D
Square()
draw(Object):void
contains(double,double):boolean
drawS(Object):void
<<Java Class>> Rectangle paint.model
rectangle2d: Rectangle2D
Rectangle()
draw(Object):void
contains(double,double):boolean
<<Java Class>> LineSegment paint.model
line: Line2D
LineSegment()
draw(Object):void
contains(double,double):boolean
<<Java Class>> Triangle paint.model
triangle2D: Polygon
Triangle()
draw(Object):void
contains(double,double):boolean
<<Java Class>> Circle paint.model
oval2D: Ellipse2D
Circle()
draw(Object):void
contains(double,double):boolean
<<Java Class>> Ellipse paint.model
oval2D: Ellipse2D
Ellipse()
draw(Object):void
contains(double,double):boolean
<<Java Interface>> Shape paint.model
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(Object):void
clone():Object
contains(double,double):boolean

Sample runs

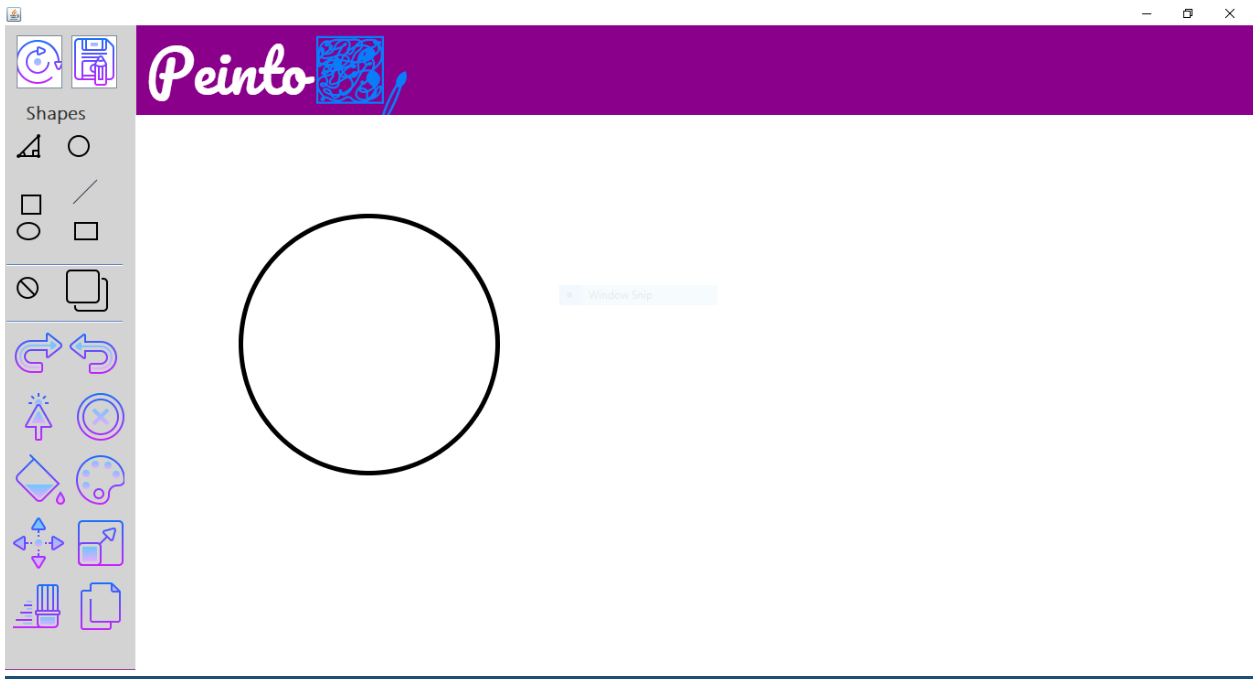


Figure 1: Drawing a Circle

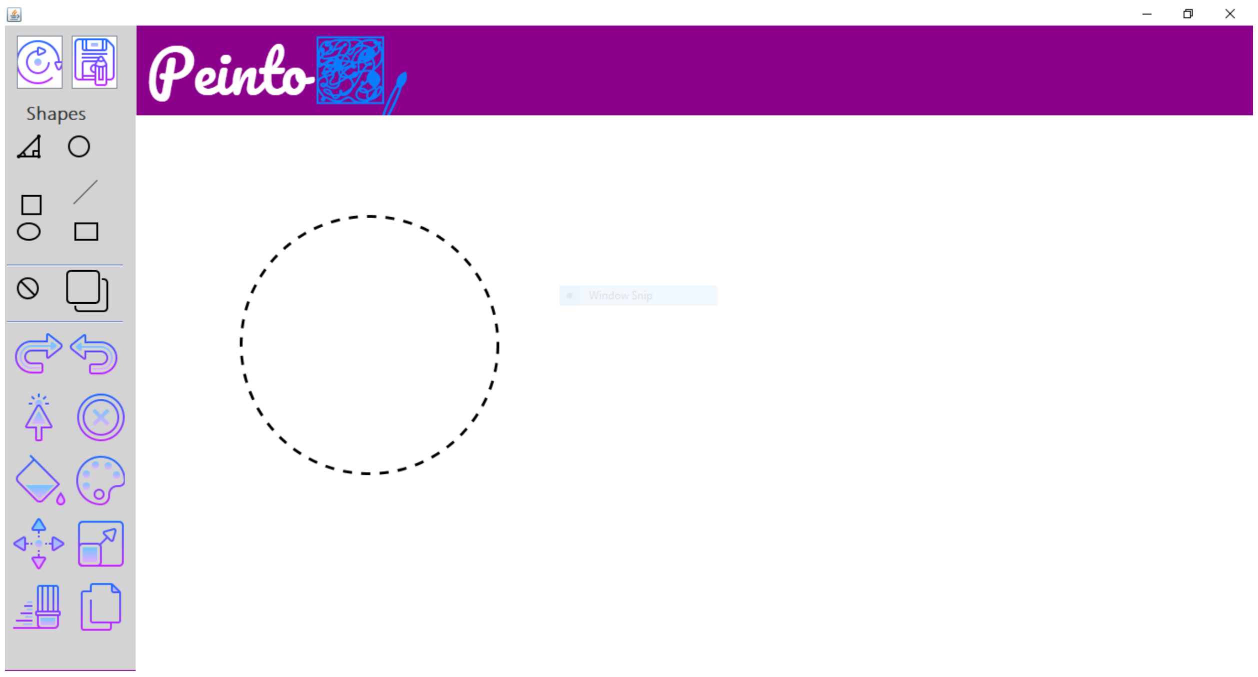


Figure 2: Selecting a circle.

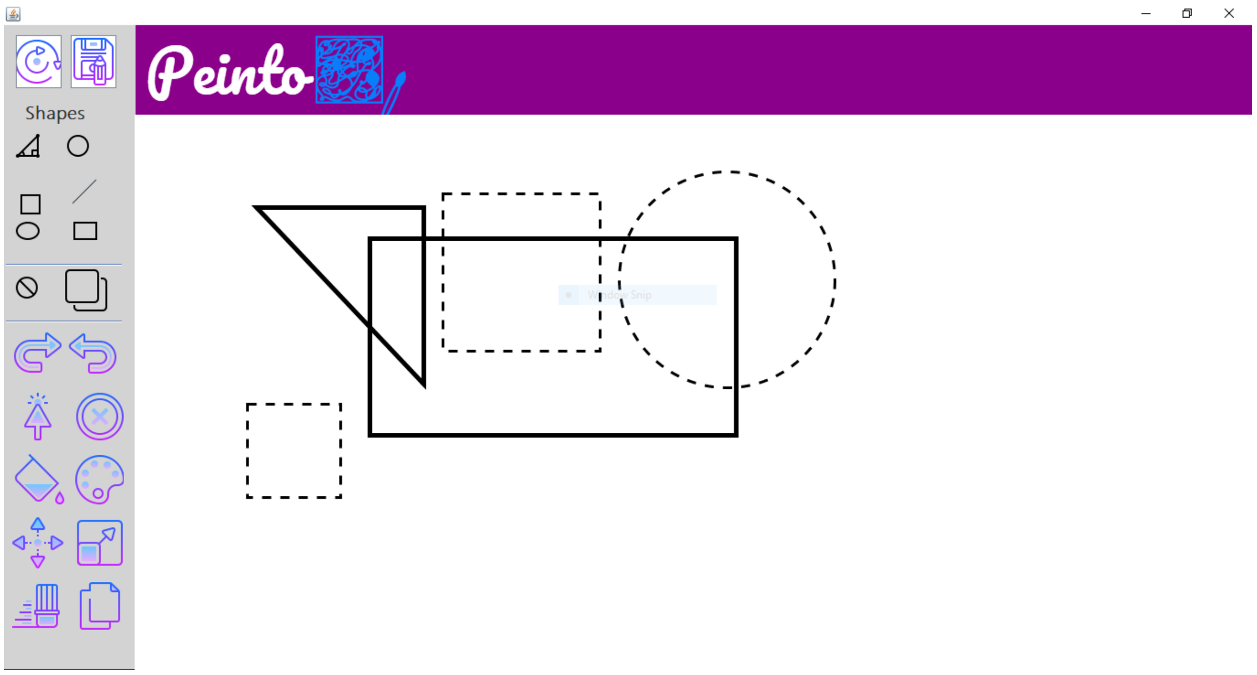


Figure 3: Selecting Multiple Shapes

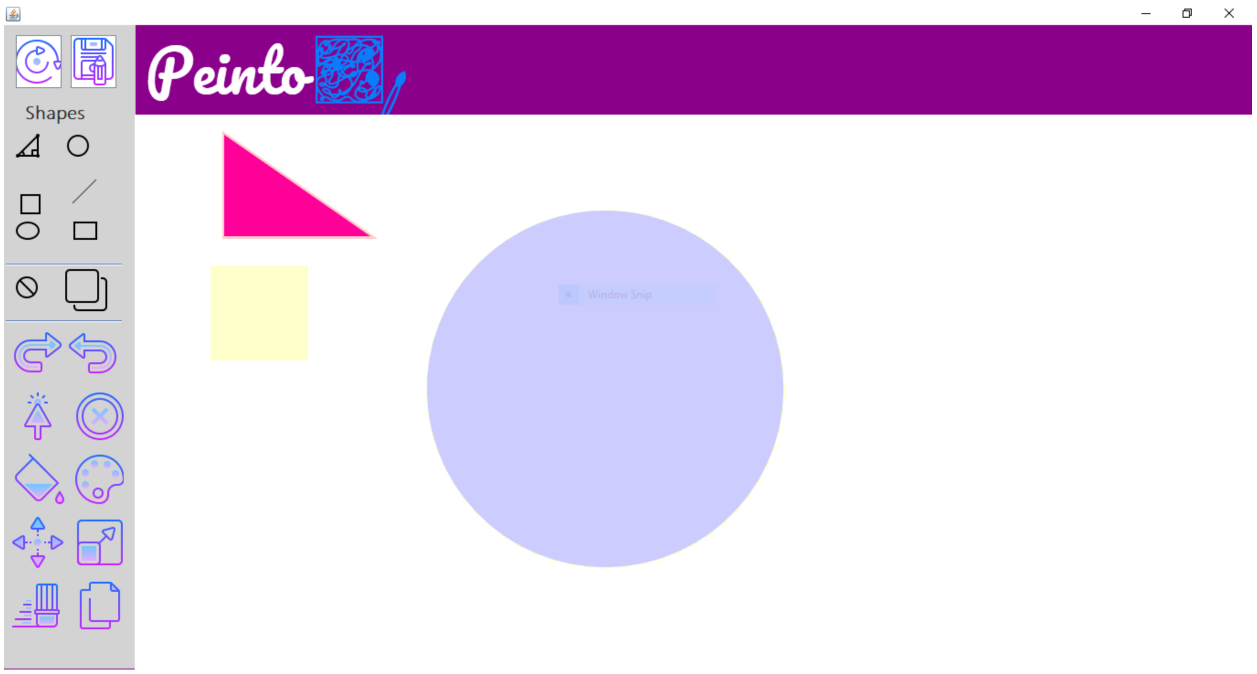


Figure 4: Drawing colored Shapes.

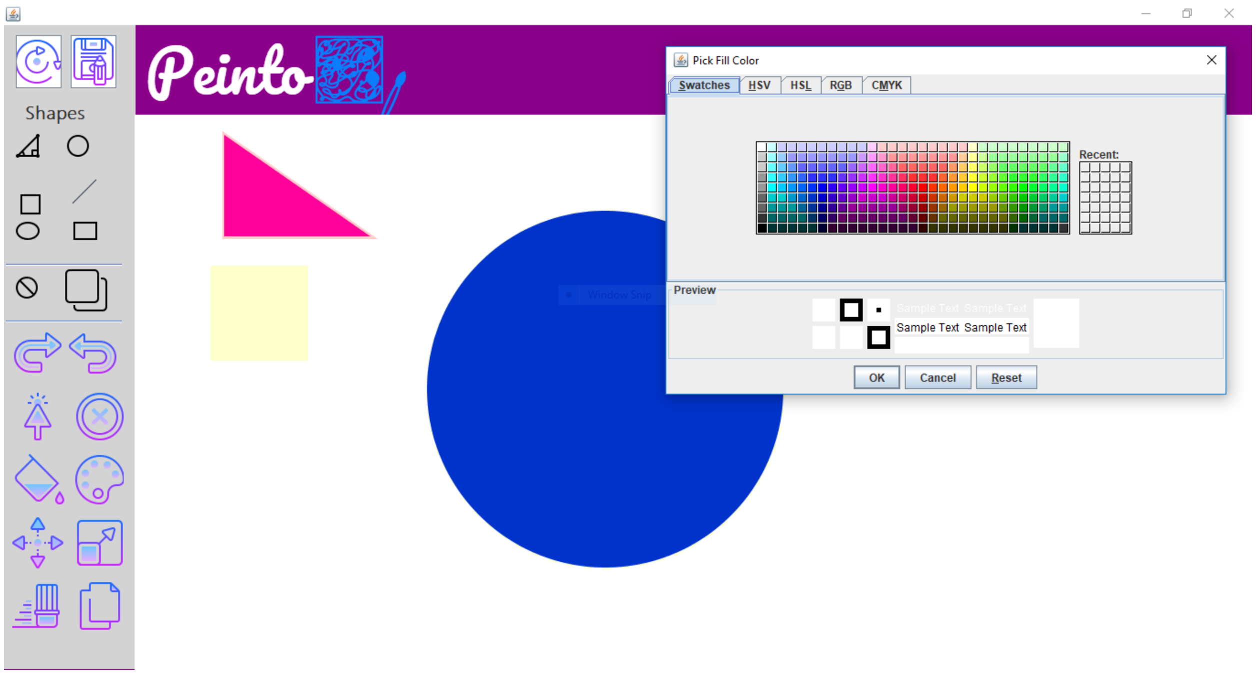


Figure 5: Changing color of selected shape.

References

Memento DP:
https://www.tutorialspoint.com/design_pattern/memento_pattern.htm

Command DP:
https://www.tutorialspoint.com/design_pattern/command_pattern.htm

JLabel mouse events for Drag and Drop:
https://stackoverflow.com/questions/5309150/jlabel-mouse-events-for-drag-and-drop/5312702#5312702

JSON: https://java2blog.com/jsonsimple-example-read-and-write-json/
https://www.geeksforgeeks.org/parse-json-java/

XML: https://www.youtube.com/watch?v=H-aTpt4NG-s