Vector Drawing App User Manual

بـــسم اللّه

Welcome to Our Modest Program User Manual

DESCRIPTION AND DESIGN

-This program supports 6 main shapes Triangle , Square , Rectangle , Circle , Ellipse and Line Segment.

-Singleton Design Pattern is applied

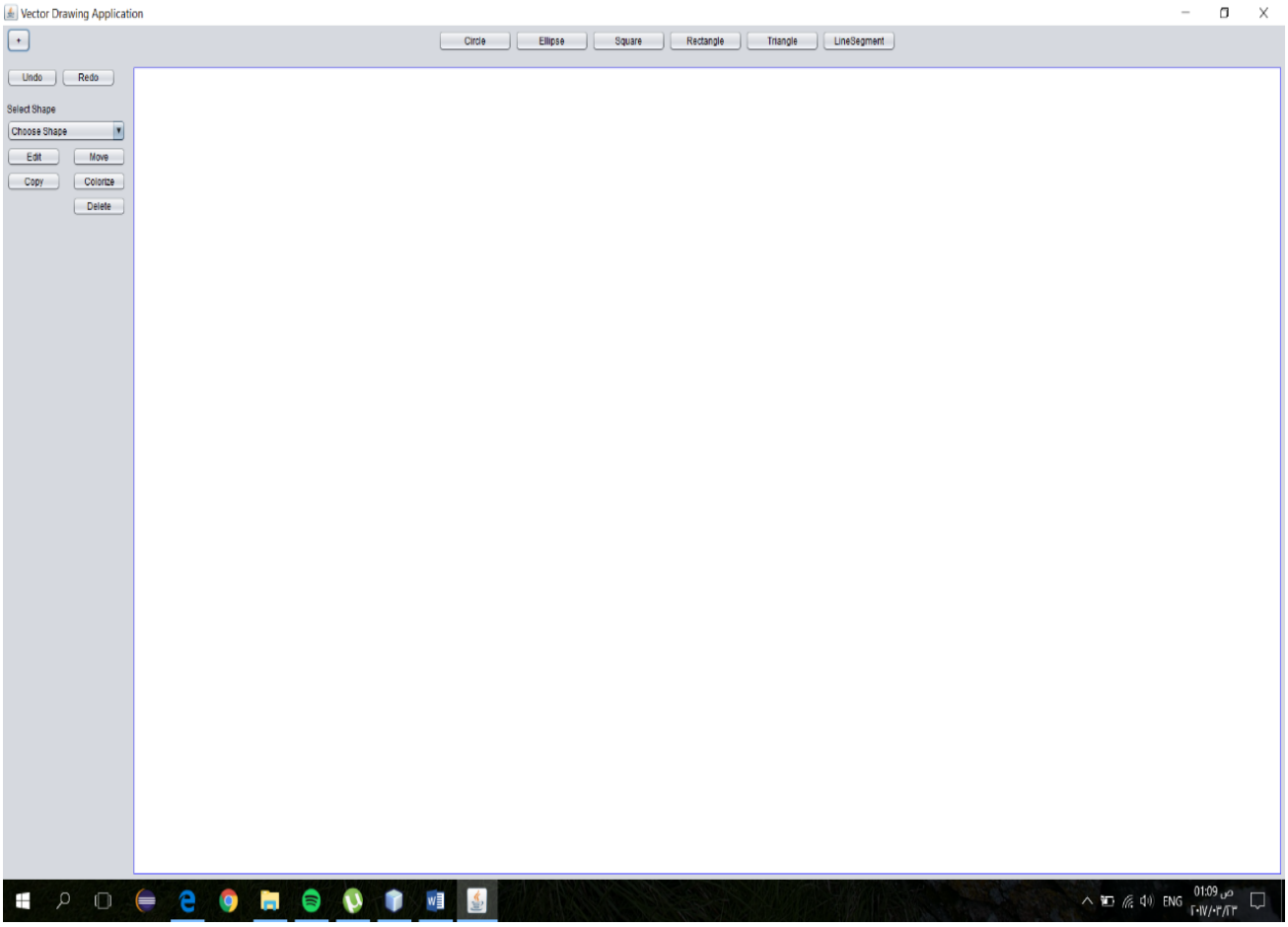
-Else there are extra shapes to be loaded in section 2

-Inheritance , Polymorphism, Abstraction encapsulation concepts are applied very well.

-Graphics 2D is nicely used to support neat Shapes .

-Program Window have fixed dimensions and cannot be maximized .

- Max X and Y Coordinates are stated so as not to get out of rage of the panel and to see nothing or a cut shape and if you do so.


Vector Drawing Application
Circle
Ellipse
Square
Rectangle
Triangle
LineSegment
Undo
Redo
Select Shape
Choose Shape
Edit
Move
Copy
Colorize
Delete
ENG
01:09 ص
٢٠٢٠/٣/١٢

ACKNOWLEDGMENT

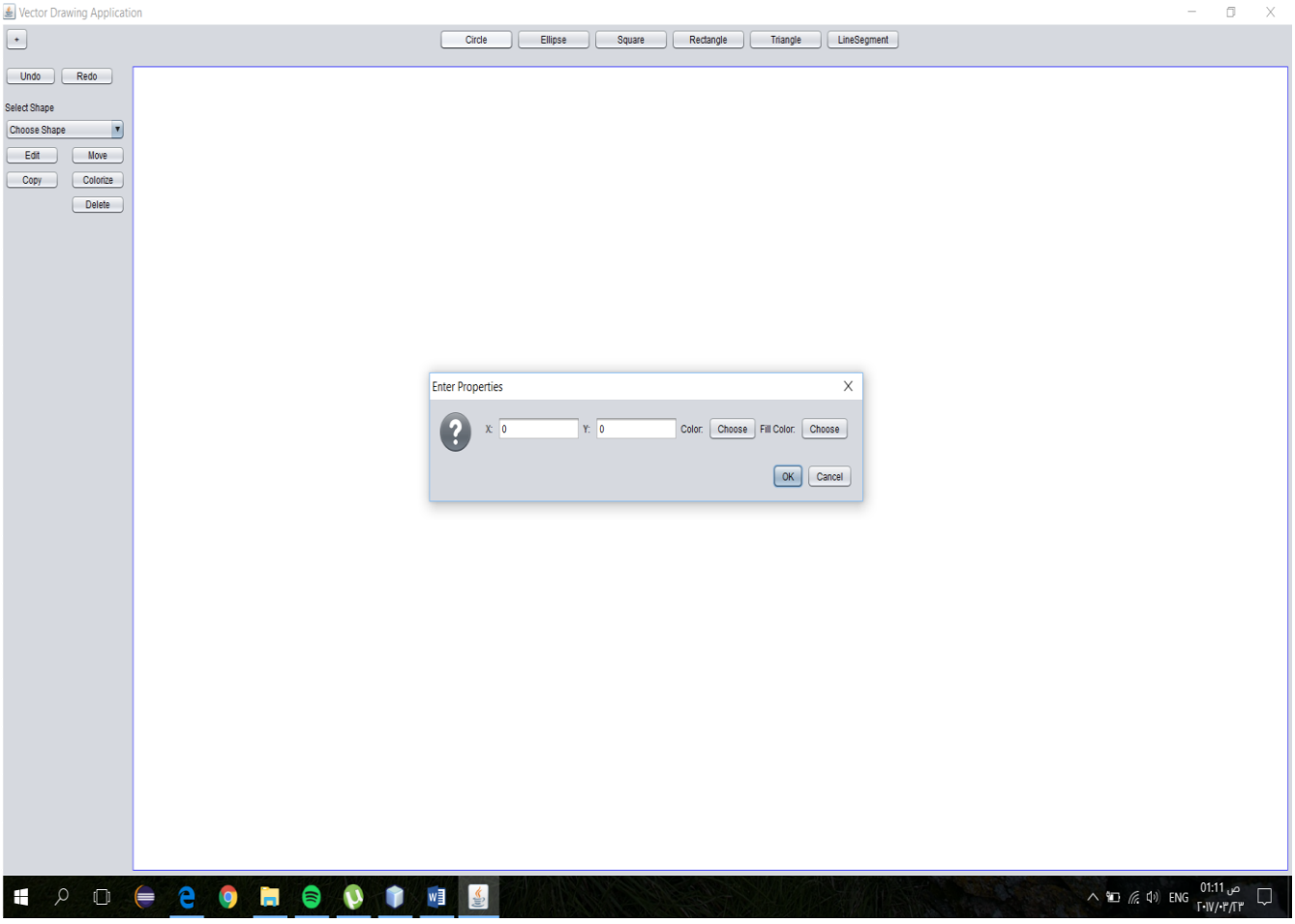
Some code parts like File chooser , color chooser , Class Loader , Dialogue and Plugin are made with the help of www.StackOverFlow.com

DESIGN AND FEATURES

SECTION 1 : Main Shapes

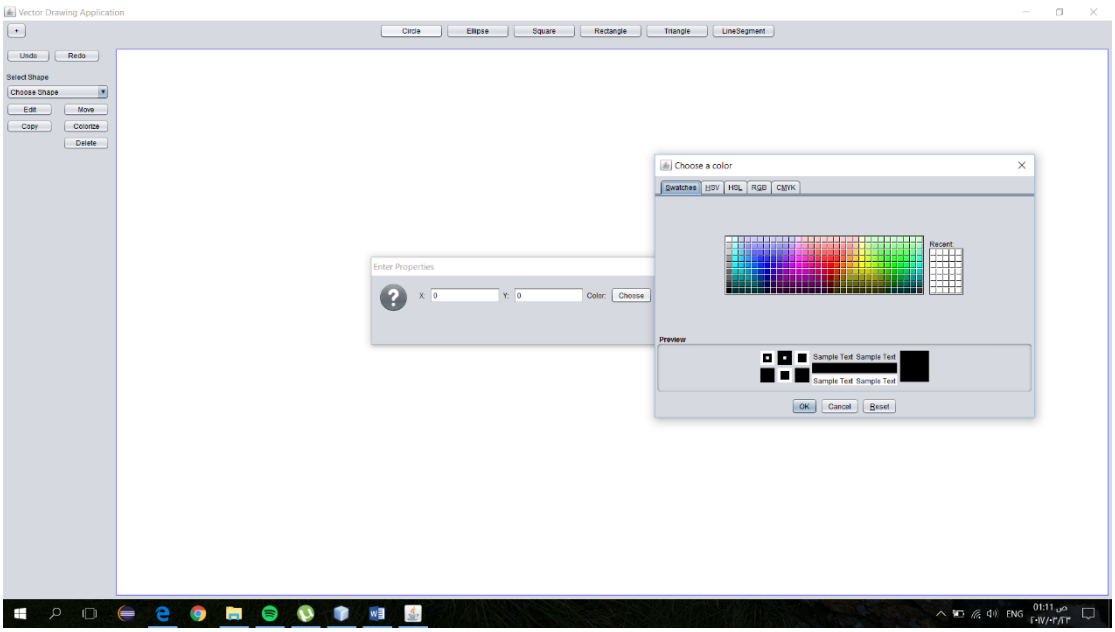
Circle

if you pressed "Circle" Button above , then a dialogue appears , you should indicate the Coordinates of your circle on the panel



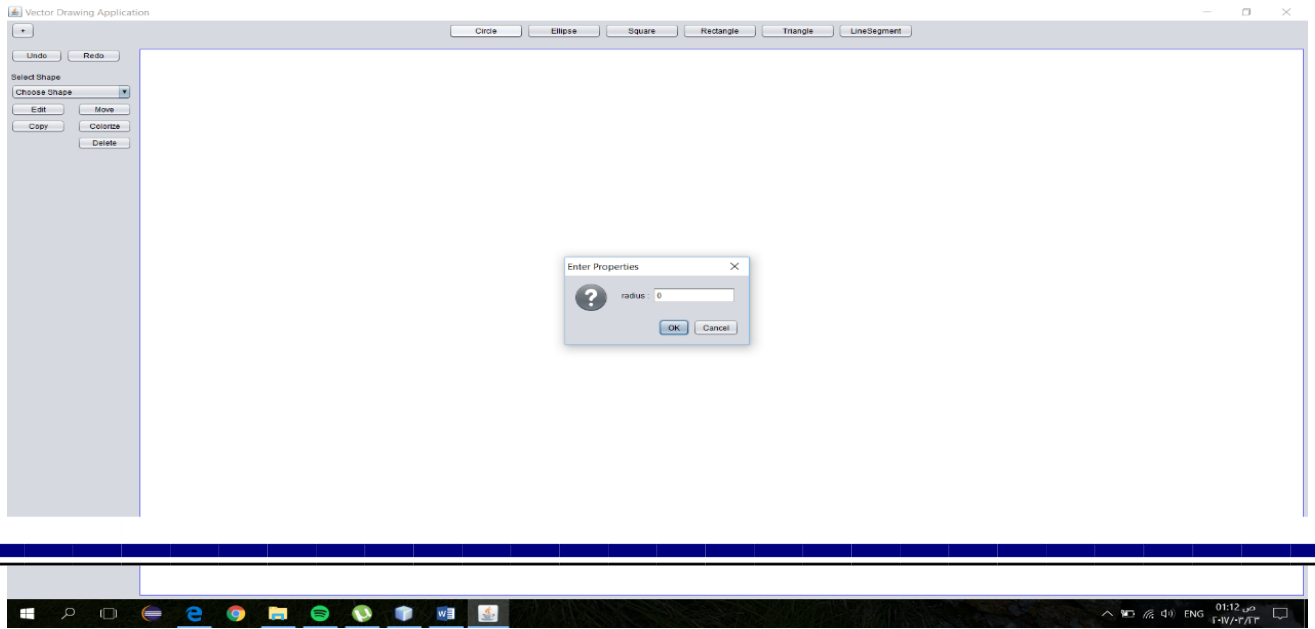
beside these fields , there is two buttons to set color and fill color that open a chart to choose whatever color you want

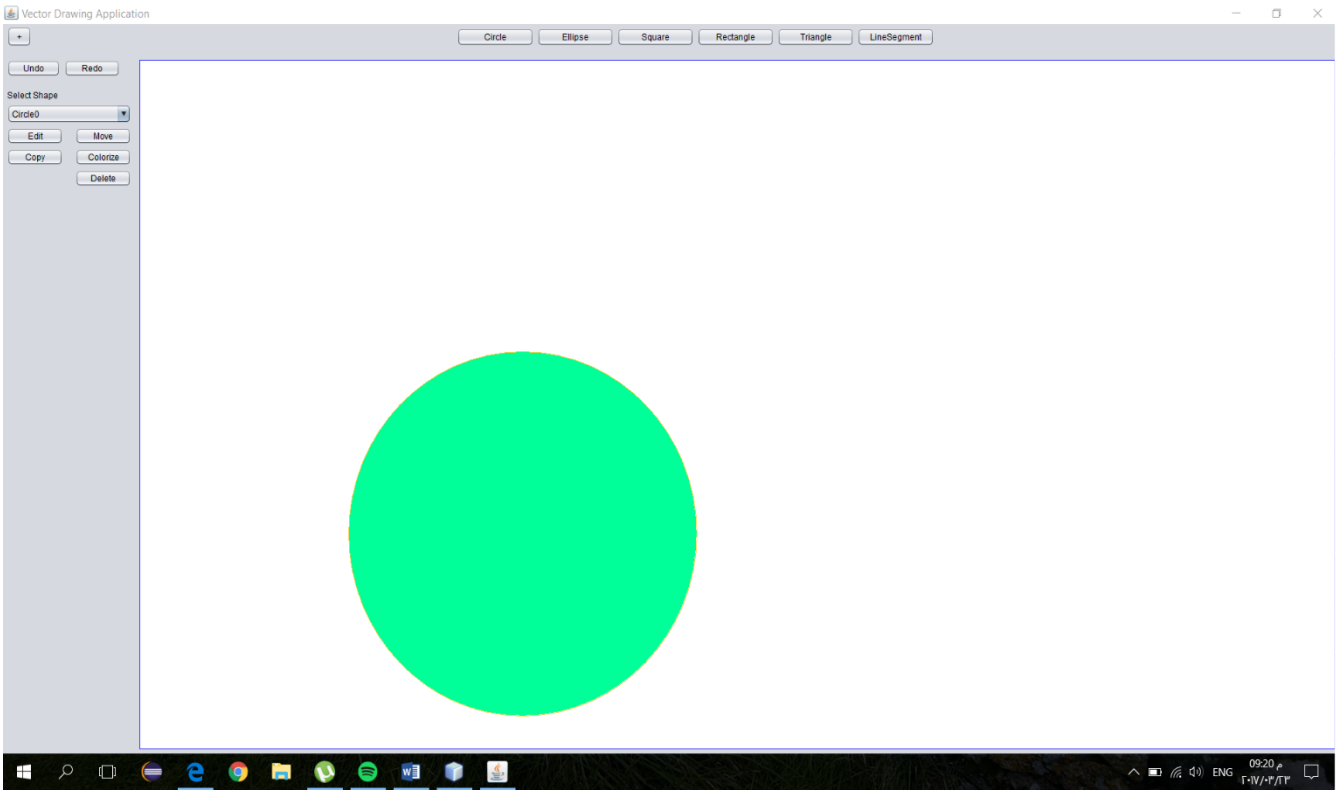
Default Color is Black and Default Fill Color is White



then , after pressing OK another dialogue appears inquires about the radius of the cirlce

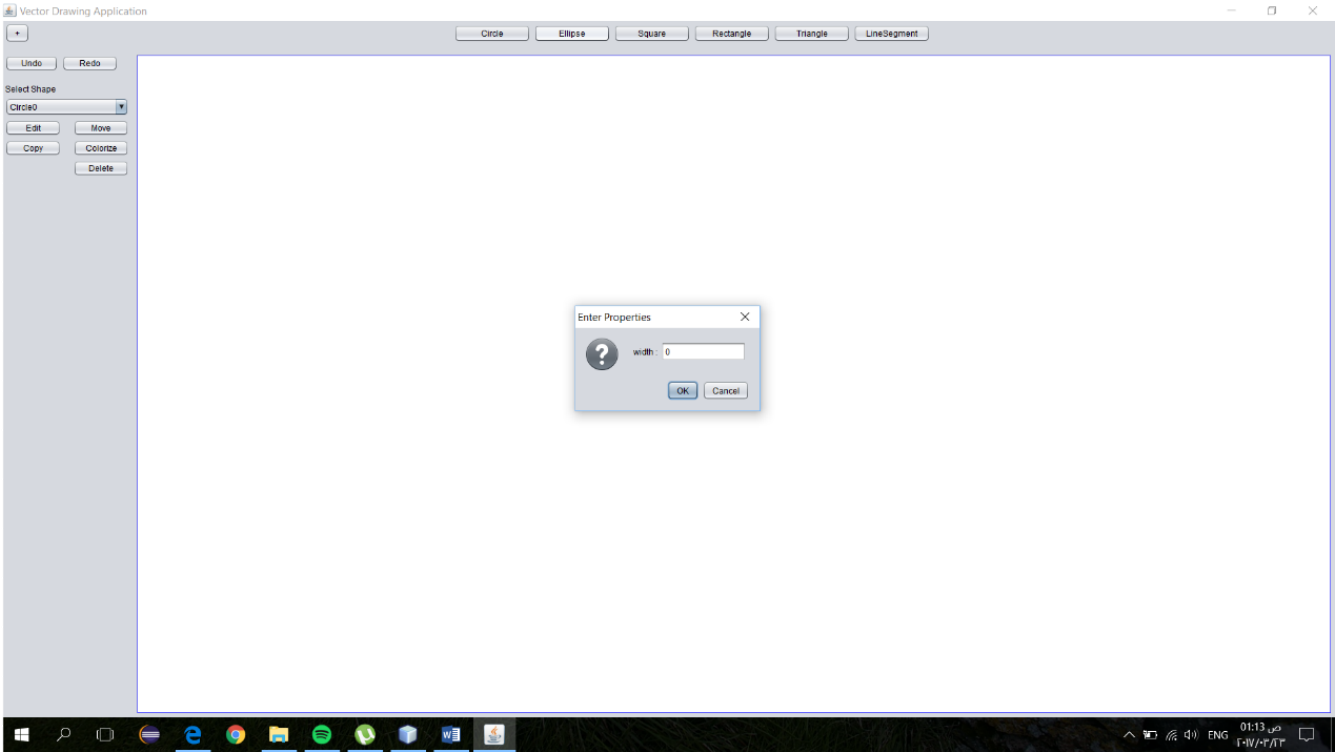
just press the last "OK" and Here is the circle




Vector Drawing Application
Circle
Ellipse
Square
Rectangle
Triangle
LineSegment
Undo
Redo
Select Shape
Circle0
Edit
Move
Copy
Colorize
Delete
ENG
09:20

Ellipse

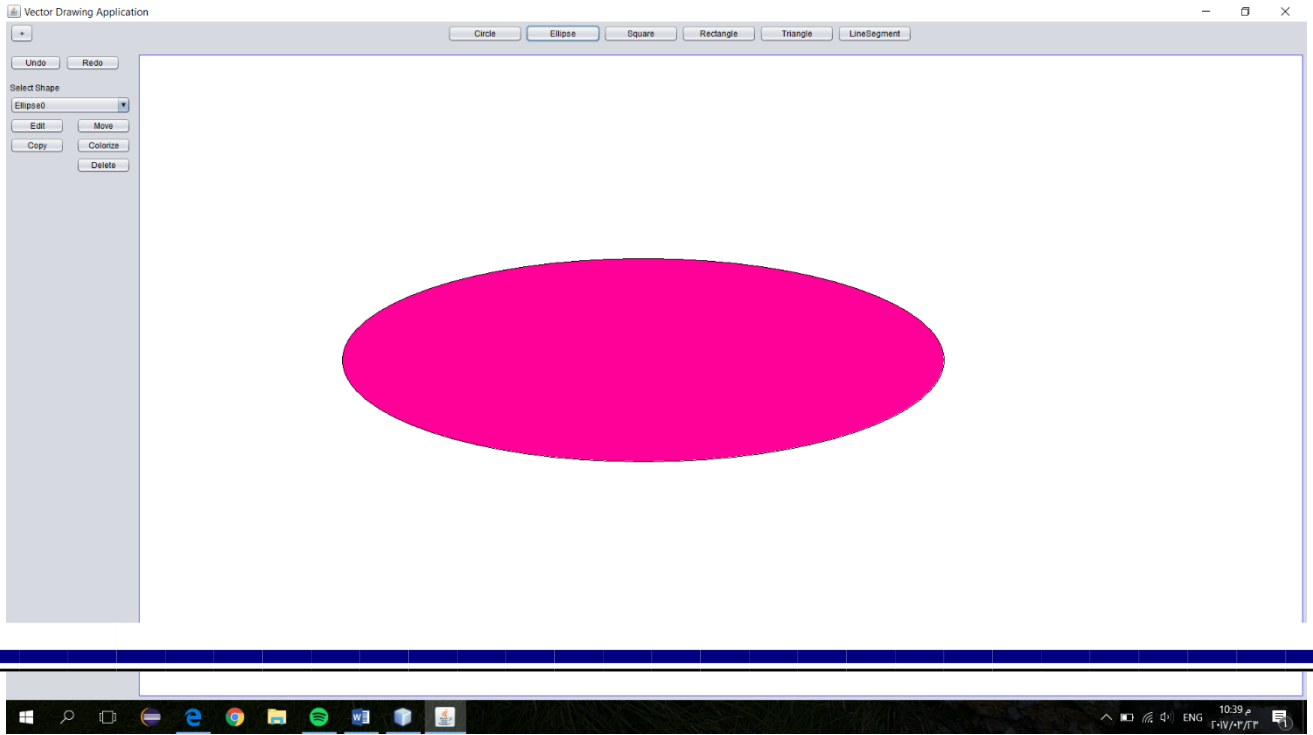
Same as the circle , we will enter X,Y but this time it's not a radius, it's Width & Height



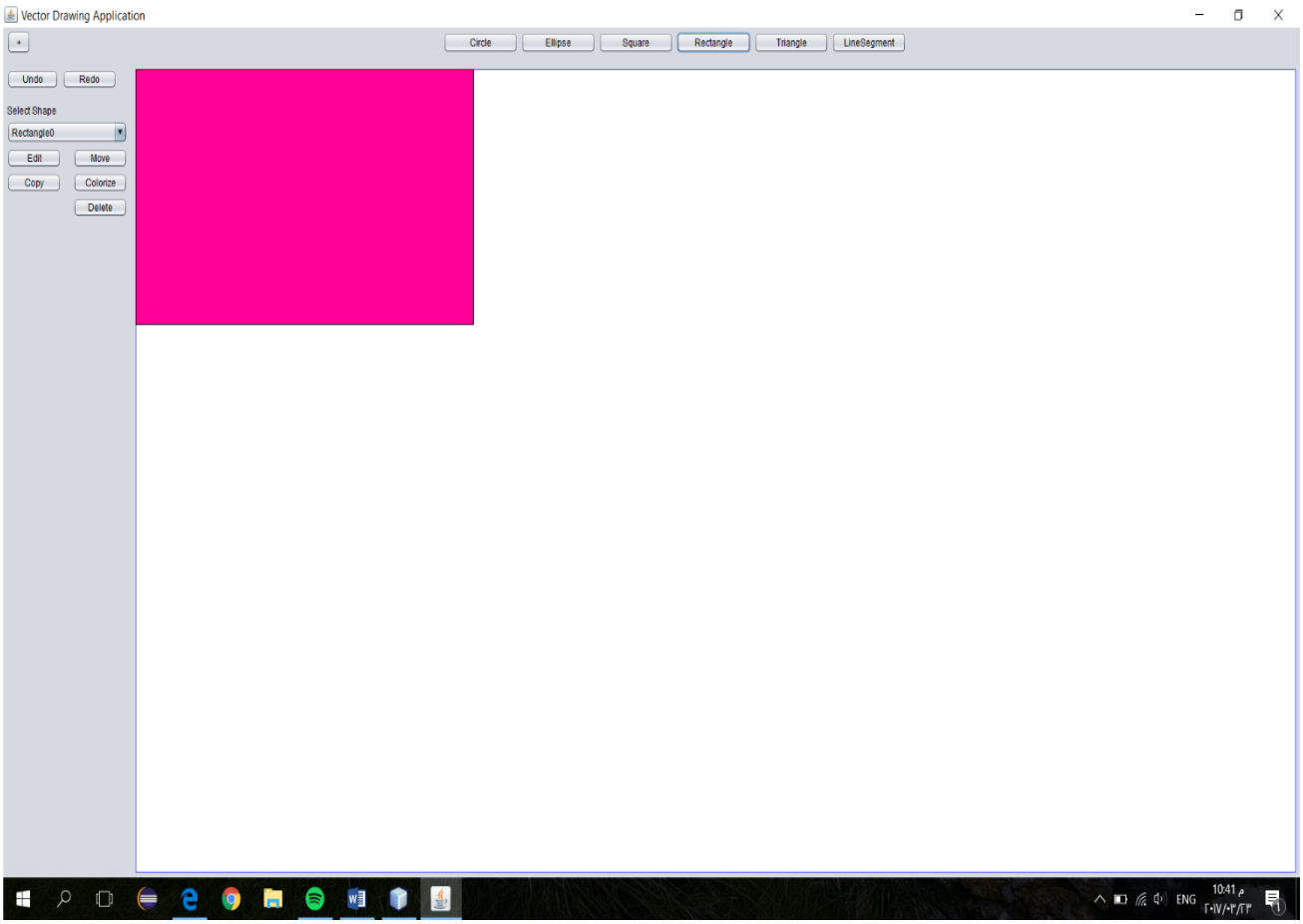
The Sample ..

Rectangle

This time , after X and Y , this shape takes two properties from you , Width and height .. You're free to set both properties with the

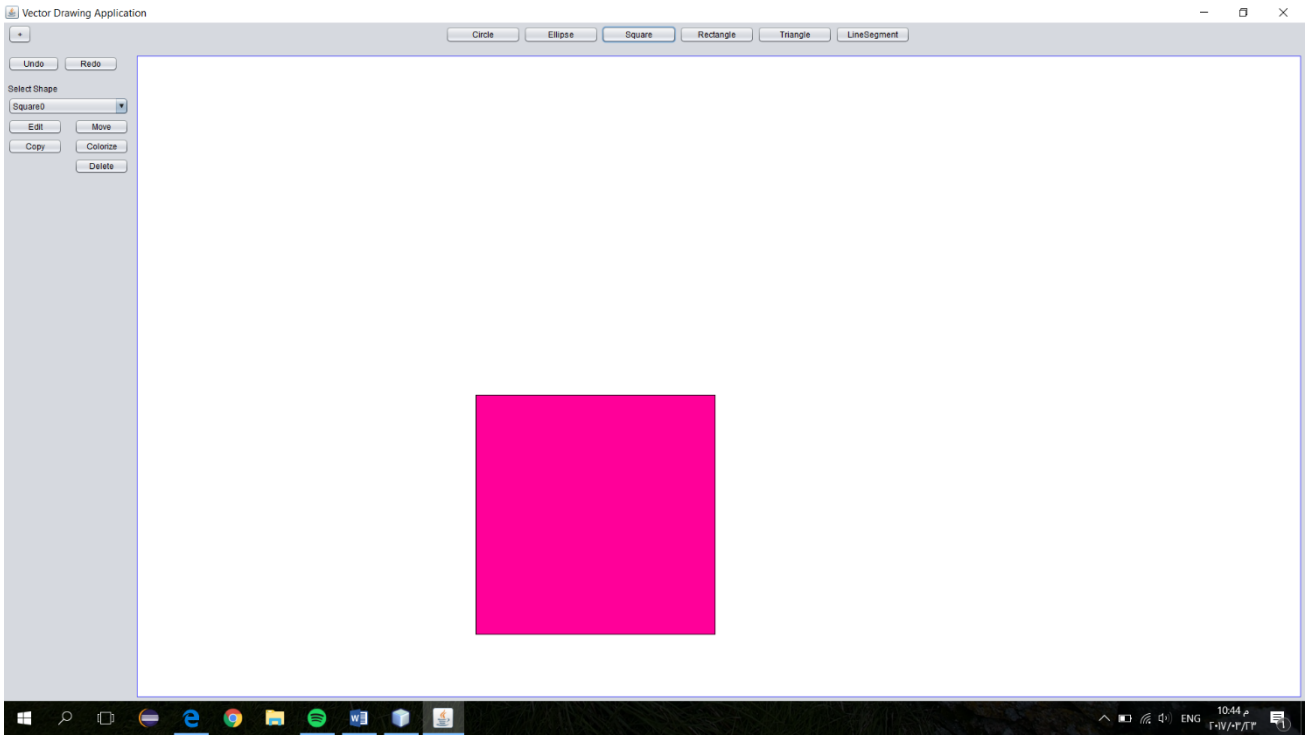


same value , it will display a square as the square is nothing but a side equal rectangle, Here is the sample ..



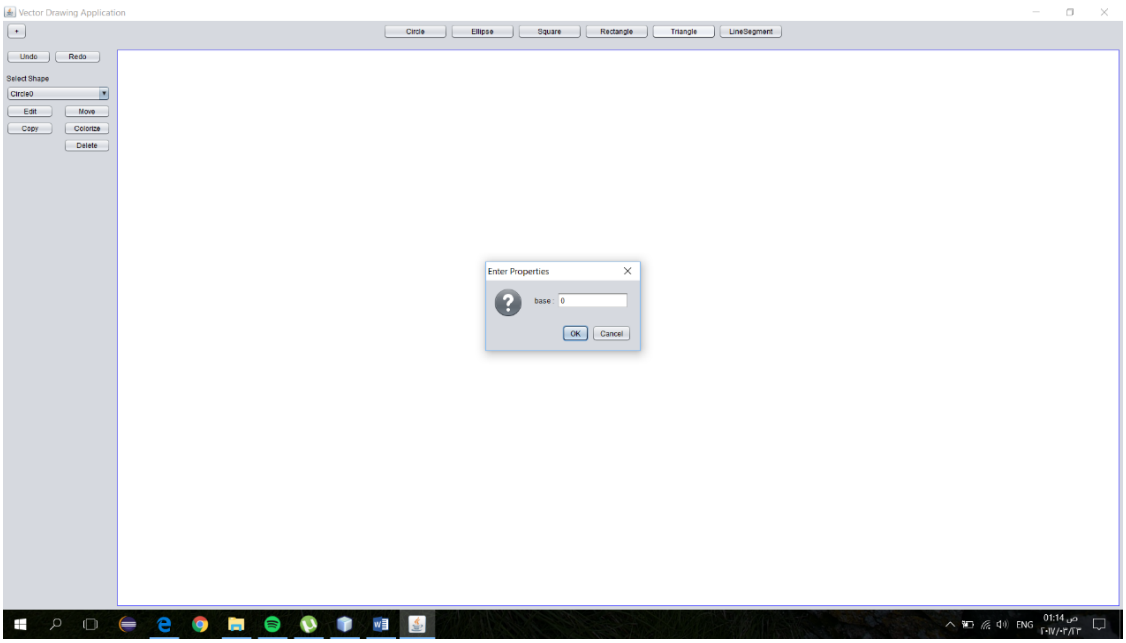
Square

As we said above , just one parameter 'Side' called Width

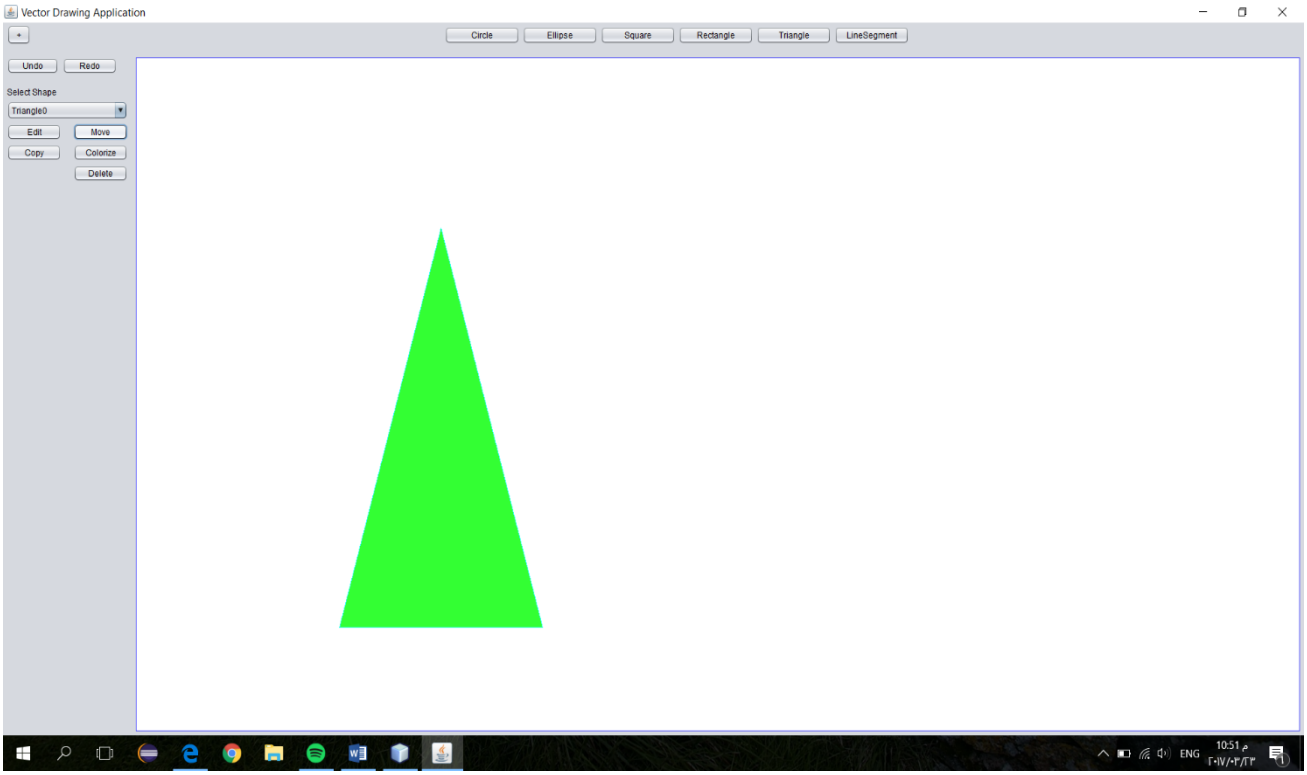


Triangle

The triangle is a polygon that have 3 vertices , regardless of that , you shall enter Base , and Height

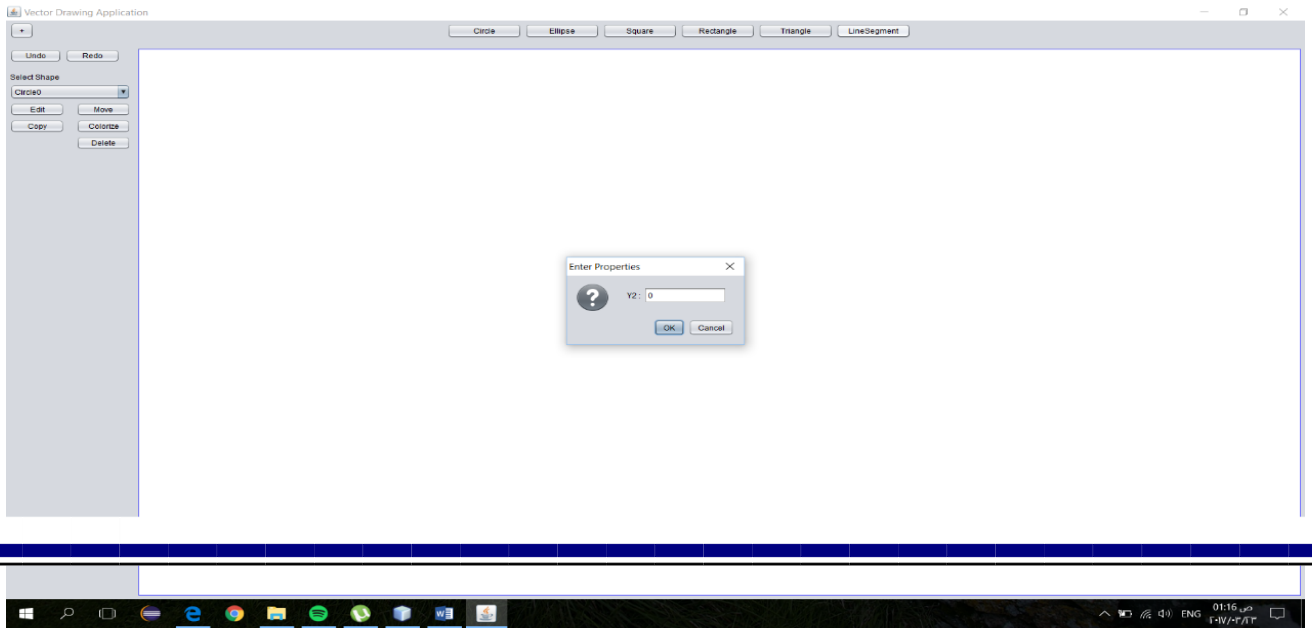


The sample ..



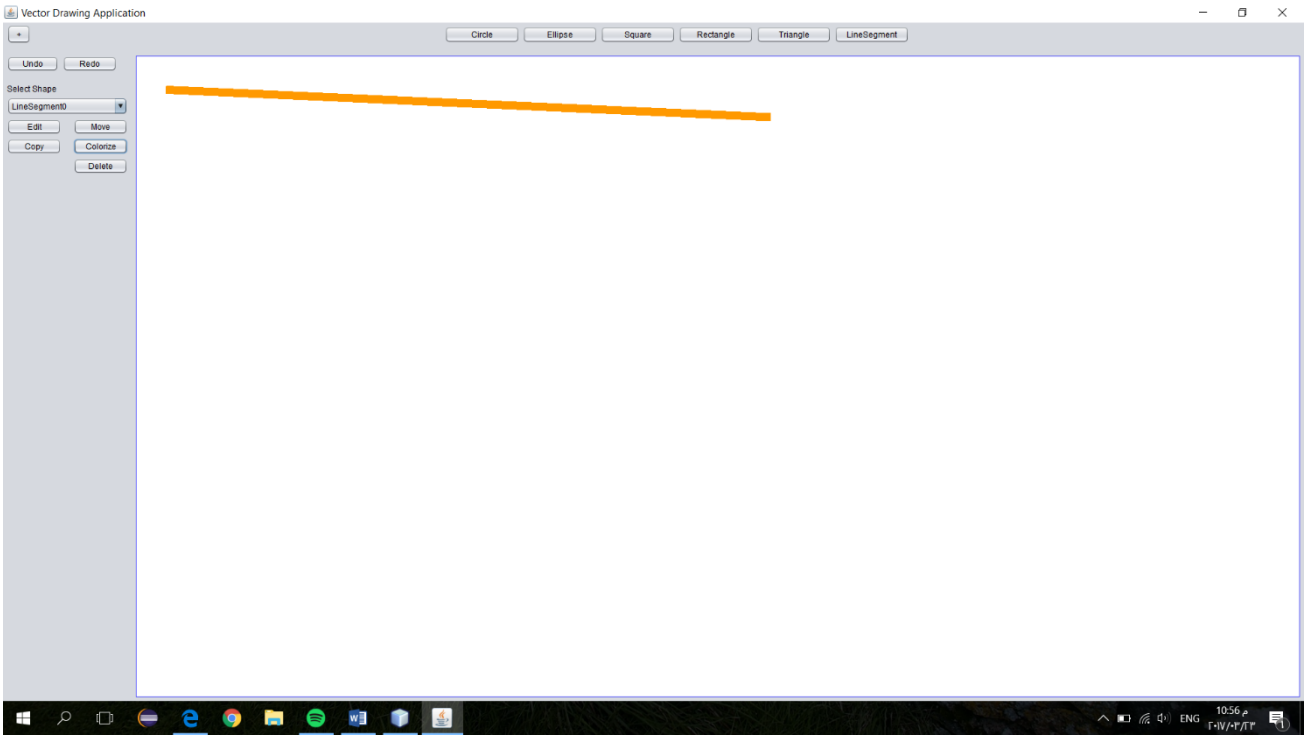
Line Segment

This one takes one point as position and you are supposed to



enter X2, Y2 point to draw the line in-between , then the line weight (Thickness)

The sample ..



THE FUNCTIONS :

Drop-Down menu

A list that contains all the shapes that have been created with an auto-incrementation to select the desired shape and to perform operations and changes

Delete

By selecting a shape drawn from the drop list menu on the left and pressing the delete button , it will disappear as you wished also it will be removed from the drop down menu … you can get it back from the undo button anyway

Colorize

This button is made specially to change the shape color or fill color to meet your desire , just press the button , the dialogue will show up once again with only color choises .

Edit

With this button you can change and re-assign values to selected shape properties like width , height , base or even line weight ,The Dialogue shows up to edit all properties but X and Y , **EXCEPT** for the line segment , it takes X2, Y2 once again , as shown

Move

This button will let you only change the X,Y Coordinates

Of the selected shape , **Except** for the line segment you are changing X,Y,X2,Y2.

Copy = Clone

-This choise will inquire a new position(X,Y) to make another copy from selected shape.

-Clone method implemented in every class of shape independently

Undo

-Stack of shapes Removes last drawn shape or restore it if it was deleted (getting back by a step) and removing from the drop-down menu , or re-inserting it again if it was removed recently.

-Limited for last 20 shapes only

-Removing a shape from the middle of the drop down menu then pressing Undo returns the shape in the tail of the drop down menu

Redo

-It's a Stack of shapes Re-drawing last removed shape whether by Undo or by the Delete function with insertion and deletion changes in the drop down menu

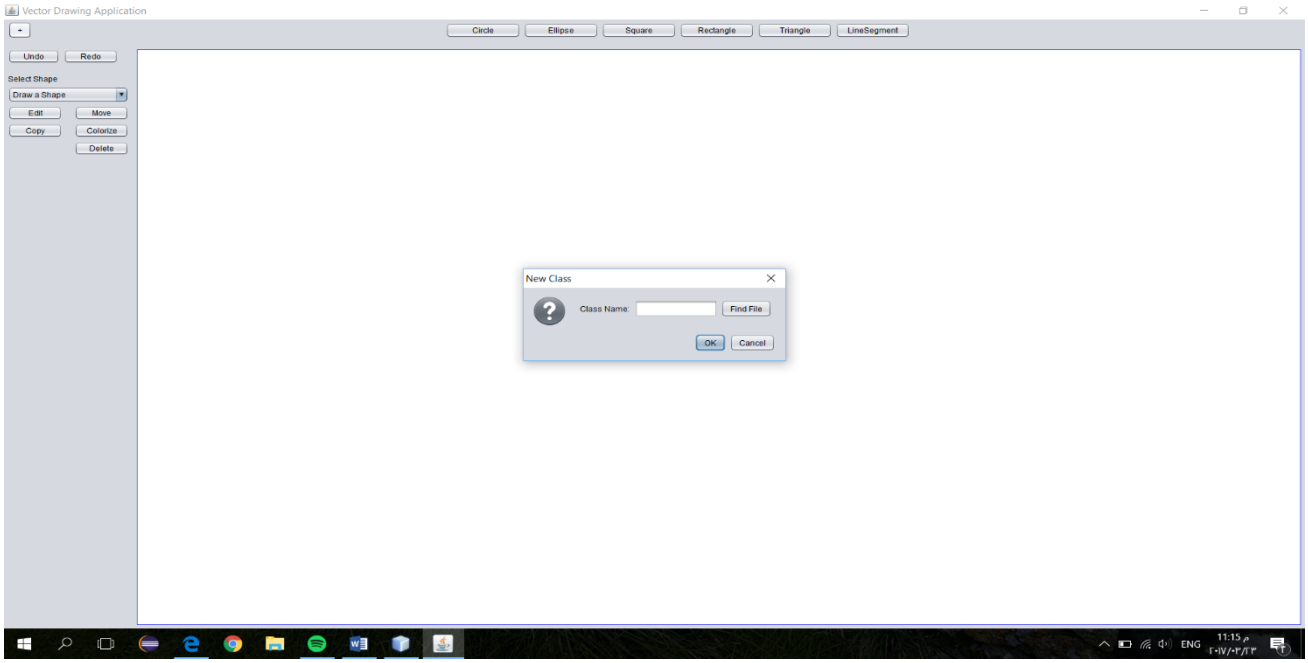
-Limited for last 20 shapes only

SECTION 2 : Plugins

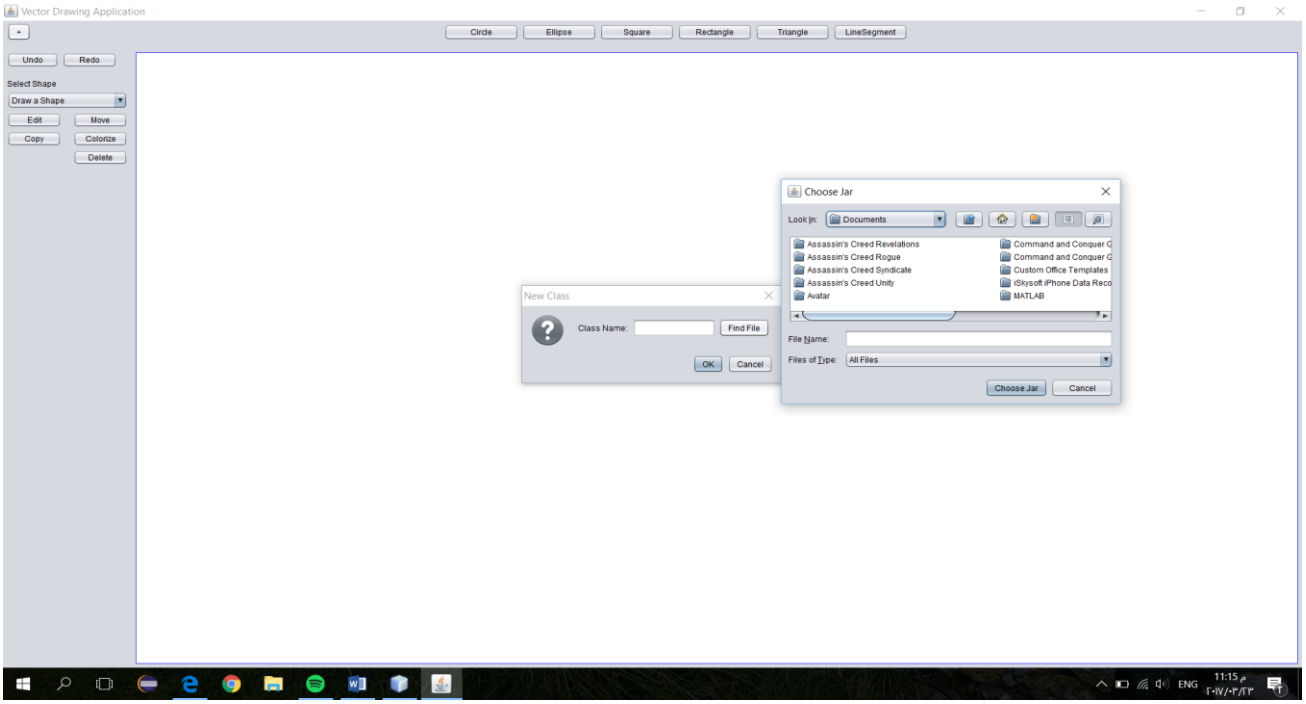
This is the plugin Part and how to load an external (.JAR) file in run time dynamically

Here we are using Star shape from a Jar file called Star.Jar and the class full name is plugin.Star

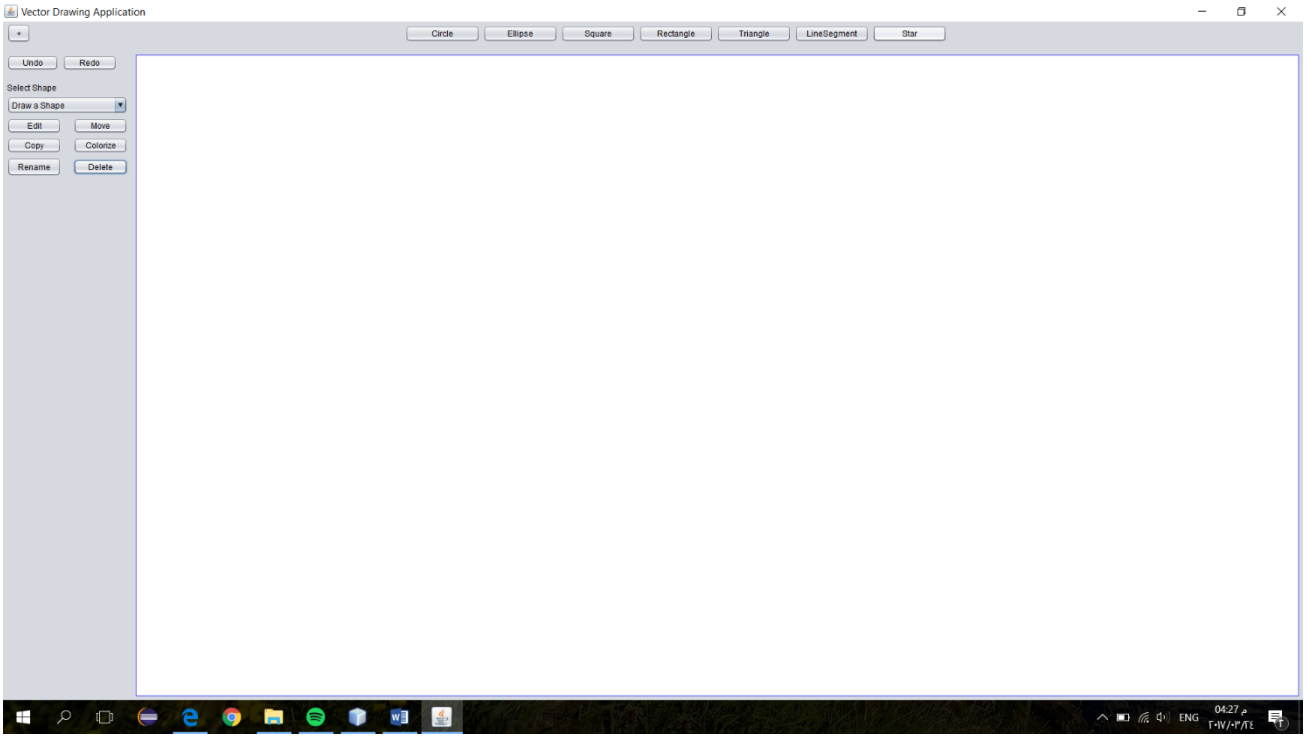
With the small '+' Icon on the very top left with a tooltip text to help you figure it out , click it , then the following dialogue



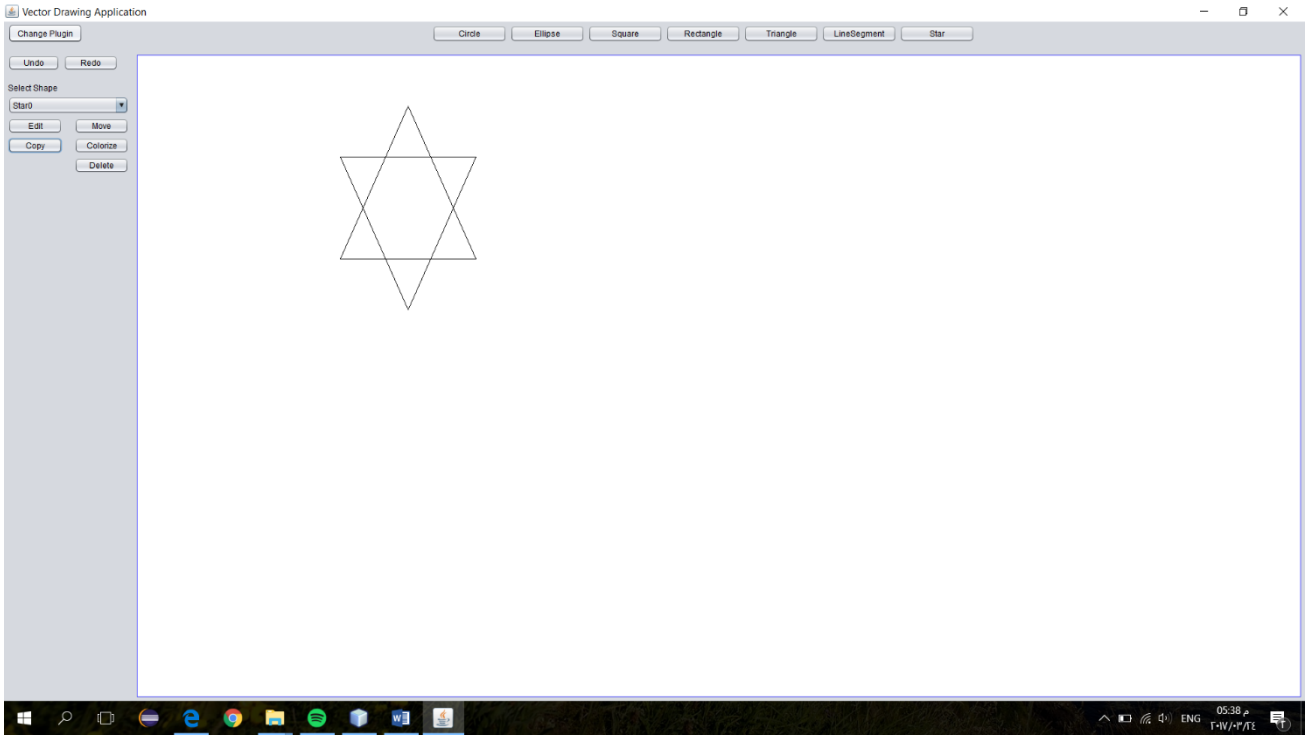
appears to find and locate the .JAR file …



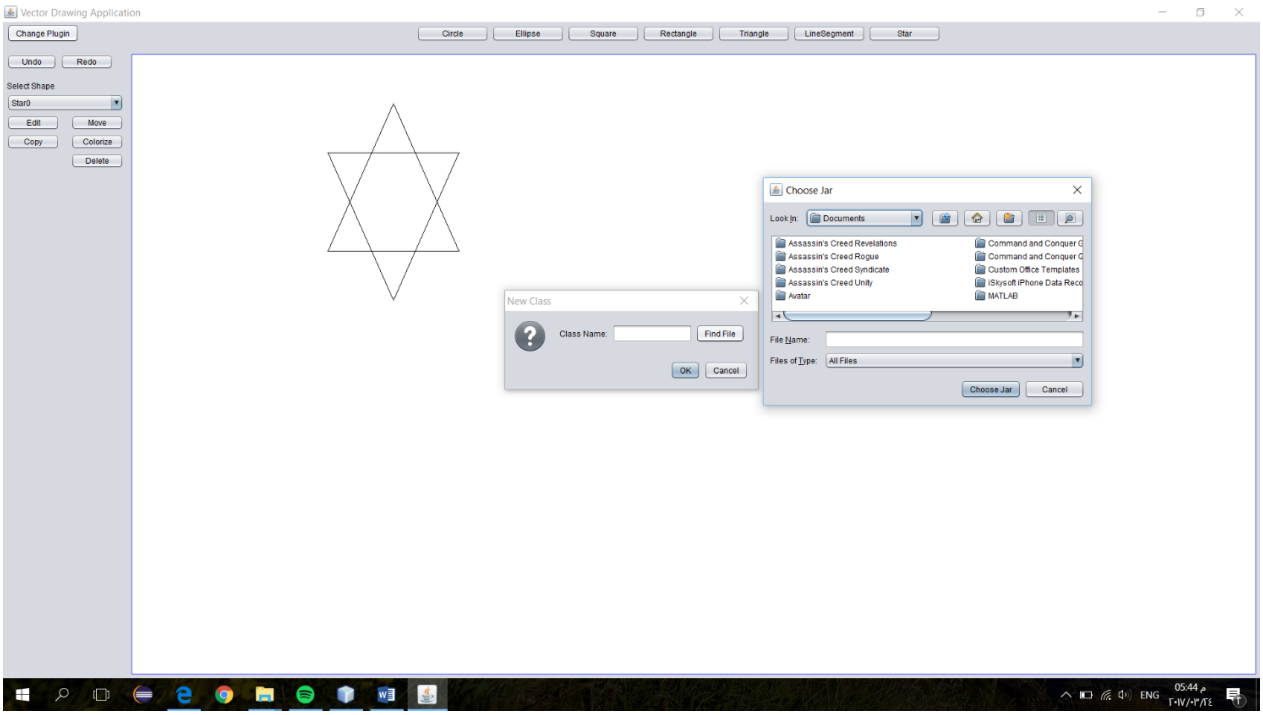
After locating it , enter in Class name field the package name followed by '.' followed by the desired shape class name and hit "OK" then a new button will appear carrying your new shape name as follows ..



Hit Star button then a dialogue will show up requesting X ,Y ,Color, Width and height .. Fill them with your preferred options then "OK"



This program support only one plugin , so if you want to change the plugin hit "Change Plugin" Button on the very top left, This Window will appear again to change it .



DESIGN DECISIONS & IMPORTANT MODULES

```
private Actions() {

    shapesList = new Stack();
    recentlyUndone = new Stack();
    command = new Stack();
    undoCommand = new Stack();
    classList = new Stack();

    classList.add(new Circle().getClass());
    classList.add(new Ellipse().getClass());
    classList.add(new Square().getClass());
    classList.add(new Rectangle().getClass());
    classList.add(new Triangle().getClass());
    classList.add(new LineSegment().getClass());
}
```

In this class called Actions , we made the default constructor

To declare 5 stacks, 2 String stacks and 3 Shape stacks

1-classList is a stack of shapes that contains supported shape in the program that adds by default all the 6 main shapes in it.

2-shapesList stack of shapes that control the drawn shapes and contain them.

3-recentlyUndone stack of shapes that contain removed shapes before adding them again by Undo or Redo.

4-command a stack of Strings that carries whether "Draw" for drawn shapes or "Delete" for removed shapes to help handling the Undo and Redo

5-undoCommand stack of String that carries same as command stack and used In undo and redo only.

```
@Override
public void installPluginShape(Class<? extends Shape> shapeClass) {
    if (classList.size() == 7) {
        classList.remove(6);
    }
    classList.add(shapeClass);
}
```

installPluginShape method is the one implemented in the interface and responsible for adding extra shapes by dynamic loading in runtime. This method allows only one plugin shape with replacement.in another words, it checks if the classList stack have less than 7 elements then it adds one new shape with new button

if it's equal to 7 AKA. You have 7 buttons already , then the new plugin shape will be installed and replace the old plugin button .

****

```
public abstract class AbstractShape implements Shape {

    protected Point position;
    protected Color color;
    protected Color fillColor;
```

This is the abstract Class that all the shapes inherit its basic attributes from like position, color, and the fillColor with their setters and getters .. as you see the identifier is Protected to support the inheritance of these fields.

```
@Override
public void undo() {

    if (command.size() == 20) {
        Stack<String> temp = new Stack();
        for (int i = 0; i < 19; i++) {
            temp.push(command.pop());
        }
        command.clear();
        for (int i = 0; i < 19; i++) {
            command.push(temp.pop());
        }
    }
    if (command.peek().equals("draw")) {
        recentlyUndone.push(shapesList.peek());
        shapesList.pop();

        undoCommand.push("draw");
    } else {
        shapesList.push(recentlyUndone.peek());
        recentlyUndone.pop();

        undoCommand.push("delete");
    }
    command.pop();
}
```

Here in the **Undo** method, there is an extra declaration of a temp Stack used to manage

Undo operations and to limit them by 20 action.

Checks first if the command String Stack have more than 20 action in it. If Yes then the last 20 actions will be Pushed in the temp Stack. Then the original command stack is Cleared. After that, The last 20

Actions will be pushed once again from temp stack to command stack, then you have your only 20 last actions.

Consequently any additional action will be added as last action and the action #21 is gone. If the elements in command are less than 20 , then it will check the first string whether it's delete then it will redraw it and re-insert it in the drop-down menu and push the shape once again through shapesList, will remove it from recentlyUndone stack and will push "delete" string through undoCommand stack … OR it's draw then it will delete it, remove it from shapesList stack and from the drop list menu and push it through recentlyUndone stack also it will push "draw" string through undoCommand .At last it will remove the executed command form command stack.

```
@Override
public void redo() {

    if (undoCommand.peek().equals("draw")) {
        shapesList.push(recentlyUndone.peek());
        recentlyUndone.pop();

        command.push("draw");

    } else {
        recentlyUndone.push(shapesList.peek());
        shapesList.pop();

        command.push("delete");
    }
    undoCommand.pop();
}
```

redo method is dealing with the **Undo** method mainly , as it checks IF the last action in undoCommand was "draw" then it will push in shapeList stack the last shape (top) of recentlyUndone stack , then pop this shape from recentlyUndone to remove it , then it will re-push string "draw" in command stack

OR otherwise it will be a "delete" action so , it will push in recentlyUndone stack the top of shapesList stack , then it will remove it from shapesList Stack, and it will re-push string "delete" in command stack .. After executing this logic it will remove this command from the undoCommand stack.

```
private void refreshSupportedShapes() {
    btnsPanel.removeAll();
    JButton[] shapeBtns = new javax.swing.JButton[Actions.getInstance().getSupportedShapes().size()];
    btnsPanel.setLayout(new FlowLayout());
    for (int i = 0; i < Actions.getInstance().getSupportedShapes().size(); i++) {
        shapeBtns[i] = new javax.swing.JButton();
        shapeBtns[i].setText(Actions.getInstance().getSupportedShapes().get(i).getSimpleName());
        shapeBtns[i].setVisible(true);
        shapeBtns[i].setPreferredSize(new java.awt.Dimension(110, 25));
        btnsPanel.add(shapeBtns[i]);
```

This method exists in the GUI class , it loops on supported shapes using size of **getSupportedShapes** method and create their buttons and launch their listeners, this method is called twice , one at the activation of the dynamic loading session and the other on the constructor of the GUI class .

****

private boolean ***showDialog****(Map<String, Double> properties, Shape shape) throws NumberFormatException {…}*

showDialog() loops on properties (Hash Map) by its size And take them from user which is X, Y , Color , Fill Color, and Special shape properties like width , height , weight …etc.

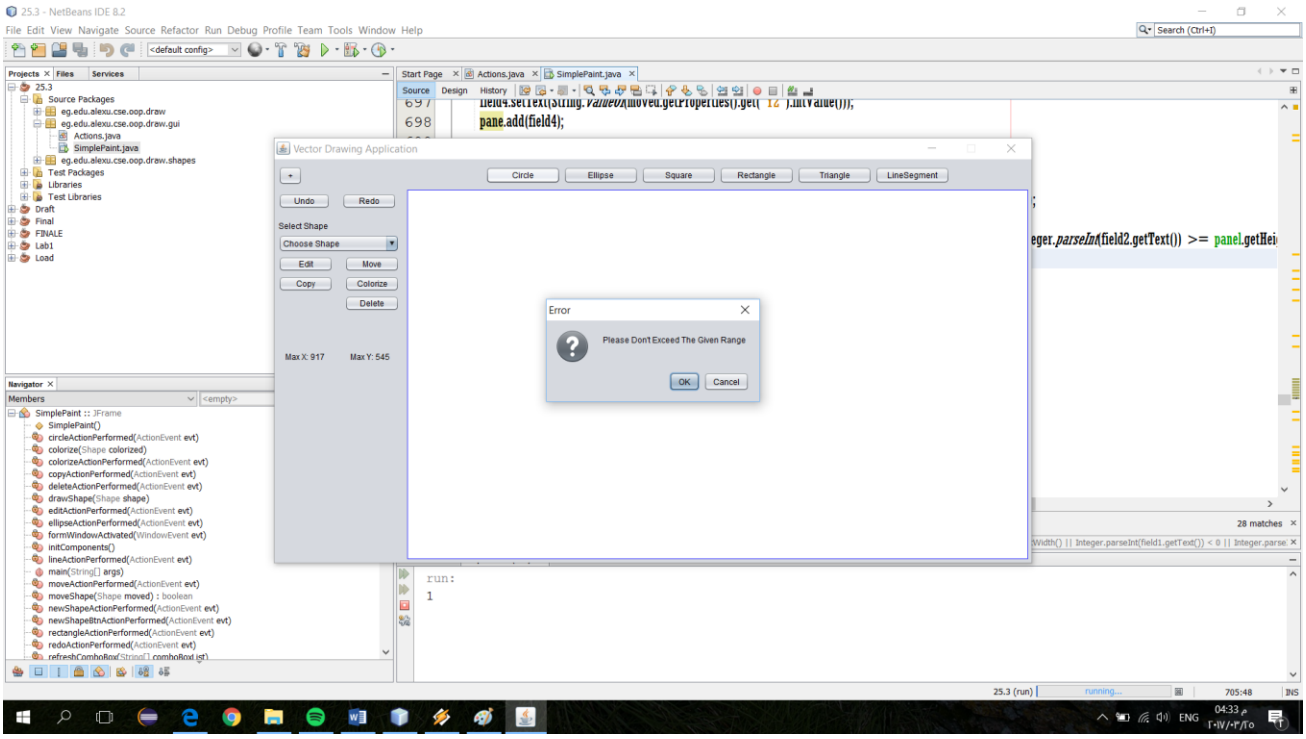
It's the responsible of the Input that the program takes and deal with user.

```
private void showError(String msg) {

    JPanel pane = new JPanel();
    JLabel errorLbl = new JLabel(msg);
    pane.add(errorLbl);

    JOptionPane.showConfirmDialog(this, pane, "Error", JOptionPane.OK_CANCEL_OPTION);
}
```

This is another method existed in the GUI class that control some situations that could happen and cause Errors in runtime with some users for example entering Very Big X,Y values then this dialogue shall appear saying "Please Don't Exceed The Given Range".



Made By /

1- Yousuf Sherif Sabry 4319
2- Shams Sherif 4079
3- Nour El-Din Moustafa 3784