CAB302 Assignment 2 Vector Design Tool – Team188

Subject Name: Software Development

Unit ID: CAB302

Unit Coordinator: Dr Timothy Chappell

Group members: Johnson KaiZhi Foo (N9915931)
Wei Zhao Wu (N10192701)
Yung Han Lin (N10094881)
MinHo Kim (N8051381)

Due Date: 02/06/2019 11:59p.m.

Table of Contents

Completeness of Functionality

In this paragraph, all the basic function and additional function will be explained here. There are two section in order to clearly explain the functionality for the users. Basic Functionality shows all the basic function like draw, colour and save file, etc. Additional function shows all the extra features for the project such as grid and multi-image support, etc.

Basic Functionality

I. Draw

```
// common shape properties
private Color color;
private Color fill;
private PenType type;
private Graphics graphics;
private int penSize;
private int beginX, beginY, endX, endY;
```

Above shows the properties for the drawing part.

1. Point

```
// Method for drawing a point
private void DrawPoint(){
    graphics.setColor(color);
    graphics.drawOval(beginX, beginY,penSize,penSize);
    graphics.fillOval(beginX,beginY,penSize,penSize);
}
```

```
// method for creating dot pen
public Pen(Graphics whichGraphics, PenType whichType, int Size, Color color, int x, int y){
    this.beginX = x;
    this.beginY = y;
    this.endX = x;
    this.endY = y;
    this.color = color;
    this.penSize = Size;
    this.type = whichType;
    this.graphics = whichGraphics;
    Draw();
}
```

The function of draw point is to draw a dot shape on the page screen. Above screenshot shows the Java code of the dot point. The first screenshot is the code of that able to draw the dot on the drawing screen. The second screenshot is to create the pen for the dot point to draw.

2. Line

```
// method for creating line pen
public Pen(Graphics whichGraphics, PenType whichType, int Size, Color color, int beginX, int beginY, int endX, int endY){
    this.beginX = beginX;
    this.beginY = beginY;
    this.endX = endX;
    this.endY = endY;
    this.color = color;
    this.penSize = Size;
    this.type = whichType;
    this.graphics = whichGraphics;
    Draw();
}
```

```
// Method for drawing a line
private void DrawLine(){
    graphics.setColor(color);
    graphics.drawLine(beginX, beginY, endX, endY);

}
```

The function of Line is to draw a straight line from pointed a to pointed b. Above screenshots shows all the codes of drawing a line. The first screenshot shows the codes that create the pen for the line to draw and the second screenshot is to shows the codes of letting the pen able to draw it into the drawing screen.

3. Rectangle

```
// method for creating circle, ellipse, rectangle pen
public Pen(Graphics whichGraphics, PenType whichType, int Size, Color color, Color fillColor, int beginX, int beginY, int endX, int endY){
    this.beginX = beginX;
    this.beginY = beginY;
    this.endX = endX;
    this.endY = endY;
    this.color = color;
    this.fill = fillColor;
    this.penSize = Size;
    this.type = whichType;
    this.graphics = whichGraphics;
    Draw();
}
```

```
// Method for drawing a rectangle
private void DrawRectangle(){
    int startX = Math.min(beginX, endX);
    int startY = Math.min(beginY, endY);
    int width = Math.abs(beginX - endX);
    int height = Math.abs(beginY - endY);

    graphics.setColor(color);
    graphics.drawRect(startX, startY, width, height);
    graphics.setColor(fill);
    graphics.fillRect(startX, startY, width, height);

}
```

The function of rectangle is to draw the shape of a long version of square into the drawing screen. The function also able to change the pen and internal parts colour. Above shows the screenshots of all codes about the rectangle. The first screenshot explained the methods of creating the pen for the rectangle and other shape like circle, and ellipse. The second screenshot cover all the methods for how the pen able to draw and colour fill.

4. Ellipse

```
// method for creating circle, ellipse, rectangle pen
public Pen(Graphics whichGraphics, PenType whichType, int Size, Color color, Color fillColor, int beginX, int beginY, int endX, int endY){
    this.beginX = beginX;
    this.beginY = beginY;
    this.endX = endX;
    this.endY = endY;
    this.color = color;
    this.fill = fillColor;
    this.penSize = Size;
    this.type = whichType;
    this.graphics = whichGraphics;
    Draw();
}
```

```
// Method for drawing an ellipse
private void DrawEllipse() {
    int startX = Math.min(beginX, endX);
    int startY = Math.min(beginY, endY);
    int width = Math.abs(beginX - endX);
    int height = Math.abs(beginY - endY);

    graphics.setColor(color);
    graphics.drawOval(startX, startY, width, height);
    graphics.setColor(fill);
    graphics.fillOval(startX, startY, width, height);
}
```

The function for ellipse is to allow users to draw an oval shape of object into the drawing screen. Above screenshots shows all the code about the ellipse function. The first screenshot explained about how the pen is create for the ellipse function to draw. Second screenshot covered all the method of how it able to draw and colour on the drawing screen.

5. Polygon

The function of polygon is to draw a polygon shape of object into the drawing screen by point out placing multiple of pointed location in order to link it together.

```
// properties for polygon only
private int step;
private ArrayList<Integer> xLocs;
private ArrayList<Integer> yLocs;
```

Above screenshot show the properties for the polygon only.

```
// method for creating polygon pen
public Pen(Graphics whichGraphics, int Size, Color color, Color fileColor, int x, int y){
    // setup common properties
    this.color = color;
    this.fill = fileColor;
    this.penSize = Size;
    this.graphics = whichGraphics;
    this.type = PenType.Poly;

    // setup polygon properties
    this.step = 0;
    this.xLocs = new ArrayList<Integer>();
    this.yLocs = new ArrayList<Integer>();
    this.xLocs.add(step, x);
    this.yLocs.add(step, y);
    graphics.setColor(color);

}
```

This is the methods for the polygon of creating a pen for only the function itself.

```
// Method for drawing a polygon
public void DrawPolygon() {
    // draw a line between new point and preview point
    AddPolygonPoint(xLocs.get(step), yLocs.get(step));

    // convert points from list to array
    int[] xs = buildInArray(xLocs);
    int[] ys = buildInArray(yLocs);

    graphics.setColor(color);
    graphics.drawPolygon(xs, ys, step);
    graphics.setColor(fill);
    graphics.fillPolygon(xs, ys, step);

}
```

Here is the method for letting the pen able to draw a polygon shape on the drawing screen. It also able to fill its solid colour and change the brush colour.

```
// Method for adding polygon point
public void AddPolygonPoint(int x, int y){
    // draw a line between new point and preview point
    graphics.drawLine(xLocs.get(step),yLocs.get(step),x,y);

    // update vertices
    step += 1;
    xLocs.add(step, x);
    yLocs.add(step, y);

}
```

```
public int GetPolygonStep() { return step; }
private int[] buildInArray(ArrayList<Integer> points){
    int[] ints = new int[points.size()];
    for (int i = 0; i < points.size(); i++) {
        ints[i] = points.get(i);
    }
    return ints;
}
```

II. Able to save into a VEC file

```
97      // Select a vec file to export
98      public void Save(ArrayList<Object> record){...}
```

III. Brush colour selection

The function of the brush colour selection is for the user to select any possible pen colour in a new window. Users able to choose colour by clicking the button of line colour and choose

whatever colour the user likes. After the user selected, the colour will show in a square panel where just below the button.

```
public JButton LineColorButton, FillColorButton ;
public JButton showLineColorBtn, showFillColorBtn;
```

(Here show the properties for the button)

```
private void CreateColorButtons() {

    Font colorFont = new Font( name: "Arial", Font.ITALIC, size: 10);

    LineColorButton = new JButton( text: "Line color");
    FillColorButton = new JButton( text: "Fill color");
    showLineColorBtn = new JButton();
    showFillColorBtn = new JButton();


    LineColorButton.setFont(colorFont);
    FillColorButton.setFont(colorFont);

    LineColorButton.setBackground(ButtonColor);
    FillColorButton.setBackground(ButtonColor);
    showLineColorBtn.setBackground(InitPenColor);
    showFillColorBtn.setBackground(InitFillColor);
```

```
    LineColorButton.setBounds( x: 30, y: 370, width: 80, height: 30);
    FillColorButton.setBounds( x: 30, y: 410, width: 80, height: 30);
    showLineColorBtn.setBounds( x: 20, y: 480, width: 60, height: 60);
    showFillColorBtn.setBounds( x: 60, y: 520, width: 60, height: 60);


    SidePanel.add(LineColorButton);
    SidePanel.add(FillColorButton);
    SidePanel.add(showLineColorBtn);
    SidePanel.add(showFillColorBtn);


    LineColorBtnTrigger LineColorBtnTrigger = new LineColorBtnTrigger();
    FillColorBtnTrigger FillColorBtnTrigger = new FillColorBtnTrigger();

    LineColorButton.addActionListener(LineColorBtnTrigger);
    FillColorButton.addActionListener(FillColorBtnTrigger);

}
```

(Above two screenshots covered all the method of creating the colour button. In here, the first screenshot explained creating the button with its font and visual background. The second screenshot will set the button location into the right-hand side for users' convenience. The last 4 codes are the function of enable users to trigger the button and make it work.)

```
private class LineColorBtnTrigger implements ActionListener{

    public void actionPerformed(ActionEvent e) { // overriding

        JColorChooser chooser = new JColorChooser(); // JColorChooser
        Color selectLineColor = chooser.showDialog( component: null, title: "Line Color", Color.ORANGE);

        InitPenColor = selectLineColor;
        showLineColorBtn.setBackground(InitPenColor);

    }
};
```

(This screenshot shows the method of triggering the button. By pressing the line colour button, the system will pop out a new window for users to choose and the last two code is to show the selected colour in a square panel just right below the button.)

IV. Flood fill tool

The function of flood fill is to allow users to select any possible colour for the internal parts of shapes. At first, users press the fill colour button. Secondly, users selected their desired colour.

```
public JButton LineColorButton, FillColorButton ;
public JButton showLineColorBtn, showFillColorBtn;
```

(Here show the properties for the button)

```
private void CreateColorButtons() {

    Font colorFont = new Font( name: "Arial", Font.ITALIC, size: 10);

    LineColorButton = new JButton( text: "Line color");
    FillColorButton = new JButton( text: "Fill color");
    showLineColorBtn = new JButton();
    showFillColorBtn = new JButton();


    LineColorButton.setFont(colorFont);
    FillColorButton.setFont(colorFont);

    LineColorButton.setBackground(ButtonColor);
    FillColorButton.setBackground(ButtonColor);
    showLineColorBtn.setBackground(InitPenColor);
    showFillColorBtn.setBackground(InitFillColor);
```

```
    LineColorButton.setBounds( x: 30, y: 370, width: 80, height: 30);
    FillColorButton.setBounds( x: 30, y: 410, width: 80, height: 30);
    showLineColorBtn.setBounds( x: 20, y: 480, width: 60, height: 60);
    showFillColorBtn.setBounds( x: 60, y: 520, width: 60, height: 60);


    SidePanel.add(LineColorButton);
    SidePanel.add(FillColorButton);
    SidePanel.add(showLineColorBtn);
    SidePanel.add(showFillColorBtn);


    LineColorBtnTrigger LineColorBtnTrigger = new LineColorBtnTrigger();
    FillColorBtnTrigger FillColorBtnTrigger = new FillColorBtnTrigger();

    LineColorButton.addActionListener(LineColorBtnTrigger);
    FillColorButton.addActionListener(FillColorBtnTrigger);

}
```

(Above two screenshots covered all the method of creating the colour fill button. In here, the first screenshot explained creating the button with its font and visual background. The second screenshot will set the button location into the right-hand side for users' convenience. The last 4 codes are the function of enable users to trigger the button and make it work.)

```
private class FillColorBtnTrigger implements ActionListener{

    public void actionPerformed(ActionEvent e) { // overriding

        JColorChooser chooser = new JColorChooser(); // JColorChooser
        Color selectFillColor = chooser.showDialog( component: null,  title: "Line Color", Color.ORANGE);

        InitFillColor = selectFillColor;
        showFillColorBtn.setBackground(InitFillColor);

    }
};
```

(This screenshot proves the method of triggering the fill button. By pressing the fill colour button, the system will pop out a new window for users to choose and the last two code is to show the selected colour in a square panel just right below the button.)

V. Undo button

The function of undo is to revert the object back to the screen that never appear before.

```
public void Undo() {
    // check to see if the record size is greater than 0
    if (record.size() > 0) {
        // refresh the screen
        paint(graphics);

        // remove the last recorded shape
        record.remove( index: record.size() - 1);

        // redraw every single shape
        for (Object currentRecord : record ) {
            Pen pen = (Pen)currentRecord;
            pen.Draw();
        }
    }

}
```

(This screenshot showed all the code for the function of undo.)

VI. VEC file import

```
44      // Select a vec file to import
45      public void Open(Color defaultPen, Color defaultFill, Tool tool, Graphics2D graphics, FrameDesign theWindow)
46          if (fileChooser.showOpenDialog( parent: this) == JFileChooser.APPROVE_OPTION) {
47              // get the selected file
48              File theFile = fileChooser.getSelectedFile();
49              String format = theFile.getName().substring(theFile.getName().length()-4);
50
51              // when opening another format file
52              if (!format.equals(supportFormat)){
53                  JOptionPane.showMessageDialog( parentComponent: this, message: "This is not a support format");
54                  return;
55              }
56
57              // actual open the file
58              theReader.Draw(defaultPen, defaultFill, tool, graphics, ReadingTheFile(theFile));
59
60              // change the name of the window
61              theWindow.getMainFrame().setTitle(theFile.getName());
62
63          }
64      }
```

Additional Functionality

```
198         // Method for creating tool shapeBtns
199         private void CreateToolButtons(){
200             // setup grid icon paths
201             gridIcon = new ImageIcon[2];
202             gridIcon[0] = new ImageIcon("image\\grid_on.png");
203             gridIcon[1] = new ImageIcon("image\\grid_off.png");
204 
205             // shapeBtns creation
206             UndoButton = new JButton (new ImageIcon("image\\undo.gif"));
207             RedoButton = new JButton (new ImageIcon("image\\redo.gif"));
208             ClearButton = new JButton(new ImageIcon("image\\clean.png"));
209             GridButton = new JButton(gridIcon[0]);
210 
211             // set shapeBtns color
212             UndoButton.setBackground(ButtonColor);
213             RedoButton.setBackground(ButtonColor);
214             ClearButton.setBackground(ButtonColor);
215             GridButton.setBackground(ButtonColor);
216 
217             // set undo and redo button disable
218             UndoButton.setEnabled(false);
219             RedoButton.setEnabled(false);
220 
221             // set shapeBtns locations
222             UndoButton.setBounds(windowWidth - 200, 5, 80, 40);
223             RedoButton.setBounds(windowWidth - 115, 5, 80, 40);
224             ClearButton.setBounds(windowWidth - 30, 5, 80, 40);
225             GridButton.setBounds(windowWidth + 55, 5, 80, 40);
226 
227             // add shapeBtns in window
228             TopPanel.add(UndoButton);
229             TopPanel.add(RedoButton);
230             TopPanel.add(ClearButton);
231             TopPanel.add(GridButton);
232 
233             // add trigger to shapeBtns
234             UndoButton.addActionListener(toolBtnTrigger);
235             RedoButton.addActionListener(toolBtnTrigger);
236             ClearButton.addActionListener(toolBtnTrigger);
237             GridButton.addActionListener(toolBtnTrigger);
238 
239         }
```

```
401     // Method for users to use tool at once
402     private class ToolBtnTrigger implements ActionListener{
403         @Override
404         public void actionPerformed(ActionEvent e) {
405             JButton clickedButton = (JButton)e.getSource();
406 
407             if (clickedButton.equals(ClearButton)){
408                 tool.Clean();
409 
410             }else if (clickedButton.equals(UndoButton)) {
411                 tool.Undo();
412 
413             } else if (clickedButton.equals(RedoButton)) {
414                 tool.Redo();
415 
416             } else if (clickedButton.equals(LineColorButton)) {
417                 tool.ChooseLineColor();
418 
419             } else if (clickedButton.equals(FillColorButton)) {
420                 tool.ChooseFillColor();
421 
422             } else if (clickedButton.equals(GridButton)){
423                 tool.SwitchGrid();
424 
425             }else if (clickedButton.equals(showLineColorBtn) || clickedButton.equals(showFillColorBtn)) {
426                 quickColorSelectionBtn = clickedButton;
427 
428             } else if (IsClickingColorPattern(clickedButton)) {
429                 UseQuickSelecteColor(clickedButton);
430 
431             }
432         }
```

I. Grid

```
736             // Method to turn on or off grid
737             public void SwitchGrid(){
738                 // decide turn on or turn off
739                 boolean isOn;
740                 if (GridButton.getIcon() == gridIcon[0]){
741                     isOn = true;
742                     GridButton.setIcon(gridIcon[1]);
743                 }else{
744                     isOn = false;
745                     GridButton.setIcon(gridIcon[0]);
746                 }
747 
748                 // actual turning the grid
749                 theGrid.EnableGrid(isOn);
750             }
751 
752 
753 
754         }
```

II. Multi-image support

```
66      // Open a vec file in a new window
67      public void OpenFileInNewWindow(){
68          FrameDesign newWindow = new FrameDesign("Choose your vec file");
69          Graphics2D newGraphics = newWindow.getPicture();
70          Tool newTool = newWindow.getTool();
71          Color penColor = newWindow.getPenColor();
72          Color fillColor = newWindow.getFillColor();
73          Open(penColor, fillColor, newTool, newGraphics, newWindow);
74
75      }
```

III. Clean/ Eraser

```
public void Clean(){
    // refresh the screen and clear the record
    record.clear();
    paint(graphics);

}
```

IV. Redo

```
665         // method to redo the last undo
666         public void Redo(){
667             // check to see if the undo histroy size is greater than 0
668             if (undoHistroy.size() > 0){
669                 // refresh the screen
670                 Refresh();
671
672                 // remove the last recorded shape fro undo histroy
673                 Pen pen = (Pen)undoHistroy.get(undoHistroy.size()-1);
674                 undoHistroy.remove(pen);
675
676                 // add pen to record list and enable undo button
677                 record.add(pen);
678                 UndoButton.setEnabled(true);
679
680                 // redraw every single shape
681                 Redraw();
682
683                 // make redo button gray again
684                 if (undoHistroy.size() == 0){
685                     RedoButton.setEnabled(false);
686                 }
687             }
688         }
689
```

V. Pen Size

```
600         // Method to lock brush size
601         private void LockBrushsize(){
602             int min = 1;
603             int max = 30;
604             int size = Integer.parseInt(brushSizeEnter.getText());
605
606             if (size > max){
607                 size = max;
608             }else if (size < min){
609                 size = min;
610             }
611
612             brushSizeEnter.setText(String.valueOf(size));
613             brushSize = size;
614         }
615
616     }
```

Communication & Team Works

Communication tools

- Discord
- Facebook
- GitBucket
- Google Document
- Intellij IDEA

Agile Software Development

CAB302 Group 188			
Team Members No.	**Name**	**Position**	**Colour**
1.n9915931	Johnson KaiZhi Foo	Report Writing & Project Management	
2.n10094881	Yung Han Lin	Report & Coding	
3.n10192701	Wei Zhao Wu	Coding & Unit Testing	
4.n8051381	MinHo Kim	Coding & GUI Design	

Project Management:

Work Distribution	
Mission	**Responsible**
Research	
Design	
GUI Function	
Shape Function	
Undo Function	
Clean Function	
Colour Selection Function	
Drop Down Window (File Menu)	
Additional Function	
Import/ Open	
Export/ Save	
Unit Testing	
JAVA Doc	
Report	

Team member Contribution:

Names	Report Writing	Coding	Creation of unit test	Timetable management
Johnson KaiZhi, Foo	✓	✓		✓
Wei Zhao, Wu	✓	✓	✓	
Minho, Kim	✓	✓	✓	
Yung Han, Lin	✓	✓		✓

Meeting Schedule:

#Semester 1- 2019					
Week 8	Looking for group members				
Week 9	Group 188 formed	Criteria Review	Research		
Week 10	Work Distribution	Start paint project	Project Design	Window Drop Down	
Week 11	PLOT	LINE	Rectangle	Ellipse	Polygon
Week 12	Undo	Import	Export	Unit Testing	
Week 13	Additional Functionality	Report	Criteria Review again		

Group meeting record:

Week No.	Meeting No.	Date
8	1	17/04/2019
9	2	01/05/2019
10	3	06/05/2019
10	4	10/05/2019
11	5	15/05/2019
11	6	18/05/2019
12	7	21/05/2019
12	8	25/05/2019
12	9	26/05/2019
13	10	28/05/2019
13	11	31/05/2019
13	12	01/06/2019
13	13	02/06/2019

Software Architecture

The project initially began by running the class of Frame Design. The main class (java file) is the execution point and the it is also as the controller as this is a standard Java Swing.

1. Public class FrameDesign
   Is to create a frame and button to draw the shapes. It also has a buttons and tools for the brush and fill colour in order to draw appointed object in the default white background. It is the only class which have the main method to run the program and build the GUI application out. This class included the method of creating the frame, menubar, buttons, tools, colour buttons and core component for the GUI to appear.

2. Private class ShapeBtnTrigger implements ActionListener
   The class of ShapeBtnTrigger is for the button of any shapes to allow the system to detect which button we pressed and check if the user is clicked on a new button. If the users did not press a new colour button for the shape, the colour for the shape will be the same as the default one.

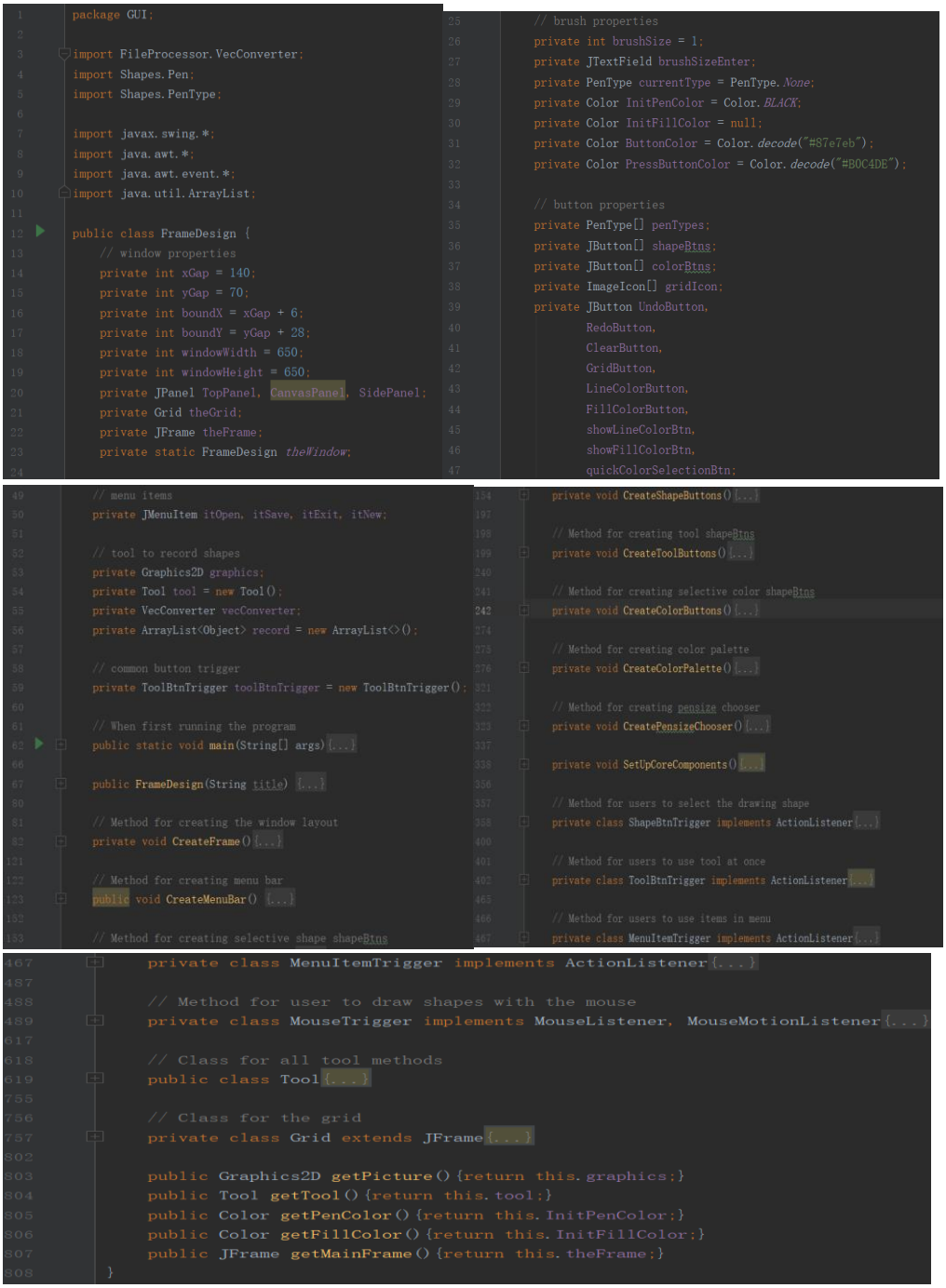
3. Private class ToolBtnTrigger implements ActionListener
   The class of ToolBtnTrigger is to detect the users which tool button he chooses. By using the if statement to check whether the users will choose clean, undo, line colour selection, fill colour selection, quick line colour selection, and quick fill colour selection. The system will check the users whether he press the button to open the colour palette by using the method of for loop statement.

4. Private class MenuItemTrigger implements ActionListener
   This class is a method for the users to use any item in the file menu. the system will detect whether the users chooses which items in the menu by using the if statement. There are four items in the file menu such as create new file, import existing file, save current file and exit the application.

5. Private class MouseTrigger implements MouseListener, MouseMotionListener
   The class of MouseTrigger is a method for the users to draw the selected shape using mouse. It able to record the shape users had drawn and draw it out when the users release the mouse click. All the trigger for the canvas panel will return its value to the private method of CreateFrame in the public class FrameDesign.
6. Public class Tool
   The Tool class is to set up all tools method and make it usable. The Tool class have the undo, clean, choose line and fill colour function. This class will return back to the private class of ToolBtnTrigger in order to make it functionable in the Jframe.

7. Public class Pen
   This class is to create a pen for all the shape to be able to draw in the Drawing screen. This methods inside this class is basically all the methods for creating the shape and received the colour for the line and fill that the users had chosen.
8. Public class VecConverter
   This class is significant for the project as it required this class to be able to read the Vec file inside or outside the application. This class will initiate the converter and select a Vec file from a pop out window to import or export. In the process of import or export, the program will start beginning a conversion to the text, pen type, colour, and coordinates to generate a Vec format file. After convert is finished, it will return its value to the class of VecReader to be able to show in the application.
9. Public class VecReader
   The main function of VecReader is to analyse the imported Vec file and show it visible in the application. The method of Draw is to record the default graphics and colour. Secondly, the system will clean the drawing screen and start to draw the imported Vec file. Lastly, all the things that we draw will save it in a new graphics and array list. The private method of DrawEverySingleShape is to create an array list and get all the detail of shapes to be able to draw.
10. Public class VecWriter
    This class of VecWriter is to create a list of variables and scan every pen's graphics to write the file in the variable of FileWriter. The method of ScanForObjs is to scan every single pen's graphic in the drawing screen and store it into the list. The method of InsertObjInfoIntoList is to convert the pen's graphic that scan from the method of ScanForObjs into a string value. Lastly, the method of WriteInInFile is to use the variable of FileWriter to write the string value into the array list that created in the method of Export to export the Vec file.

Screenshot for all the Class



```
package GUI;

import FileProcessor.VecConverter;
import Shapes.Pen;
import Shapes.PenType;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

public class FrameDesign {
    // window properties
    private int xGap = 140;
    private int yGap = 70;
    private int boundX = xGap + 6;
    private int boundY = yGap + 28;
    private int windowWidth = 650;
    private int windowHeight = 650;
    private JPanel TopPanel, CanvasPanel, SidePanel;
    private Grid theGrid;
    private JFrame theFrame;
    private static FrameDesign theWindow;

    // brush properties
    private int brushSize = 1;
    private JTextField brushSizeEnter;
    private PenType currentType = PenType.None;
    private Color InitPenColor = Color.BLACK;
    private Color InitFillColor = null;
    private Color ButtonColor = Color.decode("#87e7eb");
    private Color PressButtonColor = Color.decode("#B0C4DE");

    // button properties
    private PenType[] penTypes;
    private JButton[] shapeBtns;
    private JButton[] colorBtns;
    private ImageIcon[] gridIcon;
    private JButton UndoButton,
            RedoButton,
            ClearButton,
            GridButton,
            LineColorButton,
            FillColorButton,
            showLineColorBtn,
            showFillColorBtn,
            quickColorSelectionBtn;

    // menu items
    private JMenuItem itOpen, itSave, itExit, itNew;

    // tool to record shapes
    private Graphics2D graphics;
    private Tool tool = new Tool();
    private VecConverter vecConverter;
    private ArrayList<Object> record = new ArrayList<>();

    // common button trigger
    private ToolBtnTrigger toolBtnTrigger = new ToolBtnTrigger();

    // When first running the program
    public static void main(String[] args){...}

    public FrameDesign(String title) {...}

    // Method for creating the window layout
    private void CreateFrame(){...}

    // Method for creating menu bar
    public void CreateMenuBar() {...}

    // Method for creating selective shape shapeBtns
    private void CreateShapeButtons(){...}

    // Method for creating tool shapeBtns
    private void CreateToolButtons(){...}

    // Method for creating selective color shapeBtns
    private void CreateColorButtons(){...}

    // Method for creating color palette
    private void CreateColorPalette(){...}

    // Method for creating pensize chooser
    private void CreatePensizeChooser(){...}

    private void SetUpCoreComponents(){...}

    // Method for users to select the drawing shape
    private class ShapeBtnTrigger implements ActionListener{...}

    // Method for users to use tool at once
    private class ToolBtnTrigger implements ActionListener{...}

    // Method for users to use items in menu
    private class MenuItemTrigger implements ActionListener{...}

    private class MenuItemTrigger implements ActionListener{...}

    // Method for user to draw shapes with the mouse
    private class MouseTrigger implements MouseListener, MouseMotionListener{...}

    // Class for all tool methods
    public class Tool{...}

    // Class for the grid
    private class Grid extends JFrame{...}

    public Graphics2D getPicture(){return this.graphics;}
    public Tool getTool(){return this.tool;}
    public Color getPenColor(){return this.InitPenColor;}
    public Color getFillColor(){return this.InitFillColor;}
    public JFrame getMainFrame(){return this.theFrame;}
}
```

Advanced Object-Oriented Programming Principles

- ## Abstraction
  There is three main folders for our paint project. You will be able to find the abstraction by looking each class in every folder. The diagrams are presented in a UML diagram in order to allow user to understand easily.

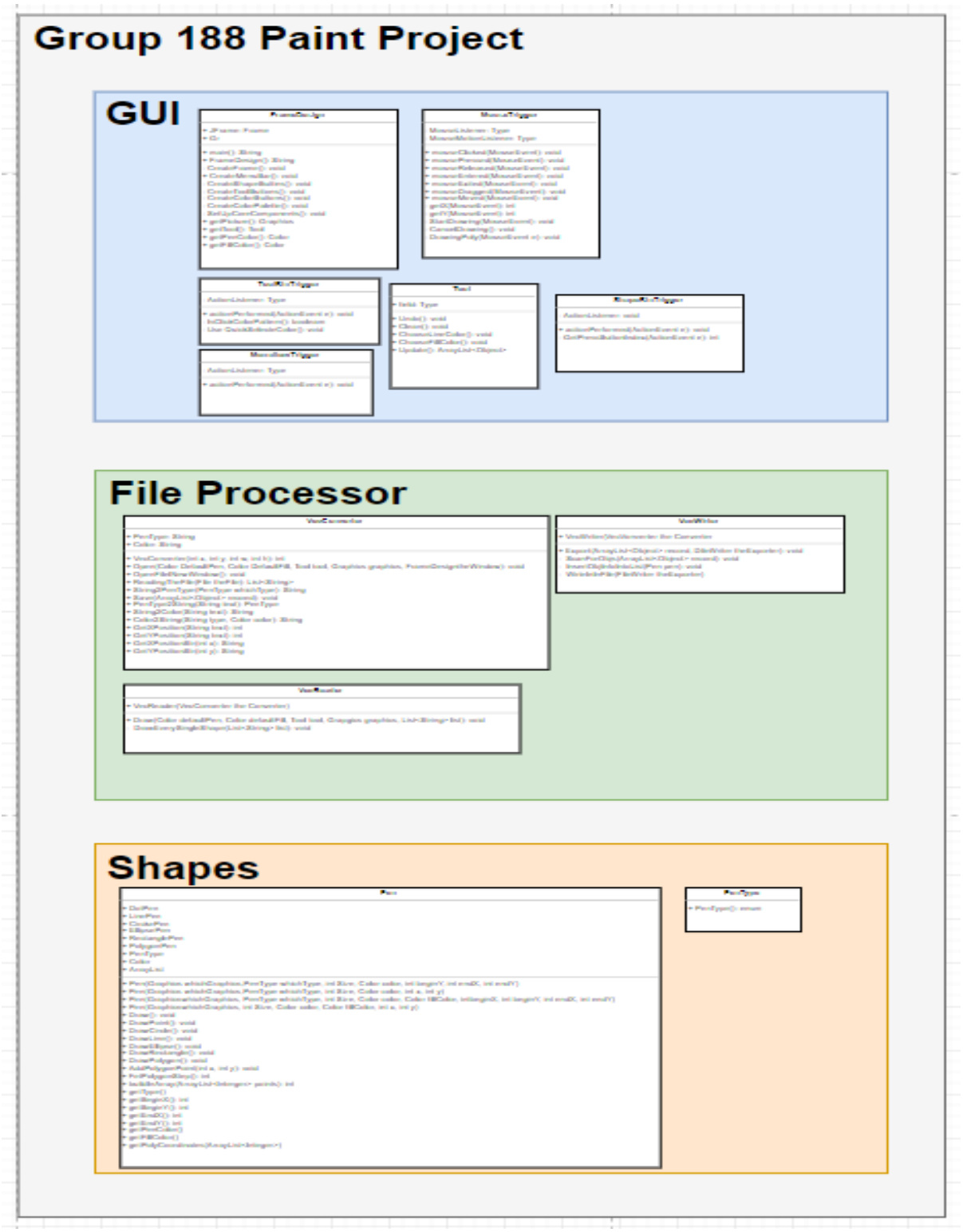
- ## Encapsulation
  By looking the important stuff for encapsulation. In every class diagram, there is a symbol for each method and the symbol of minus(-) is all the private method for the paint project.

- ## Inheritance
  The only inheritance in our code are public class FrameDesign extends JFrame, and public class VecConverter extends JFrame.

  ```
  public class VecConverter extends JFrame {
  ```

- ## Polymorphism
  All the polymorphism operation able to find in the class of ShapeBtnTrigger, ToolBtnTrigger, MenuItemTrigger, MouseTrigger. All this class is supported by method of overriding.

UML Diagrams



GUI

FrameDesign

+ JFrame: Frame
+ Gr

+ main(): String
+ FrameDesign(): String
- CreateFrame(): void
+ CreateMenuBar(): void
- CreateShapeButtins(): void
- CreateToolButtons(): void
- CreateColorButtons(): void
- CreateColorPalette(): void
- SetUpCoreComponents(): void
+ getPicture(): Graphics
+ getTool(): Tool
+ getPenColor(): Color
+ getFillColor(); Color

MouseTrigger

- MouseListener: Type
- MouseMotionListener: Type

+ mouseClicked(MouseEvent): void
+ mousePressed(MouseEvent): void
+ mouseReleased(MouseEvent): void
+ mouseEntered(MouseEvent): void
+ mouseExited(MouseEvent): void
+ mouseDragged(MouseEvent): void
+ mouseMoved(MouseEvent): void
- getX(MouseEvent): int
- getY(MouseEvent): int
- StartDrawing(MouseEvent): void
- CancelDrawing(): void
- DrawingPoly(MouseEvent e): void

ToolBtnTrigger

- ActionListener: Type

+ actionPerformed(ActionEvent e): void
- IsClickColorPattern(): booleam
- Use QuickSelecteColor(): void

Tool

+ field: Type

+ Undo(): void
+ Clean(): void
+ ChooseLineColor(): void
+ ChooseFillColor(): void
+ Update(): ArrayList<Object>

ShapeBtnTrigger

- ActionListener: void

+ actionPerformed(ActionEvent e): void
- GetPressButtonIndex(ActionEvent e): int

MenuItemTrigger

- ActionListener: Type

+ actionPerformed(ActionEvent e): void

File Processor

VecConverter

+ PenType: String
+ Color: String

+ VecConverter(int x, int y, int w, int h): int
+ Open(Color DefaultPen, Color DefaultFill, Tool tool, Graphics graphics, FrameDesigntheWindow): void
+ OpenFileINewWindow(): void
+ ReadingTheFile(File theFile): List<String>
+ String2PenType(PenType whichType): String
+ Save(ArrayList<Object> recoed): void
+ PenType2String(String text): PenType
+ String2Color(String text): String
+ Color2String(String type, Color color): String
+ GetXPosition(String text): int
+ GetYPosition(String text): int
+ GetXPositionStr(int x): String
+ GetYPositionStr(int y): String

VecWirter

+ VecWriter(VecVonverter the Converter

+ Export(ArrayList<Object> record, DileWriter theExporter): void
- ScanForObjs(ArrayList<Object> record): void
- InsertObjInfoIntoList(Pen pen): void
- WirteInInFile(FileWriter theExporter)

VecReader

+ VecReader(VecConverter the Converter)

+ Draw(Color defaultPen, Color defaultFill, Tool tool, Grapgics graphics, List<String> list): void
- DrawEverySingleShape(List<String> list): void

Shapes

Pen

+ DotPen
+ LinePen
+ CirckePen
+ EllipsePen
+ RectanglePen
+ PolygonPen
+ PenType
+ Color
+ ArrayList

+ Pen(Graphics whichGraphics,PenType whichType, int Size, Color color, int beginY, int endX, int endY)
+ Pen(Graphics whichGraphics,PenType whichType, int Size, Color color, int x, int y)
+ Pen(GraphicswhichGraphics, PenType whichType, int Size, Color color, Color fillColor, intbeginX, int beginY, int endX, int endY)
+ Pen(GraphicswhichGraphics, int Size, Color color, Color fillColor, int x, int y)
+ Draw(): void
+ DrawPoint(): void
+ DrawCircle(): void
+ DrawLine(): void
+ DrawEllipse(): void
+ DrawRectangle(): void
+ DrawPolygon(): void
+ AddPolygonPoint(int x, int y): void
+ FetPolygonStep(): int
+ buildInArray(ArrayList<Interger> points): int
+ getType()
+ getBeginX(): int
+ getBeginY(): int
+ getEndX(): int
+ getEndY(): int
+ getPenColor()
+ getFillColor()
+ getPolyCoordinates(ArrayList<Integer>)

PenType

+ PenType(): enum

User Guide

I. **First Step:** Open the application tool by running the class of FrameDesign.

II. **Second Step:** The right up corner grid button will show the grid on or off. If you turn on the grid function, the icon will change into a slash grid icon and the drawing screen will appear a grid panel. If you turn off the grid function, the icon will return to normal grid icon and the panel will disappear.

III. **Third Step:** You will be able to open the colour palette window by pressing the button of line and fill colour. You also able to select quick colour selection at the bottom left corner.

IV. **Forth Step:** You will be able to draw point by pressing the button of Point.

V. **Fifth Step:** Press the Circle button to draw a circle shape.

VI. **Sixth Step:** You can draw a rectangle shape by pressing the button of rectangle

VII. **Seventh Step:** The ellipse button is a function for you to draw an oval shape.

VIII. **Eighth Step :** Press the polygon and start point the drawing screen of at least three point to form a polygon shape.

IX. **Ninth Step:** The eraser icon at the top right is the function of cleaning all object in the drawing screen.

X. **Tenth Step:** The undo button is the middle one beside the pen thickness area. Using the undo function will allows you to revert to the previous screen.

XI. **Eleventh Step:** The redo button is to restore your screen to the latest ahead.

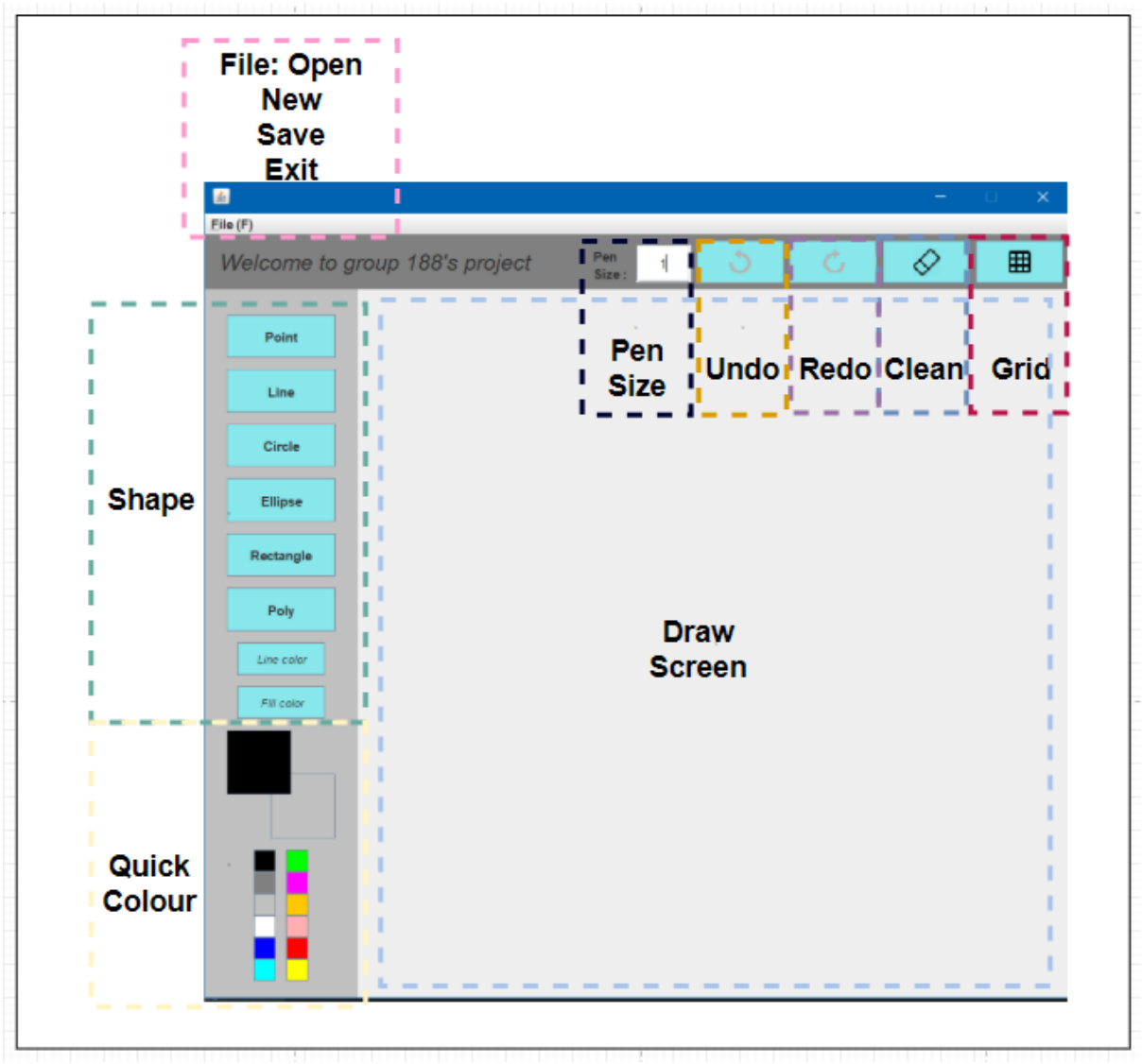
XII. **Twelfth Step:** You able to save your current drawing stuff into the format of VEC file by pressing the button of save in the file setting.

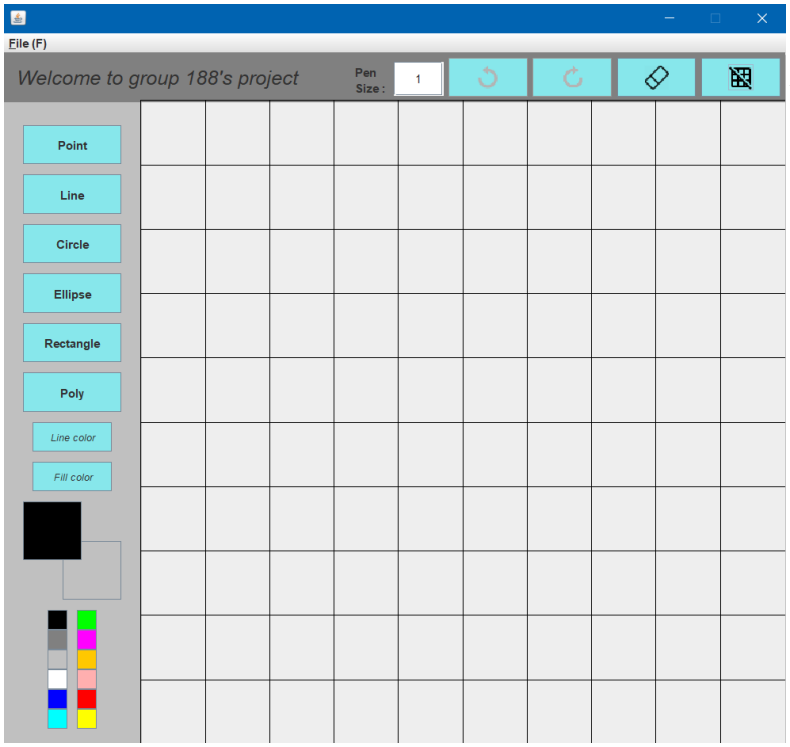
XIII. **Thirteenth Step:** You able to import other VEC file into your current drawing screen too.

XIV. **Fourteenth Step:** You able to open a new window of application and choose any vec file to import or cancel to continue your new window.

XV. **Fifteen Step:** Press the cross button at the right-hand corner side or go to the file and choose the exit to terminate the application.

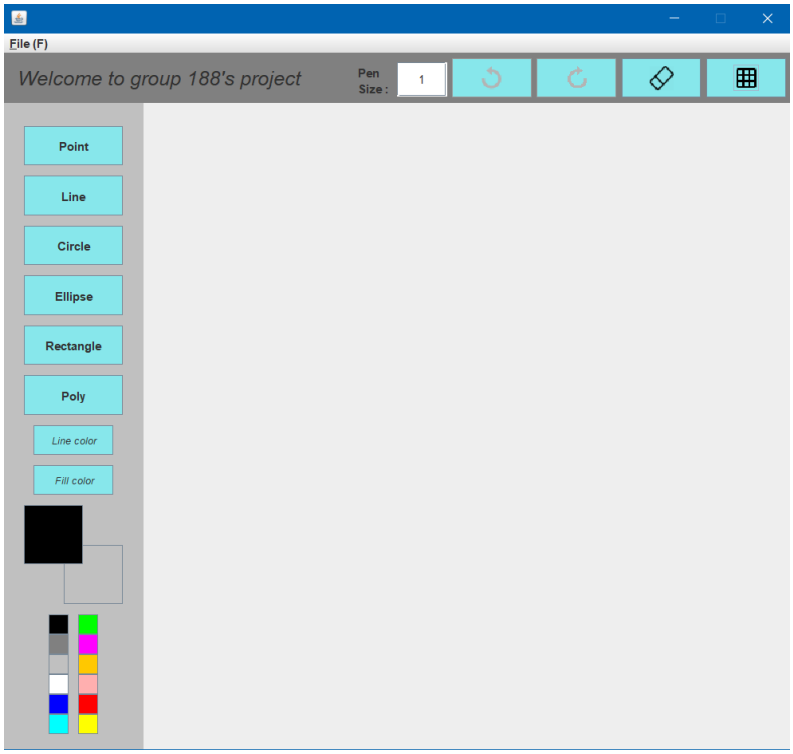
Screenshot


File: Open
New
Save
Exit
File (F)
Welcome to group 188's project
Pen Size :
Point
Line
Circle
Ellipse
Rectangle
Poly
Line color
Fill color
Shape
Quick Colour
Pen Size
Undo
Redo
Clean
Grid
Draw Screen



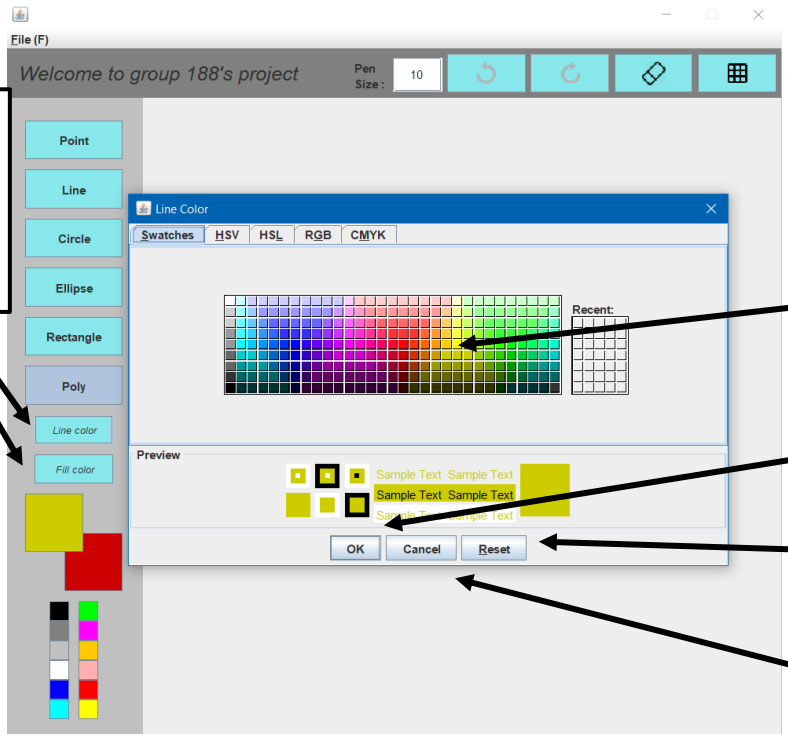
Grip turn on

Grid appear in the draw screen



Grip turn off

Grid gone after it is turn off



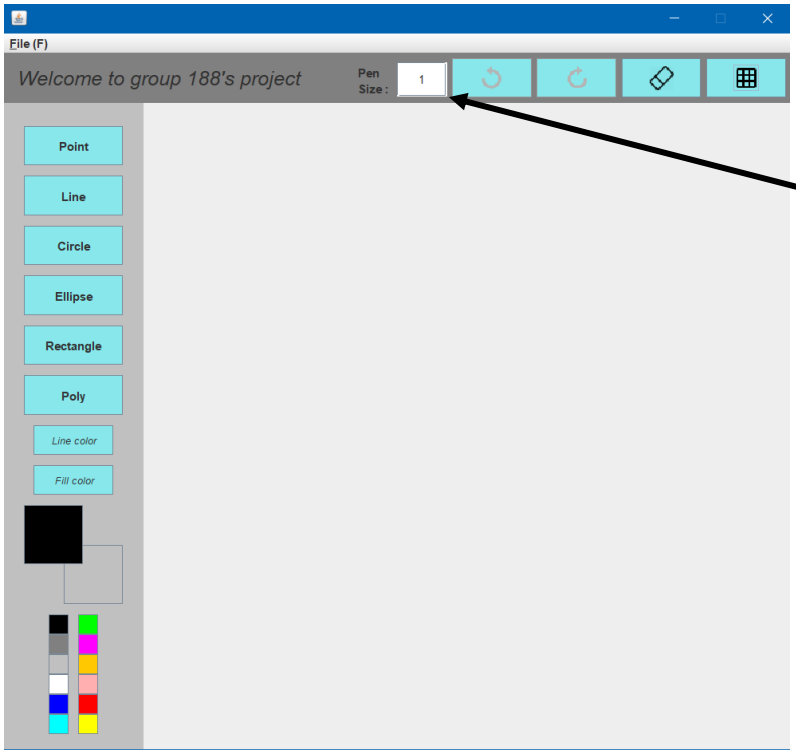
Press these two buttons to open the colour

Select any colour

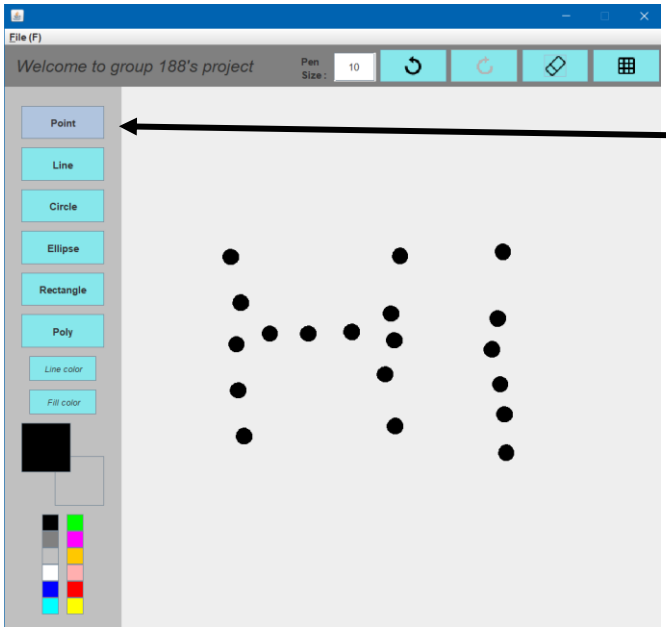
Press ok to confirm

Reset to the beginning colour

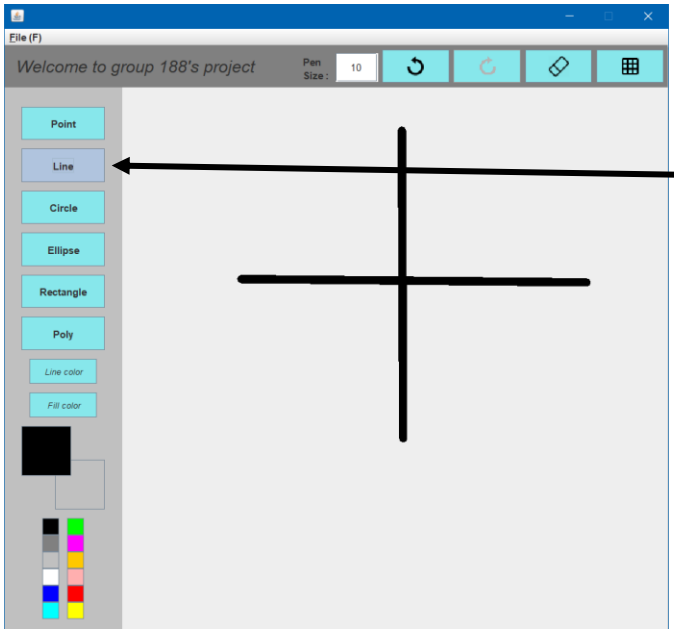
Press to make the fill colour disable



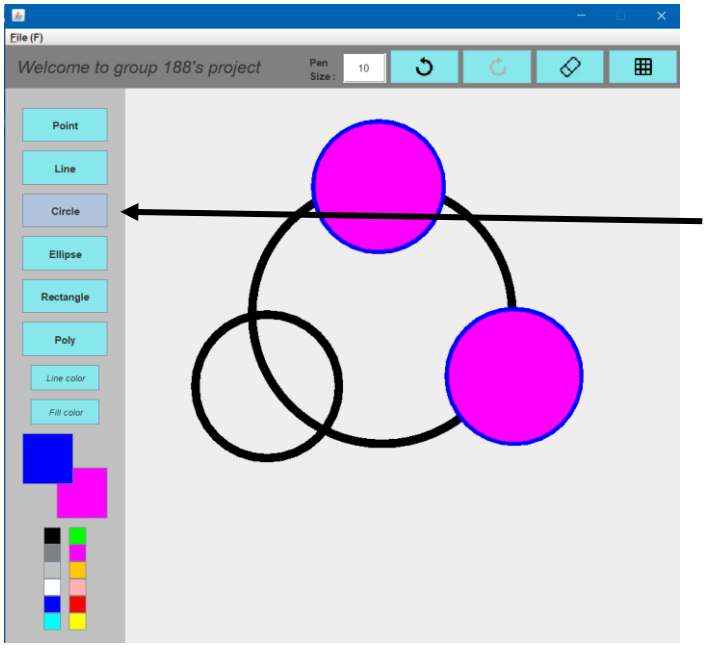
Change the pen's thickness 10 ~30

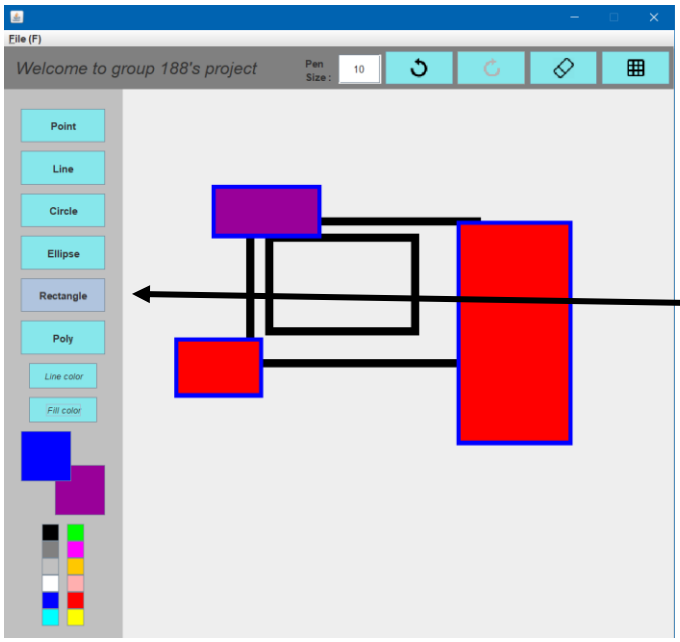


Point Function

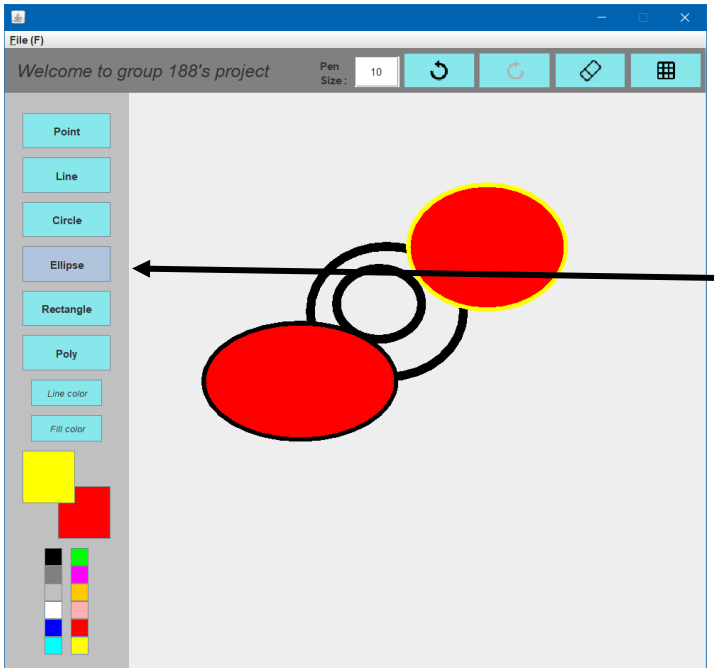


Line Function

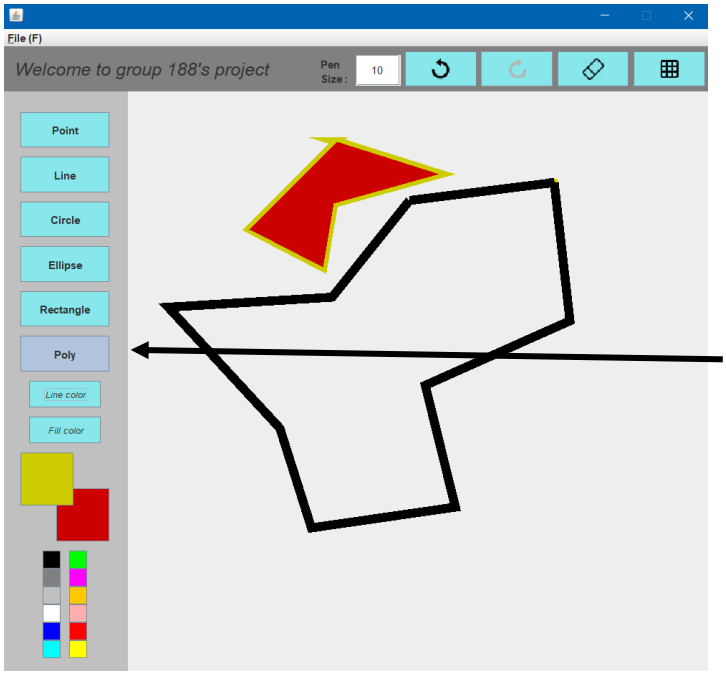




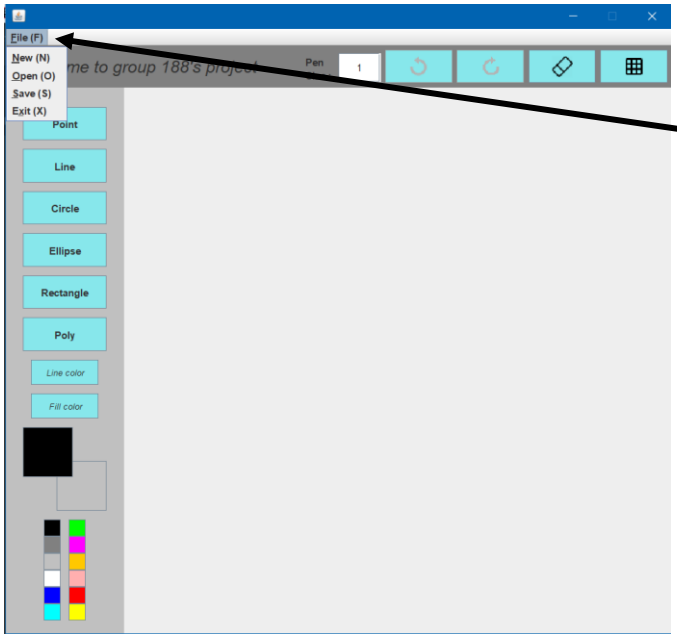
Rectangle Function



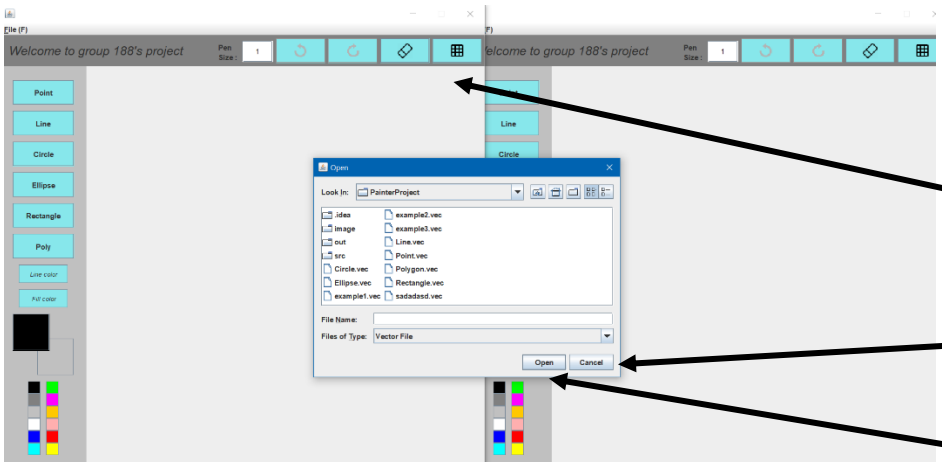
Ellipse Function



Polygon Function

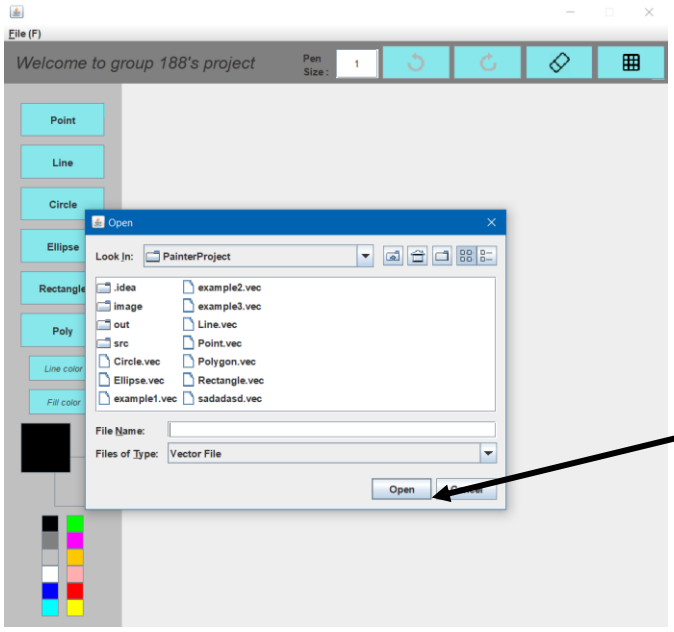


Open File Menu

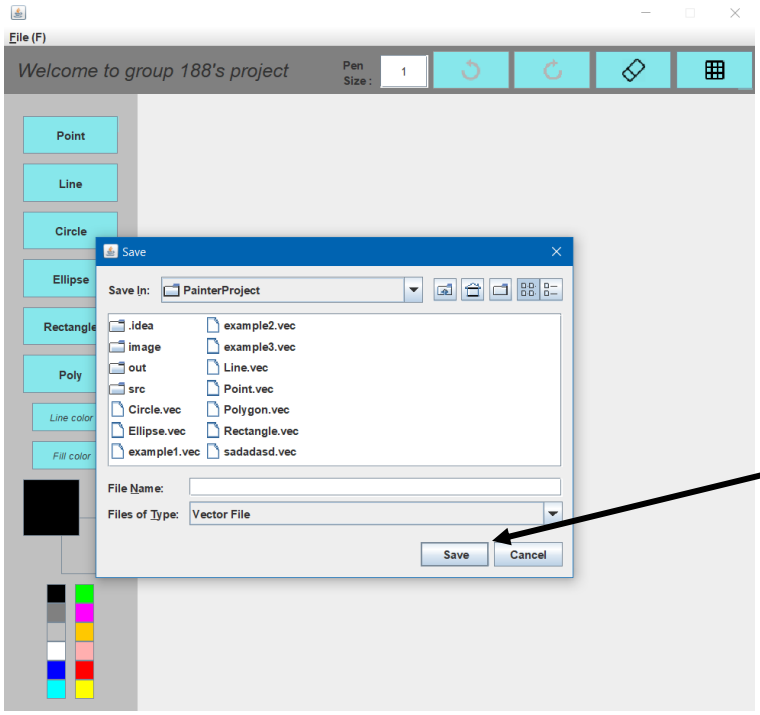


New window when u press new

Cancel to continue on new window



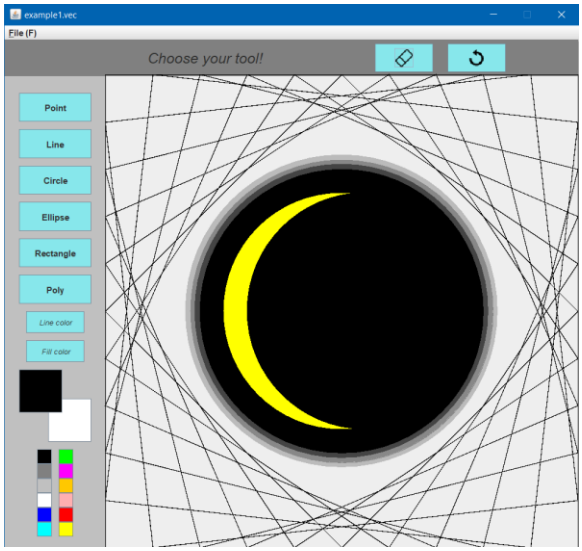
Select and open to import in the current window when u press opens in the file menu



Type your new file name and save it in vec format file

Example Screenshot

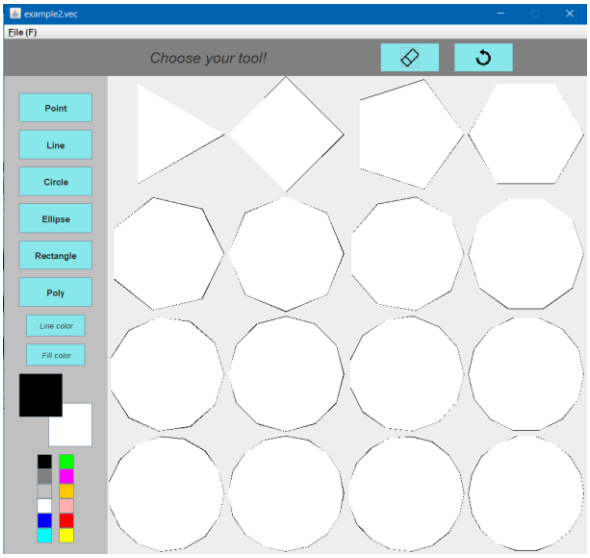
- Example 1



Vec Code:

```
LINE 0.000000 0.000000 1.000000 0.000000
LINE 1.000000 0.000000 1.000000 1.000000
LINE 1.000000 1.000000 0.000000 1.000000
LINE 0.000000 1.000000 0.000000 0.000000
LINE 0.100000 0.000000 1.000000 0.100000
LINE 1.000000 0.100000 0.900000 1.000000
LINE 0.900000 1.000000 0.000000 0.900000
LINE 0.000000 0.900000 0.100000 0.000000
LINE 0.200000 0.000000 1.000000 0.200000
LINE 1.000000 0.200000 0.800000 1.000000
LINE 0.800000 1.000000 0.000000 0.800000
LINE 0.000000 0.800000 0.200000 0.000000
LINE 0.300000 0.000000 1.000000 0.300000
LINE 1.000000 0.300000 0.700000 1.000000
LINE 0.700000 1.000000 0.000000 0.700000
LINE 0.000000 0.700000 0.300000 0.000000
LINE 0.400000 0.000000 1.000000 0.400000
LINE 1.000000 0.400000 0.600000 1.000000
LINE 0.600000 1.000000 0.000000 0.600000
LINE 0.000000 0.600000 0.400000 0.000000
LINE 0.500000 0.000000 1.000000 0.500000
LINE 1.000000 0.500000 0.500000 1.000000
LINE 0.500000 1.000000 0.000000 0.500000
LINE 0.000000 0.500000 0.500000 0.000000
LINE 0.600000 0.000000 1.000000 0.600000
LINE 1.000000 0.600000 0.400000 1.000000
LINE 0.400000 1.000000 0.000000 0.400000
LINE 0.000000 0.400000 0.600000 0.000000
LINE 0.700000 0.000000 1.000000 0.700000
LINE 1.000000 0.700000 0.300000 1.000000
LINE 0.300000 1.000000 0.000000 0.300000
LINE 0.000000 0.300000 0.700000 0.000000
LINE 0.800000 0.000000 1.000000 0.800000
LINE 1.000000 0.800000 0.200000 1.000000
LINE 0.200000 1.000000 0.000000 0.200000
LINE 0.000000 0.200000 0.800000 0.000000
LINE 0.900000 0.000000 1.000000 0.900000
LINE 1.000000 0.900000 0.100000 1.000000
LINE 0.100000 1.000000 0.000000 0.100000
LINE 0.000000 0.100000 0.900000 0.000000
LINE 1.000000 0.000000 1.000000 1.000000
LINE 1.000000 1.000000 0.000000 1.000000
LINE 0.000000 1.000000 0.000000 0.000000
LINE 0.000000 0.000000 1.000000 0.000000
PEN #BBBBBB
FILL #BBBBBB
ELLIPSE 0.17 0.17 0.83 0.83
PEN #888888
FILL #888888
ELLIPSE 0.18 0.18 0.82 0.82
PEN #444444
FILL #444444
ELLIPSE 0.19 0.19 0.81 0.81
PEN #000000
FILL #000000
ELLIPSE 0.2 0.2 0.8 0.8
FILL #FFFF00
ELLIPSE 0.25 0.25 0.75 0.75
FILL #000000
ELLIPSE 0.3 0.25 0.8 0.75
```

- Example 2



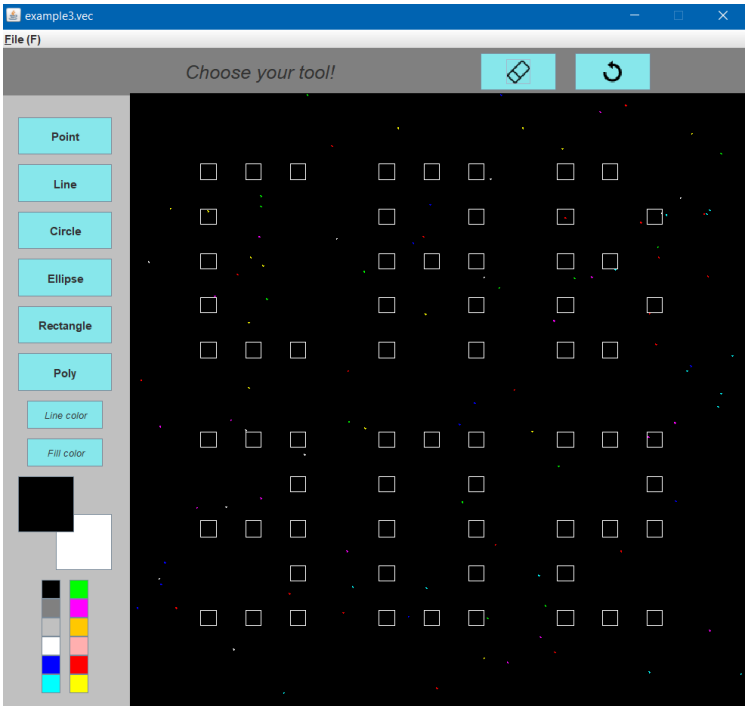
Vec Code:

```
POLYGON 0.245000 0.125000 0.065000 0.228923 0.065000 0.021077
POLYGON 0.495000 0.125000 0.375000 0.245000 0.255000 0.125000 0.375000 0.005000
POLYGON 0.745000 0.125000 0.662082 0.239127 0.527918 0.195534 0.527918 0.054466 0.662082 0.010873
POLYGON 0.995000 0.125000 0.935000 0.228923 0.815000 0.228923 0.755000 0.125000 0.815000 0.021077 0.935000 0.021077
POLYGON 0.245000 0.375000 0.199819 0.468820 0.098297 0.491991 0.016884 0.427066 0.016884 0.322934 0.098297 0.258009 0.199819 0.281180
POLYGON 0.495000 0.375000 0.459853 0.459853 0.375000 0.495000 0.290147 0.459853 0.255000 0.375000 0.290147 0.290147 0.375000 0.255000
POLYGON 0.745000 0.375000 0.716925 0.452135 0.645838 0.493177 0.565000 0.478923 0.512237 0.416042 0.512237 0.333958 0.565000 0.271077
POLYGON 0.995000 0.375000 0.972082 0.445534 0.912082 0.489127 0.837918 0.489127 0.777918 0.445534 0.755000 0.375000 0.777918 0.304466
POLYGON 0.245000 0.625000 0.225950 0.689877 0.174850 0.734156 0.107922 0.743779 0.046417 0.715690 0.009861 0.658808 0.009861 0.591192
POLYGON 0.495000 0.625000 0.478923 0.685000 0.435000 0.728923 0.375000 0.745000 0.315000 0.728923 0.271077 0.685000 0.255000 0.625000
POLYGON 0.745000 0.625000 0.731255 0.680767 0.693168 0.723758 0.639464 0.744125 0.582447 0.737202 0.535179 0.704575 0.508487 0.653718
POLYGON 0.995000 0.625000 0.983116 0.677066 0.949819 0.718820 0.901703 0.741991 0.848297 0.741991 0.800181 0.718820 0.766884 0.677066
POLYGON 0.245000 0.875000 0.234625 0.923808 0.205296 0.964177 0.162082 0.989127 0.112457 0.994343 0.065000 0.978923 0.027918 0.945534
POLYGON 0.495000 0.875000 0.485866 0.920922 0.459853 0.959853 0.420922 0.985866 0.375000 0.995000 0.329078 0.985866 0.290147 0.959853
POLYGON 0.745000 0.875000 0.736897 0.918349 0.713681 0.955843 0.678489 0.982420 0.636072 0.994488 0.592160 0.990419 0.552684 0.970762
POLYGON 0.995000 0.875000 0.987763 0.916042 0.966925 0.952135 0.935000 0.978923 0.895838 0.993177 0.854162 0.993177 0.815000 0.978923
```

```
0.459853 0.290147
0.645838 0.256823 0.716925 0.297865
0.837918 0.260873 0.912082 0.260873 0.972082 0.304466
0.046417 0.534310 0.107922 0.506221 0.174850 0.515844 0.225950 0.560123
0.271077 0.565000 0.315000 0.521077 0.375000 0.505000 0.435000 0.521077 0.478923 0.565000
0.508487 0.596282 0.535179 0.545425 0.582447 0.512798 0.639464 0.505875 0.693168 0.526242 0.731255 0.569233
0.755000 0.625000 0.766884 0.572934 0.800181 0.531180 0.848297 0.508009 0.901703 0.508009 0.949819 0.531180 0.983116 0.572934
0.007622 0.899949 0.007622 0.850051 0.027918 0.804466 0.065000 0.771077 0.112457 0.755657 0.162082 0.760873 0.205296 0.785823 0.234625
0.264134 0.920922 0.255000 0.875000 0.264134 0.829078 0.290147 0.790147 0.329078 0.764134 0.375000 0.755000 0.420922 0.764134 0.459853
0.522974 0.938172 0.507043 0.897050 0.507043 0.852950 0.522974 0.811828 0.552684 0.779238 0.592160 0.759581 0.636072 0.755512 0.678489
0.783075 0.952135 0.762237 0.916042 0.755000 0.875000 0.762237 0.833958 0.783075 0.797865 0.815000 0.771077 0.854162 0.756823 0.895838
```

```
0.826192
0.790147 0.485866 0.829078
0.767580 0.713681 0.794157 0.736897 0.831651
0.756823 0.935000 0.771077 0.966925 0.797865 0.987763 0.833958
```

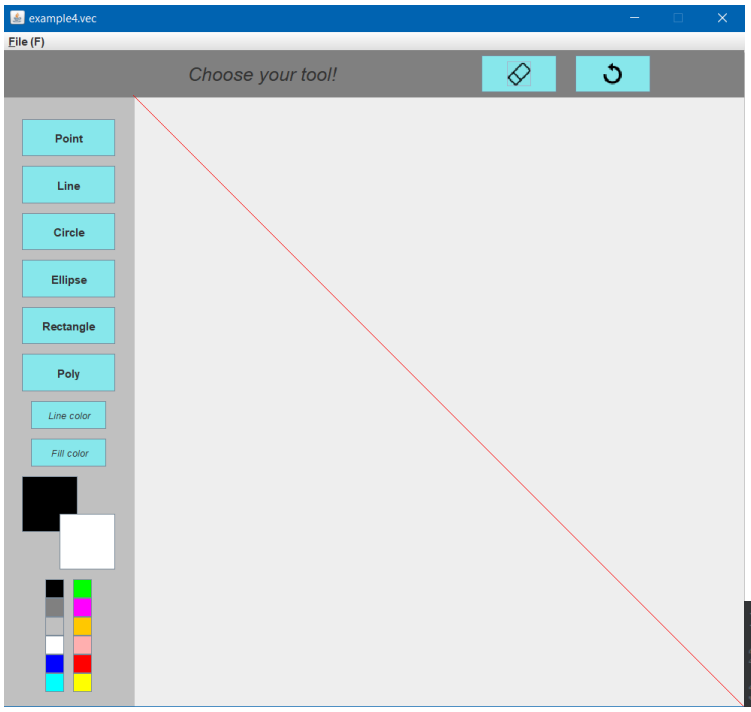
- Example 3


example3.vec
File (F)
Choose your tool!
Point
Line
Circle
Ellipse
Rectangle
Poly
Line color
Fill color

Vec Code:

```
FILL #000000
RECTANGLE 0.0 0.0 1.0 1.0
PEN #FFFFFF
PLOT 0.336492 0.235552
PLOT 0.896505 0.169719
PLOT 0.283804 0.589160
PLOT 0.865037 0.194123
PLOT 0.188216 0.548681
PLOT 0.047656 0.789851
PLOT 0.168696 0.904856
PLOT 0.587655 0.138561
PLOT 0.029540 0.274927
PLOT 0.156462 0.673318
PLOT 0.575948 0.299182
PEN #0000FF
PLOT 0.453672 0.851311
PLOT 0.603206 0.002372
PLOT 0.050746 0.799212
PLOT 0.460386 0.243586
PLOT 0.011023 0.837306
PLOT 0.535768 0.538573
PLOT 0.055615 0.764492
PLOT 0.488703 0.180064
PLOT 0.912895 0.443514
PLOT 0.561466 0.504521
PLOT 0.888733 0.664213
PEN #FF0000
PLOT 0.844652 0.357826
PLOT 0.625279 0.482355
PLOT 0.940967 0.297796
PLOT 0.478004 0.232552
PLOT 0.074824 0.837983
PLOT 0.933897 0.196005
PLOT 0.328905 0.085089
PLOT 0.018094 0.467306
PLOT 0.353928 0.451433
PLOT 0.691917 0.909236
PLOT 0.594122 0.734229
PLOT 0.832187 0.210488
PLOT 0.798462 0.745077
PLOT 0.499853 0.968454
PLOT 0.346504 0.845883
PLOT 0.855385 0.407713
PLOT 0.707696 0.201544
PLOT 0.259062 0.814577
PLOT 0.174336 0.295088
PLOT 0.806156 0.018998
PLOT 0.860874 0.267558
PEN #00FFFF
PLOT 0.790247 0.287432
PLOT 0.846070 0.942759
PLOT 0.362237 0.803654
PLOT 0.249523 0.976133
PLOT 0.958367 0.509307
PLOT 0.980413 0.427336
PLOT 0.907403 0.451916
PLOT 0.873799 0.197717
PLOT 0.995358 0.945321
PLOT 0.938648 0.196736
PLOT 0.665500 0.785974
PLOT 0.481761 0.805478
PLOT 0.962083 0.488273
PLOT 0.943092 0.190746
PEN #00FF00
PLOT 0.288151 0.003051
PLOT 0.380480 0.289850
PLOT 0.540900 0.665621
PLOT 0.675587 0.834024
PLOT 0.212548 0.183193
PLOT 0.697902 0.607534
PLOT 0.600272 0.316250
PLOT 0.859946 0.249876
PLOT 0.355729 0.534429
PLOT 0.724348 0.300078
PLOT 0.222814 0.334678
PLOT 0.582255 0.854941
PLOT 0.961742 0.097945
PLOT 0.212588 0.167211
PEN #FF00FF
PLOT 0.138183 0.330370
PLOT 0.352927 0.744955
PLOT 0.668118 0.885944
PLOT 0.765362 0.030082
PLOT 0.752226 0.299804
PLOT 0.164427 0.531754
PLOT 0.108043 0.674834
PLOT 0.689281 0.369559
PLOT 0.213782 0.658948
PLOT 0.048574 0.541660
PLOT 0.614641 0.926413
PLOT 0.944003 0.874742
PLOT 0.210552 0.233812
PLOT 0.887313 0.535415
PLOT 0.843784 0.559694
PEN #FFFF00
PLOT 0.065583 0.186858
PLOT 0.435438 0.056717
PLOT 0.480879 0.359815
PLOT 0.381813 0.546039
PLOT 0.192926 0.373676
PLOT 0.914763 0.065955
PLOT 0.127249 0.191206
PLOT 0.654603 0.550251
PLOT 0.576219 0.919617
PLOT 0.639769 0.056473
PLOT 0.216798 0.280989
PLOT 0.704353 0.090278
PLOT 0.196271 0.266697
PLOT 0.192809 0.479269
FILL OFF
PEN #FFFFFF
RECTANGLE 0.115000 0.115000 0.140000 0.140000
RECTANGLE 0.187727 0.115000 0.212727 0.140000
RECTANGLE 0.260455 0.115000 0.285455 0.140000
RECTANGLE 0.405909 0.115000 0.430909 0.140000
RECTANGLE 0.478636 0.115000 0.503636 0.140000
RECTANGLE 0.551364 0.115000 0.576364 0.140000
RECTANGLE 0.696818 0.115000 0.721818 0.140000
RECTANGLE 0.769545 0.115000 0.794545 0.140000
RECTANGLE 0.115000 0.187727 0.140000 0.212727
RECTANGLE 0.405909 0.187727 0.430909 0.212727
RECTANGLE 0.551364 0.187727 0.576364 0.212727
RECTANGLE 0.696818 0.187727 0.721818 0.212727
RECTANGLE 0.842273 0.187727 0.867273 0.212727
RECTANGLE 0.115000 0.260455 0.140000 0.285455
RECTANGLE 0.405909 0.260455 0.430909 0.285455
RECTANGLE 0.478636 0.260455 0.503636 0.285455
RECTANGLE 0.551364 0.260455 0.576364 0.285455
RECTANGLE 0.696818 0.260455 0.721818 0.285455
RECTANGLE 0.769545 0.260455 0.794545 0.285455
RECTANGLE 0.115000 0.333182 0.140000 0.358182
RECTANGLE 0.405909 0.333182 0.430909 0.358182
RECTANGLE 0.551364 0.333182 0.576364 0.358182
RECTANGLE 0.696818 0.333182 0.721818 0.358182
RECTANGLE 0.842273 0.333182 0.867273 0.358182
RECTANGLE 0.115000 0.405909 0.140000 0.430909
RECTANGLE 0.187727 0.405909 0.212727 0.430909
RECTANGLE 0.260455 0.405909 0.285455 0.430909
RECTANGLE 0.405909 0.405909 0.430909 0.430909
RECTANGLE 0.551364 0.405909 0.576364 0.430909
RECTANGLE 0.696818 0.405909 0.721818 0.430909
RECTANGLE 0.769545 0.405909 0.794545 0.430909
RECTANGLE 0.115000 0.551364 0.140000 0.576364
RECTANGLE 0.187727 0.551364 0.212727 0.576364
RECTANGLE 0.260455 0.551364 0.285455 0.576364
RECTANGLE 0.405909 0.551364 0.430909 0.576364
RECTANGLE 0.478636 0.551364 0.503636 0.576364
RECTANGLE 0.551364 0.551364 0.576364 0.576364
RECTANGLE 0.696818 0.551364 0.721818 0.576364
RECTANGLE 0.769545 0.551364 0.794545 0.576364
RECTANGLE 0.842273 0.551364 0.867273 0.576364
RECTANGLE 0.260455 0.624091 0.285455 0.649091
RECTANGLE 0.405909 0.624091 0.430909 0.649091
RECTANGLE 0.551364 0.624091 0.576364 0.649091
RECTANGLE 0.842273 0.624091 0.867273 0.649091
RECTANGLE 0.115000 0.696818 0.140000 0.721818
RECTANGLE 0.187727 0.696818 0.212727 0.721818
RECTANGLE 0.260455 0.696818 0.285455 0.721818
RECTANGLE 0.405909 0.696818 0.430909 0.721818
RECTANGLE 0.551364 0.696818 0.576364 0.721818
RECTANGLE 0.696818 0.696818 0.721818 0.721818
RECTANGLE 0.769545 0.696818 0.794545 0.721818
RECTANGLE 0.842273 0.696818 0.867273 0.721818
RECTANGLE 0.260455 0.769545 0.285455 0.794545
RECTANGLE 0.405909 0.769545 0.430909 0.794545
RECTANGLE 0.551364 0.769545 0.576364 0.794545
RECTANGLE 0.696818 0.769545 0.721818 0.794545
RECTANGLE 0.115000 0.842273 0.140000 0.867273
RECTANGLE 0.187727 0.842273 0.212727 0.867273
RECTANGLE 0.260455 0.842273 0.285455 0.867273
RECTANGLE 0.405909 0.842273 0.430909 0.867273
RECTANGLE 0.478636 0.842273 0.503636 0.867273
RECTANGLE 0.551364 0.842273 0.576364 0.867273
RECTANGLE 0.696818 0.842273 0.721818 0.867273
RECTANGLE 0.769545 0.842273 0.794545 0.867273
RECTANGLE 0.842273 0.842273 0.867273 0.867273
```

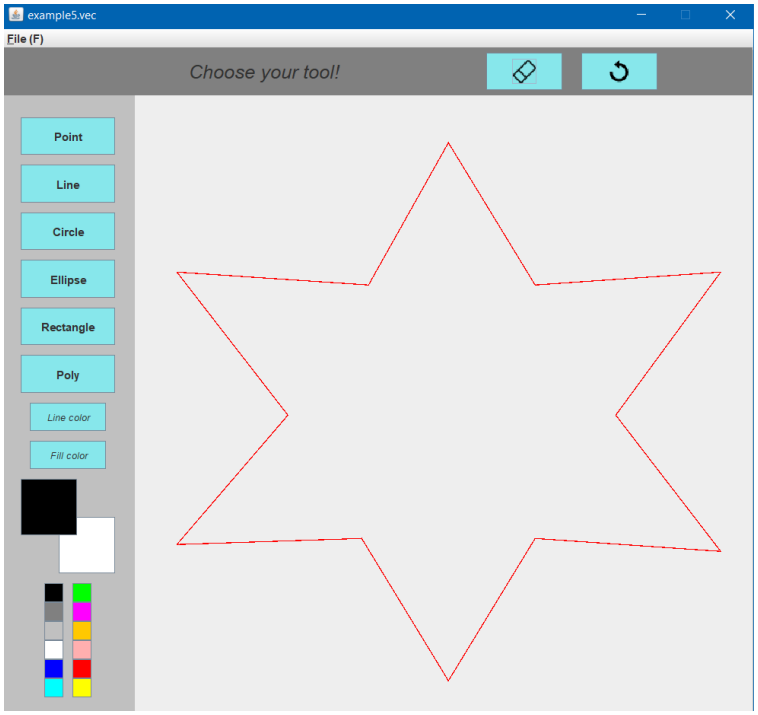
- Example 4



Vec Code:

```
PEN #FF0000
FILL #0000FF
LINE 0.0 0.0 1.0 1.0
```

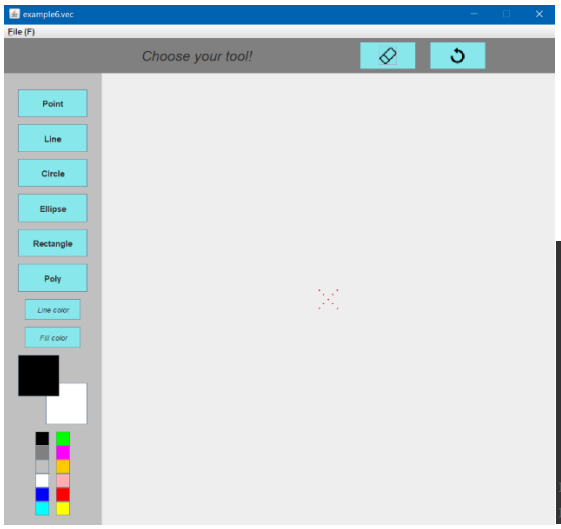
- Example 5



Vec Code:

```
PEN #FF0000
FILL #0000FF
LINE 0.5092308 0.08 0.6492308 0.30923077
PEN #FF0000
FILL #0000FF
LINE 0.6492308 0.30923077 0.9492308 0.28923076
PEN #FF0000
FILL #0000FF
LINE 0.9492308 0.28923076 0.78 0.52
PEN #FF0000
FILL #0000FF
LINE 0.78 0.52 0.9492308 0.74
PEN #FF0000
FILL #0000FF
LINE 0.9492308 0.74 0.6492308 0.72
PEN #FF0000
FILL #0000FF
LINE 0.6492308 0.72 0.5092308 0.9492308
PEN #FF0000
FILL #0000FF
LINE 0.5092308 0.9492308 0.36923078 0.72
PEN #FF0000
FILL #0000FF
LINE 0.36923078 0.72 0.06923077 0.72923076
PEN #FF0000
FILL #0000FF
LINE 0.06923077 0.72923076 0.24923077 0.52
PEN #FF0000
FILL #0000FF
LINE 0.24923077 0.52 0.06923077 0.28923076
PEN #FF0000
FILL #0000FF
LINE 0.06923077 0.28923076 0.38 0.30923077
PEN #FF0000
FILL #0000FF
LINE 0.38 0.30923077 0.5092308 0.08
```

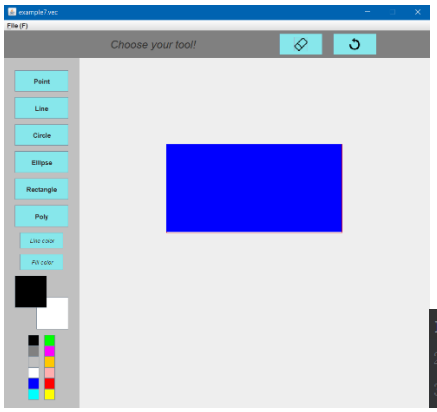
- Example 6



Vec Code:

```
PEN #FF0000
FILL #0000FF
PLOT 0.5 0.5
PLOT 0.49 0.49
PLOT 0.48 0.48
PLOT 0.51 0.49
PLOT 0.52 0.48
PLOT 0.51 0.51
PLOT 0.52 0.52
PLOT 0.49 0.51
PLOT 0.48 0.52
```

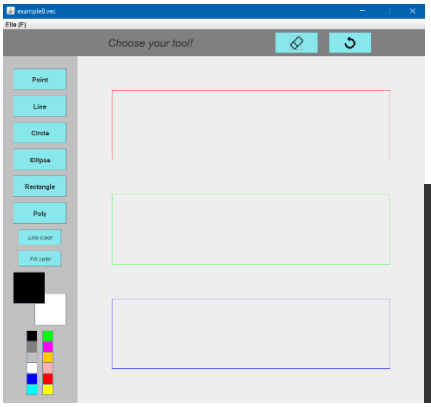
- Example 7



Vec Code:

```
PEN #FF0000
FILL #0000FF
RECTANGLE 0.25 0.25 0.75 0.5
```

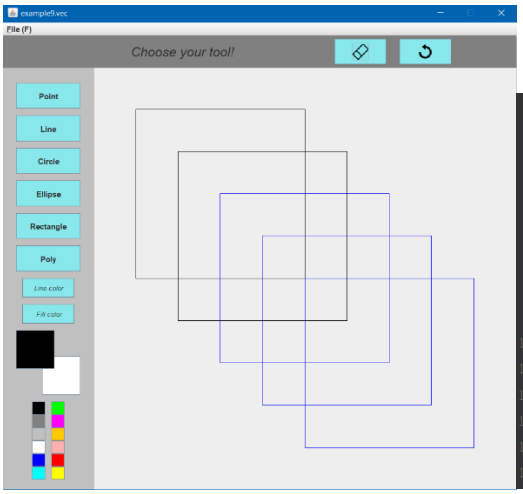
- Example 8



Vec Code:

```
PEN #FF0000
FILL OFF
RECTANGLE 0.1 0.1 0.9 0.3
PEN #00FF00
FILL OFF
RECTANGLE 0.1 0.4 0.9 0.6
PEN #0000FF
FILL OFF
RECTANGLE 0.1 0.7 0.9 0.9
```

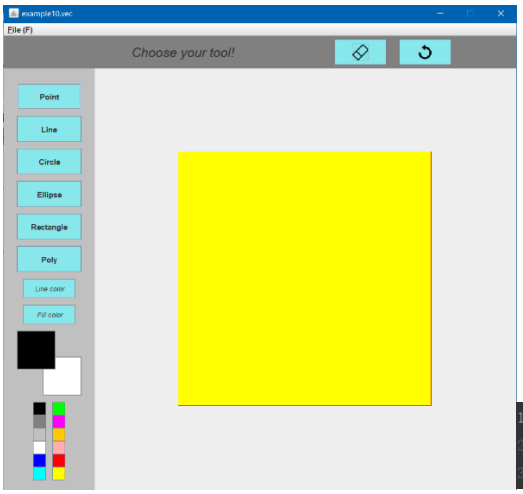
- Example 9



Vec Code:

```
PEN #000000
FILL OFF
RECTANGLE 0.1 0.1 0.5 0.5
PEN #000000
FILL OFF
RECTANGLE 0.2 0.2 0.6 0.6
PEN #0000FF
FILL OFF
RECTANGLE 0.3 0.3 0.7 0.7
PEN #0000FF
FILL OFF
RECTANGLE 0.4 0.4 0.8 0.8
PEN #0000FF
FILL OFF
RECTANGLE 0.5 0.5 0.9 0.9
```

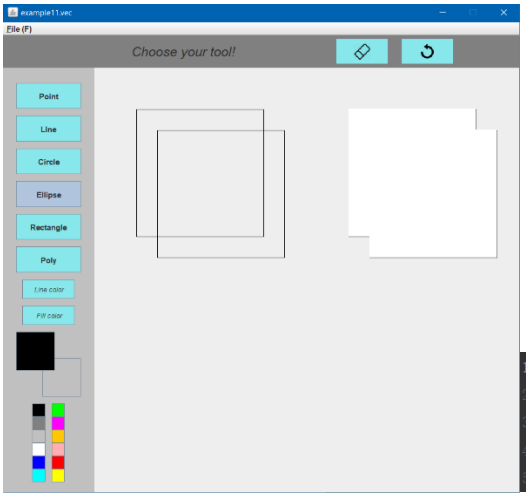
- Example 10



Vec Code:

```
PEN #FF0000
FILL #FFFF00
RECTANGLE 0.2 0.2 0.8 0.8
```

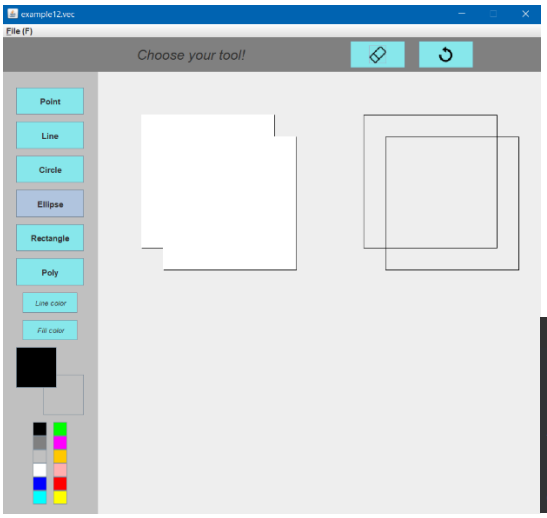
- Example 11



Vec Code:

```
RECTANGLE 0.1 0.1 0.4 0.4
RECTANGLE 0.15 0.15 0.45 0.45
FILL #FFFFFF
RECTANGLE 0.6 0.1 0.9 0.4
RECTANGLE 0.65 0.15 0.95 0.45
```

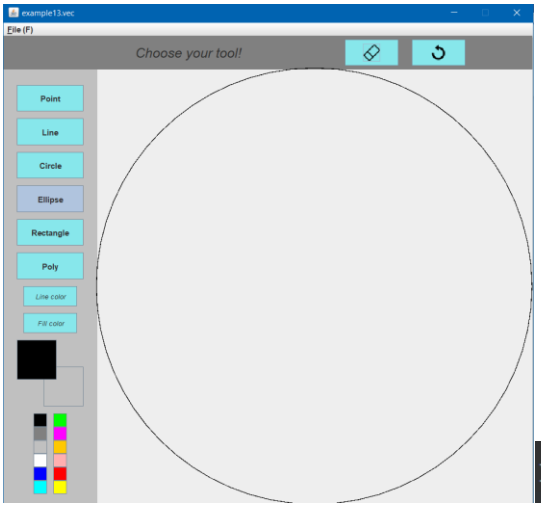
- Example 12



Vec Code:

```
FILL #FFFFFF
RECTANGLE 0.1 0.1 0.4 0.4
RECTANGLE 0.15 0.15 0.45 0.45
FILL OFF
RECTANGLE 0.6 0.1 0.9 0.4
RECTANGLE 0.65 0.15 0.95 0.45
```

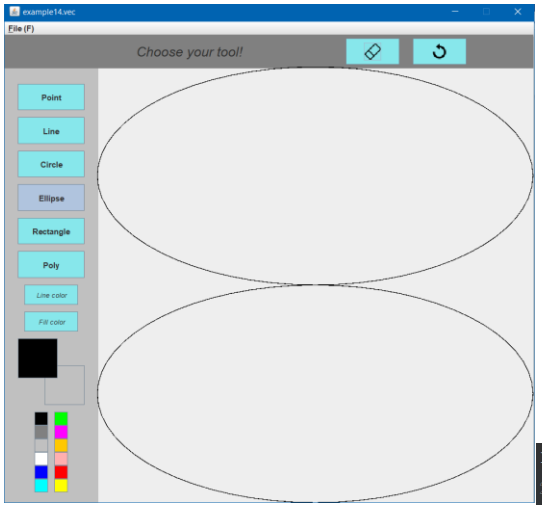
- Example 13



Vec Code:

```
ELLIPSE 0.0 0.0 1.0 1.0
```

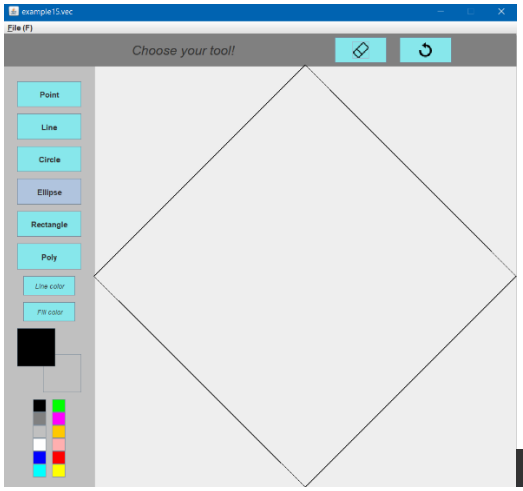
- Example 14



Vec Code:

```
1  ELLIPSE 0.0 0.0 1.0 0.5
2  ELLIPSE 0.0 0.5 1.0 1.0
```

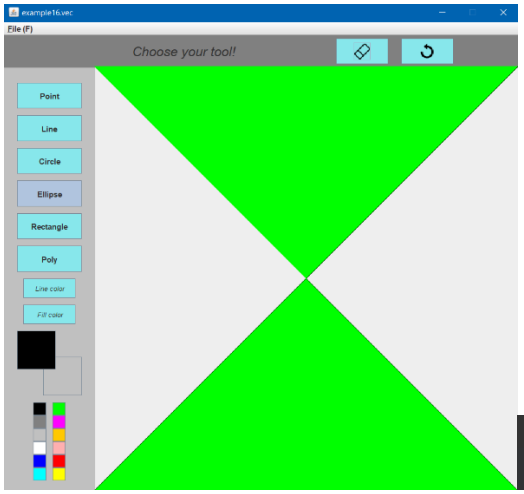
- Example 15



Vec Code:

```
1  POLYGON 0.5 0.0 1.0 0.5 0.5 1.0 0.0 0.5
```

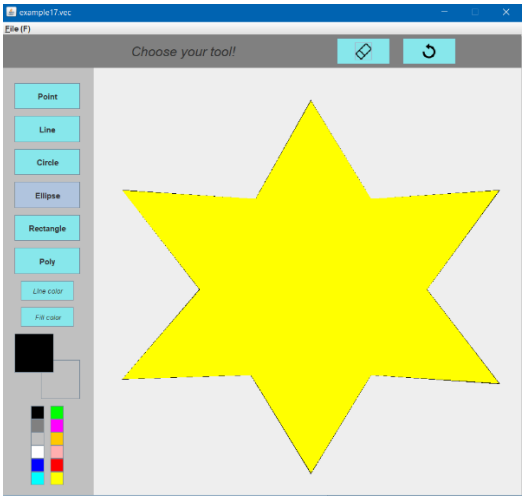
- Example 16



Vec Code:

```
1  FILL #00FF00
2  POLYGON 0.0 0.0 1.0 0.0 0.0 1.0 1.0 1.0
```

- Example 17



Vec Code:

```
1  FILL #FFFF00
2  POLYGON 0.51 0.08 0.65 0.31 0.95 0.29 0.78 0.52 0.95 0.74 0.65 0.72 0.51 0.95 0.37 0.72 0.07 0.73 0.25 0.52 0.07 0.29 0.38 0.31
```