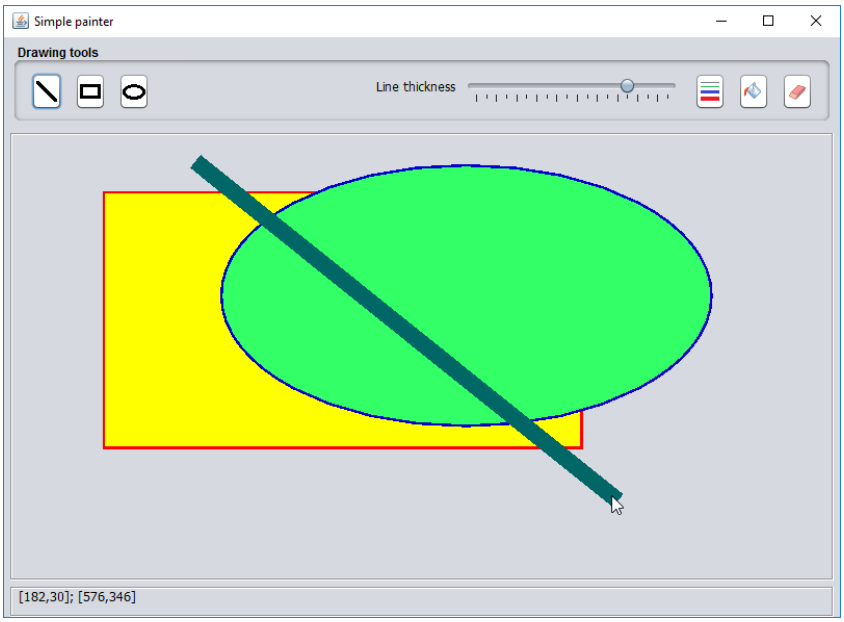
Slot 07

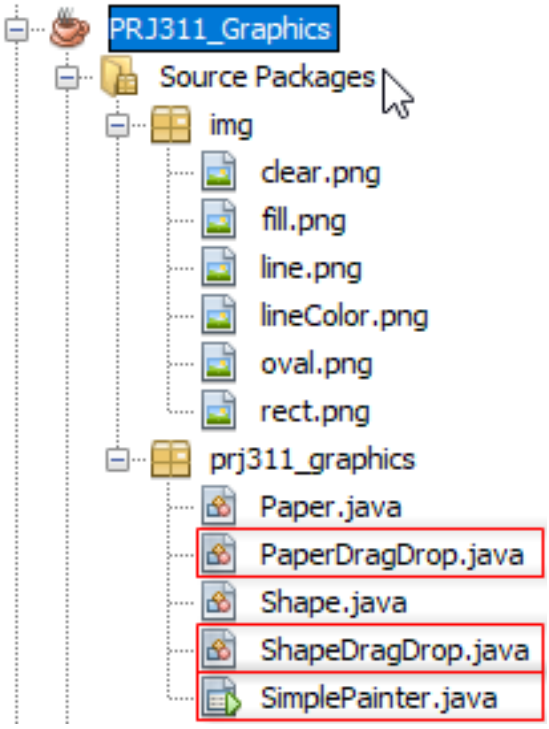
2D Graphics – Drag & Drop

Write simple painting program with following interface



- The program can draw lines, rectangles and ovals.
- User can select line color and fill color.
- When **Clear paper** button is clicked, the picture will be cleared.

The project



Step by step

Base on the given project, creates 3 new classes called **ShapeDragDrop**, **PaperDragDrop** and **SimplePainter** as below

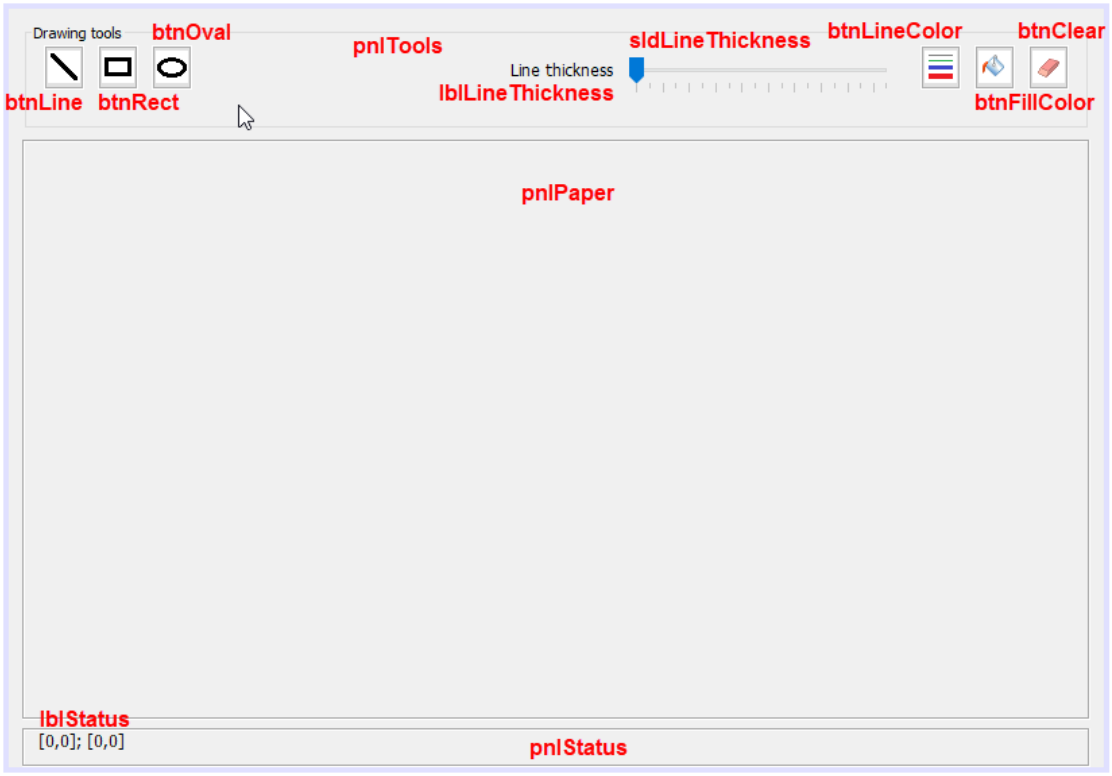
B1: Creates a class **ShapeDragDrop** base on class **Shape** that used to draw a shape include line or rectangle or oval

```
1   package prj311_graphics;
2   import java.awt.BasicStroke;
3   import java.awt.Color;
4   import java.awt.Graphics;
5   import java.awt.Graphics2D;
6
7   public class ShapeDragDrop extends Shape {
8       public ShapeDragDrop(String type, int x, int y, int width, int height,
9                            int lineThickness, Color line, Color fill) {
10          super(type, x, y, width, height, lineThickness, line, fill);
11      }
12      @Override
13      public void draw(Graphics g) {
14          Graphics2D g2D = (Graphics2D) g;
15          //draws filled shape
16          g2D.setColor(fill);
17          if (type.equals("rect")) {
18              g2D.fillRect(x, y, width, height);
19          } else  if (type.equals("oval")) {
20              g2D.fillOval(x, y, width, height);
21          }
22          //draws stroke
23          g2D.setStroke(new BasicStroke(lineThickness));
24          g2D.setColor(line);
25          if (type.equals("rect")) {
26              g2D.drawRect(x, y, width, height);
27          } else  if (type.equals("oval")) {
28              g2D.drawOval(x, y, width, height);
29          } else if (type.equals("line")) {
30              g2D.drawLine(x, y, x+width-1, y+height-1);
31          }
32      }
33  }
```

B2: Creates a class **PaperDragDrop** base on class **Paper** that extends from **JPanel** and uses to manage a list of shape

```
1   package prj311_graphics;
2
3   import java.awt.Color;
4   import java.awt.Graphics;
5
6   /**
7    *
8    * @author KhanhVH@fe.edu.vn
9    */
10  public class PaperDragDrop extends Paper {
11      public void removeLast() {
12          if (shapes.size() > 0) {
13              shapes.remove(shapes.size() - 1);
14          }
15      }
16
17      public void addShape(String type, int x, int y, int width, int height,
18                           int lineThickness, Color line, Color fill) {
19          shapes.add(new ShapeDragDrop(type, x, y, width, height, lineThickness, line, fill));
20          repaint();
21      }
22
23      @Override
24      public void paint(Graphics g) {
25          super.paintComponent(g);
26          for (Object s : shapes) {
27              ((ShapeDragDrop)s).draw(g);
28          }
29      }
30  }
```

B3: Add a JFrame that named **SimplePainter**. You must design the interface as below:



B4: Set form title is "Simple Painter".

B5: Updates a main frame to draw simple shape as below:

```
public class SimplePainter extends javax.swing.JFrame {
    private PaperDragDrop p;
    private int startX, startY, endX, endY;  // of a rectangle
    private Color lineColor, fillColor;
    private String type;
    private MouseInputAdapter dragDropListener;
    /**
     * Creates new form SimplePainter
     */
    public SimplePainter() {
        initComponents();
        this.setLocationRelativeTo(null);

        p = new PaperDragDrop();
        pnlPaper.setSize(800, 400);
        pnlPaper.setLayout(new BorderLayout());
        pnlPaper.add(p, BorderLayout.CENTER);
```

```
        dragDropListener = new MouseInputAdapter() {
            @Override
            public void mousePressed(MouseEvent evt) {
                startX = evt.getX();
                startY = evt.getY();
                lblStatus.setText("[" + startX + "," + startY + "]; [" + startX + "," + startY + "]");
                p.addShape(type, startX, startY, 1, 1, sldLineThickness.getValue(), lineColor, fillColor);
            }
            @Override
            public void mouseDragged(MouseEvent evt) {
               endX = evt.getX();
               endY = evt.getY();
               lblStatus.setText("[" + startX + "," + startY + "]; [" + endX + "," + endY + "]");
               p.removeLast();
               p.addShape(type, startX, startY, endX-startX+1, endY-startY+1, sldLineThickness.getValue(), lineColor, fillColor);
            }
            @Override
               public void mouseReleased(MouseEvent evt) {
               endX = evt.getX();
               endY = evt.getY();
               lblStatus.setText("[" + startX + "," + startY + "]; [" + endX + "," + endY + "]");
               p.removeLast();
               p.addShape(type, startX, startY, endX-startX+1, endY-startY+1, sldLineThickness.getValue(), lineColor, fillColor);
            }
        };

        p.addMouseListener(dragDropListener);
        p.addMouseMotionListener(dragDropListener);
    }
```

```
    private void btnRectActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        type = "rect";
    }

    private void btnOvalActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        type = "oval";
    }

    private void btnClearActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        p.clear();
    }

    private void btnLineActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        type = "line";
    }

    private void btnLineColorActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        lineColor = JColorChooser.showDialog(this, "Choose line color", Color.BLACK);
    }

    private void btnFillColorActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        fillColor = JColorChooser.showDialog(this, "Choose fill color", Color.WHITE);
    }
```