```java
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.util.Random;

import javax.swing.JPanel;
import javax.swing.JFrame;

public class LineArt extends JPanel {

    // Unique version ID for this class to ensure saved objects can be loaded safely
    private static final long serialVersionUID = 1L;

    // Initial width of height of the starting rectangle
    private static int width = 980;
    private static int height = 630;

    // main method to launch the program as a standalone application - no need to
    // modify
    public static void main(String[] args) {
        LineArt panel = new LineArt();
        panel.setPreferredSize(new Dimension(width + 20, height + 20)); // content size window dimensions

        JFrame frame = new JFrame("Line Art"); // Title of frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.add(panel);
        frame.pack();
        frame.setVisible(true);
    }


    /**
     * Draw the four corners of line art.  Line art displays straight lines inside a rectangle from one side to
     * a perpendicular side. The lines must be drawn in such a way that both the starting points of the lines on
     * one side and the ending points on the other side are equi-distant along the sides.  The size of the rectangle
     * is 980 pixels wide by 630 pixels high.
     *
     * @param g the Graphics object used for drawing shapes, text, and images
     */
    public void drawLineArt(Graphics g) {

        // Draw the initial rectangle
        g.drawRect(10, 10, width, height);
```

```
43
44          //Box with movable k to find center
45          int k = 540; // Used to adjust the size of the rectangle
      centered on origin
46          g.drawRect(500-k/2, 330-k/2*65/100, k, k*65/100-11);
47//        g.drawLine(0, 325, 1000, 325); Middle Lines for Ref -
      Keeping for future Ref
48//        g.drawLine(500, 0, 500, 650);  Middle Lines for Ref -
      Keeping for future Ref
49
50          // Draw bottom-left corner
51          int y = 640;
52          for (int x=890;x>=10;x-=20) {
53              g.setColor(randomColor());
54              g.drawLine(x, 640, 10, y);
55              y-=(20*0.65);
56          }
57
58          y = 495;
59          for (int x=625;x>=230;x-=10) {
60              g.setColor(randomColor());
61              g.drawLine(x, 495, 230, y);
62              y-=(10*0.65);
63          }
64
65          // Draw bottom-right corner
66          y = 640;
67          for (int x=110;x<=990;x+=20) {
68              g.setColor(randomColor());
69              g.drawLine(x, 640, 990, y);
70              y-=(20*0.65);
71          }
72
73          y = 495;
74          for (int x=375;x<=770;x+=10) {
75              g.setColor(randomColor());
76              g.drawLine(x, 495, 770, y);
77              y-=(10*0.65);
78          }
79
80          // Draw top-left corner
81          y = 10;
82          for (int x=890;x>=10;x-=20) {
83              g.setColor(randomColor());
84              g.drawLine(x, 10, 10, y);
85              y+=(20*0.65);
86          }
87
88          y = 155;
```

```
 89          for (int x=625;x>=230;x-=10) {
 90              g.setColor(randomColor());
 91              g.drawLine(x, 155, 230, y);
 92              y+=(10*0.65);
 93          }
 94
 95          // Draw top-right corner
 96          y = 10;
 97          for (int x=110;x<=990;x+=20) {
 98              g.setColor(randomColor());
 99              g.drawLine(x, 10, 990, y);
100              y+=(20*0.65);
101          }
102
103          y = 155;
104          for (int x=375;x<=770;x+=10) {
105              g.setColor(randomColor());
106              g.drawLine(x, 155, 770, y);
107              y+=(10*0.65);
108          }
109
110      }
111
112      public static Color randomColor() {
113          Random rand = new Random();
114          int randr = rand.nextInt(256);
115          int randg = rand.nextInt(256);
116          int randb = rand.nextInt(256);
117          return new Color(randr, randg, randb);
118
119      }
120
121      /**
122       * Overrides JPanel's paintComponent method to perform custom
    drawing.
123       *
124       * @param g the Graphics object used for drawing shapes,
    text, and images
125       */
126      @Override
127      protected void paintComponent(Graphics g) {
128          super.paintComponent(g); // Clears the panel before
    drawing
129          drawLineArt(g);
130      }
131
132  }
```