```
import java.awt.Dimension;
import java.awt.Graphics;
import javax.swing.JPanel;
import javax.swing.JFrame;
import java.awt.Color;

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 (980,
630)
	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
equidistant 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) {
		int run = 7;
		int pointnum = 100;
		double gapw = 0;
		double gaph = 0;
```

```
                int startx = 0;
                int starty = 0;
                int x = 0;
                int y = 0;
                int runtime = 0;
                int offsetx = 10;
                int offsety = 10;
                int tempwidth = width;
                int tempheight = height;
                double c = 1.98;
                int count = 0;
                int keepx = 0;
                int keepy = 0;
                //color
                int red = 1;
                int green = 2;
                int blue = 3;
                int cdist = 7;
                int limhighr = 255;
                int limhighg = 254;
                int limhighb = 253;
                int limlowr = 1;
                int limlowg = 1;
                int limlowb = 1;
                int dred = 1;
                int dgreen = 1;
                int dblue = 1;
                Color color = new Color(red,green,blue);
                Color back = new Color(255,255,255);
                g.setColor(back);
                g.fillRect(0, 0, width + 20, height + 20);
                //

                while (run > 0) {

                        // Draw the initial rectangle
                        color = new Color(0,0,0);
                        g.setColor(color);
                        g.drawRect(offsetx, offsety, tempwidth,
tempheight);
                        System.out.println("(" + offsetx + "," +
offsety + ")");
                        System.out.println("Width: " + tempwidth);
                        System.out.println("Height: " + tempheight);
                        System.out.println();

                        // Draw bottom-left corner
                        gapw = 1.0 * tempwidth / (pointnum + 1);
                        gaph = 1.0 * tempheight / (pointnum + 1);
                        startx = offsetx;
```

```
                        starty = tempheight + offsety - 1;
                        x = (int) Math.round(1.0 * startx + gapw);
                        y = (int) Math.round(1.0 * offsety + gaph) ;
                        runtime = pointnum;
                        count = 2;
                        while (runtime > 0) {
                                //color
                                if (red >= (limhighr - cdist)) {
                                        red = limhighr - 1 - cdist;
                                        dred = -1;
                                }
                                else if (red <= (limlowr + cdist)) {
                                        red = limlowr + 1 + cdist;
                                        dred = 1;
                                }
                                else {
                                        red = red + (dred * cdist);
                                }
                                if (green >= (limhighg - cdist)) {
                                        green = limhighg - 1 - cdist;
                                        dgreen = -1;
                                }
                                else if (green <= (limlowg + cdist)) {
                                        green = limlowg + 1 + cdist;
                                        dgreen = 1;
                                }
                                else {
                                        green = green + (dgreen *
cdist);
                                }
                                if (blue >= (limhighb - cdist)) {
                                        blue = limhighb - 1 - cdist;
                                        dblue = -1;
                                }
                                else if (blue <= (limlowb + cdist)) {
                                        blue = limlowb + 1 + cdist;
                                        dblue = 1;
                                }
                                else {
                                        blue = blue + (dblue * cdist);
                                }
                                color = new Color(red, green, blue);
                                g.setColor(color);
                                //
                                g.drawLine(startx, y, x, starty);
                                x = (int) Math.round(1.0 * startx +
(count * gapw));
                                y = (int) Math.round(1.0 * offsety +
(count * gaph));
                                runtime--;
```

```
                count++;
        }
        // Draw bottom-right corner
        startx = tempwidth + offsetx - 1;
        starty = tempheight + offsety - 1;
        x = (int) Math.round(1.0 * offsetx + gapw);
        y = (int) Math.round(1.0 * starty - gaph);
        runtime = pointnum;
        count = 2;
        while (runtime > 0) {
                //color
                if (red >= (limhighr - cdist)) {
                        red = limhighr - 1 - cdist;
                        dred = -1;
                }
                else if (red <= (limlowr + cdist)) {
                        red = limlowr + 1 + cdist;
                        dred = 1;
                }
                else {
                        red = red + (dred * cdist);
                }
                if (green >= (limhighg - cdist)) {
                        green = limhighg - 1 - cdist;
                        dgreen = -1;
                }
                else if (green <= (limlowg + cdist)) {
                        green = limlowg + 1 + cdist;
                        dgreen = 1;
                }
                else {
                        green = green + (dgreen *
cdist);
                }
                if (blue >= (limhighb - cdist)) {
                        blue = limhighb - 1 - cdist;
                        dblue = -1;
                }
                else if (blue <= (limlowb + cdist)) {
                        blue = limlowb + 1 + cdist;
                        dblue = 1;
                }
                else {
                        blue = blue + (dblue * cdist);
                }
                color = new Color(red, green, blue);
                g.setColor(color);
                //
                g.drawLine(startx, y, x, starty);
                x = (int) Math.round(1.0 * offsetx +
```

```
(count * gapw));
                                y = (int) Math.round(1.0 * starty -
(count * gaph));
                                runtime--;
                                count++;
                        }
                        // Draw top-right corner
                        startx = tempwidth + offsetx - 1;
                        starty = offsety;
                        x = (int) Math.round(1.0 * startx - gapw);
                        y = (int) Math.round(1.0 * offsety +
tempheight - 1 - gaph);
                        runtime = pointnum;
                        count = 2;
                        while (runtime > 0) {
                                //color
                                if (red >= (limhighr - cdist)) {
                                        red = limhighr - 1 - cdist;
                                        dred = -1;
                                }
                                else if (red <= (limlowr + cdist)) {
                                        red = limlowr + 1 + cdist;
                                        dred = 1;
                                }
                                else {
                                        red = red + (dred * cdist);
                                }
                                if (green >= (limhighg - cdist)) {
                                        green = limhighg - 1 - cdist;
                                        dgreen = -1;
                                }
                                else if (green <= (limlowg + cdist)) {
                                        green = limlowg + 1 + cdist;
                                        dgreen = 1;
                                }
                                else {
                                        green = green + (dgreen *
cdist);
                                }
                                if (blue >= (limhighb - cdist)) {
                                        blue = limhighb - 1 - cdist;
                                        dblue = -1;
                                }
                                else if (blue <= (limlowb + cdist)) {
                                        blue = limlowb + 1 + cdist;
                                        dblue = 1;
                                }
                                else {
                                        blue = blue + (dblue * cdist);
                                }
```

```
                                color = new Color(red, green, blue);
                                g.setColor(color);
                                //
                                g.drawLine(startx, y, x, starty);
                                x = (int) Math.round(1.0 * startx -
(count * gapw));
                                y = (int) Math.round(1.0 * offsety +
tempheight - 1 - (count * gaph));
                                runtime--;
                                count++;
                        }
                        // Draw top-left corner
                        startx = offsetx;
                        starty = offsety;
                        x = (int) Math.round(1.0 * offsetx + tempwidth
- 1 - gapw);
                        y = (int) Math.round(1.0 * starty + gaph);
                        runtime = pointnum;
                        count = 2;
                        while (runtime > 0) {
                                //color
                                if (red >= (limhighr - cdist)) {
                                        red = limhighr - 1 - cdist;
                                        dred = -1;
                                }
                                else if (red <= (limlowr + cdist)) {
                                        red = limlowr + 1 + cdist;
                                        dred = 1;
                                }
                                else {
                                        red = red + (dred * cdist);
                                }
                                if (green >= (limhighg - cdist)) {
                                        green = limhighg - 1 - cdist;
                                        dgreen = -1;
                                }
                                else if (green <= (limlowg + cdist)) {
                                        green = limlowg + 1 + cdist;
                                        dgreen = 1;
                                }
                                else {
                                        green = green + (dgreen *
cdist);
                                }
                                if (blue >= (limhighb - cdist)) {
                                        blue = limhighb - 1 - cdist;
                                        dblue = -1;
                                }
                                else if (blue <= (limlowb + cdist)) {
                                        blue = limlowb + 1 + cdist;
```

```
                                        dblue = 1;
                                }
                                else {
                                        blue = blue + (dblue * cdist);
                                }
                                color = new Color(red, green, blue);
                                g.setColor(color);
                                //
                                g.drawLine(startx, y, x, starty);
                                x = (int) Math.round(1.0 * offsetx +
tempwidth - (count * gapw));
                                y = (int) Math.round(1.0 * starty +
(count * gaph));
                                runtime--;
                                count++;
                        }

                        run--;
                        keepx = offsetx;
                        keepy = offsety;
                        offsetx = (int) Math.round(offsetx +
(tempwidth / (2 * c)));
                        offsety = (int) Math.round(offsety +
(tempheight / (2 * c)));
                        tempwidth = (int) Math.round(tempwidth - (2 *
(offsetx - keepx)));
                        tempheight = (int) Math.round(tempheight - (2
* (offsety - keepy)));
                }
        }

        /**
         * Overrides JPanel's paintComponent method to perform custom
drawing.
         *
         * @param g the Graphics object used for drawing shapes, text,
and images
         */
        @Override
        protected void paintComponent(Graphics g) {
                super.paintComponent(g); // Clears the panel before
drawing
                drawLineArt(g);
        }

}
```