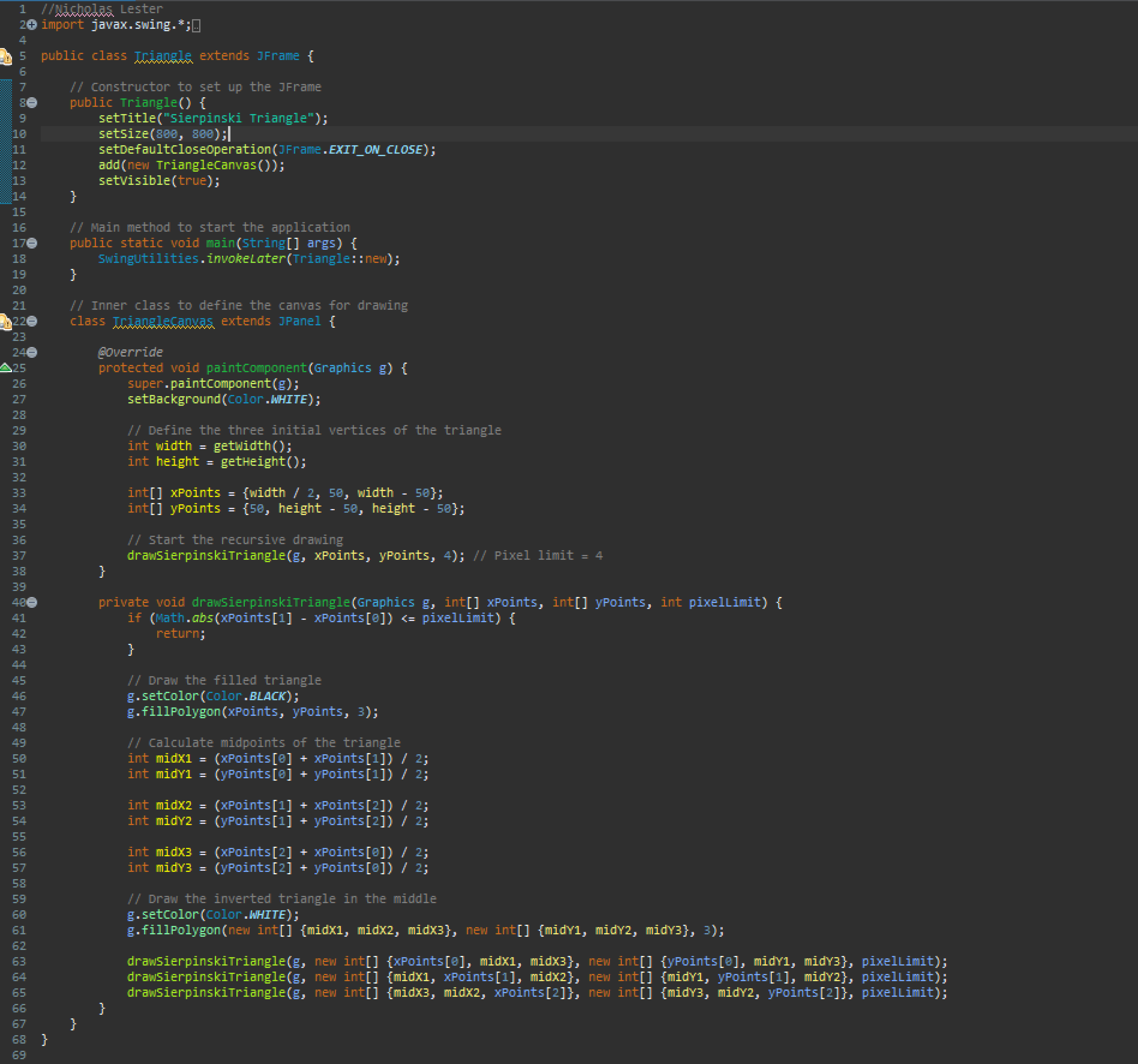

```java
//Nicholas Lester
import javax.swing.*;

public class Triangle extends JFrame {

    // Constructor to set up the JFrame
    public Triangle() {
        setTitle("Sierpinski Triangle");
        setSize(800, 800);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        add(new TriangleCanvas());
        setVisible(true);
    }

    // Main method to start the application
    public static void main(String[] args) {
        SwingUtilities.invokeLater(Triangle::new);
    }

    // Inner class to define the canvas for drawing
    class TriangleCanvas extends JPanel {

        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            setBackground(Color.WHITE);

            // Define the three initial vertices of the triangle
            int width = getWidth();
            int height = getHeight();

            int[] xPoints = {width / 2, 50, width - 50};
            int[] yPoints = {50, height - 50, height - 50};

            // Start the recursive drawing
            drawSierpinskiTriangle(g, xPoints, yPoints, 4); // Pixel limit = 4
        }

        private void drawSierpinskiTriangle(Graphics g, int[] xPoints, int[] yPoints, int pixelLimit) {
            if (Math.abs(xPoints[1] - xPoints[0]) <= pixelLimit) {
                return;
            }

            // Draw the filled triangle
            g.setColor(Color.BLACK);
            g.fillPolygon(xPoints, yPoints, 3);

            // Calculate midpoints of the triangle
            int midX1 = (xPoints[0] + xPoints[1]) / 2;
            int midY1 = (yPoints[0] + yPoints[1]) / 2;

            int midX2 = (xPoints[1] + xPoints[2]) / 2;
            int midY2 = (yPoints[1] + yPoints[2]) / 2;

            int midX3 = (xPoints[2] + xPoints[0]) / 2;
            int midY3 = (yPoints[2] + yPoints[0]) / 2;

            // Draw the inverted triangle in the middle
            g.setColor(Color.WHITE);
            g.fillPolygon(new int[] {midX1, midX2, midX3}, new int[] {midY1, midY2, midY3}, 3);

            drawSierpinskiTriangle(g, new int[] {xPoints[0], midX1, midX3}, new int[] {yPoints[0], midY1, midY3}, pixelLimit);
            drawSierpinskiTriangle(g, new int[] {midX1, xPoints[1], midX2}, new int[] {midY1, yPoints[1], midY2}, pixelLimit);
            drawSierpinskiTriangle(g, new int[] {midX3, midX2, xPoints[2]}, new int[] {midY3, midY2, yPoints[2]}, pixelLimit);
        }
    }
}
```