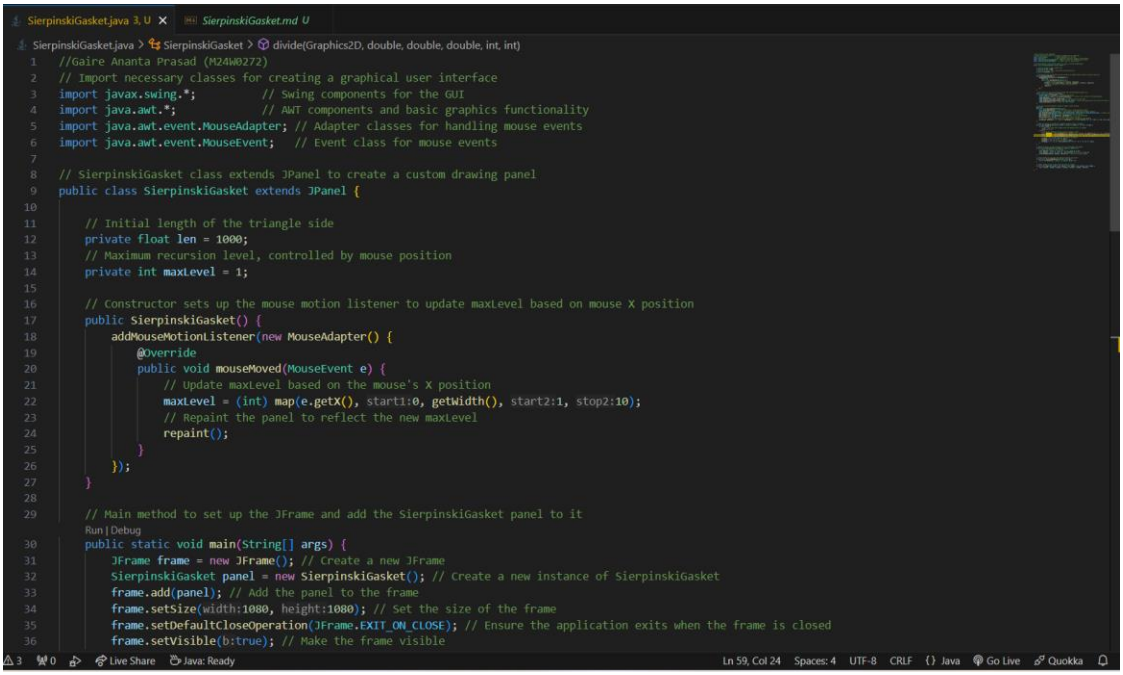
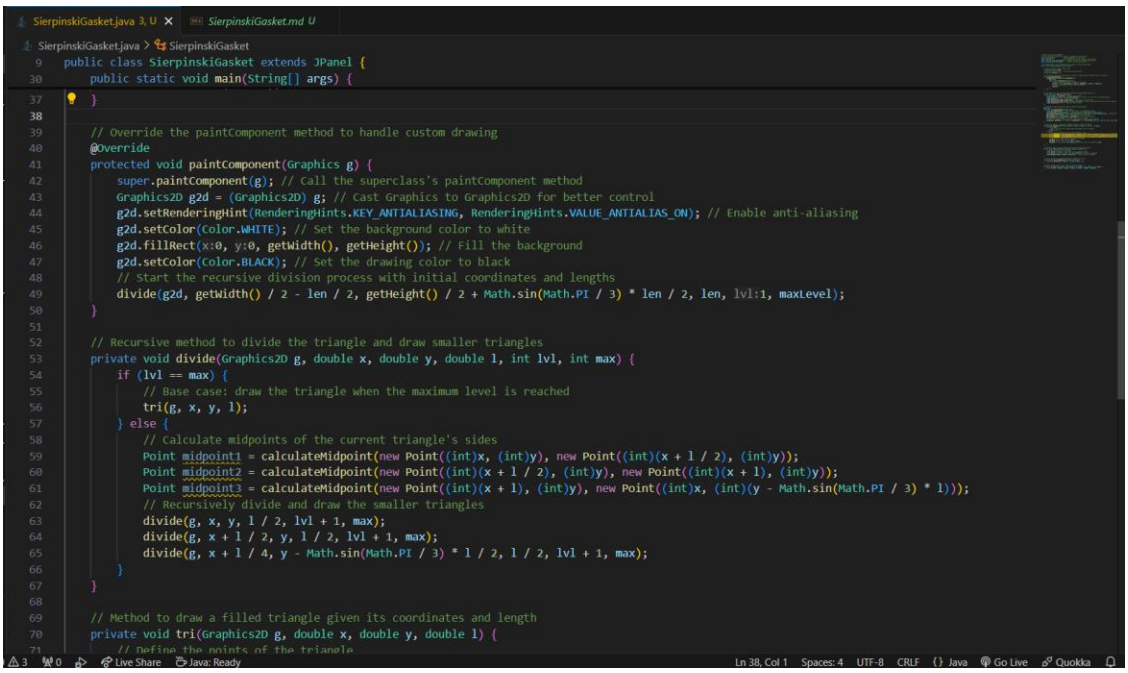
Gaire Ananta Prasad
M24W0272
Sierpinski Gasket Java code



```java
//Gaire Ananta Prasad (M24W0272)
// Import necessary classes for creating a graphical user interface
import javax.swing.*;          // Swing components for the GUI
import java.awt.*;             // AWT components and basic graphics functionality
import java.awt.event.MouseAdapter; // Adapter classes for handling mouse events
import java.awt.event.MouseEvent;   // Event class for mouse events

// SierpinskiGasket class extends JPanel to create a custom drawing panel
public class SierpinskiGasket extends JPanel {

    // Initial length of the triangle side
    private float len = 1000;
    // Maximum recursion level, controlled by mouse position
    private int maxLevel = 1;

    // Constructor sets up the mouse motion listener to update maxLevel based on mouse X position
    public SierpinskiGasket() {
        addMouseMotionListener(new MouseAdapter() {
            @Override
            public void mouseMoved(MouseEvent e) {
                // Update maxLevel based on the mouse's X position
                maxLevel = (int) map(e.getX(), start1:0, getWidth(), start2:1, stop2:10);
                // Repaint the panel to reflect the new maxLevel
                repaint();
            }
        });
    }

    // Main method to set up the JFrame and add the SierpinskiGasket panel to it
    Run | Debug
    public static void main(String[] args) {
        JFrame frame = new JFrame(); // Create a new JFrame
        SierpinskiGasket panel = new SierpinskiGasket(); // Create a new instance of SierpinskiGasket
        frame.add(panel); // Add the panel to the frame
        frame.setSize(width:1080, height:1080); // Set the size of the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Ensure the application exits when the frame is closed
        frame.setVisible(b:true); // Make the frame visible
```



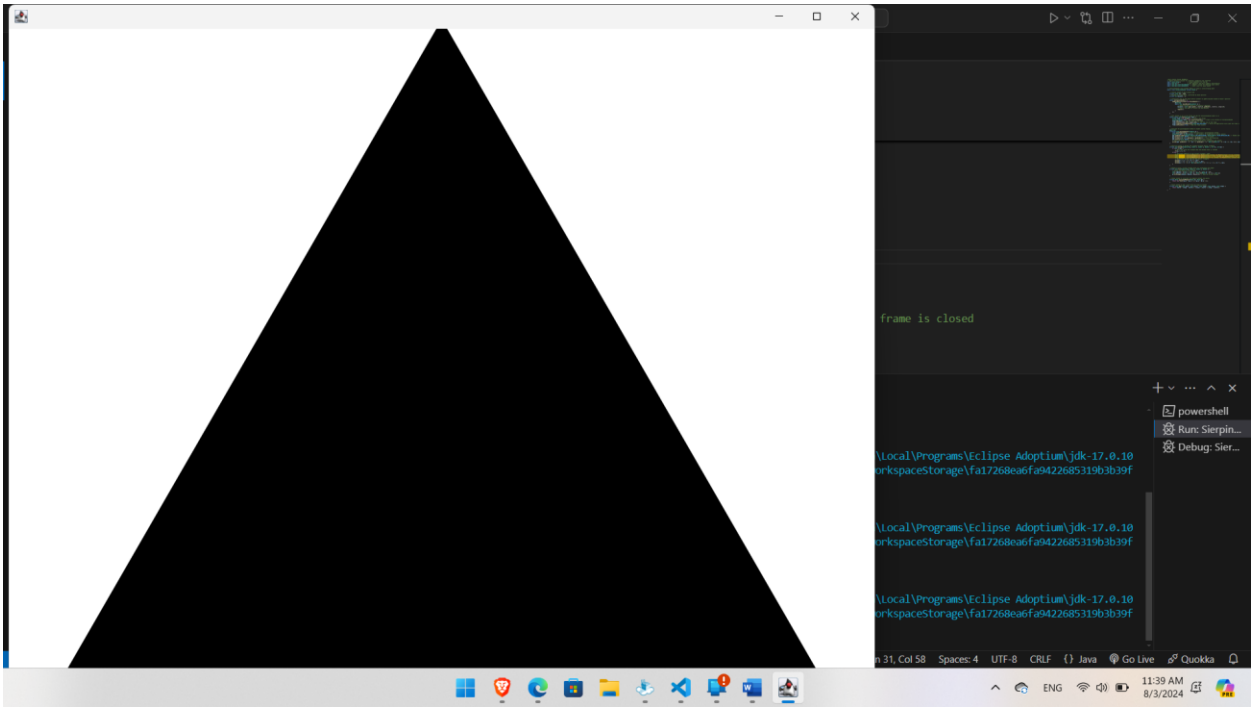
```java
public class SierpinskiGasket extends JPanel {
    public static void main(String[] args) {
    }

    // Override the paintComponent method to handle custom drawing
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g); // Call the superclass's paintComponent method
        Graphics2D g2d = (Graphics2D) g; // Cast Graphics to Graphics2D for better control
        g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON); // Enable anti-aliasing
        g2d.setColor(Color.WHITE); // Set the background color to white
        g2d.fillRect(x:0, y:0, getWidth(), getHeight()); // Fill the background
        g2d.setColor(Color.BLACK); // Set the drawing color to black
        // Start the recursive division process with initial coordinates and lengths
        divide(g2d, getWidth() / 2 - len / 2, getHeight() / 2 + Math.sin(Math.PI / 3) * len / 2, len, lvl:1, maxLevel);
    }

    // Recursive method to divide the triangle and draw smaller triangles
    private void divide(Graphics2D g, double x, double y, double l, int lvl, int max) {
        if (lvl == max) {
            // Base case: draw the triangle when the maximum level is reached
            tri(g, x, y, l);
        } else {
            // Calculate midpoints of the current triangle's sides
            Point midpoint1 = calculateMidpoint(new Point((int)x, (int)y), new Point((int)(x + l / 2), (int)y));
            Point midpoint2 = calculateMidpoint(new Point((int)(x + l / 2), (int)y), new Point((int)(x + l), (int)y));
            Point midpoint3 = calculateMidpoint(new Point((int)(x + l), (int)y), new Point((int)x, (int)(y - Math.sin(Math.PI / 3) * l)));
            // Recursively divide and draw the smaller triangles
            divide(g, x, y, l / 2, lvl + 1, max);
            divide(g, x + l / 2, y, l / 2, lvl + 1, max);
            divide(g, x + l / 4, y - Math.sin(Math.PI / 3) * l / 2, l / 2, lvl + 1, max);
        }
    }

    // Method to draw a filled triangle given its coordinates and length
    private void tri(Graphics2D g, double x, double y, double l) {
        // Define the points of the triangle
```

SierpinskiGasket.java 3, U SierpinskiGasket.md U

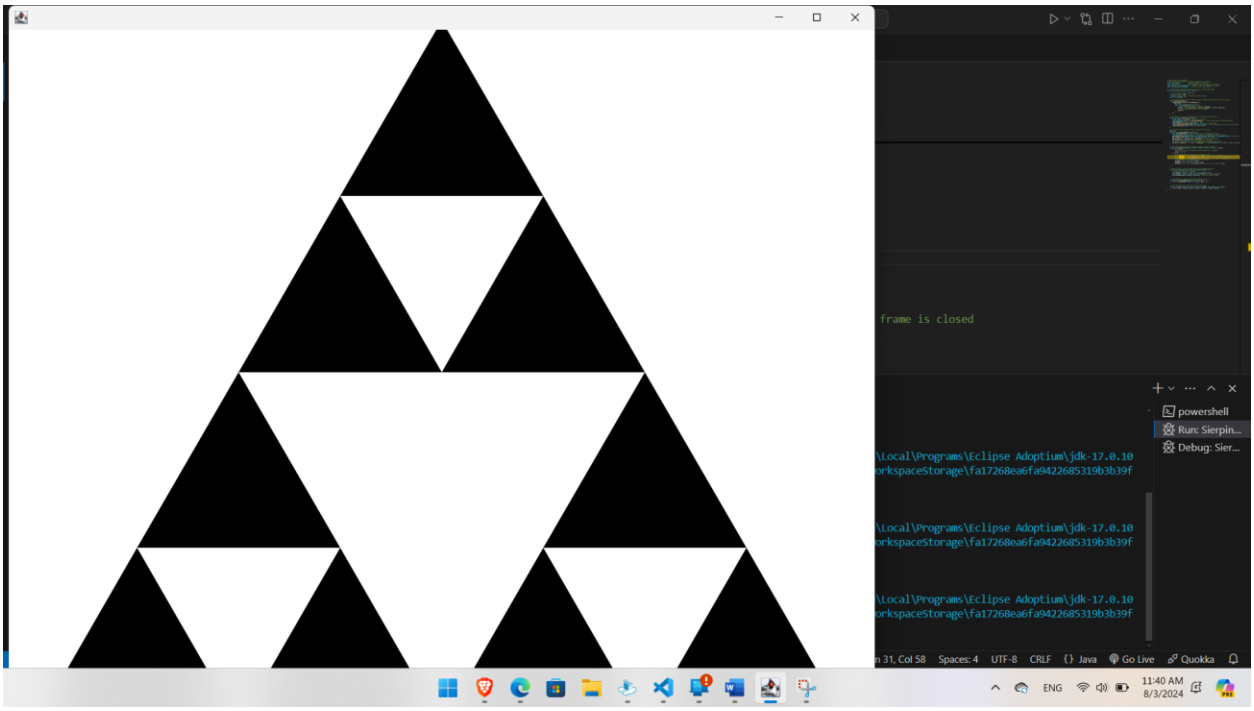
SierpinskiGasket.java > SierpinskiGasket

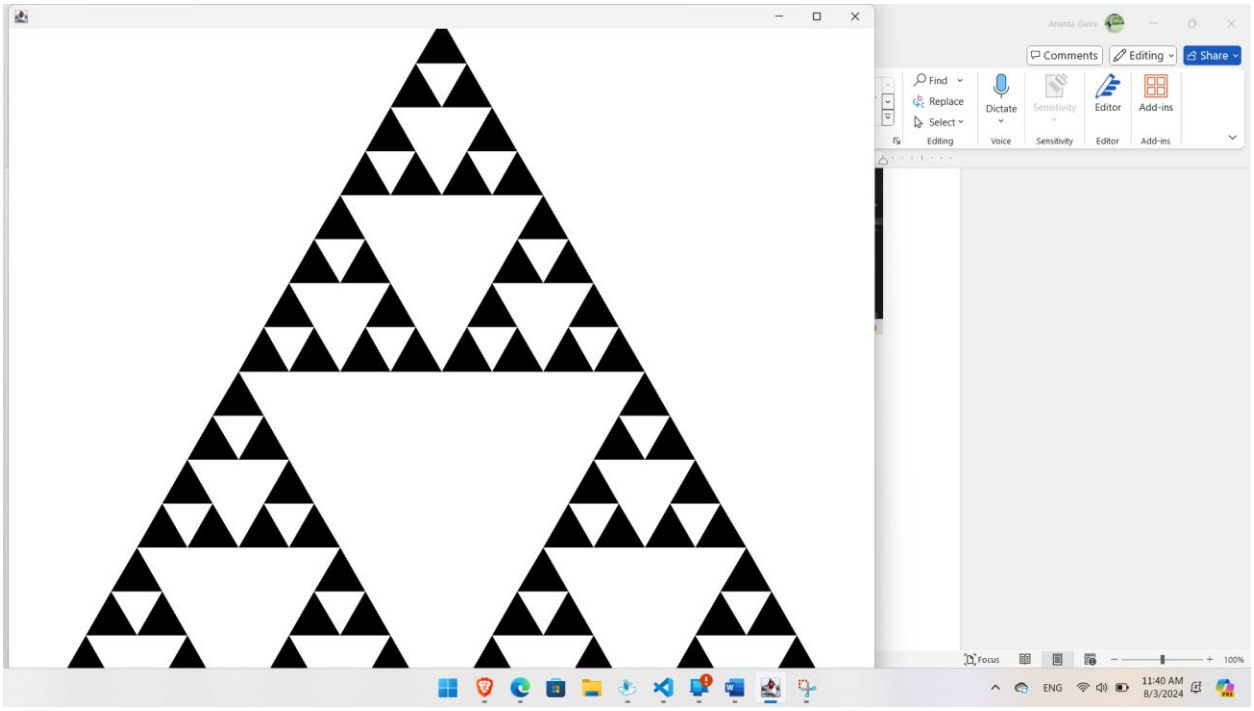
```java
public class SierpinskiGasket extends JPanel {

    // Method to draw a filled triangle given its coordinates and length
    private void tri(Graphics2D g, double x, double y, double l) {
        // Define the points of the triangle
        int[] xPoints = {(int) x, (int) (x + l / 2), (int) (x + l)};
        int[] yPoints = {(int) y, (int) (y - Math.sin(Math.PI / 3) * l), (int) y};
        g.fillPolygon(xPoints, yPoints, nPoints:3); // Draw the filled triangle
    }

    // Helper method to calculate the midpoint between two points
    private Point calculateMidpoint(Point p1, Point p2) {
        return new Point((p1.x + p2.x) / 2, (p1.y + p2.y) / 2);
    }

    // Helper method to map a value from one range to another
    private float map(float value, float start1, float stop1, float start2, float stop2) {
        return start2 + (stop2 - start2) * ((value - start1) / (stop1 - start1));
    }
}
```

Output

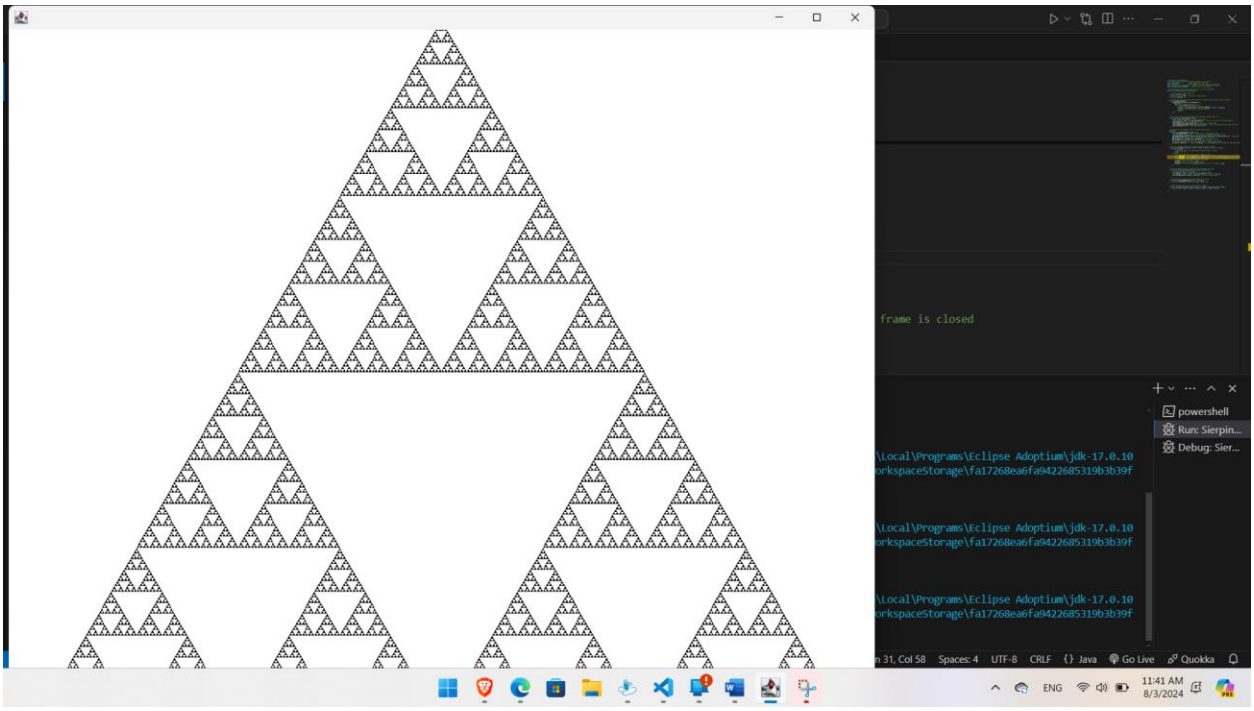


Gaire Ananta Prasad(M24W0272)



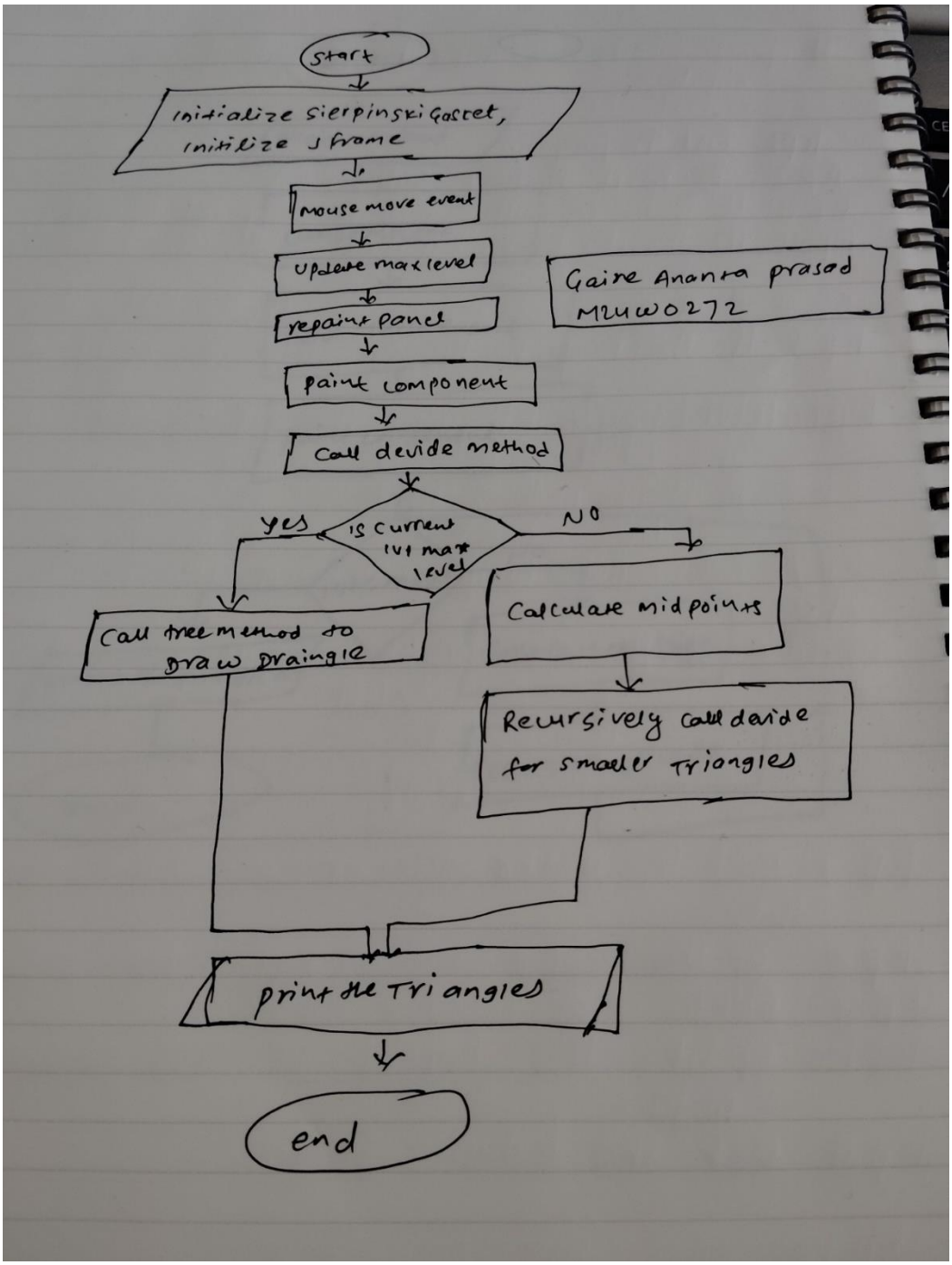


Gaire Ananta Prasad(M24W0272)

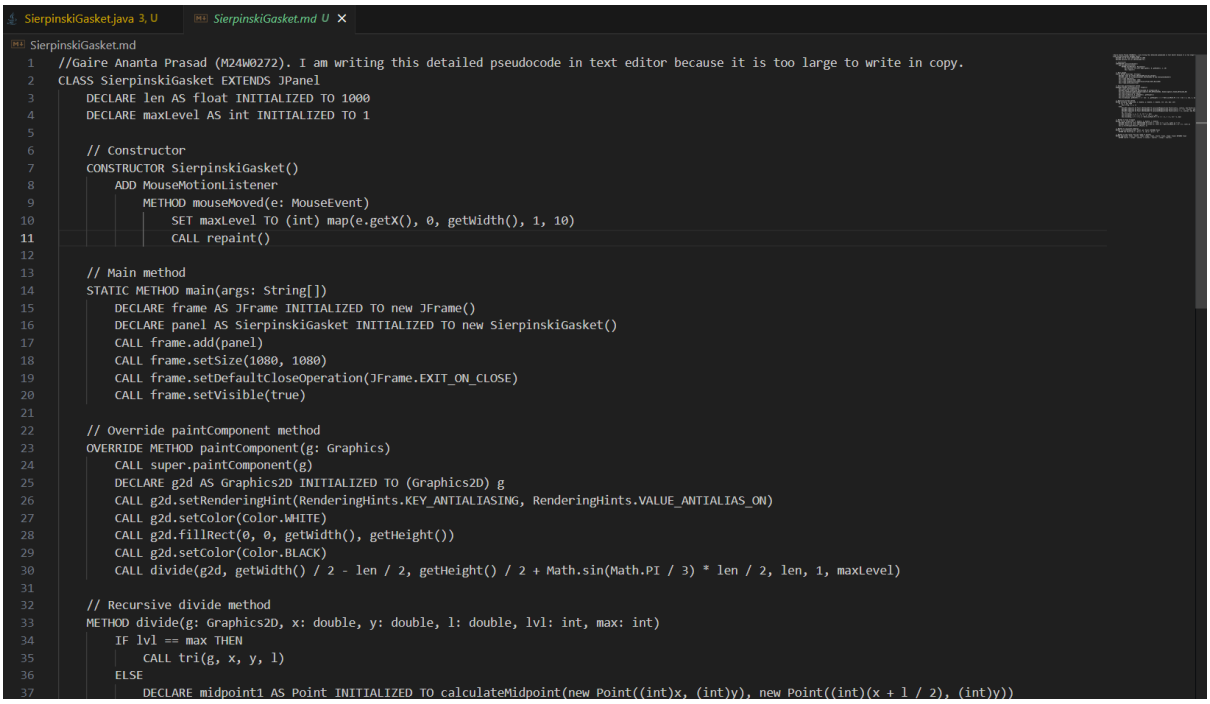


Gaire Ananta Prasad(M24W0272)

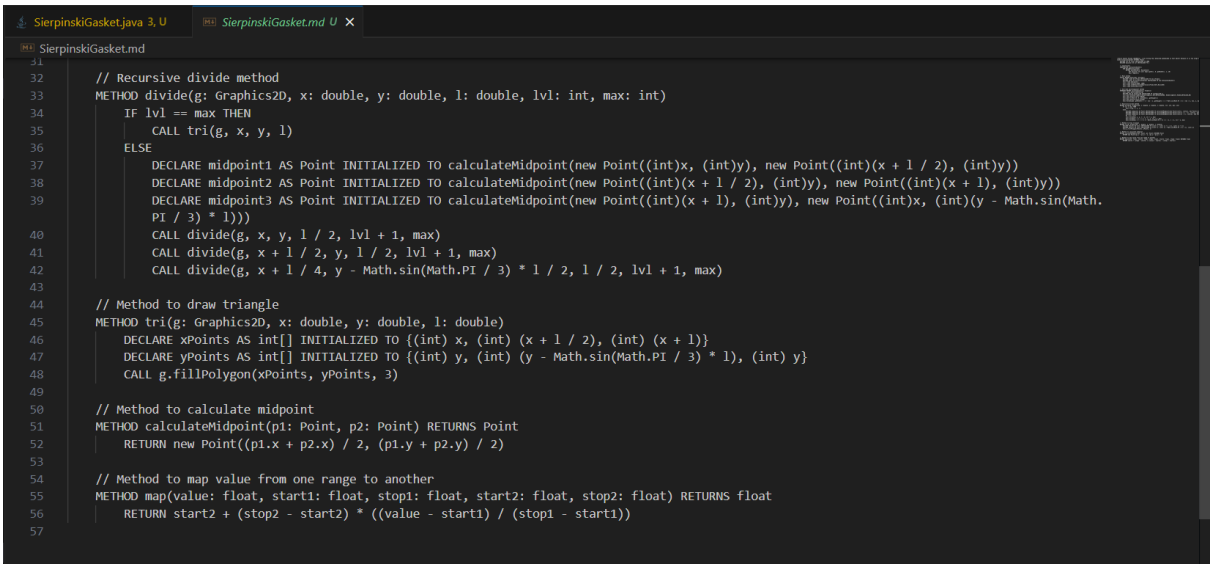
Flow-Chart


start
initialize SierpinskiGasket, initilize JFrame
mouse move event
Update max level
repaint panel
paint component
Call devide method
Is current lvl max level
yes
NO
Call tree method to draw Draingle
Calculate midpoints
Recursively call devide for smaller Triangles
print the Triangles
end
Gaire Ananta prasad M24W0272

Pseudocode



```
//Gaire Ananta Prasad (M24W0272). I am writing this detailed pseudocode in text editor because it is too large to write in copy.
CLASS SierpinskiGasket EXTENDS JPanel
    DECLARE len AS float INITIALIZED TO 1000
    DECLARE maxLevel AS int INITIALIZED TO 1

    // Constructor
    CONSTRUCTOR SierpinskiGasket()
        ADD MouseMotionListener
            METHOD mouseMoved(e: MouseEvent)
                SET maxLevel TO (int) map(e.getX(), 0, getWidth(), 1, 10)
                CALL repaint()

    // Main method
    STATIC METHOD main(args: String[])
        DECLARE frame AS JFrame INITIALIZED TO new JFrame()
        DECLARE panel AS SierpinskiGasket INITIALIZED TO new SierpinskiGasket()
        CALL frame.add(panel)
        CALL frame.setSize(1080, 1080)
        CALL frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
        CALL frame.setVisible(true)

    // Override paintComponent method
    OVERRIDE METHOD paintComponent(g: Graphics)
        CALL super.paintComponent(g)
        DECLARE g2d AS Graphics2D INITIALIZED TO (Graphics2D) g
        CALL g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON)
        CALL g2d.setColor(Color.WHITE)
        CALL g2d.fillRect(0, 0, getWidth(), getHeight())
        CALL g2d.setColor(Color.BLACK)
        CALL divide(g2d, getWidth() / 2 - len / 2, getHeight() / 2 + Math.sin(Math.PI / 3) * len / 2, len, 1, maxLevel)

    // Recursive divide method
    METHOD divide(g: Graphics2D, x: double, y: double, l: double, lvl: int, max: int)
        IF lvl == max THEN
            CALL tri(g, x, y, l)
        ELSE
            DECLARE midpoint1 AS Point INITIALIZED TO calculateMidpoint(new Point((int)x, (int)y), new Point((int)(x + l / 2), (int)y))
```



```
    // Recursive divide method
    METHOD divide(g: Graphics2D, x: double, y: double, l: double, lvl: int, max: int)
        IF lvl == max THEN
            CALL tri(g, x, y, l)
        ELSE
            DECLARE midpoint1 AS Point INITIALIZED TO calculateMidpoint(new Point((int)x, (int)y), new Point((int)(x + l / 2), (int)y))
            DECLARE midpoint2 AS Point INITIALIZED TO calculateMidpoint(new Point((int)(x + l / 2), (int)y), new Point((int)(x + l), (int)y))
            DECLARE midpoint3 AS Point INITIALIZED TO calculateMidpoint(new Point((int)(x + l), (int)y), new Point((int)x, (int)(y - Math.sin(Math.
            PI / 3) * l)))
            CALL divide(g, x, y, l / 2, lvl + 1, max)
            CALL divide(g, x + l / 2, y, l / 2, lvl + 1, max)
            CALL divide(g, x + l / 4, y - Math.sin(Math.PI / 3) * l / 2, l / 2, lvl + 1, max)

    // Method to draw triangle
    METHOD tri(g: Graphics2D, x: double, y: double, l: double)
        DECLARE xPoints AS int[] INITIALIZED TO {(int) x, (int) (x + l / 2), (int) (x + l)}
        DECLARE yPoints AS int[] INITIALIZED TO {(int) y, (int) (y - Math.sin(Math.PI / 3) * l), (int) y}
        CALL g.fillPolygon(xPoints, yPoints, 3)

    // Method to calculate midpoint
    METHOD calculateMidpoint(p1: Point, p2: Point) RETURNS Point
        RETURN new Point((p1.x + p2.x) / 2, (p1.y + p2.y) / 2)

    // Method to map value from one range to another
    METHOD map(value: float, start1: float, stop1: float, start2: float, stop2: float) RETURNS float
        RETURN start2 + (stop2 - start2) * ((value - start1) / (stop1 - start1))
```

Gaire Ananta Prasad(M24W0272)