```java
// Import necessary libraries for GUI components and graphics
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;

/**
 * Main class that creates a 3D Shape Calculator application
 * Extends JFrame to create a window-based GUI application
 */
public class Shape3D extends JFrame {
    // Private field to hold the drawing panel where shapes will be displayed
    private DrawPanel drawPanel;

    /**
     * Constructor - sets up the main window and all GUI components
     */
    public Shape3D() {
        // Set up the main window properties
        setTitle("3D Shape Viewer");                        // Window title
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);     // Close program when
window is closed
        setSize(800, 600);                                  // Set window
dimensions
        setLocationRelativeTo(null);                        // Center window on
screen

        // Create and add the drawing panel where shapes will be displayed
        drawPanel = new DrawPanel();
        add(drawPanel, BorderLayout.CENTER);                // Add to center of
window

        // Create a panel to hold all the buttons
        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new FlowLayout());            // Arrange buttons in a
row

        // Create buttons for each shape type
        JButton sphereBtn = new JButton("Sphere");
        JButton cubeBtn = new JButton("Cube");
        JButton cylinderBtn = new JButton("Cylinder");
        JButton coneBtn = new JButton("Cone");
        JButton prismBtn = new JButton("Rectangular Prism");
        JButton trianglesBtn = new JButton("Show Triangles");

        // Add action listeners to each button (what happens when clicked)
        sphereBtn.addActionListener(e -> sphere(drawPanel));
        cubeBtn.addActionListener(e -> cube(drawPanel));
        cylinderBtn.addActionListener(e -> cylinder(drawPanel));
        coneBtn.addActionListener(e -> cone(drawPanel));
        prismBtn.addActionListener(e -> rectangularPrism(drawPanel));
        trianglesBtn.addActionListener(e -> showTriangles(drawPanel));

        // Add all buttons to the button panel
        buttonPanel.add(sphereBtn);
        buttonPanel.add(cubeBtn);
        buttonPanel.add(cylinderBtn);
        buttonPanel.add(coneBtn);
        buttonPanel.add(prismBtn);
        buttonPanel.add(trianglesBtn);

        // Add button panel to the bottom of the main window
        add(buttonPanel, BorderLayout.SOUTH);
    }

    /**
     * Method to display a 3D sphere
     * Simply shows the sphere shape without any calculations
     */
    public static void sphere(DrawPanel panel) {
        // Tell the drawing panel to display a sphere with default size
        panel.setShape("Sphere (3D)", 5.0);
    }

    /**
     * Method to display a 3D cube
     * Simply shows the cube shape without any calculations
     */
    public static void cube(DrawPanel panel) {
        // Tell the drawing panel to display a cube with default size
        panel.setShape("Cube (3D)", 8.0);
    }

    /**
     * Method to display a 3D cylinder
     * Simply shows the cylinder shape without any calculations
     */
    public static void cylinder(DrawPanel panel) {
        // Tell the drawing panel to display a cylinder with default dimensions
        panel.setShape("Cylinder (3D)", 4.0, 10.0);
    }

    /**
     * Method to display a 3D cone
     * Simply shows the cone shape without any calculations
     */
    public static void cone(DrawPanel panel) {
        // Tell the drawing panel to display a cone with default dimensions
        panel.setShape("Cone (3D)", 4.0, 8.0);
    }

    /**
     * Method to display a 3D rectangular prism
     * Simply shows the rectangular prism shape without any calculations
     */
    public static void rectangularPrism(DrawPanel panel) {
        // Tell the drawing panel to display a rectangular prism with default
dimensions
        panel.setShape("Rectangular Prism (3D)", 10.0, 6.0, 4.0);
    }

    /**
     * Method to display three colored triangles
     * No calculations needed - just tells the panel to show triangles
     */
    public static void showTriangles(DrawPanel panel) {
        // Tell the drawing panel to display the three triangles
        panel.setShape("Three Triangles");
    }

    /**
     * Main method - entry point of the program
     * Creates and displays the main window
     */
    public static void main(String[] args) {
        // Use SwingUtilities to ensure GUI is created on the correct thread
        SwingUtilities.invokeLater(() -> {
            // Create a new Shape3D window and make it visible
            new Shape3D().setVisible(true);
        });
    }

    /**
     * Inner class that handles all the drawing/graphics
     * Extends JPanel to create a custom drawing area
     */
    static class DrawPanel extends JPanel {
        // Store what shape is currently being displayed
        private String currentShape = "Three Triangles";
        // Store the dimensions of the current shape (radius, height, etc.)
        private double[] dimensions = new double[3];
        // Store the position where shapes should be drawn (for movement)
        private int shapeX, shapeY;
        // Track if mouse is being dragged
        private boolean isDragging = false;
        private int lastMouseX, lastMouseY;

        /**
         * Constructor for DrawPanel - sets up mouse listeners for movement
         */
        public DrawPanel() {
            // Initialize shape position to center
            shapeX = 400;
            shapeY = 300;

            // Add mouse listener for clicking and dragging
            addMouseListener(new MouseAdapter() {
                @Override
                public void mousePressed(MouseEvent e) {
                    // Start dragging when mouse is pressed
                    isDragging = true;
                    lastMouseX = e.getX();
                    lastMouseY = e.getY();
                }

                @Override
                public void mouseReleased(MouseEvent e) {
                    // Stop dragging when mouse is released
                    isDragging = false;
                }
            });

            // Add mouse motion listener for dragging
            addMouseMotionListener(new MouseMotionAdapter() {
                @Override
                public void mouseDragged(MouseEvent e) {
                    if (isDragging) {
                        // Calculate how much the mouse moved
                        int deltaX = e.getX() - lastMouseX;
                        int deltaY = e.getY() - lastMouseY;

                        // Update shape position
                        shapeX += deltaX;
                        shapeY += deltaY;

                        // Update last mouse position
                        lastMouseX = e.getX();
                        lastMouseY = e.getY();

                        // Redraw the panel
                        repaint();
                    }
                }
            });
        }

        /**
         * Method to set what shape should be displayed and its dimensions
         * Called by the button methods above
         */
        public void setShape(String shape, double... dims) {
            this.currentShape = shape;        // Remember which shape to draw
            this.dimensions = dims;           // Remember the shape's dimensions
            repaint();                        // Trigger a redraw of the panel
        }

        /**
         * This method is automatically called whenever the panel needs to be
redrawn
         * It's where all the actual drawing happens
         */
        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);                    // Clear the panel first
            Graphics2D g2d = (Graphics2D) g;            // Convert to Graphics2D
for better drawing features
            // Enable antialiasing for smoother shapes
            g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

            // Draw the title text and instructions at the top
            g2d.setColor(Color.BLACK);
            g2d.drawString("Current Shape: " + currentShape, 10, 20);
            g2d.drawString("Click and drag to move shapes around!", 10, 40);

            // Decide what to draw based on the current shape
            if (currentShape.equals("Three Triangles")) {
                drawThreeTriangles(g2d);                // Draw the three triangles
            } else {
                drawShapeRepresentation(g2d);           // Draw 3D shape
representations
            }
        }

        /**
         * Method to draw three colored triangles arranged in a pattern
         * Two triangles on top (left and right), one at bottom center
         */
        private void drawThreeTriangles(Graphics2D g2d) {
            // Use the moveable position for triangles
            int centerX = shapeX;
            int centerY = shapeY;
            int triangleSize = 80;                      // Size of each triangle

            // Triangle 1 - Red triangle (positioned top left)
            g2d.setColor(Color.RED);
            // Define the three corner points of the triangle
            int[] x1 = {centerX - triangleSize, centerX - triangleSize/2, centerX};
            int[] y1 = {centerY - triangleSize/2, centerY - triangleSize, centerY -
triangleSize/2};
            g2d.fillPolygon(x1, y1, 3);                // Draw filled triangle with
3 points

            // Triangle 2 - Blue triangle (positioned top right)
            g2d.setColor(Color.BLUE);
            // Define the three corner points of the triangle
            int[] x2 = {centerX, centerX + triangleSize/2, centerX + triangleSize};
            int[] y2 = {centerY - triangleSize/2, centerY - triangleSize, centerY -
triangleSize/2};
            g2d.fillPolygon(x2, y2, 3);                // Draw filled triangle

            // Triangle 3 - Green triangle (positioned bottom center)
            g2d.setColor(Color.GREEN);
            // Define the three corner points of the triangle
            int[] x3 = {centerX - triangleSize/2, centerX, centerX +
triangleSize/2};
            int[] y3 = {centerY + triangleSize/2, centerY + triangleSize, centerY +
triangleSize/2};
            g2d.fillPolygon(x3, y3, 3);                // Draw filled triangle
        }

        /**
         * Method to draw visual representations of 3D shapes
         * Each shape is drawn differently based on the currentShape string
         */
        private void drawShapeRepresentation(Graphics2D g2d) {
            // Use the moveable position instead of fixed center
            int centerX = shapeX;
            int centerY = shapeY;

            // Check which type of shape to draw and draw it accordingly
            if (currentShape.contains("Sphere")) {
                // Draw sphere as a filled circle with gradient effect
                // Scale the radius for display and limit size to reasonable bounds
                int radius = Math.max(20, Math.min(100, (int) (dimensions[0] *
10)));
                // Create gradient from light blue to dark blue for 3D effect
                GradientPaint gradient = new GradientPaint(
                    centerX - radius/2, centerY - radius/2, Color.CYAN,
                    centerX + radius/2, centerY + radius/2, Color.BLUE
                );
                g2d.setPaint(gradient);
                // Draw circle centered at moveable position
                g2d.fillOval(centerX - radius, centerY - radius, radius * 2, radius
* 2);
            } else if (currentShape.contains("Cube")) {
                // Draw cube as a square with 3D effect and different colored sides
                // Scale the side length and limit to reasonable size
                int side = Math.max(20, Math.min(150, (int) (dimensions[0] * 10)));
                int offset = side / 4;                  // How much to offset for
3D effect

                // Draw the main square face of the cube (front face) - RED
                g2d.setColor(Color.RED);
                g2d.fillRect(centerX - side/2, centerY - side/2, side, side);

                // Draw the top face - ORANGE
                g2d.setColor(Color.ORANGE);
                int[] topX = {centerX - side/2, centerX - side/2 + offset, centerX
+ side/2 + offset, centerX + side/2};
                int[] topY = {centerY - side/2, centerY - side/2 - offset, centerY
- side/2 - offset, centerY - side/2};
                g2d.fillPolygon(topX, topY, 4);

                // Draw the right face - YELLOW
                g2d.setColor(Color.YELLOW);
                int[] rightX = {centerX + side/2, centerX + side/2 + offset,
centerX + side/2 + offset, centerX + side/2};
                int[] rightY = {centerY - side/2, centerY - side/2 - offset,
centerY + side/2 - offset, centerY + side/2};
                g2d.fillPolygon(rightX, rightY, 4);

                // Draw outlines for definition
                g2d.setColor(Color.BLACK);
                g2d.drawRect(centerX - side/2, centerY - side/2, side, side);
                g2d.drawPolygon(topX, topY, 4);
                g2d.drawPolygon(rightX, rightY, 4);
            } else if (currentShape.contains("Cylinder")) {
                // Draw cylinder with different colored parts
                // Scale dimensions and apply size limits
                int radius = Math.max(15, Math.min(80, (int) (dimensions[0] *
10)));
                int height = Math.max(30, Math.min(200, (int) (dimensions[1] *
10)));

                // Draw the main body (rectangle) of cylinder - BLUE
                g2d.setColor(Color.BLUE);
                g2d.fillRect(centerX - radius, centerY - height/2, radius * 2,
height);

                // Draw top oval (ellipse) of cylinder - LIGHT BLUE
                g2d.setColor(Color.CYAN);
                g2d.fillOval(centerX - radius, centerY - height/2 - 10, radius * 2,
20);

                // Draw bottom oval of cylinder - DARK BLUE
                g2d.setColor(Color.BLUE.darker());
                g2d.fillOval(centerX - radius, centerY + height/2 - 10, radius * 2,
20);
```

```
                // Add outlines
                g2d.setColor(Color.BLACK);
                g2d.drawRect(centerX - radius, centerY - height/2, radius * 2,
height);
                g2d.drawOval(centerX - radius, centerY - height/2 - 10, radius * 2,
20);
                g2d.drawOval(centerX - radius, centerY + height/2 - 10, radius * 2,
20);
            } else if (currentShape.contains("Cone")) {
                // Draw cone with different colored parts
                // Scale dimensions and apply size limits
                int radius = Math.max(15, Math.min(80, (int) (dimensions[0] *
10)));
                int height = Math.max(30, Math.min(150, (int) (dimensions[1] *
10)));

                // Draw the triangular side view of the cone - GREEN
                g2d.setColor(Color.GREEN);
                int[] x = {centerX - radius, centerX, centerX + radius};         //
x-coordinates of triangle points
                int[] y = {centerY + height/2, centerY - height/2, centerY +
height/2}; // y-coordinates of triangle points
                g2d.fillPolygon(x, y, 3);                                       //
Draw filled triangle

                // Draw the circular base of the cone - DARK GREEN
                g2d.setColor(Color.GREEN.darker());
                g2d.fillOval(centerX - radius, centerY + height/2 - 5, radius * 2,
10);

                // Add outlines
                g2d.setColor(Color.BLACK);
                g2d.drawPolygon(x, y, 3);
                g2d.drawOval(centerX - radius, centerY + height/2 - 5, radius * 2,
10);
            } else if (currentShape.contains("Rectangular")) {
                // Draw rectangular prism with different colored sides
                // Scale all three dimensions and apply size limits
                int length = Math.max(20, Math.min(120, (int) (dimensions[0] *
5)));
                int width = Math.max(20, Math.min(120, (int) (dimensions[1] * 5)));
                int height = Math.max(10, Math.min(50, (int) (dimensions[2] * 5)));
                int offset = Math.max(5, height / 4);                           //
3D offset based on height

                // Draw the main rectangular face of the prism (front face) -
MAGENTA
                g2d.setColor(Color.MAGENTA);
                g2d.fillRect(centerX - length/2, centerY - width/2, length, width);

                // Draw the top face - PINK
                g2d.setColor(Color.PINK);
                int[] topX = {centerX - length/2, centerX - length/2 + offset,
centerX + length/2 + offset, centerX + length/2};
                int[] topY = {centerY - width/2, centerY - width/2 - offset,
centerY - width/2 - offset, centerY - width/2};
                g2d.fillPolygon(topX, topY, 4);

                // Draw the right face - LIGHT GRAY
                g2d.setColor(Color.LIGHT_GRAY);
                int[] rightX = {centerX + length/2, centerX + length/2 + offset,
centerX + length/2 + offset, centerX + length/2};
                int[] rightY = {centerY - width/2, centerY - width/2 - offset,
centerY + width/2 - offset, centerY + width/2};
                g2d.fillPolygon(rightX, rightY, 4);

                // Add outlines for definition
                g2d.setColor(Color.BLACK);
                g2d.drawRect(centerX - length/2, centerY - width/2, length, width);
                g2d.drawPolygon(topX, topY, 4);
                g2d.drawPolygon(rightX, rightY, 4);
            }
        }
    }
}
```