```java
// Import statements - bringing in necessary Java libraries
import javax.swing.*;           // GUI components like JFrame, JButton, JPanel,
JOptionPane
import java.awt.*;              // Drawing capabilities, layout managers, and basic
graphics classes
import java.awt.event.ActionEvent;     // Event handling for user interactions
import java.awt.event.ActionListener;  // Interface for handling button clicks

/**
 * Main application class that creates a 2D Shape Calculator & Drawer
 * Extends JFrame to create a window application
 */
public class Shape2D extends JFrame {
    // Private field to hold the custom drawing area
    private DrawPanel drawPanel;

    /**
     * Constructor - Sets up the main window and user interface
     */
    public Shape2D() {
        // Configure the main window properties
        setTitle("2D Shape Calculator & Drawer");   // Set window title
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Close program when
window is closed
        setLayout(new BorderLayout());              // Use BorderLayout for
component positioning

        // Create and configure the drawing panel (custom canvas for shapes)
        drawPanel = new DrawPanel();
        drawPanel.setPreferredSize(new Dimension(600, 400)); // Set canvas size to
600x400 pixels
        drawPanel.setBackground(Color.WHITE);       // White background for drawing
area

        // Create panel to hold buttons with horizontal flow layout
        JPanel buttonPanel = new JPanel(new FlowLayout());

        // Create buttons for each shape type and clear function
        JButton circleBtn = new JButton("Circle");       // Button to draw circles
        JButton rectangleBtn = new JButton("Rectangle"); // Button to draw
rectangles
        JButton triangleBtn = new JButton("Triangle");   // Button to draw
triangles
        JButton clearBtn = new JButton("Clear");         // Button to clear the
drawing

        // Add event listeners to buttons - define what happens when each button is
clicked
        circleBtn.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                drawCircle();  // Call method to handle circle drawing
            }
        });

        rectangleBtn.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                drawRectangle();  // Call method to handle rectangle drawing
            }
        });
        triangleBtn.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                drawTriangle();  // Call method to handle triangle drawing
            }
        });

        clearBtn.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                drawPanel.clearShape();  // Call method to clear the drawing area
            }
        });

        // Add all buttons to the button panel
        buttonPanel.add(circleBtn);     // Add circle button
        buttonPanel.add(rectangleBtn);  // Add rectangle button
        buttonPanel.add(triangleBtn);   // Add triangle button
        buttonPanel.add(clearBtn);      // Add clear button

        // Add components to the main window using BorderLayout
        add(drawPanel, BorderLayout.CENTER);    // Drawing area takes up center
space
        add(buttonPanel, BorderLayout.SOUTH);   // Buttons go at the bottom

        // Final window setup
        pack();                     // Size window to fit all components
        setLocationRelativeTo(null); // Center window on screen
        setVisible(true);           // Make window visible to user
    }

    /**
     * Handles circle creation and area calculation
     * Gets radius from user, validates input, calculates area, and draws the
circle
     */
    private void drawCircle() {
        try {
            // Show dialog box to get radius value from user
            String input = JOptionPane.showInputDialog(this, "Enter radius:");
            if (input == null) return;  // User clicked Cancel

            // Convert string input to number
            double radius = Double.parseDouble(input);

            // Validate that radius is positive
            if (radius <= 0) {
                JOptionPane.showMessageDialog(this, "Radius must be positive!");
                return;
            }

            // Calculate circle area using formula: π × r²
            double area = Math.PI * radius * radius;

            // Display calculated area to user (formatted to 2 decimal places)
            JOptionPane.showMessageDialog(this, "Area = " + String.format("%.2f",
area));

            // Tell drawing panel to draw the circle
            drawPanel.setCircle(radius);
        } catch (NumberFormatException e) {
            // Handle case where user enters non-numeric input
            JOptionPane.showMessageDialog(this, "Invalid number!");
        }
    }

    /**
     * Handles rectangle creation and area calculation
     * Gets width and height from user, validates input, calculates area, and draws
the rectangle
     */
    private void drawRectangle() {
        try {
            // Get width and height through separate dialog boxes
            String w = JOptionPane.showInputDialog(this, "Enter width:");
            if (w == null) return;  // User clicked Cancel on width dialog
            String h = JOptionPane.showInputDialog(this, "Enter height:");
            if (h == null) return;  // User clicked Cancel on height dialog

            // Convert string inputs to numbers
            double width = Double.parseDouble(w);
            double height = Double.parseDouble(h);

            // Validate that both dimensions are positive
            if (width <= 0 || height <= 0) {
                JOptionPane.showMessageDialog(this, "Dimensions must be
positive!");
                return;
            }

            // Calculate rectangle area using formula: width × height
            double area = width * height;

            // Display calculated area to user
            JOptionPane.showMessageDialog(this, "Area = " + String.format("%.2f",
area));

            // Tell drawing panel to draw the rectangle
            drawPanel.setRectangle(width, height);
        } catch (NumberFormatException e) {
            // Handle case where user enters non-numeric input
            JOptionPane.showMessageDialog(this, "Invalid numbers!");
        }
    }

    /**
     * Handles triangle creation and area calculation
     * Gets base and height from user, validates input, calculates area, and draws
the triangle
     */
    private void drawTriangle() {
        try {
            // Get base and height measurements through separate dialog boxes
            String b = JOptionPane.showInputDialog(this, "Enter base:");
            if (b == null) return;  // User clicked Cancel on base dialog
            String h = JOptionPane.showInputDialog(this, "Enter height:");
            if (h == null) return;  // User clicked Cancel on height dialog

            // Convert string inputs to numbers
            double base = Double.parseDouble(b);
            double height = Double.parseDouble(h);

            // Validate that both dimensions are positive
            if (base <= 0 || height <= 0) {
                JOptionPane.showMessageDialog(this, "Dimensions must be
positive!");
                return;
            }

            // Calculate triangle area using formula: ½ × base × height
            double area = 0.5 * base * height;

            // Display calculated area to user
            JOptionPane.showMessageDialog(this, "Area = " + String.format("%.2f",
area));

            // Tell drawing panel to draw the triangle
            drawPanel.setTriangle(base, height);
        } catch (NumberFormatException e) {
            // Handle case where user enters non-numeric input
            JOptionPane.showMessageDialog(this, "Invalid numbers!");
        }
    }

    /**
     * Custom drawing component that extends JPanel
     * Handles the visual representation of shapes on screen
     */
    class DrawPanel extends JPanel {
        // State variables to track what shape to draw and its dimensions
        private String shapeType = "";      // Stores "circle", "rectangle",
"triangle", or empty string
        private double[] dimensions = {};   // Array holding measurements for the
current shape

        /**
         * Configure panel to draw a circle
         * @param radius The radius of the circle to draw
         */
        public void setCircle(double radius) {
            shapeType = "circle";                    // Set shape type identifier
            dimensions = new double[]{radius};      // Store radius in dimensions
array
            repaint();                              // Trigger redraw of the panel
        }

        /**
         * Configure panel to draw a rectangle
         * @param width  The width of the rectangle
         * @param height The height of the rectangle
         */
        public void setRectangle(double width, double height) {
            shapeType = "rectangle";                     // Set shape type
identifier
            dimensions = new double[]{width, height};   // Store width and height
            repaint();                                  // Trigger redraw of the
panel
        }
        /**
         * Configure panel to draw a triangle
         * @param base   The base length of the triangle
         * @param height The height of the triangle
         */
        public void setTriangle(double base, double height) {
            shapeType = "triangle";                    // Set shape type identifier
            dimensions = new double[]{base, height};  // Store base and height
            repaint();                               // Trigger redraw of the
panel
        }

        /**
         * Clear the current shape from display
         */
        public void clearShape() {
            shapeType = "";  // Reset shape type to empty (no shape)
            repaint();       // Trigger redraw to clear the panel
        }

        /**
         * Main drawing method called automatically by Swing when panel needs to be
redrawn
         * This is where all the visual rendering happens
         */
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);  // Call parent method to clear background

            // Cast to Graphics2D for advanced drawing capabilities
            Graphics2D g2d = (Graphics2D) g;
            // Enable anti-aliasing for smooth shape edges
            g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

            // Calculate center point of the panel for positioning shapes
            int centerX = getWidth() / 2;   // Horizontal center
            int centerY = getHeight() / 2;  // Vertical center

            // Draw reference grid lines (crosshairs) in light gray
            g2d.setColor(Color.LIGHT_GRAY);
            g2d.drawLine(centerX, 0, centerX, getHeight());      // Vertical line
through center
            g2d.drawLine(0, centerY, getWidth(), centerY);      // Horizontal line
through center

            // Set up drawing properties for shapes
            g2d.setColor(Color.BLUE);           // Blue color for shape outlines
            g2d.setStroke(new BasicStroke(2));  // 2-pixel thick lines

            // Draw the appropriate shape based on current shapeType
            if (shapeType.equals("circle") && dimensions.length > 0) {
                // Draw circle: scale radius by 20 for visible size on screen
                int radius = (int) (dimensions[0] * 20);
                // drawOval needs top-left corner coordinates and width/height
(diameter)
                g2d.drawOval(centerX - radius, centerY - radius, radius * 2, radius
* 2);
                // Add text label showing the radius value
                g2d.setColor(Color.BLACK);
                g2d.drawString("Circle (r=" + dimensions[0] + ")", centerX - 50,
centerY + radius + 20);

            } else if (shapeType.equals("rectangle") && dimensions.length > 1) {
                // Draw rectangle: scale dimensions by 20 for visible size
                int width = (int) (dimensions[0] * 20);
                int height = (int) (dimensions[1] * 20);
                // drawRect needs top-left corner coordinates and width/height
                g2d.drawRect(centerX - width/2, centerY - height/2, width, height);
                // Add text label showing the dimensions
                g2d.setColor(Color.BLACK);
                g2d.drawString("Rectangle (" + dimensions[0] + "x" + dimensions[1]
+ ")", centerX - 50, centerY + height/2 + 20);

            } else if (shapeType.equals("triangle") && dimensions.length > 1) {
                // Draw triangle: scale dimensions by 20 for visible size
                int base = (int) (dimensions[0] * 20);
                int height = (int) (dimensions[1] * 20);
                // Define triangle vertices: top point and two base points
                int[] xPoints = {centerX, centerX - base/2, centerX + base/2};
// X coordinates
                int[] yPoints = {centerY - height/2, centerY + height/2, centerY +
height/2}; // Y coordinates
                // Draw triangle using the three points
                g2d.drawPolygon(xPoints, yPoints, 3);
                // Add text label showing the dimensions
                g2d.setColor(Color.BLACK);
                g2d.drawString("Triangle (b=" + dimensions[0] + ", h=" +
dimensions[1] + ")", centerX - 50, centerY + height/2 + 20);
            }
        }
    }

    /**
     * Application entry point - starts the program
     * @param args Command line arguments (not used in this application)
     */
    public static void main(String[] args) {
        // Use SwingUtilities.invokeLater for thread safety
        // This ensures the GUI is created on the proper Event Dispatch Thread
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new Shape2D();  // Create and display the main application window
            }
        });
    }
}
```