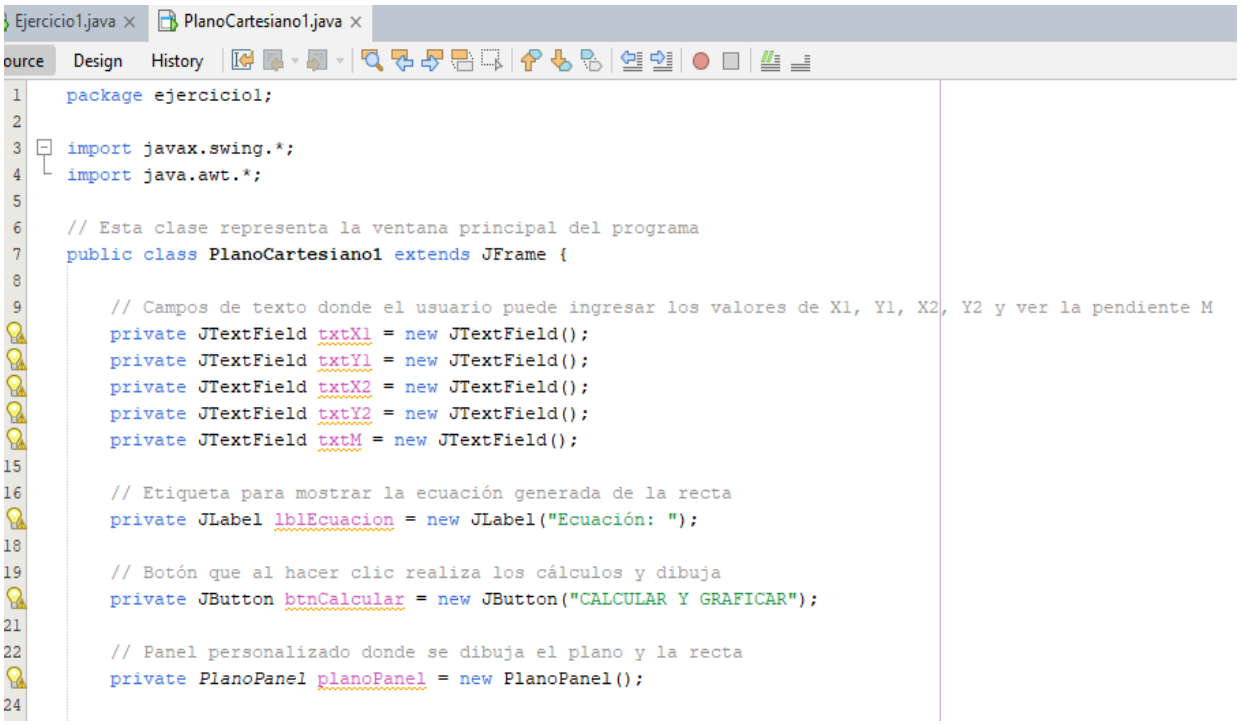


```
// Constructor de la ventana principal
public PlanoCartesiano1() {
    super("Programa #1 – Plano cartesiano");
    setDefaultCloseOperation(EXIT_ON_CLOSE); // Cierra la app al presionar la X
    setSize(950, 550); // Tamaño inicial de la ventana
    setLocationRelativeTo(null); // Centrar en pantalla
    setLayout(new BorderLayout()); // Usamos un layout tipo borde

    add(crearPanelEntrada(), BorderLayout.CENTER); // Parte izquierda: entradas
    add(planoPanel, BorderLayout.EAST); // Parte derecha: dibujo

    btnCalcular.addActionListener(e -> procesar()); // Acción del botón
}
```

```java
// Método que construye y organiza los campos y botones para ingresar los datos
private JPanel crearPanelEntrada() {
    JPanel p = new JPanel(null); // Usamos null layout para posicionar manualmente
    p.setPreferredSize(new Dimension(450, 0));
    p.setBackground(new Color(255, 255, 153)); // Fondo amarillo claro

    Font lblF = new Font("SansSerif", Font.BOLD, 14);
    Font field = new Font("Comic Sans MS", Font.PLAIN, 13);

    JLabel titulo = new JLabel("PROGRAMA #1");
    titulo.setFont(new Font("Comic Sans MS", Font.BOLD | Font.ITALIC, 16));
    titulo.setBounds(10, 10, 200, 25);
    p.add(titulo);

    // Mensajes de instrucción
    JLabel instrul = new JLabel("Ingresa los valores de cada variable");
    JLabel instru2 = new JLabel("para hacer su cálculo");
    instrul.setFont(new Font("Consolas", Font.BOLD, 14));
    instru2.setFont(instrul.getFont());
    instrul.setBounds(10, 45, 350, 20);
    instru2.setBounds(10, 65, 350, 20);
    p.add(instrul);
    p.add(instru2);

    // Añade cada campo (X1, Y1, X2, Y2, M)
    colocarCampo(p, "Variable X1:", 110, txtX1, lblF, field);
    colocarCampo(p, "Variable Y1:", 150, txtY1, lblF, field);
    colocarCampo(p, "Variable X2:", 190, txtX2, lblF, field);
    colocarCampo(p, "Variable Y2:", 230, txtY2, lblF, field);
    colocarCampo(p, "M:", 270, txtM, lblF, field);

    // La pendiente no se puede modificar manualmente
    txtM.setEditable(false);
    txtM.setBackground(Color.WHITE);

    // Botón de calcular y graficar
    btnCalcular.setBackground(Color.GREEN);
    btnCalcular.setFont(new Font("Consolas", Font.BOLD, 14));
    btnCalcular.setBounds(100, 320, 220, 30);
    p.add(btnCalcular);

    // Etiqueta para mostrar la ecuación de la recta
    lblEcuacion.setFont(new Font("Comic Sans MS", Font.PLAIN, 14));
    lblEcuacion.setForeground(new Color(148, 0, 211)); // Color morado
    lblEcuacion.setBounds(10, 370, 420, 25);
    p.add(lblEcuacion);

    return p;
}
```

```java
// Método reutilizable para agregar un campo de texto con su etiqueta
private void colocarCampo(JPanel p, String texto, int y,
                          JTextField campo, Font lblF, Font fCampo) {
    JLabel l = new JLabel(texto);
    l.setFont(lblF);
    l.setBounds(10, y, 110, 25);
    campo.setFont(fCampo);
    campo.setBounds(130, y, 140, 25);
    p.add(l);
    p.add(campo);
}

// Método que procesa los datos ingresados por el usuario y realiza los cálculos
private void procesar() {
    try {
        // Lee y convierte las entradas, permitiendo fracciones
        double x1 = parseFraccion(txtX1.getText());
        double y1 = parseFraccion(txtY1.getText());
        double x2 = parseFraccion(txtX2.getText());
        double y2 = parseFraccion(txtY2.getText());

        if (x1 == x2) {
            // Si x1 == x2, es una recta vertical
            txtM.setText("∞");
            lblEcuacion.setText("Ecuación:  x = " + x1);
            planoPanel.setRectaVertical(x1);
        } else {
            // Calculamos pendiente y ordenada al origen
            double m = (y2 - y1) / (x2 - x1);
            double b = y1 - m * x1;
            txtM.setText(String.format("%.4f", m));
            lblEcuacion.setText(
                    String.format("Ecuación:  y = %.4f x + %.4f", m, b));
            planoPanel.setRecta(m, b);
        }
```

```
            // Establece los puntos a dibujar
            planoPanel.setPuntos(x1, y1, x2, y2);
            planoPanel.repaint(); // Redibuja el panel
        } catch (NumberFormatException ex) {
            // Muestra error si el formato no es correcto
            JOptionPane.showMessageDialog(this,
                    "Ingresa números válidos (decimales o fracciones).",
                    "Error de entrada", JOptionPane.ERROR_MESSAGE);
        }
    }

    // Convierte un número o fracción (como 3/4) en un double
    private double parseFraccion(String s) {
        s = s.trim();
        if (s.contains("/")) {
            String[] partes = s.split("/");
            if (partes.length == 2) {
                double numerador = Double.parseDouble(partes[0]);
                double denominador = Double.parseDouble(partes[1]);
                return numerador / denominador;
            } else {
                throw new NumberFormatException("Fracción mal formateada.");
            }
        }
        return Double.parseDouble(s);
    }

    // Clase interna para el panel que dibuja el plano cartesiano y la línea
    private static class PlanoPanel extends JPanel {
        private Double m = null, b = null, xVertical = null;
        private Double x1 = null, y1 = null, x2 = null, y2 = null;

        // Constructor: define el tamaño del panel
        PlanoPanel() {
            setPreferredSize(new Dimension(480, 0));
        }

        // Configura la pendiente y ordenada (recta normal)
        void setRecta(double m, double b) {
            this.m = m;
            this.b = b;
            this.xVertical = null;
        }

        // Configura una recta vertical
        void setRectaVertical(double x) {
            this.xVertical = x;
            this.m = this.b = null;
        }
```

```java
// Guarda los puntos para dibujar después
void setPuntos(double x1, double y1, double x2, double y2) {
    this.x1 = x1;
    this.y1 = y1;
    this.x2 = x2;
    this.y2 = y2;
}

// Aquí se realiza el dibujo del plano y la recta
@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D) g;
    int w = getWidth(), h = getHeight();
    int cx = w / 2, cy = h / 2; // Centro del plano

    // Dibujar los ejes X y Y
    g2.setColor(Color.BLACK);
    g2.drawLine(0, cy, w, cy); // Eje X
    g2.drawLine(cx, 0, cx, h); // Eje Y
```

```java
// Marcas en los ejes (cada 20 pixeles equivale a una unidad)
for (int x = cx; x < w; x += 20) g2.drawLine(x, cy - 3, x, cy + 3); // eje X positivo
for (int x = cx; x > 0; x -= 20) g2.drawLine(x, cy - 3, x, cy + 3); // eje X negativo
for (int y = cy; y < h; y += 20) g2.drawLine(cx - 3, y, cx + 3, y); // eje Y negativo (porque el 0 está arriba)
for (int y = cy; y > 0; y -= 20) g2.drawLine(cx - 3, y, cx + 3, y); // eje Y positivo

// Dibuja la recta en rojo
g2.setColor(Color.RED);
if (m != null && b != null) {
    // Si tenemos pendiente y ordenada al origen, graficamos y = mx + b
    for (int xPixel = 0; xPixel < w - 1; xPixel++) {
        double xMath = (xPixel - cx) / 20.0; // Convertimos pixel a unidad matemática
        double yMath1 = m * xMath + b;
        double yMath2 = m * (xMath + 1.0 / 20) + b;
        int yPixel1 = cy - (int) Math.round(yMath1 * 20);
        int yPixel2 = cy - (int) Math.round(yMath2 * 20);
        g2.drawLine(xPixel, yPixel1, xPixel + 1, yPixel2);
    }
} else if (xVertical != null) {
    // Si es una línea vertical, se dibuja de arriba a abajo en x = xVertical
    int xPix = cx + (int) Math.round(xVertical * 20);
    g2.drawLine(xPix, 0, xPix, h);
}
```

```java
        // Dibuja los puntos (X1, Y1) y (X2, Y2) como cruces azules
        g2.setColor(Color.BLUE);
        if (x1 != null && y1 != null) {
            int px = cx + (int) Math.round(x1 * 20);
            int py = cy - (int) Math.round(y1 * 20);
            g2.drawLine(px - 5, py - 5, px + 5, py + 5); // Diagonal \
            g2.drawLine(px - 5, py + 5, px + 5, py - 5); // Diagonal /
        }
        if (x2 != null && y2 != null) {
            int px = cx + (int) Math.round(x2 * 20);
            int py = cy - (int) Math.round(y2 * 20);
            g2.drawLine(px - 5, py - 5, px + 5, py + 5);
            g2.drawLine(px - 5, py + 5, px + 5, py - 5);
        }
    }
}
```

```java
package ejerciciol;
import javax.swing.*;

public class Ejerciciol {

    public static void main(String[] args) {
        // Método principal para iniciar la aplicación
        SwingUtilities.invokeLater(() -> new PlanoCartesianol().setVisible(true));
    }
}
```