Ejercicios de Java Swing

Descargue el código AQUI

1. Ventana de Bienvenida

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class VentanaBienvenida extends JFrame {

    public VentanaBienvenida() {
        setTitle("Bienvenidos");          // Configurar el título de la ventana
        setSize(400, 200);                // Configurar el tamaño de la ventana
        setLocationRelativeTo(null);      // Centrar la ventana en la pantalla

        // Configurar la operación de cierre por defecto
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Crear el panel principal con un layout que centre los componentes
        JPanel panel = new JPanel(new GridBagLayout());

        // Crear el botón "Salir"
        JButton btnSalir = new JButton("Salir");

        panel.add(btnSalir);              // Agregar el botón al panel
        add(panel);                       // Agregar el panel a la ventana

        // Agregar evento al botón
        btnSalir.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.exit(0);           // Cerrar la ventana y terminar el programa
            }
        });

        setVisible(true);                // Hacer visible la ventana
    }

    public static void main(String[] args) {
        // Ejecutar la aplicación en el hilo de eventos de Swing
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new VentanaBienvenida();
            }
        });
    }
}
```

2. Ventana de Bienvenida 2

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class VentanaBienvenida2 extends JFrame {

    public VentanaBienvenida2() {
        setTitle("Bienvenidos");            // Configurar el título de la ventana
        setSize(400, 250);                  // Configurar el tamaño de la ventana
        setLocationRelativeTo(null);        // Centrar la ventana en la pantalla

        // Configurar la operación de cierre por defecto
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Crear el panel principal con GridBagLayout
        JPanel panel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10); // Espaciado entre componentes

        // Crear los botones de radio
        JRadioButton rbMasculino = new JRadioButton("Masculino");
        JRadioButton rbFemenino = new JRadioButton("Femenino");
```

```
        // Agrupar los botones de radio para que sean mutuamente excluyentes
        ButtonGroup grupoGenero = new ButtonGroup();
        grupoGenero.add(rbMasculino);
        grupoGenero.add(rbFemenino);

        // Seleccionar uno por defecto (opcional)
        rbMasculino.setSelected(true);

        // Crear el botón "Salir"
        JButton btnSalir = new JButton("Salir");

        // Posicionar los componentes en el grid
        // Fila 0: Botones de radio (masculino y femenino)
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridwidth = 2; // Ocupa 2 columnas
        panel.add(rbMasculino, gbc);

        gbc.gridx = 2;
        gbc.gridy = 0;
        gbc.gridwidth = 2; // Ocupa 2 columnas
        panel.add(rbFemenino, gbc);

        // Fila 1: Botón "Salir" (centrado)
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.gridwidth = 4;                  // Ocupa todas las columnas
        gbc.anchor = GridBagConstraints.CENTER;
        panel.add(btnSalir, gbc);
        add(panel);                         // Agregar el panel a la ventana

        // Agregar evento al botón
        btnSalir.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Mostrar el género seleccionado (opcional)
                String genero = "";
                if (rbMasculino.isSelected()) {
                    genero = "Masculino";
                } else if (rbFemenino.isSelected()) {
                    genero = "Femenino";
                }

                // Mensaje de confirmación
                JOptionPane.showMessageDialog(
                    VentanaBienvenida2.this,
                    "Has seleccionado: " + genero + "\nąGracias por usar la aplicación!",
                    "Información",
                    JOptionPane.INFORMATION_MESSAGE
                );

                System.exit(0);             // Cerrar la ventana y terminar el programa
            }
        });

        setVisible(true);                   // Hacer visible la ventana
    }

    public static void main(String[] args) {
        // Ejecutar la aplicación en el hilo de eventos de Swing
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new VentanaBienvenida2();
            }
        });
    }
}
```

3. Ventana de Bienvenida 3

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class VentanaBienvenida3 extends JFrame {

    public VentanaBienvenida3() {
        setTitle("Bienvenidos");            // Configurar el título de la ventana
        setSize(500, 350);                  // Configurar el tamaño de la ventana
        setLocationRelativeTo(null);        // Centrar la ventana en la pantalla
```

```
// Configurar la operación de cierre por defecto
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

// Crear el panel principal con GridBagLayout
JPanel panel = new JPanel(new GridBagLayout());
GridBagConstraints gbc = new GridBagConstraints();
gbc.insets = new Insets(10, 10, 10, 10); // Espaciado entre componentes

// Crear los botones de radio (Género)
JLabel lblGenero = new JLabel("Género:");

JRadioButton rbMasculino = new JRadioButton("Masculino");
JRadioButton rbFemenino = new JRadioButton("Femenino");

// Agrupar los botones de radio
ButtonGroup grupoGenero = new ButtonGroup();
grupoGenero.add(rbMasculino);
grupoGenero.add(rbFemenino);
rbMasculino.setSelected(true);

// Crear los checkboxes (Destinos de viaje)
JLabel lblDestinos = new JLabel("£Le gustaría viajar a?:");
JCheckBox chkRio = new JCheckBox("Rio de Janeiro");
JCheckBox chkParis = new JCheckBox("París");
JCheckBox chkSingapur = new JCheckBox("Singapur");

// Crear el botón "Salir"
JButton btnSalir = new JButton("Salir");
btnSalir.setPreferredSize(new Dimension(100, 40));

// Posicionar los componentes en el grid
// Fila 0: Etiqueta "Género:"
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 1;
gbc.anchor = GridBagConstraints.WEST;
panel.add(lblGenero, gbc);

// Fila 0: Botón "Masculino"
gbc.gridx = 1;
gbc.gridy = 0;
gbc.gridwidth = 1;
gbc.anchor = GridBagConstraints.WEST;
panel.add(rbMasculino, gbc);

// Fila 0: Botón "Femenino"
gbc.gridx = 2;
gbc.gridy = 0;
gbc.gridwidth = 1;
gbc.anchor = GridBagConstraints.WEST;
panel.add(rbFemenino, gbc);

// Fila 1: Etiqueta "£Le gustaría viajar a?:"
gbc.gridx = 0;
gbc.gridy = 1;
gbc.gridwidth = 3;
gbc.anchor = GridBagConstraints.WEST;
panel.add(lblDestinos, gbc);

// Fila 2: Checkbox "Rio de Janeiro"
gbc.gridx = 0;
gbc.gridy = 2;
gbc.gridwidth = 1;
gbc.anchor = GridBagConstraints.WEST;
panel.add(chkRio, gbc);

// Fila 2: Checkbox "París"
gbc.gridx = 1;
gbc.gridy = 2;
gbc.gridwidth = 1;
gbc.anchor = GridBagConstraints.WEST;
panel.add(chkParis, gbc);

// Fila 2: Checkbox "Singapur"
gbc.gridx = 2;
gbc.gridy = 2;
gbc.gridwidth = 1;
gbc.anchor = GridBagConstraints.WEST;
panel.add(chkSingapur, gbc);

// Fila 3: Botón "Salir" (centrado)
gbc.gridx = 0;
gbc.gridy = 3;
gbc.gridwidth = 3;
gbc.anchor = GridBagConstraints.CENTER;
panel.add(btnSalir, gbc);
```

```
        add(panel);                         // Agregar el panel a la ventana

        // Agregar evento al botón
        btnSalir.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Obtener el género seleccionado
                String genero = "";
                if (rbMasculino.isSelected()) {
                    genero = "Masculino";
                } else if (rbFemenino.isSelected()) {
                    genero = "Femenino";
                }

                // Obtener los destinos seleccionados
                StringBuilder destinos = new StringBuilder();
                if (chkRio.isSelected()) {
                    destinos.append(" Rio de Janeiro\n");
                }
                if (chkParis.isSelected()) {
                    destinos.append(" París\n");
                }
                if (chkSingapur.isSelected()) {
                    destinos.append(" Singapur\n");
                }

                // Si no seleccionó ningún destino
                if (destinos.length() == 0) {
                    destinos.append("(Ningún destino seleccionado)");
                }

                // Mostrar mensaje de confirmación
                JOptionPane.showMessageDialog(
                    VentanaBienvenida3.this,
                    "Género: " + genero + "\n" +
                    "Destinos seleccionados:\n" + destinos.toString(),
                    "Resumen de selección",
                    JOptionPane.INFORMATION_MESSAGE
                );

                // Cerrar la ventana y terminar el programa
                System.exit(0);
            }
        });
        setVisible(true);                   // Hacer visible la ventana
    }

    public static void main(String[] args) {
        // Ejecutar la aplicación en el hilo de eventos de Swing
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new VentanaBienvenida3();
            }
        });
    }
}
```

4. Convertidor de estatura

Este ejercicio consiste en leer la estatura, en metros, de una persona (por ejemplo: 1.70) y el programa retorna su equivalente en pies y pulgadas. Un pie tiene 12 pulgadas y una pulgada equivale a 2.54 cm. Un metro tiene 100 centímetros.

```
// Convierte la estatura de una persona, de metros y centímetros a pies y pulgadas

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ConvertidorEstatura extends JFrame {

    private JTextField txtEstatura;
    private JLabel lblResultado;

    public ConvertidorEstatura() {
        // Configuración de la ventana
        setTitle("Convertidor de Estatura");
        setSize(400, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
```

```
        setLayout(new FlowLayout());

        // Etiqueta de instrucción
        JLabel lblInstruccion = new JLabel("Ingrese su estatura en metros (ej: 1.70):");
        add(lblInstruccion);

        // Campo de texto
        txtEstatura = new JTextField(10);
        add(txtEstatura);

        // Botón para convertir
        JButton btnConvertir = new JButton("Convertir a pies y pulgadas");
        add(btnConvertir);

        // Etiqueta para mostrar el resultado
        lblResultado = new JLabel("Resultado: ");
        add(lblResultado);

        // Acción del botón
        btnConvertir.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                convertirEstatura();
            }
        });

        // También permitir convertir al presionar Enter en el campo de texto
        txtEstatura.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                convertirEstatura();
            }
        });
    }

    private void convertirEstatura() {
        try {
            // Leer la estatura ingresada
            String estaturaTexto = txtEstatura.getText().trim();

            // Validar que no esté vacío
            if (estaturaTexto.isEmpty()) {
                JOptionPane.showMessageDialog(this,
                    "Por favor, ingrese un valor.",
                    "Campo vacío",
                    JOptionPane.WARNING_MESSAGE);
                return;
            }

            // Convertir a número
            double estaturaMetros = Double.parseDouble(estaturaTexto);

            // Validar que sea un valor razonable
            if (estaturaMetros <= 0 || estaturaMetros > 3) {
                JOptionPane.showMessageDialog(this,
                    "Por favor, ingrese una estatura válida (entre 0 y 3 metros).",
                    "Valor inválido",
                    JOptionPane.ERROR_MESSAGE);
                return;
            }

            // Convertir a pies totales
            double piesTotales = estaturaMetros * 3.28084;

            // Obtener pies enteros
            int pies = (int) piesTotales;

            // Obtener pulgadas (parte decimal * 12)
            double pulgadasDecimal = (piesTotales - pies) * 12;
            int pulgadas = (int) Math.round(pulgadasDecimal);

            // Ajustar si las pulgadas llegan a 12 (redondeo)
            if (pulgadas == 12) {
                pies++;
                pulgadas = 0;
            }

            // Mostrar resultado
            lblResultado.setText(String.format("Resultado: %d pies %d pulgadas", pies, pulgadas));

        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this,
                "Por favor, ingrese un número válido (ej: 1.70).",
                "Error de formato",
                JOptionPane.ERROR_MESSAGE);
        }
    }
```

```
    public static void main(String[] args) {
        // Ejecutar la interfaz en el hilo de eventos de Swing
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new ConvertidorEstatura().setVisible(true);
            }
        });
    }
}
```

5. Hamburguesas Bertha

Un negocio de hamburguesas tiene un menú que incluye: Hamburguesa Royal, Hamburguesa Clásica, Hamburguesa Universitaria, Vaso de gaseosa grande, y Vaso de gaseosa chica. Los precios de cada producto se mostrarán en un formulario en el cual el usuario marca el producto que quiere y la cantidad. Terminada la elección, calcula el monto a pagar, el IGV y el total a pagar.

```
// Hamburguesa Royal,         15 soles
// Hamburguesa Clásica,       12 soles
// Hamburguesa Universitaria, 10 soles
// Vaso de gaseosa grande,     6 soles
// Vaso de gaseosa chica,      4 soles
//
// El usuario puede marcar la opción u opciones que desea, y la cantidad de cada producto.
// El programa calcula el monto de la venta, el Impuesto General a las Ventas (18% del total de la venta)
// y el total a pagar
//
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;// El menú de una hamburguesería muestra las opciones siguientes:

public class HamburguesasBertha extends JFrame {

    // Precios de los productos
    private static final double PRECIO_ROYAL = 15.00;
    private static final double PRECIO_CLASICA = 12.00;
    private static final double PRECIO_UNIVERSITARIA = 10.00;
    private static final double PRECIO_GASEOSA_GRANDE = 6.00;
    private static final double PRECIO_GASEOSA_CHICA = 4.00;
    private static final double IGV = 0.18;

    // Componentes de la interfaz
    private JCheckBox chkRoyal, chkClasica, chkUniversitaria;
    private JCheckBox chkGaseosaGrande, chkGaseosaChica;
    private JSpinner spinRoyal, spinClasica, spinUniversitaria;
    private JSpinner spinGaseosaGrande, spinGaseosaChica;
    private JLabel lblSubtotal, lblIgv, lblTotal;
    private JButton btnCalcular, btnResetear, btnSalir;

    public HamburguesasBertha() {
        // Configuración de la ventana
        setTitle("Hamburguesas Bertha - Sistema de Ventas");
        setSize(600, 550);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout(10, 10));

        // Panel principal con scroll
        JPanel panelPrincipal = new JPanel();
        panelPrincipal.setLayout(new BoxLayout(panelPrincipal, BoxLayout.Y_AXIS));
        panelPrincipal.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));

        // Título
        JLabel lblTitulo = new JLabel("HAMBURGUESAS BERTHA");
        lblTitulo.setAlignmentX(Component.CENTER_ALIGNMENT);
        panelPrincipal.add(lblTitulo);
        panelPrincipal.add(Box.createRigidArea(new Dimension(0, 15)));

        // Sección de Hamburguesas
        JPanel panelHamburguesas = new JPanel();
        panelHamburguesas.setBorder(BorderFactory.createTitledBorder("Hamburguesas"));
        panelHamburguesas.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(5, 5, 5, 5);
        gbc.fill = GridBagConstraints.HORIZONTAL;

        // Fila 1: Hamburguesa Royal
        gbc.gridx = 0; gbc.gridy = 0;
        chkRoyal = new JCheckBox("Hamburguesa Royal");
```

```
panelHamburguesas.add(chkRoyal, gbc);

gbc.gridx = 1;
panelHamburguesas.add(new JLabel("S/. " + PRECIO_ROYAL), gbc);

gbc.gridx = 2;
spinRoyal = new JSpinner(new SpinnerNumberModel(0, 0, 99, 1));
spinRoyal.setEnabled(false);
spinRoyal.setPreferredSize(new Dimension(60, 25));
panelHamburguesas.add(spinRoyal, gbc);

// Fila 2: Hamburguesa Clásica
gbc.gridx = 0; gbc.gridy = 1;
chkClasica = new JCheckBox("Hamburguesa Clásica");
panelHamburguesas.add(chkClasica, gbc);

gbc.gridx = 1;
panelHamburguesas.add(new JLabel("S/. " + PRECIO_CLASICA), gbc);

gbc.gridx = 2;
spinClasica = new JSpinner(new SpinnerNumberModel(0, 0, 99, 1));
spinClasica.setEnabled(false);
spinClasica.setPreferredSize(new Dimension(60, 25));
panelHamburguesas.add(spinClasica, gbc);

// Fila 3: Hamburguesa Universitaria
gbc.gridx = 0; gbc.gridy = 2;
chkUniversitaria = new JCheckBox("Hamburguesa Universitaria");
panelHamburguesas.add(chkUniversitaria, gbc);

gbc.gridx = 1;
panelHamburguesas.add(new JLabel("S/. " + PRECIO_UNIVERSITARIA), gbc);

gbc.gridx = 2;
spinUniversitaria = new JSpinner(new SpinnerNumberModel(0, 0, 99, 1));
spinUniversitaria.setEnabled(false);
spinUniversitaria.setPreferredSize(new Dimension(60, 25));
panelHamburguesas.add(spinUniversitaria, gbc);

panelPrincipal.add(panelHamburguesas);
panelPrincipal.add(Box.createRigidArea(new Dimension(0, 10)));

// Sección de Gaseosas
JPanel panelGaseosas = new JPanel();
panelGaseosas.setBorder(BorderFactory.createTitledBorder(" Gaseosas"));
panelGaseosas.setLayout(new GridBagLayout());

// Fila 1: Gaseosa Grande
gbc.gridx = 0; gbc.gridy = 0;
chkGaseosaGrande = new JCheckBox("Vaso de gaseosa grande");
panelGaseosas.add(chkGaseosaGrande, gbc);

gbc.gridx = 1;
panelGaseosas.add(new JLabel("S/. " + PRECIO_GASEOSA_GRANDE), gbc);

gbc.gridx = 2;
spinGaseosaGrande = new JSpinner(new SpinnerNumberModel(0, 0, 99, 1));
spinGaseosaGrande.setEnabled(false);
spinGaseosaGrande.setPreferredSize(new Dimension(60, 25));
panelGaseosas.add(spinGaseosaGrande, gbc);

// Fila 2: Gaseosa Chica
gbc.gridx = 0; gbc.gridy = 1;
chkGaseosaChica = new JCheckBox("Vaso de gaseosa chica");
panelGaseosas.add(chkGaseosaChica, gbc);

gbc.gridx = 1;
panelGaseosas.add(new JLabel("S/. " + PRECIO_GASEOSA_CHICA), gbc);

gbc.gridx = 2;
spinGaseosaChica = new JSpinner(new SpinnerNumberModel(0, 0, 99, 1));
spinGaseosaChica.setEnabled(false);
spinGaseosaChica.setPreferredSize(new Dimension(60, 25));
panelGaseosas.add(spinGaseosaChica, gbc);

panelPrincipal.add(panelGaseosas);
panelPrincipal.add(Box.createRigidArea(new Dimension(0, 15)));

// Sección de Resultados
JPanel panelResultados = new JPanel();
panelResultados.setBorder(BorderFactory.createTitledBorder("Resumen de la Venta"));
panelResultados.setLayout(new GridLayout(3, 2, 10, 5));
panelResultados.setPreferredSize(new Dimension(400, 80));

panelResultados.add(new JLabel("Subtotal (sin IGV):"));
lblSubtotal = new JLabel("S/. 0.00");
lblSubtotal.setForeground(Color.BLUE);
```

```
        panelResultados.add(lblSubtotal);

        panelResultados.add(new JLabel("IGV (18%):"));
        lblIgv = new JLabel("S/. 0.00");
        lblIgv.setForeground(Color.ORANGE);
        panelResultados.add(lblIgv);

        panelResultados.add(new JLabel("Total a pagar:"));
        lblTotal = new JLabel("S/. 0.00");
        lblTotal.setForeground(Color.RED);
        panelResultados.add(lblTotal);

        panelPrincipal.add(panelResultados);
        panelPrincipal.add(Box.createRigidArea(new Dimension(0, 15)));

        // Panel de Botones
        JPanel panelBotones = new JPanel(new FlowLayout(FlowLayout.CENTER, 15, 10));

        btnCalcular = new JButton("Calcular Venta");
        btnCalcular.setBackground(new Color(33, 150, 243));
        btnCalcular.setForeground(Color.WHITE);
        btnCalcular.setFocusPainted(false);

        btnResetear = new JButton(" Resetear Compra");
        btnResetear.setBackground(new Color(255, 152, 0));
        btnResetear.setForeground(Color.WHITE);
        btnResetear.setFocusPainted(false);

        btnSalir = new JButton(" Salir");
        btnSalir.setBackground(new Color(244, 67, 54));
        btnSalir.setForeground(Color.WHITE);
        btnSalir.setFocusPainted(false);

        panelBotones.add(btnCalcular);
        panelBotones.add(btnResetear);
        panelBotones.add(btnSalir);

        panelPrincipal.add(panelBotones);

        // Agregar panel principal con scroll
        JScrollPane scrollPane = new JScrollPane(panelPrincipal);
        scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
        add(scrollPane, BorderLayout.CENTER);

        // Eventos de los CheckBox para habilitar/deshabilitar Spinners
        chkRoyal.addActionListener(e -> spinRoyal.setEnabled(chkRoyal.isSelected()));
        chkClasica.addActionListener(e -> spinClasica.setEnabled(chkClasica.isSelected()));
        chkUniversitaria.addActionListener(e -> spinUniversitaria.setEnabled(chkUniversitaria.isSelected()));
        chkGaseosaGrande.addActionListener(e -> spinGaseosaGrande.setEnabled(chkGaseosaGrande.isSelected()));
        chkGaseosaChica.addActionListener(e -> spinGaseosaChica.setEnabled(chkGaseosaChica.isSelected()));

        // Evento del botón Calcular
        btnCalcular.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                calcularVenta();
            }
        });

        // Evento del botón Resetear
        btnResetear.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                resetearCompra();
            }
        });

        // Evento del botón Salir
        btnSalir.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int confirm = JOptionPane.showConfirmDialog(
                    HamburguesasBertha.this,
                    "£Está seguro que desea salir?",
                    "Confirmar salida",
                    JOptionPane.YES_NO_OPTION
                );
                if (confirm == JOptionPane.YES_OPTION) {
                    System.exit(0);
                }
            }
        });
    }

    private void calcularVenta() {
        double subtotal = 0.0;
        StringBuilder detalle = new StringBuilder();
```

```
detalle.append("DETALLE DE LA VENTA\n");
detalle.append("==================\n\n");

// Verificar Hamburguesa Royal
if (chkRoyal.isSelected()) {
    int cantidad = (int) spinRoyal.getValue();
    if (cantidad > 0) {
        double totalProducto = cantidad * PRECIO_ROYAL;
        subtotal += totalProducto;
        detalle.append(String.format("Royal x%d: S/. %.2f\n", cantidad, totalProducto));
    } else {
        JOptionPane.showMessageDialog(this,
            "Seleccionó Hamburguesa Royal pero la cantidad es 0.\nPor favor, ajuste la cantidad.",
            "Cantidad inválida",
            JOptionPane.WARNING_MESSAGE);
        return;
    }
}

// Verificar Hamburguesa Clásica
if (chkClasica.isSelected()) {
    int cantidad = (int) spinClasica.getValue();
    if (cantidad > 0) {
        double totalProducto = cantidad * PRECIO_CLASICA;
        subtotal += totalProducto;
        detalle.append(String.format("Clásica x%d: S/. %.2f\n", cantidad, totalProducto));
    } else {
        JOptionPane.showMessageDialog(this,
            "Seleccionó Hamburguesa Clásica pero la cantidad es 0.\nPor favor, ajuste la cantidad.",
            "Cantidad inválida",
            JOptionPane.WARNING_MESSAGE);
        return;
    }
}

// Verificar Hamburguesa Universitaria
if (chkUniversitaria.isSelected()) {
    int cantidad = (int) spinUniversitaria.getValue();
    if (cantidad > 0) {
        double totalProducto = cantidad * PRECIO_UNIVERSITARIA;
        subtotal += totalProducto;
        detalle.append(String.format("Universitaria x%d: S/. %.2f\n", cantidad, totalProducto));
    } else {
        JOptionPane.showMessageDialog(this,
            "Seleccionó Hamburguesa Universitaria pero la cantidad es 0.\nPor favor, ajuste la cantidad.",
            "Cantidad inválida",
            JOptionPane.WARNING_MESSAGE);
        return;
    }
}

// Verificar Gaseosa Grande
if (chkGaseosaGrande.isSelected()) {
    int cantidad = (int) spinGaseosaGrande.getValue();
    if (cantidad > 0) {
        double totalProducto = cantidad * PRECIO_GASEOSA_GRANDE;
        subtotal += totalProducto;
        detalle.append(String.format("Gaseosa Grande x%d: S/. %.2f\n", cantidad, totalProducto));
    } else {
        JOptionPane.showMessageDialog(this,
            "Seleccionó Gaseosa Grande pero la cantidad es 0.\nPor favor, ajuste la cantidad.",
            "Cantidad inválida",
            JOptionPane.WARNING_MESSAGE);
        return;
    }
}

// Verificar Gaseosa Chica
if (chkGaseosaChica.isSelected()) {
    int cantidad = (int) spinGaseosaChica.getValue();
    if (cantidad > 0) {
        double totalProducto = cantidad * PRECIO_GASEOSA_CHICA;
        subtotal += totalProducto;
        detalle.append(String.format("Gaseosa Chica x%d: S/. %.2f\n", cantidad, totalProducto));
    } else {
        JOptionPane.showMessageDialog(this,
            "Seleccionó Gaseosa Chica pero la cantidad es 0.\nPor favor, ajuste la cantidad.",
            "Cantidad inválida",
            JOptionPane.WARNING_MESSAGE);
        return;
    }
}

// Verificar que haya seleccionado al menos un producto
if (subtotal == 0.0) {
    JOptionPane.showMessageDialog(this,
        "Por favor, seleccione al menos un producto con cantidad mayor a 0.",
```

```
                "Sin productos",
                JOptionPane.WARNING_MESSAGE);
            return;
        }

        // Calcular IGV y total
        double igv = subtotal * IGV;
        double total = subtotal + igv;

        // Mostrar resultados
        lblSubtotal.setText(String.format("S/. %.2f", subtotal));
        lblIgv.setText(String.format("S/. %.2f", igv));
        lblTotal.setText(String.format("S/. %.2f", total));

        // Mostrar detalle en un cuadro de diálogo
        detalle.append("\n======================\n");
        detalle.append(String.format("Subtotal: S/. %.2f\n", subtotal));
        detalle.append(String.format("IGV (18%%): S/. %.2f\n", igv));
        detalle.append(String.format("TOTAL A PAGAR: S/. %.2f\n", total));

        JOptionPane.showMessageDialog(this,
            detalle.toString(),
            "Factura de Venta",
            JOptionPane.INFORMATION_MESSAGE);
    }

    private void resetearCompra() {
        // Desmarcar todos los checkboxes
        chkRoyal.setSelected(false);
        chkClasica.setSelected(false);
        chkUniversitaria.setSelected(false);
        chkGaseosaGrande.setSelected(false);
        chkGaseosaChica.setSelected(false);

        // Resetear spinners a 0
        spinRoyal.setValue(0);
        spinClasica.setValue(0);
        spinUniversitaria.setValue(0);
        spinGaseosaGrande.setValue(0);
        spinGaseosaChica.setValue(0);

        // Deshabilitar spinners
        spinRoyal.setEnabled(false);
        spinClasica.setEnabled(false);
        spinUniversitaria.setEnabled(false);
        spinGaseosaGrande.setEnabled(false);
        spinGaseosaChica.setEnabled(false);

        // Resetear resultados
        lblSubtotal.setText("S/. 0.00");
        lblIgv.setText("S/. 0.00");
        lblTotal.setText("S/. 0.00");

        JOptionPane.showMessageDialog(this,
            "La compra ha sido reiniciada exitosamente.",
            "Compra reiniciada",
            JOptionPane.INFORMATION_MESSAGE);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new HamburguesasBertha().setVisible(true);
        });
    }
}
```

6. Control de pesca

La pesca en una región está limitada a cierta cantidad por día. Antes de salir a pescar, usted recibe el límite de pesca. Cada pez pescado se pesa y se ingresa al sistema que va acumulando lo pescado. Cuando el valor acumulado iguala o excede al limite, la pesca finaliza. Usted puede finalizar la pesca si ingresa un valor negativo.

```
// El reglamento de pesca de una región establece el límite de pesca que una persona puede realizar
// Antes de salir a pescar, usted lee su límite de pesca.
// Cada presa que pesca, se pesa y su peso se acumula hasta que el total acumulado llega al límite,
// y la pesca termina.
// También la pesca puede terminar cuando ingresa un valor negativo, antes de llegar al límite de pesca,
// lo que indica que dejará de pescar.
//
import javax.swing.*;
import java.awt.*;
```

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.List;

public class ControlPesca extends JFrame {

    private double limitePesca;
    private double totalAcumulado = 0.0;
    private List<Double> pesosCapturados = new ArrayList<>();
    private int numeroPresa = 0;
    private boolean pescaTerminada = false;

    // Componentes de la interfaz
    private JTextField txtLimitePesca;
    private JTextField txtPesoPresa;
    private JTextArea txtAreaRegistro;
    private JLabel lblTotalAcumulado;
    private JLabel lblContadorPresa;
    private JButton btnIniciar;
    private JButton btnRegistrarPresa;
    private JButton btnReiniciar;
    private JButton btnSalir;
    private JPanel panelPesca;

    public ControlPesca() {
        // Configuración de la ventana
        setTitle("Control de Límite de Pesca");
        setSize(650, 550);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout(10, 10));

        // Panel principal
        JPanel panelPrincipal = new JPanel();
        panelPrincipal.setLayout(new BoxLayout(panelPrincipal, BoxLayout.Y_AXIS));
        panelPrincipal.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));

        // Título
        JLabel lblTitulo = new JLabel("CONTROL DE LÍMITE DE PESCA");
        lblTitulo.setAlignmentX(Component.CENTER_ALIGNMENT);
        panelPrincipal.add(lblTitulo);
        panelPrincipal.add(Box.createRigidArea(new Dimension(0, 15)));

        // Panel de Configuración (Límite de pesca)
        JPanel panelConfiguracion = new JPanel();
        panelConfiguracion.setBorder(BorderFactory.createTitledBorder("Configuración del Límite"));
        panelConfiguracion.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));

        panelConfiguracion.add(new JLabel("Límite de pesca (kg):"));
        txtLimitePesca = new JTextField(10);
        panelConfiguracion.add(txtLimitePesca);

        btnIniciar = new JButton("Iniciar Pesca");
        btnIniciar.setBackground(new Color(76, 175, 80));
        btnIniciar.setForeground(Color.WHITE);
        btnIniciar.setFocusPainted(false);
        panelConfiguracion.add(btnIniciar);

        panelPrincipal.add(panelConfiguracion);
        panelPrincipal.add(Box.createRigidArea(new Dimension(0, 10)));

        // Panel de Pesca (se habilita después de iniciar)
        panelPesca = new JPanel();
        panelPesca.setBorder(BorderFactory.createTitledBorder("Registro de Pesca"));
        panelPesca.setLayout(new BoxLayout(panelPesca, BoxLayout.Y_AXIS));
        panelPesca.setEnabled(false);

        // Subpanel para ingreso de peso
        JPanel panelIngreso = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 10));
        panelIngreso.add(new JLabel("Peso de la presa (kg):"));
        txtPesoPresa = new JTextField(10);
        txtPesoPresa.setEnabled(false);
        panelIngreso.add(txtPesoPresa);

        btnRegistrarPresa = new JButton("Registrar Presa");
        btnRegistrarPresa.setBackground(new Color(33, 150, 243));
        btnRegistrarPresa.setForeground(Color.WHITE);
        btnRegistrarPresa.setFocusPainted(false);
        btnRegistrarPresa.setEnabled(false);
        panelIngreso.add(btnRegistrarPresa);

        panelPesca.add(panelIngreso);
        panelPesca.add(Box.createRigidArea(new Dimension(0, 10)));

        // Panel de información
        JPanel panelInfo = new JPanel(new GridLayout(2, 2, 10, 5));
```

```
panelInfo.setBorder(BorderFactory.createEmptyBorder(5, 10, 5, 10));

panelInfo.add(new JLabel("Presas capturadas:"));
lblContadorPresa = new JLabel("0");
lblContadorPresa.setForeground(Color.BLUE);
panelInfo.add(lblContadorPresa);

panelInfo.add(new JLabel("Total acumulado (kg):"));
lblTotalAcumulado = new JLabel("0.00");
lblTotalAcumulado.setForeground(new Color(255, 152, 0));
panelInfo.add(lblTotalAcumulado);

panelPesca.add(panelInfo);
panelPesca.add(Box.createRigidArea(new Dimension(0, 10)));

// Área de registro de capturas
JLabel lblRegistro = new JLabel("Registro de capturas:");
lblRegistro.setAlignmentX(Component.LEFT_ALIGNMENT);
panelPesca.add(lblRegistro);

txtAreaRegistro = new JTextArea(10, 40);
txtAreaRegistro.setEditable(false);
txtAreaRegistro.setBackground(new Color(245, 245, 245));
JScrollPane scrollPane = new JScrollPane(txtAreaRegistro);
scrollPane.setPreferredSize(new Dimension(600, 150));
panelPesca.add(scrollPane);

panelPrincipal.add(panelPesca);
panelPrincipal.add(Box.createRigidArea(new Dimension(0, 10)));

// Panel de botones de control
JPanel panelBotones = new JPanel(new FlowLayout(FlowLayout.CENTER, 15, 10));

btnReiniciar = new JButton("Reiniciar Pesca");
btnReiniciar.setBackground(new Color(255, 152, 0));
btnReiniciar.setForeground(Color.WHITE);
btnReiniciar.setFocusPainted(false);
btnReiniciar.setEnabled(false);

btnSalir = new JButton("Salir");
btnSalir.setBackground(new Color(244, 67, 54));
btnSalir.setForeground(Color.WHITE);
btnSalir.setFocusPainted(false);

panelBotones.add(btnReiniciar);
panelBotones.add(btnSalir);

panelPrincipal.add(panelBotones);

add(panelPrincipal, BorderLayout.CENTER);

// ========== EVENTOS ==========

// Evento del botón Iniciar
btnIniciar.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        iniciarPesca();
    }
});

// Evento del botón Registrar Presa
btnRegistrarPresa.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        registrarPresa();
    }
});

// Evento del botón Reiniciar
btnReiniciar.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        reiniciarPesca();
    }
});

// Evento del botón Salir
btnSalir.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int confirm = JOptionPane.showConfirmDialog(
            ControlPesca.this,
            "£Está seguro que desea salir?",
            "Confirmar salida",
            JOptionPane.YES_NO_OPTION
        );
```

```
            if (confirm == JOptionPane.YES_OPTION) {
                System.exit(0);
            }
        }
    });

    // Enter en el campo de peso también registra
    txtPesoPresa.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (btnRegistrarPresa.isEnabled()) {
                registrarPresa();
            }
        }
    });

    // Enter en el campo de límite también inicia
    txtLimitePesca.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            iniciarPesca();
        }
    });
}

private void iniciarPesca() {
    try {
        String limiteTexto = txtLimitePesca.getText().trim();

        if (limiteTexto.isEmpty()) {
            JOptionPane.showMessageDialog(this,
                "Por favor, ingrese un límite de pesca válido.",
                "Límite requerido",
                JOptionPane.WARNING_MESSAGE);
            return;
        }

        limitePesca = Double.parseDouble(limiteTexto);

        if (limitePesca <= 0) {
            JOptionPane.showMessageDialog(this,
                "El límite de pesca debe ser mayor a 0 kg.",
                "Límite inválido",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        // Reiniciar variables de pesca
        totalAcumulado = 0.0;
        pesosCapturados.clear();
        numeroPresa = 0;
        pescaTerminada = false;

        // Habilitar panel de pesca
        panelPesca.setEnabled(true);
        txtPesoPresa.setEnabled(true);
        btnRegistrarPresa.setEnabled(true);
        btnReiniciar.setEnabled(true);

        // Deshabilitar configuración
        txtLimitePesca.setEnabled(false);
        btnIniciar.setEnabled(false);

        // Limpiar área de registro
        txtAreaRegistro.setText("");

        // Actualizar contadores
        actualizarContadores();

        // Agregar mensaje de inicio
        txtAreaRegistro.append("INICIO DE PESCA\n");
        txtAreaRegistro.append("Límite establecido: " + limitePesca + " kg\n");
        txtAreaRegistro.append("==================================\n\n");

        txtPesoPresa.requestFocus();

        JOptionPane.showMessageDialog(this,
            "Pesca iniciada exitosamente.\n" +
            "Límite: " + limitePesca + " kg\n" +
            "Ingrese el peso de cada presa.",
            "Pesca iniciada",
            JOptionPane.INFORMATION_MESSAGE);

    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(this,
            "Por favor, ingrese un número válido para el límite (ej: 5.5).",
            "Error de formato",
```

```
                JOptionPane.ERROR_MESSAGE);
        }
    }

    private void registrarPresa() {
        if (pescaTerminada) {
            JOptionPane.showMessageDialog(this,
                "La pesca ya ha terminado. Presione 'Reiniciar Pesca' para comenzar de nuevo.",
                "Pesca terminada",
                JOptionPane.INFORMATION_MESSAGE);
            return;
        }

        try {
            String pesotexto = txtPesoPresa.getText().trim();

            if (pesotexto.isEmpty()) {
                JOptionPane.showMessageDialog(this,
                    "Por favor, ingrese el peso de la presa.",
                    "Peso requerido",
                    JOptionPane.WARNING_MESSAGE);
                return;
            }

            double pesoPresa = Double.parseDouble(pesotexto);

            // Verificar si es valor negativo (terminar pesca)
            if (pesoPresa < 0) {
                terminarPesca("Valor negativo ingresado");
                return;
            }

            // Verificar si el peso es 0
            if (pesoPresa == 0) {
                JOptionPane.showMessageDialog(this,
                    "El peso de la presa debe ser mayor a 0 kg.",
                    "Peso inválido",
                    JOptionPane.WARNING_MESSAGE);
                return;
            }

            // Acumular peso
            numeroPresa++;
            totalAcumulado += pesoPresa;
            pesosCapturados.add(pesoPresa);

            // Registrar en el área de texto
            String registro = String.format("Presa #%d: %.2f kg  |  Acumulado: %.2f kg\n",
                numeroPresa, pesoPresa, totalAcumulado);
            txtAreaRegistro.append(registro);

            // Actualizar contadores
            actualizarContadores();

            // Limpiar campo de texto
            txtPesoPresa.setText("");
            txtPesoPresa.requestFocus();

            // Verificar si se alcanzó o superó el límite
            if (totalAcumulado >= limitePesca) {
                terminarPesca("Límite de pesca alcanzado");
            }

        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this,
                "Por favor, ingrese un número válido para el peso (ej: 2.5).\n" +
                "Ingrese un valor negativo para terminar la pesca.",
                "Error de formato",
                JOptionPane.ERROR_MESSAGE);
        }
    }

    private void terminarPesca(String motivo) {
        if (pescaTerminada) {
            return;
        }

        pescaTerminada = true;

        // Deshabilitar ingreso de pesos
        txtPesoPresa.setEnabled(false);
        btnRegistrarPresa.setEnabled(false);

        // Mostrar mensaje según el motivo
        String mensaje;
        String titulo;
        int tipoMensaje;
```

```
        if (motivo.equals("Límite de pesca alcanzado")) {
            mensaje = "ąPESCA TERMINADA!\n\n" +
                      "Motivo: Límite de pesca alcanzado\n" +
                      String.format("Total acumulado: %.2f kg\n", totalAcumulado) +
                      String.format("Límite establecido: %.2f kg\n", limitePesca) +
                      "Presas capturadas: " + numeroPresa + "\n\n" +
                      "ąExcelente pesca!";
            titulo = "Límite alcanzado";
            tipoMensaje = JOptionPane.INFORMATION_MESSAGE;
        } else { // Valor negativo
            mensaje = "ąPESCA TERMINADA!\n\n" +
                      "Motivo: Decisión del pescador\n" +
                      String.format("Total acumulado: %.2f kg\n", totalAcumulado) +
                      String.format("Límite establecido: %.2f kg\n", limitePesca) +
                      "Presas capturadas: " + numeroPresa + "\n\n" +
                      "ąBuena pesca!";
            titulo = "Pesca finalizada";
            tipoMensaje = JOptionPane.INFORMATION_MESSAGE;
        }

        // Agregar mensaje de cierre en el registro
        txtAreaRegistro.append("\n==================================\n");
        txtAreaRegistro.append("PESCA TERMINADA\n");
        txtAreaRegistro.append("Motivo: " + motivo + "\n");
        txtAreaRegistro.append(String.format("Total: %.2f kg\n", totalAcumulado));

        // Mostrar diálogo
        JOptionPane.showMessageDialog(this, mensaje, titulo, tipoMensaje);
    }

    private void actualizarContadores() {
        lblContadorPresa.setText(String.valueOf(numeroPresa));
        lblTotalAcumulado.setText(String.format("%.2f", totalAcumulado));

        // Cambiar color si está cerca del límite
        if (totalAcumulado >= limitePesca * 0.9 && totalAcumulado < limitePesca) {
            lblTotalAcumulado.setForeground(new Color(255, 0, 0)); // Rojo si está cerca
        } else {
            lblTotalAcumulado.setForeground(new Color(255, 152, 0)); // Naranja normal
        }
    }

    private void reiniciarPesca() {
        // Confirmar reinicio
        int confirm = JOptionPane.showConfirmDialog(this,
            "£Está seguro que desea reiniciar la pesca?\n" +
            "Se perderán todos los datos actuales.",
            "Confirmar reinicio",
            JOptionPane.YES_NO_OPTION);

        if (confirm != JOptionPane.YES_OPTION) {
            return;
        }

        // Resetear variables
        totalAcumulado = 0.0;
        pesosCapturados.clear();
        numeroPresa = 0;
        pescaTerminada = false;

        // Resetear componentes
        txtLimitePesca.setEnabled(true);
        txtLimitePesca.setText("");
        btnIniciar.setEnabled(true);

        panelPesca.setEnabled(false);
        txtPesoPresa.setEnabled(false);
        txtPesoPresa.setText("");
        btnRegistrarPresa.setEnabled(false);
        btnReiniciar.setEnabled(false);

        txtAreaRegistro.setText("");
        lblContadorPresa.setText("0");
        lblTotalAcumulado.setText("0.00");
        lblTotalAcumulado.setForeground(new Color(255, 152, 0));

        txtLimitePesca.requestFocus();

        JOptionPane.showMessageDialog(this,
            "Pesca reiniciada exitosamente.\n" +
            "Configure un nuevo límite para comenzar.",
            "Pesca reiniciada",
            JOptionPane.INFORMATION_MESSAGE);
    }

    public static void main(String[] args) {
```

```
        SwingUtilities.invokeLater(() -> {
            new ControlPesca().setVisible(true);
        });
    }
}
```

7. Gestor de películas

Guarde en un archivo texto el título de sus películas favoritas. Su programa debe permitir ingresar el título de una película, ignorarlo o guardarlo en un archivo texto. Cuando se carga el programa, se carga también el número de películas guardadas y los primeros títulos.

```
// Programa Java para leer desde teclado los nombres de película y guardarlos en un archivo texto

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class GestorPeliculas extends JFrame {

    private static final String ARCHIVO = "peliculas.txt";
    private int contadorPeliculas = 0;

    // Componentes de la interfaz
    private JTextField txtPelicula;
    private JLabel lblContador;
    private JTextArea txtAreaLista;
    private JButton btnGuardar;
    private JButton btnVerTodas;
    private JButton btnSalir;

    public GestorPeliculas() {
        // Configuración de la ventana principal
        setTitle("Gestor de Películas");
        setSize(500, 450);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout(10, 10));

        // Contar películas existentes al iniciar
        contadorPeliculas = contarPeliculas();

        // Panel superior (título y contador)
        JPanel panelSuperior = new JPanel();
        panelSuperior.setLayout(new BoxLayout(panelSuperior, BoxLayout.Y_AXIS));
        panelSuperior.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

        JLabel lblTitulo = new JLabel("GESTOR DE PELÍCULAS");
        lblTitulo.setAlignmentX(Component.CENTER_ALIGNMENT);
        panelSuperior.add(lblTitulo);

        JLabel lblSubtitulo = new JLabel("Guarda el título de tus películas favoritas");
        lblSubtitulo.setAlignmentX(Component.CENTER_ALIGNMENT);
        panelSuperior.add(lblSubtitulo);

        panelSuperior.add(Box.createRigidArea(new Dimension(0, 10)));

        JPanel panelContador = new JPanel(new FlowLayout(FlowLayout.CENTER));
        JLabel lblTotal = new JLabel("Películas guardadas: ");
        lblContador = new JLabel(String.valueOf(contadorPeliculas));
        lblContador.setForeground(Color.BLUE);
        panelContador.add(lblTotal);
        panelContador.add(lblContador);
        panelSuperior.add(panelContador);

        add(panelSuperior, BorderLayout.NORTH);

        // Panel central (ingreso de películas)
        JPanel panelCentral = new JPanel();
        panelCentral.setLayout(new BoxLayout(panelCentral, BoxLayout.Y_AXIS));
        panelCentral.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));

        JLabel lblIngreso = new JLabel("Ingrese el nombre de la película:");
        lblIngreso.setAlignmentX(Component.LEFT_ALIGNMENT);
        panelCentral.add(lblIngreso);

        panelCentral.add(Box.createRigidArea(new Dimension(0, 5)));
```

```
        // Campo de texto
        txtPelicula = new JTextField();
        txtPelicula.setMaximumSize(new Dimension(Integer.MAX_VALUE, 30));
        txtPelicula.setPreferredSize(new Dimension(300, 30));
        panelCentral.add(txtPelicula);

        panelCentral.add(Box.createRigidArea(new Dimension(0, 10)));

        // Panel para botones de acción
        JPanel panelBotonesAccion = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 10));
        btnGuardar = new JButton("Guardar Película");
        btnGuardar.setBackground(new Color(76, 175, 80));
        btnGuardar.setForeground(Color.WHITE);
        btnGuardar.setFocusPainted(false);

        btnVerTodas = new JButton("Ver Todas");
        btnVerTodas.setBackground(new Color(33, 150, 243));
        btnVerTodas.setForeground(Color.WHITE);
        btnVerTodas.setFocusPainted(false);

        panelBotonesAccion.add(btnGuardar);
        panelBotonesAccion.add(btnVerTodas);
        panelCentral.add(panelBotonesAccion);

        add(panelCentral, BorderLayout.CENTER);

        // Panel inferior (área de texto para mostrar lista)
        JPanel panelInferior = new JPanel(new BorderLayout());
        panelInferior.setBorder(BorderFactory.createTitledBorder("Lista de Películas"));

        txtAreaLista = new JTextArea(8, 30);
        txtAreaLista.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(txtAreaLista);
        panelInferior.add(scrollPane, BorderLayout.CENTER);

        // Botón Salir
        JPanel panelSalir = new JPanel(new FlowLayout(FlowLayout.RIGHT));
        btnSalir = new JButton("Salir");
        btnSalir.setBackground(new Color(244, 67, 54));
        btnSalir.setForeground(Color.WHITE);
        btnSalir.setFocusPainted(false);
        panelSalir.add(btnSalir);
        panelInferior.add(panelSalir, BorderLayout.SOUTH);

        add(panelInferior, BorderLayout.SOUTH);

        // Cargar la lista inicial
        cargarListaPeliculas();

        // Eventos de los botones
        btnGuardar.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                guardarPelicula();
            }
        });

        btnVerTodas.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                mostrarTodasPeliculas();
            }
        });

        btnSalir.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int confirm = JOptionPane.showConfirmDialog(
                    GestorPeliculas.this,
                    "£Está seguro que desea salir?",
                    "Confirmar salida",
                    JOptionPane.YES_NO_OPTION
                );
                if (confirm == JOptionPane.YES_OPTION) {
                    System.exit(0);
                }
            }
        });

        // Enter en el campo de texto también guarda
        txtPelicula.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                guardarPelicula();
            }
        });
    }
```

```
// Método para contar películas en el archivo
private int contarPeliculas() {
    int count = 0;
    try (BufferedReader reader = new BufferedReader(new FileReader(ARCHIVO))) {
        while (reader.readLine() != null) {
            count++;
        }
    } catch (FileNotFoundException e) {
        // El archivo no existe, es normal la primera vez
    } catch (IOException e) {
        JOptionPane.showMessageDialog(this,
            "Error al leer el archivo: " + e.getMessage(),
            "Error",
            JOptionPane.ERROR_MESSAGE);
    }
    return count;
}

// Método para guardar una película
private void guardarPelicula() {
    String nombrePelicula = txtPelicula.getText().trim();

    if (nombrePelicula.isEmpty()) {
        JOptionPane.showMessageDialog(this,
            "Por favor, ingrese el nombre de una película.",
            "Campo vacío",
            JOptionPane.WARNING_MESSAGE);
        return;
    }

    // Guardar en el archivo
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(ARCHIVO, true))) {
        writer.write(nombrePelicula);
        writer.newLine();
        contadorPeliculas++;
        lblContador.setText(String.valueOf(contadorPeliculas));

        JOptionPane.showMessageDialog(this,
            "Título de película '" + nombrePelicula + "' guardada exitosamente.\n" +
            "Total: " + contadorPeliculas + " títulos de películas",
            "Éxito",
            JOptionPane.INFORMATION_MESSAGE);

        txtPelicula.setText(""); // Limpiar campo
        txtPelicula.requestFocus(); // Enfocar para siguiente ingreso
        cargarListaPeliculas(); // Actualizar lista

    } catch (IOException e) {
        JOptionPane.showMessageDialog(this,
            "Error al guardar la película: " + e.getMessage(),
            "Error",
            JOptionPane.ERROR_MESSAGE);
    }
}

// Método para cargar la lista de películas en el área de texto
private void cargarListaPeliculas() {
    txtAreaLista.setText("");
    try (BufferedReader reader = new BufferedReader(new FileReader(ARCHIVO))) {
        String linea;
        int numero = 1;
        StringBuilder sb = new StringBuilder();

        while ((linea = reader.readLine()) != null) {
            sb.append(String.format("%d. %s%n", numero, linea));
            numero++;
        }

        if (sb.length() == 0) {
            txtAreaLista.setText("No hay películas guardadas aún.");
        } else {
            txtAreaLista.setText(sb.toString());
        }

    } catch (FileNotFoundException e) {
        txtAreaLista.setText("No hay películas guardadas aún.");
    } catch (IOException e) {
        txtAreaLista.setText("Error al leer el archivo: " + e.getMessage());
    }
}

// Método para mostrar todas las películas en una ventana emergente
private void mostrarTodasPeliculas() {
    List<String> peliculas = new ArrayList<>();
    try (BufferedReader reader = new BufferedReader(new FileReader(ARCHIVO))) {
        String linea;
```

```
            while ((linea = reader.readLine()) != null) {
                peliculas.add(linea);
            }
        } catch (FileNotFoundException e) {
            JOptionPane.showMessageDialog(this,
                "No hay películas guardadas aún.",
                "Lista vacía",
                JOptionPane.INFORMATION_MESSAGE);
            return;
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this,
                "Error al leer el archivo: " + e.getMessage(),
                "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        if (peliculas.isEmpty()) {
            JOptionPane.showMessageDialog(this,
                "No hay películas guardadas aún.",
                "Lista vacía",
                JOptionPane.INFORMATION_MESSAGE);
            return;
        }

        // Crear un cuadro de diálogo con la lista
        JDialog dialog = new JDialog(this, "Lista de Películas", true);
        dialog.setLayout(new BorderLayout());
        dialog.setSize(400, 400);
        dialog.setLocationRelativeTo(this);

        JTextArea textArea = new JTextArea();
        textArea.setEditable(false);

        StringBuilder sb = new StringBuilder();
        sb.append("PELÍCULAS GUARDADAS\n");
        sb.append("============================\n\n");
        for (int i = 0; i < peliculas.size(); i++) {
            sb.append(String.format("%d. %s\n", i + 1, peliculas.get(i)));
        }
        sb.append("\n============================\n");
        sb.append("Total: ").append(peliculas.size()).append(" películas");
        textArea.setText(sb.toString());

        JScrollPane scrollPane = new JScrollPane(textArea);
        dialog.add(scrollPane, BorderLayout.CENTER);

        JButton btnCerrar = new JButton("Cerrar");
        btnCerrar.addActionListener(e -> dialog.dispose());
        JPanel panelBoton = new JPanel();
        panelBoton.add(btnCerrar);
        dialog.add(panelBoton, BorderLayout.SOUTH);

        dialog.setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new GestorPeliculas().setVisible(true);
        });
    }
}
```