“Año del Bicentenario, de la consolidación de nuestra Independencia, y de la conmemoración de las heroicas batallas de Junín y Ayacucho”

“UNIVERSIDAD PERUANA LOS ANDES”

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Trabajo 4

Alumno:

- **Rojas Pariona Miguel Angel**

Docente: Fernandez Bejarano Raul Enrique

Asignatura: ARQUITECTURA DE SOFTWARE

Sección: B1

Ciclo: VII

Huancayo - Perú

2024

Sistema de Registro de Estudiantes

Descripción: El sistema de registro de estudiantes permitirá gestionar la información básica de los estudiantes en una universidad. A través de una interfaz gráfica, los usuarios podrán ingresar, buscar y almacenar datos de estudiantes, incluyendo su código, nombre, fecha de nacimiento, teléfono y dirección. El sistema también debe ser capaz de cargar y guardar la información de los estudiantes en un archivo de texto, asegurando que los datos sean persistentes entre sesiones.

Objetivos:

- Facilitar el registro de estudiantes en el sistema.
- Permitir la búsqueda de estudiantes por nombre.
- Almacenar la información de estudiantes de manera persistente en un archivo.
- Proporcionar una interfaz gráfica intuitiva para los usuarios.

Requerimientos Funcionales

Nombre:	***R1: Registro de Estudiantes***
Resumen:	***Permite al usuario ingresar los datos de un nuevo alumno.***
Entrada:	***Código del estudiante, nombre, teléfono, dirección, y fecha de nacimiento.***
Resultados:	***El alumno es registrado en el sistema.***

Nombre:	***R2: Cálculo de Edad***
Resumen:	***Calcular la edad del estudiante automáticamente al registrar su información.***
Entrada:	***Fecha de nacimiento del estudiante (Date)***
Resultados:	***Se calculará la edad del estudiante y se asignará al atributo correspondiente en el objeto Estudiante.***

Nombre:	***R3: Visualización de Estudiantes***
Resumen:	***Mostrar la lista de estudiantes registrados en una tabla.***
Entrada:	***Lista de estudiantes (ArrayList<Estudiante>)***
Resultados:	***La tabla de estudiantes se llenará con los datos de todos los estudiantes registrados.***

Nombre:	***R4: Búsqueda de Estudiantes***
Resumen:	***Permitir a los usuarios buscar estudiantes por su nombre.***
Entrada:	***Nombre a buscar.***
Resultados:	***La tabla se actualizará para mostrar solo los estudiantes cuyo nombre coincide con el texto de búsqueda.***

Nombre:	***R5: Carga y Guardado de Datos***
Resumen:	***Permitir al sistema cargar y guardar datos de estudiantes desde/hacia un archivo de texto.***
Entrada:	***Ruta del archivo***
Resultados:	• ***Al cargar, los estudiantes del archivo se añadirán a la lista y se mostrarán en la tabla.*** • ***Al guardar, los datos de todos los estudiantes registrados se escribirán en el archivo especificado.*** • ***Se mostrará un mensaje de confirmación al usuario sobre el éxito de la operación.***

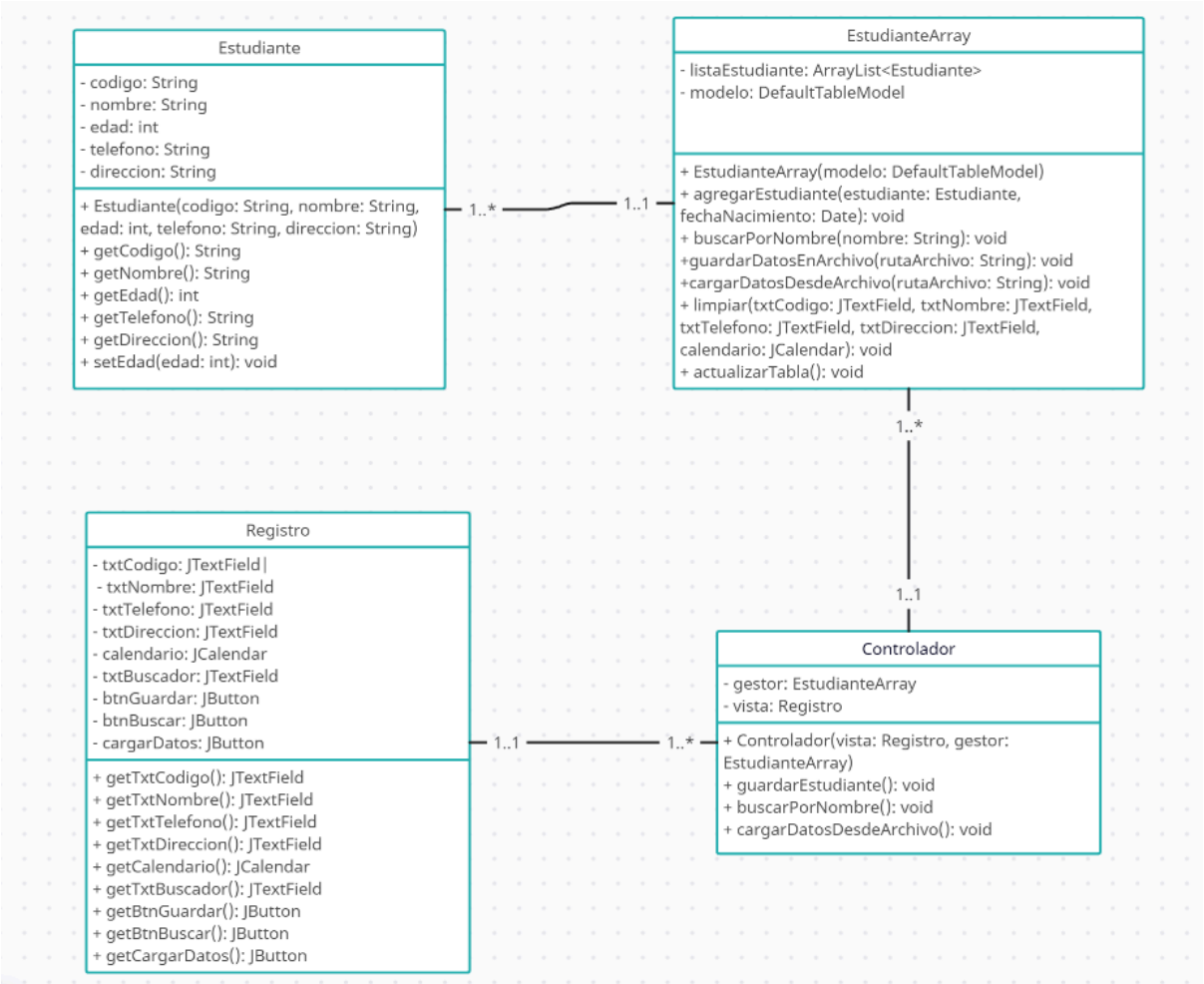
Nombre:	***R6: Limpieza de Campos***
Resumen:	***Proporcionar una función para limpiar los campos del formulario después de guardar un estudiante.***
Entrada:	***Campos de entrada del formulario.***
Resultados:	***Todos los campos del formulario se limpiarán, permitiendo al usuario ingresar nuevos datos sin interferencias.***

Diagrama de Clases para el Sistema de Registro de Estudiantes

1. **Estudiante**:
    - **Atributos**:
        - `codigo`: Identificador del estudiante.
        - `nombre`: Nombre del estudiante.
        - `edad`: Edad calculada del estudiante.
        - `telefono`: Número de teléfono del estudiante.
        - `direccion`: Dirección del estudiante.
    - **Métodos**:
        - Métodos para obtener y establecer los atributos.
2. **EstudianteArray**:
    - **Atributos**:
        - `listaEstudiante`: Lista de estudiantes.
        - `modelo`: Modelo de la tabla para mostrar los estudiantes.
    - **Métodos**:
        - Métodos para agregar, buscar, guardar y cargar datos, así como limpiar campos y actualizar la tabla.
3. **Controlador**:
    - **Atributos**:
        - `gestor`: Referencia al `EstudianteArray`.
        - `vista`: Referencia a la vista (formulario).
    - **Métodos**:
        - Métodos para guardar estudiantes, buscar por nombre y cargar datos desde un archivo.
4. **Registro**:
    - **Atributos**:
        - Campos de texto para entrada de datos (`txtCodigo`, `txtNombre`, `txtTelefono`, `txtDireccion`, `txtBuscador`) y botones (`btnGuardar`, `btnBuscar`, `cargarDatos`).
    - **Métodos**:
        - Métodos para obtener los componentes de la interfaz gráfica.

Relaciones

- **Asociación**:
    - `Controlador` tiene una asociación con `EstudianteArray` (1 a muchos).
    - `EstudianteArray` tiene una asociación con `Estudiante` (1 a muchos).
    - `Controlador` tiene una asociación con `Registro` (1 a 1).



Código

ESTUDIANTE

```
package Modelo;

public class Estudiante {
    private String codigo;
    private String nombre;
    private int edad; // Cambiado a int para que coincida con Integer.parseInt
    private String telefono;
    private String direccion;

    public Estudiante(String codigo, String nombre, int edad, String telefono, String direccion) {
        this.codigo = codigo;
        this.nombre = nombre;
```

```
        this.edad = edad;
        this.telefono = telefono;
        this.direccion = direccion;
    }


    // Constructor con edad
    public Estudiante(String codigo, String nombre, String telefono, String direccion,
int edad) {
        this.codigo = codigo;
        this.nombre = nombre;
        this.telefono = telefono;
        this.direccion = direccion;
        this.edad = edad;
    }

    // Getters y Setters
    public String getCodigo() { return codigo; }
    public String getNombre() { return nombre; }
    public String getTelefono() { return telefono; }
    public String getDireccion() { return direccion; }
    public int getEdad() { return edad; }
    public void setEdad(int edad) { this.edad = edad; }
}
```

ESTUDIANTE ARRAY

```
package Modelo;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author USER
```

```
 */
public class EstudianteArray {

    private ArrayList<Estudiante> listaEstudiante;
    private DefaultTableModel modelo;

    // Constructor que acepta DefaultTableModel
    public EstudianteArray(DefaultTableModel modelo) {
        this.listaEstudiante = new ArrayList<>();
        this.modelo = modelo;
    }

public void agregarEstudiante(Estudiante estudiante, Date fechaNacimiento) {
    if (listaEstudiante.size() >= 5) {
        JOptionPane.showMessageDialog(null, "No se pueden registrar más de 5
estudiantes.");
        return;
    }

    // Calcular la edad basada en la fecha de nacimiento
    int edad = calcularEdad(fechaNacimiento);
    estudiante.setEdad(edad);  // Asigna la edad calculada

    listaEstudiante.add(estudiante);
    actualizarTabla(); // Actualizar la tabla al agregar un estudiante
    guardarDatosEnArchivo("C:\\Users\\migue\\Downloads\\Semana
05\\estudiantes.txt");
}

    // Método para calcular la edad en función de la fecha de nacimiento
    public int calcularEdad(Date fechaNacimiento) {
        Calendar cal = Calendar.getInstance();
        cal.setTime(fechaNacimiento);
        int añoNacimiento = cal.get(Calendar.YEAR);
        int añoActual = Calendar.getInstance().get(Calendar.YEAR);
        return añoActual - añoNacimiento;
    }

    public void actualizarTabla() {
        // Limpiar la tabla actual
        modelo.setRowCount(0);

        // Agregar todos los estudiantes a la tabla
        for (Estudiante e : listaEstudiante) {
```

```
            modelo.addRow(new Object[]{e.getCodigo(), e.getNombre(), e.getEdad(),
e.getTelefono(), e.getDireccion()});
        }
    }

    // Método para limpiar los campos del formulario
    public void limpiar(javax.swing.JTextField txtCodigo, javax.swing.JTextField
txtNombre,
            javax.swing.JTextField txtTelefono, javax.swing.JTextField txtDireccion,
            com.toedter.calendar.JCalendar calendario) {
        txtCodigo.setText("");
        txtNombre.setText("");
        txtTelefono.setText("");
        txtDireccion.setText("");
        calendario.setDate(null);
    }

    public void buscarPorNombre(String nombre) {
        // Limpiar la tabla actual
        modelo.setRowCount(0);

        // Filtrar la lista por nombre y agregar los resultados a la tabla
        for (Estudiante e : listaEstudiante) {
            if (e.getNombre().toLowerCase().contains(nombre.toLowerCase())) {
                modelo.addRow(new Object[]{e.getCodigo(), e.getNombre(), e.getEdad(),
e.getTelefono(), e.getDireccion()});
            }
        }
    }

    public ArrayList<Estudiante> getListaEstudiantes() {
        return listaEstudiante;
    }

    public void guardarDatosEnArchivo(String rutaArchivo) {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(rutaArchivo))) {
            for (Estudiante estudiante : listaEstudiante) {
                writer.write(estudiante.getCodigo() + ","
                        + estudiante.getNombre() + ","
                        + estudiante.getEdad() + ","
                        + estudiante.getTelefono() + ","
                        + estudiante.getDireccion());
                writer.newLine();
            }
```

```
            JOptionPane.showMessageDialog(null, "Datos guardados exitosamente en "
+ rutaArchivo);
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "Error al guardar los datos: " +
e.getMessage());
        }
    }

    public void cargarDatosDesdeArchivo(String rutaArchivo) {
        try (BufferedReader reader = new BufferedReader(new
FileReader(rutaArchivo))) {
            String linea;
            boolean archivoVacio = true;

            listaEstudiante.clear();
            modelo.setRowCount(0);

            while ((linea = reader.readLine()) != null) {
                archivoVacio = false;
                String[] datos = linea.split(",");
                if (datos.length == 5) {
                    // En lugar de crear un nuevo estudiante aquí, puedes usar los datos
para crear el objeto en otro lugar si es necesario.
                    Estudiante estudiante = new Estudiante(datos[0], datos[1],
Integer.parseInt(datos[2]), datos[3], datos[4]);

                    listaEstudiante.add(estudiante);
                    modelo.addRow(new Object[]{datos[0], datos[1], datos[2], datos[3],
datos[4]});
                }
            }

            if (archivoVacio) {
                JOptionPane.showMessageDialog(null, "El archivo está vacío, la tabla y la
lista han sido limpiadas.");
            }

        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "Error al cargar los datos: " +
e.getMessage());
        }
    }
}
```

REGISTRO

```
package Vista;
import Controlador.Controlador;
import Modelo.EstudianteArray;
import javax.swing.table.DefaultTableModel;
/**
 *
 * @author miguel
 */
public class Registro extends javax.swing.JFrame {

  private DefaultTableModel modelo;
    private EstudianteArray gestor;
    private Controlador controlador;
    /**
     * Creates new form Registro
     */
public Registro() {
    initComponents();
    setLocationRelativeTo(null);

    // Inicializar el modelo de la tabla
    modelo = new DefaultTableModel();
    modelo.addColumn("Código");
    modelo.addColumn("Nombre");
    modelo.addColumn("Edad");
    modelo.addColumn("Teléfono");
    modelo.addColumn("Dirección");

    // Asignar el modelo a la tabla
    tblEstudiante.setModel(modelo);

    // Crear el gestor con el modelo ya inicializado
    gestor = new EstudianteArray(modelo);
    controlador = new Controlador(this, gestor);

    /*
    // Cargar los datos desde el archivo al iniciar el programa
    String rutaArchivo = "D:\\DISCO D\\INGENIERIA DE SISTEMAS\\VII
CICLO\\Esctructura de software\\Semana 05\\estudiantes.txt";
    gestor.cargarDatosDesdeArchivo(rutaArchivo);
    */
}
```

```
// Método get para el botón btnBuscar
    public javax.swing.JButton getBtnBuscar() {
        return btnBuscar;
    }

    public javax.swing.JButton getBtnGuardar() {
        return btnGuardar;
    }

   public javax.swing.JButton getCargarDatos() {
        return btnCargarDatos;
    }
public javax.swing.JTextField getTxtDireccion() {
    return txtDireccion;
}

    public javax.swing.JTextField getTxtNombre() {
        return txtNombre;
    }

    public javax.swing.JTextField getTxtCodigo() {
        return txtCodigo;
    }
    public com.toedter.calendar.JCalendar getCalendario() {
        return Calendario;
    }

    public javax.swing.JTextField getTxtTelefono() {
        return txtTelefono;
    }
    public javax.swing.JTextField getTxtBuscador() {
        return txtBuscador;
    }

    public javax.swing.JTextField getJTextDireccion() {
        return txtDireccion;
    }

    public javax.swing.JTextField getTxtBuscar() {
        return txtBuscador;
    }

    private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {
         controlador.buscarPorNombre();
```

```
    }

    private void txtBuscadorActionPerformed(java.awt.event.ActionEvent evt) {

    }

    private void txtNombreActionPerformed(java.awt.event.ActionEvent evt) {

    }

    private void txtCodigoActionPerformed(java.awt.event.ActionEvent evt) {

    }

    private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
        controlador.guardarEstudiante();
    }

    private void btnCargarDatosActionPerformed(java.awt.event.ActionEvent evt) {
     controlador.cargarDatosDesdeArchivo();
    }

    private void txtDireccionActionPerformed(java.awt.event.ActionEvent evt) {

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look
and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
```

```
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Registro.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Registro.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Registro.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Registro.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Registro().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private com.toedter.calendar.JCalendar Calendario;
private javax.swing.JButton btnBuscar;
private javax.swing.JButton btnCargarDatos;
private javax.swing.JButton btnGuardar;
private com.jtattoo.plaf.hifi.HiFiLookAndFeel hiFiLookAndFeel1;
private javax.swing.JLabel jCodigo;
private com.toedter.calendar.JDateChooser jDateChooser1;
private javax.swing.JLabel jFecha;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jNombre;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JLabel jTelefono;
private javax.swing.JLabel jTitulo;
private javax.swing.JTable tblEstudiante;
```

```
    private javax.swing.JTextField txtBuscador;
    private javax.swing.JTextField txtCodigo;
    private javax.swing.JTextField txtDireccion;
    private javax.swing.JTextField txtNombre;
    private javax.swing.JTextField txtTelefono;
    // End of variables declaration

}
```

CONTROLADOR

```
package Controlador;

import Modelo.Estudiante;
import Modelo.EstudianteArray;
import Vista.Registro;
import java.util.Calendar;
import java.util.Date;
import javax.swing.JOptionPane;

/**
 *
 * @author USER
 */
public class Controlador {

    private EstudianteArray gestor;
    private Registro vista;

    public Controlador(Registro vista, EstudianteArray gestor) {
        this.vista = vista;
        this.gestor = gestor;

        // Asignar eventos usando los getters de los botones
        this.vista.getBtnGuardar().addActionListener(e -> guardarEstudiante());
        this.vista.getBtnBuscar().addActionListener(e -> buscarPorNombre());
        this.vista.getCargarDatos().addActionListener(e ->
cargarDatosDesdeArchivo());
    }

public void guardarEstudiante() {
    // Usar los getters para acceder a los campos
    if (vista.getTxtCodigo().getText().isEmpty() ||
```

```
        vista.getTxtNombre().getText().isEmpty() ||
        vista.getTxtTelefono().getText().isEmpty() ||
        vista.getCalendario().getDate() == null ||
        vista.getTxtDireccion().getText().isEmpty()) {
        JOptionPane.showMessageDialog(vista, "Por favor, complete todos los
campos.");
        return;
    }

    // Crear un nuevo estudiante y asignar los valores de los campos
Estudiante estudiante = new Estudiante(
    vista.getTxtCodigo().getText(),
    vista.getTxtNombre().getText(),
    vista.getTxtTelefono().getText(),
    vista.getTxtDireccion().getText(),
    0 // Puedes pasar 0 temporalmente, ya que la edad se calculará en
EstudianteArray
);

    // Agregar el estudiante al gestor, pasando la fecha de nacimiento
    gestor.agregarEstudiante(estudiante, vista.getCalendario().getDate());

    // Limpiar los campos después de guardar
    gestor.limpiar(vista.getTxtCodigo(), vista.getTxtNombre(), vista.getTxtTelefono(),
vista.getTxtDireccion(), vista.getCalendario());

    // Actualizar la tabla
    gestor.actualizarTabla();

    String rutaArchivo = "C:\\Users\\migue\\Downloads\\Semana 05\\estudiantes.txt";
    gestor.guardarDatosEnArchivo(rutaArchivo);
}


    public void buscarPorNombre() {
        String nombreBuscado = vista.getTxtBuscador().getText();
        gestor.buscarPorNombre(nombreBuscado); // Actualiza la tabla según el texto
buscado
    }

    public void cargarDatosDesdeArchivo() {
        // Cargar los datos desde el archivo al iniciar el programa
        String rutaArchivo = "C:\\Users\\migue\\Downloads\\Semana
05\\estudiantes.txt";
```

```
        gestor.cargarDatosDesdeArchivo(rutaArchivo);
    }

    private int calcularEdad(Date fechaNacimiento) {
        Calendar nacimiento = Calendar.getInstance();
        nacimiento.setTime(fechaNacimiento);

        Calendar hoy = Calendar.getInstance();
        int edad = hoy.get(Calendar.YEAR) - nacimiento.get(Calendar.YEAR);

        // Verificar si ya cumplió años este año
        if (hoy.get(Calendar.DAY_OF_YEAR) <
nacimiento.get(Calendar.DAY_OF_YEAR)) {
            edad--;
        }

        return edad;
    }
}
```

MAIN

```
package com.mycompany.semana04_rojas_miguel;

import Vista.Registro;

/**
 *
 * @author miguel
 */
public class Semana04_Rojas_Miguel {

    public static void main(String[] args) {
        try {
            // Set HiFi Look and Feel from JTattoo

javax.swing.UIManager.setLookAndFeel("com.jtattoo.plaf.hifi.HiFiLookAndFeel");
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Semana04_Rojas_Miguel.class.getName()).log(ja
va.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
```

```
java.util.logging.Logger.getLogger(Semana04_Rojas_Miguel.class.getName()).log(ja
va.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Semana04_Rojas_Miguel.class.getName()).log(ja
va.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Semana04_Rojas_Miguel.class.getName()).log(ja
va.util.logging.Level.SEVERE, null, ex);
        }

         java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Registro().setVisible(true);
        }
    });
    }
}
```