Objetos, GUI y Composición en Java

Usaremos **Swing** para la interfaz gráfica y aplicaremos el principio de **separar la lógica de negocio (objetos) de la interfaz de usuario**.

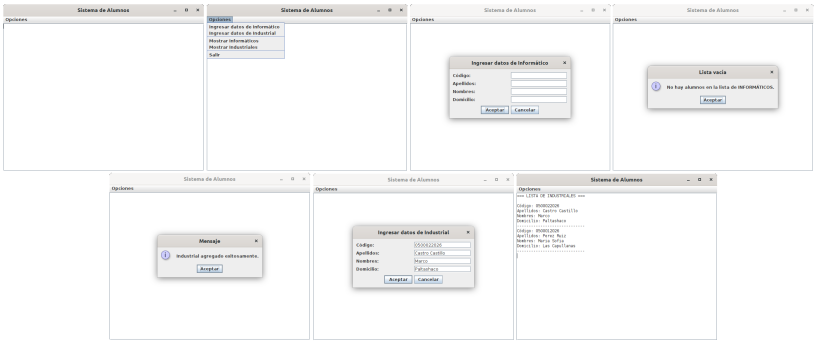
Estructura del proyecto

1. **Clase** `Alumno` (modelo)
2. **Clase** `AlumnoGUI` (interfaz gráfica con menú)
3. **Colecciones** para almacenar informáticos e industriales

Explicación del código

Descargar código: https://github.com/willycruzy/Java-GUI

- **Clase Alumno**
  - Implementa `Comparable<Alumno>` para permitir ordenamiento.
  - El método `compareTo()` ordena primero por **Apellidos** y luego por **Nombres**.
  - El método `toString()` formatea los datos para mostrarlos en forma elegante.
- **Clase AlumnoGUI**
  - Usa dos `ArrayList<Alumno>` para almacenar informáticos e industriales.
  - **Menú** con `JMenuBar` y `JMenuItem` para las 5 opciones requeridas.
  - **Validación**: Al ingresar datos, se verifica que ningún campo esté vacío usando `trim()` y `isEmpty()`
  - **Ordenamiento**: Usa `Collections.sort(lista)` que automáticamente usa el `compareTo()` de la clase `Alumno`.
  - **Muestra**: Los datos se muestran en un `JTextArea` con *scroll*.
- **Características**
  - Clase `Alumno` con los 4 atributos.
  - Instancias `AlInformatica` y `AlIndustrial` (listas).
  - Menú con 5 opciones.
  - Validación de campos vacíos.
  - Mostrar ordenado por Apellidos y Nombres.



Composición

La **composición** en Java es un **mecanismo** de la *Programación Orientada a Objetos* (**POO**) que **permite reutilizar código creando una nueva clase que agrupa objetos de otras clases preexistentes**, estableciendo una relación de tipo *tiene-un* o *parte de un todo*.

A diferencia de la **herencia** (relación *es-un*), la **composición** es una asociación fuerte y estática donde:

- **La clase contenedora posee las instancias de las clases compuestas**, que normalmente se declaran como ***private***.
- **Los objetos contenidos dependen del objeto contenedor**; si este se destruye, los objetos internos también dejan de existir o perder su contexto funcional (ejemplo: un Motor no puede funcionar sin el Coche que lo compone).
- **Permite cambiar la implementación en tiempo de ejecución** y facilita la reutilización de código sin crear jerarquías rígidas de clases.

Estructura de Composición para Estudiantes

```
public class Estudiante {
    // COMPOSICIÓN: El estudiante ESTÁ COMPUESTO POR tres objetos
    private DatosPersonales datosPersonales;    // Componente 1
    private DatosAcademicos datosAcademicos;    // Componente 2
    private DatosContacto datosContacto;        // Componente 3
}
```

Ventajas

Aspecto	Beneficio
Separación de responsabilidades	Cada componente maneja su propia lógica
Reutilización	Los componentes pueden usarse en otros contextos
Mantenibilidad	Cambios en un componente no afectan a los demás
Escalabilidad	Fácil agregar nuevos componentes (ej. DatosLaborales)
Pruebas	Cada componente se puede probar independientemente

```
Estudiante (compuesto por)
|--- DatosPersonales (nombre, apellidos, domicilio)
|--- DatosAcademicos (código, carrera, promedio)
|--- DatosContacto (email, teléfono, redes sociales)
```

La GUI accede a los componentes a través del objeto compuesto

```
estudiante.getDatosPersonales().getNombres();
estudiante.getDatosAcademicos().getPromedio();
estudiante.getDatosContacto().getEmail();
```

Composición clara y visible

```
Estudiante
|--- DatosPersonales  (nombre, apellidos, domicilio, fecha nac.)
|--- DatosAcademicos  (código, carrera, promedio, ciclo, estado)
|--- DatosContacto    (email, teléfono, celular, redes sociales)
```

Características

- Cada componente tiene su propia lógica y responsabilidad
- El objeto `Estudiante` delega tareas a sus componentes
- La GUI interactúa con el objeto compuesto
- Fácil de extender (ej. agregar `DatosLaborales`, `DatosFamiliares`, etc.)

Funcionalidades de la GUI

- Ingreso de estudiantes con pestañas organizadas
- Tabla con filtros por carrera
- Detalle completo del estudiante
- Cambio de estado académico
- Exportar resumen
- Eliminar estudiantes

Ver en anexo el código de:

- DatosPersonales.java
- DatosAcadémicos.java
- DatosContacto.java
- Estudiante.java
- GestorEstudiantesGUI.java

Descargar código: https://github.com/willycruzy/Java-GUI

Anexos

Clase Alumno.java (Modelo)

```
import java.util.Objects;

public class Alumno implements Comparable<Alumno> {
    private String codigo;
    private String apellidos;
    private String nombres;
    private String domicilio;

    // Constructor
    public Alumno(String codigo, String apellidos, String nombres, String domicilio) {
        this.codigo = codigo;
        this.apellidos = apellidos;
        this.nombres = nombres;
        this.domicilio = domicilio;
    }

    // Getters y Setters
    public String getCodigo() { return codigo; }
    public void setCodigo(String codigo) { this.codigo = codigo; }

    public String getApellidos() { return apellidos; }
    public void setApellidos(String apellidos) { this.apellidos = apellidos; }

    public String getNombres() { return nombres; }
    public void setNombres(String nombres) { this.nombres = nombres; }

    public String getDomicilio() { return domicilio; }
    public void setDomicilio(String domicilio) { this.domicilio = domicilio; }

    // Para ordenar por Apellidos y luego Nombres
    @Override
    public int compareTo(Alumno otro) {
        int compararApellidos = this.apellidos.compareTo(otro.apellidos);
        if (compararApellidos != 0) {
            return compararApellidos;
        }
        return this.nombres.compareTo(otro.nombres);
    }

    // Para mostrar los datos en el JTextArea
    @Override
    public String toString() {
        return "Código: " + codigo +
               "\nApellidos: " + apellidos +
               "\nNombres: " + nombres +
               "\nDomicilio: " + domicilio +
               "\n------------------------------";
    }
}
```

AlumnoGUI.java - Clase Principal con la GUI

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

public class AlumnoGUI extends JFrame {
    // Colecciones para almacenar alumnos
    private ArrayList<Alumno> listaInformaticos;
    private ArrayList<Alumno> listaIndustriales;

    // Componentes de la GUI
    private JTextArea areaMostrar;

    public AlumnoGUI() {
        // Inicializar listas
        listaInformaticos = new ArrayList<>();
        listaIndustriales = new ArrayList<>();

        // Configurar la ventana principal
        setTitle("Sistema de Alumnos");
        setSize(600, 500);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());

        // Crear el menú
        crearMenu();

        // Área para mostrar información
        areaMostrar = new JTextArea();
        areaMostrar.setEditable(false);
        areaMostrar.setFont(new Font("Monospaced", Font.PLAIN, 12));
        JScrollPane scrollPane = new JScrollPane(areaMostrar);
        add(scrollPane, BorderLayout.CENTER);

        setVisible(true);
    }

    private void crearMenu() {
        JMenuBar menuBar = new JMenuBar();

        // Menú "Opciones"
        JMenu menuOpciones = new JMenu("Opciones");

        // Opción 1: Ingresar Informático
        JMenuItem itemIngresarInfo = new JMenuItem("Ingresar datos de Informático");
        itemIngresarInfo.addActionListener(e -> ingresarAlumno("Informático"));
        menuOpciones.add(itemIngresarInfo);

        // Opción 2: Ingresar Industrial
        JMenuItem itemIngresarInd = new JMenuItem("Ingresar datos de Industrial");
        itemIngresarInd.addActionListener(e -> ingresarAlumno("Industrial"));
        menuOpciones.add(itemIngresarInd);

        menuOpciones.addSeparator();

        // Opción 3: Mostrar Informáticos
        JMenuItem itemMostrarInfo = new JMenuItem("Mostrar Informáticos");
        itemMostrarInfo.addActionListener(e -> mostrarAlumnos(listaInformaticos, "INFORMÁTICOS"));
        menuOpciones.add(itemMostrarInfo);

        // Opción 4: Mostrar Industriales
        JMenuItem itemMostrarInd = new JMenuItem("Mostrar Industriales");
        itemMostrarInd.addActionListener(e -> mostrarAlumnos(listaIndustriales, "INDUSTRIALES"));
        menuOpciones.add(itemMostrarInd);

        menuOpciones.addSeparator();

        // Opción 5: Salir
```

```
        JMenuItem itemSalir = new JMenuItem("Salir");
        itemSalir.addActionListener(e -> System.exit(0));
        menuOpciones.add(itemSalir);

        menuBar.add(menuOpciones);
        setJMenuBar(menuBar);
    }

    // Método para ingresar un alumno (validando campos vacíos)
    private void ingresarAlumno(String tipo) {
        // Diálogo personalizado con JOptionPane
        JPanel panel = new JPanel(new GridLayout(4, 2, 5, 5));
        JTextField txtCodigo = new JTextField(15);
        JTextField txtApellidos = new JTextField(15);
        JTextField txtNombres = new JTextField(15);
        JTextField txtDomicilio = new JTextField(15);

        panel.add(new JLabel("Código:"));
        panel.add(txtCodigo);
        panel.add(new JLabel("Apellidos:"));
        panel.add(txtApellidos);
        panel.add(new JLabel("Nombres:"));
        panel.add(txtNombres);
        panel.add(new JLabel("Domicilio:"));
        panel.add(txtDomicilio);

        int resultado = JOptionPane.showConfirmDialog(
            this,
            panel,
            "Ingresar datos de " + tipo,
            JOptionPane.OK_CANCEL_OPTION,
            JOptionPane.PLAIN_MESSAGE
        );

        if (resultado == JOptionPane.OK_OPTION) {
            // Validar que ningún campo esté vacío
            String codigo = txtCodigo.getText().trim();
            String apellidos = txtApellidos.getText().trim();
            String nombres = txtNombres.getText().trim();
            String domicilio = txtDomicilio.getText().trim();

            if (codigo.isEmpty() || apellidos.isEmpty() || nombres.isEmpty() || domicilio.isEmpty()) {
                JOptionPane.showMessageDialog(
                    this,
                    "Todos los campos son obligatorios. No se permiten campos vacíos.",
                    "Error de validación",
                    JOptionPane.ERROR_MESSAGE
                );
                return;
            }

            // Crear el objeto Alumno
            Alumno nuevoAlumno = new Alumno(codigo, apellidos, nombres, domicilio);

            // Agregar a la lista correspondiente
            if (tipo.equals("Informático")) {
                listaInformaticos.add(nuevoAlumno);
                JOptionPane.showMessageDialog(this, "Informático agregado exitosamente.");
            } else {
                listaIndustriales.add(nuevoAlumno);
                JOptionPane.showMessageDialog(this, "Industrial agregado exitosamente.");
            }
        }
    }

    // Método para mostrar alumnos ordenados
    private void mostrarAlumnos(ArrayList<Alumno> lista, String titulo) {
        if (lista.isEmpty()) {
            JOptionPane.showMessageDialog(
                this,
                "No hay alumnos en la lista de " + titulo + ".",
                "Lista vacía",
                JOptionPane.INFORMATION_MESSAGE
```

```
            );
            areaMostrar.setText("");
            return;
        }

        // Ordenar la lista usando Comparable (Apellidos y luego Nombres)
        Collections.sort(lista);

        // Construir el texto a mostrar
        StringBuilder sb = new StringBuilder();
        sb.append("=== LISTA DE ").append(titulo).append(" ===\n\n");
        for (Alumno alumno : lista) {
            sb.append(alumno.toString()).append("\n");
        }

        areaMostrar.setText(sb.toString());
    }

    // Método main
    public static void main(String[] args) {
        // Ejecutar en el hilo de eventos de Swing
        SwingUtilities.invokeLater(() -> new AlumnoGUI());
    }
}
```

Composición

DatosPersonales.java

```
public class DatosPersonales {
    private String nombres;
    private String apellidos;
    private String domicilio;
    private String fechaNacimiento;

    public DatosPersonales(String nombres, String apellidos, String domicilio, String fechaNacimiento) {
        this.nombres = nombres;
        this.apellidos = apellidos;
        this.domicilio = domicilio;
        this.fechaNacimiento = fechaNacimiento;
    }

    // Getters y Setters
    public String getNombres() { return nombres; }
    public void setNombres(String nombres) { this.nombres = nombres; }

    public String getApellidos() { return apellidos; }
    public void setApellidos(String apellidos) { this.apellidos = apellidos; }

    public String getDomicilio() { return domicilio; }
    public void setDomicilio(String domicilio) { this.domicilio = domicilio; }

    public String getFechaNacimiento() { return fechaNacimiento; }
    public void setFechaNacimiento(String fechaNacimiento) { this.fechaNacimiento = fechaNacimiento; }

    // Método para obtener nombre completo
    public String getNombreCompleto() {
        return nombres + " " + apellidos;
    }

    @Override
    public String toString() {
        return "DATOS PERSONALES:\n" +
               "  Nombres: " + nombres + "\n" +
               "  Apellidos: " + apellidos + "\n" +
               "  Domicilio: " + domicilio + "\n" +
               "  F. Nacimiento: " + fechaNacimiento;
    }
}
```

DatosAcademicos.java

```
public class DatosAcademicos {
    private String codigo;
    private String carrera;
    private double promedio;
    private int ciclo;
    private String estado; // "Activo", "Inactivo", "Graduado"

    public DatosAcademicos(String codigo, String carrera, double promedio, int ciclo, String estado) {
        this.codigo = codigo;
        this.carrera = carrera;
        this.promedio = promedio;
        this.ciclo = ciclo;
        this.estado = estado;
    }

    // Getters y Setters
    public String getCodigo() { return codigo; }
    public void setCodigo(String codigo) { this.codigo = codigo; }

    public String getCarrera() { return carrera; }
    public void setCarrera(String carrera) { this.carrera = carrera; }

    public double getPromedio() { return promedio; }
    public void setPromedio(double promedio) { this.promedio = promedio; }

    public int getCiclo() { return ciclo; }
    public void setCiclo(int ciclo) { this.ciclo = ciclo; }

    public String getEstado() { return estado; }
    public void setEstado(String estado) { this.estado = estado; }

    // Métodos de negocio
    public boolean esAprobado() {
        return promedio >= 11.0;
    }

    public String obtenerCondicion() {
        if (promedio >= 15) return "Excelente";
        else if (promedio >= 12) return "Bueno";
        else if (promedio >= 11) return "Aprobado";
        else return "Desaprobado";
    }

    @Override
    public String toString() {
        return "DATOS ACADÉMICOS:\n" +
               "  Código: " + codigo + "\n" +
               "  Carrera: " + carrera + "\n" +
               "  Promedio: " + String.format("%.2f", promedio) + "\n" +
               "  Ciclo: " + ciclo + "\n" +
               "  Estado: " + estado + "\n" +
               "  Condición: " + obtenerCondicion();
    }
}
```

DatosContacto.java

```
public class DatosContacto {
    private String email;
    private String telefono;
    private String celular;
    private String redesSociales;

    public DatosContacto(String email, String telefono, String celular, String redesSociales) {
        this.email = email;
        this.telefono = telefono;
        this.celular = celular;
        this.redesSociales = redesSociales;
    }

    // Getters y Setters
    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }

    public String getTelefono() { return telefono; }
    public void setTelefono(String telefono) { this.telefono = telefono; }

    public String getCelular() { return celular; }
    public void setCelular(String celular) { this.celular = celular; }

    public String getRedesSociales() { return redesSociales; }
    public void setRedesSociales(String redesSociales) { this.redesSociales = redesSociales; }

    @Override
    public String toString() {
        return "DATOS DE CONTACTO:\n" +
               "  Email: " + email + "\n" +
               "  Teléfono: " + telefono + "\n" +
               "  Celular: " + celular + "\n" +
               "  Redes Sociales: " + redesSociales;
    }
}
```

Estudiante.java

```
public class Estudiante implements Comparable<Estudiante> {
    // COMPOSICIÓN: El estudiante está compuesto por 3 objetos
    private DatosPersonales datosPersonales;
    private DatosAcademicos datosAcademicos;
    private DatosContacto datosContacto;
    private String tipo; // "Informático" o "Industrial"

    // Constructor que recibe los tres componentes
    public Estudiante(DatosPersonales datosPersonales,
                      DatosAcademicos datosAcademicos,
                      DatosContacto datosContacto,
                      String tipo) {
        this.datosPersonales = datosPersonales;
        this.datosAcademicos = datosAcademicos;
        this.datosContacto = datosContacto;
        this.tipo = tipo;
    }

    // Getters para acceder a los componentes
    public DatosPersonales getDatosPersonales() { return datosPersonales; }
    public DatosAcademicos getDatosAcademicos() { return datosAcademicos; }
    public DatosContacto getDatosContacto() { return datosContacto; }
    public String getTipo() { return tipo; }
    public void setTipo(String tipo) { this.tipo = tipo; }

    // Métodos de conveniencia (delegación)
    public String getNombreCompleto() {
        return datosPersonales.getNombreCompleto();
    }

    public boolean estaAprobado() {
        return datosAcademicos.esAprobado();
    }

    public String getCodigo() {
        return datosAcademicos.getCodigo();
    }

    // Implementación de Comparable para ordenar
    @Override
    public int compareTo(Estudiante otro) {
        int compararApellidos = this.datosPersonales.getApellidos()
                                .compareTo(otro.datosPersonales.getApellidos());
        if (compararApellidos != 0) {
            return compararApellidos;
        }
        return this.datosPersonales.getNombres()
                 .compareTo(otro.datosPersonales.getNombres());
    }

    // Mostrar estado completo del estudiante
    public String mostrarEstadoCompleto() {
        StringBuilder sb = new StringBuilder();
        sb.append("------------------------------------------\n");
        sb.append("    ESTUDIANTE DE ").append(tipo.toUpperCase()).append("\n");
        sb.append("------------------------------------------\n\n");
        sb.append(datosPersonales.toString()).append("\n\n");
        sb.append(datosAcademicos.toString()).append("\n\n");
        sb.append(datosContacto.toString()).append("\n\n");
        sb.append("Resumen:\n");
        sb.append("  Nombre completo: ").append(getNombreCompleto()).append("\n");
        sb.append("  Estado académico: ").append(estaAprobado() ? "Aprobado" : "Desaprobado").append("\n");
        sb.append("  Condición: ").append(datosAcademicos.obtenerCondicion()).append("\n");
        sb.append("------------------------------------------");
        return sb.toString();
    }

    @Override
    public String toString() {
        return datosPersonales.getApellidos() + ", " +
               datosPersonales.getNombres() +
```

```
                " (" + datosAcademicos.getCodigo() + ") - " +
                datosAcademicos.getCarrera();
    }
}
```

```
import javax.swing.*;
import javax.swing.border.TitledBorder;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.util.ArrayList;
import java.util.Collections;

public class GestorEstudiantesGUI extends JFrame {
    // COMPOSICIÓN: La GUI contiene listas de estudiantes
    private ArrayList<Estudiante> listaInformaticos;
    private ArrayList<Estudiante> listaIndustriales;
    private Estudiante estudianteSeleccionado;

    // Componentes GUI
    private JTable tablaEstudiantes;
    private DefaultTableModel modeloTabla;
    private JTextArea areaDetalle;
    private JComboBox<String> comboFiltro;
    private JLabel lblContador;

    public GestorEstudiantesGUI() {
        // Inicializar listas
        listaInformaticos = new ArrayList<>();
        listaIndustriales = new ArrayList<>();
        estudianteSeleccionado = null;

        // Configurar ventana
        setTitle("Gestor de Estudiantes - Ejemplo de Composición");
        setSize(1000, 700);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout(10, 10));

        // Crear barra de menú
        crearMenu();

        // Panel principal dividido
        JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT);
        splitPane.setDividerLocation(350);

        // Panel superior: Tabla
        JPanel panelTabla = crearPanelTabla();
        splitPane.setTopComponent(panelTabla);

        // Panel inferior: Detalles y controles
        JPanel panelDetalle = crearPanelDetalle();
        splitPane.setBottomComponent(panelDetalle);

        add(splitPane, BorderLayout.CENTER);
        add(crearPanelEstadisticas(), BorderLayout.SOUTH);

        // Cargar datos de ejemplo
        cargarDatosEjemplo();
        actualizarTabla();

        setVisible(true);
    }

    private void crearMenu() {
        JMenuBar menuBar = new JMenuBar();

        JMenu menuEstudiante = new JMenu("Estudiante");

        JMenuItem itemIngresar = new JMenuItem("Ingresar Nuevo Estudiante");
        itemIngresar.addActionListener(e -> mostrarDialogoIngreso());

        JMenuItem itemEditar = new JMenuItem("Editar Estudiante Seleccionado");
        itemEditar.addActionListener(e -> editarEstudiante());

        JMenuItem itemEliminar = new JMenuItem("Eliminar Estudiante");
        itemEliminar.addActionListener(e -> eliminarEstudiante());

        menuEstudiante.add(itemIngresar);
```

```
        menuEstudiante.add(itemEditar);
        menuEstudiante.addSeparator();
        menuEstudiante.add(itemEliminar);

        JMenu menuVer = new JMenu("Ver");

        JMenuItem itemMostrarDetalle = new JMenuItem("Mostrar Detalle Completo");
        itemMostrarDetalle.addActionListener(e -> mostrarDetalleCompleto());

        JMenuItem itemOrdenar = new JMenuItem("Ordenar por Apellidos");
        itemOrdenar.addActionListener(e -> ordenarLista());

        menuVer.add(itemMostrarDetalle);
        menuVer.add(itemOrdenar);

        JMenu menuAyuda = new JMenu("Ayuda");
        JMenuItem itemAcerca = new JMenuItem("Acerca de Composición");
        itemAcerca.addActionListener(e -> mostrarAyuda());
        menuAyuda.add(itemAcerca);

        menuBar.add(menuEstudiante);
        menuBar.add(menuVer);
        menuBar.add(menuAyuda);
        setJMenuBar(menuBar);
    }

    private JPanel crearPanelTabla() {
        JPanel panel = new JPanel(new BorderLayout());
        panel.setBorder(new TitledBorder("Lista de Estudiantes"));

        // Panel de filtros
        JPanel panelFiltros = new JPanel(new FlowLayout(FlowLayout.LEFT));
        panelFiltros.add(new JLabel("Filtrar por:"));

        comboFiltro = new JComboBox<>(new String[]{"Todos", "Informáticos", "Industriales"});
        comboFiltro.addActionListener(e -> actualizarTabla());
        panelFiltros.add(comboFiltro);

        // Botón actualizar
        JButton btnActualizar = new JButton("Actualizar");
        btnActualizar.addActionListener(e -> actualizarTabla());
        panelFiltros.add(btnActualizar);

        panel.add(panelFiltros, BorderLayout.NORTH);

        // Tabla
        String[] columnas = {"Código", "Apellidos", "Nombres", "Carrera", "Promedio", "Estado"};
        modeloTabla = new DefaultTableModel(columnas, 0) {
            @Override
            public boolean isCellEditable(int row, int column) {
                return false;
            }
        };

        tablaEstudiantes = new JTable(modeloTabla);
        tablaEstudiantes.setRowHeight(25);
        tablaEstudiantes.getSelectionModel().addListSelectionListener(e -> {
            if (!e.getValueIsAdjusting()) {
                mostrarDetalleSeleccionado();
            }
        });

        JScrollPane scroll = new JScrollPane(tablaEstudiantes);
        panel.add(scroll, BorderLayout.CENTER);

        return panel;
    }

    private JPanel crearPanelDetalle() {
        JPanel panel = new JPanel(new BorderLayout());
        panel.setBorder(new TitledBorder("Detalles del Estudiante"));

        areaDetalle = new JTextArea();
```

```
        areaDetalle.setEditable(false);
        areaDetalle.setFont(new Font("Monospaced", Font.PLAIN, 12));
        areaDetalle.setBackground(new Color(240, 248, 255));
        JScrollPane scroll = new JScrollPane(areaDetalle);
        panel.add(scroll, BorderLayout.CENTER);

        // Panel de botones de acción rápida
        JPanel panelBotones = new JPanel(new FlowLayout(FlowLayout.CENTER));

        JButton btnVerCompleto = new JButton("Ver Estado Completo");
        btnVerCompleto.addActionListener(e -> mostrarDetalleCompleto());

        JButton btnCambiarEstado = new JButton("Cambiar Estado Académico");
        btnCambiarEstado.addActionListener(e -> cambiarEstadoAcademico());

        panelBotones.add(btnVerCompleto);
        panelBotones.add(btnCambiarEstado);
        panel.add(panelBotones, BorderLayout.SOUTH);

        return panel;
    }

    private JPanel crearPanelEstadisticas() {
        JPanel panel = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 5));
        panel.setBorder(new TitledBorder("Estadísticas"));
        panel.setBackground(new Color(240, 240, 240));

        lblContador = new JLabel();
        actualizarEstadisticas();
        panel.add(lblContador);

        JButton btnExportar = new JButton("Exportar Resumen");
        btnExportar.addActionListener(e -> exportarResumen());
        panel.add(btnExportar);

        return panel;
    }

    private void mostrarDialogoIngreso() {
        // Creamos un panel con tabs para organizar los datos
        JTabbedPane tabbedPane = new JTabbedPane();

        // Panel 1: Datos Personales
        JPanel panelPersonal = new JPanel(new GridLayout(5, 2, 5, 5));
        JTextField txtNombres = new JTextField();
        JTextField txtApellidos = new JTextField();
        JTextField txtDomicilio = new JTextField();
        JTextField txtFechaNac = new JTextField();

        panelPersonal.add(new JLabel("Nombres:"));
        panelPersonal.add(txtNombres);
        panelPersonal.add(new JLabel("Apellidos:"));
        panelPersonal.add(txtApellidos);
        panelPersonal.add(new JLabel("Domicilio:"));
        panelPersonal.add(txtDomicilio);
        panelPersonal.add(new JLabel("Fecha Nacimiento:"));
        panelPersonal.add(txtFechaNac);

        // Panel 2: Datos Académicos
        JPanel panelAcademico = new JPanel(new GridLayout(6, 2, 5, 5));
        JTextField txtCodigo = new JTextField();
        JComboBox<String> cmbCarrera = new JComboBox<>(new String[]{"Ingeniería Informática", "Ingeniería Industrial"});
        JTextField txtPromedio = new JTextField();
        JTextField txtCiclo = new JTextField();
        JComboBox<String> cmbEstado = new JComboBox<>(new String[]{"Activo", "Inactivo", "Graduado"});

        panelAcademico.add(new JLabel("Código:"));
        panelAcademico.add(txtCodigo);
        panelAcademico.add(new JLabel("Carrera:"));
        panelAcademico.add(cmbCarrera);
        panelAcademico.add(new JLabel("Promedio:"));
        panelAcademico.add(txtPromedio);
        panelAcademico.add(new JLabel("Ciclo:"));
```

```
panelAcademico.add(txtCiclo);
panelAcademico.add(new JLabel("Estado:"));
panelAcademico.add(cmbEstado);

// Panel 3: Datos de Contacto
JPanel panelContacto = new JPanel(new GridLayout(4, 2, 5, 5));
JTextField txtEmail = new JTextField();
JTextField txtTelefono = new JTextField();
JTextField txtCelular = new JTextField();
JTextField txtRedes = new JTextField();

panelContacto.add(new JLabel("Email:"));
panelContacto.add(txtEmail);
panelContacto.add(new JLabel("Teléfono:"));
panelContacto.add(txtTelefono);
panelContacto.add(new JLabel("Celular:"));
panelContacto.add(txtCelular);
panelContacto.add(new JLabel("Redes Sociales:"));
panelContacto.add(txtRedes);

tabbedPane.addTab("Personal", panelPersonal);
tabbedPane.addTab("Académico", panelAcademico);
tabbedPane.addTab("Contacto", panelContacto);

int resultado = JOptionPane.showConfirmDialog(
    this,
    tabbedPane,
    "Ingresar Nuevo Estudiante",
    JOptionPane.OK_CANCEL_OPTION,
    JOptionPane.PLAIN_MESSAGE
);

if (resultado == JOptionPane.OK_OPTION) {
    // Validar campos obligatorios
    if (txtNombres.getText().trim().isEmpty() ||
        txtApellidos.getText().trim().isEmpty() ||
        txtCodigo.getText().trim().isEmpty()) {
        JOptionPane.showMessageDialog(this,
            "Nombres, Apellidos y Código son obligatorios.",
            "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    try {
        // Crear los tres objetos componentes
        DatosPersonales dp = new DatosPersonales(
            txtNombres.getText().trim(),
            txtApellidos.getText().trim(),
            txtDomicilio.getText().trim(),
            txtFechaNac.getText().trim()
        );

        DatosAcademicos da = new DatosAcademicos(
            txtCodigo.getText().trim(),
            (String) cmbCarrera.getSelectedItem(),
            Double.parseDouble(txtPromedio.getText().trim()),
            Integer.parseInt(txtCiclo.getText().trim()),
            (String) cmbEstado.getSelectedItem()
        );

        DatosContacto dc = new DatosContacto(
            txtEmail.getText().trim(),
            txtTelefono.getText().trim(),
            txtCelular.getText().trim(),
            txtRedes.getText().trim()
        );

        // Crear el estudiante usando COMPOSICIÓN
        String tipo = ((String) cmbCarrera.getSelectedItem()).contains("Informática")
                    ? "Informático" : "Industrial";

        Estudiante estudiante = new Estudiante(dp, da, dc, tipo);
```

```
            // Agregar a la lista correspondiente
            if (tipo.equals("Informático")) {
                listaInformaticos.add(estudiante);
            } else {
                listaIndustriales.add(estudiante);
            }

            actualizarTabla();
            JOptionPane.showMessageDialog(this,
                "Estudiante agregado exitosamente.",
                "Éxito", JOptionPane.INFORMATION_MESSAGE);

        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this,
                "Error en el formato de números (Promedio o Ciclo).",
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}

private void actualizarTabla() {
    modeloTabla.setRowCount(0);

    String filtro = (String) comboFiltro.getSelectedItem();
    ArrayList<Estudiante> listaMostrar = new ArrayList<>();

    if (filtro.equals("Todos")) {
        listaMostrar.addAll(listaInformaticos);
        listaMostrar.addAll(listaIndustriales);
    } else if (filtro.equals("Informáticos")) {
        listaMostrar.addAll(listaInformaticos);
    } else if (filtro.equals("Industriales")) {
        listaMostrar.addAll(listaIndustriales);
    }

    // Ordenar por apellidos
    Collections.sort(listaMostrar);

    for (Estudiante e : listaMostrar) {
        Object[] fila = {
            e.getCodigo(),
            e.getDatosPersonales().getApellidos(),
            e.getDatosPersonales().getNombres(),
            e.getDatosAcademicos().getCarrera(),
            String.format("%.2f", e.getDatosAcademicos().getPromedio()),
            e.estaAprobado() ? "Aprobado" : "Desaprobado"
        };
        modeloTabla.addRow(fila);
    }

    actualizarEstadisticas();
}

private void mostrarDetalleSeleccionado() {
    int fila = tablaEstudiantes.getSelectedRow();
    if (fila >= 0) {
        String filtro = (String) comboFiltro.getSelectedItem();
        ArrayList<Estudiante> listaMostrar = new ArrayList<>();

        if (filtro.equals("Todos")) {
            listaMostrar.addAll(listaInformaticos);
            listaMostrar.addAll(listaIndustriales);
        } else if (filtro.equals("Informáticos")) {
            listaMostrar.addAll(listaInformaticos);
        } else if (filtro.equals("Industriales")) {
            listaMostrar.addAll(listaIndustriales);
        }

        Collections.sort(listaMostrar);

        if (fila < listaMostrar.size()) {
            estudianteSeleccionado = listaMostrar.get(fila);
            areaDetalle.setText(estudianteSeleccionado.mostrarEstadoCompleto());
```

```
            }
        }
    }

    private void mostrarDetalleCompleto() {
        if (estudianteSeleccionado == null) {
            JOptionPane.showMessageDialog(this,
                "Por favor, seleccione un estudiante de la tabla.",
                "Sin selección", JOptionPane.WARNING_MESSAGE);
            return;
        }

        JOptionPane.showMessageDialog(this,
            estudianteSeleccionado.mostrarEstadoCompleto(),
            "Detalle Completo del Estudiante",
            JOptionPane.INFORMATION_MESSAGE);
    }

    private void editarEstudiante() {
        if (estudianteSeleccionado == null) {
            JOptionPane.showMessageDialog(this,
                "Por favor, seleccione un estudiante de la tabla.",
                "Sin selección", JOptionPane.WARNING_MESSAGE);
            return;
        }
        // Aquí iría la lógica para editar (similar al ingreso)
        JOptionPane.showMessageDialog(this,
            "Función de edición en desarrollo.\n" +
            "Estudiante seleccionado: " + estudianteSeleccionado.getNombreCompleto(),
            "Editar", JOptionPane.INFORMATION_MESSAGE);
    }

    private void eliminarEstudiante() {
        if (estudianteSeleccionado == null) {
            JOptionPane.showMessageDialog(this,
                "Por favor, seleccione un estudiante de la tabla.",
                "Sin selección", JOptionPane.WARNING_MESSAGE);
            return;
        }

        int confirmar = JOptionPane.showConfirmDialog(this,
            "¿Está seguro de eliminar a:\n" +
            estudianteSeleccionado.getNombreCompleto() + "\n" +
            "Código: " + estudianteSeleccionado.getCodigo(),
            "Confirmar eliminación",
            JOptionPane.YES_NO_OPTION);

        if (confirmar == JOptionPane.YES_OPTION) {
            if (estudianteSeleccionado.getTipo().equals("Informático")) {
                listaInformaticos.remove(estudianteSeleccionado);
            } else {
                listaIndustriales.remove(estudianteSeleccionado);
            }
            estudianteSeleccionado = null;
            areaDetalle.setText("");
            actualizarTabla();
            JOptionPane.showMessageDialog(this,
                "Estudiante eliminado exitosamente.",
                "Éxito", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void cambiarEstadoAcademico() {
        if (estudianteSeleccionado == null) {
            JOptionPane.showMessageDialog(this,
                "Por favor, seleccione un estudiante de la tabla.",
                "Sin selección", JOptionPane.WARNING_MESSAGE);
            return;
        }

        String[] opciones = {"Activo", "Inactivo", "Graduado"};
        String nuevoEstado = (String) JOptionPane.showInputDialog(
            this,
```

```
            "Seleccione el nuevo estado para:\n" +
            estudianteSeleccionado.getNombreCompleto(),
            "Cambiar Estado Académico",
            JOptionPane.QUESTION_MESSAGE,
            null,
            opciones,
            estudianteSeleccionado.getDatosAcademicos().getEstado()
        );

        if (nuevoEstado != null) {
            estudianteSeleccionado.getDatosAcademicos().setEstado(nuevoEstado);
            actualizarTabla();
            mostrarDetalleSeleccionado();
            JOptionPane.showMessageDialog(this,
                "Estado actualizado a: " + nuevoEstado,
                "Éxito", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void ordenarLista() {
        Collections.sort(listaInformaticos);
        Collections.sort(listaIndustriales);
        actualizarTabla();
        JOptionPane.showMessageDialog(this,
            "Listas ordenadas por apellidos y nombres.",
            "Ordenado", JOptionPane.INFORMATION_MESSAGE);
    }

    private void actualizarEstadisticas() {
        int total = listaInformaticos.size() + listaIndustriales.size();
        int aprobados = 0;

        for (Estudiante e : listaInformaticos) {
            if (e.estaAprobado()) aprobados++;
        }
        for (Estudiante e : listaIndustriales) {
            if (e.estaAprobado()) aprobados++;
        }

        lblContador.setText(String.format(
            "Total: %d estudiantes | Aprobados: %d | Desaprobados: %d | " +
            "Informáticos: %d | Industriales: %d",
            total, aprobados, total - aprobados,
            listaInformaticos.size(), listaIndustriales.size()
        ));
    }

    private void exportarResumen() {
        StringBuilder sb = new StringBuilder();
        sb.append("=== RESUMEN DE ESTUDIANTES ===\n\n");

        sb.append("INFORMÁTICOS (").append(listaInformaticos.size()).append("):\n");
        Collections.sort(listaInformaticos);
        for (Estudiante e : listaInformaticos) {
            sb.append("  • ").append(e.toString()).append("\n");
        }

        sb.append("\nINDUSTRIALES (").append(listaIndustriales.size()).append("):\n");
        Collections.sort(listaIndustriales);
        for (Estudiante e : listaIndustriales) {
            sb.append("  • ").append(e.toString()).append("\n");
        }

        JTextArea textArea = new JTextArea(sb.toString());
        textArea.setEditable(false);
        textArea.setFont(new Font("Monospaced", Font.PLAIN, 12));

        JOptionPane.showMessageDialog(this,
            new JScrollPane(textArea),
            "Resumen de Estudiantes",
            JOptionPane.INFORMATION_MESSAGE);
    }
```

```
    private void mostrarAyuda() {
        String ayuda = """
            SISTEMA DE GESTIÓN DE ESTUDIANTES
            ---------------------------------

            COMPOSICIÓN EN ACCIÓN:

            Cada ESTUDIANTE está compuesto por:
            1. DatosPersonales (nombre, apellidos, domicilio, fecha nac.)
            2. DatosAcademicos (código, carrera, promedio, ciclo, estado)
            3. DatosContacto (email, teléfono, celular, redes sociales)

            PRINCIPIOS DEMOSTRADOS:
            • Composición: Estudiante TIENE UN objeto de cada tipo
            • Delegación: Métodos que usan los componentes
            • Encapsulamiento: Cada componente maneja sus datos
            • Reutilización: Los componentes pueden usarse en otros contextos

            FUNCIONALIDADES:
            • Ingresar estudiantes con todos sus datos
            • Ver listado con filtros por carrera
            • Ver detalle completo de cada estudiante
            • Ordenar por apellidos y nombres
            • Cambiar estado académico
            • Exportar resumen
            """;

        JOptionPane.showMessageDialog(this,
            ayuda,
            "Ayuda - Composición con Estudiantes",
            JOptionPane.INFORMATION_MESSAGE);
    }

    private void cargarDatosEjemplo() {
        // Crear algunos estudiantes de ejemplo usando COMPOSICIÓN
        DatosPersonales dp1 = new DatosPersonales("Ana", "García Pérez", "Calle 123", "15/05/2000");
        DatosAcademicos da1 = new DatosAcademicos("2024001", "Ingeniería Informática", 15.5, 5, "Activo");
        DatosContacto dc1 = new DatosContacto("ana@email.com", "555-1234", "999-888-777", "@anagarcia");
        listaInformaticos.add(new Estudiante(dp1, da1, dc1, "Informático"));

        DatosPersonales dp2 = new DatosPersonales("Carlos", "López Mendoza", "Av. Principal 456", "20/08/1999");
        DatosAcademicos da2 = new DatosAcademicos("2024002", "Ingeniería Informática", 12.0, 6, "Activo");
        DatosContacto dc2 = new DatosContacto("carlos@email.com", "555-5678", "999-666-555", "@carloslopez");
        listaInformaticos.add(new Estudiante(dp2, da2, dc2, "Informático"));

        DatosPersonales dp3 = new DatosPersonales("María", "Rodríguez Castro", "Calle 789", "10/12/2001");
        DatosAcademicos da3 = new DatosAcademicos("2024003", "Ingeniería Industrial", 8.5, 3, "Activo");
        DatosContacto dc3 = new DatosContacto("maria@email.com", "555-9012", "999-444-333", "@mariarod");
        listaIndustriales.add(new Estudiante(dp3, da3, dc3, "Industrial"));

        DatosPersonales dp4 = new DatosPersonales("Juan", "Martínez Sánchez", "Av. Secundaria 321", "25/03/2000");
        DatosAcademicos da4 = new DatosAcademicos("2024004", "Ingeniería Industrial", 14.0, 7, "Activo");
        DatosContacto dc4 = new DatosContacto("juan@email.com", "555-3456", "999-222-111", "@juanmartinez");
        listaIndustriales.add(new Estudiante(dp4, da4, dc4, "Industrial"));
    }

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