INSTITUTO TECNOLOGICO DE NUEVO LEÓN

INGENIERÍA EN SISTEMAS COMPUTACIONALES

(Unidad 2 Temas y Librerías)

Proyecto Unidad 2

Nombre: Angel Fernando Lopez Serratos

Matrícula:23480843

Nombre: César Emilio Oliva Vázquez

Matricula:23480864

Nombre: Omar Alejandro Arriaga Reyna

Matricula; 23480854

Nombre: Alan Alfonso Coronado Martinez

Matricula: 23480777

Nombre: Julio Andres Herrera Martinez

Matricula: 22480923

10 de Marzo de 2025

Introducción

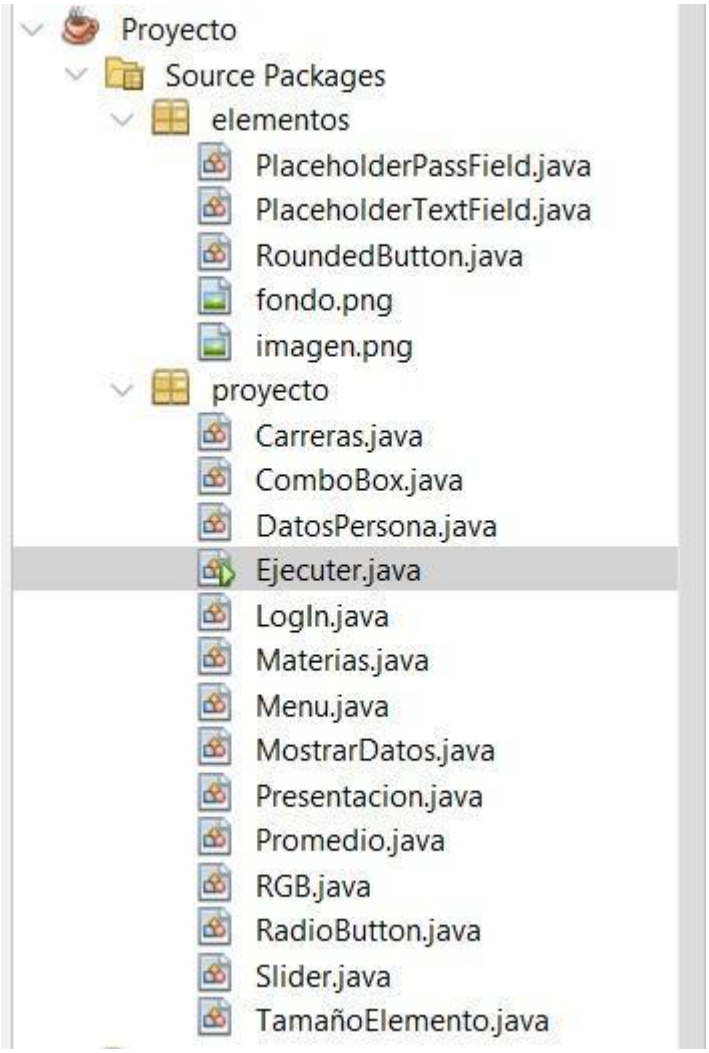
El presente proyecto tiene como objetivo desarrollar una interfaz gráfica que integre y gestione diversas interfaces dentro de un mismo entorno, optimizando la interacción del usuario con el sistema. Como parte de la materia de Tópicos de Programación, este trabajo explora la implementación de componentes gráficos como Combo Box, Radio Button, Sliders, y otros elementos visuales para mejorar la experiencia de uso. A través de esta práctica, se busca fortalecer el conocimiento en el desarrollo de interfaces gráficas y su aplicación en sistemas interactivos.

El desarrollo de interfaces gráficas es un aspecto fundamental en la ingeniería de software, ya que permite la creación de entornos más accesibles e intuitivos para los usuarios

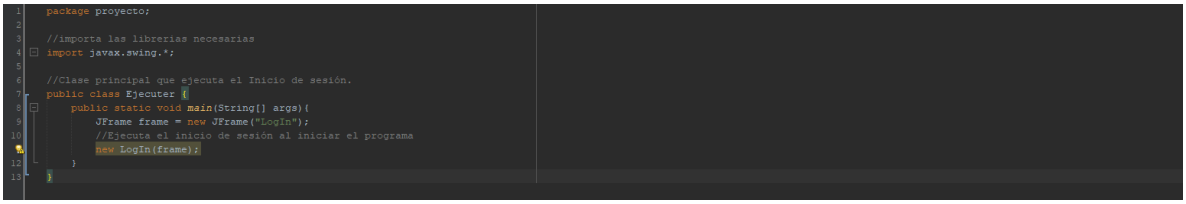
A través del desarrollo de este proyecto, los participantes podrán reforzar sus conocimientos en programación orientada a eventos, manipulación de interfaces gráficas y diseño de experiencias de usuario. Asimismo, se fomentará el aprendizaje sobre buenas prácticas en la creación de interfaces, garantizando que los componentes sean accesibles, intuitivos y funcionales.

Códigos

Estructura



Código Ejecuter



```java
package proyecto;

//importa las librerias necesarias
import javax.swing.*;

//Clase principal que ejecuta el Inicio de sesión.
public class Ejecuter {
    public static void main(String[] args){
        JFrame frame = new JFrame("LogIn");
        //Ejecuta el inicio de sesión al iniciar el programa
        new LogIn(frame);
    }
}
```

Código con la clase main que llama al inicio de sesión

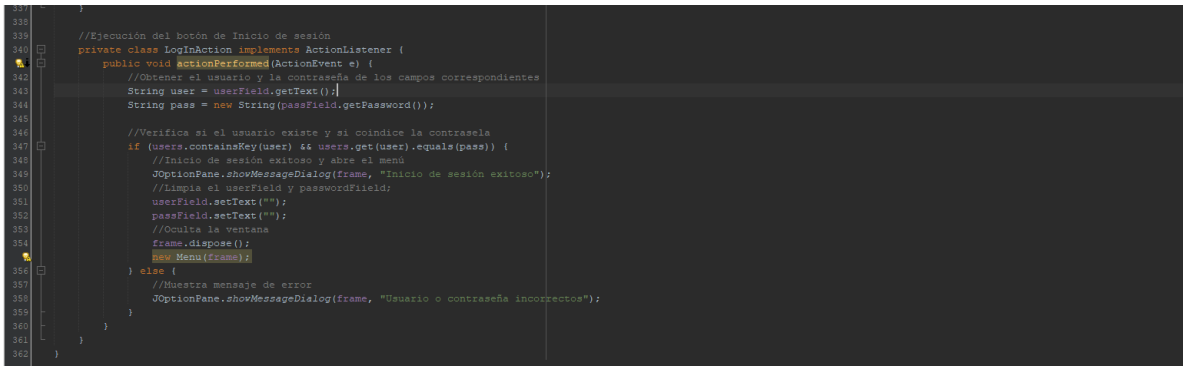
Código Login

```java
package proyecto;

//Se importan las librerias a usar
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.HashMap;

//"librerias" personalizadas utilizadas
import elementos.RoundedButton;
import elementos.PlaceholderTextField;
import elementos.PlaceholderPassField;

//La clase LogIn heredada de un JPanel. Panel personalizado
//Muestra el Inicio de sesión o registro
public class LogIn extends JPanel{
    //Se declara fuera por que se usa en otros metodo fuera del main
    private HashMap<String, String> users = new HashMap<>();
    private JFrame frame;
    private JPanel logIn, register, descLogIn, descRegister;
    private JPanel panelLogIn, panelRegister;
    private JTextField userField;
    private JPasswordField passField;
    private static LogIn instancia;

    //Declarando un color que se usará en el diseño
    public Color azul = new Color(100, 149, 237);

    public LogIn(JFrame frame){
        //Cambia el diseño de los OptionPane
        UIManager.put("OptionPane.messageFont", new Font("Inter", Font.PLAIN, 12));

        //Configuración de la ventana
        frame = new JFrame("Iniciar Sesión o Registrarse");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 520);
        frame.setResizable(false); //Bloquea la opción de maximizar
        frame.setLayout(new BorderLayout());

        this.frame = frame;
        instancia = this;

        //Centra la ventana al iniciar
        frame.setLocationRelativeTo(null);

        //Mostrar Inicialmente el panel de Inicio de Sesión
        frame.add(MostrarLogIn());
        frame.setVisible(true);

        //Enfoca la ventana al iniciar para evitar que enfoque otros elementos
        SwingUtilities.invokeLater(() -> {
            this.frame.requestFocusInWindow();
        });

        agregarListener(frame);
    }

    //Listener del focus
    public void agregarListener(JFrame frame) {
        frame.addMouseListener(new java.awt.event.MouseAdapter() {
            @Override
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                frame.requestFocusInWindow(); // Quita el foco del JTextField
            }
        });
    }


    //Metodo que muestra los paneles de la sección de Iniciar Sesión
    private JPanel MostrarLogIn(){
        //Panel que contendrá los dos paneles
        panelLogIn = new JPanel(new BorderLayout());

        //Panel que muestra los elementos de inicio de sesión
        logIn = new JPanel();
        logIn.setLayout(null);
        logIn.setPreferredSize(new Dimension(400, 520));
        logIn.setBackground(Color.WHITE);

        //Elementos del panel de Inicio de sesión
        JLabel textLogIn = new JLabel("Iniciar Sesión");
        textLogIn.setBounds(110, 120, 300, 30);
        textLogIn.setFont(new Font("Inter", Font.PLAIN, 30));

        JLabel textUse = new JLabel("Ingrese su usuario y contraseña");
        textUse.setBounds(97, 160, 250, 30);
        textUse.setFont(new Font("Inter", Font.PLAIN, 14));

        userField = new PlaceholderTextField("Usuario", 20);
        userField.setBounds(100, 210, 200, 30);
        userField.setFont(new Font("Inter", Font.PLAIN, 12));

        passField = new PlaceholderPassField("Contraseña", 20);
        passField.setBounds(100, 250, 200, 30);
        passField.setFont(new Font("Inter", Font.PLAIN, 12));
```

```
        registrarse.setFont(new Font("Inter", Font.PLAIN, 14));
        registrarse.setBounds(105, 250, 250, 30);
        registrarse.setForeground(Color.WHITE);

        //Creación del botón redondeado
        JButton btnCR = new RoundedButton("Registrarse");
        btnCR.setBounds(125, 280, 150, 30);
        btnCR.setBackground(Color.WHITE);
        btnCR.setForeground(azul);
        btnCR.setFont(new Font("Inter", Font.PLAIN, 14));


        //Funcionalidad del boton cambiar a registrarse
        btnCR.addActionListener((ActionEvent e)->{
            CambiarVista(MostrarRegister());
        });

        //Agregar elementos al panel de instrucciones
        descLogIn.add(titulo);
        descLogIn.add(texto1);
        descLogIn.add(texto2);
        descLogIn.add(registrarse);
        descLogIn.add(btnCR);
        descLogIn.add(fondo);

        //Se agregan los paneles al panel principal
        panelLogIn.add(logIn, BorderLayout.WEST);
        panelLogIn.add(descLogIn, BorderLayout.EAST);

        //Regresa el panel con todo el contenido
        return panelLogIn;
    }

    //Metodo que retorna los paneles de la sección de Iniciar Sesión
    private JPanel MostrarRegister(){
        //Panel que contendrá los dos paneles
        panelRegister = new JPanel(new BorderLayout());

        //Panel que muestra el panel de instrucciones
        descRegister = new JPanel();
        descRegister.setLayout(null);
        descRegister.setPreferredSize(new Dimension(400, 520));
        descRegister.setBackground(azul);

        //Imagen de fondo
        JLabel fondo = new JLabel(new ImageIcon(getClass().getResource("/elementos/imagen.png")));
        fondo.setBounds(0, 0, 400, 520); // Ajusta al tamaño de la ventana

        //Diseño y acomodo de los elementos
        //Diseño y acomodo de los elementos
        JLabel titulo = new JLabel("¡Hola!");
        titulo.setFont(new Font("Inter", Font.PLAIN, 30));
        titulo.setBounds(160, 140, 150, 30);
        titulo.setForeground(Color.WHITE);

        JLabel texto1 = new JLabel("Registrese con sus datos para usar");
        JLabel texto2 = new JLabel("todas las funciones del programa");

        texto1.setFont(new Font("Inter", Font.PLAIN, 14));
        texto1.setBounds(85, 140, 300, 100);
        texto1.setForeground(Color.WHITE);

        texto2.setFont(new Font("Inter", Font.PLAIN, 14));
        texto2.setBounds(95, 155, 300, 100);
        texto2.setForeground(Color.WHITE);

        JLabel ingresar = new JLabel("¿Ya tienes una cuenta? Intente");
        ingresar.setFont(new Font("Inter", Font.PLAIN, 14));
        ingresar.setBounds(100, 250, 250, 30);
        ingresar.setForeground(Color.WHITE);

        JButton btnCL = new RoundedButton("Iniciar Sesión");
        btnCL.setBounds(125, 280, 150, 30);
        btnCL.setBackground(Color.WHITE);
        btnCL.setForeground(azul);
        btnCL.setFont(new Font("Inter", Font.PLAIN, 14));

        //Funcionalidad del boton cambiar a logIn
        btnCL.addActionListener((ActionEvent e)->{
            CambiarVista(MostrarLogIn());
        });

        //Agregar elementos al panel de instrucciones
        descRegister.add(titulo);
        descRegister.add(texto1);
        descRegister.add(texto2);
        descRegister.add(ingresar);
        descRegister.add(btnCL);
        descRegister.add(fondo);



        //Panel que muestra los elementos de registrarse
        register = new JPanel();
        register.setLayout(null);
        register.setPreferredSize(new Dimension(400, 520));
        register.setBackground(Color.WHITE);

```

```
        //Elementos del panel de Inicio de sesión
        JLabel textRegister = new JLabel("Registrarse");
        textRegister.setBounds(120, 120, 300, 40);
        textRegister.setFont(new Font("Inter", Font.PLAIN, 30));

        JLabel textUse = new JLabel("Cree un usuario y contraseña");
        textUse.setBounds(100, 160, 250, 30);
        textUse.setFont(new Font("Inter", Font.PLAIN, 14));
        textRegister.setFont(new Font("Inter", Font.PLAIN, 30));

        userField = new PlaceholderTextField("Usuario", 20);
        userField.setBounds(100, 210, 200, 30);
        userField.setFont(new Font("Inter", Font.PLAIN, 12));

        passField = new PlaceholderPassField("Contraseña", 20);
        passField.setBounds(100, 250, 200, 30);
        passField.setFont(new Font("Inter", Font.PLAIN, 12));

        //Creación del botón redondeado
        //Creación del botón redondeado
        JButton registerButton = new RoundedButton("Registrarse");
        registerButton.setBounds(125, 300, 150, 30);

        registerButton.setBackground(new Color(100, 149, 237));
        registerButton.setForeground(Color.WHITE);
        registerButton.setFocusPainted(false);
        registerButton.setFont(new Font("Inter", Font.PLAIN, 14));


         //Funcionalidad del botón. Registra
        registerButton.addActionListener(new RegisterAction());


        //Agregar elementos al panel de registro
        register.add(textRegister);
        register.add(textUse);
        register.add(userField);
        register.add(passField);
        register.add(registerButton);


        //Se agregan los paneles al panel principal
        panelRegister.add(descRegister, BorderLayout.WEST);
        panelRegister.add(register, BorderLayout.EAST);

        //Regresa el panel con todo el contenido
        return panelRegister;
    }
```

```
    }

    //Cambia la vista del panel
    private void CambiarVista(JPanel panel){
        //Elimina el contenido del panel de contenido
        frame.getContentPane().removeAll();
        //Agrega el nuevo panel
        frame.add(panel);
        frame.revalidate();
        frame.repaint();
        //Enfoca la ventana al iniciar para evitar que enfoque otros elementos
        SwingUtilities.invokeLater(() -> {
            frame.requestFocusInWindow();
        });

    }

    //Ejecucción del botón registro
    private class RegisterAction implements ActionListener{
        public void actionPerformed(ActionEvent e) {
            //Obtener el usuario y la contraseña de los campos correspondientes
            String user = userField.getText();
            String pass = new String(passField.getPassword());

            //Verifica si el usuario o la contraseña están vacios
            if (user.isEmpty() || pass.isEmpty()) {
                //Muestra mensaje de error con un panel personalizado
                JOptionPane.showMessageDialog(frame, "Usuario y contraseña no pueden estar vacios");
                //Si el usuario ya existe
            } else if (users.containsKey(user)) {
                JOptionPane.showMessageDialog(frame, "El usuario ya existe");
                //Si no existe el usuario y los datos están completos
            } else {
                users.put(user, pass);
                JOptionPane.showMessageDialog(frame, "Usuario registrado exitosamente");
               CambiarVista(MostrarLogIn());
            }

        }
    }

    //Hace visible de nuevo la ventana de logIn
    public void mostrarVentana(){
        frame.setVisible(true);
    }

    public static LogIn getInstancia(){
        return instancia;
    }
```



```
    }

    //Ejecución del botón de Inicio de sesión
    private class LogInAction implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            //Obtener el usuario y la contraseña de los campos correspondientes
            String user = userField.getText();
            String pass = new String(passField.getPassword());

            //Verifica si el usuario existe y si coindice la contrasela
            if (users.containsKey(user) && users.get(user).equals(pass)) {
                //Inicio de sesión exitoso y abre el menú
                JOptionPane.showMessageDialog(frame, "Inicio de sesión exitoso");
                //Limpia el userField y passwordFiield;
                userField.setText("");
                passField.setText("");
                //Oculta la ventana
                frame.dispose();
                new Menu(frame);
            } else {
                //Muestra mensaje de error
                JOptionPane.showMessageDialog(frame, "Usuario o contraseña incorrectos");
            }
        }
    }
}
```

1. Estructura General

- **Clase:** LogIn es una clase pública que hereda de JPanel, lo que la convierte en un panel personalizado.
- **Propósito:** Crear una interfaz con dos secciones: "Iniciar Sesión" y "Registrarse", permitiendo a los usuarios alternar entre ellas.
- **Librerías**:
    - javax.swing.* y java.awt.*: Para componentes gráficos y diseño.
    - HashMap: Para almacenar usuarios y contraseñas en memoria (clave: usuario, valor: contraseña).
    - Clases personalizadas (RoundedButton, PlaceholderTextField, PlaceholderPassField): Botones redondeados y campos de texto con placeholders.
- **Atributos principales:**
    - users: Un HashMap que guarda los usuarios registrados.
    - frame: El marco principal (JFrame) de la ventana.
    - logIn, register, descLogIn, descRegister: Paneles para las vistas de inicio de sesión y registro.
    - userField y passField: Campos de texto para ingresar usuario y contraseña.
    - instancia: Variable estática para implementar el patrón Singleton (una sola instancia de LogIn).

---

2. Constructor (LogIn(JFrame frame))

- **Función**: Configura la ventana principal y muestra el panel de inicio de sesión por defecto.
- **Detalles**:
    - Define el título de la ventana como "Iniciar Sesión o Registrarse".
    - Establece un tamaño fijo (800x520 píxeles) y desactiva la opción de redimensionar.
    - Usa BorderLayout como diseño principal.
    - Centra la ventana en la pantalla con setLocationRelativeTo(null).
    - Llama a MostrarLogIn() para mostrar el panel de inicio de sesión y hace visible la ventana.
    - Agrega un listener para mantener el foco en la ventana y evitar que se enfoque en los campos de texto al iniciar.

---

3. Método MostrarLogIn()

- **Función**: Crea y devuelve un panel con la interfaz de inicio de sesión.
- **Estructura**:
    - panelLogIn: Panel principal con diseño BorderLayout que contiene dos subpaneles:
        - logIn: Panel izquierdo (blanco) con los elementos para iniciar sesión.
        - descLogIn: Panel derecho (color azul) con información y un botón para cambiar a "Registrarse".
    - Elementos de logIn:
        - Etiquetas: "Iniciar Sesión" y "Ingrese su usuario y contraseña".
        - userField: Campo de texto con placeholder "Usuario".
        - passField: Campo de contraseña con placeholder "Contraseña".
        - Botón "Iniciar Sesión" (RoundedButton): Color azul, inicia la acción de login (clase LogInAction).
    - Elementos de descLogIn:
        - Fondo con imagen (imagen.png).
        - Texto de bienvenida ("¡Bienvenido!", instrucciones).
        - Botón "Registrarse": Cambia la vista a MostrarRegister() usando CambiarVista().

---

4. Método MostrarRegister()

- **Función**: Crea y devuelve un panel con la interfaz de registro.
- **Estructura**:
  - panelRegister: Panel principal con diseño BorderLayout que contiene dos subpaneles:
    - descRegister: Panel izquierdo (azul) con información y un botón para volver a "Iniciar Sesión".
    - register: Panel derecho (blanco) con los elementos para registrarse.
  - Elementos de register:
    - Etiquetas: "Registrarse" y "Cree un usuario y contraseña".
    - userField y passField: Campos reutilizados para usuario y contraseña.
    - Botón "Registrarse" (RoundedButton): Color azul, registra al usuario (clase RegisterAction).
  - Elementos de descRegister:
    - Fondo con imagen (imagen.png).
    - Texto de bienvenida ("¡Hola!", instrucciones).
    - Botón "Iniciar Sesión": Cambia la vista a MostrarLogIn().

---

5. Método CambiarVista(JPanel panel)

- **Función:** Cambia dinámicamente el contenido de la ventana entre las vistas de inicio de sesión y registro.
- **Detalles**:
  - Elimina el contenido actual del frame con removeAll().
  - Agrega el nuevo panel (panelLogIn o panelRegister).
  - Actualiza la interfaz con revalidate() y repaint().
  - Asegura que el foco esté en la ventana.

---

6. Clase RegisterAction

- **Función**: Maneja la lógica del botón "Registrarse".
- **Detalles**:
  - Obtiene el usuario y contraseña de los campos userField y passField.
  - Validaciones:
    - Si alguno está vacío: Muestra un mensaje de error.
    - Si el usuario ya existe en users: Muestra "El usuario ya existe".
    - Si todo es válido: Agrega el par usuario-contraseña al HashMap y cambia a la vista de inicio de sesión con un mensaje de éxito.

7. Clase LogInAction

- **Función**: Maneja la lógica del botón "Iniciar Sesión".
- **Detalles**:
    - Obtiene el usuario y contraseña ingresados.
    - Verifica:
        - Si el usuario existe en users y la contraseña coincide: Muestra un mensaje de éxito, limpia los campos, cierra la ventana y abre una nueva ventana Menu.
        - Si falla: Muestra "Usuario o contraseña incorrectos".

---

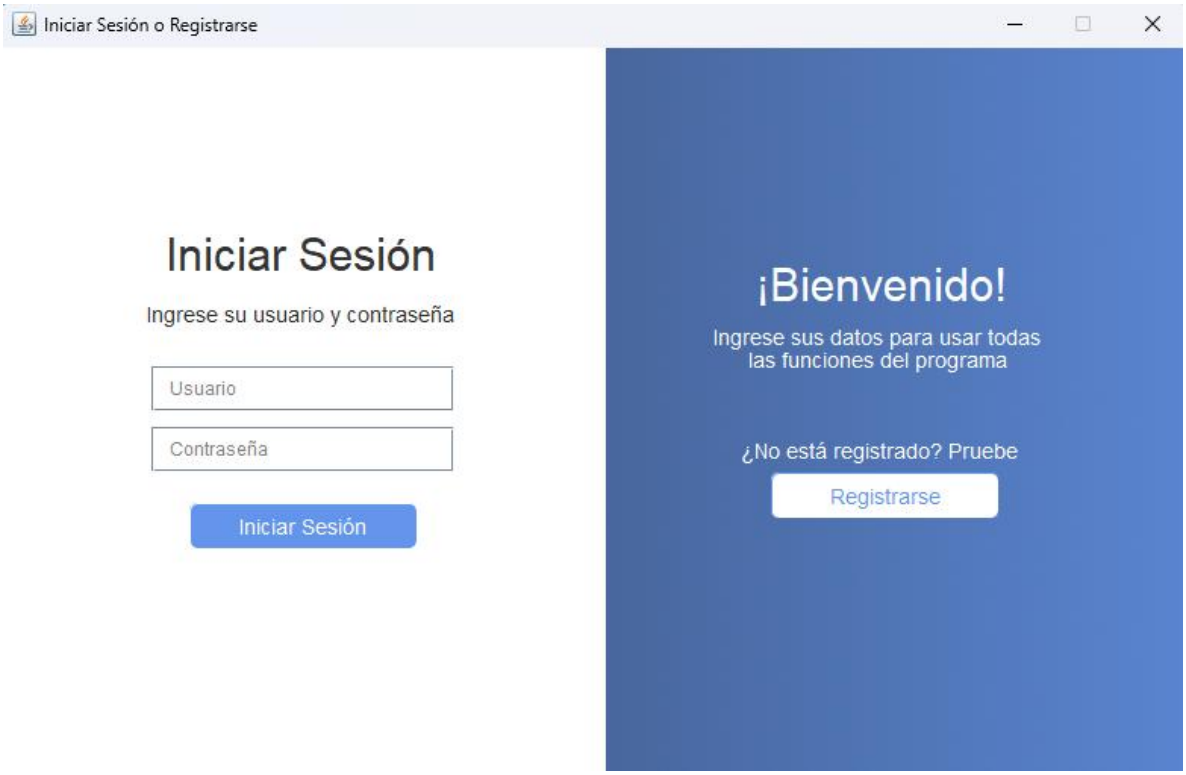
8. Métodos Auxiliares

- agregarListener(JFrame frame): Agrega un listener para que la ventana recupere el foco al hacer clic en ella.
- mostrarVentana(): Hace visible la ventana nuevamente.
- getInstancia(): Devuelve la instancia única de LogIn (patrón Singleton).

---

9. Diseño y Estilo

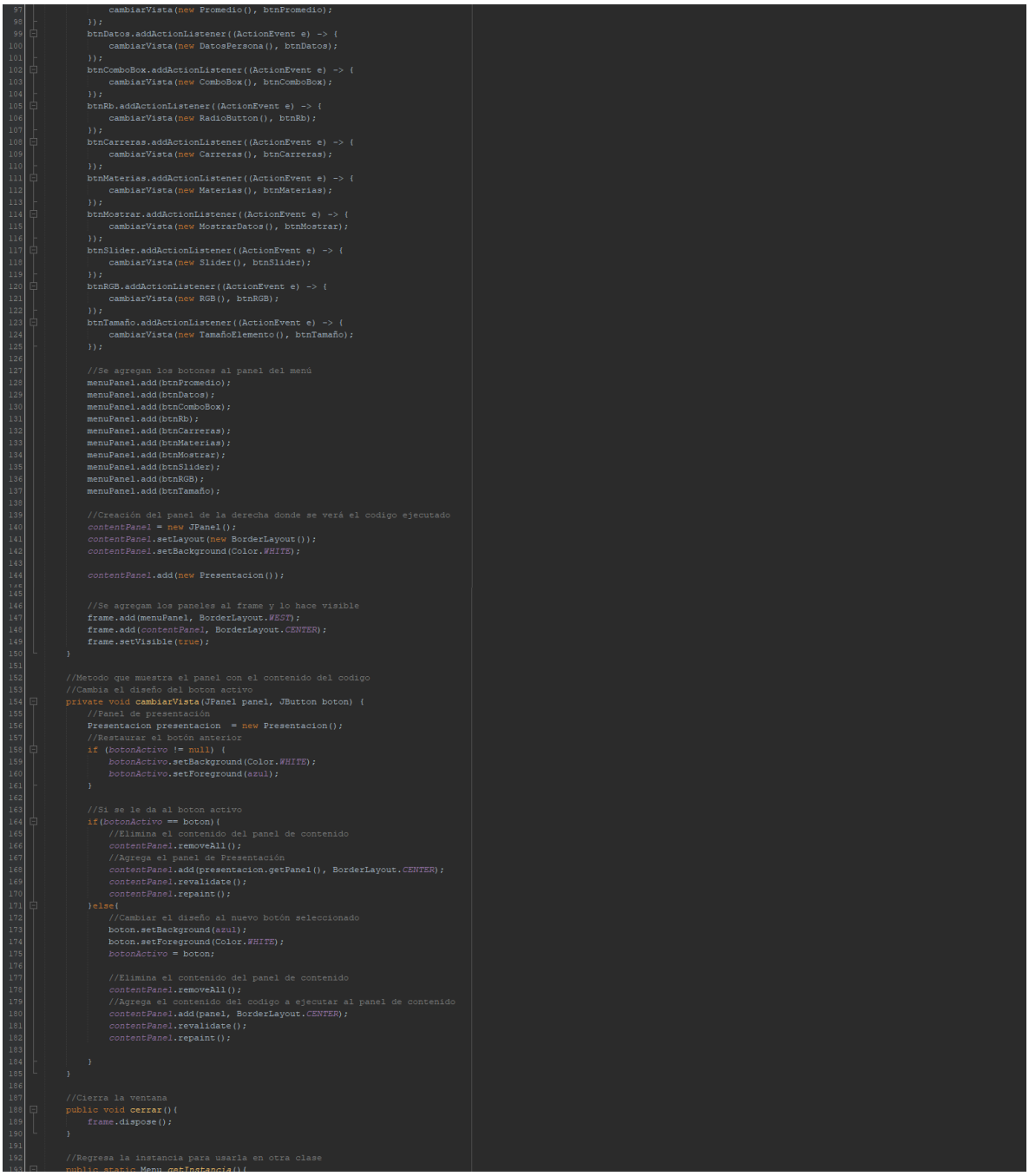
- Colores: Usa un azul personalizado (100, 149, 237) para fondos y botones.
- Fuentes: "Inter" para un diseño moderno y legible.
- Componentes personalizados:
    - RoundedButton: Botones con bordes redondeados.
    - PlaceholderTextField y PlaceholderPassField: Campos con texto de placeholder.


Iniciar Sesión o Registrarse
Iniciar Sesión
Ingrese su usuario y contraseña
Usuario
Contraseña
Iniciar Sesión
¡Bienvenido!
Ingrese sus datos para usar todas las funciones del programa
¿No está registrado? Pruebe
Registrarse

Código Menú

```
        JButton btnDatos = new RoundedButton("Datos");
        btnDatos.setBounds(50, 71, 150, 30);
        btnDatos.setBackground(Color.WHITE);
        btnDatos.setForeground(azul);

        JButton btnComboBox = new RoundedButton("ComboBox");
        btnComboBox.setBounds(50, 115, 150, 30);
        btnComboBox.setBackground(Color.WHITE);
        btnComboBox.setForeground(azul);

        JButton btnRb = new RoundedButton("Radio Button");
        btnRb.setBounds(50, 159, 150, 30);
        btnRb.setBackground(Color.WHITE);
        btnRb.setForeground(azul);

        JButton btnCarreras = new RoundedButton("Carreras");
        btnCarreras.setBounds(50, 203, 150, 30);
        btnCarreras.setBackground(Color.WHITE);
        btnCarreras.setForeground(azul);

        JButton btnMaterias = new RoundedButton("Materias");
        btnMaterias.setBounds(50, 251, 150, 30);
        btnMaterias.setBackground(Color.WHITE);
        btnMaterias.setForeground(azul);

        JButton btnMostrar = new RoundedButton("Mostrar Datos");
        btnMostrar.setBounds(50, 295, 150, 30);
        btnMostrar.setBackground(Color.WHITE);
        btnMostrar.setForeground(azul);

        JButton btnSlider = new RoundedButton("Slider");
        btnSlider.setBounds(50, 339, 150, 30);
        btnSlider.setBackground(Color.WHITE);
        btnSlider.setForeground(azul);

        JButton btnRGB = new RoundedButton("RGB");
        btnRGB.setBounds(50, 387, 150, 30);
        btnRGB.setBackground(Color.WHITE);
        btnRGB.setForeground(azul);

        JButton btnTamaño = new RoundedButton("Cambio Tamaño");
        btnTamaño.setBounds(50, 431, 150, 30);
        btnTamaño.setBackground(Color.WHITE);
        btnTamaño.setForeground(azul);

        //Ejecución de las clases al presionar los botones
        btnPromedio.addActionListener((ActionEvent e) -> {
            //Ejecuta el metodo cambiarVista
            cambiarVista(new Promedio(), btnPromedio);
```

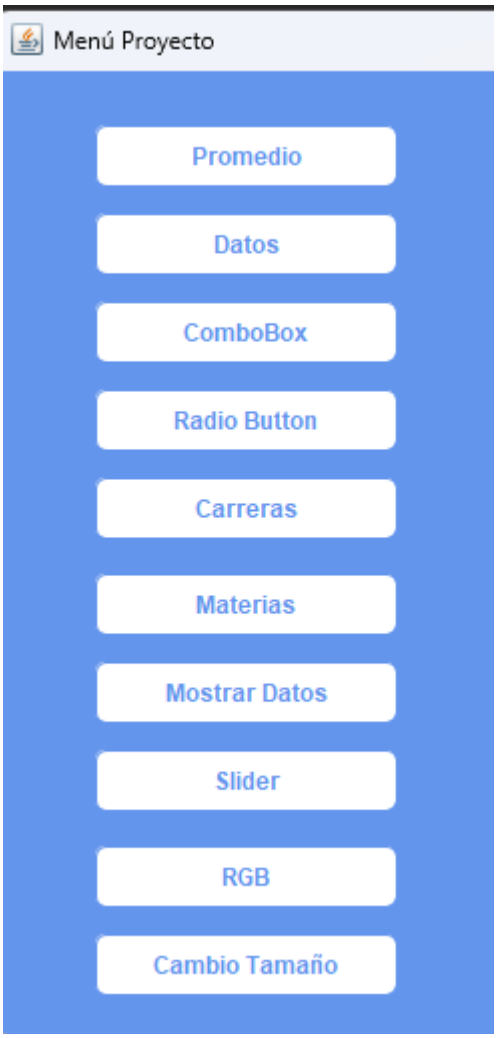
```
package proyecto;

//Se importan las librerias a usar
import elementos.RoundedButton;
import java.awt.BorderLayout;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;

//La clase Menú heredada de un JPanel. Panel personalizado
//Esta clase ejecuta la función del menú con todos los codigos en ella.
public class Menu extends JPanel{
    //Se declara fuera por que se usa en otro metodo fuera del constructor
    private static JPanel contentPanel; //Se declara el panel que cambiará todos los codigos
    private static JButton botonActivo = null; //Se declara el boton que guardará el activo
    private JFrame frame;
    private static Menu instancia;
    //Declarando un color que se usará en el diseño
    public Color azul = new Color(100, 149, 237);

    //Constructor
    public Menu(JFrame frame){
        //Se crea el frame que contendrá dos paneles y se hacen sus configuraciones
        frame = new JFrame("Menú Proyecto");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 520);
        frame.setResizable(false); //Bloquea la opción de maximizar
        frame.setLayout(new BorderLayout());

        this.frame = frame;
        instancia = this; //Guarda la instancia en una variable estatica

        //Centra la ventana al iniciar
        frame.setLocationRelativeTo(null);

        //Se crea el panel del lado izquierdo (Menú) que contiene los botones que ejecutan los códigos
        JPanel menuPanel = new JPanel();
        menuPanel.setLayout(null);
        //Configuración de tamaño y fondo
        menuPanel.setBackground(azul);
        menuPanel.setPreferredSize(new Dimension(250, 520));

        //Creación y diseño de los botones
        JButton btnPromedio = new RoundedButton("Promedio");
        btnPromedio.setBounds(50, 27, 150, 30);
        btnPromedio.setBackground(Color.WHITE);
        btnPromedio.setForeground(azul);

        JButton btnDatos = new RoundedButton("Datos");
```



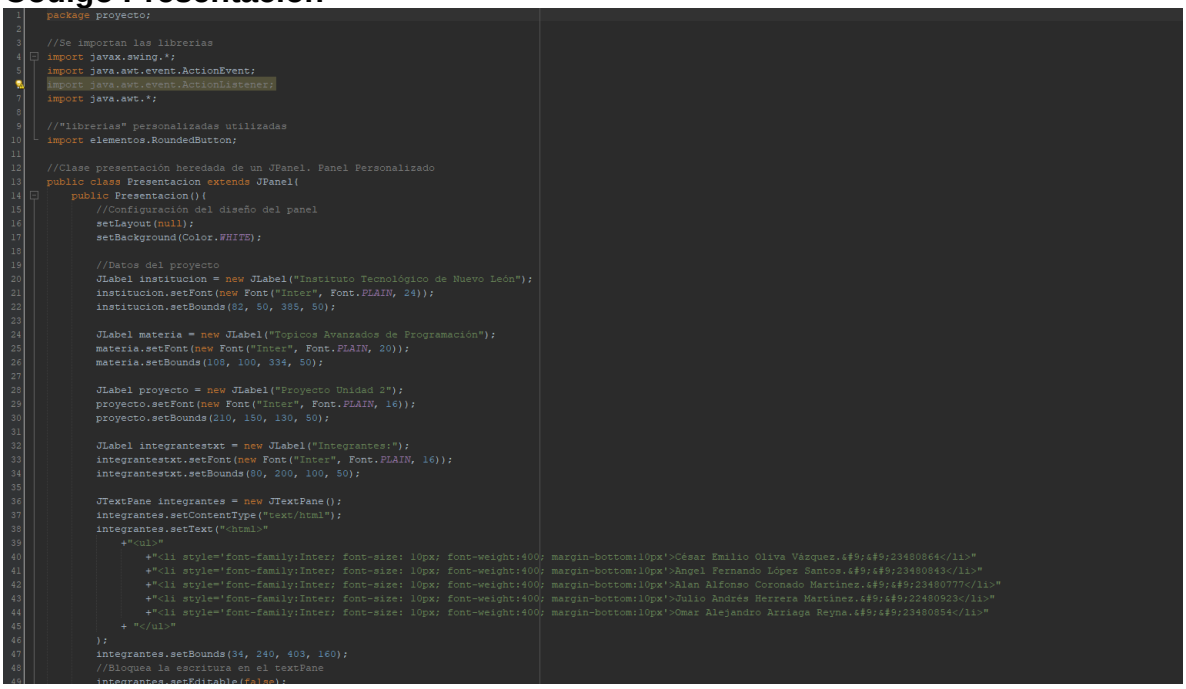
```
            cambiarVista(new Promedio(), btnPromedio);
        });
        btnDatos.addActionListener((ActionEvent e) -> {
            cambiarVista(new DatosPersona(), btnDatos);
        });
        btnComboBox.addActionListener((ActionEvent e) -> {
            cambiarVista(new ComboBox(), btnComboBox);
        });
        btnRb.addActionListener((ActionEvent e) -> {
            cambiarVista(new RadioButton(), btnRb);
        });
        btnCarreras.addActionListener((ActionEvent e) -> {
            cambiarVista(new Carreras(), btnCarreras);
        });
        btnMaterias.addActionListener((ActionEvent e) -> {
            cambiarVista(new Materias(), btnMaterias);
        });
        btnMostrar.addActionListener((ActionEvent e) -> {
            cambiarVista(new MostrarDatos(), btnMostrar);
        });
        btnSlider.addActionListener((ActionEvent e) -> {
            cambiarVista(new Slider(), btnSlider);
        });
        btnRGB.addActionListener((ActionEvent e) -> {
            cambiarVista(new RGB(), btnRGB);
        });
        btnTamaño.addActionListener((ActionEvent e) -> {
            cambiarVista(new TamañoElemento(), btnTamaño);
        });

        //Se agregan los botones al panel del menú
        menuPanel.add(btnPromedio);
        menuPanel.add(btnDatos);
        menuPanel.add(btnComboBox);
        menuPanel.add(btnRb);
        menuPanel.add(btnCarreras);
        menuPanel.add(btnMaterias);
        menuPanel.add(btnMostrar);
        menuPanel.add(btnSlider);
        menuPanel.add(btnRGB);
        menuPanel.add(btnTamaño);

        //Creación del panel de la derecha donde se verá el codigo ejecutado
        contentPanel = new JPanel();
        contentPanel.setLayout(new BorderLayout());
        contentPanel.setBackground(Color.WHITE);

        contentPanel.add(new Presentacion());


        //Se agregam los paneles al frame y lo hace visible
        frame.add(menuPanel, BorderLayout.WEST);
        frame.add(contentPanel, BorderLayout.CENTER);
        frame.setVisible(true);
    }

    //Metodo que muestra el panel con el contenido del codigo
    //Cambia el diseño del boton activo
    private void cambiarVista(JPanel panel, JButton boton) {
        //Panel de presentación
        Presentacion presentacion  = new Presentacion();
        //Restaurar el botón anterior
        if (botonActivo != null) {
            botonActivo.setBackground(Color.WHITE);
            botonActivo.setForeground(azul);
        }

        //Si se le da al boton activo
        if(botonActivo == boton){
            //Elimina el contenido del panel de contenido
            contentPanel.removeAll();
            //Agrega el panel de Presentación
            contentPanel.add(presentacion.getPanel(), BorderLayout.CENTER);
            contentPanel.revalidate();
            contentPanel.repaint();
        }else{
            //Cambiar el diseño al nuevo botón seleccionado
            boton.setBackground(azul);
            boton.setForeground(Color.WHITE);
            botonActivo = boton;

            //Elimina el contenido del panel de contenido
            contentPanel.removeAll();
            //Agrega el contenido del codigo a ejecutar al panel de contenido
            contentPanel.add(panel, BorderLayout.CENTER);
            contentPanel.revalidate();
            contentPanel.repaint();

        }
    }

    //Cierra la ventana
    public void cerrar(){
        frame.dispose();
    }

    //Regresa la instancia para usarla en otra clase
    public static Menu getInstancia(){
```

La clase Menu en Java, parte del paquete proyecto, crea una interfaz gráfica usando Swing. Hereda de JPanel y organiza una ventana (JFrame) de 800x520 píxeles con dos secciones: un panel izquierdo (menú) y un panel derecho (contenido). El menú, de fondo azul, contiene 10 botones personalizados (RoundedButton) que al hacer clic cambian el contenido del panel derecho, mostrando diferentes paneles como Promedio, DatosPersona, etc. Inicialmente muestra Presentacion. Usa un patrón singleton para mantener una única instancia, y el método cambiarVista gestiona los cambios de contenido y el estilo del botón activo. Otros métodos incluyen cerrar para cerrar la ventana y getInstancia para acceder a

la instancia única. Requiere clases externas como RoundedButton y los paneles asociados a cada botón.



Código Presentación



```java
package proyecto;

//Se importan las librerias
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.*;

//"librerias" personalizadas utilizadas
import elementos.RoundedButton;

//Clase presentación heredada de un JPanel. Panel Personalizado
public class Presentacion extends JPanel{
    public Presentacion(){
        //Configuración del diseño del panel
        setLayout(null);
        setBackground(Color.WHITE);

        //Datos del proyecto
        JLabel institucion = new JLabel("Instituto Tecnológico de Nuevo León");
        institucion.setFont(new Font("Inter", Font.PLAIN, 24));
        institucion.setBounds(82, 50, 388, 50);

        JLabel materia = new JLabel("Topicos Avanzados de Programación");
        materia.setFont(new Font("Inter", Font.PLAIN, 20));
        materia.setBounds(108, 100, 334, 50);

        JLabel proyecto = new JLabel("Proyecto Unidad 2");
        proyecto.setFont(new Font("Inter", Font.PLAIN, 16));
        proyecto.setBounds(210, 150, 130, 50);

        JLabel integrantestxt = new JLabel("Integrantes:");
        integrantestxt.setFont(new Font("Inter", Font.PLAIN, 16));
        integrantestxt.setBounds(80, 200, 100, 50);

        JTextPane integrantes = new JTextPane();
        integrantes.setContentType("text/html");
        integrantes.setText("<html>"
            +"<ul>"
                +"<li style='font-family:Inter; font-size: 10px; font-weight:400; margin-bottom:10px'>César Emilio Oliva Vázquez.&#9;&#9;23480864</li>"
                +"<li style='font-family:Inter; font-size: 10px; font-weight:400; margin-bottom:10px'>Angel Fernando López Santos.&#9;&#9;23480843</li>"
                +"<li style='font-family:Inter; font-size: 10px; font-weight:400; margin-bottom:10px'>Alan Alfonso Coronado Martinez.&#9;&#9;23480777</li>"
                +"<li style='font-family:Inter; font-size: 10px; font-weight:400; margin-bottom:10px'>Julio Andrés Herrera Martinez.&#9;&#9;22480923</li>"
                +"<li style='font-family:Inter; font-size: 10px; font-weight:400; margin-bottom:10px'>Omar Alejandro Arriaga Reyna.&#9;&#9;23480854</li>"
            + "</ul>"
        );
        integrantes.setBounds(34, 240, 403, 160);
        //Bloquea la escritura en el textPane
        integrantes.setEditable(false);
```

Descripción

- **Propósito: Este código crea un panel de presentación para el proyecto. Muestra información estática (institución, materia, proyecto, integrantes) y un botón funcional para cerrar sesión y regresar a una ventana de inicio de sesión.**
- **Diseño: Usa un layout nulo para posicionar manualmente los componentes con coordenadas específicas.**
- **Estilo: Fuentes "Inter" de diferentes tamaños, fondo blanco, un botón rojo con texto blanco y una lista de integrantes en formato HTML.**
- **Dependencias externas:**
    - **RoundedButton (clase personalizada para el botón).**
    - **Menu y LogIn (clases singleton que manejan el menú y la ventana de login, respectivamente).**


Menú Proyecto
Promedio
Datos
ComboBox
Radio Button
Carreras
Materias
Mostrar Datos
Slider
RGB
Cambio Tamaño
Instituto Tecnológico de Nuevo Le...
Topicos Avanzados de Programación
Proyecto Unidad 2
Integrantes:
• César Emilio Oliva Vázquez. 23480864
• Angel Fernando López Santos. 23480843
• Alan Alfonso Coronado Martínez. 23480777
• Julio Andrés Herrera Martínez. 22480923
• Omar Alejandro Arriaga Reyna. 23480854
Cerrar Sesión

Código Promedio

```java
package proyecto;

//Se importan las librerias a usar
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;

//"librerias" personalizadas utilizadas
import elementos.RoundedButton;
import elementos.PlaceholderTextField;

//La clase Promedio heredada de un JPanel. Panel personalizado
public class Promedio extends JPanel{
    public Promedio() {
        //Configuración del diseño del panel
        setLayout(null);
        setBackground(Color.WHITE);

        //Creación de Labels y Textfields para ingresar datos
        JLabel labelCal1 = new JLabel("Calificación 1:");
        labelCal1.setBounds(20, 30, 150, 30);

        JTextField textCal1 = new PlaceholderTextField("",20);
        textCal1.setBounds(110, 30, 150, 30);

        JLabel labelCal2 = new JLabel("Calificación 2:");
        labelCal2.setBounds(20, 80, 150, 30);

        JTextField textCal2 = new PlaceholderTextField("",20);
        textCal2.setBounds(110, 80, 150, 30);

        JLabel labelCal3 = new JLabel("Calificación 3:");
        labelCal3.setBounds(20, 130, 150, 30);

        JTextField textCal3 = new PlaceholderTextField("",20);
        textCal3.setBounds(110, 130, 150, 30);

        JLabel labelCal4 = new JLabel("Calificación 4:");
        labelCal4.setBounds(20, 180, 150, 30);

        JTextField textCal4 = new PlaceholderTextField("",20);
        textCal4.setBounds(110, 180, 150, 30);

        JLabel labelCal5 = new JLabel("Calificación 5:");
        labelCal5.setBounds(20, 230, 150, 30);

        JTextField textCal5 = new PlaceholderTextField("",20);
        textCal5.setBounds(110, 230, 150, 30);
```

```java
        //Creación del boton para hacer la función del promedio
        JButton boton = new RoundedButton("Promedio");
        boton.setBounds(20, 280, 100, 30);
        boton.setBackground(new Color(100, 149, 237));
        boton.setForeground(Color.WHITE);

        //Evento de acción al dar click en el botón
        boton.addActionListener((ActionEvent e) -> {
            //Intente calcular el promedio o evitar el error
            try {
                //Convertir el valor de los textFields en double
                double cal1 = Double.parseDouble(textCal1.getText());
                double cal2 = Double.parseDouble(textCal2.getText());
                double cal3 = Double.parseDouble(textCal3.getText());
                double cal4 = Double.parseDouble(textCal4.getText());
                double cal5 = Double.parseDouble(textCal5.getText());
                double promedio = (cal1+cal2+cal3+cal4+cal5)/5;
                //Mostrar el resultado mediante una ventana de dialogo
                JOptionPane.showMessageDialog(this, "Promedio: "+promedio);
                // Captura el error si la conversión de String a número falla
            } catch (NumberFormatException ex) {
                //Muestra mensaje de error
                JOptionPane.showMessageDialog(this, "Ingresa datos validos", "Error", JOptionPane.ERROR_MESSAGE);
            }
        });

        //Se agregan los elementos creados
        add(labelCal1);
        add(textCal1);
        add(labelCal2);
        add(textCal2);
        add(labelCal3);
        add(textCal3);
        add(labelCal4);
        add(textCal4);
        add(labelCal5);
        add(textCal5);
        add(boton);
    }
}
```

1. Estructura General

- **Clase**: Promedio es una clase pública que hereda de JPanel, lo que la convierte en un panel personalizado.
- **Propósito**: Crear un formulario para ingresar cinco calificaciones, calcular su promedio y mostrar el resultado.
- **Librerías**:
    - javax.swing.* y java.awt.*: Para componentes gráficos y diseño.
    - Clases personalizadas (RoundedButton, PlaceholderTextField): Un botón redondeado y campos de texto con placeholder.

---

2. Constructor (Promedio())

- **Función**: Configura el panel y agrega los elementos gráficos.
- **Detalles**:
    - Establece un diseño nulo (setLayout(null)), usando posicionamiento manual.
    - Define el fondo del panel como blanco (Color.WHITE).

---

3. Componentes Gráficos

- Campos de Texto
  - Calificación 1:
    - labelCal1: Etiqueta "Calificación 1:" en (20, 30), tamaño 150x30 píxeles.
    - textCal1: Campo de texto (PlaceholderTextField) en (110, 30), tamaño 150x30 píxeles, capacidad para 20 caracteres.
  - Calificación 2:
    - labelCal2: Etiqueta "Calificación 2:" en (20, 80), tamaño 150x30 píxeles.
    - textCal2: Campo de texto en (110, 80), tamaño 150x30 píxeles.
  - Calificación 3:
    - labelCal3: Etiqueta "Calificación 3:" en (20, 130), tamaño 150x30 píxeles.
    - textCal3: Campo de texto en (110, 130), tamaño 150x30 píxeles.
  - Calificación 4:
    - labelCal4: Etiqueta "Calificación 4:" en (20, 180), tamaño 150x30 píxeles.
    - textCal4: Campo de texto en (110, 180), tamaño 150x30 píxeles.
  - Calificación 5:
    - labelCal5: Etiqueta "Calificación 5:" en (20, 230), tamaño 150x30 píxeles.
    - textCal5: Campo de texto en (110, 230), tamaño 150x30 píxeles.
- Botón
  - boton: Botón redondeado (RoundedButton) con texto "Promedio":
    - Ubicado en (20, 280) con tamaño 100x30 píxeles.
    - Fondo azul (100, 149, 237) y texto blanco.

---

4. Lógica del Botón (ActionListener)

- **Función**: Maneja el evento al hacer clic en el botón "Promedio".
- **Detalles**:
  - Usa un bloque try-catch para manejar errores de conversión numérica.
  - Obtención y cálculo:
    - Convierte el texto de cada campo (textCal1 a textCal5) a double usando Double.parseDouble.
    - Calcula el promedio: (cal1 + cal2 + cal3 + cal4 + cal5) / 5.

---

5. Agregar Componentes

- **Función**: Añade todos los elementos gráficos al panel Promedio.
- **Componentes agregados:** labelCal1, textCal1, labelCal2, textCal2, labelCal3, textCal3, labelCal4, textCal4, labelCal5, textCal5, boton.

---

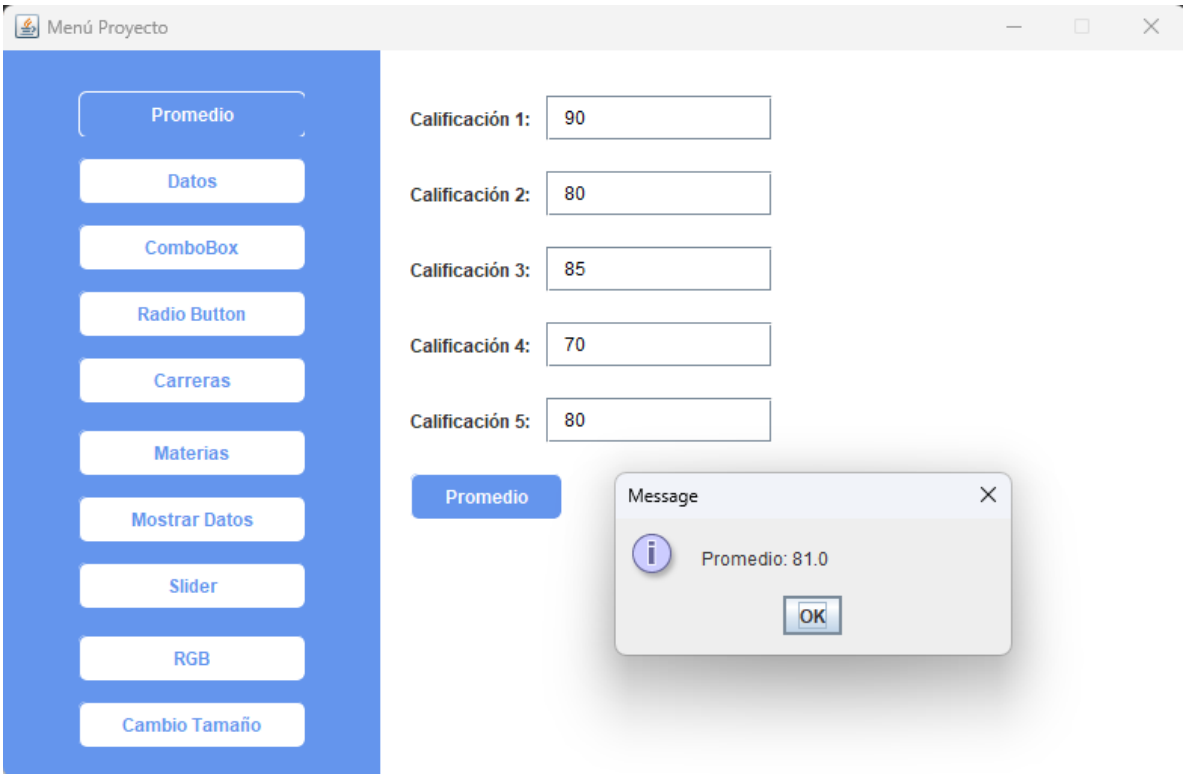
6. Flujo General

1. El panel muestra cinco campos de texto para ingresar calificaciones y un botón.
2. El usuario ingresa valores numéricos en los campos.
3. Al hacer clic en "Promedio":
    - Si todos los campos contienen números válidos, calcula y muestra el promedio.
    - Si algún campo no es un número válido (e.g., texto o vacío), muestra un mensaje de error.

---

7. Diseño y Estilo

- **Colores**: Fondo blanco para el panel, azul (100, 149, 237) para el botón con texto blanco.
- **Posicionamiento**: Uso de coordenadas manuales (setBounds) para un diseño estático.
- **Componentes personalizados:**
    - RoundedButton: Botón con bordes redondeados.
    - PlaceholderTextField: Campos de texto (sin placeholders visibles en este caso).


Menú Proyecto
Promedio
Datos
ComboBox
Radio Button
Carreras
Materias
Mostrar Datos
Slider
RGB
Cambio Tamaño
Calificación 1:
90
Calificación 2:
80
Calificación 3:
85
Calificación 4:
70
Calificación 5:
80
Promedio
Message
Promedio: 81.0
OK

Código Datos Personales

```java
package proyecto;

//Se importan las librerias a usar
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.*;

//"librerias" personalizadas utilizadas
import elementos.RoundedButton;
import elementos.PlaceholderTextField;

//La clase DatosPersona heredada de un JPanel. Panel personalizado
public class DatosPersona extends JPanel{
    public DatosPersona() {
        //Configuración del diseño del panel
        setLayout(null);
        setBackground(Color.WHITE);

        //Creación de Labels y Textfields para ingresar datos
        JLabel labelNombre = new JLabel("Nombre: ");
        labelNombre.setBounds(20, 30, 150, 30);

        JTextField textNombre = new PlaceholderTextField("",20);
        textNombre.setBounds(110, 30, 150, 30);

        JLabel labelApellido = new JLabel("Apellido:");
        labelApellido.setBounds(20, 80, 150, 30);

        JTextField textApellido = new PlaceholderTextField("",20);
        textApellido.setBounds(110, 80, 150, 30);

        JLabel labelEdad = new JLabel("Edad:");
        labelEdad.setBounds(20, 130, 150, 30);

        JTextField textEdad = new PlaceholderTextField("",20);
        textEdad.setBounds(110, 130, 150, 30);

        JLabel labelMateria = new JLabel("Materia:");
        labelMateria.setBounds(20, 180, 150, 30);

        JTextField textMateria = new PlaceholderTextField("",20);
        textMateria.setBounds(110, 180, 150, 30);

        JLabel labelCali = new JLabel("Calificación:");
        labelCali.setBounds(20, 230, 150, 30);

        JTextField textCali = new PlaceholderTextField("",20);
        textCali.setBounds(110, 230, 150, 30);
```

```java
        //Creación del boton para mostrar los datos ingresados
        JButton boton = new RoundedButton("Mostrar Datos");
        boton.setBounds(20, 280, 150, 30);
        boton.setBackground(new Color(100, 149, 237));
        boton.setForeground(Color.WHITE);

        //Declaración del arreglo para hacer la comparación con el label de nombre
        String[] numeros = new String[]{"1", "2", "3", "4", "5", "6", "7", "8", "9", "0"};

        //Evento de acción al dar click en el botón
        boton.addActionListener((ActionEvent e) -> {
            //Obtener el texto en los campos correspondientes
            String nombre = textNombre.getText();
            String apellido = textApellido.getText();
            String materia = textMateria.getText();

            //Intente convertir en numeros enteros o evitar el error
            try {
                int edad = Integer.parseInt(textEdad.getText());
                int cali = Integer.parseInt(textCali.getText());

                //Boolean para verificar si encontró un numero en el campo de nombre y apellido
                boolean encontrado = false;
                //Se recorre el arreglo para compararlo con los campos
                for (int i = 0; i < numeros.length; i++) {
                    if (nombre.contains(numeros[i]) || apellido.contains(numeros[i]) || materia.contains(numeros[i])) {
                        //Si los campos contienen un numero cambiar el estado del booleano a verdadero
                        encontrado = true;
                    }
                }
                //Si no encontró numeros en campos de String
                if (!encontrado) {
                    //Muestra los datos en una ventana de dialogo
                    JOptionPane.showMessageDialog(this, "Nombre: " + nombre + "\nApellido: " + apellido + "\nEdad: " + edad
                            + "\nMateria: " + materia + "\nCalificación: " + cali);
                } else {
                    //Muestra error
                    JOptionPane.showMessageDialog(this, "Ingresa datos validos", "Error", JOptionPane.ERROR_MESSAGE);
                }
            // Captura el error si la conversión de String a número falla en campos de edad y calificación
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(this, "Ingresa datos validos", "Error", JOptionPane.ERROR_MESSAGE);
            }
        });

        //Se agregan los elementos creados
        add(labelNombre);
        add(textNombre);
```

```java
        //Se agregan los elementos creados
        add(labelNombre);
        add(textNombre);
        add(labelApellido);
        add(textApellido);
        add(labelEdad);
        add(textEdad);
        add(labelMateria);
        add(textMateria);
        add(labelCali);
        add(textCali);
        add(boton);
    }
}
```

1. Estructura General

- **Clase**: DatosPersona es una clase pública que hereda de JPanel, lo que la convierte en un panel personalizado.
- **Propósito**: Crear un formulario para capturar datos (nombre, apellido, edad, materia y calificación), validar la entrada y mostrar la información en un mensaje.
- **Librerías**:
    - javax.swing.* y java.awt.*: Para componentes gráficos y diseño.
    - Clases personalizadas (RoundedButton, PlaceholderTextField): Un botón redondeado y campos de texto con placeholder.

---

2. Constructor (DatosPersona())

- **Función**: Configura el panel y agrega los elementos gráficos.
- **Detalles**:
    - Establece un diseño nulo (setLayout(null)), usando posicionamiento manual.
    - Define el fondo del panel como blanco (Color.WHITE).

---

3. Componentes Gráficos

- Campos de Texto
  - Nombre:
    - labelNombre: Etiqueta "Nombre:" en (20, 30), tamaño 150x30 píxeles.
    - textNombre: Campo de texto (PlaceholderTextField) en (110, 30), tamaño 150x30 píxeles, capacidad para 20 caracteres.
  - Apellido:
    - labelApellido: Etiqueta "Apellido:" en (20, 80), tamaño 150x30 píxeles.
    - textApellido: Campo de texto en (110, 80), tamaño 150x30 píxeles.
  - Edad:
    - labelEdad: Etiqueta "Edad:" en (20, 130), tamaño 150x30 píxeles.
    - textEdad: Campo de texto en (110, 130), tamaño 150x30 píxeles.
  - Materia:
    - labelMateria: Etiqueta "Materia:" en (20, 180), tamaño 150x30 píxeles.
    - textMateria: Campo de texto en (110, 180), tamaño 150x30 píxeles.
  - Calificación:
    - labelCali: Etiqueta "Calificación:" en (20, 230), tamaño 150x30 píxeles.
    - textCali: Campo de texto en (110, 230), tamaño 150x30 píxeles.
- Botón
  - boton: Botón redondeado (RoundedButton) con texto "Mostrar Datos":
    - Ubicado en (20, 280) con tamaño 150x30 píxeles.
    - Fondo azul (100, 149, 237) y texto blanco.
- Arreglo de Números
  - numeros: Arreglo de cadenas con dígitos del "0" al "9", usado para validar que los campos de texto (nombre, apellido, materia) no contengan números.

---

4. Lógica del Botón (ActionListener)

- **Función**: Maneja el evento al hacer clic en el botón "Mostrar Datos".
- **Detalles**:
    - Obtención de datos:
        - nombre, apellido, materia: Textos de los campos correspondientes.
        - edad y cali: Intentan convertirse de texto a enteros (Integer.parseInt).
    - Validación:
        - Usa un bloque try-catch para manejar errores de conversión numérica.
        - Variable encontrado: Booleano que indica si se encontró un número en nombre, apellido o materia.
        - Recorre el arreglo numeros y verifica con contains() si alguno está presente en los campos de texto.

---

5. Agregar Componentes

- Función: Añade todos los elementos gráficos al panel DatosPersona.
- Componentes agregados: labelNombre, textNombre, labelApellido, textApellido, labelEdad, textEdad, labelMateria, textMateria, labelCali, textCali, boton.

---

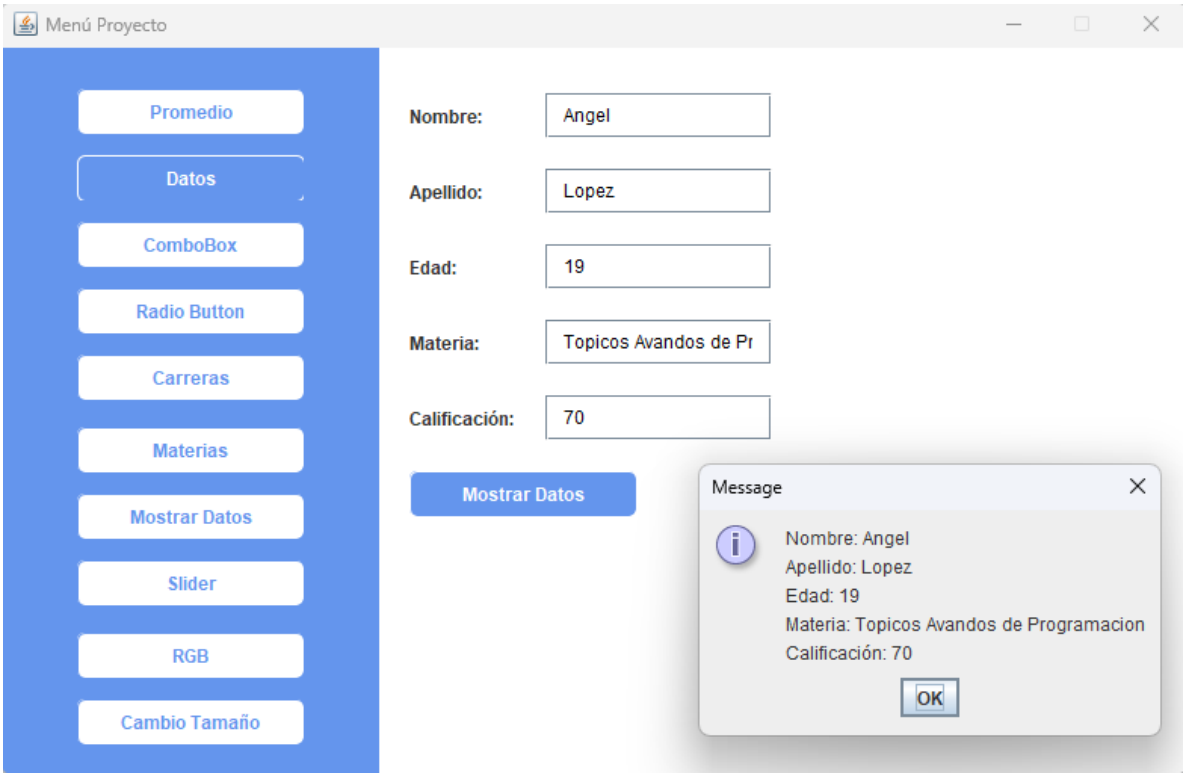
6. Flujo General

1. El panel muestra un formulario con campos para nombre, apellido, edad, materia y calificación, más un botón.
2. El usuario ingresa datos en los campos.
3. Al hacer clic en "Mostrar Datos":
    - Si los campos de texto (nombre, apellido, materia) no tienen números y los campos numéricos (edad, calificación) son válidos, muestra los datos en un mensaje.
    - Si hay números en los campos de texto o los campos numéricos no son enteros válidos, muestra un mensaje de error.

---

7. Diseño y Estilo

- Colores: Fondo blanco para el panel, azul (100, 149, 237) para el botón con texto blanco.
- Posicionamiento: Uso de coordenadas manuales (setBounds) para un diseño estático.
- Componentes personalizados:
  - RoundedButton: Botón con bordes redondeados.
  - PlaceholderTextField: Campos de texto (sin placeholders visibles en este caso).



Código ComboBox

```java
package proyecto;

//Se importan las librerias a usar
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.*;
import java.awt.event.ActionListener;

//Librerias personalizadas a utilizar
import elementos.RoundedButton;


//La clase heredada de un JPanel. Panel personalizado
public class ComboBox extends JPanel{
    public ComboBox() {
        //Configuración del diseño del panel
        setLayout(null);
        setBackground(Color.WHITE);

        //Opciones a mostrar en el comboBox
        String opciones[] = {"Opcion 1", "Opcion 2", "Opcion 3", "Opcion 4", "Opcion 5", "Opcion 6", "Opcion 7", "Opcion 8",
        "Opcion 9", "Opcion 10", "Opcion 11", "Opcion 12", "Opcion 13", "Opcion 14", "Opcion 15", "Opcion 16",
        "Opcion 17", "Opcion 18", "Opcion 19", "Opcion 20"};

        //Creción del componente comboBox
        JComboBox<String> comboBox = new JComboBox<>(opciones);
        comboBox.setMaximumRowCount(5);
        comboBox.setBounds(20, 30, 150, 30);
        comboBox.setBackground(Color.WHITE);

        //Creación del botón
        JButton boton = new RoundedButton("Mostrar Selección");
        boton.setBounds(250, 30, 150, 30);
        boton.setBackground(new Color(100, 149, 237));
        boton.setForeground(Color.WHITE);

        //Evento de acción al dar click en el botón
        boton.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                //Obtener el item seleccionado del comboBox
                String seleccion = (String) comboBox.getSelectedItem();
                //Muestra una ventana con la selección
                JOptionPane.showMessageDialog(ComboBox.this, "Opcion seleccionada: "+seleccion, "Selección" ,JOptionPane.INFORMATION_MESSAGE);
            }
        });

        //Se agregan los elementos creados
        add(comboBox);
```

```java
        //Evento de acción al dar click en el botón
        boton.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                //Obtener el item seleccionado del comboBox
                String seleccion = (String) comboBox.getSelectedItem();
                //Muestra una ventana con la selección
                JOptionPane.showMessageDialog(ComboBox.this, "Opcion seleccionada: "+seleccion, "Selección" ,JOptionPane.INFORMATION_MESSAGE);
            }
        });

        //Se agregan los elementos creados
        add(comboBox);
        add(boton);
    }
}
```

1. Estructura General

- **Clase**: ComboBox es una clase pública que hereda de JPanel, lo que la convierte en un panel personalizado.
- **Propósito**: Mostrar una lista desplegable con 20 opciones genéricas y permitir al usuario seleccionar una, mostrando la selección al hacer clic en un botón.
- **Librerías**:
    - javax.swing.* y java.awt.*: Para componentes gráficos y diseño.
    - Clase personalizada (RoundedButton): Un botón con bordes redondeados.

---

2. Constructor (ComboBox())

- Función: Configura el panel y agrega los elementos gráficos.
- Detalles:
    - Establece un diseño nulo (setLayout(null)), lo que implica posicionamiento manual de los componentes.
    - Define el fondo del panel como blanco (Color.WHITE).

---

3. Componentes Gráficos

- Lista Desplegable
    - opciones: Arreglo de cadenas con 20 elementos genéricos: "Opcion 1" hasta "Opcion 20".
    - comboBox: Lista desplegable (JComboBox) que muestra las opciones:
        - Ubicada en (20, 30) con tamaño 150x30 píxeles.
        - Fondo blanco (Color.WHITE).
        - Máximo de 5 opciones visibles al desplegarse (setMaximumRowCount(5)).
- Botón
    - boton: Botón redondeado (RoundedButton) con texto "Mostrar Selección":
        - Ubicado en (250, 30) con tamaño 150x30 píxeles.
        - Fondo azul (100, 149, 237) y texto blanco.

---

4. Lógica del Botón (ActionListener)

- **Función**: Maneja el evento al hacer clic en el botón "Mostrar Selección".
- **Detalles**:
    - Obtiene la opción seleccionada en comboBox con getSelectedItem() y la convierte a String (seleccion).
    - Muestra un JOptionPane con el mensaje:

---

5. Agregar Componentes

- **Función**: Añade los elementos gráficos al panel ComboBox.
- **Componentes agregados:** comboBox y boton.

---

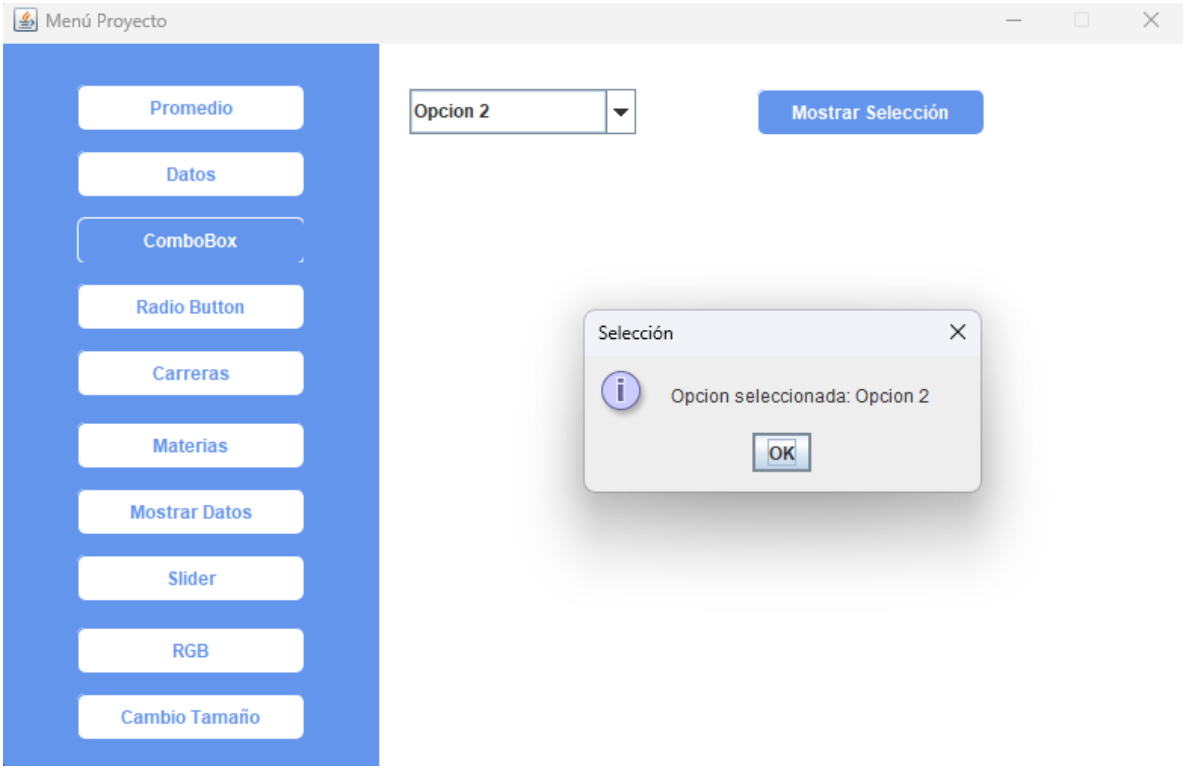
6. Flujo General

1. El panel se inicializa con una lista desplegable que contiene 20 opciones y un botón.
2. El usuario selecciona una opción de la lista.
3. Al hacer clic en "Mostrar Selección", se muestra un mensaje con la opción elegida.

---

7. Diseño y Estilo

- Colores: Fondo blanco para el panel y el JComboBox, azul (100, 149, 237) para el botón con texto blanco.
- Posicionamiento: Uso de coordenadas manuales (setBounds) para un diseño estático.
- Componente personalizado:
    - RoundedButton: Botón con bordes redondeados.



Código RadioButton

```java
package proyecto;

//Se importan las librerias a usar
import java.awt.event.ActionEvent;
import javax.swing.*;
import java.awt.*;
//"librerias" personalizadas utilizadas
import elementos.RoundedButton;

//La clase heredada de un JPanel. Panel personalizado
public class RadioButton extends JPanel{
    public RadioButton() {
        setLayout(null);
        setBackground(Color.WHITE);

        //Creación y posición de componentes
        JLabel carreras = new JLabel("Carreras ITNL:");
        carreras.setBounds(20, 30, 150, 30);

        JRadioButton amb = new JRadioButton("Ing. Ambiental");
        amb.setBounds(20, 65, 250, 30);
        amb.setBackground(Color.WHITE);

        JRadioButton ige = new JRadioButton("Ing. Gestión Empresarial");
        ige.setBounds(20, 100, 250, 30);
        ige.setBackground(Color.WHITE);

        JRadioButton iem = new JRadioButton("Ing. Electromecánica");
        iem.setBounds(20, 135, 250, 30);
        iem.setBackground(Color.WHITE);

        JRadioButton ie = new JRadioButton("Ing. Eléctrica");
        ie.setBounds(20, 170, 250, 30);
        ie.setBackground(Color.WHITE);

        JRadioButton ind = new JRadioButton("Ing. Industrial");
        ind.setBounds(20, 205, 250, 30);
        ind.setBackground(Color.WHITE);

        JRadioButton im = new JRadioButton("Ing. Mecatrónica");
        im.setBounds(20, 240, 250, 30);
        im.setBackground(Color.WHITE);

        JRadioButton isc = new JRadioButton("Ing. Sistemas Computacionales");
        isc.setBounds(20, 275, 250, 30);
        isc.setBackground(Color.WHITE);

        JRadioButton semi = new JRadioButton("Ing. Semiconductores");
        semi.setBounds(20, 310, 250, 30);
        semi.setBackground(Color.WHITE);

        //agrega todos los radio buttons a un grupo de selección
        ButtonGroup opciones = new ButtonGroup();
        opciones.add(amb);
        opciones.add(ige);
        opciones.add(iem);
        opciones.add(ie);
        opciones.add(ind);
        opciones.add(im);
        opciones.add(isc);
        opciones.add(semi);

        //Creación del boton para mostrar la elección
        JButton boton = new RoundedButton("Mostrar Selección");
        boton.setBounds(20, 355, 150, 30);
        boton.setBackground(new Color(100, 149, 237));
        boton.setForeground(Color.WHITE);

        //Evento de accion al dar click en el botón
        boton.addActionListener((ActionEvent e) -> {
            //Verifica la opción del radioButton seleccionada
            if (amb.isSelected()) {
                //Muestra una ventana con la selección
                JOptionPane.showMessageDialog(this, "Seleccionaste: Ing. Ambiental");
            } else if (ige.isSelected()) {
                JOptionPane.showMessageDialog(this, "Seleccionaste: Ing. Gestión Empresarial");
            } else if (iem.isSelected()) {
                JOptionPane.showMessageDialog(this, "Seleccionaste: Ing. Electromecanica");
            } else if (ie.isSelected()) {
                JOptionPane.showMessageDialog(this, "Seleccionaste: Ing. Electrica");
            } else if (ind.isSelected()) {
                JOptionPane.showMessageDialog(this, "Seleccionaste: Ing. Industrial");
            } else if (im.isSelected()) {
                JOptionPane.showMessageDialog(this, "Seleccionaste: Ing. Mecatronica");
            } else if (isc.isSelected()) {
                JOptionPane.showMessageDialog(this, "Seleccionaste: Ing. Sistemas Computacionales");
            } else if (semi.isSelected()) {
                JOptionPane.showMessageDialog(this, "Seleccionaste: Ing. SemiConductores");
            } else {
                JOptionPane.showMessageDialog(this, "No has seleccionado ninguna opción");
            }
        });

        //Se agregan los elementos creados
        add(carreras);
        add(amb);
        add(ige);
        add(iem);
        add(ie);
        add(ind);
        add(im);
        add(isc);
        add(semi);
        add(boton);
    }
}
```

1. Estructura General

- **Clase**: RadioButton es una clase pública que hereda de JPanel, lo que la convierte en un panel personalizado.
- **Propósito**: Crear un formulario con botones de radio para elegir una carrera del "ITNL" (Instituto Tecnológico Nacional de Laredo, presumiblemente) y mostrar la opción seleccionada.
- **Librerías**:
    - javax.swing.* y java.awt.*: Para componentes gráficos y diseño.
    - Clase personalizada (RoundedButton): Un botón con bordes redondeados.

---

2. Constructor (RadioButton())

- **Función**: Configura el panel y agrega los elementos gráficos.
- **Detalles**:
    - Establece un diseño nulo (setLayout(null)), usando posicionamiento manual.
    - Define el fondo del panel como blanco (Color.WHITE).

---

3. Componentes Gráficos

- Etiqueta
  - carreras: Etiqueta "Carreras ITNL:" ubicada en (20, 30) con tamaño 150x30 píxeles.
- Botones de Radio (JRadioButton)
  - Lista de carreras disponibles, cada una con su propio JRadioButton:
    - amb: "Ing. Ambiental" en (20, 65).
    - ige: "Ing. Gestión Empresarial" en (20, 100).
    - iem: "Ing. Electromecánica" en (20, 135).
    - ie: "Ing. Eléctrica" en (20, 170).
    - ind: "Ing. Industrial" en (20, 205).
    - im: "Ing. Mecatrónica" en (20, 240).
    - isc: "Ing. Sistemas Computacionales" en (20, 275).
    - semi: "Ing. Semiconductores" en (20, 310).
  - Propiedades:
    - Todos tienen tamaño 250x30 píxeles.
    - Fondo blanco (setBackground(Color.WHITE)).
  - Agrupación:
    - Todos los JRadioButton se añaden a un ButtonGroup (opciones), asegurando que solo uno pueda estar seleccionado a la vez.
- Botón
  - boton: Botón redondeado (RoundedButton) con texto "Mostrar Selección":
    - Ubicado en (20, 355) con tamaño 150x30 píxeles.
    - Fondo azul (100, 149, 237) y texto blanco.

---

4. Lógica del Botón (ActionListener)

- **Función**: Maneja el evento al hacer clic en el botón "Mostrar Selección".
- **Detalles**:
  - Verifica cuál JRadioButton está seleccionado usando isSelected() en una serie de condiciones if-else.
  - Muestra un JOptionPane con el mensaje correspondiente:
    - "Seleccionaste: [carrera]" si hay una opción seleccionada.
    - "No has seleccionado ninguna opción" si no hay selección.
  - Ejemplos de mensajes:
    - Si amb está seleccionado: "Seleccionaste: Ing. Ambiental".
    - Si isc está seleccionado: "Seleccionaste: Ing. Sistemas Computacionales".

---

5. Agregar Componentes

- **Función**: Añade todos los elementos gráficos al panel RadioButton.
- **Componentes agregados:** carreras, amb, ige, iem, ie, ind, im, isc, semi, boton.

---

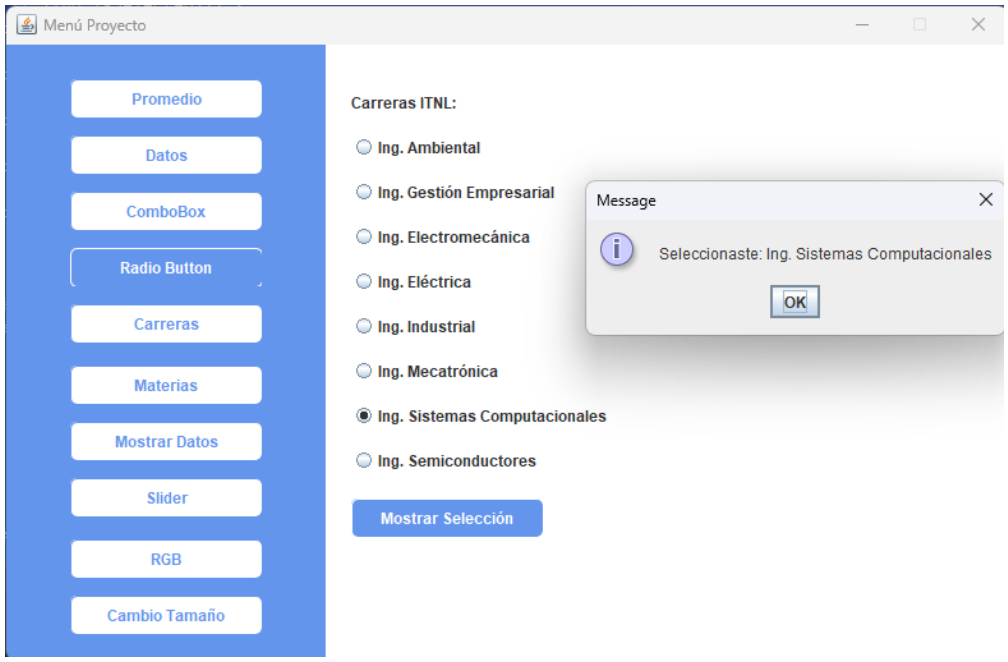
6. Flujo General

1. El panel muestra una lista de carreras como botones de radio y un botón para confirmar la selección.
2. El usuario selecciona una carrera (solo una opción es posible gracias al ButtonGroup).
3. Al hacer clic en "Mostrar Selección":
    - Si hay una carrera seleccionada, muestra un mensaje con la opción elegida.
    - Si no hay selección, muestra un mensaje indicando que no se eligió nada.

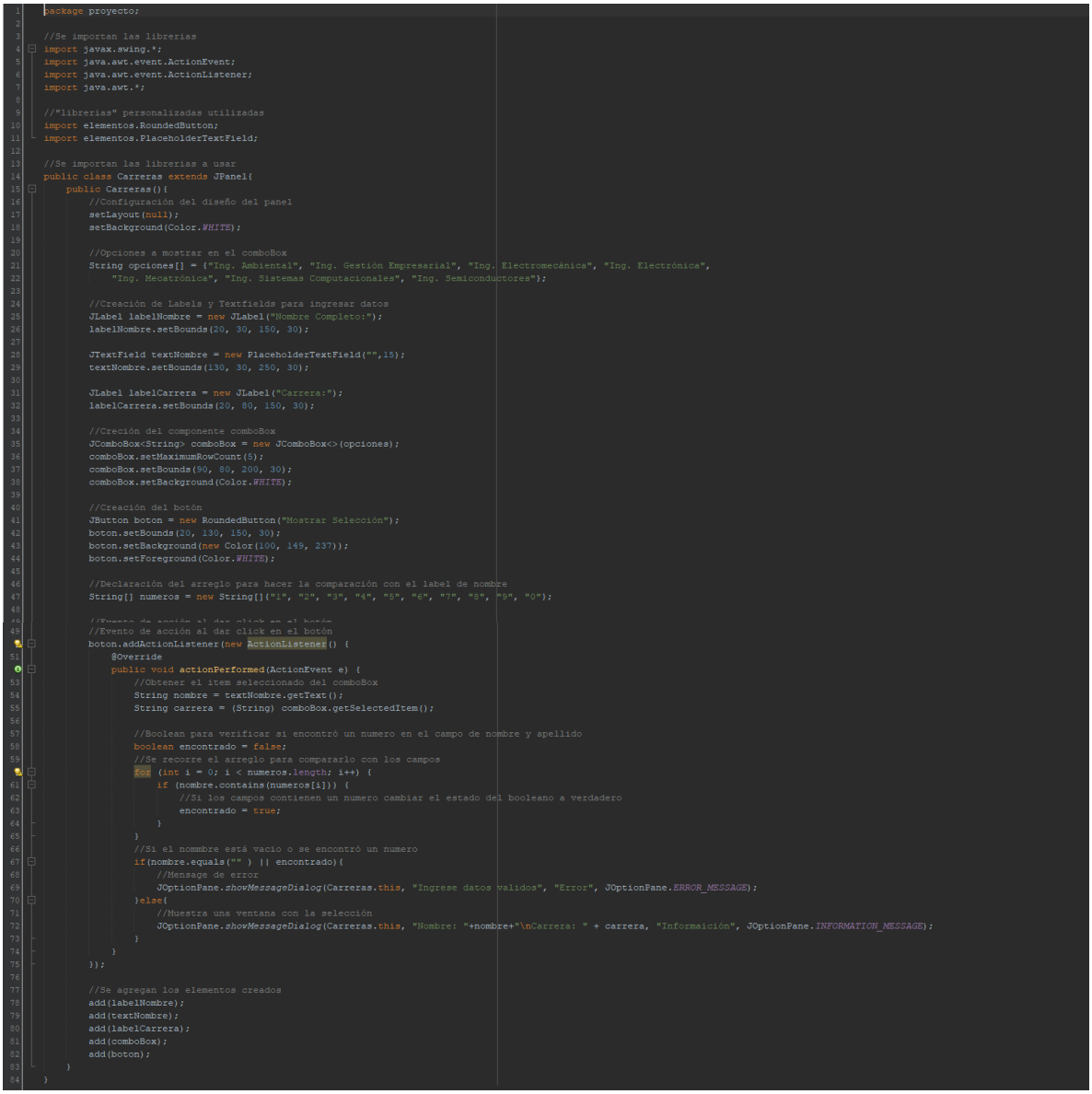
---

7. Diseño y Estilo

- Colores: Fondo blanco para el panel y los botones de radio; azul (100, 149, 237) para el botón con texto blanco.
- Posicionamiento: Uso de coordenadas manuales (setBounds) para un diseño estático.
- Componente personalizado:
    - RoundedButton: Botón con bordes redondeados.


Menú Proyecto
Promedio
Datos
ComboBox
Radio Button
Carreras
Materias
Mostrar Datos
Slider
RGB
Cambio Tamaño
Carreras ITNL:
Ing. Ambiental
Ing. Gestión Empresarial
Ing. Electromecánica
Ing. Eléctrica
Ing. Industrial
Ing. Mecatrónica
Ing. Sistemas Computacionales
Ing. Semiconductores
Mostrar Selección
Message
Seleccionaste: Ing. Sistemas Computacionales
OK

Codigo Carreras



```java
package proyecto;

//Se importan las librerias
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.*;

//"librerias" personalizadas utilizadas
import elementos.RoundedButton;
import elementos.PlaceholderTextField;

//Se importan las librerias a usar
public class Carreras extends JPanel{
    public Carreras(){
        //Configuración del diseño del panel
        setLayout(null);
        setBackground(Color.WHITE);

        //Opciones a mostrar en el comboBox
        String opciones[] = {"Ing. Ambiental", "Ing. Gestión Empresarial", "Ing. Electromecánica", "Ing. Electrónica",
            "Ing. Mecatrónica", "Ing. Sistemas Computacionales", "Ing. Semiconductores"};

        //Creación de Labels y Textfields para ingresar datos
        JLabel labelNombre = new JLabel("Nombre Completo:");
        labelNombre.setBounds(20, 30, 150, 30);

        JTextField textNombre = new PlaceholderTextField("",15);
        textNombre.setBounds(130, 30, 250, 30);

        JLabel labelCarrera = new JLabel("Carrera:");
        labelCarrera.setBounds(20, 80, 150, 30);

        //Creción del componente comboBox
        JComboBox<String> comboBox = new JComboBox<>(opciones);
        comboBox.setMaximumRowCount(5);
        comboBox.setBounds(90, 80, 200, 30);
        comboBox.setBackground(Color.WHITE);

        //Creación del botón
        JButton boton = new RoundedButton("Mostrar Selección");
        boton.setBounds(20, 130, 150, 30);
        boton.setBackground(new Color(100, 149, 237));
        boton.setForeground(Color.WHITE);

        //Declaración del arreglo para hacer la comparación con el label de nombre
        String[] numeros = new String[]{"1", "2", "3", "4", "5", "6", "7", "8", "9", "0"};

        //Evento de acción al dar click en el botón
        boton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                //Obtener el item seleccionado del comboBox
                String nombre = textNombre.getText();
                String carrera = (String) comboBox.getSelectedItem();

                //Boolean para verificar si encontró un numero en el campo de nombre y apellido
                boolean encontrado = false;
                //Se recorre el arreglo para compararlo con los campos
                for (int i = 0; i < numeros.length; i++) {
                    if (nombre.contains(numeros[i])) {
                        //Si los campos contienen un numero cambiar el estado del booleano a verdadero
                        encontrado = true;
                    }
                }
                //Si el nommbre está vacio o se encontró un numero
                if(nombre.equals("") || encontrado){
                    //Mensage de error
                    JOptionPane.showMessageDialog(Carreras.this, "Ingrese datos validos", "Error", JOptionPane.ERROR_MESSAGE);
                }else{
                    //Muestra una ventana con la selección
                    JOptionPane.showMessageDialog(Carreras.this, "Nombre: "+nombre+"\nCarrera: " + carrera, "Informaición", JOptionPane.INFORMATION_MESSAGE);
                }
            }
        });

        //Se agregan los elementos creados
        add(labelNombre);
        add(textNombre);
        add(labelCarrera);
        add(comboBox);
        add(boton);
    }
}
```

1. Estructura General

- **Clase**: Carreras es una clase pública que hereda de JPanel, lo que la convierte en un panel personalizado.
- **Propósito**: Crear un formulario para capturar el nombre completo de una persona y su carrera, validando que el nombre no contenga números ni esté vacío, y mostrar la selección en un mensaje.
- **Librerías**:
    - javax.swing.* y java.awt.*: Para componentes gráficos y diseño.
    - Clases personalizadas (RoundedButton, PlaceholderTextField): Un botón redondeado y un campo de texto con placeholder.

2. Constructor (Carreras())

- **Función**: Configura el panel y agrega los elementos gráficos.
- **Detalles**:
    - Establece un diseño nulo (setLayout(null)), lo que significa que los componentes se posicionan manualmente con coordenadas.
    - Define el fondo del panel como blanco (Color.WHITE).

---

3. Componentes Gráficos

- Campo de Nombre
    - labelNombre: Etiqueta "Nombre Completo:" ubicada en (20, 30) con tamaño 150x30 píxeles.
    - textNombre: Campo de texto (PlaceholderTextField) con capacidad para 15 caracteres, ubicado en (130, 30) con tamaño 250x30 píxeles. No tiene placeholder definido explícitamente (se pasa "").
- Lista de Carreras
    - opciones: Arreglo de cadenas con 7 carreras disponibles:
        - "Ing. Ambiental", "Ing. Gestión Empresarial", "Ing. Electromecánica", "Ing. Electrónica", "Ing. Mecatrónica", "Ing. Sistemas Computacionales", "Ing. Semiconductores".
    - labelCarrera: Etiqueta "Carrera:" ubicada en (20, 80) con tamaño 150x30 píxeles.
    - comboBox: Lista desplegable (JComboBox) que muestra las opciones:
        - Ubicada en (90, 80) con tamaño 200x30 píxeles.
        - Fondo blanco.
        - Máximo de 5 opciones visibles al desplegarse (setMaximumRowCount(5)).
- Botón
    - botón: Botón redondeado (RoundedButton) con texto "Mostrar Selección":
        - Ubicado en (20, 130) con tamaño 150x30 píxeles.
        - Fondo azul (100, 149, 237) y texto blanco.
- Arreglo de Números
    - números: Arreglo de cadenas con dígitos del "0" al "9", usado para validar que el nombre no contenga números.

---

4. Lógica del Botón (ActionListener)

- Función: Maneja el evento al hacer clic en el botón "Mostrar Selección".
- Detalles:
    - Obtención de datos:
        - nombre: Texto ingresado en textNombre.
        - carrera: Opción seleccionada en comboBox.
    - Validación:
        - Variable encontrado: Booleano que indica si se encontró un número en el nombre.
        - Recorre el arreglo numeros y verifica si alguno está presente en nombre usando contains().
        - Condiciones de error:
            - Si nombre está vacío (nombre.equals("")).
            - Si encontrado es true (hay un número en el nombre).
        - Si hay error: Muestra un JOptionPane con el mensaje "Ingrese datos válidos" y un ícono de error.

5. Agregar Componentes

- Función: Añade todos los elementos gráficos al panel Carreras.
- Componentes agregados: labelNombre, textNombre, labelCarrera, comboBox, boton.

---

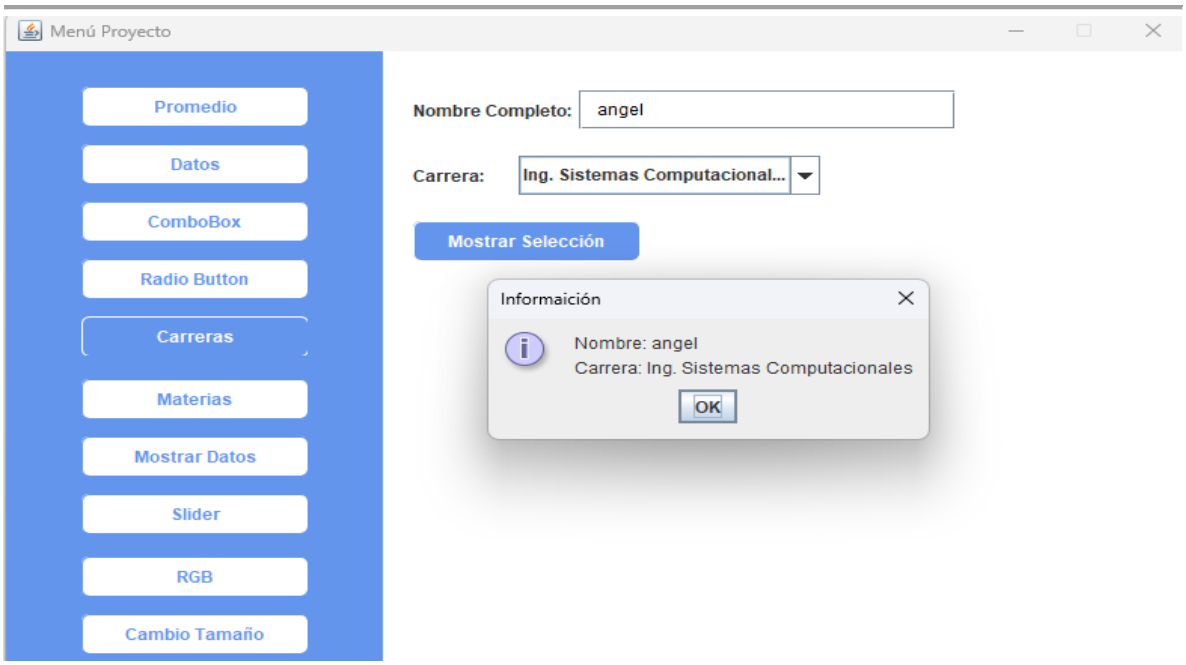
6. Flujo General

1. El panel se inicializa con un campo de texto para el nombre, una lista desplegable de carreras y un botón.
2. El usuario ingresa su nombre y selecciona una carrera.
3. Al hacer clic en "Mostrar Selección":
    - Si el nombre está vacío o contiene números, muestra un mensaje de error.
    - Si el nombre es válido, muestra un mensaje con el nombre y la carrera seleccionada.

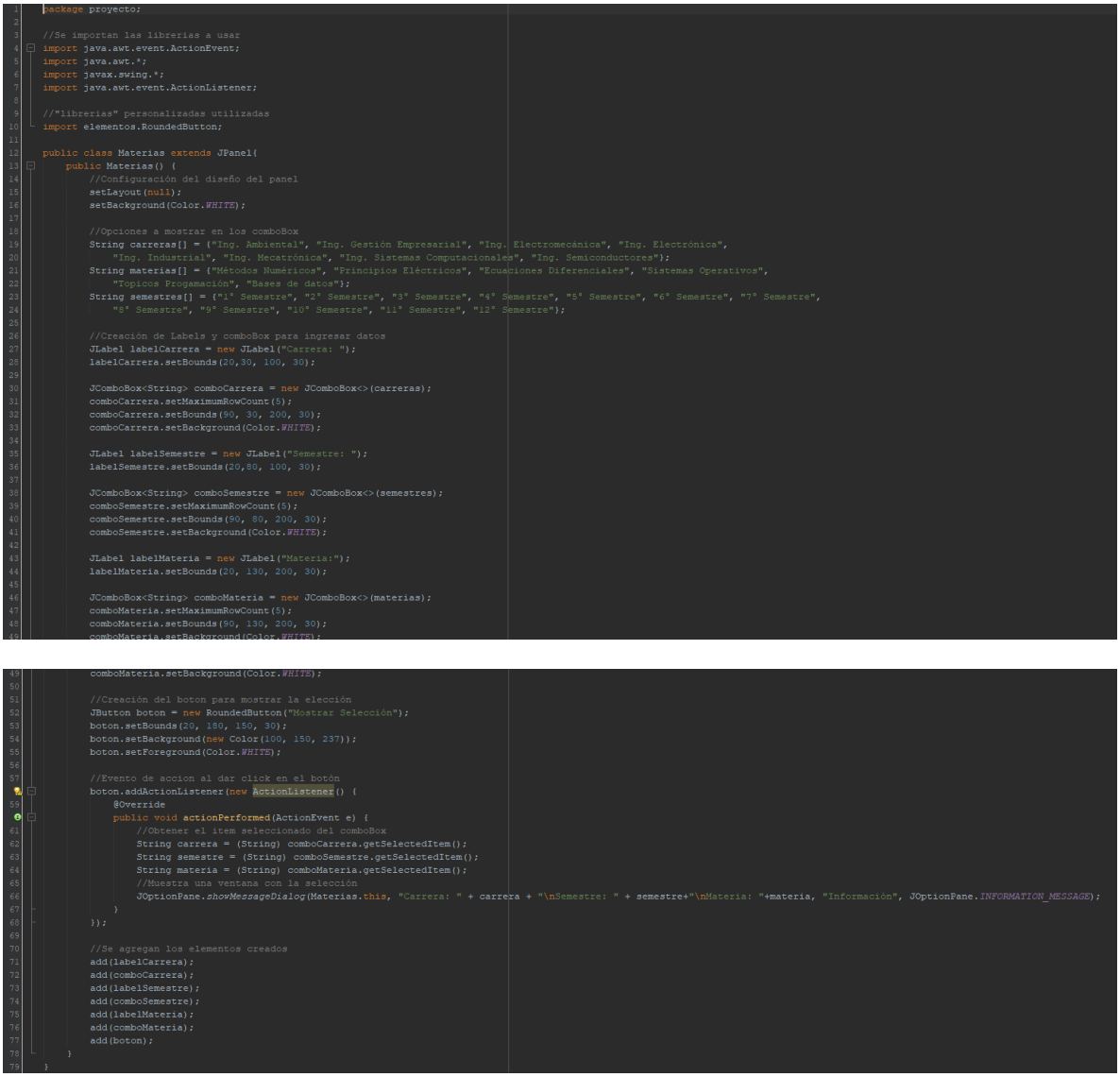
---

7. Diseño y Estilo

- Colores: Fondo blanco para el panel, azul (100, 149, 237) para el botón con texto blanco.
- Posicionamiento: Uso de coordenadas manuales (setBounds) para un diseño estático.
- Componentes personalizados:
    - RoundedButton: Botón con bordes redondeados.
    - PlaceholderTextField: Campo de texto (aunque en este caso no usa un placeholder visible).



Código Materias



```java
package proyecto;

//Se importan las librerias a usar
import java.awt.event.ActionEvent;
import java.awt.*;
import javax.swing.*;
import java.awt.event.ActionListener;

//"librerias" personalizadas utilizadas
import elementos.RoundedButton;

public class Materias extends JPanel{
    public Materias() {
        //Configuración del diseño del panel
        setLayout(null);
        setBackground(Color.WHITE);

        //Opciones a mostrar en los comboBox
        String carreras[] = {"Ing. Ambiental", "Ing. Gestión Empresarial", "Ing. Electromecánica", "Ing. Electrónica",
            "Ing. Industrial", "Ing. Mecatrónica", "Ing. Sistemas Computacionales", "Ing. Semiconductores"};
        String materias[] = {"Métodos Numéricos", "Principios Eléctricos", "Ecuaciones Diferenciales", "Sistemas Operativos",
            "Topicos Progamación", "Bases de datos"};
        String semestres[] = {"1° Semestre", "2° Semestre", "3° Semestre", "4° Semestre", "5° Semestre", "6° Semestre", "7° Semestre",
            "8° Semestre", "9° Semestre", "10° Semestre", "11° Semestre", "12° Semestre"};

        //Creación de Labels y comboBox para ingresar datos
        JLabel labelCarrera = new JLabel("Carrera: ");
        labelCarrera.setBounds(20,30, 100, 30);

        JComboBox<String> comboCarrera = new JComboBox<>(carreras);
        comboCarrera.setMaximumRowCount(5);
        comboCarrera.setBounds(90, 30, 200, 30);
        comboCarrera.setBackground(Color.WHITE);

        JLabel labelSemestre = new JLabel("Semestre: ");
        labelSemestre.setBounds(20,80, 100, 30);

        JComboBox<String> comboSemestre = new JComboBox<>(semestres);
        comboSemestre.setMaximumRowCount(5);
        comboSemestre.setBounds(90, 80, 200, 30);
        comboSemestre.setBackground(Color.WHITE);

        JLabel labelMateria = new JLabel("Materia:");
        labelMateria.setBounds(20, 130, 200, 30);

        JComboBox<String> comboMateria = new JComboBox<>(materias);
        comboMateria.setMaximumRowCount(5);
        comboMateria.setBounds(90, 130, 200, 30);
        comboMateria.setBackground(Color.WHITE);

        //Creación del boton para mostrar la elección
        JButton boton = new RoundedButton("Mostrar Selección");
        boton.setBounds(20, 180, 150, 30);
        boton.setBackground(new Color(100, 150, 237));
        boton.setForeground(Color.WHITE);

        //Evento de accion al dar click en el botón
        boton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                //Obtener el item seleccionado del comboBox
                String carrera = (String) comboCarrera.getSelectedItem();
                String semestre = (String) comboSemestre.getSelectedItem();
                String materia = (String) comboMateria.getSelectedItem();
                //Muestra una ventana con la selección
                JOptionPane.showMessageDialog(Materias.this, "Carrera: " + carrera + "\nSemestre: " + semestre+"\nMateria: "+materia, "Información", JOptionPane.INFORMATION_MESSAGE);
            }
        });

        //Se agregan los elementos creados
        add(labelCarrera);
        add(comboCarrera);
        add(labelSemestre);
        add(comboSemestre);
        add(labelMateria);
        add(comboMateria);
        add(boton);
    }
}
```

1. Estructura General

- **Clase**: Materias es una clase pública que hereda de JPanel, lo que la convierte en un panel personalizado.
- **Propósito**: Crear un formulario con tres listas desplegables (JComboBox) para elegir una carrera, un semestre y una materia, y mostrar las selecciones al usuario.
- **Librerías**:
    - javax.swing.* y java.awt.*: Para componentes gráficos y diseño.
    - Clase personalizada (RoundedButton): Un botón con bordes redondeados.

---

2. Constructor (Materias())

- **Función**: Configura el panel y agrega los elementos gráficos.
- **Detalles**:
    - Establece un diseño nulo (setLayout(null)), usando posicionamiento manual.
    - Define el fondo del panel como blanco (Color.WHITE).

---

3. Componentes Gráficos

- Listas Desplegables
    - Carrera:
        - carreras: Arreglo con 8 opciones: "Ing. Ambiental", "Ing. Gestión Empresarial", "Ing. Electromecánica", "Ing. Electrónica", "Ing. Industrial", "Ing. Mecatrónica", "Ing. Sistemas Computacionales", "Ing. Semiconductores".
        - labelCarrera: Etiqueta "Carrera:" en (20, 30), tamaño 100x30 píxeles.
        - comboCarrera: Lista desplegable en (90, 30), tamaño 200x30 píxeles, fondo blanco, máximo 5 opciones visibles (setMaximumRowCount(5)).
    - Semestre:
        - semestres: Arreglo con 12 opciones: "1° Semestre" hasta "12° Semestre".
        - labelSemestre: Etiqueta "Semestre:" en (20, 80), tamaño 100x30 píxeles.
        - comboSemestre: Lista desplegable en (90, 80), tamaño 200x30 píxeles, fondo blanco, máximo 5 opciones visibles.
    - Materia:
        - materias: Arreglo con 6 opciones: "Métodos Numéricos", "Principios Eléctricos", "Ecuaciones Diferenciales", "Sistemas Operativos", "Topicos Programación", "Bases de datos".
        - labelMateria: Etiqueta "Materia:" en (20, 130), tamaño 200x30 píxeles.
        - comboMateria: Lista desplegable en (90, 130), tamaño 200x30 píxeles, fondo blanco, máximo 5 opciones visibles.
- Botón
    - boton: Botón redondeado (RoundedButton) con texto "Mostrar Selección":
        - Ubicado en (20, 180) con tamaño 150x30 píxeles.
        - Fondo azul (100, 150, 237) y texto blanco.

---

4. Lógica del Botón (ActionListener)

- **Función**: Maneja el evento al hacer clic en el botón "Mostrar Selección".
- **Detalles**:
    - Obtiene las opciones seleccionadas de cada JComboBox:
        - carrera de comboCarrera.
        - semestre de comboSemestre.
        - materia de comboMateria.

---

5. Agregar Componentes

- Función: Añade todos los elementos gráficos al panel Materias.
- Componentes agregados: labelCarrera, comboCarrera, labelSemestre, comboSemestre, labelMateria, comboMateria, boton.

---

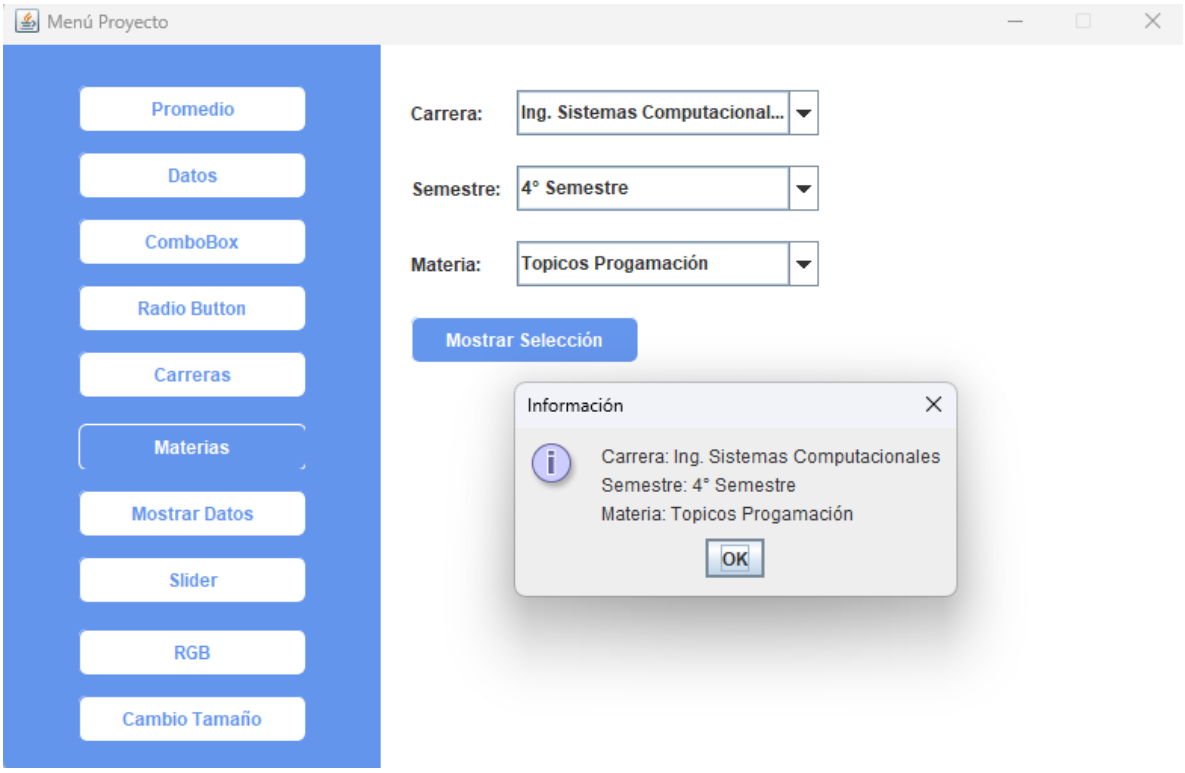
6. Flujo General

1. El panel muestra tres listas desplegables para seleccionar una carrera, un semestre y una materia, junto con un botón.
2. El usuario elige una opción de cada lista.
3. Al hacer clic en "Mostrar Selección", se muestra un mensaje con las opciones seleccionadas.

---

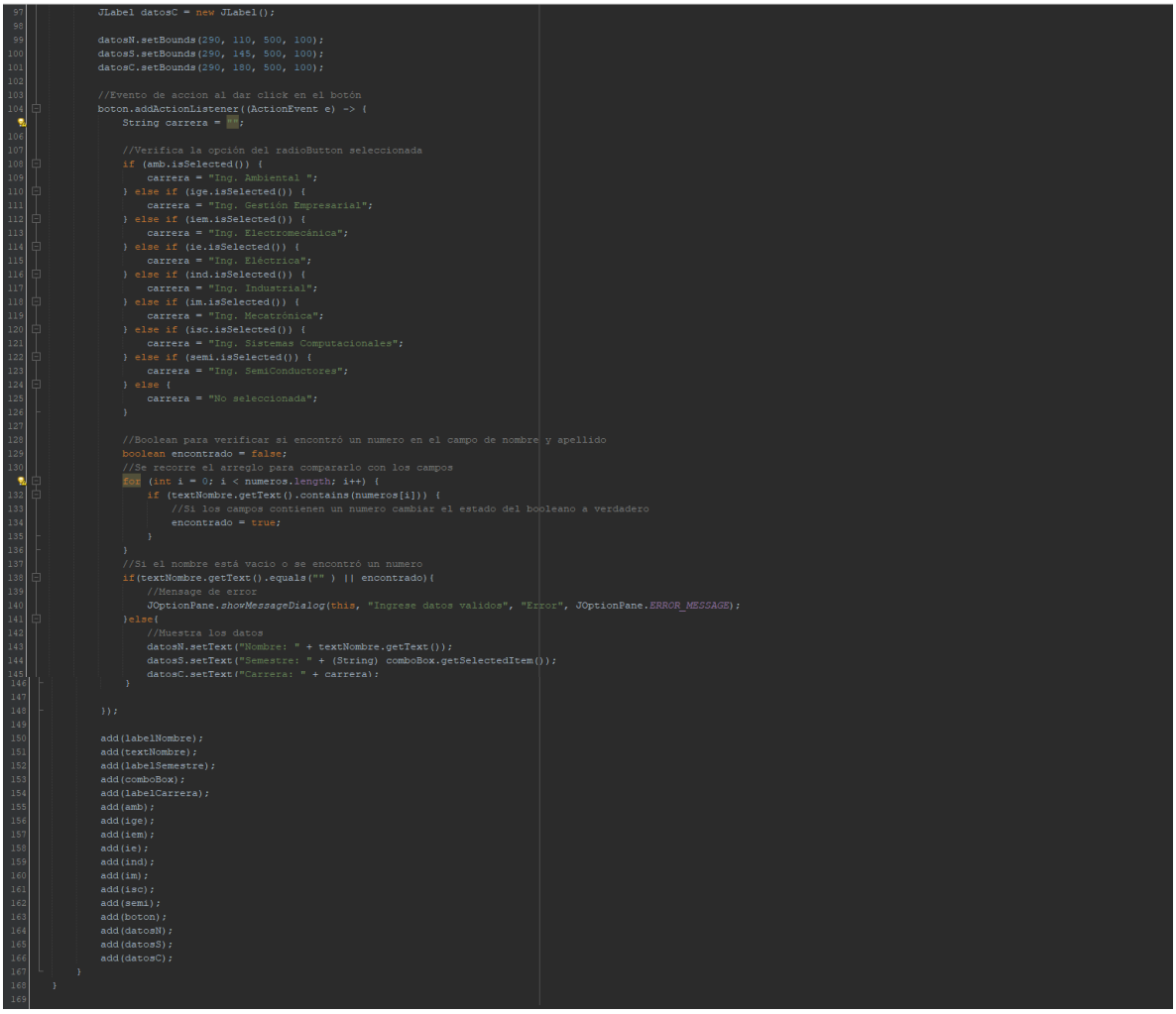
7. Diseño y Estilo

- Colores: Fondo blanco para el panel y los JComboBox, azul (100, 150, 237) para el botón con texto blanco.
- Posicionamiento: Uso de coordenadas manuales (setBounds) para un diseño estático.
- Componente personalizado:
    - RoundedButton: Botón con bordes redondeados.


Menú Proyecto
Promedio
Datos
ComboBox
Radio Button
Carreras
Materias
Mostrar Datos
Slider
RGB
Cambio Tamaño
Carrera:
Ing. Sistemas Computacional...
Semestre:
4° Semestre
Materia:
Topicos Progamación
Mostrar Selección
Información
Carrera: Ing. Sistemas Computacionales
Semestre: 4° Semestre
Materia: Topicos Progamación
OK

Código Mostrar Datos

```java
package proyecto;

//Se importan las librerias a usar
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.*;

//"librerias" personalizadas utilizadas
import elementos.PlaceholderTextField;
import elementos.RoundedButton;

//La clase Promedio heredada de un JPanel. Panel personalizado
public class MostrarDatos extends JPanel{
    public MostrarDatos() {
        //Configuración del diseño del panel
        setLayout(null);
        setBackground(Color.WHITE);

        //Opciones del comboBox
        String[] semestre = {"1° Semestre", "2° Semestre", "3° Semestre", "4° Semestre", "5° Semestre", "6° Semestre", "7° Semestre",
            "8° Semestre", "9° Semestre", "10° Semestre", "11° Semestre", "12° Semestre"};

        //Creación de Labels y Textfields para ingresar datos
        JLabel labelNombre = new JLabel("Nombre: ");
        labelNombre.setBounds(20, 30, 200, 30);

        JTextField textNombre = new PlaceholderTextField("",20);
        textNombre.setBounds(90, 30, 200, 30);

        JLabel labelSemestre = new JLabel("Semestre: ");
        labelSemestre.setBounds(20, 70, 200, 30);

        JComboBox<String> comboBox = new JComboBox<>(semestre);
        comboBox.setMaximumRowCount(5);
        comboBox.setBounds(90, 70, 200, 30);
        comboBox.setBackground(Color.WHITE);

        JLabel labelCarrera = new JLabel("Carrera: ");
        labelCarrera.setBounds(20, 110, 200, 30);

        //Creación de los radioButtons
        JRadioButton amb = new JRadioButton("Ing. Ambiental");
        amb.setBounds(20, 145, 250, 30);
        amb.setBackground(Color.WHITE);

        JRadioButton ige = new JRadioButton("Ing. Gestión Empresarial");
        ige.setBounds(20, 180, 250, 30);
        ige.setBackground(Color.WHITE);

        JRadioButton iem = new JRadioButton("Ing. Electromecánica");
        iem.setBounds(20, 215, 250, 30);
        iem.setBackground(Color.WHITE);

        JRadioButton ie = new JRadioButton("Ing. Electrica");
        ie.setBounds(20, 250, 250, 30);
        ie.setBackground(Color.WHITE);

        JRadioButton ind = new JRadioButton("Ing. Industrial");
        ind.setBounds(20, 285, 250, 30);
        ind.setBackground(Color.WHITE);

        JRadioButton im = new JRadioButton("Ing. Mecatronica");
        im.setBounds(20, 320, 250, 30);
        im.setBackground(Color.WHITE);

        JRadioButton isc = new JRadioButton("Ing. Sistemas Computacionales");
        isc.setBounds(20, 355, 250, 30);
        isc.setBackground(Color.WHITE);

        JRadioButton semi = new JRadioButton("Ing. Semiconductores");
        semi.setBounds(20, 390, 250, 30);
        semi.setBackground(Color.WHITE);

        //agrega todos los radio buttons a un grupo de selección
        ButtonGroup carreras = new ButtonGroup();
        carreras.add(amb);
        carreras.add(ige);
        carreras.add(iem);
        carreras.add(ie);
        carreras.add(ind);
        carreras.add(im);
        carreras.add(isc);
        carreras.add(semi);

        //Creación del boton para mostrar la elección
        JButton boton = new RoundedButton("Mostrar Selección");
        boton.setBounds(20, 435, 150, 30);
        boton.setBackground(new Color(100, 149, 237));
        boton.setForeground(Color.WHITE);

        //Declaración del arreglo para hacer la comparación con el label de nombre
        String[] numeros = new String[]{"1", "2", "3", "4", "5", "6", "7", "8", "9", "0"};

        //Creación de etiquetas para mostrar los elementos en el mismo panel
        JLabel datosN = new JLabel();
        JLabel datosS = new JLabel();
        JLabel datosC = new JLabel();
```



```
        JLabel datosC = new JLabel();

        datosN.setBounds(290, 110, 500, 100);
        datosS.setBounds(290, 145, 500, 100);
        datosC.setBounds(290, 180, 500, 100);

        //Evento de accion al dar click en el botón
        boton.addActionListener((ActionEvent e) -> {
            String carrera = "";

            //Verifica la opción del radioButton seleccionada
            if (amb.isSelected()) {
                carrera = "Ing. Ambiental ";
            } else if (ige.isSelected()) {
                carrera = "Ing. Gestión Empresarial";
            } else if (iem.isSelected()) {
                carrera = "Ing. Electromecánica";
            } else if (ie.isSelected()) {
                carrera = "Ing. Eléctrica";
            } else if (ind.isSelected()) {
                carrera = "Ing. Industrial";
            } else if (im.isSelected()) {
                carrera = "Ing. Mecatrónica";
            } else if (isc.isSelected()) {
                carrera = "Ing. Sistemas Computacionales";
            } else if (semi.isSelected()) {
                carrera = "Ing. SemiConductores";
            } else {
                carrera = "No seleccionada";
            }

            //Boolean para verificar si encontró un numero en el campo de nombre y apellido
            boolean encontrado = false;
            //Se recorre el arreglo para compararlo con los campos
            for (int i = 0; i < numeros.length; i++) {
                if (textNombre.getText().contains(numeros[i])) {
                    //Si los campos contienen un numero cambiar el estado del booleano a verdadero
                    encontrado = true;
                }
            }
            //Si el nombre está vacio o se encontró un numero
            if(textNombre.getText().equals("" ) || encontrado){
                //Mensage de error
                JOptionPane.showMessageDialog(this, "Ingrese datos validos", "Error", JOptionPane.ERROR_MESSAGE);
            }else{
                //Muestra los datos
                datosN.setText("Nombre: " + textNombre.getText());
                datosS.setText("Semestre: " + (String) comboBox.getSelectedItem());
                datosC.setText("Carrera: " + carrera);
            }

        });

        add(labelNombre);
        add(textNombre);
        add(labelSemestre);
        add(comboBox);
        add(labelCarrera);
        add(amb);
        add(ige);
        add(iem);
        add(ie);
        add(ind);
        add(im);
        add(isc);
        add(semi);
        add(boton);
        add(datosN);
        add(datosS);
        add(datosC);
    }
}
```

Te explicaré y describiré este código Java paso a paso. Este código crea un formulario gráfico en Java utilizando Swing para recolectar y mostrar información de un estudiante (nombre, semestre y carrera).

Estructura general

1. **Paquete y Clase**:
    - Se define en el paquete proyecto.
    - La clase MostrarDatos extiende JPanel, lo que significa que es un panel personalizado que contendrá componentes gráficos.
2. **Importaciones**:
    - Usa bibliotecas estándar de Java (javax.swing, java.awt) para la interfaz gráfica.

- Importa clases personalizadas (PlaceholderTextField y RoundedButton) del paquete elementos, que parecen ser componentes personalizados.

Constructor MostrarDatos()

El constructor configura el panel y añade todos los componentes gráficos.

Configuración inicial

- setLayout(null): Usa un diseño absoluto (coordenadas manuales) en lugar de un administrador de diseño automático.
- setBackground(Color.WHITE): Establece el fondo del panel en blanco.

Componentes del formulario

1. **ComboBox para Semestre**:
   - Se crea un arreglo semestre con opciones del 1° al 12° semestre.
   - JComboBox<String> comboBox: Un menú desplegable con las opciones del arreglo.
   - setMaximumRowCount(5): Limita a 5 las opciones visibles antes de mostrar una barra de desplazamiento.
   - Posición: (90, 70, 200, 30).
2. **Campo de Nombre**:
   - JLabel labelNombre: Etiqueta "Nombre:" en (20, 30, 200, 30).
   - JTextField textNombre: Campo de texto con un placeholder (usando PlaceholderTextField), ubicado en (90, 30, 200, 30).
3. **RadioButtons para Carrera**:
   - Se crean 8 JRadioButton para diferentes carreras de ingeniería (Ambiental, Gestión Empresarial, etc.).
   - Cada uno tiene su posición específica (e.g., amb.setBounds(20, 145, 250, 30)).

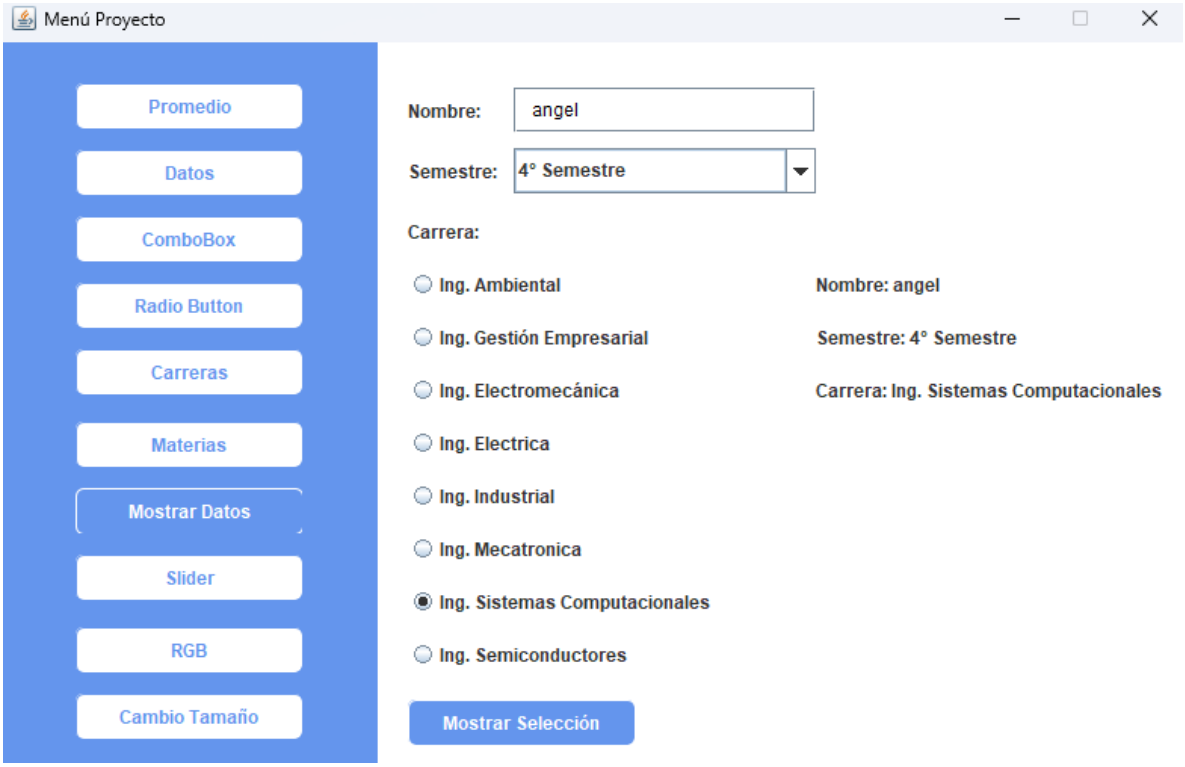
- Todos se agrupan en un ButtonGroup (carreras) para que solo uno pueda estar seleccionado a la vez.

4. **Botón "Mostrar Selección"**:
    - JButton boton: Usa RoundedButton (botón personalizado con bordes redondeados).
    - Color de fondo: azul claro (new Color(100, 149, 237)).
    - Posición: (20, 435, 150, 30).
5. **Etiquetas para mostrar resultados**:
    - Tres JLabel (datosN, datosS, datosC) para mostrar el nombre, semestre y carrera seleccionados.
    - Posicionados a la derecha del formulario (e.g., datosN.setBounds(290, 110, 500, 100)).

Lógica del botón (Evento ActionListener)

El botón tiene un evento que se activa al hacer clic:

1. **Determinación de la carrera**:
    - Verifica cuál JRadioButton está seleccionado con una serie de if-else.
    - Asigna el nombre de la carrera a la variable carrera. Si no hay selección, usa "No seleccionada".
2. **Validación del nombre**:
    - Define un arreglo numeros con dígitos del 0 al 9.
    - Recorre el texto ingresado en textNombre para verificar si contiene números.
    - Si el nombre está vacío ("") o contiene números (encontrado = true), muestra un mensaje de error con JOptionPane.showMessageDialog.
3. **Mostrar datos**:
    - Si la validación pasa, actualiza las etiquetas datosN, datosS y datosC con el nombre, semestre y carrera seleccionados.


Menú Proyecto
Promedio
Datos
ComboBox
Radio Button
Carreras
Materias
Mostrar Datos
Slider
RGB
Cambio Tamaño
Nombre:
angel
Semestre:
4° Semestre
Carrera:
Ing. Ambiental
Ing. Gestión Empresarial
Ing. Electromecánica
Ing. Electrica
Ing. Industrial
Ing. Mecatronica
Ing. Sistemas Computacionales
Ing. Semiconductores
Mostrar Selección
Nombre: angel
Semestre: 4° Semestre
Carrera: Ing. Sistemas Computacionales

Código Slider

```java
package proyecto;

//Se importan las librerias a usar
import javax.swing.*;
import java.awt.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

//Librerias personalizadas utilizadas
import elementos.PlaceholderTextField;

//La clase heredada de un JPanel. Panel personalizado
public class Slider extends JPanel{
    public Slider() {
        //Configuración del diseño del panel
        setLayout(null);
        setBackground(Color.WHITE);

        //Creación y configuración de componentes
        JLabel label = new JLabel("Valor del slider: ");
        label.setBounds(200, 30, 150, 30);
        JSlider slider = new JSlider(0, 100, 50);
        slider.setMajorTickSpacing(10);
        slider.setMinorTickSpacing(5);
        slider.setPaintTicks(true);
        slider.setPaintLabels(true);
        slider.setBounds(20, 60, 500, 100);
        slider.setBackground(Color.WHITE);

        JTextField textField = new PlaceholderTextField("", 5);
        textField.setText("50");
        textField.setBounds(320,30, 50, 30);

        //Creación de checkbox
        JCheckBox checkBox = new JCheckBox("Activar Actualización automática", true);
        checkBox.setBounds(170, 170, 215, 30);
        checkBox.setBounds(20, 60, 500, 100);
        checkBox.setBackground(Color.WHITE);

        //Evento de accion al interactuar con el slider
        slider.addChangeListener(new ChangeListener(){
            @Override
            //Pendiente a cambios en el valor del slider
            public void stateChanged(ChangeEvent e){
                //Condición que se cumple cuando el checkbox está seleccionado
                if(checkBox.isSelected()){
                    int valor = slider.getValue();
                    //Actualiza el valor del slider en texto
                    label.setText("Valor del slider: "+valor);
                    textField.setText(String.valueOf(valor));
                }
            }
        });

        //Cambia el valor del slider
        textField.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                //Intente cambiar el valor del slider
                try{
                    int valor = Integer.parseInt(textField.getText());
                    //Verifica que el valor sea valido
                    if(valor >=0 && valor <=100){
                        slider.setValue(valor);
                    }else{
                        //Muestra una ventana con información
                        JOptionPane.showMessageDialog(Slider.this, "Ingrese un numero entre 0 y 100");
                    }
                // Captura el error si la conversión de String a número falla
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(Slider.this, "Ingrese un numero", "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        add(label);
        add(slider);
        add(textField);
        add(checkBox);
    }
}
```

1. Estructura General

- **Clase**: Slider es una clase pública que hereda de JPanel, lo que la convierte en un panel personalizado.
- **Propósito**: Crear una interfaz interactiva donde un deslizador muestra un valor, que se refleja en una etiqueta y un campo de texto, con la opción de habilitar/deshabilitar la actualización automática mediante una casilla de verificación.
- **Librerías**:
    - javax.swing.* y java.awt.*: Para componentes gráficos y diseño.
    - javax.swing.event.ChangeListener: Para manejar eventos del deslizador.
    - java.awt.event.*: Para manejar eventos del campo de texto.
    - Clase personalizada (PlaceholderTextField): Campo de texto con placeholder.

---

2. Constructor (Slider())

- **Función**: Configura el panel y agrega los elementos gráficos.
- **Detalles**:
    - Establece un diseño nulo (setLayout(null)), usando posicionamiento manual.
    - Define el fondo del panel como blanco (Color.WHITE).

---

3. Componentes Gráficos

- **Etiqueta**
  - label: Etiqueta "Valor del slider:" ubicada en (200, 30) con tamaño 150x30 píxeles. Muestra el valor actual del deslizador.
- **Deslizador (JSlider)**
  - slider:
    - Rango de 0 a 100, valor inicial 50.
    - Marcas mayores cada 10 (setMajorTickSpacing(10)), marcas menores cada 5 (setMinorTickSpacing(5)).
    - Muestra marcas (setPaintTicks(true)) y etiquetas numéricas (setPaintLabels(true)).
    - Ubicado en (20, 60) con tamaño 500x100 píxeles.
    - Fondo blanco (setBackground(Color.WHITE)).
- **Campo de Texto**
  - textField: Campo de texto (PlaceholderTextField) con capacidad para 5 caracteres:
    - Valor inicial "50" (coincide con el deslizador).
    - Ubicado en (320, 30) con tamaño 50x30 píxeles.
    - Sin placeholder visible (se pasa "").
- **Casilla de Verificación**
  - checkBox: Casilla "Activar Actualización automática", marcada por defecto (true):
    - Ubicada en (20, 60) con tamaño 500x100 píxeles (nota: parece haber un error en el código, ya que se reasigna la misma posición que el deslizador; probablemente debería ser diferente, como (170, 170)).
    - Fondo blanco (setBackground(Color.WHITE)).

---

4. Lógica de los Eventos

- **Evento del Deslizador (ChangeListener)**
  - Función: Actualiza la etiqueta y el campo de texto cuando el valor del deslizador cambia, si la casilla está marcada.
  - Detalles:
    - Obtiene el valor del deslizador (slider.getValue()).
    - Si checkBox.isSelected() es true:
      - Actualiza la etiqueta con "Valor del slider: [valor]".
      - Actualiza el campo de texto con el valor convertido a cadena (String.valueOf(valor)).
- **Evento del Campo de Texto (ActionListener)**
  - Función: Permite al usuario ingresar un valor manualmente para ajustar el deslizador.
  - Detalles:
    - Usa un bloque try-catch para manejar errores de conversión.
    - Convierte el texto ingresado (textField.getText()) a entero (Integer.parseInt).
    - Validación:
      - Si el valor está entre 0 y 100: Ajusta el deslizador con slider.setValue(valor).
      - Si está fuera del rango: Muestra un JOptionPane con "Ingrese un numero entre 0 y 100".
    - Error: Si el texto no es un número, muestra un JOptionPane con "Ingrese un numero" y un ícono de error (ERROR_MESSAGE).

---

5. Agregar Componentes

- **Función**: Añade todos los elementos gráficos al panel Slider.
- **Componentes** agregados: label, slider, textField, checkBox.

---

6. Flujo General

1. El panel muestra un deslizador (inicialmente en 50), una etiqueta con el valor, un campo de texto con "50" y una casilla marcada.
2. Con la casilla marcada:
   - Mover el deslizador actualiza automáticamente la etiqueta y el campo de texto.
3. Con la casilla desmarcada:
   - El deslizador no actualiza la etiqueta ni el campo de texto.
4. Campo de texto:
   - Escribir un valor y presionar Enter ajusta el deslizador si es un número válido entre 0 y 100; de lo contrario, muestra un error.

7. Diseño y Estilo

- **Colores**: Fondo blanco para el panel, deslizador y casilla.
- **Posicionamiento**: Uso de coordenadas manuales (setBounds), pero hay un error: el checkBox y el slider tienen la misma posición (20, 60, 500, 100), lo que causaría superposición. Probablemente debería ser (170, 170, 215, 30) como se menciona inicialmente en el código.
- **Componente personalizado:** PlaceholderTextField (sin placeholder visible).

Código RGB

```java
package proyecto;

//Se importan las librerias a usar
import javax.swing.*;
import java.awt.*;
import javax.swing.event.ChangeListener;

//La clase heredada de un JPanel. Panel personalizado
public class RGB extends JPanel {
    //Atributos del código.
    private JSlider redSlider, greenSlider, blueSlider;
    private JCheckBox enableCheckBox;
    private JPanel colorPanel;

    public RGB() {
        //Configuración del diseño del panel
        setLayout(new BorderLayout());
        setBackground(Color.WHITE);

        // Panel para mostrar el color (centro)
        colorPanel = new JPanel();
        colorPanel.setBackground(Color.BLACK);
        colorPanel.setPreferredSize(new Dimension(200, 200));
        add(colorPanel, BorderLayout.CENTER);

        // Sliders RGB
        redSlider = new JSlider(0, 255, 0);
        greenSlider = new JSlider(0, 255, 0);
        blueSlider = new JSlider(0, 255, 0);

        // Configuración de los sliders
        configureSlider(redSlider);
        configureSlider(greenSlider);
        configureSlider(blueSlider);

        // Checkbox
        enableCheckBox = new JCheckBox("Habilitar", true);
        enableCheckBox.setBackground(Color.WHITE);
        enableCheckBox.addItemListener(e -> updateColor());

        // Panel de controles (abajo derecha)
        JPanel controlPanel = new JPanel();
        controlPanel.setBackground(Color.WHITE);
        controlPanel.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(5, 5, 5, 5); // Espaciado

        // Agregar componentes al panel de control
        gbc.gridx = 0;
        gbc.gridy = 0;
        controlPanel.add(new JLabel("Rojo:"), gbc);
        gbc.gridy = 1;
        controlPanel.add(new JLabel("Verde:"), gbc);
        gbc.gridy = 2;
        controlPanel.add(new JLabel("Azul:"), gbc);

        gbc.gridx = 1;
        gbc.gridy = 0;
        controlPanel.add(redSlider, gbc);
        gbc.gridy = 1;
        controlPanel.add(greenSlider, gbc);
        gbc.gridy = 2;
        controlPanel.add(blueSlider, gbc);

        gbc.gridx = 0;
        gbc.gridy = 3;
        gbc.gridwidth = 2;
        controlPanel.add(enableCheckBox, gbc);

        // Panel contenedor para alinear abajo-derecha
        JPanel southPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));
        southPanel.setBackground(Color.WHITE);
        southPanel.add(controlPanel);
        add(southPanel, BorderLayout.SOUTH);

        // Listeners para los sliders
        ChangeListener slideListener = e -> updateColor();
        redSlider.addChangeListener(slideListener);
        redSlider.setBackground(Color.WHITE);
        greenSlider.addChangeListener(slideListener);
        greenSlider.setBackground(Color.WHITE);
        blueSlider.addChangeListener(slideListener);
        blueSlider.setBackground(Color.WHITE);
    }

    //Configuración del slider
    private void configureSlider(JSlider slider) {
        slider.setMajorTickSpacing(50);
        slider.setPaintTicks(true);
        slider.setPaintLabels(true);
        slider.setPreferredSize(new Dimension(150, 50));
    }

    //Actualiza el color del panel
    private void updateColor() {
        //Verifica si está seleccionado el checkbox para hacer el cambio de color
        if (enableCheckBox.isSelected()) {
            int red = redSlider.getValue();
            int green = greenSlider.getValue();
            int blue = blueSlider.getValue();
            colorPanel.setBackground(new Color(red, green, blue));
        } else {
            //Color negro si no está activo el checkbox
            colorPanel.setBackground(Color.BLACK);
        }
    }
}
```

1. Estructura General

- **Clase**: RGB es una clase pública que hereda de JPanel, lo que la convierte en un panel personalizado.
- **Propósito**: Crear un selector de colores RGB interactivo con deslizadores y una casilla de verificación, mostrando el color resultante en tiempo real.
- **Librerías**:
    - javax.swing.* y java.awt.*: Para componentes gráficos y diseño.
    - javax.swing.event.ChangeListener: Para manejar eventos de los deslizadores.
- **Atributos**:
    - redSlider, greenSlider, blueSlider: Deslizadores para ajustar los valores de rojo, verde y azul (0-255).
    - enableCheckBox: Casilla de verificación para habilitar/deshabilitar los cambios de color.
    - colorPanel: Panel que muestra el color resultante.

---

2. Constructor (RGB())

- **Función**: Configura el panel y organiza los componentes gráficos.
- **Detalles**:
  - Usa BorderLayout como diseño principal (en lugar de null como en ejemplos anteriores).
  - Establece el fondo del panel como blanco (Color.WHITE).
- **Componentes Gráficos**
  - Panel de Color (colorPanel):
    - Ubicado en el centro (BorderLayout.CENTER).
    - Fondo inicial negro (Color.BLACK).
    - Tamaño preferido de 200x200 píxeles.
  - Deslizadores (JSlider):
    - redSlider, greenSlider, blueSlider: Rango de 0 a 255, valor inicial 0.
    - Configurados con configureSlider() para mostrar marcas y etiquetas.
  - Casilla de Verificación (enableCheckBox):
    - Texto "Habilitar", seleccionada por defecto (true).
    - Fondo blanco.
    - Listener (addItemListener) para actualizar el color al cambiar su estado.
  - Panel de Controles (controlPanel):
    - Usa GridBagLayout para organizar etiquetas y deslizadores.
    - Contiene:
      - Etiquetas "Rojo:", "Verde:", "Azul:" (columna izquierda).
      - Deslizadores correspondientes (columna derecha).
      - Casilla de verificación debajo, abarcando ambas columnas.
    - Fondo blanco.
  - Panel Sur (southPanel):
    - Usa FlowLayout para centrar controlPanel.
    - Ubicado en la parte inferior (BorderLayout.SOUTH).
- **Eventos**
  - Listener de los deslizadores:
    - Un ChangeListener común (slideListener) llama a updateColor() cuando se ajusta cualquier deslizador.
    - Fondo de los deslizadores establecido como blanco.

---

3. Método configureSlider(JSlider slider)

- **Función:** Configura las propiedades visuales de los deslizadores.
- **Detalles:**
  - Espaciado de marcas principales cada 50 unidades (setMajorTickSpacing(50)).
  - Muestra marcas (setPaintTicks(true)) y etiquetas numéricas (setPaintLabels(true)).
  - Tamaño preferido de 150x50 píxeles.

---

4. Método updateColor()

- **Función**: Actualiza el color del colorPanel basado en los valores de los deslizadores y el estado de la casilla de verificación.
- **Detalles**:
  - Si enableCheckBox está seleccionado:
    - Obtiene los valores de redSlider, greenSlider y blueSlider (0-255).
    - Crea un nuevo color con new Color(red, green, blue) y lo aplica al colorPanel.
  - Si no está seleccionado:
    - Establece el fondo del colorPanel como negro (Color.BLACK).

---

5. Flujo General

1. El panel muestra un área de color (inicialmente negra) en el centro y un panel de controles en la parte inferior.
2. El usuario ajusta los deslizadores de rojo, verde y azul (0-255).
3. Si la casilla "Habilitar" está marcada, el color del colorPanel cambia en tiempo real según los valores de los deslizadores.
4. Si la casilla se desmarca, el color vuelve a negro, ignorando los deslizadores.

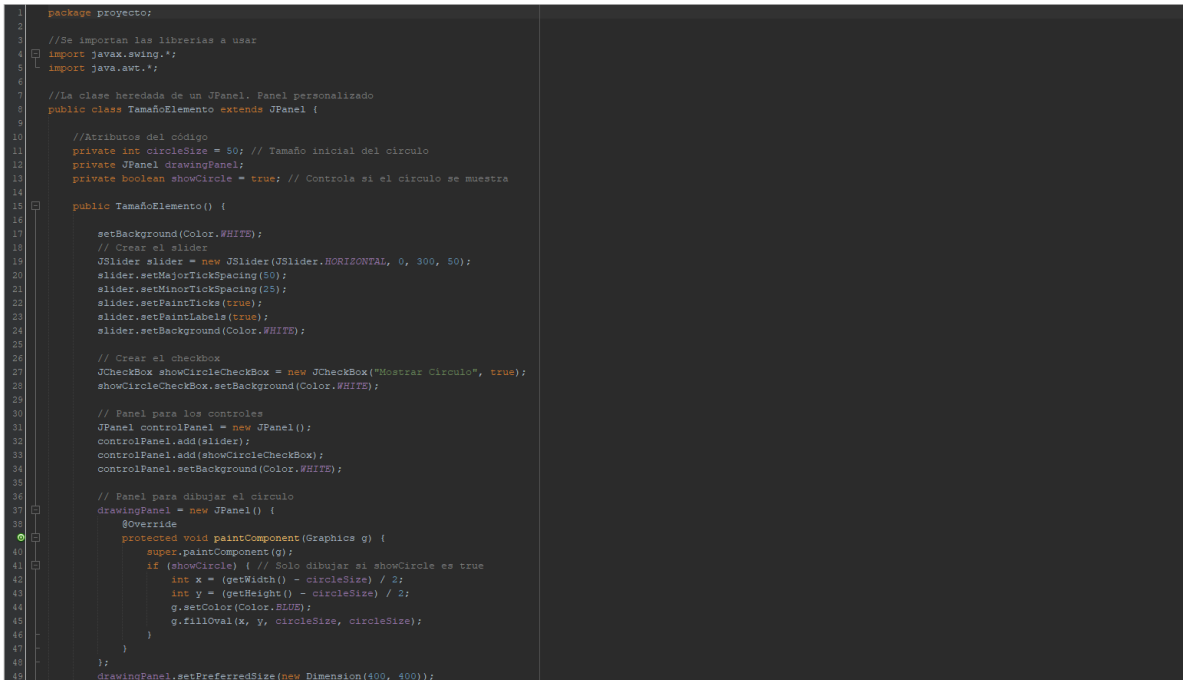
---

6. Diseño y Estilo

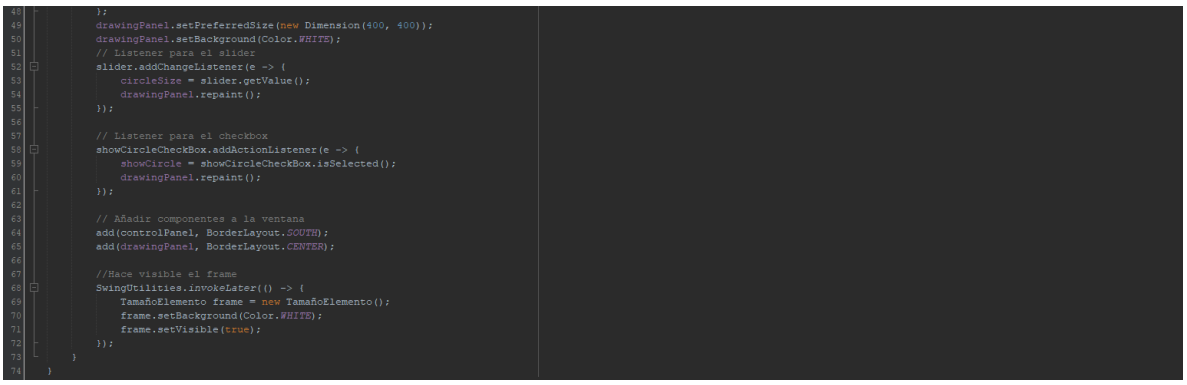
- Colores: Fondo blanco para el panel principal, los deslizadores y los subpaneles; negro inicial para el colorPanel.
- Diseño: Uso de BorderLayout y GridBagLayout para un diseño más estructurado y adaptable que en ejemplos anteriores con null layout.
- Interactividad: Los deslizadores tienen marcas y etiquetas para facilitar la selección de valores.


Menú Proyecto
Promedio
Datos
ComboBox
Radio Button
Carreras
Materias
Mostrar Datos
Slider
RGB
Cambio Tamaño
Rojo:
0 50 100 150 200 250
Verde:
0 50 100 150 200 250
Azul:
0 50 100 150 200 250
Habilitar

Código Tamaño Elemento



```java
package proyecto;

//Se importan las librerias a usar
import javax.swing.*;
import java.awt.*;

//La clase heredada de un JPanel. Panel personalizado
public class TamañoElemento extends JPanel {

    //Atributos del código
    private int circleSize = 50; // Tamaño inicial del circulo
    private JPanel drawingPanel;
    private boolean showCircle = true; // Controla si el circulo se muestra

    public TamañoElemento() {

        setBackground(Color.WHITE);
        // Crear el slider
        JSlider slider = new JSlider(JSlider.HORIZONTAL, 0, 300, 50);
        slider.setMajorTickSpacing(50);
        slider.setMinorTickSpacing(25);
        slider.setPaintTicks(true);
        slider.setPaintLabels(true);
        slider.setBackground(Color.WHITE);

        // Crear el checkbox
        JCheckBox showCircleCheckBox = new JCheckBox("Mostrar Circulo", true);
        showCircleCheckBox.setBackground(Color.WHITE);

        // Panel para los controles
        JPanel controlPanel = new JPanel();
        controlPanel.add(slider);
        controlPanel.add(showCircleCheckBox);
        controlPanel.setBackground(Color.WHITE);

        // Panel para dibujar el circulo
        drawingPanel = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                if (showCircle) { // Solo dibujar si showCircle es true
                    int x = (getWidth() - circleSize) / 2;
                    int y = (getHeight() - circleSize) / 2;
                    g.setColor(Color.BLUE);
                    g.fillOval(x, y, circleSize, circleSize);
                }
            }
        };
        drawingPanel.setPreferredSize(new Dimension(400, 400));
```



```java
        };
        drawingPanel.setPreferredSize(new Dimension(400, 400));
        drawingPanel.setBackground(Color.WHITE);
        // Listener para el slider
        slider.addChangeListener(e -> {
            circleSize = slider.getValue();
            drawingPanel.repaint();
        });

        // Listener para el checkbox
        showCircleCheckBox.addActionListener(e -> {
            showCircle = showCircleCheckBox.isSelected();
            drawingPanel.repaint();
        });

        // Añadir componentes a la ventana
        add(controlPanel, BorderLayout.SOUTH);
        add(drawingPanel, BorderLayout.CENTER);

        //Hace visible el frame
        SwingUtilities.invokeLater(() -> {
            TamañoElemento frame = new TamañoElemento();
            frame.setBackground(Color.WHITE);
            frame.setVisible(true);
        });
    }
}
```

1. Estructura General

- **Clase**: TamañoElemento es una clase pública que hereda de JPanel, lo que la convierte en un panel personalizado.
- **Propósito**: Crear una interfaz interactiva donde el usuario puede ajustar el tamaño de un círculo (de 0 a 300 píxeles) y decidir si se muestra o no.
- **Librerías**:
    - javax.swing.* y java.awt.*: Para componentes gráficos y diseño.
- **Atributos**:
    - circleSize: Entero que define el tamaño inicial del círculo (50 píxeles).
    - drawingPanel: Panel donde se dibuja el círculo.
    - showCircle: Booleano que controla si el círculo es visible (true por defecto

2. Constructor (TamañoElemento())

- **Función**: Configura el panel y organiza los componentes gráficos.
- **Detalles**:
    - Establece el fondo del panel como blanco (setBackground(Color.WHITE)).
    - Usa BorderLayout como diseño principal (aunque no se especifica explícitamente en setLayout, se infiere por el uso de BorderLayout.SOUTH y CENTER).
- **Componentes Gráficos:**
    - Deslizador (JSlider):
        - slider: Deslizador horizontal con rango de 0 a 300, valor inicial 50.
        - Marcas mayores cada 50 (setMajorTickSpacing(50)), marcas menores cada 25 (setMinorTickSpacing(25)).
        - Muestra marcas (setPaintTicks(true)) y etiquetas numéricas (setPaintLabels(true)).
        - Fondo blanco (setBackground(Color.WHITE)).
    - Casilla de Verificación (JCheckBox):
        - showCircleCheckBox: Casilla "Mostrar Círculo", marcada por defecto (true).
        - Fondo blanco (setBackground(Color.WHITE)).
    - Panel de Controles (controlPanel):
        - Contiene el deslizador y la casilla.
        - Fondo blanco (setBackground(Color.WHITE)).
        - Usa diseño predeterminado (FlowLayout implícito).
    - Panel de Dibujo (drawingPanel):
        - Panel personalizado que sobrescribe paintComponent para dibujar el círculo.
        - Tamaño preferido de 400x400 píxeles.
    - Fondo blanco (setBackground(Color.WHITE)).
    - Lógica de dibujo:
        - Si showCircle es true, dibuja un círculo azul centrado con tamaño circleSize.
        - Usa g.fillOval(x, y, circleSize, circleSize) para dibujar el círculo, calculando x e y para centrarlo.
- **Eventos**
    - Listener del Deslizador:
        - slider.addChangeListener: Actualiza circleSize con el valor del deslizador y repinta drawingPanel (repaint()).
    - Listener de la Casilla:
        - showCircleCheckBox.addActionListener: Cambia showCircle según el estado de la casilla y repinta drawingPanel.

- **Inicialización**
    - Nota crítica: Hay un problema en el código. Dentro del constructor, se intenta crear y mostrar una nueva instancia de TamañoElemento con SwingUtilities.invokeLater. Esto es incorrecto porque:
        - TamañoElemento es un JPanel, no un JFrame, y no debería manejarse como una ventana independiente aquí.
        - Esto crea un bucle innecesario y no funciona como se espera. Probablemente debería eliminarse y manejarse en una clase principal con un JFrame.

---

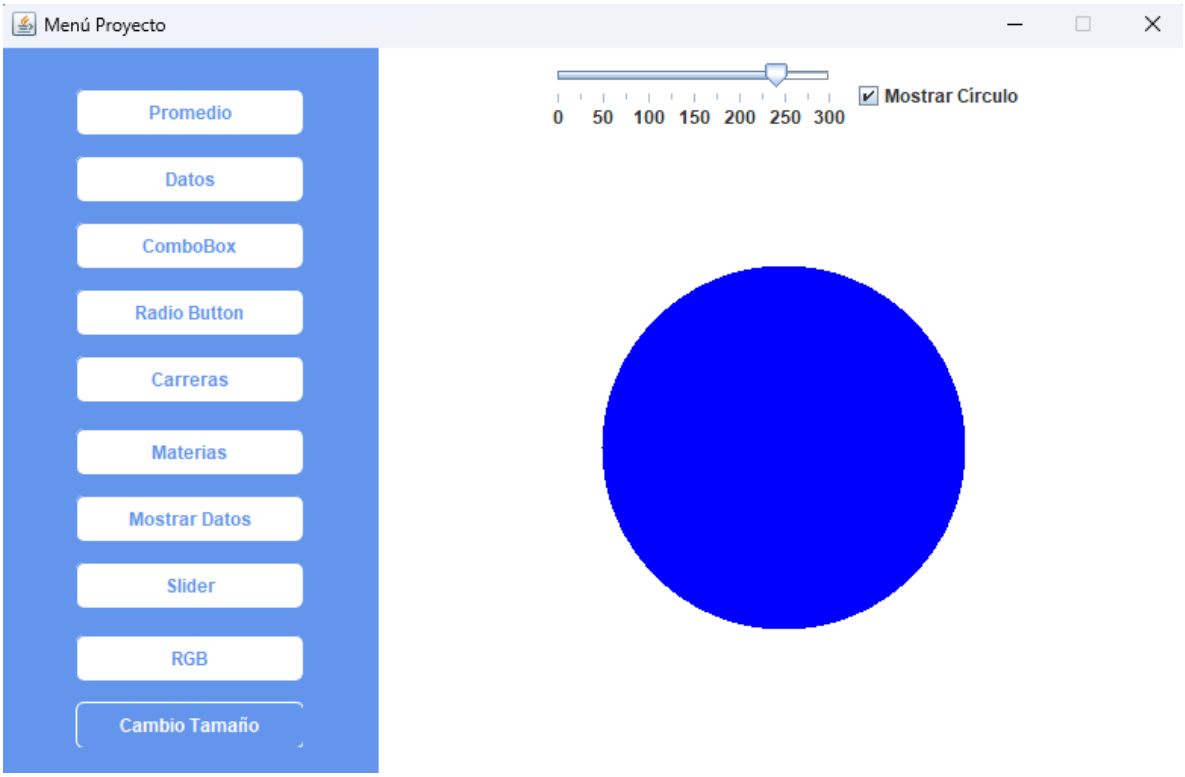
3. Flujo General

1. El panel muestra un círculo azul de 50x50 píxeles centrado en un área de 400x400, con un deslizador y una casilla debajo.
2. Ajustar tamaño:
    - Mover el deslizador cambia el tamaño del círculo en tiempo real (0 a 300 píxeles).
3. Mostrar/Ocultar:
    - Marcar/desmarcar la casilla muestra u oculta el círculo.
4. El círculo se redibuja automáticamente con cada cambio (repaint()).

---

4. Diseño y Estilo

- Colores: Fondo blanco para todos los componentes; círculo azul (Color.BLUE).
- Diseño: Usa BorderLayout para organizar el panel de dibujo en el centro y los controles en la parte inferior.
- Interactividad: El deslizador tiene marcas y etiquetas para facilitar la selección.


Menú Proyecto
Promedio
Datos
ComboBox
Radio Button
Carreras
Materias
Mostrar Datos
Slider
RGB
Cambio Tamaño
0 50 100 150 200 250 300
Mostrar Círculo

Clase PlaceholderPassField

¿Qué hace un "placeholder" en un campo de contraseña?

Un "placeholder" (en español, marcador de posición) es un texto que aparece dentro de un campo de entrada en una página web, como un formulario, para darle al usuario una pista sobre qué debe escribir. En el caso de un campo de contraseña

 Qué **hace:** Muestra el texto "Ingresa tu contraseña" dentro del campo antes de que el usuario escriba algo. Cuando el usuario empieza a teclear, ese texto desaparece.

 Para **qué sirve:** Ayuda a que el usuario entienda qué tipo de información debe poner ahí (en este caso, una contraseña).

Clase PlaceholderTextField

PlaceholderTextField es una clase personalizada que probablemente extiende o mejora la funcionalidad de un JTextField de Java Swing. Su propósito principal es mostrar un texto de "placeholder" (un texto de ayuda o sugerencia) dentro del campo de texto cuando está vacío. Este texto sirve para indicar al usuario qué tipo de información debe ingresar, y suele desaparecer cuando el usuario comienza a escribir. Es una característica que no está incluida de forma nativa en JTextField, por lo que se crea esta clase para añadirla y mejorar la experiencia del usuario en la interfaz gráfica.

Clase RoundedButton

Ayuda a poner redondo el boerde de los botones

Conclusión

La creación de esta interfaz gráfica permitió profundizar en el diseño e implementación de interfaces dentro de un mismo sistema, resaltando la importancia de la organización y reutilización de componentes visuales. Durante el desarrollo, se abordaron conceptos clave de Tópicos de Programación, aplicándolos en un entorno práctico y funcional. Como resultado, se logró una interfaz intuitiva y adaptable, sentando las bases para proyectos más complejos en el futuro.

En conclusión, este proyecto representa una oportunidad para aplicar y consolidar los conocimientos adquiridos en la materia de Tópicos de Programación, promoviendo el desarrollo de habilidades prácticas en la implementación de interfaces gráficas. Al final del trabajo, se espera obtener un sistema funcional que integre diversos componentes gráficos y facilite la interacción del usuario con el sistema, contribuyendo así al desarrollo de aplicaciones más eficientes y amigables.

Asimismo, se experimentó con diversas técnicas de optimización en la gestión de eventos y actualización de la interfaz, lo que permitió mejorar la fluidez y el tiempo de respuesta del sistema. Esto resultó en una aplicación más robusta, adaptable y preparada para escalar en complejidad en futuros proyectos. El proceso de prueba y ajuste fue fundamental para garantizar la usabilidad del sistema, permitiendo detectar áreas de mejora y refinar la interacción con el usuario.