UCSM Esc. Prof. de Ingeniería de Sistemas

INFORME DE PRÁCTICAS

Noviembre - 2024

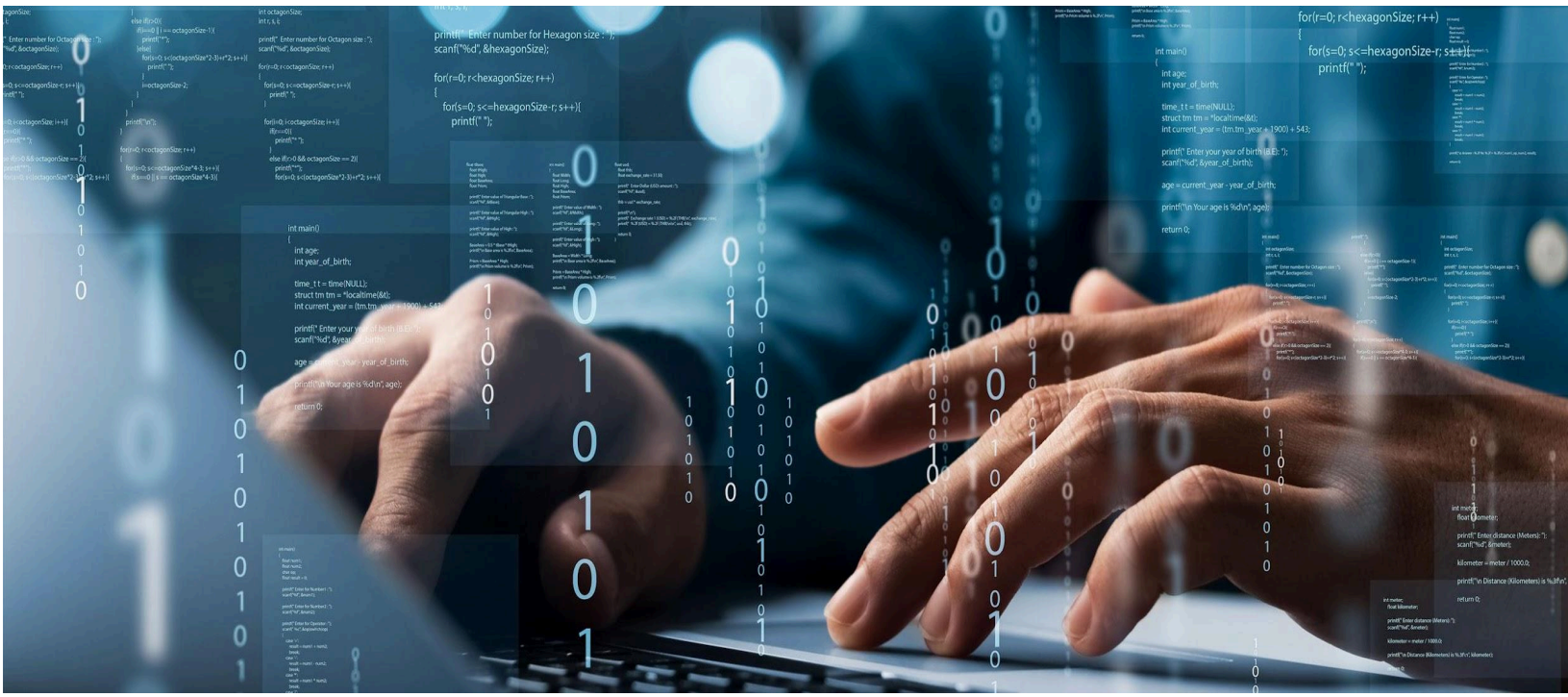
LENGUAJES DE PROGRAMACIÓN III

Práctica N° 09:

JAVA SWING BÁSICO: INTRODUCCIÓN Y CONTROLES

Elaborado por:

DIAZ ACOSTA KAHORI FERNANDA



GRUPO N° 05

LENGUAJES DE PROGRAMACIÓN

Presentado por:

72019650	DIAZ ACOSTA KAHORI FERNANDA	100%

ÍNDICE

1. ACTIVIDADES

1.1 Según el código visto en el marco teórico implementen tres ventanas que utilicen diferentes tipos de administradores de esquemas FlowLayout, BorderLayout y GridLayout.

1.2 Para la implementación de cada uno de los administradores, creen ventanas con títulos diferentes, agréguenles diferentes funcionalidades según los temas vistos en el marco teórico.

ENLACE GITHUB:
https://github.com/KahoriDiazUCSM/Laboratorio9_/tree/main/ACTIVIDAD1

Ventana Inicio

- **Descripción:** En esta ventana se distribuyen dos botones: Ingresar y Registrarse, utilizando el esquema de diseño **FlowLayout.**

- **super("Ingresar o Registrarse"):** Asigna el título de la ventana como "Ingresar o Registrarse".

- **Componentes:**
  - **Botón Ingresar:** Se implementa utilizando JButton.
  - **Botón Registrarse:** Se implementa utilizando JButton.

- **Detalles de los botones:**
  - **Botón Ingresar:**
    + Tiene un ActionListener que se ejecuta cuando el botón es presionado.
    + Abre una instancia de la clase **VentanaIngresar**, mostrando la **ventana de inicio de sesión.**
  - **Botón Registrarse:**
    + Tiene un ActionListener que se ejecuta cuando el botón es presionado.
    + Abre una instancia de la clase **VentanaRegistrarse**, mostrando la **ventana de registro.**

- **Diseño:**
  Utiliza FlowLayout, para organizar los componentes en una fila horizontal que se ubicará en el centro.

- **Configuraciones adicionales:**
- **setSize(300, 100):** Tamaño de la ventana.

- **setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE):** Finaliza la aplicación al cerrar esta ventana.

```
/*************************************************************
* NOMBRE : VentanaInicio.java
* DESCRIPCIO´N: Clase para Ventana de Inicio
*************************************************************/
package actividad;

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


public class VentanaInicio extends JFrame {
    private JButton botonIngresar; -> Botón Ingresar usa JButton
    private JButton botonRegistrarse; -> Botón Registrarse usa JButton

    public VentanaInicio() {
        super("Ingresar o Registrarse"); -> Nombre de la ventana

        setLayout(new FlowLayout());


        //Botón Ingresar - > Evento Ingresar-Ventana Ingresar

        botonIngresar = new JButton("Ingresar");
        botonIngresar.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent EventoIngresar) {
                VentanaIngresar ventanaIngresar1 = new VentanaIngresar();
                ventanaIngresar1.setVisible(true);
            }
        }
        );
        add(botonIngresar);


        //Botón Registrarse- > Evento Registrarse -Ventana Registrarse

        botonRegistrarse = new JButton("Registrarse");
        botonRegistrarse.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent EventoRegistrarse) {
                VentanaRegistrarse ventanaRegistrarse1 = new VentanaRegistrarse();
                ventanaRegistrarse1.setVisible(true);
            }
        }
        );
        add(botonRegistrarse);

        setSize(500, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Ventana Ingresar

- **Descripción:** En esta ventana se distribuyen campos para ingresar el usuario y la contraseña, junto con un botón Entrar, utilizando el esquema de diseño **GridLayout**.

- **super("Ingresar"):** Asigna el título de la ventana como "Ingresar".

- **Componentes:**
  - **Etiqueta Usuario:** Muestra el texto "Usuario:".
  - **Campo de Usuario:** Implementado con JTextField, permite al usuario ingresar su nombre de usuario.
  - **Etiqueta Contraseña:** Muestra el texto "Contraseña:".
  - **Campo de Contraseña:** Implementado con JPasswordField, permite ingresar una contraseña.
  - **Botón Entrar:** Implementado con JButton.

- **Diseño:**
  Utiliza **GridLayout**, organizando los componentes en 6 filas y 2 columnas.

- **Configuraciones adicionales:**

  setSize(300, 150): Tamaño de la ventana**.**

  setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE): Finaliza la aplicación al cerrar esta ventana.

```
/*********************************************************
* NOMBRE : VentanaIngresaar.java
* DESCRIPCIO´N: Ventana que se abre al accionar el boton Ingresar
*********************************************************/
package actividad;

import javax.swing.*;
import java.awt.GridLayout;

public class VentanaIngresar extends JFrame {
    private JTextField campoUsuario; -> Campo de texto para ingresar USUARIO
    private JPasswordField campoContrasena;-> Campo para ingresar CONTRASEÑA
    private JButton btnEntrar; -> Botón ENTRAR

    public VentanaIngresar() {
        super("Ingresar");  -> Nombre de la ventana

        setLayout(new GridLayout(6, 2)); -> GridLayout con 6 filas y 2 columnas para
distribuir etiquetas, campos de texto y un botón

        add(new JLabel("Usuario:")); -> Etiqueta que muestra la palabra usuario
```

```
        campoUsuario = new JTextField();
        add(campoUsuario);

        add(new JLabel("Contraseña:")); -> Etiqueta que muestra la palabra contraseña
        campoContrasenia = new JPasswordField();
        add(campoContrasenia);

        // Botón de entrar
        btnEntrar = new JButton("Entrar");
        add(btnEntrar);

        setSize(500, 300);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }
}
```

Ventana Registrarse

Descripción:

En esta ventana se distribuyen campos para registrar un nuevo usuario y una nueva contraseña, junto con un botón Registrar, utilizando el esquema de diseño BorderLayout y GridLayout.

super("Registrarse"): Asigna el título de la ventana como "Registrarse".

Componentes:

- **Etiqueta Nuevo Usuario:** Muestra el texto "Nuevo Usuario:".
- **Campo de Usuario:** Implementado con JTextField, permite ingresar el nuevo nombre de usuario.
- **Etiqueta Nueva Contraseña:** Muestra el texto "Nueva Contraseña:".
- **Campo de Contraseña:** Implementado con JPasswordField, permite ingresar una nueva contraseña.
- **Botón Registrar:** Implementado con JButton, colocado en la parte inferior de la ventana.

Diseño:

BorderLayout:

Divide la ventana en regiones. Aquí, se utiliza:

Centro: Un panel central que utiliza un GridLayout para organizar las etiquetas y campos en una cuadrícula de 2 filas por 2 columnas.

Sur: Un botón Registrar en la parte inferior.

GridLayout:

Organiza los componentes del panel central.

Configuraciones adicionales:

setSize(300, 150): tamaño de la ventana.

setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE): Finaliza la aplicación al cerrar esta ventana.

```
/*************************************************************
* NOMBRE : VentanaRegistrarse.java
* DESCRIPCIÓN: Ventana que se abre al accionar el boton Registrarse
*************************************************************/
package actividad;

import javax.swing.*;
import java.awt.BorderLayout;
import java.awt.GridLayout;

public class VentanaRegistrarse extends JFrame {
    private JTextField campoUsuario; -> Campo de texto para ingresar USUARIO
    private JPasswordField campoContrasena; -> Campo para ingresar CONTRASEÑA
    private JButton botonRegistrarse; -> Botón Registrarse

    public VentanaRegistrarse() {
        super("Registrarse"); -> Nombre de la ventana

        setLayout(new BorderLayout());

        // Ingresar Uusuario y contraseña - parte central
        JPanel panelCentral = new JPanel(new GridLayout(2, 2));
        panelCentral.add(new JLabel("Nuevo Usuario:")); Etiqueta que muestra la palabra
Nuevo usuario
        campoUsuario = new JTextField();
        panelCentral.add(campoUsuario);

        panelCentral.add(new JLabel("Nueva Contraseña:"));
        campoContrasena = new JPasswordField();
        panelCentral.add(campoContrasena);
        add(panelCentral, BorderLayout.CENTER); -> centro

        // Botón de registrar - parte inferior organización
        botonRegistrarse = new JButton("Registrar");
        add(botonRegistrarse, BorderLayout.SOUTH);-> sur, inferior

        setSize(500, 300);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }
}
```

1.3 Hagan capturas de pantalla por cada implementación de administrador de esquema de tal forma que se evidencie cómo se organiza el contenido y las interacciones que se puede realizar en cada ventana.

Prueba

Es la clase principal del programa.

Contiene el método main, que es el punto de entrada de la aplicación.

VentanaInicio ventanaInicio = new VentanaInicio();:

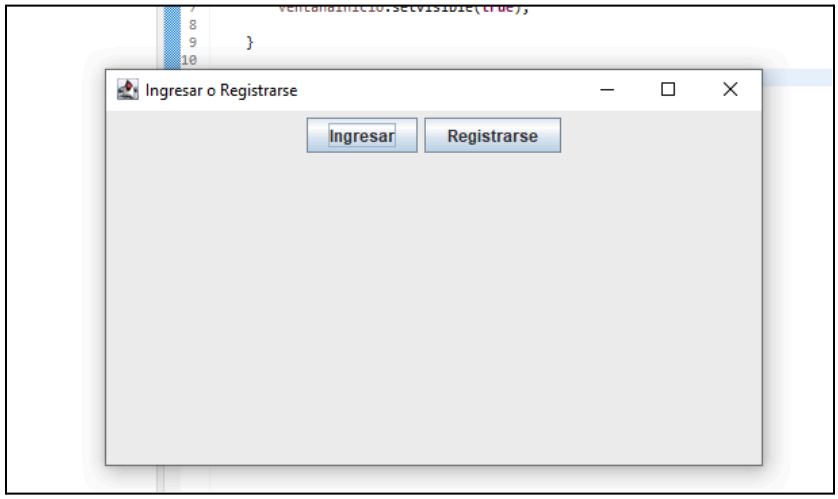
Crea una instancia de la clase **VentanaInicio** - ventana principal.

ventanaInicio.setVisible(true);:

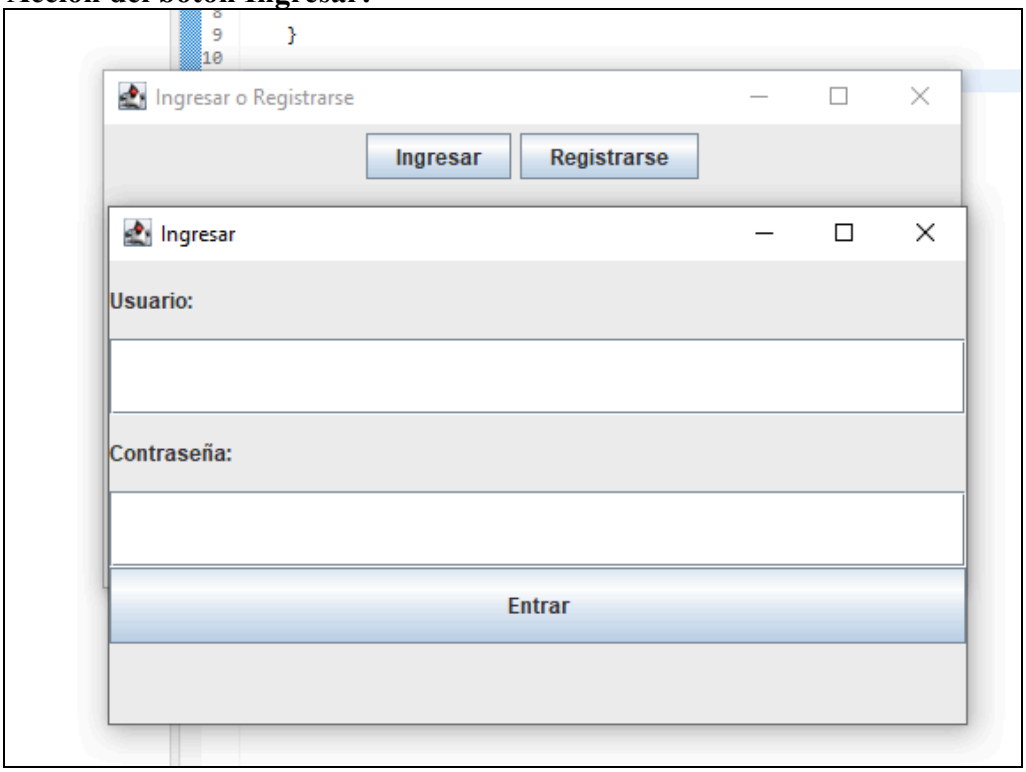
Hace visible la ventana **VentanaInicio**.

```
/*************************************************************
* NOMBRE : Prueba.java
* DESCRIPCIÓN: Prueba del programa
*************************************************************/
package actividad;
import javax.swing.*;

public class Prueba {
    public static void main(String[] args) {
        VentanaInicio ventanaInicio = new VentanaInicio();
        ventanaInicio.setVisible(true);

    }

}
```

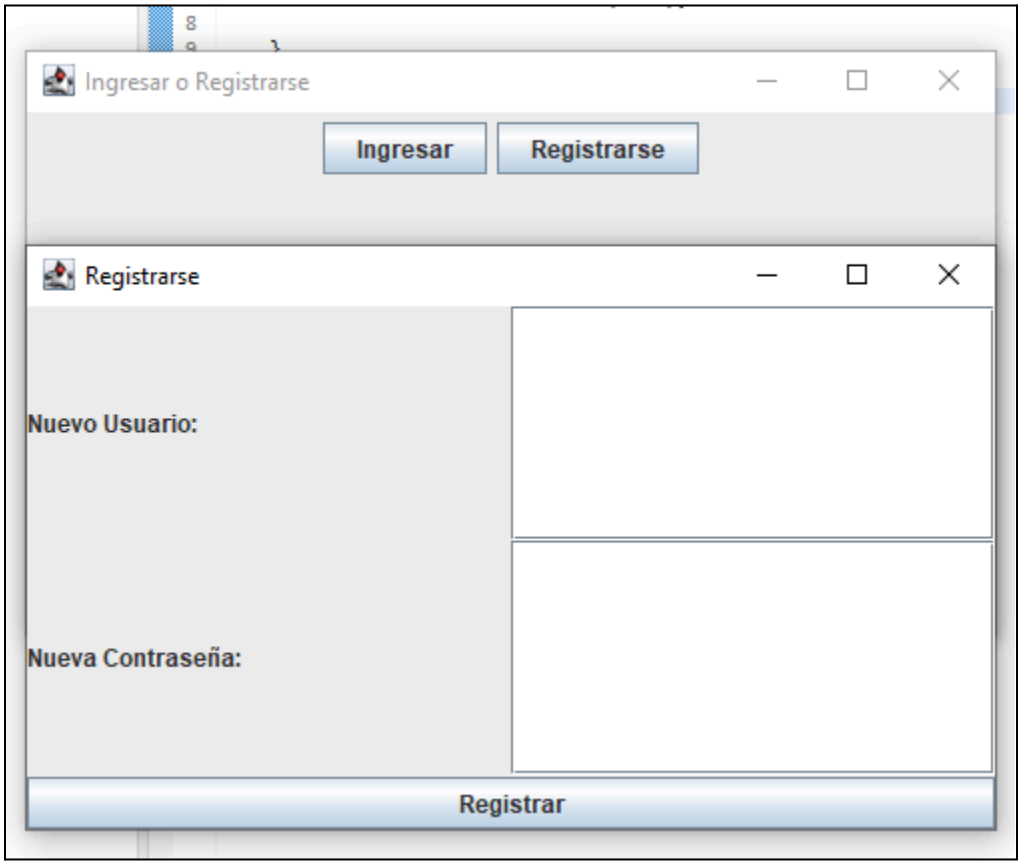
RESULTADO:



Acción del botón Ingresar:



Acción del botón Registrarse:



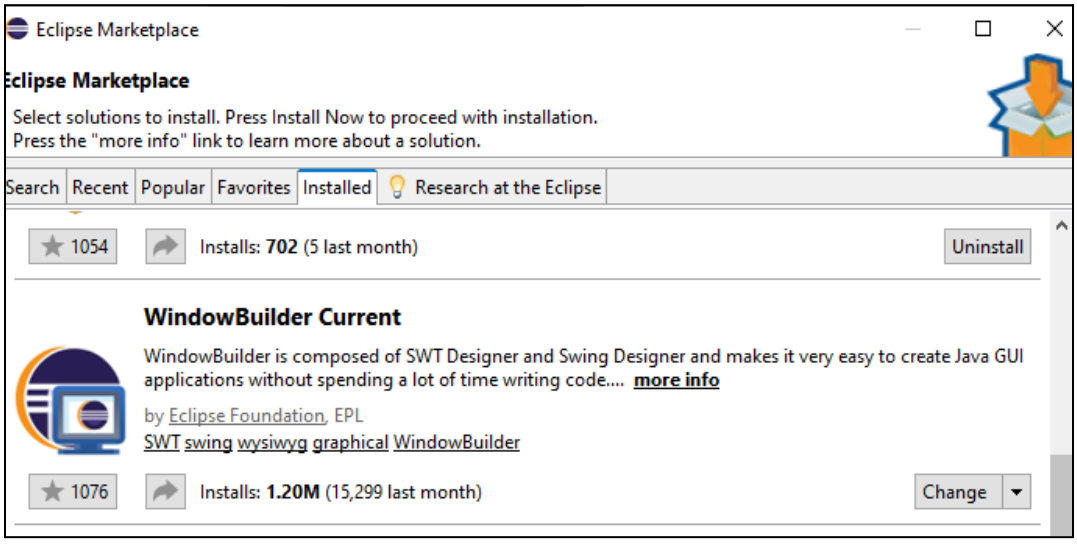
2. EJERCICIOS

2.1 EJERCICIO 1
GUI Swing a través de operaciones “soltar y arrastrar”

a. Como es el proceso de creación de una GUI utilizando herramientas de “soltar y arrastrar” y haga capturas de pantalla que evidencien el mismo.

Proceso:

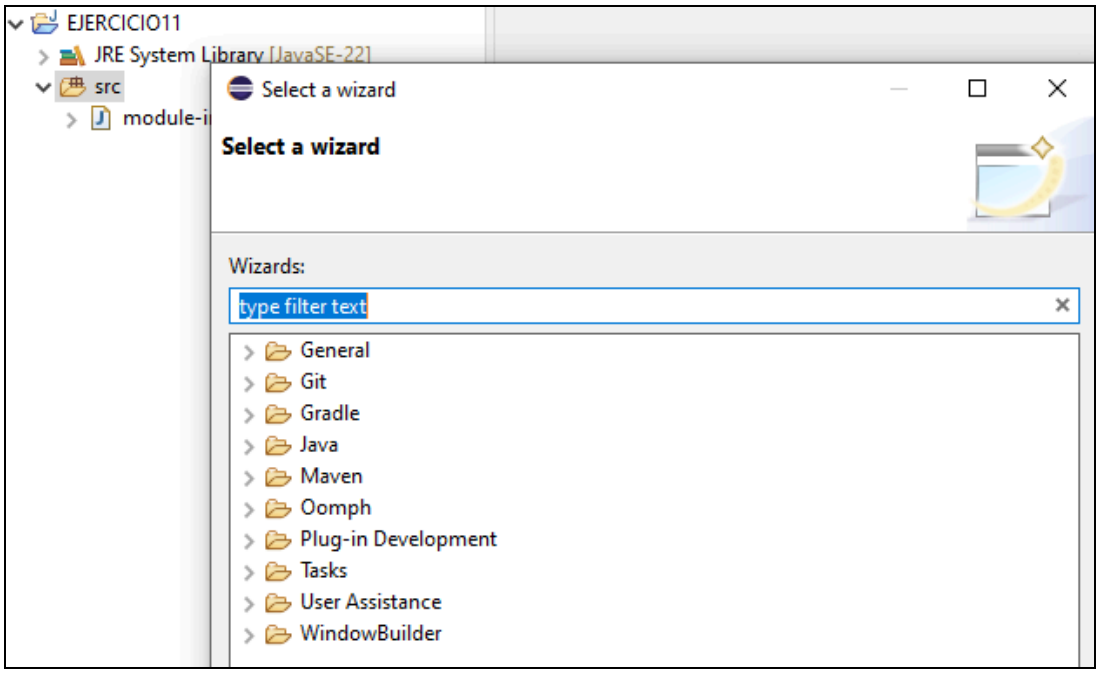
- **Instalación de plugin WindowBuilder**
  Se obtiene en Eclipse Marketplace



- **Instalación de plugin WindowBuilder**
  Crear una nueva clase con extensión de JFrame:

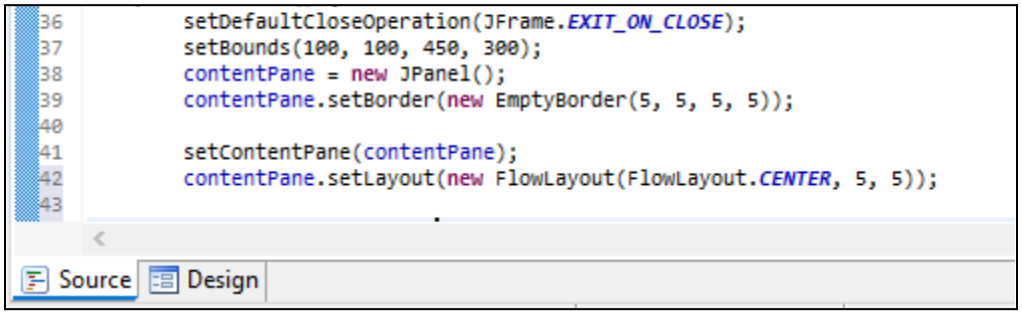
  Haz clic derecho en src -> New -> Class.

  Seleccionar la casilla public static void main(String[] args).



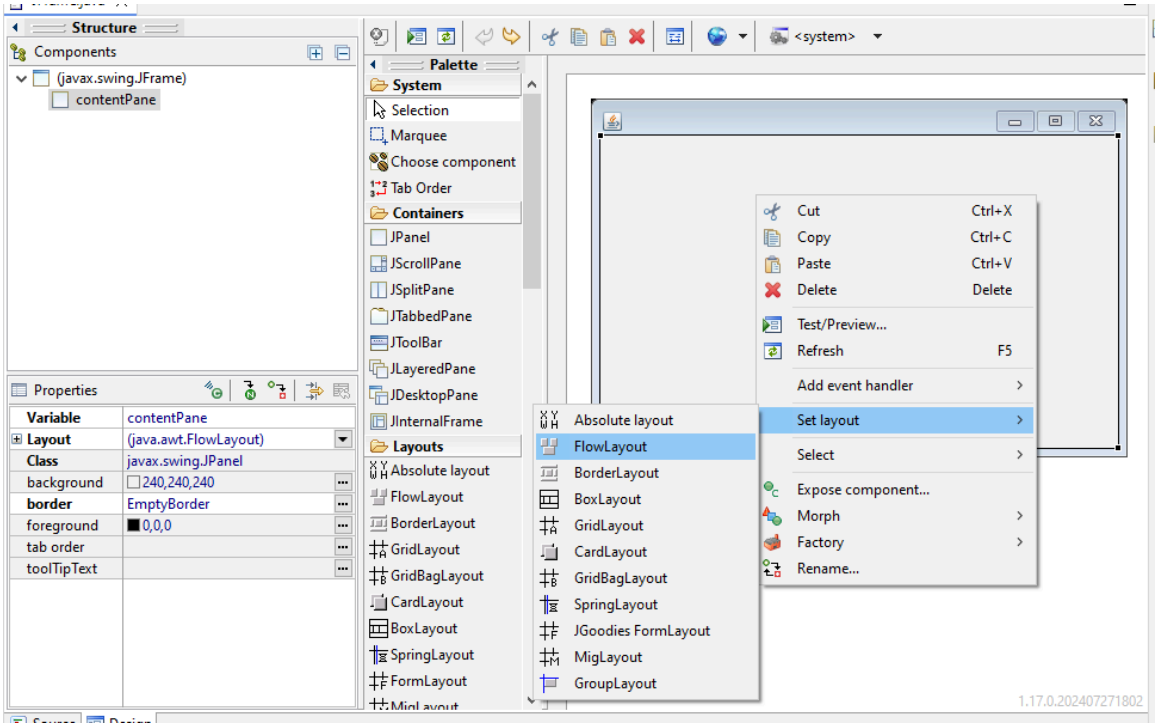
- **Abrir WindowBuilder para diseño visual**

  Clase en modo diseño:
  Seleccionar Open With > WindowBuilder Editor.



Design

Configurar el layout:

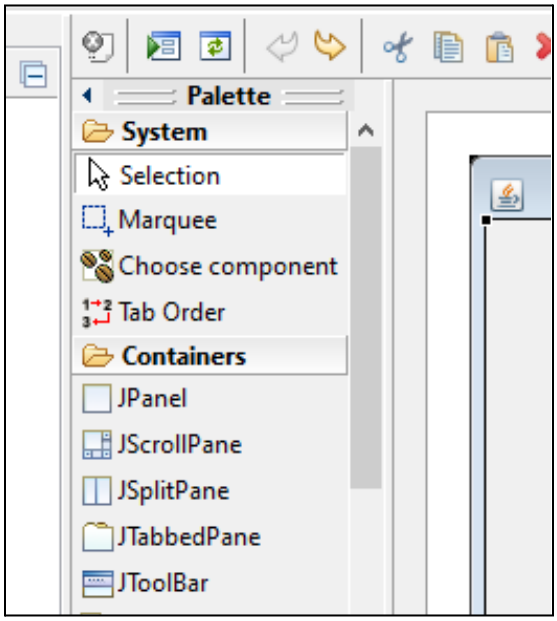


Agregar componentes:

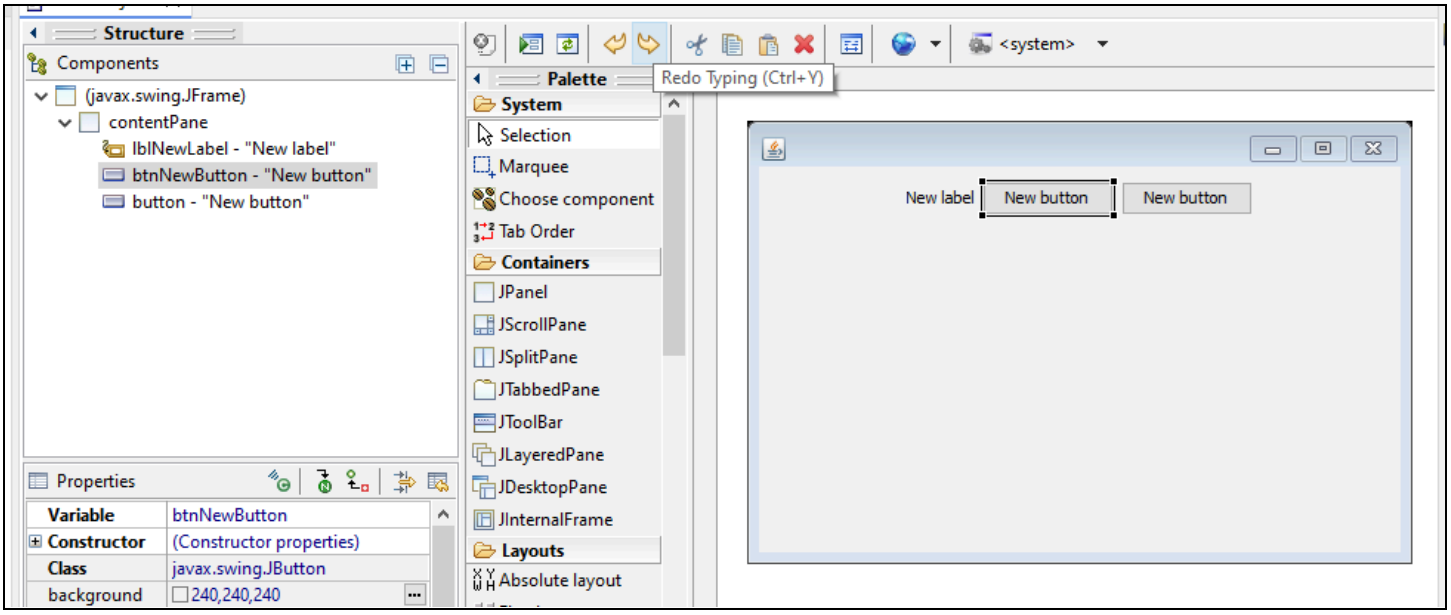
Componentes desde el panel Palette

Etiquetas (**JLabel**): Para mostrar texto.

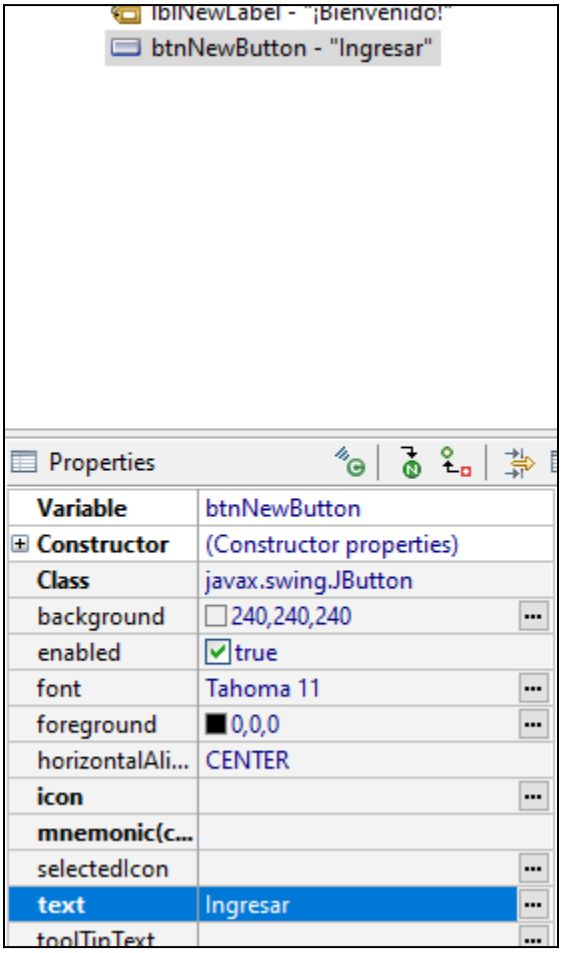
Botones (**JButton**): Para acciones como "Ingresar" y "Registrarse".

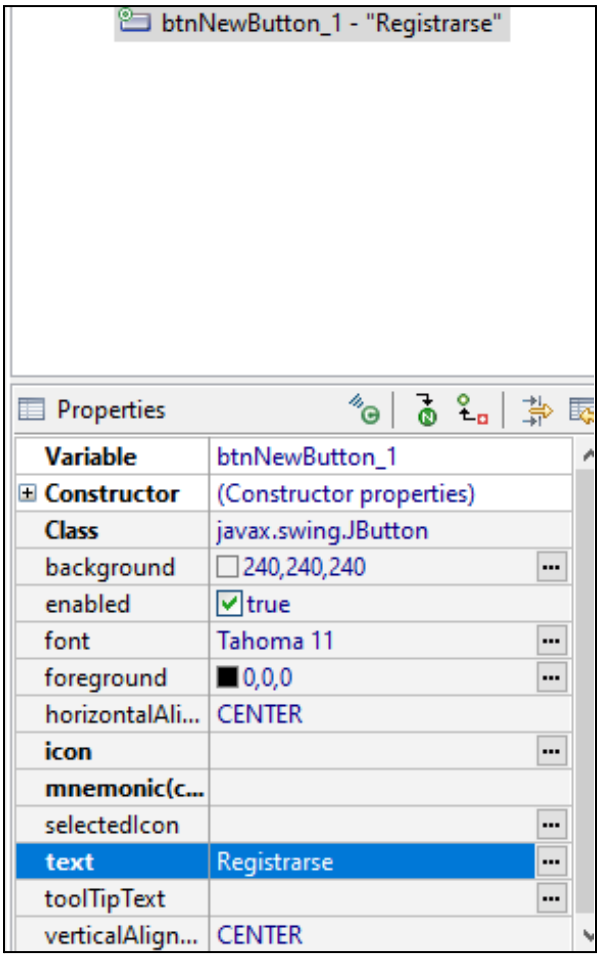


Insertar botones y etiqueta



Detalles





b. Ventajas y desventajas de crear una GUI desde código vs con herramientas de "soltar y arrastrar"

Código	Ventajas	Desventajas
	- Control total sobre el diseño. - Es independiente del IDE	- El diseño puede ser tedioso y complicado al no tener una vista en vivo del diseño

Soltar y arrastrar	Ventajas	Desventajas
	- Facilita modificaciones rápidas al diseño. - Vista intuitiva.	- Dependencia del IDE - Dificultad para ajustar detalles avanzados. - Si no se conocen bien las herramientas es algo complicado.

2.1.1 EJERCICIO 2 - CORREGIDO

ENLACE GITHUB:
https://github.com/KahoriDiazUCSM/Laboratorio9_/tree/main/EJERCICIO2
Implemente una pequeña aplicación que simule el ingreso de datos para la compra de pasajes de una empresa de transporte terrestre. Para esto deberá utilizar todos los conceptos y componentes ya vistos. Una vez que se ingresen todos los datos del pasajero, se deberá mostrar un resumen de sus datos en un cuadro de Diálogo luego de presionar un botón. A continuación, se muestran cómo se utilizarían los componentes para ingresar un tipo de información específica:

Nombre de Componente	Propósito que cumplira en la GUI de compra de pasaje
Etiquetas	Rotular que tipo de información se ingresara en cada componente
Campos de Texto	Ingreso de nombres, documento de identidad y fecha de viaje
Botón de Comando	Reiniciar o blanquear todos los componentes y mostrar diálogo
Casillas de Verificación	Indicar servicio opcional para pasajero (audifonos, manta, revistas)
Botones de Opción	Indicar si el pasajero quiere viajar en 1er o 2do piso
Cuadros combinados	Permitir al pasajero elegir un lugar de origen y de destino
Lista	Elegir entre 3 calidades de servicio (económico, standard, VIP)

Pasajero

Descripción: Representa un pasajero que compra un pasaje. Almacena su nombre y DNI.

Componentes:

- **String nombre:** Nombre del pasajero.
- **String dni:** DNI del pasajero.

Métodos:

- **Constructor:** Inicializa los atributos nombre y dni.
- **Getters y setters:** Métodos para obtener o modificar el nombre y DNI.

```
/*************************************************************
* NOMBRE : Pasajero.java
* DESCRIPCIO´N: Clase Pasajero
*************************************************************/
package ejercicio2.modelo;


public class Pasajero {
    private String nombre;
    private String dni;


    public Pasajero(String nombre, String dni) {
        this.nombre = nombre;
        this.dni = dni;
    }

    //Getters y setters
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getDni() {
        return dni;
    }

    public void setDni(String dni) {
        this.dni = dni;
    }
}
```

Clase DNI - VALIDA NÚMERO DE DÍGITOS(8)

```
/*************************************************************
* NOMBRE : DNI.java
*************************************************************/
public class DNI {
    private String numero;

    public DNI(String numero) {
        setNumero(numero);
    }

    public String getNumero() {
        return numero;
    }

    public void setNumero(String numero) {
        this.numero = numero;
    }

    public boolean validarDNI() {
        // Verificar que el DNI sea un número y tenga exactamente 8 dígitos
        return numero.matches("\\d{8}");
    }
}
```

Clase Fecha - VALIDA formato de fecha

```
import java.text.ParseException;
import java.text.SimpleDateFormat;

public class Fecha {
    private int anio;
    private int dia;
    private int mes;

    public Fecha() {
    }

    public Fecha(int dia, int mes, int anio) {
        setAnio(anio);
        setDia(dia);
        setMes(mes);
    }

    public int getAnio() {
        return anio;
    }

    public int getDia() {
        return dia;
    }
```

```
    public int getMes() {
        return mes;
    }

    public void setAnio(int anio) {
        this.anio = anio;
    }

    public void setDia(int dia) {
        this.dia = dia;
    }

    public void setMes(int mes) {
        this.mes = mes;
    }

    public boolean validarFecha() {
        boolean correcto = false;

        try {
            // Formato de fecha (día/mes/año)
            SimpleDateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy");
            formatoFecha.setLenient(false); // No permite fechas inválidas
            // Comprobación de la fecha
            formatoFecha.parse(this.dia + "/" + this.mes + "/" + this.anio);
            correcto = true;
        } catch (ParseException e) {
            // Si la fecha no es correcta, pasará por aquí
            correcto = false;
        }

        return correcto;
    }
}
```

Clase Reserva_Compra

- **Descripción**: Esta clase representa la reserva de un pasaje de autobús. Contiene información del pasajero, su origen, destino, piso asignado, servicios opcionales y la calidad del servicio seleccionada.

- **Componentes**:
- **Pasajero pasajero**: Objeto que almacena la información del pasajero (nombre y DNI).
- **String origen**: Ciudad o terminal de origen.
- **String destino**: Ciudad o terminal de destino.
- **String piso**: Piso del autobús seleccionado (1er o 2do piso).
- **List<String> serviciosOpcionales**: Lista de servicios opcionales, como audífonos, manta, etc.
- **String calidadServicio**: Tipo de calidad del servicio seleccionado (por ejemplo, "VIP").

- **Métodos**:
- **Constructor**: Inicializa todos los atributos de la clase.
- **Getters y setters**: Métodos para acceder y modificar los valores de los atributos.
- **generarResumen()**: Devuelve un resumen en forma de texto con toda la información de la reserva.

```
/*********************************************************
* NOMBRE : Reserva_Compra.java
* DESCRIPCIÓN: Clase Reserva_Compra
*********************************************************/
package ejercicio2.modelo;


import java.util.List;


public class Reserva_Compra {
    private Pasajero pasajero;
    private String origen;
    private String destino;
    private String piso;
    private List<String> serviciosOpcionales;
    private String calidadServicio;

    // Constructor
    public Reserva_Compra(Pasajero pasajero, String origen, String destino, String
piso,
                          List<String> serviciosOpcionales, String calidadServicio) {
        this.pasajero = pasajero;
        this.origen = origen;
```

```
        this.destino = destino;
        this.piso = piso;
        this.serviciosOpcionales = serviciosOpcionales;
        this.calidadServicio = calidadServicio;
    }

    // Getters y setters
    public Pasajero getPasajero() {
        return pasajero;
    }

    public void setPasajero(Pasajero pasajero) {
        this.pasajero = pasajero;
    }

    public String getOrigen() {
        return origen;
    }

    public void setOrigen(String origen) {
        this.origen = origen;
    }

    public String getDestino() {
        return destino;
    }

    public void setDestino(String destino) {
        this.destino = destino;
    }

    public String getPiso() {
        return piso;
    }

    public void setPiso(String piso) {
        this.piso = piso;
    }

    public List<String> getServiciosOpcionales() {
        return serviciosOpcionales;
    }

    public void setServiciosOpcionales(List<String> serviciosOpcionales) {
        this.serviciosOpcionales = serviciosOpcionales;
    }

    public String getCalidadServicio() {
        return calidadServicio;
    }

    public void setCalidadServicio(String calidadServicio) {
        this.calidadServicio = calidadServicio;
    }
```

```
    // Resumen de la reserva
    public String generarResumen() {
        StringBuilder resumen = new StringBuilder();
        resumen.append("Reserva de Pasaje:\n");
        resumen.append("Nombre: ").append(pasajero.getNombre()).append("\n");
        resumen.append("DNI: ").append(pasajero.getDni()).append("\n");
        resumen.append("Origen: ").append(origen).append("\n");
        resumen.append("Destino: ").append(destino).append("\n");
        resumen.append("Piso: ").append(piso).append("\n");
        resumen.append("Servicios Opcionales: ").append(String.join(", ",
serviciosOpcionales)).append("\n");
        resumen.append("Calidad del Servicio: ").append(calidadServicio).append("\n");
        return resumen.toString();
    }
}
```

CompraPasajesVista

- **Descripción:** Interfaz gráfica de la aplicación para la compra de pasajes. Permite al usuario ingresar sus datos personales, seleccionar origen, destino, piso, servicios opcionales y calidad del servicio.

- **Componentes:**
- **Etiquetas (JLabel):**
    - Nombre:, DNI:, Fecha de Viaje:, Origen:, Destino:, etc.
- **Campos de texto (JTextField):**
    - nombreField, dniField, fechaViajeField para ingresar datos personales.
- **Combos (JComboBox):**
    - origenCombo y destinoCombo para seleccionar las ciudades.
- **Radios (JRadioButton):**
    - primerPiso y segundoPiso para elegir el piso del autobús.
- **Checkboxes (JCheckBox):**
    - servicioAudifonos, servicioManta, servicioRevistas para servicios opcionales.
- **Lista (JList):**
    - servicioList para seleccionar la calidad del servicio.
- **Botones (JButton):**
    - btnResumen para generar un resumen de la reserva.
    - btnReiniciar para reiniciar los campos.

- **Diseño:** Utiliza un diseño GridLayout para organizar los componentes en filas y columnas

.

- **Configuraciones adicionales:**
- **setSize(400, 300):** Define el tamaño de la ventana.
- **setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE):** Finaliza la aplicación al cerrar esta ventana.

```
/*************************************************************
* NOMBRE : CompraPasajesVista.java
* DESCRIPCIO´N: CompraPasajesVista
*************************************************************/
package ejercicio2.vista;


import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.*;
import javax.swing.*;

public class CompraPasajesVista extends JFrame {
    // Componentes para el formulario
    public JTextField nombreField;
    public JTextField dniField;
    public JTextField fechaViajeField;
    public JComboBox<String> origenn;
    public JComboBox<String> destinoo;
    public JRadioButton primerPiso;
    public JRadioButton segundoPiso;
    public ButtonGroup grupoPiso;
    public JCheckBox servicioAudifonos;
    public JCheckBox servicioManta;
    public JCheckBox servicioRevistas;
    public JList<String> servicioList;
    public JButton botonResumen;
    public JButton botonReiniciar;

    // Constructor
    public CompraPasajesVista() {
        setTitle("Compra de Pasajes");
        setSize(400, 500);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new GridLayout(10, 2));

        // Nombre
        add(new JLabel("Nombre:"));
        nombreField = new JTextField();
        add(nombreField);

        // DNI
        add(new JLabel("DNI:"));
        dniField = new JTextField();
        add(dniField);

        // Fecha de viaje
```

```
        add(new JLabel("Fecha de Viaje:"));
        fechaViajeField = new JTextField();
        add(fechaViajeField);

        // Origen
        add(new JLabel("Origen:"));
        origenn = new JComboBox<>(new String[]{"Arequipa", "Lima", "Cusco"});
        add(origenn);

        // Destino
        add(new JLabel("Destino:"));
        destinoo = new JComboBox<>(new String[]{"Ayacucho", "Ica", "Puno"});
        add(destinoo);

        // Piso
        add(new JLabel("Piso:"));
        primerPiso = new JRadioButton("1er Piso");
        segundoPiso = new JRadioButton("2do Piso");
        grupoPiso = new ButtonGroup();
        grupoPiso.add(primerPiso);
        grupoPiso.add(segundoPiso);
        JPanel pisoPanel = new JPanel(new FlowLayout());
        pisoPanel.add(primerPiso);
        pisoPanel.add(segundoPiso);
        add(pisoPanel);

        // Servicios extra
        add(new JLabel("Servicios Extras:"));
        servicioAudifonos = new JCheckBox("Audífonos");
        servicioManta = new JCheckBox("Manta");
        servicioRevistas = new JCheckBox("Revistas");
        JPanel serviciosPanel = new JPanel(new FlowLayout());
        serviciosPanel.add(servicioAudifonos);
        serviciosPanel.add(servicioManta);
        serviciosPanel.add(servicioRevistas);
        add(serviciosPanel);

        // Calidad del servicio
        add(new JLabel("Calidad del Servicio:"));
        servicioList = new JList<>(new String[]{"Económicoo", "Ejecutivo", "VIP"});
        add(new JScrollPane(servicioList));

        // Botones
        botonResumen = new JButton("Mostrar Resumen");
        botonReiniciar = new JButton("Reiniciar");
        add(botonResumen);
        add(botonReiniciar);

        setVisible(true);
    }
}
```

CompraPasajesControlador - Corregido para hacer validaciones

Referencia: https://es.stackoverflow.com/questions/174706/como-validar-fecha

- **Descripción:** Controlador que maneja la interacción entre la vista y los modelos. Captura datos, genera el resumen y reinicia los campos.

- **Componentes:**
- **Vista:** Objeto de la clase CompraPasajesVista-
- **Acciones:**
  - **mostrarResumen():** Obtiene los datos de la vista, crea objetos de Pasajero y Reserva_Compra, y muestra un resumen en un cuadro de diálogo.
  - **reiniciarCampos():** Limpia todos los campos y reinicia las selecciones de la vista.

- **Eventos manejados:**
- **Botón Resumen:** Genera y muestra el resumen de la reserva.
- **Botón Reiniciar:** Limpia todos los campos y componentes seleccionados.

```
/*************************************************************
* NOMBRE : CompraPasajesControlador.java
* DESCRIPCIO´N: CompraPasajesControlador
*************************************************************/
package ejercicio2.controlador;

import ejercicio2.modelo.Pasajero;
import ejercicio2.modelo.Reserva_Compra;
import ejercicio2.vista.CompraPasajesVista;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class CompraPasajesControlador {
    private CompraPasajesVista vista;

    public CompraPasajesControlador(CompraPasajesVista vista) {
        this.vista = vista;

        // Botón de resumen -> acción
        vista.botonResumen.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                mostrarResumen();
            }
        });

        // Botón de reiniciar
        vista.botonReiniciar.addActionListener(new ActionListener() {
```

```
            @Override
            public void actionPerformed(ActionEvent e) {
                reiniciarCampos();
            }
        });
    }

    private void mostrarResumen() {
        // Validar campos obligatorios
        String nombre = vista.nombreField.getText();
        String dni = vista.dniField.getText();
        String fechaViaje = vista.fechaViajeField.getText();

        // Validar nombre
        if (nombre.trim().isEmpty()) {
            JOptionPane.showMessageDialog(vista, "El nombre no puede estar vacío.",
"Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        // Validar DNI
        if (!validarDNI(dni)) {
            return;
        }

        // Validar fecha de viaje
        if (!validarFechaViaje(fechaViaje)) {
            return;
        }

        // Validar que se haya seleccionado un piso
        if (!vista.primerPiso.isSelected() && !vista.segundoPiso.isSelected()) {
            JOptionPane.showMessageDialog(vista, "Debe seleccionar un piso.", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }

        // Validar que se haya seleccionado una calidad de servicio
        if (vista.servicioList.getSelectedValue() == null) {
            JOptionPane.showMessageDialog(vista, "Debe seleccionar una calidad de
servicio.", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        // Si todas las validaciones pasan, generar el resumen
        Pasajero pasajero = new Pasajero(nombre, dni);
        String origen = (String) vista.origenn.getSelectedItem();
        String destino = (String) vista.destinoo.getSelectedItem();
        String piso = vista.primerPiso.isSelected() ? "1er Piso" : "2do Piso";
        List<String> servicios = new ArrayList<>();
        if (vista.servicioAudifonos.isSelected()) servicios.add("Audífonos");
        if (vista.servicioManta.isSelected()) servicios.add("Manta");
        if (vista.servicioRevistas.isSelected()) servicios.add("Revistas");
        String calidadServicio = vista.servicioList.getSelectedValue();
```

```
        Reserva_Compra reserva = new Reserva_Compra(pasajero, origen, destino, piso,
servicios, calidadServicio);
        JOptionPane.showMessageDialog(vista, reserva.generarResumen());
    }

    private boolean validarDNI(String dni) {
        DNI dniObj = new DNI(dni);
        if (!dniObj.validarDNI()) {
            JOptionPane.showMessageDialog(vista, "El DNI debe contener exactamente 8
dígitos.", "Error", JOptionPane.ERROR_MESSAGE);
            return false;
        }
        return true;
    }

    private boolean validarFechaViaje(String fechaViaje) {
        // Verificar que la fecha tenga el formato DD/MM/AAAA
        if (!fechaViaje.matches("\\d{2}/\\d{2}/\\d{4}")) {
            JOptionPane.showMessageDialog(vista, "La fecha de viaje debe tener el
formato DD/MM/AAAA.", "Error", JOptionPane.ERROR_MESSAGE);
            return false;
        }

        // Separar los componentes de la fecha
        String[] partes = fechaViaje.split("/");
        int dia = Integer.parseInt(partes[0]);
        int mes = Integer.parseInt(partes[1]);
        int anio = Integer.parseInt(partes[2]);

        // Validar la fecha utilizando la clase Fecha
        Fecha fecha = new Fecha(dia, mes, anio);
        if (!fecha.validarFecha()) {
            JOptionPane.showMessageDialog(vista, "La fecha de viaje no es válida.",
"Error", JOptionPane.ERROR_MESSAGE);
            return false;
        }

        return true;
    }

    private void reiniciarCampos() {
        vista.nombreField.setText("");
        vista.dniField.setText("");
        vista.fechaViajeField.setText("");
        vista.origenn.setSelectedIndex(0);
        vista.destinoo.setSelectedIndex(0);
        vista.grupoPiso.clearSelection();
        vista.servicioAudifonos.setSelected(false);
        vista.servicioManta.setSelected(false);
        vista.servicioRevistas.setSelected(false);
        vista.servicioList.clearSelection();
    }
}
```

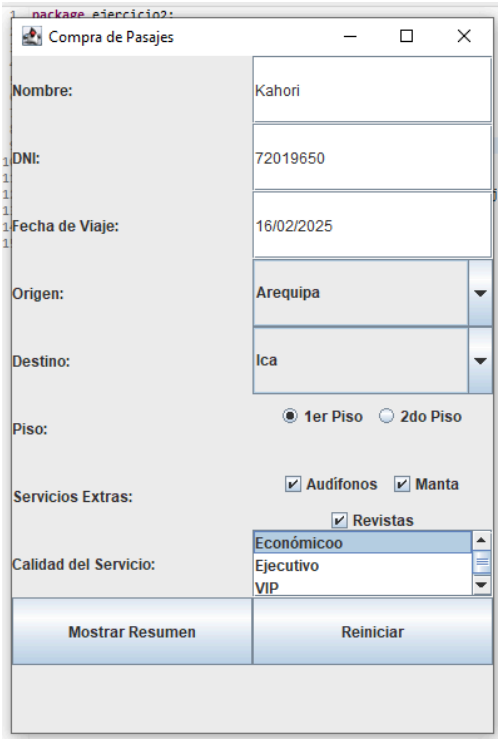
AppCompraPasajes

Descripción:

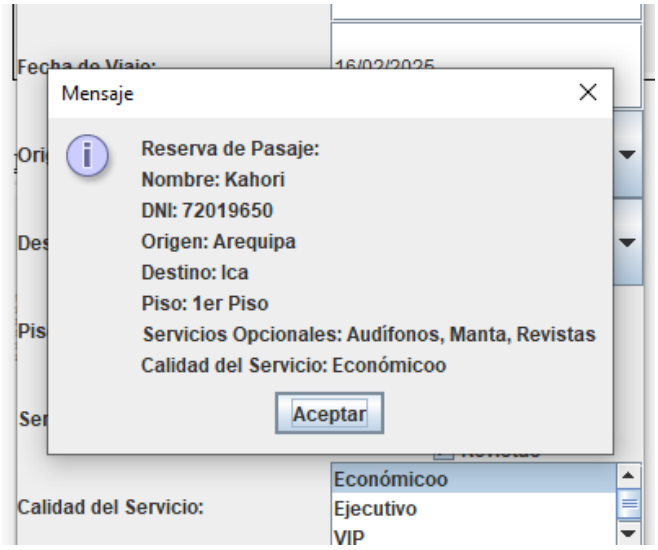
La clase AppCompraPasajes es el punto de entrada del programa. Inicializa las clases de Vista y Controlador..

```
/**********************************************************
* NOMBRE : AppCompraPasajes.java
* DESCRIPCIO´N: Prueba
**********************************************************/
package ejercicio2;

import ejercicio2.controlador.CompraPasajesControlador;
import ejercicio2.vista.CompraPasajesVista;

public class AppCompraPasajes {
    public static void main(String[] args) {

        CompraPasajesVista vista = new CompraPasajesVista();


        CompraPasajesControlador controlador = new
CompraPasajesControlador(vista);
    }
}
```

Prueba



Mostrar Resumen



CORRECCIONES

Clase Fecha: Valida fechas utilizando **SimpleDateFormat**.

Clase DNI: Valida que el DNI sea un número de 8 dígitos.

Clase CompraPasajesControlador: Se integraron las validaciones de fecha y DNI en el flujo de la aplicación.

3. BIBLIOGRAFÍA

Eclipse Foundation. (s.f.). WindowBuilder. Recuperado de https://eclipse.dev/windowbuilder/

JaviDev. (s.f.). Java Swing GUI Tutorial - Make a GUI in 13 Minutes [Video]. YouTube. Recuperado de https://www.youtube.com/watch?v=43fmwIcXdMY