**Asignatura**

Programación orientada a objetos

**Estudiantes**

Juan Camilo Herrera Sibaja

María Alejandra Castaño Tobón

Juan Camilo Hincapié Uribe

**Docente**

Walter Hugo Arboleda

**Grupo**

4

**Universidad Nacional de Colombia**

**Medellín**

**Herencia**

**Ejercicio 4.1 Resuelto**

**Clase: Cuenta**

package pruebacuenta;

public class Cuenta {

protected float saldo;

/\* Atributo que define el número de consignaciones realizadas en

una cuenta bancaria \*/

protected int numeroConsignaciones = 0;

// Atributo que define el número de retiros de una cuenta bancaria

protected int numeroRetiros = 0;

// Atributo que define la tasa anual de intereses de una cuenta bancaria

protected float tasaAnual;

/\* Atributo que define la comisión mensual que se cobra a una

cuenta bancaria \*/

protected float comisionMensual = 0;

public Cuenta(float saldo, float tasaAnual) {

this.saldo = saldo;

this.tasaAnual = tasaAnual;

}

public void consignar(float cantidad) {

saldo = saldo + cantidad;

/\* Se actualiza el saldo con la cantidad

consignada \*/

// Se actualiza el número de consignaciones realizadas en la cuenta

numeroConsignaciones = numeroConsignaciones + 1;

}

public void retirar(float cantidad) {

float nuevoSaldo = saldo - cantidad;

/\* Si la cantidad a retirar no supera el saldo, el retiro no se puede

realizar \*/

if (nuevoSaldo >= 0) {

saldo -= cantidad;

numeroRetiros = numeroRetiros + 1;

} else {

System.out.println("La cantida a retirar excede el saldo actual.");

}

}

public void calcularInterés() {

float tasaMensual = tasaAnual / 12; /\* Convierte la tasa anual en

mensual \*/

float interesMensual = saldo \* tasaMensual;

saldo += interesMensual; /\* Actualiza el saldo aplicando el interés

mensual \*/

}

public void extractoMensual() {

saldo -= comisionMensual;

calcularInterés();

}

}

**Clase: CuentaAhorros**

package pruebacuenta;

public class CuentaAhorros extends Cuenta{

/\* Atributo que identifica si una cuenta está activa; lo está si su saldo

es superior a 10000 \*/

private boolean activa;

public CuentaAhorros(float saldo, float tasa) {

super(saldo, tasa);

if (saldo < 10000) /\* Si el saldo es menor a 10000, la cuenta no

se activa \*/

activa = false;

else

activa = true;

}

public void retirar(float cantidad) {

if (activa) // Si la cuenta está activa, se puede retirar dinero

super.retirar(cantidad);

}

public void consignar(float cantidad) {

if (activa) // Si la cuenta está activa, se puede consignar dinero

super.consignar(cantidad); /\* Invoca al método consignar de

la clase padre \*/

}

public void extractoMensual() {

/\* Si la cantidad de retiros es superior a cuatro, se genera una

comisión mensual \*/

if (numeroRetiros > 4) {

comisionMensual += (numeroRetiros - 4) \* 1000;

}

super.extractoMensual(); // Invoca al método de la clase padre

/\* Si el saldo actualizado de la cuenta es menor a 10000, la

cuenta no se activa \*/

if ( saldo < 10000 )

activa = false;

}

public void imprimir() {

System.out.println("Saldo = $ " + saldo);

System.out.println("Comisión mensual = $ " + comisionMensual);

System.out.println("Número de transacciones = " + (numeroConsignaciones + numeroRetiros));

System.out.println();

}

}

**Clase: CuentaCorriente**

package pruebacuenta;

public class CuentaCorriente extends Cuenta{

/\* Atributo que define un sobregiro de la cuenta que surge cuando el

saldo de la cuenta es negativo \*/

float sobregiro;

public CuentaCorriente(float saldo, float tasa) {

super(saldo, tasa); // Invoca al constructor de la clase padre

sobregiro = 0; // Inicialmente no hay sobregiro

}

public void retirar(float cantidad) {

float resultado = saldo - cantidad; // Se calcula un saldo temporal

/\* Si el valor a retirar supera el saldo de la cuenta, el valor

excedente se convierte en sobregiro y el saldo es cero \*/

if (resultado < 0) {

sobregiro = sobregiro - resultado;

saldo = 0;

} else {

super.retirar(cantidad); /\* Si no hay sobregiro, se realiza un

retiro normal \*/

}

}

public void consignar(float cantidad) {

float residuo = sobregiro - cantidad;

// Si existe sobregiro la cantidad consignada se resta al sobregiro

if (sobregiro > 0) {

if ( residuo > 0) { /\* Si el residuo es mayor que cero, se libera

el sobregiro \*/

sobregiro = 0;

saldo = residuo;

} else { /\* Si el residuo es menor que cero, el saldo es cero y

surge un sobregiro \*/

sobregiro = -residuo;

saldo = 0;

}

} else {

super.consignar(cantidad); /\* Si no hay sobregiro, se realiza

una consignación normal \*/

}

}

public void extractoMensual() {

super.extractoMensual(); // Invoca al método de la clase padre

}

public void imprimir() {

System.out.println("Saldo = $ " + saldo);

System.out.println("Cargo mensual = $ " + comisionMensual);

System.out.println("Número de transacciones = " +

(numeroConsignaciones + numeroRetiros));

System.out.println("Valor de sogregiro = $" +

(numeroConsignaciones + numeroRetiros));

System.out.println();

}

}

**Clase: PruebaCuenta**

package pruebacuenta;

import java.util.Scanner;

public class PruebaCuenta {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.println("Cuenta de ahorros");

System.out.println("Ingrese saldo inicial= $");

float saldoInicialAhorros = input.nextFloat();

System.out.print("Ingrese tasa de interés= ");

float tasaAhorros = input.nextFloat();

CuentaAhorros cuenta1 = new

CuentaAhorros(saldoInicialAhorros, tasaAhorros);

System.out.print("Ingresar cantidad a consignar: $");

float cantidadDepositar = input.nextFloat();

cuenta1.consignar(cantidadDepositar);

System.out.print("Ingresar cantidad a retirar: $");

float cantidadRetirar = input.nextFloat();

cuenta1.retirar(cantidadRetirar);

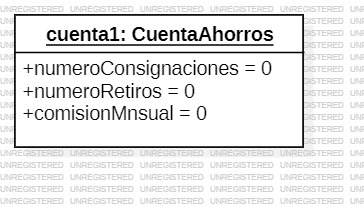
cuenta1.extractoMensual();

cuenta1.imprimir();

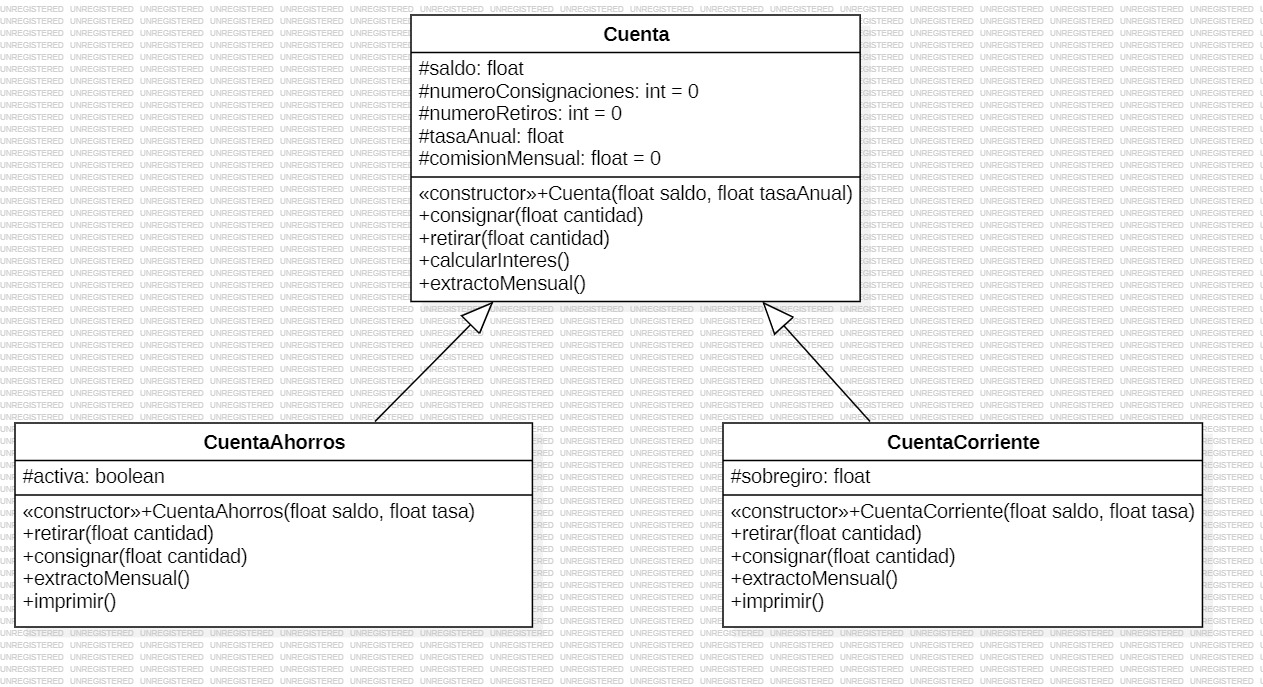
}

}

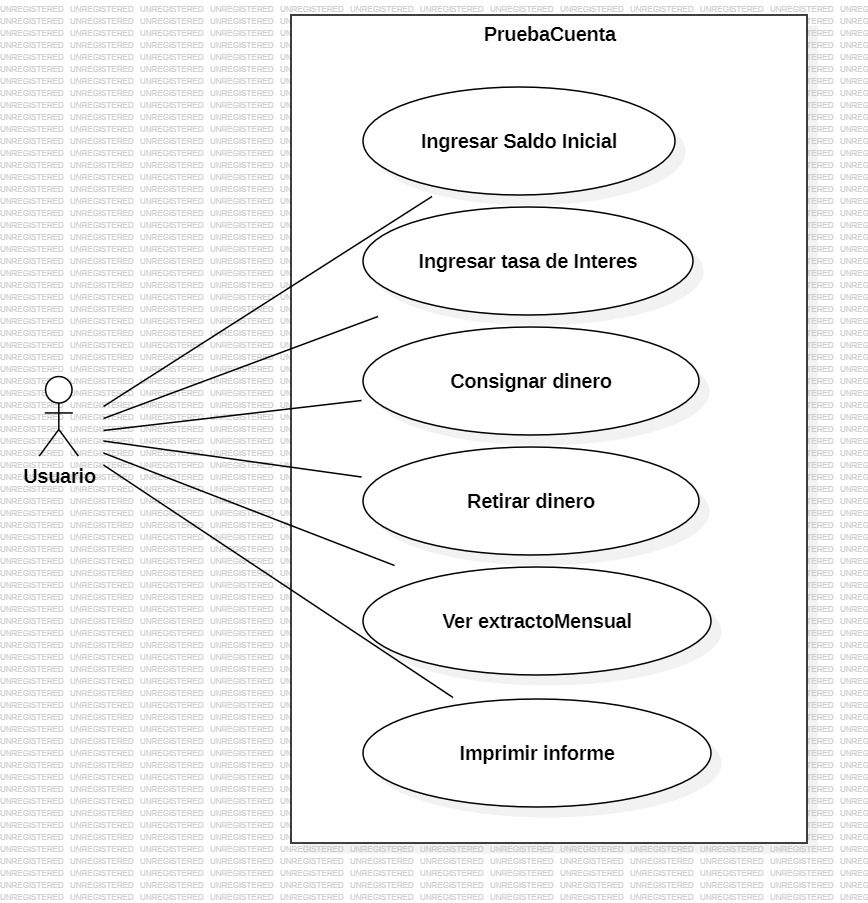
**Diagrama de objetos**



**Diagrama de clases**



**Diagrama de casos de uso**



**Paquete swing**

**Ejercicio 8.1 Resuelto**

**Clase: VentanaPrincipal**

package Personas;

import javax.swing.DefaultListModel;

import javax.swing.JOptionPane;

import javax.swing.event.ListSelectionEvent;

public class VentanaPrincipal extends javax.swing.JFrame {

ListaPersonas lista = new ListaPersonas();

DefaultListModel modelo = new DefaultListModel();

private int indice = -1;

public VentanaPrincipal() {

initComponents();

setTitle("Personas");

setResizable(false);

setLocationRelativeTo(null);

listaNombres.addListSelectionListener((ListSelectionEvent evt) -> {

indice = listaNombres.getSelectedIndex(); // Actualiza el índice

System.out.println("Índice seleccionado: " + indice); // Debug: Verifica el valor de índice

});

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

campoNombre = new javax.swing.JTextField();

campoApellidos = new javax.swing.JTextField();

campoTelefono = new javax.swing.JTextField();

campoDireccion = new javax.swing.JTextField();

añadir = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

listaNombres = new javax.swing.JList<>();

eliminar = new javax.swing.JButton();

borrarLista = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jLabel1.setText("Nombre:");

jLabel2.setText("Apellidos:");

jLabel3.setText("Teléfono:");

jLabel4.setText("Dirección:");

añadir.setText("Añadir");

añadir.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

añadirActionPerformed(evt);

}

});

jScrollPane1.setViewportView(listaNombres);

eliminar.setText("Eliminar");

eliminar.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

eliminarActionPerformed(evt);

}

});

borrarLista.setText("Borrar Lista");

borrarLista.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

borrarListaActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(20, 20, 20)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(37, 37, 37)

.addComponent(eliminar)

.addGap(27, 27, 27)

.addComponent(borrarLista))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel2)

.addComponent(jLabel3)

.addComponent(jLabel4)

.addComponent(jLabel1))

.addGap(30, 30, 30)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(campoNombre)

.addComponent(campoApellidos)

.addComponent(campoTelefono)

.addComponent(campoDireccion, javax.swing.GroupLayout.DEFAULT\_SIZE, 171, Short.MAX\_VALUE)

.addGroup(layout.createSequentialGroup()

.addGap(11, 11, 11)

.addComponent(añadir))))

.addComponent(jScrollPane1)))

.addContainerGap(26, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(27, 27, 27)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel1)

.addComponent(campoNombre, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel2)

.addComponent(campoApellidos, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel3)

.addComponent(campoTelefono, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jLabel4)

.addComponent(campoDireccion, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

.addComponent(añadir)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 106, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(eliminar)

.addComponent(borrarLista))

.addContainerGap(17, Short.MAX\_VALUE))

);

pack();

}// </editor-fold>

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

Persona p = new Persona(campoNombre.getText(),campoApellidos.getText(),campoTelefono.getText(), campoDireccion.getText());

lista.añadirPersona(p);

String elemento = campoNombre.getText() + "-" + campoApellidos.getText() + "-" + campoTelefono.getText() + "-" + campoDireccion.getText();

modelo.addElement(elemento);

listaNombres.setModel(modelo);

campoNombre.setText("");

campoApellidos.setText("");

campoTelefono.setText("");

campoDireccion.setText("");

}

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

lista.borrarLista();

modelo.clear();

}

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

if (indice >= 0 && indice < modelo.getSize()) {

modelo.removeElementAt(indice);

lista.eliminarPersona(indice);

indice = -1;

} else {

JOptionPane.showMessageDialog(null, "Debe seleccionar un elemento","Error", JOptionPane.ERROR\_MESSAGE);

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new VentanaPrincipal().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton añadir;

private javax.swing.JButton borrarLista;

private javax.swing.JTextField campoApellidos;

private javax.swing.JTextField campoDireccion;

private javax.swing.JTextField campoNombre;

private javax.swing.JTextField campoTelefono;

private javax.swing.JButton eliminar;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JList<String> listaNombres;

// End of variables declaration

}

**Clase: ListaPersonas**

package Personas;

import java.util.\*;

public class ListaPersonas {

Vector listaPersonas;

public ListaPersonas() {

listaPersonas = new Vector();

}

public void añadirPersona(Persona p) {

listaPersonas.add(p);

}

public void eliminarPersona(int i) {

listaPersonas.removeElementAt(i);

}

public void borrarLista() {

listaPersonas.removeAllElements();

}

}

**Clase: Persona**

package Personas;

public class Persona {

String nombre;

String apellidos;

String telefono;

String direccion;

public Persona(String nombre, String apellidos, String telefono, String direccion) {

this.nombre = nombre;

this.apellidos = apellidos;

this.telefono = telefono;

this.direccion = direccion;

}

}

**Clase: Principal**

package Personas;

public class Principal {

public static void main(String[] args) {

VentanaPrincipal miVentanaPrincipal;

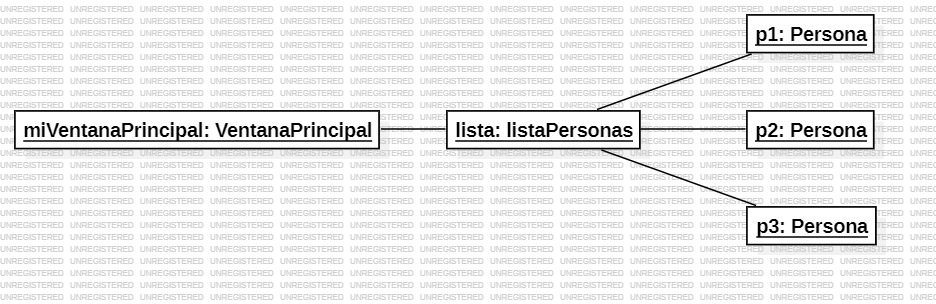
miVentanaPrincipal= new VentanaPrincipal();

miVentanaPrincipal.setVisible(true);

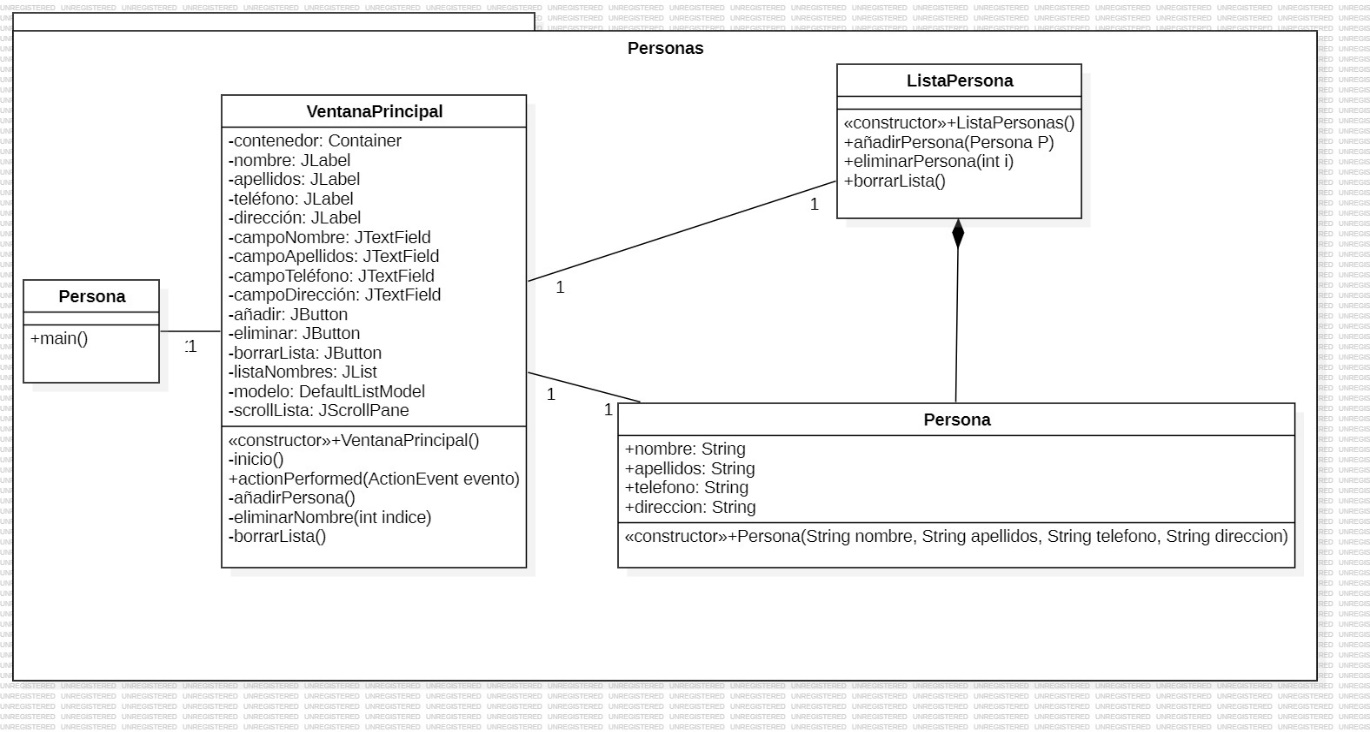
}

}

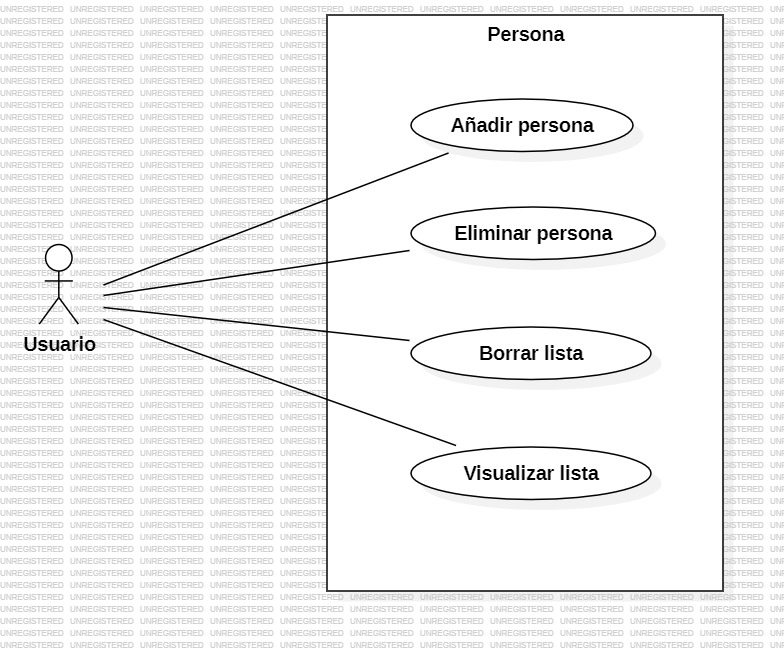
**Diagrama de objetos**

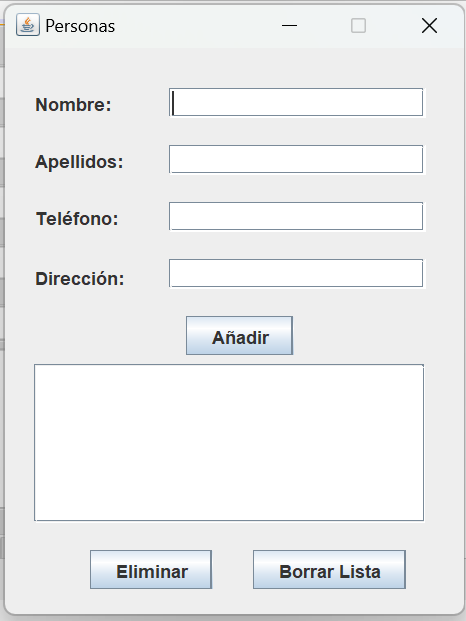
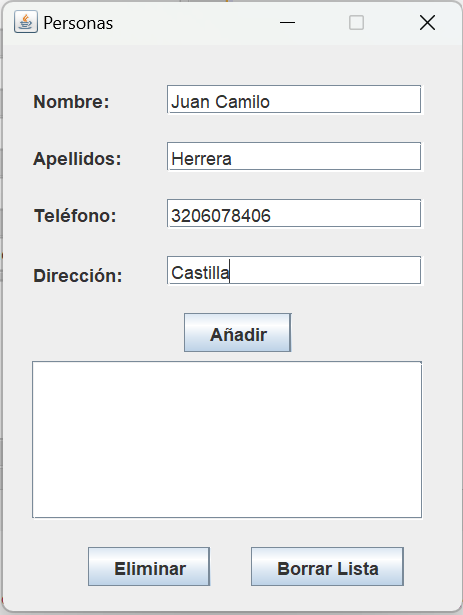


**Diagrama de clases**



**Diagrama de casos de uso**



******Interfaz grafica**

****