Universidad Corporativa Coppel | Programas Académicos

Ingeniería en Desarrollo de Software

Actividad | 2 |

Programa Banco Mexicano (Parte 1)

Lenguajes de Programación IV

Ingeniería en Desarrollo de Software

TUTOR: Ing. Aarón Iván Salazar Macías

ALUMNO: Joaquin Herman Valenzuela Aridjis

FECHA: 17/04/2025

Índice

Introducción

En el entorno bancario actual, la automatización de operaciones básicas como depósitos y retiros representa una necesidad fundamental para optimizar la experiencia del cliente y mejorar la eficiencia de los procesos internos. Esta actividad tiene como objetivo desarrollar un sistema en Java 8 que permita simular el funcionamiento inicial de un programa bancario orientado a clientes, el cual será ejecutado a través de un menú interactivo en consola. El desarrollo se basará en el paradigma de la programación orientada a objetos, aplicando estructuras de control como “if, while y switch-case” para gestionar el flujo de la aplicación. En esta primera etapa se implementarán las funcionalidades correspondientes al registro de depósitos y retiros, mientras que las opciones de consulta de saldo y salida se abordarán en una fase posterior. Esta práctica permitirá reforzar el diseño modular del código y la capacidad para desarrollar soluciones que respondan a requerimientos del mundo real, utilizando buenas prácticas de programación.

Descripción

La presente actividad forma parte de la segunda entrega del curso Lenguajes de Programación IV, y tiene como propósito diseñar un programa de consola que permita simular operaciones bancarias básicas para los clientes de una institución ficticia llamada Banco Mexicano. Utilizando Java 8 como lenguaje de programación y siguiendo el paradigma orientado a objetos, se busca implementar un menú interactivo con las opciones de depósito, retiro y una estructura base para las futuras funciones de saldo y salida. El sistema debe solicitar entradas por teclado, procesar las operaciones correspondientes y mantener un flujo adecuado utilizando estructuras de control como switch y if. En esta primera etapa se enfocará en el manejo de la lógica de depósitos múltiples y el control de retiros, validando entradas y regresando al menú principal según la interacción del usuario. Este proyecto representa una aproximación práctica a

la programación estructurada y modular, permitiendo al estudiante aplicar sus conocimientos en un escenario funcional y cotidiano.

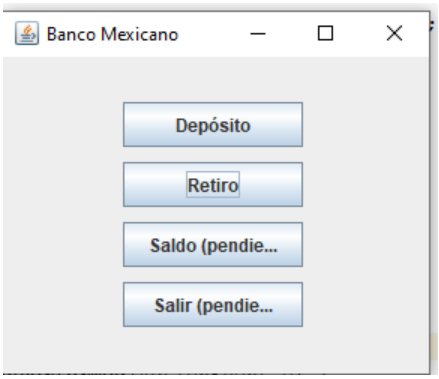
Justificación

El desarrollo de un programa que simule operaciones bancarias básicas como depósitos y retiros representa una oportunidad clave para aplicar los conocimientos adquiridos en programación orientada a objetos y estructuras de control. Este tipo de sistemas, aunque en su versión más simple, refleja problemáticas reales que enfrentan instituciones financieras en la gestión de transacciones cotidianas. Al implementar una solución que permita la interacción mediante un menú en consola, se refuerzan habilidades fundamentales como el control del flujo del programa, la validación de datos ingresados por el usuario y el diseño modular basado en clases. Además, se promueve la capacidad de interpretar requerimientos funcionales y traducirlos en soluciones concretas dentro del entorno de desarrollo Java. Esta actividad permite al estudiante visualizar cómo sus habilidades técnicas pueden resolver necesidades reales, aportando eficiencia, automatización y precisión en la ejecución de tareas que tradicionalmente se realizaban de forma manual, lo que resulta aplicable en múltiples sectores laborales.

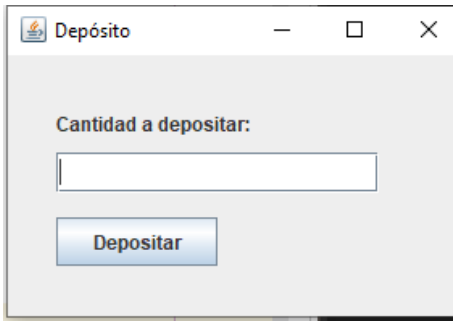
Desarrollo

Interfaz.

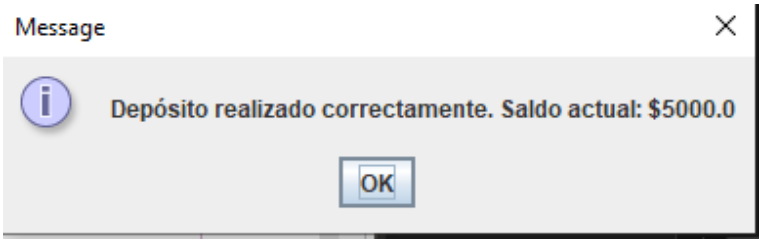
En esta primera imagen se puede observar como la GUI esta diseñada de manera sencilla, actualmente cuenta con 4 botones, aunque solamente estan operativos 2 de ellos, siendo estos el boton de deposito y retiro



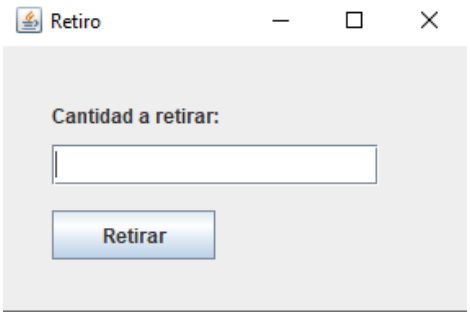
Al presionar el boton deposito, se nos despliega una nueva ventana en la cual indicaremos la cantidad a depositar



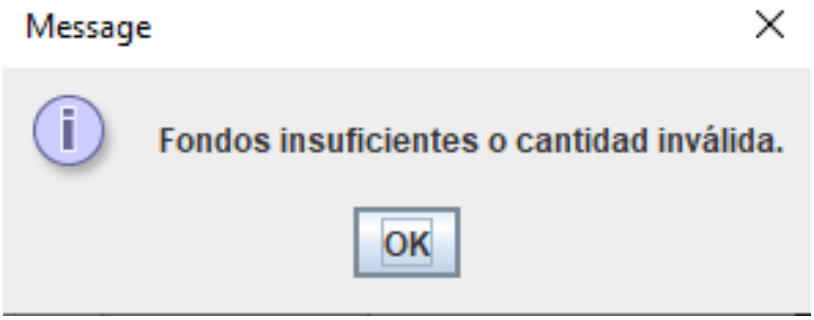
Una vez realizado el deposito nos arroja un mensaje de confirmacion y nos entrega el nuevo saldo de la cuenta



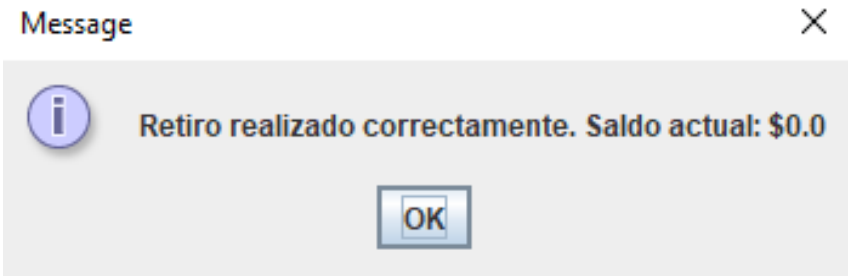
Por otro lado, la opcion retiro nos arroja una nueva ventana y nos permite ingresar una cantidad a retirar.



En caso de que la cantidad a retirar sea mayor que la cantidad disponible, nos arrojara el siguiente mensaje de “fondos insuficientes”

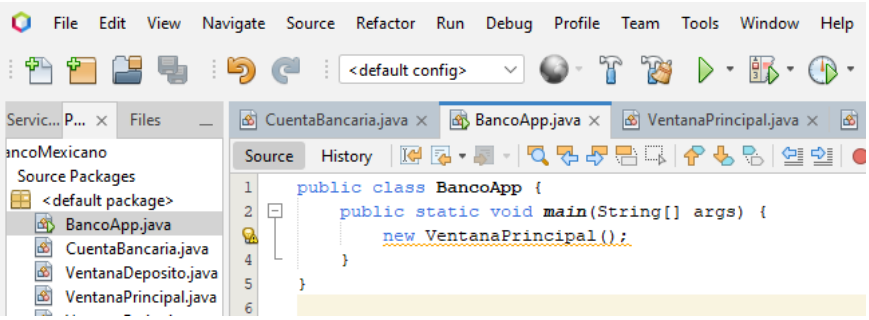


Y en caso de que la cantidad a retirar sea menor o igual al saldo, este arrojara un mensaje de confirmacion y el nuevo saldo restante



Codificación.

Metodo Main para arrancar el programa y evitar problemas



Codigo de la clase cuenta bancaria, para que realice y almacene la informacion de cada deposito y retiro

```
public class CuentaBancaria {
    private double saldo;

    public CuentaBancaria() {
        saldo = 0.0;
    }

    public void depositar(double cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
            System.out.println("Deposito exitoso. Nuevo saldo: $" + saldo);
        }else {
            System.out.println("Cantidad invalida. Intenta nuevamente.");
        }
    }

    public boolean retirar(double cantidad) {
        if (cantidad > 0 && cantidad <= saldo) {
         saldo -= cantidad;
            System.out.println("Retiro exitoso. Nuevo saldo: $" + saldo);
            return true;
        } else {
            System.out.println("Fondos insuficientes o cantidad inválida.");
            return false;
        }
    }


    public double getSaldo() {
        return saldo;
    }
}
```

Codigo de la clase ventanaPrincipal para la creacion del menu con cada uno de sus botones y la funcionalidad de los mismos

```
public class VentanaPrincipal extends JFrame {

    private CuentaBancaria cuenta;

    public VentanaPrincipal() {
        cuenta = new CuentaBancaria();

        setTitle("Banco Mexicano");
        setSize(300, 250);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(null);

        JButton btnDeposito = new JButton("Depósito");
        btnDeposito.setBounds(80, 30, 120, 30);
        add(btnDeposito);

        JButton btnRetiro = new JButton("Retiro");
        btnRetiro.setBounds(80, 70, 120, 30);
        add(btnRetiro);

        JButton btnSaldo = new JButton("Saldo (pendiente)");
        btnSaldo.setBounds(80, 110, 120, 30);
        add(btnSaldo);

        JButton btnSalir = new JButton("Salir (pendiente)");
        btnSalir.setBounds(80, 150, 120, 30);
        add(btnSalir);

        // Acción botón Depósito
        btnDeposito.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                new VentanaDeposito(cuenta);
            }
        });

        // Acción botón Retiro
        btnRetiro.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                new VentanaRetiro(cuenta);
            }
        });

        setVisible(true);
    }
}
```

Codigo de la clase ventanaDeposito, la cual permite incrementar el valor de la variable “saldo” esta cuenta con un mensaje de aceptación del deposito y en caso de que se ingrese una variable invalida como una letra o algun signo, un mensaje indicado que se debe de corregir el monto

```
import javax.swing.*;
import java.awt.event.*;

public class VentanaDeposito extends JFrame {

    public VentanaDeposito(CuentaBancaria cuenta) {
        setTitle("Depósito");
        setSize(300, 200);
        setLocationRelativeTo(null);
        setLayout(null);

        JLabel lblCantidad = new JLabel("Cantidad a depositar:");
        lblCantidad.setBounds(30, 30, 200, 25);
        add(lblCantidad);

        JTextField txtCantidad = new JTextField();
        txtCantidad.setBounds(30, 60, 200, 25);
        add(txtCantidad);

        JButton btnDepositar = new JButton("Depositar");
        btnDepositar.setBounds(30, 100, 100, 30);
        add(btnDepositar);

        btnDepositar.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                    double cantidad = Double.parseDouble(txtCantidad.getText());
                    cuenta.depositar(cantidad);
                    JOptionPane.showMessageDialog(null, "Depósito realizado correctamente. Saldo actual: $" + cuenta.getSaldo());
                    dispose();
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(null, "Ingresa una cantidad válida.");
                }
            }
        });


        setVisible(true);
```

Finalmente el codigo de la clase ventanaRetiro, esta al igual que la ventana deposito cuenta con su boton de accion, pero esta esta equipada con un mensaje extra que se activa en caso que la cantidad a retirar sea mayor a el saldo disponible en la cuenta

```
import javax.swing.*;
import java.awt.event.*;

public class VentanaRetiro extends JFrame {

    public VentanaRetiro(CuentaBancaria cuenta) {
        setTitle("Retiro");
        setSize(300, 200);
        setLocationRelativeTo(null);
        setLayout(null);

        JLabel lblCantidad = new JLabel("Cantidad a retirar:");
        lblCantidad.setBounds(30, 30, 200, 25);
        add(lblCantidad);

        JTextField txtCantidad = new JTextField();
        txtCantidad.setBounds(30, 60, 200, 25);
        add(txtCantidad);

        JButton btnRetirar = new JButton("Retirar");
        btnRetirar.setBounds(30, 100, 100, 30);
        add(btnRetirar);

    btnRetirar.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            try {
                double cantidad = Double.parseDouble(txtCantidad.getText());
                boolean exito = cuenta.retirar(cantidad);

                if (exito) {
                    JOptionPane.showMessageDialog(null, "Retiro realizado correctamente. Saldo actual: $" + cuenta.getSaldo());
                    dispose();
                } else {
                    JOptionPane.showMessageDialog(null, "Fondos insuficientes o cantidad inválida.");
                }

            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(null, "Ingresa una cantidad válida.");
            }
        }
    });


        setVisible(true);
    }
```

Conclusión

La realización de esta actividad permitió aplicar de manera práctica los principios de la programación orientada a objetos mediante el desarrollo de un sistema bancario con interfaz gráfica en Java. A lo largo del proyecto se diseñó un menú interactivo que facilita al usuario la realización de depósitos y retiros, con una retroalimentación visual clara y validaciones que aseguran la integridad de los datos ingresados. Esta implementación refuerza la importancia de utilizar estructuras de control como if y switch, así como el uso de clases, métodos y encapsulamiento para mantener una arquitectura modular y escalable. Además, la migración desde un sistema en consola hacia una interfaz gráfica mediante el uso de componentes Swing, representa un avance significativo en la creación de aplicaciones orientadas al usuario final. Este ejercicio no solo fortalece habilidades técnicas, sino que también desarrolla la capacidad de análisis, detección de errores y mejora continua, aspectos esenciales para el desempeño profesional en el ámbito del desarrollo de software.

Referencias

Valenzuela Aridjis, J. H. (2025). Repositorio académico de actividades: Lenguajes de Programación IV [Repositorio GitHub]. GitHub. https://github.com/Aridjis/LDP4.git