

## Actividad | #2 | Programa Banco

### Mexicano (parte 1)

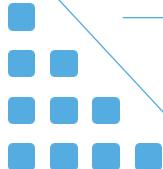
Ingeniería en Desarrollo de Software



TUTOR: Aarón Iván Salazar Macías

ALUMNO: Edgar Enrique Cuamea Ochoa

FECHA: 19 de noviembre del 2025



## Contenido

Introducción.....	3
Descripción.....	4
Justificación.....	5
Desarrollo.....	8
Conclusión.....	22
Referencias.....	24

## **Introducción.**

En esta actividad continuaremos con lo visto en la actividad anterior donde realizamos un programa para mostrar nuestro índice de masa corporal, en este caso realizaremos una interfaz gráfica para un programa que debe de tener algunas características en su uso, en este caso se necesita tener un menú que tenga las diferentes opciones a seleccionar como depósitos, retiros, consulta de saldos y un botón adicional para poder salir del programa, al realizar un depósito, el programa debe de pedir la cantidad a depositar y si requiere realizar otro depósito para volver a capturar datos, si no es así, deberá de regresarnos a la pantalla principal, el programa debe de pedir la cantidad del retiro en caso de elegir esa opción, diseñaremos el programa con las herramientas que nos proporciona el programa de NetBeans siguiendo el paradigma de la programación orientada a objetos definiendo las acciones de cada botón y celdas de texto que realizaremos en NetBeans.

### **Descripción.**

En esta ocasión realizaremos un programa que podría funcionar en un cajero automático, ya que principalmente se utilizara para la consulta de saldos, realizar depósitos y retiros en este programa, usaremos el programa de NetBeans para el diseño de la interfaz, utilizando etiquetas de texto, cuadros donde podemos ingresar texto además de diferentes botones de confirmación por lo que diseñando esta interfaz aprenderemos sobre el diseño de las interfaces además de modificar las diferentes acciones que se realizaran en el programa como los diferentes botones de selección para los depósitos o retiros además de los botones de confirmación y los botones para poder salir del programa por lo que en este caso también debemos poner atención a la gramática de las diferentes acciones que programaremos para evitar errores en el código por lo que un error puede hacer que el programa no compile y no funcione correctamente o ni siquiera podamos abrir el programa ya que un solo símbolo o mayúscula que pongamos mal puede estropear el código.

### **Justificación.**

Utilizaremos NetBeans como programa para desarrollar nuestro programa de banco, utilizaremos java 8 para poder realizar el programa que codificaremos, utilizaremos diferentes formas como textos, cuadros para ingresar textos y botones para el diseño de las páginas además de codificar las diferentes funciones para poder realizar los depósitos y retiros además de mostrar los diferentes mensajes del programa como el mensaje de depósito realizado exitosamente o de retiro, por lo que aprenderemos sobre la sintaxis de la programación orientada a objeto creando clases para las diferentes funciones además de poder realizar una interfaz gráfica funcional ya que este programa debe de ser fácil de entender y debe de ser funcional por lo que tendremos que tener cuidado en la codificación además de poder practicar con otro lenguaje de programación como java ya que nos servirá para poder encontrar similitudes con los demás lenguajes de programación como C++, C# y algunos otros ya que podremos encontrar algunas similitudes en estos lenguajes así como su estructura.

### Investigación.

Investiga un ejemplo de cómo declarar una clase con sus propiedades y métodos en cada uno de los siguientes lenguajes de programación:

1.Java. En este lenguaje se utiliza la palabra public class para declarar una clase publica seguido del nombre y corcheas para el cuerpo definiendo atributos y otras opciones

```
public class NombreDeLaClase{
```

```
private string Nombre; para declarar atributos
```

```
public int sumar(int a, int b) {
```

```
return a + b;
```

```
}, de esta forma creamos una clase, definimos un atributo y un método que devolvería un resultado de una suma, solo deberíamos definir que variables se utilizaran antes de la operacion
```

2.Python. En este lenguaje es mas sencillo solo con las palabras class NombreDeLaClase:

Los atributos se definen con los comandos def\_init(self, valor1, valor2):

```
Self.atributo1 = valor1
```

```
Self.atributo2 = valor 2
```

Para declarar métodos se utiliza el comando def

```
def suma(self):
```

```
return f {valor1 + valor2};
```

3.C#. Para este lenguaje de programación se utiliza una sintaxis similar que java ya que se utiliza

public class NombreDeLaClase{ } utilizando la misma gramática solo que antes de este código se debe de poner el código namespace classes; para poder organizar el código, dentro de la corchea van los atributos y métodos como public string color; y el método dentro de otra corchea {get {return color;}}

Set{ color= value:}

4.C++. Similar a los lenguajes anteriores, este lenguaje utiliza la palabra class NombreDeLaClase {} seguido de las variables public o private dentro de la corchea para declarar atributos como Int edad; y para los métodos es void edad(){std::cout<<“ esta es mi edad”<< std::endl;

5.PHP. en este lenguaje de programación se utiliza una sintaxis parecida a las anteriores con el nombre Class NombreDeLaClase{} dentro de las corcheas se ingresaran los diferentes atributos, los métodos los definimos con el código función utilizando anteriormente si queremos que el método sea publico, privado o protected.

## Desarrollo.

Contextualización:

Los clientes de Banco Mexicano necesitan un programa que les permita a sus clientes el realizar depósitos, retiros y consultas de su saldo. Por lo que necesitan que un ingeniero en desarrollo de software genere una base de datos que atienda a esta necesidad.

Utilizar el lenguaje Java 8 y el entorno de programación para realizar un programa con los siguientes requerimientos:

La pantalla principal debe contar con un menú que tenga las siguientes opciones. Además, deberá solicitar la respuesta por teclado:

Depósito

Retiro

Saldo

Salir

Respuesta: En caso de que el usuario ingrese la opción **Depósito**, el programa deberá ser capaz de capturar por teclado los siguientes datos:

Cantidad a depositar

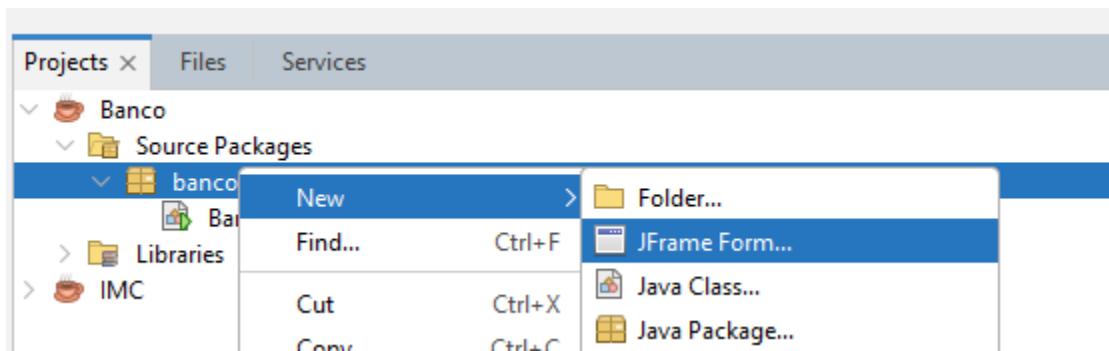
Preguntar si desea realizar otro depósito; en caso de que se seleccione la opción “Si”, deberá mostrar nuevamente la pantalla de Depósito, si la respuesta es “No”, deberá mandar al menú principal.

En caso de que el usuario ingrese la opción **Retiro**, el programa deberá ser capaz de capturar por teclado los siguientes datos:

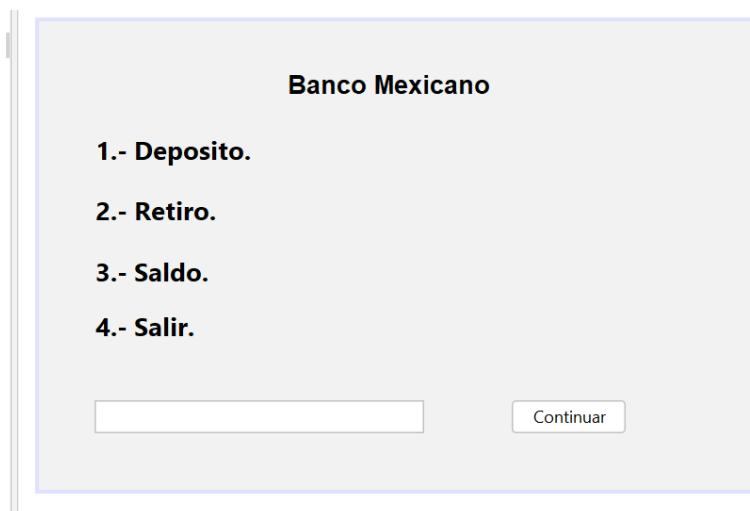
Cantidad a retirar.

Posteriormente deberá de mandar al menú principal.

Con la contextualización anterior realizamos un nuevo proyecto y agregamos un nuevo JFrame Form



Una vez creada nuestro JFrame, crearemos el siguiente cuadro con las herramientas que nos proporciona NetBeans



Realizamos esta ventana para con las diferentes opciones a elegir además de un cuadro donde debemos ingresar el numero de la opción a la que queramos entrar por lo que, una vez diseñado el menú, realizaremos la página de depósito y retiro creando 2 diferentes jframes



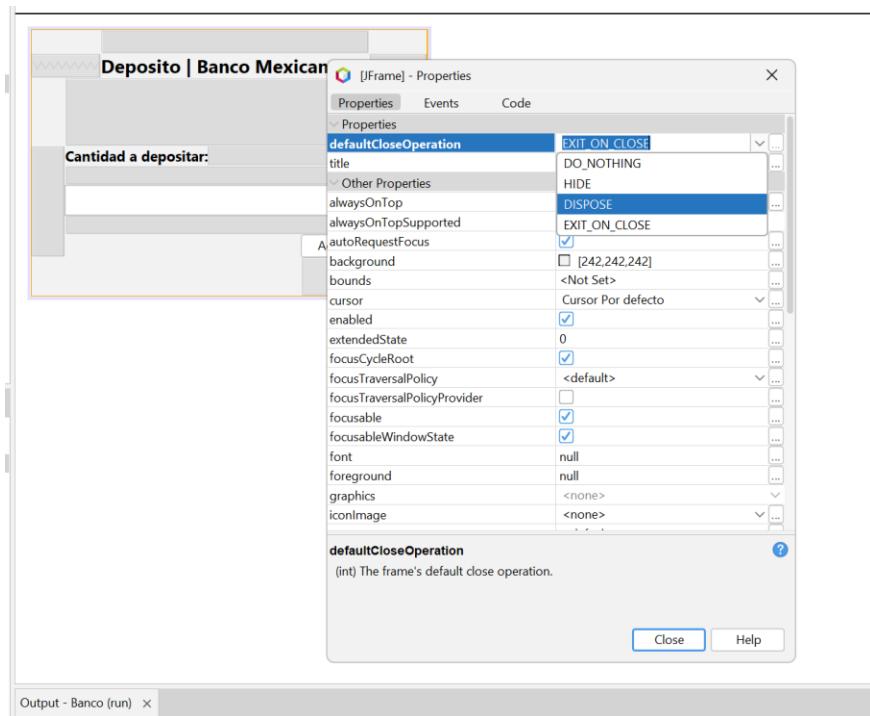
En este caso solo realizare el diseño de las páginas ya que cambiare después el nombre de los diferentes elementos al momento de utilizarlos, una vez creada la página de depósitos, realizaremos lo mismo con los retiros,



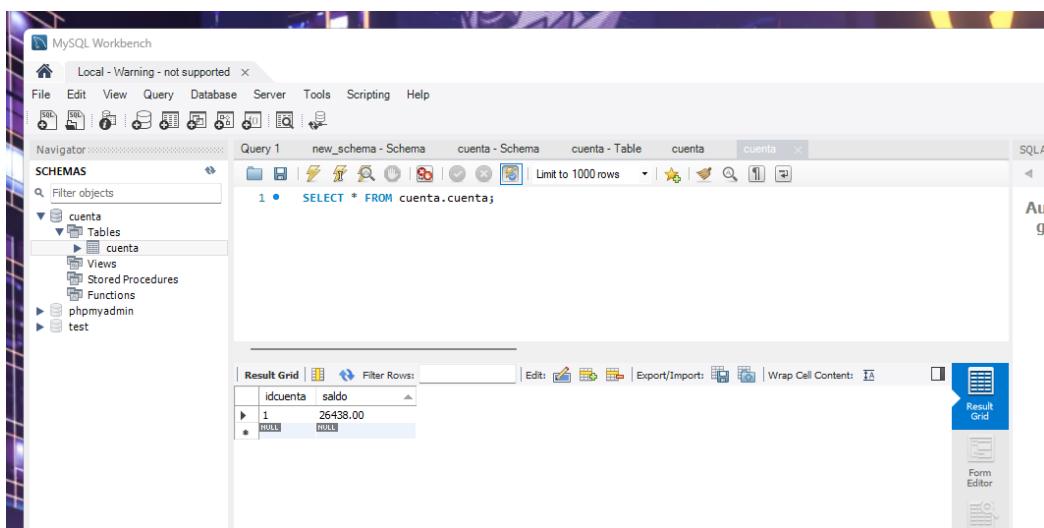
Una vez que tengamos los diseños, programaremos la función para poder seleccionar el retiro y el depósito por lo que creamos un evento en el botón de continuar en nuestro frame de menu y escribimos el código en el siguiente espacio

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    try{  
        String opcion = JtfOpcion.getText();  
        int opc = Integer.parseInt(opcion);  
  
        switch(opc){  
            case 1:  
                // en esta opcion abriremos nuestra pagina de Depositos  
                Deposito deposito = new Deposito();  
                deposito.show();  
                break;  
            case 2:  
                // en esta pagina abriremos la opcion de retiros  
                Retiro retiro = new Retiro();  
                retiro.show();  
                break;  
            case 3:  
                // en esta opcion abriremos la pagina de saldos  
                Saldo saldo = new Saldo();  
                saldo.show();  
                break;  
            default:  
                // saldremos de la aplicacion  
                this.dispose();  
                break;  
        }  
  
    } catch(Exception e){  
        JOptionPane.showConfirmDialog(null,"Error" + e);  
    }  
}
```

Después de presionar el botón de depositar creamos un try catch que ejecutara el código que escribamos dentro del try y mostrara un error en caso de tenerlo con el catch, con string opcion registraremos el texto escrito en nuestro cuadro de texto para las diferentes opciones, en este caso le cambiamos el nombre a JtfOpcion, el valor tomado será un entero y lo agregamos a la variable opc, definimos un switch para tomar diferentes rutas dependiendo de el caso, en el caso del Deposito, crearemos una variable llamada deposito para poder llamar el JFrame llamado Deposito, de igual forma con los demás casos llamando sus respectivos Frames, en caso de dar continuar y no ingresar una opción previamente, mostrara un error con el catch del final

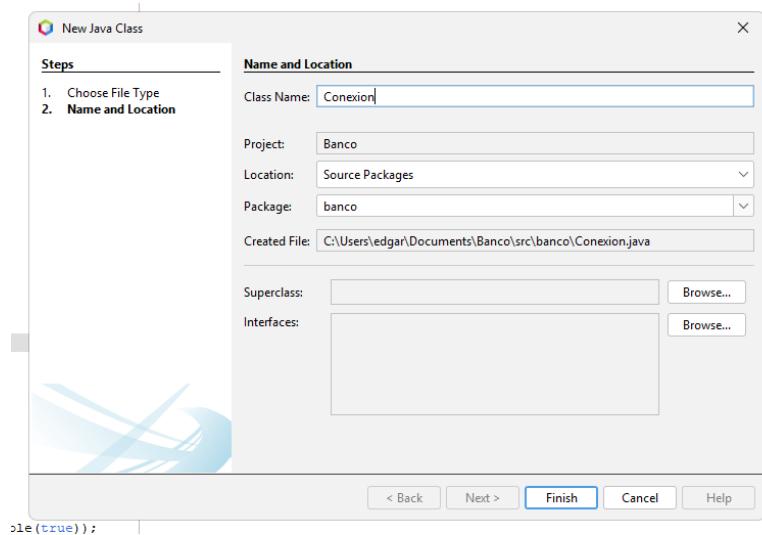


Modificamos cada uno de los frames seleccionando la opción de dispose en cada frame para cerrar la ventana del frame actual y no toda la aplicación



Una vez realizado estos pasos, creamos una base de datos, en nuestro caso utilizamos MYSQL workbench, ya que es parecido a SQL management studio además de probar una aplicación de

bases de datos diferente de las que he utilizado, asignamos nuestra base de datos, ingresamos tablas con sus respectivos valores así como algunos datos en nuestro saldo para poder verificar después que la conexión y la aplicación funcionan correctamente



Creamos una nueva clase en nuestro proyecto de banco llamada conexión donde escribiremos el siguiente código

```

import java.sql.Connection;
import java.sql.DriverManager;
import javax.swing.JOptionPane;

/*
 * @author edgar
 */
public class Conexion {
    public Connection getConnection() {

        Connection con = null;
        String base = "cuenta";
        String url = "jdbc:mysql://localhost:3306/" + base;
        String user = "root";
        String password = "";

        try {
            Class.forName("com.mysql.jdbc.Driver");
            con = (Connection) DriverManager.getConnection(url, user, password);
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Error" + e);
        }
        return null;
    }
}

```

En este caso importamos 3 paquetes que necesitamos, una para la conexión con SQL, una para los

drivers y la de JOptionPane para poder mostrar mensajes, creamos una clase publica llamada conexión y asignamos sus atributos donde esta clase cuenta con una variable llamada connection que debe de realizar la conexión con la base de datos, a esta variable la conectamos con la variable con para poder recoger datos, ingresaremos a la base de datos llamada cuenta con la url de nuestra base de datos local donde nuestro usuario será root ya que no asignamos uno y la contraseña en blanco, creamos un try catch donde realizamos la conexión con la base de datos creada en SQL y mostrara un error en caso de haberlo con el catch, una vez realizada la clase conexión realizamos lo siguiente en el evento del botón depositar

```

import java.sql.Connection;
import java.sql.PreparedStatement;
import javax.swing.JOptionPane;

/*
 * @author edgar
 */
public class Deposito extends javax.swing.JFrame {

    private static final java.util.logging.Logger logger = java.util.logging.Logger.getLogger(Deposito.class.getName());

    /**
     * Creates new form Deposito
     */
    public Deposito() {
        initComponents();
    }

    /**
     * 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")
    // Generated Code

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Connection con;
        Conexion conn = new Conexion();

        try{
            con = conn.getConnection();
            PreparedStatement ps = con.prepareStatement("UPDATE cuenta SET saldo= saldo + ? WHERE idcuenta=1");
            ps.setString(1, Cantidad.getText());

            int res = ps.executeUpdate();
            if(res > 0) {
                JOptionPane.showMessageDialog(null, "Deposito Exitoso");
            } else {
                JOptionPane.showMessageDialog(null, "Error al depositar");
            }
            Cantidad.setText("");
            con.close();
        }catch (Exception e){
        }
    }
}

```

Llamamos a la conexión con la variable con para poder conectarnos a sql y con un try catch preparamos algunos comandos que puede leer SQL donde actualizaremos la base de datos cuenta

además de la tabla saldo donde sumaremos la cantidad ingresada en nuestro cuadro de texto llamado Cantidad en la cuenta donde la idcuenta sea igual a 1, el texto del cuadro de texto será un entero por lo que ejecutamos el comando, si el entero es mayor a 0, mostrara un mensaje de depósito correcto, si no es así, mostrara un error, el cuadro de texto lo dejamos en blanco en caso de realizar otro depósito y cerramos la conexión con SQL

En mi caso tuve errores al realizar la conexión ya que faltaba una librería para poder realizar dichas conexiones así que descargamos la librería desde la página oficial en plataforma independiente compatible con mi sistema operativo Windows 11 en su formato zip

## ④ MySQL Community Downloads

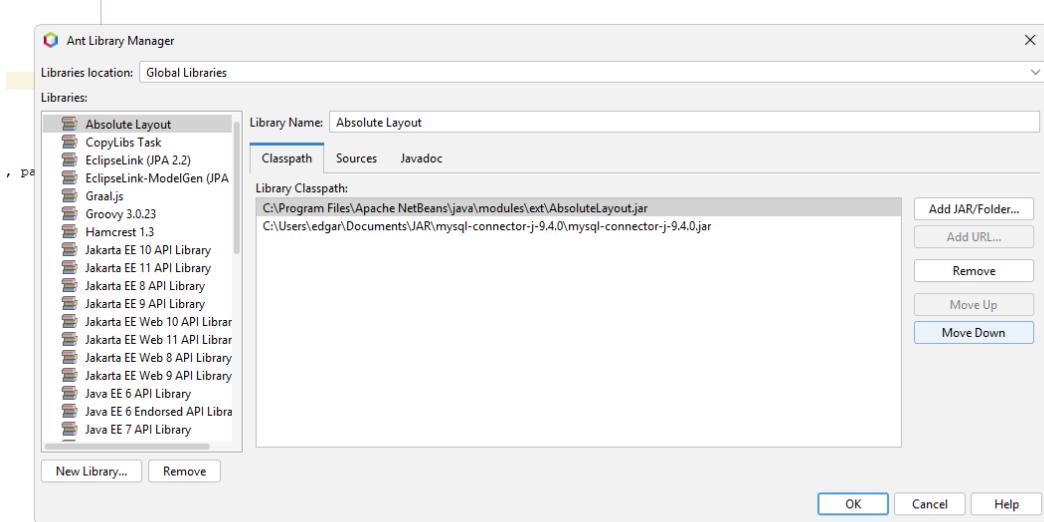
◀ Connector/J

The screenshot shows the MySQL Community Downloads page for the Connector/J 9.5.0 release. At the top, there are tabs for "General Availability (GA) Releases" (which is selected), "Archives", and a help icon. Below the tabs, the title "Connector/J 9.5.0" is displayed. A dropdown menu labeled "Select Operating System" is set to "Platform Independent". Two download options are listed:

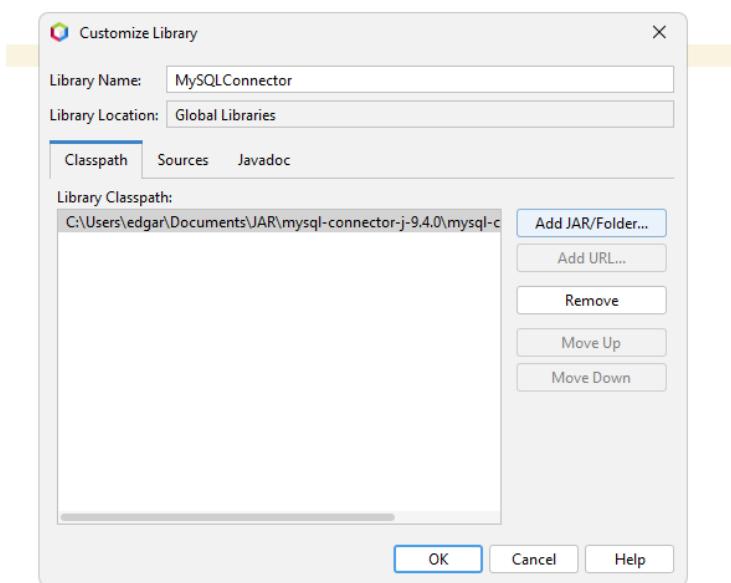
File Type	Version	Size	Action
Platform Independent (Architecture Independent), Compressed TAR Archive (mysql-connector-j-9.5.0.tar.gz)	9.5.0	4.3M	<a href="#">Download</a>
Platform Independent (Architecture Independent), ZIP Archive (mysql-connector-j-9.5.0.zip)	9.5.0	5.1M	<a href="#">Download</a>

Below the download links, a note in a box says: "We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download." It includes links for "MD5 checksums" and "GnuPG signatures".

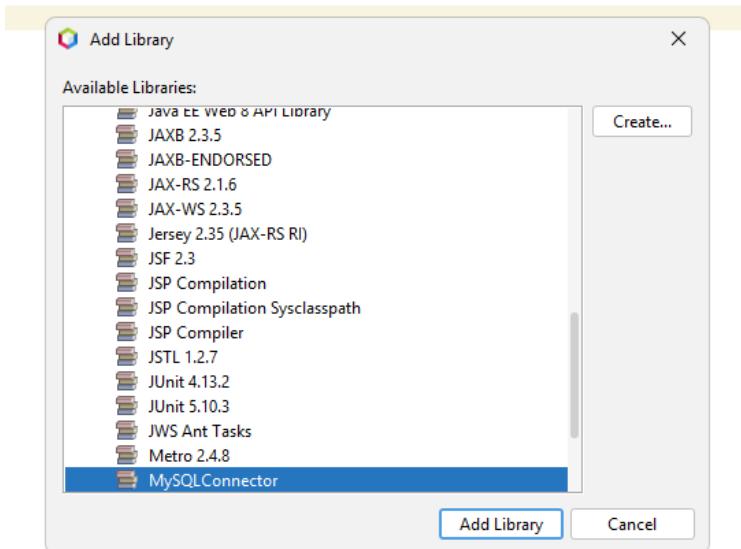
Una vez descargada realizamos lo siguiente



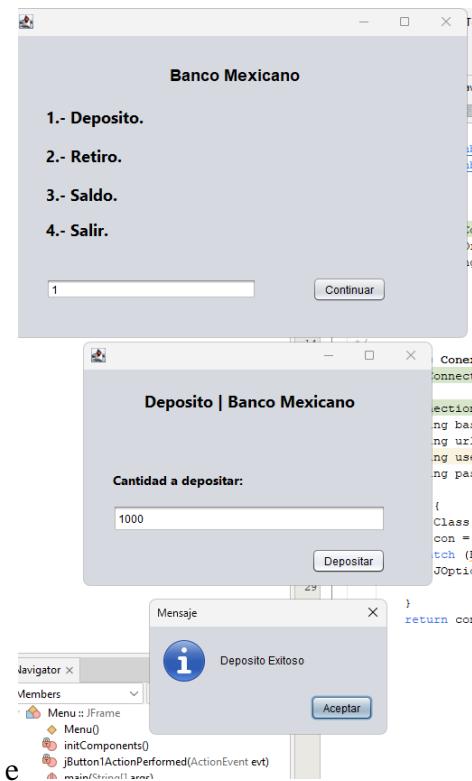
La carpeta zip que descargamos la descomprimimos en una ruta donde tengamos claro done lo realizamos y guardamos el archivo, agregamos el archivo .jar en la librería de NetBeans y aceptamos y guardamos los datos



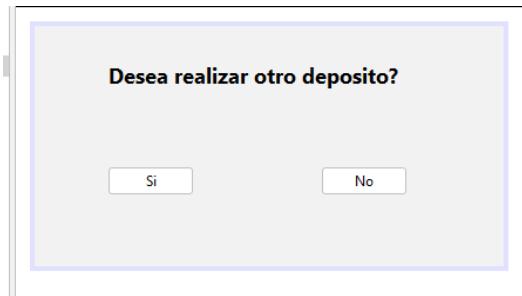
En nuestra librería de nuestro proyecto de banco realizamos lo mismo, asignamos un nombre a la librería y añadimos el archivo que descargamos



En mi caso esta nombrado como MySQLConnector y añadimos la librería, al ejecutar el programa realizamos un deposito



agregamos un nuevo frame para preguntar si queremos realizar otro deposito



Al presionar sobre si, cerramos la ventana con this.dispose(); y podremos realizar otro depósito.

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{ this.dispose();
        } catch(Exception e){
            JOptionPane.showConfirmDialog(null,"Error" + e);
    }
}

```

Al presionar no, cerrara las 2 ventanas regresandonos a la pantalla principal con este código.

En nuestra código de deposito

```

Cantidad.setText("");
con.close();

Deposito2 deposito2 = new Deposito2(this);
deposito2.setVisible(true);

} catch (Exception e) {

```

Al final del código agregamos la opción que nos abrirá el jframe para confirmar si queremos realizar otro deposito, por lo que asignamos el jframe deposito a la variable deposito2 con la palabra this para poder obtener la referencia del jframe desde otro jframe, en este caso desde el jframe donde cerraremos las ventanas

```


    | */
    public class Deposito2 extends javax.swing.JFrame {

        private static final java.util.logging.Logger logger = java.util.logging.Logger.getLogger(Deposito.class.getName());
        private javax.swing.JFrame VentanaDepositoAnterior;

        /** Creates new form Deposito2 ...3 lines */
        public Deposito2(){
            this(null);
        }
        public Deposito2(javax.swing.JFrame depositoRecibido) {
            initComponents();
            this.VentanaDepositoAnterior = depositoRecibido;
        }
    }


```

Añadimos una variable publica para poderla llamar en alguna otra pagina, llamada ventanadepositoanterior con la función javax.swing.JFrame que controla operaciones como cerrar ventanas y algunas etiquetas, botones y diferentes opciones, con public Deposito2(){ this(null)} permitimos que el compilador funcione sin dar la referencia del deposito, con this.ventanadepositoanterior = deposito recibido le damos la capacidad de poder acceder al JFrame de deposito desde esta pagina para poder cerrarla, al final escribimos lo siguiente

```


99
100    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
101        // TODO add your handling code here:
102        if (VentanaDepositoAnterior != null){
103            this.VentanaDepositoAnterior.dispose();
104        }
105        this.dispose();
106    }
107


```

Al presionar el botón no, realizamos una comparativa, si la ventana de depósito es diferente a null, entonces cerraremos la ventana con la variable que tiene guardada la referencia del JFrame de deposito con .dispose y cerraremos el cuadro que abrimos con this.dispose, una vez realizando verificamos que el depósito se realizó exitosamente por lo que verificamos el saldo de la cuenta en nuestra base de datos y vemos que aumento la cantidad del saldo

The screenshot shows a MySQL Workbench interface with a 'Result Grid' tab selected. A single row is displayed with the following data:

	idcuenta	saldo
▶	1	27439.00
*	NULL	NULL

Copiamos el código para el depósito y lo ingresamos al evento del botón retirar y cambiamos algunos datos como el nombre del cuadro de texto de los retiros, así como los mensajes del retiro exitoso y error al retirar además de modificar el comando SQL para que nos reste la cantidad tomada del cuadro de texto, todo el código será parecido con las mismas funciones de llamar la conexión con la base de datos

```

Generated Code

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Connection con;
    Conexion conn = new Conexion();

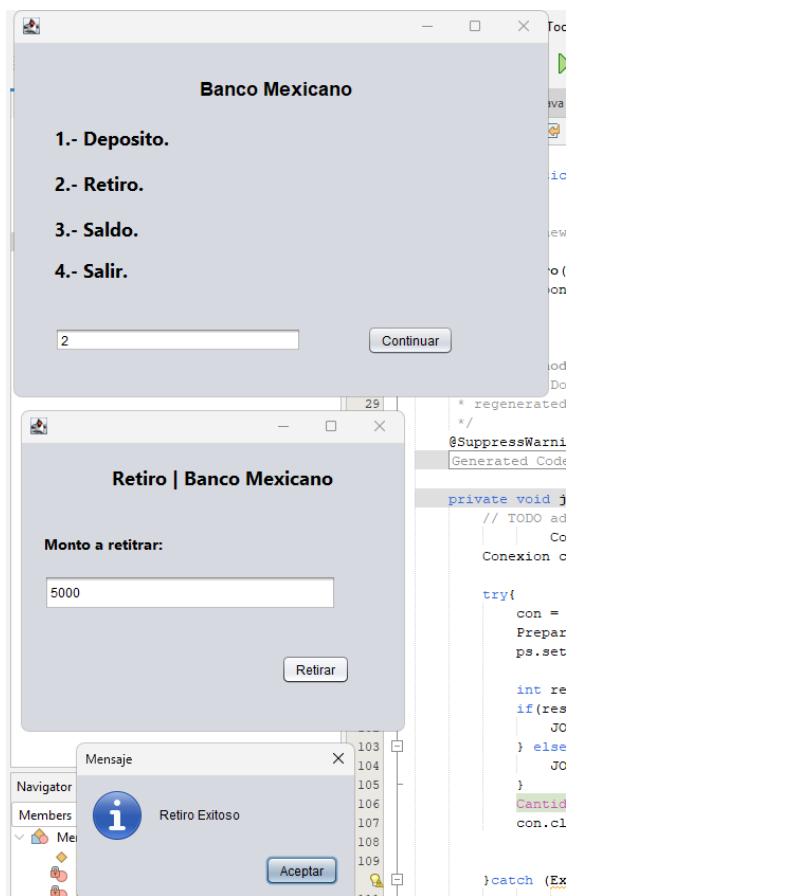
    try{
        con = conn.getConnection();
        PreparedStatement ps = con.prepareStatement("UPDATE cuenta SET saldo= saldo - ? WHERE idcuenta=1");
        ps.setString(1, Cantidad.getText());

        int res = ps.executeUpdate();
        if(res > 0) {
            JOptionPane.showMessageDialog(null, "Deposito Exitoso");
        } else {
            JOptionPane.showMessageDialog(null, "Error al retirar");
        }
        Cantidad.setText("");
        con.close();

    }catch (Exception e){
    }
}

```

En este caso no preguntaremos si queremos realizar otro retiro, solo cerraremos la ventana para volver al inicio



Al realizar el retiro en la aplicación verificamos que el saldo de nuestra cuenta en la base de datos sea restado correctamente por lo que vemos lo siguiente

Result Grid		Filter Rows:	Edit:
	idcuenta	saldo	
▶	1	22439.00	
*	NULL	NULL	

Agregamos una línea al final del código de retiros poniendo lo siguiente

```

}
Cantidad.setText("");
con.close();
this.dispose();

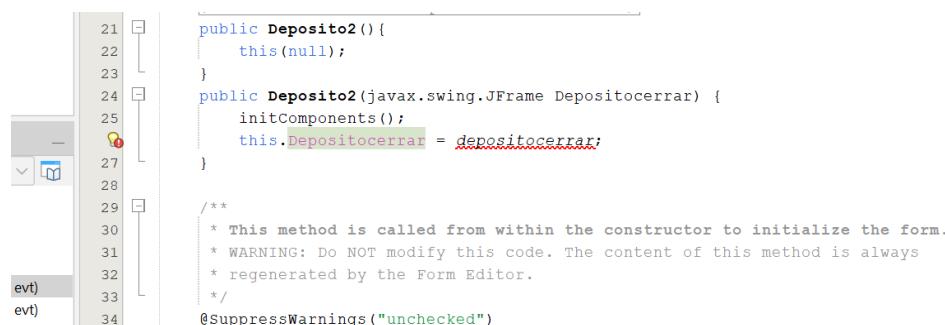
```

Esto hará que al final donde se muestra el mensaje de retiro exitoso, cerrara la pagina de retiro y nos regresara al menú donde podremos seleccionar otra opción.

De esta forma podremos crear una conexión con la base de datos donde en la siguiente actividad realizaremos la consulta de saldos, así como la opción de salir de la aplicación y algunas mejoras en la página para poder mostrar

### Conclusion.

Con la realización de esta actividad tuve muchos problemas a la cerra la página de depósito desde la página de depósito 2 para mandarnos a la página principal, así como la conexión con la base de datos, en el código de deposito 2 donde preguntamos si se desea realizar otro deposito realice lo siguiente



```

21
22
23
24
25
26
27
28
29
30
31
32
33
34

```

```

public Deposito2() {
    this(null);
}
public Deposito2(javax.swing.JFrame Depositocerrar) {
    initComponents();
    this.Depositocerrar = Depositocerrar;
}

/**
 * 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")

```

Tengo un error de gramática al presentar la inicial como mayúscula y después como minúscula, realice los cambios y aún tenía errores cambie las variables a deposito ya que pensé que de esta forma llamaría a la pantalla de deposito para poderla cerrar por lo que no funciono y aun así tenia

errores ya que el compilador no sabía si llamábamos una clase o era algún método por lo que termine investigando y cambiando las variables hasta que dejara de tener un error, ya que tuve muchos problemas al agregar la clase privada para que solo sea visible en esta jframe así como esta función de poder recibir la referencia de la página depósito en la de deposito2 ya que al cerrarlo solo realizamos la comparativa, si es diferente a null, lo cerramos, con la conexión de la base de datos tuve algunos problemas ya que debemos de descargar las librerías e instalarlas manualmente ya que en visual studio la podremos descargar desde el propio IDE por lo que utilizamos xamp y my sql para la base de datos, su funcionamiento es similar a otras aplicaciones de bases de datos que hemos utilizado anteriormente por lo que fue fácil crear las tablas y atributos, solo en lo que tarde mas fue en hacer que la una ventana pudiera cerrar otra y devolvernos a la pantalla principal ya que establecí variables que se llamaban depositocerrar y Deposito lo que confundía la variable con el jframe y me daba errores en su compilación además de cambiar las variables para poder cerrar la pagina

## Referencias.

*Video conferencing, web conferencing, webinars, screen sharing.* (s. f.-f). Zoom.

[https://academiaglobal-](https://academiaglobal-mx.zoom.us/rec/play/AOhI8WAFo7DxH3EqlDdRdcZQIv6fP9kd81AkELNRrDC7q9bPYUWYgg0G8l5uT_jrb_JyhIZQHE6-8eMo.xe64afpEXSfzoxJ?eagerLoadZvaPages=sidemenu.billing.plan_management&isReferralProgramEnabled=false&isReferralProgramAvailable=false&accessLevel=meeting&canPlayFromShare=true&from=share_recording_detail&continueMode=true&componentName=rec-play&originRequestUrl=https%3A%2F%2Facademiaglobal-mx.zoom.us%2Frec%2Fshare%2FvUaUixJZGeIsu5-Crs-LZEnacXzGevNzgNI5CSBVmccn_Dq9vcYF25tjq-nb1KHe.dFLgOOjwhP-b-te8)

[mx.zoom.us/rec/play/AOhI8WAFo7DxH3EqlDdRdcZQIv6fP9kd81AkELNRrDC7q9bPYUWYgg0G8l5uT\\_jrb\\_JyhIZQHE6-8eMo.xe64afpEXSfzoxJ?eagerLoadZvaPages=sidemenu.billing.plan\\_management&isReferralProgramEnabled=false&isReferralProgramAvailable=false&accessLevel=meeting&canPlayFromShare=true&from=share\\_recording\\_detail&continueMode=true&componentName=rec-play&originRequestUrl=https%3A%2F%2Facademiaglobal-mx.zoom.us%2Frec%2Fshare%2FvUaUixJZGeIsu5-Crs-LZEnacXzGevNzgNI5CSBVmccn\\_Dq9vcYF25tjq-nb1KHe.dFLgOOjwhP-b-te8">https://academiaglobal-](https://academiaglobal-mx.zoom.us%2Frec%2Fshare%2FvUaUixJZGeIsu5-Crs-LZEnacXzGevNzgNI5CSBVmccn_Dq9vcYF25tjq-nb1KHe.dFLgOOjwhP-b-te8)

Hackemi. (s. f.-i). *Hackemi/Lenguajes\_Prog4*. GitHub.

[https://github.com/Hackemi/Lenguajes\\_Prog4](https://github.com/Hackemi/Lenguajes_Prog4)

González, J. D. M. (2021, 29 junio). *Objetos y clases*.

<https://www.programarya.com/Cursos/Java/Objetos-y-Clases>

 *Crear clase.* (s. f.). El Libro de Python. <https://ellibrodepython.com/crear-clase-python>

BillWagner. (s. f.). *Tutorial sobre clases y objetos - C#*. Microsoft Learn.

<https://learn.microsoft.com/es-es/dotnet/csharp/fundamentals/tutorials/classes>

TylerMSFT. (s. f.). *Clases C++ como tipos de valor.* Microsoft Learn.

<https://learn.microsoft.com/es-es/cpp/cpp/value-types-modern-cpp?view=msvc-170>

*Clases en PHP.* (s. f.). aulab.es. [https://aulab.es/articulos-guias-avanzadas/43/clases-en-](https://aulab.es/articulos-guias-avanzadas/43/clases-en-php)

[php](#)