

Actividad | #3 | Programa Banco

Mexicano **Lenguajes de**

Programación IV

Ingeniería en Desarrollo de Software



TUTOR: Aarón Iván Salazar Macías

ALUMNO: Erick Alfredo Quiroz Figueroa

FECHA: 08/08/2025

Contenido

Introducción	3
Descripción	4
Justificación	5
Desarrollo.....	6
Interfaz	6
Codificación	8
Conclusión	17

Introducción

Como sabemos la programación en Java es una de las mas importantes hoy en día ya que gracias a las tecnologías Java fue el primer lenguaje en usar API en sus aplicaciones, y, por consiguiente, el primero en usar al 100% el paradigma orientado a objetos ya que si nos basamos en la historia el primero en usarlo fue C++ con la utilización de clases y objetos y todo lo que conlleva dicho paradigma.

Pero Java utilizo una interfaz única para poder utilizar los componentes de un programa de una manera mas sencilla creando una IDE exclusiva para su programación eficiente.

Para que un programa sea ejecutado correctamente en nuestro equipo es necesario correrlo en la maquina virtual de Java que por sus siglas en ingles podemos encontrarlo como JVM y se puede instalar en un paquete que la misma empresa proporciona para que sea fácil usar el intérprete.

Como todos los lenguajes de programación que existen podemos encontrar un kit con las librerías necesarias en el lenguaje Java para que las funciones y métodos a utilizar en la codificación hagan funcionar el programa sin ningún error en él.

Descripción

Como pudimos observar en la actividad anterior, creamos una aplicación en el lenguaje Java en la cual hicimos la simulación de un cajero automático, el cual hace los siguientes movimientos: “Deposito”, “Retiro”, “Consulta de Saldo” y “Salir”. En cada una de las opciones programamos la opción de que al ingresar una cantidad nos mandara una caja de texto en la que nos dijera o nos preguntara si queríamos realizar otro deposito o retiro según sea el caso, y conforme a lo elegido pasar al siguiente movimiento.

Ahora dejando en la aplicación esos eventos programados, realizaremos su complemento que es realizar la conexión con un gestor de base de datos, en este caso con MySQL para poder guardar las cantidades ingresadas y se reflejen en la aplicación o en este caso en la tabla de la Base de Datos creada.

Al momento de ingresar la cantidad nos tendrá que arrojar el mensaje de que el depósito fue exitoso caso contrario será que hubo algún error, de igual manera sucederá el mismo escenario con la opción de retiro.

Justificación

Como sabemos la complejidad de un sistema determina que tan bien este estructurado y organizado el código que hay detrás de este, ya que esto nos permite observar e interactuar de manera sencilla con el programa.

El motivo de estas actividades es poder conocer un poco mas a fondo como es que se crea la interfaz de un programa y como es que se puede programar dándole un funcionamiento optimo empezando por la simplicidad del código, dándole una estructura y entendimiento lo mas sencillo que se pueda realizar.

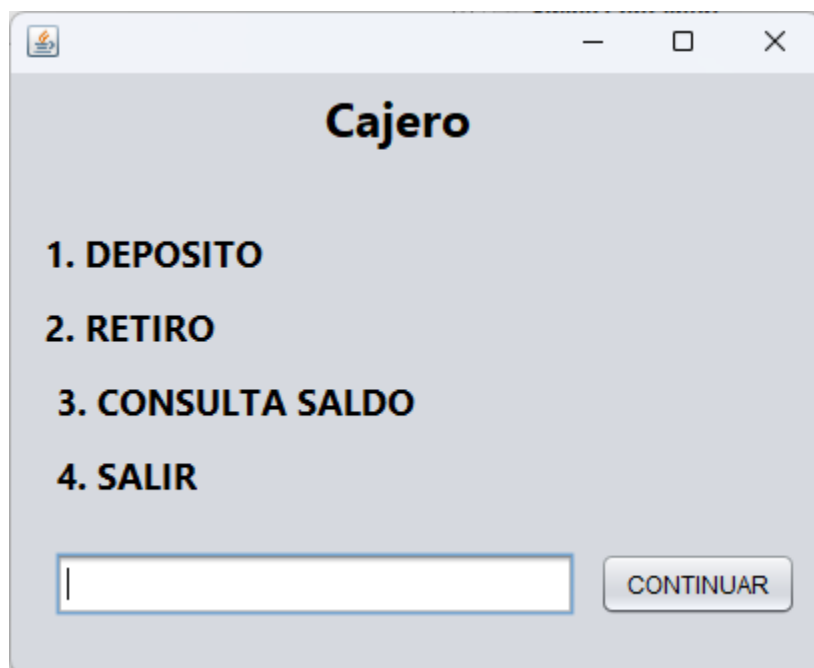
Sabemos que a parte de Java existen cientos de lenguajes de alto nivel que nos permiten crear interfaces graficas de manera sencilla y se pueden hacer conexiones entre dichos lenguajes para tener un mayor control sobre el sistema.

Dicho esto, pasaremos a terminar y realizar la actividad.

Desarrollo

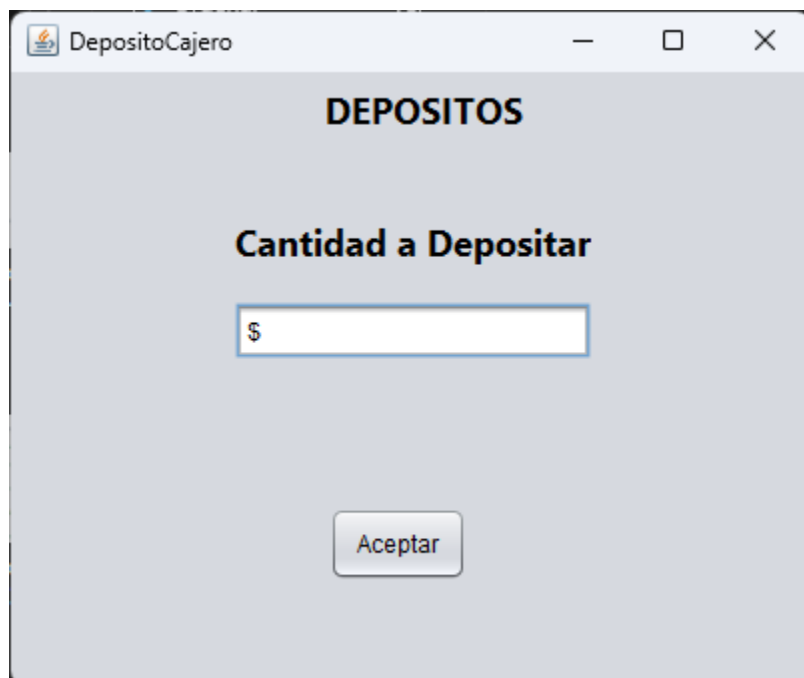
Interfaz

Interfaz gráfica principal del programa con las opciones a utilizar.



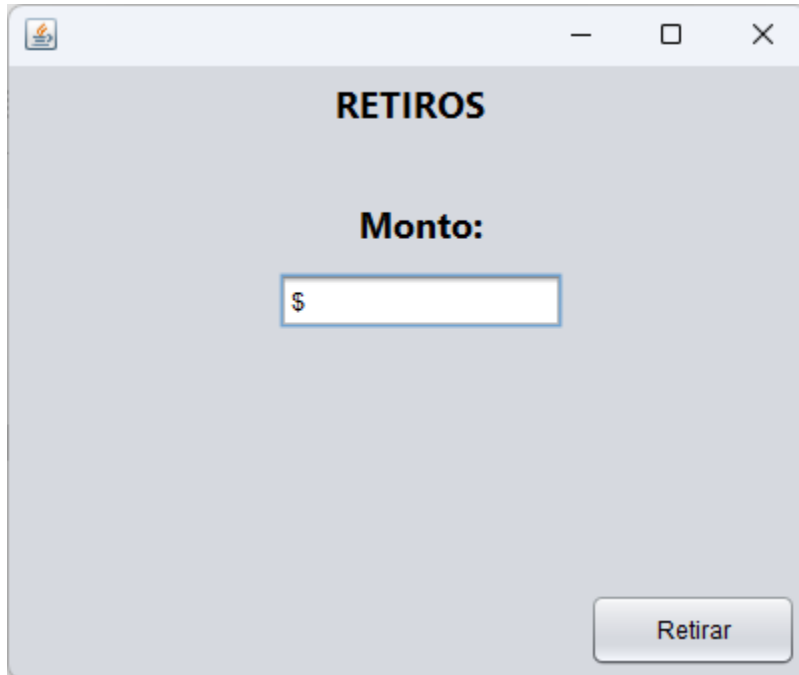
The screenshot shows a window titled "Cajero" with a light gray background. At the top, there is a title bar with a small icon on the left and standard window controls (minimize, maximize, close) on the right. The main content area lists four options in bold black text: "1. DEPOSITO", "2. RETIRO", "3. CONSULTA SALDO", and "4. SALIR". Below these options is a white text input field with a blue border. To the right of the input field is a button labeled "CONTINUAR".

Interfaz de "Depósitos".

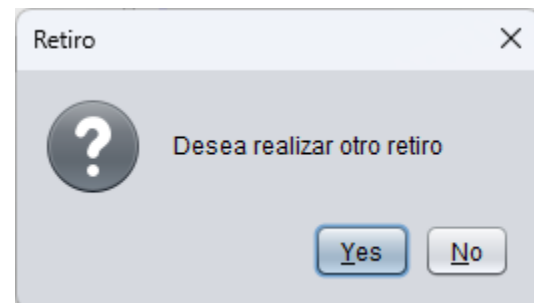
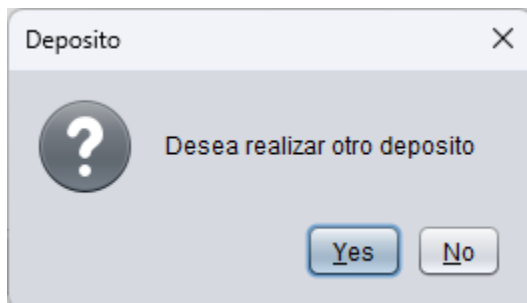


The screenshot shows a window titled "DepositoCajero" with a light gray background. The title bar includes a small icon and window controls. The main content area has the title "DEPOSITOS" in bold black text. Below it is the label "Cantidad a Depositar" in bold black text. Underneath the label is a white text input field with a blue border, starting with a "\$" symbol. At the bottom center of the window is a button labeled "Aceptar".

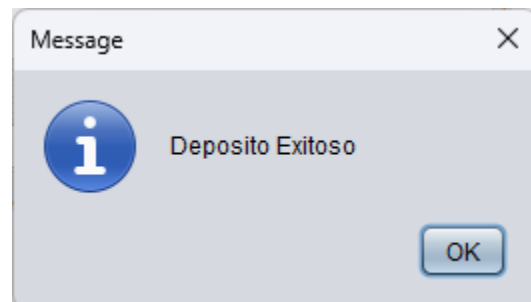
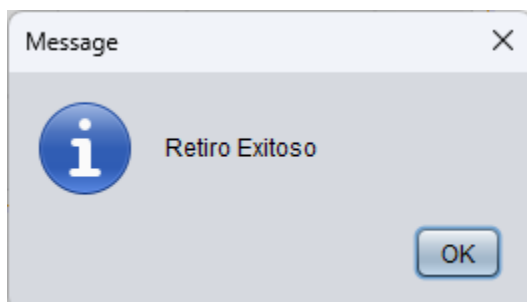
Interfaz gráfica de “Retiros”.



Interfaces de mensajes de confirmación de “Deposito” y “Retiro”.

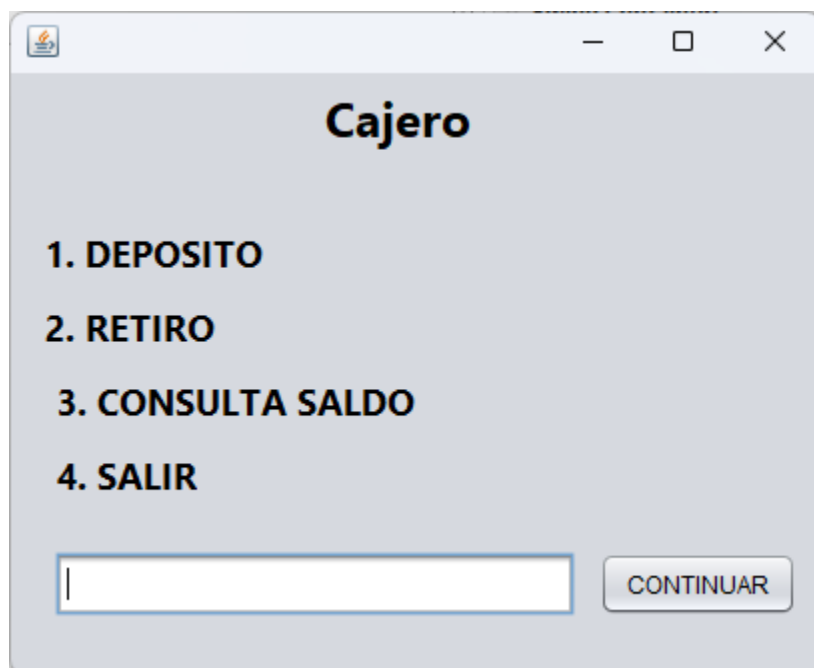


Interfaces de “Deposito” y “Retiro” exitosos. Por alguna extraña razón no me aparecen las interfaces de error.



Codificación

Como punto de partida tenemos la interfaz principal del “Cajero” con las diferentes opciones enumeradas.

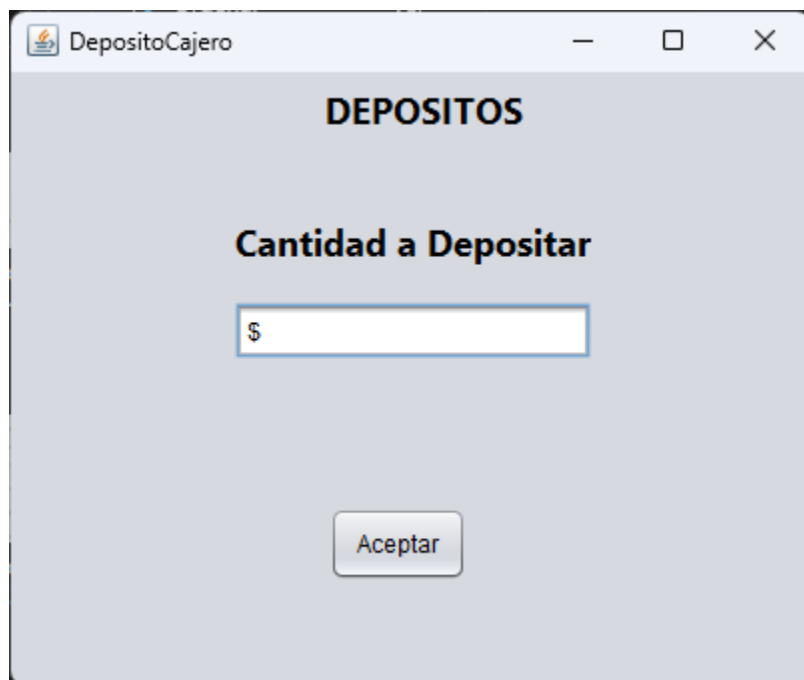


Cajero

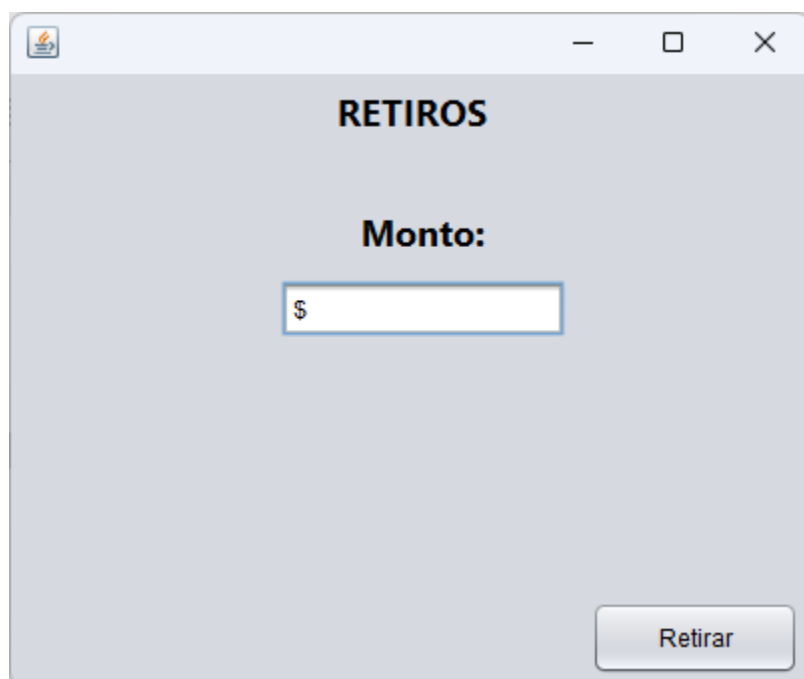
- 1. DEPOSITO**
- 2. RETIRO**
- 3. CONSULTA SALDO**
- 4. SALIR**

CONTINUAR

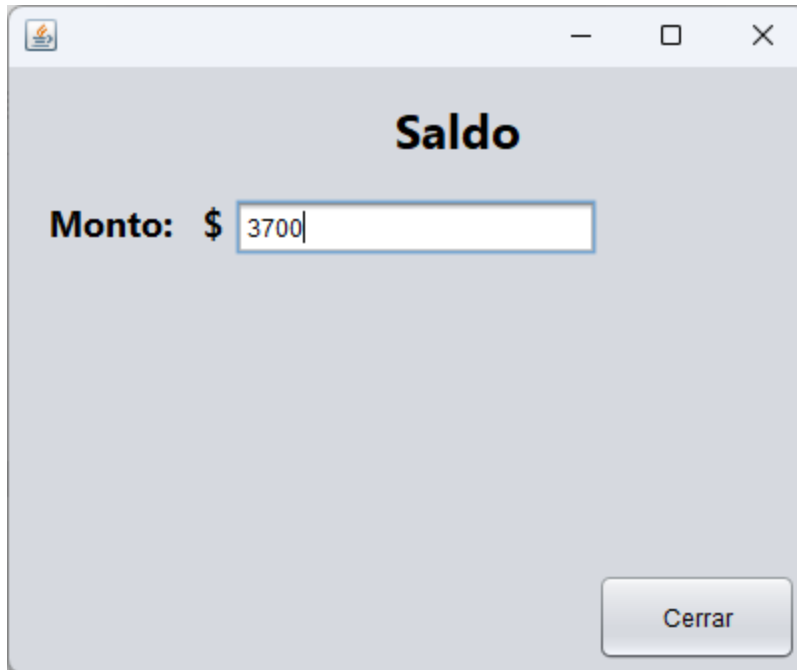
Y de aquí se despliegan las interfaces siguientes.



The screenshot shows a window titled 'DepositoCajero' with standard Windows window controls (minimize, maximize, close). The main content area has a light gray background and is titled 'DEPOSITOS' in bold black text. Below the title, the text 'Cantidad a Depositar' is displayed in bold. Underneath this is a text input field with a blue border, containing a dollar sign '\$'. At the bottom center of the window is a button labeled 'Aceptar'.



The screenshot shows the same 'DepositoCajero' window, but now displaying the 'RETIROS' interface. The title 'RETIROS' is at the top in bold black text. Below it, the text 'Monto:' is displayed in bold. Underneath is a text input field with a blue border, containing a dollar sign '\$'. At the bottom right of the window is a button labeled 'Retirar'.



Como es primordial veremos el código de nuestra interfaz principal que nos ayudara a saltar a la

```
private void jMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
    String opcion = jMenuItem.getText();

    switch (opcion)
    {
        case "1":
            Deposito deposito = new Deposito();
            deposito.show();
            break;
        case "2":
            Retiro retiro = new Retiro();
            retiro.show();
            break;
        case "3":
            Saldo saldo = new Saldo();
            saldo.show();
            break;
        case "4":
            JOptionPane.showMessageDialog(null, "Gracias Por Usar El Cajero UMI");
            this.dispose();
            break;
        default:
            JOptionPane.showMessageDialog(null, "No ha seleccionado ninguna opcion");
            break;
    }

    jMenuItem.setText("");
}
```

siguiente interfaz poniendo el numero de la opción de la siguiente manera:

Para que nuestra interfaz de “Depósitos” funcione tenemos que codificar lo siguiente:

```
private void jBDepositoActionPerformed(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, jTextDeposito.getText());  
  
        int res = ps.executeUpdate();  
    }  
}
```

En esta parte del código hacemos la conexión con la Base de Datos usando el comando *UPDATE* – *SET* el cual nos permite actualizar la información ingresada para reflejarla en la base de datos hecha en MySQL.

A continuación, tenemos el siguiente código:

```

    if(res > 0)
    {
        JOptionPane.showMessageDialog(null, "Deposito Exitoso");
    }
    else
    {
        JOptionPane.showMessageDialog(null, "Error al depositar");
    }
}
catch(Exception e)
{
    System.err.print(e);
}

```

En este fragmento mostramos el uso de nuestra condicional *IF – ELSE* el cual nos ayuda a ver que acciones tomar si se cumple una opcion y que pasa si no se cumple del todo.

Despues, tenemos lo siguiente:

```

int respuesta = JOptionPane.showConfirmDialog(this, "Desea realizar otro deposito", "Deposito",
        JOptionPane.YES_NO_OPTION);
if(respuesta == JOptionPane.YES_OPTION)
{
    Deposito deposito2 = new Deposito();
    deposito2.setVisible(true);
    this.dispose();
}
else
{
    this.dispose();
}

```

Este fragmento al mismo tiempo que muestra el uso de una condicional nos ayuda a ver que se manda llamar un Panel de opciones de Si – No para ver si se realiza otro deposito o no, esto después de haber realizado el depósito.

Después pasamos a la interfaz de “Retiro” donde los códigos tienen la misma función solo que con variables y funciones diferentes a la parte de la interfaz de “Deposito”.

```

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, jTextFieldRetiro.getText());

    int res = ps.executeUpdate();

```

Esta es la parte de la conexión.

Después tenemos la parte de código que pertenece a la los mensajes de situación del retiro, si este fue exitoso o no.

```

        if(res > 0)
        {
            JOptionPane.showMessageDialog(null, "Retiro Exitoso");
        }
        else
        {
            JOptionPane.showMessageDialog(null, "Error al retirar");
        }
    }
    catch(Exception e)
    {
        System.err.print(e);
    }

```

Por último, tenemos la parte del código donde se nos pregunta si se realizara otro retiro o no.

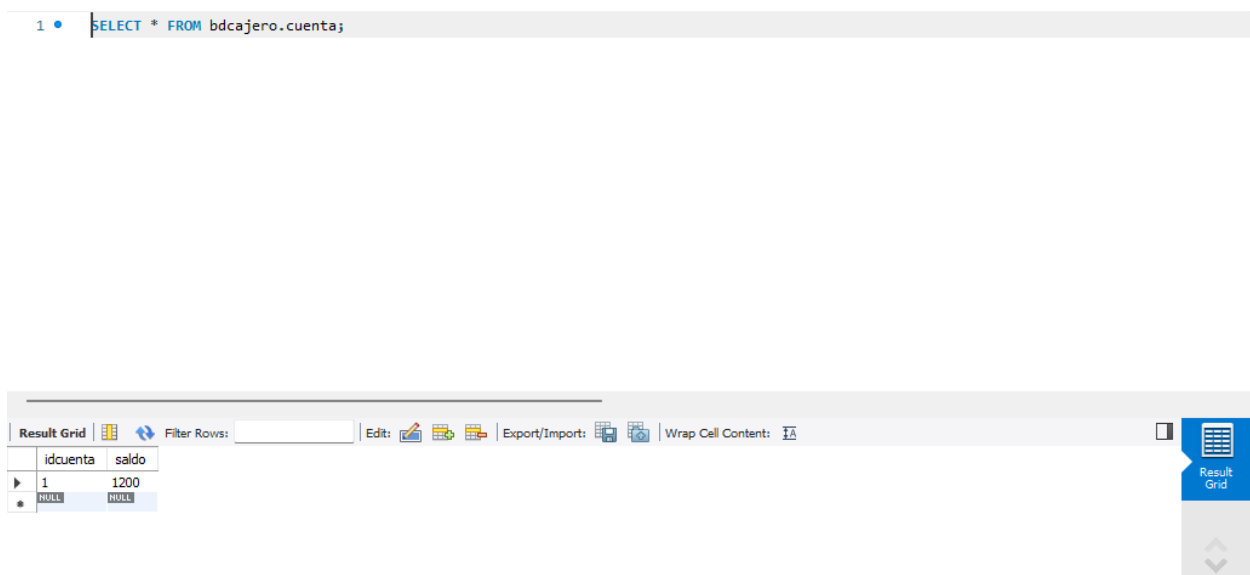
```
int respuesta = JOptionPane.showConfirmDialog(this, "Desea realizar otro retiro", "Retiro",
    JOptionPane.YES_NO_OPTION);
if(respuesta == JOptionPane.YES_OPTION)
{
    Retiro retiro2 = new Retiro();
    retiro2.setVisible(true);
    this.dispose();
}
else
{
    this.dispose();
}
```

Para la consulta de saldo tenemos que hacer la creación de la BD en MySQL lo cual el código quedaría de la siguiente manera:

```
create database bdcajero;

create table cuenta
(
    idcuenta int primary key AUTO_INCREMENT not null,
    saldo float not null
);
```

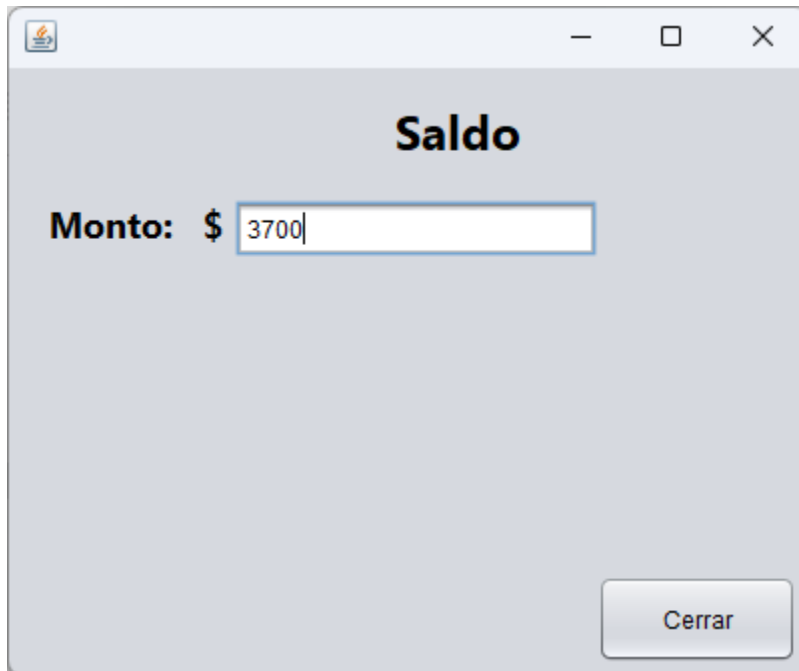
Y ya corriendo o haciendo la consulta de nuestra BD quedaría algo así:



The screenshot shows a MySQL database interface. At the top, a query is entered in the SQL editor: `SELECT * FROM bdcajero.cuenta;`. Below the editor, the results are displayed in a table with two columns: `idcuenta` and `saldo`. The table contains three rows: a row with `1` and `1200`, a row with `NULL` and `NULL`, and a row with `NULL` and `NULL`. The interface includes a toolbar with various icons for editing, exporting, and filtering, and a sidebar on the right with a 'Result Grid' button.

idcuenta	saldo
1	1200
NULL	NULL
NULL	NULL

Para que nuestra interfaz de saldo salga con datos ingresados de esta manera:



Es necesario generar y programar el código como se muestra a continuación:

```
public Saldo() {
    initComponents();

    Connection con;
    Conexion conn = new Conexion();

    try
    {
        con = conn.getConnection();

        PreparedStatement ps = con.prepareStatement("SELECT saldo FROM cuenta WHERE idcuenta = 1");

        ResultSet rs = ps.executeQuery();

        if(rs.next())
        {
            jTextSaldo.setText(rs.getString("saldo"));
        }
        else
        {
            JOptionPane.showMessageDialog(null, "Error");
        }
    }
    catch(Exception e)
    {
        System.err.print(e);
    }
}
```

Bien, para que todo esto funcione y la conexión sea exitosa necesitamos crear la conexión de Java con MySQL y para ello creamos una clase llamada “Conexión” que nos ayudara con dicha tarea y se muestra de la siguiente manera:

```
public class Conexion {  
  
    public Connection getConnection()  
    {  
        Connection con = null;  
        String base = "bdcajero";  
        String url = "jdbc:mysql://localhost:3306/"+base;  
        String user = "root";  
        String pass = "";  
  
        try  
        {  
            Class.forName("com.mysql.jdbc.Driver");  
            con = (Connection) DriverManager.getConnection(url, user, pass);  
        }  
        catch (Exception e)  
        {  
            System.err.print(e);  
        }  
  
        return con;  
    }  
}
```


Conclusión

Para terminar, podemos decir que el saber estructurar y darle su lugar a cada línea de código es importante ya que nos ayuda a identificar de manera sencilla que es lo que hace y como es que este funciona, que parte del código funciona primero y cual le sucede después, para ello, en ocasiones es importante poner comentarios para separar ya que sin estos podemos confundirnos y hasta perdernos en el código.

Saber programar estos sistemas, nos ayudan a despertar nuestra lógica y ver cómo es que podríamos darle solución a los distintos problemas que los sistemas presentan a lo largo del tiempo, siempre y cuando nos haga ahorrar recursos y tiempo para no hacerlo tedioso y costoso.

Analizar y comprender un lenguaje como tal nos ayuda a verificar y conocer como es que podemos dar solución al sistema y así poder corregir de manera rápida la parte del sistema que nos este causando problemas en el momento.