

Actividad 3 Programa Banco Mexicano (parte 2)

Lenguajes de Programación IV

Ingeniería en Desarrollo de Software

Tutor: Aarón Iván Salazar Macías

Alumno: Blanca Patricia Rosas Torres

Fecha: julio de 2023

Contenido

| | |
|---------------------|----|
| Introducción | 3 |
| Descripción | 3 |
| Justificación | 4 |
| Desarrollo..... | 5 |
| Interfaz | 8 |
| Codificación..... | 10 |
| Conclusión | 12 |
| Referencias..... | 13 |

Introducción

En el presente documento se ve el trabajo realizado durante esta materia, un ejemplo de programación en lenguaje java usando el ide NetBeans, se crea una base de datos en MySQL para guardar la información, la actividad pasada se centra en el diseño de la interfaz, es decir como se vera el proyecto, tamaños de fuentes, colores, posiciones, se nombraron los botones que llevaran las acciones a realizar, estos serian los eventos del proyecto, los que se van a programar en esta actividad, para que mediante la interfaz el usuario decida la acción a realizar y esto lleve al evento seleccionado, se usa un conector para poder tener acceso a la base creada en MySQL desde NetBeans, esto para que lo que se haga en los formularios se guarde en la base de datos creada, cada evento generado tendrá un enlace a ella mediante una clase creada llamada conexión que también se creara para este proyecto.

Descripción

Se necesita desarrollar un programa para que los clientes del banco mexicano puedan consultar su saldo, realizar retiros y depósitos, todo esto debe guardarse en una base de datos con la finalidad de que cada cliente pueda realizar sus movimientos usando su id, al crear estos eventos las entradas de los clientes deberán de ser por teclado, es decir sus repuestas o acciones a realizar deberán corresponder a un criterio dado por ejemplo si quiere retirar deberá seleccionar el numero que corresponda a retirar, se vera como una pregunta de opción múltiple donde la respuesta es lo que se va a realizar, todo debe estar basado en la programación orientada a objetos como ya se realizo en las actividades anteriores en lenguaje java con ide NetBeans declarando sentencias, como Switch, if, else, etc., la que pueda ayudar a llevar el proyecto con éxito, partiendo de el diseño de los JFrame de la actividad número dos.

Justificación

Java trabaja con un sistema de delegación de responsabilidades, esto viene dado por que la forma en la que trabajan los eventos, un objeto fuente crea y dispara un evento y un objeto interesado en el mismo lo recibe y maneja el evento. Entonces este último objeto es a quien se le delega la responsabilidad de hacer algo con el evento y debe tener dos componentes, el primero es escuchar o estar pendiente del evento y el segundo es hacer algo cuando reciba la señal que espera, por ello es que habíamos indicado los términos Listener y Handling. (solvetic, 2023)

Un evento en Java es un objeto que se crea cuando algo cambia dentro de una interfaz gráfica de usuario. Si un usuario hace clic en un botón, hace clic en un cuadro combinado o escribe caracteres en un campo de texto, etc., se activa un evento y se crea el objeto de evento relevante.

El manejo de eventos en Java se compone de dos elementos clave:

- El origen del evento , que es un objeto que se crea cuando ocurre un evento. Java proporciona varios tipos de estas fuentes de eventos,
- El detector de eventos , el objeto que "escucha" los eventos y los procesa cuando ocurren.

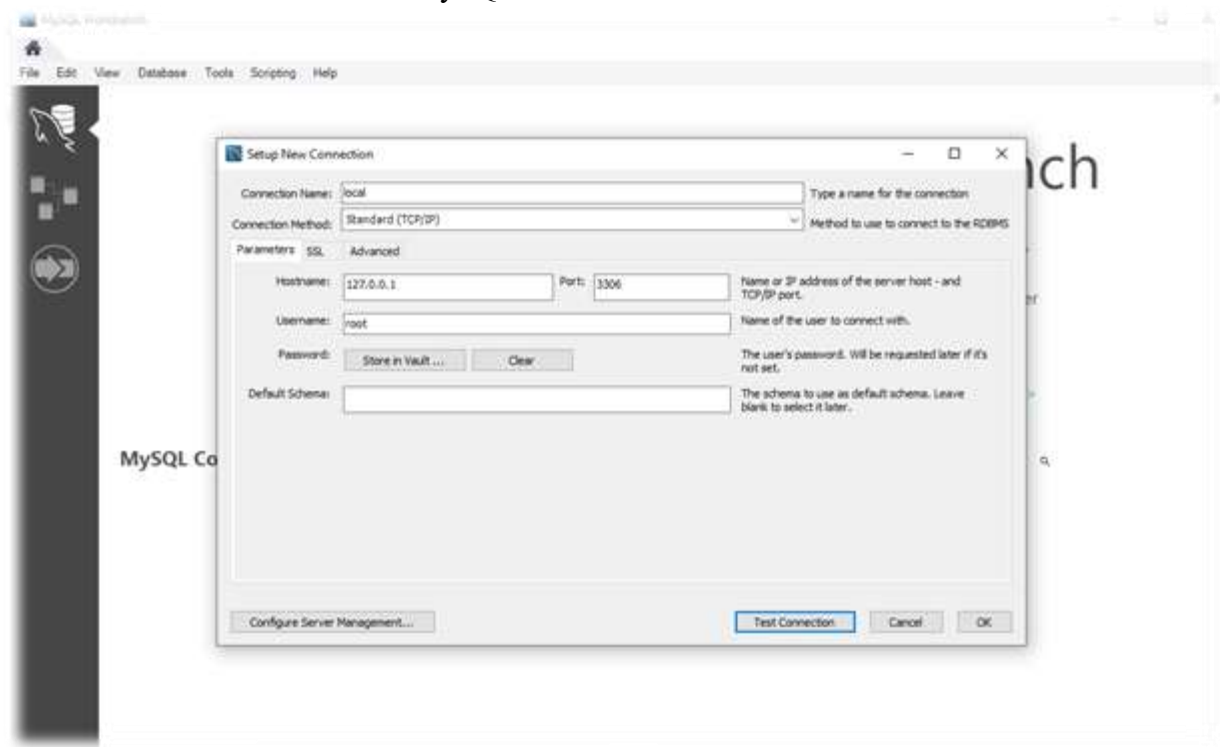
Hay varios tipos de eventos y oyentes en Java: cada tipo de evento está vinculado a un oyente correspondiente.

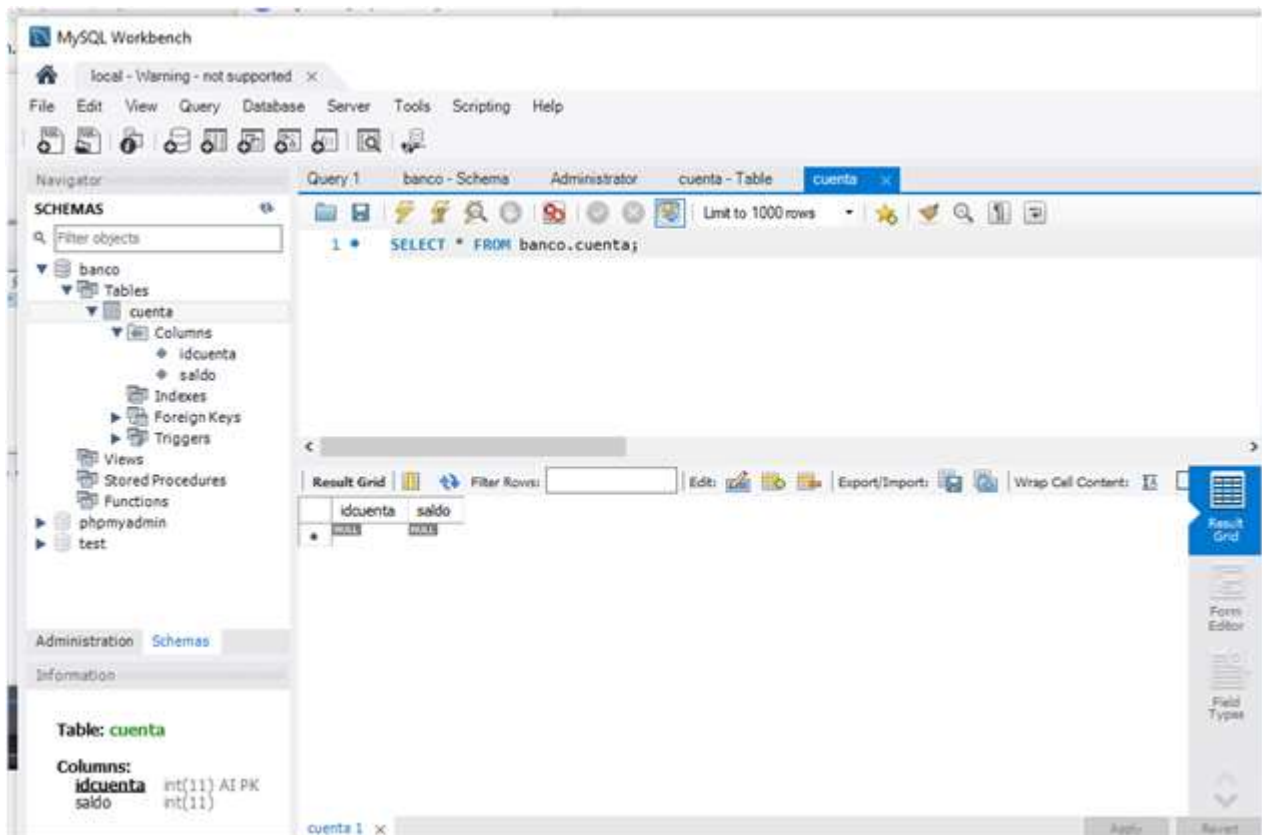
- **ActionEvent** : Representa un elemento gráfico en el que se hace clic, como un botón o un elemento de una lista. Oyente relacionado: **ActionListener**.

- ContainerEvent : representa un evento que ocurre en el propio contenedor de la GUI, por ejemplo, si un usuario agrega o elimina un objeto de la interfaz. Oyente relacionado: ContainerListener.
- KeyEvent : Representa un evento en el que el usuario presiona, teclea o suelta una tecla. Oyente relacionado: KeyListener.
- WindowEvent : Representa un evento relacionado con una ventana, por ejemplo, cuando se cierra, activa o desactiva una ventana. Oyente relacionado: WindowListener.
- MouseEvent : representa cualquier evento relacionado con un mouse, como cuando se hace clic o se presiona un mouse. Oyente relacionado: MouseListener. (Greelane, 2019)

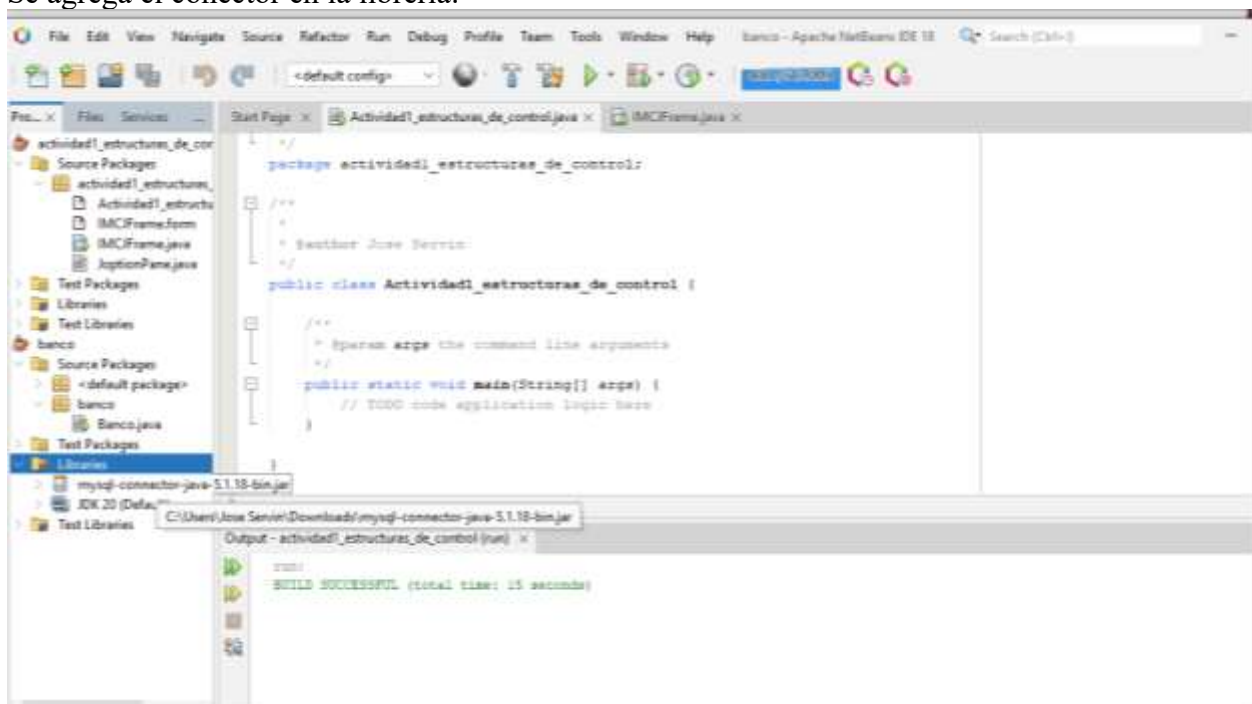
Desarrollo

Creación de la base de datos en MySQL:

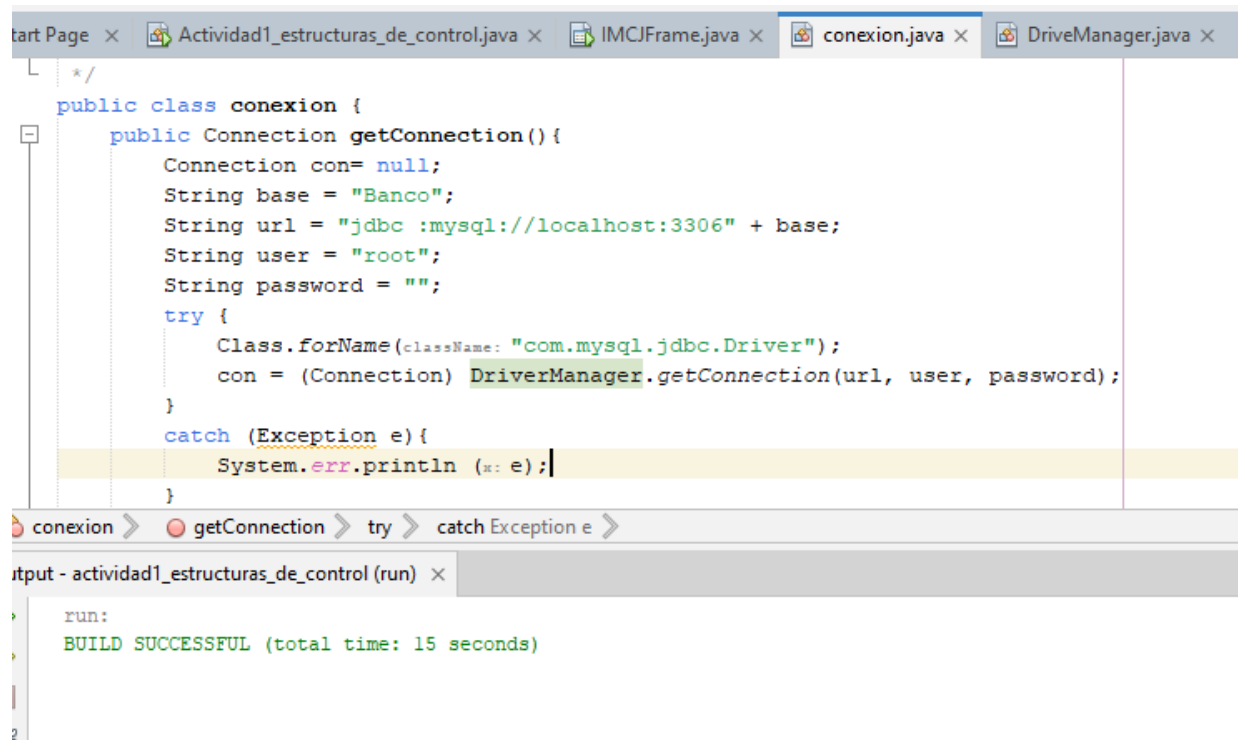
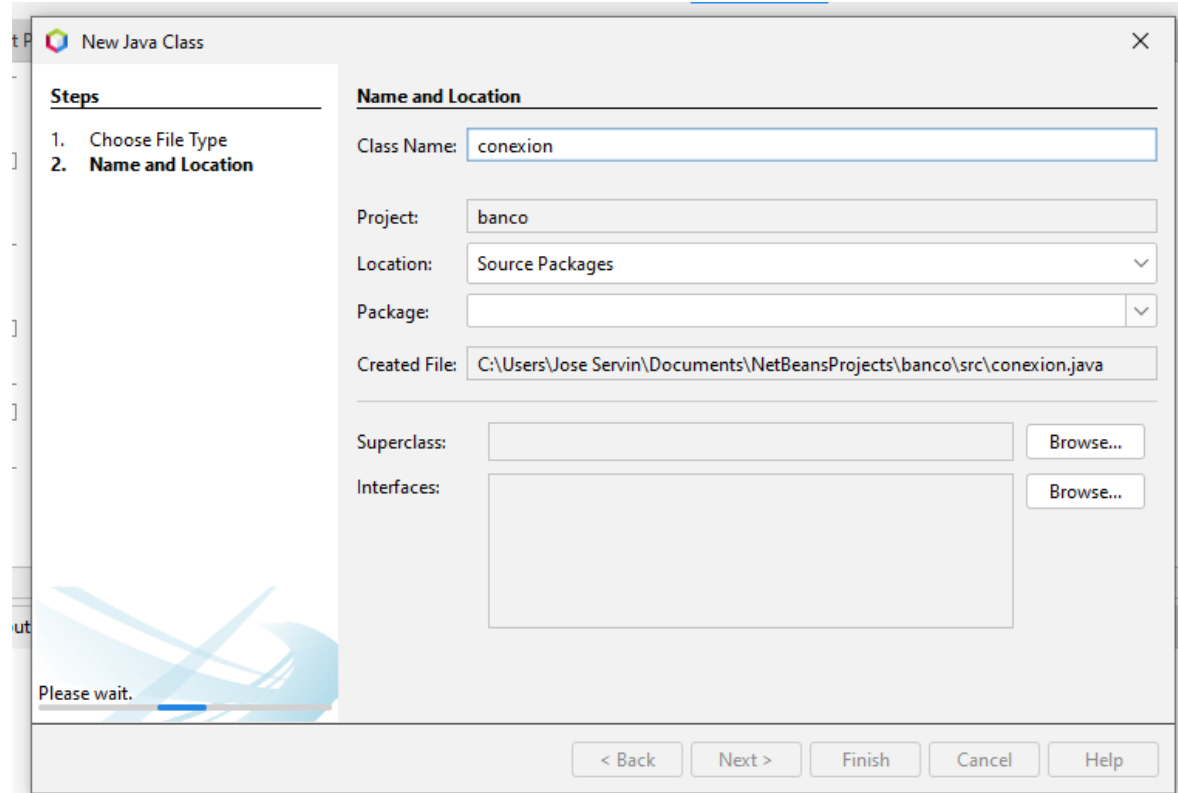




Se agrega el conector en la librería:



Código de la clase conexión:

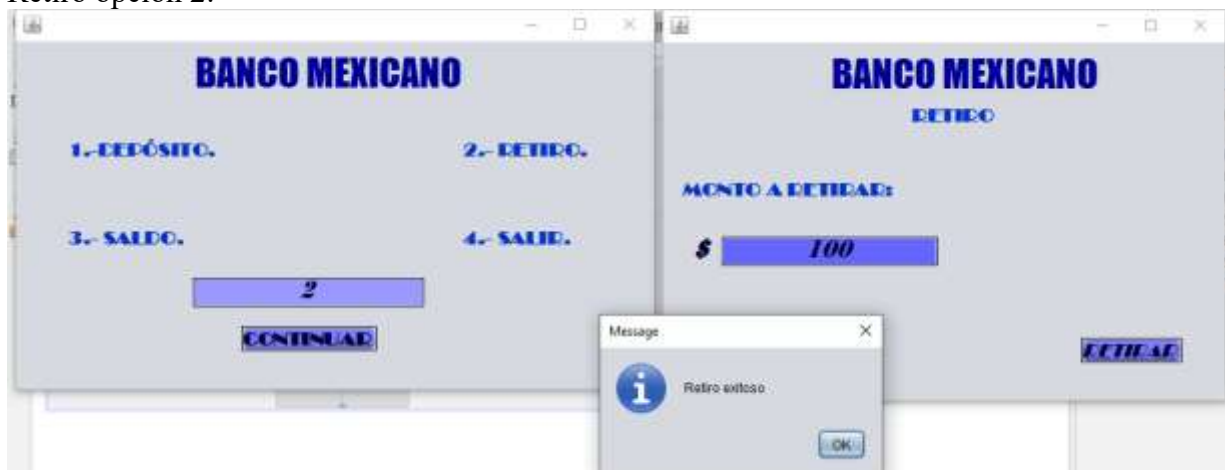


Interfaz

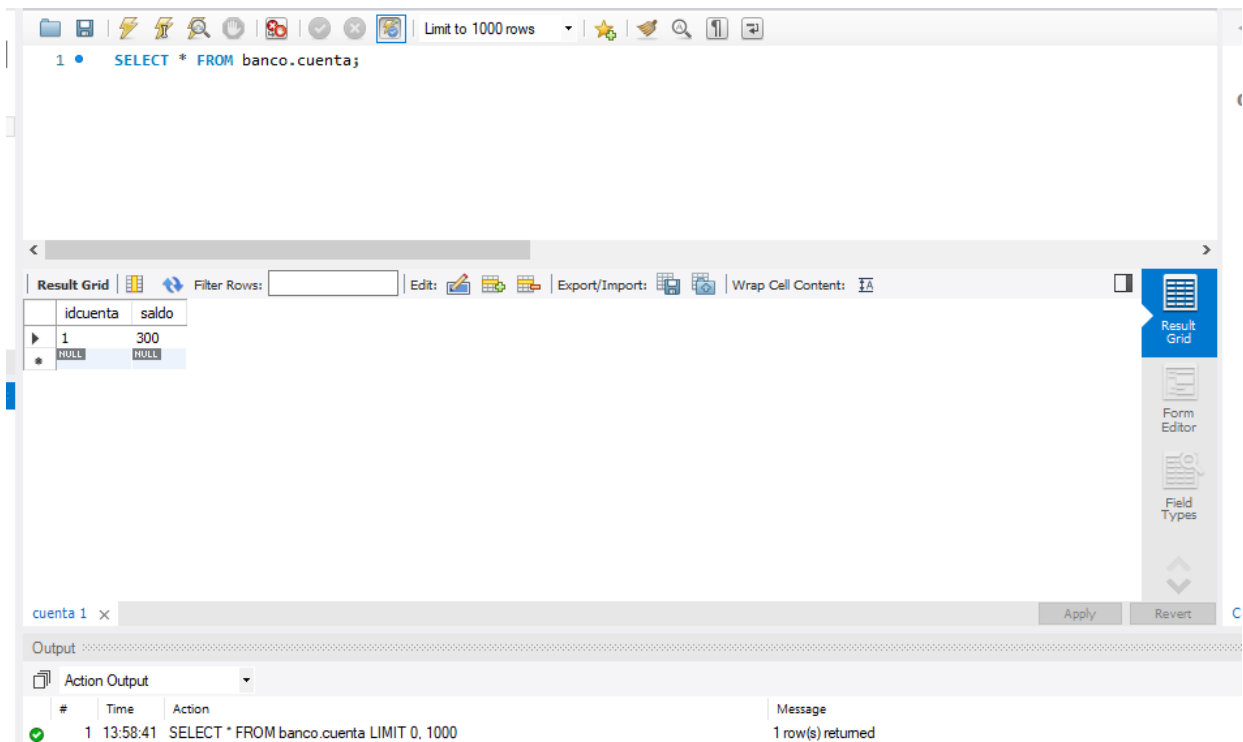
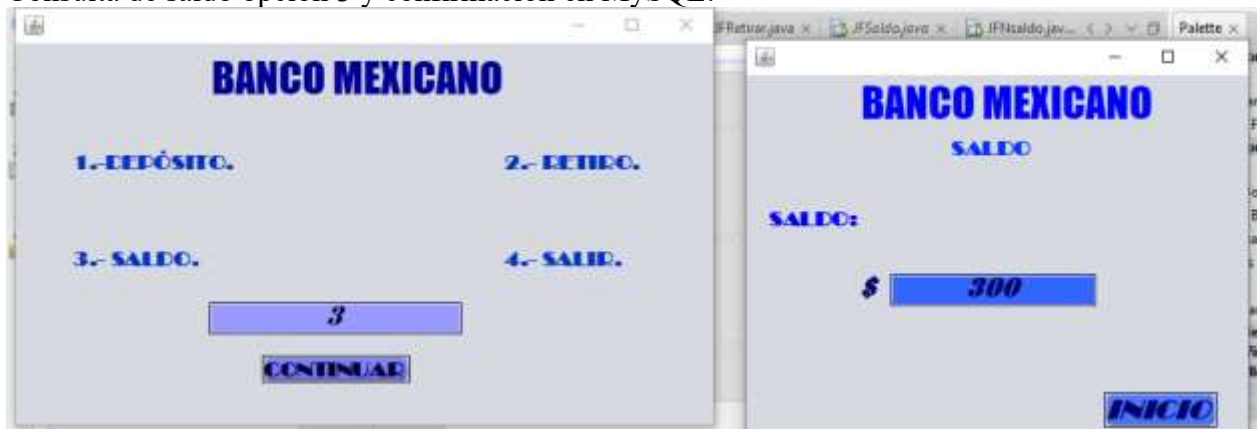
Deposito opción 1:



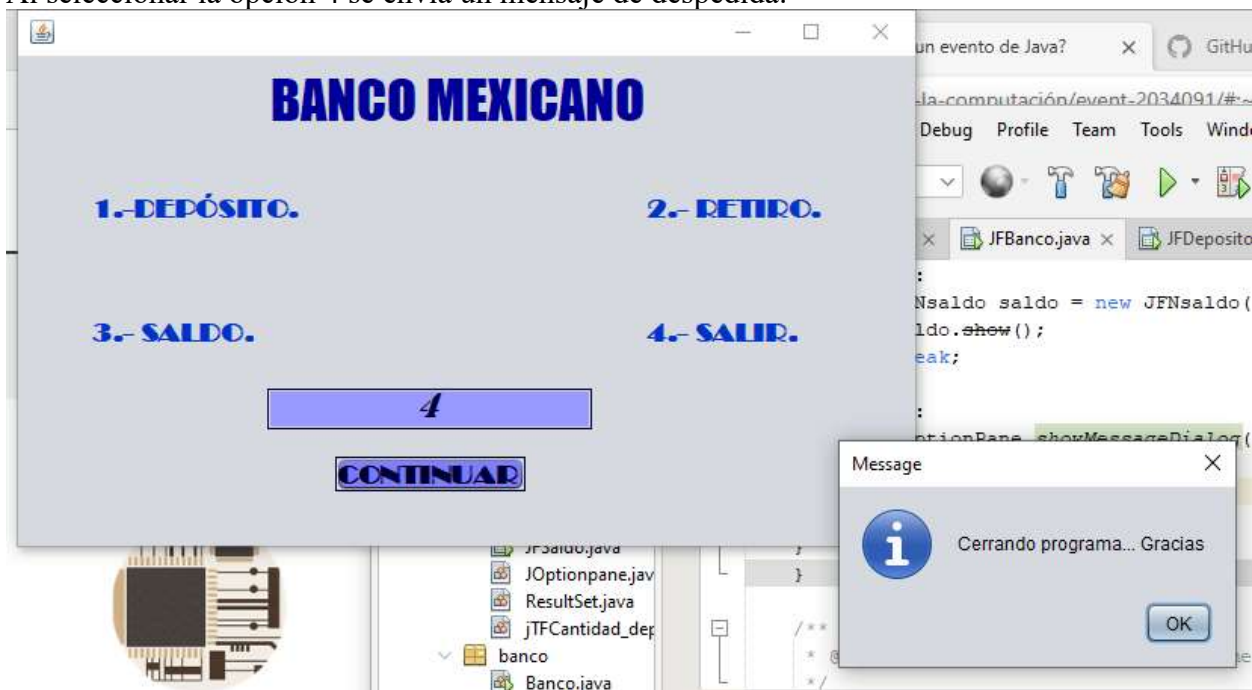
Retiro opción 2:



Consulta de saldo opción 3 y confirmación en MySQL:

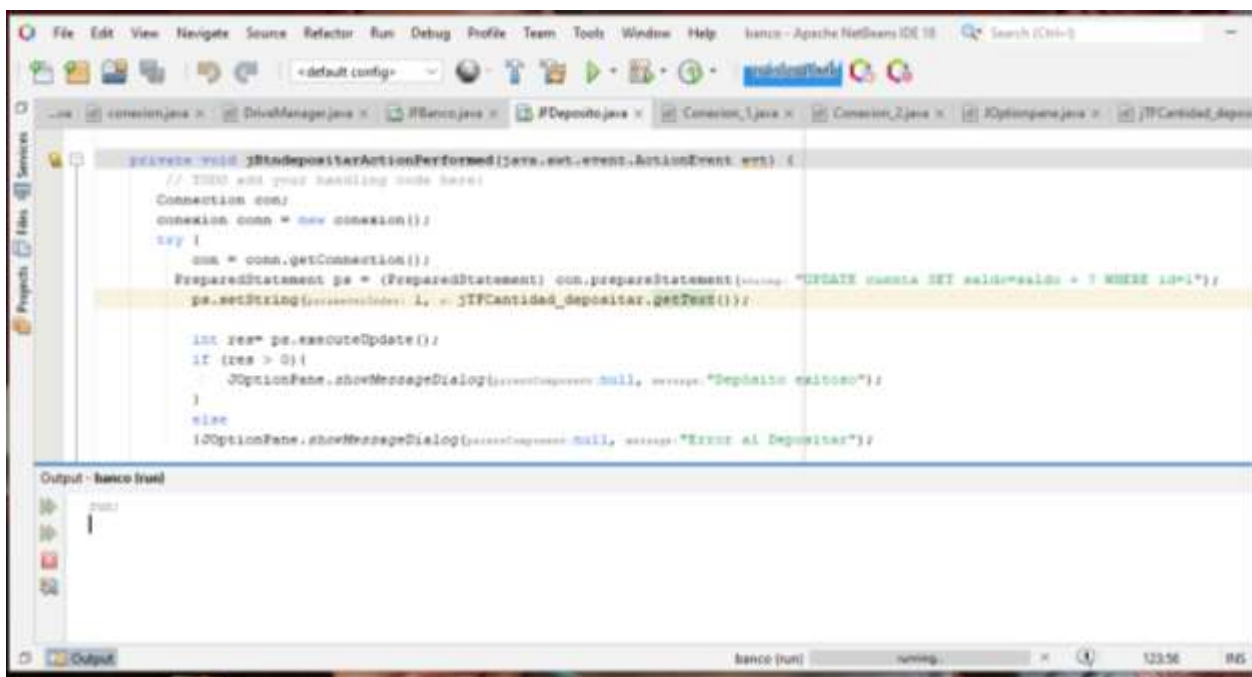


Al seleccionar la opción 4 se envía un mensaje de despedida:



Codificación

Código de la opción depositar, este es igual al de retirar solo cambian detalles como el signo de + y los mensajes:



Código creado para la opción 3 consulta de saldo, esta se genera en el constructor no como acción directa del botón:



```

/*
 * Created new form JFMsaldo
 */
public JFMsaldo() {
    initComponents();

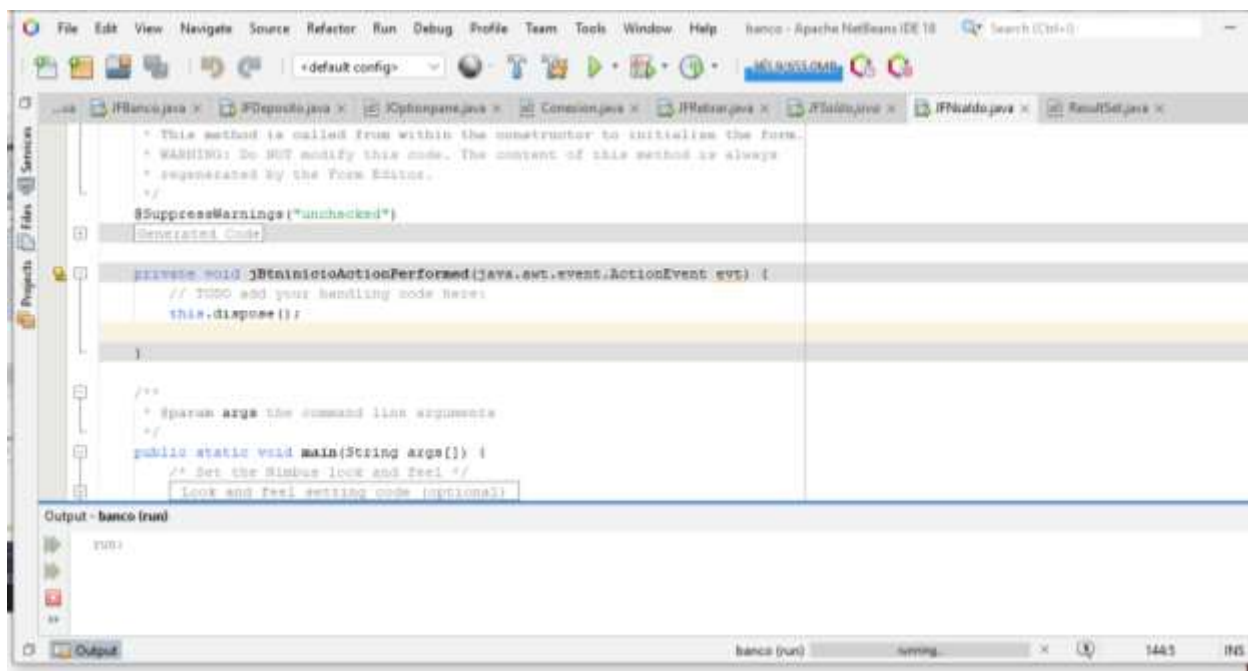
    Connection con;
    Conexion conx = new Conexion();
    try {
        con = conx.getConnection();
        PreparedStatement ps = (PreparedStatement) con.prepareStatement("SELECT saldo FROM cuenta WHERE idcuenta = 1");

        java.sql.ResultSet rs = ps.executeQuery();

        if (rs.next()) {
            jTextSaldo.setText(rs.getString("saldo"));
        } else {
            JOptionPane.showMessageDialog(this, "Error");
        }
    } catch (Exception e) {
        System.out.println(e);
    }
}

```

En el botón de la opción 3 se programa para cerrarse y así regrese al inicio:



```

/*
 * 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 jBTInicioActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.dispose();
}

/**
 * Param args the command line arguments
 */
public static void main(String args[]) {
    // Set the Nimbus look and feel
    // Look and feel setting code (optional)
}

```

Output - banco (run)

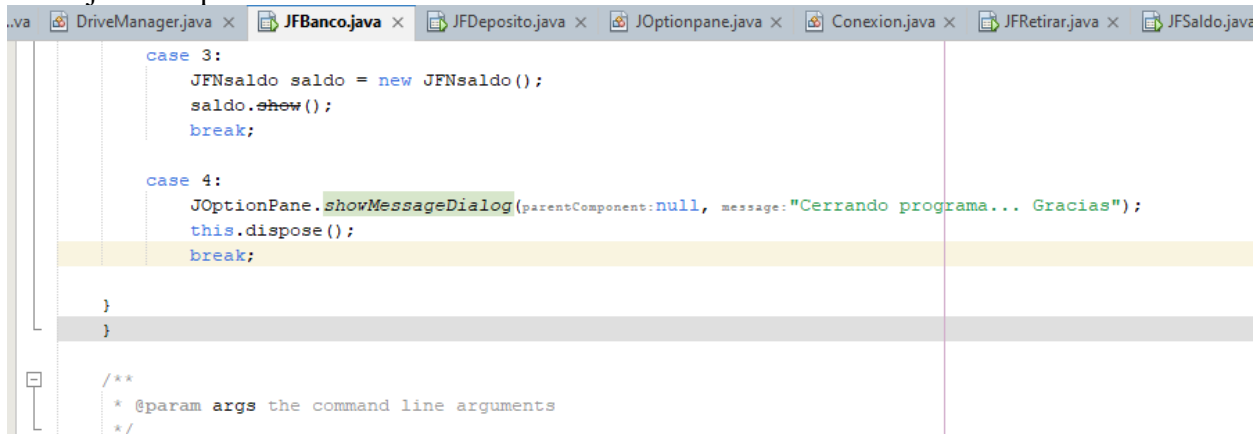
```

run:

```

El código de la opción 4 para salir del programa, se agregó un JoptionPane para enviar el

mensaje de despedida:



```
case 3:
    JFNSaldo saldo = new JFNSaldo();
    saldo.show();
    break;

case 4:
    JOptionPane.showMessageDialog(parentComponent:null, message:"Cerrando programa... Gracias");
    this.dispose();
    break;

}

}

/**
 * @param args the command line arguments
 */
```

Conclusión

Para la elaboración del código se deben ir agregando las librerías necesarias por ejemplo en el ultimo caso de switch se hace un dispose, pero se agrega el JoptionPane que es el que nos manda la ventanita del mensaje y para poder utilizarla sin que nos marque error se debe importar, esto sucede en la mayoría de los eventos programados, debido a que la interfaz de NetBeans es muy amigable al momento de copiar o escribir alguna línea que necesita alguna importación lo marca, cabe mencionar que esta actividad se ve sencilla pero en el desarrollo no salió todo de acuerdo a la tutoría pasaron varios incidentes que llevaron a la creación de un nuevo JFrame de saldo ya que el que se tenia se le borraron algunas líneas de código, errores de los que se aprende, el uso de mayúsculas y minúsculas así como optar por mejor copiar y pegar los objetos ya que en ocasiones una sola letra causa un error.

GitHub: github.com/97677256/lenguajes-de-programaci-n-4

Referencias

Greelane. (03 de 07 de 2019). Obtenido de <https://www.greelane.com/es/ciencia-tecnolog%c3%ada-matem%c3%a1ticas/ciencias-de-la-computaci%c3%b3n/event-2034091/#:~:text=Estos%20son%20algunos%20de%20los%20tipos%20de%20eventos,teclea%20o%20suelta%20una%20tecla.%20...%20M%C3%A1s%20elementos>

solvetic. (2023). Obtenido de <https://www.solvetic.com/tutoriales/article/1231-manejo-de-eventos-en-java/>