	UNIVERSIDAD DON BOSCO **FACULTAD DE ESTUDIOS TECNOLÓGICOS**
CICLO I	**GUIA DE LABORATORIO #6** **Programación Orientada a Objetos** **Interfaces gráficas y JDBC con Netbeans**

I.OBJETIVOS

- Que el estudiante aplique sus conocimientos de base de datos.
- Que el estudiante pueda conectar un a aplicación gráfica con una BD.

II. INTRODUCCIÓN

En esta guía continuaremos el uso de Swing y las conexiones a bases de datos. Como antes se había mencionado, Netbeans posee excelentes asistentes para el diseño de aplicaciones gráficas, combinado con JDBC se pueden desarrollar aplicaciones potentes para escritorio.

III. PROCEDIMIENTO

1. Crear un nuevo proyecto con el nombre de **"Guia6poo1"**
2. Para poder realizar una conexión crearemos una clase que tenga los métodos necesarios y que se llame **"Conexion.java"**, agregar el siguiente **código:**

```
package guia6poo1;

importjava.sql.*;

publicclassConexion {
private Connection conexion =null;
private Statement s =null;
privateResultSetrs=null;
private String query="";

//Contructor
publicConexion() throwsSQLException{
try
      {
//obtenemos el driver de para mysql
Class.forName("com.mysql.jdbc.Driver");
// Se obtiene una conexión con la base de datos.
conexion = DriverManager.getConnection (
"jdbc:mysql://localhost/Guia6poo1","root", "");
// Permite ejecutar sentencias SQL sin parámetros
s = conexion.createStatement();
          }
```

```
    catch (ClassNotFoundException e1) {
    //Error si no puedo leer el driver de MySQL
                        System.out.println("ERROR:No encuentro el driver de la BD:
    "+e1.getMessage());
            }
        }
    //Metodo que permite obtener los valores del resulset
    publicResultSetgetRs() {
    returnrs;
    }
    //Metodo que permite fijar la tabla resultado de la pregunta
    //SQL realizada
    publicvoidsetRs(String consulta)  {
    try {
    this.rs = s.executeQuery(consulta);
    } catch (SQLException e2) {
            System.out.println("ERROR:Fallo en SQL: "+e2.getMessage());
            }
        }

    //Metodo que recibe un sql como parametro que sea un update,insert.delete
    publicvoidsetQuery(String query) throwsSQLException {
    this.s.executeUpdate(query);
        }

    //Metodo que cierra la conexion
    publicvoidcerrarConexion() throwsSQLException{
    conexion.close();
        }
    }
```

Nota: No olvide agregar el driver MySQL JDBC Driver al proyecto (recomendado) o a la dirección respectiva.

3. Crear una base de datos llamada **“Guia6poo1”**.
4. Ejecutar el archivo SQL proveído por su instructor. No olvide utilizar respectivamente esa base de datos. Si tiene problemas consulte a su instructor.
5. Crear el paquete “**sv.edu.udb.util**” y en el crear la clase **CheckPassword**, esta deberá contener el siguiente código:

```java
package sv.edu.udb.util;
/**
 *
 * @author Rafael Torres
 */
public class CheckPassword {
    public boolean verificarPassword(char passArray[]){
        for (int i = 0; i < passArray.length; i++) {
                        char c = passArray[i];
                        // Si no es letra o numero entonces no es valido
                        if (!Character.isLetterOrDigit(c)){
                            return false;
                        }
            }
        return true;
    }
}
```

6. Nuestro propósito para esta guíaserá crear un formulario de ingreso que permita validar un usuario que sea de tipo **"admin"** si esto es verdadero pasaremos a un frame de tipo Mdi que permitirá abrir un mantenimiento básico, para ello crear un JframeForm dentro del paquete "**guia6poo1**", que se vea de la siguiente manera.

No olvide importar el paquete java.sql.* dentro del código del Formulario

```java
import java.sql.*;

public class Logueo ex
```

7. Cambiar las propiedades de cada componte según como se muestra en la siguiente tabla.

Control	Propiedad	Evento	Código	Valor
jTextField1	name		Variable name	txtNombre
jPasswordField1	name		Variable name	txtPassword
jButton1	text	actionPerformed		Limpiar
jButton2	text	actionPerformed		Ingresar

8. Modificar el constructor de este Jframe para que quede de la siguiente forma, esto para que la ventana aparezca posicionada en el centro.

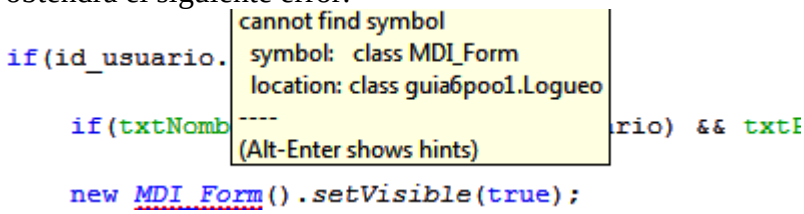
```
public Logueo() {
    initComponents();
    setLocationRelativeTo(null);
}
```

9. Ahora utilizamos el evento **actionPerformed** del botón ingresar, dentro de él digitar el siguiente código:

```
private void jButton2ActionPerformed(java.awt.event.ActionEventevt) {
        String id_tipo="";
        String usuario="";
        String password="";
      try {
CheckPasswordverificar= new CheckPassword();
        char passArray[] = txtPassword.getPassword();
        if(verificar.verificarPassword(passArray)){
           String pass=new String(passArray);
Conexion  con = new Conexion();//creamos el objeto para la conexion
con.setRs("selectid_tipo_usuario,usuario,passwordfrom usuarios where usuario=" + "'" + txtNombre.getText() + "'");//consulta
ResultSet valor=(ResultSet) con.getRs();//obtenemos los valores

valor.next();//nos movemos al unico registro devuelto
id_tipo=valor.getString(1);//obtenemos el id del tipo de usuario
            usuario=valor.getString(2);//obtenemos el usuario
password=valor.getString(3);//obtenemos el password del usuario

System.out.println("Password: "+ pass +" " +password  );
System.out.println("Usuario: "+ txtNombre.getText() +" " +usuario  );
          //verificamos si el usuario y el password de la base son iguales a los ingresados en los txt
if(txtNombre.getText().equals(usuario) &&pass.equals(password)){
//si este es un usuario de tipo administrador
System.out.println("Id: "+ id_tipo  );
               if (id_tipo.equals("0")){
                  new MDI_Form().setVisible(true);
this.dispose();
               }
               else{
JOptionPane.showMessageDialog(this, "Ustedes un empleado");
               }
            }
            else{
JOptionPane.showMessageDialog(this, "Usuario ó Password incorrecto");
            }



        }else
        {
JOptionPane.showMessageDialog(this, "El passworContieneCaracteresInvalidos");
```

```
            }
        } catch (SQLException ex) {

Logger.getLogger(Logueo.class.getName()).log(Level.SEVERE, null, ex);
}

    }
```

Tome en cuenta que obtendrá el siguiente error:



Esto es debido a que el formulario que se llamará MDI_Form será creado en el paso 11 de esta guía.

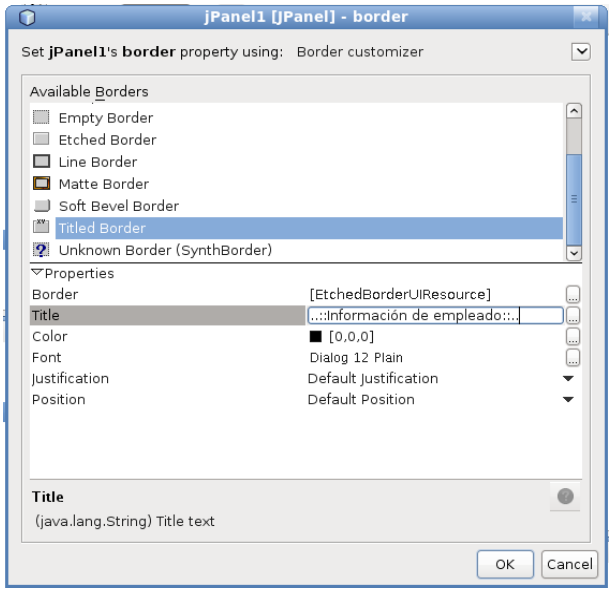
IMPORTANTE: PARA ESTA GUÍA EL USUARIO PRIVILEGIADO DE SU SISTEMA SERÁ root Y SU PASSWORD SERÁ root

10. Para darle funcionalidad al botón Limpiar seleccionar el evento **actionPerformed** y agregar el siguiente código:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    txtNombre.setText("");
    txtPassword.setText("");
}
```

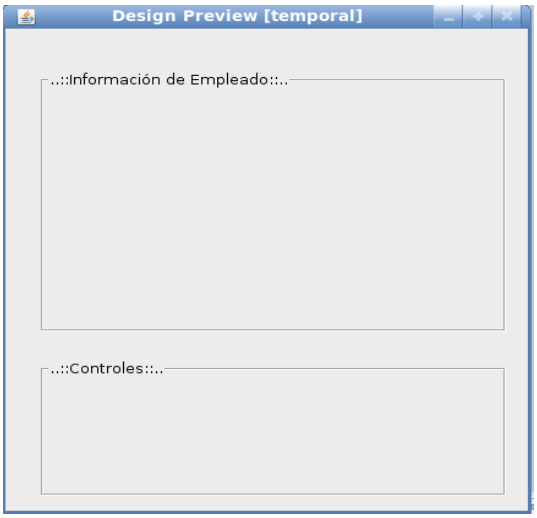
11. Finalizado la creación del formulario de ingreso de usuario, pasaremos a crear un formulario Mdi dentro del paquete "**guia6poo1**" llamado **"MDI_Form"** y agregar un Jframe interno (JInternalFrameForm) el cual se llamara **"MantenimientoEmpleados"** dentro del mismo paquete.

12. Agregar 2 Panel al FrameMantenimientoEmpleados, ahora seleccionar uno de los controlespanel ir a las propiedades y seleccionar **"border"**, aparecerá una pantalla como la



siguiente

Ya en la ventana seleccionar **"TitledBorder"** y modificar la propiedad title como se ve en la imagen, modificar el segundo **Panel** para que al final se vea de la siguiente manera.



13. Modificar el Frame interno para que se vea de la siguiente manera.

14. Cambiar las propiedades de cada componte según como se muestra en la siguiente tabla.

Control	Propiedad	Eventos	Código	Valor
JInternalFrame	title			Ingreso Persona
jLabel1	text			ID:

			Variable Name	lblId
jLabel2	text			Ingrese los nombres:
			Variable Name	lblNombres
jLabel3	text			Ingrese los apellidos:
			Variable Name	lblApellidos
jLabel4	text			Ingrese la edad:
			Variable Name	lblEdad
jLabel5	text			Nombre de usuario
			Variable Name	lblNombreUsuario
jLabel6	text			Password:
			Variable Name	lblPasswordUsuario
jLabel7	text			Tipo de Usuario:
			Variable Name	lblTipoUsuario
jTextField1	name			txtID
jTextField2	name			txtNombres
jTextField3	name			txtApellidos
jTextField4	name			txtEdad
jTextField5	name			txtNombreUsuario
jPasswordField1	name			txtPassword
jComboBox1	name			cmbTipoUsuario
jButton1	name	actionPerformed		btnIngresar
	text			Ingresar
jButton2	name	actionPerformed		btnAnterior
	text			Anterior
jButton3	name	actionPerformed		btnSiguiente
	text			Siguiente
jButton4	name	actionPerformed		btnLimpiar
	text			Limpiar

15. Importar antes de la clase el paquete java.sql.* luego, dentro de la clase, agregar las siguientes propiedades e inicializar el constructor tal y como se muestra en la siguiente imagen.

```
/** Creates new form MantenimientoEmpleados */
ResultSet empleados;
ResultSet llenarcombo;
static  int bandera=0;

public MantenimientoEmpleados() throws SQLException {
    initComponents();
     iniciarValores();//Permite inicializar valores
 }
```

Nota:iniciarValores() es un método que va a crear en el siguiente paso, no tome en cuenta el error de "*método no definido"*.

16. Como se puede observar en el constructor aparte de que inicializar los componentes, también llama a un método creado por nosotros.Usted debe crearlo y codificarlo tal y como se ve en la imagen:

```
    Conexion  con = new Conexion();//creación del objeto
public void iniciarValores() throws SQLException{

    con.setRs("select * from usuarios ");//consulta de todos los usuarios
    //atributo resulset que obtiene los valores de la clase Conexion
    //devueltos por el metodo getRS().
    empleados=(ResultSet) con.getRs();
    empleados.last();//Nos permite ir al ultimo registro
    empleados.beforeFirst();//permite volver al registro inicial
    empleados.next();//movemos al primer registro

    //Creacion del objeto con2 que permite llenar con valores
    //el Jcombobox y tambien para el ingreso de datos.
    Conexion con2=new Conexion();
    //obtenemos todos los campos de la tabla tipo_usuarios
    con2.setRs("select * from tipo_usuarios");
    cmbTipoUsuario.removeAllItems();          // Limpiamos el combobox
    //atributo resulset que obtiene los valores de la clase Conexion
    //devueltos por el metodo getRS().
    llenarcombo=(ResultSet) con2.getRs();
    while(llenarcombo.next()){//recorremos todos los campos
        //agregamos los item de los tipos de usuarios que se encuentran
        //en el campo 2
        cmbTipoUsuario.addItem(llenarcombo.getString(2));
    }
    con2.cerrarConexion();//cerramos la conexion de este objeto
    llenarTxtbox();         // Metodo para llenar campos de el fomulario
    btnAnterior.setEnabled(false);
    btnSiguiente.setEnabled(true);
}
```

Nota:llenarTxtbox() es un método que va a crear en el paso 18, no tome en cuenta el error de "*método no definido"*.

17. Ahora pasaremos a crear la funcionalidad de los botones Anterior, Siguiente y Limpiar, seleccionar el evento **actionPerformed**y los digitaremos tal y como se muestra en la imagen.

Para el botón Anterior

```
private void btnAnteriorActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        anterior();
    } catch (SQLException ex) {
        try {
            anterior();
            //JOptionPane.showMessageDialog(this, "No existen registros anteriores");
        } catch (SQLException ex1) {
            Logger.getLogger(MantenimientoEmpleados.class.getName()).log(Level.SEVERE, null, ex1);
        }
    }

}
```

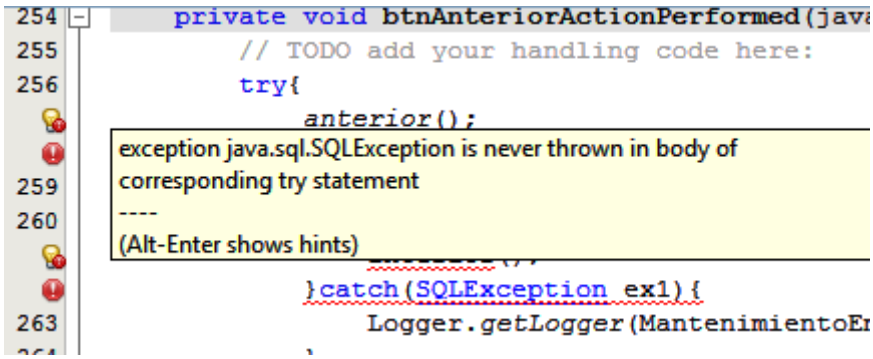
Para el botón Limpiar

```
private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    limpiarText();//metodo general para limpiar los txtbox
}
```

Para el botón Siguiente

```
private void btnSiguienteActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        siguiente();

    } catch (SQLException ex) {
        try {
            siguiente();
        } catch (SQLException ex1) {
            Logger.getLogger(MantenimientoEmpleados.class.getName()).log(Level.SEVERE, null, ex1);
        }
            //JOptionPane.showMessageDialog(this, "Fin de registro");

    }
}
```

Nota: todos los métodos mostrados se crearán en el siguiente paso, no tome en cuenta el error de "*método no definido*" y los llamados de atención de SQLException no lanzado.



18. Como se puede observar cada evento llama métodos importantes que generan la funcionalidad de los botones Anterior y Siguiente, los métodos son detallados a continuación (si no comprende su funcionamiento consultar con su instructor).

```
private void siguiente() throws SQLException{
    //Si el cursor no esta despues del ultimo registro se
    //podra mover al siguiente
    if(empleados.isAfterLast()==false){
        btnAnterior.setEnabled(true);
        empleados.next(); //mover al siguiente registro
        llenarTxtbox();
    }
    else{
        //Si se estamos despues del ultimo registro desactivaremos el boton
        //y regresamos al ultimo registro valido
        JOptionPane.showMessageDialog(this, "Ya no existen mas registros para recorrer");
        empleados.previous();
        btnSiguiente.setEnabled(false);
    }
}
```

```
private void anterior() throws SQLException{
    //Si el cursor no esta antes del ultimo registro se
    //podra mover al anterior
     if(empleados.isBeforeFirst()==false){
            btnSiguiente.setEnabled(true);
            empleados.previous();
            llenarTxtbox();
     }
     else{
         //Si estamos despues del antes del ultimo registro desctivaremos el boton
         //y regresamos al primer registro valido
            JOptionPane.showMessageDialog(this, "Ya no existen mas registros para recorrer");
            empleados.next();
            btnAnterior.setEnabled(false);
     }

}
```

```
    //Las cajas pasan a un valor de vacío
    private void limpiarText(){
    txtID.setText("");
    txtNombres.setText("");
    txtApellidos.setText("");
    txtEdad.setText("");
    txtNombreUsuario.setText("");
    cmbTipoUsuario.setSelectedIndex(0);
    txtPassword.setText("");
    }

    //Llena las cajas de texto con los valores de la consulta
    private void llenarTxtbox() throws SQLException{
        txtID.setText(String.valueOf(empleados.getInt(1)));
        txtNombres.setText(empleados.getString(2));
        txtApellidos.setText(empleados.getString(3));
        txtEdad.setText(String.valueOf(empleados.getInt(4)));
        cmbTipoUsuario.setSelectedIndex(empleados.getInt(5));
        txtNombreUsuario.setText(empleados.getString(6));
        txtPassword.setText(empleados.getString(7));

    }
```

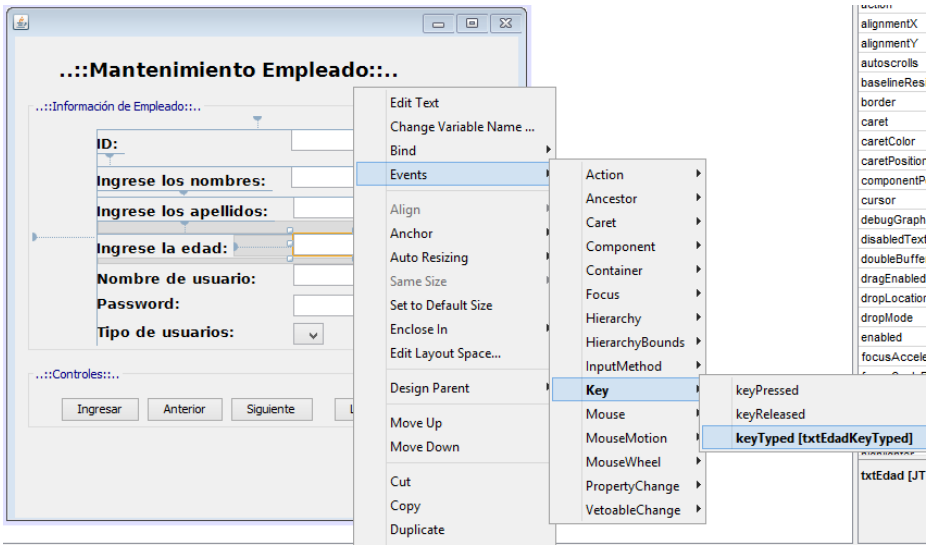
19. Ahora generaremos la funcionalidad del botón ingresar para agregar un nuevo registro a la base de datos, digite el código necesario para este paso, que utiliza el eventoactionPerformed.

```
private void btnIngresarActionPerformed(java.awt.event.ActionEventevt) {
      try {
         // TODO add your handling code here:
         if(btnIngresar.getText().equals("Ingresar")){
btnIngresar.setText("Guardar");
btnAnterior.setEnabled(false);
btnSiguiente.setEnabled(false);
limpiarText();

         }
         else{
Conexion con2=new Conexion();
CheckPasswordverificar= new CheckPassword();
```

```
                char passArray[] = txtPassword.getPassword();

                if(verificar.verificarPassword(passArray)){
                   String password = new String(txtPassword.getPassword());
      con2.setQuery("Insert into usuarios values("+txtID.getText() + ",\"" + txtNombres.getText() +
"\",\"" +
txtApellidos.getText() + "\"," +Integer.parseInt(txtEdad.getText()) +
                    "," + cmbTipoUsuario.getSelectedIndex() +
                    ",\"" +txtNombreUsuario.getText()  +  "\",\""+
                    password +
"\")");
                    con2.cerrarConexion();
btnIngresar.setText("Ingresar");
JOptionPane.showMessageDialog(this, "Usuario Ingresado Exitosamente");
limpiarText();
                  con2.cerrarConexion();
iniciarValores();
                }
else{
JOptionPane.showMessageDialog(this, "El passwor Contiene Caracteres Invalidos");
                }
           }
       } catch (SQLException ex) {
          Logger.getLogger(MantenimientoEmpleados.class.getName()).log(Level.SEVERE,      null,
ex);
       }

   }
```

20. Ya que podemos tener problemas con el campo de edad, agregaremos una validación para evitar que se escriban caracteres diferentes de dígitos, para ello realizar lo siguiente:

    - Ubicarse sobre el controltxtEdad y agregar el evento keyTyped.



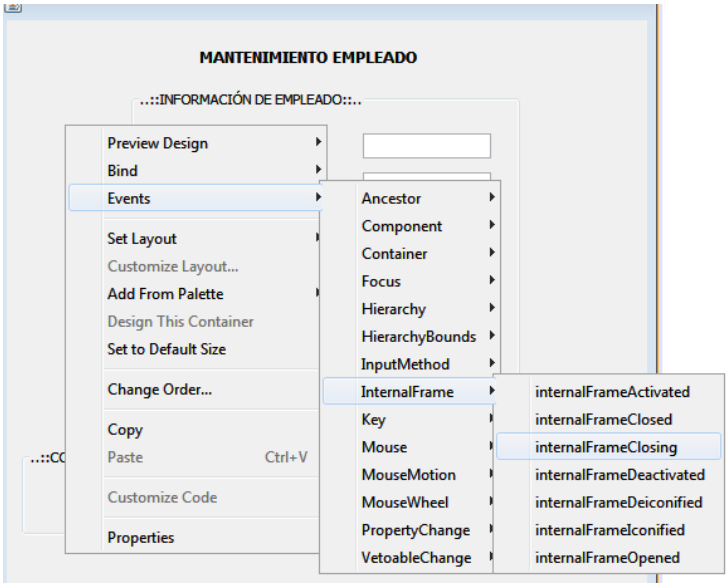
    - Modificar el evento tal y como se muestra a continuación

```
private void txtEdadKeyTyped(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    if(!Character.isDigit(evt.getKeyChar()))
    {

          Toolkit.getDefaultToolkit().beep();
            evt.consume();
    }
}
```

Se dará cuenta que ahora ya no deja escribir caracteres que no sean números.

21. Ya que abrimos una conexión al inicializar el formulario también deberemos de cerrar la conexión al cerrar el formulario, para ello debemos de apoyarnos de los pasos necesarios vistos en la guia5.

    - Seleccionar el Jframe interno llamado "**MantenimientoEmpleados**", ir a sus propiedades y modificar la propiedad DefaultCloseOperation →desplegar las opciones y seleccionar**DO_NOTHING**para eliminar la funcionalidad al dar click en la "X".Si ahora intenta correr la aplicación y da click en la "X" podrá observar que no pasa nada.
    - Modificar el **evento "InternalFrameClosing"**, agregaremos el método cerrar y lo modificaremos como se muestra a continuación.




```
private void cerrar(javax.swing.event.InternalFrameEvent evt) {
    try {
        // TODO add your handling code here:
        bandera = 0; //valor restaurado para el nuevo frame
        this.dispose();
        con.cerrarConexion();
    } catch (SQLException ex) {
        Logger.getLogger(MantenimientoEmpleados.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Note que la variable **bandera** es utilizada para mantener dos estados o 0 ó 1 que servirán para evitar que se abran varios hijos del mismo tipo en frameMdi

22. Ahora agregaremos la funcionalidad al Mdi padre este debe se debe de ver de la siguiente manera.

23. Modificar el método constructor para que quede de la siguiente manera

```
public MDI_Form() {
    initComponents();
    setExtendedState(MDI_Form.MAXIMIZED_BOTH);
}
```

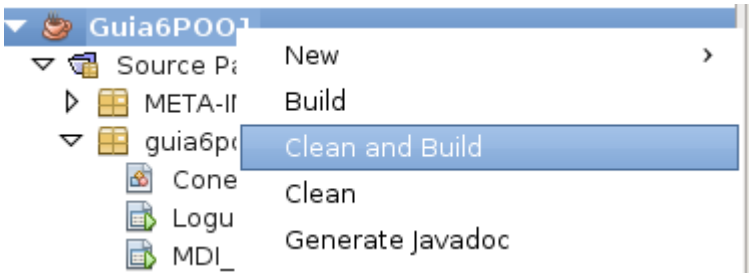
Maximiza la ventana

24. Ahora solo nos queda agregar al hijo al **desktopPane**del padre, como se mencionó en el punto 21 la propiedad **bandera** es utilizada para evitar que se sigan agregando más hijos del mismo tipo al padre.

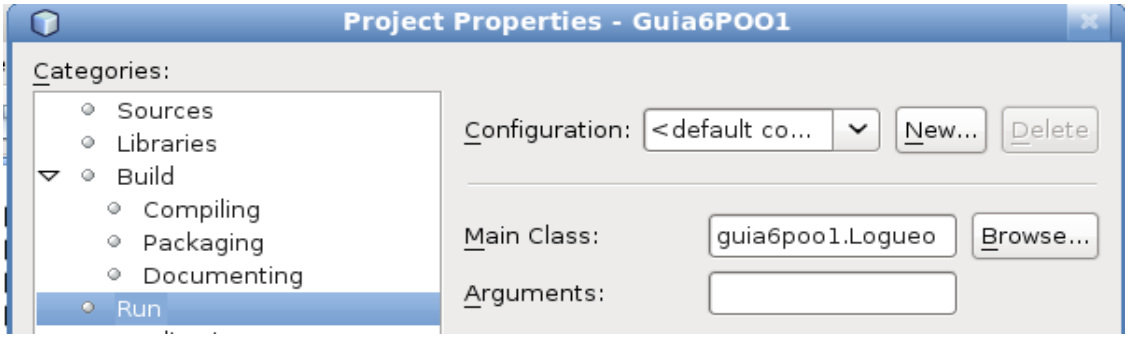
Nota: No olvide importar java.sql.* para **MDI_Form**

```
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        // TODO add your handling code here:

        if(MantenimientoEmpleados.bandera==0){
           MantenimientoEmpleados empleados = new MantenimientoEmpleados();
           desktopPane.add(empleados);
           empleados.show();
           MantenimientoEmpleados.bandera=1;
        }

    } catch (SQLException ex) {
        Logger.getLogger(MDI_Form.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

25. Por último, ya teniendo aplicación final resulta complicado tener que abrir NetBeans para correr la aplicación. Por eso, crearemos un archivo ejecutable.Las aplicaciones en Java poseen la extensión**.jar**, nos apoyaremos de la bondades del IDE NetBeans y solo daremos clic derecho sobre el proyecto y dar clic sobre **"Clean and Build"**.

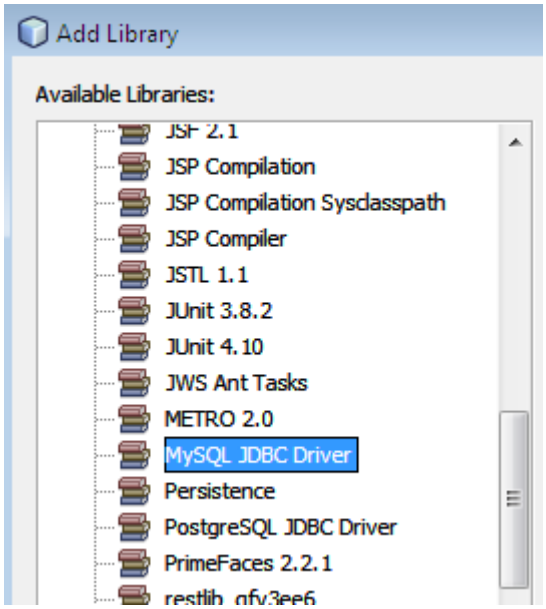


El proyecto comenzará a correr desde aquella clase que se haya establecido como **main**,si usted quiere modificar el **main** original deberá irse a las propiedades del proyecto, aparecerá una ventana como la siguiente. Seleccionar **Run** y establecer el main que queremos por defecto.



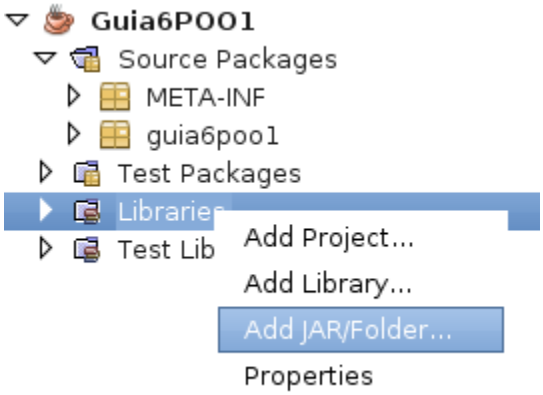
Nota1: No olvidar que por cada cambio deberá genera el nuevo **jar.**
Nota2: Si los métodos que utilicen la conexión dan errores seguramente es porque no se agregado el conector de mysql.

Recuerde las formas de agregar el conector a partir de las librerías:

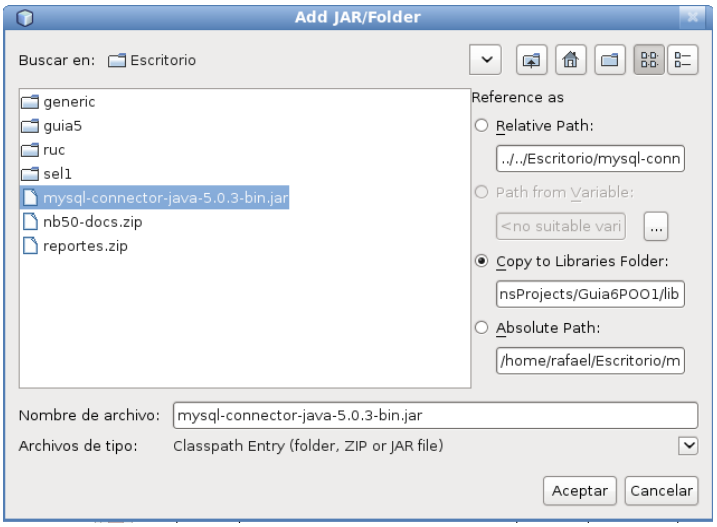


También puede agregar una librería de la que usted disponga:

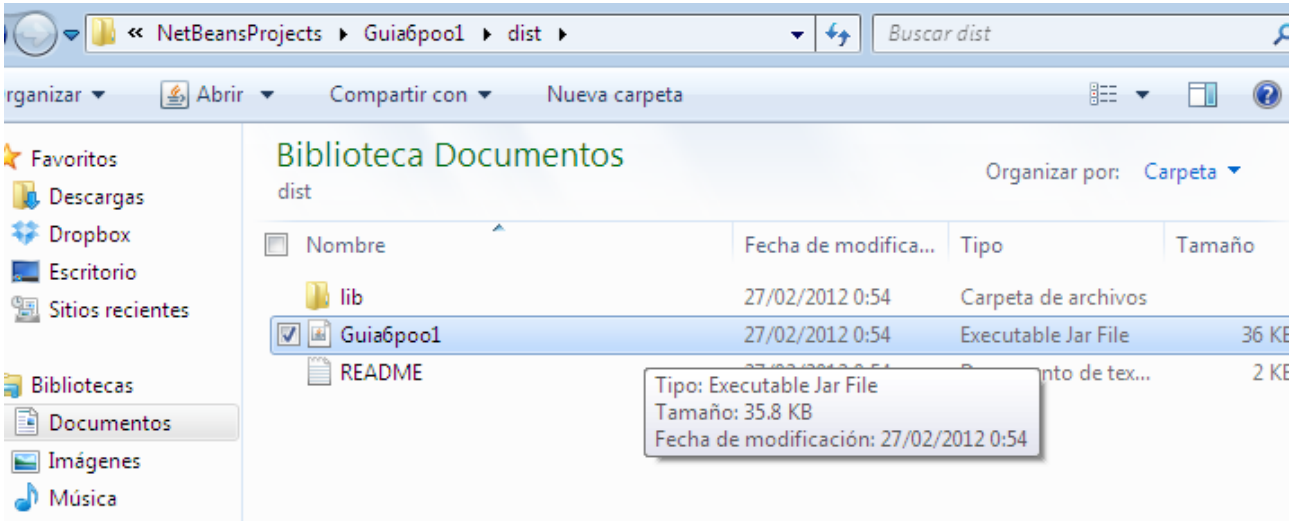
->Seleccionar **Add JAR/Folder**



->Buscar el conector y dar click en aceptar.



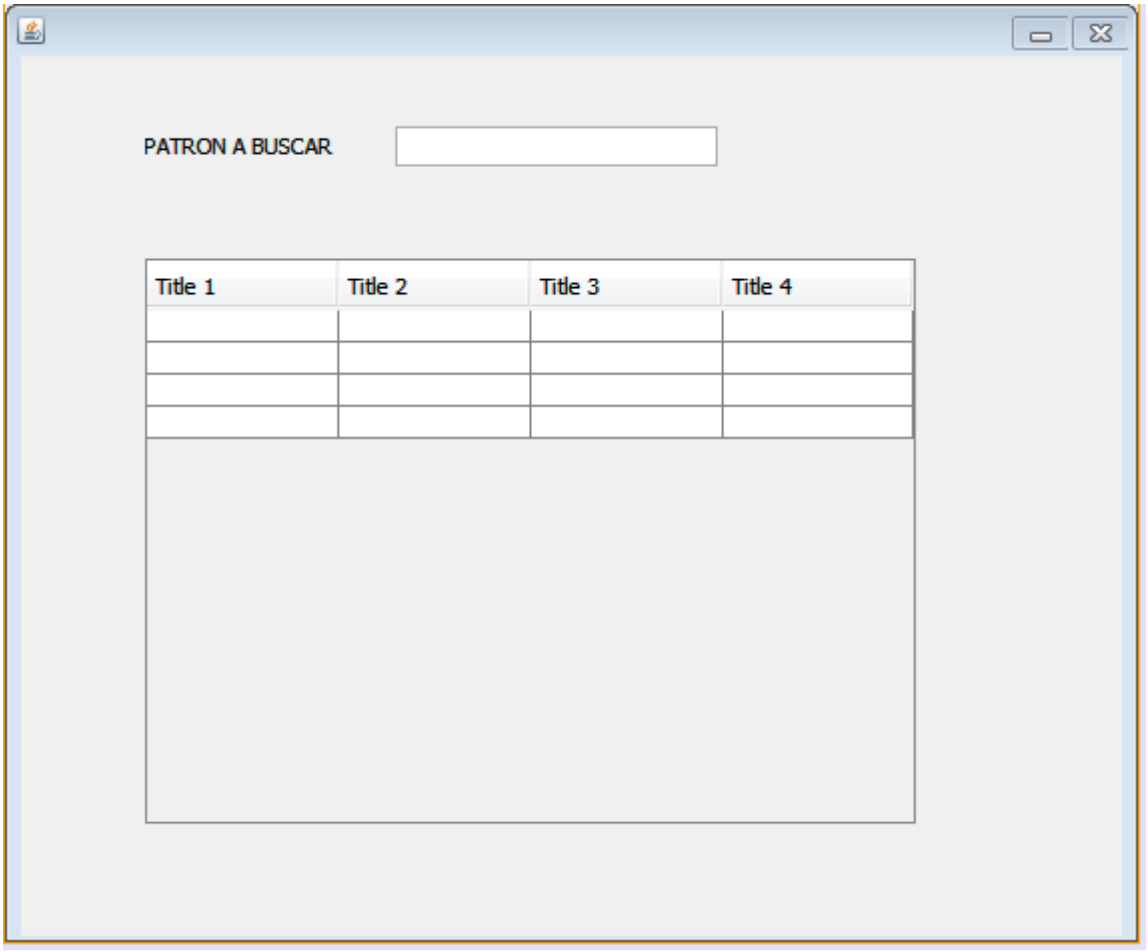
Finalmente si todo ha salido bien, puede ejecutar la aplicación dando doble clic a su archivo .jar:



El archivo .jar se encuentra en la carpeta generada llamada **dist**

Mostrando datos en un JTable:

26. Utilizaremos el jTable, para este punto debe de crear un nuevo JframeForm en el paquete "**guia6poo1**" que se llame **Jtable** y que tendrá un aspecto gráfico similar al mostrado en la siguiente imagen.



27. Cambiar las propiedades de cada componte según como se muestra en la siguiente tabla.

Control	Propiedad	Código	Valor
jTextField1	name	Variable name	txtBusqueda
jTable1	name	Variable name	jTableResultado

28. Agregar el siguiente código que permite declarar un atributo de tipo **DefaultTableModel**y los valores de inicialización para nuestra tabla que van dentro del constructor.

Nota: De ser necesario, debe importar javax.swing.table.DefaultTableModel; o utilizar "**Fiximports**", además de importar java.sql.* Puede guiarse de la imagen siguiente:

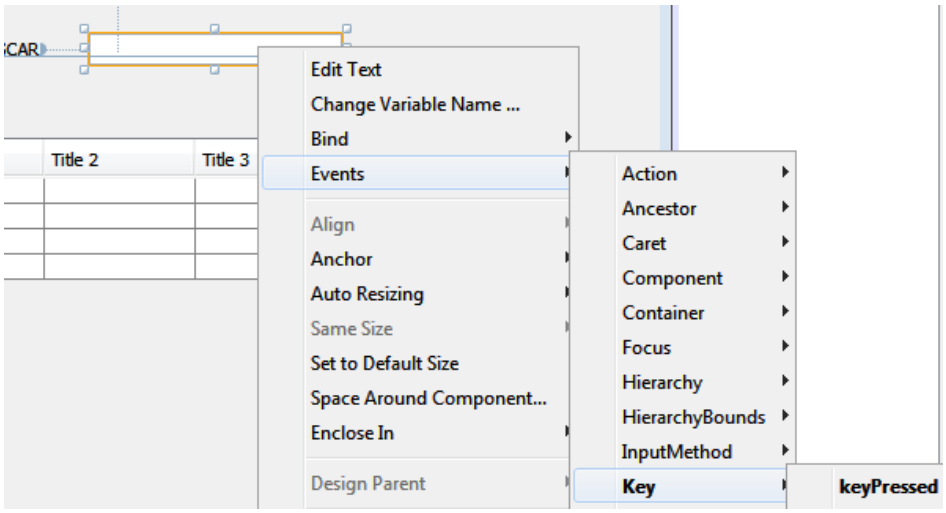
```
Design  History
package guia6poo1;

import javax.swing.table.DefaultTableModel;
import java.sql.*;
import java.util.logging.Level;
import java.util.logging.Logger;
public class JTable extends javax.swing.JInternalFrame

    DefaultTableModel   modelo1 = null;
    static int bandera=0;
    ResultSet resultado= null;
    Conexion con3= new Conexion();

    public JTable() throws SQLException{

        bandera = 1;
        initComponents();

        Object [][] data = null;

        String [] columns= {
            "ID","Nombres","Apellidos","Edad"
         };
        modelo1 = new DefaultTableModel(data, columns);
        this.jTableResultado.setModel(modelo1);

          con3.setRs("select * from usuarios");

          generarListado();

    }
```

Nota: El método generarListado aún no ha sido definido, por lo que debe ignorar el error de método no encontrado.

Agregar el método generarListado luego del contructor

```
void generarListado () throws SQLException{
    resultado = con3.getRs();
      while(resultado.next()){//agregando una nueva fila a la tabla
          Object [] newRow={ resultado.getInt(1),resultado.getString(2),resultado.getString(3),
          resultado.getString(4)};
          modelo1.addRow(newRow);
      }

      //es una buena práctica cerrar el ResultSet
      resultado.close();

}
```

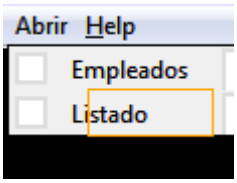
29. Ahora dentro en la caja de texto denominada txtBusqueda, agregue el manejador del evento KeyPressed



Agregue el siguiente código:

```
private void txtBusquedaKeyPressed(java.awt.event.KeyEvent evt) {

    while(modelo1.getRowCount()!=0) modelo1.removeRow(0);

    con3.setRs("select * from usuarios where concat(Nombres,' ',Apellidos ) "
            + "like '%" + this.txtBusqueda.getText() + "%'");

        try{
        generarListado();
        }catch(SQLException ex){
            Logger.getLogger(JTable.class.getName()).log(Level.SEVERE,null,ex);
        }
}
```

Agregue un nuevo menú al MDI



Y maneje el evento ActionPerformed (Puede dar doble clic en nuevo MenuItem para hacerlo de inmediato), agregando este código.

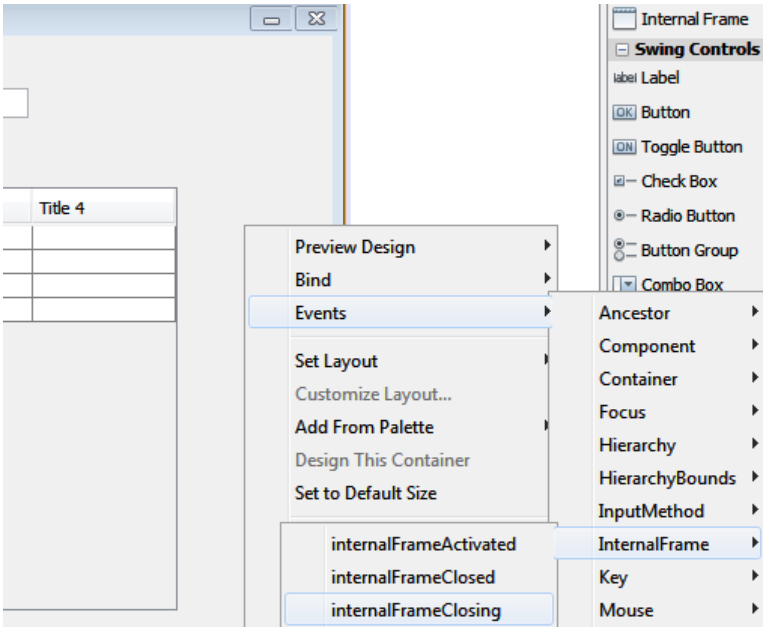
```
    //AGREGANDO UN NUEVO FORMULARIO
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {

        try{
            if(JTable.bandera==0){
                JTable listado = new JTable();
                desktopPane.add(listado);
                listado.show();

            }
        }catch(SQLException ex){
            Logger.getLogger(JTable.class.getName()).log(Level.SEVERE,null,ex);
        }
    }
```

Al igual que el paso 15, es necesario que manejemos un "bandera" para este formulario.

30. Ya que abrimos una conexión al inicializar el formulario también deberemos de cerrar la conexión al cerrar este formulario (exactamente lo mismo que el paso 21).

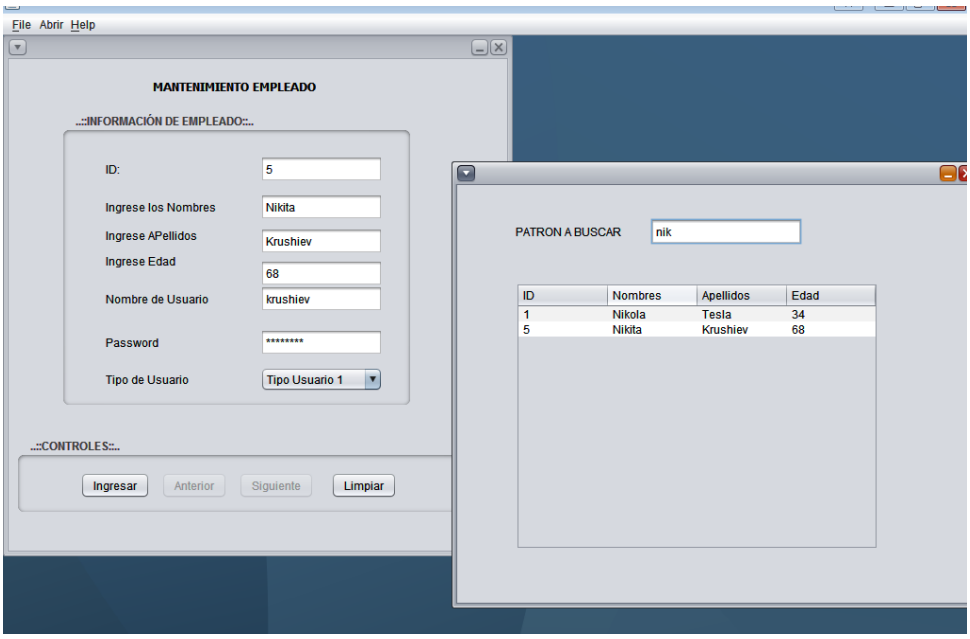
    - Seleccionar el Jframe interno llamado "**JTable**", ir a sus propiedades y modificar la propiedad DefaultCloseOperation->desplegar las opciones y seleccionar DO_NOTHING para eliminar la funcionalidad al dar click en la "X".

31. Modificar el **evento "InternalFrameClosing"**, agregaremos el método cerrar y lo modificaremos como se muestra a continuación.




```
    private void formInternalFrameClosing(javax.swing.event.InternalFrameEvent evt) {
try{
        bandera = 0;
        this.dispose();
        con3.cerrarConexion();
        }catch(SQLException ex){
            Logger.getLogger(JTable.class.getName()).log(Level.SEVERE,null, ex);
        }
                // TODO add your handling code here:
    }
```

32. Ejecutar, probar, crear el respectivo .jar y evaluar el resultado.



IV. EJERCICIOS COMPLEMENTARIOS

- Crear lo que falta del mantenimiento "eliminar,actualizar, buscar"(Agregar estas características al formulario de MantenimientoEmpleados)
- Crear un mantenimiento para las tablas
  -Alumno
  -Materias
  -AlumnoMateria

  Esta ya se había realizado con anterioridad, solo que ahora deberá utilizar formularios para interactuar con el usuario.