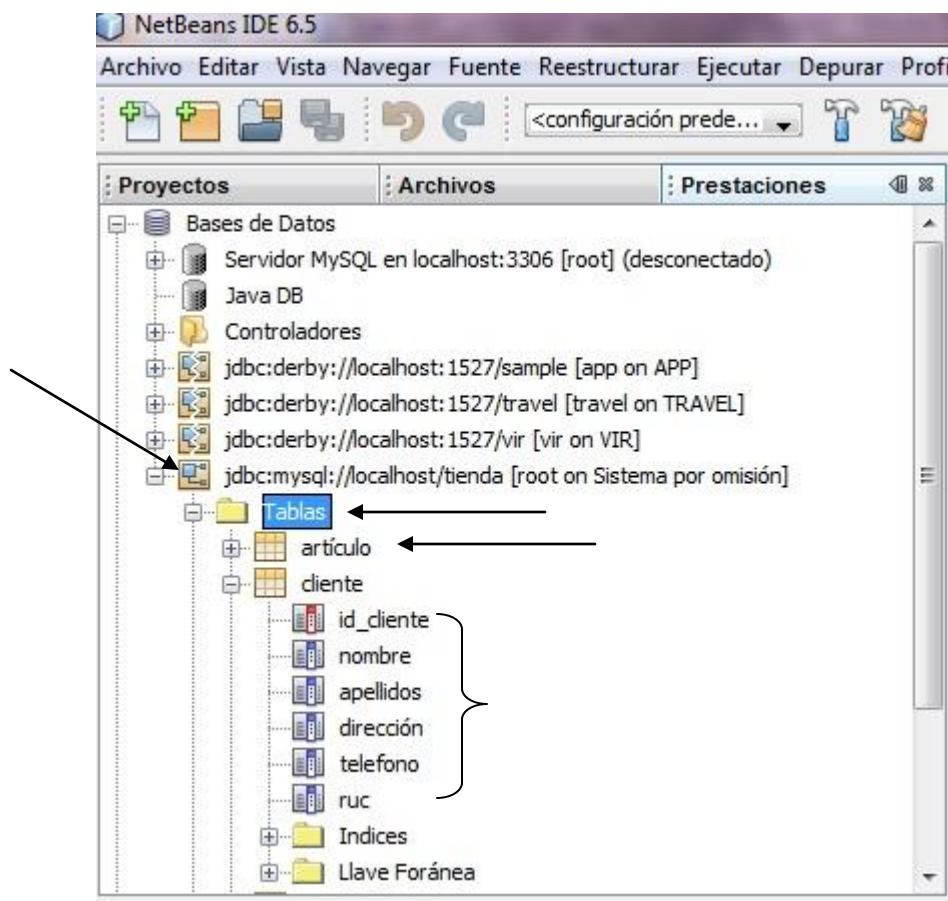


En el tutorial anterior se creó una Base de datos llamada “tienda” con tres tablas en el que usamos como motor de base de datos el MySQL y luego hicimos una conexión desde el entorno de NetBeans BD usando la API JDBC con una aplicación Java.

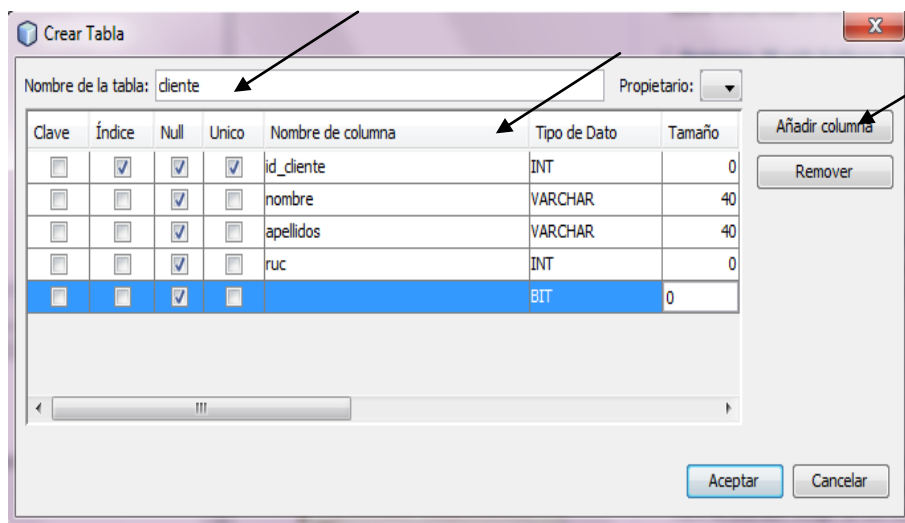
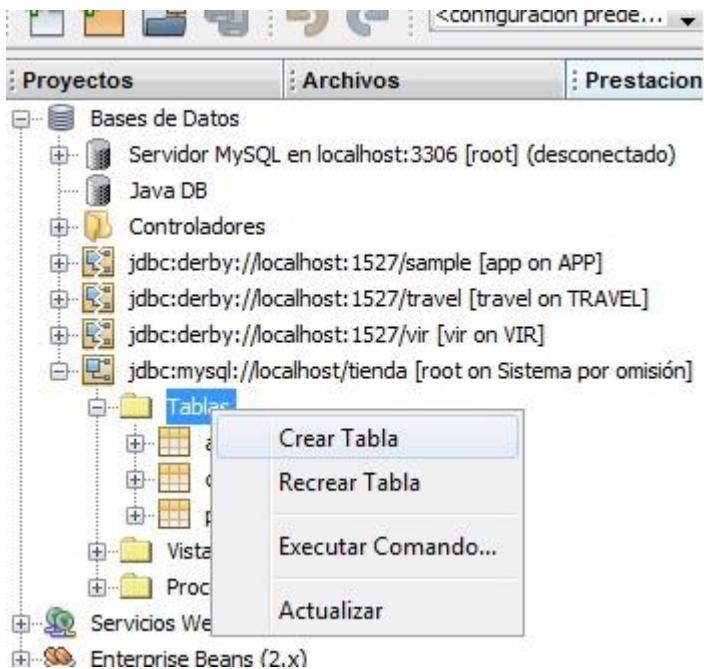
Ahora ya realizada dicha conexión, procederemos a usar el entorno de NetBeans para acceder a la Base de Datos y el paquete Java.SQL todo esto mediante un formulario ya que NetBeans nos permite hacer operaciones sobre la base de datos como crear y borrar tablas, agregar y eliminar columnas, agregar, modificar y eliminar registros de datos como realizar consultas.

Acá podemos apreciar la Base de Datos “tienda” conectada y con las tablas creadas, incluso también podemos ver sus campos.



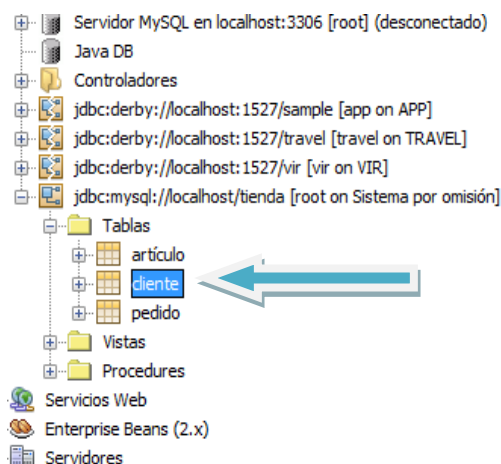
Si no tendríamos tablas en la Base de Datos, también podemos crearla desde el NetBeans, lo veremos a continuación.

Lo que hacemos es anti clic en Tablas y seleccionamos crear nueva tabla y procedemos a crear las tablas que crean conveniente.



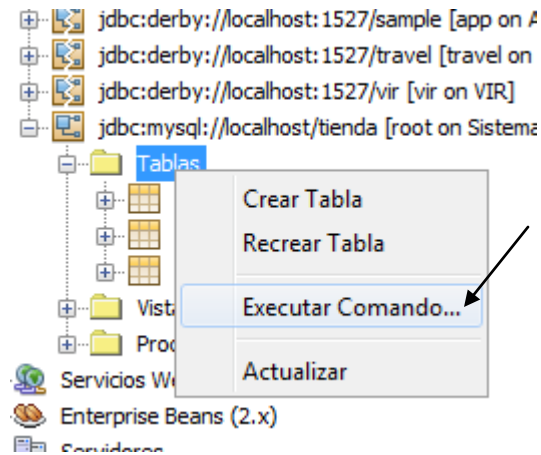
Añadir más datos

Se visualizara esta ventana, el cual es muy familiar al que usamos en el MySQL al ahora de crear las tablas.

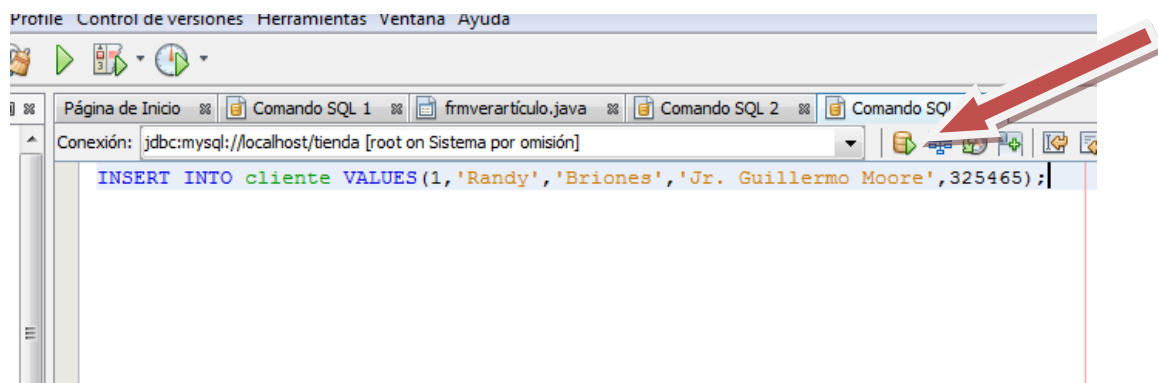


Luego de darle en aceptar, se visualizara en el entorno de NetBeans la nueva tabla creada, En el caso que hare esta vez, no añadí tablas por este entorno porque ya las había creado en el MySQL.

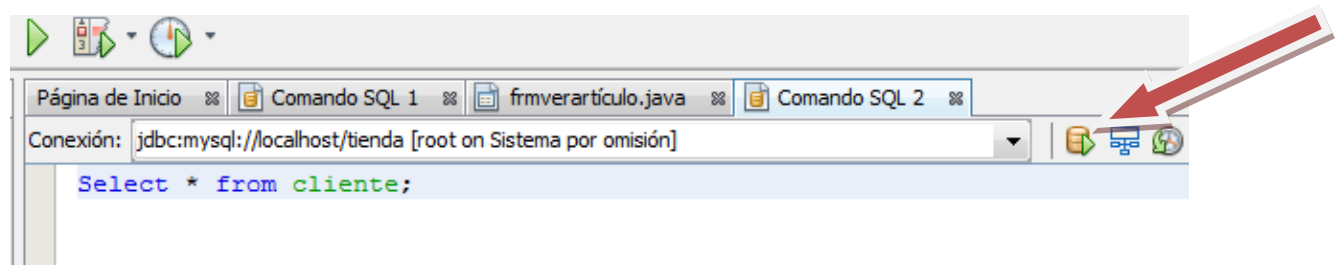
Ahora si empecemos a insertar registros en las tablas a través del comando insertar. Damos clic en el nodo Tablas y podremos ver el menú flotante en el cual seleccionaremos la opción Ejecutar Comando... en la imagen pueden visualizar mejor.



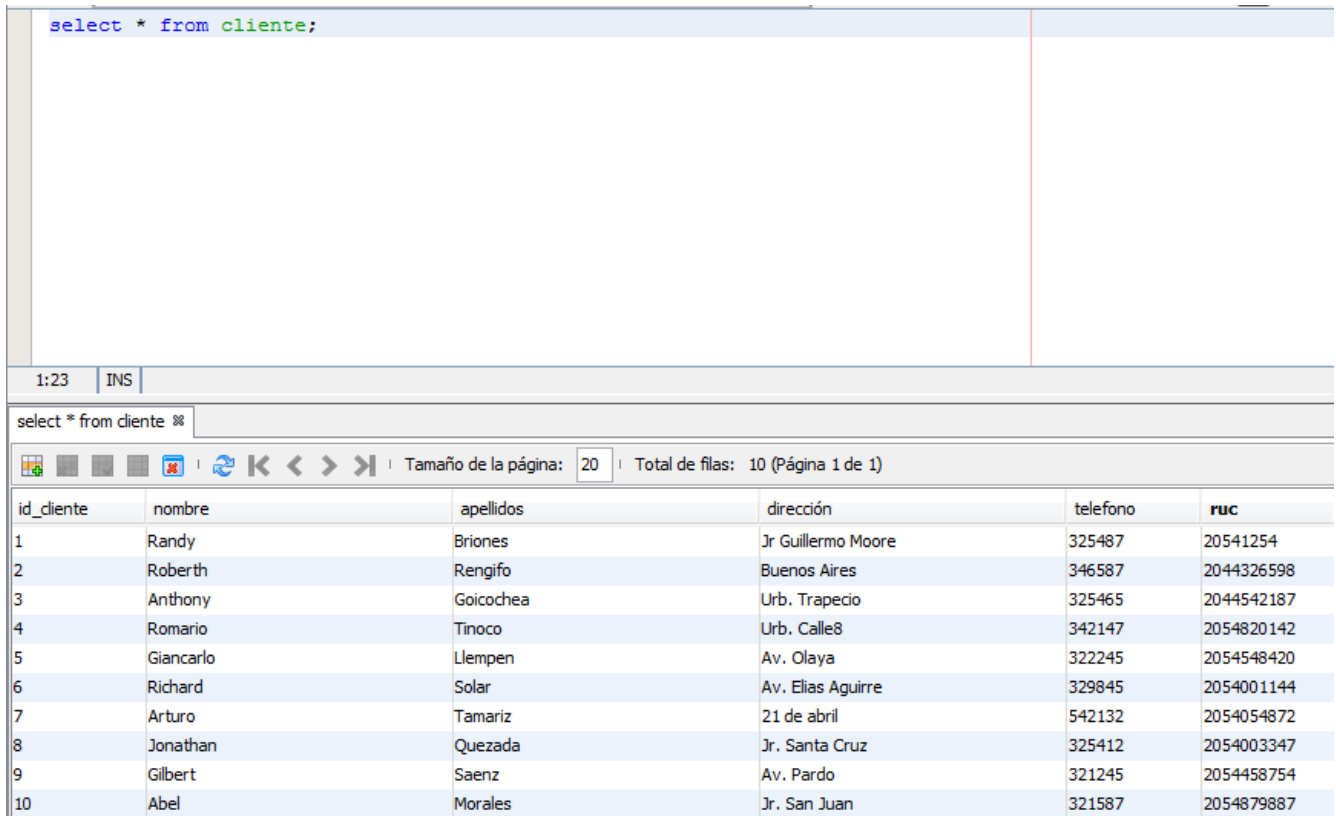
Escribimos el comando para insertar registro, y luego procedemos a ejecutar. (clic donde señala la flecha roja para ejecutar)



Ahora consultaremos el registro insertado con el comando select.



Acá visualizamos el resultado al utilizar el comando select para mostrar los registros.



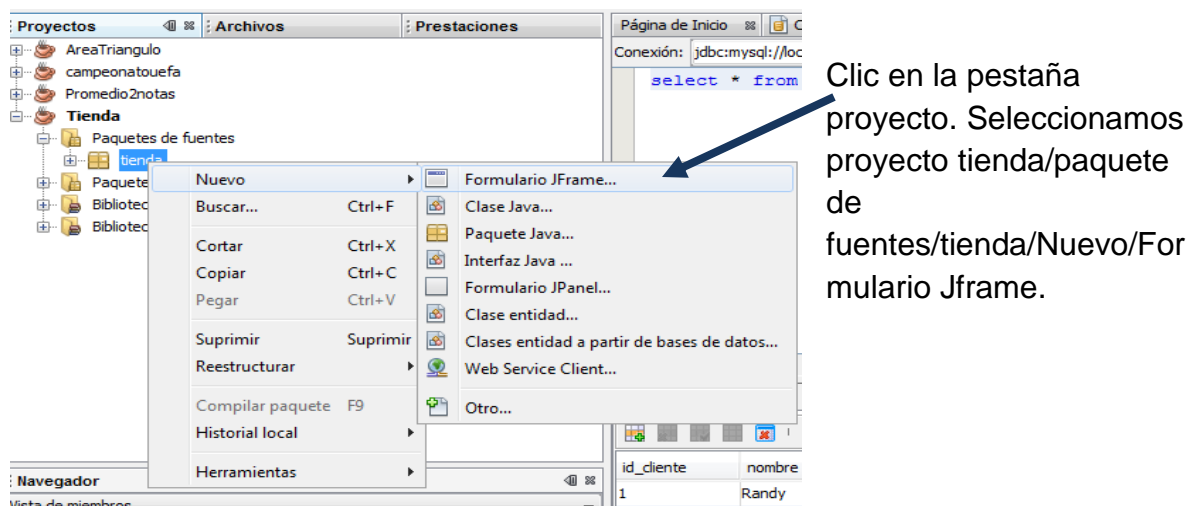
```
select * from cliente;
```

id_cliente	nombre	apellidos	dirección	telefono	ruc
1	Randy	Briones	Jr Guillermo Moore	325487	20541254
2	Roberth	Rengifo	Buenos Aires	346587	2044326598
3	Anthony	Goicochea	Urb. Trapecio	325465	2044542187
4	Romario	Tinoco	Urb. Calle8	342147	2054820142
5	Giancarlo	Llempen	Av. Olaya	322245	2054548420
6	Richard	Solar	Av. Elias Aguirre	329845	2054001144
7	Arturo	Tamariz	21 de abril	542132	2054054872
8	Jonathan	Quezada	Jr. Santa Cruz	325412	2054003347
9	Gilbert	Saenz	Av. Pardo	321245	2054458754
10	Abel	Morales	Jr. San Juan	321587	2054879887

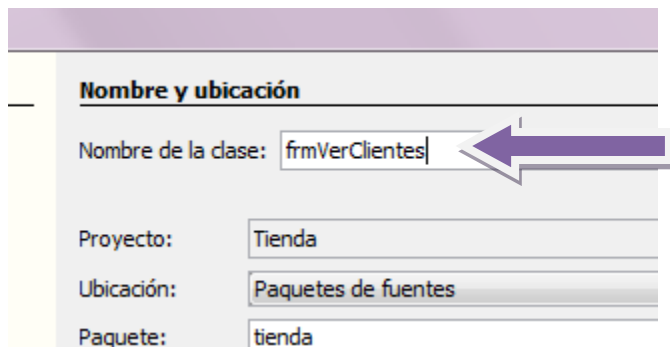
Como verán el entorno de NetBeans nos ofrece la oportunidad de acceder y manipular los datos y las estructuras de los elementos que conforman una base de datos.

Entonces ya sabiendo insertar y mostrar datos, plasmaremos estos registros en un formulario para que sea mejor visualizado.

Creamos un objeto JFrame para abrir un nuevo formulario.



Visualizaremos la ventana del formulario. La cual denominare frmVerClientes.
Procedemos a dar clic en Terminar.



Nombre y ubicación

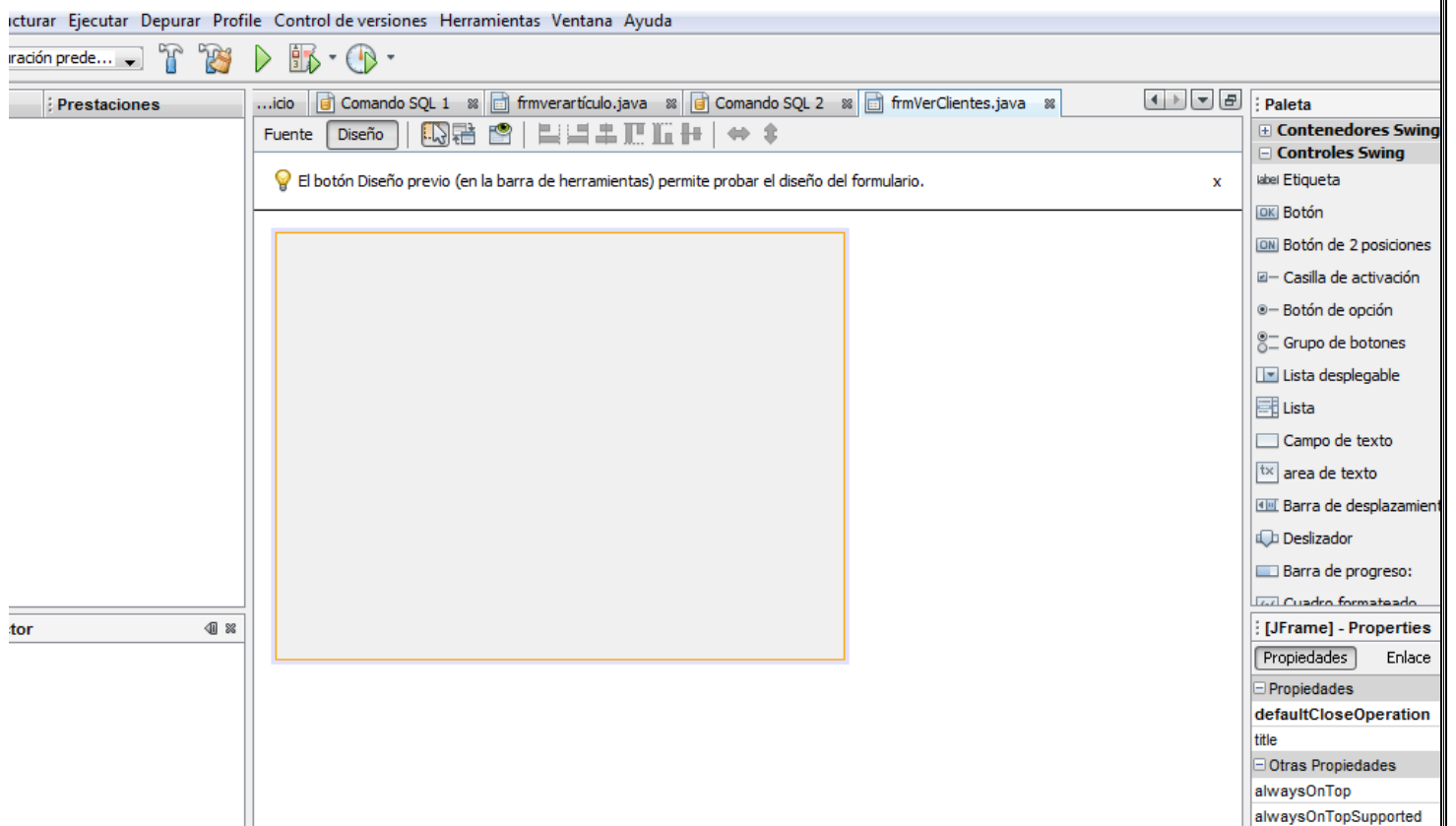
Nombre de la clase:

Proyecto:

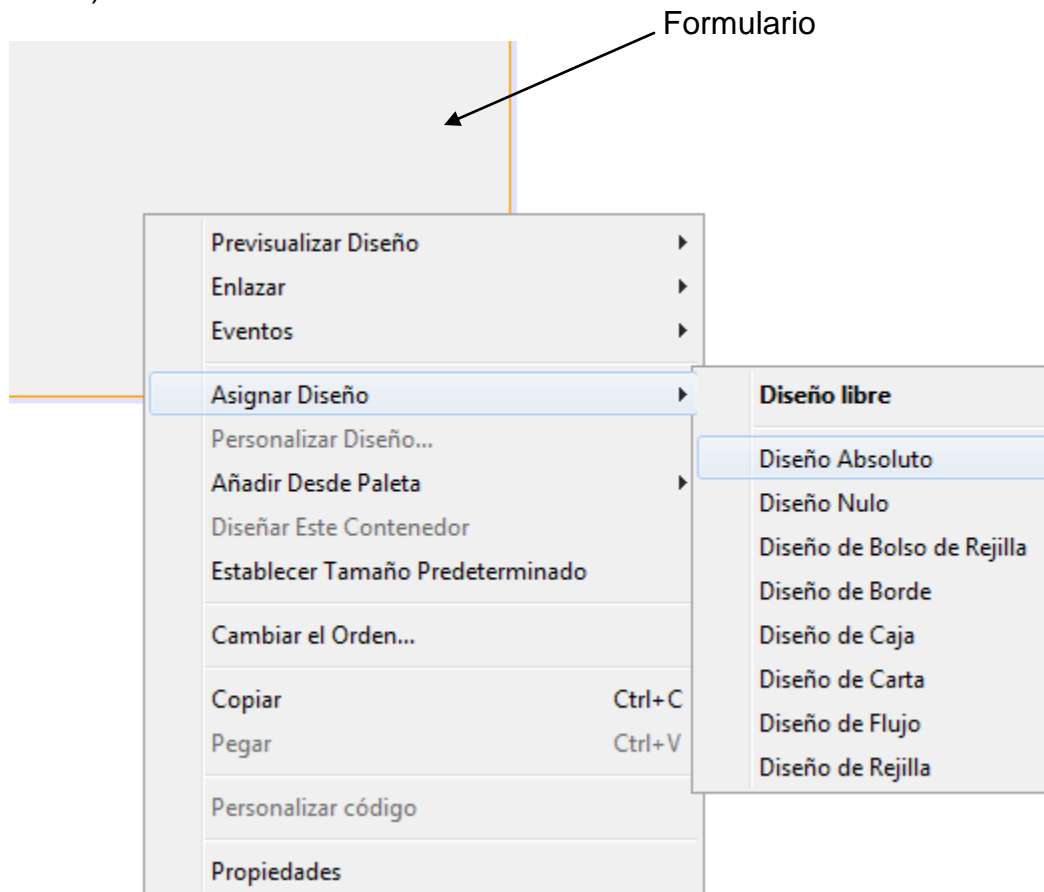
Ubicación:

Paquete:

A continuación se mostrara de la siguiente manera:



Seleccionamos la opción Asignar Diseño/*Diseño Absoluto*, lo cual nos permitirá usar los objetos de control más libremente. (anti clic en el nuevo formulario)



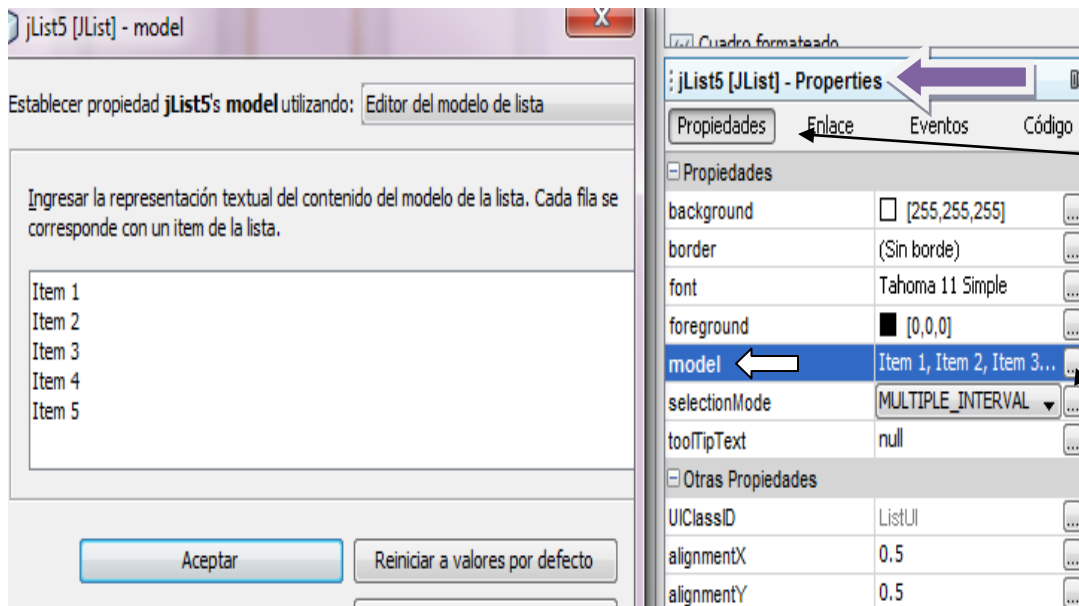
Colocamos los objetos de control quedando el diseño del formulario de la siguiente manera:

Diagram illustrating the components of a Java Swing window titled "Clientes de la Tienda".

The window contains the following elements:

- Header Area:** A row of labels: `id_cliente`, `Nombre`, `Apellido`, `Dirección`, `Teléfono`, and `Ruc`. These are identified by the callout "Etiquetas/label".
- Content Area:** Six empty rectangular boxes, each containing a list of items: `Item 1`, `Item 2`, `Item 3`, `Item 4`, and `Item 5`. These are identified by the callout "Listas".
- Footer Area:** A button labeled "Cerrar", identified by the callout "Botón".

Borramos los Items de cada objeto Jlist y colocamos los nombres a cada objeto de Control. Recuerden que para eliminar los Items de cada Jlist hay que hacer uso de model que se encuentra en la ventana de propiedades.



Realizamos lo siguiente:

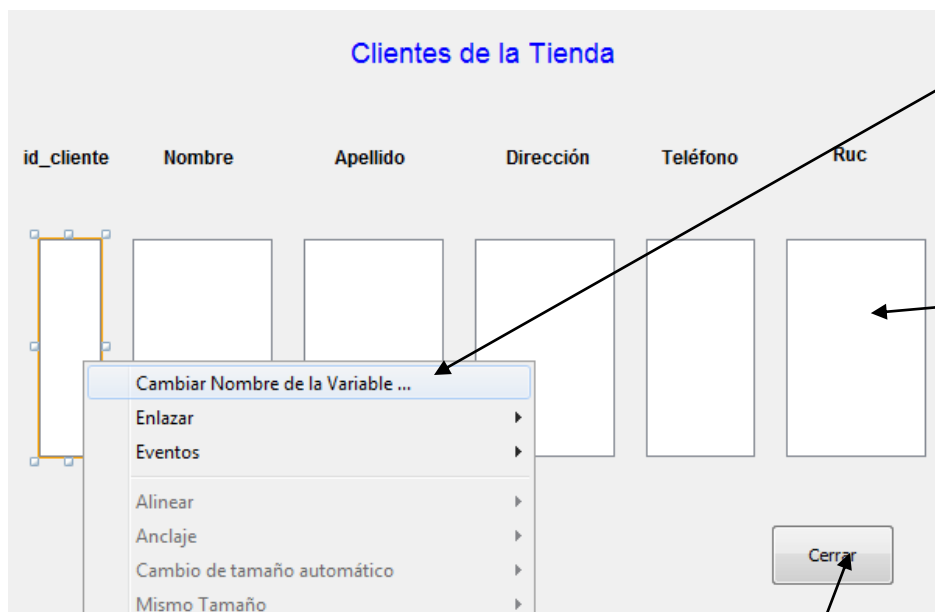
Pestaña Propiedades

Se visualiza la opción model (fecha blanca)

Parte derecha: clic en:

Veremos la ventana siguiente:

Borramos los Items y ponemos aceptar. (Lo mismo con las demás listas).



Ahora anti clic y procedemos a cambiar los nombre de las variables. De las listas y botones.

Istruc
Isttelefono
Istdireccion
Istapellido
Istnombre
Istidcliente

btncerra

Ahora procedemos a la programación:

Nota: El botón cerrar podrá funcionar con esta línea de código: `dispose ();`



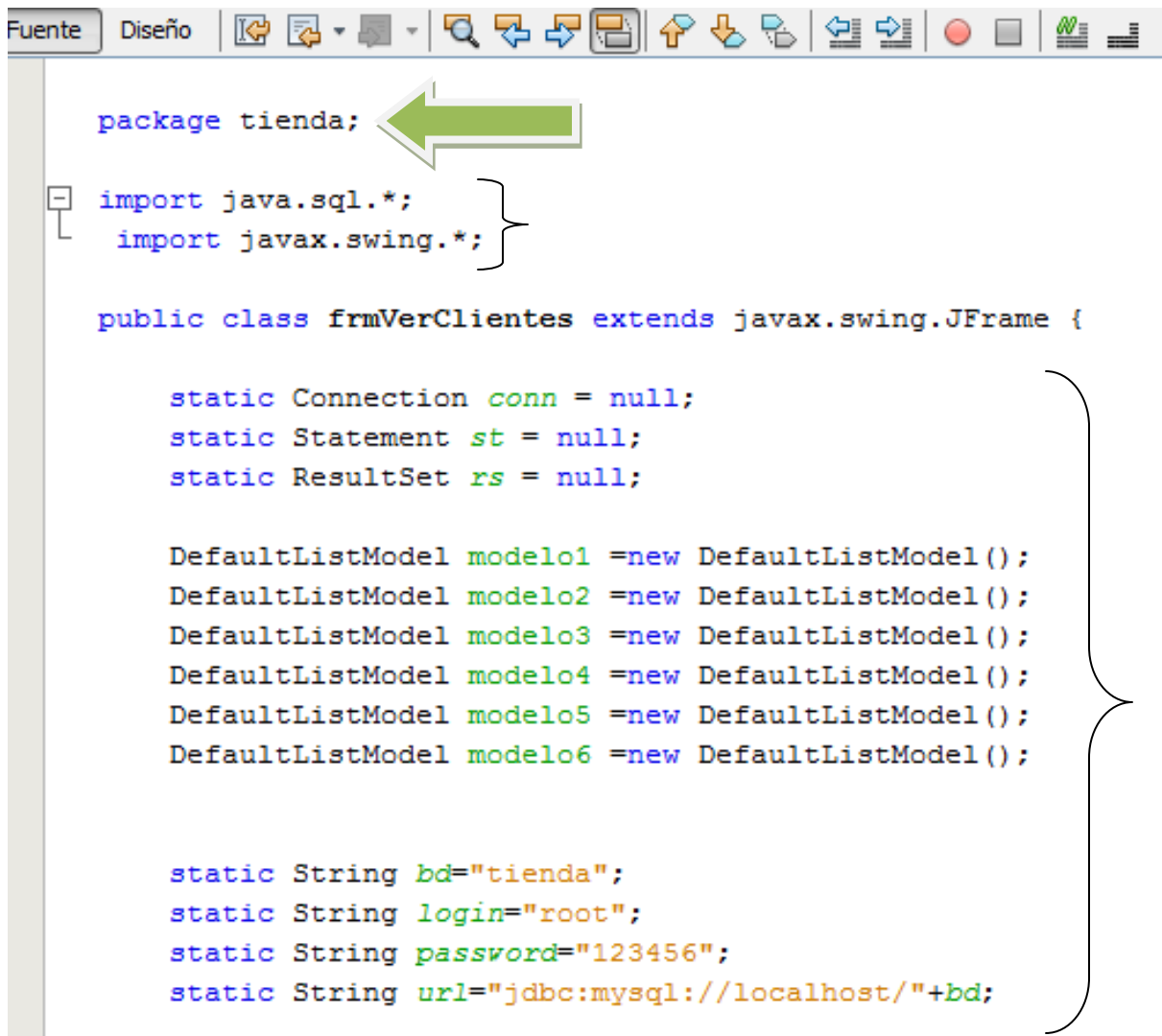
Doble clic,

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent ev) {  
    dispose();  
}
```

Procedemos ahora en la pestaña fuente en donde ira la parte de código.

```
package tienda;  
  
/**  
 * To change this template, choose Tools | Templates  
 * and open the template in the editor.  
 */  
  
/**  
 * frmVerClientes.java  
 *  
 * Created on 24/09/2009, 01:44:57 PM  
 */  
  
public class frmVerClientes extends javax.swing.JFrame {
```


Dentro del paquete tienda ira la siguiente codificación:



```
package tienda;

import java.sql.*;
import javax.swing.*;

public class frmVerClientes extends javax.swing.JFrame {

    static Connection conn = null;
    static Statement st = null;
    static ResultSet rs = null;

    DefaultListModel modelo1 =new DefaultListModel();
    DefaultListModel modelo2 =new DefaultListModel();
    DefaultListModel modelo3 =new DefaultListModel();
    DefaultListModel modelo4 =new DefaultListModel();
    DefaultListModel modelo5 =new DefaultListModel();
    DefaultListModel modelo6 =new DefaultListModel();

    static String bd="tienda";
    static String login="root";
    static String password="123456";
    static String url="jdbc:mysql://localhost/"+bd;
```

Las líneas de código que están en llaves:

Primera llave:

Java.sql: conexión con la base de datos y **javax.swing:** para poder utilizar la clase JOptionPane para la visualización de un mensaje a través de su método ShowMessageDialog.

En la segunda llave: defino las variables que voy a usar. Vendría hacer los atributos de la clase **frmVerClientes**. Como tenemos establecer conexión usaremos un objeto Connection (conn), para hacer una operación de consulta usaremos un objeto Statement (st) y para almacenar los resultados de la consulta usaremos un objeto ResultSet (rs). Las cuales se declaran nulas, están vacías. (null)

Como la aplicación hace uso de los Jlist se tiene crear 6 objetos instanciados de la clase DefaultListModel.


DefaultListModel: es el tipo de variable.

Modelo=new: declarando estas variables.

DefaultListModel: Este en un método constructor igual que el nombre de la clase.

Luego seguimos con la programación:

```
public static Connection Enlace(Connection conn)throws SQLException
{
    try
    {
        Class.forName("com.mysql.jdbc.Driver");
        conn = DriverManager.getConnection(url,login,password);
    }
    catch(ClassNotFoundException c)
    {
        JOptionPane.showMessageDialog(null,c);
    }
    return conn;
}
```



Preparamos variables para indicar el nombre de la base de datos, el login, el password y el url. En esta parte se ha diseñado un método denominado Enlace que permitirá establecer conexión con la base de datos “tienda”.

En la llave la flecha verde “Base de Datos”

Flecha roja: indica que el objeto conexión (conn) ya no es nulo.

Luego podemos ver que la clase frmVerClientes tienen un método del mismo nombre frmVerClientes (), esto es lo que se denomina método constructor. Deseamos que los datos se muestren en los objetos Jlist al momento de la ejecución del formulario debemos programar en el método antes mencionado.

```

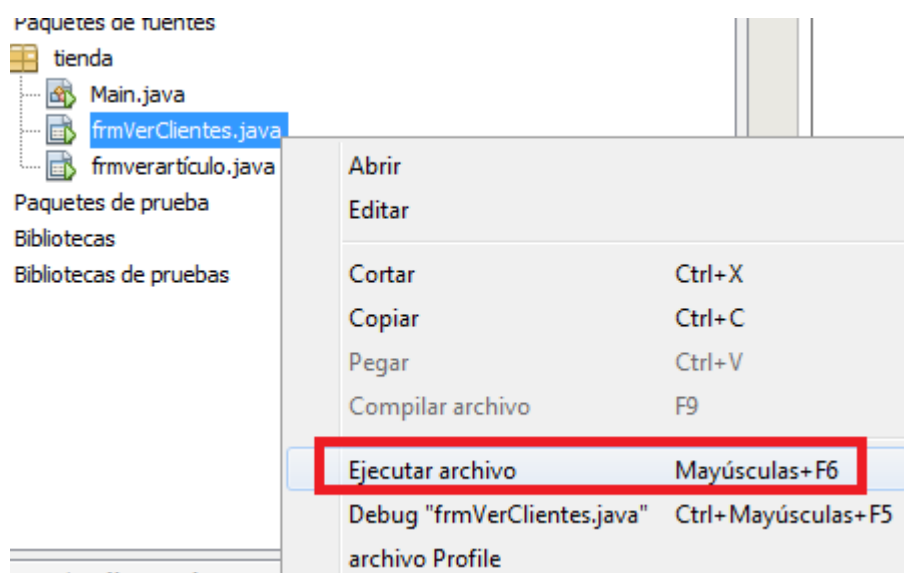
public frmVerClientes() {
    initComponents();
    lstidcliente.setModel(modelo1);
    lstnombre.setModel(modelo2);
    lstapellido.setModel(modelo3);
    lstdireccion.setModel(modelo4);
    lsttelefono.setModel(modelo5);
    lstruc.setModel(modelo6);

    try
    {
        conn=Enlace(conn);
        st=conn.createStatement();
        rs=st.executeQuery("select * from cliente");
        while(rs.next())
        {
            modelo1.addElement(rs.getString(1));
            modelo2.addElement(rs.getString(2));
            modelo3.addElement(rs.getString(3));
            modelo4.addElement(rs.getString(4));
            modelo5.addElement(rs.getString(5));
            modelo6.addElement(rs.getString(6));
        }
    }
    catch (SQLException c)
    {
        JOptionPane.showMessageDialog(null,"Error"+c.getMessage());
    }
}

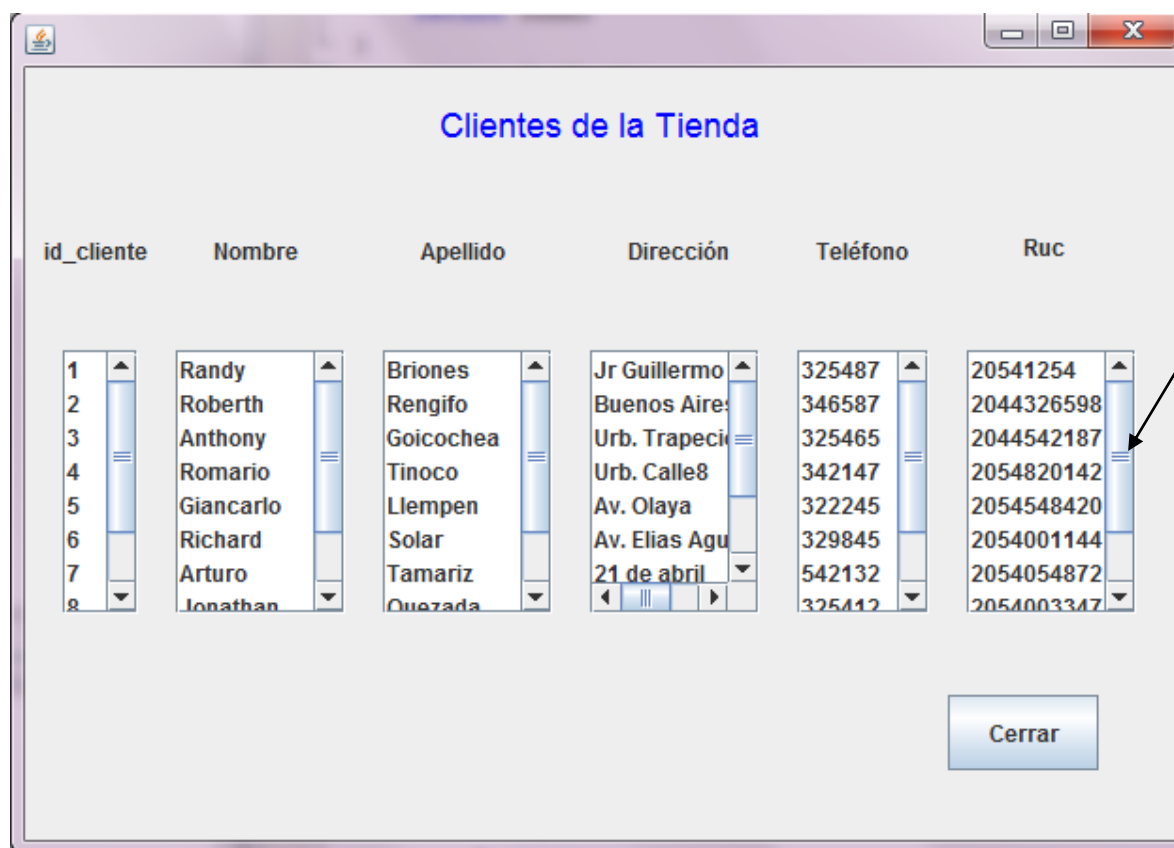
```

Los objetos instanciados de la clase DefaultListModel deben ser vinculados a cada uno de las cajas de listas. Luego usando el bloque try { } que sirve para Interceptar errores y si lo hubiera, ejecutaría lo programado en el bloque catch { }, establecemos la conexión usando el método Enlace, creamos el objeto **st** de tipo Statement (se encuentra listo para realizar una operación) y luego ejecutamos una Sentencia de consulta con select * from cliente (en este caso) cuyo resultado va ocasionar que los datos se almacenen en el objeto rs del tipo ResultSet. Finalmente con el método next se logra desplazar a través de los registros de datos para ir llenado los objetos modelo1, modelo2, modelo3, modelo4, modelo5 y modelo6. De esta forma llenamos los objetos Jlist lo que nos permitirá ver en el formulario los datos de la tabla de Cliente.

Ahora ejecutamos, clic derecho en: frmVerClientes.java

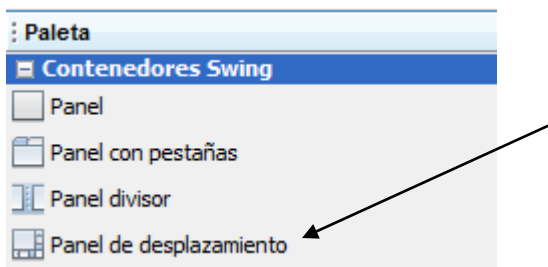


Y se mostrara de esta manera:



Por cierto las cajas de listas aparecen con paneles de desplazamiento. (son opcionales) generalmente cuando hay muchos registros.

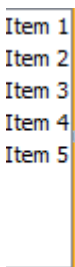
Nota: Los Paneles de Desplazamiento se encuentran dentro de Paletas/Contenedores Swing



Siempre se colocan primero los paneles de desplazamiento y luego encima las cajas de listas.



Panel de Desplazamiento



Caja de lista

Lista Encima del panel

