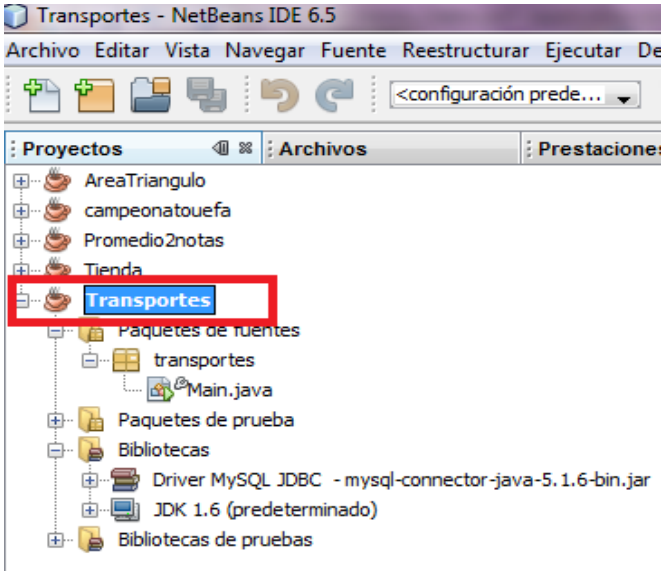
JTABLE_ MANTENIMIENTO DE DATOS EN NETBEANS

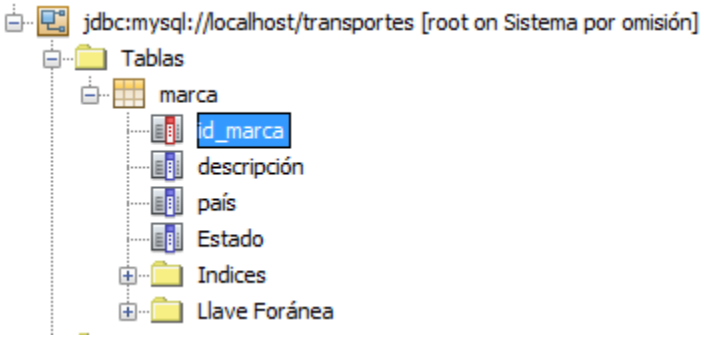
Tenemos la Base de Datos Transportes, que la cree en el localhost y solo hice la conexión con el NetBeans.



La Base de Datos transportes no tiene tablas, ahora procederé a crear una tabla denominada “marca” que tendrá la siguiente estructura:

idmarca int not null autoincrement
descripción varchar(20)
país varchar(20)
Estado bit
idmarca (clave primaria)

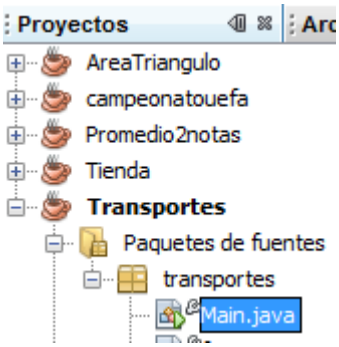
Nota: La tabla “marca” será creada directamente desde el NetBeans.

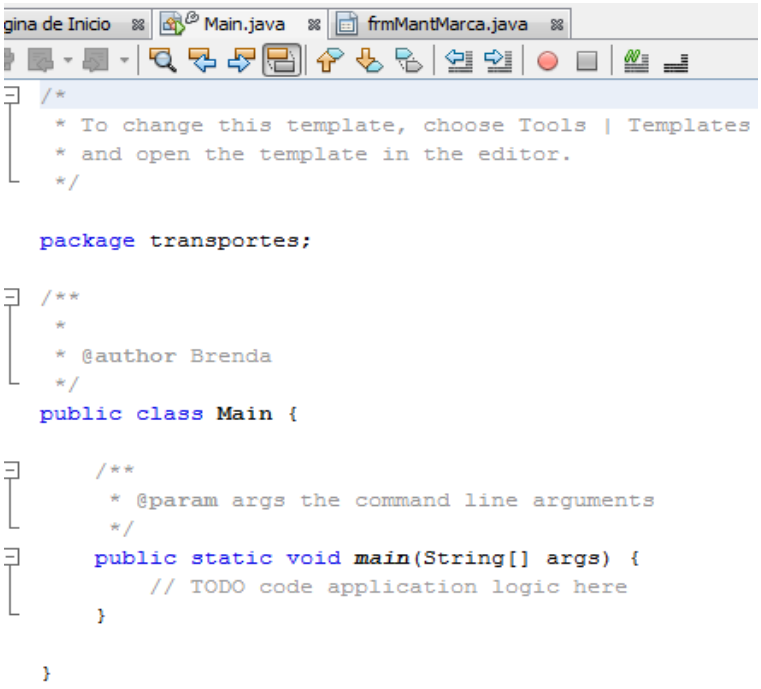


Ya esta creada la tabla con sus campos. (en el entorno NetBeans)

Ahora si entremos en detalles con la tabla “marca” creare una aplicación que permita el mantenimiento de la tabla utilizando el Objeto JTable para la visualización de los datos.

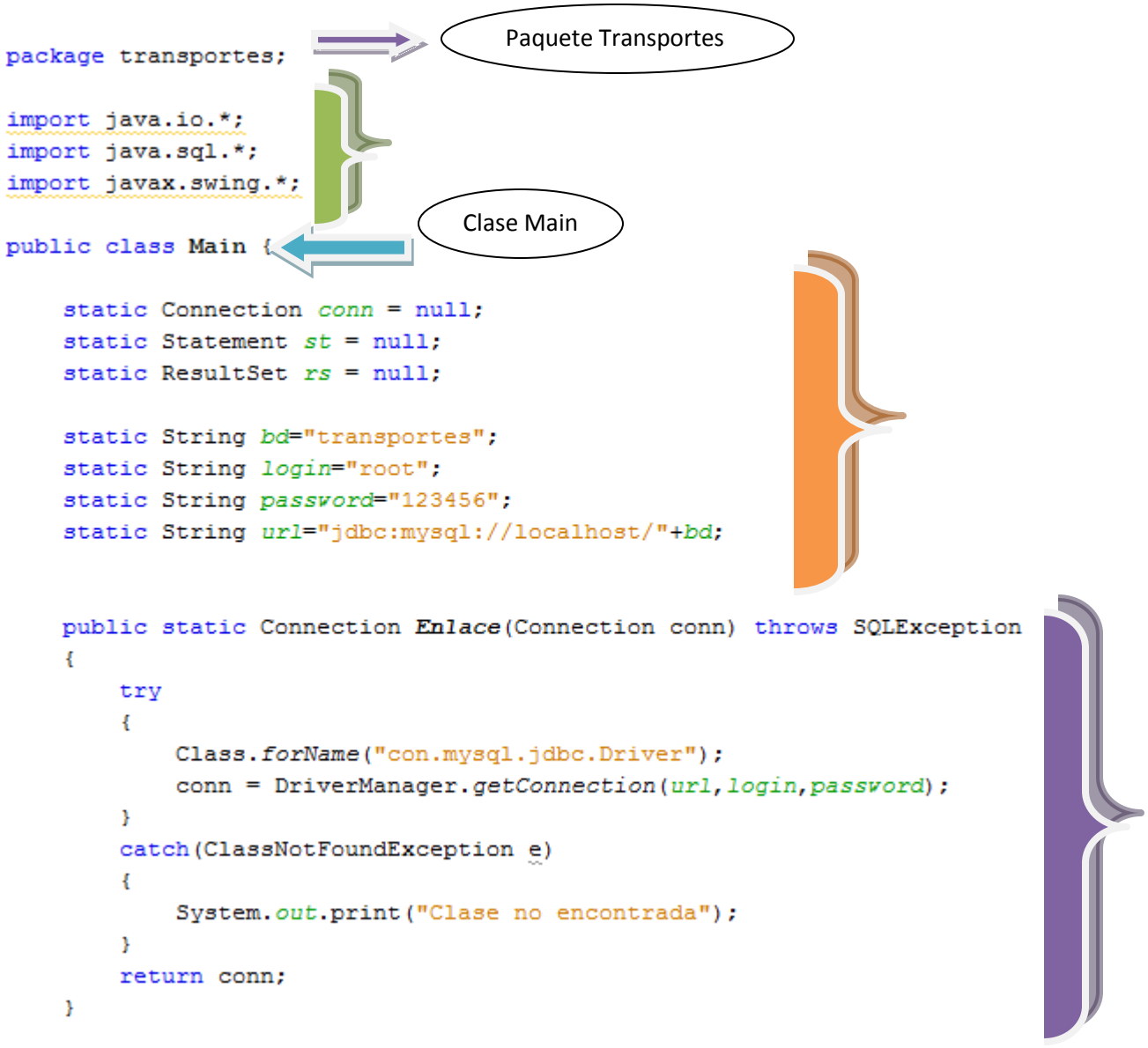
Nos vamos a la pestaña Proyectos donde vamos a crear métodos en la clase Main, necesarios para la conexión con la base de datos y el enlace con la tabla de marca.





Es allí en donde ingresaremos el siguiente código.

A continuación (en la llave verde) agregamos los paquetes: **java.io, java.sql y javax.swing.**
El paquete **java.io** contiene clases que soportan entrada/salida. Las clases del paquete son principalmente streams; sin embargo, se incluye una clase para ficheros de acceso aleatorio. Las clases centrales del paquete son InputStream y OutputStream las cuales son clases abstractas base para leer de y escribir a streams de bytes, respectivamente. Y el **javax.swing** es necesario para el uso de los objetos del paquete swing principalmente el **JOptionPane** para la visualización de posibles mensajes de error. Estando dentro de la **clase Main** establecemos las variables conn del tipo Connection(llave anaranjada), **st** del tipo Statement y **rs** del tipo ResultSet.
También definimos las varibales bd, login, password y url.
(Llave morada) creamos el primer método estático denominado ***Enlace*** que devolverá un objeto del tipo Connection. Este método nos ayudará establecer la conexión con la base de datos transportes.

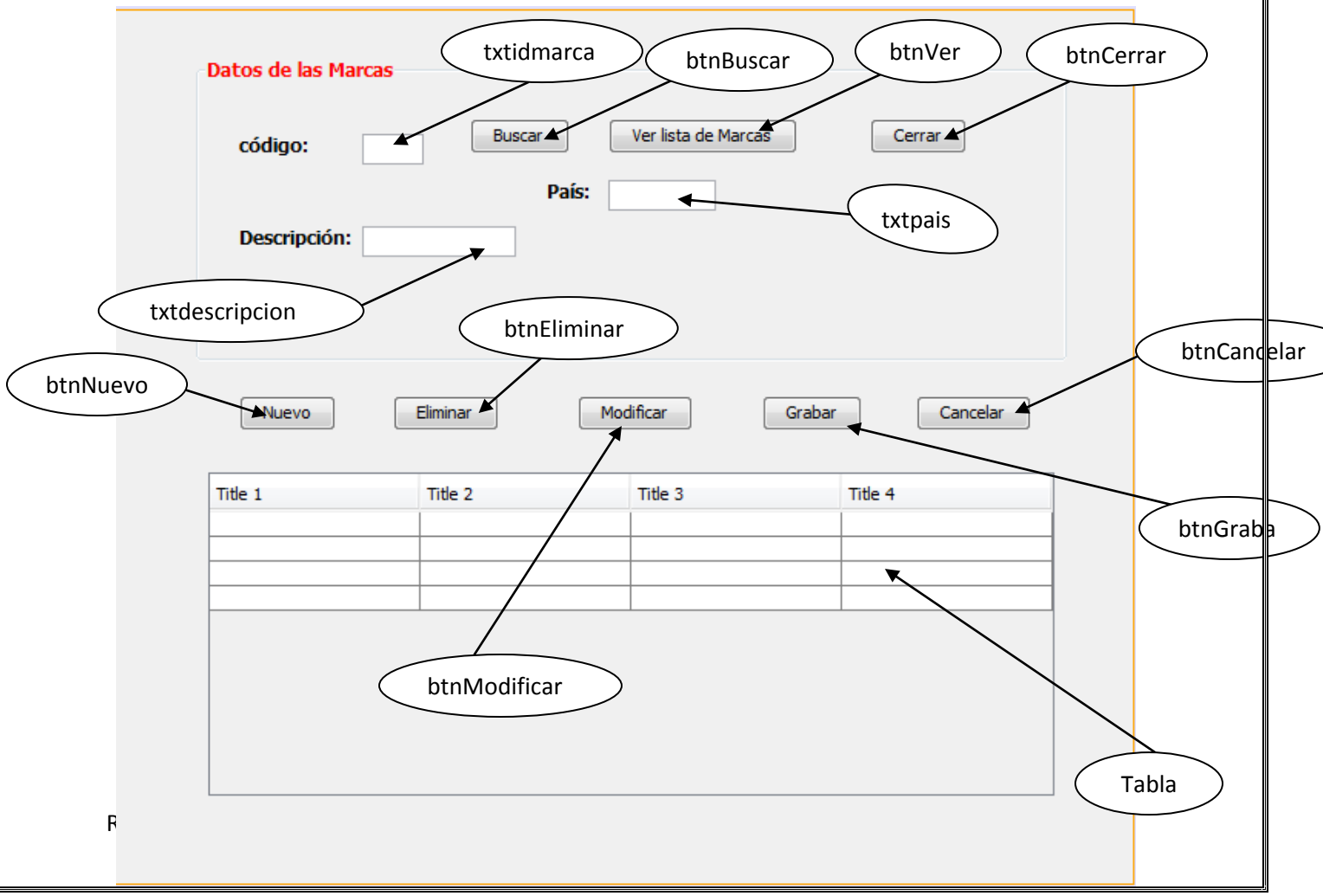



```
package transportes;

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

public class Main {

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

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


    public static Connection Enlace(Connection conn) throws SQLException
    {
        try
        {
            Class.forName("con.mysql.jdbc.Driver");
            conn = DriverManager.getConnection(url,login,password);
        }
        catch(ClassNotFoundException e)
        {
            System.out.print("Clase no encontrada");
        }
        return conn;
    }
```

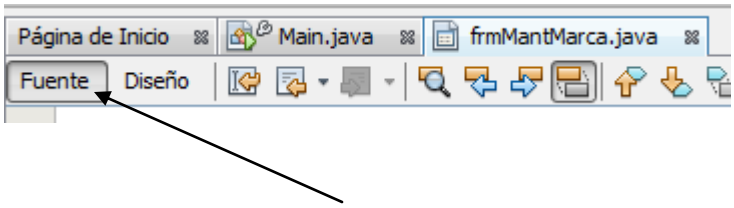
Seguimos creando métodos. Se crea el método ***sta*** que devolverá un objeto del tipo Statement el cual permite crear el objeto del tipo Statement a partir del objeto **Connection** a través del método ***createStatement ().*** Finalmente se necesita tener un método denominado ***EnlEst*** que permitirá enlazarnos con la tabla de marca aplicando el método ***executeQuery()*** estableciendo a través del comando de consulta “select” a todos los campos de la tabla de “marca” solo aquellos cuyo estado es igual a 1, es decir, las marcas de buses que están habilitados para su uso. Si observamos en cada uno de los métodos se está usando throws SQLException, esto quiere decir que los métodos usan excepciones (intercepción de errores) para los errores que se pueden presentar durante la conexión y acceso de datos.

```java
public static Statement sta(Statement st) throws SQLException
{
    conn=Enlace(conn);
    st=conn.createStatement();
    return st;
}

public static ResultSet EnlEst(ResultSet rs) throws SQLException
{
    st=sta(st);
    rs=st.executeQuery("Select * from marca where estado=1");
    return rs;
}
```

Nota: La tabla de marca tiene realmente 3 campos, pero el último es para indicar si el marca de bus esta habilitado para su uso o no lo está, esto quiere decir si hacemos una eliminación se procederá a cambiar el estado a 0 (eliminación lógica) y cada vez que grabemos los datos de un nuevo estadio se habilitará.



```
package transportes;

import java.io.*;
import java.sql.*;
import javax.swing.*;
import javax.swing.table.*;
import transportes.Main.*;
```



Importamos los paquetes siguientes paquetes:
java.sql: para acceder a base de datos
javax.swing: para el uso de los controles visuales
javax.swing.table: para el manejo de las clases del paquete table y el paquete transportes que contiene a la clase Main con lo cual podremos hacer uso de todos los métodos que tenga.

A continuación realizamos la construcción de la clase frmMantMarca, estableciendo como variables o atributos ***conn***, ***st*** y ***rs***. Como se está utilizando un objeto JTable se define la variable ***dtm*** del tipo **DefaulTableModel**. En el método constructor hacemos uso del método **activaBotones (esta parte la veremos más adelante)**, se establece un vector o arreglo del tipo String donde se coloca los títulos que serán de cada una de las columnas del objeto **JTable**. A partir del método **setColumnIdentifiers** indicamos los datos del vector títulos al objeto ***dtm*** y con el método setModel vinculamos el objeto ***dtm*** al objeto JTable denominado ***tablamarca***. Los métodos **setSize** y **setLocation** es para establecer el tamaño y la localización del formulario en la pantalla del computador.

```
public class frmMantMarca extends javax.swing.JFrame {

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

    DefaultTableModel dtm = new DefaultTableModel();

    public frmMantMarca() {
        initComponents();
        activaBotones(true,false,false,false);
        String titulos[]={"id_marca","descripción","país","Estado"};
        dtm.setColumnIdentifiers(titulos);
        tablamarca.setModel(dtm);
        setSize(530,230);
        setLocation(200,200);
    }
```

El método **activaBotones** es para habilitar o inhabilitar el uso de los botones de comando, esto dependerá en qué circunstancias que nos encontremos en la ejecución de la aplicación de mantenimiento de datos de marcas. Con el método **limpiarDatos** se limpia los cuadros de textos.

```
        public void activaBotones(boolean n, boolean e, boolean m, boolean g)
        {
           btnNuevo.setEnabled(n);
           btnEliminar.setEnabled(e);
           btnModificar.setEnabled(m);
           btnGrabar.setEnabled(g);
        }
public void limpiarObjetos()
{
    txtidmarca.setText("");
    txtpais.setText("");
    txtdescripcion.setText("");
`
```

Nombre por las primeras letras n = nuevo, e= eliminar, m= modificar, g= grabar

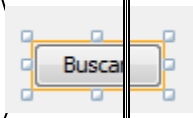
(“ “) comillas cuadro de texto vacio.

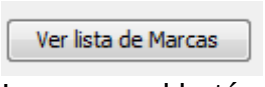
A continuación:

En el botón Buscar:

En el botón de comando **btnBuscar** si está habilitado después de dar clic en dicho botón, se procederá a la conexión con la base de datos, luego en la variable ***rs*** se almacenará los datos provenientes de la tabla marca. En la variable ***b*** se coloca el valor ingresado en el cuadro de texto **txtidmarca**. Se define una variable booleana para manejar la situación de éxito o fracaso de la búsqueda.
En la sentencia while utilizamos el método next que pertenece al objeto ***rs***, es decir, es un método de la interfaz ResulSet. El método **next** devuelve verdadero si encuentra la primera fila de información, las siguientes veces se desplaza en cada registro almacenado en el ***rs***. La sentencia if que se encuentra dentro del while, su condición lógica se hará verdadero cuando encuentre el código de estadio buscado, haciendo que los cuadros de textos se muestren los demás datos, es entonces que la variable ***encuentra*** recién se hace verdadero.

```
private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt){

    String b;
    if(btnNuevo.isEnabled())
    {
        try
        {
                conn=Main.Enlace(conn);
                rs=Main.EnlEst(rs);
                b=txtidmarca.getText();
                boolean encuentra=false;
                while(rs.next())
                {
                    if(b.equals(rs.getString(1)))
                    {
                       txtpais.setText((String)rs.getString(2));
                       txtdescripcion.setText((String)rs.getString(3));
                       this.activaBotones(true, true, true, false);
                       encuentra=true;
                       break;

                    }
                }
                if(encuentra==false)
                {
                    txtidmarca.setText("No existe");
                    txtidmarca.requestFocus();
                }
    }
  catch(SQLException e)
  {
      JOptionPane.showMessageDialog(null,"Error BD"+e.getMessage());
  }
  }
  }
```

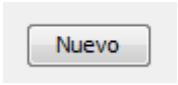




Luego en el botón: Ver listas de Marcas de comando btnVer, consiste en aumentar el tamaño del formulario para visualizar el objeto JTable. Posteriormente se establece la conexión con la base de datos y en la variable ***rs*** se almacena los datos provenientes de la tabla de Estadio. Se define un vector denominado datos de tamaño 4 elementos del tipo String que servirá colocar los datos de una fila para luego agregarlo al objeto ***dtm*** que está vinculado al objeto JTable llamado

tablamarca. Pero antes de agregarlo debemos asegurarnos que no exista fila alguna de datos en el modelo ***dtm*** y por ende en la ***tablamarca***. El bucle de la sentencia while permite colocar en cada elemento del arreglo los datos extraídos de una fila que almacena el objeto ***rs,*** esto es posible ya que el método getString, indicando la posición de la columna, podemos obtener el dato de la fila actual. Con el método addRow logramos crear una fila con los datos del vector ***datos*** en el objeto dtm y como está vinculado a la tablamarca entonces se podrá ver los registros agregados.

```
    private void btnVerActionPerformed(java.awt.event.ActionEvent evt) {
    try
{
    this.setSize(530,500);
    int f, i;
    conn=Main.Enlace(conn);
    rs=Main.EnlEst(rs);
    String datos[]=new String[4];
    f=dtm.getRowCount();
    if(f>0)
        for(i=0; i<f; i++)
            dtm.removeRow(0);
    while(rs.next())
    {
        datos[0]=(String)rs.getString(1);
        datos[1]=(String)rs.getString(2);
        datos[2]=(String)rs.getString(3);
        datos[3]=(String)rs.getString(4);
        dtm.addRow(datos);
    }
}
    catch(SQLException e)
       {
            JOptionPane.showMessageDialog(null,"Error BD"+e.toString());
       }
    }
```

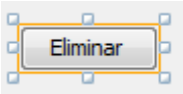


Para el botón de comando **btnNuevo**, limpiamos los cuadros de textos con el método limpiarObjetos. Se inhabilita el cuadro de texto **txtidmarca** y se envía el cursor al cuadro de texto **txtpais**. Se inhabilta los botones de comando a excepción de grabar ya que estamos en el momento de ingresar nuevos datos y proceder a almacenar.

```
    private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {
    limpiarObjetos();
txtidmarca.setEnabled(false);
txtdescripcion.requestFocus();
activaBotones(false,false,false,true);
    }
```

En el botón de comando **btnGrabar** se inicia visualizando un mensaje de confirmación para proceder a grabar, esto se logra usando el método **showConfirmDialog** de la clase JOptionPane. Si la respuesta es Sí entonces la sentencia if su condición lógica se hará verdadera y por lo tanto establecemos conexión con la base de datos transportes, pasamos los datos ingresados a variables como ***descrip*** y ***país***. En la variable comando establecemos la instrucción con el comando INSERT para luego usar el método ***executeUpdate*** quien procederá a grabar los datos. Posteriormente se procede a cerrar la conexión con el método **close** del objeto connection ***conn***.

```
    private void btnGrabarActionPerformed(java.awt.event.ActionEvent evt) {
   int resp;
resp=JOptionPane.showConfirmDialog(null,"¿Desea Grabar el registro?", "Pregunta",0);
if(resp==0)
{
    try
    {
        conn=Main.Enlace(conn);
        st=Main.sta(st);
        rs=Main.EnlEst(rs);
        String descrip, pais, comando;
        descrip=txtdescripcion.getText();
        pais=txtpais.getText();
        comando="INSERT INTO marca (descripción, país, Estado) VALUES";
        comando=comando+"('"+descrip+"','"+pais+"',1)";
        st.executeUpdate(comando);
        conn.close();
        activaBotones(true,false,false,false);
        limpiarObjetos();
        txtidmarca.setEnabled(true);
    }
    catch(SQLException e)
    {
        JOptionPane.showMessageDialog(null,"Error "+e.toString());
    }
}
    }
```



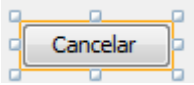
En el botón de comando **btnEliminar**, también se procede a través de un mensaje confirmar si procede la eliminación de los datos de Marcas. Si la respuesta es afirmativa se procede a conectarse a la base de datos y en la variable de memoria ***id*** se almacenada el código de la marca ingresado a través del cuadro de texto txtidmarca. Se construye la instrucción usando el comando UPDATE, luego ejecutamos la eliminación lógica haciendo que el campo estado sea igual a cero y se cierra la conexión con la base de datos.

```
    private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
   int resp;
resp=JOptionPane.showConfirmDialog(null,"¿Desea Eliminar el registro?", "Pregunta",0);
if(resp==0)
{
    try
    {
      conn=Main.Enlace(conn);
        st=Main.sta(st);
        rs=Main.EnlEst(rs);
        int cod;
        String comando;
        cod=Integer.parseInt(txtidmarca.getText());
        comando="UPDATE marca SET estado=0 where id_marca="+String.valueOf(cod);
        st.executeUpdate(comando);
        conn.close();
        activaBotones(true,false,false,false);
        limpiarObjetos();
    }
    catch(SQLException e)
    {
        JOptionPane.showMessageDialog(null,"Error "+e.toString());
    }
}
    }
```

En el botón de comando **btnModificar**, al igual que de grabar o eliminar se procede a confirmar a través de un mensaje si se procede a la modificación de datos. Una vez salvados los datos ingresados en los cuadros de textos en variables de memoria se prepara la instrucción en la variable de memoria ***comando***. Usamos el comando UPDATE para actualizar los datos. Se procede a

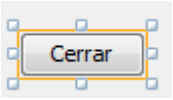
ejecutar el comando con el método **executeUpdate** y se cierra la conexión con el método **close**.

```
    private void btnModificarActionPerformed(java.awt.event.ActionEvent evt) {
   int resp;
resp=JOptionPane.showConfirmDialog(null,"¿Desea Modificar los datos?", "Pregunta",0);
if(resp==0)
{
    try
    {
        conn=Main.Enlace(conn);
        st=Main.sta(st);
        rs=Main.EnlEst(rs);
        String descrip, pais, comando;
        int cod;
        cod=Integer.parseInt(txtidmarca.getText());
        descrip=txtdescripcion.getText();
        pais=txtpais.getText();
        comando="UPDATE marca SET descripción='"+descrip+"', país='"+pais+"',";
        comando=comando+" where id_marca="+String.valueOf(cod);
        st.executeUpdate(comando);
        conn.close();
        activaBotones(true,false,false,false);
        limpiarObjetos();
    }
    catch(SQLException e)
    {
        JOptionPane.showMessageDialog(null,"Error "+e.toString());
    }
    }
    }
```



En el botón de comando Cancelar, luego de dar respuesta afirmativa se procede a limpiar los cuadros de textos, habilita el cuadro de texto txtidmarca para su uso y se vuelve a su estado inicial.

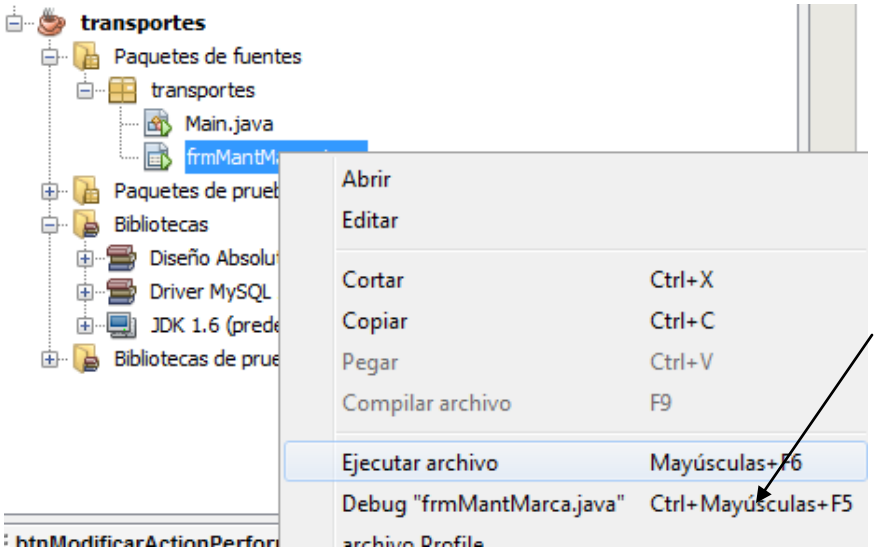
```
        private void btnCancelarActionPerformed(java.awt.event.ActionEvent evt) {
int resp;
resp=JOptionPane.showConfirmDialog(null,"Desea Cancelar el proceso?", "Pregunta",0);
if(resp==0)
{
        limpiarObjetos();
        txtidmarca.setEnabled(true);
        activaBotones(true,false,false,false);
}
        }
```



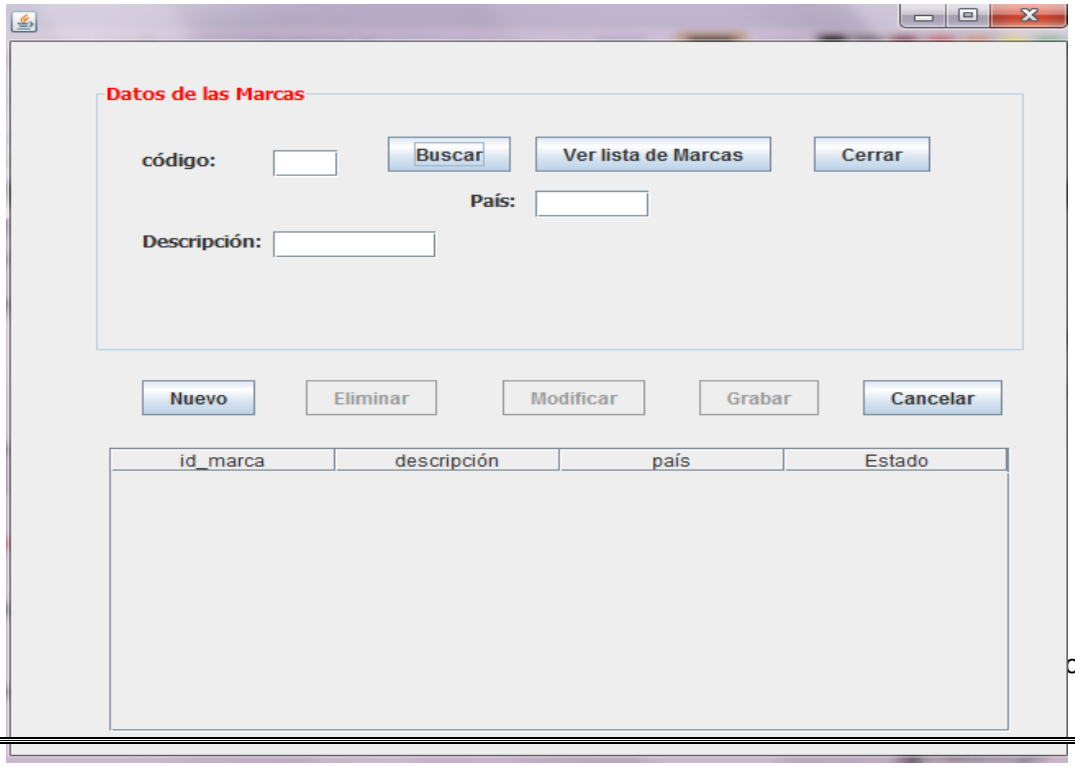
El botón de comando **btnCerrar**, con el método dispose () se cierra el Formulario.

```
        private void btnCerrarActionPerformed(java.awt.event.ActionEvent evt) {
dispose();
        }
```

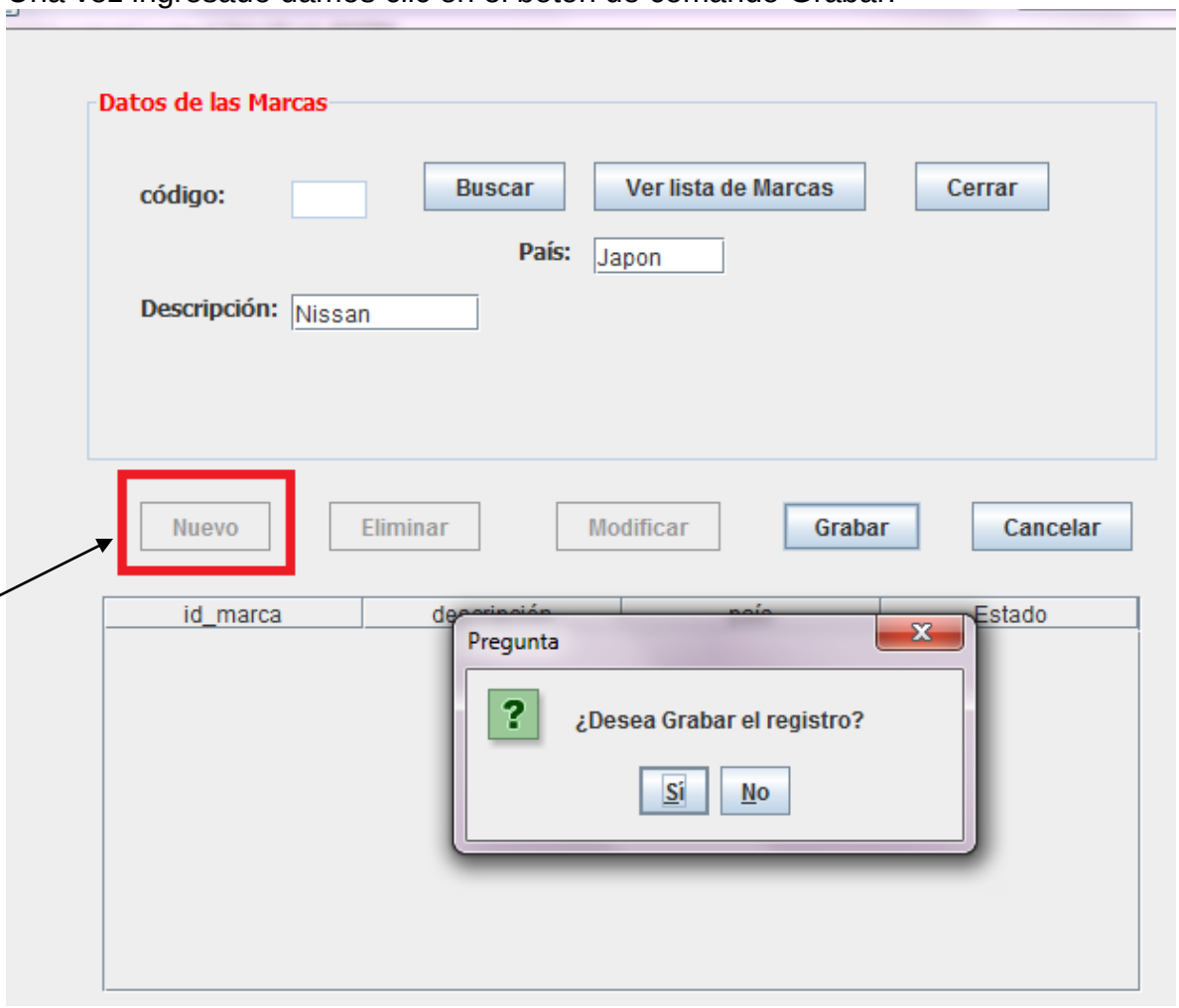
Procedemos a ejecutar el formulario, seleccionado Ejecutar archivo.



Observamos el formulario ejecutado.

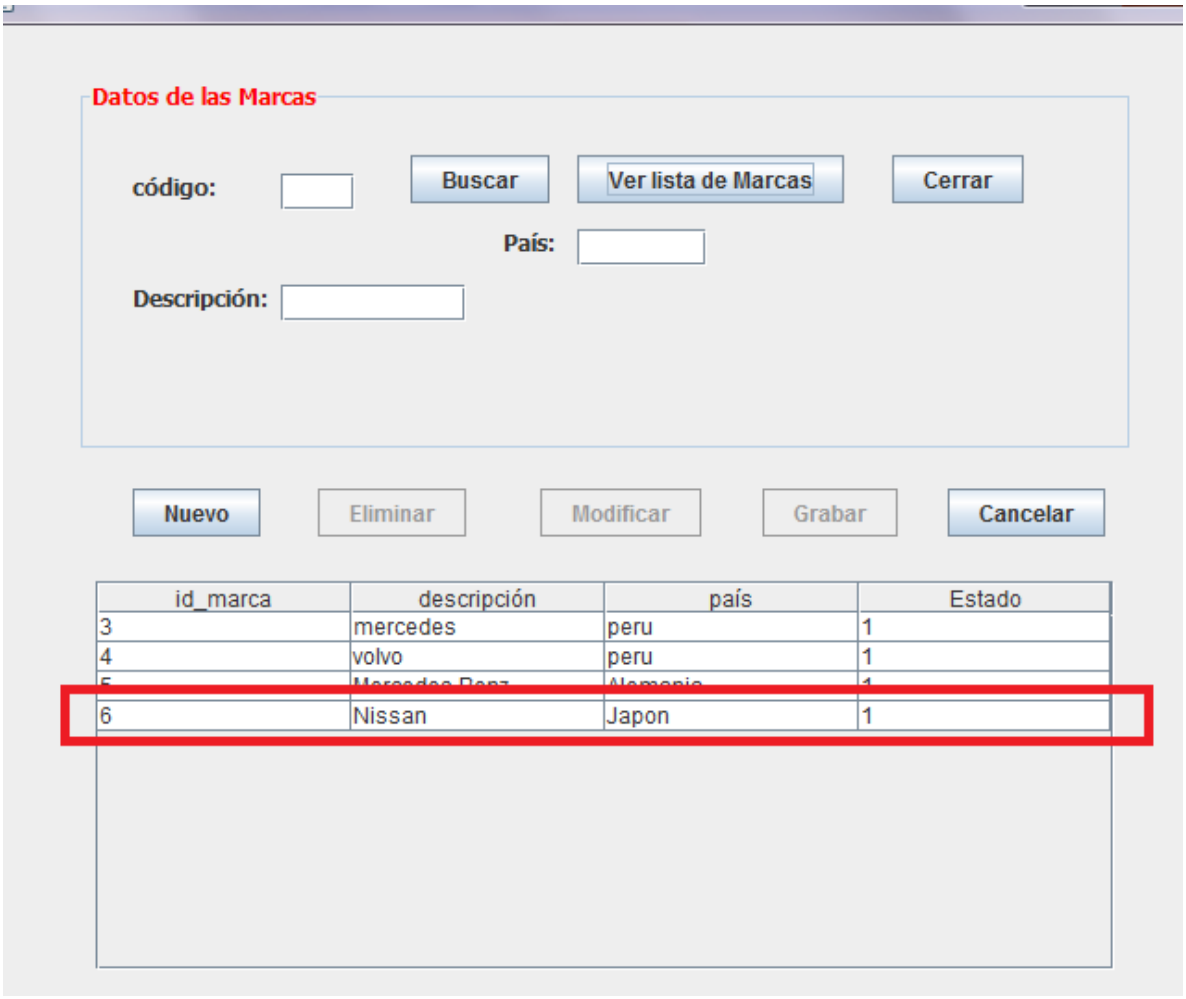


Al dar clic en el botón de comando Nuevo podemos proceder a ingresar datos. Una vez ingresado damos clic en el botón de comando Grabar.



Al dar clic en Nuevo, el cursor nos re direccionara automáticamente al campo de texto txtdescripcion por que así fue especificado en el código: **requestFocus()**

Luego de grabar, al dar clic en el botón de comando Ver Lista de Estadios, el Formulario se mostrará de la siguiente forma.



Nota: Si gustan pueden realizar las demás funciones, como Buscar por código poder Modificar o Eliminar un registro, y si no quieren realizar esa función pues solo seleccionar en cancelar, automáticamente se cancelara la operación.