Prácticas Ingeniería del Software 3º

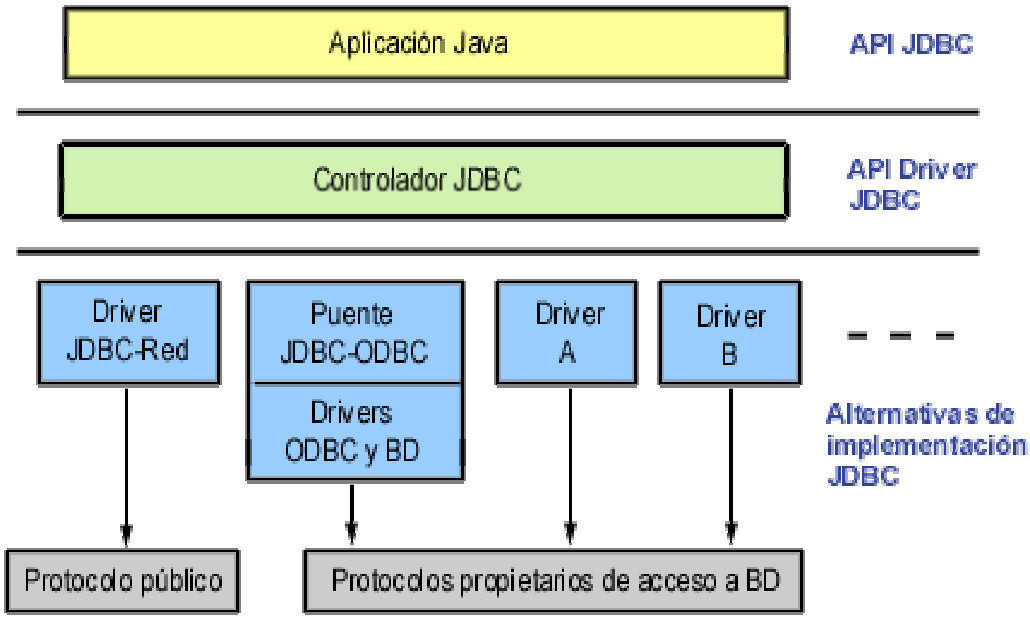
JDBC

JAVA con Bases de Datos

UNIVERSIDAD DE
CASTILLA-LA MANCHA
ES de Informática de Ciudad Real

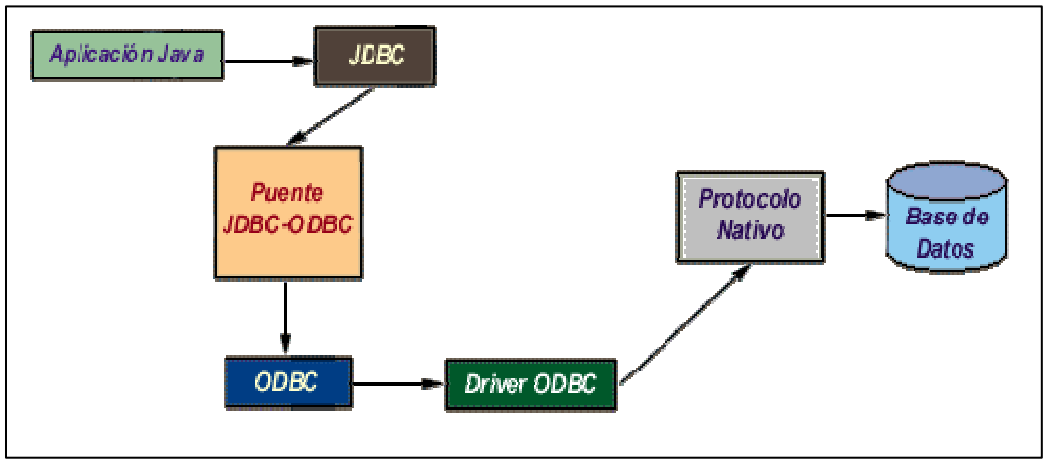
Introducción

- JDBC es una especificación de un **conjunto de clases** y **métodos de operación** que permiten a cualquier programa **Java acceder** a **sistemas de bases de datos** de forma **homogénea**

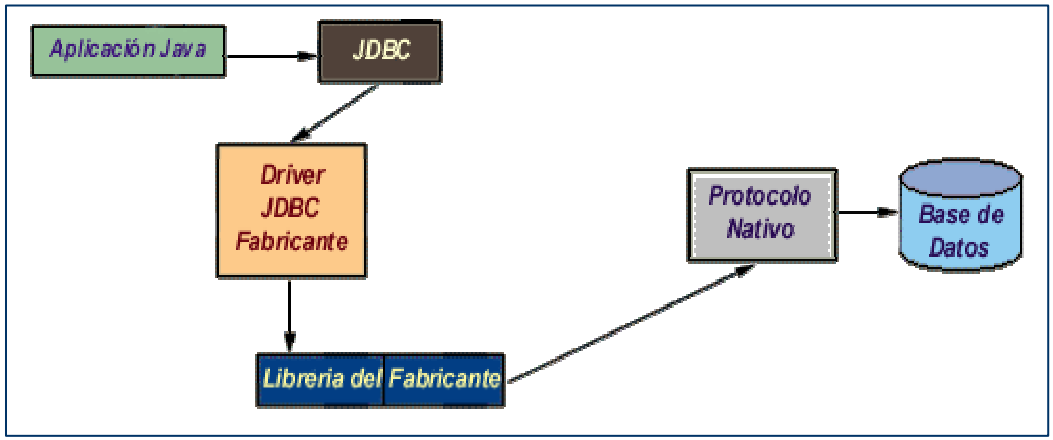


Drivers JDBC

- ***Puente JDBC-ODBC***
  - Facilidad Configuración
  - Más lento

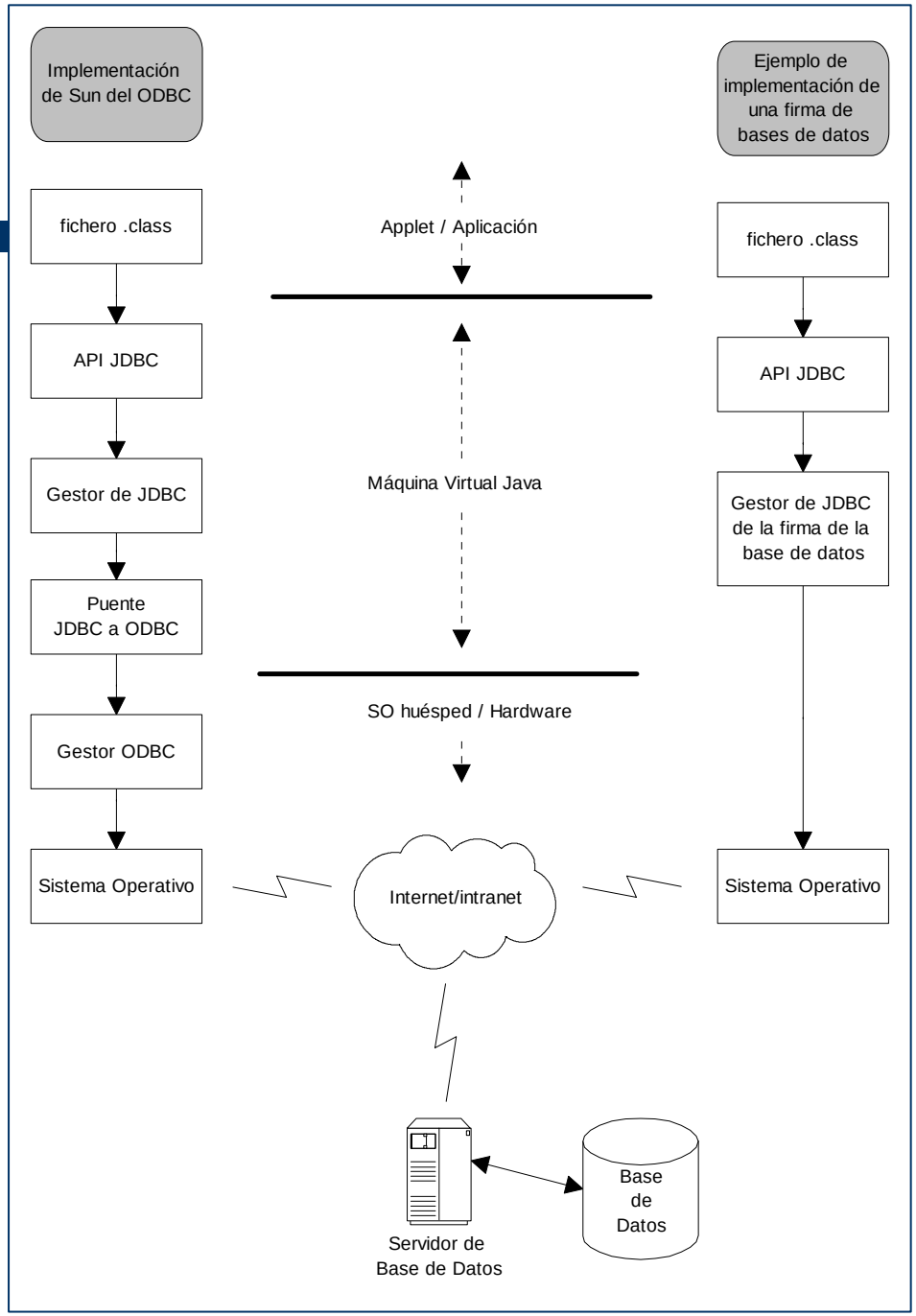


- ***Java Binario***
  - No hay capa ODBC
  - Necesario *Driver* Fabricante en el cliente



- ***Otros:***
  - ✓ 100% JAVA Protocolo Nativo
  - ✓ 100% JAVA Protocolo Independiente

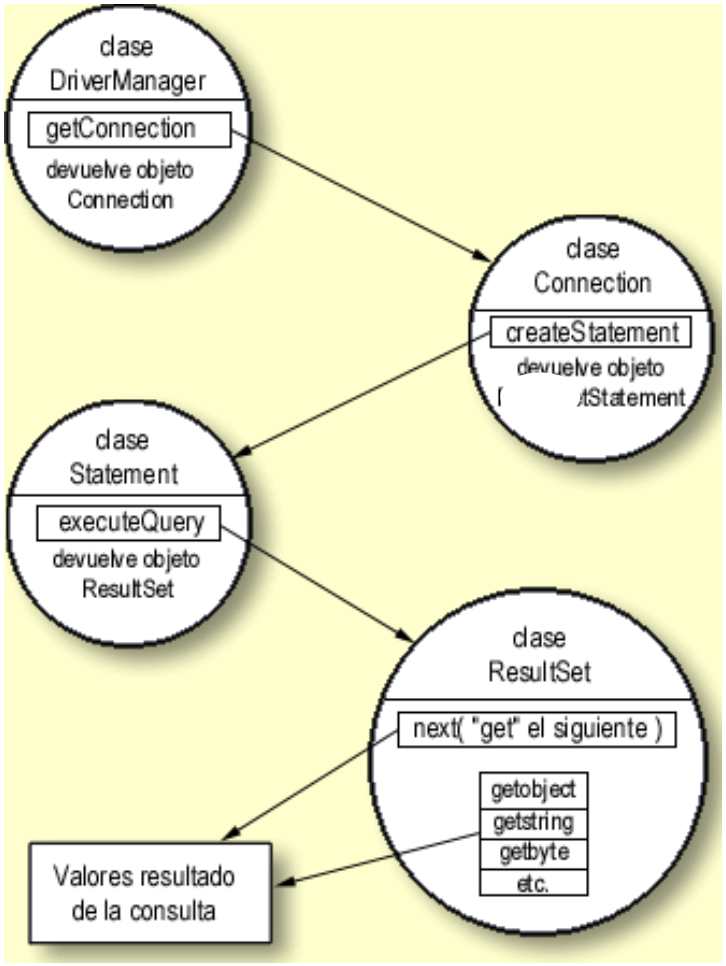
Comunicación con la BBDD



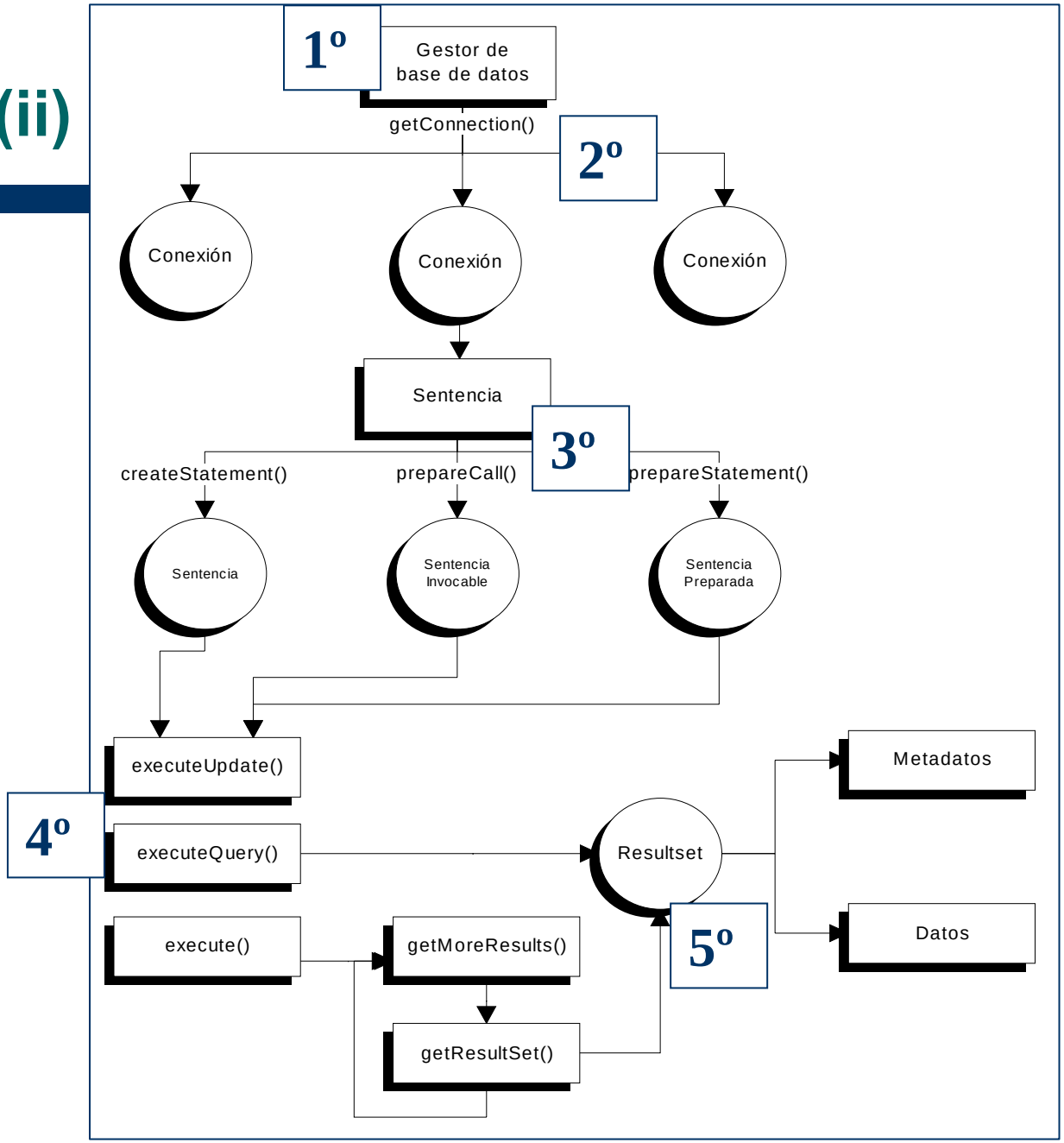
Proceso (i)

1. Establecer Driver
2. Establecer conexión
3. Crear sentencia
4. Ejecutar sentencia
5. Procesar resultados
6. Finalizar sentencia
7. Cerrar conexión

En JAVA



Proceso (ii)

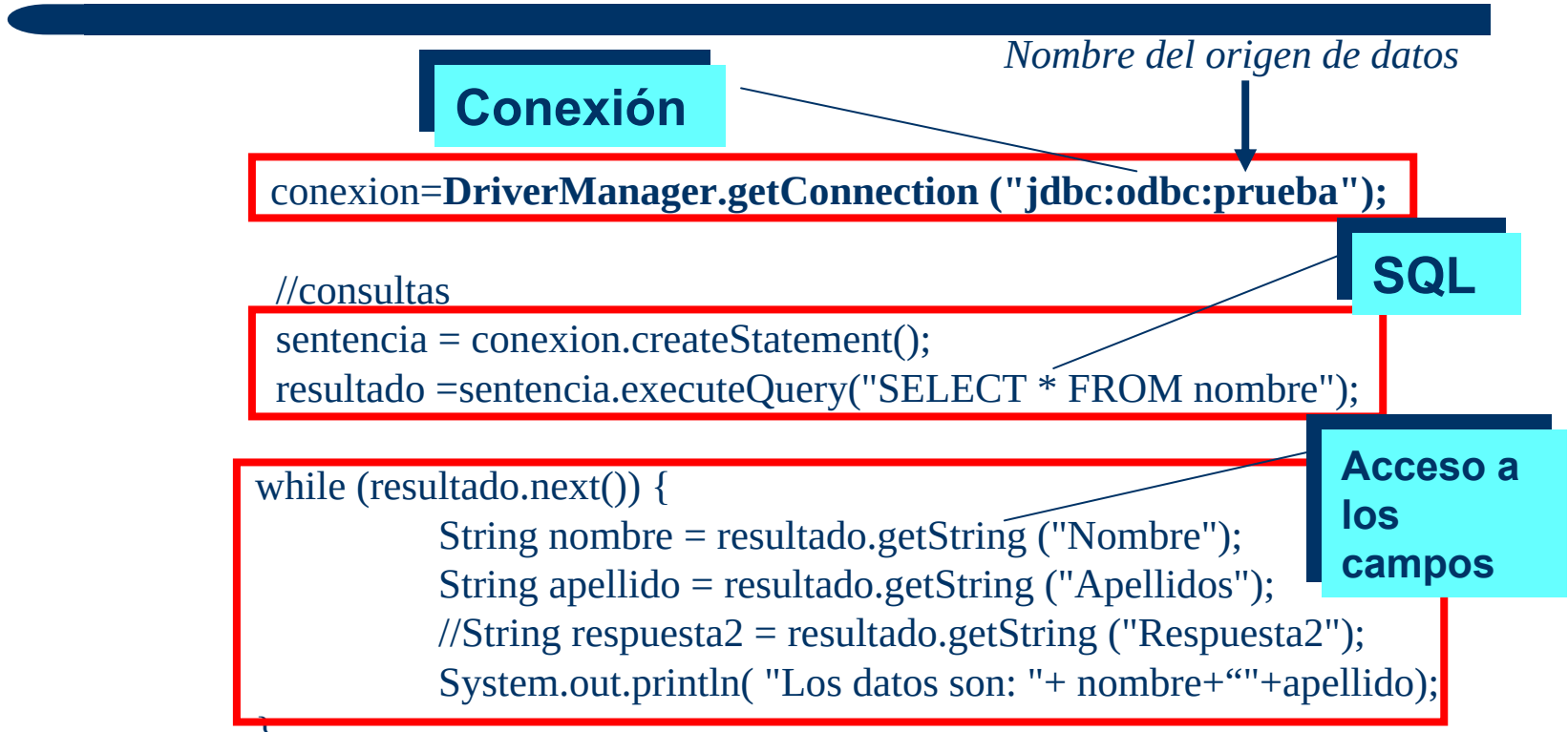

1º
Gestor de base de datos
getConnection()
2º
Conexión
Conexión
Conexión
Sentencia
3º
createStatement()
prepareCall()
prepareStatement()
Sentencia
Sentencia Invocable
Sentencia Preparada
4º
executeUpdate()
executeQuery()
execute()
Resultset
5º
Metadatos
Datos
getMoreResults()
getResultSet()

Ejemplo (i)

```
import java.sql.*;

class Ejercicio{
        static public void main (String [] args){
                Connection conexion;
                Statement  sentencia;
                ResultSet  resultado;
                System.out.println("Iniciando programa");
                //Se carga el driver JDBC-ODBC
                try{
                Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
                }
        catch (Exception e) {
          System.out.println("No se pudo cargar el puente JDBC-ODBC");
          return;
        }
        }
}
```

Driver

Ejemplo (ii)


Nombre del origen de datos
Conexión
conexion=DriverManager.getConnection ("jdbc:odbc:prueba");
SQL
//consultas
sentencia = conexion.createStatement();
resultado =sentencia.executeQuery("SELECT * FROM nombre");
Acceso a los campos
while (resultado.next()) {
String nombre = resultado.getString ("Nombre");
String apellido = resultado.getString ("Apellidos");
//String respuesta2 = resultado.getString ("Respuesta2");
System.out.println("Los datos son: "+ nombre+“"+apellido);
}
}
catch (Exception e){
System.out.println (e);
return;
}
System.out.println ("fin");
}}

Ejemplo Drivers

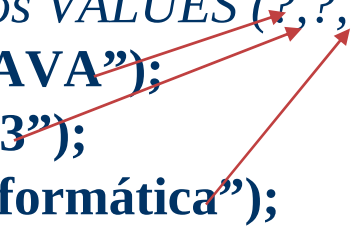
```
public static void conectar(String tipoConexion, String servidor) throws Exception {
 case 0: // JDBC -> ODBC
   driver="sun.jdbc.odbc.JdbcOdbcDriver";
   url="jdbc:odbc:domosim";
   txt="JDBC - ODBC";
   break;

 case 1: // JDBC -> INTERBASE
   driver="interbase.interclient.Driver";
   url="jdbc:interbase://"+maq+"/c:/domosim/data/domosim.gdb";
   txt="JDBC - InterBase";
   break;

 case 2: // JDBC -> MYSQL
   driver="org.gjt.mm.mysql.Driver";
   url="jdbc:mysql://"+maq+"/domosim";
   txt="JDBC - MySQL";
   break;
 }
Class.forName(driver).newInstance();
con=DriverManager.getConnection(url,user,clave);
```

Ejemplo PreparedStatement
Consulta de datos con condición

```
import java.sql.*;
final String DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
final String BBDD = "jdbc:odbc:ARTICULOS";
try {
   Class.forName(DRIVER);
   Connection conexion = DriverManager.getConnection(BBDD);
   PreparedStatement sentencia = conexion.prepareStatement(
      "SELECT * FROM articulos WHERE Titulo=?");
   sentencia.setString(1, "La boda del Príncipe");
   ResultSet resultado = sentencia.executeQuery();
   while (resultado.next()) {
            lista.addItem(resultado.getString("Titulo"));
   }
   resultado.close(); sentencia.close(); conexion.close();
} catch (Exception e) {
   System.out.println("Error: " + e); }
```

Ejemplo PreparedStatement
Inserción de datos

```
import java.sql.*;
final String DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
final String BBDD = "jdbc:odbc:ARTICULOS";
try {
   Class.forName(DRIVER);
   Connection conexion = DriverManager.getConnection(BBDD);
   PreparedStatement insercion = conexion.prepareStatement(
      "INSERT INTO articulos VALUES (?,?,?)");
   insercion.setString(1," JAVA");
   insercion.setString(2,"123");
   insercion.setString(3,"Informática");
   int resultado = insercion.executeUpdate();
   insercion.close(); conexion.close();
} catch (Exception e) {
   System.out.println("Error: " + e);
}
```



El estándar JDBC
Tipos de Clases

TIPO	Clase JDBC
Implementación	**java.sql.Driver** **java.sql.DriverManager** **java.sql.DriverPropertyInfo**
Conexión a base de datos	**java.sql.Connection**
Sentencias SQL	**java.sql.Statement** **java.sql.PreparedStatement** **java.sql.CallableStatement**
Datos	**java.sql.ResulSet**
Errores	**java.sql.SQLException** **java.sql.SQLWarning**

Clases de JDBC
java.sql.DriverManager

- Lleva el control de los gestores JDBC disponibles
  - Es posible que existan varios dentro del sistema
  - Por defecto, carga todos los disponibles en *sql.drivers*
  - El gestor cargado debería registrarse con el método *registerDriver*
- Sintaxis utilizada: URL's
  - jdbc:<subprotocolo>:<parámetros>
  - jdbc:odbc:NOTICIAS:UID=Sistema;PWD=SistemaPW
- Seguridad
  - Hay que tener presente el modelo de seguridad

```
final String BBDD = "jdbc:odbc:ARTICULOS";
Connection conexion =
     DriverManager.getConnection(BBDD);
```

Clases de JDBC
java.sql.Driver

- Gestor de información y configuración general
- Se carga durante la inicialización
  - mediante
    - DriverManager.registerDriver
    - Class.forName
- Se le pedirá información a lo largo del programa
- Residirá en memoria
- Métodos:
  - connect
  - getPropertyInfo

```
final String DRIVER ="sun.jdbc.odbc.JdbcOdbcDriver";
Class.forName(DRIVER);
```

Clases de JDBC
java.sql.Connection

- Puntero a la base de datos
- Proporciona el contexto de trabajo para los objetos Statement y ResultSet
- Soporta propiedades de transacción
  - setAutoCommit
  - commit
  - rollback

```
Connection conexion =
    DriverManager.getConnection(BBDD);
```

Clases de JDBC
java.sql.Statement

- Ejecución de una sentencia SQL
  - executeQuery
    - Sentencias SELECT
    - devuelve un ResultSet
  - executeUpdate
    - Sentencias INSERT, DELETE, UPDATE, CREATE
    - Devuelve un entero
  - execute
    - Sentencias desconocidas en tiempo de compilación o sentencias que devuelven resultados complejos
    - *Devuelve true/false*

```
Statement select = conexion.createStatement();

ResultSet resultado =

      select.executeQuery("SELECT * FROM ACTIVIDAD");
```

Clases de JDBC
java.sql.PreparedStatement

- Extiende Statement para añadir sentencias precompiladas SQL
  - Compila la sentencia SQL la primera vez
  - Sentencias que son llamadas más de una vez en el programa

- Soporta parámetros de entrada
  - setInt, setFloat, setLong, setString

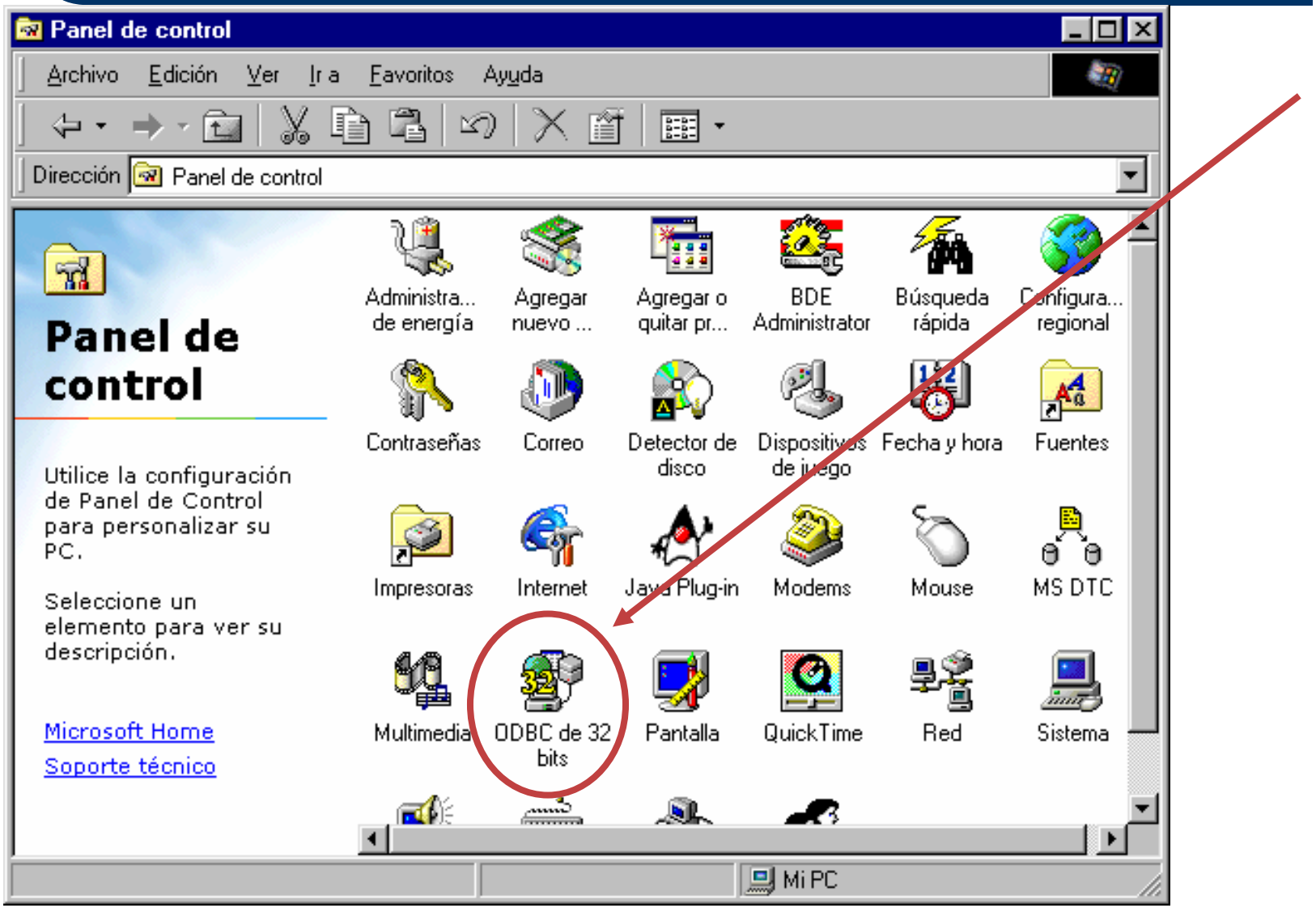
```
PreparedStatement select = conexion.prepareStatement(
        "SELECT * FROM articulos WHERE Titulo=?");
select.setString(1,”Mi titulo”);
ResultSet resultado = select.executeQuery();
```

Clases de JDBC
java.sql.ResultSet

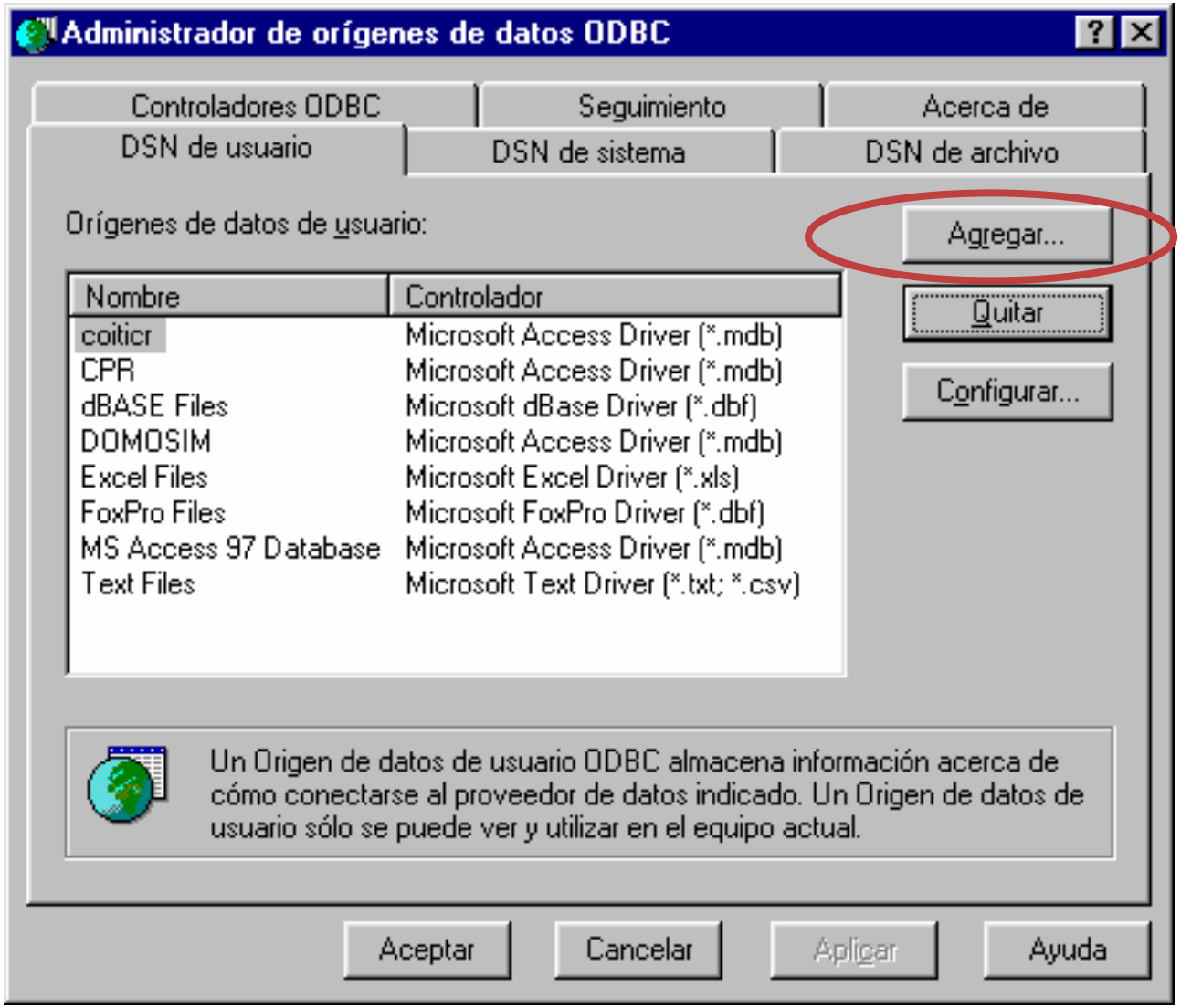
- Contiene los datos resultado de una sentencia SQL
- Se recuperan secuencialmente en filas
  - next sirve para avanzar una fila
- Se puede acceder a los datos de las columnas en cualquier orden
  - índice de posición
  - nombre del campo
  - Métodos: getString, getFloat, getInt, etc.
  - Método wasNull()

```
PreparedStatement select = conexion.prepareStatement(
        "SELECT * FROM articulos WHERE Titulo=?");
select.setString(1,"Mi titulo");
ResultSet resultado = select.executeQuery();
while (resultado.next())
      System.out.println(resultado.getString("Titulo"));
```

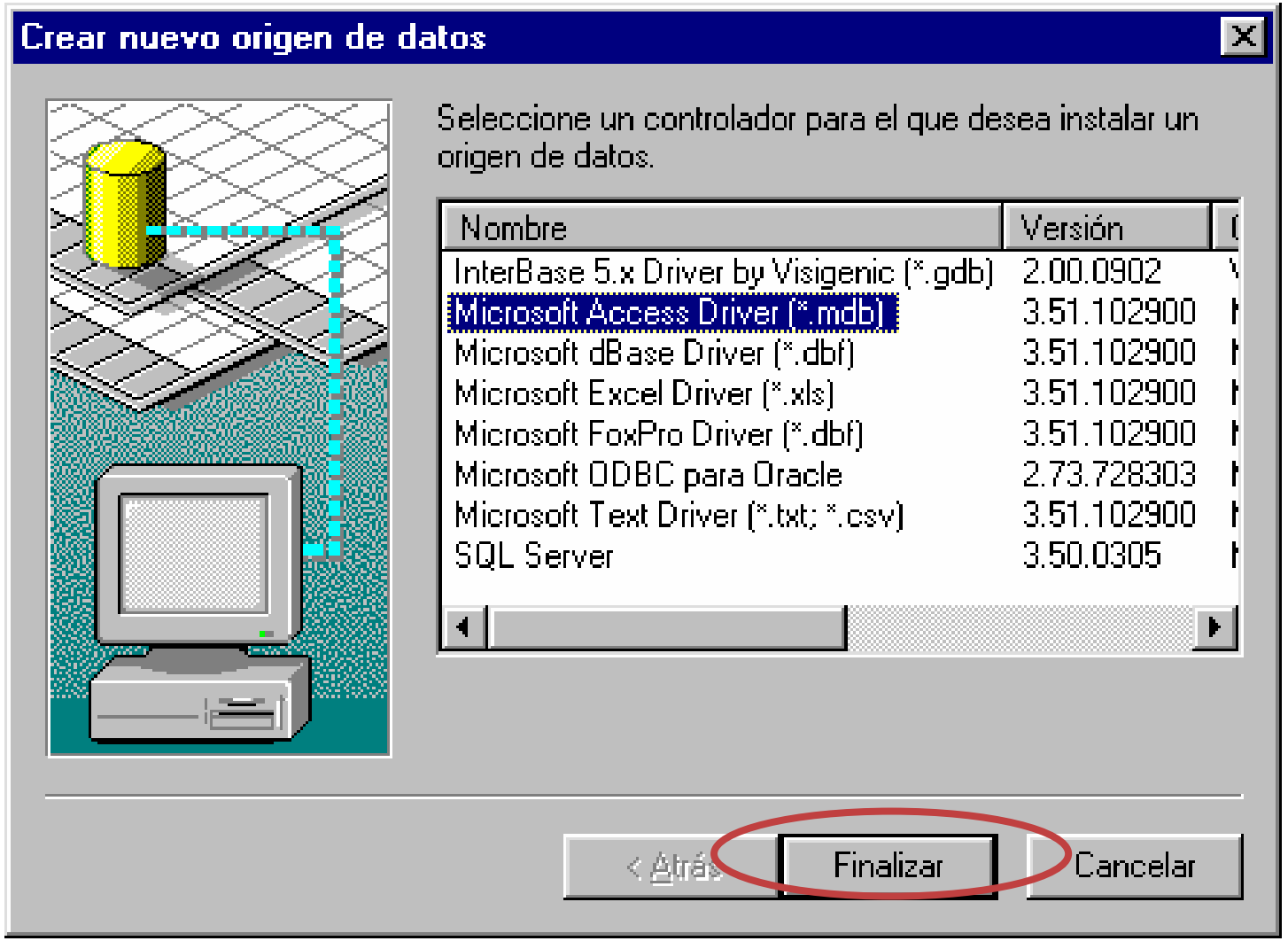
El estándar JDBC
Configuración JDBC:ODBC



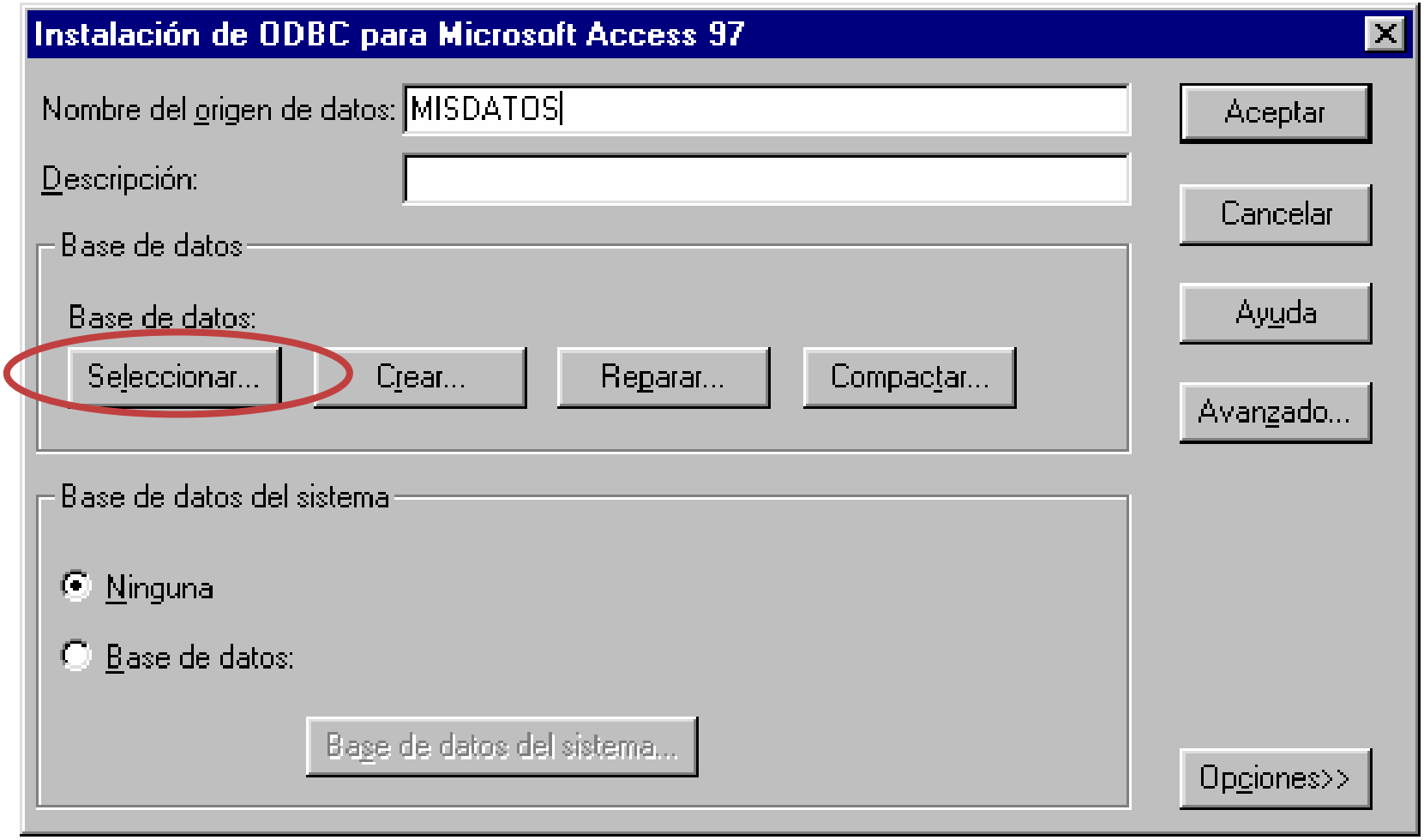
El estándar JDBC
Configuración JDBC:ODBC



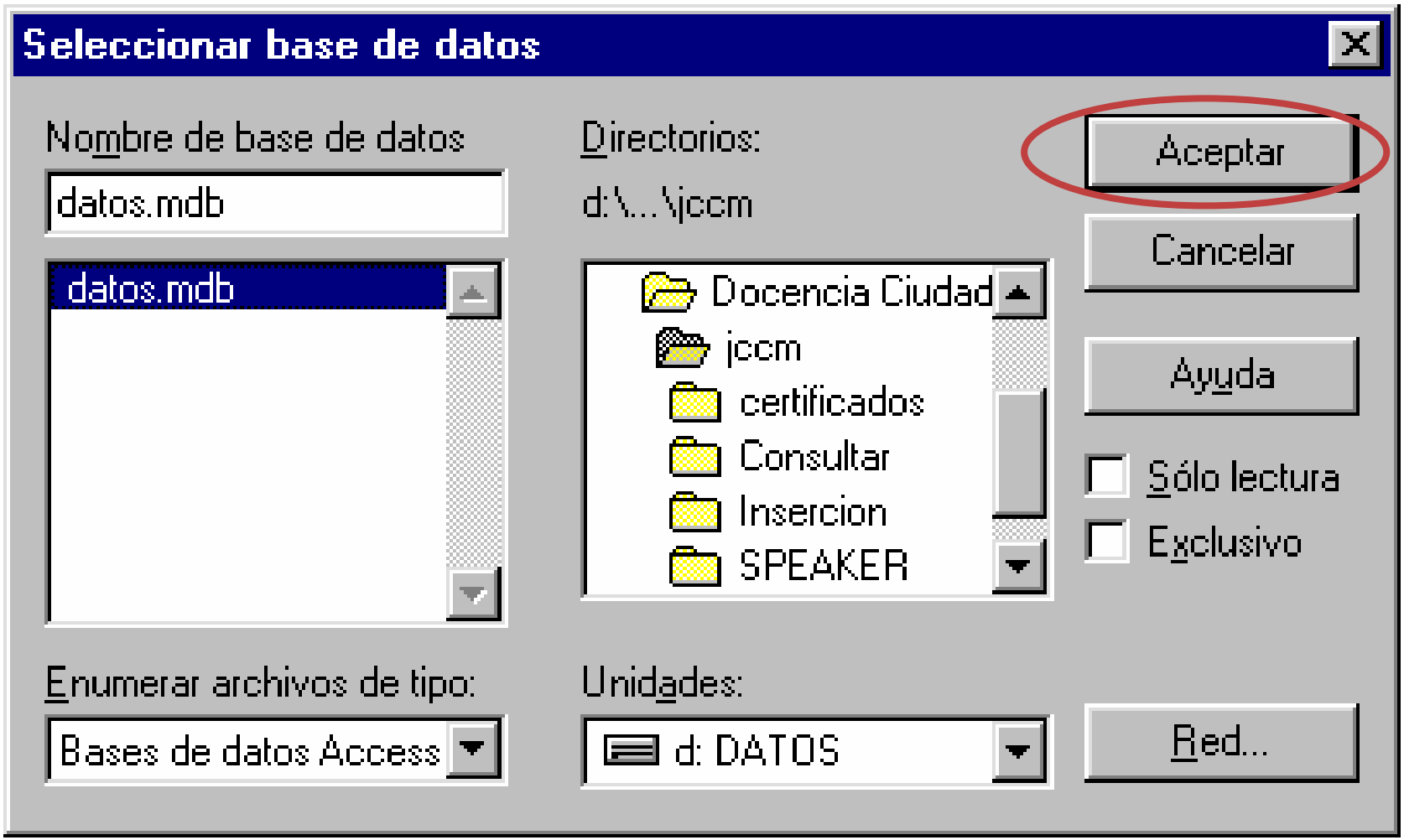
El estándar JDBC
Configuración JDBC:ODBC



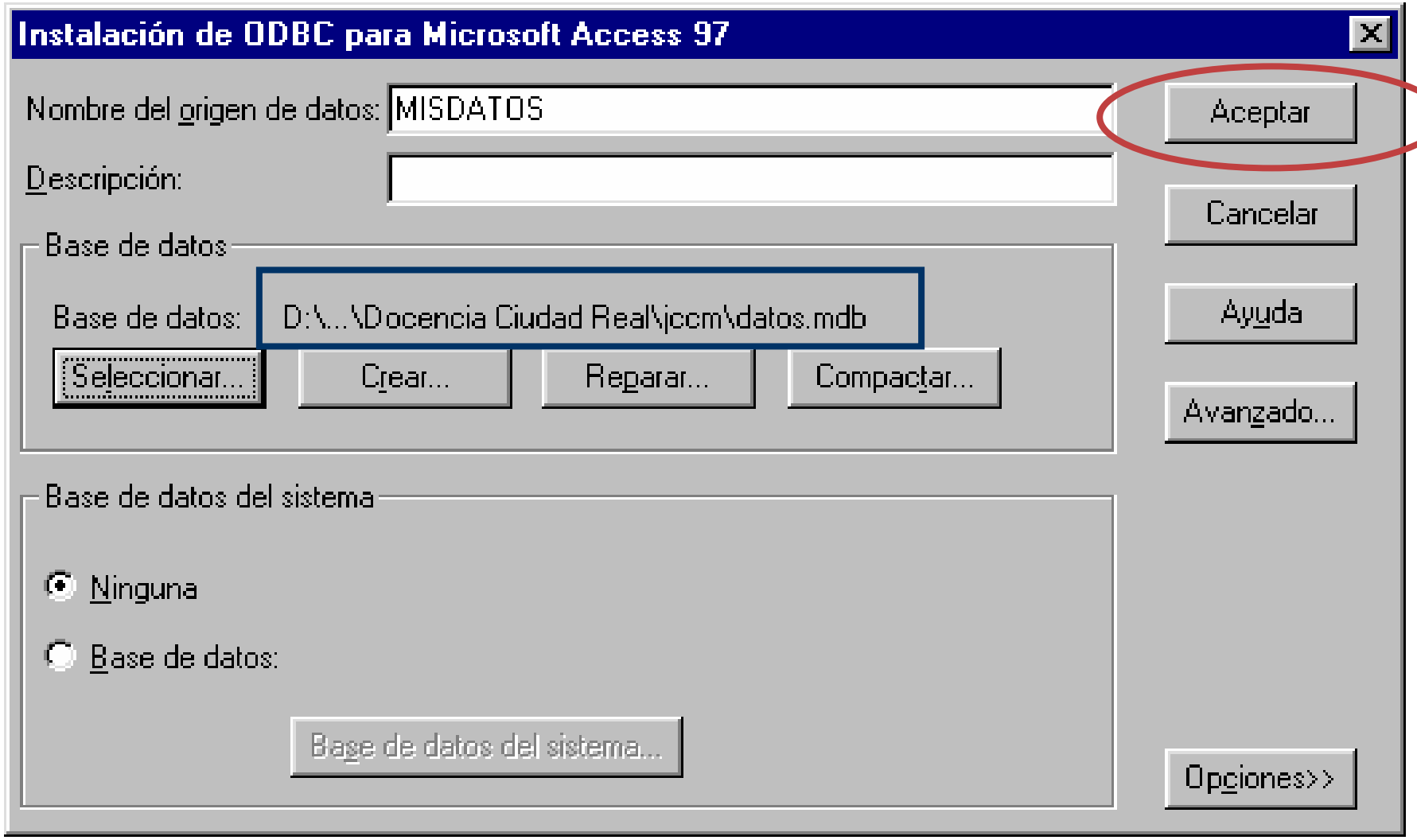
El estándar JDBC
Configuración JDBC:ODBC



El estándar JDBC
Configuración JDBC:ODBC



El estándar JDBC
Configuración JDBC:ODBC



El estándar JDBC
Configuración JDBC:ODBC

Ejercicio

Crear en Access una base de datos con una tabla llamada Alumnos con tres campos: Nombre, Apellido y DNI. El último campo es de tipo numérico, el resto de tipo cadena. Configure en el ODBC la base de datos. A continuación implementar un programa Java que se conecte a la base de datos e inserte 3 filas en la tabla alumnos. Posteriormente, mostrar por pantalla el contenido de la tabla. Para terminar realizar una modificación del nombre que coincida con un determinado DNI. Se debe utilizar la clase *PreparedStatement* para realizar la modificación.

Solución (1/3)

```
import java.sql.*;
import java.io.*;
class Alumnos{
    static public void main (String [] args){
      Connection conexion;
      Statement  sentencia;
      ResultSet  resultado;
      BufferedReader leer= new BufferedReader (new
            InputStreamReader (System.in));
       System.out.println("Iniciando programa");
//Se carga el drvier JDBC-ODBC
       try{
        Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
        } catch (Exception e) {
        System.out.println("No se pudo cargar el puente JDBC-ODBC");
         return;}
```

Solución (2/3)

```
try{
 //Se establece la conexión con la base de datos
conexion=DriverManager.getConnection ("jdbc:odbc:prueba");
//consultas
  sentencia = conexion.createStatement();
 //anadir tres tuplas a la base de datos

 sentencia.execute("INSERT INTO Alumnos VALUES  (1,'Aurora',’Vizcaíno’,591)");
 sentencia.execute("INSERT INTO Alumnos VALUES  (2,‘Ismael',’Caballero',593)");
 sentencia.execute("INSERT INTO Alumnos VALUES  (3,'Juan','Paz',533)");

resultado =sentencia.executeQuery("SELECT * FROM Alumnos");
 while (resultado.next()) {
    String nombre = resultado.getString ("Nombre");
    String apellido = resultado.getString ("Apellido");
    int dni = resultado.getInt ("DNI");
    System.out.println( "Los datos son: "+ nombre+ " "+dni);
}
```

Solución (3/3)

```
//Modificar un nombre por el que lee por teclado, usar ?
PreparedStatement sentencia2=conexion.prepareStatement
            ("UPDATE Alumnos SET Nombre = ? WHERE DNI=591");
sentencia2.setString(1, leer.readLine());
//Se ejecuta la sentencia
sentencia2.execute();
resultado =sentencia.executeQuery("SELECT * FROM Alumnos");
 while (resultado.next()) {
            String nombre = resultado.getString ("Nombre");
            String apellido = resultado.getString ("Apellido");
            int dni = resultado.getInt ("DNI");
            System.out.println( "Los datos son: "+ nombre+ " "+dni);
}
sentencia.close(); conexion.close();
}
 catch (Exception e){  System.out.println (e);return;            }
System.out.println ("fin");
}
```

Otro Problema

- A partir de la experiencia del problema anterior, cree una clase con los atributos para los datos de los *drivers* y las conexiones y métodos necesarios para hacer consultas parametrizadas de selección, inserción, actualización y borrado de tal forma que los parámetros se pasen como argumentos, gestionando adecuadamente las excepciones. Dicha clase **no** debe escribir nada a la línea de comandos. Probar con el DSN creado para el problema anterior. Utilizar el patrón Agente.