

EJERCICIO GUIADO. JAVA. ACCESO A BASE DE DATOS

Acceso a Base de Datos desde una aplicación Java

El objetivo de todas las hojas guiadas anteriores dedicadas a las bases de datos, es finalmente aprender a crear un fichero que contenga toda la información que la empresa necesita gestionar. Es decir, crear un fichero de base de datos.

Este fichero se incluirá con el programa java que se realice. Nuestro programa java accederá a este fichero continuamente para añadir nuevos datos, o modificar datos, eliminar datos, o extraer datos según lo ordene el usuario del programa.

En esta hoja guiada, se verán los pasos necesarios para conectar un proyecto java con un fichero de base de datos creado en Access.

Pasos Generales para preparar una Aplicación Java para acceder a una Base de Datos

Para preparar nuestra aplicación Java para que pueda acceder a una Base de Datos, es necesario realizar tres pasos:

1. Cargar el controlador de la base de datos.

El *controlador* define el tipo de base de datos que se va a usar (base de datos de Access, o de MySQL, o de cualquier otro gestor de base de datos)

En nuestro caso, tendremos que indicar el controlador para base de datos de Access.

2. Crear un objeto conexión (*Connection*)

Para crear este objeto hay que indicar la situación del fichero de base de datos, el usuario y la contraseña de dicha base de datos. El objeto conexión abre el fichero de la base de datos.

3. Crear un objeto sentencia (*Statement*)

El objeto sentencia se crea a partir del objeto conexión anterior. Los objetos sentencia permiten realizar acciones sobre la base de datos usando instrucciones SQL.

Es decir, a través del objeto sentencia introduciremos datos en la base de datos, eliminaremos datos, haremos modificaciones, y extraeremos datos de la base de datos.

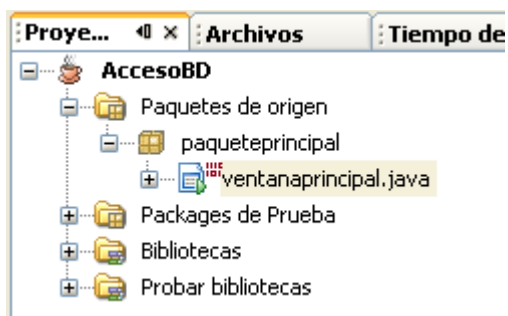
Así pues, este objeto es vital. Este objeto es el que realmente permite el acceso a los datos de la base de datos y la manipulación de dicha base de datos.

EJERCICIO GUIADO Nº 1

PLANTEAMIENTO

En este ejercicio se pretende crear una pequeña aplicación de bases de datos que permita simplemente mostrar los datos de los trabajadores almacenados en la base de datos MANEMPSA.

1. Entre en NetBeans. Crea un nuevo proyecto llamado *AccesoBD*. Dentro de este proyecto crea un paquete principal llamado *paquetepincipal* y dentro de él un JFrame llamado *ventanaprincipal*:



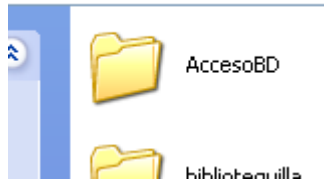
2. En la parte superior de la ventana añade un botón con el texto *Ver Datos Trabajadores* que se llame *btnVerDatos*.



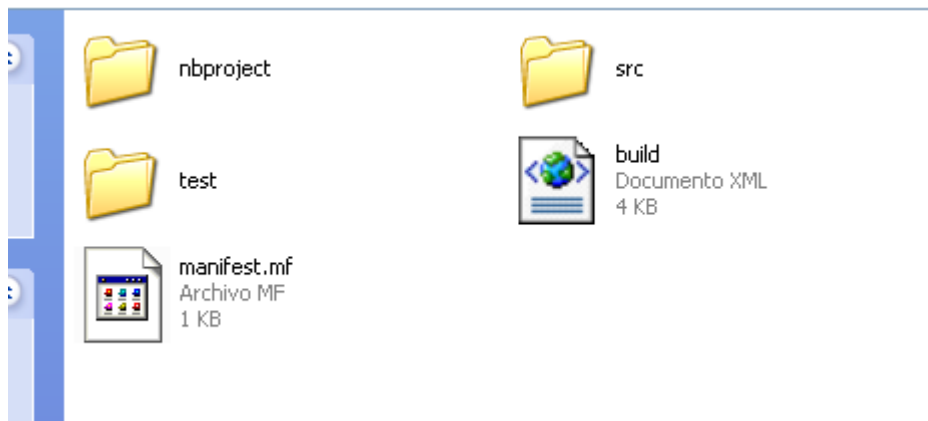
3. Se pretende simplemente que al pulsar el botón *btnVerDatos* aparezcan en un JOptionPane datos sobre los trabajadores almacenados en la base de datos.

SITUACIÓN DEL FICHERO DE BASE DE DATOS

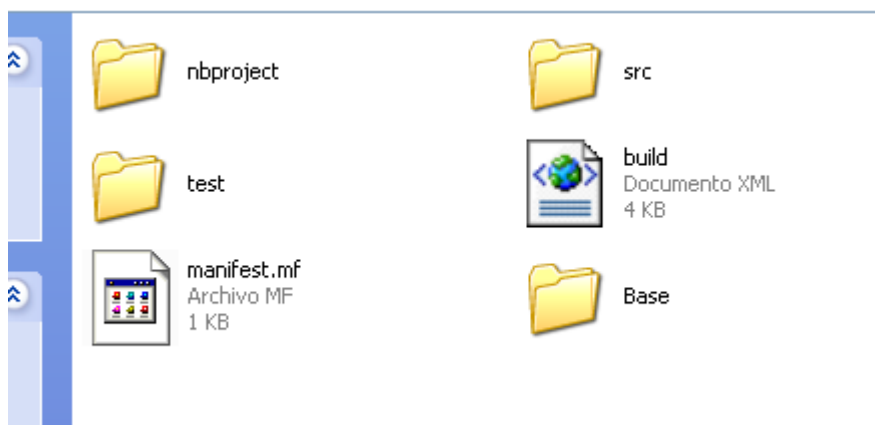
4. Antes de empezar con la programación de nuestra aplicación, introduciremos el fichero de base de datos en la carpeta del proyecto java que estamos haciendo. Para ello, accede a la carpeta del proyecto *AccesoBD*. Esta carpeta se encontrará allá donde estés guardando tus proyectos:



5. Entra dentro de esta carpeta y observarás una serie de carpetas creadas por el propio NetBeans. (Entre ellas, la más interesante es la llamada *src*, la cual es la que contiene los ficheros con el código java)



6. Para tener todo bien organizado, crearemos una carpeta llamada *Base* dentro de la carpeta de proyecto, de manera que su contenido quede así:



7. Ahora lo único que tiene que hacer es copiar el fichero de base de datos *MANEMPSA.MDB* que se encuentra en Mis Documentos dentro de la carpeta *Base*. (La extensión de los ficheros de base de datos de Access es *MDB*)

8. De esta manera, tenemos el fichero de base de datos que hemos creado con Access guardado dentro de la misma carpeta de proyecto java que usará dicha base de datos. Se recomienda que esto lo haga cada vez que programe una aplicación de bases de datos Java.
9. Ahora ya podemos volver al NetBeans y continuar con nuestro trabajo.

PREPARACIÓN DE LA APLICACIÓN JAVA PARA EL ACCESO A LA BASE DE DATOS

10. Para poder acceder y manipular una base de datos, es necesario tener dos objetos:

- Un objeto del tipo *Connection*, al que llamaremos *conexion*. Este objeto define la conexión con la base de datos.
- Un objeto del tipo *Statement*, al que llamaremos *sentencia*. Este objeto permite manipular la base de datos.

11. Así pues, lo primero que haremos será definir estos objetos como globales en la clase de la ventana principal, para así poderlos usar desde cualquier lado:

```
L */
public class ventanaprincipal extends javax.swing.JFrame {

    Connection conexion;
    Statement sentencia;

    /** Creates new form ventanaprincipal */
    public ventanaprincipal() {
        initComponents();
    }
}
```

Aparecerán subrayados ya que será necesario indicar el import para las clases *Connection* y *Statement*. Estos import son respectivamente:

```
java.sql.Connection
```

y

```
java.sql.Statement
```

Agrega los import correspondientes para eliminar los errores.

12. Una vez definidos los objetos *conexión* y *sentencia*, necesarios para el acceso a la base de datos, prepararemos nuestro programa para que pueda acceder a la base de datos MANEMPSA.MDB. Esto se hará en el constructor.

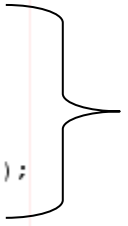
13. En primer lugar, añade al constructor una llamada a una función *PrepararBaseDatos*:

```
/** Creates new form ventanaprincipal */
public ventanaprincipal() {
    initComponents();
    PrepararBaseDatos(); ←
}
```

14. Crea el método *PrepararBaseDatos* debajo del constructor y empieza a programar lo siguiente:

```
/** Creates new form ventanaprincipal */
public ventanaprincipal() {
    initComponents();
    PrepararBaseDatos();
}

void PrepararBaseDatos() {
    try {
        String controlador="sun.jdbc.odbc.JdbcOdbcDriver";
        Class.forName(controlador).newInstance();
    } catch(Exception e) {
        JOptionPane.showMessageDialog(null,"Error al cargar el controlador");
    }
}
```



El código que acabas de programar es el primer paso a realizar para el acceso a una base de datos: **La Carga del Controlador**.

Recuerda que el *controlador* le indica a Java que tipo de base de datos usaremos: Access, MySQL, etc...

El controlador que le indica al java que usaremos una base de datos de Access viene definido a través de la siguiente cadena de texto:

```
sun.jdbc.odbc.JdbcOdbcDriver
```

Y la forma de activar dicho controlador es a través de la instrucción:

```
Class.forName(controlador).newInstance();
```

Donde *controlador* es una variable de cadena que contiene la cadena anterior.

Básicamente, lo que hacen estas dos líneas de código, es preparar a Java para poder usar Access.

En el caso de que se quisiera usar una base de datos realizada en otro programa que no fuera Access, habría que cambiar la cadena de texto correspondiente a su controlador.

Por ejemplo, para usar una base de datos creada con el gestor de base de datos MySQL se usa la cadena: `com.mysql.jdbc.Driver`.

En nuestro caso siempre usaremos Access.

También puedes observar que es obligatorio encerrar el código de la carga del controlador entre `try ... catch` para capturar cualquier error imprevisto.

15. Ahora añade el siguiente código a continuación del anterior:

```
try {
    String DSN = "jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ="+
        "Base\\MANEMPSA.MDB";
    String user = "";
    String password = "";
    conexion=DriverManager.getConnection(DSN,user,password);
} catch(Exception e) {
    JOptionPane.showMessageDialog(null,"Error al realizar la conexión");
}
```

El código que acabas de añadir se corresponde con el segundo paso para acceder a una base de datos: **Crear el objeto Conexión**.

El objeto *conexión* es el que efectúa la conexión real con la base de datos. Se podría decir que es el objeto que permite abrir la puerta del fichero de base de datos para entrar en él. Para construir este objeto *conexión* (el cual ya está declarado en la parte global de la clase) hacen falta tres datos:

- El nombre del usuario que manipulará la base de datos. En el caso de Access no necesitamos indicar ningún nombre de usuario, por eso verás la instrucción:

```
String user = "";
```

- El password del usuario que manipulará la base de datos. En el caso de Access tampoco necesitaremos ningún password, por eso verás la instrucción:

```
String password = "";
```

- La DSN de la base de datos. DSN significa “nombre del origen de datos” y es una cadena de texto algo compleja que contiene información sobre el fichero de base de datos que queremos usar. La parte inicial de la cadena de la DSN siempre será igual para las bases de datos de Access:

```
"jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ="
```

Sin embargo, lo que nos interesa realmente de esta cadena es su parte final, donde hay que indicar el camino del fichero de base de datos al que accederemos. En nuestro caso, indicaremos el camino de MANEMPSA.MDB, el fichero de la base de datos.

Observa como indicamos el camino del fichero: Base\\MANEMPSA.MDB

Base es la carpeta donde hemos guardado el fichero, dentro de nuestro proyecto java, y MANEMPSA.MDB como ya sabes es el fichero de Access que contiene la base de datos. Observa la necesidad de escribir dos barras.

En el código podrás ver la creación de la DSN:

```
String DSN = "jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ="+
    "Base\\MANEMPSA.MDB";
```

Finalmente se usa la instrucción

```
conexion=DriverManager.getConnection(DSN,user,password);
```

para crear la conexión. Será necesario añadir un import para la clase *DriverManager*.

Al igual que con el primer paso, es necesario rodear la creación de la conexión con un `try...catch` para capturar cualquier error inesperado que se pueda producir en este paso.

16. Ahora añade el siguiente código a continuación del anterior:

```
try {  
    sentencia=conexion.createStatement(  
        ResultSet.TYPE_SCROLL_INSENSITIVE,  
        ResultSet.CONCUR_READ_ONLY);  
} catch(Exception e) {  
    JOptionPane.showMessageDialog(null,"Error al crear el objeto sentencia");  
}
```

Este código que acabas de añadir se corresponde con el tercer paso necesario para poder acceder a una base de datos: **Creación del objeto Sentencia.**

El objeto *sentencia* será el que nos permita ejecutar órdenes SQL sobre la base de datos. Es decir, el objeto que nos permite actuar y manipular la base de datos. Este objeto es vital, y es el objetivo de toda esta preparación.

El objeto *sentencia* se crea a partir del objeto *conexión* creado en el paso anterior, usando la siguiente instrucción:

```
sentencia=conexion.createStatement(  
    ResultSet.TYPE_SCROLL_INSENSITIVE,  
    ResultSet.CONCUR_READ_ONLY);
```

Y como sucedió en los dos pasos anteriores, es necesario rodear esta instrucción con un try...catch para capturar cualquier tipo de error inesperado que se pudiera producir.

17. Así pues ya tenemos preparado nuestro programa para acceder a la base de datos. Esta preparación se realiza en el momento en que se ejecuta el programa, ya que hemos introducido este código en el constructor.

Es cierto que este código puede resultar bastante abstracto y complejo, pero tiene la gran ventaja de que siempre es igual.

Para nuestros ejercicios, solo tendremos que cambiar el nombre de la base de datos que se esté usando en ese momento. El resto del código queda igual.

Observa:


```

void PrepararBaseDatos() {
    try {
        String controlador="sun.jdbc.odbc.JdbcOdbcDriver";
        Class.forName(controlador).newInstance();
    } catch(Exception e) {
        JOptionPane.showMessageDialog(null,"Error al cargar el controlador");
    }

    try {
        String DSN = "jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ="+
                     "Base\\MANEMPSA.MDB";
        String user = "";
        String password = "";
        conexion=DriverManager.getConnection(DSN,user,password);
    } catch(Exception e) {
        JOptionPane.showMessageDialog(null,"Error al realizar la conexión");
    }

    try {
        sentencia=conexion.createStatement(
            ResultSet.TYPE_SCROLL_INSENSITIVE,
            ResultSet.CONCUR_READ_ONLY);
    } catch(Exception e) {
        JOptionPane.showMessageDialog(null,"Error al crear el objeto sentencia");
    }
}

```

El método *PrepararBaseDatos* siempre será igual, solo cambiará el nombre de la base de datos a usar.

18. El objetivo de todo este código de preparación para el acceso al fichero de la base de datos es obtener un objeto llamado *sentencia* que nos posibilitará la manipulación de los datos de la base de datos, usando órdenes SQL.

En este ejercicio guiado usaremos el objeto *sentencia* para averiguar información acerca de los trabajadores.

REALIZAR CONSULTAS SQL USANDO EL OBJETO SENTENCIA

19. Cuando se pulse el botón *Ver Datos de Trabajadores* tendremos que extraer los datos de la tabla *trabajadores* para poder mostrarlos. Para ello, escribe el siguiente código dentro del evento *actionPerformed* del botón *btnVerDatos*:

```
private void btnVerDatosActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO: Agregue su código aquí:  
    try {  
        ResultSet r=sentencia.executeQuery("select * from trabajadores order by nombre");  
    } catch(Exception e) {  
        JOptionPane.showMessageDialog(null,"Error al consultar la tabla trabajadores");  
    }  
}
```

Observa el código:

```
ResultSet r = sentencia.executeQuery("select * from trabajadores order by nombre");
```

El objeto *sentencia* se usa para dar órdenes a la base de datos. Esas órdenes se dan usando el lenguaje de consulta SQL.

Se usa el método *executeQuery* del objeto *sentencia* para ejecutar la consulta SQL *"select * from trabajadores order by nombre"*. Esta consulta extraerá todos los datos de la tabla *trabajadores* ordenados por nombre.

El método *executeQuery* recibe como parámetro una cadena representando la consulta SQL. No es necesario indicar el punto y coma final de la consulta SQL.

El resultado de la consulta se guarda en un objeto del tipo *ResultSet* al que se ha llamado simplemente "r". Los objetos *ResultSet* almacenan el resultado de una consulta SQL. (Será necesario incluir el *import* necesario para la clase *ResultSet*)

Y como puedes observar, es necesario rodear la ejecución de una consulta SQL con un *try...catch* para capturar errores inesperados al realizar la consulta.

LOS OBJETOS RESULTSET

20. Debes imaginarte el objeto *ResultSet* *r* como una tabla que contiene el resultado de la consulta SQL que se ha ejecutado. En nuestro caso, la consulta SQL que hemos ejecutado ha extraído toda la tabla *trabajadores*. Por tanto nuestro *ResultSet* contiene toda la tabla *trabajadores*.

El objeto *r* por tanto podría representarse así:

Trabajadores					
DNI	Nombre	Apellidos	Sueldo	Fecha	Matricula
BOF					
21.123.123-A	Ana	Ruiz	1200	02/03/2002	3322-ASR
22.333.444-C	Francisco	López	1000	01/06/2006	1144-BBB
12.321.567-B	Juan	Pérez	1120	04/05/2002	4433-ABB
EOF					

21. La fila BOF significa “comienzo de fichero” y representa una fila anterior al primer registro del *ResultSet*.

La fila EOF significa “final de fichero” y representa una fila posterior al último registro del *ResultSet*.

La flecha indica la posición actual donde estamos situados dentro de la tabla del *ResultSet*.

22. Añade la siguiente línea al código del *actionPerformed*:

```
private void btnVerDatosActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO: Agregue su código aquí:  
    try {  
        ResultSet r=sentencia.executeQuery("select * from trabajadores order by nombre");  
        r.next();  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(null,"Error al consultar la tabla trabajadores");  
    }  
}
```

El método *next* del *ResultSet* hará que avancemos una fila en el *ResultSet*. Es decir, ahora estaremos situados en la primera fila del *ResultSet* (la flecha avanza una posición)

Trabajadores					
DNI	Nombre	Apellidos	Sueldo	Fecha	Matricula
BOF					
21.123.123-A	Ana	Ruiz	1200	02/03/2002	3322-ASR
22.333.444-C	Francisco	López	1000	01/06/2006	1144-BBB
12.321.567-B	Juan	Pérez	1120	04/05/2002	4433-ABB
EOF					

23. Ahora que estamos situados en la posición del primer trabajador (Ana), podemos extraer información referente a sus campos. Añade el siguiente código al *actionPerformed* del botón:

```
private void btnVerDatosActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO: Agregue su código aquí:  
    String info;  
  
    try {  
        ResultSet r=sentencia.executeQuery("select * from trabajadores order by nombre");  
        r.next();  
  
        info="El trabajador se llama "+r.getString("nombre")+" "+r.getString("apellidos")+  
            " y cobra "+r.getString("sueldo");  
  
        JOptionPane.showMessageDialog(null,info);  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(null,"Error al consultar la tabla trabajadores");  
    }  
}
```

Lo que se ha hecho primero es declarar una variable de cadena llamada *info*.

Luego, a esta variable se le ha asignado una concatenación de cadenas:

```
info="El trabajador se llama "+r.getString("nombre")+" "+r.getString("apellidos")+  
    " y cobra "+r.getString("sueldo");
```

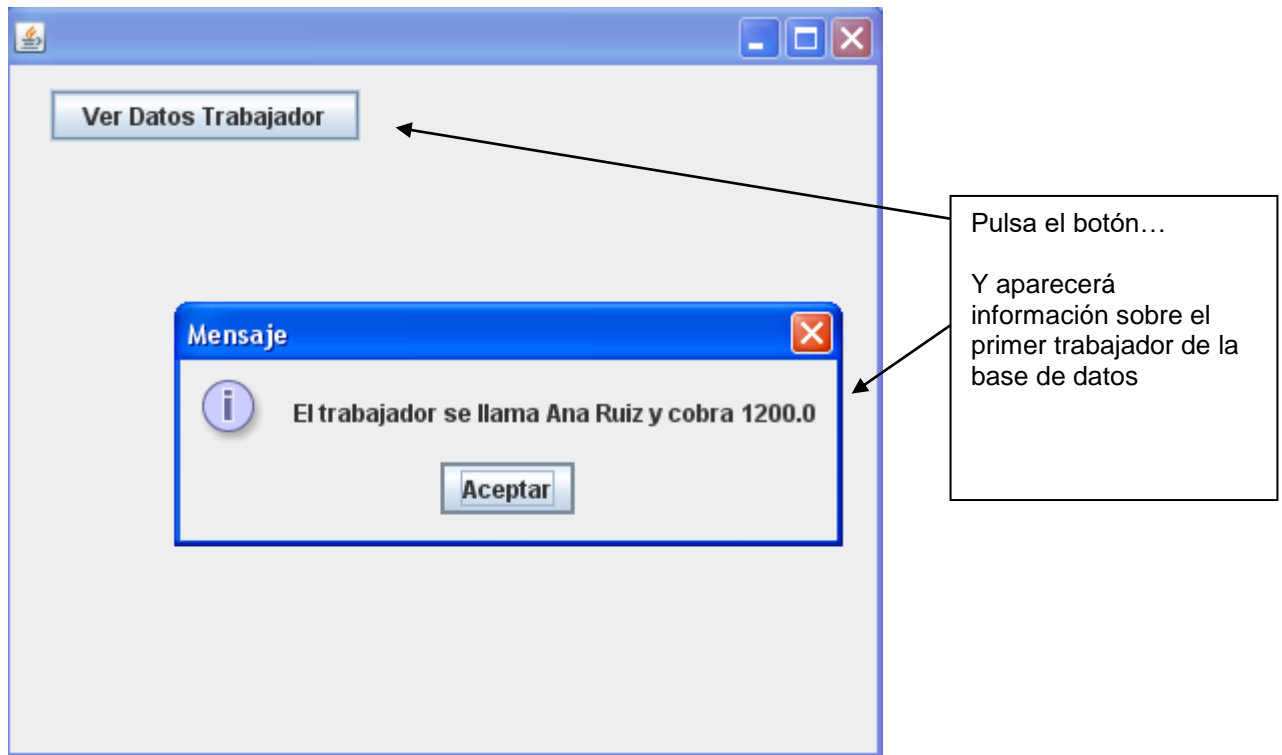
Lo interesante de esto es el método *getString* del objeto *ResultSet* *r*. El método *getString* permite extraer una cadena con el valor del campo indicado como parámetro.

En nuestro caso:

<i>r.getString("nombre")</i>	→ Extrae el nombre del trabajador actual ("Ana")
<i>r.getString("apellidos")</i>	→ Extrae los apellidos del trabajador actual ("Ruiz")
<i>r.getString("sueldo")</i>	→ Extrae el sueldo del trabajador actual ("1200")

Luego se muestra la cadena *info* en un simple *JOptionPane*.

24. Ya puedes ejecutar el programa.



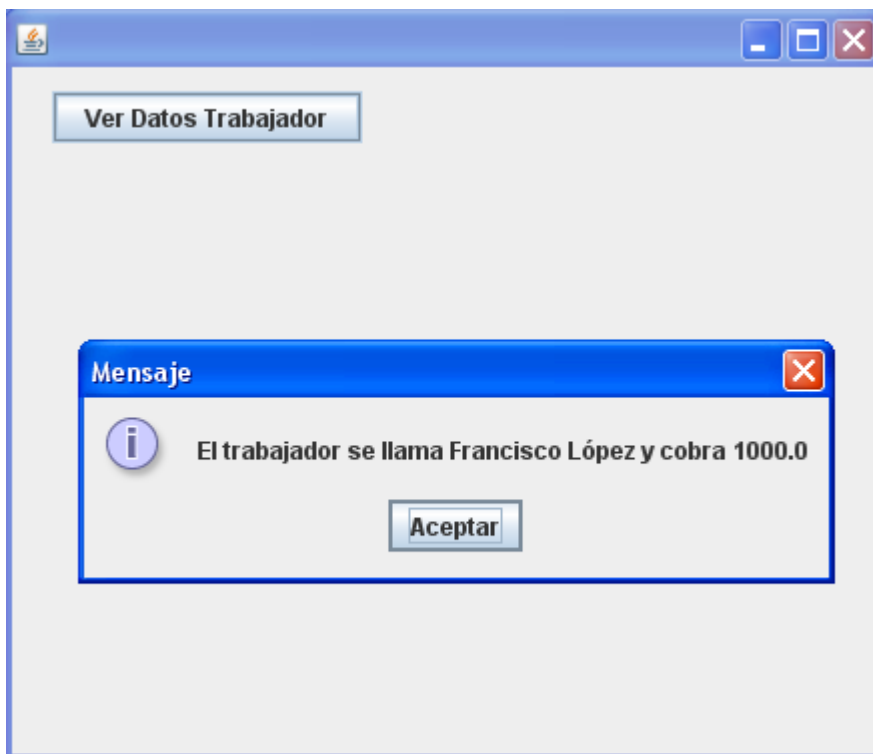
25. Sigamos haciendo cambios en el código del botón para entender mejor el funcionamiento de los *ResultSet*. Añade la siguiente línea:

```
private void btnVerDatosActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO: Agrega su código aquí:  
    String info;  
  
    try {  
        ResultSet r=sentencia.executeQuery("select * from trabajadores order by nombre");  
        r.next();  
        → r.next();  
  
        info="El trabajador se llama "+r.getString("nombre")+" "+r.getString("apellidos")+  
            " y cobra "+r.getString("sueldo");  
  
        JOptionPane.showMessageDialog(null,info);  
  
    } catch(Exception e) {  
        JOptionPane.showMessageDialog(null,"Error al consultar la tabla trabajadores");  
    }  
  
}
```

26. Se ha añadido un segundo *next*. Esto producirá que la flecha avance dos posiciones en el *ResultSet*, y por tanto se coloque en la segunda fila:

Trabajadores					
DNI	Nombre	Apellidos	Sueldo	Fecha	Matricula
BOF					
21.123.123-A	Ana	Ruiz	1200	02/03/2002	3322-ASR
→ 22.333.444-C	Francisco	López	1000	01/06/2006	1144-BBB
12.321.567-B	Juan	Pérez	1120	04/05/2002	4433-ABB
EOF					

27. Esto quiere decir que si se ejecuta el programa se mostrará información sobre *Francisco López*. Compruébalo:



28. Los objetos *ResultSet* poseen diversos métodos para cambiar la posición actual en la tabla del *ResultSet*. Dicho de otro modo: “para mover la flecha”. Veamos algunos de estos métodos (se supone que el objeto *ResultSet* se llama *r*):

<code>r.next();</code>	→ Mueve la flecha a la siguiente fila
<code>r.previous();</code>	→ Mueve la flecha a la fila anterior
<code>r.first();</code>	→ Mueve la flecha a la primera fila
<code>r.last();</code>	→ Mueve la flecha a la última fila
<code>r.beforeFirst()</code>	→ Mueve la flecha a la fila BOF
<code>r.afterLast()</code>	→ Mueve la flecha a la fila EOF
<code>r.absolute(n)</code>	→ Mueve la flecha a la fila <i>n</i> del <i>ResultSet</i> .
Las filas se empiezan a numerar por 1.	

29. Haga el siguiente cambio en el *actionPerformed* simplemente para experimentar:

```
private void btnVerDatosActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO: Agregue su código aquí:  
    String info;  
  
    try {  
        ResultSet r=sentencia.executeQuery("select * from trabajadores order by nombre");  
        r.afterLast();  
        r.previous();  
  
        info="El trabajador se llama "+r.getString("nombre")+" "+r.getString("apellidos")+  
            " y cobra "+r.getString("sueldo");  
  
        JOptionPane.showMessageDialog(null,info);  
  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(null,"Error al consultar la tabla trabajadores");  
    }  
}
```

En este caso la flecha se coloca en EOF (*afterLast*) y luego retrocede una fila (*previous*). Por tanto, al ejecutar el programa se mostrarán los datos del último trabajador. Comprueballo.



Trabajadores					
DNI	Nombre	Apellidos	Sueldo	Fecha	Matricula
BOF					
21.123.123-A	Ana	Ruiz	1200	02/03/2002	3322-ASR
22.333.444-C	Francisco	López	1000	01/06/2006	1144-BBB
12.321.567-B	Juan	Pérez	1120	04/05/2002	4433-ABB
EOF					

30. Otro experimento. Cambie ahora el código de esta forma:

```
private void btnVerDatosActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO: Agregue su código aquí:  
    String info;  
  
    try {  
        ResultSet r=sentencia.executeQuery("select * from trabajadores order by nombre");  
        r.absolute(2);  
        r.next();  
        r.previous();  
        r.previous();  
  
        info="El trabajador se llama "+r.getString("nombre")+" "+r.getString("apellidos")+  
            " y cobra "+r.getString("sueldo");  
  
        JOptionPane.showMessageDialog(null,info);  
  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(null,"Error al consultar la tabla trabajadores");  
    }  
}
```

Este código coloca la flecha en la fila 2 (*absolute(2)*), luego avanza una fila (*next*), luego retrocede una fila (*previous*) y finalmente retrocede una fila (*previous*)

Así pues, finalmente, la flecha queda colocada en la primera fila. Por lo tanto se muestran los datos del primer trabajador. Compruébalo.



Trabajadores					
DNI	Nombre	Apellidos	Sueldo	Fecha	Matricula
BOF					
21.123.123-A	Ana	Ruiz	1200	02/03/2002	3322-ASR
22.333.444-C	Francisco	López	1000	01/06/2006	1144-BBB
12.321.567-B	Juan	Pérez	1120	04/05/2002	4433-ABB
EOF					

31. Como ves, podemos movernos dentro del contenido del *ResultSet* gracias a todos estos métodos, para luego poder extraer los datos de la fila correspondiente.

Ahora, estudiaremos la forma de recorrer todas las filas del *ResultSet* para así extraer la información de todos sus registros.

32. Vamos a modificar nuestro código para que se muestren todos los trabajadores del *ResultSet*. Para ello, realiza el siguiente cambio:

```
private void btnVerDatosActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO: Agregue su código aquí:  
    String info=""; ←  
  
    try {  
        ResultSet r=sentencia.executeQuery("select * from trabajadores order by nombre");  
  
        {  
            r.beforeFirst();  
            while (r.next()) {  
                info=info+r.getString("nombre")+" "+r.getString("apellidos")+" "+r.getString("sueldo")+"\n";  
            }  
        }  
  
        JOptionPane.showMessageDialog(null,info);  
  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(null,"Error al consultar la tabla trabajadores");  
    }  
}
```

En este código se ha inicializado la variable *info* a la cadena vacía "" y luego se ha añadido un bucle bastante interesante:

```
r.beforeFirst();  
while (r.next()) {  
    info=info+r.getString("nombre")+" "+r.getString("apellidos")+"  
    "+r.getString("sueldo")+"\n";  
}
```

Analicemos este bucle:

- Lo primero que se hace es colocar explícitamente la flecha en la fila BOF, es decir, antes del primer trabajador:

```
r.beforeFirst();
```

- Luego tenemos un bucle *mientras* que comienza así:

```
while (r.next()) {
```

El método *next* intenta colocar la flecha en la siguiente fila, y si lo hace bien, devuelve el valor *verdadero*. Cuando no se puede avanzar más, el método *next* devolverá *falso*.

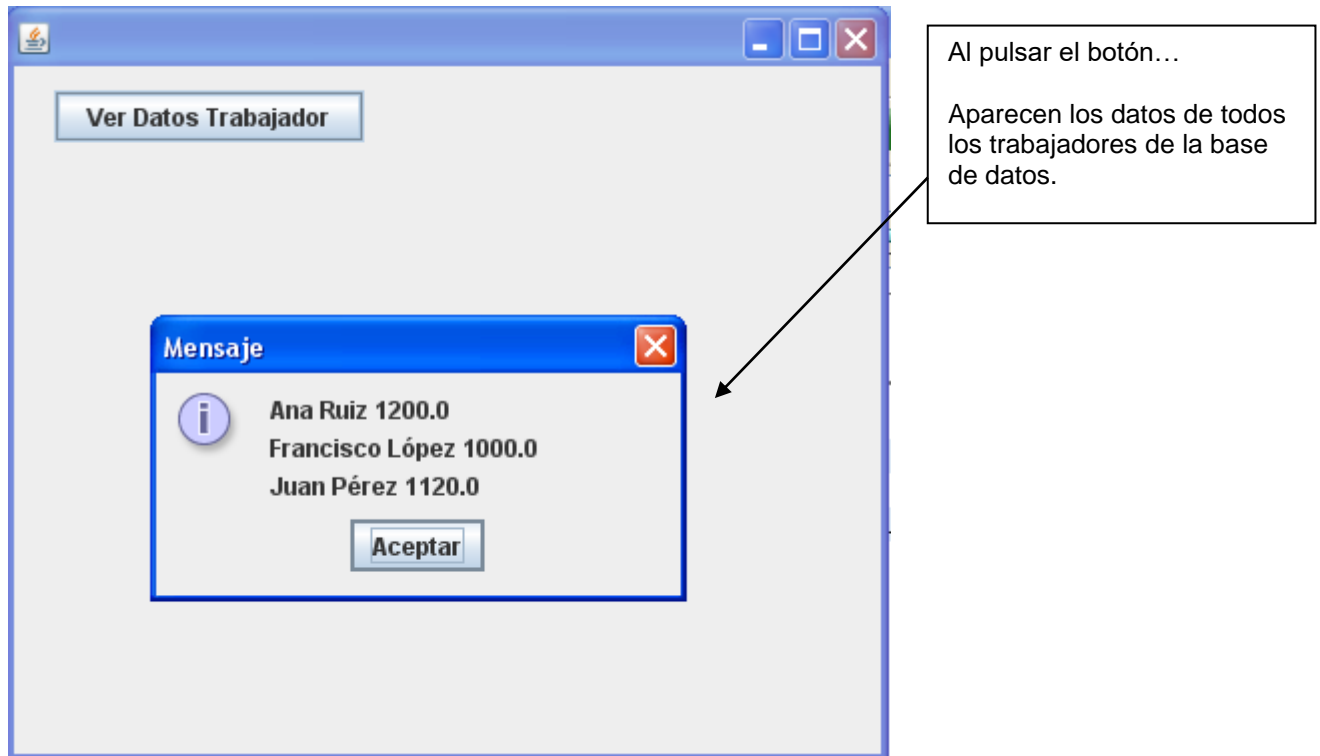
Así pues, este *while* significa "*mientras se haya podido avanzar una fila, haz lo siguiente*". O dicho de otro modo, este bucle se repetirá para cada fila de la tabla del *ResultSet*.

- Si analizamos el contenido del bucle, básicamente veremos una concatenación de cadenas dentro de la variable *info*. En cada vuelta del bucle se concatenará el nombre, apellidos y sueldo de cada trabajador.

```
info=info+r.getString("nombre")+" "+r.getString("apellidos")+"  
"+r.getString("sueldo")+"\n";
```

- Finalmente se visualiza la variable *info* en un JOptionPane.

33. Ejecuta el programa y comprueba su funcionamiento.



34. El bucle que acabas de programar es un código “clásico” para manipular un *ResultSet*. Siempre que quieras recorrer todas las filas del *ResultSet* harás algo como esto:

```
r.beforeFirst();  
while (r.next()) {  
    manipulación de la fila  
}
```

35. Se ha visto que para obtener un dato de la fila actual se usa el método *getString* indicando como parámetro el dato que se quiere obtener.

Por ejemplo:

```
r.getString("sueldo");
```

Obtiene el sueldo del trabajador señalado actualmente con la flecha. Este sueldo se obtiene convertido en cadena.

36. Los *ResultSet* poseen otros métodos para obtener los datos convertidos en números como son:

```
getInt("campo")
```

y

```
getDouble("campo")
```

para obtener el dato en entero o double, respectivamente.

Esto nos permite el realizar operaciones con los datos extraídos del *ResultSet*.

37. Como ejemplo de esto último, realice la siguiente mejora al programa:

```
private void btnVerDatosActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO: Agregue su código aquí:  
    String info="";  
    double totalsu=0; //total sueldo ←  
  
    try {  
        ResultSet r=sentencia.executeQuery("select * from trabajadores order by nombre");  
  
        r.beforeFirst();  
        while (r.next()) {  
            info=info+r.getString("nombre")+" "+r.getString("apellidos")+" "+r.getString("sueldo")+"\n";  
            totalsu=totalsu+r.getDouble("sueldo"); ←  
        }  
  
        JOptionPane.showMessageDialog(null,info);  
        JOptionPane.showMessageDialog(null,"La suma de los sueldos es "+totalsu); ←  
  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(null,"Error al consultar la tabla trabajadores");  
    }  
}
```

En este nuevo código se ha añadido una variable double acumuladora llamada *totalsu*, donde sumaremos todos los sueldos.

Dentro del bucle, se van acumulando el sueldo de cada trabajador. Observa el uso de *getDouble* para obtener el campo sueldo de forma numérica, en vez de usar *getString*:

```
totalsu=totalsu+r.getDouble("sueldo");
```

Y finalmente usamos otro *JOptionPane* para ver la suma de los sueldos calculada.

Ejecuta el programa y comprueba su funcionamiento.

38. Una vez finalizado el programa, es una buena costumbre cerrar la base de datos que estamos manejando. Esto se hace cerrando la “conexión” con la base de datos.

Para hacer esto se usa el método *close* del objeto *conexión*.

Esto se hará en el momento en que se finalice el programa, es decir, en el evento *windowClosing* de la ventana principal:

```
private void formWindowClosing(java.awt.event.WindowEvent evt) {  
    // TODO: Agregue su código aquí:  
    try {  
        conexion.close();  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(null, "No se pudo cerrar la base de datos");  
    }  
}
```

Como ves, también hay que encerrar este código entre try...catch para evitar posibles errores.

CONCLUSIÓN

Para crear un programa Java que pueda usar una base de datos será necesario realizar los siguientes pasos:

- Colocar el fichero de base de datos en una subcarpeta de la carpeta del proyecto java.
- Preparar el acceso a la base de datos (en el constructor)
 - * Se crearán dos objetos: *conexión (Connection)* y *sentencia (Statement)*
 - * Se cargará el controlador del tipo de base de datos a usar
 - * Se creará el objeto conexión indicando el fichero de la base de datos.
 - * Se creará el objeto sentencia a partir del objeto conexión
- Se usará el objeto *sentencia* para ejecutar consultas SQL en la base de datos.
- Las consultas SQL ejecutadas en la base de datos se almacenan en objetos del tipo *ResultSet*
- Un objeto *ResultSet* tiene forma de tabla conteniendo el resultado de la consulta SQL
 - * Los objetos *ResultSet* tienen métodos para seleccionar el registro de la tabla
 - * Los objetos *ResultSet* tienen métodos que permiten extraer el dato de un campo en concreto.