EJERCICIO GUIADO. JAVA. ACCESO A BASE DE DATOS

Recapitulando…

Para hacer una aplicación java que acceda a una base de datos se tiene que…

- Introducir la base de datos en una subcarpeta del proyecto.
- Preparar la base de datos desde el constructor.
- Usar el objeto sentencia cada vez que se quiera consultar la base de datos o actuar sobre ella.
- Los resultados de las consultas ejecutadas sobre la base de datos se recogerán en objetos *ResultSet* que contendrán los datos devueltos por la consulta.
- Cerrar la conexión a la base de datos cuando ya no haya que usarla más.

Consultar la Base de Datos

En la hoja anterior se vio que se puede usar el objeto *sentencia* para ejecutar una consulta SQL en la base de datos. Al hacer esto se consigue un objeto *ResultSet* que contiene el resultado de la consulta.

El contenido del *ResultSet* tiene forma de tabla, y podemos extraer la información colocándonos en la fila correspondiente del *ResultSet* y luego usando los métodos:

```
getString
getDouble
getInt
```

según queramos extraer el dato en forma de cadena, de número real o número entero.

En esta hoja guiada se insistirá sobre la forma de manipular los datos contenidos en un ResultSet

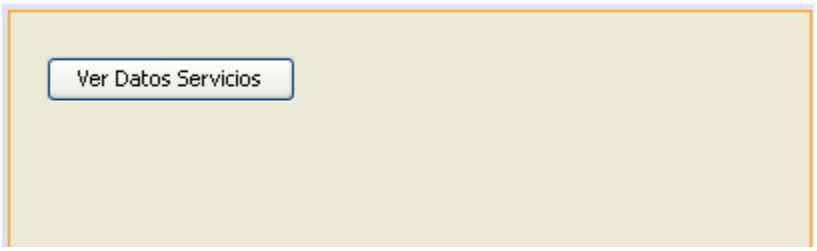
EJERCICIO GUIADO Nº 1

PLANTEAMIENTO

Se quiere realizar una pequeña aplicación de base de datos que nos muestre información sobre los servicios realizados en la empresa MANEMPSA. Para ello, siga los pasos que se indican a continuación:

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

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



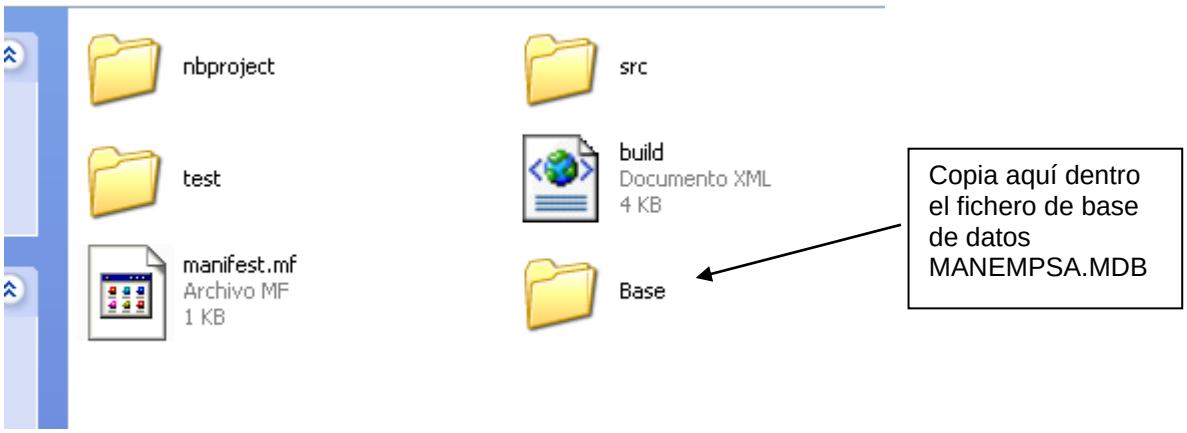
3. Se pretende simplemente que al pulsar el botón *btnServicios* aparezcan en un JOptionPane datos sobre los servicios almacenados en la base de datos.

SITUACIÓN DEL FICHERO DE BASE DE DATOS

4. Como se vio en la hoja anterior, interesa colocar el fichero de la base de datos que se va a usar en una subcarpeta de la carpeta de proyecto que se está haciendo.

   Así pues, entre en la carpeta de proyecto *ServiciosBD*

5. Y dentro de ella crea una carpeta *Base*. Dentro de la carpeta *Base* copia el fichero de base de datos MANEMPSA.MDB, el cual se encuentra dentro de la carpeta Mis Documentos.



6. Ahora ya podemos volver al NetBeans y continuar con nuestro trabajo.

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

7. Preparar nuestro proyecto para que permita el acceso a la base de datos MANEMPSA.MDB es un proceso complejo, aunque afortunadamente siempre se hace igual.

   Solo tiene que añadir el siguiente código a su ventana principal:

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

    Connection conexion;
    Statement sentencia;

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

Declara los objetos globales *conexión* y *sentencia*

Haz la llamada al método *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");
   }

   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");
   }
}
```

Recuerda que el método *PrepararBaseDatos* siempre será igual, solo tienes que indicar aquí el nombre de la base de datos a usar.

REALIZAR CONSULTAS SQL USANDO EL OBJETO SENTENCIA

8. Ahora que hemos preparado nuestro proyecto para poder usar la base de datos MANEMPSA.MDB, ya podemos programar el botón para visualizar los servicios. Entra dentro del *actionPerformed* de este botón y programa lo siguiente:

```
    private void btnServiciosActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:

       String info="";

       try {
        ResultSet r=sentencia.executeQuery("select * from servicios order by cantidad");

        r.beforeFirst();
        while (r.next()) {
           info=info+r.getString("tipo")+" "+r.getString("cantidad")+"\n";
        }
        JOptionPane.showMessageDialog(null,info);

       } catch(Exception e) {
           JOptionPane.showMessageDialog(null,"Error al consultar la tabla servicios");
       }

    }
```

Si observas el código, lo que hace es ejecutar la consulta SQL

```
select * from servicios order by cantidad
```

la cual extrae todos los servicios almacenados en la tabla *servicios* ordenados por *cantidad* de menor a mayor.

El resultado de esta consulta se almacena en un *ResultSet* y se usa un bucle típico que recorre el *ResultSet* y muestra el tipo de cada servicio y la cantidad:

```
while (r.next()) {
   info=info+r.getString("tipo")+" "+r.getString("cantidad")+"\n";
}
```

Puedes ejecutar el programa para ver como funciona.

9. Aunque no es vital para el programa, añadamos el cierre de la conexión de la base de datos en el *windowClosing* de nuestra ventana:

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

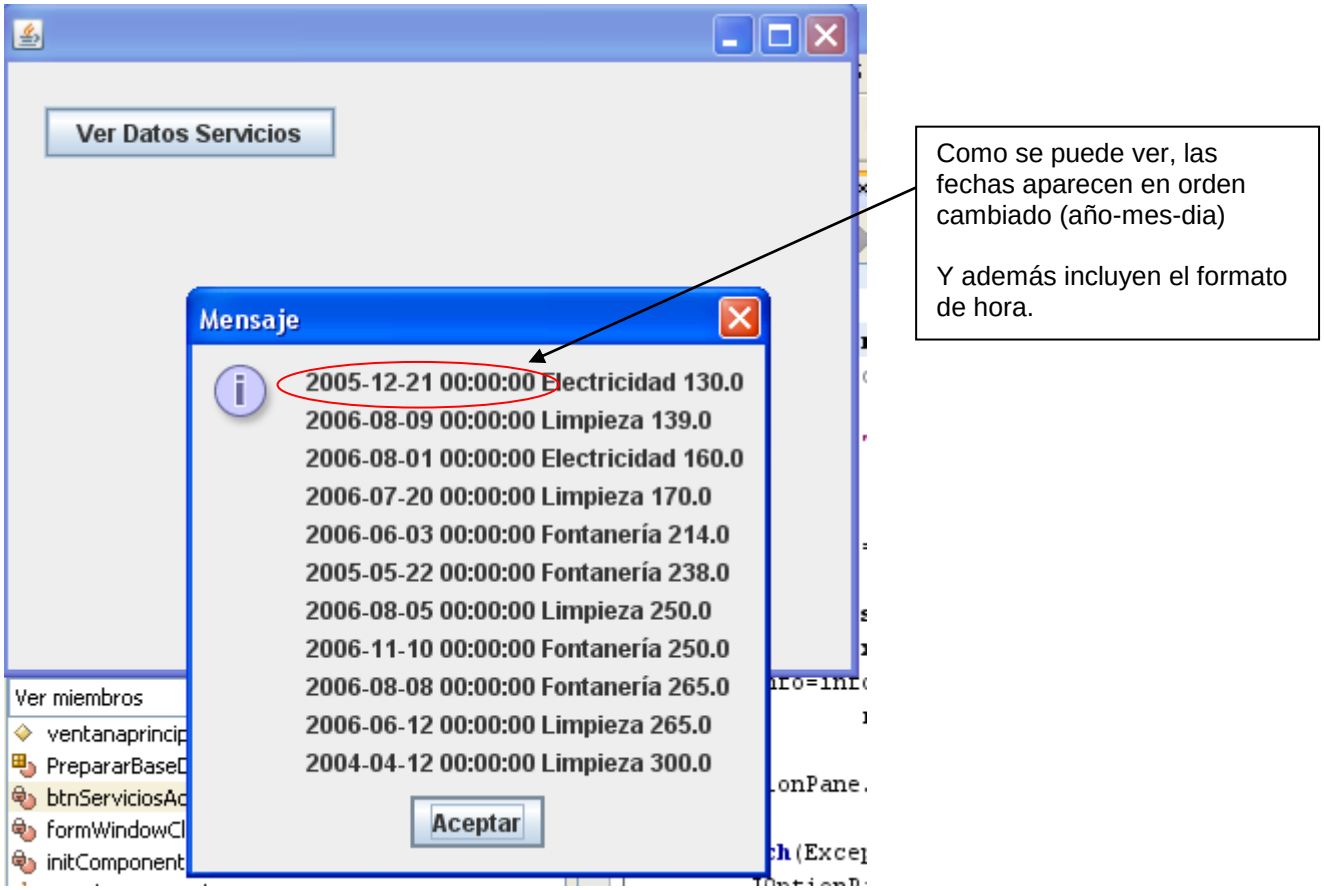
EXTRAER FECHAS DEL RESULTSET

10. Se va a mejorar el programa de forma que se muestre de cada servicio el tipo, la cantidad y la fecha en que se hizo. Por tanto, haz el siguiente cambio en el código del *actionPerformed* del botón *btnServicios*:

```
private void btnServiciosActionPerformed(java.awt.event.ActionEvent evt) {
TODO: Agrege su codigo aqui:

    String info="";

    try {
     ResultSet r=sentencia.executeQuery("select * from servicios order by cantidad");

     r.beforeFirst();
     while (r.next()) {
        info=info+r.getString("fecha")+" "+r.getString("tipo")+" "+
                r.getString("cantidad")+"\n";
     }
     JOptionPane.showMessageDialog(null,info);

    } catch(Exception e) {
        JOptionPane.showMessageDialog(null,"Error al consultar la tabla servicios");
    }

}
```

Simplemente estamos cambiando la concatenación de la cadena *info* de forma que aparezca la fecha de cada servicio, el tipo y la cantidad.

11. Ejecuta el programa y observa el resultado.

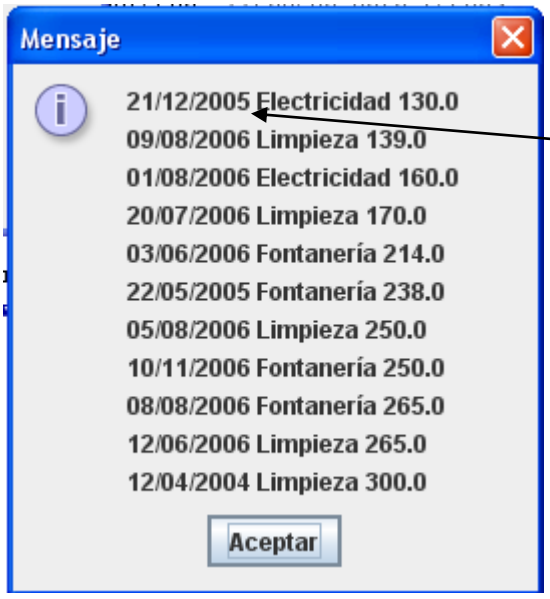


12. Como se ha podido observar, las fechas extraídas del *ResultSet* tienen un formato distinto al que usamos normalmente.

    Para mejorar la presentación de las fechas extraídas del *ResultSet* haz los siguientes cambios en el código:

```
    private void btnServiciosActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        String info="";
        String cadfecha; //cadena para fechas
        String caddia; //cadena para el dia
        String cadmes; //cadena para el mes
        String cadanio; //cadena para el año

        try {
         ResultSet r=sentencia.executeQuery("select * from servicios order by cantidad");

         r.beforeFirst();
         while (r.next()) {

            cadfecha=r.getString("fecha");
            cadanio=cadfecha.substring(0,4);
            cadmes=cadfecha.substring(5,7);
            caddia=cadfecha.substring(8,10);
            cadfecha=caddia+"/"+cadmes+"/"+cadanio;

            info=info+cadfecha+" "+r.getString("tipo")+" "+
                    r.getString("cantidad")+"\n";
         }
         JOptionPane.showMessageDialog(null,info);

        } catch(Exception e) {
            JOptionPane.showMessageDialog(null,"Error al consultar la tabla servicios");
        }
    }
```

El código se ha mejorado de forma que la fecha aparezca en un formato español correcto. Ejecuta el programa para comprobar el resultado:



Observa como ahora las fechas aparecen correctamente…

13. Estudiemos el código que acabamos de añadir:

    Lo primero que se ha hecho es crear varias variables de cadenas para contener el día, mes y año de la fecha así como la fecha completa.

```
String cadfecha; //cadena para fechas
String caddia; //cadena para el dia
String cadmes; //cadena para el mes
String cadanio; //cadena para el año
```

    Dentro del bucle extraemos la fecha del *ResultSet* y la almacenamos en la variable *cadfecha*:

```
cadfecha=r.getString("fecha");
```

    Ahora mismo, la variable *cadfecha* contendrá una cadena como la siguiente:

2	0	0	5	-	1	2	-	2	1	-	0	0	:	0	0	:	0	0

    Para extraer el año de la cadena, extraemos los caracteres comprendidos entre la posición 0 y la posición 3. Esto se hace usando el método *substring* de la siguiente forma:

```
cadanio=cadfecha.substring(0,4);
```

    Para extraer el mes de la cadena, tendremos que extraer los caracteres comprendidos entre la posición 5 y la posición 6. Esto se hace usando el método *substring* de la siguiente forma:

```
cadmes=cadfecha.substring(5,7);
```

    Para extraer el dia de la cadena, tendremos que extraer los caracteres comprendidos entre la posición 8 y la posición 9 de la cadena. Esto se hace usando el método *substring* de la siguiente forma:

```
caddia=cadfecha.substring(8,10);
```

    Una vez extraídos *dia*, *mes* y *año*, de la cadena, podemos concatenarlos formando una fecha en formato dia/mes/año de la siguiente forma:

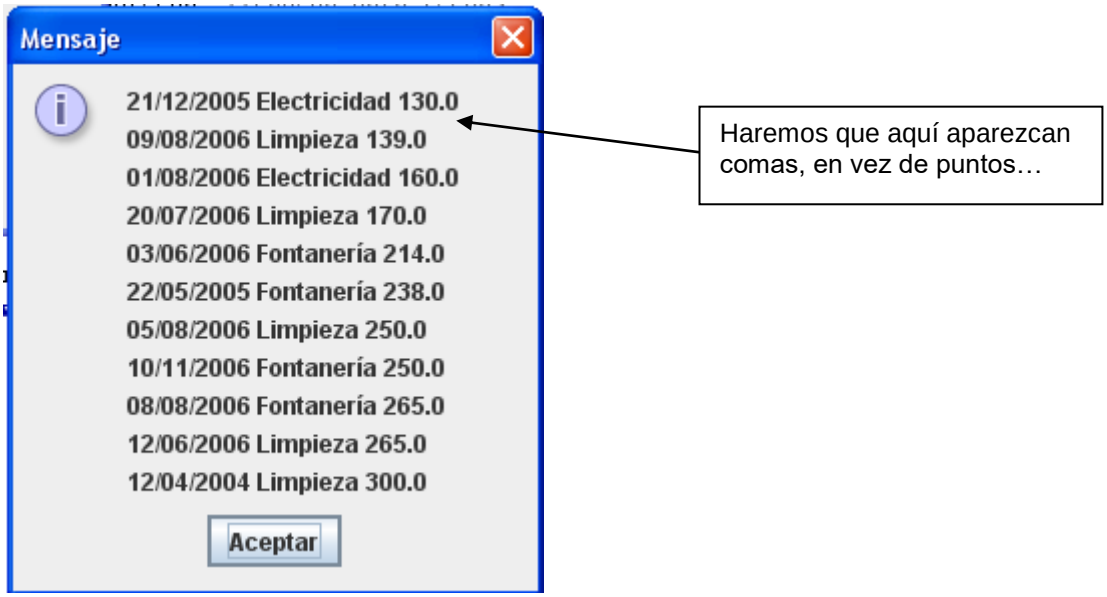
```
cadfecha=caddia+"/"+cadmes+"/"+cadanio;
```

    Así pues, finalmente tenemos una variable *cadfecha* que será la que se visualizará en el JOptionPane:

```
info=info+cadfecha+" "+r.getString("tipo")+" "+
                    r.getString("cantidad")+"\n";
```

PRESENTACIÓN DE COMAS DECIMALES

14. Ahora mejoraremos el programa para que los costes de los servicios aparezcan con coma decimal, en vez de punto decimal:



Modifica el código de la siguiente forma:

```
    private void btnServiciosActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        String info="";
        String cadfecha; //cadena para fechas
        String caddia; //cadena para el dia
        String cadmes; //cadena para el mes
        String cadanio; //cadena para el año
        String cadcoste; //cadena para el coste

        try {
         ResultSet r=sentencia.executeQuery("select * from servicios order by cantidad");

         r.beforeFirst();
         while (r.next()) {
            cadfecha=r.getString("fecha");
            cadanio=cadfecha.substring(0,4);
            cadmes=cadfecha.substring(5,7);
            caddia=cadfecha.substring(8,10);
            cadfecha=caddia+"/"+cadmes+"/"+cadanio;

            cadcoste=r.getString("cantidad");
            cadcoste=cadcoste.replace(".",",");

            info=info+cadfecha+" "+r.getString("tipo")+" "+cadcoste+"\n";
         }
         JOptionPane.showMessageDialog(null,info);

        } catch(Exception e) {
            JOptionPane.showMessageDialog(null,"Error al consultar la tabla servicios");
        }
    }
```

Se ha añadido una variable de cadena llamada *cadcoste* que almacenará el coste de cada servicio.

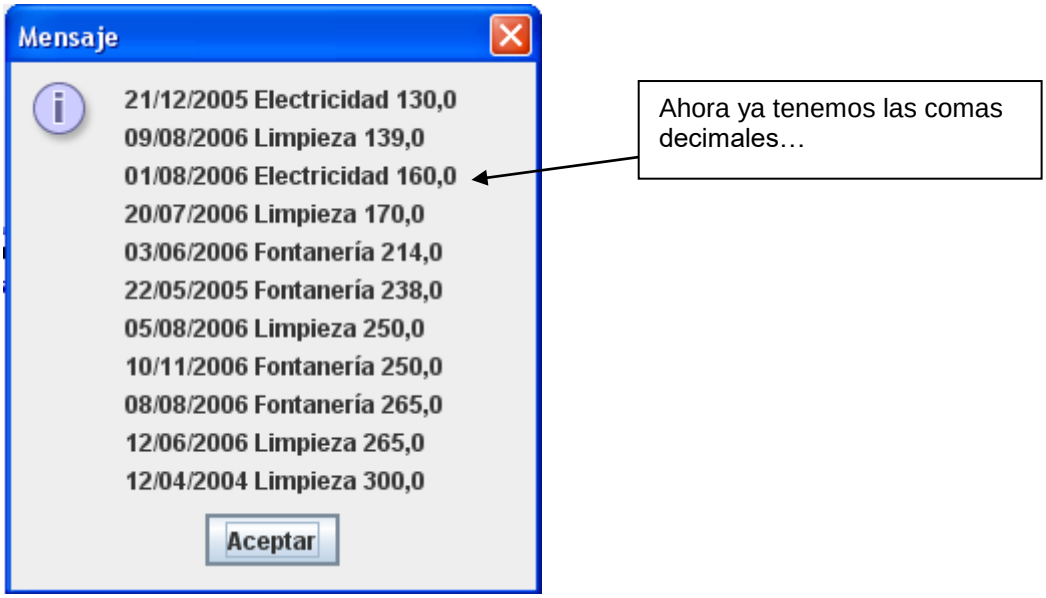
En el código del bucle, recogemos la cantidad en dicha variable y luego usamos el método de cadena *replace* para reemplazar los puntos por comas:

```
cadcoste=r.getString("cantidad");
cadcoste=cadcoste.replace(".",",");
```

Finalmente, mostramos la cadena de coste en la concatenación:

```
info=info+cadfecha+" "+r.getString("tipo")+" "+cadcoste+"\n";
```

Ejecuta el programa y observa el resultado:



VALORES NULOS

15. Es posible que algún campo de algún registro de la tabla esté vacío. Es decir, que sea nulo. Si esto ocurre, entonces al extraer dicho dato de la tabla usando *getString* aparecerá el valor *null* en el JOptionPane.

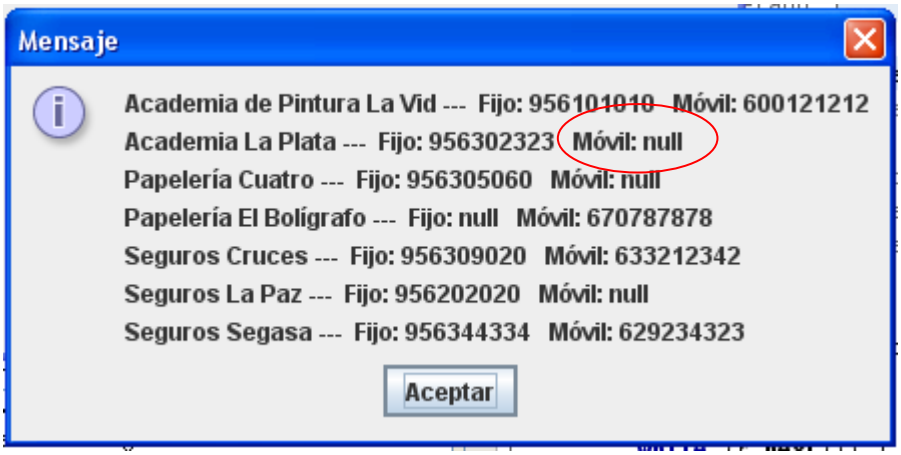
16. Para comprobar esta circunstancia, agrega un nuevo botón a la ventana principal con el texto "*Ver Datos de Clientes*". Llámalo por ejemplo *btnClientes*.

17. Al pulsar este botón aparecerá el listado de clientes de la empresa. Concretamente debe aparecer el nombre del cliente, el teléfono 1 y el teléfono 2. Para ello añade el siguiente código dentro del evento *actionPerformed* del botón.

```
    private void btnClientesActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        String info="";
        String nomcli; //cadena el nombre de cliente
        String tfno1; //cadena para telefono 1
        String tfno2; //cadena para telefono 2

        try {
         ResultSet r=sentencia.executeQuery("select * from clientes order by nombre");

         r.beforeFirst();
         while (r.next()) {
            nomcli=r.getString("nombre");
            tfno1=r.getString("tfno1");
            tfno2=r.getString("tfno2");

            info=info+nomcli+" ---  "+"Fijo: "+tfno1+"   "+"Móvil: "+tfno2+"\n";
         }
         JOptionPane.showMessageDialog(null,info);

        } catch(Exception e) {
            JOptionPane.showMessageDialog(null,"Error al consultar la tabla clientes");
        }
    }
```

Este código es prácticamente igual que el anterior. Simplemente ejecuta una consulta SQL usando el objeto *sentencia* que permite extraer el contenido de la tabla *clientes*, y luego recorre el *ResultSet* mostrando los campos *nombre, teléfono 1 y teléfono 2* en un JOptionPane.

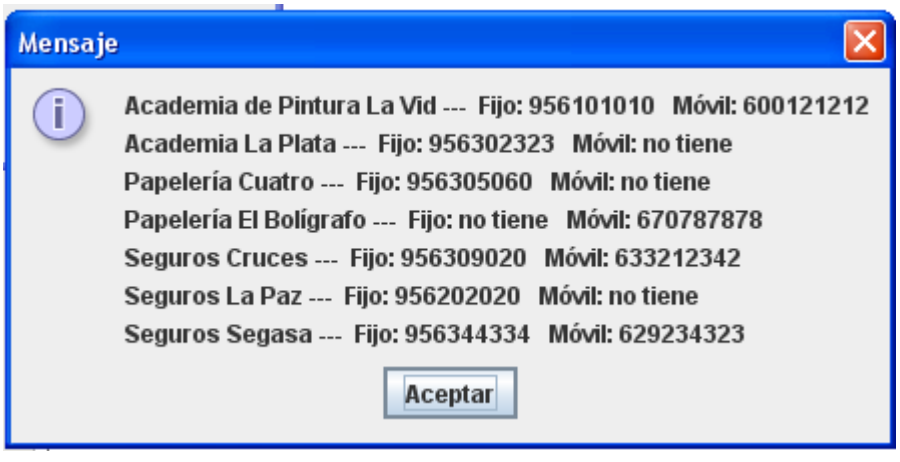
18. Ejecuta el programa ahora y prueba a pulsar este nuevo botón. Observa el resultado. Cada vez que un cliente no tenga un teléfono, aparecerá el valor “null” en el JOptionPane:



19. Vamos a arreglar esto de forma que aparezca el texto “no tiene” en vez de la cadena “null”. Modifique el código como se indica:

```
    private void btnClientesActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        String info="";
        String nomcli; //cadena el nombre de cliente
        String tfno1; //cadena para telefono 1
        String tfno2; //cadena para telefono 2

        try {
         ResultSet r=sentencia.executeQuery("select * from clientes order by nombre");

         r.beforeFirst();
         while (r.next()) {
            nomcli=r.getString("nombre");
            tfno1=r.getString("tfno1");
            if (tfno1==null) {
                tfno1="no tiene";
            }
            tfno2=r.getString("tfno2");
            if (tfno2==null) {
                tfno2="no tiene";
            }

            info=info+nomcli+" ---  "+"Fijo: "+tfno1+"   "+"Móvil: "+tfno2+"\n";
         }
         JOptionPane.showMessageDialog(null,info);

        } catch(Exception e) {
            JOptionPane.showMessageDialog(null,"Error al consultar la tabla clientes");
        }
    }
```

Como puedes ver, lo que se hace ahora es comprobar si el valor extraído del *ResultSet* es null, y en ese caso, se concatena la cadena *no tiene*. En caso contrario se concatena el valor del campo.



CONCLUSIÓN

A través del objeto *sentencia* podemos ejecutar una consulta SQL en una base de datos.

El resultado de la consulta se almacena en un objeto del tipo *ResultSet*.

Al extraer el valor de un campo fecha desde el objeto *ResultSet* observaremos que tiene el siguiente formato:

Año – Mes – Dia – Hora : Minutos : Segundos

Así pues puede ser necesario realizar cambios en esta cadena.

Al extraer el valor de un campo numérico real, obtendremos un número con punto decimal. Quizás sea necesario cambiar este punto por una coma decimal.

Cuando un campo está vacío, al intentar extraer su valor obtendremos el valor *null*.