EJERCICIO GUIADO. JAVA. ACCESO A BASE DE DATOS

Recapitulando. Consultas de selección. Consultas de Acción.

Al estudiar SQL, vimos que existían dos tipos de instrucciones.

- Consultas de selección (SELECT)

  Estas consultas permiten extraer datos de la base de datos. Dicho de otro modo, permiten visualizar información de la base de datos que cumpla un criterio.

  Estas consultas no afectan a la base de datos, simplemente muestran información sobre la propia base de datos.

- Consultas de acción. (INSERT, DELETE, UPDATE)

  Estas consultas realizan una acción sobre la base de datos. Esta acción puede ser:

  - Insertar un nuevo registro en una tabla (INSERT)
  - Borrar un registro o registros de una tabla (DELETE)
  - Modificar los datos de un registro o registros de la tabla (UPDATE)

Ejecución de consultas de selección y de consultas de acción.

Hasta ahora se han realizado programas java que ejecutaban consultas de selección sobre la base de datos.

Recuerda que para ejecutar estas consultas se usa el método *executeQuery* del objeto *sentencia* y el resultado de la consulta se almacena en un objeto *ResultSet*:

```
ResultSet r = sentencia.executeQuery("select . . .");
```

En esta hoja guiada veremos la ejecución de consultas de acción sobre la base de datos desde la aplicación java. Este tipo de consultas se ejecutan usando el método *executeUpdate* del objeto *sentencia*, y no devuelven un resultado concreto, ya que simplemente actúan sobre la base de datos modificando de alguna manera su contenido.

Así pues, para realizar un alta en la base de datos se usará:

```
sentencia.executeUpdate("insert . . .");
```

Para realizar una modificación en la base de datos se usará:

```
sentencia.executeUpdate("update . . . ");
```

Para realizar una eliminación en la base de datos se usará:

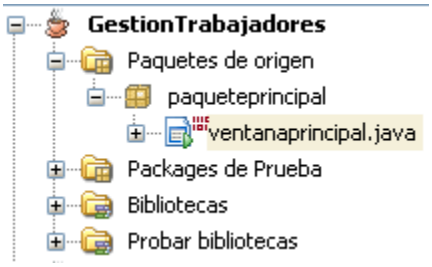
```
sentencia.executeUpdate("delete . . . ");
```

EJERCICIO GUIADO Nº 1

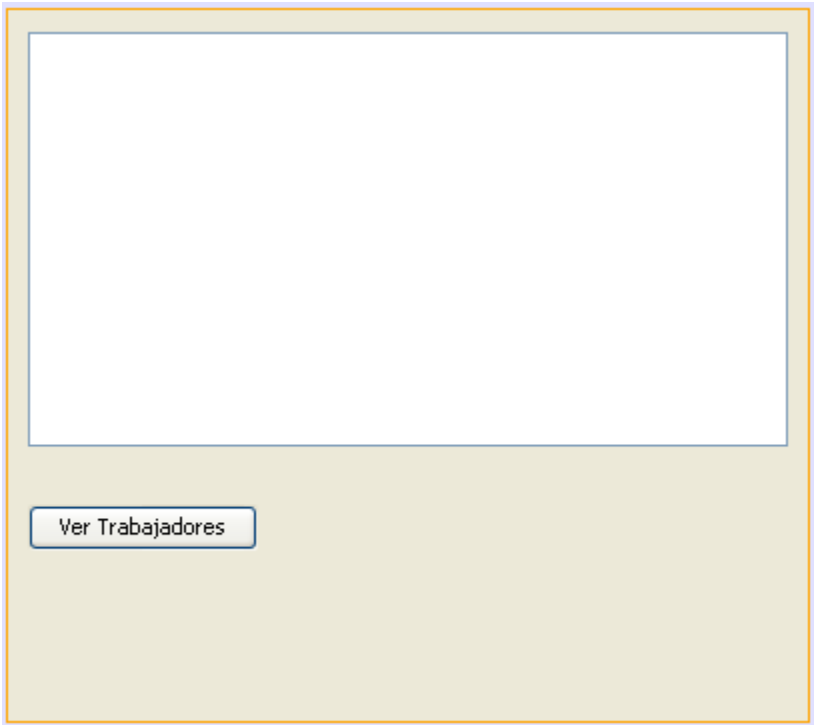
PLANTEAMIENTO

Se quiere realizar una aplicación de base de datos que manipule los datos de los trabajadores de la base de datos MANEMPSA. Esta aplicación permitirá ver el listado de trabajadores y además permitirá introducir nuevos trabajadores.

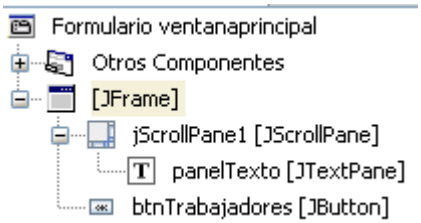
1. Entre en NetBeans. Crea un nuevo proyecto llamado *GestionTrabajadores*. Dentro de este proyecto crea un paquete principal llamado *paqueteprincipal* y dentro de él un JFrame llamado *ventanaprincipal*:



2. Añade a la ventana un JTextPane y un botón de momento:



El botón se llamará *btnTrabajadores* y el JTextPane se llamará *panelTexto*.



3. Para que este programa pueda trabajar con la base de datos MANEMPSA tendrá que prepararlo haciendo lo siguiente:

   - Crear la subcarpeta Base y copiar en ella el fichero de base de datos MANEMPSA.MDB que tiene en la carpeta Mis Documentos.

   - Añadir al programa los objetos *conexión (Connection) y sentencia (Statement)* como globales.

   - Crear el procedimiento *PrepararBaseDatos* y llamarlo desde el constructor.

   - Cerrar la conexión desde el evento *windowClosing*

   Realice estos cuatro pasos que se han indicado antes de continuar.

4. Ya se puede programar el botón *btnTrabajadores.* Se pretende que al pulsar este botón aparezca en el panel *panelTexto* el contenido de la tabla *trabajadores*.

   Para ello, en el *actionPerformed* del botón *btnTrabajadores* programe lo siguiente:

```
    private void btnTrabajadoresActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        MostrarTodos();
    }
```

5. Como puede ver, es una llamada a un método al que se le ha dado el nombre *MostrarTodos*. Este método se encargará de mostrar todos los trabajadores en el *panelTexto*.

   Programe el método *MostrarTodos* de la siguiente forma:

```
void MostrarTodos() {
   String info="";
   //cadenas para los campos
   String caddni,cadnombre,cadapell,cadsueldo,cadfecha,cadmat;

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

    r.beforeFirst();
    while (r.next()) {
       caddni=r.getString("dni");
       cadnombre=r.getString("nombre");
       cadapell=r.getString("apellidos");
       cadsueldo=r.getString("sueldo");
       cadsueldo = cadsueldo.replace(".",",");
       cadfecha=r.getString("fecha");
       cadfecha=cadfecha.substring(8,10)+"/"+cadfecha.substring(5,7)+"/"+
                cadfecha.substring(0,4);
       cadmat=r.getString("matricula");
       if (cadmat==null || cadmat.equals("")) {
           cadmat="sin coche";
       }
       info=info+caddni+"  --  "+cadnombre+" "+cadapell+"  --  "+
                 cadsueldo+"  --  "+cadfecha+"  --  "+cadmat+"\n";
    }
    panelTexto.setText(""); //vacio el panel de texto
    panelTexto.setText(info); //meto la cadena info
   } catch(Exception e) {
       JOptionPane.showMessageDialog(null,"Error al consultar la tabla trabajadores");
   }
}
```

El código de este método no debe resultarle ya desconocido.

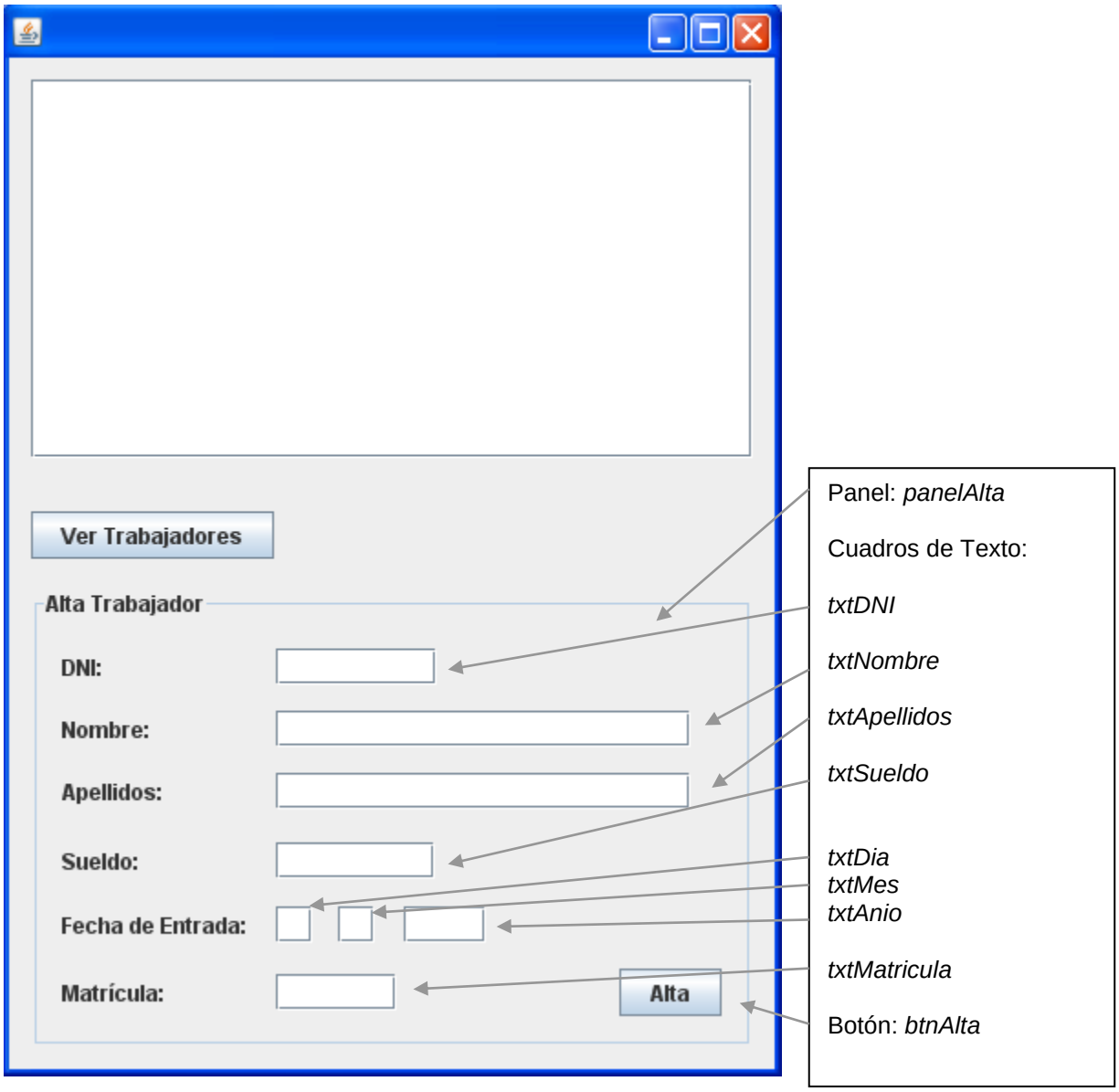
Básicamente lo que hace es ejecutar una consulta SQL que recoge todos los datos de la tabla trabajadores y luego muestra dichos datos en el panel de la ventana.

Se muestra el listado procurando que las fechas aparezcan con el formato dia-mes-año, que los sueldos aparezcan con la coma decimal y que si el campo matrícula fuera nulo o la cadena vacía “”, entonces aparezca el texto “sin coche”.

La razón por la que se ha programado este código en un método aparte llamado *MostrarTodos* en vez de hacerlo directamente en el *actionPerformed* del botón se entenderá más adelante, cuando avancemos en el ejercicio guiado.

6. Ejecute el programa y pruebe el botón *btnTrabajadores*. Compruebe que realiza su cometido.

7. Ahora que ya tenemos un programa capaz de visualizar a los trabajadores, añada los siguientes elementos a la ventana principal:



8. El objetivo de estos elementos es el siguiente:

   El usuario introducirá los datos de un nuevo trabajador en las casillas indicadas.

   Luego, al pulsar el botón de *Alta*, se introducirá en la tabla *trabajadores* los datos del nuevo trabajador y aparecerá en el panel la lista actualizada de trabajadores incluyendo al nuevo que se ha añadido.

   Para ello tendrás que programar en el botón *Alta* lo siguiente:

```
    private void btnAltaActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        String consulta="";

        try {
            consulta="insert into trabajadores values (";
            consulta+="'"+txtDNI.getText()+"',";
            consulta+="'"+txtNombre.getText()+"',";
            consulta+="'"+txtApellidos.getText()+"',";
            consulta+=txtSueldo.getText()+",";
            consulta+="#"+txtMes.getText()+"/"+txtDia.getText()+"/"+txtAnio.getText()+"#,";
            consulta+="'"+txtMatricula.getText()+"')";

            sentencia.executeUpdate(consulta);
            MostrarTodos();
        } catch(Exception e) {
            JOptionPane.showMessageDialog(null,"Error al introducir el nuevo trabajador");
        }

    }
```

Analicemos este código detenidamente.

Lo primero que hay que tener en cuenta es que se realiza una concatenación de cadenas dentro de la variable consulta.

Observa el uso de **+=** para concatenar. Ten en cuenta que es lo mismo poner esto:

```
consulta += "'"+txtDNI.getText()+"',";
```

que poner esto:

```
consulta = consulta + "'"+txtDNI.getText()+"',";
```

Solo que el uso de += acorta las instrucciones.

Si se analiza la concatenación de las cadenas, se observará que el resultado es una instrucción SQL del tipo INSERT INTO. Es decir, una instrucción SQL que permite la inserción de un nuevo registro en la tabla trabajadores.

Por ejemplo, supongamos que introducimos los siguientes valores en los cuadros de texto:

```
txtDNI →          11.111.111-A
txtNombre →       María
txtApellidos →    Ruiz
txtSueldo →       1100
txtDia →          10
txtMes →          4
txtAnio →         2001
txtMatricula →    4433RET
```

La concatenación en la variable *consulta* resultaría lo siguiente (en azul los valores de los cuadros de texto, en rosa las cadenas que se concatenan):

```
insert into trabajadores values ('11.111.111-A','María','Ruiz',1100,#4/10/2001#,'4433RET')
```

Es decir, se sigue la misma estrategia que en la hoja anterior. Se concatenan trozos de cadenas y datos introducidos por el usuario hasta conseguir una cadena con forma de instrucción SQL.

En este ejemplo, la instrucción SQL construida por concatenación es una instrucción INSERT INTO que permite introducir en la tabla trabajadores los datos de un nuevo trabajador.

El objeto que se encarga de ejecutar dentro de la base de datos la instrucción SQL recién construida es el objeto *sentencia*:

```
sentencia.executeUpdate(consulta);
```

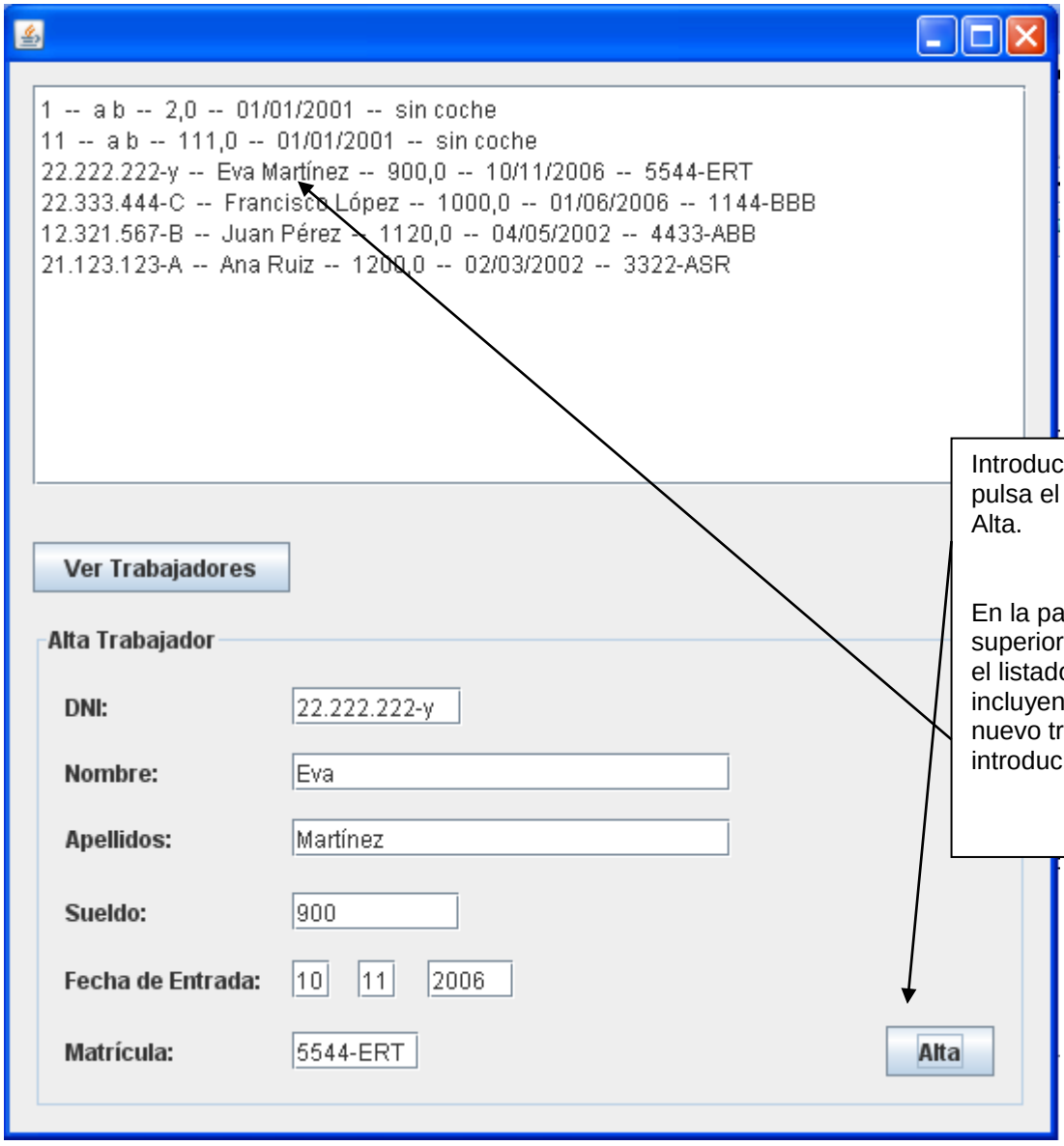
Como se puede observar, para ejecutar instrucciones SQL de acción ya no se usa el método *executeQuery*, sino que se usa el método *executeUpdate*.

Una vez ejecutada la introducción del nuevo registro en la base de datos, se llama al procedimiento *MostrarTodos* el cual se encarga de mostrar todo el contenido de la tabla trabajadores en el panel de texto. Gracias a este método, veremos como se rellena el panel y observaremos al nuevo registro recién introducido.

La razón de que se programara aparte el método *MostrarTodos* para mostrar el listado completo de trabajadores ha sido debido a la intención de llamar a este método desde otros lugares del programa. Es decir, la intención ha sido centralizar código y evitar su repetición.

Hay que tener en cuenta que todo este código está rodeado de un try ... catch para evitar cualquier error inesperado. Ten en cuenta que pueden producirse errores fácilmente si introducimos valores incorrectos en los cuadros de texto.

9. Finalmente ejecute el programa y compruebe su funcionamiento añadiendo varios trabajadores a la tabla:



Introduce datos y pulsa el botón Alta.

En la parte superior aparecerá el listado completo incluyendo al nuevo trabajador introducido.

CONCLUSIÓN

Se pueden ejecutar instrucciones SQL del tipo INSERT INTO (insertar registros) usando el objeto sentencia.

Para este tipo de instrucciones hay que usar el método *executeUpdate*.

Normalmente, será necesario construir una cadena de consulta a través de la concatenación de subcadenas y datos introducidos por el usuario en cuadros de texto.

Este tipo de instrucciones SQL no devuelven ningún *ResultSet*, ya que no extraen datos de las tablas, sino que modifican el contenido de éstas.