EJERCICIO GUIADO. JAVA. ACCESO A BASE DE DATOS

Tablas (JTable)

Como se ha estudiado en las hojas guiadas anteriores, se pueden extraer datos de la base de datos a través de consultas SQL de tipo SELECT. Los datos extraídos se almacenan en objetos de tipo *ResultSet*.

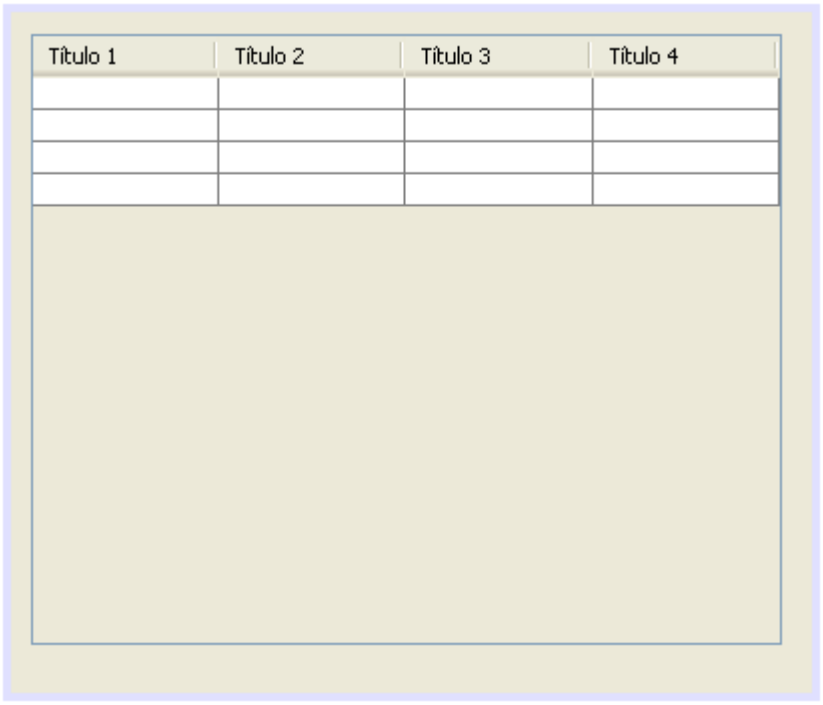
Luego, solo hay que analizar el contenido del objeto *ResultSet* para extraer los datos que contiene y trabajar con ellos.

En las hojas anteriores hemos extraído los datos del *ResultSet* y los hemos presentado en un JOptionPane o en un JTextPane. Sin embargo, una mejor opción para presentar el contenido de un *ResultSet* es usar objetos del tipo JTable, ya que estos objetos tienen forma de tabla.

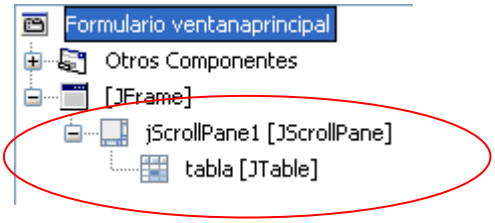
En esta hoja guiada se estudiarán los objetos JTable (sin tener en cuenta a las bases de datos) Una vez que entendamos el funcionamiento de los objetos JTable, los usaremos en posteriores hojas guiadas para presentar dentro de ellos el contenido de consultas SQL.

EJERCICIO GUIADO Nº 1

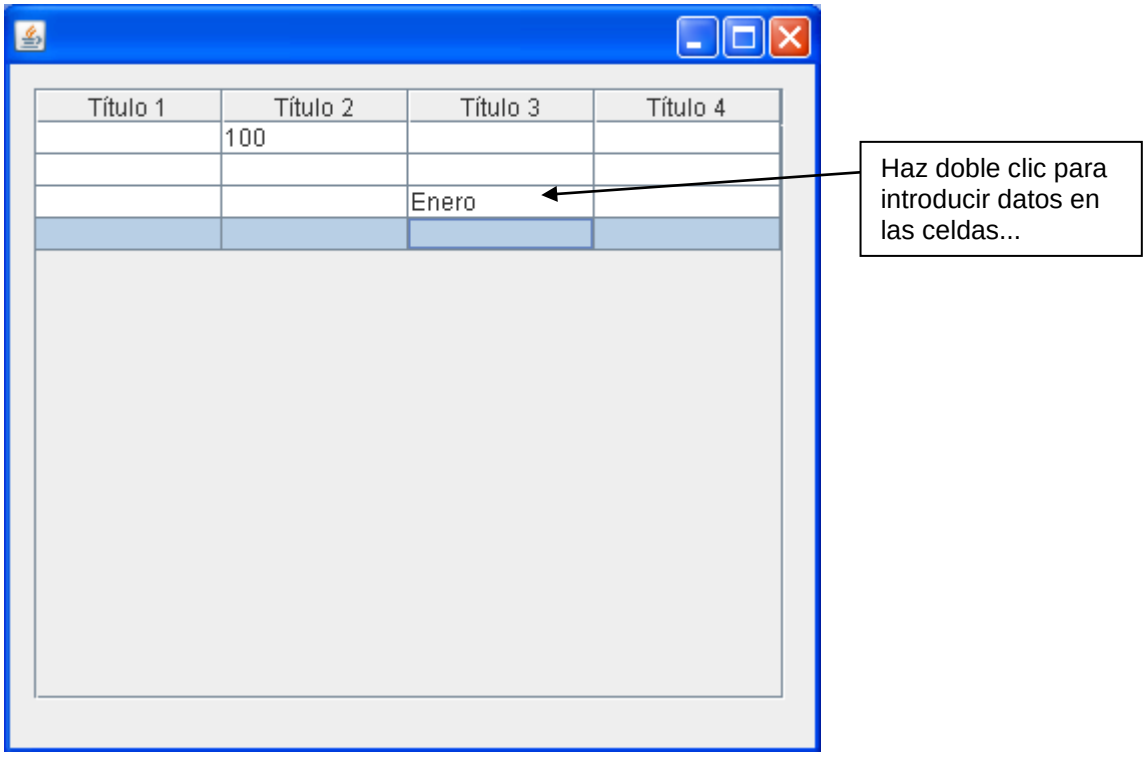
1. Crear un nuevo proyecto en NetBeans. En la ventana principal de dicho proyecto agregue un objeto *JTable*:



El nombre del objeto *JTable* será simplemente *tabla*. Debes tener en cuenta que los objetos *JTable* se añaden siempre dentro de un panel de desplazamiento *JScrollPane*:



2. El objeto *tabla* que se acaba de introducir tiene por defecto cuatro columnas con los nombres “Título 1”, “Título 2”, “Título 3” y “Título 4”, y contiene cuatro filas vacías. Puede ejecutar el programa para ver el funcionamiento del objeto tabla. Pruebe a introducir algún dato en las celdas de la tabla...



3. Aprenderemos ahora a configurar determinados aspectos de la tabla que vamos a usar. Para ello añada al constructor una llamada a un método *PreparарTabla* que programaremos a continuación.

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

4. Programa ahora el método *PreparарTabla* de la siguiente forma:

```
DefaultTableModel m;

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

void PrepararTabla() {

    String titulos[] = {"Nombre","Apellidos","Teléfono","Edad","Ciudad"};
    m = new DefaultTableModel(null,titulos);
    tabla.setModel(m);

}
```

La primera línea del código define un array de String con los títulos de la tabla, es decir, con las columnas de la tabla.

A continuación, se construye un objeto del tipo *DefaultTableModel*, o dicho de otra manera, un *modelo* de tabla. El objeto *m* está declarado como variable global.

MODELOS (recordatorio)

Hay que recordar que existen objetos en java que contienen un objeto *modelo*, encargado de contener los datos del objeto. Un ejemplo de ello son las listas y los combos (cuadros desplegables)

Para definir los datos contenidos en el objeto, primero había que definir el *modelo* y luego asignar el modelo al objeto.

El caso de las tablas es igual. Para introducir datos en la tabla primero hay que configurar su objeto *modelo* (que será de la clase *DefaultTableModel*) y luego asignárselo a la tabla.

En el código, se define como global un objeto llamado *m*, de tipo *DefaultTablaModel*. Luego, al construir el objeto se deben indicar dos parámetros: *null* y el vector de títulos de columnas:

```
DefaultTableModel m = new DefaultTableModel(null,titulos);
```

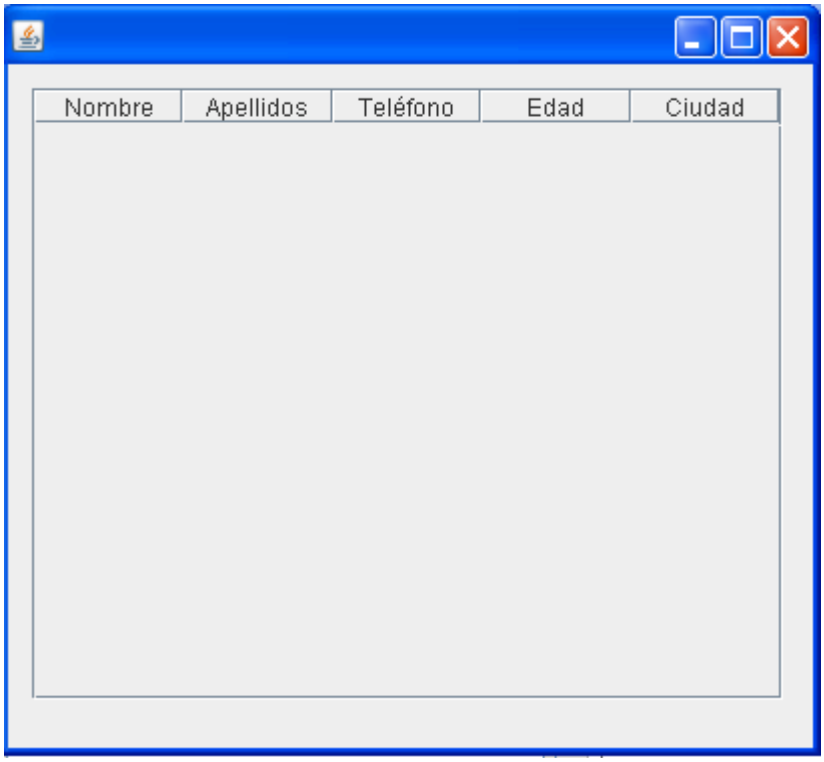
El valor null hace que la tabla aparezca vacía en un principio, mientras que el vector títulos define las columnas que tendrá la tabla.

Una vez construido de esta forma el objeto *modelo* m, este se asigna al objeto tabla:

```
tabla.setModel(m);
```

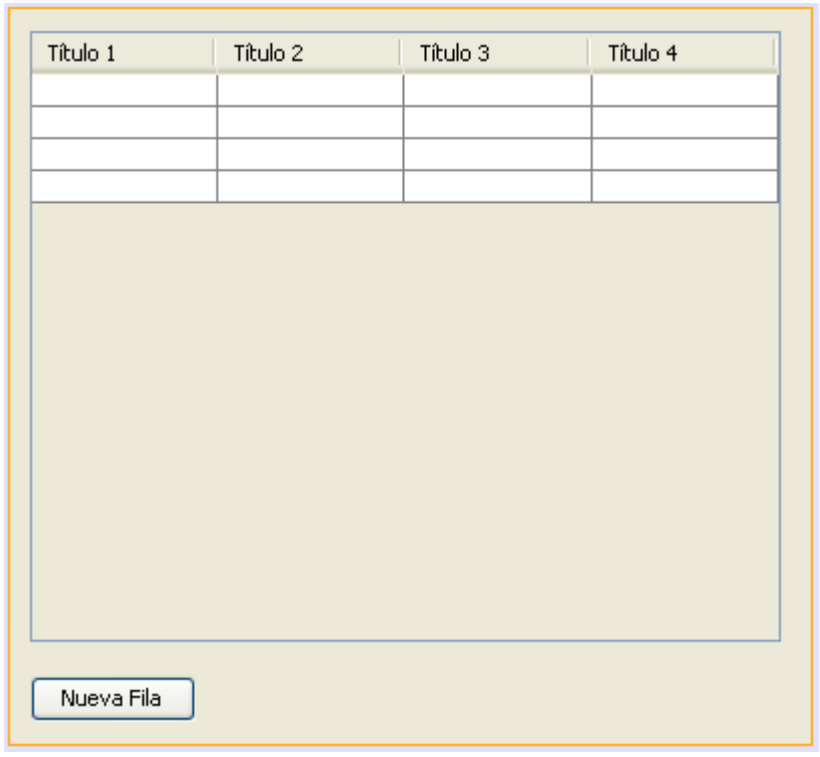
La razón por la que se ha declarado el objeto *modelo* m como global es que será usado en otros métodos del programa.

5. Ejecuta el programa y observa el resultado:



Como puedes observar, se ha creado una tabla vacía con cinco columnas correspondientes al vector de títulos.

6. Vamos a añadir un botón a la ventana con el texto *Nueva Fila*, al cual llamaremos *btnNueva*:



7. Dentro del botón *btnNueva* programa lo siguiente

```
    private void btnNuevaActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        m=(DefaultTableModel) tabla.getModel();
        String filavacia[]=new String[5];
        m.addRow(filavacia);
    }
```

Este código añadirá una nueva fila en blanco a la tabla. Estudiemos cada línea:

La primera línea recoge el modelo de la tabla a través del método *getModel*. El modelo es recogido en la variable global *m* que creamos anteriormente. Observa como es necesario realizar un cast (en rojo) a la hora de asignar el modelo a la variable m:

```
m = (DefaultTableModel) tabla.getModel();
```

El resultado de esta instrucción es que volvemos a tener disponible el *modelo* de la tabla en la variable *m*, y por tanto, podemos manipular los datos contenidos en la tabla a través de esta variable.

Por ejemplo, se usará el modelo *m* para añadir una fila en blanco en la tabla. Para ello, basta con crear un vector de cadenas de 5 elementos (ya que hay cinco columnas):

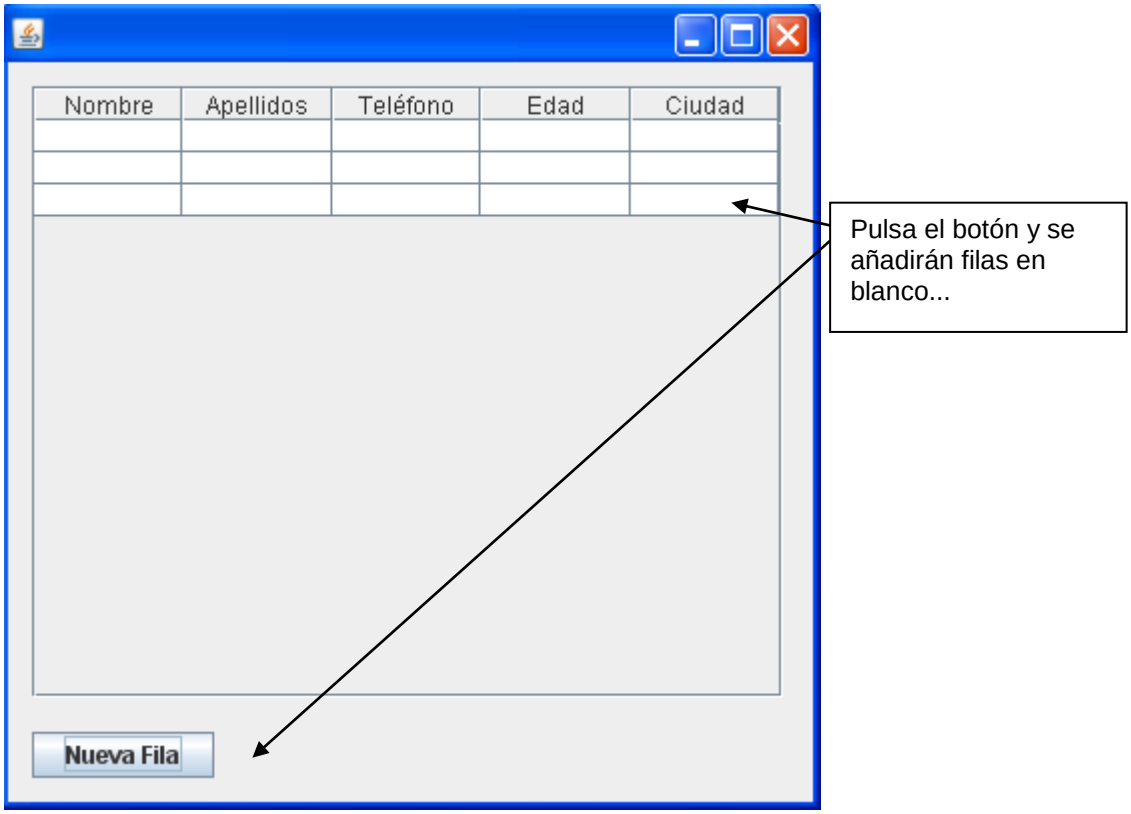
```
String filavacia[]=new String[5];
```

Y luego, a través del método *addRow* (añadir fila), añadir una fila correspondiente a los valores del vector:

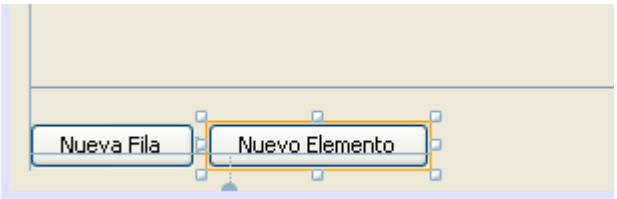
```
m.addRow(filavacia);
```

Debido a que el vector está vacío, la fila que se añade está vacía también. Si el vector contuviera algún dato, estos datos aparecerían en la fila que se añade (esto se verá a continuación)

7. Ejecuta el programa y pulsa varias veces el botón *Nueva Fila*. Observarás como se van añadiendo filas vacías a la tabla.



8. Para entender bien el funcionamiento de la inserción de filas en la tabla, añada otro botón llamado *btnNuevoElemento* con el texto *Nuevo Elemento*:

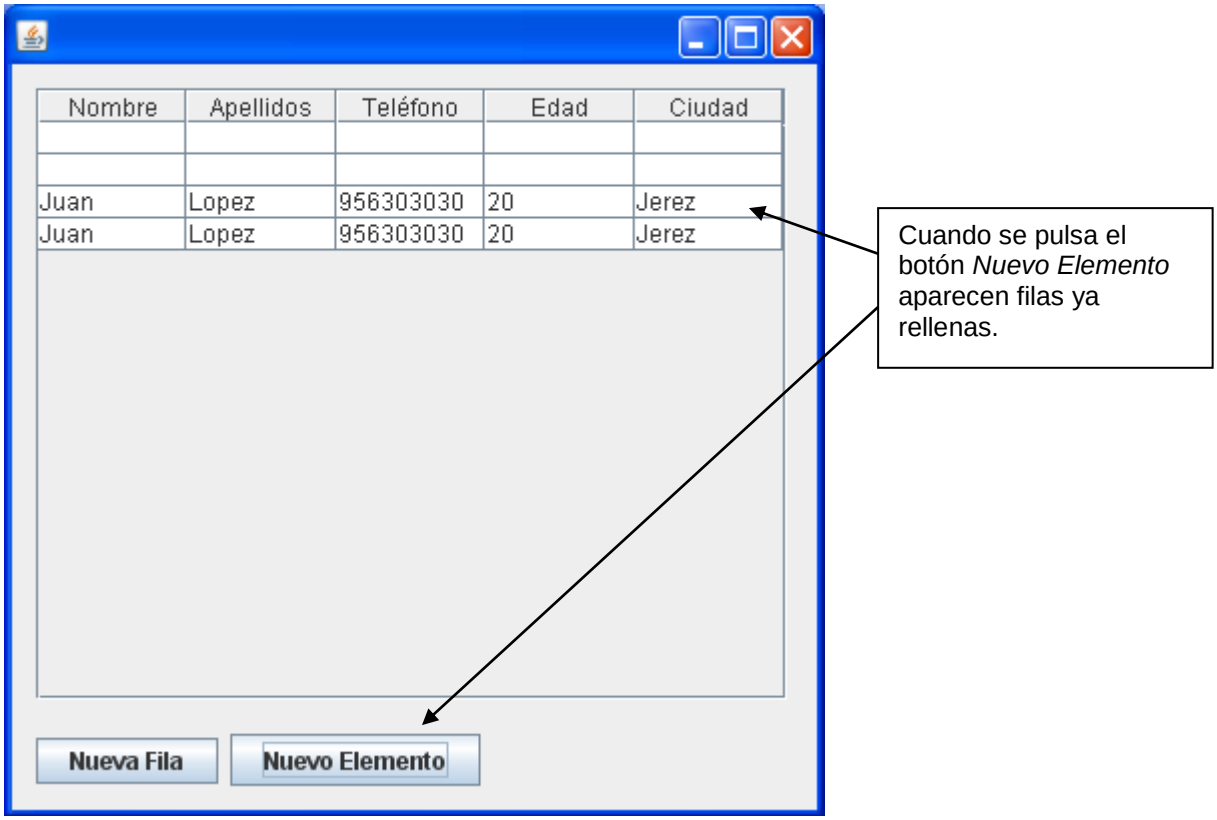


9. En este botón programe lo siguiente:

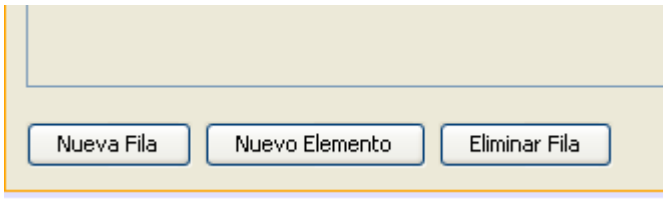
```
    private void btnNuevoElementoActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        m=(DefaultTableModel) tabla.getModel();
        String filaelemento[]={"Juan","Lopez","956303030","20","Jerez"};
        m.addRow(filaelemento);
    }
```

Puedes observar que el código es parecido al anterior, solo que en este caso se añade una fila correspondiente a un vector relleno de datos. Esto quiere decir que cuando se añada esta fila al modelo aparecerá con los datos del vector.

10. Ejecuta el programa y comprueba su funcionamiento, observarás que al pulsar el nuevo botón aparecen filas rellenas.



11. Sigamos experimentando con la tabla. Ahora añadiremos un botón *btnEliminar* con el texto “Eliminar Fila”:



12. Este botón eliminará la fila que esté seleccionada en ese momento. Para ello programe en este botón el siguiente código:

```
    private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        int filasel = tabla.getSelectedRow();
        if (filasel==-1) {
            JOptionPane.showMessageDialog(null,"No hay ninguna fila seleccionada");
        } else {
            m=(DefaultTableModel) tabla.getModel();
            m.removeRow(filasel);
        }
    }
```

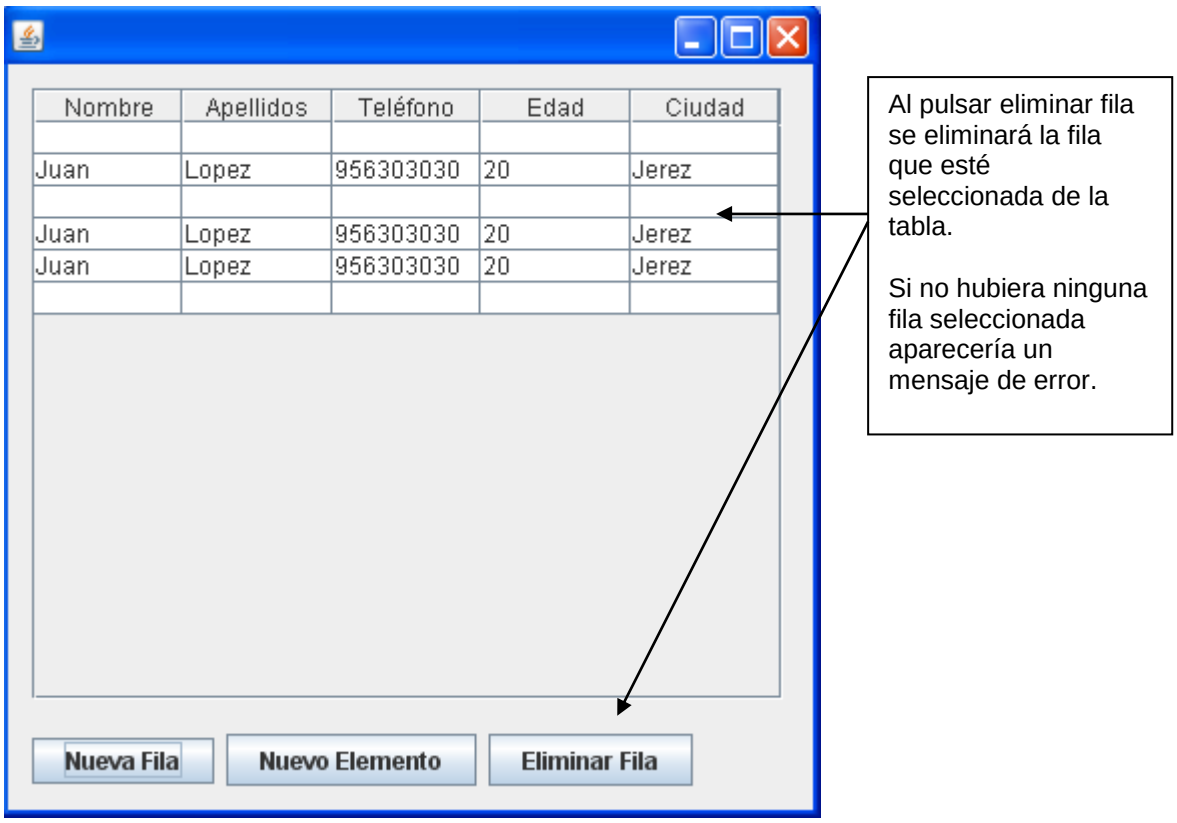
Estudiemos este código. En primer lugar usamos un método que poseen los objetos JTable llamado *getSelectedRow* que nos dice el número de la fila que está seleccionada en este momento.

Si no hubiera ninguna fila seleccionada, el método *getSelectedRow* devuelve el valor –1. Esto lo usamos en el if que viene a continuación para mostrar un mensaje de error diciendo que es necesario seleccionar la fila que se va a borrar.

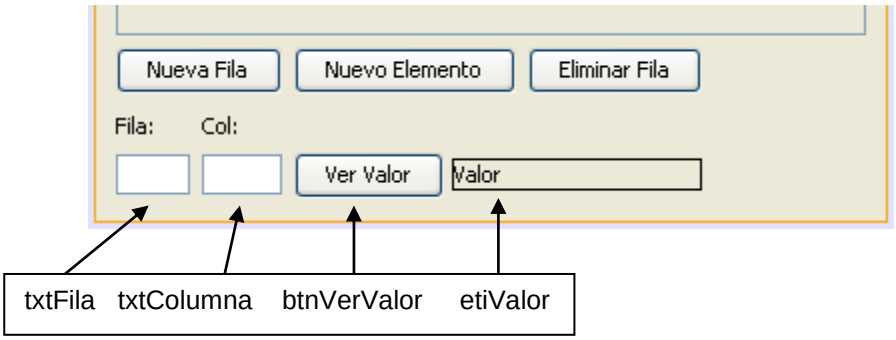
En el caso de que haya alguna fila seleccionada (es decir, que no se haya devuelto el valor –1) entonces efectuamos el borrado.

Para borrar recogemos el modelo de la tabla y usamos un método llamado *removeRow* para borrar la fila seleccionada (la cual se pasa como parámetro)

13. Ejecuta el programa y añade varias filas. Luego prueba a seleccionar alguna y pulsar el botón *Eliminar*. Observa también lo que sucede cuando intentas eliminar una fila cuando no hay ninguna seleccionada.



14. Sigamos aprendiendo nuevas cualidades de las tablas. Añada lo siguiente a su ventana:



15. Lo que se pretende es lo siguiente: El usuario introducirá un valor en la fila y la columna, por ejemplo: Fila 3, columna 4, y al pulsarse el botón *Ver Valor* aparecerá en la etiqueta *Valor* el contenido de la casilla situada en la posición 3, 4 (fila 3 columna 4)

    Así pues programe lo siguiente en el botón *Ver Valor*:

```
    private void btnVerValorActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        int fila;
        int col;
        String valor;

        try {
            //recojo los valores de los cuadros de texto fila y columna
            fila = Integer.parseInt(txtFila.getText());
            col = Integer.parseInt(txtColumna.getText());

            m=(DefaultTableModel) tabla.getModel();
            valor=(String) m.getValueAt(fila,col);
            etiValor.setText(valor);
        } catch(Exception e) {
            JOptionPane.showMessageDialog(null,"Error al recoger el valor");
        }
    }
```

Estudiemos el código.

Primero, se recogen los valores introducidos en los cuadros de texto de la fila y columna y se convierten a enteros:

```
fila = Integer.parseInt(txtFila.getText());
col = Integer.parseInt(txtColumna.getText());
```

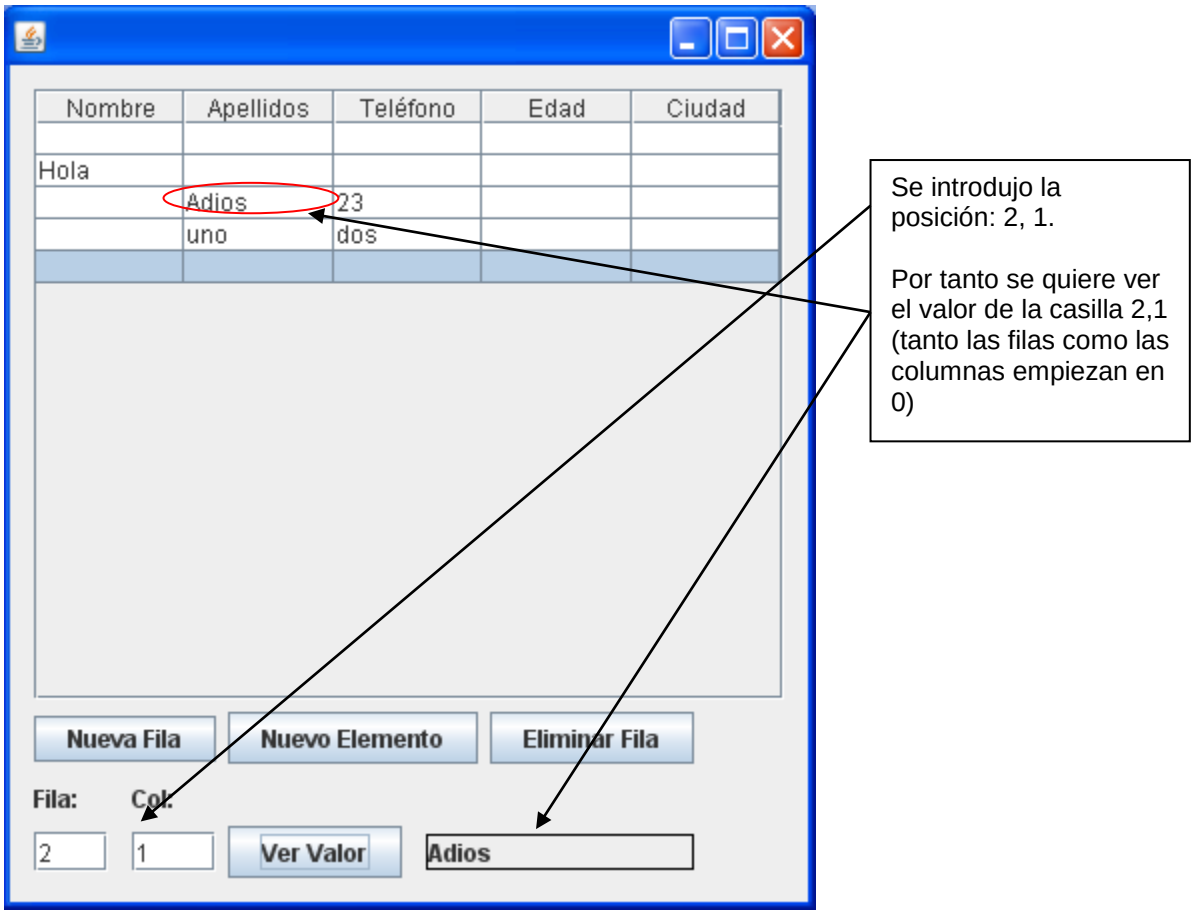
Luego, se extrae el modelo de la tabla y se usa un método llamado *getValueAt*. Este método permite extraer el valor de una casilla de la tabla, indicando la fila y columna de esa celda. El resultado lo hemos almacenado en una variable *valor* de tipo cadena. Es necesario realizar un cast (en rojo):

```
valor = (String) m.getValueAt(fila,col);
```

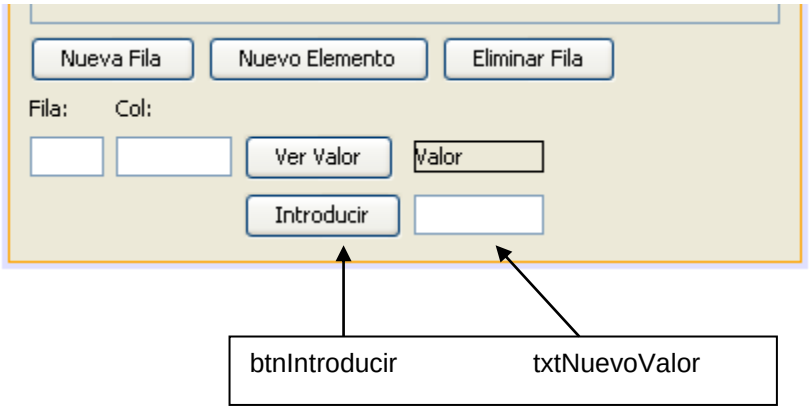
Finalmente, una vez extraído el valor de la celda de la posición *fila, col* este se coloca en la etiqueta *etiValor*.

Este código es susceptible de dar error, por ejemplo si el usuario introduce unos valores de fila y columna incorrectos. Por eso ha sido incluido dentro de un try...catch.

16. Ejecuta el programa y añade varias filas. Introduce algunos valores en las filas. Luego prueba a introducir un valor de fila y columna y pulsa el botón *Ver Valor*:



17. Sigamos experimentando. Añade lo siguiente a la ventana:



18. Se pretende lo siguiente. El usuario introducirá un valor de fila y de columna en los cuadros de texto correspondientes. Luego, el usuario introducirá un valor en el cuadro de texto *txtNuevoValor*. Finalmente pulsará el botón *btnIntroducir*.

    El resultado será que se introduce el valor escrito en la posición de la tabla indicada por la fila y columna introducida.

    Para hacer esto, programe lo siguiente dentro del botón *btnIntroducir*:

```
    private void btnIntroducirActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        int fila;
        int col;
        String valor;

        try {
            fila = Integer.parseInt(txtFila.getText());
            col = Integer.parseInt(txtColumna.getText());
            valor = txtNuevoValor.getText();

            m=(DefaultTableModel) tabla.getModel();
            m.setValueAt(valor,fila,col);
        } catch(Exception e) {
            JOptionPane.showMessageDialog(null,"Error al introducir el valor");
        }
    }
```

Estudiemos el código detenidamente.

Lo primero que se hace es extraer de los cuadros de texto la fila, columna y el valor. La fila y columna son convertidas a enteros.

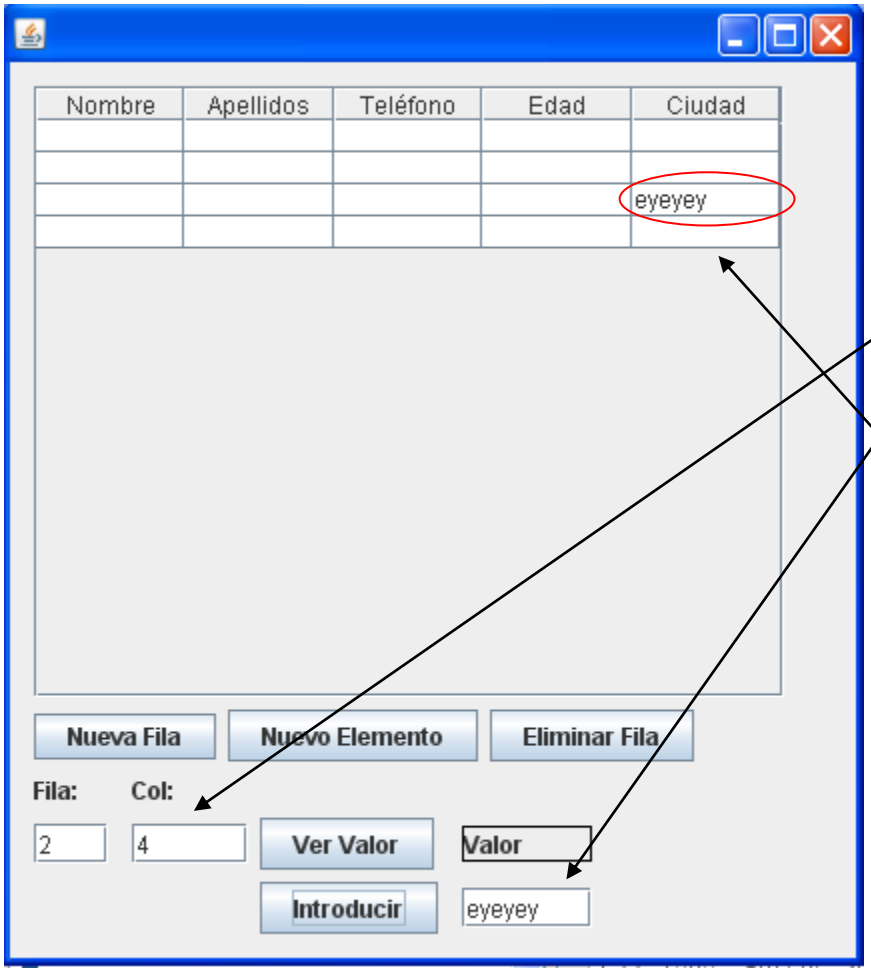
```
fila = Integer.parseInt(txtFila.getText());
col = Integer.parseInt(txtColumna.getText());
valor = txtNuevoValor.getText();
```

Luego se extrae (como siempre) el modelo de la tabla y se usa un método suyo llamado *setValueAt* que permite introducir un valor en una celda de la tabla. Concretamente, se introduce el valor escrito por el usuario en la celda con fila y columna indicadas:

```
m=(DefaultTableModel) tabla.getModel();
m.setValueAt(valor,fila,col);
```

Este código es susceptible de tener errores de ejecución (por ejemplo que el usuario introduzca un valor equivocado en la fila y columna) por eso está rodeado de un try...catch.

19. Ejecuta el programa. Añade varias filas en blanco. Luego introduce un valor para la fila y la columna y escribe un valor que quieras introducir. Al pulsar el botón *Introducir* dicho valor aparecerá en la celda correspondiente:



Introduce una fila y columna.

Luego introduce un dato.

Y al pulsar el botón *Introducir* el resultado es que el dato se introduce en la celda indicada por la fila y columna.

Recuerda que las filas y columnas empiezan a numerarse por 0 en un JTable.

CONCLUSIÓN

Existe una clase llamada *JTable* que permite crear objetos con forma de tabla.

Estos objetos son ideales para mostrar el contenido de las tablas de una base de datos, aunque pueden ser usados de forma independiente también para mostrar cualquier tipo de datos.

Los objetos *JTable* contienen un modelo de datos de tipo *DefaultTableModel*.

Cada vez que se quiera trabajar con una tabla será necesario construir su modelo y asignárselo a la tabla.

Cada vez que se quiera modificar de alguna forma los datos de una tabla, será necesario acceder a su modelo y trabajar con él.

A través del modelo de una tabla podemos añadir nuevas filas a la tabla, eliminar filas, extraer datos de las celdas de la tabla o añadir datos a las celdas de la tabla.