EJERCICIO GUIADO. JAVA. ACCESO A BASE DE DATOS

Ordenación y cálculos

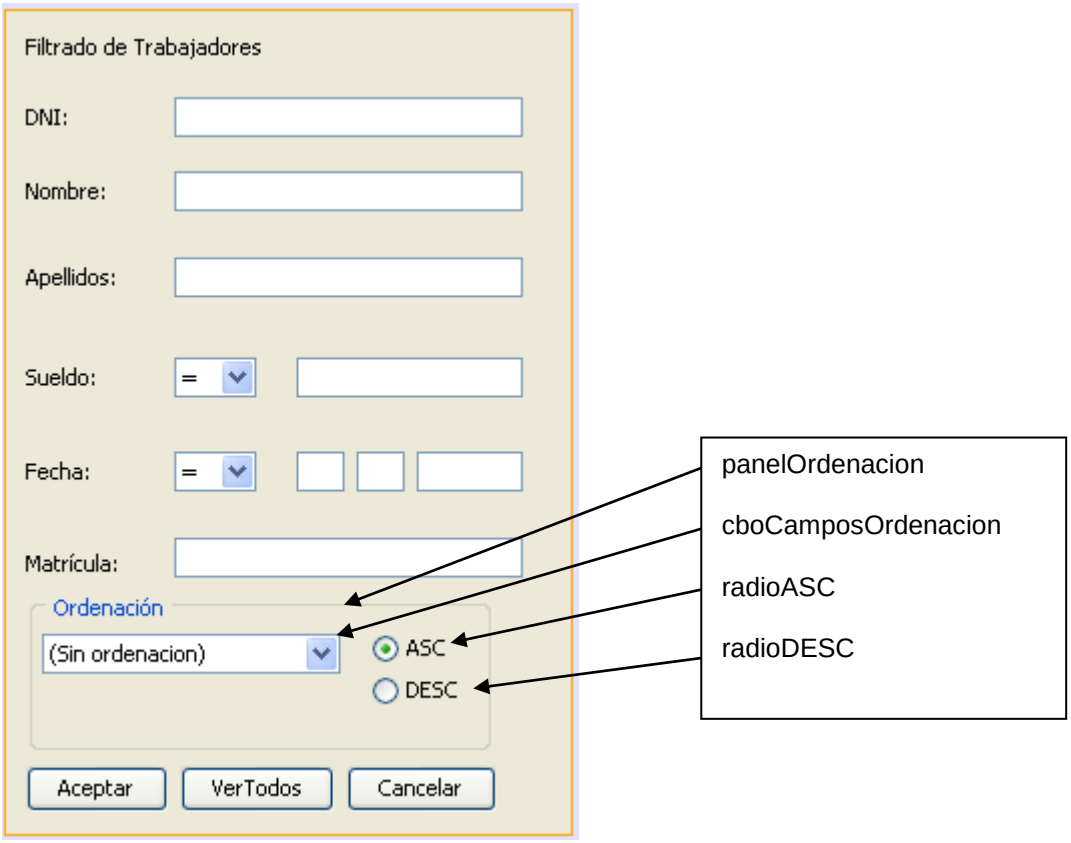
Se van a añadir ciertas mejoras al programa que se viene realizando en las últimas hojas guiadas. En esta hoja programaremos ciertas opciones de ordenación así como realizaremos determinados cálculos sobre la tabla trabajadores.

EJERCICIO GUIADO Nº 1

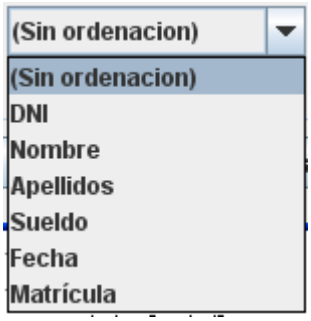
1. Abrir la aplicación de la hoja guiada anterior.

2. Vamos a modificar el cuadro de diálogo de filtrado de forma que no solo sea capaz de filtrar, sino que también permita que el listado que se ha filtrado aparezca ordenado según un campo.

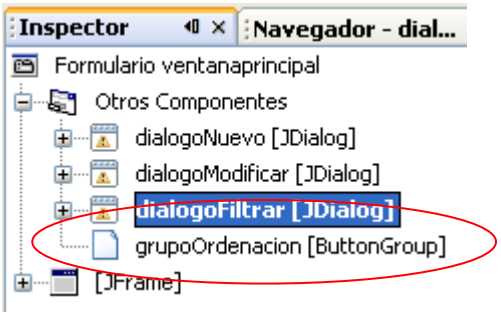
   Para ello, añada lo siguiente al diálogo de filtrado:



NOTA Nº1: El combo *cboCamposOrdenacion* contendrá los siguientes elementos: *(Sin Ordenación), DNI, Nombre, Apellidos, Sueldo, Fecha, Matricula*:



NOTA Nº2: Es necesario añadir un *ButtonGroup* al programa, al que llamaremos *grupoOrdenacion*. Los botones de opción *radioASC y radioDESC* deben pertenecer a dicho grupo.



3. La idea es la siguiente. Cuando el usuario quiera hacer un filtrado podrá indicar que el listado aparezca ordenado por algún campo, seleccionándolo en el combo de ordenación. También puede indicar la forma de ordenación (ascendente o descendente) a través de los botones de opción.

   Al pulsar Aceptar el listado de trabajadores no solo saldrá filtrado, sino que también saldrá ordenado.

   Para ello tendremos que realizar modificaciones en el código del botón *Aceptar*. Añade el siguiente código en el *actionPerformed* del botón *Aceptar* (justo antes de la ejecución de la consulta):

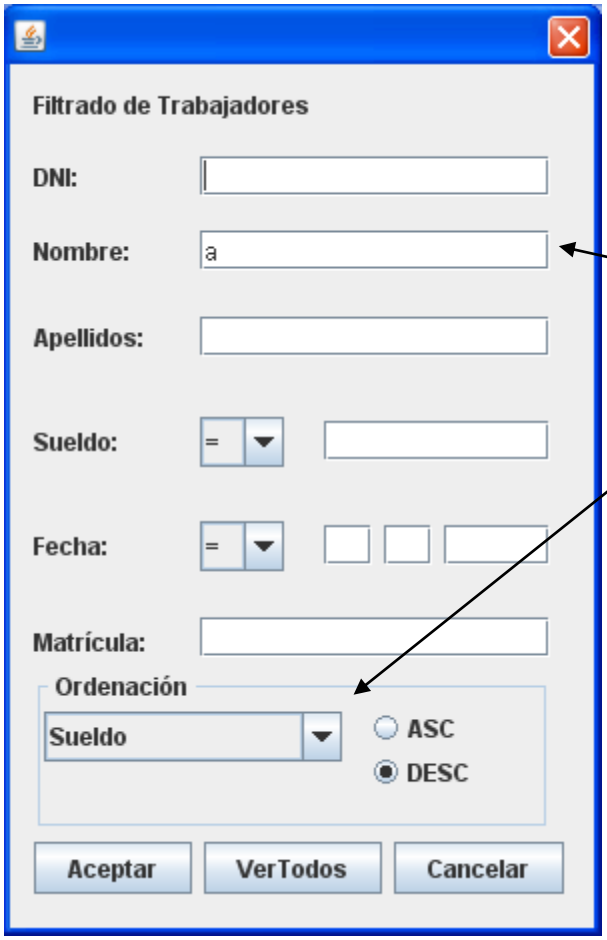
```
//Ordenación
String campoorden = (String)cboCamposOrdenacion.getSelectedItem();
if (!campoorden.equals("(Sin ordenacion)")) {
    if (radioASC.isSelected()) {
        sql+=" order by "+campoorden+" ASC";
    } else {
        sql+=" order by "+campoorden+" DESC";
    }
}

//Ejecutamos la consulta creada
ResultSet r = sentencia.executeQuery(sql);

String titulos[] = {"DNI","Nombre","Apellidos","Sueldo","Fecha","Ma
m=new DefaultTableModel(null,titulos);
```

   Estudiemos el código. Lo primero que se hace es extraer el valor del combo desplegable de los campos de ordenación.

   Si este combo no contiene el valor "*(Sin ordenación)*" significará que se desea realizar una ordenación por el campo seleccionado. Así pues se concatena a la variable *sql* la cláusula *order by* con el campo que se ha seleccionado en el combo.

   Se controla también la forma de ordenación teniendo en cuenta el botón de opción que esté activado, y según esto, se añade la cadena "ASC" o "DESC" para ordenar ascendentemente o descendentemente.

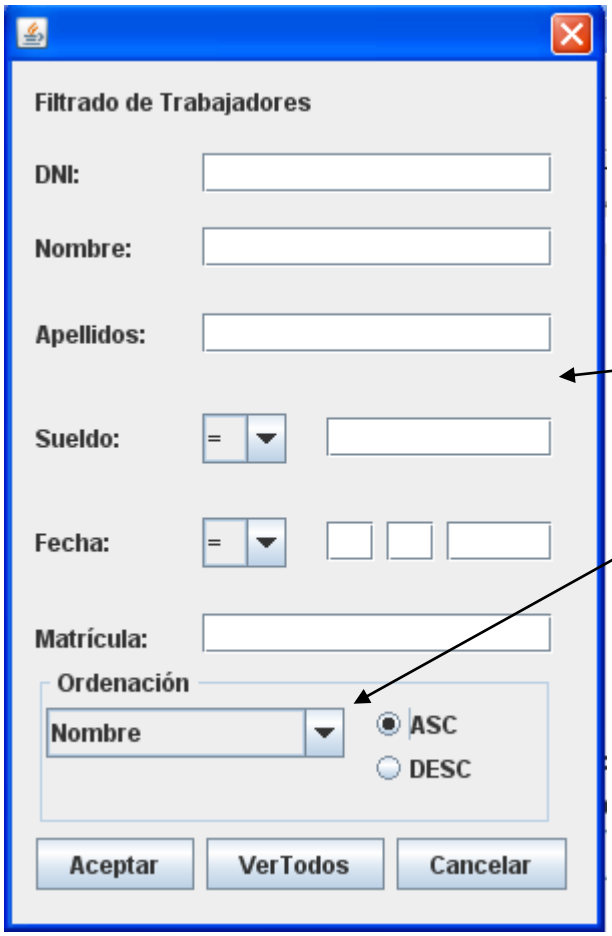
4. Ejecuta el programa y prueba a hacer un filtrado eligiendo un campo para ordenar y observa el resultado:



En este ejemplo se pretende visualizar a todos los trabajadores que contengan una “a” en su nombre...

... y el listado saldrá ordenado por el campo sueldo descendentemente.

5. Una ventaja de esto es que se puede mostrar el listado completo de trabajadores ordenado por el campo que se quiera siempre y cuando no se indique ninguna condición:

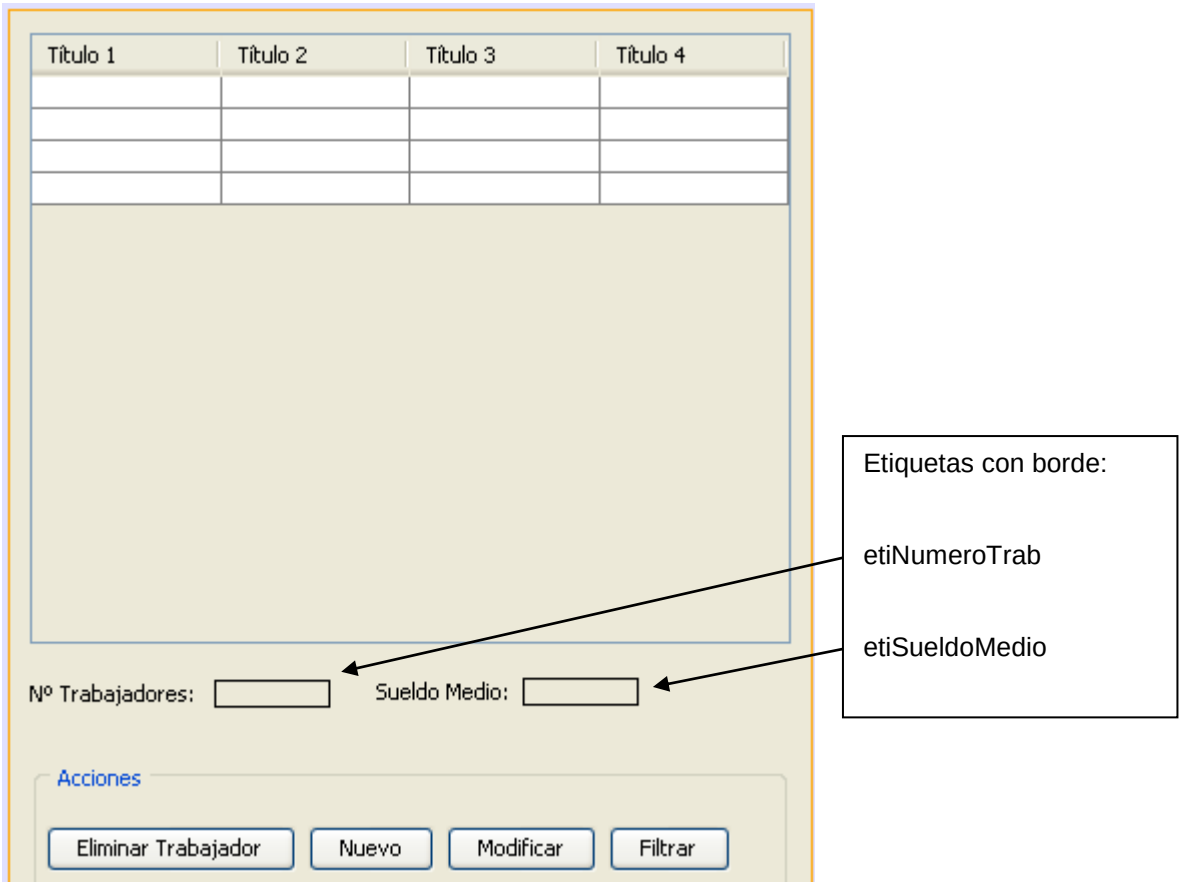


En este ejemplo aparecerían todos los trabajadores, ya que no se ha indicado ninguna condición.

... y el listado saldrá ordenado por el nombre ascendentemente.

6. Ahora vamos a añadir una nueva mejora al programa. La idea es que se visualice junto al JTable de trabajadores los siguientes datos: el número de trabajadores que se muestra y la media de los sueldos.

   Para ello, añade las siguientes modificaciones en el diseño de la ventana principal:



7. Ahora programaremos un método que sea capaz de calcular el número de trabajadores y el sueldo medio a partir del contenido de un *ResultSet*.

   Este método recibirá un *ResultSet* como parámetro, y lo que hará será analizar el contenido de este *ResultSet* para calcular y luego mostrar el número de trabajadores y el sueldo medio.

   Accede al código de tu programa y añade el siguiente procedimiento:

```
void HacerCalculos(ResultSet r) {

    int ntrab=0; //numero de trabajadores
    double ssueldo=0; //suma de sueldos
    double media;  //media

    try {
        r.beforeFirst();
        while(r.next()) {
            ntrab++;
            ssueldo=ssueldo+r.getDouble("sueldo");
        }
        media=ssueldo/ntrab;

        etiNumeroTrab.setText(""+ntrab);
        etiSueldoMedio.setText(""+media);

    } catch(Exception e) {
        JOptionPane.showMessageDialog(null,"Error al realizar cálculos");
    }
}
```

Este código básicamente toma el *ResultSet* pasado como parámetro y lo recorre aumentando en uno la variable *ntrab* cada vez que se pasa al siguiente trabajador.

Por otro lado se calcula la suma de los sueldos de los trabajadores del *ResultSet*.

Una vez terminado el bucle solo hay que dividir la suma de los sueldos entre el número de trabajadores para obtener la media, y estos datos se colocan en las etiquetas correspondientes.

8. Interesa que cada vez que se muestren todos los trabajadores en el JTable se realicen estos cálculos, así pues debe añadir la siguiente línea en el método *MostrarTrabajadores*.

```
void MostrarTrabajadores() {
    String fecha;
    try {
        //Extraer datos de la tabla y almacenarlos en el resultset
        ResultSet r = sentencia.executeQuery("select * from trabajadores");
        //Crear el modelo y definir la cabecera de la tabla
        String titulos[] = {"DNI","Nombre","Apellidos","Sueldo","Fecha","Matrícula"};
        m=new DefaultTableModel(null,titulos);

        String fila[] = new String[6];
        while(r.next()) {
            fila[0]=r.getString("DNI");
            fila[1]=r.getString("Nombre");
            fila[2]=r.getString("Apellidos");
            fila[3]=r.getString("Sueldo").replace(".",",");
            fecha=r.getString("Fecha");
            fecha=fecha.substring(8,10)+"/"+fecha.substring(5,7)+"/"+fecha.substring(0,4);
            fila[4]=fecha;
            fila[5]=r.getString("Matricula");
            m.addRow(fila);
        }
        tabla.setModel(m);
        HacerCalculos(r);

    } catch (Exception e) {
        JOptionPane.showMessageDialog(null,"Error al extraer los datos de la tabla");
    }
}
```

Esa línea es una llamada al método que acabamos de programar, y hará que después de que se presenten en el JTable todos los trabajadores se rellenen las etiquetas de los cálculos.

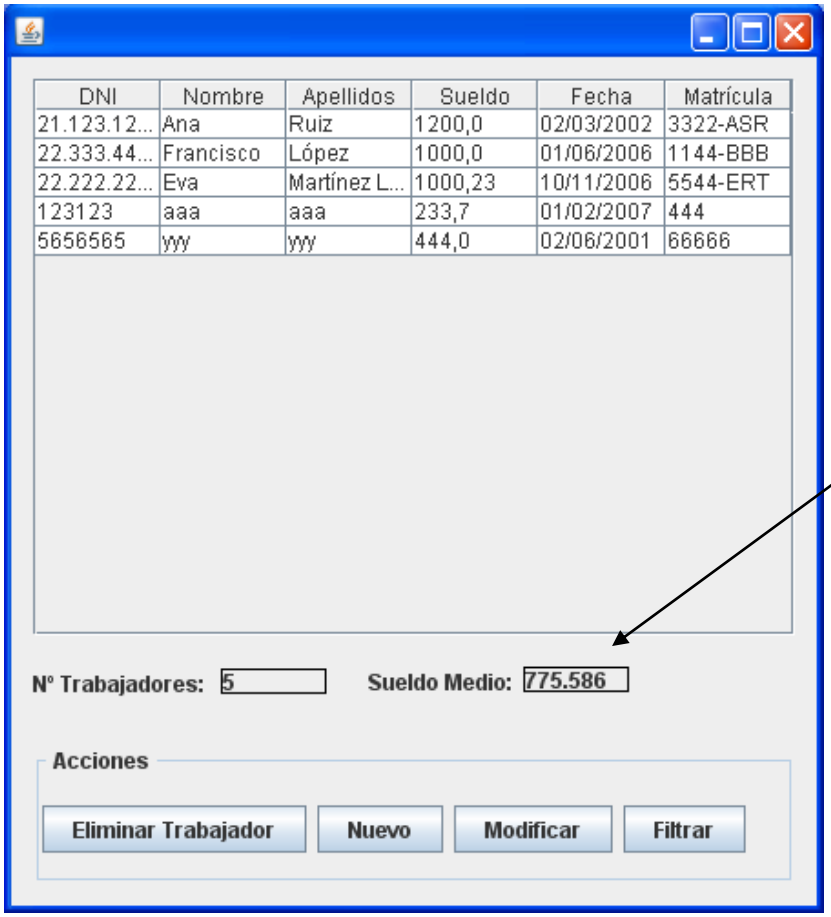
9. También interesa que cuando se realice un filtrado aparezca el número de trabajadores y el sueldo medio correspondiente al listado que se acaba de filtrar. Para ello debes añadir la siguiente línea al código del botón *Aceptar* del cuadro de diálogo de filtrado:

```
                m.addRow(fila);
            }
            tabla.setModel(m);
            HacerCalculos(r);
            dialogoFiltrar.dispose();

        } catch(Exception e) {
            JOptionPane.showMessageDialog(null,"Error al filtrar la tabla trabajadores"+e);
        }

    }
```

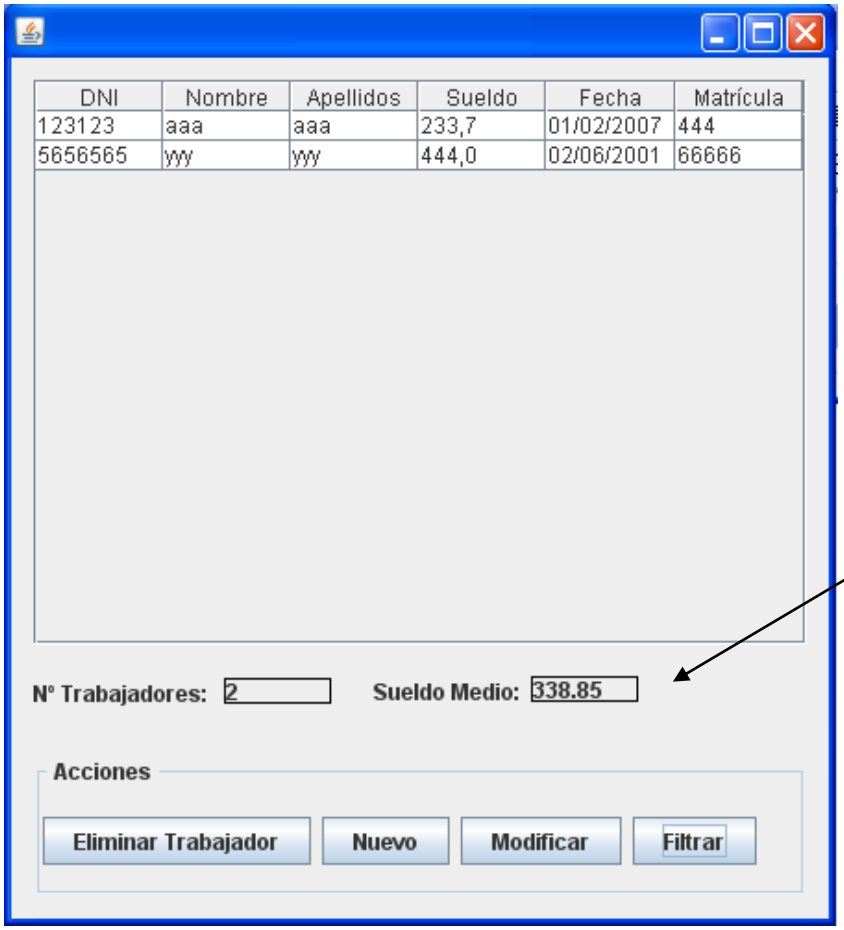
De nuevo se realiza la llamada al método de los cálculos justamente después de colocar en la tabla el listado de trabajadores, en este caso filtrado.

10. Ejecuta el programa y observa como al iniciar el programa ya aparecen en las etiquetas los cálculos (ya que el constructor llama al método *MostrarTrabajadores* que a su vez llama al método *HacerCalculos*)

    También puedes comprobar como al hacer un filtrado se muestran los cálculos correspondientes al listado filtrado, ya que en el botón *Aceptar* del dialogo de filtrado se hace una llamada al método *HacerCalculos*.



Al iniciarse el programa aparece el listado completo de trabajadores y se calcula el número de trabajadores y el sueldo medio...



Si se hace un filtrado (en este ejemplo se visualizan los trabajadores con menos de 1000 de sueldo) los cálculos se corresponderán con el listado filtrado...

CONCLUSIÓN

Todo programa de gestión de datos de una tabla debería tener opciones de ordenación para mostrar los datos de la tabla ordenados como el usuario quiera.

La opción de ordenación puede estar situada junto a las opciones de filtrado para permitir de esta manera que los filtrados aparezcan ordenados según le interese al usuario.

Suele ser habitual que al lado del JTable aparezcan ciertos cálculos estadísticos relativos a la tabla.