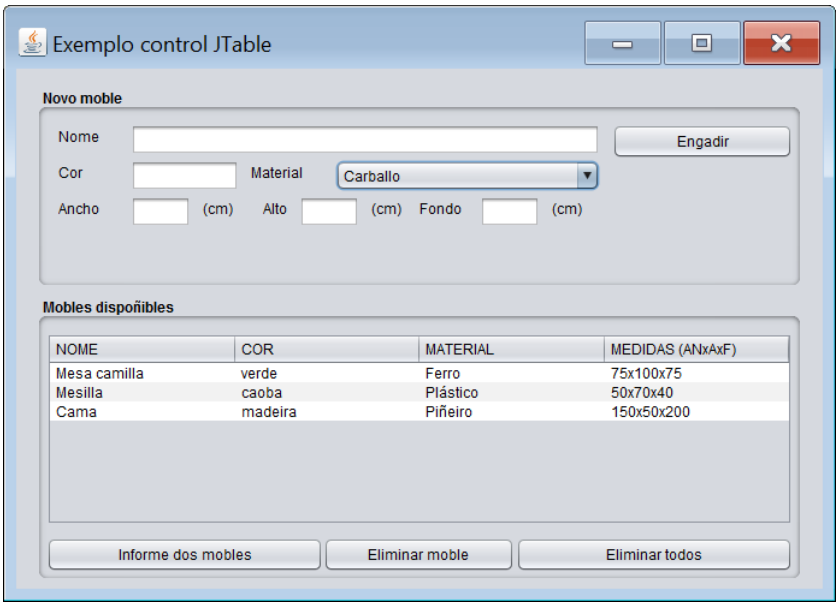
1. Compoñente Táboa (JTable)

Unha táboa é un compoñente gráfico composto por filas e columnas de celas, sendo nestas últimas nas que se almacena a información que conten a táboa. O compoñente gráfico JTable permite a visualización dunha gran cantidade de información ben organizada dun só golpe de vista. A información contida nun compoñente da clase JTable pode ser modificada dinamicamente en función de distintos eventos. A xestión das táboas en java fai emprego do paradigma vista controlador (ver listas). O compoñente lista defínese a través da clase javax.swing.JTable. Graficamente o seu aspecto é o seguinte:



Na imaxe anterior podemos ver unha táboa que conten unha serie de valores definidos dinamicamente a través do emprego dun programa.

Propiedades do control JTable empregadas máis habitualmente e que pódense establecer a través do entorno de programación:

Propiedade	Función
Background	Cor de fondo do compoñente
Border	Borde do compoñente
Cursor	Mediante esta propiedade indícase a imaxe que amosará o punteiro do rato ao navegar sobre o compoñente
Enabled	Se esta propiedade está activada o compoñente será funcional. No caso de que esta propiedade estea desactivada o compoñente será visualizable, pero o usuario non poderá interaccionar con el.
Focusable	Se esta propiedade está activada o compoñente entrará na roda de reparto do foco de xeito que ao premer a tecla de cambio de foco (habitualmente o tabulador) nalgún momento tomará o foco da aplicación (cando sexa a súa quenda na roda de reparto do foco). Pola contra, se esta propiedade está desactivada o compoñente non entrará na roda de reparto do foco e soamente será posible acceder a el mediante o rato.
Font	Características da fonte do compoñente (tipo de fonte, tamaño, ...)
Foreground	Cor do texto do compoñente
GridColor	Cor da reixa da táboa

Model	Modelo de datos. Estrutura de datos que contén a información que se almacena na táboa
RowHeight	Altura das filas da táboa
RowSelectionAllowed	Activando esta propiedade indicamos que ao seleccionar un elemento na táboa seleccionarase a fila enteira á que pertence ese elemento. Se desactivamos esta propiedade, ao seleccionar un elemento na táboa unicamente seleccionarase a cela dese elemento.
SelectionBackground	Cor de fondo da selección
SelectionForeground	Cor do texto da selección
SelectionModel	Tipo de selección que podemos facer sobre os elementos da lista. Pode ser unha selección Single Selection (unicamente pódese seleccionar un elemento da táboa), Single Interval Selection (pódense seleccionar varios elementos da táboa, pero deben ser contiguos) ou Multiple Interval Selection (pódense seleccionar varios elementos da táboa e non é preciso que sexan contiguos)
ShowHorizontalLines	Mediante esta propiedade indícase se as liñas horizontais da reixa da táboa serán ou non serán visibles
ShowVerticalLines	Mediante esta propiedade indícase se as liñas verticais da reixa da táboa serán ou non serán visibles
ToolTipText	Mediante esta propiedade establécese unha mensaxe (tooltip) que será amosado ao deixar o punteiro do rato sobre o compoñente. É habitual empregar esta mensaxe para indicar cal é a utilidade do compoñente

Eventos do control JTable empregados máis habitualmente e que pódense establecer a través do entorno de programación:

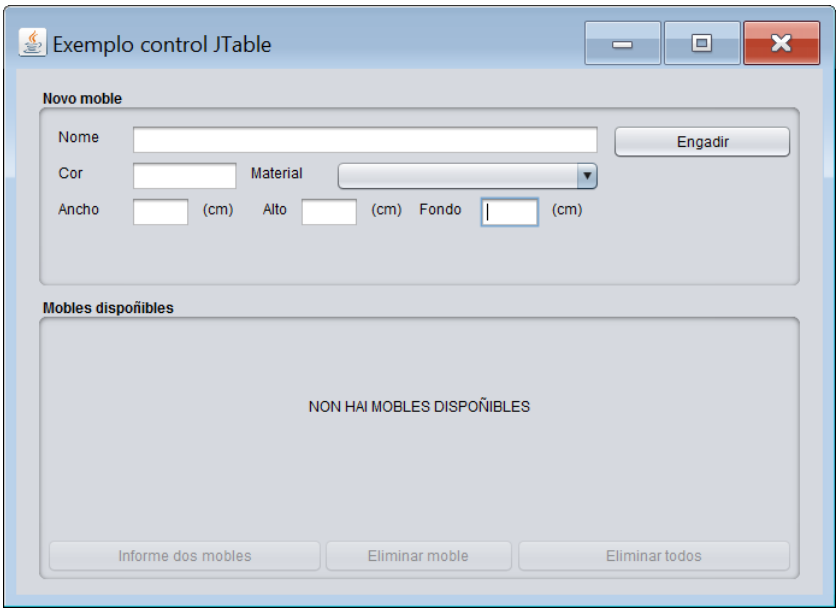
Evento	**Lanzamento**
MouseClicked	Este evento é disparado cando facemos clic co rato sobre o control. Permite contar o número de clics realizados, polo cal realmente sóese empregar para detectar o dobre clic sobre algún elemento da táboa.

Métodos do control JTable empregados máis habitualmente:

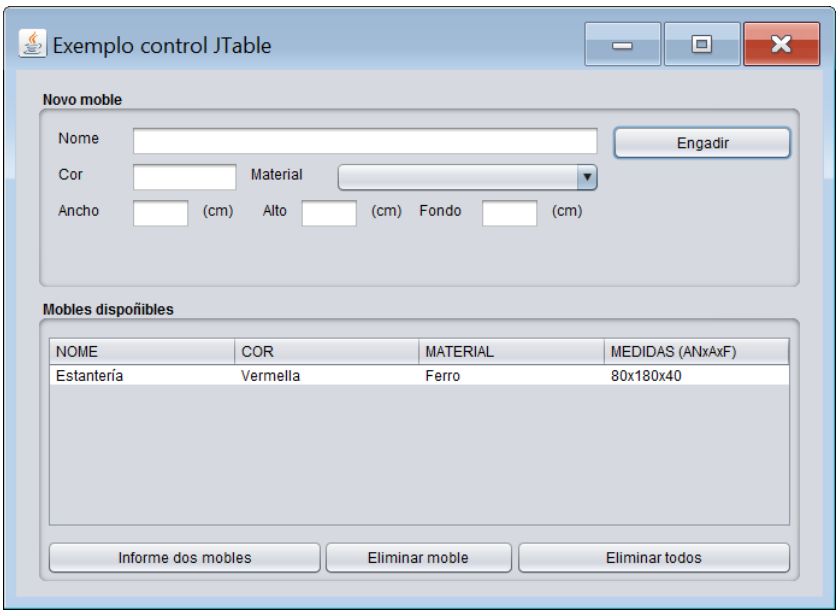
Método	**Función**
public int getSelectedRow()	Devolve o índice da fila seleccionada na táboa. Devolve -1 se non hai ningunha fila seleccionada.
public int getSelectedRowCount()	Devolve o número de filas seleccionadas
public int[] getSelectedRows()	Devolve os índices das filas seleccionadas na táboa. Devolve un arrai baleiro se non hai ningunha fila seleccionada.

Xestión de táboas empregando NetBeans

A continuación imos desenvolver o seguinte formulario:



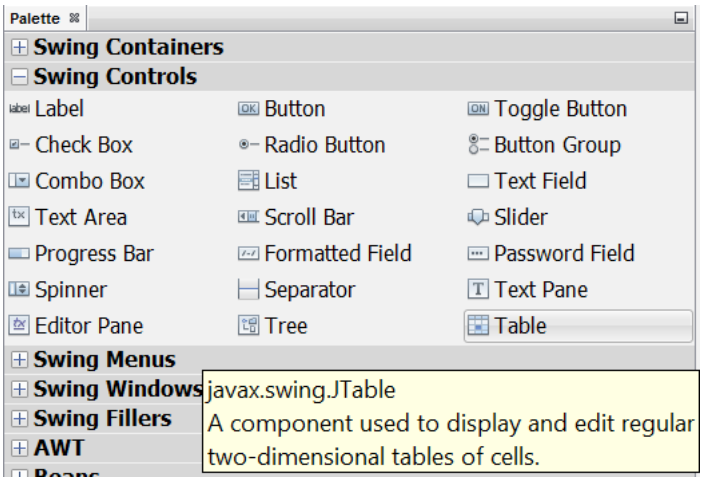
O formulario consta dunha táboa, a cal atópase no panel de mobles dispoñibles. A táboa emprégase para engadir nela os mobles que definamos desde o panel novo moble. Como pódese observar na imaxe anterior, a táboa inicialmente non se visualiza. Imos facer que o noso programa só amose a táboa se conten datos. No caso contrario unicamente amosará unha etiqueta informando de que non hai mobles dispoñibles. No caso de que teñamos introducido algún dato na táboa, este será o aspecto da aplicación:



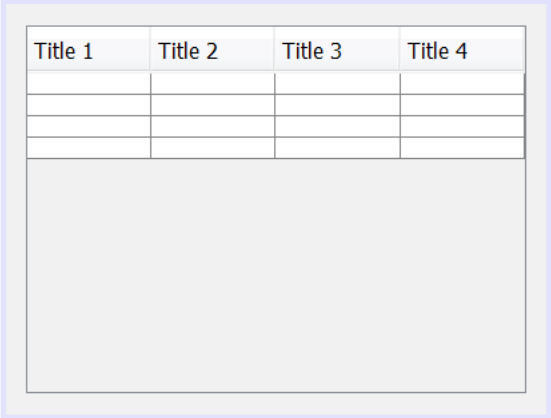
Como pódese observar na imaxe, agora amósase unha táboa, a cal contén información organizada en filas e columnas. Ademais, como un detalle menor fixémonos nos tres botóns que acompañan á táboa. Empréganse para xestionar a información da táboa. Só estarán habilitados cando teñamos información que manipular na táboa. Adicionalmente, no caso de que fagamos dobre clic sobre algunha fila da lista, amosarase a información ao respecto desa fila. A táboa será de tipo multiselección de intervalo múltiple. As caixas de texto Ancho, Alto e

Fondo só admitirán números e a súa lonxitude estará limitada a cinco caracteres. Calquera error que poida acontecer ao longo do emprego do programa debe ser reportado ao usuario.

Para engadir un compoñente de tipo JTable no noso formulario primeiramente seleccionarémolo da paleta de compoñentes do entorno de programación:



e arrastrámolo ata o formulario:

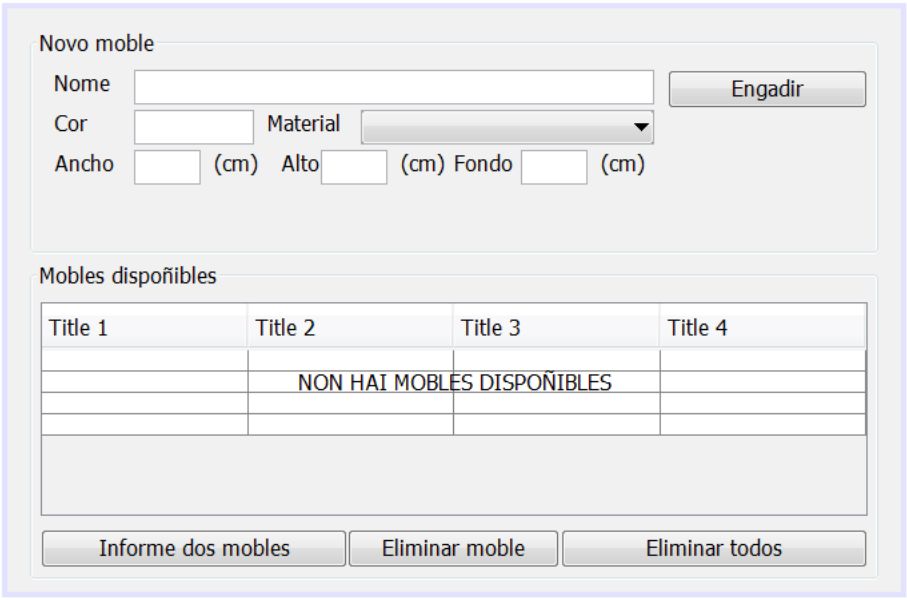


Cando arrastramos unha táboa ao noso formulario en realidade créanse dous compoñentes: un JScrollPane e dentro deste a táboa. Isto é debido a que cando engadimos elementos dentro do compoñente táboa, pode ocorrer que teñamos máis elementos na táboa dos que se poden visualizar en ela. Para desprazarnos polos elementos da táboa podemos empregar os cursores, aínda que é máis doado facelo mediante barras de desprazamento. O problema é que o compoñente táboa non ten barras de desprazamento. Aquí é onde entra en acción o entorno de programación creando a táboa dentro dun JScrollPane. Como resultado, temos que aínda que a táboa non ten barras de desprazamento, de cara ao usuario aparenta que as ten, xa que estas son provistas polo JScrollPane.

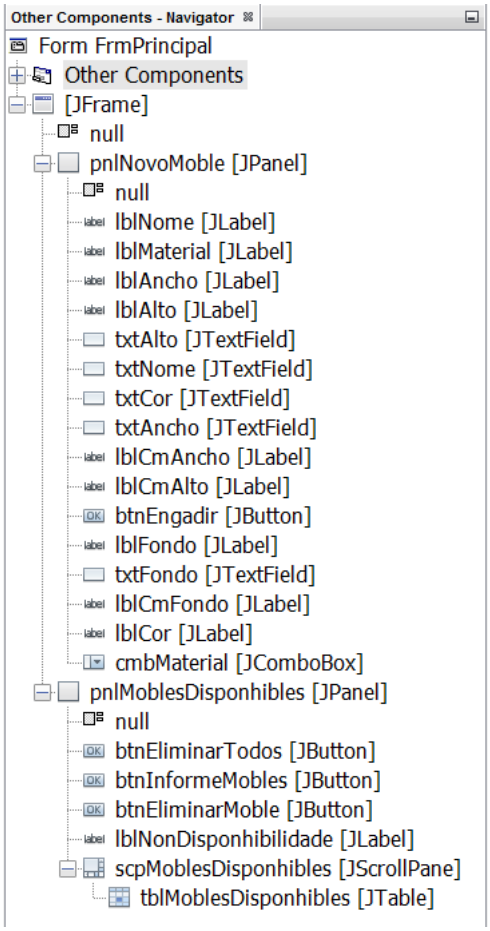
Unha vez situado o compoñente dentro do formulario, adaptámolo para que teña o aspecto visual que desexamos que teña.

No caso que estamos desenvolvendo necesitamos engadir dez etiquetas,

cinco caixas de texto, catro botóns, dous paneis, un combo e unha táboa. Unha vez engadidos e colocados no seu lugar correspondente, este será o resultado do deseño:



O noso formulario quedará configurado do seguinte xeito:



Unha vez que temos colocados os compoñentes que necesitamos hai que configurar o seu aspecto. Neste caso faremos as seguintes modificacións:

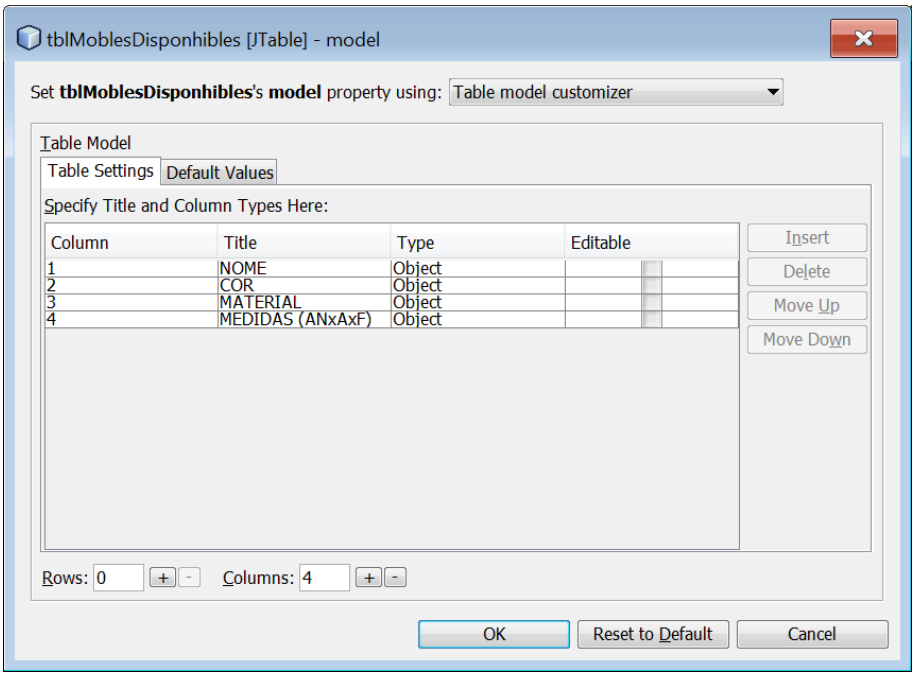
- Para o combo cmbMaterial, modificamos a súa propiedade model para que teña os seguintes valores: Carballo, Cerdeira, Faia, Ferro, Piñeiro, Plástico. Ademais á súa propiedade selectedIndex asignámoslle o valor -1 para que de entrada ningún valor do combo sexa preseleccionado.
- A tódolos botóns do panel pnlMoblesDisponhibles deshabilitámoslles a súa propiedade enabled para que ao arrancar a aplicación aparezan deshabilitados.

Para facer que a táboa non sexa visible ao iniciar a aplicación, temos que modificar a súa propiedade visible, pero o entorno empregado para o desenvolvemento da aplicación non permite facelo graficamente, así que o temos que facer por código:

```
    public FrmPrincipal() {
        initComponents();
        scpMoblesDisponhibles.setVisible(false);
```

O mellor sitio para facer isto é no construtor do formulario xusto despois de pintar os seus compoñentes (initComponents). O método empregado é o método setVisible sobre o compoñente scpMoblesDisponhibles, ao cal pasámoslle false para indicar que non o queremos visualizar. A partir de agora, o compoñente existirá no noso contedor, pero non será visible. En xeral a tódolos compoñentes gráficos pódeselles aplicar este método e serve para facelos visibles (pasando true ao método) o invisibles (pasando false ao método).

Atributos da táboa: imos indicar o número de filas e de columnas da táboa, as cabeceiras da táboa, o tipo de obxectos almacenables na táboa e a posibilidade de edición da táboa. Para elo imos empregar unha pantalla de configuración á que se accede premendo sobre o botón asociado á propiedade model do compoñente tblMoblesDisponhibles. Imos configurar os datos desa pantalla de xeito que obtemos a seguinte configuración para a nosa táboa: cero filas e 4 columnas. Para a primeira columna, a súa cabeceira será NOME, poderase almacenar nela calquera tipo de obxecto (Object) e non será editable (casilla editable desmarcada). Para a segunda, terceira e carta columna só variarán os textos de as súas cabeceiras, que serán COR, MATERIAL e MEDIDAS (ANxAxF) respectivamente. Finalmente esta será a pantalla de configuración que xeraremos para a nosa táboa:



O comportamento visual da aplicación xa está resolto (unicamente deberemos pulir o detalle de cando amosar ou non a táboa e habilitar ou deshabilitar o seus botóns adxuntos). Agora imos desenvolver a súa funcionalidade.

Anteriormente, mencionamos que o funcionamento da táboa baséase no paradigma vista controlador segundo o cal o compoñente internamente garda a información que xestiona empregando unha estrutura de datos determinada mentres que o que amosa non ten que coincidir coa información gardada internamente, senón que é o programador o que pode decidir que é o amosado no compoñente gráfico. Segundo o que acabamos de explicar, para empregar o paradigma vista controlador necesitamos dous elementos, o compoñente gráfico (tblMoblesDisponhibles) que xa o temos e unha estrutura de datos onde almacenar a súa información. Este almacén de datos será un obxecto da clase javax.swing.table.DefaultTableModel. Este obxecto permite xestionar dinamicamente o seu contido (engadir obxectos, eliminalos, modificalos, ...). Neste caso imos declarar o almacén de datos como un atributo da clase, de xeito que podamos acceder a el desde calquera método do formulario:

```
    private DefaultTableModel modeloMobles;
```

Unha vez definida a estrutura de datos que vai funcionar como almacén dos datos da táboa, temos que ligala ao compoñente gráfico de tipo táboa que vai amosar os seus datos. Neste caso imos empregar a configuración realizada sobre a propiedade model da táboa e imos indicar que o almacén de datos imos empregar vai ser o definido a través da propiedade model da táboa. Este almacén de datos xa ten predefinidas unha serie de características (cero filas, catro columnas, cabeceiras, tipos de obxectos almacenables e indicación da non posibilidade de editar as celas). Todas estas características tamén poderíanse indicar a través dun construtor específico da clase DefaultTableModel, pero xa que o entorno de programación nos proporciona unha ferramenta para facelo, podemos facer emprego dela. Resumindo, á variable que imos empregar para referenciar a información almacenada na táboa, ímoslle asignar o modelo definido graficamente para a táboa mediante a seguinte liña de código:

```
modeloMobles=(DefaultTableModel)tblMoblesDisponhibles.getModel();
```

No construtor do formulario e unha vez pintados os seus compoñentes (método initComponentes), facemos emprego do método getModel sobre a táboa. O método getModel obtén unha referencia ao modelo asignado a un compoñente gráfico. Por ultimo asignamos esta referencia á variable modeloMobles, a cal imos empregar para xestionar o almacén de datos asignado á táboa.

De seguido imos ver como engadimos un elemento á táboa. Cando pulsamos o botón Engadir, valídase o formulario. Se todo é correcto xérase un obxecto de tipo Moble cos datos introducidos, amosamos a táboa e habilitamos os botóns asociados á táboa. Por último executamos as seguintes liñas:

```
//Engadir o moble á táboa
modeloMobles.setRowCount(modeloMobles.getRowCount()+1);
modeloMobles.setValueAt(moble, modeloMobles.getRowCount()-1 , 0);
modeloMobles.setValueAt(moble.getCor(), modeloMobles.getRowCount()-1 , 1);
modeloMobles.setValueAt(moble.getMaterial(), modeloMobles.getRowCount()-1 , 2);
String medidas=moble.getAncho()+"x"+moble.getAlto()+"x"+moble.getFondo();
modeloMobles.setValueAt(medidas, modeloMobles.getRowCount()-1 , 3);
```

Analicemos estas liñas: a primeira liña é a seguinte:

```
modeloMobles.setRowCount(modeloMobles.getRowCount()+1);
```

Con esta liña estamos creando unha nova fila ao final do almacén de datos asociado á táboa. Nesta nova fila almacenaremos os obxectos que despois reflectiranse na táboa ligada ao almacén de datos. O método setRowCount establece o número de filas que ten o almacén de datos. O método getRowCount devolve o número de filas que ten o almacén de datos. Neste caso estamos establecendo no almacén de datos tantas filas como ten máis unha, é dicir, estamos engadindo unha fila ao almacén de datos.

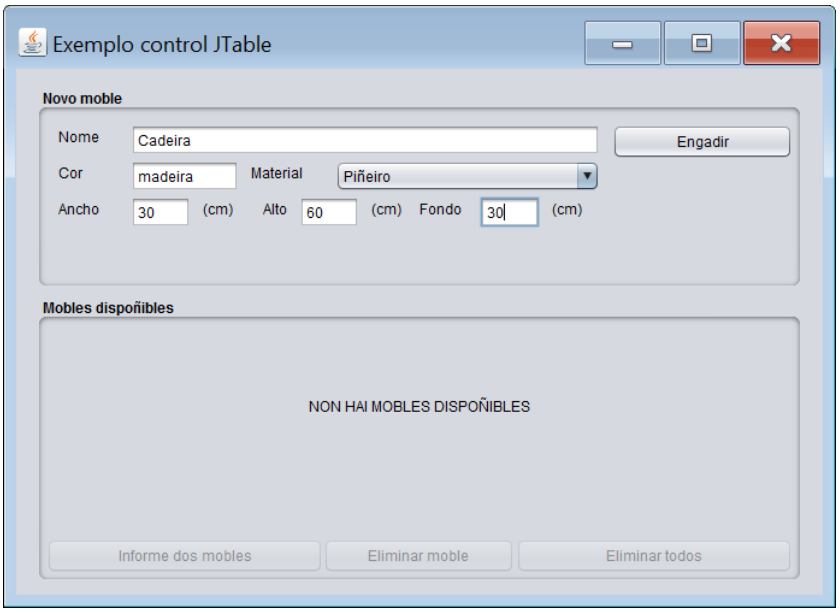
As seguintes liñas son as que se empregan para inserir os obxectos na nova fila creada no almacén de datos. Coa seguinte liña de código:

```
modeloMobles.setValueAt(moble, modeloMobles.getRowCount()-1 , 0);
```

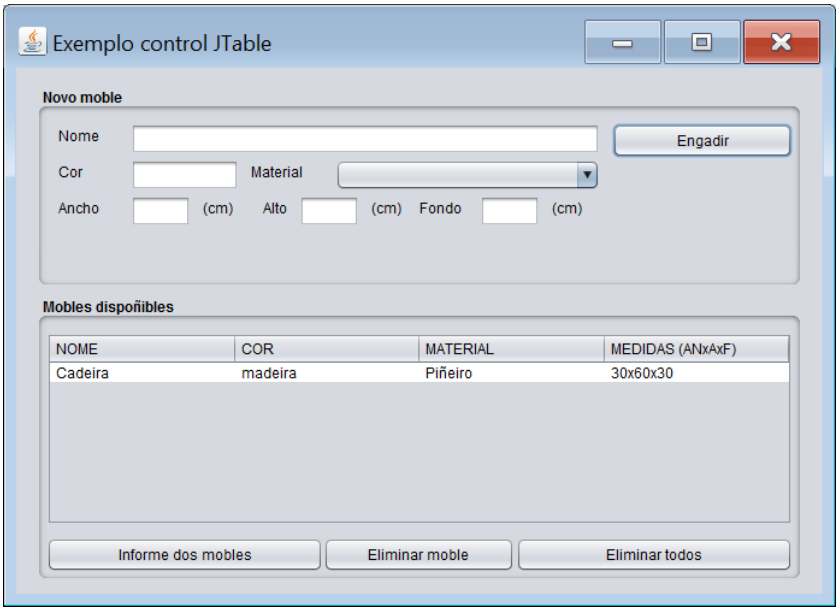
Engadimos ao almacén de datos o obxecto moble que conten os datos do novo moble creado. Como o almacén de datos é unha matriz de celas, debemos indicar a fila e a columna donde imos colocar o obxecto. Neste caso, a fila indicámoslla con modeloMobles.getRowCount()-1. Restámoslle un xa que a primeira fila do almacén de datos é a cero. A columna, neste caso é a primeira, a cal no almacén de datos é a cero. Ao igual que ocorría con listas e combos, ao engadir un obxecto no almacén de datos ligado a un compoñente táboa, o obxecto almacénase completo no almacén de datos, pero na cela correspondente do compoñente táboa vaise visualizar unicamente a información que devolve o método toString do obxecto gardado no almacén de datos.

De igual xeito coas seguintes liñas engadimos na nova fila creada no almacén de datos os valores para a segunda, terceira e carta columna, os cales visualizaranse no compoñente táboa. En cada cela podemos almacenar o obxecto que queiramos. Neste caso optouse por almacenar un Moble (que conten toda a información referente a ese moble) na primeira columna do modelo, mentres que nas outras tres almacenáronse obxectos de tipo String.

Por exemplo se inserimos os seguintes valores:



Cando pulsemos sobre o botón Engadir este será o resultado:



O almacén interno ligado ao compoñente gráfico almacena na súa primeira columna un obxecto de clase Moble cuxos atributos son nome: Cadeira, cor: madeira, ancho: 30, alto: 60 e fondo: 30, pero o compoñente gráfico unicamente amosa na primeira columna o valor que devolve a aplicación do método toString do obxecto de clase Moble, é dicir, Cadeira. Respecto á segunda columna do almacén de datos, almacena un String cuxo valor é madeira. Como o método toString dun String é o propio String, na segunda cela do compoñente visual amosase o valor madeira. O mesmo ocorre coa terceira e carta columnas, almacenan Strings e amosan os seus valores no compoñente gráfico.

A continuación imos ver como é implementado o botón Informe dos mobles. Ao pulsar sobre este botón amósase toda a información almacenada sobre cada un dos obxectos seleccionados na táboa. Este é o código asociado ao botón:

```
    private void btnInformeMoblesActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:

        // Comprobar se hai mobles na taboa
        if(modeloMobles.getRowCount()==0)
        {
            JOptionPane.showMessageDialog(this, "Non hai mobles dispoñibles");
            return;
        }

        //Comprobar se seleccionamos algun moble
        if(tblMoblesDisponhibles.getSelectedRowCount()==0)
        {
            JOptionPane.showMessageDialog(this, "Debe seleccionar ao menos un moble");
            return;
        }

        //Recuperar os mobles do modelo

        int posicionesMobles[]=tblMoblesDisponhibles.getSelectedRows();

        String resultado="";
        for(int i=0;i<posicionesMobles.length;i++)
        {
            Moble moble=(Moble)modeloMobles.getValueAt(posicionesMobles[i], 0);
            resultado+="Nome: "+moble.getNome()+" Cor: "+moble.getCor()+
                    " Material: "+moble.getMaterial()+" Ancho: "+moble.getAncho()+
                    " Alto: "+moble.getAlto()+" Fondo: "+moble.getFondo()+"\n";

        }

        JOptionPane.showMessageDialog(this, resultado);
    }
```

O primeiro que temos que facer é comprobar se a táboa está baleira, xa que de ser así, non teremos información que amosar. Para elo executamos a seguinte liña:

```
        if(modeloMobles.getRowCount()==0)
```

O método getRowCount aplícase sobre o almacén de datos que é o que realmente conten a información. Devólvenos o número de filas que conten o almacén de datos.

A continuación comprobamos se o usuario seleccionou algunha fila da táboa. Para elo executamos a seguinte liña:

```
        if(tblMoblesDisponhibles.getSelectedRowCount()==0)
```

O método getSelectedRowCount aplícase sobre o compoñente táboa e devolve o número de filas seleccionadas na táboa. No caso de que devolva -1 quere dicir que non hai ningunha fila seleccionada.

A continuación temos que recuperar cales son as filas que o usuario seleccionou na táboa. Para elo executamos a seguinte liña:

```
        int posicionesMobles[]=tblMoblesDisponhibles.getSelectedRows();
```

O método getSelectedRows aplícase sobre o compoñente táboa e devolve un arrai de int que conten os índices das filas marcadas na táboa (a primeira fila é a cero). Como a táboa é de selección múltiple hai que empregar este método para recuperar tódolos elementos marcados. Se fora de selección simple, poderíamos empregar este método o tamén o método getSelectedRow.

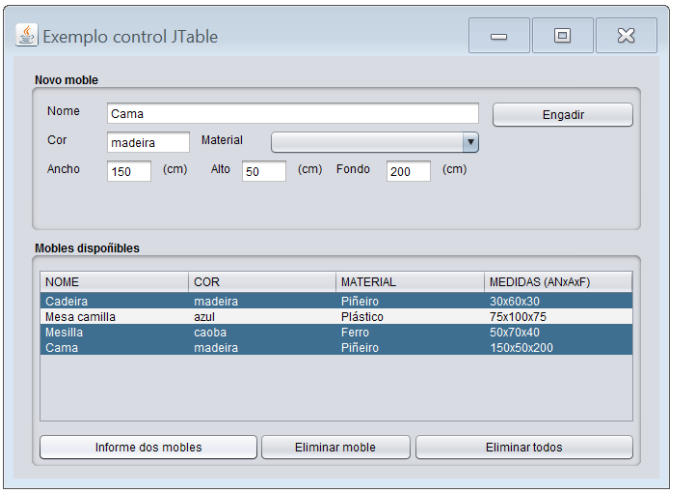
Finalmente imos recuperar tódolos elementos seleccionados na táboa. Para elo empregamos o método getValueAt, o cal aplícase sobre o almacén de datos que conten a información:

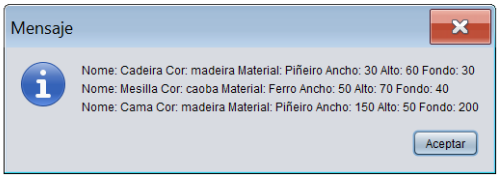
```
            Moble moble=(Moble)modeloMobles.getValueAt(posicionesMobles[i], 0);
```

Como pódese observar, para recuperar un valor do almacén de datos hai que indicar a súa posición dentro del. Isto faise pasando a súa fila e a súa columna. Neste caso a fila conseguímola do arrai de filas marcadas, mentres que a columna é a cero, xa que na columna cero estamos almacenando o obxecto

completo. Agora que foi recuperado un obxecto da clase Moble podemos acceder aos seus atributos a través de os seus métodos getter e setter.

A continuación amósase o resultado de premer sobre o botón Informe dos mobles:

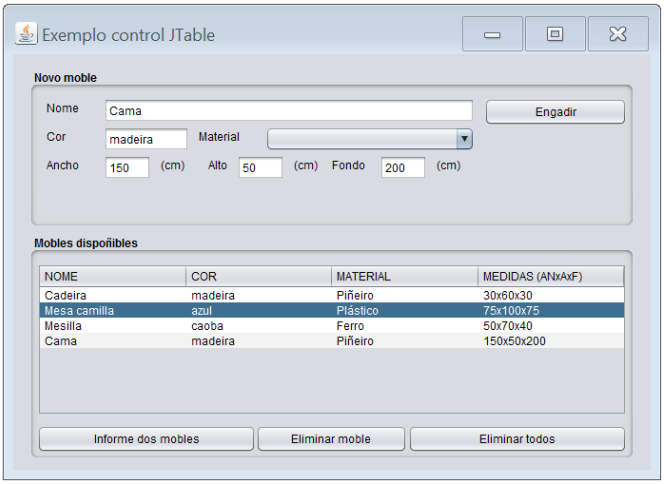


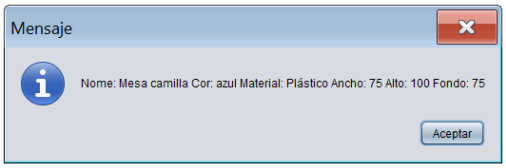


Outro dos requisitos da nosa aplicación é que ao facer dobre clic sobre algún elemento da táboa sexa amosada a información ao respecto dese elemento. Para elo, temos que implementar o evento mouseClicked da táboa:

```
    private void tblMoblesDisponhiblesMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        if(evt.getClickCount()==2)
        {
            //Recuperar os moble seleccionado
            Moble moble=(Moble)modeloMobles.getValueAt(tblMoblesDisponhibles.getSelectedRow(), 0);
            String resultado="Nome: "+moble.getNome()+" Cor: "+moble.getCor()+
                        " Material: "+moble.getMaterial()+" Ancho: "+moble.getAncho()+
                        " Alto: "+moble.getAlto()+" Fondo: "+moble.getFondo();
            JOptionPane.showMessageDialog(this, resultado);
        }
    }
```

A detección do dobre clic xa foi analizada cando vimos as listas. Respecto ao xeito de recuperar a información da fila sobre a que fixemos dobre clic, é similar ao empregado no botón Informe dos mobles visto anteriormente. A continuación amósase o resultado de facer dobre clic sobre un elemento da táboa:





O seguinte botón que imos ver como implementar é o botón Eliminar moble. Ao pulsar sobre este botón elimínanse os mobles seleccionados na táboa. Unha vez realizadas as validacións, a liña que se encarga de eliminar cada un dos elementos seleccionados na táboa é a seguinte:

```
            modeloMobles.removeRow(posicionesMobles[i]);
```

O método removeRow aplícase sobre o almacén de datos. Mediante este método elimínanse tódolos obxectos almacenados na fila que se pasa como parámetro no almacén de datos. Debido á ligazón establecida entre o almacén de datos e o compoñente gráfico táboa, ao eliminar o obxecto do almacén de datos, tamén desaparece da táboa.

O último botón que imos ver como implementar é o botón Eliminar todos. Ao pulsar sobre este botón elimínanse tódolos mobles da táboa. Unha vez realizadas as validacións, a liña que se encarga de eliminar tódolos elementos da táboa é a seguinte:

```
        modeloMobles.setRowCount(0);
```

O xeito de eliminar tódolos elementos do almacén de datos é empregar o seu método setRowCount para indicar que queremos que a partir de agora o modelo de datos teña cero filas, é dicir, que quede baleiro. Debido á ligazón establecida entre o almacén de datos e o compoñente gráfico táboa, ao eliminar tódolos obxectos do almacén de datos, tamén desaparecen da táboa.