

<b>Universidad Técnica Nacional</b> <b>Ingeniería del software</b> <b>Sede San Carlos</b>	Programación I
Prof. Allan Murillo Alfaro	Interfaces gráficas

## Objetivos

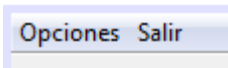
Que el estudiante sea capaz de:

- Desarrollar programas con interfaces de usuario utilizando el generador de NetBeans

### 1. Guía

Vamos a crear una aplicación que utilice algunos de los controles con que cuenta la librería Swing.

- Cree un nuevo proyecto y proporcionele el nombre que desee.
- Luego, agregue un JFrame al que va a llamar Ventana.
- Ahora, seleccione el control Menú Bar que se encuentra en Swing Menús y arrástrelo al formulario. Para cambiar las opciones del menú haga click derecho en la opción y seleccione "Edit Text". Cambie las opciones para que queden como se muestra en la siguiente imagen.

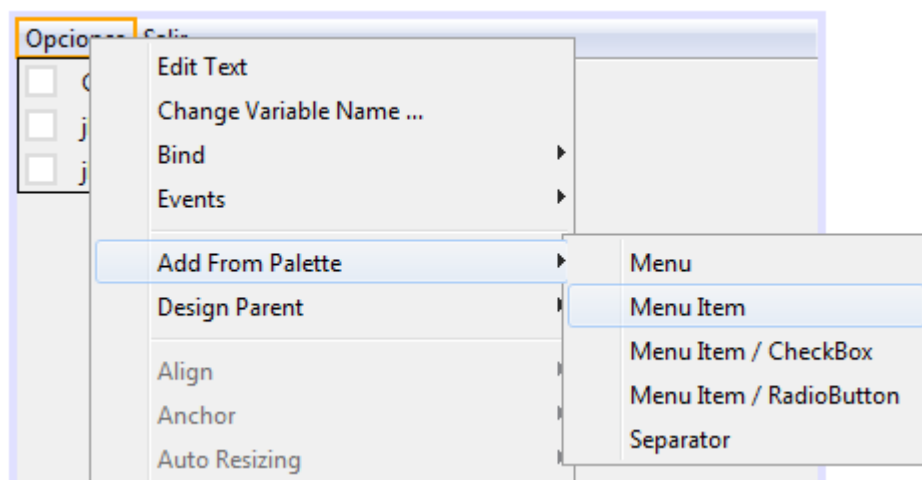


- Seguidamente, implemente la opción Salir agregando el siguiente código. Pero, antes debe hacer click derecho en la opción y seleccionar Events, Action, ActionPerformed.

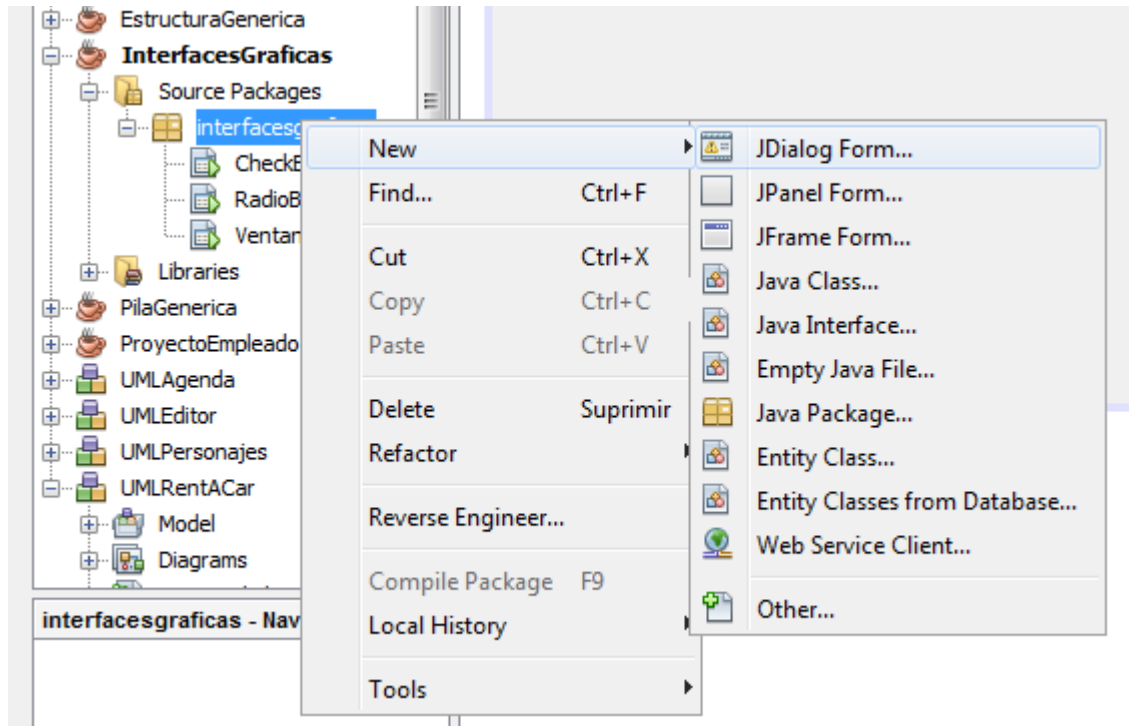
```
System.exit(0);
```

Nota: Tome en consideración que cuando el usuario utiliza el ratón, el teclado o cualquier otros dispositivo físico para interaccionar con las componentes de una interfaz gráfica, la máquina "virtual" (en ejecución) produce un evento que representa dicha interacción.

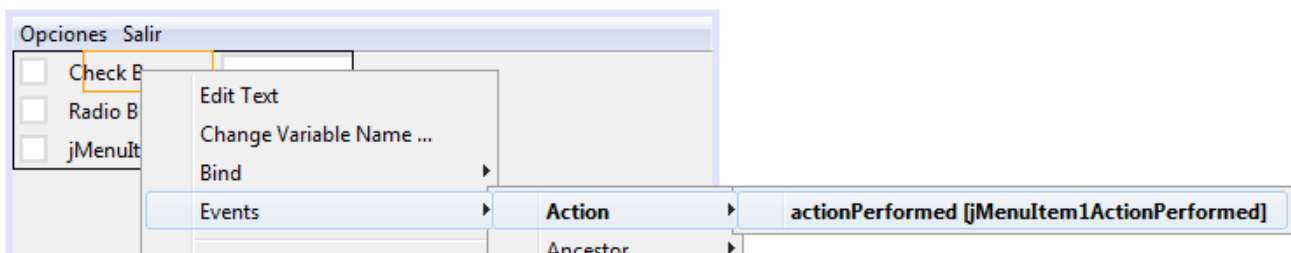
- Para agregar ítems a cada opción, usted debe dar click derecho y seleccionar la opción "Add From Palette" y luego Menu Item.



- Ahora, haz click derecho en el nuevo Item para cambiarle el nombre. Y, escribe CheckBox.
- Lo que queremos hacer es tener una nueva ventana que al dar click en el Item agregado anteriormente se muestra. Para ello, debes agregar un JDialog Form y llamarlo CheckBox.



- Si la opción no aparece en la lista, selecciona "Other" y búscala.
- Para hacer que al dar click en el Item del menú aparezca esta nueva ventana, debes dar click derecho en el Item y agregar el Evento actionPerformed.



- Posteriormente, debes agregar el siguiente código para que aparezca la ventana.

```
CheckBox ventanaSecundaria= new CheckBox(this,true);
ventanaSecundaria.pack();
ventanaSecundaria.setVisible(true);
```

- Al correr la aplicación, veras que las ventanas aparecen en la esquina izquierda en la parte superior de la pantalla de la computadora. Para centrarla en la pantalla solamente se debe agregar en ambos constructores la siguiente línea de código.

```
setLocationRelativeTo(null);
```

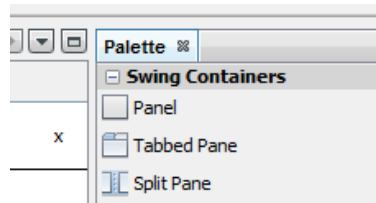
- Seguidamente, trabajaremos en el JDialog creado anteriormente. Debes agregar un botón "Aceptar" llamado btnAceptar y una etiqueta llamada etiResultado.
- También, debes añadir tres cuadros de verificación. Estos cuadros son objetos del tipo JCheckBox. Cambia el texto de ellos, de forma que aparezca "Perro", "Gato" y "Ratón".
- Debe cambiar el nombre de cada uno de ellos. Se llamarán: chkPerro, chkGato, chkRaton.
- La ventana tendrá el siguiente aspecto cuando termine:



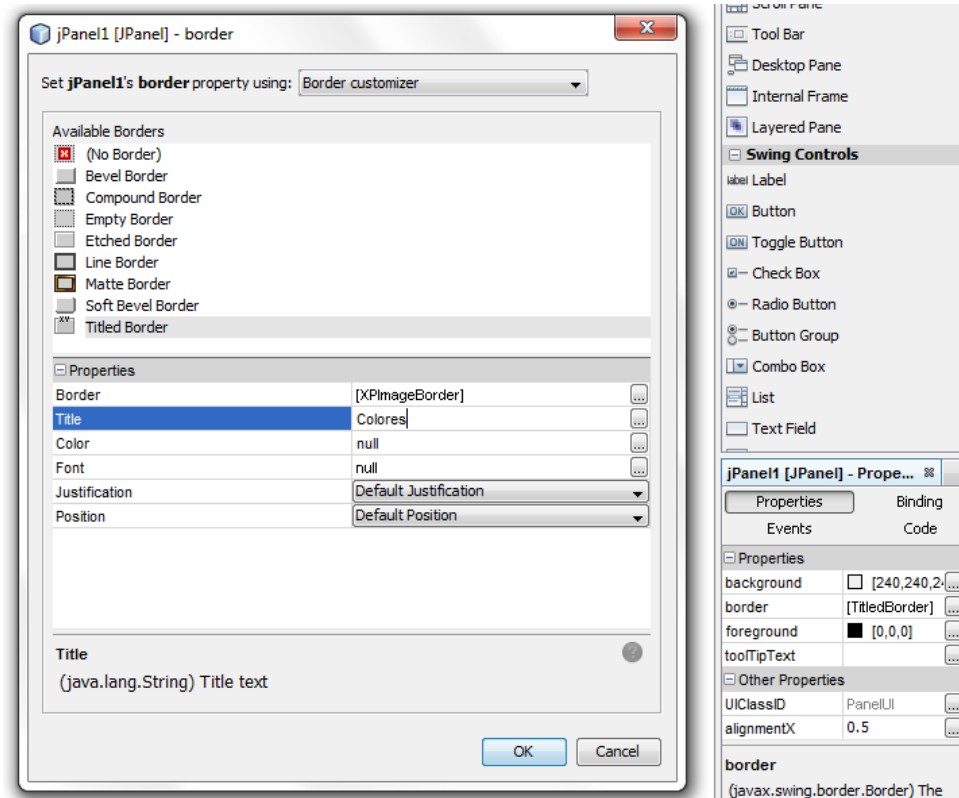
- El programa debe funcionar de la siguiente forma:  
Cuando el usuario pulse aceptar, en la etiqueta aparecerá un mensaje indicando qué animales han sido “seleccionados”. Para ello hay que programar el evento actionPerformed del botón Aceptar. En ese evento añade el siguiente código:

```
String mensaje="Animales elegidos: ";
if (chkPerro.isSelected()) {
    mensaje=mensaje+"Perro ";
}
if (chkGato.isSelected()) {
    mensaje=mensaje+"Gato ";
}
if (chkRaton.isSelected()) {
    mensaje=mensaje+"Raton ";
}
etiResultado.setText(mensaje);
```

- Ahora, aprenderemos a utilizar los radiobutton. Para ello agrega una nueva opción en el menú que llame a un nuevo JDialog. En el que debes agregar un panel. Un panel es una zona rectangular que puede contener elementos (botones, etiquetas, etc).



- Una vez añadido el panel al JDialog, le pondremos un borde para poder localizarlo fácilmente. Debes hacer lo siguiente:
  - o Selecciona el panel que has añadido.
  - o Activa la propiedad Border (botón con tres puntos)
  - o Busca el tipo de borde llamado Title y escribe el título Colores.



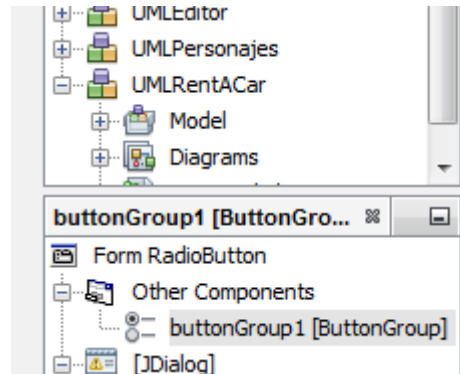
- Además, agrega al JDialog Un botón “Aceptar” llamado btnAceptar. Y un label o etiqueta con borde llamada etiResultado.
- Tu ventana debe quedar más o menos así:



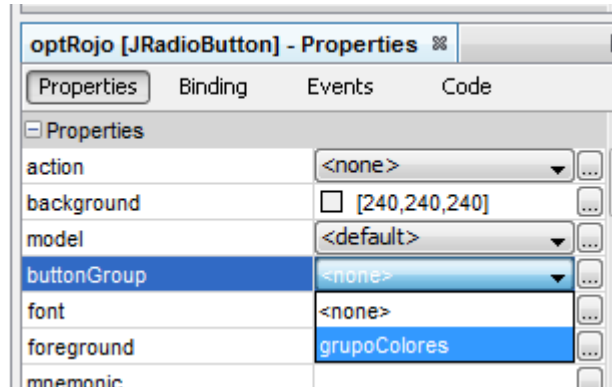
- A continuación debes añadir tres botones de opción (botones de radio) dentro del panel. Estos botones son objetos del tipo JRadioButton.
- Añade tres JRadioButton y cambia el texto de ellos, de forma que aparezca “Rojo”, “Verde” y “Azul”.
- Debe cambiar el nombre de cada uno de ellos. Se llamarán: optRojo, optVerde, optAzul.
- La ventana tendrá el siguiente aspecto cuando termine:



- Si ejecuta el programa, observará que pueden seleccionarse varios colores a la vez. Esto no es interesante, ya que los botones de opción se usan para activar solo una opción entre varias.
- Hay que hacer que solo un botón de opción pueda estar seleccionado a la vez. Para ello, debe añadir un nuevo objeto. Realice los siguientes pasos:
  - o Añada un objeto del tipo ButtonGroup al formulario. ¡Atención! Este objeto es invisible, y no se verá en el formulario, sin embargo, lo podréis ver en el Inspector, en la parte de “Otros Componentes”:



- o Tienes que darle un nombre al ButtonGroup. El nombre será “grupoColores”.
- o Ahora, hay que conseguir que los tres botones pertenezcan al mismo grupo. Es decir, que pertenezcan al grupo grupoColores.
- o Selecciona el botón de opción optRojo y cambia su propiedad buttonGroup, indicando que pertenece al grupo colores (observa la imagen):



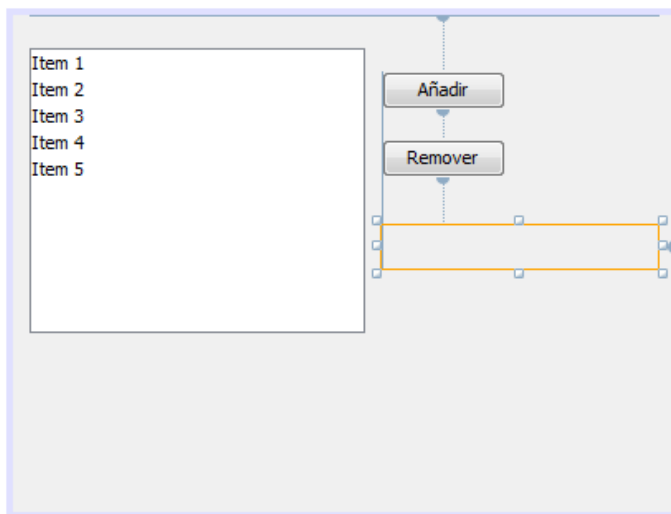
- Haz lo mismo con los botones optVerde y optAzul.
- Acabas de asociar los tres botones de opción a un mismo grupo. Esto produce que solo una de las tres opciones pueda estar activada. Pruébalo ejecutando el programa.
- Ahora interesa que la opción “Rojo” salga activada desde el principio. Una forma de hacer esto es programando en el “Constructor” lo siguiente:

```
optRojo.setSelected(true);
```

- El método setSelected hace que se pueda activar o desactivar un botón de opción.
- Prueba el programa. Observa como la opción Rojo está activada inicialmente.
- El programa no está terminado aún. Interesa que cuando el usuario pulse el botón Aceptar, en la etiqueta aparezca el color elegido. Para ello, en el actionPerformed del botón Aceptar programe lo siguiente:

```
String mensaje="Color elegido: ";
if (optRojo.isSelected()) {
    mensaje=mensaje+"Rojo";
} else if (optVerde.isSelected()) {
    mensaje=mensaje+"Verde";
} else if (optAzul.isSelected()) {
    mensaje=mensaje+"Azul";
}
etiResultado.setText(mensaje);
```

- El siguiente control que aprenderemos a utilizar en el JList. Nuevamente, agrega un nuevo JDialog y en la ventana principal un nuevo Item que llame a dicha ventana.
- En el JDialog creado agrega un List y dale por nombre lista.
- Agrega un botón que diga Añadir y otro que diga Remover. Así como un label, al que no es necesario cambiarle el nombre. La ventana quedará similar a la siguiente imagen.



- Para poder utilizar la clase DefaultListModel, es necesario importar la siguiente librería: javax.swing.\*
- Declara las siguientes variables globales en el JDialog.

```
private DefaultListModel modelo; //lista vacía
private int seleccion=-1; //control de los elementos que se introducirán en el JList
```

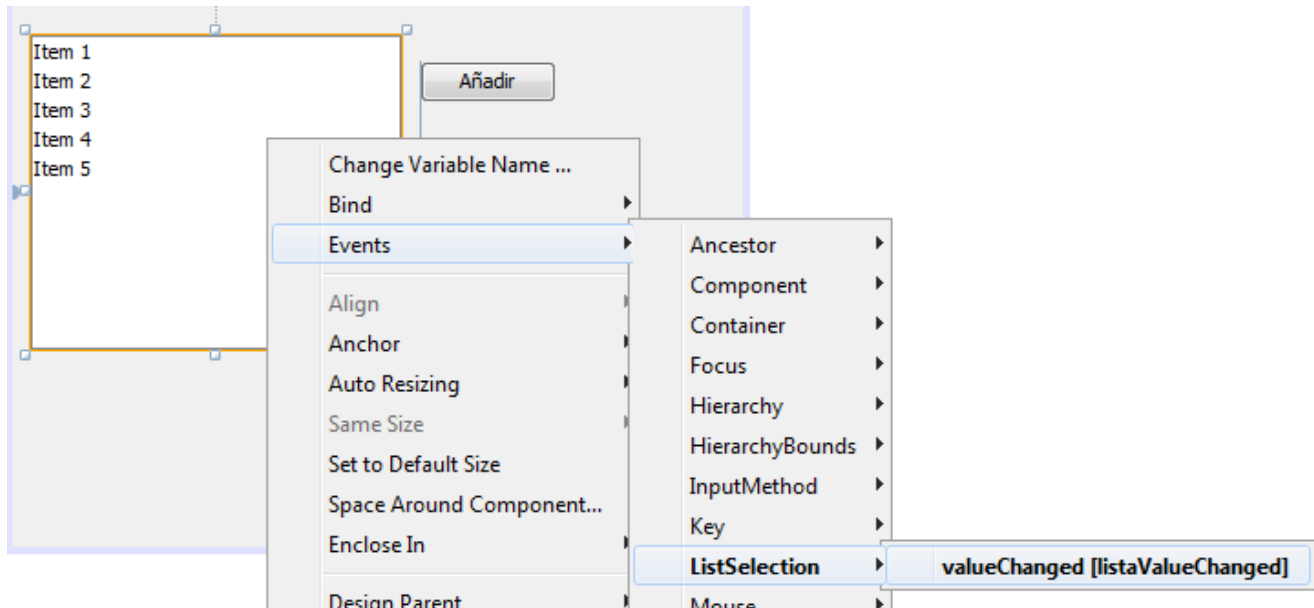
- En el constructor agrega las siguientes líneas de código.

```
modelo = new DefaultListModel();
lista.setModel(modelo);
```

- En el botón de añadir componentes, vamos a utilizar una ventana de diálogo para buscar el archivo e insertar el nombre en el list.

```
JFileChooser fileChooser = new JFileChooser();
int result = fileChooser.showOpenDialog(null);
if ( result == JFileChooser.APPROVE_OPTION ){
    String name = fileChooser.getSelectedFile().getName();
    modelo.addElement(name);
}
```

- En el evento “ValueChanged” del list, vamos a incorporar el código que sirve para obtener el elemento seleccionado.

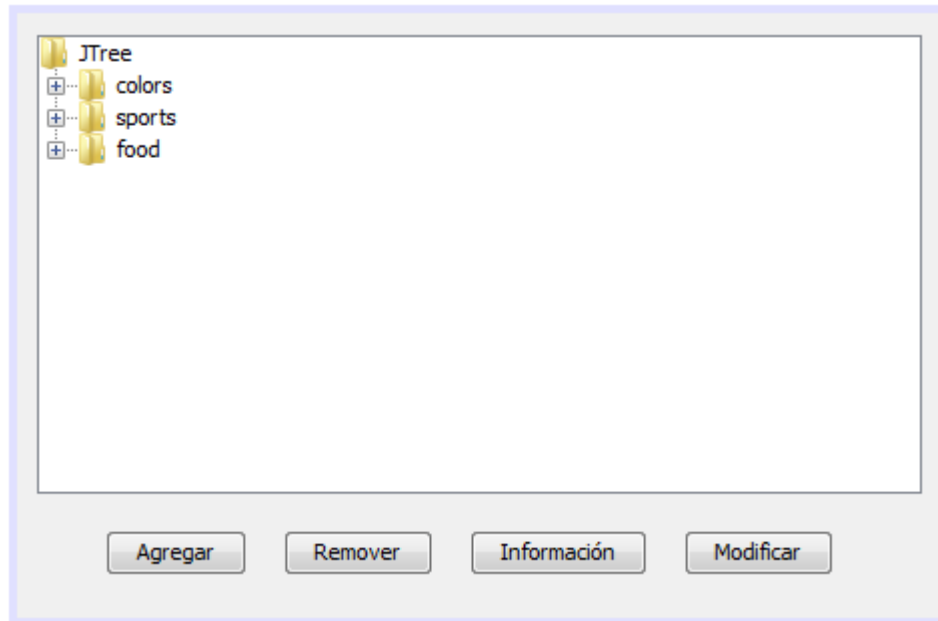


```
seleccion = lista.getSelectedIndex();
if (seleccion >=0){
    jLabel1.setText("Archivo seleccionado: " + seleccion);
}
else
{
    jLabel1.setText("Archivo seleccionado:");
}
```

- Para remover un elemento agregamos el evento “actionPerformed” al botón Remove e incorporamos el siguiente código.

```
if (seleccion >=0){
    modelo.removeElementAt(seleccion);
}
```

- Otro control interesante de aprender a utilizar es el JTree. De igual forma que en los casos anteriores, agregue un nuevo JDialog y un ítem en el menú que llame a la ventana creada.
- Seguidamente agregue un JTree y cuatro botones, obteniendo un resultado como el siguiente.



- Primero debe importar la siguiente librería: `import javax.swing.tree.*;`
- Declare las siguientes variables globales:

```
DefaultMutableTreeNode Titulo=null;  
DefaultTreeModel modelo=null;
```

- Necesitamos definir un modelo que contenga la información que deseamos se muestre en el árbol. Por ello, vamos a incorporar el siguiente método en el que primero se define el padre que sería el título y luego una serie de categorías y subcategorías. Definiendo en cada uno de ellos la posición en donde irá ubicado el elemento.

```
public DefaultTreeModel cargarArbol() {  
    DefaultMutableTreeNode categorias = null;  
    DefaultMutableTreeNode subcategoria = null;  
  
    Titulo = new DefaultMutableTreeNode("Centro de Administración");  
    modelo = new DefaultTreeModel(Titulo);  
  
    categorias = new DefaultMutableTreeNode("Adm. Redes");  
    subcategoria = new DefaultMutableTreeNode("Dept. de planificación");  
    modelo.insertNodeInto(categorias, Titulo, 0);  
    modelo.insertNodeInto(subcategoria, categorias, 0);  
  
    categorias = new DefaultMutableTreeNode("Adm. de Laboratorios");  
    subcategoria = new DefaultMutableTreeNode("Dept. de Tecnología");  
    modelo.insertNodeInto(categorias, Titulo, 1);  
    modelo.insertNodeInto(subcategoria, categorias, 0);  
  
    categorias = new DefaultMutableTreeNode("Dept. de Investigación");  
    subcategoria = new DefaultMutableTreeNode("Dept. A");  
    modelo.insertNodeInto(categorias, Titulo, 2);  
}
```



```

        modelo.insertNodeInto(subcategoria, categorias, 0);

        return modelo;
    }

```

- En el constructor debe cargarle a JTree el modelo por lo que debe agregar la siguiente línea de código.

```

jTree1.setModel(cargarArbol());

```

- Para agregar un elemento al árbol, añada al botón Agregar el evento ActionPerformed. Y, copie el siguiente código. En el que primero se solicita un texto, luego se obtiene el objeto seleccionado y por último se agrega el elemento al modelo.

```

String cadena=JOptionPane.showInputDialog(this,"Ingrese un departamento");
DefaultMutableTreeNode parentNodo=null;
TreePath parentPath=jTree1.getSelectionPath();

if(parentPath==null){

}
else{
    parentNodo=(DefaultMutableTreeNode)parentPath.getLastPathComponent();
}
DefaultMutableTreeNode child = new DefaultMutableTreeNode(cadena);
modelo.insertNodeInto(child, parentNodo, parentNodo.getChildCount());

```

- Para eliminar un elemento, se obtiene el objeto seleccionado y el modelo. Y, se elimina el objeto del modelo. Copie el código en el evento ActionPerformed del botón Remove.

```

DefaultMutableTreeNode parentNode = null;
TreePath currentSelection = jTree1.getSelectionPath();

if (currentSelection != null) {
    parentNode = (DefaultMutableTreeNode) currentSelection.getLastPathComponent();
    DefaultTreeModel model = ((DefaultTreeModel) jTree1.getModel());
    model.removeNodeFromParent(parentNode);
}

```

- Para mostrar la información, agregue el evento al botón. Y, luego copie el siguiente código.

```

DefaultMutableTreeNode node = (DefaultMutableTreeNode) jTree1.getLastSelectedPathComponent();
if (node == null) //Nothing is selected.
{
    return;
}
Object nodeInfo = node.getUserObject();

```

```
JOptionPane.showMessageDialog(this, nodeInfo.toString());
```

- Por último, para modificar la información. Debe agregar el evento al botón y este es el código que se requiere para modificar. En primer lugar se obtiene el nodo seleccionado, luego se pide la nueva información y finalmente obtiene el modelo y se modifica el objeto.

```
DefaultMutableTreeNode node = (DefaultMutableTreeNode) jTree1.getLastSelectedPathComponent();
    if (node == null) //Nothing is selected.
    {
        return;
    }

    Object nodeInfo = node.getUserObject();
    String cadena = JOptionPane.showInputDialog(this, "Ingrese el nuevo nombre",nodeInfo.toString());
    TreePath currentSelection = jTree1.getSelectionPath();
    if (currentSelection != null) {
        node = (DefaultMutableTreeNode) currentSelection.getLastPathComponent();
        node.setUserObject(cadena);
        DefaultTreeModel model = ((DefaultTreeModel) jTree1.getModel());
        model.nodeChanged(node);
    }
```

- El JTable es un componente de mucho ayuda para mostrar datos. Por ello ahora trabajaremos con dicho componente. De este modo, es necesario que agregue un nuevo JDialog y un Item en el menú.
- Luego, debe agregar el componente Jtable, un label, un textbox y tres botones. Según su muestra en la siguiente imagen.

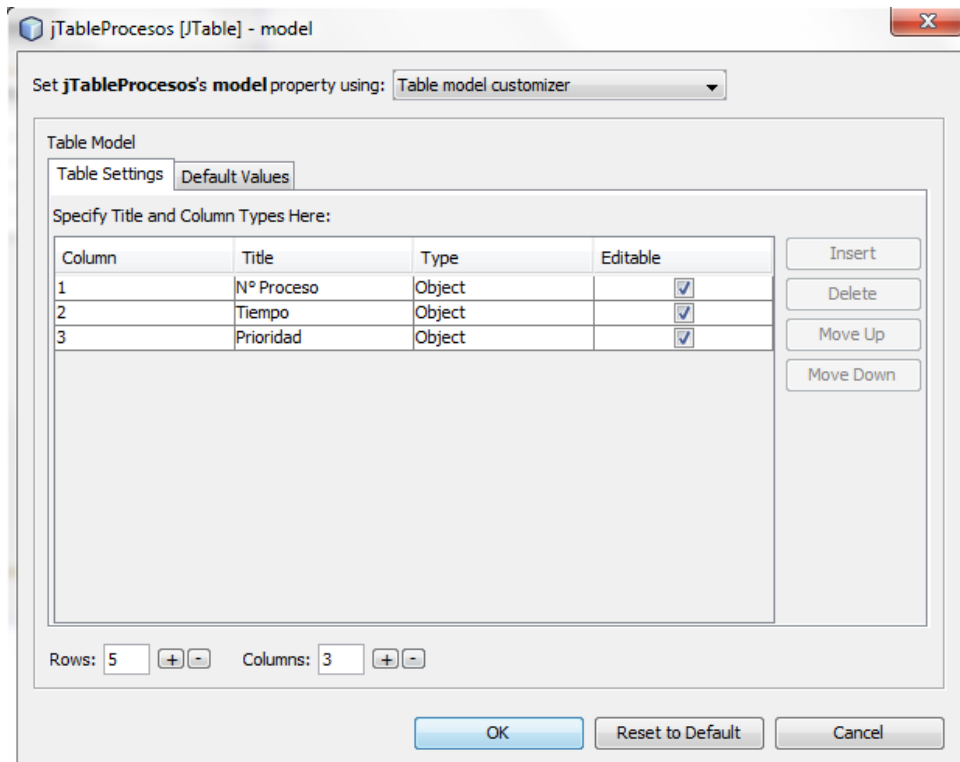
The screenshot shows a Java Swing window with a light gray background. At the top, there is a label "Tiempo:" followed by a text input field and a "Buscar" button. Below this is a JTable with three columns: "N° Proceso", "Tiempo", and "Prioridad". The table has five rows, numbered 1 to 5 in the first column. To the right of the table are two buttons, "+" and "-", stacked vertically. Below the table is a large, empty rectangular area.

N° Proceso	Tiempo	Prioridad
1		
2		
3		
4		
5		

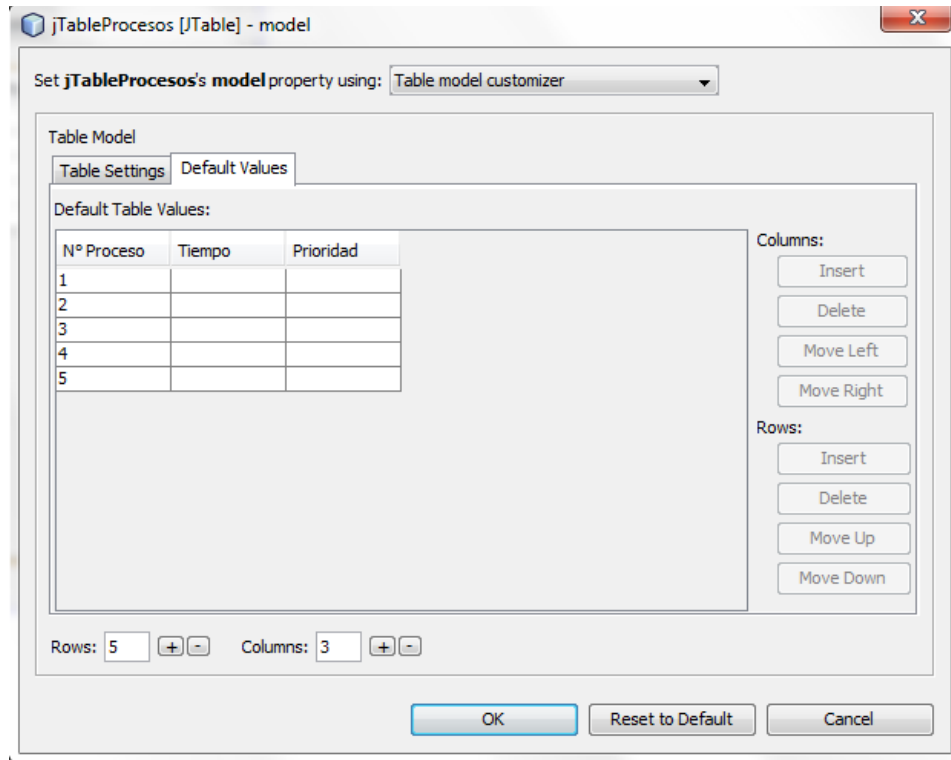
- Para cambiar la tabla se debe modificar la propiedad denominada model.

Properties		
autoCreateColumnsFromModel	<input checked="" type="checkbox"/>	...
autoCreateRowSorter	<input type="checkbox"/>	...
background	<input type="checkbox"/> [255,255,255]	...
border	(No Border)	...
font	Tahoma 11 Plain	...
foreground	<input checked="" type="checkbox"/> [0,0,0]	...
model	[TableModel]	...

- Se desplegará una ventana en la que puedes modificar la cantidad de columnas.



- También, puedes agregar elementos por defecto como se muestra en la siguiente imagen.



- Modifica el nombre del textBox por txtElemento y el del Table por jTableProcesos.
- Importe la librería: `import javax.swing.table.*;`
- En el evento actionPerformed del botón "+", agregue el siguiente código:

```
DefaultTableModel temp = (DefaultTableModel) jTableProcesos.getModel();
Object nuevo[] = {temp.getRowCount()+1, "", ""};
temp.addRow(nuevo);
```

- En el evento actionPerformed del botón "-", agregue el siguiente código para eliminar la última fila de la tabla.

```
try
{
    DefaultTableModel temp = (DefaultTableModel) jTableProcesos.getModel();
    temp.removeRow(temp.getRowCount()-1);
}
catch(ArrayIndexOutOfBoundsException e){}
```

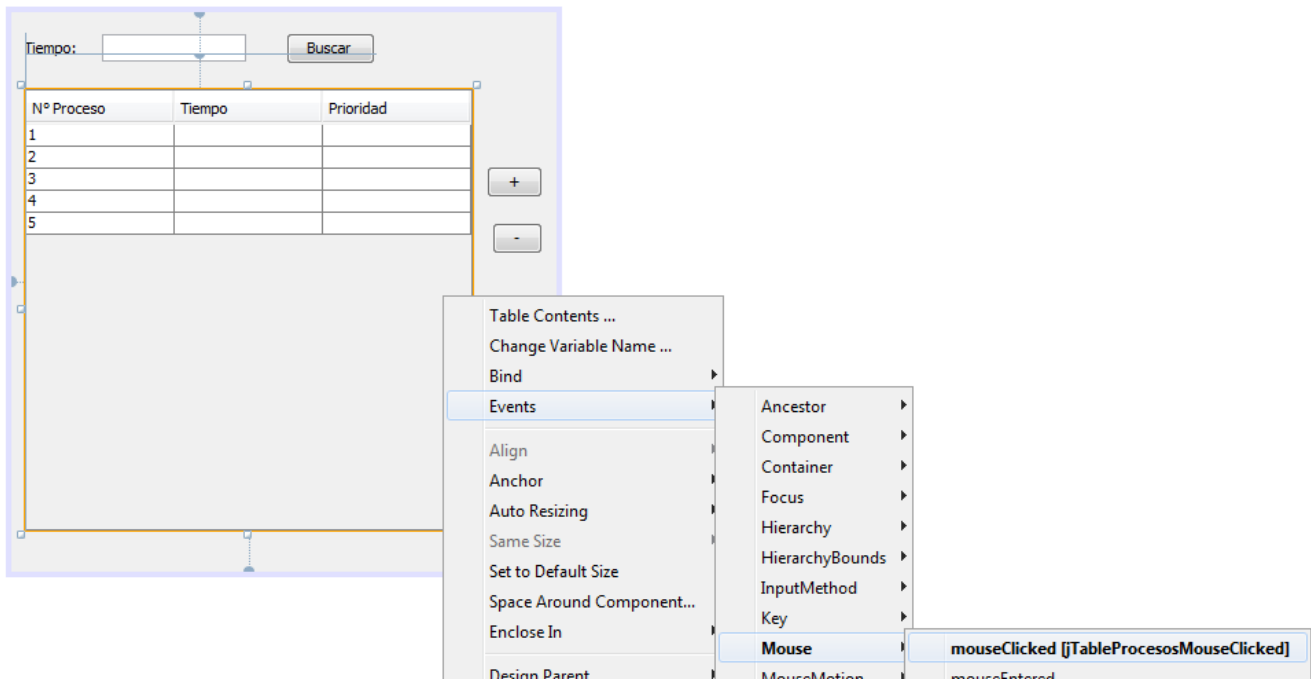
- Para buscar un elemento dentro de la tabla, las siguientes líneas de código lo realizan con el campo tiempo.

```
String ele = txtElemento.getText();

for (int i = 0; i < jTableProcesos.getRowCount(); i++) {
    if (jTableProcesos.getValueAt(i, 1).equals(ele)) {
        jTableProcesos.changeSelection(i, 1, false, false);
        break;
    }
}
```

}

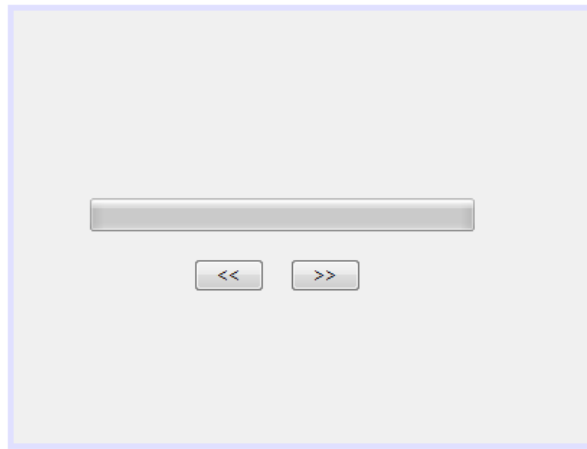
- Modifique el código anterior para que realice la búsqueda con el campo prioridad.
- Ahora, agregue el evento mouseClicked a la tabla.



- Y, en el evento anterior agregue el código que permite mostrar la información del elemento seleccionado.

```
TableModel tablaModelo;  
tablaModelo = (TableModel) jTableProcesos.getModel();  
boolean avanzar = true;  
  
int registro = jTableProcesos.getSelectedRow();  
int columna = jTableProcesos.getSelectedColumn();  
  
if (registro == -1) {  
    avanzar = false;  
} else if (columna == -1) {  
    avanzar = false;  
}  
  
if (avanzar) {  
    String strResultado = tablaModelo.getValueAt(  
        jTableProcesos.getSelectedRow(),  
        jTableProcesos.getSelectedColumn()).toString();  
    JOptionPane.showMessageDialog(null, "Dato seleccionado : " + strResultado);  
} else {  
    JOptionPane.showMessageDialog(null, "No se ha seleccionado un registro");  
}
```

- El control Progress Bar es bastante fácil de usar. Agregue un nuevo JDialog y un Item en el menú que llame dicha ventana.
- Coloque en la ventana un progress Bar y dos botones. El resultado será similar a la imagen.



- Lo que haremos es que un botón incremente la barra y otro lo disminuya. Para ello, debe declarar la siguiente variable como global: `int cont=0;`
- Luego, en el botón de incrementar debe agregar el evento y el siguiente código.

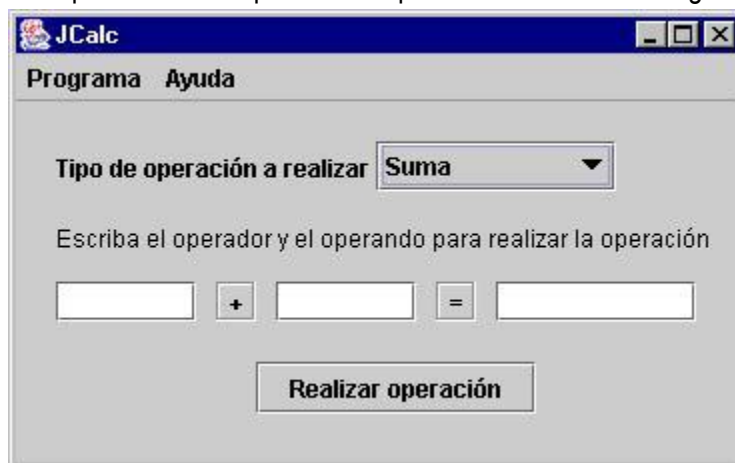
```
cont=cont+10;
jProgressBar1.setValue(cont);
```

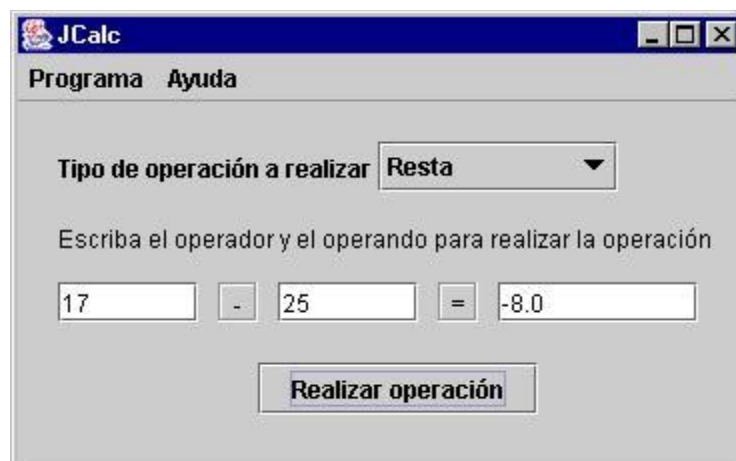
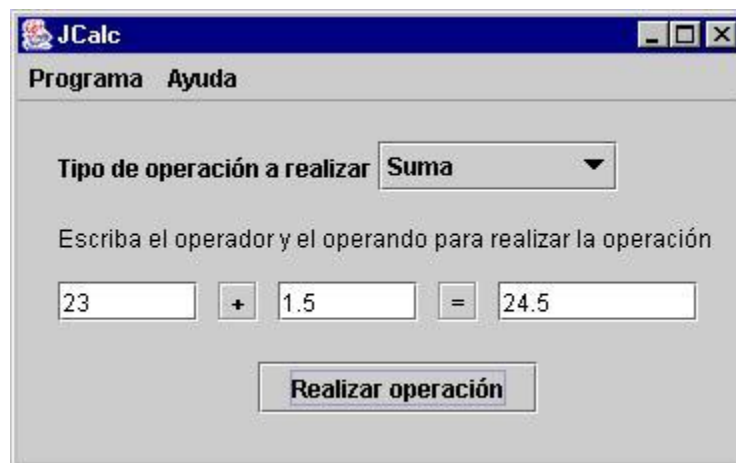
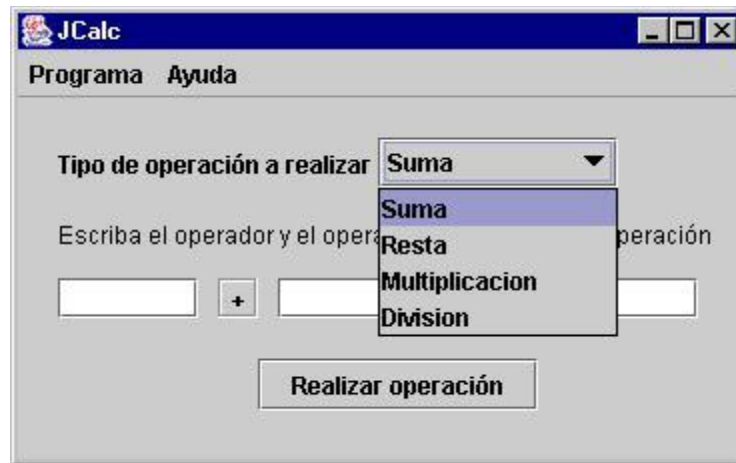
- Con el botón de disminuir, haremos similar solamente que el contador se disminuya en 10. Y agregamos una validación para que el contador no disminuya menos de cero.

```
if(cont>0)
    cont=cont-10;
jProgressBar1.setValue(cont);
```

## 2. Ejercicios

- Elabore una aplicación que realice las operaciones que se muestran en las siguientes pantallas.





JCalc

Programa Ayuda

Tipo de operación a realizar Multiplicacion ▼

Escriba el operador y el operando para realizar la operación

34 \* 3.5 = 119.0

Realizar operación

JCalc

Programa Ayuda

Tipo de operación a realizar Division ▼

Escriba el operador y el operando para realizar la operación

124 / 11 = 11.2727272727

Realizar operación

JCalc

Programa Ayuda

Tipo de operación a realizar Division ▼

Escriba el operador y el operando para realizar la operación

4763 / 0 = Infinity

Realizar operación