



Ingeniería del Software 2
Grado en Ingeniería Informática en Sistemas de Información - Curso 2012/2013
Enseñanzas Prácticas y de Desarrollo
EPD5: Programación con Java Swing usando NetBeans

Objetivos

- Conocer el API Swing y las mejoras que presenta frente al AWT.
- Manejar los componentes principales de JComponent: frames, etiquetas, iconos, botones, áreas de texto y componentes de elección.
- Dominar los conceptos de evento e interfaz de usuario
- Desarrollar una interfaz de usuario con NetBeans

Conceptos Teóricos Previos

1. Introducción.

En este apartado se pretende realizar una pequeña introducción a Swing, con la que se intenta mostrar al lector una visión general sobre el mundo de las interfaces gráficas de usuario, y más concretamente, sobre el desarrollo de éstas con Java y sus APIs. Como tan sólo se trata de una introducción al tema, lo que se pretende es captar la atención del alumno y mostrar las ventajas que ofrece éste tipo de programación.

En primer lugar se realizará una introducción a Swing, describiendo el entorno en que se encuentra, tratando por ello algunos aspectos relaciones con AWT, JFC y JKD. Además, se realizará una breve descripción de las interfaces de usuario. Posteriormente, se describe Swing con más profundidad, analizando su arquitectura y sus componentes principales. Y Por último, se realizará una pequeña introducción al concepto y manejo de eventos en Swing.

2. Introducción a Swing

Swing es una de las mejoras principales que ha experimentado el JDK en su versión 1.2 con respecto a la versión 1.1, y representa la nueva generación de AWT (*Abstract Window Toolkit*). También es una de las APIs de las Clases de Fundamentos de Java (JFC), lo cual es el resultado de un esfuerzo de colaboración entre SUN, Netscape, IBM y otras empresas. Lo que da a Swing su importancia es el poder que ofrece para desarrollar interfaces gráficas de usuario (GUI) para *applets* y aplicaciones. La cantidad y calidad de los controles GUI que ofrece Swing no tiene rival en ningún otro juego de herramientas GUI.

El origen de los controles GUI que presenta Swing lo encontramos en las Clases de Fundamentos de Internet de Netscape (IFC). Los componentes Swing van más allá de las IFC, hasta el punto de que no hay un parecido apreciable entre los componentes Swing y los de las IFC. Swing ofrece también la posibilidad de cambiar fácil y rápidamente el aspecto y sensación (L&F) de un único componente o grupo de éstos. Esta posibilidad, que se conoce como aspecto y sensación conectables (PL&F), es un sello distintivo de Swing.

3. Introducción a las Interfaces de Usuario y el AWT

Para poder apreciar la importancia de Swing, haremos primero una introducción a las interfaces de usuario y al AWT.

El interfaz de usuario es la parte del programa que permite a éste interactuar con el usuario. Las interfaces de usuario pueden adoptar muchas formas, que van desde la simple línea de comandos hasta las interfaces gráficas que proporcionan las aplicaciones más modernas. El interfaz de usuario es uno de los aspectos más importante de cualquier aplicación. Una aplicación sin un interfaz fácil, impide que los usuarios saquen el máximo rendimiento del programa. Java proporciona los elementos básicos para construir interfaces de usuario a través del AWT, y opciones para mejorarlas mediante Swing, que sí permite la creación de interfaces de usuario de gran impacto y sin demasiados quebraderos de cabeza por parte del programador.

Al nivel más bajo, el sistema operativo transmite información desde el ratón y el teclado como dispositivos de entrada al programa. El AWT fue diseñado pensando en que el programador no tuviese que preocuparse de detalles como controlar el movimiento del ratón o leer el teclado, ni tampoco atender a detalles como la escritura en pantalla. El AWT constituye una librería de clases



orientada a objetos para cubrir estos recursos y servicios de bajo nivel. Debido a que el lenguaje de programación Java es independiente de la plataforma en que se ejecuten sus aplicaciones, AWT también lo es. AWT proporciona un conjunto de herramientas para la construcción de interfaces gráficas que tienen una apariencia y se comportan de forma semejante en todas las plataformas en que se ejecute. Los elementos de interfaz proporcionados por el AWT están implementados utilizando *toolkits* nativos de las plataformas, preservando una apariencia semejante a todas las aplicaciones que se creen para esa plataforma. Este es un punto fuerte del AWT, pero también tiene la desventaja de que un interfaz gráfico diseñado para una plataforma, puede no visualizarse correctamente en otra diferente. Estas carencias del AWT son subsanadas en parte por Swing, y en general por las JFC.

4. Swing, el AWT y las JFC

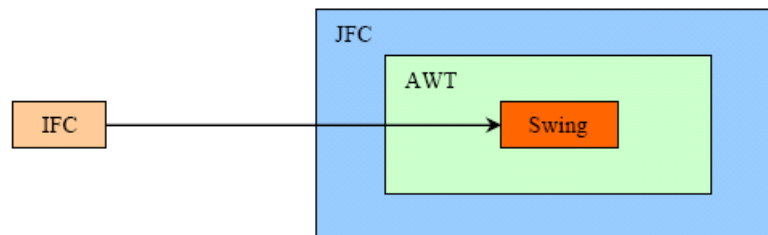


Figura 1. Swing, IFC, JFC y AWT

La Figura 1 muestra la relación existente entre Swing, el AWT y las JFCs. Las JFCs subsumen y amplían el AWT original, y constan de las siguientes APIs principales:

- AWT
- Swing
- Java 2D
- Drag-and-Drop
- Accessibility

Aunque Swing está separado del AWT, se implementa en términos de clases AWT básicas. El AWT proporciona la interfaz entre el sistema de ventanas nativo subyacente y los componentes GUI de Java. Swing utiliza esta interfaz, pero no se apoya en componentes del AWT para hacer uso de objetos nativos. En lugar de ello, los componentes Swing están escritos en Java puro. Esto ofrece ventajas significativas. Permite a los componentes Swing ser independientes del sistema de ventanas nativo, lo cual implica que pueden ejecutarse en cualquier sistema de ventanas que admita el AWT. También permite a los componentes Swing ser independientes de cualquier limitación de los sistemas de ventanas nativos. Esta independencia permite a Swing controlar y adaptar su aspecto y sensación (de ahí la aparición de PL&F).

Entre los componentes nuevos que incluye Swing hay desde paneles tabulados y bordes estilizados hasta barras deslizadoras y efectos giratorios. Estos componentes nuevos, en sí mismos, hacen que Swing constituya un agregado de primera magnitud a la API Java. La galería de componentes Swing, que se encuentra en <http://java.sun.com/products/jfc/swingdoc-current>, muestra algunos de ellos. Swing también incorpora un programa de demostración llamado *SwingSet* (<http://java.sun.com/products/javawebstart/demos-nojavascript.html>).

Por otro lado, Swing ofrece una implementación de Java puro de muchos de los componentes del AWT. Estos componentes tienen la misma funcionalidad que los componentes del AWT y todas las ventajas de Swing. Swing es compatible con el AWT, y los componentes Swing se pueden utilizar con los del AWT. Sin embargo, los componentes Swing sólo se pueden usar con el modelo de eventos del JDK 1.1 o superior.

La arquitectura PL&F de Swing facilita la personalización tanto del aspecto como del comportamiento de cualquier control Swing o de cualquier grupo de estos controles. Swing también incorpora varios L&F predefinidos, entre los que reincluye el Metal L&F predeterminado, el Motif L&F y el Windows L&F. Los L&F para Macintosh y otras plataformas también se están desarrollando.

5. La jerarquía de componentes Swing

Swing consta de nueve paquetes y cientos de clases e interfaces. No obstante, la clase *JComponent* de `java.awt.swing` es la clase superior de la jerarquía de componentes Swing. La clase *JComponent* es una subclase de `java.awt.container` y, por tanto,



es a la vez un componente y un contenedor en el sentido del AWT. Dado que JComponent es la superclase de todos los componentes Swing, todos ellos descienden de java.awt.Container y java.awt.Component.

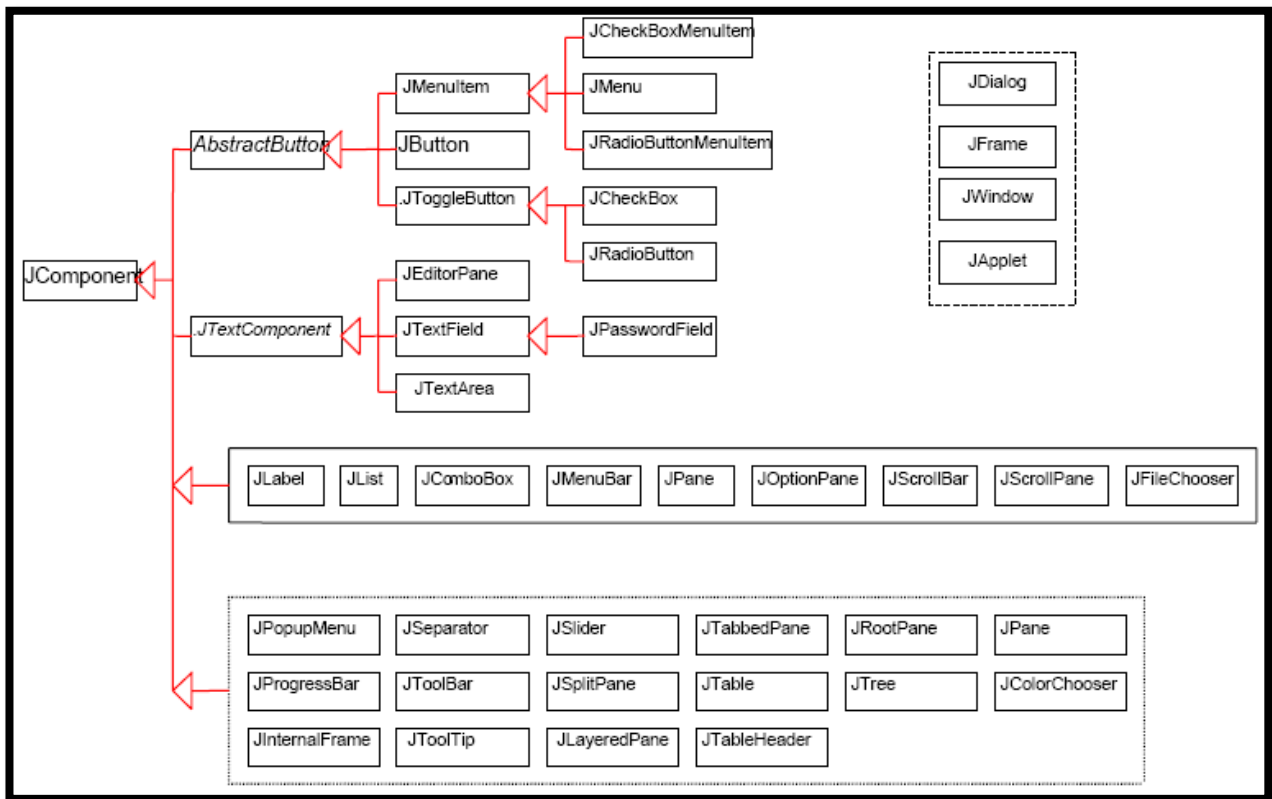


Figura 2. Jerarquía de componentes Swing

La Figura 2 muestra la jerarquía de componentes Swing. La primera cosa que se debe tener en cuenta es que todos los componentes empiezan por la letra J, seguida por el tipo de componente que admite la clase.

6. Construcción de GUI en Swing

Una vez determinado qué es Swing y cómo se estructura, pasaremos a describir los conceptos y clases principales que intervienen en la construcción de GUI usando Swing.

6.1. Swing: Ventana JFrame

Descripción:

- Ventana dependiente de escritorio
- Posee decoración, icono y controles

Constructores

- JFrame(String titulo);

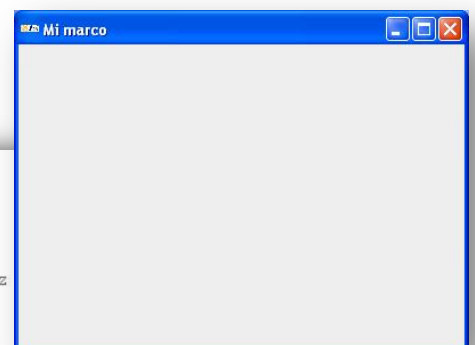
Propiedades y métodos:

- setSize(int x, int y);
- setLocation(int x,int y);
- setVisible(boolean b);
- pack(); //colocar
- add(Component...);
- setIconImage(ImageIcon i)
- Tamaño pantalla: Dimension dim= getToolkit().getScreenSize();

```
package gui;
import javax.swing.*;

/**
 *
 * @author Norberto Diaz-Diaz
 */
public class Ventana {

    public static void main(String[] args){
        JFrame frame = new JFrame("Mi marco");
        frame.setSize(400,300);
        frame.setVisible(true);
        ImageIcon g= new ImageIcon("upo.gif");
        frame.setIconImage(g.getImage());
    }
}
```





6.2. Swing: Propiedades de componentes

- text: Texto presente en el componente
- icon: Icono asociado al componente (gif, jpg)
- toolTipText: Texto alternativo (curso sobre componente)
- font
- background } Estilo
- foreground }
- doubleBuffered: Dibujo (gráficos)
- border: Estilos de bordes del componente
- preferredSize } Ajuste de tamaños preferentes
- minimumSize }
- maximumSize }
- mnemonic: Acceso alternativo por Teclado (ALT + carácter 'C') → `comp.setMnemonic('C');`

6.3. Etiquetas e Iconos

Añade componentes no interactivos al interfaz.

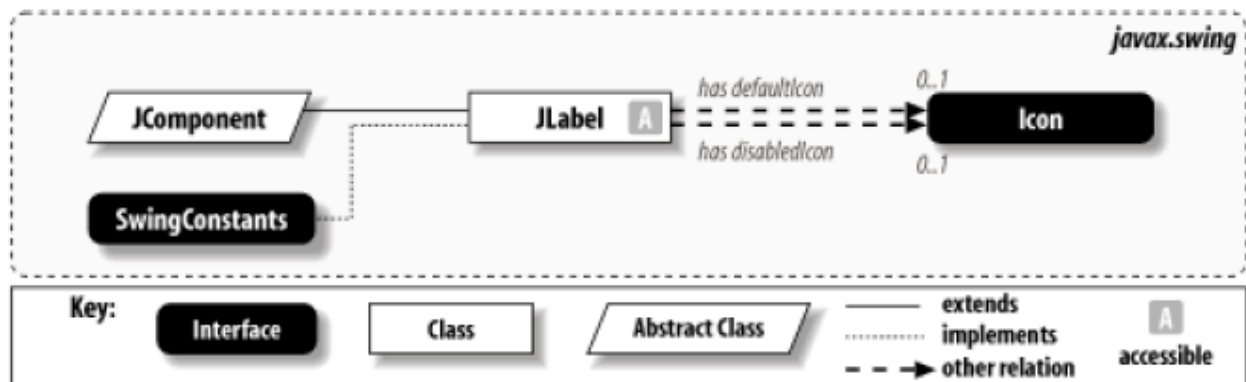


Figura 3. Jerarquía de Etiquetas e Iconos

6.3.1. JLabel

Descripción:

- Incluye texto y/o imagen no seleccionable en una línea
- Control de la alineación

Constructores

- JLabel(String s);
- JLabel(Icon i);
- JLabel(String s, int alin_hor); // LEFT | RIGHT | CENTER
- JLabel(Icon i, int alin_hor);

Propiedades y métodos:

- String texto;
- Icon icono;
- int horizontalAlignment
- setText(String s);
- String getText();
- setIcon(new ImageIcon("upo.gif"));
- setHorizontalAlignment(int); // LEFT, CENTER, RIGHT
- setVerticalAlignment(int); // TOP, CENTER, BOTTOM

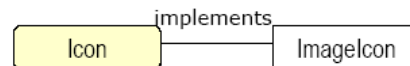




6.3.1. Iconos (ImageIcon)

Descripción:

- Implementa clase abstracta Icon
- Visualiza imágenes .jpg/.gif
- Componentes → método *setIcon*



Constructores

- ImageIcon(String Fichero)

Propiedades y métodos:

- String texto;
- Icon icono;
- int horizontalAlignment
- setImage(String s);
- getImage()

```
ImageIcon icono=new ImageIcon("ok.gif");  
...  
Icono.setImage("upo.gif");  
JLabel et=new JLabel(icono);
```

6.4. Botones

Los botones son componentes que capturan eventos generados por el usuario (por pulsación). Éstos pueden representar texto y/o iconos. El estado de los botones es definido por la clase DefaultButtonModel, mientras que ButtonGroup permite agrupar varios de éstos.

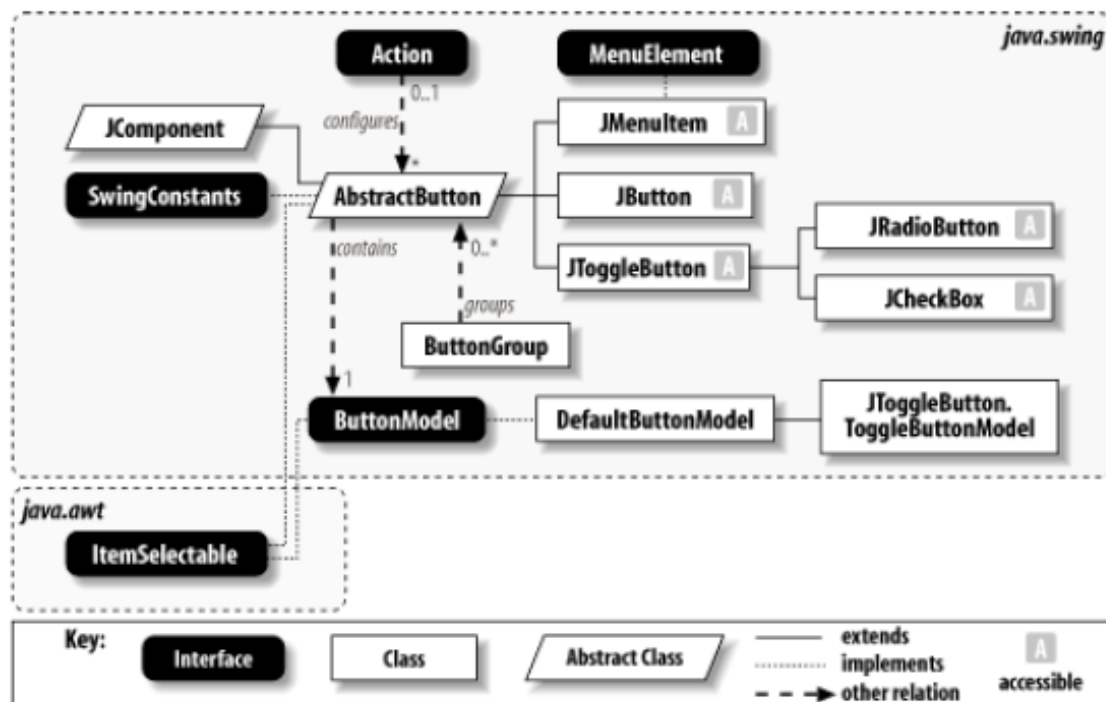


Figura 4. Jerarquía y relaciones de los distintos componentes de Botones



6.4.1. DefaultButtonModel

En la figura 5, se resume los diferentes conceptos que hay que tener en cuenta para manejar el estado de un botón.

Propiedad	Tipo Dato	get	is	set	Comentario
actionCommand	String	X		X	Texto asociado al botón (para un ActionEvent) para el manejo de eventos
enabled	boolean		X	X	Estado de Habilitado (debe estar habilitado para ser pulsado)
group	ButtonGroup			X	Contenedor (si existe) del botón
mnemonic	int	X		X	Tecla de atajo junto a la combinación de tecla relativa al comportamiento del Look&Feel por defecto. Tecla definida mediante código (p.e. KeyEvent.VK_A).
pressed	boolean		X	X	Estado de pulsado
rollover	boolean		X	X	Indica si el cursor está sobre el botón
selected	boolean		X	X	Indica si el botón está seleccionado

Evento	Descripción
ActionEvent	El botón es presionado
ChangeEvent	Ha sucedido algún cambio en las propiedades (el estado) del botón
ItemEvent	El botón permuta de pulsado a soltado

Figura 5. Conceptos asociados al estado de un botón (DefaultButtonModel)

6.4.2. JButton

Descripción:

- Control de pulsación
- Puede contener iconos

Constructores

- JButton(String text)
- JButton(Icon icon)
- JButton(String text, Icon icon)



Propiedades y métodos:

- String texto;
- Icon icono;
- int mnemonic
- setText(String);
- setIcon(Icon);
- setRolloverIcon(Icon); //muestra al estar encima del botón
- setDisabledSelectedIcon();
- setPressedIcon();

```
JButton jb = new JButton();
jb.setIcon(new ImageIcon("image.jpg"));
```

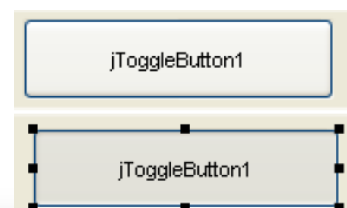
6.4.3. JToggleButton

Descripción:

- Botón con estado (seleccionado/deseleccionado)

Constructores

- JToggleButton(String text)
- JToggleButton(Icon icon)
- JToggleButton(String text, Boolean selected);



```
JToggleButton jtb = new JToggleButton();
jtb.setSelected(true);
```



Propiedades y métodos:

- boolean getSelected();
- void setSelected(boolean)

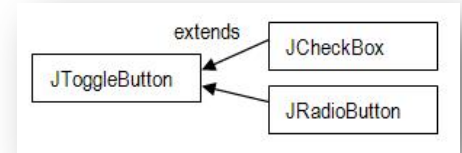
6.4.4. JCheckBox/JRadioButton

Descripción:

- Control de pulsación
- Puede contener iconos

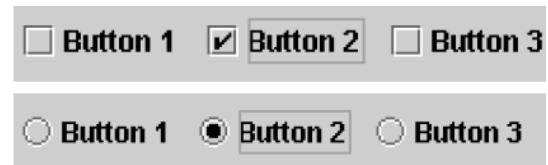
Constructores

- JCheckBox/JRadioButton(String text [, boolean selec])
- JCheckBox/JRadioButton(Icon icon [, boolean selec])
- JCheckBox/JRadioButton(String text, Icon icon [, boolean selec])



Propiedades y métodos:

- String texto;
- Icon icono;
- int mnemonic;
- boolean selec;
- setText(String);
- setIcon(Icon);
- boolean isSelected();
- setSelected(Boolean b)
- Boolean isSelected();
- Emite un ItemEvent al cambiar de estado (ItemStateChanged): ItemEvent.SELECTED, ItemEvent.DESELECTED.



6.4.5. ButtonGroup

Descripción:

- Agrupa un conjunto de JRadioButton
- Permite simular la Selección excluyente

Constructores

- ButtonGroup()

Propiedades y métodos:

- String texto;
- Icon icono;
- int mnemonic;
- boolean selec;
- add/remove(AbstractButton); //añade/quita botones
- setSelected(ButtonModel m, boolean b);
- boolean isSelected(ButtonModel m); //comprueba si está seleccionado
- int getButtonCount(); //número de botones

```
ButtonGroup bg1 = new ButtonGroup();  
bg1.add(new JCheckBox("Bold"));  
bg1.add(new JCheckBox("Italic"));
```





6.5. Texto

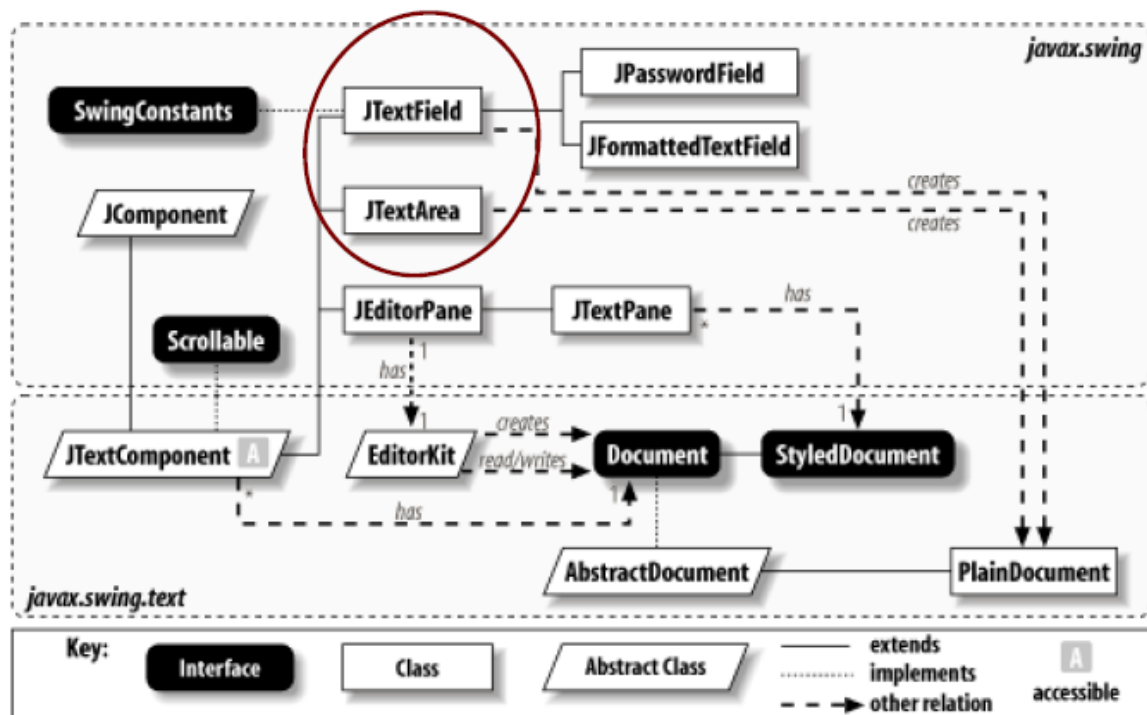


Figura 6. Jerarquía y relaciones de los distintos componentes de Texto

6.5.1. JTextField/JPasswordField

	JTextField	JPasswordField
Descripción	Texto editable (una línea)	Contraseña (una línea)
Constructores	JTextField();	JPasswordField();
	JTextField(String, int); //nº columnas	JPasswordField(String, int); //nº columnas
	JTextField(String);	JPasswordField(String);
Propiedades	text, editable, columns	
Métodos	setText(String)	setPassword(String);
	setColumns(int); //nº columnas	setColumns(int); //nº columnas
	setEditable(boolean editable)	setEchoChar(char); //carácter
		boolean echoCharIsSet();

6.5.2. JTextArea

Descripción:

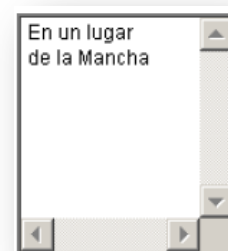
- Entrada de texto (filas y columnas)
- Permite simular la Selección excluyente

Constructores

- JTextArea(String t, int n_fila, int n_col);

Propiedades y métodos:

- text, editable, columns, rows
- lineCount





- setText(String);
- String getSelectedText();
- insert(String s, int pos); //inserta texto en pos.
- replaceRange(String s, int inicio, int fin); //sustituye texto en el rango
- append(String s); //añade al final

6.6. Elección

Seguidamente se presentan aquellos componentes que permiten elegir una opción entre una serie de alternativas. Estos componentes permiten una selección simple/múltiple. El modo de selección múltiple depende de Look&Feel; usando CTRL, SHIFT...

6.6.1. JList

El componente JList consta de tres partes: un modelo de datos (ListModel), un modelo de selección (ListSelectionModel) y un modelo de representación (ListCellRenderer). Cada una de estas parte y la relación entre ellos y el componente JList es recogido en la La figura 7.

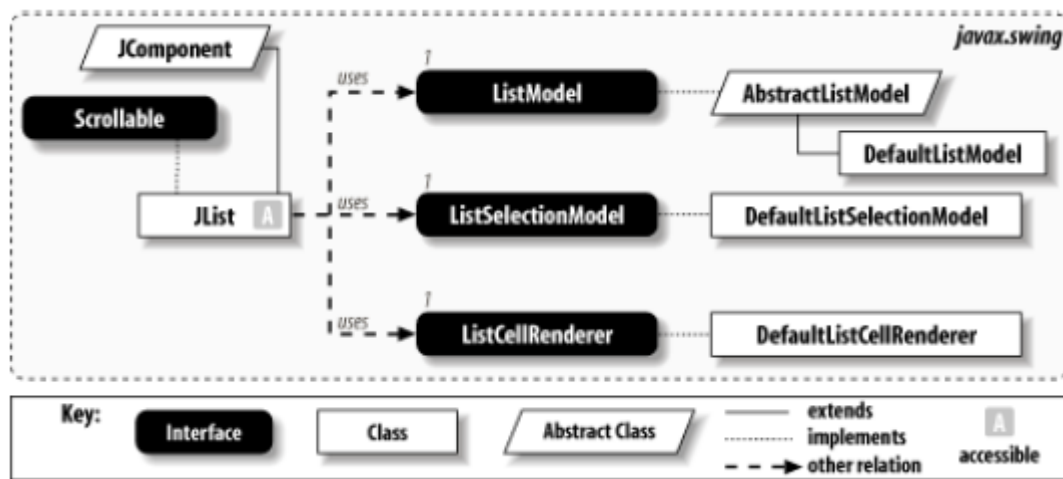


Figura 7. Jerarquía y relaciones del componente JList

Por otro lado, la figura 8 presenta las nociones más importantes y que hay que tener en cuenta para usar el componente JList.

Propiedad	Tipo	Get	Set	Descripción
selectionMode	int	X	X	Modo (MULTIPLE_INTERVAL_SELECTION)
VisibleRowCount	int	X	X	Número de columnas visibles
selectedIndex	Int	X	X	Devuelve el índice del ítem seleccionado
Model	ListModel	X	X	Asocia un modelo de datos a la lista

Figura 8. Conceptos más relevante del componente JList

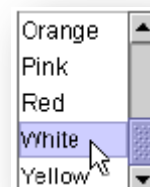
Seguidamente, se presenta un resumen de las características fundamentales de JList:

Descripción:

- Lista de elementos (para selección)
- Admite diferentes modos de selección
- No dispone de barra de desplazamiento (scroll) automáticamente, sino que se debe insertar en un panel JScrollPane.
- Listas sencillas (tamaño fijo) o variable (ListModel)

Constructores

- JList(Object[] elementos); //array de String para visualizar.
- JList(ListModel modelo); // modelo de datos complejos





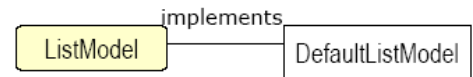
Propiedades y métodos:

- String texto;
- Icon icono;
- int modo; //SINGLE_SELECTION, SINGLE_INTERVAL_SELECTION, MULTIPLE_INTERVAL_SELECTION
- void set VisibleRowCount(int filas); //columnas visible
- void setSelectionMode(int modo); //modo selección

6.6.2. ListModel/DefaultListModel

Descripción:

- Lista de elementos (para selección)
- Adecuado para: listas largas, cambiantes, de objetos, etc.
- Control para el manejo de la lista (Interfaz ListModel).
- Existe una implementación predefinida (DefaultListModel)



Propiedades y métodos:

- String texto;
- Icon icono;
- int modo; //SINGLE_SELECTION, SINGLE_INTERVAL_SELECTION, MULTIPLE_INTERVAL_SELECTION
- void set VisibleRowCount(int filas); //columnas visible
- void setSelectionMode(int modo); //modo selección
- void addElement(Object o); //añade elemento al final de la lista
- void removeElement(Object o); //elimina la primera ocurrencia del objeto en la lista
- void removeAllElementes();

6.6.3. JComboBox

El componente JComboBox combina una entrada de texto con una lista desplegable. De forma que, el primer elemento (índice 0) aparece como seleccionado por defecto.

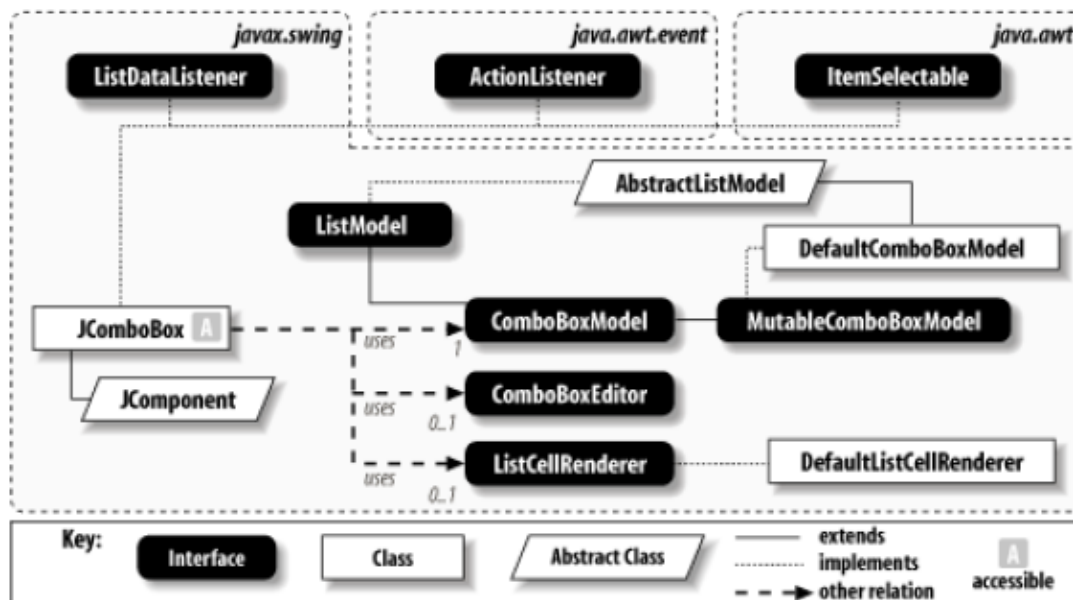


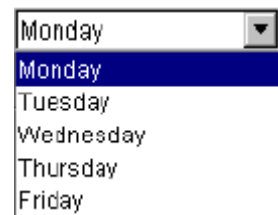
Figura 9. Jerarquía y relaciones del componente JComboBox

Nótese, en la figura 9, que ComboBoxModel extiende a ListBoxModel para permitir una única selección. Mientras que MutableComboBoxModel permite listas editables.

Seguidamente, se presenta un resumen de las características fundamentales de JComboBox:

Descripción:

- Combinación de entrada de texto con lista desplegable
- Posee ScrollBar automático
- El primer elemento aparece como seleccionado

Constructores

- JComboBox()
- JComboBox(ComboBoxModel aModel);
- JComboBox(Object[] items); //array de elementos

Propiedades y métodos:

- int getSelectedItem; //selección actual
- void setMaximumRowCount(int n); //número de elementos a mostrar en el combo
- int getItemCount(); //contabilizar nº de elementos
- void setEditable (boolean b); //si permite realizar cambios en la lista

6.6.4. ComboBoxModel/DefaultComboBoxModel

Descripción:

- Modelo para representar los datos de un JComboBox
- Permite edición

Constructor:

- DefaultComboBoxModel();

```
JComboBox jcb = new JComboBox();
DefaultComboBoxModel m=new DefaultComboBoxModel();
m.addElement ("Monday"); m.addElement ("Tuesday");
m.addElement ("Wednesday"); m.addElement ("Thursday");
m.addElement ("Friday");
jcb.setModel (m);
jcb.setEditable (true);
```

Propiedades y métodos:

- void addElement(Object o); //añade elemento al final de la lista
- void insertElementAt(Object obj, int index); //inserta un element en la posición dada
- void removeElement(Object o); //elimina la primera ocurrencia del objeto en la lista
- void removeAllElements();

7. Manejo de Eventos en Swing

Cada vez que el usuario teclea un carácter o pulsa un botón del ratón, ocurre un evento. Cualquier componente puede ser notificado del evento. Todo lo que tiene que hacer es implementar el interfaz apropiado y ser registrado como un oyente del evento fuente en cuestión. Los componentes Swing pueden generar muchas clases de evento.

El paquete java.awt.swing.event define una serie de interfaces auditoras de eventos y clases de eventos que se usan con los componentes Swing. Además, muchos de los componentes Swing también utilizan eventos del AWT.

En la figura 10 se muestran algunos ejemplos:

Acción que resulta en el evento	Tipo de oyente
El usuario pulsa un botón, presiona Return mientras teclea en un campo de texto, o elige un ítem de menú.	ActionListener
El usuario elige un frame (ventana principal).	WindowListener
El usuario pulsa un botón del ratón mientras el cursor está sobre un componente.	MouseListener
El usuario mueve el cursor sobre un componente.	MouseMotionListener
El componente se hace visible.	ComponentListener
El componente obtiene el foco del teclado.	FocusListener
Cambia la tabla o la selección de una lista.	ListSelectionListener

Figura 10. Ejemplos de Eventos en Swing

Cada evento está representado por un objeto que ofrece información sobre el evento e identifica la fuente. Las fuentes de los eventos normalmente son componentes, pero otros tipos de objetos también pueden ser fuente de eventos. Además, cada fuente de evento puede tener varios oyentes registrados. Incluso, es posible que un solo oyente pueda registrarse con varias fuentes de eventos.



Bibliografía

Básicas:

- API J2SE: <http://java.sun.com/j2se/1.5.0/docs/api/>
- The Swing Tutorial: <http://java.sun.com/docs/books/tutorial/uiswing/index.html>
- The Swing Tutorial (traducción): <http://www.programacion.com/java/tutorial/swing/>
- Learning Swing by Example: <http://java.sun.com/docs/books/tutorial/uiswing/learn/index.html>
- *Diseño Interfaz Usuario*:
Roger S. Pressman. Ingeniería del Software. Un enfoque práctico (Edición 6 Traducido). Editorial Mac Graw Hill. ISBN 970-10-573-3. Capítulo 12, páginas 350-381

Libros (on-line):

- Thinking in Java, 3rd Edition: <http://www.mindview.net/Books/TIJ/>

Varios:

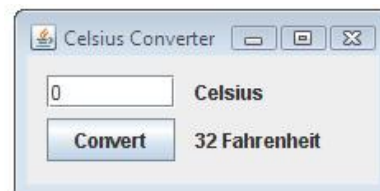
- Swing Example: <http://java.sun.com/products/javawebstart/demos-nojavascript.html>

Experimento

Una vez presentados las nociones básicas sobre interfaces gráficas de usuario, el uso de Swing y sus componentes y el manejo de eventos, pasaremos a realizar una introducción a la programación de GUI con Swing y NetBeans IDE.

La finalidad de esta sección es introducir el manejo de la API Swing con NetBeans desarrollado una aplicación simple que convierta la temperatura medida en grados Celsius a grados Fahrenheit. Su interfaz gráfica será básica, centrándonos sólo en un subconjunto de componentes disponibles de Swing. Para la creación de ésta usaremos el constructor NetBeans IDE GUI, el cual hace que la creación de la interfaz de usuario sea cuestión de arrastrar y soltar.

La GUI final de esta aplicación tendrá el siguiente aspecto:



➤ Paso 1. Crear el proyecto CelsiusConverter

Como primer paso, deberemos crear un proyecto de aplicación general en Netbeans.

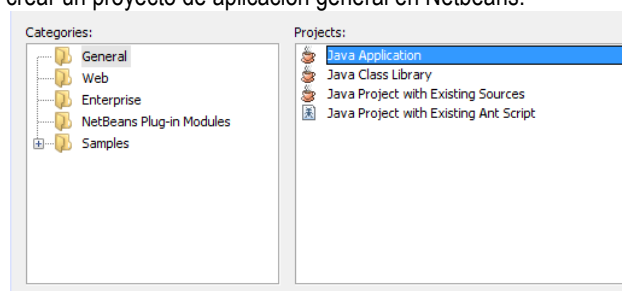


Figura 11. Creación de proyecto

Seguidamente deberemos añadir un JFrameForm, que será la clase Swing responsable del frame principal de la aplicación. Con tal fin, pinchando con el botón derecho sobre el nombre del proyecto, elija New→JFrame Form. El nombre de esta clase será "CelsiusConverterGUI".

Una vez finalizado el proceso anterior, el panel derecho mostrará el aspecto de la clase en cuestión (ver figura 12). Es sobre esta pantalla donde se arrastrará, soltará y manipulará los distintos componentes Swing.

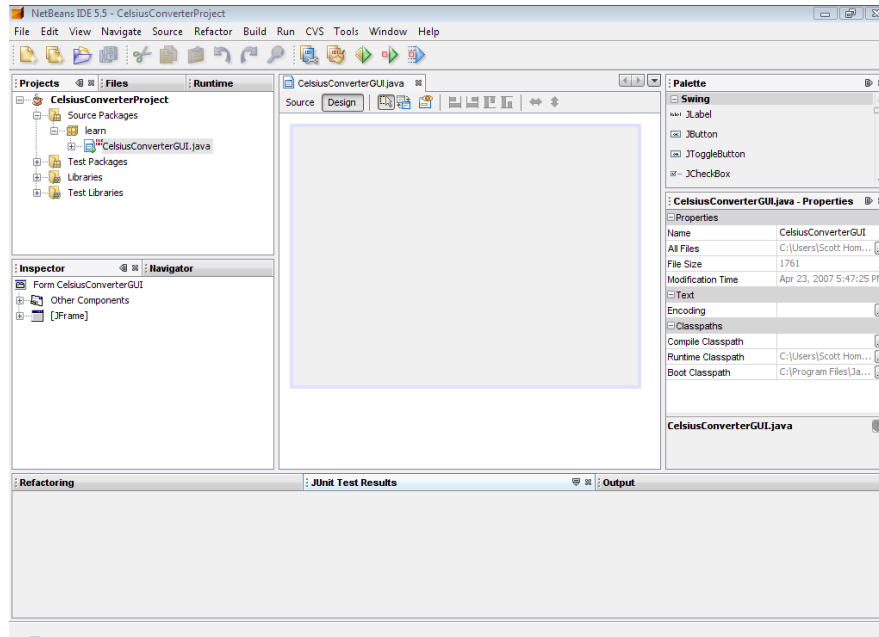


Figura 12. Pantalla de NetBeans para manipular Swing

En la figura 12 se puede observar cuatro partes nuevas fundamentales: paleta, área de diseño, editor de propiedades y el panel de revisión (*inspector*).

La paleta contiene todos los componentes que ofrece Swing.

El área de diseño, parte central superior, es donde se visualizará la construcción de la GUI. Ésta consta de dos vistas: de código y de diseño. Esta última es la vista por defecto, aunque se pueden intercambiar las dos vistas indistintamente. La figura 13 muestra el código fuente de la clase principal de nuestra aplicación. En esta se puede observar que NetBeans ha creado automáticamente un método privado denominado *initComponent*, que será donde se inicializarán los distintos componentes de la GUI.

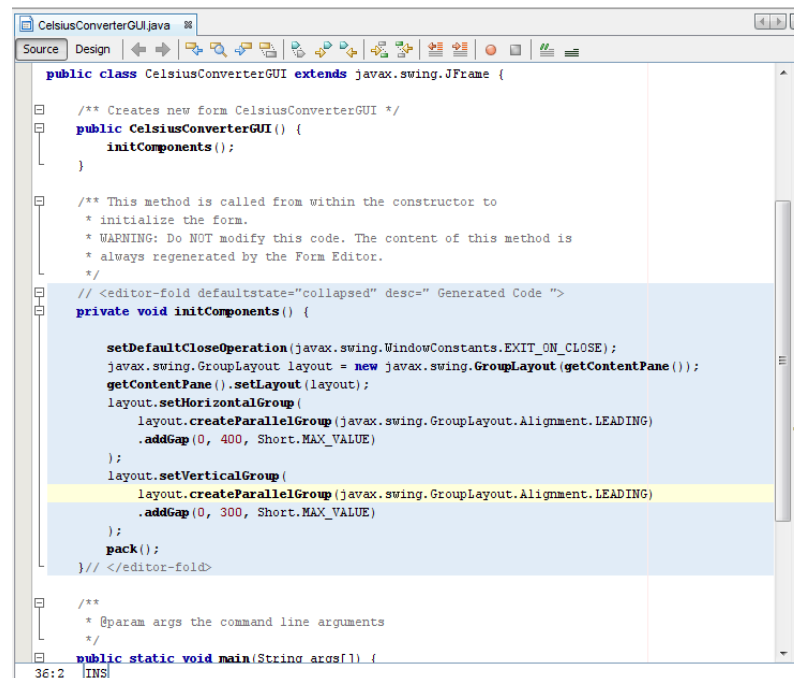


Figura 13. Código fuente, autogenerado, de CelsiusConverterGUI.java



Por otro lado, el editor de propiedades, que como su propio nombre indica, permite editar las propiedades de cada componente. Este panel es muy intuitivo, mostrando por cada fila una propiedad concreta, la cual puedes modificar con un simple click sin entrar en el código fuente del programa.

Por último, el panel revisor provee de una representación gráfica de los componentes de la aplicación. Este panel es muy útil para cambiar el nombre de las variables que se genera automáticamente.

➤ Paso 2. Creación de la GUI

Llegado este paso, donde tenemos creado la clase principal de nuestra interfaz gráfica, añadiremos los diferentes componentes necesarios para realizar el cometido que nos propusimos al comienzo.

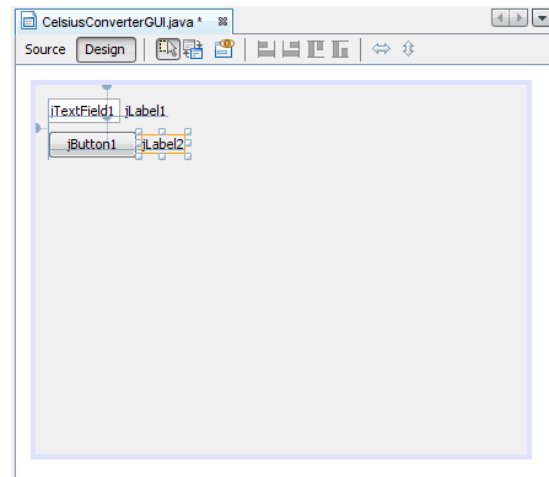
En primer lugar, cambiaremos el título de la aplicación. Para ello deberemos acceder a las propiedades del JFrame e indicaremos en el editor de propiedades que el título es: "Celsius Converter".

Seguidamente, añadimos un JTextField, que será el componente encargado de almacenar el número de grados Celsius. Con tal fin, desplazamos desde la paleta al área de diseño un componente de la mencionada clase. De la misma forma, añadimos dos componentes JLabel y uno JButton.

Una vez insertado todos los componentes de los que constará nuestra aplicación pasaremos a darle las propiedades que deseamos a cada uno de ellos. La primera propiedad a cambiar son los nombres de las dos etiquetas, las cuales las tenemos que cambiar a "Celsius" y "Fahrenheit" para JLabel1 y JLabel2, respectivamente. De la misma manera, modificamos el texto del botón a "Convert".

Posteriormente, se ha de establecer el mismo tamaño para el botón y el área de texto. Para ello, y teniendo seleccionado ambos componentes, se ha de hacer clic con el botón derecho para, posteriormente, elegir: "Same Size→Same Width".

Finalmente, queda reducir el JFrame para que no exista espacio vacío en exceso.

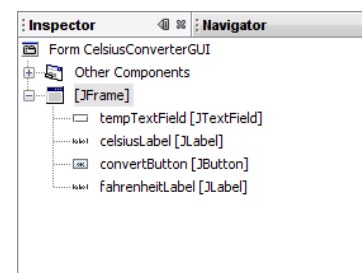


➤ Paso 3. Añadir funcionalidad

Como último paso, queda añadir la funcionalidad requerida a los distintos componentes.

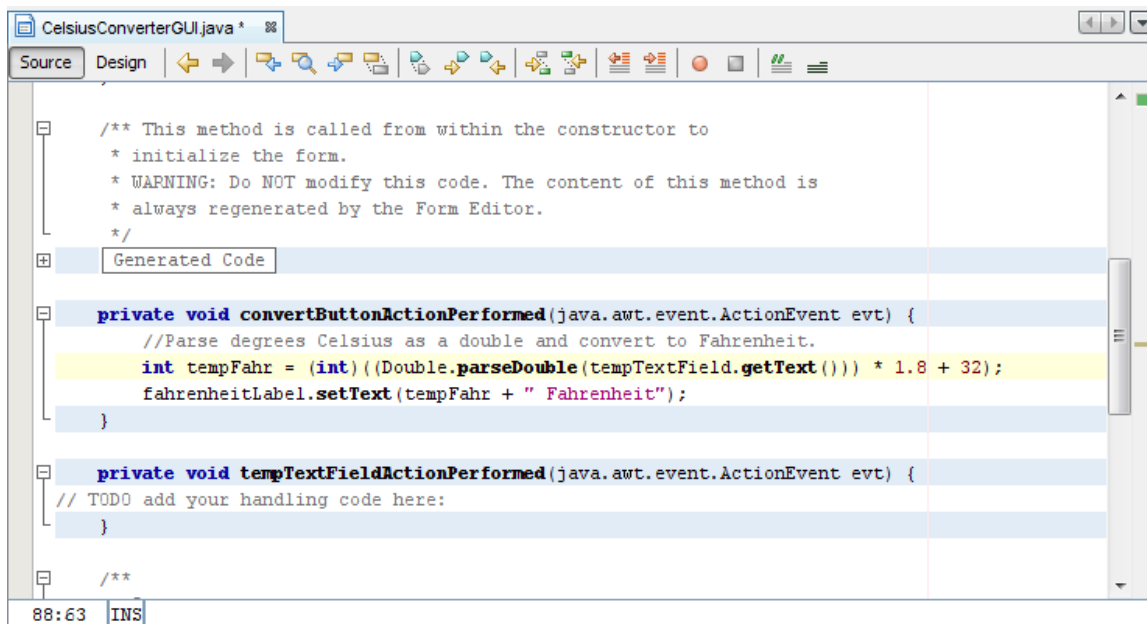
Ya que a partir de ahora tendremos que navegar hacia el código fuente es muy recomendable cambiar el nombre de las variables que tienen asociado cada componente. Nótese que cada uno de los componentes insertados en el paso anterior son, en realidad, atributos de nuestra clase principal. Para cambiar el nombre de estas variables basta con realizarlo en el panel "Inspector".

Llegado a este punto, lo único que queda es darle la funcionalidad al botón de que cuando se pinche sobre él calcule la conversión de los grados indicados en "tempTextField" y cambie el texto de "fahrenheitLabel". Para ello debemos incluir al botón un "Event Listeners" que escuche el evento mencionado. Con tal fin, desde la venta de diseño, pulsaremos sobre el componente botón en cuestión el botón derecho del ratón y elegiremos "Event→ActionPerformed". De esta manera, se generará el código del controlador de tal evento. Nótese que el código del controlador se almacena en un método () y que este es invocado cuando ocurre tal evento.





Por último, queda incluir el código pertinente para realizar la conversión solicitada ($^{\circ}\text{F} = ^{\circ}\text{C} \cdot 1.8 + 32$).



```
/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
Generated Code

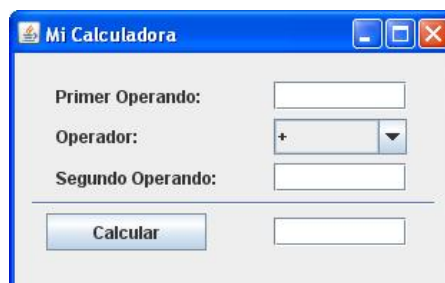
private void convertButtonActionPerformed(java.awt.event.ActionEvent evt) {
    //Parse degrees Celsius as a double and convert to Fahrenheit.
    int tempFahr = (int) ((Double.parseDouble(tempTextField.getText())) * 1.8 + 32);
    fahrenheitLabel.setText(tempFahr + " Fahrenheit");
}

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

/**
```

Ejercicio

Realizar una aplicación que simule una calculadora básica. Las operaciones que debe permitir son: suma, resta, multiplicación y división. El aspecto de la aplicación deberá ser el siguiente:



Además de las operaciones anteriores, debe notificar, en una ventana diferente, de cualquier error que se produzca en la calculadora. Los fallos pueden ser: división por cero o que se introduzca letras y no números. Para este último se debe hacer usando excepciones.

Por último, el campo que presenta la solución deberá tener a falso los campos "editable" y "enable". De esta forma no se podrá reescribir este campo ni posicionar el foco de la aplicación.

Problemas

EJ1. (30 mins) Cree una aplicación swing que contenga un botón e informe en la misma ventana el número de veces que se ha hecho clic sobre el botón.

EJ2. (20 mins) Modifique el ejercicio anterior utilizando un botón con estados y cuente el número de veces que el botón ha estado seleccionado.



EJ3. (60 mins) Desarrolle una aplicación swing que pueda ser usado en una votación entre varios candidatos. La idea es que se seleccione a un candidato y después se cliquee sobre el botón de “votar”. Este proceso se repetirá hasta que demos por cerrada la votación. Una vez haya finalizo, se informará en un diálogo externo (usando JDialog) el candidato que ha ganado.

EJ4. (90 mins) Implemente una aplicación capaz de capturar los datos de asignaturas y alumnos. La aplicación tendrá una interfaz gráfica con tres paneles bien diferenciados. Uno dedicado a capturar los datos de las asignaturas, otro de los alumnos y el último para establecer qué alumnos pertenecen a qué asignatura.



Datos de la Práctica

Autor del documento: Norberto Díaz Díaz (Marzo 2013).

Revisiones:

Estimación temporal:

- Parte presencial: 120 minutos.
 - Experimento: 60 minutos.
 - Ejercicio: 60 minutos.
- Parte no presencial: 270 minutos.
 - Lectura y estudio del guión y Bibliografía: 70 minutos
 - Problemas: 200 minutos