# SWING

Encarnación Sánchez Gallego

# CONTENIDOS

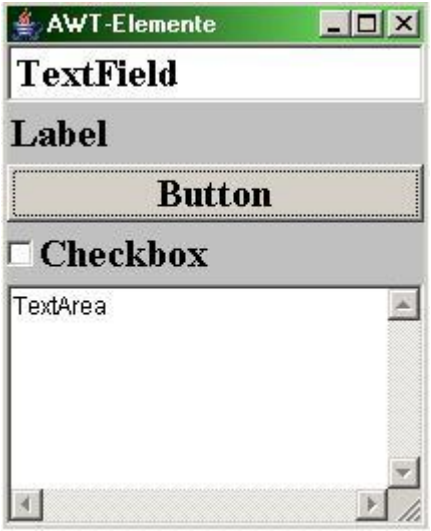
- ORIGEN DE LAS LIBRERÍAS GRÁFICAS
- LIBRERÍA AWT
- LIBRERÍA SWING

# ORIGEN DE LAS LIBRERÍAS GRÁFICAS

Desde las primeras versiones de Java, han existido **bibliotecas** de Clases Java o **API** (Application Programming Interface - Interfaz de programación de aplicaciones) que han aportado mayor funcionalidad a nuestros programas.

De hecho una de las razones de su éxito es la posibilidad desde las primeras versiones, de una gran cantidad de paquetes de clases destinadas al diseño de interfaces gráficas (mientras que en otros lenguajes dependían de terceros).

En concreto, uno de estos paquetes de librerías o bibliotecas disponible desde la versión original e incluida después en la versión 1.1 (principios del **97**) es java.awt o **AWT** (Abstract Window Toolkit). Este paquete contiene clases destinadas a la creación de objetos gráficos e imágenes.

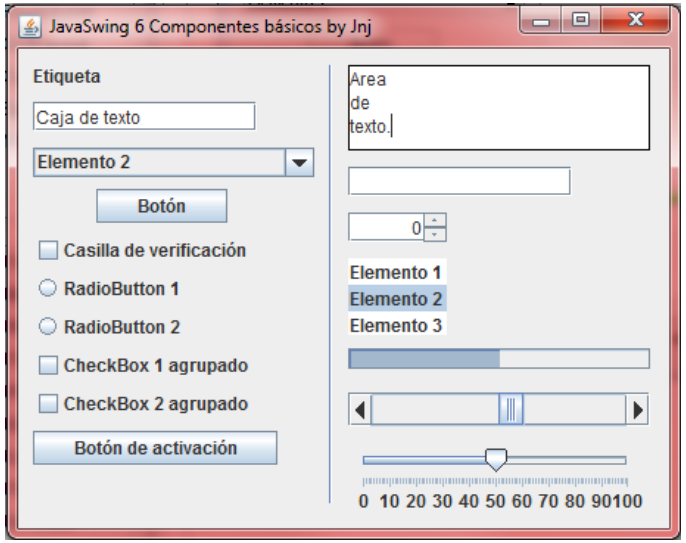


# ORIGEN DE LAS LIBRERÍAS GRÁFICAS

A finales del año **98**, en la versión 1.2 de Java se integra la biblioteca o API gráfica **Swing** en las clases básicas convirtiéndose en la librería utilizada para realizar aplicaciones gráficas al incluir widgets tales como cajas de texto, botones, listas desplegables y tablas.

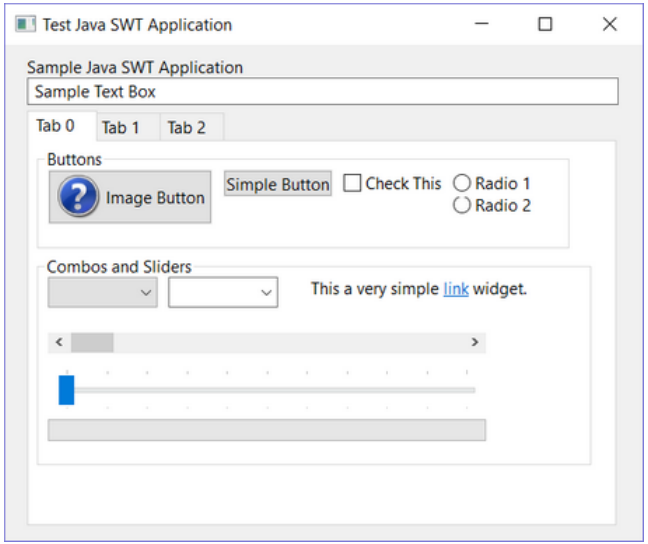
**Swing** está basado en la AWT (Abstract Window Toolkit) y es un kit de herramientas de gráficos, interfaz de usuario y sistema de ventanas independiente de la plataforma original de Java.

Programar interfaces gráficas es más complejo que interfaces de texto en las que solo es necesario programar la entrada y salida por teclado, pero también permite una interacción más rica y versátil, dando como resultado una experiencia de usuario más sencilla y amigable.

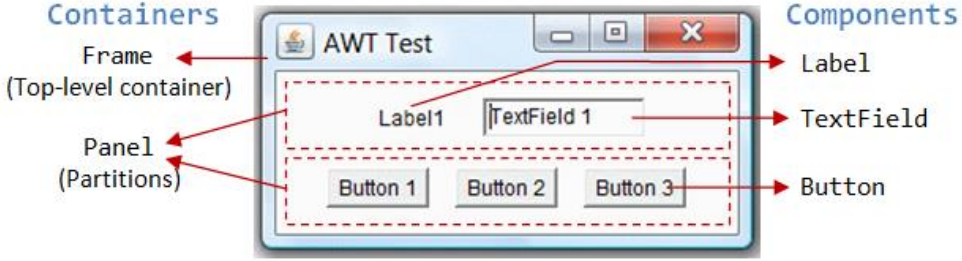


# ORIGEN DE LAS LIBRERÍAS GRÁFICAS

Aunque no será objeto de este tema, más adelante, en **2003**, surge la librería **SWT** (Standard Widget Toolkit), un conjunto de componentes para construir interfaces gráficas en Java o widgets, y que es  desarrollada por el proyecto Eclipse. Recupera la idea original de la biblioteca AWT de utilizar componentes nativos, con lo que adopta un estilo más consistente en todas las plataformas, pero evita caer en las limitaciones de esta.



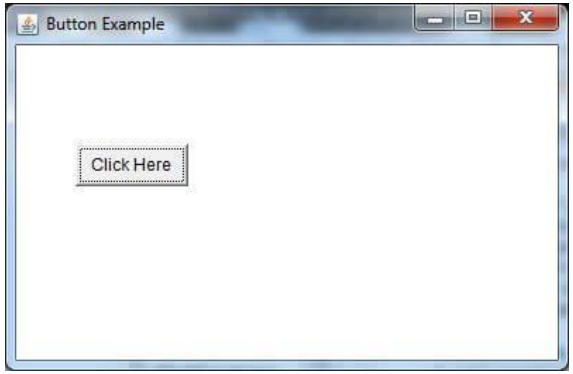
# LIBRERÍA AWT




Como ya se ha dicho, una de las razones del éxito de Java es la posibilidad desde las primeras versiones, de una gran cantidad de paquetes de clases destinadas al diseño de interfaces gráficas mientras que en otros lenguajes (C, C++, etc.), las librerías de componentes gráficos dependían del fabricante del compilador particular que se decidiera utilizar.

Un buen programador de Java debe conocer la variedad de **clases gráficas** y sus inmensas posibilidades de diseño además de los aspectos básicos del lenguaje y su filosofía. Por tanto, es necesario un conocimiento detallado de las clases del API (application program interface) de Java y sus posibilidades. El primer grupo de clases gráficas de Java se encuentra en la librería **AWT (Abstract Window Toolkit)**.

# LIBRERÍA AWT




A partir de la librería AWT se han generado otros conjuntos de clases como Swing pero es importante conocer y entender primero esta librería ya que Swing contiene demasiado de AWT como para ignorarlo y así, si se comprenden los conceptos pasar posteriormente a Swing no supondrá ningún problema.

El concepto básico de la programación gráfica es el **componente**. Un componente es cualquier cosa que tenga un tamaño, una posición, pueda pintarse en pantalla y pueda recibir eventos (es decir, pueda recibir acciones por parte del usuario). Un ejemplo típico es un botón; el evento más normal que puede recibir es que el usuario de la aplicación pulse el botón.

# LIBRERÍA AWT


Los **conceptos básicos** de la programación gráfica con AWT se resumen en:

- Todos los **componentes** son o bien **contenedores** (como Frame) o bien componentes **básicos** (como Button).
- Los componentes básicos siempre deben encontrarse en un contenedor para ser visibles.
- Los contenedores a su vez se pueden insertar en otros contenedores.
- La **posición** de un componente en un contenedor depende del tamaño del contenedor y de su estilo (en inglés, Layout).
- Todo componente incluye un atributo que se encarga de **dibujarlo** (de tipo **Graphics**).
- Si queremos que un componente haga algo cuando le ocurra un **evento** determinado debemos asociar un objeto (tipo **Listener**) para dicho evento.

# LIBRERÍA AWT

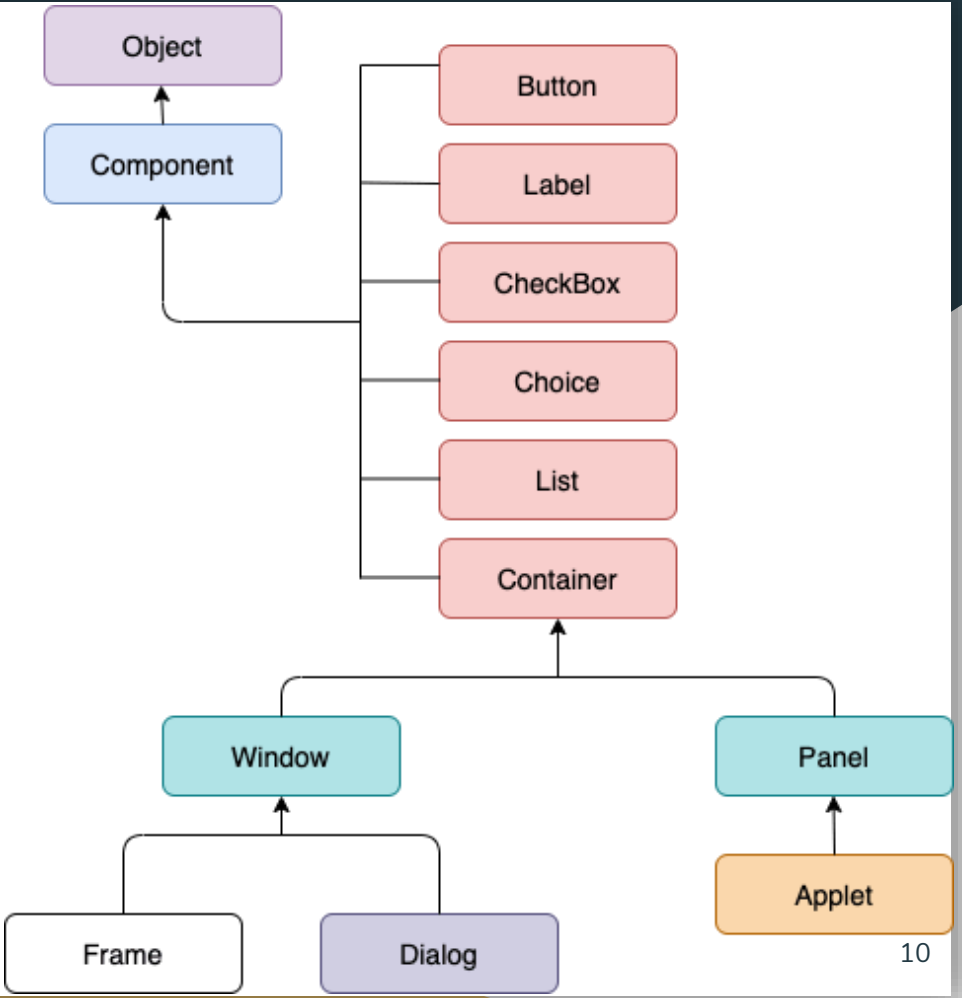
En general al crear una aplicación gráfica con AWT debemos realizar una delegación de eventos:

1. La función main se encargará de crear la ventana principal.
2. En dicha ventana (contenedor) se dispondrán los componentes.
3. Para cada componente se indicará a qué clase debe avisar cuando le suceda un evento asociar Listener).
4. Se programará cada clase para los eventos y a partir de este momento será el propio lenguaje el que se encargará de avisar a dichas clases cuando ocurra el evento provocado por las acciones del usuario.

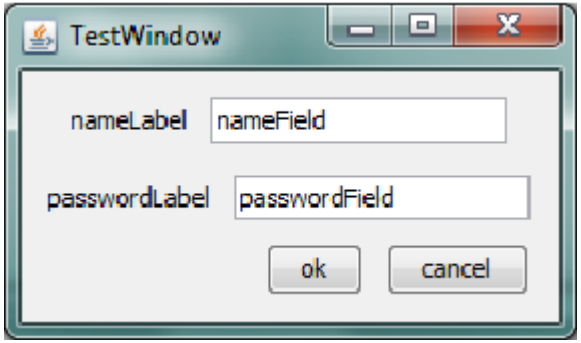
Con este modelo, se pierde el concepto de ejecución secuencial que hemos visto en los programas con entrada/salida de texto.

# LIBRERÍA AWT.

En AWT la clase principal es COMPONENT



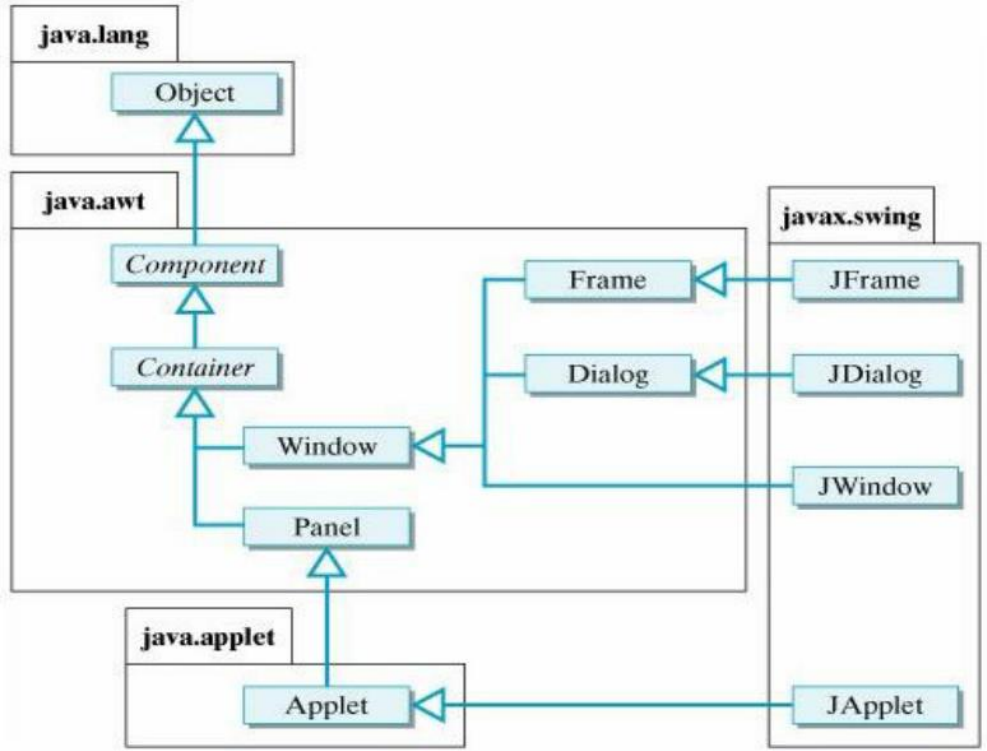
# LIBRERÍA SWING




Como ya se ha mencionado, el conjunto de clases Swing nació a partir de AWT, simplificando los aspectos más engorrosos de AWT, dando mayor flexibilidad al programador para diseñar sus propios componentes gráficos, con el uso de beans, e incorporando numerosos componentes nuevos.

Por cada componente en AWT, el correspondiente componente en Swing suele tener el mismo nombre pero precedido de una letra J. Por ejemplo, los botones se representan en AWT mediante objetos de la clase Button y en Swing mediante objetos de la clase JButton, las ventanas mediante las clases Frame y JFrame, y así sucesivamente.

# LIBRERÍA SWING



# 1. El origen de Swing

Nace para proporcionar un entorno visual sencillo para permitir la comunicación e interacción del usuario con el programa.

Por ejemplo en un navegador de Internet, un editor de textos, una aplicación de móvil, etc. Todos tienen ventanas, testo, imágenes, botones, menús, barras, etc. que nos permiten ver la información representada por el programa e interactuar con el para realizar acciones como visitar una página web, escribir un texto con formato, guardar o abrir archivos del disco duro, escuchar música, etc.

En Java la programación de aplicaciones gráfica se realiza utilizando Swing, una biblioteca gráfica que incluye widgets para interfaz gráfica de usuario tales como cajas de texto, botones, listas desplegables y tablas. Swing está basado en la Abstract Window Toolkit (AWT, en español Kit de Herramientas de Ventana Abstracta) es un kit de herramientas de gráficos, interfaz de usuario, y sistema de ventanas independiente de la plataforma original de Java.

# 2. Componentes y contenedores

Una GUI Swing consta de dos elementos clave: **componentes** y **contenedores**. Sin embargo, esta distinción es principalmente conceptual porque todos los contenedores también son componentes. a diferencia entre los dos se encuentra en su propósito: un componente es un control visual independiente, como un botón o campo de texto. Un contenedor contiene un grupo de componentes.

Por lo tanto, un contenedor es un tipo especial de componente que está diseñado para contener otros componentes. Además, para que se muestre un componente, debe mantenerse dentro de un contenedor. Por lo tanto, **todas las GUI de Swing tendrán al menos un contenedor**. Como los contenedores son componentes, un contenedor también puede contener otros contenedores. Esto permite a Swing definir lo que se llama una **jerarquía de contención**, en la parte superior debe ser un contenedor de nivel superior.

# 2.1. Componentes

- En general, los componentes Swing se derivan de la clase **JComponent**. (Las únicas excepciones a esto son los cuatro contenedores de nivel superior, que se describen en la siguiente sección). JComponent proporciona la funcionalidad que es común a todos los componentes. Por ejemplo, JComponent admite la **look and feel** conectables. JComponent hereda las clases AWT **Container** y **Component**. Por lo tanto, un componente Swing está integrado y es compatible con un componente AWT.

- Todos los componentes de Swing están representados por clases definidas dentro del paquete **javax.swing**. La siguiente tabla muestra los nombres de clase para los componentes Swing (incluidos los utilizados como contenedores):

- Observe que todas las clases de componentes comienzan con la letra **J**. Por ejemplo, la clase para una etiqueta es **JLabel**, la clase para un botón es **JButton** y la clase para para una casilla de verificación es JCheckBox

| | | | | | |
|---|---|---|---|---|---|
| JApplet (obsoleto JDK 9) | JButton | JCheckBox | JCheckBoxMenuItem | JColorChooser | JComboBox |
| JComponent | JDesktopPane | JDialog | JEditorPane | JFileChooser | JFormattedTextField |
| JFrame | JInternalFrame | JLabel | JLayer | JLayeredPane | JList |
| JMenu | JMenuBar | JMenuItem | JOptionPane | JPanel | JPasswordField |
| JPopupMenu | JProgressBar | JRadioButton | JRadioButtonMenuItem | JRootPane | JScrollBar |
| JScrollPane | JSeparator | JSlider | JSpineer | JSplitPane | JTabbedPane |
| JTable | JTextArea | JTextField | JTextPane | JTogglebutton | JToolBar |
| JToolTip | JTree | JViewport | JWindows | | |

# 2.2. Contenedores

Swing define dos tipos de contenedores. Los primeros son contenedores de nivel superior: **JFrame**, **JApplet**, **JWindow** y **JDialog**. (JApplet, que admite applets basados en Swing, ha sido descartado por JDK 9.) Estos contenedores no heredan **JComponent**. Sin embargo, heredan las clases AWT **Component** y **Container**. A diferencia de otros componentes de Swing, que son *lightweight*, los contenedores de nivel superior son *heavyweight*. Esto hace que los contenedores de nivel superior sean un caso especial en la biblioteca de componentes Swing.

Como su nombre lo indica, un contenedor de nivel superior debe estar en la parte superior de una jerarquía de contención. Un contenedor de nivel superior no está contenido en ningún otro contenedor. Además, cada jerarquía de contención debe comenzar con un contenedor de nivel superior. El más comúnmente utilizado para las aplicaciones es **JFrame**.

El segundo tipo de contenedor compatible con Swing es el contenedor lightweight. Los contenedores *lightweight* heredan **JComponent**. Ejemplos de contenedores lightweight son **JPanel**, **JScrollPane** y **JRootPane**. Los contenedores lightweight a menudo se usan para organizar y administrar colectivamente grupos de componentes relacionados porque un contenedor lightweight se puede contener dentro de otro contenedor. Por lo tanto, puede usar contenedores lightweight para crear subgrupos de controles relacionados que están contenidos dentro de un contenedor externo.

Tabla de Administrador de esquemas Swing Java.

| Layout Managers | Definición |
|---|---|
| FLowLayout | Un diseño simple que posiciona los componentes de izquierda a derecha, de arriba hacia abajo. |
| BorderLayout | Posiciones de componentes dentro del centro o los bordes del contenedor. Este es el diseño predeterminado para un content pone.. |
| GridLayout | Presenta diferentes componentes de tamaño dentro de un grid flexible. |
| GridBagLayout | Presenta los componentes dentro de un grid. |
| BoxLayout | Expone componentes vertical u horizontalmente dentro de un box. |
| SpingLayout | Presenta componentes sujetos a una serie de restricciones. |

# HOLA MUNDO CON SWING

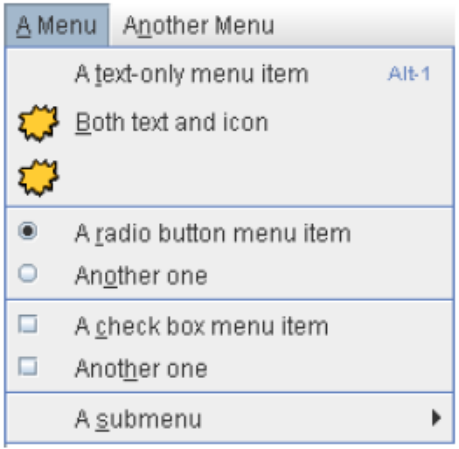
```
import javax.swing.JFrame;                    /* HolaMundo.java. Ejemplo de "Hola Mundo" en Swing. */
import javax.swing.JLabel;
import javax.swing.BorderFactory;
import javax.swing.border.Border;
class HolaMundo {
   public static void main(String[] args) {
      JFrame frame = new JFrame("Hola Mundo en Swing"); //creamos un JFrame con un título
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //definimos que la ejecución termine al cerrar el
JFrame
      JLabel label = new JLabel("Hola Mundo", JLabel.CENTER); //creamos una etiqueta con el texto y borde invisible
      Border empty = BorderFactory.createEmptyBorder(100, 100, 100, 100);
      label.setBorder(empty);
      frame.getContentPane().add(label);  //agregamos la etiqueta en el panel de contenido principal
      frame.pack();  //le damos el tamaño adecuado al JFrame según el tamaño preferido de sus componentes
      frame.setLocationRelativeTo(null);  //centramos el JFrame en la pantalla
      frame.setVisible(true); //hacemos visible el JFrame
   }
}
```

# SWING

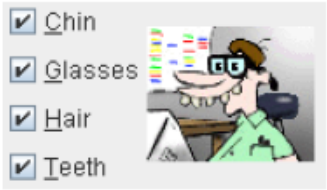
Introducción
2. Crear una aplicación Swing con Netbeans
3. Jerarquía de componentes
4. JFrame
5. Eventos
6. Componentes
7. Cuadros de diálogo JOptionPane
8. Ejemplo de aplicación gráfica

# CONTROLES BÁSICOS DE SWING

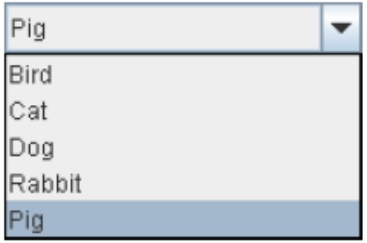

JButton




JMenu

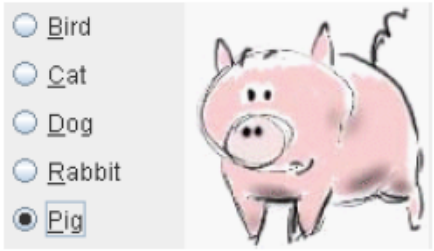



JCheckBox




JComboBox


JList




JRadioButton

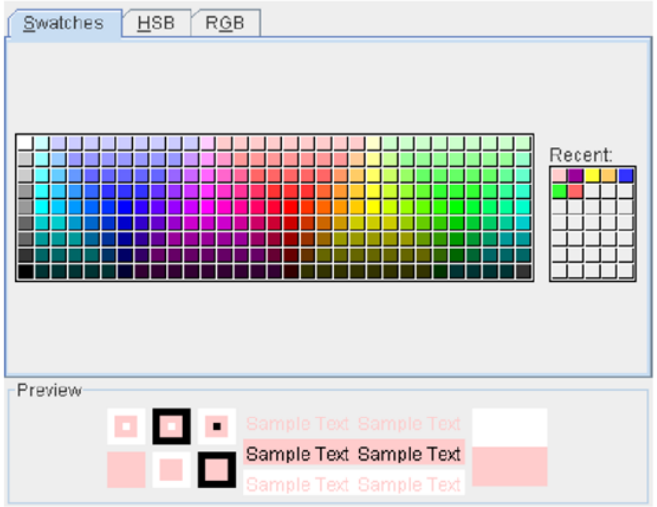

JSlider


JPasswordField

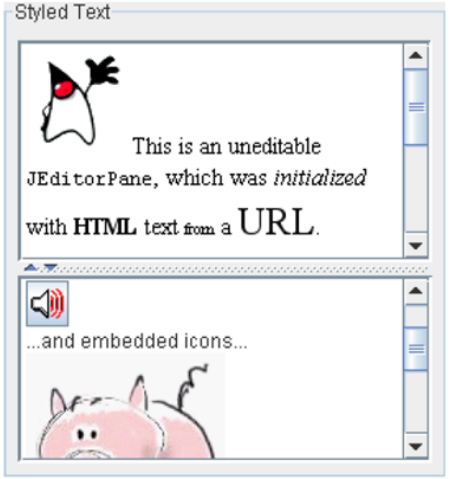

JTextField


JSpinner

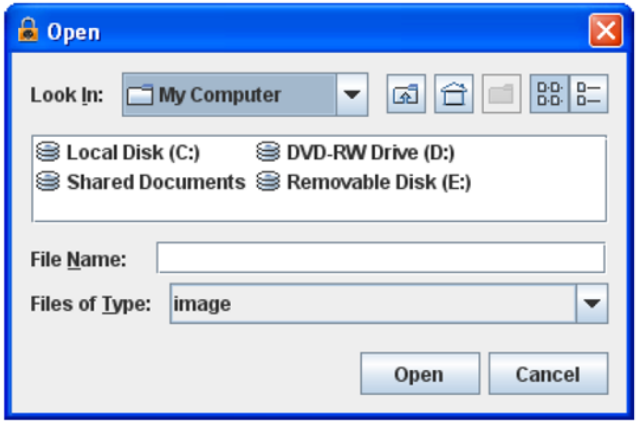
# COMPONENTES INTERACTIVOS




JColorChooser



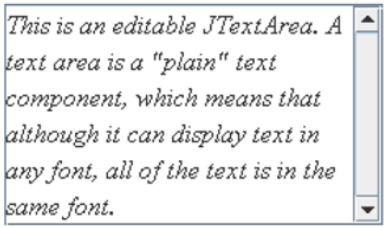

JEditorPane and JTextPane




JFileChooser

| Host | User | Password | Last Modified |
|---|---|---|---|
| Biocca Games | Freddy | !#asf6Awwzb | Mar 16, 2006 |
| zabble | ichabod | Tazb!34$fZ | Mar 6, 2006 |
| Sun Developer | fraz@hotmail.co... | AasW541!fbZ | Feb 22, 2006 |
| Heirloom Seeds | shams@gmail.... | bkz[ADF78! | Jul 29, 2005 |
| Pacific Zoo Shop | seal@hotmail.c... | vbAf124%z | Feb 22, 2006 |

JTable




JTextArea


JTree

# INFORMACIÓN NO EDITABLE

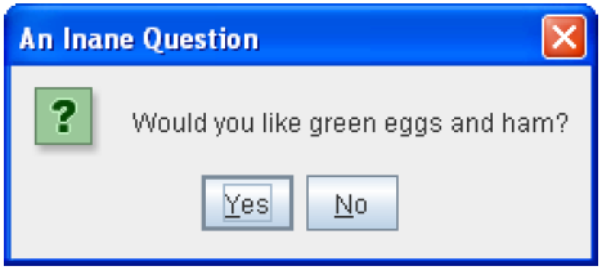

JLabel


JToolTip

JSeparator

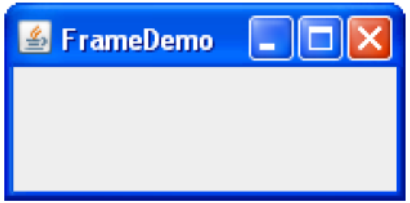

JProgressBar

# CONTENEDORES DE ALTO NIVEL




JDialog

JApplet




JFrame

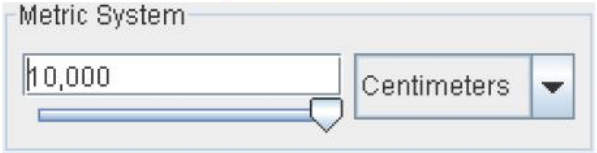
# CONTENEDORES DE PROPÓSITO GENERAL


JTabbedPane

JToolBar

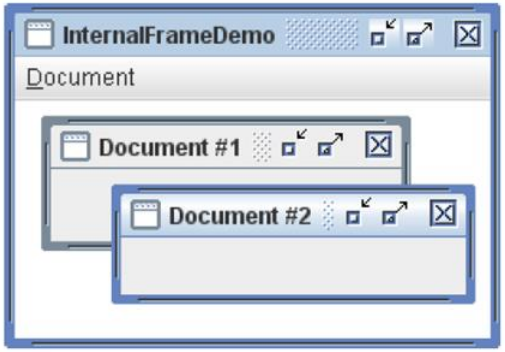
JScrollPane


JSplitPane

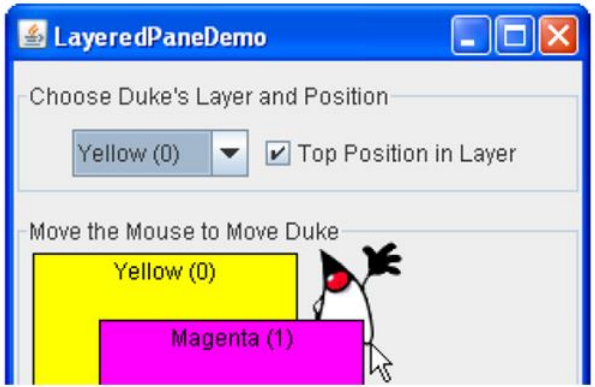



JPanel

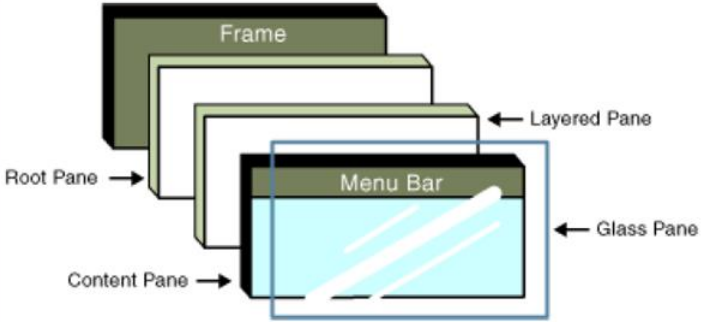
# CONTENEDORES DE PROPÓSITO ESPECÍFICO




JInternalFrame




JLayeredPane




Root pane

# Bibliografía y webgrafía


- http://gpd.sip.ucm.es/rafa/docencia/programacion/tema5/tema5.html
- http://codejavu.blogspot.com/2013/08/que-es-java-swing.html
- https://guru99.es/java-swing-gui/
- http://gpd.sip.ucm.es/rafa/docencia/programacion/tema5/tema5.html
- http://codejavu.blogspot.com/2013/08/que-es-java-swing.html