Interfaz Gráfica de Usuario con Swing

Índice de contenido

La Interfaz Gráficaz de Usuario

Interfaz gráfica de usuario (En inglés Graphic User Interface, también conocido con su acrónimo GUI) es un método para facilitar la interacción del usuario con la computadora a través de la utilización de un conjunto de imágenes y objetos pictóricos (ventanas, botones, menús, iconos, etc.) además de texto.

Surge como evolución de la línea de comandos de los primeros sistemas operativos y es pieza fundamental en un entorno gráfico.

GUI es un acrónimo del vocablo inglés Graphical User Interface.

Douglas Engelbart, además de inventor del ratón de computadora, desarrolló la primera interfaz gráfica en los años 1960 en EE.UU. en los laboratorios de XEROX. Fue introducida posteriormente al público en las computadoras Apple Macintosh en 1984, y a las masas hasta 1993 con la primera versión popular del sistema operativo Windows 3.0.

Una GUI, presenta una interfaz de imágenes con un programa. La GUI confiere al programa un "aspecto" y una "sensación" distintivos. Gracias a la GUI, el usuario no tiene que dedicar tanto tiempo a tratar de recordar cuál es la secuencia de digitaciones y qué es lo que hace y puede pasar más tiempo usando el programa de forma productiva.

Java nos ofrece 2 paquetes para la creación de interfaces gráficas de usuario: AWT y SWING.

Un vistazo al paquete AWT

AWT (Abstract Window Toolkit) es un paquete de Java diseñado para crear interfaces de usuario y para dibujar gráficos e imágenes. La mayoría de los componentes AWT descienden de la clase `java.awt.Component` como podemos ver en la figura 1.

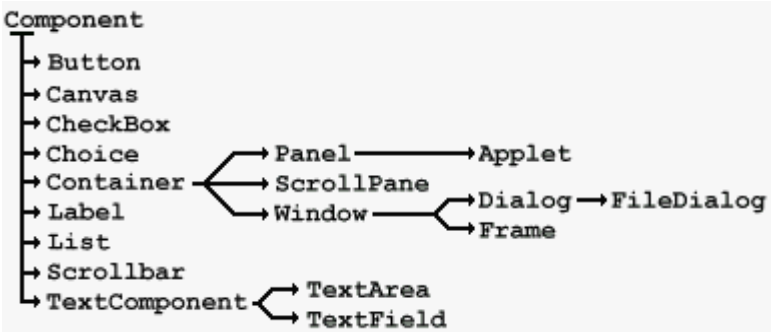


Figura 1: Jerarquía parcial de componentes de AWT

Un vistazo al paquete SWING

Swing es un extenso conjunto de componentes que van desde los más simples, como etiquetas, hasta los más complejos, como tablas, árboles, y documentos de texto con estilo. Casi todos los componentes Swing descienden de un mismo padre llamado `JComponent` que desciende de la clase de AWT `Container`. Es por ello que Swing es más una capa encima de AWT que una sustitución del mismo. La figura 2 muestra una parte de la jerarquía de `JComponent`. Si la comparas con la jerarquía de `Component` notarás que para cada componente AWT hay otro equivalente en Swing que empieza con "J". La única excepción es la clase de AWT `Canvas`, que se puede reemplazar con `JComponent`, `JLabel`, o `JPanel`. Asimismo te percatarás de que existen algunas clases Swing sin su correspondiente homólogo. La figura 2 representa sólo una pequeña fracción del paquete Swing, pero esta fracción son las clases con las que te enfrentarás más a menudo. El resto de Swing existe para suministrar un amplio soporte y la posibilidad de personalización a los componentes estas clases definen.

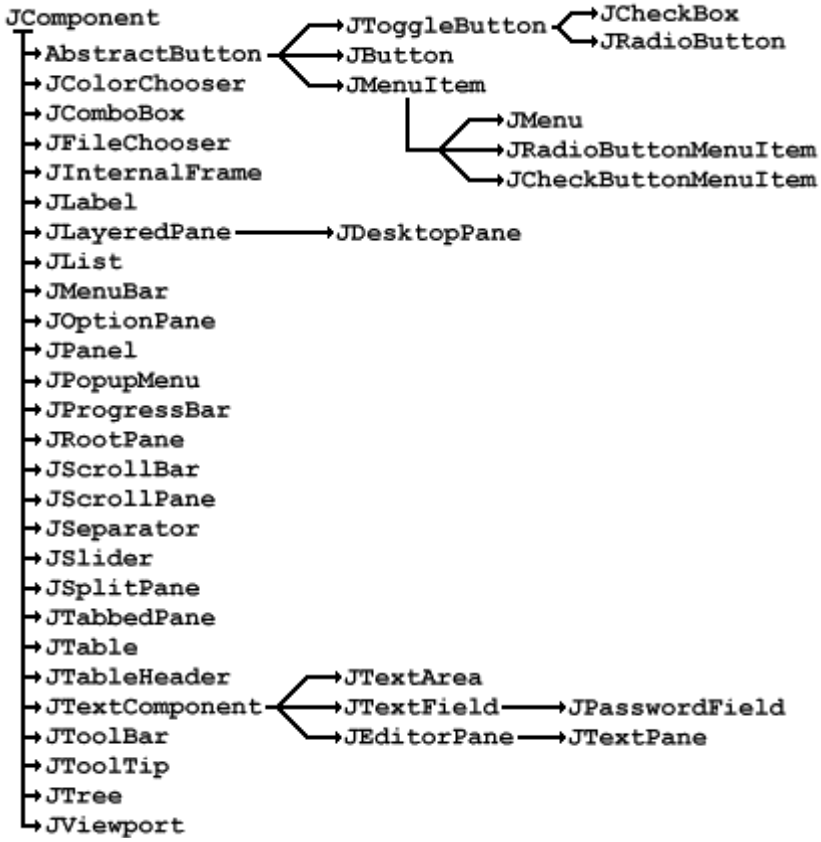


Figura 2: Parte de la jerarquía de componentes de SWING

Componentes y Contenedores

Las GUI se construyen a partir de *componentes* de GUI. Un componente de GUI es un objeto visual con el que el usuario puede interactuar a través dispositivos de entrada como ser el ratón, teclado, etc. Estos *componentes* necesitan de *contenedores* para ser mostrados. Estos *contenedores* son: Ventana con Marco (`JFrame`), Ventana sin Marco (`JWindow`) y Ventana de Diálogo (`JDialog`).

Figura 3: JFrame, JWindow y JDialog

La clase `JComponent`

Como todo buen javero sabe, `Object` es la superclase de todos las clases que están escritas y que se escriben. Así mismo, la clase `JComponent` es la superclase de todos los componentes gráficos de swing. Esta clase pertenece al paquete `javax.swing`.

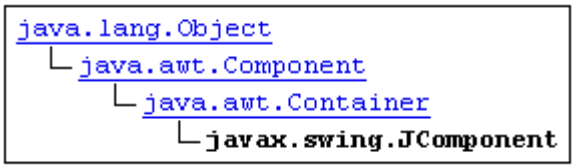


Figura 4: Jerarquía de la clase JComponent

La clase `JComponent` es la base de todos los componentes *Swing*. Es una clase derivada de la clase `java.awt.Container`.

La clase `JComponent` añade bastantes cosas a la clase `Container`, como la posibilidad de dibujar bordes predefinidos alrededores de componentes, manejo de imágenes, añadir objetos `PropertyChangeListener`, que son notificados cuando se cambia el valor de la propiedad, y mucho más.

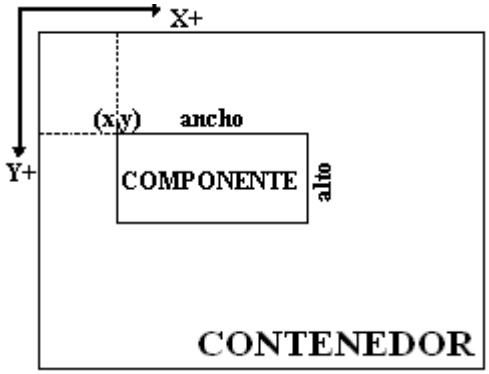


Figura 5: Representación de un componente sobre un contenedor

Para mayor referencia presentaremos a continuación los constructores y métodos más importantes de esta clase (cuidado, también se listarán algunos métodos heredados de la clase `Container` y `Component`).

Constructores

Constructor	Descripción
`JComponent()`	`Constructor por defecto de JComponent.`

Métodos

Tipo de retorno	Nombre del Método	Descripción
`int`	`getX()`	`Retorna la coordenada en X del componente, respecto de su contenedor padre.`
`int`	`getY()`	`Retorna la coordenada en Y del componente, respecto de su contenedor padre.`
`int`	`getWidth()`	`Retorna el ancho del componente.`
`int`	`getHeight()`	`Retorna el alto del componente.`
`void`	`setLocation(int,int)`	`Establece la ubicación del componente a la posición (x,y)`

Tipo de retorno	*Nombre del Método*	*Descripción*
void	setSize(int,int)	Establece la dimensión del componente con el ancho y alto especificados.
void	setBounds(int,int,int,int)	Establece la ubicación y la dimensión del componente (x, y, ancho y alto).
String	getToolTipText()	Retorna el mensaje descriptivo del componente.
void	setToolTipText(String)	Estable el mensaje descriptivo del componente.
void	grabFocus()	Establece el foco al componente.
boolean	isOpaque()	Retorna true si el componente es completamente opaco.
void	setOpaque(boolean)	Establece si el componente debería ser opaco o no.
Color	getBackground()	Retorna el color de fondo del componente.
void	setBackground(Color)	Establece el color de fondo del componente.
Color	getForeground()	Retorna el color de primer plano componente.
void	setForeground(Color)	Establece el color de primer plano del componente.
void	setBorder(Border)	Establece el borde del componente.
Border	getBorder()	Retorna el borde del componente.
void	setEnabled(boolean)	Establece si el componente debería estar habilitado o no.
boolean	isEnabled()	Retorna el resultado de comprobar si está habilitado o no lo está.
void	setFont(Font)	Establece la fuente del componente.
Font	getFont()	Retorna la fuente del componente.

Tipo de retorno	*Nombre del Método*	*Descripción*
`void`	`setVisible(boolean)`	`Establece si el componente debería mostrarse o no.`
`boolean`	`isVisible()`	`Retorna el resultado de comprobar si el componente se muestra.`
`void`	`updateUI()`	`Restablece la propiedad de UI del componente.`

Y como `JComponent` es la superclase de casi todos los componentes en Swing (Figura 2), pues todos ellos heredan estos métodos.

La forma de agregar *componentes* a un *contenedor* es utilizando el método `add()`.

Todo contenedor está asociado a un objeto llamado *Administrador de diseño*, éste se encarga básicamente de darle ubicación *(x,y)*, y también dimensión *(ancho y alto)* al componente respecto al contenedor. Iremos explicando detalladamente cada uno de estos tópicos.

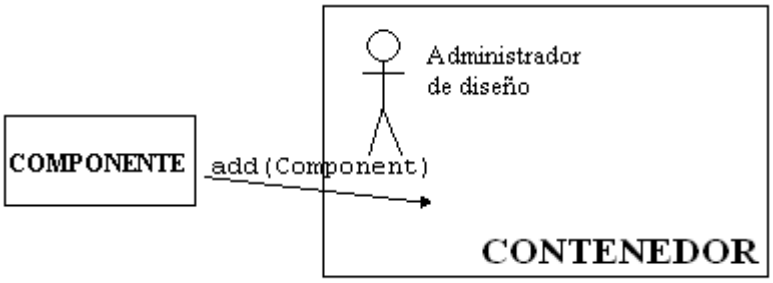


Figura 6: Representación componente añadiendose a un contenedor

Veamos un pequeño ejemplo y vamos a crear la siguiente interfaz gráfica de usuario:

Figura 7: Primer Ejemplo

```
import javax.swing.*;
import java.awt.*;

public class Ejemplo1
{
   public static void main(String[] args)
   {
      JFrame ventana = new JFrame("Swing es poder!!!");
      JLabel etiqueta = new JLabel("Ingresa un número:");
      JTextField campo = new JTextField(5);
      JButton boton = new JButton("Hazme click");

      ventana.setLayout(new FlowLayout());
      ventana.add(etiqueta);
      ventana.add(campo);
      ventana.add(boton);

      ventana.setSize(200,100);
      ventana.setVisible(true);
   }
}
```

Código 1: Ejemplo1.java

Cómo funciona

El código anterior crea una ventana con título **"Swing es Poder!!!"** y con tres componentes dentro de el: una etiqueta de texto con el texto **"Ingresa un número"**, un campo de texto de 4 columnas de ancho y un botón que exhibe el texto **"Hazme click"**.

Contenedores Básicos

La clase `JFrame`

La clase `JFrame` implementa un objeto ventana con marco. Esta clase es derivada de la clase `java.awt.Frame`.

Éste contenedor los emplearemos para situar en él todos los demás componentes que necesitemos para el desarrollo de la interfaz de nuestro programa. Para añadir un componente al contenedor utilizamos directamente el método `add()` en Java 1.5 y en versiones anteriores la técnica pasa por utilizar `getContentPane().add()`.

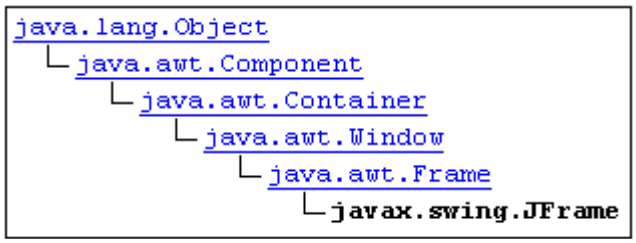


Figura 8: Jerarquía de la clase JFrame

A continuación presentaremos los constructores y métodos más importantes de esta clase (ojo, también se listarán algunos métodos que están repartidos a lo largo de todos sus ascendientes).

Constructores

Constructor	*Descripción*
`JFrame()`	`Constructor por defecto de JFrame. Construye una ventana que inicialmente es invisible.`
`JFrame(String)`	`Constructor que crea una ventana con el título especificado e inicialmente invisible.`

Métodos

Tipo de retorno	*Nombre del Método*	*Descripción*
`Container`	`getContentPane()`	`Retorna el objeto contentPane de la`

Tipo de retorno	*Nombre del Método*	*Descripción*
		ventana.
void	setContentPane(Container)	Establece la dimensión del componente con el ancho y alto especificados.
Component	getGlassPane()	Retorna el objeto glassPane de la ventana.
void	setGlassPane(Component)	Establece la propiedad glassPane.
JMenuBar	getJMenuBar()	Retorna el objeto barra de menú que esta sobre la ventana.
void	setJMenuBar(JMenuBar)	Establece la barra de menú de esta ventana.
JLayeredPane	getLayeredPane()	Retorna el objeto layeredpane de esta ventana.
void	setLayeredPane(JLayeredPane)	Establece la propiedad layeredpane.
void	remove(Component)	Remueve el componente especificado del contenedor.
void	removeAll()	Remueve todos los componentes de la ventana.
Component	add(Component)	Añade el componente especificado y retorna su referencia.
void	setIconImage(Image)	Establece la imagen que ha de ser el icono del para esta ventana.
void	setLayout(LayoutManager)	Establece el administrador de diseño especificado.
String	getTitle()	Retorna el título de la ventana.
void	setTitle(String)	Establece el título de la ventana.

Tipo de retorno	*Nombre del Método*	*Descripción*
boolean	isResizable()	Verifica si la ventana puede ser redimensionada por el usuario.
void	setResizable(boolean)	Establece si la ventana debería ser redimensionada por el usuario.
void	setCursor(Cursor)	Establece el cursor de la ventana.
void	setUndecorated(boolean)	Habilita o deshabilita decoraciones en la ventana.
void	setEnabled(boolean)	Establece si el componente debería estar habilitado o no.
boolean	isEnabled()	Retorna el resultado de comprobar si está habilitado o no lo está.
void	dispose()	Libera los recursos nativos de la pantalla.
Font	getFont()	Retorna la fuente del componente.
void	setFont(Font)	Establece la fuente del componente.
boolean	isVisible()	Retorna el resultado de comprobar si el componente se muestra.
void	setVisible(boolean)	Establece si el componente debería mostrarse o no.
void	updateUI()	Restablece la propiedad de UI del componente.
boolean	isAlwaysOnTop()	Verifica si es que la ventana siempre está visible.
void	setAlwaysOnTop(boolean)	Establece si la ventana siempre está visible.
void	setCursor(Cursor)	Establece el cursor de la ventana.

Tipo de retorno	Nombre del Método	Descripción
void	toBack()	Envia la ventana atrás.
void	toFront()	Envia la ventana adelante.
int	getComponentCount()	Retorna la cantidad de componentes de la ventana.
int	getX()	Retorna la coordenada en X del componente, respecto de su contenedor padre.
int	getY()	Retorna la coordenada en Y del componente, respecto de su contenedor padre.
int	getWidth()	Retorna el ancho del componente.
int	getHeight()	Retorna el alto del componente.
void	setLocation(int,int)	Establece la ubicación del componente a la posición (x,y)
void	setSize(int,int)	Establece la dimensión del componente con el ancho y alto especificados.
void	setSize(Dimension)	Establece la dimensión del componente con el objeto Dimension dado.
void	setBounds(int,int,int,int)	Establece la ubicación y la dimensión del componente (x, y, ancho y alto).
Color	getBackground()	Retorna el color de fondo del componente.
void	setBackground(Color)	Establece el color de fondo del componente.
Color	getForeground()	Retorna el color de primer plano componente.
void	setForeground(Color)	Establece el color de

Tipo de retorno	Nombre del Método	Descripción
		primer plano del componente.
void	setFont(Font)	Establece la fuente del componente.
Font	getFont()	Retorna la fuente del componente.
void	setVisible(boolean)	Establece si el componente debería mostrarse o no.
boolean	isVisible()	Retorna el resultado de comprobar si el componente se muestra.

Un objeto `JFrame` puede servir como cualquier otro contenedor principal: ventana (`JWindow`), cuadro de diálogo (`JDialog`) o cualquier otro.

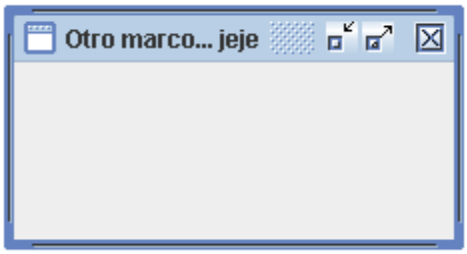
Veamos cómo podemos hacer que un `JFrame` se muestre como un `JWindow`.

```
import javax.swing.JFrame;

public class VentanaSinMarco
{
   public static void main(String[] args)
   {
      JFrame ventana = new JFrame();
      ventana.setUndecorated(true);
      ventana.setSize(300,300);
      ventana.setVisible(true);
   }
}
```

Código 2: VentanaSinMarco.java

Ahora para hacer que nuestra ventana utilice otro marco:



Aquí está el código:

```
import javax.swing.JFrame;
import javax.swing.JRootPane;

public class VentanaConOtroMarco
{
   public static void main(String[] args)
   {
      JFrame ventana = new JFrame("Otro marco... jeje");

      ventana.setUndecorated(true);
      ventana.getRootPane().setWindowDecorationStyle(JRootPane.FRAME);

      ventana.setSize(230,120);
      ventana.setVisible(true);
   }
}
```

Código 3: VentanaConOtroMarco.java

Y para darle otros estilos de decoración de ventana, simplemente utilizamos las constantes proporcionadas por la clase `JRootPane`:

- NONE
- FRAME
- PLAIN_DIALOG
- INFORMATION_DIALOG
- ERROR_DIALOG
- COLOR_CHOOSER_DIALOG

- FILE_CHOOSER_DIALOG
- QUESTION_DIALOG
- WARNING_DIALOG

La forma habitual en la que será usado un `JFrame` es mediante la herencia, así de la siguiente forma:

```
[...]
class MiVentana extends JFrame
{
   [...]
}
```

Componentes Básicos

Si queremos usar con eficacia los componentes de GUI, debemos entender y comprender a la perfección la jerarquía de la clase JComponent. Gran parte de la funcionalidad de cada componente se deriva de ésta clase o sus ascendientes. Toda clase que hereda de la clase JComponent es un componente.

Componente	***Descripción***
Etiqueta	Componente en el que puede exhibirse texto no editable.
Botón	Componente que dispara un evento cuando se hace click sobre él.
Campo de texto	Componente en el que se puede introducir datos mediante el teclado.

Tabla 1: Componentes Básicos de una interfaz de usuario

Figura 9: Componentes básicos

La clase `JLabel`

Una etiqueta puede exhibir texto, imagen o una combinación de ambos. Las etiquetas se crean con la clase `JLabel` que hereda directamente de la clase `JComponent`.

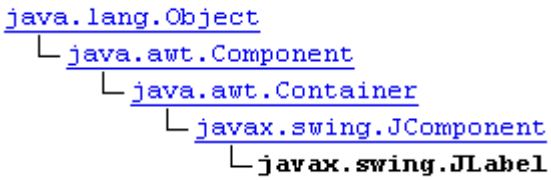


Figura 10: Jerarquía de la clase JLabel

Vayamos a ver con más detalle esta clase (no con tanto detalle, sólo lo más importante, ya sabes que para más detalle están los JAVADOCS):

Constructores

Constructor	*Descripción*
`JLabel()`	Construye una etiqueta sin texto y sin imagen por defecto.
`JLabel(Icon)`	Construye una etiqueta con una imagen especificada.
`JLabel(String)`	Crea una etiqueta con el texto especificado.
`Jlabel(String,Icon,int)`	Construye una etiqueta con el texto, imagen y alineación especificada[1].

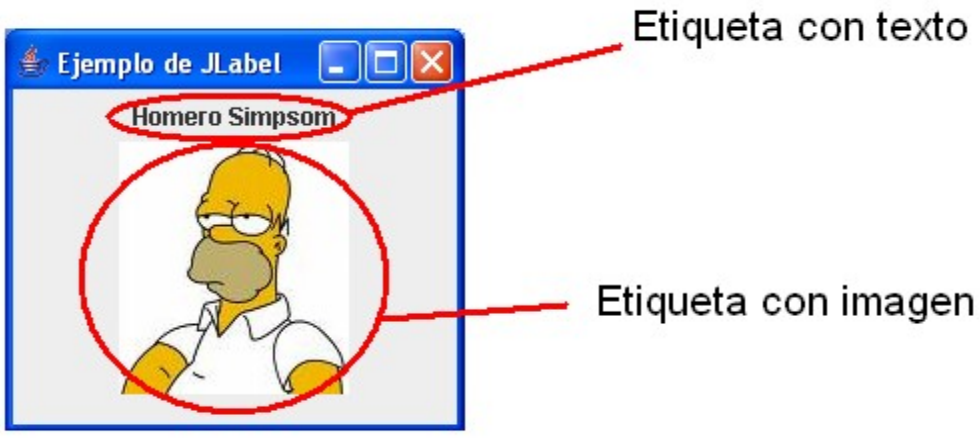
1 Esta alineación es una constante de la clase SwingConstants, clase que será vista más adelante.

Métodos

Tipo de retorno	*Nombre del Método*	*Descripción*
`String`	`getText()`	`Retorna el texto que se exhibe sobre la etiqueta.`
`void`	`setText(String)`	`Establece el texto que ha de exhibirse sobre la etiqueta.`
`Icon`	`getIcon()`	`Retorna la imagen que se exhibe sobre la etiqueta.`
`void`	`setIcon(Icon)`	`Establece la imagen que se ha de ver sobre la etiqueta.`
`int`	`getDisplayedMnemonic()`	`Devuelve el código ASCII del caracter memotécnico.`
`void`	`setDisplayedMnemonic(char)`	`Indica que caracter del texto ha de ser memotécnico.`

Pueden usarse más métodos para esta clase (recuerda que JLabel hereda de JComponent).

Ahora un pequeño ejemplo para ver algo de esta clase. Vamos a crear el siguiente GUI:



Es una ventana que tiene dos componentes: una etiqueta con texto y una etiqueta con imagen. Para ambos componentes utilizamos `JLabel`, y aquí está el código:

```
import java.awt.FlowLayout;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;


public class EjemploJLabel extends Jframe
{
   private JLabel texto;
   private JLabel imagen;

   public EjemploJLabel()
   {
      super("Ejemplo de JLabel");

      texto = new JLabel("Homero Simpsom");
      imagen = new JLabel(new ImageIcon( getClass().
                                  getResource("imagenes/homero.jpg")));
      setLayout(new FlowLayout());
      add(texto);
      add(imagen);
   }

   public static void main(String[] args) {
      EjemploJLabel x = new EjemploJLabel();
      x.setSize(230, 200);
      x.setVisible(true);
   }
}
```

Para el anterior ejemplo, la imagen homero.jpg debería estar en la carpeta **imagenes** y está ser una subcarpeta donde se encuentra nuestro código fuente y binarios:

```
-Mis Ejemplos Swing/
  |-imagenes/
     |-homero.jpg
  |-EjemploJLabel.java
  |-EjemploJLabel.class
```

La clase `JButton`

Una botón al igual que una etiqueta, puede exhibir texto, imagen o una combinación de ambos, pero este al ser presionado lanza un evento. Los botones pueden ser utilizados creando instancias de la clase `JButton`.

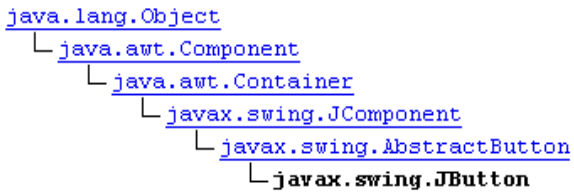


Figura 11: Jerarquía de la clase JButton

Existen otro tipo de botones que veremos más adelante.

Constructores

Constructor	*Descripción*
`JButton()`	Crea un botón sin texto o imagen.
`JButton(Icon)`	Crea un botón con una imagen especificada.
`JButton(String)`	Crea un botón con una texto especificado.
`JButton(String,Icon)`	Crea un botón con texto e imagen especificados.

Métodos

Tipo de retorno	*Nombre del Método*	*Descripción*
`String`	`getText()`	Retorna el texto que se exhibe sobre el botón.
`void`	`setText(String)`	Establece el texto que ha de exhibirse sobre el botón.
`Icon`	`getIcon()`	Retorna la imagen que se exhibe sobre el botón.
`void`	`setIcon(Icon)`	Establece la imagen que se ha de ver sobre el botón

FIN DE LA VERSIÓN DRAFT 1.0