



# TÉCNICAS DE PROGRAMACIÓN ORIENTADA A OBJETOS

07/05/2024 y 09/05/2024

UPN.EDU.PE

# **UNIDAD 2: RELACIONES DE CLASES DE HERENCIA SIMPLE Y MÚLTIPLE, COMPONENTES SWING Y ACCESO A DATOS**



## Sesión 07

- Componentes AWT



# OBJETIVOS

- Escribir programas para dibujar elementos gráficos
- Escribir programas con interfaces de usuario elaboradas
- Conocer la tecnología Applet
- Conocer las API para mostrar imágenes y reproducir sonidos



# **LOGRO DE LA SESIÓN:**

Al término de la sesión de aprendizaje, el estudiante entiende los conceptos de Arquitectura N Capas y componentes SWING.



# **UNIDAD 2: RELACIONES DE CLASES DE HERENCIA SIMPLE Y MÚLTIPLE, COMPONENTES SWING Y ACCESO A DATOS**



## Sesión 14

- Paquete SWING
- Componentes
- Estructuras
- Ejemplos



# INTERFACES DE USUARIO (GIU) CON JAVA

- Java proporciona las clases necesarias para el desarrollo de aplicaciones con interfaces de usuario interactivas
- Las clases proporcionan los componentes GUI necesarios para crear aplicaciones y applets Java
- Las clases derivan de la Java Foundation Classes (JFCs) que es una parte importante del Java SDK que es una colección de cinco APIs
  - AWT, Swing, Java2D, Accessibility, Drag and Drop



# AWT Y SWING

- AWT (Abstract Windows Toolkit) y Swing son clases para el desarrollo de GUI's.
- Algunos componentes de AWT usan código nativo y por ello es dependiente de la plataforma.
- Swing está escrito completamente en Java por lo que es independiente de la plataforma:
  - Las aplicaciones distribuidas entre varias plataformas misma apariencia.
  - Se puede considerar como el reemplazo de AWT.



# COMPONENTES AWT

AbstractWindowsToolkit(AWT): java.awt

– GUIelements:

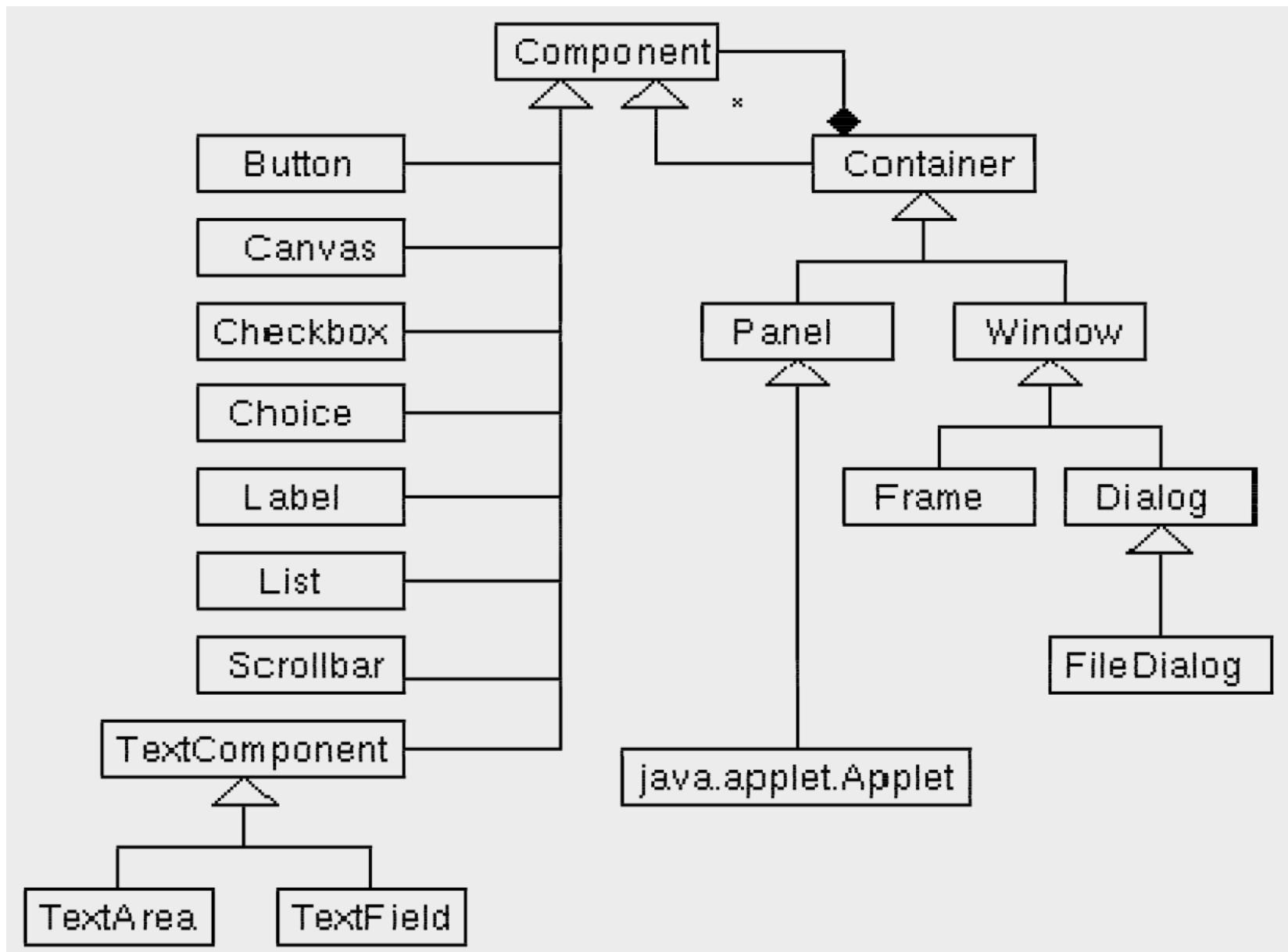
- Container(Panel,Frame,Dialog,etc.)
- Primitive(Button,Label,Checkbox,Scrollbar,etc.)

– Layoutmanagers:FlowLayout,BorderLayout,etc.

– Supportingclasses:

- Event handling
  - java.awt.event package
- Graphics
  - Color, Font, Graphics, etc.
  - Geometry
    - Point, Rectangle, Dimension, etc.
  - Imaging
    - Image class and java.awt.image package

# JERARQUÍA DE COMPONENTES AWT



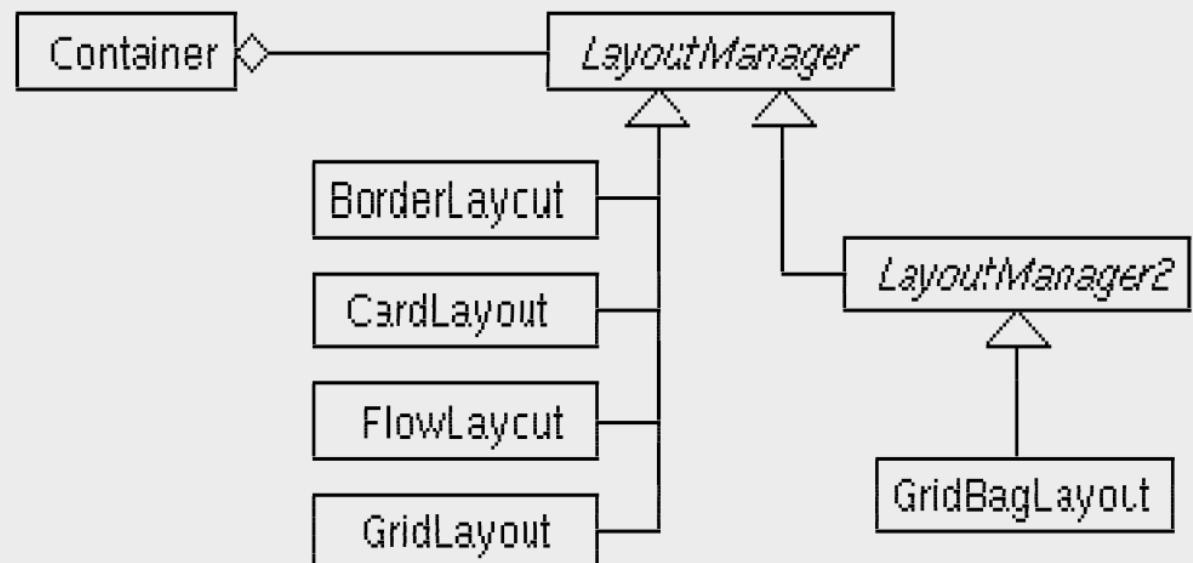
# ↑ LOS COMPONENTES GIU SE COLOCAN EN CONTENEDORES

Clase AWT	Descripción
Component	An abstract class for objects that can be displayed on the console and interact with the user. The root of all other AWT classes
Container	An abstract subclass of the <i>Component</i> class. A component that can contain other AWT components.
Panel	Extends the <i>Container</i> class. A frame or window without the titlebar, the menubar nor the border. Superclass of the Applet class
Window	Extends the <i>Container</i> class. A Window object is a top-level window with no borders and nomenubar. (default BorderLayout)
Frame	Entends the <i>Window</i> class. A window with atitle, menubar, border, and resizing corners.



# GESTORES DEL LAYOUT

- Los gestores de Layout determinan la disposición y tamaño de los componentes dentro de un contenedor
  - Las posiciones y tamaños de los componentes se ajustan automáticamente cuando la ventana cambia de tamaño.
- Los clases de los gestores de layout en Java son:
  - FlowLayout
  - BorderLayout
  - GridLayout
  - GridBagLayout
  - CardLayout





# GESTORES DEL LAYOUT: MÉTODOS

- Ajuste del gestor de layout void setLayout (LayoutManager mgr)
- Si se pasa null, no se usa un gestor de layout
- Si no se usa un gestor es necesario posicionar los elementos manualmente

```
public void setBounds(int x, int y,  
                      int width,int height)
```

Método de la clase Component: Tedioso cuando se tiene varios objetos puesto que se tiene que usar para cada objeto



# GESTOR FLOWLAYOUT:

- Es el gestor por defecto de la clase y subclases Panel – La clase Applet es una subclase de Panel
- Coloca los componentes de izquierda a derecha y de arriba, abajo, empezando en la esquina superior izquierda

## Resumen de Constructores

FlowLayout()	Constructs a new FlowLayout with a centered alignment and a default 5-unit horizontal and vertical gap.
FlowLayout(int align)	Constructs a new FlowLayout with the specified alignment and a default 5-unit horizontal and vertical gap.
FlowLayout(int align, int hgap, int vgap)	Creates a new flow layout manager with the indicated alignment and the indicated horizontal and vertical gaps.



## GESTOR FLOWLAYOUT:

- Gap
  - Espaciado entre componentes
  - Medido en pixels
- Valores de alineamiento posibles FlowLayout.LEFT FlowLayout.CENTER FlowLayout.RIGHT



# EJEMPLO DE GESTOR FLOWLAYOUT:

```
import java.awt.*;
class FlowLayoutDemo extends Frame {
    public static void main(String args[]) {
        FlowLayoutDemo fld = new FlowLayoutDemo();
        fld.setLayout(new FlowLayout(FlowLayout.RIGHT,
                                     10, 10));
        fld.add(new Button("UNO"));
        fld.add(new Button("DOS"));
        fld.add(new Button("TRES"));
        fld.setSize(100, 100);
        fld.setVisible(true);
    }
}
```

Resultado:





# GESTOR BORDERLAYOUT

- Es el gestor por defecto de la clase y subclases Window
  - Incluye los tipos Frame y Dialog
- Divide el objeto Container en cinco partes donde se añaden objetos Component (North, South, East, West, Center).

## Resumen de Constructores

BorderLayout()	Constructs a new border layout with no gaps between components.
BorderLayout(int hgap, int vgap)	Constructs a border layout with the specified gaps between components.

Los parámetros *hgap* y *vgap* se refieren al espaciado entre los componentes dentro del contenedor.



# USO DEL GESTOR BORDERLAYOUT

Para añadir un componente a una región específica:

- Usar el método add y pasar dos argumentos:
  - Componente a añadir
  - Región donde se debe colocar el componente
- Regiones válidas:
  - BorderLayout.NORTH
  - BorderLayout.SOUTH
  - BorderLayout.EAST
  - BorderLayout.WEST
  - BorderLayout.CENTER



# EJEMPLO DE GESTOR BORDERLAYOUT

```
import java.awt.*;
class BorderLayoutDemo extends Frame {
    public static void main(String args[]) {
        BorderLayoutDemo bld = new BorderLayoutDemo();
        bld.setLayout(new BorderLayout(10, 10));
        bld.add(new Button("NORTE"), BorderLayout.NORTH);
        bld.add(new Button("SUR"), BorderLayout.SOUTH);
        bld.add(new Button("ESTE"), BorderLayout.EAST);
        bld.add(new Button("OESTE"), BorderLayout.WEST);
        bld.add(new Button("CENTRO"), BorderLayout.CENTER);
        bld.setSize(200, 200);
        bld.setVisible(true);
    }
}
```



Resultado



# GESTOR GRIDLAYOUT

- Parecido a FlowLayout
  - La posición de los componentes va de izquierda a derecha y de arriba abajo
  - Empieza a añadir componentes en la esquina superior izquierda
- Divide el contenedor en un número de filas y columnas – Las regiones son de igual tamaño
  - Ignora el tamaño del componente principal



# GESTOR GRIDLAYOUT

## Resumen de Constructores

GridLayout()	Creates a grid layout with a default of one column per component, in a single row.
GridLayout(int rows, int cols)	Creates a grid layout with the specified number of rows and columns.
GridLayout(int rows, int cols, int hgap, int vgap)	Creates a grid layout with the specified number of rows and columns.



# EJEMPLO DE GESTOR GRIDLAYOUT

```
import java.awt.*;
class GridLayoutDemo extends Frame {
    public static void main(String args[]) {
        GridLayoutDemo gld = new GridLayoutDemo();
        • gld.setLayout(new GridLayout(2, 3, 4, 4));
        • gld.add(new Button("UNO"));
        • gld.add(new Button("DOS"));
        gld.add(new Button("TRES"));
        gld.add(new Button("CUATRO"));
        gld.add(new Button("CINCO"));

        gld.setSize(200, 200);
        gld.setVisible(true);
    }
}
```



Resultado



# PANELS Y LAYOUT COMPLEJOS:

- Para layouts complejos:
  - Se puede combinar los diferentes gestores de layouts – Uso de panels al mismo tiempo
- Recordar que:
  - Un Panel es un Container y un Component
  - Se puede insertar componentes en un Panel
  - Se puede añadir Panel a un Container



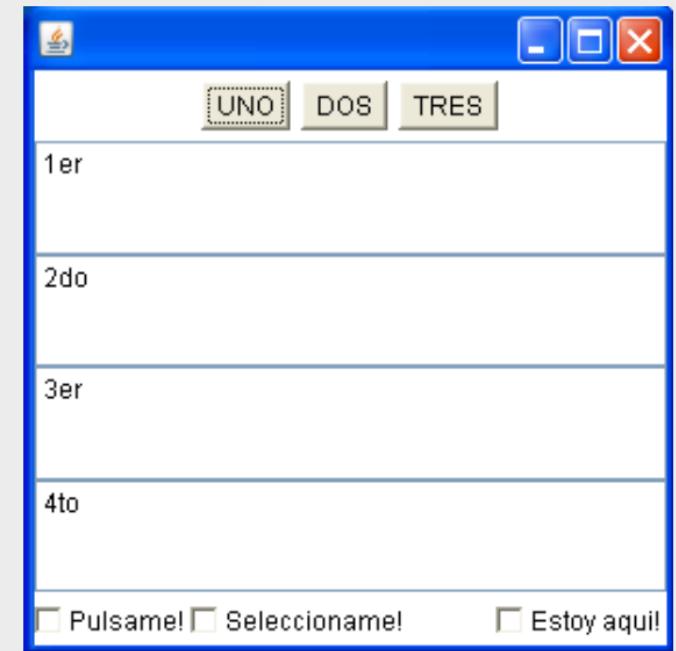
# EJEMPLO DE PANELS Y LAYOUT COMPLEJOS:

```
import java.awt.*;
class ComplexLayout extends Frame {
    public static void main(String args[]) {
        ComplexLayout cl = new ComplexLayout();
        Panel panelNorth = new Panel();
        Panel panelCenter = newPanel();
        Panel panelSouth = newPanel();
        /* North Panel */
        // Panels usan FlowLayout por defecto
        panelNorth.add(new Button("UNO"));
        panelNorth.add(new Button("DOS"));
        panelNorth.add(new Button("TRES"));
        /* Center Panel */
        panelCenter.setLayout(new GridLayout(4,1));
        panelCenter.add(new TextField(" 1 er"));
        panelCenter.add(new TextField(" 2 do"));
        panelCenter.add(new TextField(" 3 er"));
        panelCenter.add(new TextField("4to"));
```



# EJEMPLO DE PANELS Y LAYOUT COMPLEJOS:

```
/* South Panel */
panelSouth.setLayout(new BorderLayout());
panelSouth.add(new Checkbox("Seleccioname!"),
              BorderLayout.CENTER);
panelSouth.add(new Checkbox("Estoy aqui!"),
              BorderLayout.EAST);
panelSouth.add(new Checkbox("Pulsame!"),
              BorderLayout.WEST);
/* Adding the Panels to the Frame */
//Frames use BorderLayout by default
cl.add(panelNorth, BorderLayout.NORTH);
cl.add(panelCenter, BorderLayout.CENTER);
cl.add(panelSouth, BorderLayout.SOUTH);
cl.setSize(300,300); cl.setVisible(true);
}
```



Resultado

# ↑ ANT Y SWING

- Los componentes de Swing tienen nombres que comienzan con J.
  - ◆ Ejemplo: Button en AWT es JButton en Swing
- Los componentes de AWT están en el paquete `java.awt`, los de Swing en `javax.swing`.
- ```
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.*;
```



# PAQUETES JFC / SWING:

- **javax.swing.plaf**
- **javax.swing.plaf.basic**
- **javax.swing.plaf.metal**
- **javax.swing.plaf.multi**

- **javax.swing**
- **javax.swing.table**
- **javax.swing.tree**
- **javax.swing.border**
- **javax.swing.colorchooser**
- **javax.swing.filechooser**

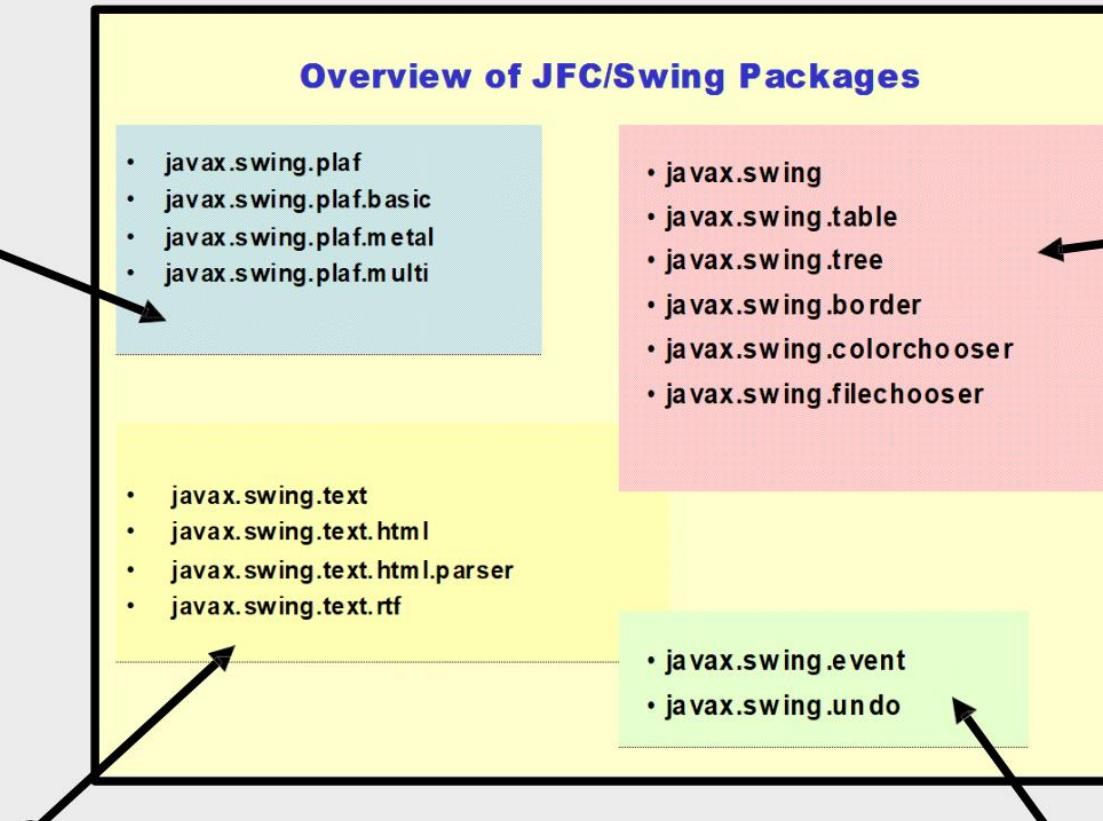
- **javax.swing.text**
- **javax.swing.text.html**
- **javax.swing.text.html.parser**
- **javax.swing.text.rtf**

- **javax.swing.event**
- **javax.swing.undo**



# PAQUETES JFC / SWING:

**Control del  
“Look & Feel”  
de Swing**



**Componentes,  
incluyendo  
componentes  
complejos**

**Widgets basados en texto  
( incluyendo html/rtf )**

**Paquetes nuevos de eventos**

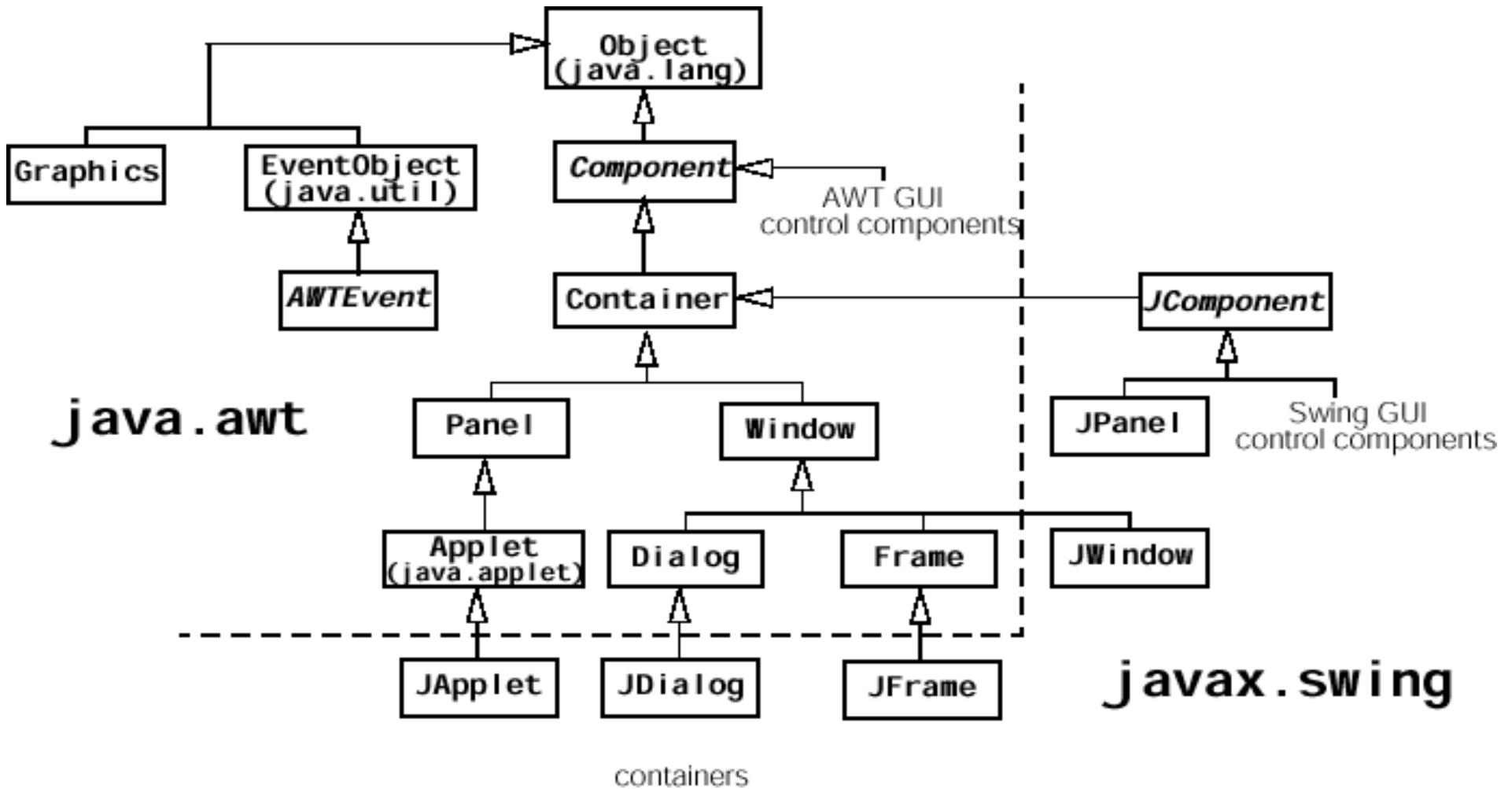


# APLICACIONES BASADAS EN GIU:

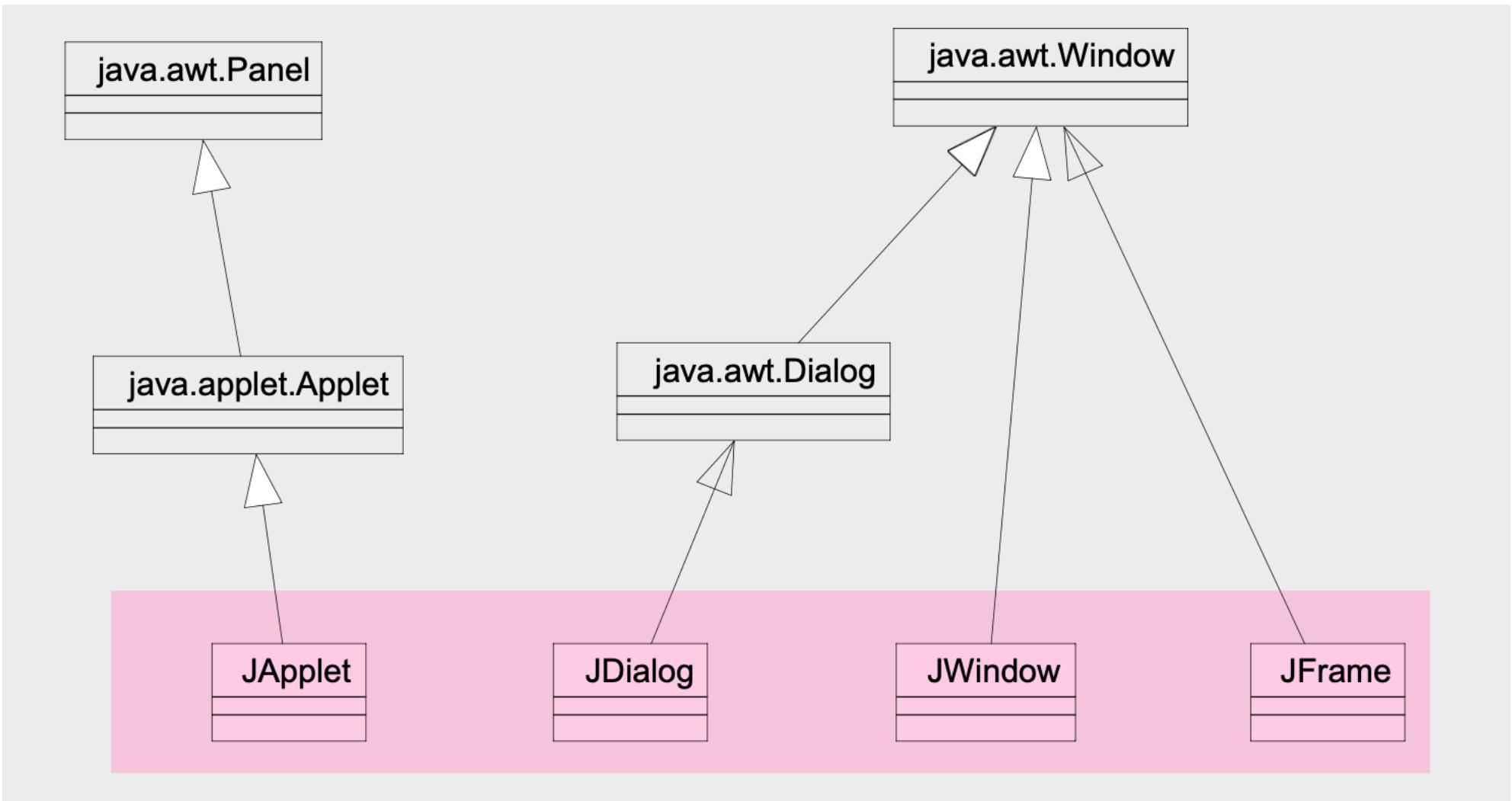
El desarrollo de una aplicación basada en GUI requiere la comprensión de:

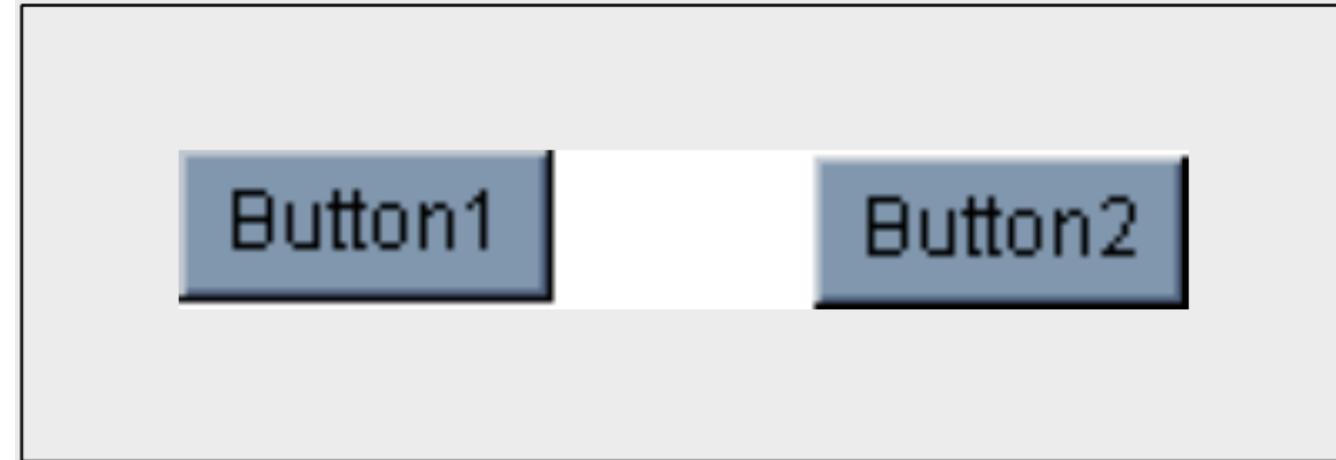
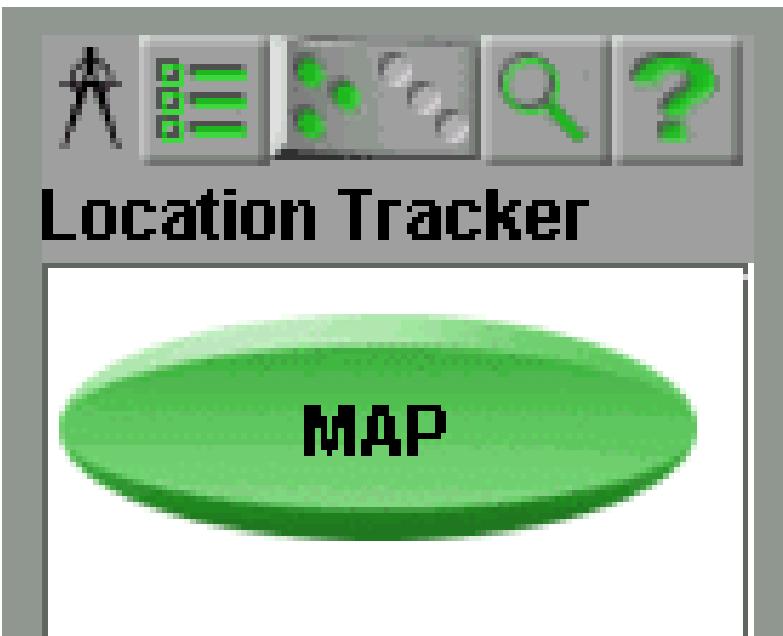
- Estructura de la jerarquía de herencia, que define el comportamiento y atributos de los componentes en la GUI de la aplicación.
- Estructura de la jerarquía de contenedores, que define cómo se disponen todos los componentes en la GUI de la aplicación.
- Manejo de eventos.

# ↑ JERARQUÍA DE HERENCIA



# ↑ HERENCIA “HEAVY”

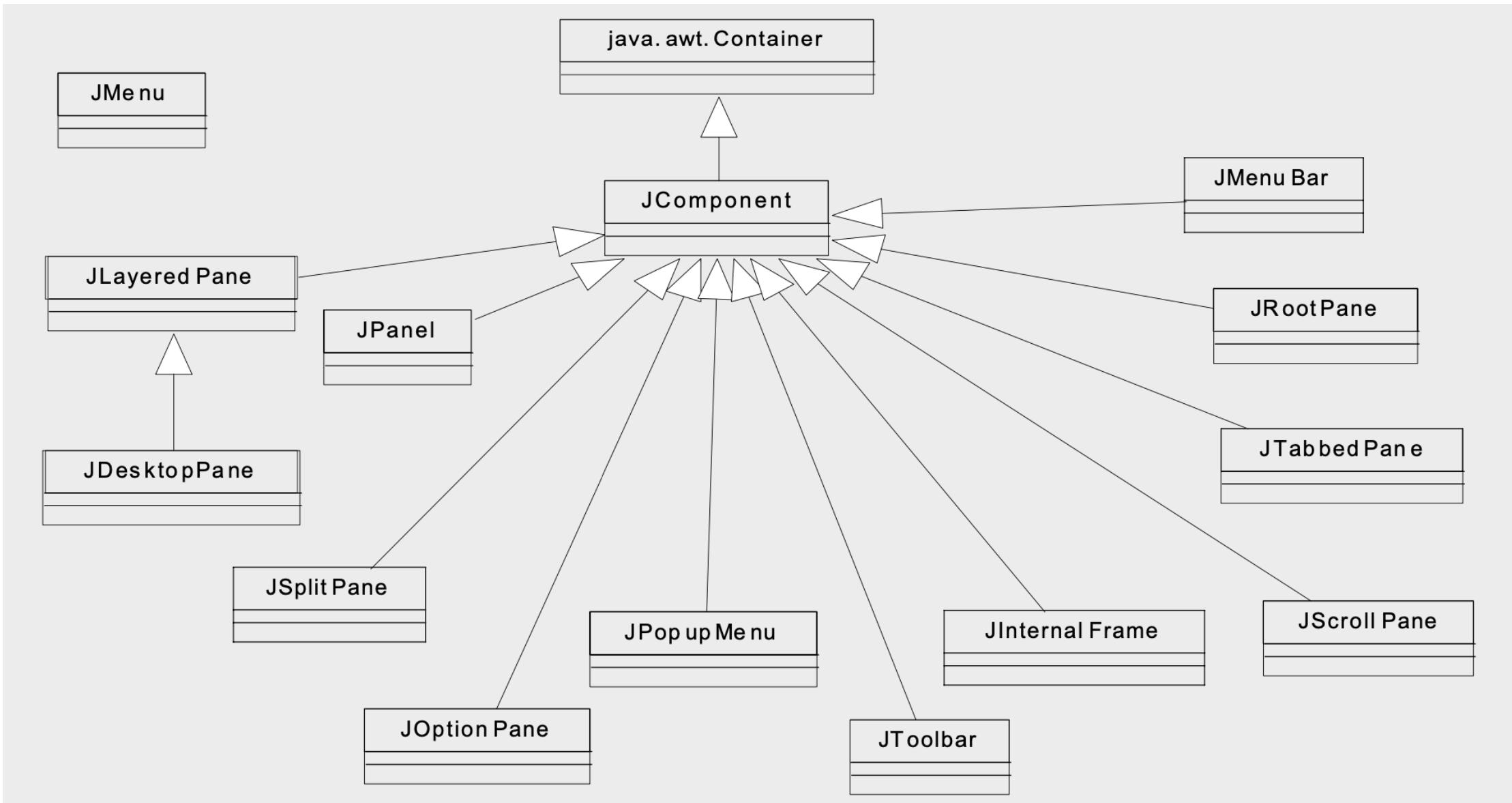




# HEAVYWEIGHT VS. LIGHTWEIGHT

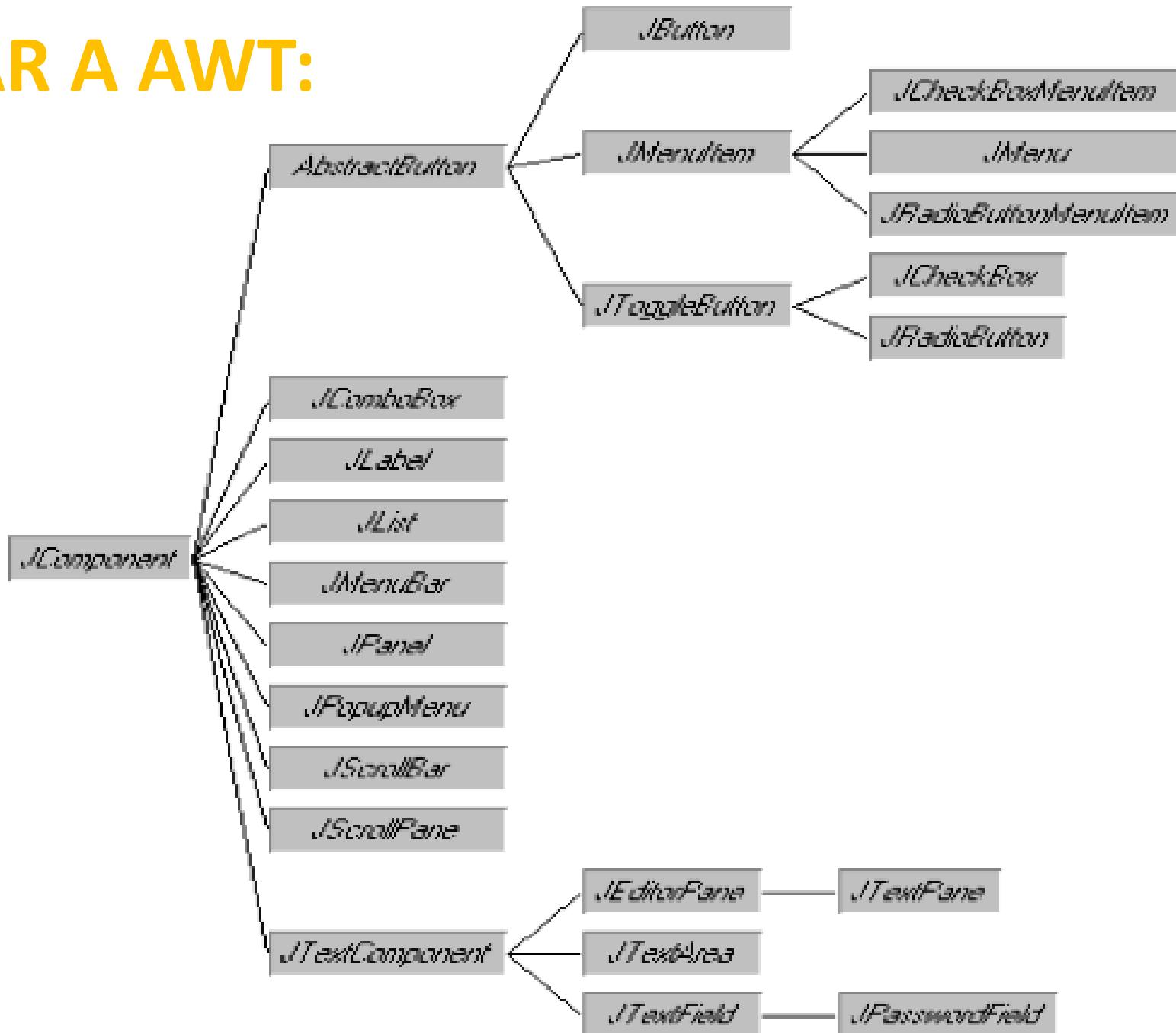


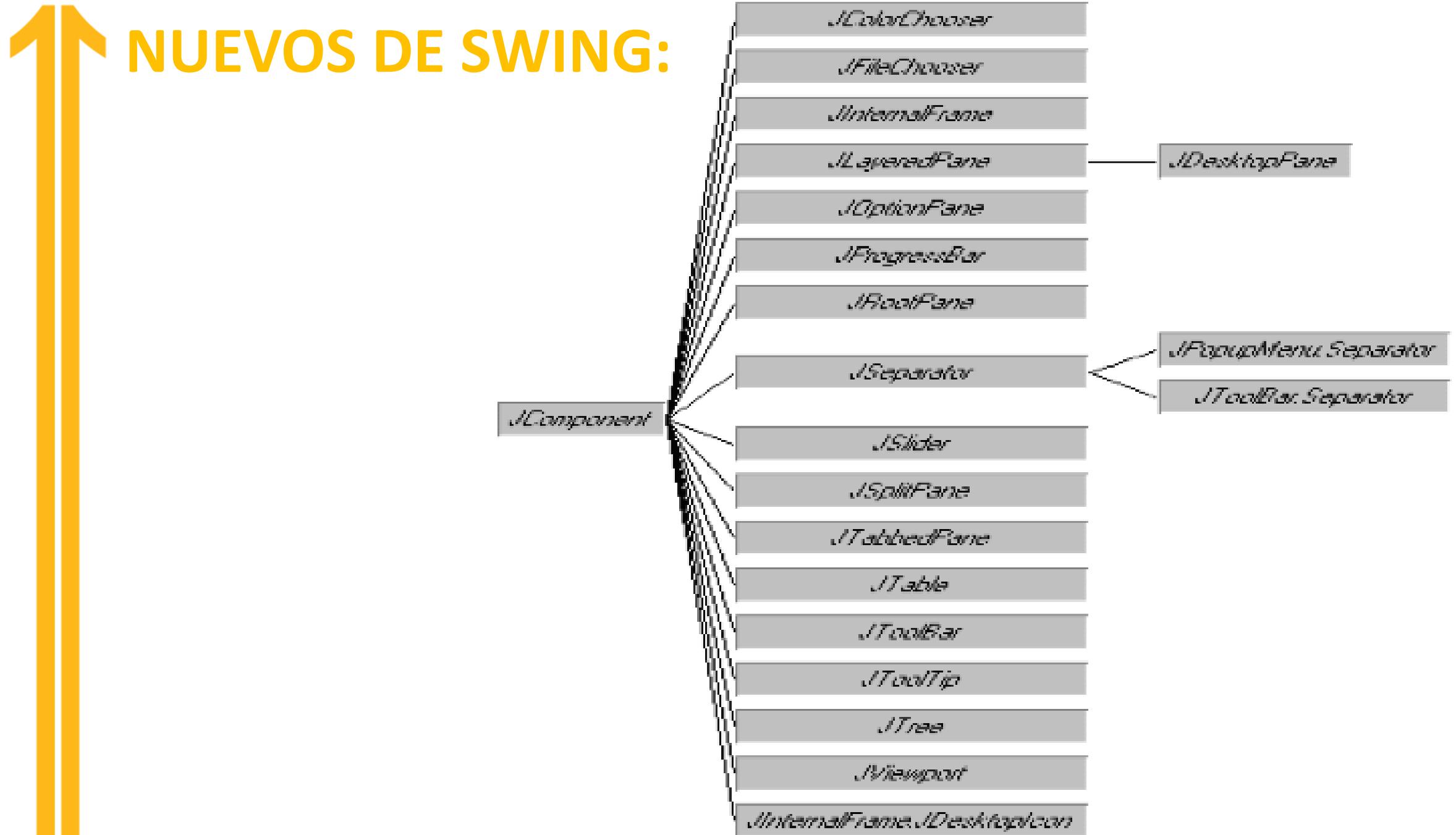
# CONTENEDORES EN SWING:





# SIMILAR A AWT:







# COMPONENTES DE SWING

La clase Component (y sus subclases) proveen soporte para manejo de eventos, cambio del tamaño de un componente, control de color y fuentes, pintado.

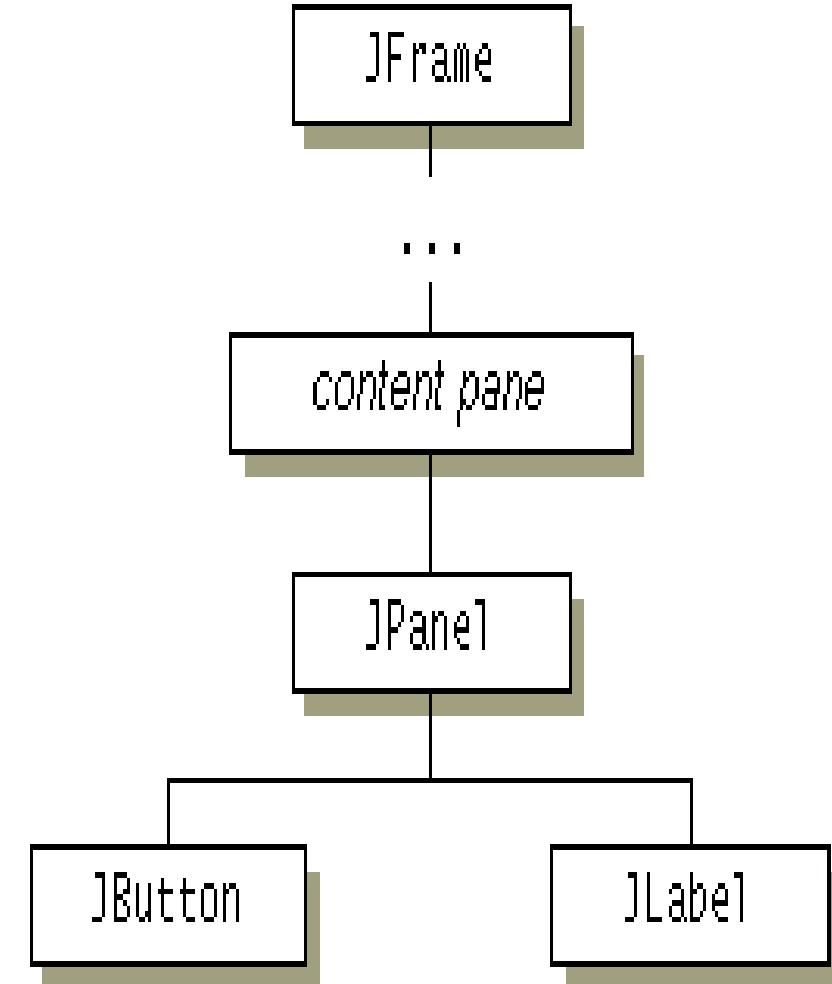
Un componente es un objeto de una subclase concreta. Se distinguen dos clases de componentes:

- Componentes de control de la GUI: la interacción de la GUI con el usuario se realiza a través de ellos.
- Contenedores: contienen otros componentes (u otros contenedores).

# CONTENEDORES:

- Anidamiento de componentes (Jerarquía de contenedores en contraste con la Jerarquía de herencia). Cada programa Swing contiene al menos una.
- Usan un Layout Manager para determinar cómo se disponen los componentes en los contenedores.
- Swing provee 4 contenedores de alto nivel (ventana base de la GUI): JFrame, JApplet, JDialog y JWindow.
- La jerarquía está compuesta de diferentes capas. Cada contenedor de alto nivel contiene un contenedor intermedio conocido como “content pane”. En casi todos los programas no es necesario conocer qué hay entre el contenedor de alto nivel y el content pane.

# ↑ JERARQUÍA DE CONTENEDORES:





# JERARQUÍA DE CONTENEDORES:

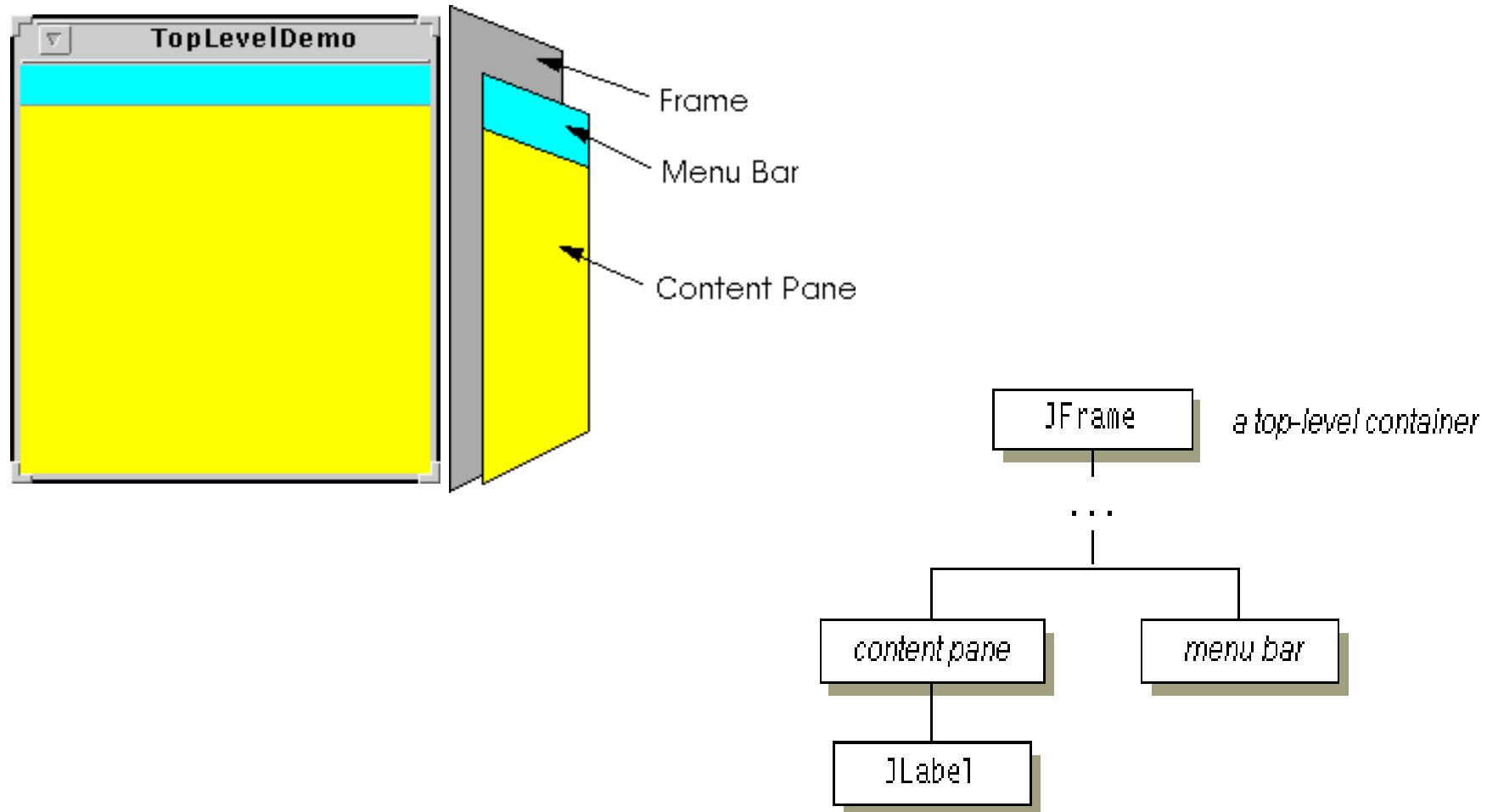
La apariencia de una GUI está determinada por:

- La jerarquía de contenedores
- El Layout Manager de cada contenedor
- Las propiedades de los componentes individuales

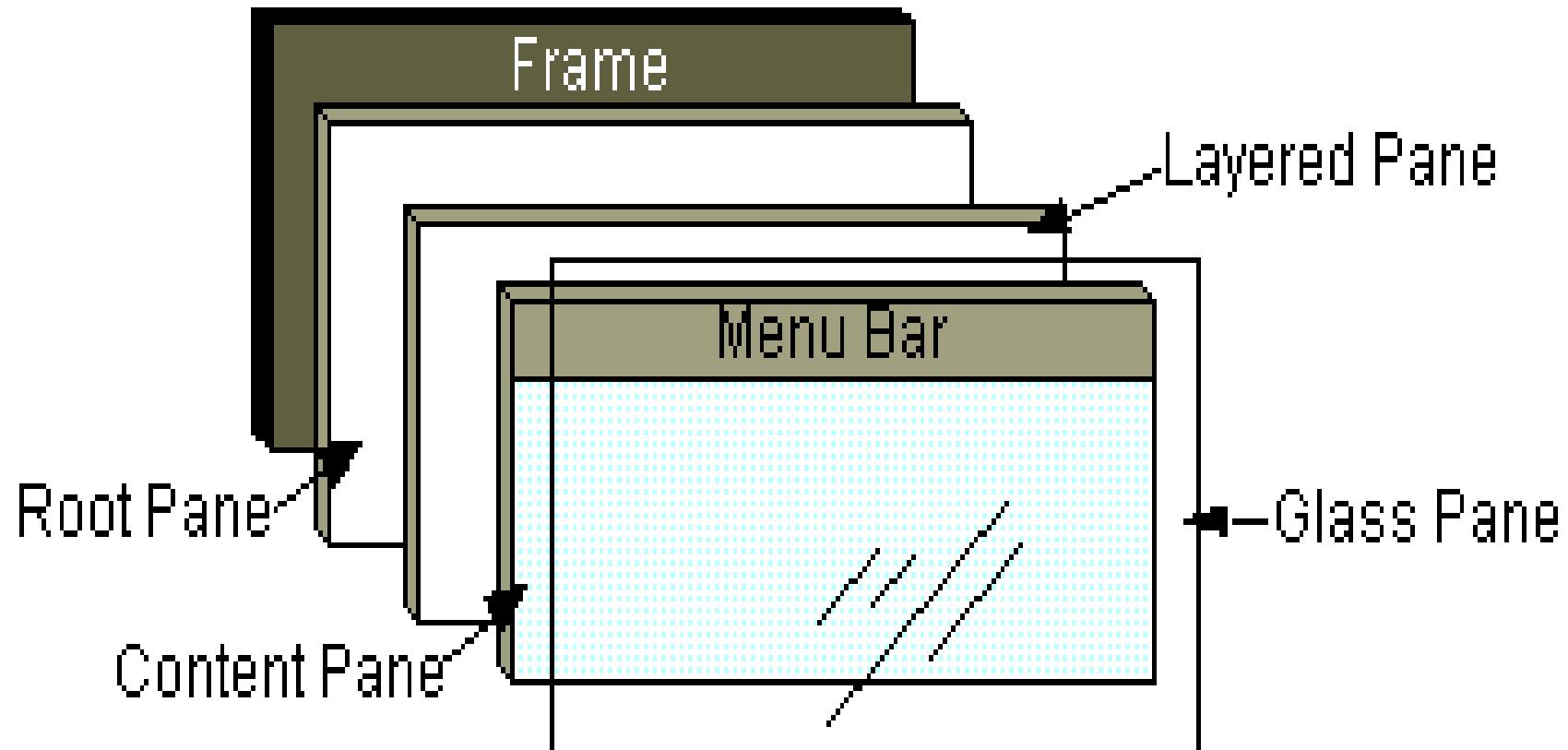
Todos estos ítems trabajan en conjunto para determinar el efecto visual final.



# ESTRUCTURA DE UN JFRAME



# ↑ ESTRUCTURA DE UN JFRAME



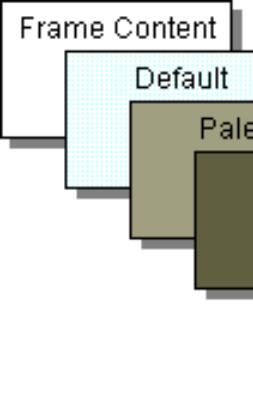
# ROOT PANES

- “Añadido” en forma invisible al contenedor de alto nivel.
- Creado por Swing cuando instancia un contenedor de alto nivel.
- Maneja prácticamente todo entre el contenedor de alto nivel y los componentes atómicos.
- Tener en cuenta si necesita interceptar clicks del mouse o pintar sobre varios componentes.
- Es una instancia de `JLayeredPane` la que contiene la barra de menús y el content pane.



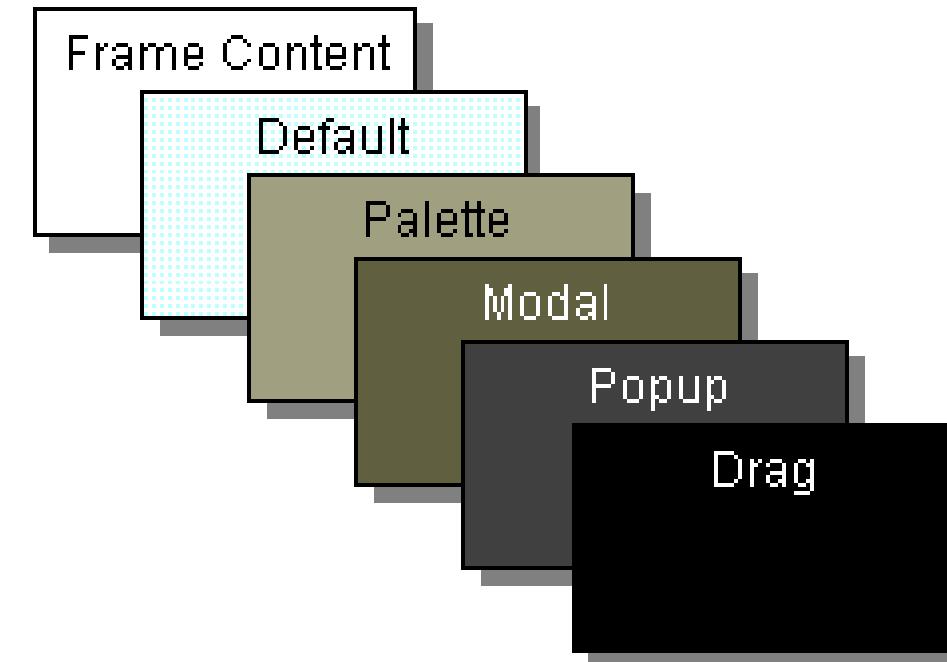
# CONTENT PANES

- Usualmente es un JPanel.
- En la mayoría de las aplicaciones Swing contiene casi todo, excepto la barra de menú.
- Debe ser creado explícitamente.



# ED PANES

- Provisto por root pane pero también puede crearse.
- Contenedor con profundidad, tal que componentes que se superponen (ej:popup menus) pueden aparecer unos encima de otros (z-buffering).



# ↑ GLASS PANE

- Util para pintar o interceptar eventos (por ejemplo: bloquear todos los eventos del mouse) en un área que contenga uno o más componentes.





# CONTENEDORES DE ALTO NIVEL





# CONTENEDORES GENERALES

A Label on a Panel

Color and font test:

- red
- blue
- green
- small

Panel

ScrollPane

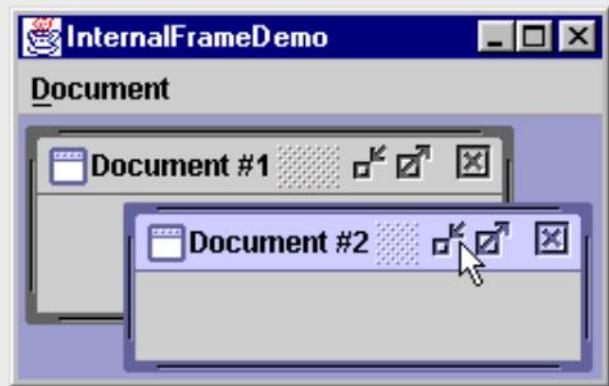
SplitPane

TabbedPane

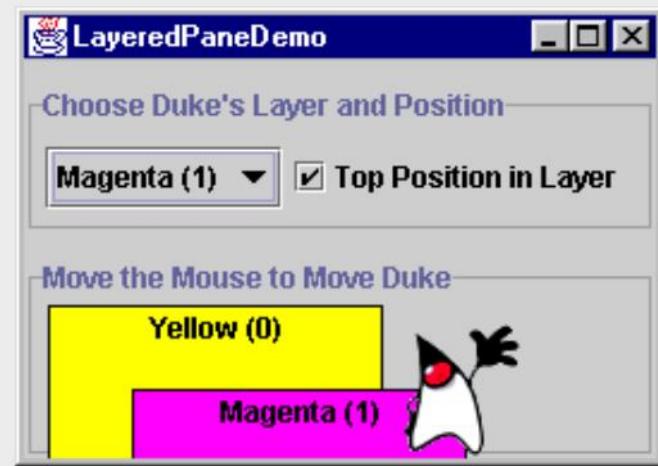
Toolbar



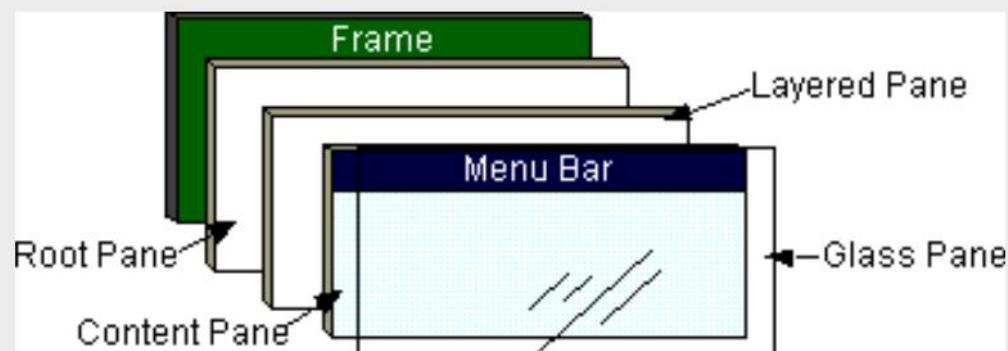
# CONTENEDORES ESPECIALES



InternalFrame



LayeredPane



Root Pane

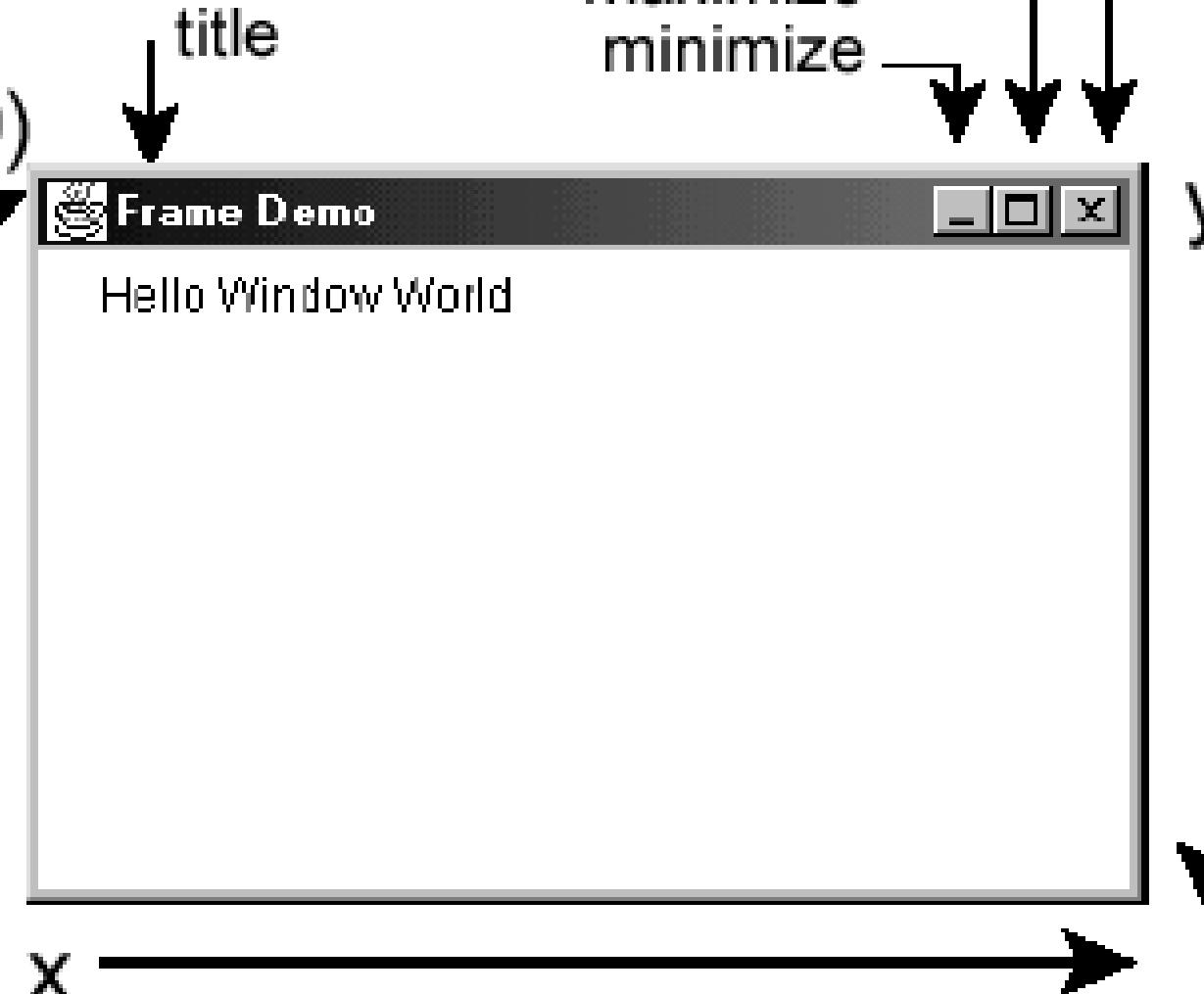


# JFRAME:

Java™ frame coordinate system

(0,0)

system pull-down  
menu button



- Measured in pixels
- Everything in a Frame is drawn on a Graphics context



# ↑ ALGUNOS MÉTODOS DE JFRAME:

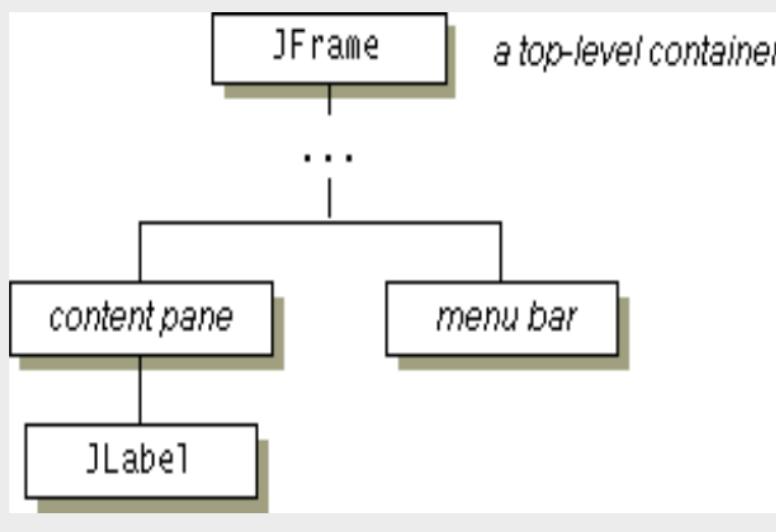
- Para añadir componentes al content pane:

```
myFrameInstance.getContentPane().add(myComponent);
```

Usual  
“this”

Requerido por  
Instancias de  
**JFrame,**  
**JDialog** y  
**JInternalFrame**

Ej.:  
“myLabel”



# ↑ ALGUNOS MÉTODOS DE JFRAME:

- Para construir una ventana con un título y mostrarla:

```
JFrame theWindow = new JFrame( "Graffiti" );
theWindow.show( );
//idem theWindow.setVisible(true);
```



- Para determinar su tamaño:

```
theWindow.setJMenuBar(cyanMenuBar);
theWindow.setSize( 220, 100 );//o mejor
theWindow.pack();
```



# NUEVA FUNCIONALIDAD DE JFRAME:

setDefaultCloseOperation(int) es la más importante:

- DO NOTHING ON CLOSE
- HIDE ON CLOSE (default) oculta el frame cuando el usuario lo cierra pero no se deshace de los recursos del sistema asociados (puedo volver a mostrar).
- DISPOSE ON CLOSE oculta el frame y llama al método dispose(), para liberar recursos.
- EXIT ON CLOSE, cierra la aplicación (System.exit(0))



# HELLO WORLD EN SWING:

```
import javax.swing.*;  
  
public class HelloWorldSwing {  
  
    public static void main(String[] args) {// crear un nuevo frame  
        JFrame frame = new JFrame("HelloWorldSwing");  
  
// crear una etiqueta y añadir al frame  
        JLabel label = new JLabel("Hello World");  
        frame.getContentPane().add(label);  
  
// especifica la operación de cierre de la ventana  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
// muestra el frame en pantalla  
        frame.pack(); frame.setVisible(true);}}}
```



# ↑ COMPONENTES ATÓMICOS:

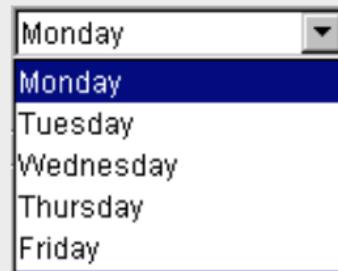
Componentes de tipo JPanel almacenan otros objetos de la GUI tales como: botones, etiquetas, campos de texto, etc. Estos objetos gráficos son considerados componentes atómicos, puesto que no pueden almacenar otros objetos de la GUI.



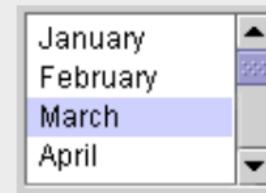
# COMPONENTES ATÓMICOS BÁSICOS:



Buttons



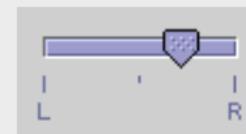
ComboBox



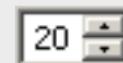
List



Menu



Slider



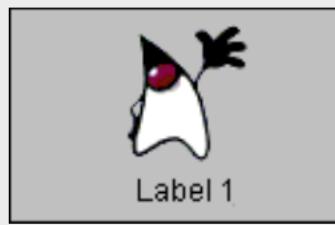
Spinner



TextFields



# COMPONENTES ATÓMICOS NO EDITABLES:



Label



Progress Bar



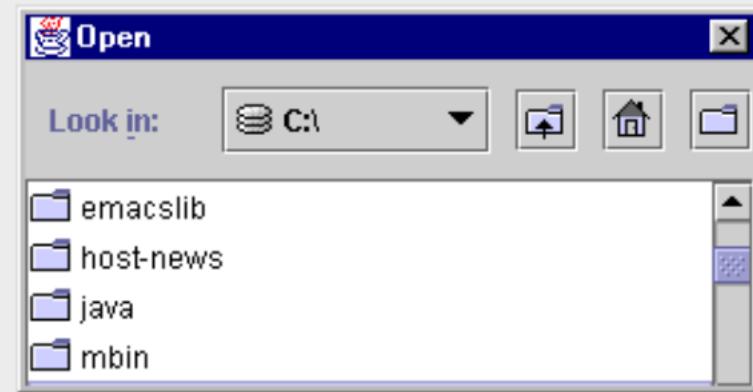
Tooltip



# OTROS COMPONENTES ATÓMICOS:



Color Chooser



File Chooser

| First Na... | Last Name |
|-------------|-----------|
| Mark        | Andrews   |
| Tom         | Ball      |
| Alan        | Chung     |
| Jeff        | Dinkins   |

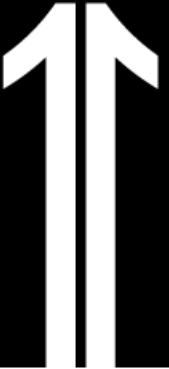
Table

Verify that the RJ45 cable is connected to the WAN plug on the back of the Pipeline unit.

Tex



Tree



# REFERENCIAS DIGITALES

- <https://javadesdecero.es/poo/herencia-java-tipos-ejemplos/>
- [https://www.mundojava.net/la-herencia-en-javascript.html?Pg=java\\_inicial\\_4\\_4\\_6.html](https://www.mundojava.net/la-herencia-en-javascript.html?Pg=java_inicial_4_4_6.html)
- [https://www.aprenderaprogramar.com/index.php?option=com\\_content&view=article&id=653: ejemplo-de-herencia-en-javascript-uso-de-palabras-clave-extends-y-super-constructores-con- herencia-cu00686b&catid=68&Itemid=188](https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=653: ejemplo-de-herencia-en-javascript-uso-de-palabras-clave-extends-y-super-constructores-con- herencia-cu00686b&catid=68&Itemid=188)
- [http://profesores.fi-b.unam.mx/carlos/java/java\\_basico3\\_4.html](http://profesores.fi-b.unam.mx/carlos/java/java_basico3_4.html)

11 GRACIAS

