DISEÑO ORIENTADO A OBJETOS

Prof: Esp. Ing. Agustín Fernandez

Swing

- Todo nace con la biblioteca gráfica Internet Foundation Classes (IFC) desarrollada para Java originalmente por Netscape y que se publicó en 1996.
- Como vimos, desde sus inicios, el entorno Java ya contaba con AWT como biblioteca de componentes gráficos y también vimos que AWT estaba concebida como una API estandarizada que permitía utilizar los componentes nativos de cada sistema operativo.
- Lo cual en la práctica hizo que esta tecnología no funcionase como se espera.

Swing

- En cambio los componentes de IFC eran mostrados y controlados directamente por código Java independiente de la plataforma.
- Además al estar enteramente desarrollada en Java aumentaba su portabilidad lo cual aseguraba un comportamiento idéntico en diferentes plataformas.
- Entonces 1997, Sun Microsystems y Netscape Communications Corporation anunciaron su intención de combinar IFC con otras tecnologías de las Java Foundation Classes dando inicio al proyecto Swing.

Swing

- A los componentes suministrados originalmente por la IFC se los llama con frecuencia como “componentes ligeros”, dado que no requieren reservar recursos nativos del sistema de ventanas del sistema operativo.
- Esto hace que los componentes **Swing sean más pequeños y más eficaces que sus homólogos de AWT.**
- Además de esto, Swing introdujo un mecanismo que permitía que el aspecto de cada componente de una aplicación pudiese cambiar sin introducir cambios sustanciales en el código de la aplicación.
- Esto permitió a Swing emular la apariencia de los componentes nativos del sistema operativo anfitrión manteniendo las ventajas de la independencia de la plataforma. **Look and Feel Conectable**

Swing

- **Look and Feel Conectable**

```
public static void main(String[] args) {
    try {
      UIManager.setLookAndFeel("Look and feel valido");
    } catch (Exception e) {
    }
    ...//Trabajar normalmente ...
}
```

- Posibles Look and Feel

```
"javax.swing.plaf.metal.MetalLookAndFeel"
"com.sun.java.swing.plaf.windows.WindowsLookAndFeel"
"com.sun.java.swing.plaf.motif.MotifLookAndFeel"
"javax.swing.plaf.mac.MacLookAndFeel"
```

Swing

• Componentes Complejos
Dependencia fuerte de código peer nativo. Cada uno de los componentes complejos está estrechamente asociado a un componente peer nativo del entorno cliente.
Cada uno se presenta en su propia ventana opaca.
Los componentes complejos se desarrollaron en las primeras versiones de AWT.
Casi todos los primeros componentes AWT eran complejos.
Todos los contenedores de nivel superior son complejos y proporcionan el contexto para los componentes y los contenedores sencillos.
Incluyen algunos componentes de nivel superior Swing (JFrame, JApplet, JDialog).

Swing

Componentes Sencillos
Independencia de código peer nativo.
Pueden tener fondos transparentes.
Casi todos los componentes Swing son sencillos.
Cuando se muestran, su forma puede ser no rectangular.
Se deben visualizar en un contenedor complejo.
Sin embargo, los componentes sencillos son más flexibles visualmente porque pueden ser transparentes y su forma puede ser no rectangular. Estas funciones permiten que los componentes sencillos se adapten fácilmente a diferentes aspectos.
Los componentes sencillos no tienen ningún peer nativo, ya que el código Java los presenta directamente. Por lo tanto, los componentes sencillos son más portátiles.

Swing

- Todos los componentes Swing forman parte del paquete javax.swing, que se agregó en JDK 1.2.
- Swing ha transformado el desarrollo UI de Java proporcionando componentes sencillos y ligeros independientemente del sistema operativo en el que se ejecuta la aplicación.
- Swing ofrece muchos más tipos de componentes UI para la interacción de usuario que los que proporcionaba AWT.

Swing

- Planificando el UI Layout (Cont.)
  - Casi todas las aplicaciones gráficas tienen un área de presentación principal (normalmente una ventana principal).
  - En Java, la ventana principal se denomina contenedor de nivel superior.
  - El contenedor de nivel superior se considera la raíz de la jerarquía de contención para la ventana o el área.

Swing

- Planificando el UI Layout (Cont.)
  - Una ventana principal se puede dividir en regiones o secciones, que se representan mediante contenedores intermedios y, al final, los componentes que contienen los datos de usuario o que aceptan la entrada de usuario.
  - Estos componentes se posicionarán dentro de los contenedores intermedios o de nivel superior.
  - Los contenedores intermedios y de nivel superior, junto con sus componentes, forman una jerarquía de contención.
  - Cada contenedor utiliza un gestor de diseños para controlar el tamaño y la ubicación de los componentes dentro de un contenedor.

Swing

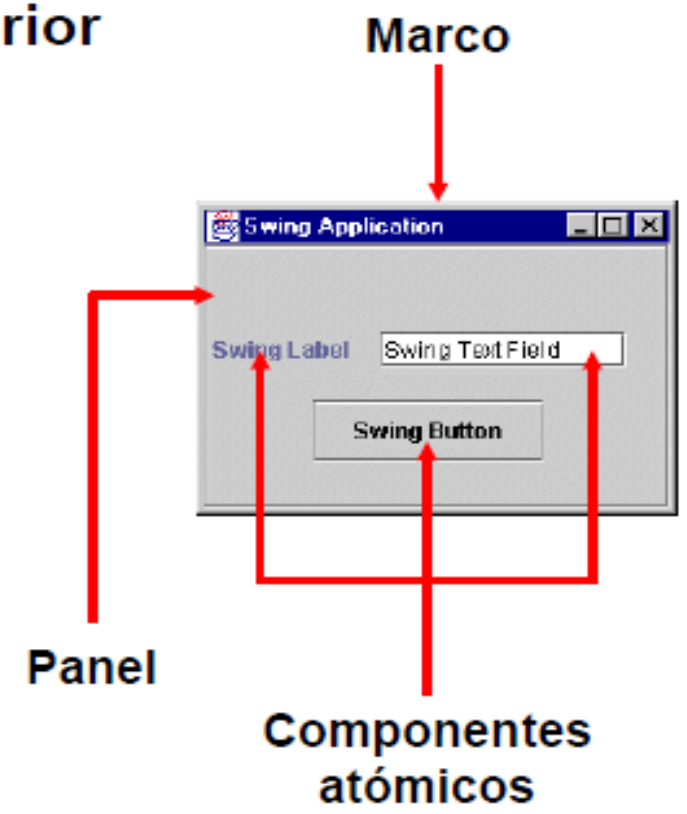
- Planificando el UI Layout (Cont.)
  - Los contenedores de nivel superior no se pueden colocar dentro de otro contenedor de nivel superior y, normalmente, contienen un contenedor intermedio denominado panel de contenido.
  - Los contenedores de nivel superior que se utilizan con más frecuencia son JFrame, JDialog y Japplet.
  - Los contenedores intermedios simplifican el modo de organización de los elementos visuales dentro de un contenedor de nivel superior y pueden contener otros contenedores intermedios y componentes atómicos de nivel inferior.

Swing

- Planificando el UI Layout (Cont.)
  - Por ejemplo, un panel puede estar anidado dentro de otro panel. Los contenedores intermedios más comunes son JPanel, JScrollPane, JSplitPane y JToolBar.
  - Los componentes atómicos son entidades autosuficientes (u objetos gráficos) que se utilizan para presentar información al usuario o para recibir datos del mismo.
  - Los componentes atómicos más comunes son JButton, JLabel y JTextField. Existen muchos componentes atómicos para texto, recuadros combinados, casillas de control, listas, etc.

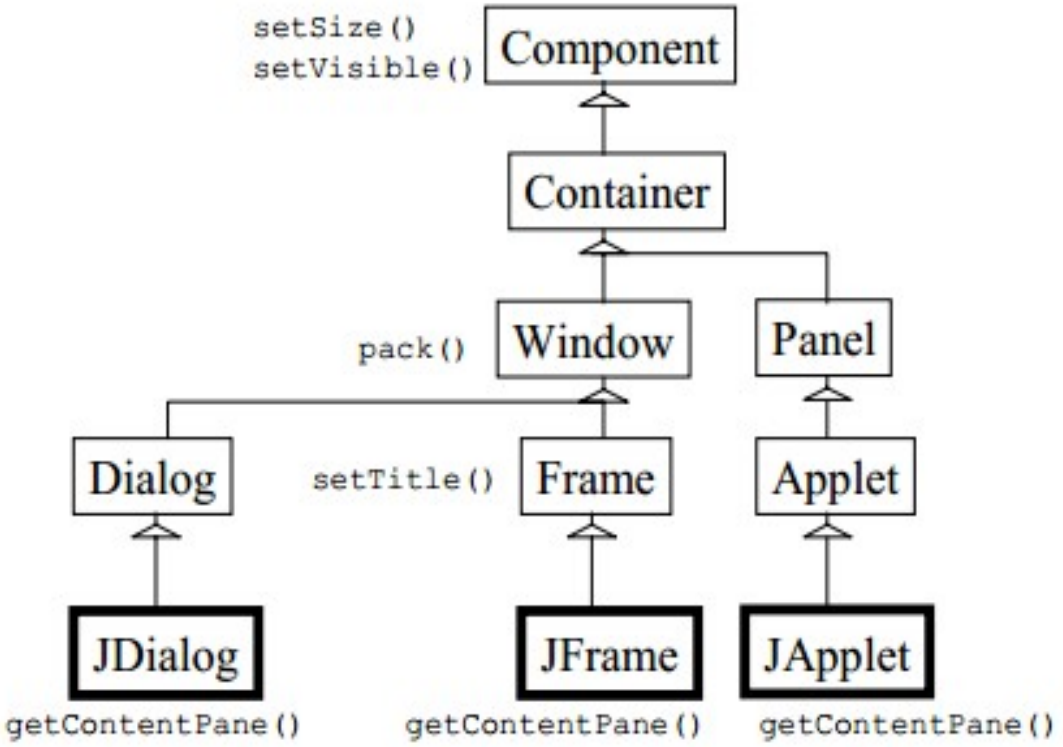
Swing

- Planificando el UI Layout (Cont.)
  - **Contenedores de nivel superior**
    - **Marco**
    - **Diálogo**
    - **Applet**
  - **Contenedores intermedios**
    - **Panel**
    - **Panel de desplazamiento**
  - **Componentes atómicos**
    - **Etiqueta**
    - **Elementos de texto**
    - **Botones**



Swing

- Planificando el UI Layout (Cont.)



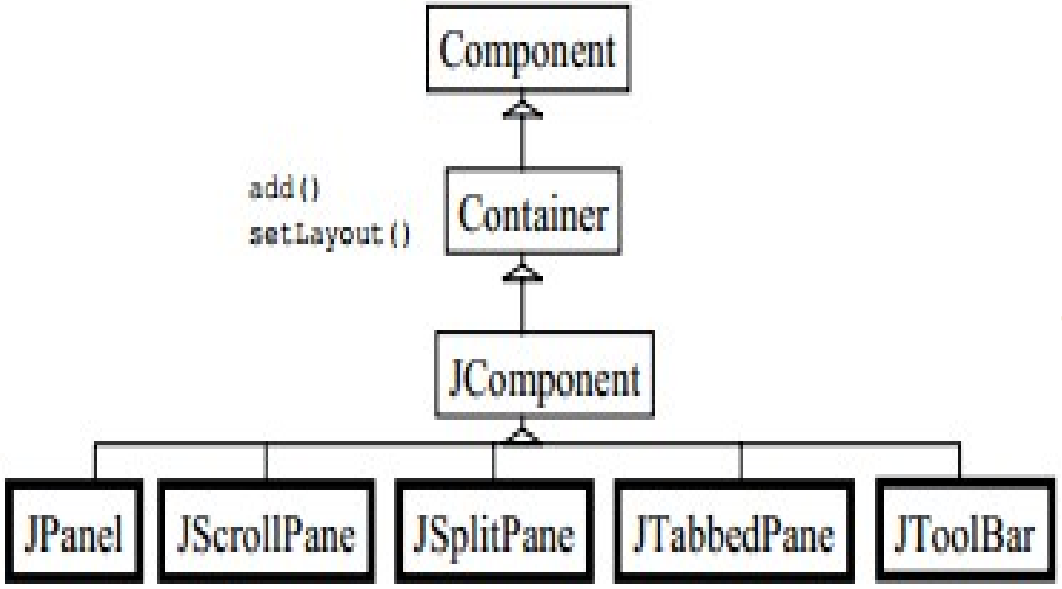
Swing

- Planificando el UI Layout (Cont.)

  - Swing proporciona `JFrame`, `JDialog` o `JApplet` con propiedades modificables como, por ejemplo:
    - Un panel de contenido para componentes o contenedores intermedios, mediante los métodos `getContentPane()` o `setContentPane()`
    - Un borde, mediante un método `setBorder()`
    - Un título, mediante un método `setTitle()`
    - Decoraciones de ventana como, por ejemplo, botones para cerrar y minimizar (se excluyen los applets)
  - AWT proporciona `Frame`, `Dialog` o `Applet`.
    - Estos elementos no proporcionan propiedades (como, por ejemplo, bordes o un panel de contenido).

Swing

- Planificando el UI Layout (Cont.)



Swing

- Planificando el UI Layout (Cont.)
  - **Diseñados para contener componentes (o contenedores): Se pueden anidar dentro de otros contenedores**
  - **Tipos de contenedores intermedios:**
    - **Paneles para agrupar contenedores o componentes**
    - **Paneles de desplazamiento para agregar barras de desplazamiento en componentes que pueden aumentar como, por ejemplo, una lista o un área de texto**
    - **Paneles de división para visualizar dos componentes en un área fija, que puede ajustar el usuario**
    - **Paneles de separador para contener varios componentes, mostrando sólo uno a la vez, según la selección del usuario**
    - **Barras de herramientas para agrupar componentes como, por ejemplo, botones**
    - **Marcos internos para ventanas anidadas**

Swing

- Planificando el UI Layout (Cont.)
  - Si se anidan los contenedores intermedios dentro de otros contenedores, puede controlar el diseño de la aplicación.
  - **Paneles:** son los contenedores intermedios que se utilizan con más frecuencia. Se implementan con la clase JPanel, normalmente, se utilizan para agrupar componentes para la presentación lógica al usuario. Un JPanel puede utilizar cualquier gestor de diseños; utiliza por defecto FlowLayout y se puede definir su borde en cualquier valor.
  - **Panel de desplazamiento:** proporciona barras de desplazamiento para componentes grandes o para componentes que pueden aumentar. Se implementa con JScrollPane.

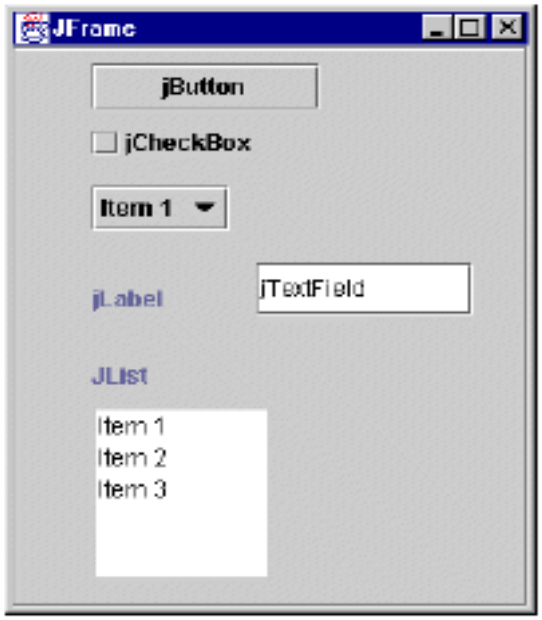
Swing

- Planificando el UI Layout (Cont.)
  - **Panel de división:** este contenedor se utiliza para presentar dos componentes en una cantidad fija de espacio, permitiendo al mismo tiempo que el usuario ajuste el espacio dedicado a cada elemento. El panel de división se implementa con JSplitPane.
  - **Panel de separador:** este contenedor tiene varios componentes, pero el usuario sólo puede ver uno a la vez. El usuario puede cambiar de un componente a otro haciendo clic en los separadores visibles. Los separadores se implementan con JTabbedPane.
  - **Barra de herramientas:** además de contener varios componentes, el usuario puede colocar en otra ubicación las instancias de JToolBar.
  - **Marco interno:** Los contenedores de nivel superior pueden soportar marcos o ventanas internas, que se implementan mediante JInternalFrame y se utilizan mejor con JDesktopPane.

Swing

- Planificando el UI Layout (Cont.)
  - **Componentes Atómicos:**

- Botones
- Casillas de control
- Recuadros combinados
- Texto
- Listas
- Etiquetas



Swing

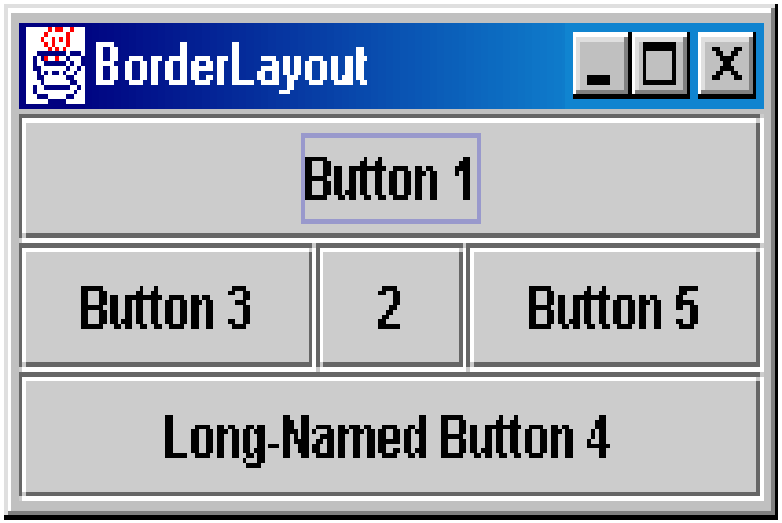
- Gestores de Diseño:
  - Sirven para controlar el proceso de colocación de componentes en un contenedor en tiempo de ejecución.
  - Cada contenedor tiene un gestor de diseños por defecto.
  - Al final, el gestor de diseños controla el diseño y la posición de los componentes dentro del contenedor.
  - Sin embargo, cada componente puede proporcionar indicaciones sobre sí mismo para ayudar al gestor de diseños (por ejemplo, su posición y tamaño preferidos).

Swing

- Gestores de Diseño: (Cont.)
  - Java proporciona muchos gestores de diseños.
  - Los cinco siguientes son los que se utilizan con más frecuencia:
  - **java.awt.BorderLayout:** gestor de diseños por defecto para los contenedores Jframe:
    - Organiza el contenedor en cinco áreas, denominadas North, South, East, West y Center.
    - El área Center se amplia hasta rellenar todo el espacio disponible.
    - Muestra un solo componente por área, para salvar esta restricción se deben usar componentes intermedios.

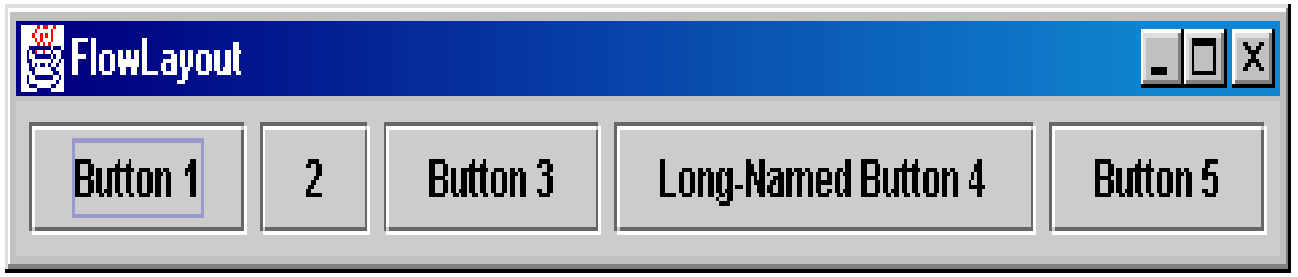
Swing

- Gestores de Diseño: (Cont.)
- **BorderLayout:**



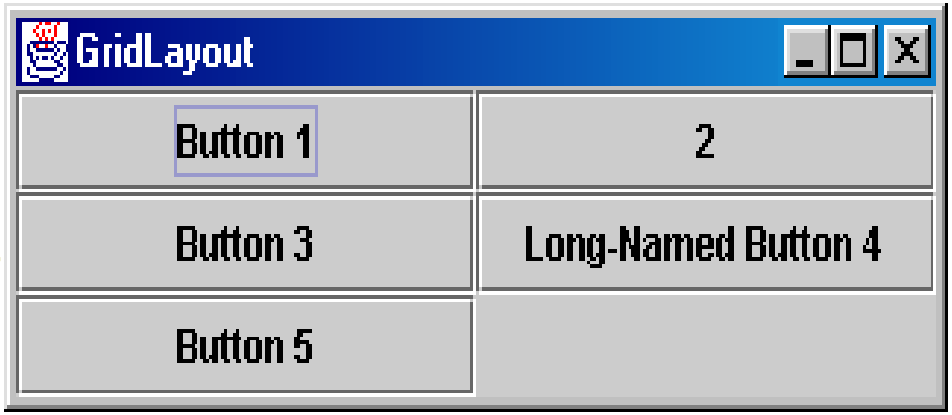
Swing

- Gestores de Diseño: (Cont.)
  - **java.awt.FlowLayout:** gestor de diseños por defecto para Jpanel:
    - Organiza los elementos de izquierda a derecha y, después, de arriba a abajo.
    - Las filas se pueden centrar (por defecto), justificar a la derecha o justificar a la izquierda.



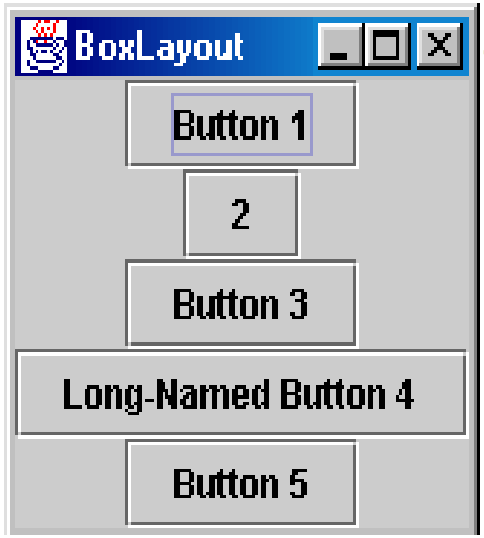
Swing

- Gestores de Diseño: (Cont.)
  - **java.awt.GridLayout:** organiza los elementos en una cuadricula de filas y columnas con celdas del mismo tamaño.



Swing

- Gestores de Diseño: (Cont.)
  - **java.awt.BoxLayout:** Coloca a los componentes a lo largo de un eje:
    - Define dos constantes X_AXIS,Y_AXIS.
    - En el constructor debemos indicar el contenedor y la orientación de los componentes BoxLayout(Container, int).
    - Los componentes no tienen igual tamaño (como en GridLayout).

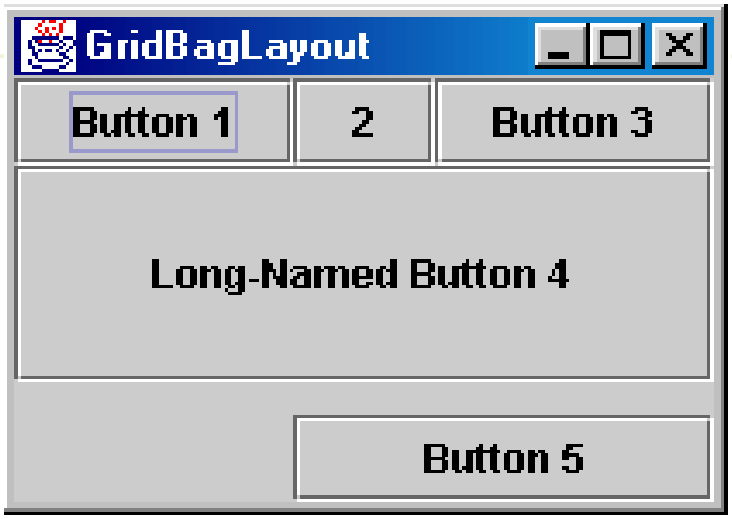


Swing

- Gestores de Diseño: (Cont.)
  - **java.awt.GridBagLayout:** organiza los elementos en una cuadrícula de filas y columnas con diferentes tamaños de celda:
    - Este es el gestor de diseños más flexible y más complejo de todos.
    - Permite que los componentes abarquen varias celdas de columna y de fila.
    - Además, los componentes pueden ofrecer indicaciones o sugerencias sobre cómo prefieren aparecer.
    - Por ejemplo, un componente puede especificar cuánto espacio se debe definir automáticamente a su alrededor, dentro y fuera de su celda.

Swing

- Gestores de Diseño: (Cont.)
  - **java.awt.GridBagLayout:** (Cont.)
    - También puede especificar tamaño mínimo, máximo y preferido de cada componente.
    - Los componentes pueden abarcar varias celdas en las dos direcciones: fila y columna. El tamaño de los componentes que ocupan la fila o la columna determina los tamaños de fila y de columna.



Swing

- Gestores de Diseño: (Cont.)
  - **Combinación de Gestores de contenido:**