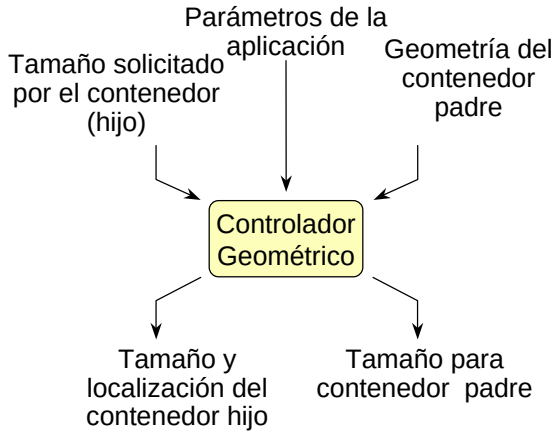
JAVA

5. Controladores Geométricos

http://giig.ugr.es/ ~mgea/docencia/diu
Ultima actualización: 2/Nov/2004

1. Controladores geométricos

- **Tipos**
  - RootLayout
  - FlowLayout
  - BorderLayout
  - CardLayout
  - GridLayout
  - GridBagLayout...

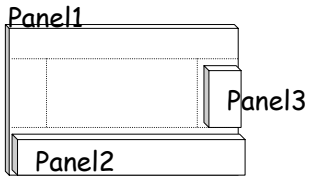


- Controlan posición y tamaño de componentes en pantalla
- Gestores de restricciones geométricas

1. Controladores geométricos (cont.)

❖ Jerarquía de Controladores

Panel1 (BorderLayout)
- Panel2 (FlowLayout)
- Panel3 (BoxLayout)

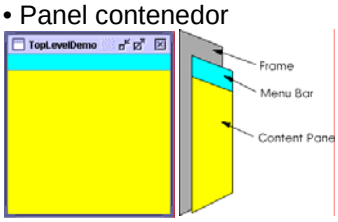


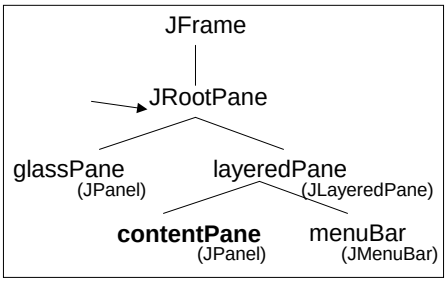
- El controlador geométrico está asociado a un contenedor
- Los elementos se añaden en el contenedor con el método (add)
- Podemos utilizar diferentes controladores para gestionar la pantalla
- Las restricciones internas poseen mayor prioridad que las externas

```
JPanel p1 = new JPanel();      JPanel p2 = new JPanel();
cg = new BorderLayout();              // crea un C.G.
p1.setLayout(new Flowlayout());       // asocia a contenedor
p1.add(p2, BorderLayout.SOUTH);  // añade elementos
```


2. RootLayout

- Aplicable a JFrame (por defecto)
- Maneja varios contenedores
- Gestiona:
  - Panel invisible (foco mouse)
  - Barra de menu (opcional)
  - Panel contenedor

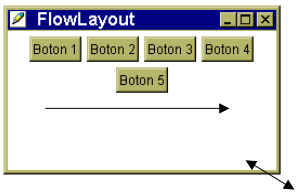





```
JFrame f = new JFrame();
f.getContentPane().setLayout(new Flowlayout());  // nuevo controlador
f.getContentPane().add(...);                      // añade componentes
f.setJMenuBar(new JMenuBar());                    // añade barra menú
```

3. FlowLayout

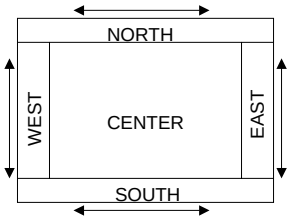
- Ubicación de izda a drcha
- Definir alineación (LEFT, RIGHT, CENTER)
- Control espacio entre controles (xgap, ygap)
- Ajuste automático (mientras quepan)
- Gestor predefinido en JPanel




```
Panel p = new JPanel();
FlowLayout cg = new FlowLayout(FlowLayout.RIGHT));  // ajuste a derecha
p.setLayout (cg);      // asociar a contenedor
p.add(boton1);         // añade componentes
p.add( . . .);
```


4. BorderLayout

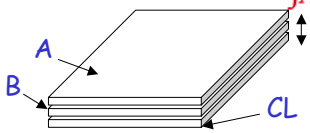
- Los objetos se ubican en 5 zonas
- Cada zona posee expansión diferente
- Útil para definir barra de estado
- Gestor predefinido en JFrame




```
JPanel p = new JPanel();
p.setLayout (new BorderLayout());
p.add(boton1, BorderLayout.NORTH);  // se indica su ubicación
p.add(boton2, BorderLayout.EAST);
```

5. CardLayout

- Gestión de contenedores por capas (Pila)
- Comparten la misma área de pantalla
- Cubren el área completa (padre)
- Métodos para conmutación

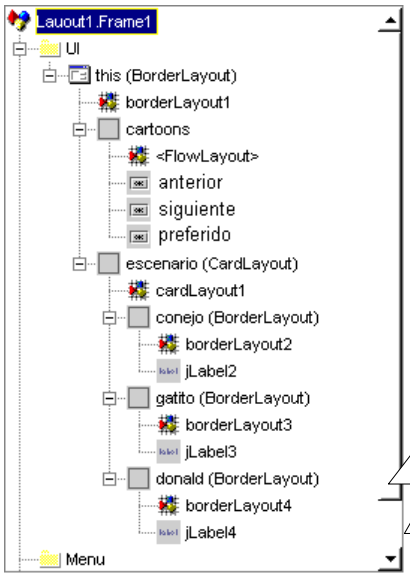


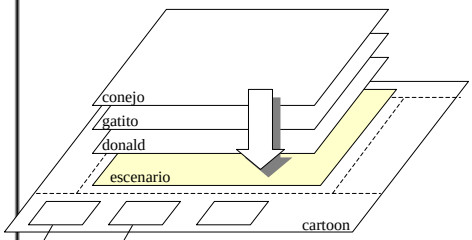

```
Panel p = new JPanel();
CardLayout CL = new CardLayout();
p.setLayout (CL);
p.add(new JPanel(),"A");
p.add(new JPanel(),"B");
...
CL.next(p);
CL.previous(p);
```

Métodos:

CardLayout ();
add (Component comp, String Identif);
next (contenedor);
previous (contenedor)
show (contenedor, string nombre);

5. CardLayout (cont.)

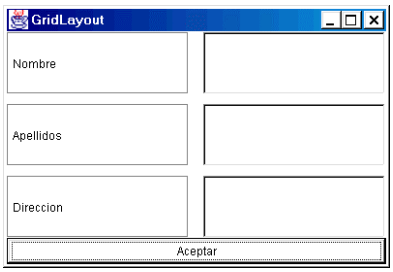





```
cardLayout1.previous(escenario);
cardLayout1.next(escenario);
cardLayout1.show(escenario,"donald");
```

6. GridLayout

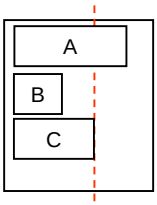
- Objetos ubicados en array (filas×cols)
- Las celdas poseen el mismo tamaño
- Controlar el espaciado (vgap, hgap)
- Se expanden para cubrir todo el área

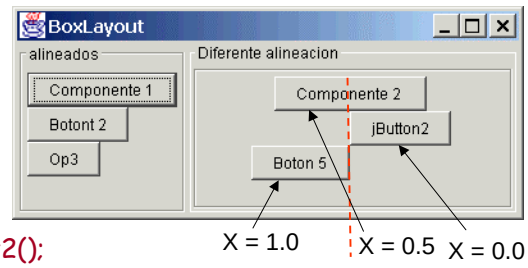



```
Panel p = new Panel();
GridLayout cg = new GridLayout(3,2);  // indicación de filas x col
cg.setHgap(15);                       // separación horizontal entre celdas
cg.setVgap(10);                       // separación vertical entre celdas
p.add(new Label("Nombre"));           // Ubicación por orden (fila 0 col 0)
p.add(new JTextField());              // Ubicación (fila 0 col 1)
```


7. BoxLayout

- Distribuye componentes de arriba-abajo (filas) o izda.-dcha (columnas)
- Esta distribución preserva alineamientos y los tamaños de cada componente
- Alineación: Eje X (X_AXIS), Eje Y (Y_AXIS)
- Alineación componentes: *buttonx*.**setAlignmentX/Y**(float) [0..1]






```
Panel p = new JPanel();
BoxLayout2 box = new BoxLayout2();
p.setLayout(box);
box.setAxis(BoxLayout.Y_AXIS);
boton5.setAlignmentX(1.0);
p.add(boton, null);
```

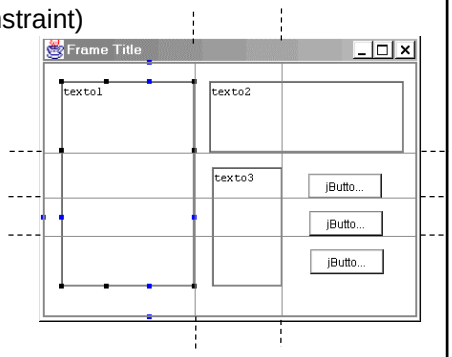
8. GridBagLayout

Distribución reticular de tamaño y ajuste variable

- Uso Complejo pero más flexible
- Cada celda posee restricciones (GridBagConstraint)

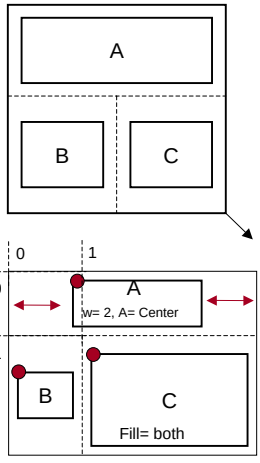
GridBagConstraints

```
(int gridx, int gridy,   // posicion de la celda
 int gridwidth, int gridheight, // ancho x alto
  double weightx,  // porcentaje de peso en x
  double weighty, // porcentaje de peso en x
  int anchor,     // Anclaje en la celda
  int fill,           // Relleno (orientación)
  Insets insets,  // Encuadre: distancia bordes celda
  int ipadx, int ipady)  // espacio extra del componente
```




```
Panel p = new Panel();
GridBagLayout cg = new GridBagLayout();
p.setLayout(gridBagLayout1)
restr = new GridBagConstraints  (0, 0, 1, 4, 1.0, 1.0
  ,GridBagConstraints.CENTER, GridBagConstraints.BOTH,
   new Insets(20, 19, 32, 0), 66, 194)
p. add(jTextArea1, restr);
```


8. GridBagLayout (cont.)



Posición
Colocación en fila x columna

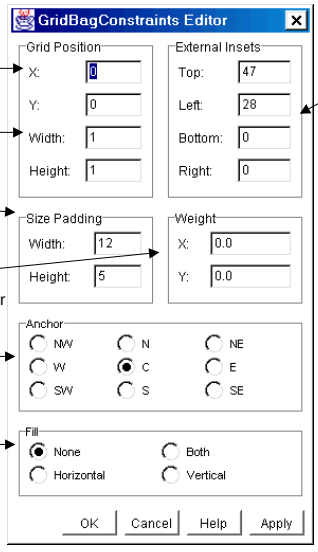
Tamaño
Nº de celdas de ocupa (en hor/vert)

Espacio extra
Espacio adicional para el objeto

Peso
% de espacio que se distribuye alrededor de la celda en caso de redimension

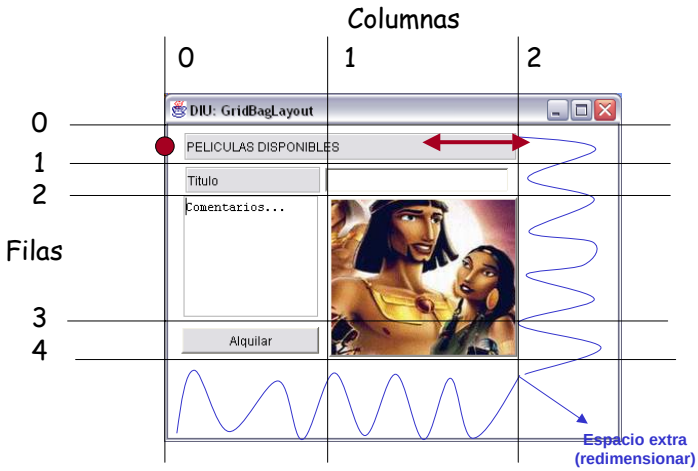
Alineación
Colocación del objeto (dentro de la celda) en caso de redimensionamiento

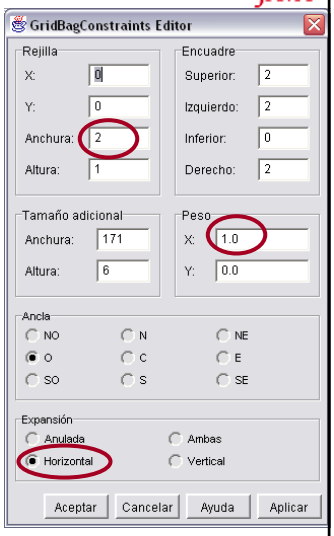
Ajuste
Expansión del objeto (en la celda) en caso de redimensionamiento



Encuadre
Distancia Bordes celda

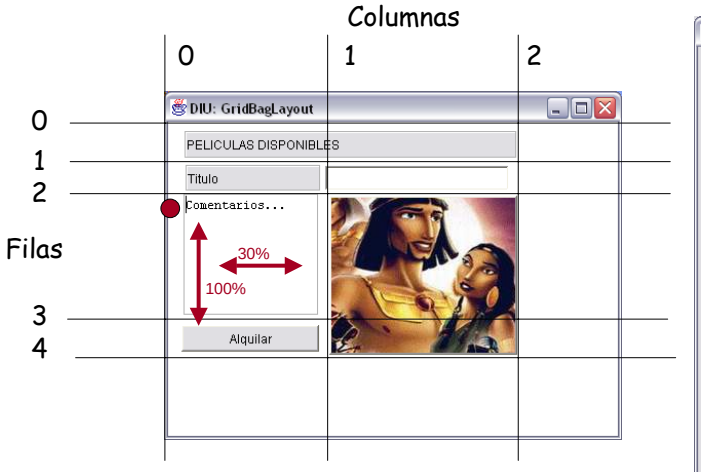
8. GridBagLayout (cont.)





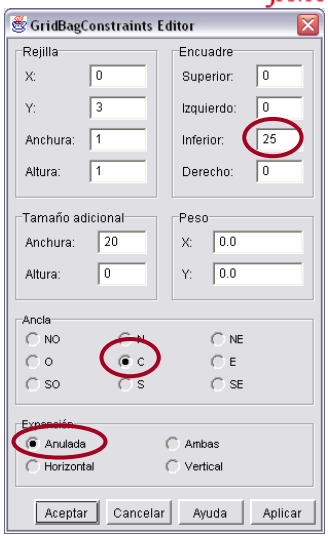
● Anclaje
↔ Expansión

8. GridBagLayout (cont.)



● Anclaje
↔ Expansión

8. GridBagLayout (cont.)



9. JTabbedPane

- Permite tener varios componentes (JPanel) compartiendo el mismo espacio.
- Se puede seleccionar el componente mediante pestañas
- Permite controlar cambios dinámicos (insertar/eliminar pestañas)
- Métodos:

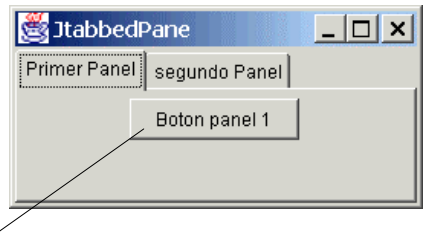

```
addTab(String, Icon, Component)
addTab(String, Component)
remove(Component)
removeTabAt(int)
removeAll()
insertTab(String, Icon, Component, String, int)
setSelectedComponent (Component);  // mostrar panel
setTabPlacement (JTabbedPane.BOTTOM);
  // Ubicación pestaña, posibilidades JTabbedPane.[TOP|BOTTOM|LEFT|RIGHT]
```

(ej. Ventanas.prj)

9. JTabbedPane (cont.)

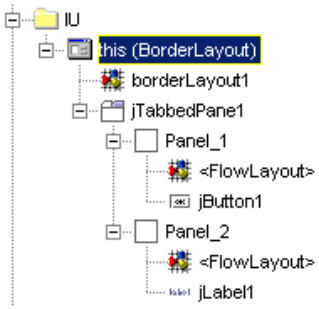
Ejemplo :

```
JTabbedPane p = new JTabbedPane();
p.addTab("Primer Panel",Panel_1);
p.addTab("segundo Panel", Panel_2);
Panel_2.add(boton);
```



actionPerformed

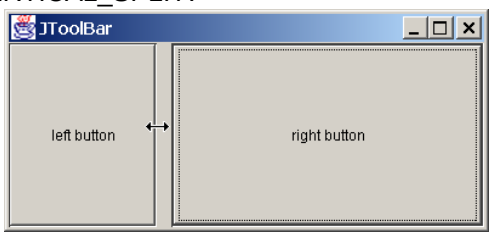
```
p.setSelectedComponent (jPanel2)
```



10. JSplitPane

- Contiene dos componentes (JPanel) separados por un divisor
- Se puede cambiar el tamaño relativo de cada área
- Orientación: HORIZONTAL_SPLIT, VERTICAL_SPLIT.
- Métodos:

JSplitPane(int);
void setDividerSize(int) // tamaño separador
void setDividerLocation(int) // ubicación
void setTopComponent(Component)
void setBottomComponent(Component)
void setLeftComponent(Component)
void setRightComponent(Component)
void add(Component) // sólo dos componentes




```
p = new JSplitPane(JsplitPane.HORIZONTAL_SPLIT);
p.setDividerSize(15);
p.setDividerLocation(250);
split.setRightComponent(new jTree());
```