Tema 6
Swing: Layouts

Proyecto de Análisis y Diseño de Software

2º Ingeniería Informática

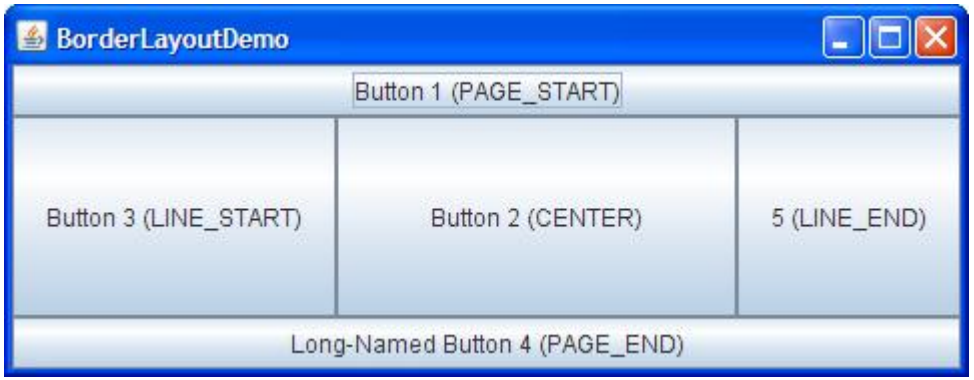
Universidad Autónoma de Madrid

Layout

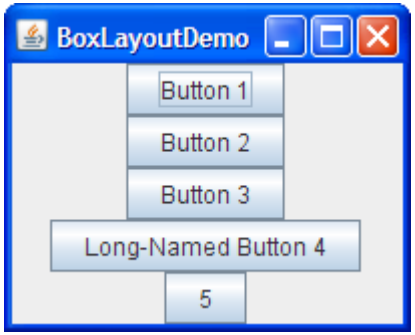
- Son estrategias para la (re-)colocación de componentes en un contenedor.
- Cada estrategia la implementa un gestor de layout, que se asigna al contenedor.
- El gestor se encarga de colocar los componentes del contenedor de acuerdo a la estrategia.
- Es posible anidar componentes con distintos layouts.

Ejemplos de Layouts

BorderLayout: Los elementos se pueden colocar en Norte, Sur, Este, Oeste o Centro.

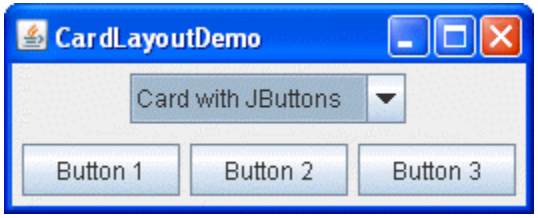


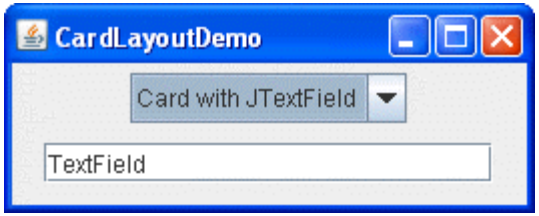
BoxLayout: Pone los componentes en una única fila o columna.



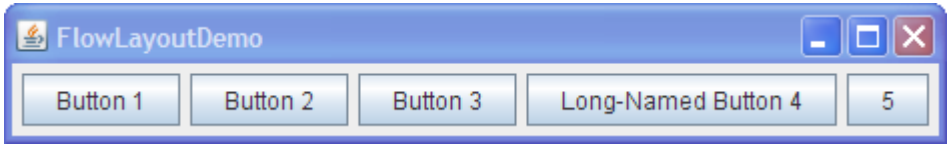
Ejemplos de Layouts

CardLayout: Permite mostrar diversos componentes en distintos instantes de tiempo. También se puede hacer con pestañas (JTabbedPane).



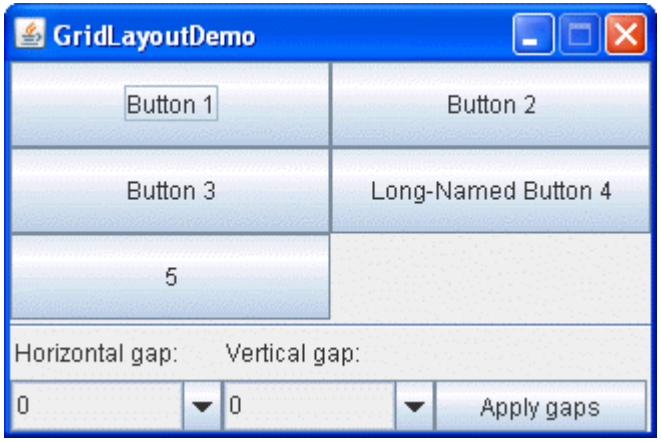


FlowLayout: Pone los elementos en una fila, y empieza otra si no caben. Es el layout por defecto de JPanel.

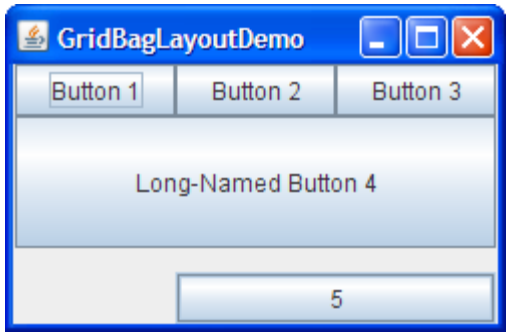


Ejemplos de Layouts

GridLayout: Pone los elementos en una rejilla.

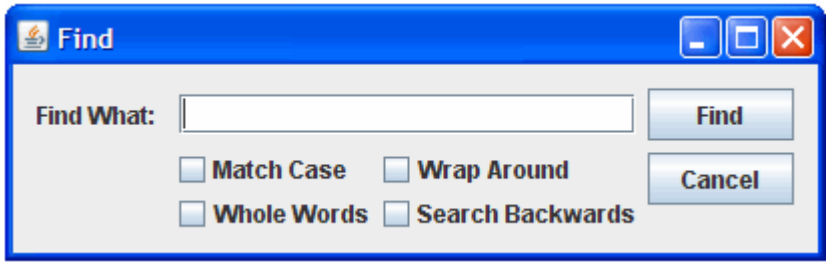


GridBagLayout: Pone los elementos en una rejilla, cada celda puede tener tamaño distinto, y un elemento puede ocupar más de una celda.

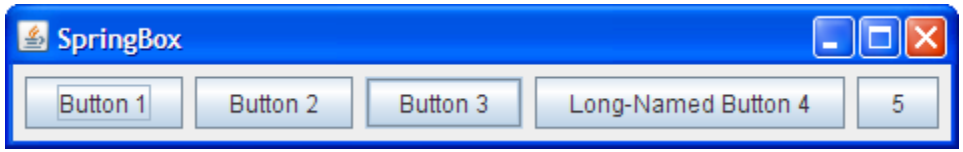


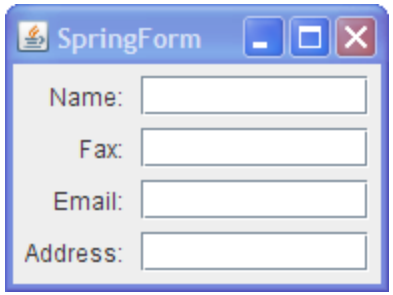
Ejemplos de Layouts

GroupLayout: Layout más sofisticado que requiere la especificación de la colocación horizontal y vertical de cada componente.



SpringLayout: Permite definir restricciones sobre la distancia entre los bordes de cada componente.





Pasos

1. Establecer el gestor de Layout.
2. Añadir los componentes al contenedor.
3. Establecer tamaños mínimo, máximo y preferidos de los componentes (opcional).
   - Muchos gestores de layout no los usan, pero sí: BoxLayout, SpringLayout y GroupLayout
4. Definir el espacio entre los componentes (opcional).
5. Establecer la orientación del contenedor (opcional).

```
public JPanel construir() {
   // Estos son los elementos que queremos
   // colocar en el contenedor
   JButton buttons[]={ new JButton("Mesa 0"),
                       new JButton("Mesa 1"),
                       new JButton("Mesa 2"),
                       new JButton("Mesa 3"),
                       new JButton("Mesa 4")
   };

   // Contenedor donde colocaremos los botones
   JPanel p = new JPanel();
   // Establecemos un layout de rejilla
   p.setLayout(new GridLayout(0,2));
   // Añadir cada botón al contenedor
   for (JButton j : buttons) {
     p.add(j); // Sin parametros, se añaden
   }           // por orden
   // Cambiamos el orden por defecto
   p.applyComponentOrientation(
    ComponentOrientation.RIGHT_TO_LEFT);

   return p;
}
```

Pasos

1. Establecer el gestor de Layout.
2. Añadir los componentes al contenedor.
3. Establecer tamaños mínimo, máximo y preferidos de los componentes (opcional).
   - Muchos gestores de layout no los usan, pero sí: BoxLayout, SpringLayout y GroupLayout
4. Definir el espacio entre los componentes (opcional).
5. Establecer la orientación del contenedor (opcional).

```
public JPanel construir() {
   // Estos son los elementos que queremos
   // colocar en el contenedor
   JButton buttons[]={ new JButton("Mesa 0"),
                       new JButton("Mesa 1"),
                       new JButton("Mesa 2"),
                       new JButton("Mesa 3"),
                       new JButton("Mesa 4")
   };

   // Contenedor donde colocaremos los botones
   JPanel p = new JPanel();
   // Establecemos un layout de rejilla
   p.setLayout(new GridLayout(0,2));
   // Añadir cada botón al contenedor
   for (JButton j : buttons) {
     p.add(j); // Sin parametros, se añaden
   }           // por orden
   // Cambiamos el orden por defecto
   p.applyComponentOrientation(
    ComponentOrientation.RIGHT_TO_LEFT);

   return p;
}
```

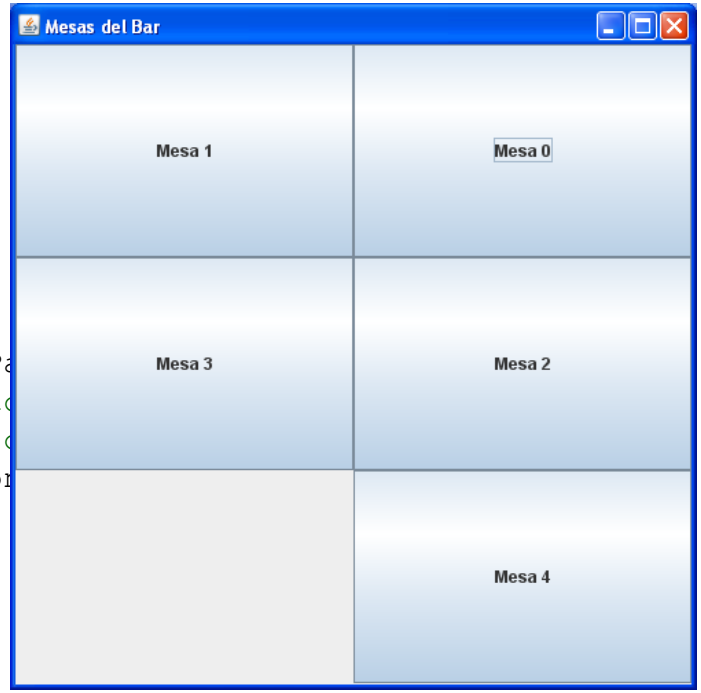
Pasos

1. Establecer el gestor de Layout.
2. Añadir los componentes al contenedor.
3. Establecer tamaños mínimo, máximo y preferidos de los componentes (opcional).
   - Muchos gestores de layout no los usan, pero sí: BoxLayout, SpringLayout y GroupLayout
4. Definir el espacio entre los componentes (opcional).
5. Establecer la orientación del contenedor (opcional).

```
public JPanel construir() {
   // Estos son los elementos que queremos
   // colocar en el contenedor
   JButton buttons[]={ new JButton("Mesa 0"),
                       new JButton("Mesa 1"),
                       new JButton("Mesa 2"),
                       new JButton("Mesa 3"),
                       new JButton("Mesa 4")
   };

   // Contenedor donde colocaremos los botones
   JPanel p = new JPanel();
   // Establecemos un layout de rejilla
   p.setLayout(new GridLayout(0,2));
   // Añadir cada botón al contenedor
   for (JButton j : buttons) {
     p.add(j); // Sin parametros, se añaden
   }           // por orden
   // Cambiamos el orden por defecto
   p.applyComponentOrientation(
    ComponentOrientation.RIGHT_TO_LEFT);

   return p;
}
```

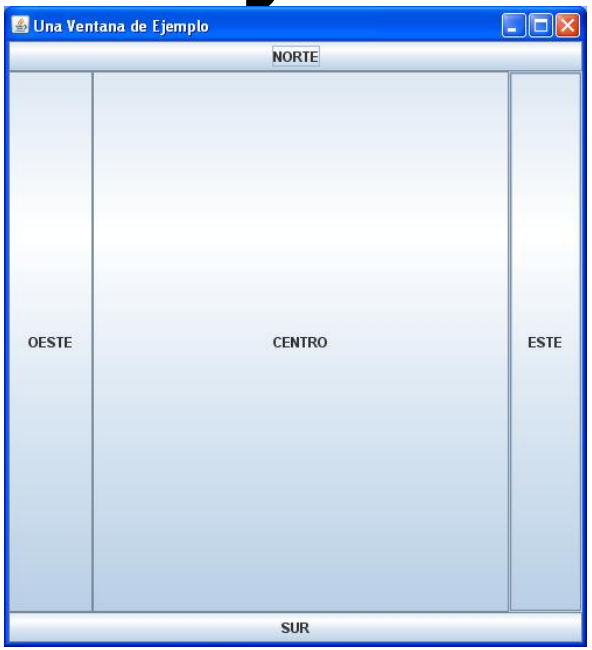
Pasos

1. Establecer el gestor de Layout.
2. Añadir los componentes al contenedor.
3. Establecer tamaños mínimo, máximo y preferidos de los componentes (opcional).
   - Muchos gestores de layout no los usan, pero sí: BoxLayout, SpringLayout y GroupLayout
4. Definir el espacio entre los componentes (opcional).
5. Establecer la orientación del contenedor (opcional).




```
public JPa
    // Esto
    // colo
    JButton
    };

    // Contenedor donde colocaremos los botones
    JPanel p = new JPanel();
    // Establecemos un layout de rejilla
    p.setLayout(new GridLayout(0,2));
    // Añadir cada botón al contenedor
    for (JButton j : buttons) {
      p.add(j); // Sin parametros, se añaden
    }           // por orden
    // Cambiamos el orden por defecto
    p.applyComponentOrientation(
     ComponentOrientation.RIGHT_TO_LEFT);

    return p;
}
```

Otro Ejemplo: Border Layout

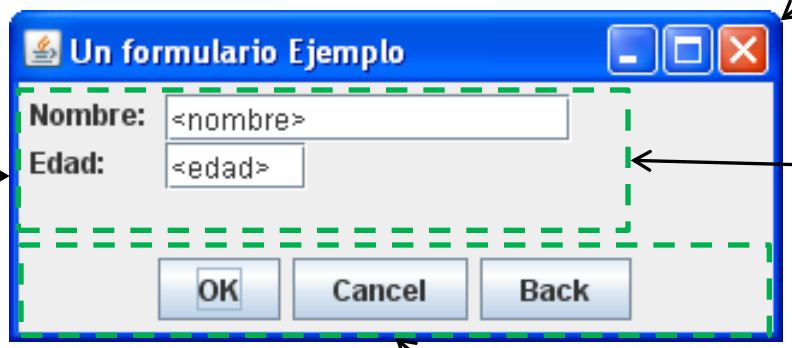



```
JPanel jp = new JPanel();
// Añadimos un layout de borde
jp.setLayout(new BorderLayout());

// Creamos los elementos a incluir en el panel
JButton button1    = new JButton("NORTE");
JButton button2    = new JButton("SUR");
JButton button3    = new JButton("ESTE");
JButton button4    = new JButton("OESTE");
JButton button5    = new JButton("CENTRO");

// Añadimos los elementos
jp.add(button1,  BorderLayout.NORTH);  // 2° par=posición
jp.add(button2,  BorderLayout.SOUTH);
jp.add(button3,  BorderLayout.EAST);
jp.add(button4,  BorderLayout.WEST);
jp.add(button5,  BorderLayout.CENTER);
```

Todos los componentes (en particular el del centro) se expandirán hasta ocupar todo el espacio disponible en el contenedor.
Podemos usar componentes anidados, con layouts distintos.

Componentes anidados

JFrame: BorderLayout



JPanel con SpringLayout, colocado en BorderLayout.CENTER del JFrame.

JPanel con FlowLayout, colocado en BorderLayout.SOUTH del JFrame.

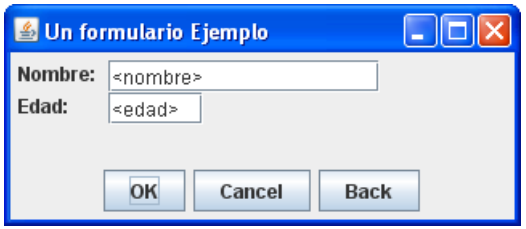
Si no añadimos componentes en norte, este u oeste, el panel no reserva espacio para ellos.

Componentes anidados (1/2)

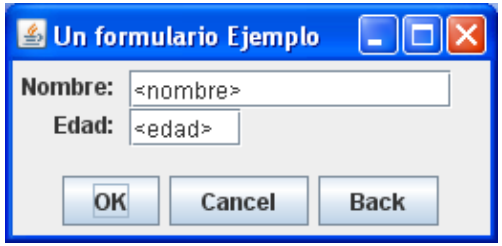
```
class VentanaFormulario extends JFrame {
  private JPanel botonera       = new JPanel();
  private Formulario formulario = new Formulario(); // Hereda de JPanel
  private JButton ok            = new JButton("OK");
  private JButton cancel        = new JButton("Cancel");
  private JButton back          = new JButton("Back");

  public VentanaFormulario() {
    super("Un formulario Ejemplo");
    Container cp = this.getContentPane();      // Obtener el contenedor del Frame
    cp.setLayout(new BorderLayout());          // Le ponemos un layout de borde

    // La botonera (JPanel) tiene por defecto FlowLayout, así que no hacemos nada
    botonera.add(ok);       // En el flowlayout, los componentes se añaden en orden,...
    botonera.add(cancel);   // por defecto de izquierda a derecha...
    botonera.add(back);     // si no hubiera espacio, se ponen en varias filas

    cp.add(botonera, BorderLayout.SOUTH);      // Colocamos la botonera al sur
    cp.add(formulario, BorderLayout.CENTER);   // El formulario irá en el centro

    this.pack();            // Importante: hace que los subcomponentes se coloquen…
                            // de acuerdo al layout y con sus tamaños preferidos.
    this.setVisible(true);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

Componentes anidados (2/2)

```
class Formulario extends JPanel {
  private JLabel label, label2;
  private JTextField field, field2;

  public Formulario() {
    SpringLayout layout = new SpringLayout();          // Layout basado en restricciones...
    this.setLayout(layout);                            // muy flexible, pero de bajo nivel.

    // Componentes a colocar...
    label  = new JLabel("Nombre: ");
    field  = new JTextField("<nombre>", 15);
    label2 = new JLabel("Edad: ");
    field2 = new JTextField("<edad>", 5);

    // La izquierda (WEST) de label estará a 5 pixels de la izquierda del contenedor
    layout.putConstraint(SpringLayout.WEST, label, 5, SpringLayout.WEST, this);
    // El norte (NORTH) de label estará a 5 pixels del norte del contenedor
    layout.putConstraint(SpringLayout.NORTH, label, 5, SpringLayout.NORTH, this);

    // La izquierda de field estará a 5 pixels desde el borde derecho (EAST) de label
    layout.putConstraint(SpringLayout.WEST, field, 5, SpringLayout.EAST, label);
    // El norte de field estará a 5 pixels desde el norte del contenedor
    layout.putConstraint(SpringLayout.NORTH, field, 5, SpringLayout.NORTH, this);

    // La izquierda de label2 estará a 0 pixels (alineada) del borde izquierdo de label
    layout.putConstraint(SpringLayout.WEST, label2, 0, SpringLayout.WEST, label);
    // El norte de label2 estará a 5 pixels del borde inferior (SOUTH) de label
    layout.putConstraint(SpringLayout.NORTH, label2, 8, SpringLayout.SOUTH, label);

    // La izquierda de field2 alienada con la izquierda de field
    layout.putConstraint(SpringLayout.WEST, field2, 0, SpringLayout.WEST, field);
    // El norte de field2, 5 pixels más abajo de field
    layout.putConstraint(SpringLayout.NORTH, field2, 5, SpringLayout.SOUTH, field);

    this.add(label); this.add(field);
    this.add(label2); this.add(field2);
    this.setPreferredSize(new Dimension(250,50));      // importante: tamaño preferido de este panel
    this.setVisible(true);  }
}
```

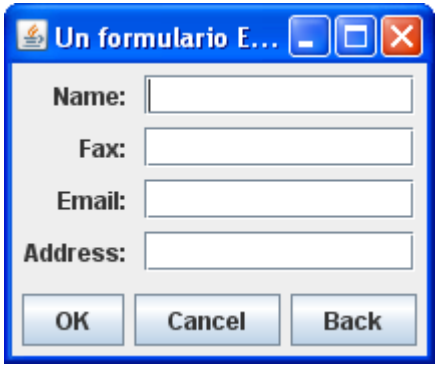


```
class Formulario extends JPanel {
  private JLabel label, label2;
  private JTextField field, field2;

  public Formulario() {
    SpringLayout layout = new SpringLayout();          // Layout basado en restricciones...
    this.setLayout(layout);                            // muy flexible, pero de bajo nivel.

    // Componentes a colocar...
    label  = new JLabel("Nombre: ");
    field  = new JTextField("<nombre>", 15);
    label2 = new JLabel("Edad: ");
    field2 = new JTextField("<edad>", 5);

    // La izquierda (WEST) de label estará a 5 pixels de la izquierda del contenedor
    layout.putConstraint(SpringLayout.WEST, label, 5, SpringLayout.WEST, this);
    // El norte (NORTH) de label estará a 5 pixels del norte del contenedor
    layout.putConstraint(SpringLayout.NORTH, label, 5, SpringLayout.NORTH, this);

    // La izquierda de field estará a 5 pixels desde el borde derecho (EAST) de label
    layout.putConstraint(SpringLayout.WEST, field, 5, SpringLayout.EAST, label);
    // El norte de field estará a 5 pixels desde el norte del contenedor
    layout.putConstraint(SpringLayout.NORTH, field, 5, SpringLayout.NORTH, this);

    // La derecha de label2 estará a 0 pixels (alineada) del borde derecho de label
    layout.putConstraint(SpringLayout.EAST, label2, 0, SpringLayout.EAST, label);
    // El norte de label2 estará a 5 pixels del borde inferior (SOUTH) de label
    layout.putConstraint(SpringLayout.NORTH, label2, 8, SpringLayout.SOUTH, label);

    // La izquierda de field2 alienada con la izquierda de field
    layout.putConstraint(SpringLayout.WEST, field2, 0, SpringLayout.WEST, field);
    // El norte de field2, 5 pixels más abajo de field
    layout.putConstraint(SpringLayout.NORTH, field2, 5, SpringLayout.SOUTH, field);

    this.add(label); this.add(field);
    this.add(label2); this.add(field2);
    this.setPreferredSize(new Dimension(250,50));      // importante: tamaño preferido de este panel
    this.setVisible(true);  }
}
```



Alinear componentes

```
public Formulario() {
  SpringLayout layout = new SpringLayout();
  this.setLayout(layout);

  // Etiquetas que vamos a poner
  String[] labels = {"Name: ", "Fax: ", "Email: ", "Address: "};
  int numPairs = labels.length;

  // Crear las etiquetas y campos de edición
  for (int i = 0; i < numPairs; i++) {
    JLabel l = new JLabel(labels[i], JLabel.TRAILING);    // 2° parám= alineación horizontal
    this.add(l);                                          // Lo añadimos sin ninguna restricción
    JTextField textField = new JTextField(10);            // 10=tamaño del campo en columnas
    l.setLabelFor(textField);                             // El componente que l etiqueta
    this.add(textField);                                  // Lo añadimos sin ninguna restricción
  }

  // Este método de utilidad se encarga de colocar los componentes
  SpringUtilities.makeCompactGrid(this, // Contenedor donde vamos a colocar los componentes
                                  numPairs, 2, // número de filas y cols
                                  6, 6,        // initX, initY
                                  6, 6);       // xPad, yPad (separación x e y)
  this.setVisible(true);
}
```



La clase SpringUtilities está:
http://download.oracle.com/javase/tutorial/uiswing/examples/layout/SpringGridProject/src/layout/SpringUtilities.java

¿Cómo pasar de un panel a otro en la misma ventana?

- Pestañas o bien usar *CardLayout*.

- Cada componente que es gestionado por un CardLayout es como una carta en una pila de cartas.
  - Sólo el componente que está más arriba en la pila es visible.

- Podemos cambiar el componente visible:
  - Mostrar el primero o el último de la pila.
  - Mostrar el siguiente o el anterior al actual.
  - Mostrar uno específico, que seleccionamos mediante un nombre (un String).

CardLayout

Funcionamiento. Paso 1.

```
// Declaramos el panel con las cartas
JPanel cartas;
final static String BUTTONPANEL = "Carta con JButtons";
final static String TEXTPANEL   = "Carta con JTextField";

// Creamos e inizializamos cada una de las cartas
JPanel card1 = new JPanel();
...
JPanel card2 = new JPanel();
...

// Creamos el panel que contiene las cartas
cards = new JPanel(new CardLayout());
cards.add(card1, BUTTONPANEL);
cards.add(card2, TEXTPANEL);
```

CardLayout

Funcionamiento. Paso 1.

```
// Declaramos el panel con las cartas
JPanel cartas;
final static String BUTTONPANEL = "Carta con JButtons";
final static String TEXTPANEL   = "Carta con JTextField";

// Creamos e inizializamos cada una de las cartas
JPanel card1 = new JPanel();
...
JPanel card2 = new JPanel();
...

// Creamos el panel que contiene las cartas
cards = new JPanel(new CardLayout());
cards.add(card1, BUTTONPANEL);
cards.add(card2, TEXTPANEL);
```

Panel que contendrá las cartas

Strings que identifican a cada una de las cartas

CardLayout

Funcionamiento. Paso 1.

```
// Declaramos el panel con las cartas
JPanel cartas;
final static String BUTTONPANEL = "Carta con JButtons";
final static String TEXTPANEL   = "Carta con JTextField";

// Creamos e inizializamos cada una de las cartas
JPanel card1 = new JPanel();
...
JPanel card2 = new JPanel();
...

// Creamos el panel que contiene las cartas
cartas = new JPanel(new CardLayout());
cartas.add(card1, BUTTONPANEL);
cartas.add(card2, TEXTPANEL);
```

Paneles con cada una de las cartas

añadimos dichas cartas al panel contenedor, con add, indicando su String identificativo

CardLayout

Funcionamiento. Paso 2.

```
JPanel comboBoxPane = new JPanel(); //usa FlowLayout por defecto
String comboBoxItems[] = { BUTTONPANEL, TEXTPANEL };

JComboBox cb = new JComboBox(comboBoxItems);    // Controlará qué carta se muestra

cb.setEditable(false);                          // Un combo sólo de selección
cb.addItemListener(this);                       // Se llamará a itemStateChanged al seleccionar
comboBoxPane.add(cb);                           // Añadimos el combo a un panel padre
...
pane.add(comboBoxPane, BorderLayout.NORTH);     // Mostramos el combo
pane.add(cartas, BorderLayout.CENTER);          // Mostramos el panel de las cartas
...



// Este método es necesario para la interfaz ItemListener,
// permite elegir qué panel se mostrará
public void itemStateChanged(ItemEvent evt) {
   CardLayout cl = (CardLayout)(cartas.getLayout());  // Obtener el layout de las cartas
   cl.show(cartas, (String)evt.getItem());    // muestra la carta correspondiente al
                                              // string elegido en el combo.
}
```

CardLayout

Funcionamiento. Paso 2.

Un ComboBox va a controlar qué panel se muestra:

- Creamos el combo, con items correspondientes a los Strings de las cartas.
- En itemStateChanged cambiamos la carta que se muestra

```
JPanel comboBoxPane = new JPanel(); //usa FlowLayo
String comboBoxItems[] = { BUTTONPANEL, TEXTPANEL

JComboBox cb = new JComboBox(comboBoxItems);    //

cb.setEditable(false);                          // Un combo sólo de selección
cb.addItemListener(this);                       // Se llamará a itemStateChanged al seleccionar
comboBoxPane.add(cb);                           // Añadimos el combo a un panel padre
...
pane.add(comboBoxPane, BorderLayout.NORTH);     // Mostramos el combo
pane.add(cartas, BorderLayout.CENTER);          // Mostramos el panel de las cartas
...
```

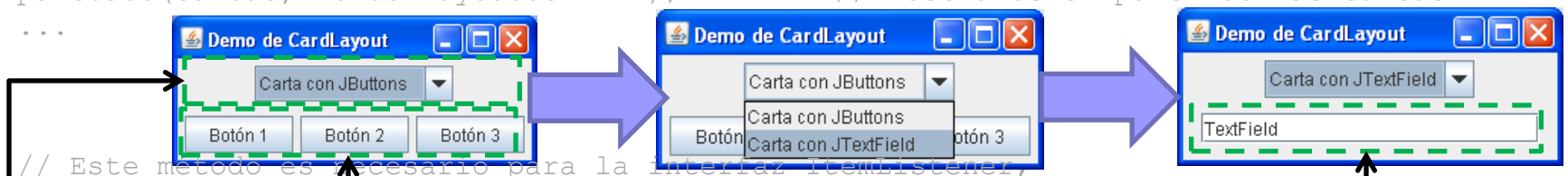

```
// Este método es necesario para la interfaz ItemListener,
// permite elegir qué panel se mostrará
public void itemStateChanged(ItemEvent evt) {
   CardLayout cl = (CardLayout)(cartas.getLayout());  // Obtener el layout de las cartas
   cl.show(cartas, (String)evt.getItem());    // muestra la carta correspondiente al
                                              // string elegido en el combo.
}
```

CardLayout

Funcionamiento. Paso 2.

Un ComboBox va a controlar qué panel se muestra:
- Creamos el combo, con items correspondientes a los Strings de las cartas.
- En itemStateChanged cambiamos la carta que se muestra

```
JPanel comboBoxPane = new JPanel(); //usa FlowLay
String comboBoxItems[] = { BUTTONPANEL, TEXTPANEL

JComboBox cb = new JComboBox(comboBoxItems);    //

cb.setEditable(false);                          // Un combo sólo de selección
cb.addItemListener(this);                       // Se llamará a itemStateChanged al seleccionar
comboBoxPane.add(cb);                           // Añadimos el combo a un panel padre
...
pane.add(comboBoxPane, BorderLayout.NORTH);     // Mostramos el combo
pane.add(cartas, BorderLayout.CENTER);          // Mostramos el panel de las cartas
...
```




```
// Este método es necesario para la interfaz ItemListener,
// permite elegir qué panel se mostrará
public void itemStateChanged(ItemEvent evt) {
    CardLayout cl = (CardLayout)(cartas.getLayout());  // Obtener el layout de las cartas
```

Panel con el combo (comboBoxPane)

Panel con las cartas (cartas). Se muestra la carta con id BUTTONPANEL

Panel con las cartas (cartas). Se muestra la carta con id TEXTPANEL

Referencias

- Tutorial de Swing:
  http://docs.oracle.com/javase/tutorial/uiswing/

- JavaDoc del API de Swing:
  http://download.oracle.com/javase/6/docs/api/javax/swing/package-summary.html

- Tutorial sobre layouts:
  http://download.oracle.com/javase/tutorial/uiswing/layout/index.html