Layouts

Vamos a analizar en detalle y con ejemplos algunos de ellos.

FlowLayout

Es el más simple y el que se utiliza por defecto en todos los paneles si no se fuerza el uso de alguno de los otros. Los componentes añadidos a un panel con **FlowLayout** se encadenan en forma de lista. La cadena es horizontal, de izquierda a derecha, y se puede seleccionar el espaciado entre cada componente.

Si el contenedor se cambia de tamaño en tiempo de ejecución, las posiciones de los componentes se ajustarán automáticamente, para colocar el máximo número posible de componentes en la primera línea.

Los componentes se alinean según se indique en el constructor. Si no se indica nada, se considera que los componentes que pueden estar en una misma línea estarán centrados, pero también se puede indicar que se alineen a izquierda o derecha en el contenedor.

Ejemplo 15: Para ilustrarlo mejor, vamos a crear un JFrame con 5 botones centrados y con una separación inicial entre ellos de 3px. Al hacer clic en cualquier botón, se incrementará dicha separación en 5px. Además, al redimensionar el JFrame veremos cómo se adaptarán los componentes al nuevo tamaño:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class Ejemplo15 extends JFrame implements ActionListener {
JButton boton1, boton2, boton3, boton4, boton5;
FlowLayout miFlowLayout;
public Ejemplo15() {
//Instancia un objeto FlowLayout object alineado al centro y con una separacion de 3px en horizonal y vertical
miFlowLayout = new FlowLayout(FlowLayout.CENTER,3,3);
//Uso este FlowLayout para que sea el controlador de posicionamiento de mi objeto JFrame
setLayout(miFlowLayout);
//Creo 5 botones y los añado a mi objeto JFrame
boton1 = new JButton("Botón 1");
boton2 = new JButton("Botón 2");
boton3 = new JButton("Botón 3");
boton4 = new JButton("Botón 4");
boton5 = new JButton("Botón 5");
add(boton1);
add(boton2);
```

```
add(boton3);
add(boton4);
add(boton5);
//Añado los botones al ActionListener
boton1.addActionListener(this);
boton2.addActionListener(this);
boton3.addActionListener(this);
boton4.addActionListener(this);
boton5.addActionListener(this);
//Configurar y mostrar JFrame
initPantalla();
}
private void initPantalla() {
setTitle("Ejemplo 15"); //Título del JFrame
setSize(250,150); //Dimensiones del JFrame
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Cerrar proceso al cerrar ventana
setVisible(true); //Mostrar JFrame
}
public static void main(String[] args) {
new Ejemplo15();
}
@Override
public void actionPerformed(ActionEvent e) {
//Al hacer clic en un botón, añado 5px de espacio horizontal y vertical entre botones
miFlowLayout.setHgap(miFlowLayout.getHgap() + 5);
miFlowLayout.setVgap(miFlowLayout.getVgap() + 5);
//Fijo el nuevo layout al formulario
setLayout(miFlowLayout);
//Valido el formulario para asegurarme de que se actualiza en pantalla
validate();
}
}
```


BorderLayout

La composición **BorderLayout** (de borde) proporciona un esquema más complejo de colocación de los componentes en un panel. Es el layout o composición que utilizan por defecto JFrame y JDialog. La composición utiliza cinco zonas para colocar los componentes sobre ellas:

- Norte: parte superior del panel (NORTH o PAGE_START)
- Sur: parte inferior del panel (SOUTH o PAGE_END)
- Este: parte derecha del panel (EAST o LINE_END)
- Oeste: parte izquierda del panel (WEST o LINE_START)
- Centro: parte central del panel, una vez que se hayan rellenado las cuatro partes (CENTER)

Este controlador de posicionamiento resuelve los problemas de cambio de plataforma de ejecución de la aplicación, pero limita el número de componentes que pueden ser colocados en contenedor a cinco; aunque, si se va a construir un interfaz gráfico complejo, algunos de estos cinco componentes pueden contenedores, con lo cual el número de componentes puede verse ampliado.

En los cuatro lados, los componentes se colocan y redimensionan de acuerdo a sus tamaños preferidos y a los valores de separación que se hayan fijado al contenedor.

El tamaño prefijado , el tamaño máximo y el tamaño mínimo (**setPreferredSize(new Dimension(WIDTH,HEIGHT))**, **setMaximumSize(new Dimension(WIDTH,HEIGHT))** y **setMinimumSize(new Dimension(WIDTH,HEIGHT))**) son informaciones muy importantes en este caso, ya que un botón o panel puede ser redimensionado a proporciones cualesquiera; sin embargo, el diseñador puede fijar un tamaño preferido para obtener una mejor visualización de cada componente.

Ejemplo 16: crea una ventana a través de un objeto **JFrame** y coloca cinco objetos **JButton**, sin funcionalidad alguna, utilizando un **BorderLayout** como manejador de composición y con una separación inicial entre ellos de 3px. Al hacer clic en cualquier botón, se incrementará dicha separación en 5px. Además, al redimensionar el JFrame veremos cómo se adaptarán los componentes al nuevo tamaño:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import static java.awt.BorderLayout.*;
public class Ejemplo16 extends JFrame implements ActionListener {
JButton boton1, boton2, boton3, boton4, boton5;
BorderLayout miBorderLayout;
public Ejemplo16() {
//Instancia un objeto BorderLayout con una separacion de 3px en horizonal y vertical
miBorderLayout = new BorderLayout(3,3);
//Uso este BorderLayout para que sea el controlador de posicionamiento de mi objeto JFrame
setLayout(miBorderLayout);
//Creo 5 botones y los añado a mi objeto JFrame
boton1 = new JButton("Sur");
boton2 = new JButton("Oeste");
boton3 = new JButton("Norte");
boton4 = new JButton("Este");
```

```
boton5 = new JButton("Centro");
add(boton1, SOUTH);
add(boton2, WEST);
add(boton3, NORTH);
add(boton4, EAST);
add(boton5, CENTER);
//Añado los botones al ActionListener
boton1.addActionListener(this);
boton2.addActionListener(this);
boton3.addActionListener(this);
boton4.addActionListener(this);
boton5.addActionListener(this);
//Configurar y mostrar JFrame
initPantalla();
}
private void initPantalla() {
setTitle("Ejemplo 16"); //Título del JFrame
setSize(250,150); //Dimensiones del JFrame
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Cerrar proceso al cerrar ventana
setVisible(true); //Mostrar JFrame
}
public static void main(String[] args) {
new Ejemplo16();
}
@Override
public void actionPerformed(ActionEvent e) {
//Al hacer clic en un botón, añado 5px de espacio horizontal y vertical entre botones
miBorderLayout.setHgap(miBorderLayout.getHgap() + 5);
miBorderLayout.setVgap(miBorderLayout.getVgap() + 5);
//Fijo el nuevo layout al formulario
setLayout(miBorderLayout);
//Valido el formulario para asegurarme de que se actualiza en pantalla
validate();
}
}
```


Se observará que se puede cambiar de tamaño el objeto JFrame, y que dentro de los límites, los componentes se van cambiado a la vez, para acomodarse al tamaño que va adoptando la ventana. Aunque todavía hay problemas que no se han eliminado, ya se podrá redimensionar la ventana para conseguir que los botones desaparezcan o se trunquen los rótulos. Sin embargo, es una forma muy

flexible de posicionar componentes en una ventana y no tener que preocuparse de su redimensionamiento y colocación (dentro de unos límites normales).

CardLayout

Este es el tipo de composición que se utiliza cuando se necesita una zona de la ventana que permita colocar distintos componentes en esa misma zona. Este *layout* suele ir asociado con botones de selección, de tal modo que cada selección determina el panel (grupo de componentes) que se presentarán.

Ejemplo 17: Crea una interfaz de usuario basado en un objeto JFrame, que contiene a los dos objetos JPanel que lo conforman. Uno de los **JPanel** en la parte superior permitirá alternar entre diferentes fichas (*cards*), utilizando el **CardLayout**. El otro **JPanel** mostrará la ficha correspondiente con varios objetos **JButton**.

```
import java.awt.BorderLayout;
import java.awt.CardLayout;
import java.awt.Color;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
public class Ejemplo17 extends JFrame {
CardLayout tarjetas;
JPanel panelTarjetas;
public Ejemplo17() {
//Centro el JFrame en la pantalla
setLocationRelativeTo(null);
//Inicializo un border layout para el JFrame
setLayout(new BorderLayout());
//Creo un panel para el botón superior con fondo rojo
JPanel panelSuperior = new JPanel();
panelSuperior.setBackground(Color.RED);
//Añado un botón para intercambiar tarjetas y le añado el action listener
JButton cambiarTarjeta = new JButton("Cambiar tarjeta");
cambiarTarjeta.addActionListener(new ActionListener(){
@Override
public void actionPerformed(ActionEvent event){
tarjetas.next(panelTarjetas);
}
});
//Añado el botón al panel de pestañas
```

```
panelSuperior.add(cambiarTarjeta);
//Añado el panel de pestañas a la parte superior del JFrame
add(panelSuperior,BorderLayout.NORTH);
//Inicializo el layout y el panel para las tarjetas
tarjetas = new CardLayout();
panelTarjetas = new JPanel();
panelTarjetas.setLayout(tarjetas);
//Inicializo 2 tarjetas, cada una es un JPanel con un color de fondo
JPanel primeraTarjeta = new JPanel();
JPanel segundaTarjeta = new JPanel();
primeraTarjeta.setBackground(Color.GREEN);
segundaTarjeta.setBackground(Color.BLUE);
//Añado botones a las dos tarjetas
nuevoBoton(primeraTarjeta, "Manzanas");
nuevoBoton(primeraTarjeta, "Naranjas");
nuevoBoton(primeraTarjeta, "Plátanos");
nuevoBoton(segundaTarjeta, "Lechugas");
nuevoBoton(segundaTarjeta, "Tomates");
nuevoBoton(segundaTarjeta, "Cebollas");
//Añado las dos tarjetas al panel de tarjetas
panelTarjetas.add(primeraTarjeta, "Frutas");
panelTarjetas.add(segundaTarjeta, "Verduras");
//Muestro la primera
tarjetas.show(panelTarjetas, "Frutas");
//Añado el panel de tarjetas a la parte central del border layout
add(panelTarjetas,BorderLayout.CENTER);
//Configurar y mostrar JFrame
initPantalla();
}
private void initPantalla() {
setTitle("Ejemplo 17"); //Título del JFrame
setSize(400,300); //Dimensiones del JFrame
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Cerrar proceso al cerrar
ventana
setVisible(true); //Mostrar JFrame
}
//Nuevo botón
private void nuevoBoton(JPanel panel, String titulo){
JButton boton = new JButton(titulo);
```

```
panel.add(boton);
}
public static void main(String[] args) {
new Ejemplo17();
}
}
```
view raw java-swing-026.java hosted with ❤ by GitHub

GridLayout

La composición **GridLayout** proporciona gran flexibilidad para situar componentes. El controlador de posicionamiento se crea con un determinado número de filas y columnas y los componentes van dentro de las celdas de la tabla así definida.

Si el contenedor es alterado en su tamaño en tiempo de ejecución, el sistema intentará mantener el mismo número de filas y columnas dentro de los márgenes de separación que se hayan indicado. En este caso, estos márgenes tienen prioridad sobre el tamaño mínimo que se haya indicado para los componentes, por lo que puede llegar a conseguirse que sean de un tamaño tan pequeño que sus etiquetas sean ilegibles.

El siguiente ejemplo muestra la Calculadora desarrollada anteriormente, pero en este caso utilizando **FlowLayout** para display y botón de resultado, y **GridLayout** para los 16 botones centrales.

```
import java.awt.*;
import static java.awt.Font.PLAIN;
import javax.swing.JButton;
import javax.swing.JFrame;
import static javax.swing.JFrame.EXIT_ON_CLOSE;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.SwingConstants;
import javax.swing.border.LineBorder;
public class Ejemplo18 extends JFrame {
JButton boton[], botonResultado; //array de botones y botón de resultado
JLabel display; //display de operaciones
JPanel panelBotones; //panel para los botones (salvo el de resultado)
int ancho = 50, alto = 50; //para posicionar botones
FlowLayout miFlowLayout;
GridLayout miGridLayout;
public Ejemplo18() {
initDisplay();
initBotones();
initBotonResultado();
initPantalla();
}
```

```
private void initDisplay(){
display = new JLabel("0");
display.setPreferredSize(new Dimension(230,60));
display.setOpaque(true);
display.setBackground(Color.black);
display.setForeground(Color.green);
display.setBorder(new LineBorder(Color.DARK_GRAY));
display.setFont(new Font("MONOSPACED",PLAIN,24));
display.setHorizontalAlignment(SwingConstants.RIGHT);
add(display);
}
private void initBotones(){
//Inicializo panel de botones y su gridlayout de 4 columnas y 4 filas
panelBotones = new JPanel();
panelBotones.setBackground(Color.black);
miGridLayout = new GridLayout(4,4,10,10);
panelBotones.setLayout(miGridLayout);
add(panelBotones);
//Array de textos de los botones
String[] texto_boton = new String[]{"0",".","C","+","1","2","3","-
","4","5","6","*","7","8","9","/"};
//Inicializo array de botones
boton = new JButton[16];
for (int i=0; i<=15; i++){
//Inicializo botón con su texto
boton[i] = new JButton(texto_boton[i]);
boton[i].setPreferredSize(new Dimension(ancho,alto));
boton[i].setFont(new Font("MONOSPACED",PLAIN,16));
boton[i].setOpaque(true);
boton[i].setFocusPainted(false);
boton[i].setBackground(Color.DARK_GRAY);
boton[i].setBorder(new LineBorder(Color.DARK_GRAY));
boton[i].setForeground(Color.WHITE);
//añado botón al panel de botones
panelBotones.add(boton[i]);
}
}
private void initBotonResultado(){
botonResultado = new JButton("RESULTADO");
botonResultado.setPreferredSize(new Dimension(230,alto));
```

```
botonResultado.setFont(new Font("MONOSPACED",PLAIN,16));
botonResultado.setOpaque(true);
botonResultado.setFocusPainted(false);
botonResultado.setBackground(Color.DARK_GRAY);
botonResultado.setBorder(new LineBorder(Color.DARK_GRAY));
botonResultado.setForeground(Color.WHITE);
add(botonResultado);
}
private void initPantalla(){
miFlowLayout = new FlowLayout(FlowLayout.CENTER,10,10);
setLayout(miFlowLayout);
setTitle("Ejemplo 18");
setMinimumSize(new Dimension(255,415));
setResizable(false);
getContentPane().setBackground(Color.black);
setDefaultCloseOperation(EXIT_ON_CLOSE);
setVisible(true);
}
public static void main(String[] args) {
new Ejemplo18();
}
}
```

view raw java-swing-027.java hosted with ❤ by GitHub

GridBagLayout

Es igual que la composición de **GridLayout**, con la diferencia que los componentes no necesitan tener el mismo tamaño. Es quizá el controlador de posicionamiento más sofisticado de los que actualmente soporta AWT.

Dada la complejidad de trabajar con este *layout* y que en posteriores capítulos se van a utilizar librerías más modernas para la creación de interfaces, no se van a implementar ejemplos de **GridBagLayout**, prefiriendo para ganar en sencillez con la combinación de *layouts* y paneles (como en el caso anterior).

BoxLayout

El controlador de posicionamiento **BoxLayout** es uno de los dos que incorpora Java a través de Swing y permite colocar los componentes a lo largo del eje X o del eje Y. También posibilita que los componentes ocupen diferente espacio a lo largo del eje principal.

En un controlador **BoxLayout** sobre el eje Y, los componentes se posicionan de arriba hacia abajo

en el orden en que se han añadido. Al contrario en en el caso del **GridLayout**, aquí se permite que los componentes sean de diferente tamaño a lo largo del eje Y, que es el eje principal del controlador de posicionamiento, en este caso.

En el eje que no es principal, **BoxLayout** intenta que todos los componentes sean tan anchos como el más ancho, o tan alto como el más alto, dependiendo de cuál sea el eje principal. Si un componente no puede incrementar su tamaño, el **BoxLayout** mira las propiedades de alineamiento en X e Y para determinar donde colocarlo.

OverlayLayout

El controlador de posicionamiento **OverlayLayout**, es el otro que se incopora a Java con *Swing*, y es un poco diferente a todos los demás. Se dimensiona para contener el más grande de los componentes y superpone cada componente sobre los otros.

La clase **OverlayLayout** no tiene un constructor por defecto, así que hay que crearlo dinámicamente en tiempo de ejecución.

No obstante, en la bibliografía adjunta existen ejemplos y especificaciones de **GridBagLayout** que permitirían, por ejemplo, que la calculadora se hiciera con un único *layout*, así como de **BoxLayout** y **OverlayLayout**.

Igualmente y para finalizar el capítulo, se ofrece el siguiente **ENLACE** por si el alumno está interesado en crear su propio layout (**CustomLayout**).