# Objetos java swing

completar apuntes cos objetos `JComponent` mais utilizados...

## Bordes nos objetos

A practicamente todos os objetos que heredan de JComponent se lle pode aplicar un borde, gracias a función `.setBorder()`

\* Pasaremoslle un objeto `MatteBorder`, que nos permite indicar un grosor distinto de borde pa cada lado.

\* `new MatteBorder(arriba,izquierda,abaixo,dereita,[Color | Imagen]);`

````java

import javax.swing.\*;

import javax.swing.border.MatteBorder;

import java.awt.\*;

JPanel panel = new JPanel();

panel.setBorder(new MatteBorder(0,10,0,10,Color.RED));

````

# Ventanas e paneles

## Ventanas

Para ventanas usaremos `JFrame`

````java

import javax.swing.\*;

JFrame frame=new JFrame();

frame.setTitle("Titulo da ventana");

frame.setSize(800,600); //tamaño da ventana en px

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); //accion ao cerrar a ventana

frame.setLayout(new GridBagLayout); //tipo de layout da ventana

````

## Paneles

Para paneles usaremos `JPanel`, nos que meteremos os elementos.

\* Cada panel pode ter un \*\*layout\*\* distinto

````java

JPanel panel = new JPanel();

panel.setBackground("#FFFF"); //ponlle color de fondo

panel.setLayout(new BorderLayout()); //cambia o layout

panel.add(elementoJComponent); //añade o elemento ao panel (os argumentos varian dependendo do layout)

````

# Layouts

Podese aplicar distintos layouts a os objectos que eredan de `Component`, como JFrame ou JPanel.

## GridBagLayout

Distribue os compoñentes nunha cuadricula, pero as celdas da cuadricula poden cambiar de tamaño flexiblemente, ademais que se poden agrandar as columna e filas.

Polo que vin se sabes usar este ben podes aplicalo a practicamente todo.

Usa a clase `GridBagConstraints` para indicar a configuración de posicion dos elementos. Indicar que `se pode usar o mismo objeto sempre`, xa que

cando se aplica a un componente, o layout toma unha instantanea dos valores do obxeto nese momento e desvinculanse desa instancia en especifico.

Esto polo que vin tamen aplica a todas as clases usadas nos atributos do GridBagConstraints, ainda que se modifique a clase despois non afecta se xa se fixo o .add()

````java

import javax.swing.\*;

import java.awt.\*;

JPanel panel=new JPanel();

panel.setLayout(new GridBagLayout());

JLabel elemento=new JLabel("label");

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx=0; //indica a columna na que estará o elemento

gbc.gridy=1; //indica a fila na que estará o elemento

gbc.gridwidth=1; //indica cantas columnas ocupa o elemento

gbc.gridheight=2; //indica cantas filas ocupa o elemento

gbc.weightx=1; //peso que terá o compoñente horizontalmente a hora de ocupar espazo dispoñible

gbc.weighty=1; //o mismo pero verticalmente

gbc.fill=GridBagConstraints.BOTH; //indica no caso de quedar espazo dispoñible na celda, se o elemento o ocupará

// ou

gbc.anchor=GridBagConstraints.CENTER; // indica onde se coloca o elemento se non ocupa toda a celda

panel.add(elemento, gbc);

````

### Weightx e weighty

Controla canto do espacio \*\*adicional\*\* do contenedor 🟦 ocupa unha celda ⛓️ con respecto as demais, para eso compara os weight das celdas. As celdas con números mais grandes serán as que ocuparán mais.

O calculo que fai será o seguinte:

1. Poñamonos que temos 2 elementos, con weightx `0.1` e `2.0`

2. Sumanse todos os valores, o que nos da `2.1`

3. Para cada elemento, dividese o seu valor entre o valor total:

4. elem1: 0.1/2.1 ≈ `4,76%` do espacio extra

5. elem2: 2.0/2.1 ≈ `95.24%` do espacio extra

Ejemplo mais claro: no caso dun contedor con dous elementos con weight 2.0 e 1.0, o de 2.0 ocupara o doble de espacio dispoñible que o de 1.0

### Fill

Indica se o compoñente se expandirá para ocupar todo o espazo dispoñible na celda.

⚠️ Non confundir concepto con weight, xa que `fill actua sobre o compoñente`, e di se este se expande ou non dentro da celda. Sen embargo, `weight actúa sobre a celda`, comparandose co das demais celdas. ⚠️

```java

import java.awt.\*;

GridBagConstraints gbc=new GridBagConstraints();

gbc.fill=GridBagConstraints.BOTH; //expandirase vertical e horizontalmente

gbc.fill=GridBagConstraints.HORIZONTAL; //solo horizontalmente

gbc.fill=GridBagConstraints.VERTICAL; //solo verticalmente

gbc.fill=GridBagConstraints.NONE; //non se expandirá

```

### Anchor

Indica onde se posicionará o elemento na celda. Polo que non ten sentido usalo se o elemento ten o fill activado.

````

Norte(North)

|

Noroeste(NorthWest) | Nordeste(NorthEast)

|

Oeste(West) -----+----- Este(East)

|

Suroeste(SouthWest) | Sureste(SouthEast)

|

Sur(South)

````

Permite os seguintes valores:

````java

import java.awt.\*;

gbc.anchor=GridBagConstraints.CENTER //(por defecto) colocao no centro da celda

//pois eso, as posicions da brujula

gbc.anchor=GridBagConstraints.[NORTH | NORTHEAST | EAST | SOUTHEAST | SOUTH | SOUTHWEST | WEST | NORTHWEST]

````

Pero para `textos` por ejemplo, usanse os siguientes.

Porque o inicio dunha liña nun idioma pode ser a esquerda e noutro a dereita, esto adaptao automaticamente según algúns valores da nosa aplicación.

````java

import java.awt.\*;

gbc.anchor=GridBagConstraints.FIRST\_LINE\_START; //ao inicio da primeira liña

gbc.achor=GridBagConstraints.FIRST\_LINE\_END; //ao fin da primeira liña

gbc.achor=GridBagConstraints.LAST\_LINE\_START; // principio da ultima

gbc.achor=GridBagConstraints.LAST\_LINE\_END; //fin da ultima

````

### Insets

Funciona como o `padding` en html. Añade espacio en blanco entre o componente e a celda na que esta.

\* `Insets(arriba,esquerda,abaixo,dereita)`

````java

import java.awt.\*;

gbc.insets=new Insets(10,0,10,0); // 10px arriba e abaixo

````

❗❗ Recalcar que se se usa o mismo GridBagConstraints para varios compoñentes, se indicamos o `.insets` e mais tarde o queremos

quitar para outro compoñente NON se pode poñer a `null`, habería que facer un novo cos campos vacios. Ainda que se se quixera aforrar memoria e non crear tantos obxetos novos sería boa practica ter algúns xa definidos ou usar un solo e ilo modificando:

\* `new Insets(0,0,0,0)`