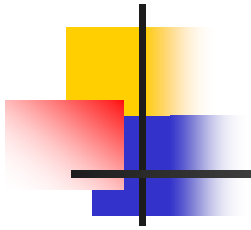
Introducción a las Interfaces Gráficas de Usuario en Java

Ingeniería del Software II
Curso 2010/2011

Sergio Ilarri Artigas
silarri@unizar.es



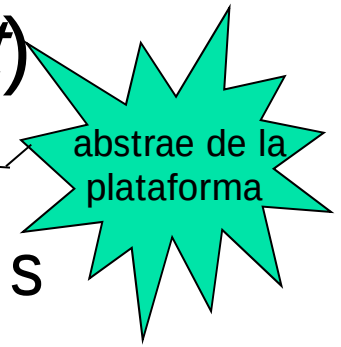
Índice

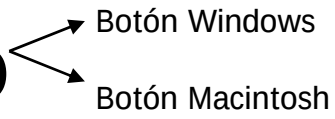
- Swing:
  - Swing vs. AWT
  - Ejemplos: HelloWorld, SwingApplication
  - Manejadores de eventos
  - Otros: Layout Managers, Borders, L&F
- Applets:
  - Introducción
  - Ciclo de vida
  - Restricciones de seguridad

Swing

AWT vs. Swing (I)

- *AWT* (***Abstract*** *Window Toolkit*)
  - Desde Java 1.0
  - Facilidades básicas para crear GUIs
  - Facilidades básicas para gráficos
  - Sobre el sistema GUI nativo del SO
  - En Java 1.1 se extendió para permitir componentes gráficos ligeros (*lightweight*)





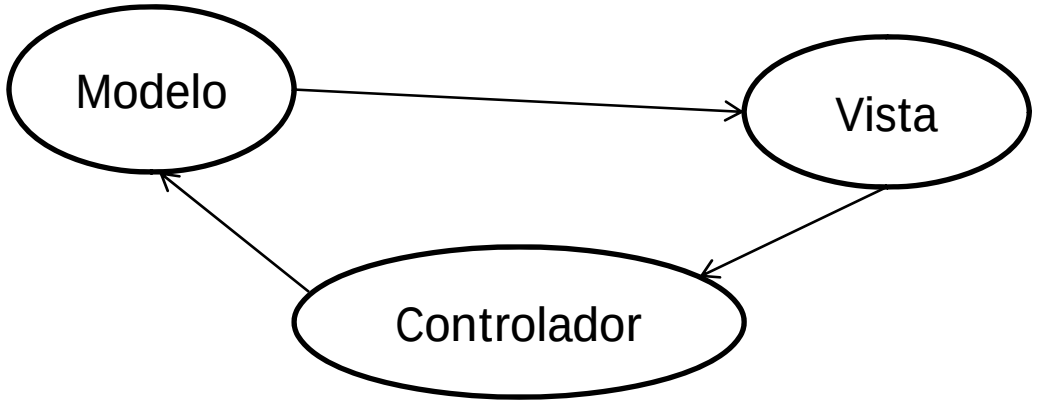
No asociados a componentes GUI nativos

AWT vs. Swing (II)

- *Swing*
  - Parte de Java 2 y como extensión en Java 1.1
  - Extensión de AWT
  - Todos sus componentes son ligeros:
    - Responsables de su propia representación gráfica
    - Más portable (100% Java, sin componentes nativos)
    - *Pluggable Look and Feel*
  - Mucho más completo y potente

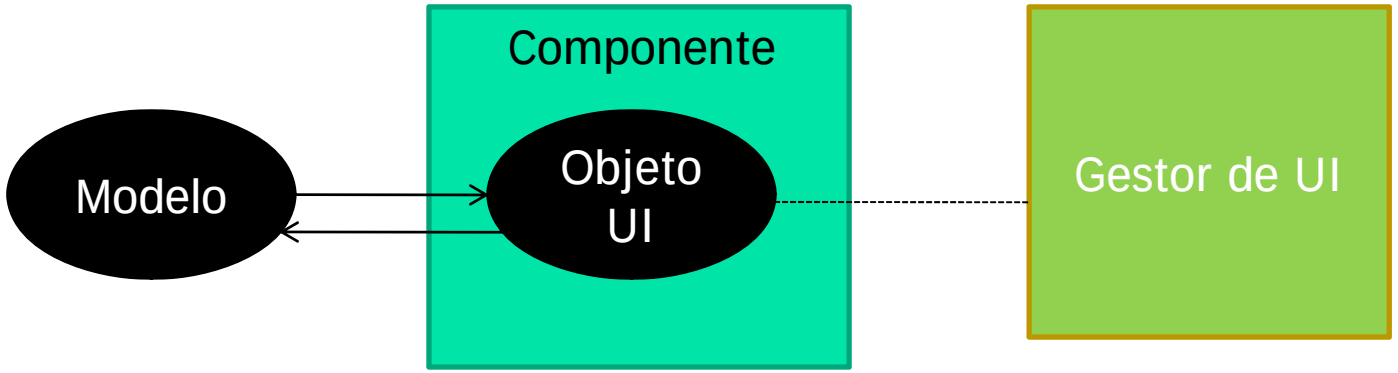
Arquitectura MVC (I)

- *Model-View-Controller*
- División de una aplicación visual en 3 partes:
  - Modelo: representa los datos de la aplicación
  - Vista: es la representación visual de los datos
  - Controlador: recibe entradas del usuario a través de la vista y actualiza el modelo en consecuencia

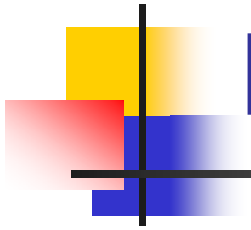


Arquitectura MVC (II)

- El primer prototipo de Swing seguía ese estilo
- Pero luego se determinó que la vista y el controlador requerían un fuerte acoplamiento
- *Separable model architecture* (*quasi-MVC*)
  - *UI* (User Interface) *Object* = *UI delegate* = *delegate object*
    - Vista + controlador



http://java.sun.com/products/jfc/tsc/articles/architecture/



Paquetes Swing en Java 1.4

javax.accessibility

javax.swing

javax.swing.border

javax.swing.colorchooser

javax.swing.event

javax.swing.filechooser

javax.swing.plaf

javax.swing.plaf.basic

javax.swing.plaf.metal

javax.swing.plaf.multi

javax.swing.table

javax.swing.text

javax.swing.text.html

javax.swing.text.parser

javax.swing.text.rtf

javax.swing.tree

javax.swing.undo

java.awt, java.awt.event

Ejemplo: HelloWorld

```
import javax.swing.*;
public class HelloWorldSwing {
 private static void createAndShowGUI() {
  JFrame.setDefaultLookAndFeelDecorated(true);
  JFrame frame = new JFrame("HelloWorldSwing");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  JLabel label = new JLabel("Hello World");
  frame.getContentPane().add(label);
  frame.pack();
  frame.setVisible(true);
 }
 public static void main(String[] args) {
  javax.swing.SwingUtilities.invokeLater(new Runnable() {
   public void run() {createAndShowGUI();}
  });
 }
}
```

Top-level container (otros: *JDialog*, *JApplet*)

Se añade al *content pane*

Evita problemas (*thread-safe*)

Ejemplo: SwingApplication (I)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SwingApplication implements ActionListener {

  private static String labelPrefix =
    "Number of button clicks: ";
  private int numClicks = 0;
  final JLabel label = new JLabel(labelPrefix + "0    ");

  ...
}
```

Implementa manejadores de eventos *action* , luego puede registrarse como *listener* de un objeto gráfico

Ejemplo: SwingApplication (II)

```
public Component createComponents() {
  JButton button = new JButton("I'm a Swing button!");
  button.setMnemonic(KeyEvent.VK_I);
  button.addActionListener(this);
  label.setLabelFor(button);
  JPanel pane = new JPanel(new GridLayout(0, 1));
  pane.add(button);
  pane.add(label);
  pane.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30);
  return pane;
}
```

Este objeto va a escuchar eventos *action* del botón

Píxels: top, left, right, bottom

JPanel : contenedor para agrupar componentes
GridLayout : *layout manager*
EmptyBorder : crear “hueco”/separación

Ejemplo: SwingApplication (III)

```
public void actionPerformed(ActionEvent e) {
  numClicks++;
  label.setText(labelPrefix + numClicks);
}
```

Implementa el *interface ActionListener*

Cuando se haga *click* en el botón

Hay un único *event-dispatching thread*
(para el tratamiento de eventos y repintado).
Por tanto, ¡hay que ser rápidos!

Ejemplo: SwingApplication (IV)

```
private static void createAndShowGUI() {
  JFrame.setDefaultLookAndFeelDecorated(true);
  JFrame frame = new JFrame("SwingApplication");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  SwingApplication app = new SwingApplication();
  Component contents = app.createComponents();
  frame.getContentPane().add(contents, BorderLayout.CENTER);
  frame.pack();
  frame.setVisible(true);
}

public static void main(String[] args) {
  javax.swing.SwingUtilities.invokeLater(new Runnable() {
    public void run() {createAndShowGUI();}
  });
 }
```

Algunos Manejadores de Eventos

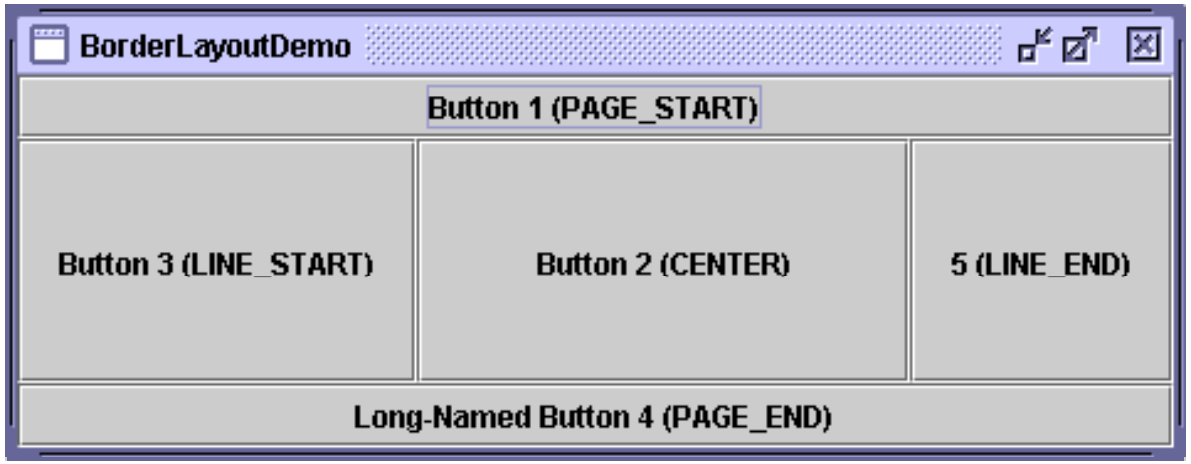
Acción que desencadena el evento	Manejador
El usuario hace click en un botón, presiona *Enter* al escribir texto en un campo, o selecciona una opción de menú	*ActionListener*
El usuario cierra una ventana	*WindowListener*
El usuario presiona un botón del ratón cuando el cursor está sobre un componente	*MouseListener*
El usuario mueve el ratón sobre un componente	*MouseMotionListener*
Un componente se hace visible	*ComponentListener*
Un componente obtiene el foco del teclado	*FocusListener*
Cambia la opción seleccionada en una lista	*ListSelectionListener*
Cambia cualquier propiedad de un componente (ej.: el texto de una etiqueta)	*PropertyChangeListener*

Hay muchos ejemplos de distintos tipos de manejadores de eventos en:

http://java.sun.com/docs/books/tutorial/uiswing/events/intro.html

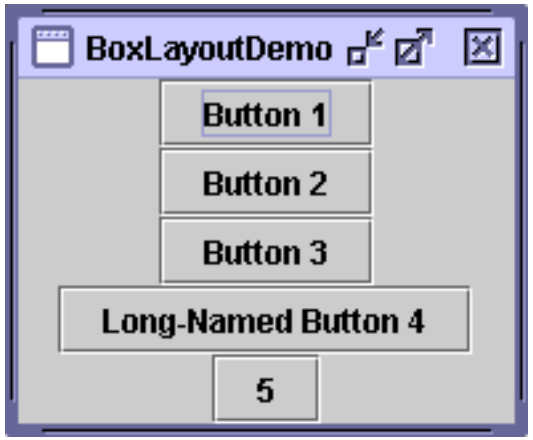
Layout Managers (I)

java.awt
javax.swing

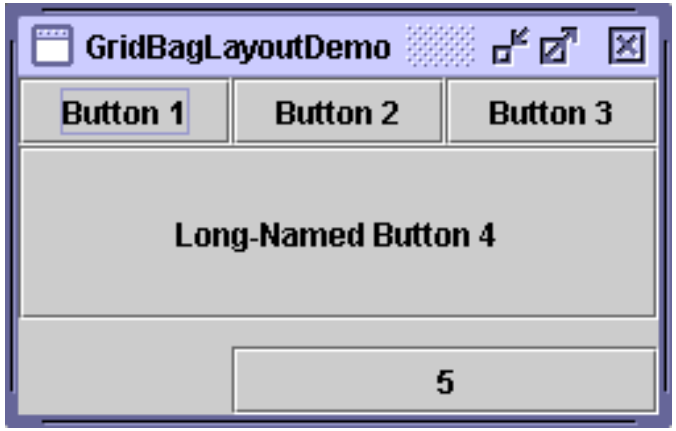


5 áreas: *top*, *bottom*, *right*, *left*, *center*
El área *center* acapara todo el espacio sobrante

Por defecto en los *content pane*
(diálogos, *frames* y *applets*)



1 única fila
o columna



-Sitúa en una rejilla

-Las filas y columnas pueden tener diferentes alturas y anchuras

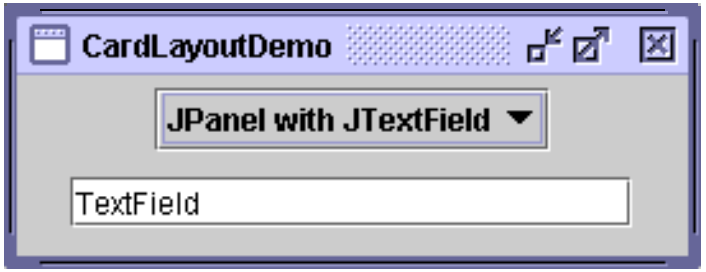
-Un componente puede ocupar varias celdas

Layout Managers (II)

1 única fila

Si se llena, empieza otra

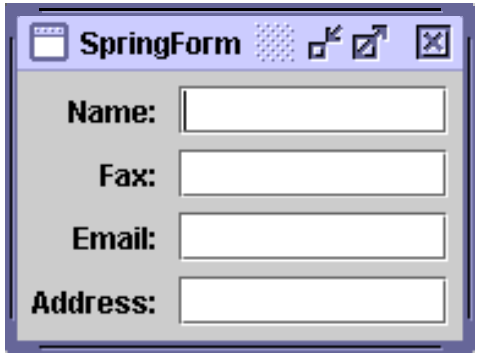
Por defecto en un *JPanel*



Contiene componentes distintos dependiendo de la selección

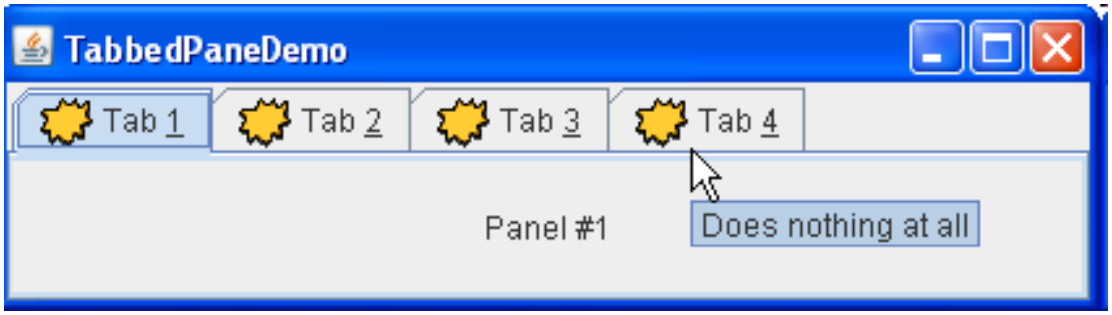
Layout Managers (III)

Componentes de igual tamaño
Rejilla de filas y columnas



Permite especificar relaciones (distancias)
entre componentes

Layout Managers (VI)



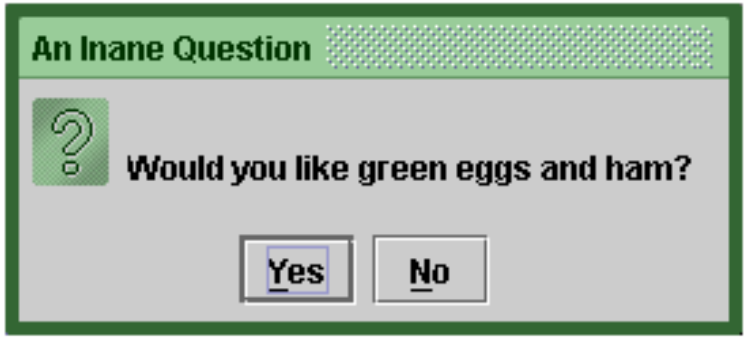
No es recomendable utilizar posicionamiento absoluto

Componentes (I)

Guía visual e interactiva de los componentes de Swing:
http://download.oracle.com/javase/tutorial/ui/features/components.html
Los componentes Jxxx, menos los contenedores de nivel superior, heredan de:
javax.swing.JComponent

1) Contenedores de nivel superior

Applet

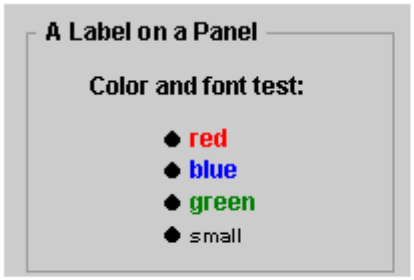


Dialog

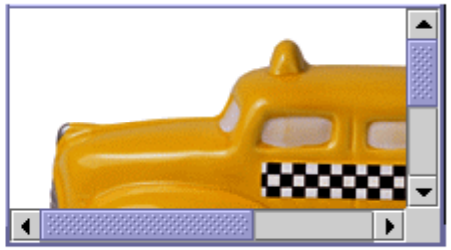
Frame

Componentes (II)

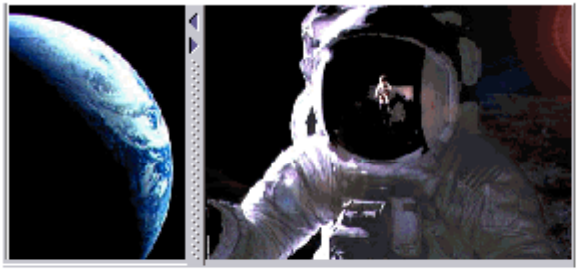
2) Contenedores de propósito general



Panel



Scroll pane



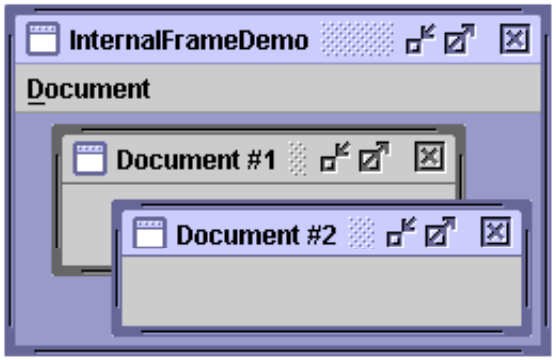
Split pane

Tabbed pane

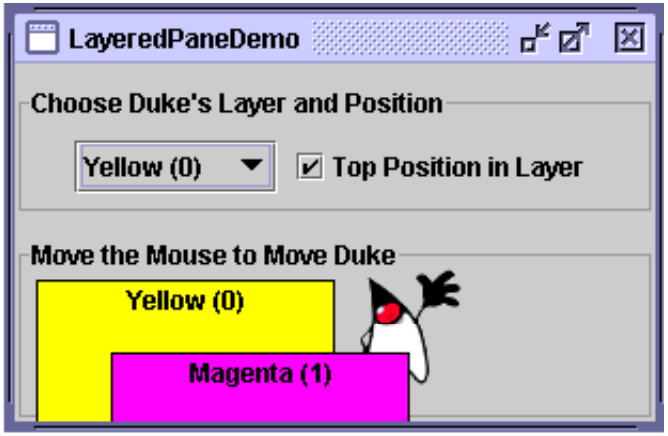
Tool bar

Componentes (III)

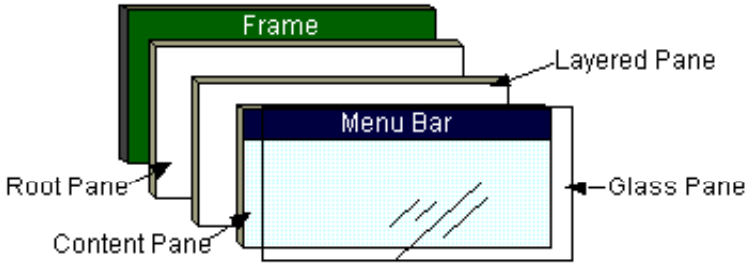
3) Contenedores de propósito especial



Internal frame



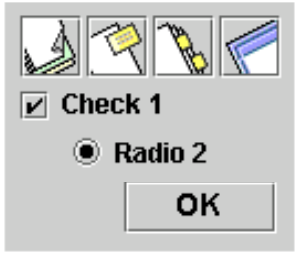
Layered pane



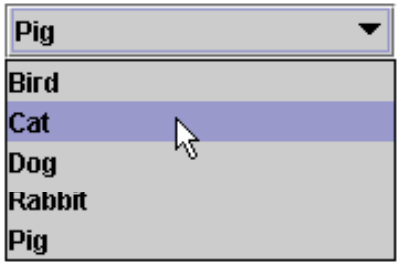
Root pane

Componentes (IV)

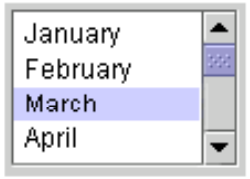
4) Controles básicos



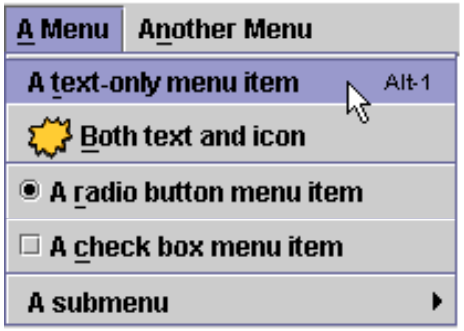
Buttons



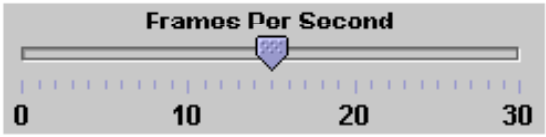
Combo box



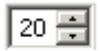
List



Menu



Slider

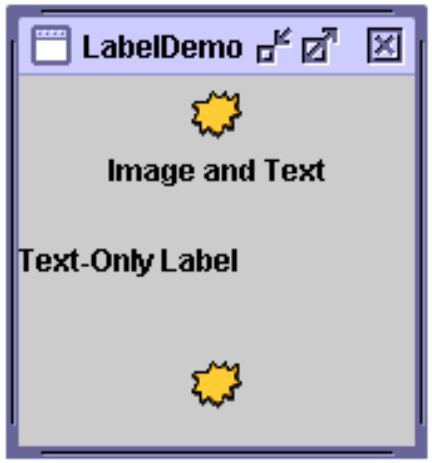


Spinner

Text field or Formatted text field

Componentes (V)

5) Elementos de información no editables



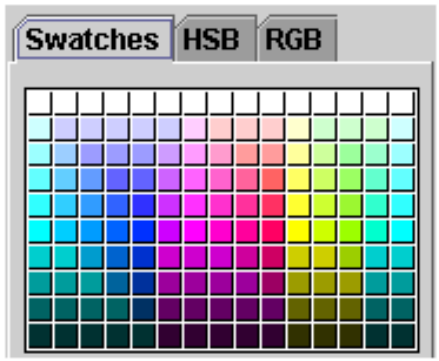
Label

Progress bar

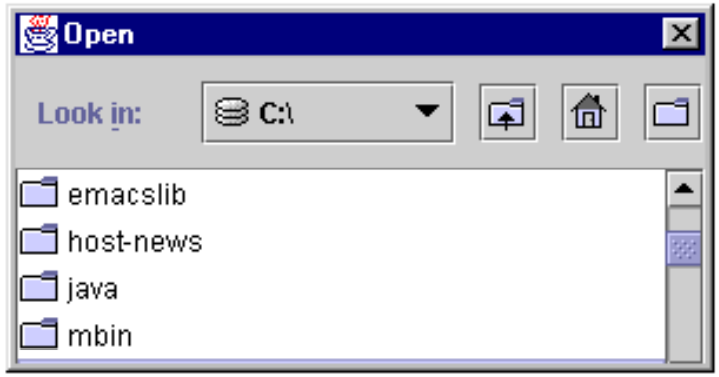
Tool tip

Componentes (VI)

6) Elementos de información interactivos



Color chooser

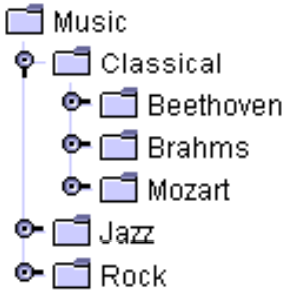


File chooser

First Name	Last Name	Favorite Food
Jeff	Dinkins	
Ewan	Dinkins	
Amy	Fowler	
Hania	Gajewska	
David	Geary	

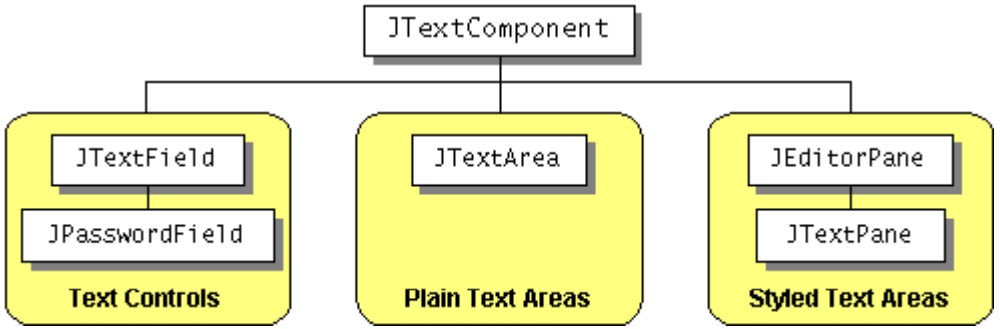
Table

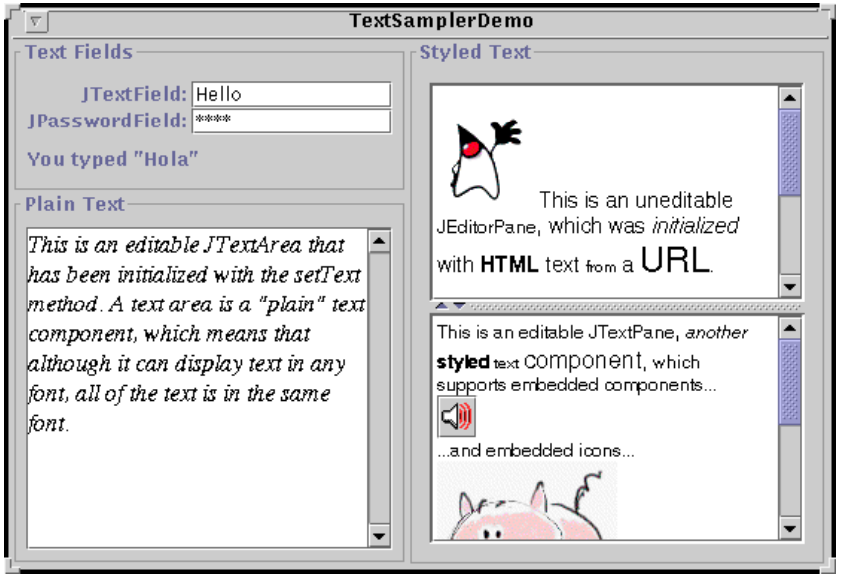
Text



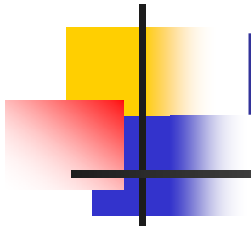
Tree

Componentes (VII): Texto



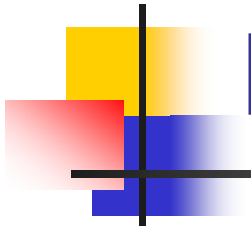


Componentes (VIII): Nombres (I)



- Botones (javax.swing.JButton)
- Casillas de verificación (javax.swing.JCheckBox)
- Campos de texto de una línea (javax.swing.JTextField)
- Campos de texto y edición de varias líneas (javax.swing.JTextArea)
- Etiquetas (javax.swing.JLabel)
- Listas (javax.swing.JList)
- Menús contextuales (javax.swing.Popup)

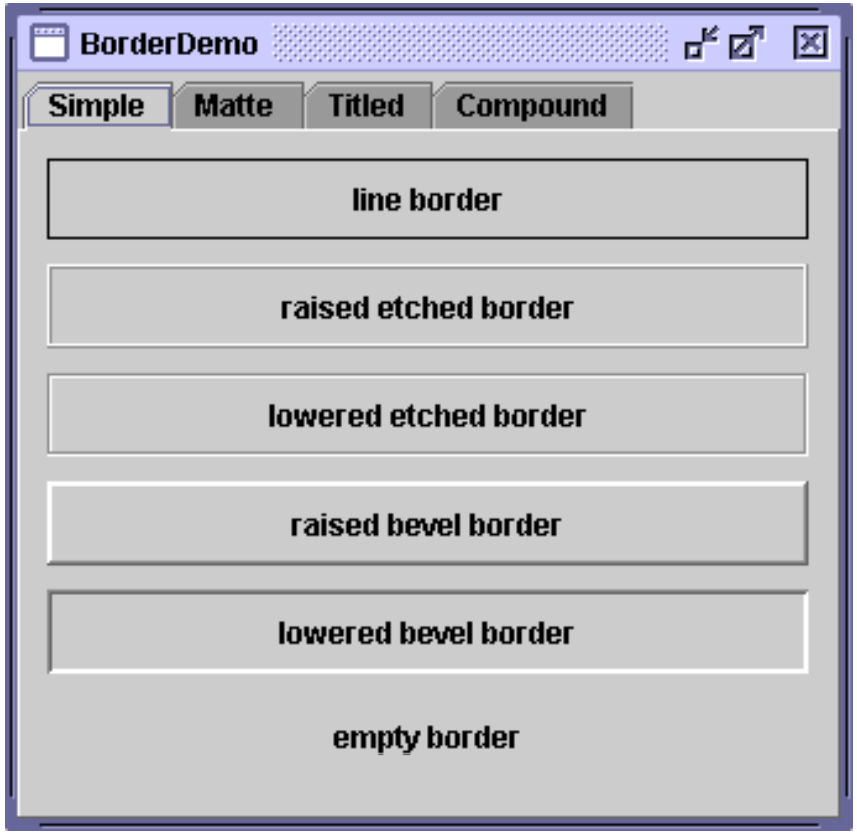
Componentes (IX): Nombres (II)

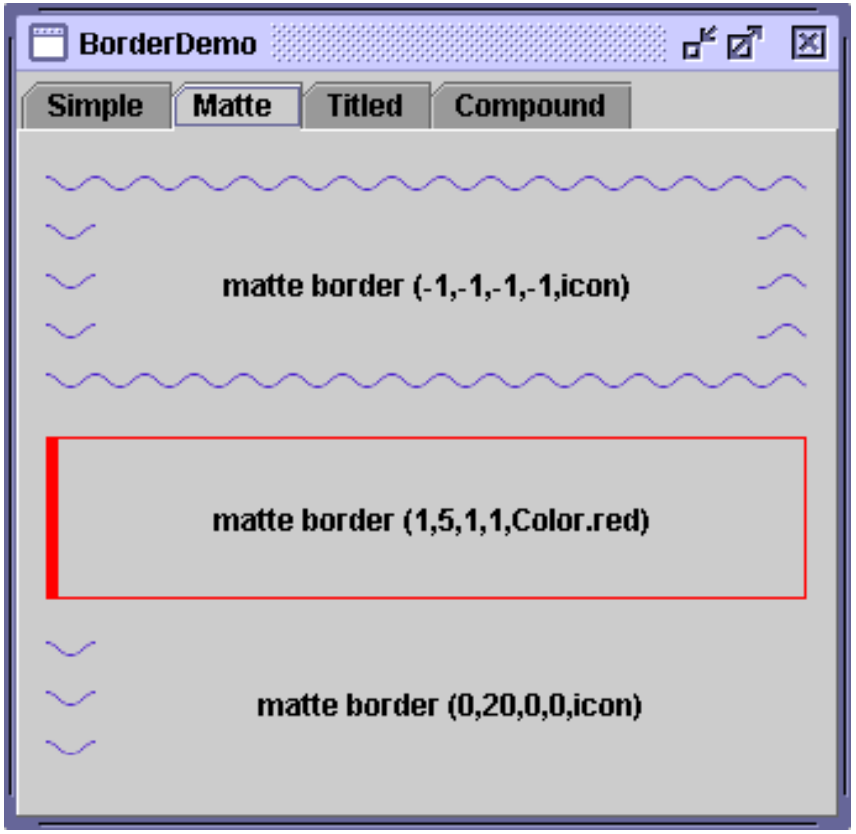


- Barras de desplazamiento (javax.swing.JScrollBar)
- Sliders (javax.swing.JSlider)
- Áreas de dibujo (java.awt.Canvas)
- Menus (javax.swing.JMenu,javax.swing.JMenuBar javax.swing.JMenuItem, javax.swing.JCheckBoxMenuItem)
- Contenedores (javax.swing.JPanel, javax.swing.JWindow and its subclasses)
- ...

Algunos Tipos de *Borders* (I)

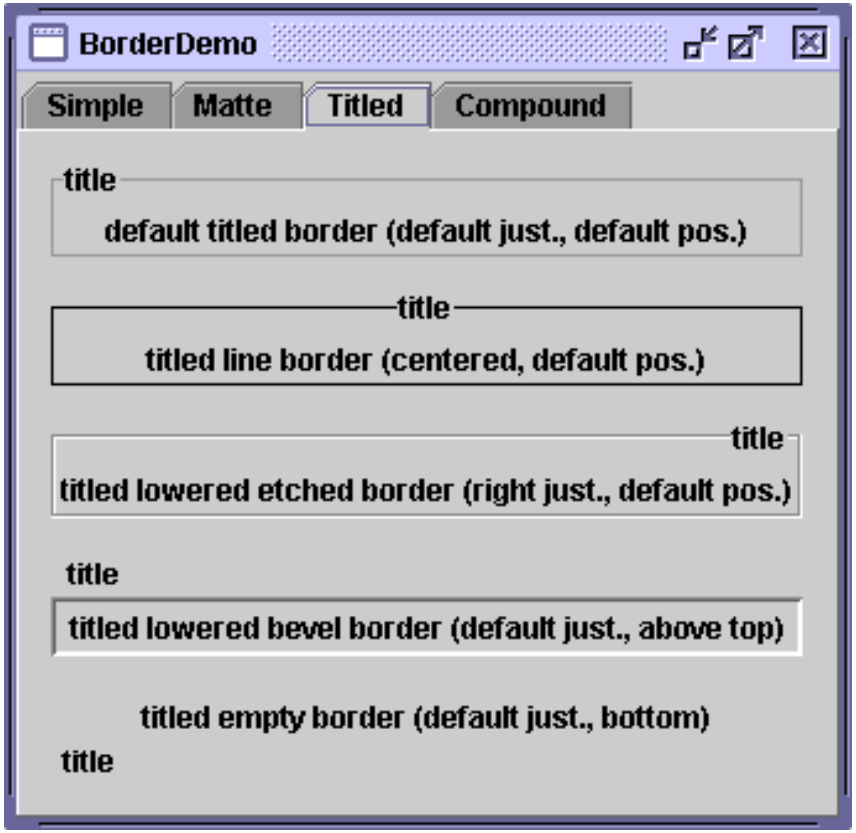
Elementos clave: clase *BorderFactory* y método *setBorder* de *JComponent*

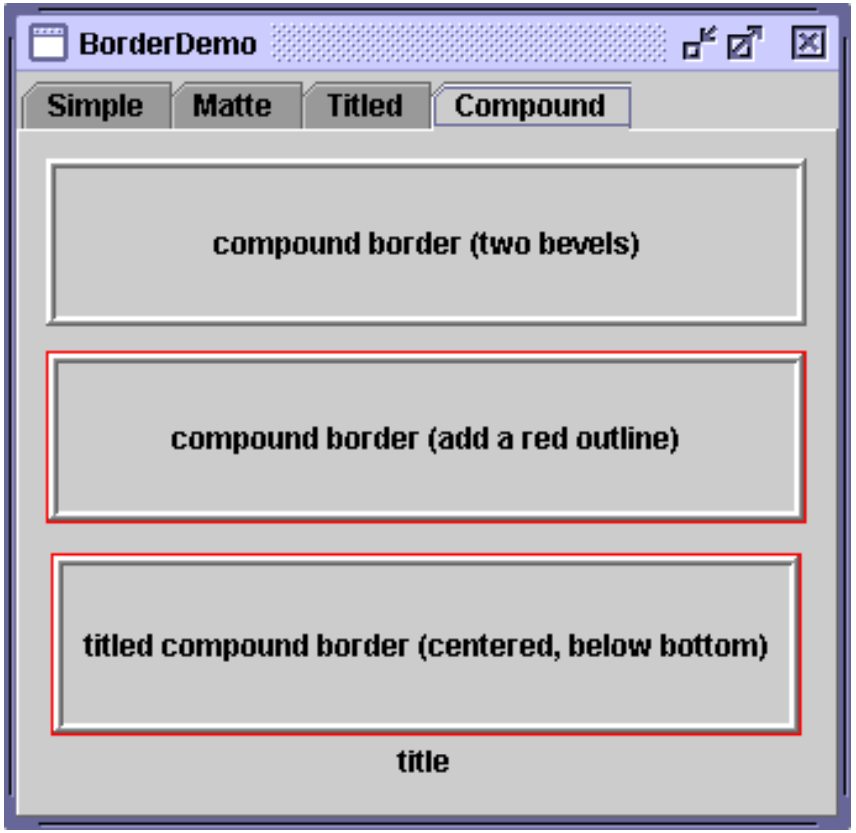




Paquete: javax.swing.border

Algunos Tipos de *Borders* (II)

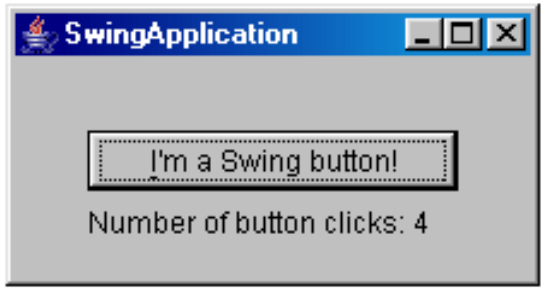




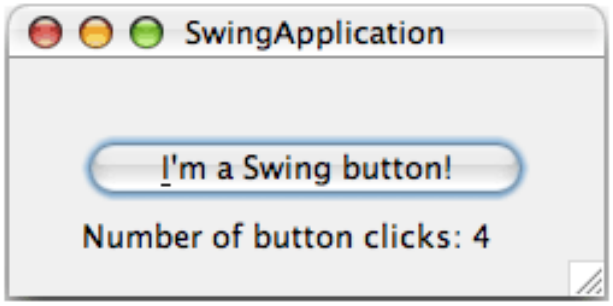
Algunos *Look and Feels*

Java look and feel

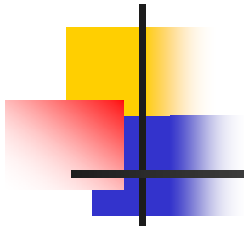
GTK+ look and feel



Windows look and feel



Mac OS look and feel



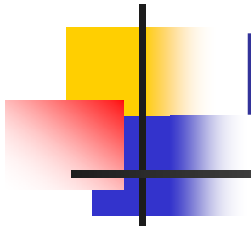
Ejemplo: Dibujar Círculo (I)

```
package is2.silarri;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class CircleDrawer extends JFrame {
   ... /* Desarrollado en las dos transparencias siguientes */
   }

class PanelWithCircle extends JPanel {
   public void paintComponent(Graphics g)
   {
      super.paintComponent(g);
       g.drawOval( 300, 300, 60, 60 );
   }
}
```



Ejemplo: Dibujar Círculo (II)

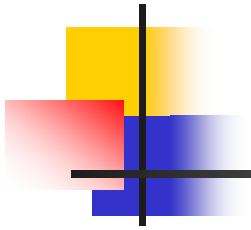
```
public CircleDrawer() {
        super("Circle example for IS2 - Sergio Ilarri, " +
                        "October 8, 2006");
        PanelWithCircle pwc = new PanelWithCircle();
        Container container = getContentPane();
        container.add(pwc);

        setSize(600, 600);
        setVisible(true);
   }
```

Ejemplo: Dibujar Círculo (III)

```
public static void main(String[] args) {
        CircleDrawer cd = new CircleDrawer();
        cd.addWindowListener (new WindowAdapter() {
                public void windowClosing ( WindowEvent e )
                {
                        System.exit(0);
                }
        });
   }
```

Applets



Introducción

- Programa Java que un navegador puede descargar y poner en ejecución
- Embebido en una página web
- Se ejecuta en un entorno seguro del navegador (*sandbox*)
- Clases:
  - *java.applet.Applet* : interface estándar
  - *javax.swing.Japplet* : si se usa Swing

Ciclo de Vida de un Applet

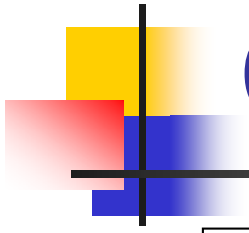
~ constructor

- init: inicialización del applet

sobrecargar

- start: invocado automáticamente después de init y al recargar la página
- stop:
  - Invocado automáticamente al pasar a otra página
  - Puede usarse para detener animación
- destroy: cuando se cierra el navegador

Sólo 1 vez



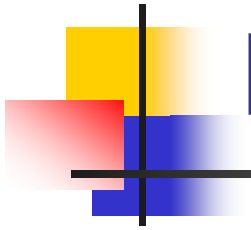
Carga de Applets

```
<applet code=AppletWorld.class width="200"

height="200"> </applet>
```

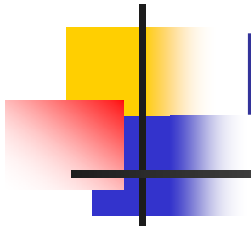
```
import javax.swing.JApplet;
import java.awt.Graphics;
public class HelloWorld extends JApplet {
   public void paint(Graphics g) {
      g.drawRect(0, 0, getSize().width - 1,
         getSize().height - 1);
      g.drawString("Hello world!", 5, 15);
   }
}
```

Ejecutado por un *Java plug-in*



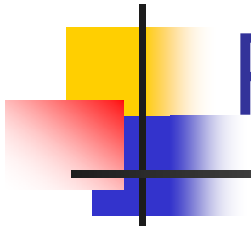
Manejo de Componentes

- *add*
  - Añade un componente
- *remove*
  - Elimina un componente
- *setLayout*
  - Establece un gestor de *layout*



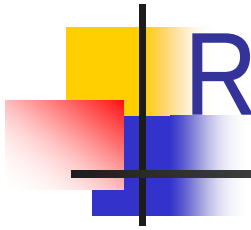
Restricciones de Seguridad (I)

- Un *applet descargado* no puede:
  - Cargar librerías o definir métodos nativos
  - Leer o escribir ficheros
  - Establecer conexiones de red (excepto a su máquina origen)
  - Ejecutar programas
  - Leer ciertas propiedades del sistema
  - Sus ventanas son diferentes a las de las aplicaciones



Restricciones de Seguridad (II)

- Cada navegador tiene su objeto *SecurityManager* para detectar violaciones
- Si se produce una violación, se lanza una *SecurityException*



Restricciones de Seguridad (III)

- Pero los *applets* sí pueden:
  - Establecer conexiones de red con su máquina origen
  - Mostrar documentos HTML
  - Invocar métodos públicos de otros *applets* de la misma página
  - Si se cargan desde el sistema de ficheros local, no tienen restricciones

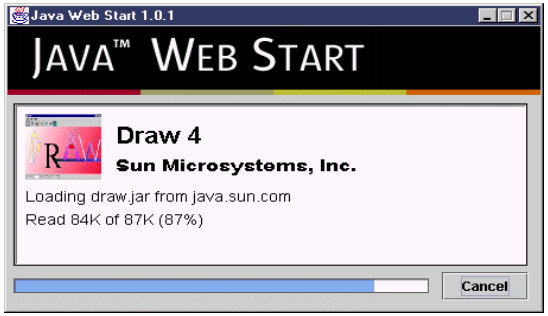
Diferencias con Programas

- No hay método *main*
- No hay constructor, la inicialización se hace en *init* y *start*
- Extiende a *Applet* o *JApplet*
- Los componentes GUI se añaden directamente al applet (no a un *content pane*)... excepto al extender *JApplet*

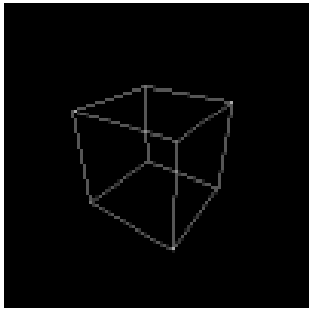
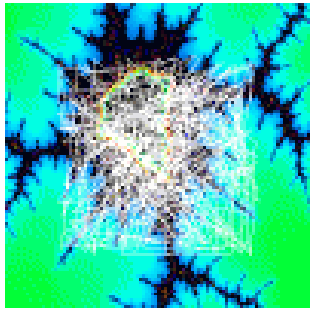
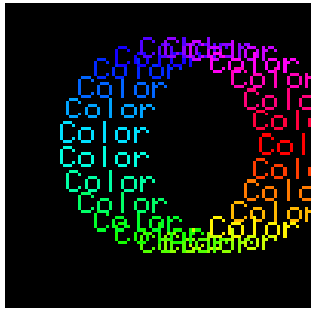
contenedor de alto nivel

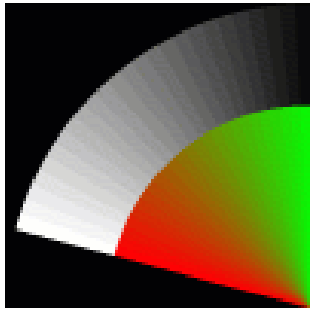
Comentarios Finales

- Gran parte del éxito de Java se debe a los *applets*
- Herramienta ligera para su visualización: *appletviewer*
- En la actualidad, *Java Web Start* (JNLP) juega un papel similar para aplicaciones



Algunos Ejemplos

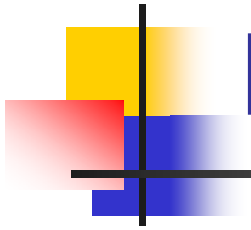






http://www.dgp.toronto.edu/~mjmcguff/learn/java/

http://java.sun.com/applets/

http://download.oracle.com/javase/1.5.0/docs/relnotes/demos.html

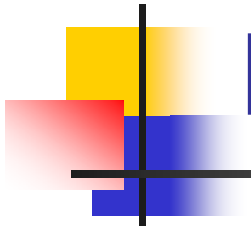


Referencias

- The Swing Tutorial.
  http://java.sun.com/docs/books/tutorial/ui/index.html

- Swing Second Edition, Matthew Robinson y Pavel Vorobiev, 2003, 912 páginas, Manning Publications Co., ISBN 193011088X. Versión previa gratuita en Word en: http://www.manning.com/robinson2/

- Applets.
  http://java.sun.com/applets/



Fin

Gracias por vuestra atención