

Interfaz gráfica de usuario y eventos

Introducción

¿GUI?

- Graphical user interface

> Interfaz gráfica de usuario

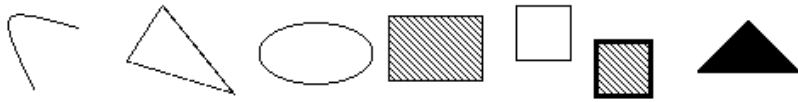
...programa o entorno que gestiona la interacción con el usuario con base en relaciones visuales como iconos, menús o un puntero»

Sample document

Close Paginate

XEROX 8010 Star Information System

Star provides integrated text and graphics. A variety of type sizes and styles may be used.



Description	Price
Peas	\$0,39
Beans	\$0,50

Sample

Close

This is some text in a text frame.

Form field

Button



Start

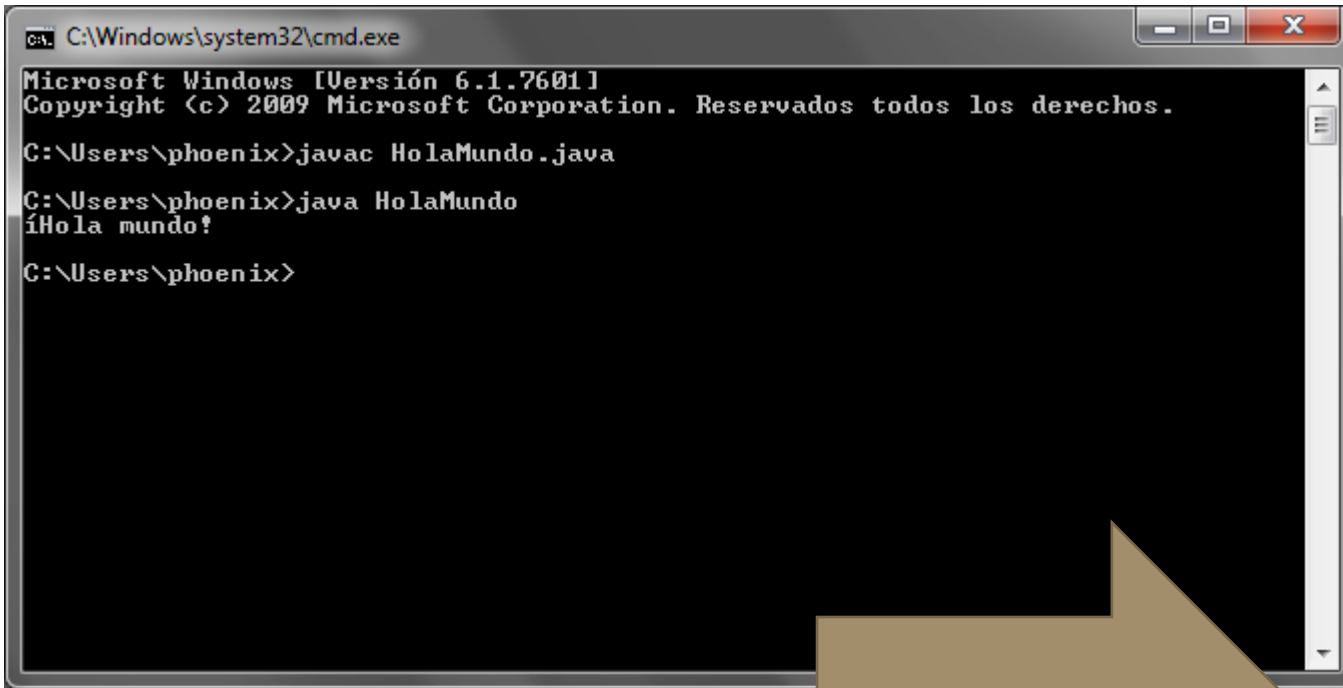
martin 



4:57

Wednesday
September 14

En nuestro caso...

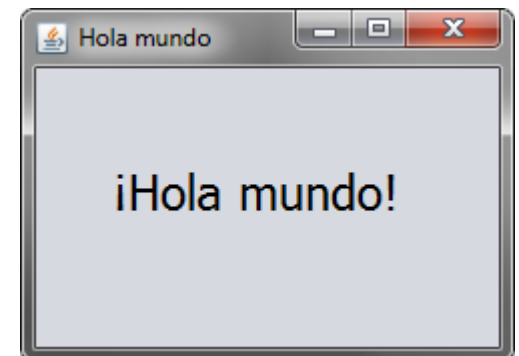
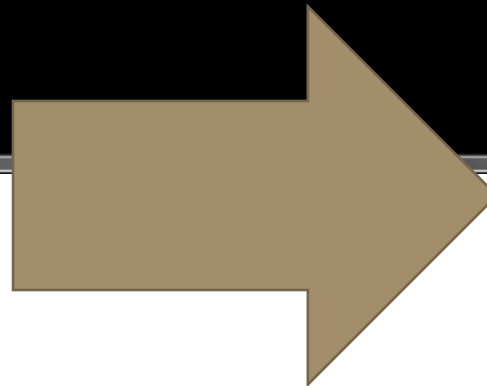


```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\phoenix>javac HolaMundo.java

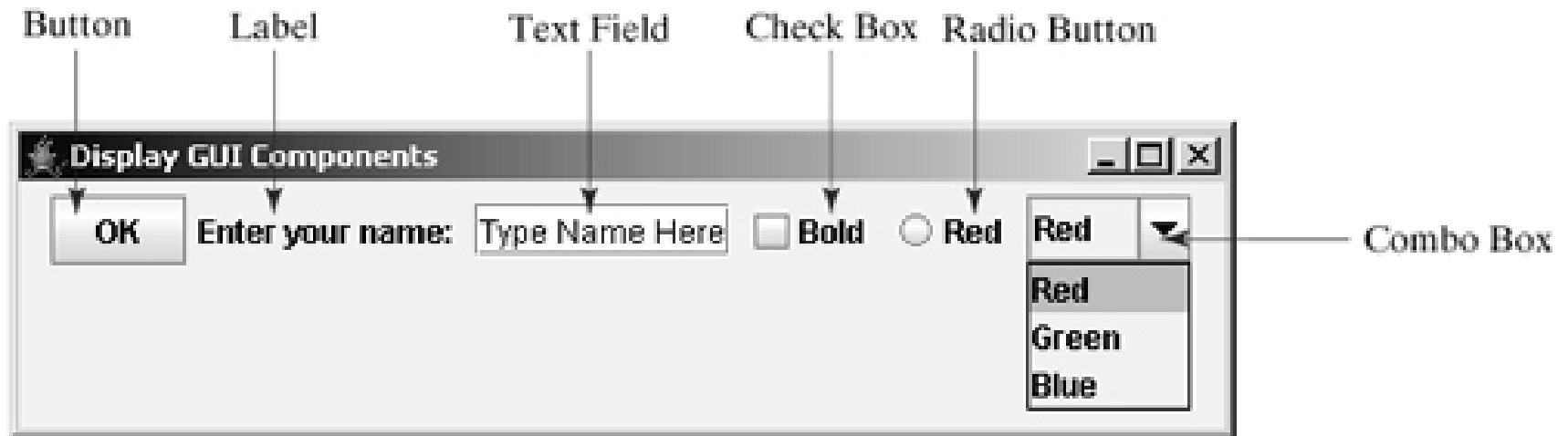
C:\Users\phoenix>java HolaMundo
¡Hola mundo!

C:\Users\phoenix>
```



Swing & AWT

- ¿Qué son?
 - Conjuntos de librerías que agrupan componentes para desarrollar interfaces gráficas



AWT (Abstract Window Toolkit)

- Fue el primero en Java (y viene en todas sus versiones)
- dependen de código nativo del sistema operativo para realizar su funcionalidad
- Su paquete es `java.awt`

Swing

- Implementado «enteramente» en Java
 - ... no tiene código nativo del sistema operativo
- Los nombres de algunas clases de Swing inician con la letra «J» para no confundirlas con las equivalentes en AWT (por ejemplo JButton y Button)
- Su paquete es javax.swing

- Junto con **Java 2D** (clases para manipular imágenes y gráficos planos), **AWT** y **Swing** conforman el **JFC (Java Foundation Classes)**
 - ... framework para construir interfaces gráficas de usuario basadas en Java

Diagrama jerárquico de clases para programación GUI en Java

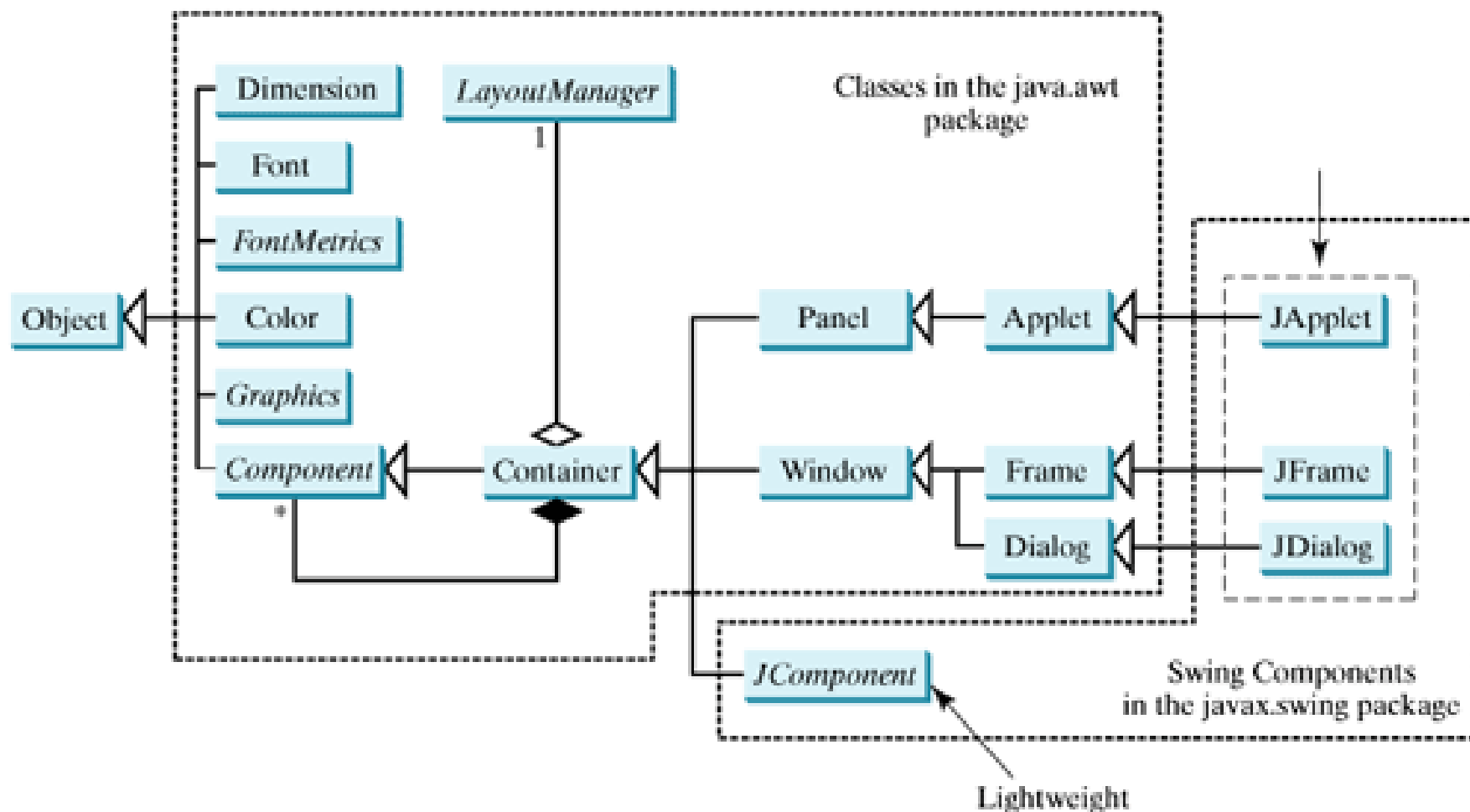
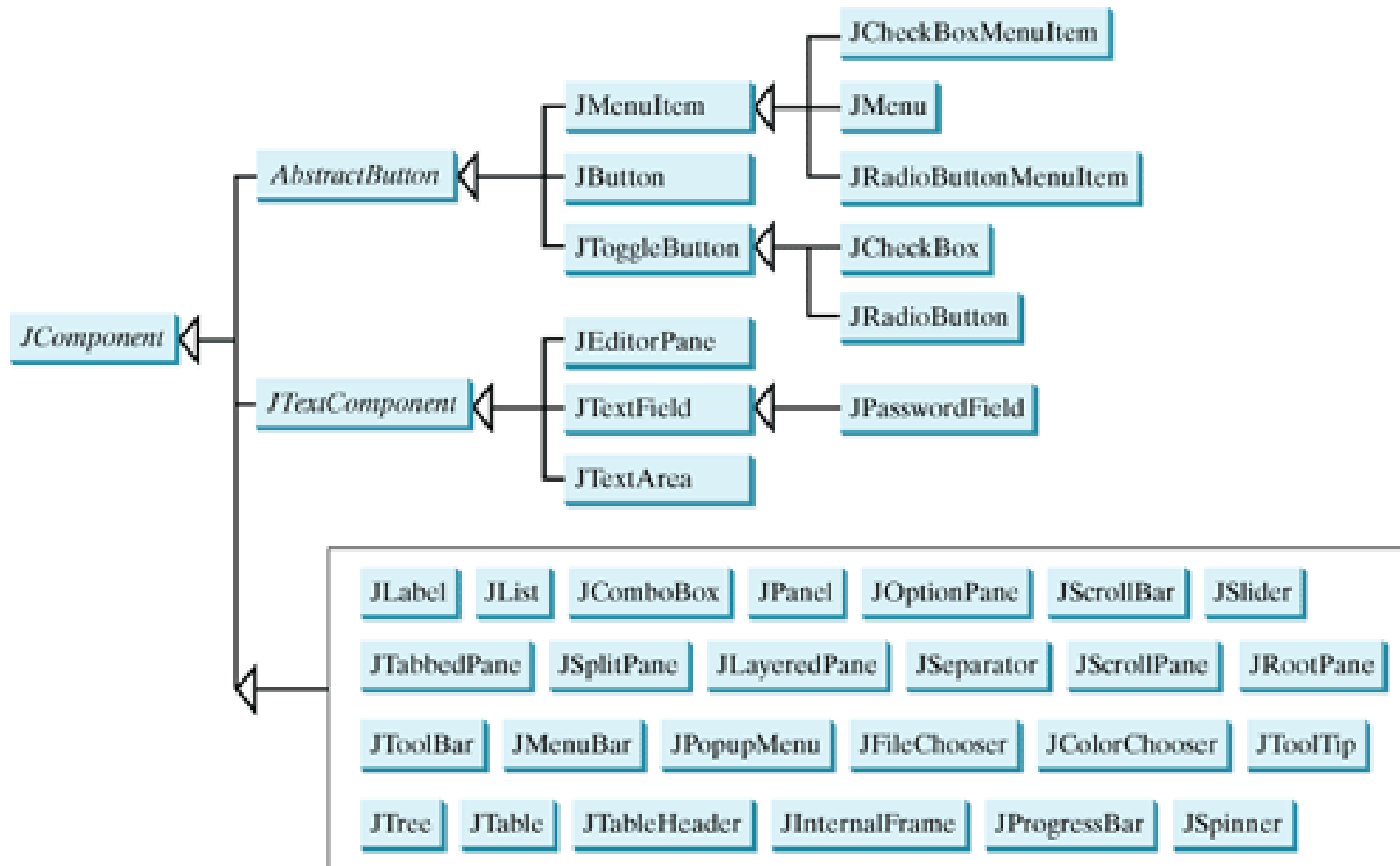


Diagrama jerárquico de clases para programación GUI en Java (cont.)



Clasificación GUI

- Se dividen en tres:
 1. Contenedores
 2. Componentes
 3. Auxiliares

Contenedores

Clases contenedor

- Son componentes GUI usados para «contener» otros componentes
- Son contenedores AWT:
 - Window, Panel, Applet, Frame y Dialog
- Son contenedores Swing:
 - Container, JFrame, JDialog, JApplet, y JPanel

Contenedores Swing

- Container
 - Usado para agrupar componentes.
 - Frames, panels y applets son ejemplos de contenedores
- JFrame
 - Es una ventana no contenida dentro de otra ventana.
 - Es el componente base para construir las interfaces basadas en Swing
- JDialog
 - Es una ventana emergente o de mensaje, usada de manera temporal para recibir información del usuario o dar alguna notificación
- JApplet
 - Es subclase de Applet. Utilizada para crear applets basados en Swing
- JPanel
 - Es un contenedor invisible para agrupar componentes de interfaz de usuario.
 - Los paneles pueden ser anidados (“uno dentro de otro”)
 - Pueden ser utilizados como “lienzo” para dibujar gráficos

Frames

`javax.swing.JFrame`

```
+JFrame()  
+JFrame(title: String)  
+setSize(width: int, height: int): void  
+setLocation(x: int, y: int): void  
+setVisible(visible: boolean): void  
+setDefaultCloseOperation(mode: int): void  
+setLocationRelativeTo (c: Component):  
    void
```

Creates a default frame with no title.

Creates a frame with the specified title.

Specifies the size of the frame.

Specifies the upper-left-corner location of the frame.

Sets true to display the frame.

Specifies the operation when the frame is closed.

Sets the location of the frame relative to the specified component.

If the component is null, the frame is centered on the screen.

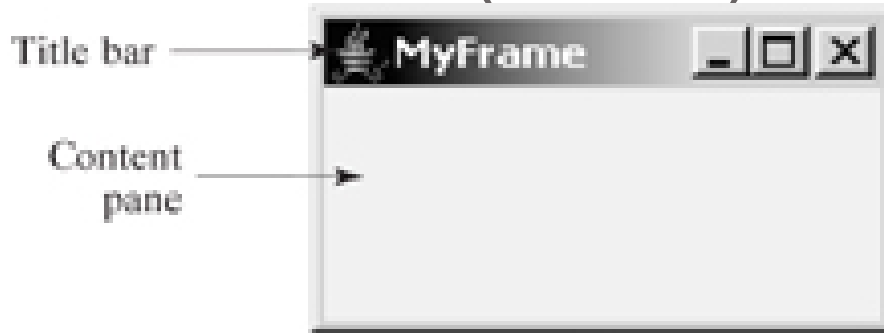
Frames (cont.)

```
import javax.swing.*;

public class MyFrame {
    public static void main(String[] args) {
        JFrame frame = new JFrame("MyFrame");
        frame.setSize(400, 300);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

- El frame no es desplegado hasta que `frame.setVisible(true);` es invocado
- `frame.setSize(400, 300);` especifica el tamaño (400px ancho, 300px alto), si no es usado el formulario se ajusta sólo hasta desplegar la barra de título
- `setVisible()` y `setSize()` están definidos en la clase `Component` y son heredados a `JFrame`, por lo que estos métodos son utilizados también por otras subclases

Frames (cont.)



- Invocando `frame.setLocationRelativeTo(null)` el frame es centrado en la pantalla
- `frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)` le dice al programa que termine cuando se cierra el formulario, si está opción no se especifica el programa seguirá en ejecución aún y cuando el formulario se haya cerrado

Frames (cont.)

- Propiedades comunes de un JFrame:
 - **contentPane**
 - Panel de contenido del frame
 - **iconImage**
 - Imagen que representa el frame, reemplaza la imagen por default de Java en la barra de título y cuando el frame es minimizado

```
Image image = (new ImageIcon(Filename)).getImage();
frame.setIconImage(image);
```
 - **JMenuBar**
 - Barra de menú opcional
 - **resizable**
 - Valor booleano que indica si el frame puede cambiar de tamaño

```
frame.setResizable(Value);
```
 - **title**
 - Especifica el título del frame
 - `frame.setTitle(Title);`

Frames (cont.)

- Para añadir componentes a un frame utilizar el método `add()` :

```
// Añadir un botón al frame  
JButton jbtOK = new JButton("OK");  
frame.add(jbtOK);
```

- Este método añade el componente al panel de contenido del frame.
 - El botón sería colocado al centro y ocupando todo el frame, esto debido al layout manager que el frame tiene por default
- Puede usarse la siguiente instrucción para eliminar el botón:

Actividad

1. Crear una clase Java de nombre PruebaFrames
2. En el método principal crear un objeto Frame con titulo «Mi primer frame»
3. Especificar el tamaño a 500px de alto por 400px de ancho
4. Guardar y ejecutar en línea de comandos

¿qué es lo que sucede?

Actividad (cont.)

5. Ubicarlo en las coordenadas (0, 0) de la pantalla
6. Hacer visible el frame
7. Guardar y ejecutar

¿qué pasa si cierran el frame?

8. Comentar la línea de ubicación anterior y agregar lo necesario para centrar el frame en la pantalla
9. Configurar que la aplicación termine al cerrar el frame
10. Guardar y ejecutar

Actividad (cont.)

11. Configurar una imagen para mostrar en la barra de título
12. Establecer que el frame no pueda cambiar de tamaño
13. Cambiar el título del frame a «Mi primer frame modificado»
14. Guardar y ejecutar

Componentes gui

Clases componente GUI

- JComponent es la superclase de todos los componentes Swing
- Como JComponent es una clase abstracta no es posible usar `new JComponent ()`, es necesario usar los constructores de las subclases

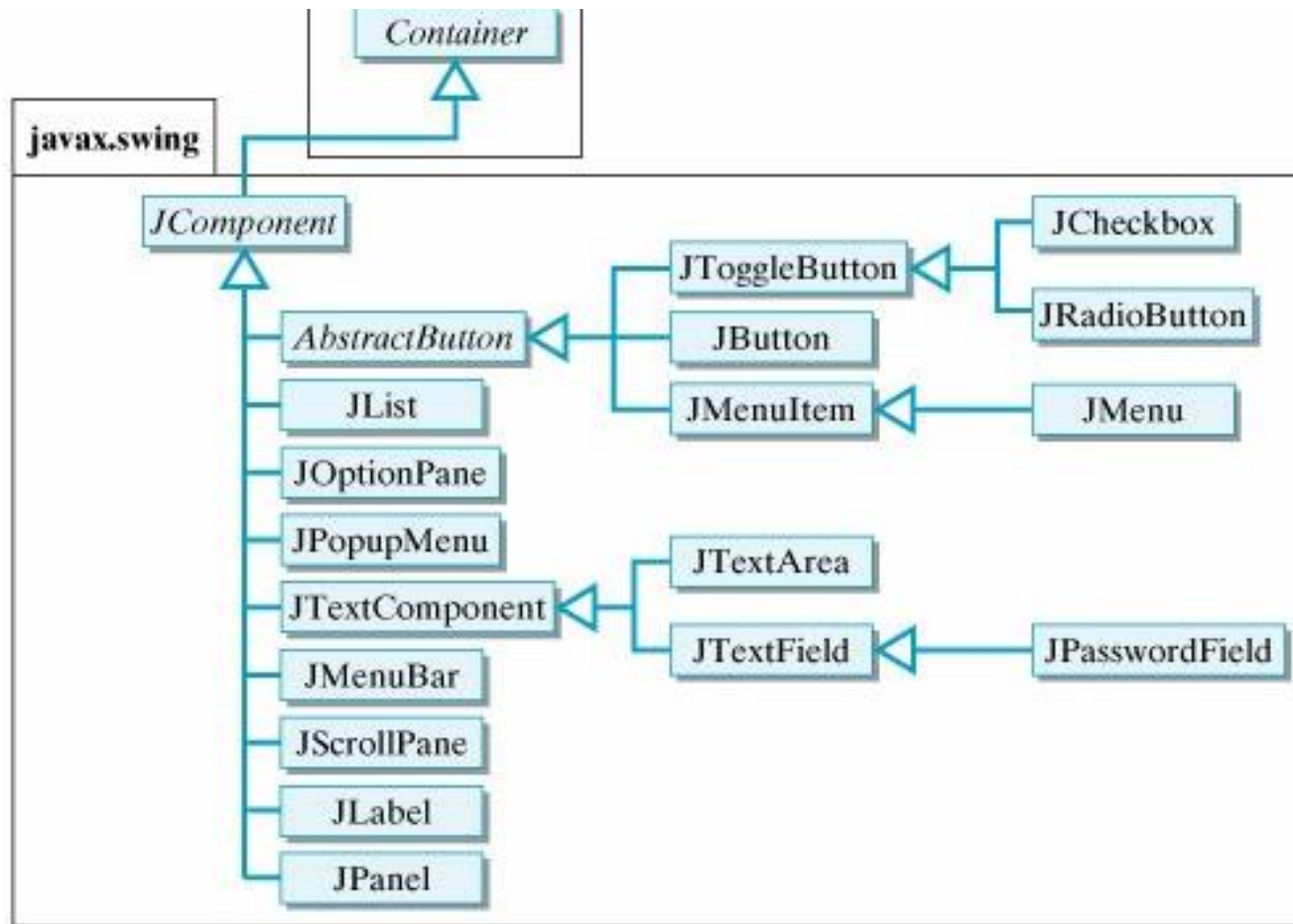
Clases componente GUI (cont.)

- ¿Cuál sería el resultado de ejecutar el siguiente fragmento de código?

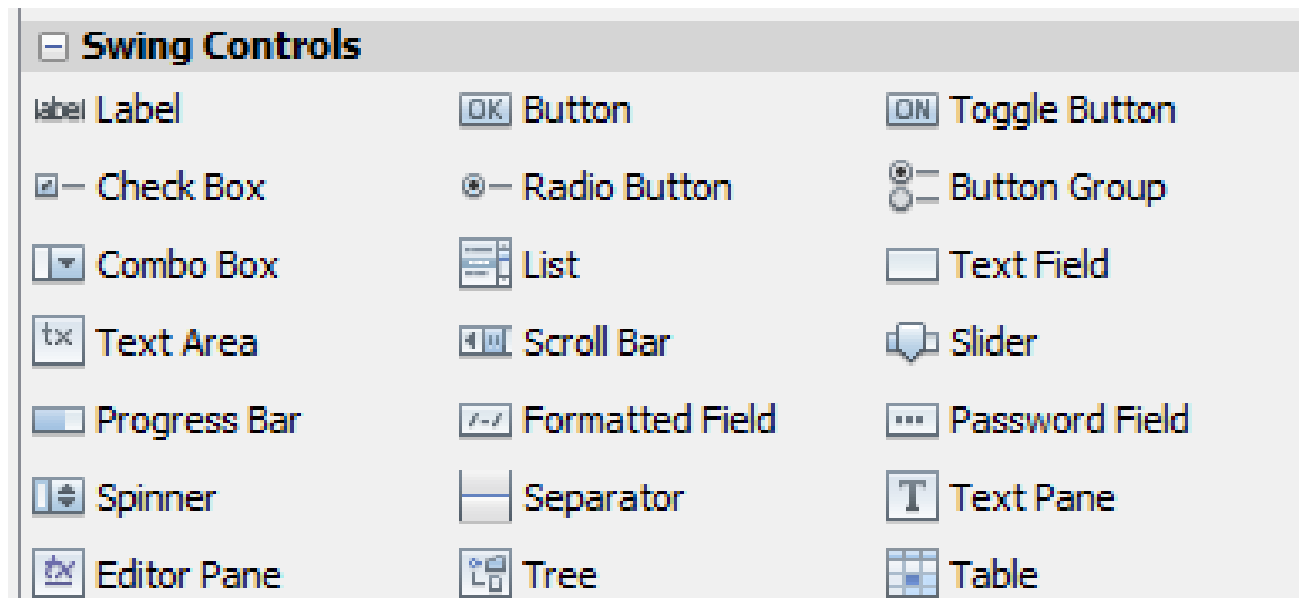
```
JButton jbtkOK = new JButton("OK");  
System.out.println(jbtkOK instanceof JButton);  
System.out.println(jbtkOK instanceof AbstractButton);  
System.out.println(jbtkOK instanceof JComponent);  
System.out.println(jbtkOK instanceof Container);  
System.out.println(jbtkOK instanceof Component);  
System.out.println(jbtkOK instanceof Object);
```

True, para todos los casos

Clases componentes GUI



JComponents- Controles



Tamaño, posición y alineación

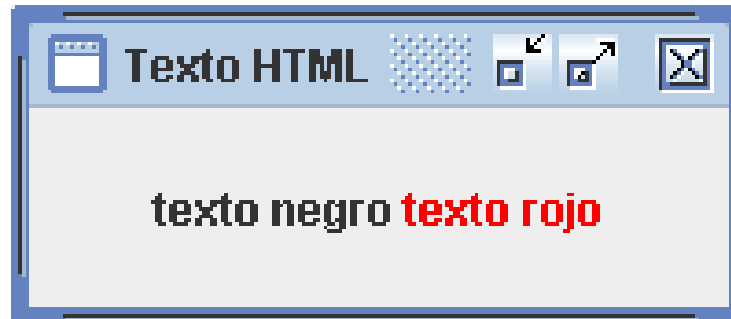
- Las siguientes propiedades controlan el tamaño y la posición de un control
 - `location`: (Point) Posición del control
 - `size`: (Dimension) Tamaño del control
 - `minimumSize`: Tamaño mínimo
 - `maximumSize`: Tamaño máximo
 - `preferredSize`: Tamaño preferido
 - `bounds`: (Rectangle) Posición y tamaño
 - `alignmentX` / `alignmentY`: (numero entre 0 y 1) posición relativa del control dentro del contenedor
- El efecto de estas propiedades depende del layout manager

Etiquetas

- Objetos **JLabel**
- Las etiquetas pueden tener texto y/o icono
- Algunas Propiedades:
 - **text**
 - **icon / disabledIcon**

Utilizar HTML

- En las propiedades de texto de los controles Swing se puede incluir HTML
 - ```
label.setText(
 "<html>texto negro " +
 "" +
 "texto rojo</html>"
);
```





# Iconos

- Los iconos se utilizan para proporcionar imagen a diversos controles
- Interfaz **Icon**
  - **iconHeight / iconWidth**
  - **paintIcon(Component, Graphics, x, y)**
- Clase **ImageIcon**
  - Propiedades: **image, description**
  - Se construye a partir de un objeto Image, ruta a fichero, URL o un array de bytes con los datos de la imagen
    - Soporta JPEG, PNG, GIF y XBM

# Botones

- `AbstractButton`: clase base de todos los botones
  - `JButton`: botón estándar
  - `JToggleButton`: botón bi-estado
    - `JCheckBox`: casilla de verificación
    - `JRadioButton`: botón de opción
- Propiedades:
  - **`Text`, `icon`, `pressedIcon`, `disabledIcon`, `rolloverIcon`, `selectedIcon`**

# ButtonGroup

- La clase ButtonGroup agrupa lógicamente varios botones (normalmente JRadioButton)
  - Sólo uno de los botones puede estar seleccionado a la vez
  - No es un control visual
- Propiedades:
  - **buttonCount, elements, selection....**
- Métodos:
  - **add(AbstractButton) / remove(AbstractButton)**
  - **isSelected(ButtonModel) / setSelected(ButtonModel, boolean)**

# Controles de texto

- JTextField: campo de texto
- JPasswordField: campo de texto oculto
- JFormattedTextField: campo de texto con patrón (e-mail, cuentas bancarias,...)
- JTextArea: área de texto multilínea
- JEditorPane: Como JTextArea pero admite distintos formatos de letra, html y rtf por defecto, se pueden meclar fuentes,colores e imagenes
- JTextPane: Más complejo que el anterior..

Más info: <http://www.chuidiang.com/java/ejemplos/JEditorPane-JTextPane/JEditorPane-JTextPane.php>

# JTextComponent

- Clase base de los controles de texto
- Propiedades:
  - text, selectedText, selectionStart, selectionEnd, editable, margin, disabledTextColor, selectedTextColor....
- Métodos
  - select(start,end), selectAll(), replaceSelection(nuevoTexto), getText(inicio,longitud)...

# JTextField

- Propiedades:
  - columns
  - horizontalAlignment
- JPasswordField
  - Propiedades:
    - echoChar, getPassword()

# JTextArea

- Propiedades:
  - columns, rows...
- Métodos
  - append(texto)
  - insert(texto,pos)
  - replaceRange(texto,inicio,fin)
  - ....

# Practica controles básicos

- Formulario de entrada de datos
  - Nombre
  - CheckBox Mayor de edad
  - NIF
  - Estado civil Botones de opcion (Soltero, casado, viudo,...)



# Clases Auxiliares

---

# Clases auxiliares

- No son subclases de Component
- Son usadas para describir propiedades de componentes GUI, están en el paquete `java.awt`

**Nota.** Los componentes Swing no reemplazan todas las clases AWT, sólo las clases de componentes GUI de AWT (como `Button`, `TextField`, `TextArea`,...)

# Clases auxiliares (cont.)

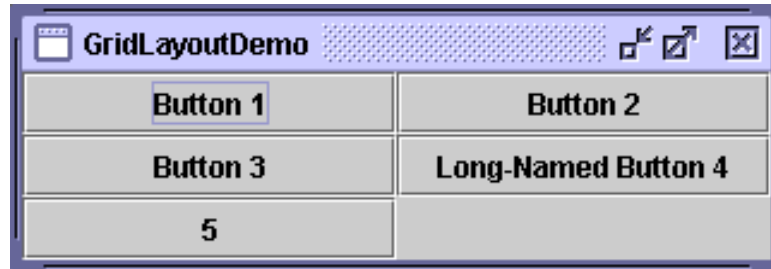
- Graphics
  - Clase abstracta que proporciona un contexto gráfico para dibujar líneas y figuras simples
- Color
  - Trata los colores de los componentes GUI. Útil para especificar fondos, color de letras o de líneas, figuras, etc.
- Font
  - Especifica tipo, estilo y tamaño de letras mostradas en componentes GUI
- FontMetrics
  - Clase abstracta utilizada para obtener propiedades de los tipos de letra
- Dimension
  - Encapsula el largo y ancho de un componente (en precisión entera)
- LayoutManager
  - Es una interface que especifica como son alineados los componentes dentro de un contenedor

# Layout Managers

- Los controles obtienen su posición y tamaño de acuerdo al gestor de disposición (layout manager) que utilice su contenedor
  - FlowLayout
  - BorderLayout
  - CardLayout
  - GridLayout
  - GridBagLayout
  - BoxLayout
  - SpringLayout
  - Null
- `Container contentPane = frame.getContentPane();`  
`contentPane.setLayout(new FlowLayout());`
- Más info:  
[http://chuwiki.chuidiang.org/index.php?title=Uso\\_de\\_Layouts](http://chuwiki.chuidiang.org/index.php?title=Uso_de_Layouts)

# GridLayout

- Alinea los controles en forma de tabla



- Parámetros del constructor:
  - **rows, columns:** N° de filas / columnas
    - Si se indica 0, no hay límite (tantas como controles se añadan)
  - **hgap, vgap:** Separación horizontal y vertical entre los controles

# FlowLayout

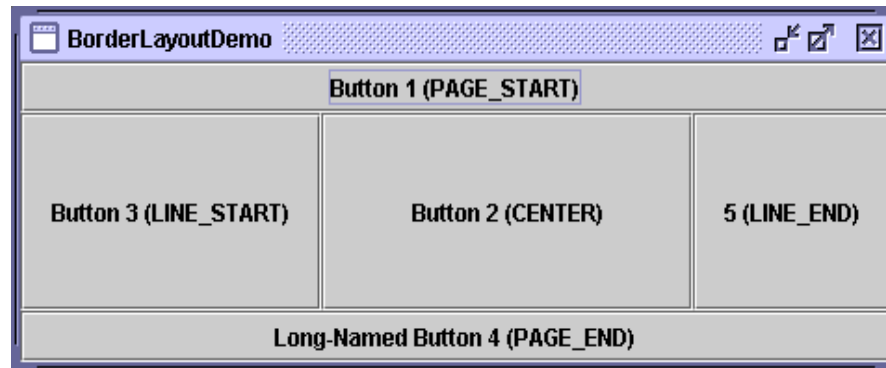
- Los controles se disponen de izquierda a derecha como en un procesador de textos
  - Si no caben, ocupan varias líneas



- Parámetros del constructor:
  - **align**: Alineación de los controles
    - LEFT, RIGHT, CENTER
  - **hgap**: Separación entre los controles de una línea
  - **vgap**: Separación entre líneas

# BorderLayout

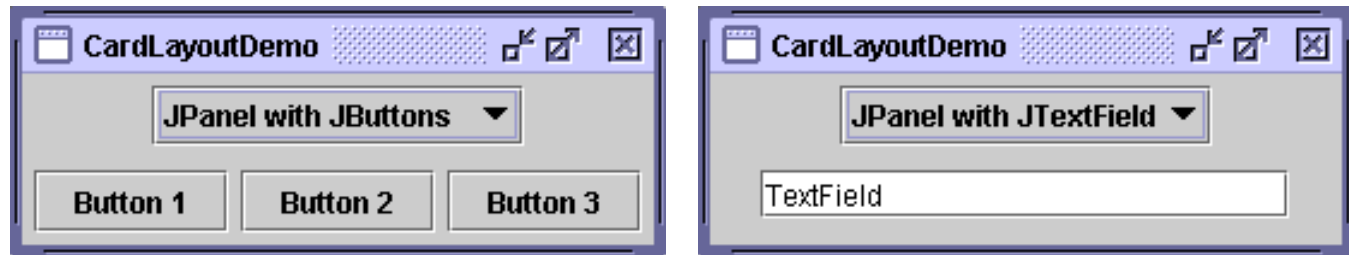
- Los controles se alinean a los bordes de la ventana, ocupando todo el espacio libre



- Parámetros del constructor:
  - **hgap, vgap**: Separación horizontal / vertical entre los controles
- Parámetro de add(): Alineación del control
  - BorderLayout.**NORTH** / BorderLayout.**PAGE\_START**
  - BorderLayout.**SOUTH** / BorderLayout.**PAGE\_END**
  - BorderLayout.**WEST** / BorderLayout.**LINE\_START**
  - BorderLayout.**EAST** / BorderLayout.**LINE\_END**
  - BorderLayout.**CENTER**

# CardLayout

- Los controles se muestran uno cada vez, como páginas
  - Normalmente, se añaden como páginas paneles con varios controles

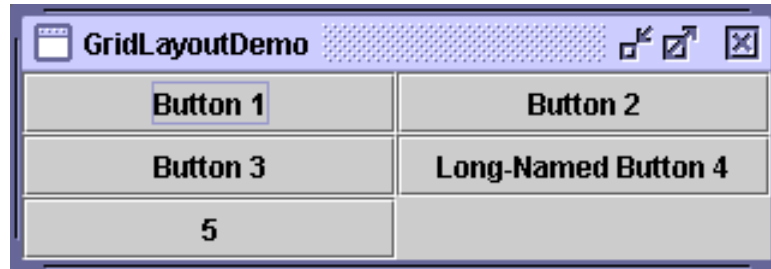


- Parámetros del constructor:
  - **hgap** / **vgap**: Márgen horizontal y vertical para los controles página
- Parámetro de `add()`: nombre de la página
- Métodos:
  - **first(Container)**, **previous(Container)**, **next(Container)**, **last(Container)**: Navegación entre las páginas del contenedor
  - **show (Container, name)**: Muestra la página del contenedor que tiene el nombre indicado



# GridLayout

- Alinea los controles en forma de tabla



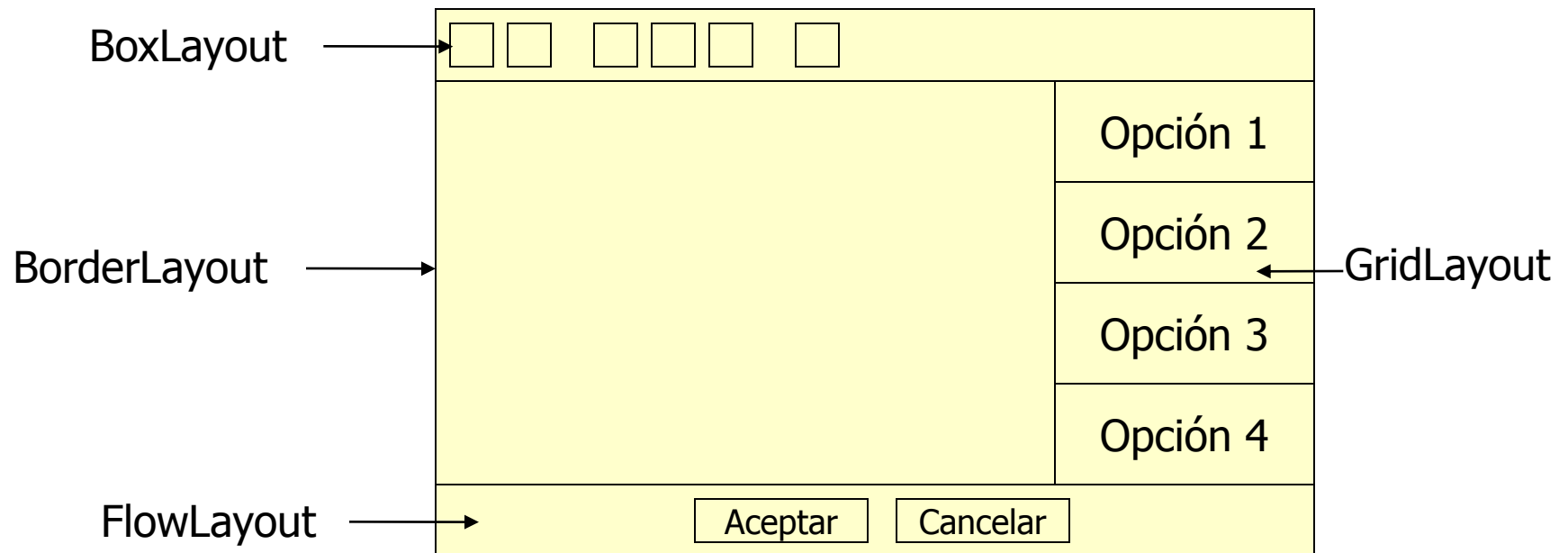
- Parámetros del constructor:
  - **rows, columns:** N° de filas / columnas
    - Si se indica 0, no hay límite (tantas como controles se añadan)
  - **hgap, vgap:** Separación horizontal y vertical entre los controles

# Null Layout

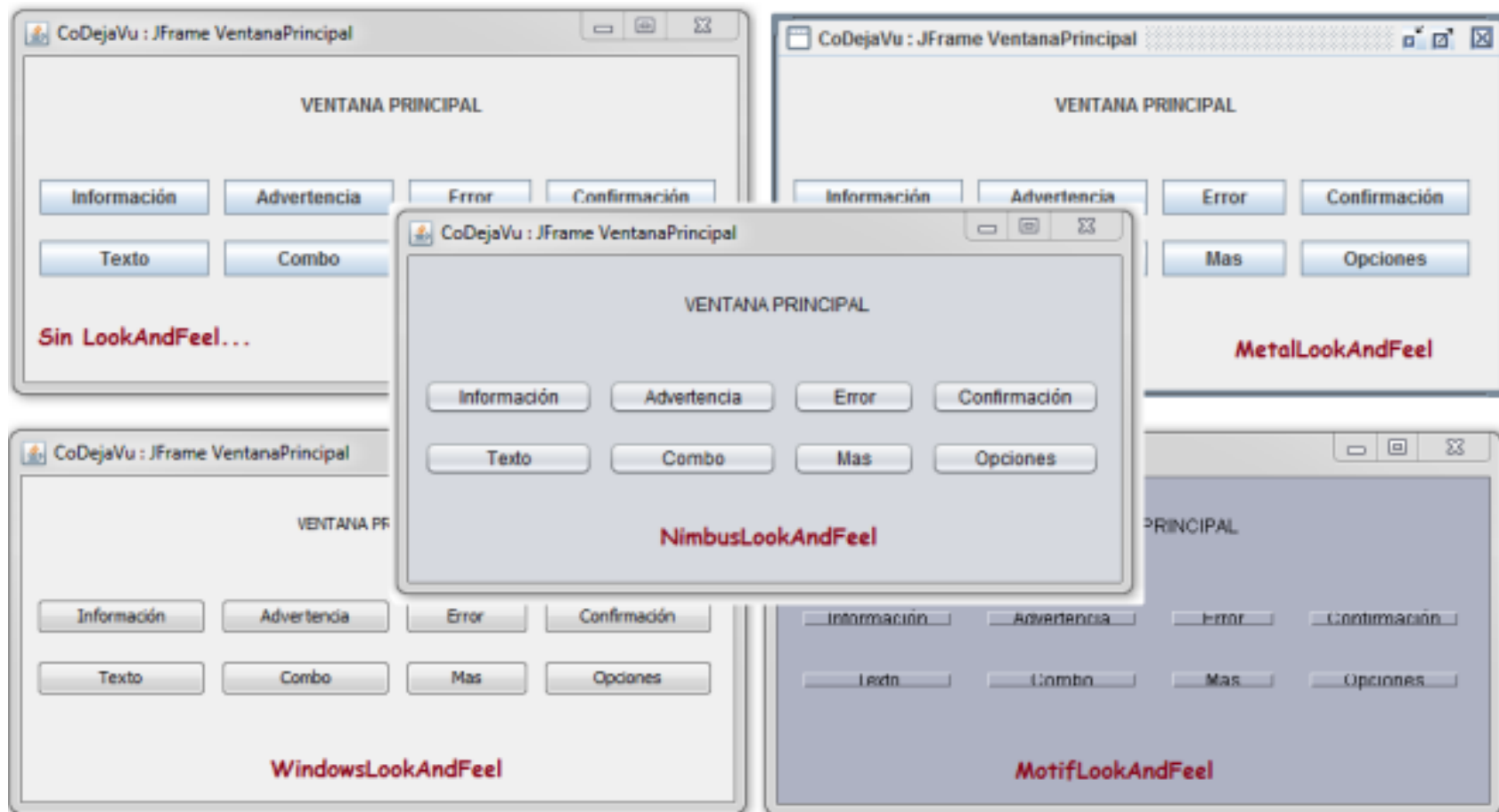
- Si no se especifica layout, los controles se colocan utilizando posiciones absolutas
  - Propiedades location y size
- Más sencillo para construir ventanas de tamaño fijo

# Combinar layouts

- Normalmente, una ventana se compone de varios paneles cada uno con el layout más apropiado



# Look & Feel



# Look & Feel

- <http://codejavu.blogspot.com.es/2014/05/ejemplo-look-and-feel-en-java.html>

- Utilización:

`UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel");`

# Eventos

---

# Programación dirigida por eventos

- Imaginen un programa para imprimir los primeros 100 números de la serie de Fibonacci o un programa para recibir dos números y escribir el resultado de su suma

**¿quién controla que se ejecutará primero y que después?**

- Ahora imaginen Microsoft Word con un nuevo documento abierto

**¿quién controla la secuencia de ejecución?**

# Programación dirigida por eventos (cont)

- Cada vez que el usuario escribe un carácter, oprime un botón del mouse, hace un movimiento con el cursor del mouse, presiona una combinación de teclas, ocurre un evento.
- El objeto que recibe el evento (un botón, un área de texto, un panel, una lista, entre otros), es notificado en tiempo de ejecución de que recibió el evento.
- Todo lo que se debe hacer es implementar la interfaz apropiada (event handler) y registrarla como un escucha (event listener) en el componente GUI (event source u objeto que va a recibir el evento) apropiado



# Manejando eventos

- Para cada tipo de evento XXX existe:
  - Un objeto XXXEvent, que contiene la información que se genera con el evento
  - Un interfaz XXXListener que se debe implementar para atender el evento (gestor de evento)
  - Dos métodos addXXXListener y removeXXXListener en los controles que soportan el evento para añadir y quitar gestores de eventos a los event sources
- Ejemplo (para pulsación de botones, menús, etc...):
  - ActionEvent
  - ActionListener
  - addActionListener
  - removeActionListener

# Tipos de escuchadores

- Los eventos están agrupados de acuerdo a su naturaleza en los siguientes grupos:
  - **ActionListener**: acciones sobre componentes.
  - **WindowListener**: cierre o manipulación una ventana (Frame/Dialog).
  - **MouseListener**: presión de un botón del mouse mientras el cursor está sobre el componente.
  - **MouseMotionListener**: movimiento del cursor sobre un componente.
  - **ComponentListener**: visibilidad del componentes.
  - **FocusListener**: obtención del foco del teclado.
  - **ListSelectionListener**: selección de ítems dentro de una lista.

# Tipos de escuchadores

- Cuando un usuario hace click sobre un botón o presiona la tecla Return mientras digitaba en un textfield o escoje una opción de un menú, se genera un evento, que tiene los siguientes elementos:
  - La fuente del evento (event source): Es el componente que origina el evento.
  - El escuchador: (event listener) es el encargado de atrapar o escuchar el evento.
  - El manejador del evento (event handler), es el método que permite implementar la interfaz, es decir el ejecutor!!. Este método:
    - Recibe un objeto evento (ActionEvent) el cuál tiene información sobre el evento que sucedió,
    - Descifra el evento, con dicho objeto, y
    - Procesa lo solicitado por el usuario.

# Ejemplo manejo de eventos

```
public class PrimerSwing {
 private static void mostrarVentana() {
 JFrame frame = new JFrame("Primer ventana");
 frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 JLabel label = new JLabel("Hello world");
 frame.getContentPane().add(label);
 frame.setSize(300, 300);
 JButton boton = new JButton("Haz click");
 frame.getContentPane().add(boton);
 frame.setVisible(true);
 boton.addActionListener(new ActionListener() {
 public void actionPerformed(ActionEvent ae) {
 JButton boton=(JButton) ae.getSource();
 boton.setText("he cambiado");
 }
 });
 }

 public static void main(String[] args) {
 mostrarVentana();
 }
}
```

Víctor Custodio, 2016

# Manejando eventos

Para atender un evento se debe implementar el interfaz

Listener:

- En una clase externa
  - El número de clases crece muy rápido
- En la propia clase ventana
  - Es complicado distinguir eventos del mismo tipo que provengan de distintos controles
- En una clase interna
  - Es la solución más utilizada
  - El código puede resultar complejo
  - Se suelen utilizar clases internas anónimas

# Implementar en Listener en la propia clase

```
public class SegundoSwing implements ActionListener {
 private static int numero_clicks = 0;
 private static void mostrarVentana() {
 JFrame frame = new JFrame("Nuestra primera ventana");
 frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 JButton boton = new JButton("Comienza a dar click!");
 frame.getContentPane().add(boton);
 frame.setVisible(true);
 boton.addActionListener(this);
 }
 public void actionPerformed(ActionEvent ae) {
 JButton boton=(JButton) ae.getSource();
 boton.setText("he cambiado");
 }
 public static void main(String[] args) {
 mostrarVentana();
 }
}
```

# Ejercicio

- Modifica el ejemplo anterior para realizar algunos cambios más cuando se clickea el botón:
  - Cambia la fuente de letra del botón
  - Cambia el título del frame

# Manejando eventos

- Algunos interfaces Listener contienen muchos métodos
  - Por ejemplo, el interfaz WindowListener incluye:
    - windowOpened, windowClosing, windowClosed, windowIconified, windowDeiconified, windowActivated, windowDeactivated
- En estos casos, el JDK incluye una clase XXXAdapter que realiza una implementación por defecto de todos los métodos
  - Sólo se sobrescriben los métodos que interesa



# Manejando eventos

- Los gestores o handlers de eventos se suelen implementar utilizando clases anónimas

- Al añadir el listener, se utiliza la sintaxis:

- `addXXXListener(new XXXListener() {  
 // Cuerpo de la clase anónima  
});`

```
JFrame ventana = new JFrame("Prueba de eventos");

ventana.addWindowListener(new WindowAdapter() {
 public void windowClosing(WindowEvent evt){
 System.exit(0);
 }
});
```

# Actividad

- Implementar una calculadora básica (+, -, \*, /, =)