

## Construcción de GUI en Java



- Tecnología Swing
  - NetScape (IFC) , IBM, Lighthouse Design
- JFC
  - AWT, Java 2D, Accessibility, Drag and Drop, Swing
- Cambios importantes desde la versión 1.1 a 1.2
  - Versión 1.1 AWT
  - Versión 1.2 en adelante incluyen JFC: SWING
  - Actualmente, los navegadores no entienden JFC
- Swing está apoyado en parte en AWT
- AWT Abstract Window Toolkit
  - La librería se encuentra en el paquete java.awt

1

## Elementos de Swing



- Componentes y contenedores
  - Componentes. Aspecto visible del interfaz
    - botones, etiquetas, campos de texto, etc
    - Se sitúan dentro de algún contenedor
  - Contenedores. Almacenes de componentes
    - Pueden contener a otros contenedores
    - Dos tipos
      - Superiores: JApplet, JFrame, JDialog,
      - Intermedios: JPanel, JScrollPane, JSplitPane, JTabbedPane, JToolBar y otros más especializados

3

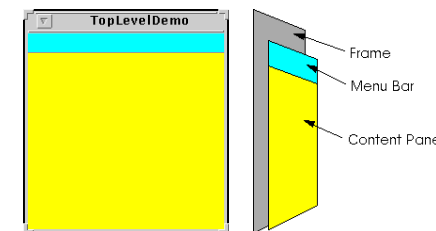
## AWT v SWING



- Por cada elemento de AWT existe un elemento en el sistema operativo que lo representa
  - El resultado final dependerá de este elemento
- Problema:
  - Hay facilidades que algún sistema operativo no tiene por lo que AWT define lo mínimo común
- Swing elimina este problema
  - Define lo máximo
  - Necesita los paquetes (y subpaquetes)
    - java.awt.\* y javax.swing.\*

2

## Contenedores superiores I

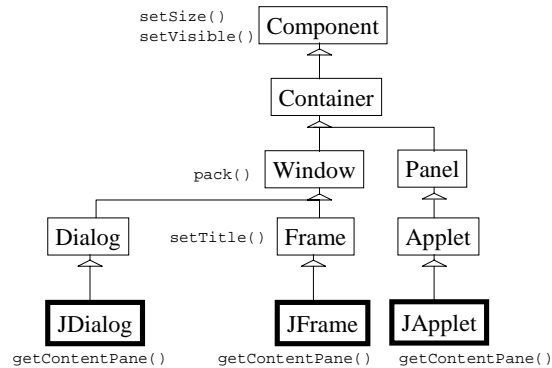


- Disponen de un panel de contenidos (ContentPane)
- Pueden opcionalmente disponer de un menú

```
Container cpane = unaFrame.getContentPane();  
unaFrame.setContentPane(unPanel);  
unaFrame.setJMenuBar(unMenuBar);
```

4

## Contenedores Superiores II



5

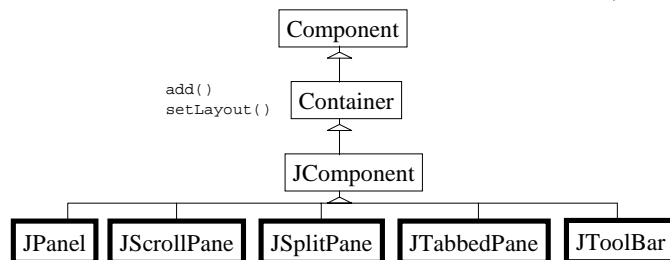
## Construcción de un GUI. Pasos a seguir



- I. Crear un contenedor superior y obtener su contenedor intermedio
- II. Seleccionar un gestor de esquemas para el contenedor intermedio
- III. Crear los componentes adecuados
- IV. Agregarlos al contenedor intermedio
- V. Dimensionar el contenedor superior
- VI. Mostrar el contenedor superior

7

## Contenedores Intermedios



- El contenedor más utilizado es JPanel

6

## I. Crear un contenedor superior



- Hay tres clases contenedores superiores
  - JFrame, JDialog, JApplet
    - JFrame -> Aplicación
      - Ventana de nivel superior con bordes y título
      - setTitle(), getTitle(), setIconImage()
    - JApplet -> Applets
    - JDialog -> Diálogos
  - Métodos de instancia ...
 

```

void pack()
Container getContentPane()
void setContentPane(Container)
void setJMenuBar(Menu)
                    
```

8

## II. Gestor de esquemas para cont. intermedio



- Determinan como encajan los componentes dentro de los contenedores
  - Cada contenedor tiene un gestor propio
    - Por defecto JPanel tienen BorderLayout
  - Los gestores (son clases) existentes son:
    - FlowLayout, BorderLayout, GridLayout, GridBagLayout, CardLayout, BoxLayout, ...
  - Puede no utilizarse el gestor y colocar los elementos con setPosition() (no recomendable)
- Para asignar un gestor de esquemas  
`contenedor.setLayout(new FlowLayout())`

9

## IV. Agregar componentes al contenedor



- Se hace a través del método add() de los contenedores

```
JFrame f = new JFrame("Un ejemplo");
Container cpane = f.getContentPane();
cpane.setLayout(new FlowLayout());
JButton bSi = new JButton("SI");
JButton bNo = new JButton("NO");
JLabel l = new JLabel("Nombre");
cpane.add(l);
cpane.add(bSi);
cpane.add(bNo);
```

  - El orden es importante
- A un contenedor intermedio también se le pueden agregar otros contenedores intermedios

11

## III. Crear componentes



- Cada componente viene determinado por una clase
- Hay que crear un objeto de esa clase

```
JButton bSi = new JButton("SI");
JButton bNo = new JButton("NO");
JLabel l = new JLabel("Nombre");
...
```

10

## V. Dimensionar el contenedor superior I



- Especifica el tamaño del contenedor superior
- El método a llamar es  
`void setSize(int anchura, int altura)`

```
JFrame f = new JFrame("Un ejemplo");
....
f.setSize(int anchura, int altura)
```

12

## V. Dimensionar el contenedor superior II



- Una alternativa a utilizar el método `setSize()` es el método `pack()`, que calcula el tamaño de la ventana teniendo en cuenta
    - El gestor de esquemas
    - El número y orden de los componentes añadidos
    - La dimensión de los componentes (preferida)
      - `void setPreferredSize(Dimension)`
      - `void setMinimumSize(Dimension)`
      - `void setMaximumSize(Dimension)`
- ```
JFrame f = new JFrame("Un ejemplo");
....
f.pack();
```

13

## Ejemplo GUI



```
import java.awt.*;
import javax.swing.*;
class GUI01 {
    public static void main(String [] args) {
        JFrame f = new JFrame("Un ejemplo.");
        Container cpane = f.getContentPane();
        cpane.setLayout(new FlowLayout());
        JButton bSi = new JButton("SI");
        JButton bNo = new JButton("NO");
        JLabel l = new JLabel("Nombre");
        cpane.add(l);
        cpane.add(bSi);
        cpane.add(bNo);
        f.pack();
        f.setVisible(true);
    }
}
```

15

## VI. Mostrar el contenedor superior



- Para hacerlo visible o invisible se utiliza el método `setVisible(boolean)`
- Este método es válido para mostrar u ocultar componentes y contenedores

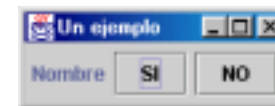
```
JFrame f = new JFrame("Un ejemplo");
....
f.setVisible(true)
```

14

## Ejemplo GUI



- `java GUI01`
  - Sólo las funciones de maximizar y minimizar, cambiar tamaño y mover están operativas
  - Los botones Si y No ceden cuando se pulsan pero no realizan ninguna acción
  - La ventana no se cierra normalmente
  - No tiene el aspecto de una ventana Windows



16



## Ejemplo GUI

- GUI01n
  - Realizar la ventana anterior como subclase de JFrame
    - En el constructor llamar a `super(String)`
    - Todo la interface gráfica se crea en el constructor
  - Luego, una clase con `main`
    - Crea un objeto de la clase
    - Hace `pack()` y lo muestra con `setVisible(true)`

17



## Iconos

- En algunos constructores y métodos aparece un argumento `Icon` que representa un icono
- `Icon` es una interface.
- Para cargar un icono desde un fichero

```
Icon i =  
    new ImageIcon("c:\\misIconos\\bruja.gif")  
  
– O bien  
ImageIcon i =  
    new ImageIcon("c:\\misIconos\\bruja.gif")
```

19



## GUI en Swing

- Queda por conocer:
  - Controlar el aspecto de la aplicación
    - Look and Feel
  - Usar adecuadamente los gestores de esquemas
  - Estudiar en detalle los componentes
  - Asociar acciones a los componentes

18



## Aspecto de la aplicación

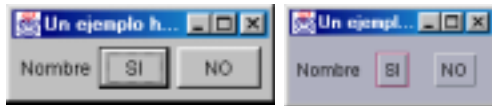
```
public static void main(String[] args) {  
    try {  
        UIManager.setLookAndFeel("Look and feel valido");  
    } catch (Exception e) {  
    }  
    ...//Trabajar normalmente ...  
}
```

- Posibles Look and Feel

```
"javax.swing.plaf.metal.MetalLookAndFeel"  
"com.sun.java.swing.plaf.windows.WindowsLookAndFeel"  
"com.sun.java.swing.plaf.motif.MotifLookAndFeel"  
"javax.swing.plaf.mac.MacLookAndFeel"
```

20

## Otros aspectos



21

## FlowLayout



- Los componentes fluyen de izquierda a derecha y de arriba a abajo.
- Su tamaño se ajusta al texto que presentan
- Al cambiar el tamaño de la ventana, puede cambiar la disposición



23

## Gestores de Esquemas



- Clases que determinan cómo se distribuirán los componentes dentro del contenedor.
- La mayoría definidos en `java.awt`
  - `FlowLayout`
  - `BorderLayout`
  - `GridLayout`
  - `GridBagLayout`
  - `CardLayout` (Swing propone alternativa)
  - `BoxLayout` (nueva en Swing: `javax.swing`)
- `JPanel` por defecto dispone de un `BorderLayout`

22

## BorderLayout



- Divide el contenedor en 5 partes
    - `NORTH`, `SOUTH`, `EAST`, `WEST` y `CENTER`
    - Los componentes se ajustan hasta rellenar completamente cada parte
    - Si algún componente falta, se ajusta con el resto (menos el centro si hay cruzados)
    - Para añadir al contenedor se utiliza una versión de `add` que indica la zona en la que se añade (Constantes definidas en la clase)
- ```
add(bSi, BorderLayout.NORTH)
```

24

## BorderLayout



Sin Norte ni Este



Ejercicio: Realizar este GUI  
GUI02.java

25

## BoxLayout



- Coloca a los componentes a lo largo de un eje.
  - Define dos constantes `X_AXIS`, `Y_AXIS`
- En el constructor debemos indicar
  - El contenedor y la orientación de los componentes `BoxLayout(Container, int)`
  - Los componentes no tienen igual tamaño (como en `GridLayout`)
- Existe la clase `Box` para facilitar la construcción
  - Es un `Container`
- El orden a la hora de agregar determina la posición (de izda a drcha y de arriba a abajo)
 

```
cpane.setLayout(new BoxLayout(this, BoxLayout.X_AXIS))
```

27

## GridLayout



- Divide al componente en una rejilla (grid)
- En el constructor debemos indicar el número de filas y de columnas
- Los componentes se mantienen de igual tamaño dentro de cada celdilla
- El orden a la hora de agregar determina la posición (de izda a drcha y de arriba a abajo)

```
cpane.setLayout(new GridLayout(2,3))
```

- Dos filas y tres columnas

26

## GUI complejos I



- Podemos utilizar un contenedor intermedio en lugar de un componente para agregarlo a otro contenedor intermedio
- Este nuevo contenedor intermedio podrá:
  - incorporar sus propios componentes
  - tener su propio gestor de esquemas

28

## GUI Complejos II



```
JFrame f = new JFrame("Un ejemplo.");
f.getContentPane().add(new BorderLayout());
JPanel p = new JPanel();
JButton bp1 = new JButton("Panel1");
JButton bp2 = new JButton("Panel2");
```

```
p.setLayout(new GridLayout(2,1));
p.add(bp1);
p.add(bp2);
```

...

```
f.getContentPane().add(p, BorderLayout.WEST);
```

...



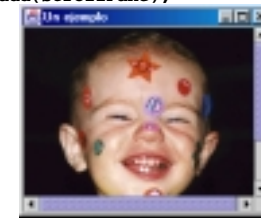
Ejercicio: jugar con setVisible  
GUI03.java

29

## JScrollPane II



```
import java.awt.*;
import javax.swing.*;
class GUIISP {
    public static void main(String [] args) {
        JFrame frame = new JFrame("Un ejemplo");
        ImageIcon ii = new ImageIcon("carapin.gif");
        JLabel label = new JLabel(ii);
        JScrollPane scrollPane = new JScrollPane(label);
        frame.getContentPane().add(scrollPane);
        frame.pack();
        frame.setVisible(true);
    }
}
```



31

## JScrollPane I



- Permite hacer scroll a un componente (u otro contenedor intermedio)

- Constructores

```
JScrollPane JScrollPane(JComponent);
```

```
JScrollPane JScrollPane(JComponent, int, int);
```

- Constantes para control del scroll

VERTICAL\_SCROLLBAR\_AS\_NEEDED

HORIZONTAL\_SCROLLBAR\_AS\_NEEDED

VERTICAL\_SCROLLBAR\_ALWAYS

HORIZONTAL\_SCROLLBAR\_ALWAYS

VERTICAL\_SCROLLBAR\_NEVER

HORIZONTAL\_SCROLLBAR\_NEVER

30

## JSplitPane I



- Divide una ventana en dos

- Vertical u horizontal
- Movimiento visible o no

- Constructores (entre otros)

```
SplitPane(int, JComponent, JComponent)
```

```
SplitPane(int, boolean, JComponent, JComponent)
```

- Constantes

HORIZONTAL\_SPLIT

VERTICAL\_SPLIT

- Métodos de instancia

```
setOneTouchExpandable(boolean);
```

```
setDividerLocation(int);
```

32



## JSplitPane II



```
import java.awt.*;
import javax.swing.*;
class GUI5IP {
    public static void main(String [] args) {
        JFrame frame = new JFrame("Un ejemplo");
        ImageIcon ii = new ImageIcon("carapin.gif");
        JLabel label1 = new JLabel(ii);
        JLabel label2 = new JLabel("Cara bonita");
        JSplitPane splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
                                              label1, label2);

        splitPane.setOneTouchExpandable(true);
        frame.getContentPane().add(splitPane);
        frame.pack();
        frame.setVisible(true);
    }
}
```

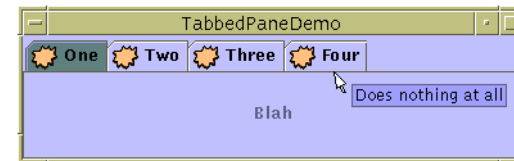


33

## JTabbedPane



```
ImageIcon icon = new ImageIcon("images/middle.gif");
JTabbedPane tabbedPane = new JTabbedPane();
Component panel1 = makeTextPanel("Blah");
tabbedPane.addTab("One", icon, panel1, "Does nothing");
tabbedPane.setSelectedIndex(0);
Component panel2 = makeTextPanel("Blah blah");
tabbedPane.addTab("Two", icon, panel2, "Does twice as much");
Component panel3 = makeTextPanel("Blah blah blah");
tabbedPane.addTab("Three", icon, panel3, "Still does nothing");
Component panel4 = makeTextPanel("Blah blah blah blah");
tabbedPane.addTab("Four", icon, panel4, "Does nothing at all");
```



35

## JTabbedPane I



- Permite simular carpetas sobre la ventana
- Constructores (entre otros)

```
JTabbedPane()
JTabbedPane(int)
```

- Constantes

```
TOP    BOTTOM    LEFT    RIGHT
```

- Métodos de instancia

```
addTab(String, Component)
addTab(String, Icon, Component)
addTab(String, Icon, Component, String)
setSelectedIndex(int);
```

34

## JToolBar I



- Crea un barra de botones
  - Debe incluirse en un contenedor con BorderLayout
  - Usualmente contiene botones con iconos
- Constructor
 

```
JToolBar()
JToolBar(int) // HORIZONTAL VERTICAL
```
- Métodos de instancia
 

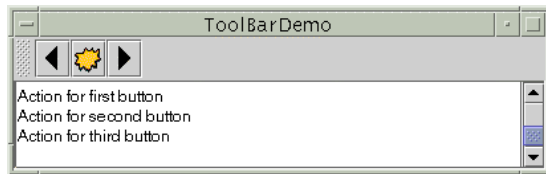
```
addSeparator()
setFloatable(boolean) // flota por defecto
```

36

## JToolBar II

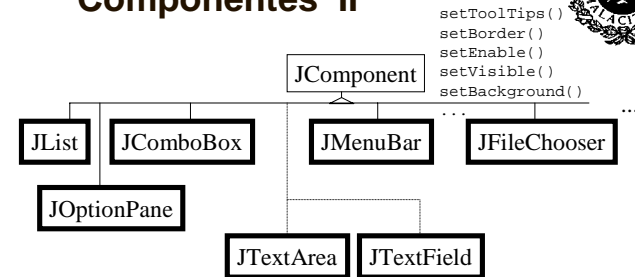


```
JToolBar toolBar = new JToolBar();
JButton button = null;
button = new JButton(new ImageIcon("images/left.gif"));
toolBar.add(button);
button = new JButton(new ImageIcon("images/middle.gif"));
toolBar.add(button);
button = new JButton(new ImageIcon("images/right.gif"));
toolBar.add(button);
pane.add(toolBar, BorderLayout.NORTH);
```



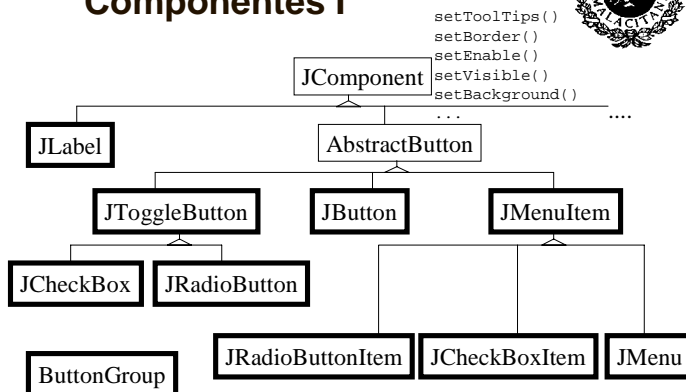
37

## Componentes II



39

## Componentes I



38

## Componentes III



### • Métodos heredados de JComponent

- Color getBackground()
- void setBackground(Color)
- Graphics getGraphics()
- String getName()
- Toolkit getToolkit()
- void setEnabled(boolean)
- void setVisible(boolean)
- void paint(Graphics g)
- void repaint()
- void setBorder()
- ...

40

## Bordes I



- En `javax.swing.borders` existen una serie de clases que permiten dar un borde a un componente. Hay nueve clases:
  - `AbstractBorder`
  - `BevelBorder`
  - `CompoundBorder`
  - `EmptyBorder`
  - `EtchedBorder`
  - `LineBorder`
  - `MatteBorder`
  - `SoftBevelBorder`
  - `TitleBorder`

41

## JButton



- Crea botones de pulsación
- Constructores
  - `JButton()`
  - `JButton(String)`
  - `JButton(String, Icon)`
  - `JButton(Icon)`
- Métodos
  - `String getText()`
  - `void setText(String)`
  - ...

43

## Bordes II



- Para cambiar el borde de un componente
  - `public void setBorder(Border)`
- La clase `javax.swing.BorderFactory` tiene métodos de clase para crear bordes
  - Múltiples peticiones de crear un mismo borde devuelven el mismo borde.

```
JButton b = new JButton("Aceptar");  
b.setBorder(BorderFactory.createTitleBorder("Boton"))
```

42

## JLabel



- Es una etiqueta con una línea de texto.
- Constructores
  - `JLabel([String,] [Icon,] [int])`
- Constantes
  - `LEFT`                      `RIGHT`                      `CENTER`
- Métodos de instancia
  - `String getText()`
  - `void setText(String)`
  - ...

44

## JCheckBox



- Marcadores
- Constructores  
`JCheckBox([String,] [Icon,] [boolean])`
- Métodos de instancia  
`String getText()`  
`void setText(String)`  
`boolean isSelected()`  
`void setSelected(boolean)`  
...

45

## Ejemplo con botones I



```
import java.awt.*;
import javax.swing.*;
class GUI04 {
    public static void main(String [] args) {
        JFrame f = new JFrame("Ejemplo de Botones");

        JButton bNorte = new JButton("Norte");
        JLabel lSur = new JLabel("Este es el Sur",
                                JLabel.CENTER);
        JCheckBox cEste = new JCheckBox("Este", true);
        JButton bCentro = new JButton("Centro");
        JRadioButton cp1 = new JRadioButton("RB1");
        JRadioButton cp2 = new JRadioButton("RB2", true);

        ButtonGroup gcb = new ButtonGroup();
        gcb.add(cp1);
        gcb.add(cp2);
```

47

## JRadioButtons y ButtonGroup



- Botones circulares
- Se agrupan de manera que sólo uno esté pulsado
- Constructores  
`JRadioButtons([String,] [Icon,] [boolean])`
- Métodos de instancia  
Igual que `JCheckBox`
- Para agruparlos, se crea una instancia de `ButtonGroup` y se añaden con `add(AbstractButton)`

46

## Ejemplo con botones II



```
JPanel prb = new JPanel();
prb.setLayout(new GridLayout(2,1));
prb.add(cp1);
prb.add(cp2);

Container contP = f.getContentPane();
contP.add(bNorte, BorderLayout.NORTH);
contP.add(lSur, BorderLayout.SOUTH);
contP.add(cEste, BorderLayout.EAST);
contP.add(prb, BorderLayout.WEST);
contP.add(bCentro, BorderLayout.CENTER);
f.pack();
f.setVisible(true);
}
```



48



## JTextField

- Permite editar texto en una línea.
- Constructores

```
JTextField ([String,] [int])
```
- Métodos

```
String getText()
String getText(int,int) // offset y len
void setEditable(boolean)
boolean isEditable()
...
```

  - Tiene una subclase que enmascara el eco (\* u otro símbolo)

```
JPasswordField
```
  - con un método de instancia

```
char [] getPassword()
```

49



## JList I

- Muestra una lista de elementos para su selección.
- Constructores

```
JList()
JList(Vector)
JList(Object [])
JList(ListModel)
```
- Métodos de instancia

```
int getSelectedIndex() // -1 si no hay
int [] getSelectedIndices()
Object getSelectedValue()
Object [] getSelectedValues()
boolean isSelectedIndex(int)
boolean isSelectionEmpty([])
void setListData(Object)
void setListData(Vector)
```

51



## JTextArea

- Permite editar texto en un area.
- Constructores

```
JTextArea ([String,] [int,int])
```
- Métodos

```
void append(String)
void insert(String,int)
igual que JTextField
...
```

50



## JList II

- Métodos de instancia

```
void setSelectionMode(int)
get selectionMode()
...
```
- Constantes

```
ListSelectionModel.SINGLE_SELECTION
ListSelectionModel.SINGLE_INTERVAL_SELECTION
ListSelectionModel.MULTIPLE_INTERVAL_SELECTION
```

52

## JComboBox



- Permite la selección de un ítem de entre varios.
- No está desplegado como JList
- Constructores
  - `JComboBox()`                      `JComboBox(Object [])`
  - `JComboBox(Vector)`        `JComboBox(ListModel)`
- Métodos de instancia
  - `int getSelectedIndex()`
  - `Object getSelectedItem()`
  - `void setSelectedIndex(int)`
  - `boolean isEditable()`
  - `void setEditable(boolean)`

53

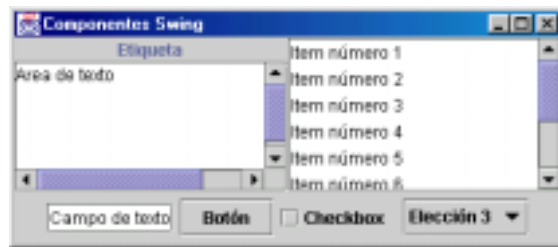
## JDialog



- Es un elemento de visualización al igual que Frame
  - Se suele crear y no visualizar hasta que sea necesario `setVisible(true)`
  - Para ocultarla `setVisible(false)`
  - Para eliminarla `dispose()`
- `JDialog(Frame, String, boolean)`
  - Frame es la ventana padre
  - String, el título
  - boolean indica si es modal o no
  - `new JDialog(f, "Ventana modal", true);`
  - `Container getContentPane()`

55

## Ejercicio



GUIcomp.java

54

## JOptionPane



- Clase que contiene métodos de clase para crear distintas ventanas de mensajes (modales)
- Métodos de clase
  - `showConfirmDialog(...)`
    - Realiza una pregunta de confirmación como Si, No Cancelar
  - `showInputDialog(...)`
    - Espera una entrada
  - `showMessageDialog(...)`
    - Informa de algo que ha ocurrido
  - `showOptionDialog(...)`
    - Unifica las tres anteriores.

56

## JOptionPane II



- Argumentos de los métodos `showXXXDialog(...)`

<code>Component padre</code>	Puede ser null
<code>Object mensaje</code>	Usualmente un <code>String</code>
<code>String titulo</code>	De la ventana
<code>int tipoOpcion</code>	
<code>ERROR_MESSAGE</code>	<code>INFORMATION_MESSAGE</code>
<code>WARNING_MESSAGE</code>	<code>QUESTION_MESSAGE</code>
<code>PLAIN_MESSAGE</code>	
<code>int tipoMensaje</code>	
<code>DEFAULT_OPTION</code>	<code>YES_NO_OPTION</code>
<code>YES_NO_CANCEL_OPTION</code>	<code>OK_CANCEL_OPTION</code>
<code>Icon icono</code>	Hay uno por defecto
<code>Object [] opciones</code>	
<code>Object valorInicial</code>	

57

## JOptionPane IV



```
showConfirmDialog(Component padre,  
                  Object mensaje,  
                  String title,  
                  int optionType,  
                  int messageType,  
                  Icon icon)  
  
Hay más constructores para este tipo de ventanas  
JOptionPane.showConfirmDialog(  
    null,  
    "Esta seguro",  
    "Ventana de Seguridad",  
    JOptionPane.YES_NO_OPTION);
```

59

## JOptionPane III



- Valores devueltos por los métodos `showXXXDialog(...)`

<code>YES_OPTION</code>
<code>NO_OPTION</code>
<code>CANCEL_OPTION</code>
<code>OK_OPTION</code>
<code>CLOSED_OPTION</code>

58

## JFileChooser



- Es un elemento de visualización al igual que permite la selección de un fichero
  - Manipula nombres de ficheros
- Constructores

```
JFileDialog()  
JFileDialog(String)    // path  
JFileDialog(File)
```
- Métodos de instancia

```
int showOpenDialog(Component)  
int showCloseDialog(Component)  
File getSelectedFile()  
File [] getSelectedFiles()
```

60

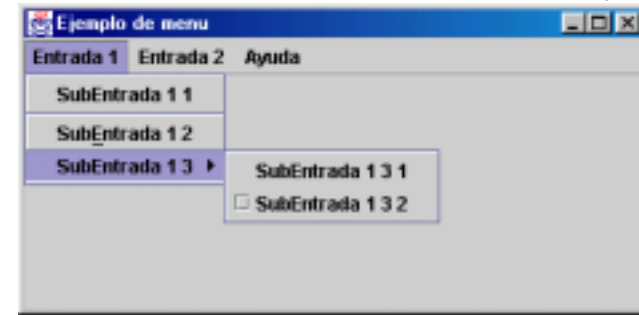
## Menús



- Se pueden añadir a los contenedores superiores
- Para añadir un menú
  - `void setJMenuBar(JMenuBar)`
- Tres elementos básicos
  - Barra de Menú (`JMenuBar`)
  - Entrada de Menú (`JMenu`)
  - Item de entrada (`JMenuItem` y ....)
- El menú de ayuda se añade a un `JMenuBar` (aún no está implementado)
  - `void setHelpMenu(JMenu)`

61

## Ejemplo de Menú



GUI07.java

63

## Menús



- Un item puede ser a su vez un menú
  - Para añadir a un `JMenuBar` una entrada
    - `void add(JMenu)`
  - Para añadir a un `JMenu`
    - `void add(JMenuItem)`
    - `void add(JMenuItem, MenuShortcut)`
    - `void addSeparator()`
  - Para manejar los items y entradas
    - `void setEnabled(boolean)`
    - `boolean isEnabled()`
- Un `CheckboxMenuItem` se puede seleccionar
  - `boolean getState()`
  - `void setState(boolean)`

62

## KeyEvent



- La clase `KeyEvent` incluye constantes para acelerar las pulsaciones para llegar a un de un `JMenuItem`
  - `VK_1, VK_2, ...`
  - `VK_A, VK_B, ...`
  - `VK_F1, VK_F2, ...`
- La clase se encuentra en el paquete `java.awt.event`
  - `add(new JMenuItem("Entrada", KeyEvent.VK_E));`

64



## PopupMenu



- Crea menús aislados (en cualquier ventana)
  - `JPopupMenu()`
- Contiene elementos de menu
  - `add(MenuItem)`
- Debe activarse en un componente dada una posición de visualización
  - `show(JComponent, int x, int y)`

65

## El Modelo de Eventos



- Un componente (o menú componente) puede disparar un evento  
`java.awt.event`      `javax.swing.event`
- Cuando un evento se dispara, es recogido por objetos “oyentes” (listeners) que realizan la acción apropiada
- Cada oyente debe pertenecer a una clase que implemente cierta interface dependiendo del evento

67

## El Modelo de Eventos



- La versión 1.0 incluía un modelo de eventos diferente
- Desde la versión 1.1 incluye el nuevo modelo de eventos que se mantiene como definitivo en 1.3 (incluso para la librería Swing que añade algo)
- Sólo vamos a ver el modelo nuevo

66

## El Modelo de Eventos



- Para que un oyente esté pendiente de un componente, se debe registrar en él
- El registro es realizado a través de un método del componente sobre el que se registra
  - `addXxxxxListener(XxxxxListener)`
    - El receptor es el componente que queremos escuchar
    - El argumento será el objeto oyente
    - `XxxxxListener` indica la interface que va a implementar
  - Por ejemplo, dado la interface `ActionListener`, un objeto se registra por medio de
    - `addActionListener(ActionListener)`

68

## Interfaces en java.awt.event I



Interfaces	Métodos
ActionListener	actionPerformed(ActionEvent)
AdjustmentListener	adjustmentValueChanged(AdjustmentEvent)
ComponentListener	componentHidden(ComponentEvent) componentMoved(ComponentEvent) componentResized(ComponentEvent) componentShown(ComponentEvent)
ContainerListener	componentAdded(ContainerEvent) componentRemoved(ContainerEvent)
FocusListener	focusGained(FocusEvent) focusLost(FocusEvent)
ItemListener	itemStateChanged(ItemEvent)
KeyListener	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)

69

## Eventos



- Los eventos se implementan como subclases de `java.util.EventObject`  
`Object getSource()`
  - Los eventos se encuentran en los paquetes  
`java.awt.event` y `javax.swing.event`
    - Interfaces `XxxxxListener`
    - Evento `EventXxxxx`
  - Como ya hemos dicho, para registrar a un oyente se utiliza
    - Registro `addXxxxxListener(XxxxxListener)`
  - Todos los métodos de la interface tendrán como argumento
    - Evento `EventXxxxx`

71

## Interfaces en java.awt.event II



Interfaces	Métodos
MouseListener	mouseClicked(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mousePressed(MouseEvent) mouseReleased(MouseEvent)
MouseMotionListener	mouseDragged(MouseEvent) mouseMoved(MouseEvent)
TextListener	textValueChanged(TextEvent)
WindowListener	windowActivated(WindowEvent) windowClosed(WindowEvent) windowClosing(WindowEvent) windowDeactivated(WindowEvent) windowDeiconified(WindowEvent) windowIconified(WindowEvent) windowOpened(WindowEvent)

70

## ActionListener I



- Se lanza si:
  - Se pulsa un botón de cualquier tipo
  - Doble pulsación en un item de una lista
  - Selección de una opción de menú
  - Pulsar retorno en un campo de texto
- Ejemplos:
  - `GUI01c.java`
    - El oyente es la propia ventana
  - y `GUI01c1.java`
    - El oyente es un objeto con visibilidad

72

## ActionListener II



- Si un oyente está pendiente de varios objetos, puede
  - preguntar por quién lo ha activado
    - `Object getSource()`
  - añadir junto con el registro una acción
    - `addActionCommand(String)`
    - y consultarla desde el oyente
      - `String getActionCommand()`
- ver GUI01c2.java
- ver GUIDia.java (uso de Dialog)

73

## ItemListener



- Se dispara si se pulsa en
  - `JCheckBox`
  - `JCheckBoxMenuItem`
  - `JList` (simple pulsación)

```
void itemStateChanged(ItemEvent)
```
- `ItemEvent`
  - `Object getItemSelectable()`
  - `int getStateChanged()`
    - Puede ser `ItemEvent.SELECTED` o `ItemEvent.DESELECTED`

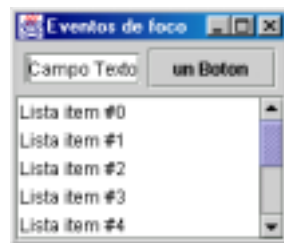
75

## FocusListener



- Controla cuando un componente gana o pierde el foco

```
void focusGained(FocusEvent)
void focusLost(FocusEvent)
```
- Ejemplo: GUI08.java



74

## KeyListener



- Se dispara
  - Cuando se pulsa o libera una tecla en un componente que tiene el foco

```
void keyTyped(KeyEvent)
void keyPressed(KeyEvent)
void keyReleased(KeyEvent)
```
- Métodos de `KeyEvent`
  - `int getKeyChar()`, `int getKeyCode()`,
  - `int getModifiersText()`, ...
  - Define como constante cualquier pulsación. Ver documentación

76

## MouseListener



- Para actuar con el ratón desde cualquier componente (no los movimientos)

```
void mouseClicked(MouseEvent)
void mouseEntered(MouseEvent)
void mouseExited(MouseEvent)
void mousePressed(MouseEvent)
void mouseReleased(MouseEvent)
```

- MouseEvent

- int getClickCount()
- int getX(), int getY(), Point getPoint()
- boolean isPopupTrigger()
- Define como constante el tipo de pulsación. Ver documentación.

77

## MouseMotionListener



- Para actuar con los movimientos del ratón desde cualquier componente

```
void mouseDragged(MouseEvent)
void mouseMoved(MouseEvent)
```

- MouseInputListener
  - javax.swing.event
  - Hereda de ambas interfaces
  - Ejemplo GUI10.java

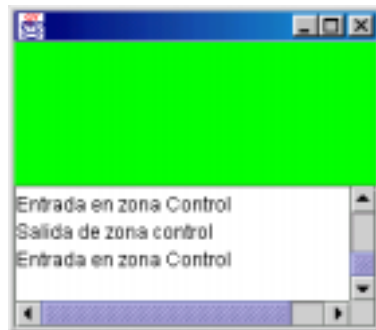


79

## MouseListener



- GUI09.java



78

## WindowListener



- Controla el movimiento de una ventana (window)

```
void windowClosing(WindowEvent)
void windowOpened(WindowEvent)
void windowIconified(WindowEvent)
void windowDeiconified(WindowEvent)
void windowClosed(WindowEvent)
void windowActivated(WindowEvent)
void windowDeactivated(WindowEvent)
```

- WindowEvent

- int getWindow()
- Ejemplo GUI11.java

80

## Adaptadores



- Es tedioso tener que implementar todas las funciones de una interface.
  - Por ejemplo, de un `WindowListener` puede interesar sólo el método de cierre, es decir `windowClosing`.
  - Si se implementa la interface hay que implementar todas las funciones, aunque estén vacías
- Una solución, los adaptadores
  - Son clases que implementan una interface con un comportamiento por defecto.
  - Podemos crear nuestros oyentes como subclases de estos adaptadores y redefinir cualquier método.

81

## Clase Anónima para WindowAdapter



```
class Ventana extends JFrame {  
    public Ventana() {  
        super("Eventos de ventana");  
        addWindowListener(new WindowAdapter() {  
            public void windowClosing(WindowEvent e) {  
                System.exit(0);  
            }  
        });  
    }  
}
```

- Ver GUI12a.java

83

## Adaptadores



- Hay adaptadores para
  - `ComponentListener`    `ComponentAdapter`
  - `ContainerListener`   `ContainerAdapter`
  - `FocusListener`        `FocusAdapter`
  - `KeyListener`           `KeyAdapter`
  - `MouseListener`        `MouseAdapter`
  - `MouseMotionListener` `MouseMotionAdapter`
  - `WindowListener`      `WindowAdapter`
- Ejemplo GUI11.java con adapter es GUI12.java

82

## La clase Graphics I



- Controla la parte gráfica del lenguaje
  - Cada componente dispone del método `public void paintComponent(Graphics)`
  - Redefiniendo este método podemos pintar sobre él.
  - Un objeto `Graphics` contiene:
    - El objeto de tipo **Component** sobre el que se pinta
    - Un origen de traslación para coordenadas de pintado y *clipping*
    - La región actual ocupada por el componente
    - El color actual
    - La fuente de caracteres actual
    - La operación lógica actual para utilizar con píxeles (XOR o Paint)
    - La actual alteración de color XOR

84

## La clase Graphics II



- Algunos métodos de interés

```
clearRect(int x,int y,int anchura,int altura )
```

```
copyArea(int x,int y,int anc,int alt,int  
xDes,int yDes )
```

```
setColor(Color c)
```

```
translate(int, int)
```

85

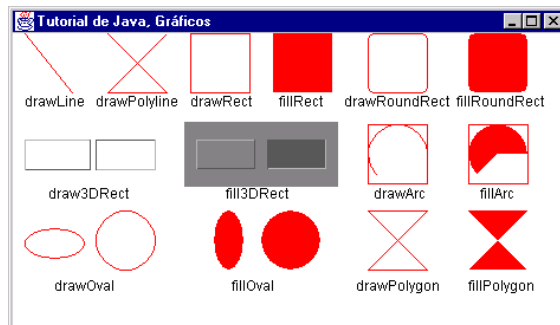
## Dibujar con Swing



- `paintComponent(Graphics)`
  - Invocado por el sistema cada vez que necesita pintar
    - Si se redefine, se debe llamar primero al mismo método del super para que pinte el fondo.
  - `repaint()`
  - `repaint(int x, int y, int w, int h)`
    - Invocado por el usuario.

87

## La clase Graphics III

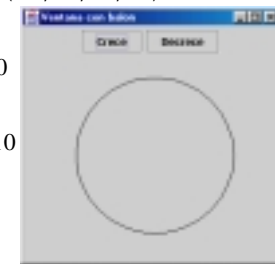


86

## Ejemplo



- Para pintar un círculo enviar al objeto Graphics el mensaje `drawOval(int x,int y,int rx,int ry)`
- Inicialmente es `drawOval(60,60,20,20)`
- `crece`
  - Incrementa el radio en 10
- `decrece`
  - Decrementa el radio en 10
- Se cierra correctamente
  - `GUIad.java` y
  - `GUIada.java`



88

## La clase Graphics IV



- Métodos para pintar textos
  - `drawString(String, int, int)`
- Cambio de fuentes
  - `setFont(Font)`
- Fuentes. En el constructor hay que indicar
  - Nombre: "Helvetica", "Courier", ...
  - Familia: `Font.PLAIN`, `Font.BOLD`, `Font.ITALIC`
  - Tamaño: 12, 14, 16, ...

89

## Applets I



- Aplicaciones diseñadas para ser ejecutadas en un navegador
- JApplet es un contenedor superior
  - No dispone de método `main()`
  - Dispone de métodos
    - `init()` Método de inicialización. Aquí se debe construir la forma del applet
    - `start()` Se ejecuta justo antes de la presentación de un applet
    - `stop()` Se ejecuta justo antes de la ocultación de un applet
    - `destroy()` Cuando desaparece el applet

91

## La clase Graphics V



- Las imágenes son objetos de la clase `Image`
  - Para pintar una imagen hay varios métodos  
`drawImage(Image, int, int, int, int ImageObserver)`
    - El método termina aún cuando la imagen no esté completamente pintada
    - `ImageObserver` es un objeto que es avisado cuando termine de pintar la imagen
  - Dos forma de obtener una imagen  
`Toolkit.getDefaultToolkit().getImage("dibujos.gif");`  
`(new ImageIcon("miImagen.gif")).getImage();`
- Ver `Imagen.java`

90

## Applets II



- Cualquier applet debe heredar de la clase `JApplet`
- Un applet debe ser una clase pública
- No puede acceder a los recursos locales
  - Certificados

```
public class SimpleApplet extends JApplets {  
    public void init() {  
        JLabel label = new JLabel("Hola mundo");  
        getContentPane().add(label, BorderLayout.CENTER);  
    }  
}
```

92

## Arrancar un Applets



- La ejecución debe realizarse desde una página HTML

```
<applet codebase=. code=SimpleApplet.class width=350 height=60>
...
</applet>
```

donde

codebase : Indica el camino donde buscar las clases  
code : Indica el nombre de la clase a lanzar  
width : Indica la anchura de la ventana  
height : Indica la altura de la ventana  
aling : Indica la alineación center, left, right

- Ver lanzaSimple.html y lanzaMuestra.html (appletViewer)
- Herramientas de conversión para navegadores no Swing

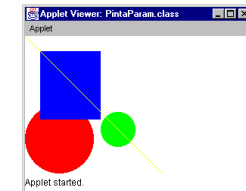
93

## Ejemplo de applets



- Ejercicio: LanzaPinta.html

```
<applet codebase=. code=PintaParam.class width=300 height=200>
<param name= figura1 value= circulo-verde-100-100-45>
<param name= figura2 value= circulo-rojo-0-90-90>
<param name= figura3 value= rectangulo-azul-20-20-80-90>
<param name= figura4 value= linea-amarillo-0-0-200-200>
</applet>
```



95

## Applets



- URL `getDocumentBase()`
  - Valor actual del documento presente en el navegador
- URL `getCodeBase()`
  - Directorio del código fuente del applet
- URL se encuentra en el paquete `java.net`
- void `showDocument(URL)`
  - pide que muestre el documento indicado
- void `showStatus(String)`
  - Saca un mensaje por la barra de estado del navegador

94

## Applets en ficheros .jar



- Se pueden juntar varias clases de un applet en un fichero .jar
  - Optimiza la comunicación
    - `jar cf misClases.jar *.class`
  - En el fichero .html

```
< applet code = "laClase.class"
      archive = "misClases.jar"
      width = 300 height = 150>
</applet>
```
  - El fichero .jar debe incluirse en CLASSPATH

96