

Unidad 11

Componentes y contenedores Swing

Programación
1º D.A.M.

1

Contenido

1. Administradores de diseño
2. Componentes Swing
3. Contenedores Swing

2

1. Administradores de diseño

1. FlowLayout
2. GridLayout
3. BorderLayout
4. BoxLayout
5. GridBagLayout
6. Ubicación absoluta
7. Administrador de apariencia

3

1. Administradores de diseño

- Determinan posición de controles en un contenedor
- GUI's Java vs. GUI's en otros lenguajes
 - Otros lenguajes → Orientadas sólo a una plataforma
 - Medidas y posiciones de controles con coordenadas absolutas
 - Java → No orientadas a una plataforma concreta
 - Medidas y posiciones de controles con administradores de diseño
 - Método `setLayout` de la clase `Container`
 - Un único parámetro → `LayoutManager`, interfaz que implementan
 - `FlowLayout`
 - `GridLayout`
 - `BoxLayout`
 - `BorderLayout`
 - `GridBagLayout`
 - ...
- Elementos añadidos al contenedor con método `add`

4

1. Administradores de diseño

- Administradores derivados de AWT
 - `BorderLayout`
 - `CardLayout`
 - `FlowLayout`
 - `GridBagLayout`
 - `GridLayout`

5

1. Administradores de diseño

- Administradores definidos en Swing
 - `BoxLayout`
 - Usado por el contenedor `Box`
 - Dispone a sus hijos en una fila o columna
 - `OverlayLayout`
 - Usado por `AbstractButton`
 - Superpone sus hijos
 - `ScrollPaneLayout`
 - Usado por `JScrollPane`
 - `ViewportLayout`
 - Usado por `JViewport`

6

1.1. FlowLayout

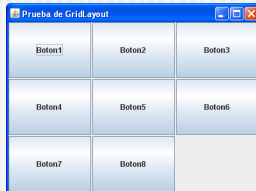
- Componentes distribuidos de izquierda a derecha y de arriba abajo.
- Distribución sencilla y efectiva



7

1.2. GridLayout

- Distribución en forma de malla con filas y columnas
 - Celdas del mismo tamaño
 - Componentes desde celda superior izquierda a inferior derecha
- Método `add` del contenedor admite segundo parámetro: número de celda en que colocar el componente



8

1.3. BorderLayout

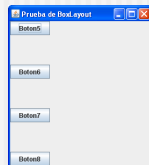
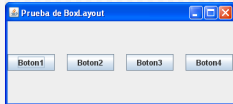
- Coloca componentes alrededor de los bordes de un contenedor
 - `BorderLayout.NORTH`
 - `BorderLayout.SOUTH`
 - `BorderLayout.EAST`
 - `BorderLayout.WEST`
 - `BorderLayout.CENTER`



9

1.4. BorderLayout

- Distribuye componentes en una fila o columna
- Pensado para filas y columnas de botones
- Incorporado por Swing
 - javax.swing
 - También incorpora la clase contenedor Box
 - Manipulación interna del administrador BorderLayout
 - `Box horizontal = Box.createHorizontalBox();`
 - `Box vertical = Box.createVerticalBox();`



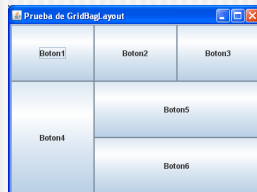
10

1.5. GridBagLayout

- Administrador más flexible de todos
- Manipulación más compleja
- Coloca componentes en relación a sí mismos
 - Se consigue cualquier diseño
 - Requiere esfuerzo y "paciencia"
 - Uso recomendado con programas de diseño
 - NetBeans
- Usa la clase GridBagConstraints
 - Controla posición y propiedades de componentes añadidos a contenedores GridBagLayout

11

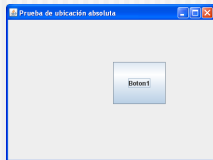
1.5. GridBagLayout



12

1.6. Ubicación absoluta

- Colocación libre de componentes mediante píxeles
 - Válida sólo cuando la ventana tiene tamaño fijo
 - Uso desaconsejado
- Pasos a seguir
 - `contenedor.setLayout(null)`
 - `componente.setLocation(posX, posY);`
 - `componente.setSize(ancho, alto);`
 - `contenedor.add(componente);`



13

1.7. Administrador de apariencia

- Clase UIManager
 - Aportación de Swing
 - `javax.swing`
 - Cambio de la apariencia según esquemas preestablecidos
 - Visualización independiente de la plataforma, determinada por esquemas
 - Metal → apariencia por defecto
 - Motif → apariencia X-Window
 - Windows → apariencia Windows
 - Método `setLookAndFeel`
 - Cambio de la apariencia
 - Argumentos
 - `javax.swing.plaf.metal.MetalLookAndFeel`
 - `com.sun.java.swing.plaf.motif.MotifLookAndFeel`
 - `com.sun.java.swing.plaf.windows.WindowsLookAndFeel`

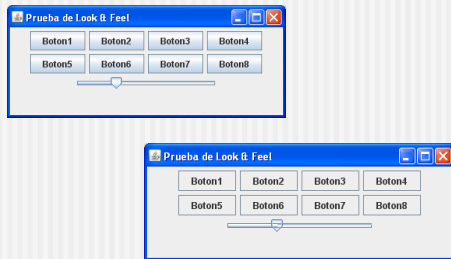
14

1.7. Administrador de apariencia

```
try {
    UIManager.setLookAndFeel(UIManager
        .getCrossPlatformLookAndFeelClassName());
} catch (Exception e) {
}
```

15

1.7. Administrador de apariencia



16

2. Componentes Swing

- 1. Etiquetas
- 2. Cuadros de texto
- 3. Cuadros de contraseña
- 4. Botones
- 5. Casillas de activación
- 6. Botones de opción
- 7. Viewport
- 8. Paneles de desplazamiento
- 9. Barras de desplazamiento
- 10. Deslizadores
- 11. Listas
- 12. Listas combinadas
- 13. Cuadros de diálogo
 - 14. Diálogos genéricos
 - 15. Mensajes al usuario
 - 16. Selector de colores
 - 17. Selector de archivos
- 18. Menús
- 19. Menús contextuales
- 20. Paneles con pestañas

17

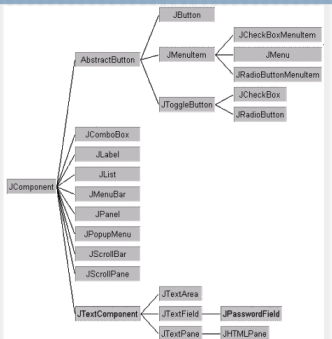
2. Componentes Swing

■ Jerarquía

- `java.lang.Object`
 - `java.awt.Component`
 - `java.awt.Container`
 - `javax.swing.JComponent`

18

2. Componentes Swing



19

2. Componentes Swing

■ JButton	→	Botón
■ JCheckBox	→	Botón de comprobación
■ JCheckBoxMenuItem	→	Botón de comprobación para menús
■ JColorChooser	→	Selector de colores
■ JComboBox	→	Entrada de texto con lista de valores
■ JComponent	→	Raíz de jerarquía de componentes Swing
■ JEditorPane	→	Editor de texto
■ JFileChooser	→	Selector de ficheros
■ JLabel	→	Etiqueta
■ JList	→	Lista
■ JMenu	→	Menú dentro de JMenuBar u otro menú.
■ JMenuBar	→	Barra de menús
■ JMenuItem	→	Elemento seleccionado en un menú
■ JOptionPane	→	Ventanas de diálogo
■ JPasswordField	→	Entrada de passwords
■ JPopupMenu	→	Ventana con un menú

20

2. Componentes Swing

■ JProgressBar	→	Barra de progreso
■ JRadioButton	→	Botón excluyente
■ JRadioButtonMenuItem	→	Botón excluyente para usar en menús
■ JScrollbar	→	Barra de desplazamiento
■ JSeparator	→	Líneas de separación
■ JSlider	→	Deslizador
■ JTable	→	Tabla
■ JTextArea	→	Edición de múltiples líneas de texto plano
■ JTextComponent	→	Raíz de los editores de texto
■ JTextField	→	Edición de una línea de texto plano
■ JTextPane	→	Subclase de JTextArea para hacer procesadores de texto
■ JToggleButton	→	Padre de JCheckBox y JRadioButton
■ JToolBar	→	Barra de herramientas o acciones
■ JToolTip	→	Ventana informativa
■ JTree	→	Árboles

21

2.1. Etiquetas. [JLabel](#)

- Textos de una línea para dar información textual
- Usadas en ventanas y applets
- Se pueden asociar a un componente concreto
- Variantes
 - Etiquetas HTML
 - Uso de estos tags en las etiquetas
 - Formato de texto y párrafo html
 - Comienzo del texto con la etiqueta <html>
 - Etiquetas gráficas
 - Admiten imágenes en su interior
 - Basadas en la interfaz [Icon](#)
 - La clase [ImageIcon](#) implementa dicha interfaz

22

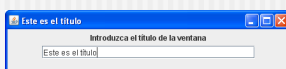
2.1. Etiquetas. [JLabel](#)



23

2.2. Cuadros de texto. [JTextField](#)

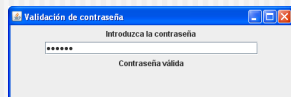
- Permiten introducir texto al programa
 - Una sola línea de texto
 - Puede ser alineado
 - Para más líneas se usa [JTextArea](#)
- Al pulsar Enter tras introducir texto, evento `ActionEvent`
- Uno de los controles más usados



24

2.3. Cuadros de contraseña. [JPasswordField](#)

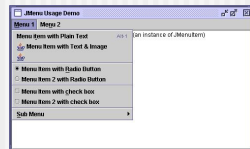
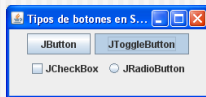
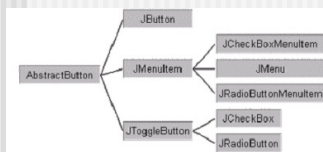
- Subclase de `JTextField`
- El texto escrito queda oculto
 - Normalmente con asteriscos
 - `char getEchoChar()`
 - `setEchoChar(char c)`
- Se recoge la contraseña introducida
 - `char[] getPassword()`



25

2.4. Botones

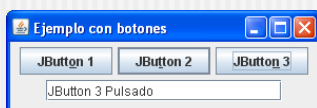
- Jerarquía en los botones



26

2.4. Botones. [JButton](#)

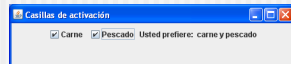
- Botones, fundamentales en las GUI
- Principal tipo de botón
 - Hereda de [AbstractButton](#)
- Al pulsarse generan evento `ActionEvent`
 - Capturado para dar funcionalidad al botón



27

2.5. Casillas de activación. [JCheckBox](#)

- Puede ser activado y desactivado
- Permite elegir una serie de opciones independientes
- Relación de herencia
 - [AbstractButton](#)
 - [JToggleButton](#)
 - [JCheckBox](#)
- Puede tener imágenes asociadas a cada estado
 - Métodos
 - `setIcon` → Imagen para estado normal
 - `setSelectedIcon` → Imagen para estado seleccionado
- Eventos
 - `ActionEvent` → La casilla es seleccionada
 - `ItemEvent` → Cambia el estado de la casilla
 - `ItemListener`
 - `itemStateChanged`



28

2.6. Botones de opción. [JRadioButton](#)

- Similares a los botones de activación
- Se usan para seleccionar una opción de entre varias
- Relación de herencia
 - [AbstractButton](#)
 - [JToggleButton](#)
 - [JRadioButton](#)
- Agrupación de botones de radio: [ButtonGroup](#)
 - Para asegurar que sólo se puede elegir una opción
- Métodos
 - `add` → Añadir un botón al grupo
 - `remove` → Quitar un botón del grupo



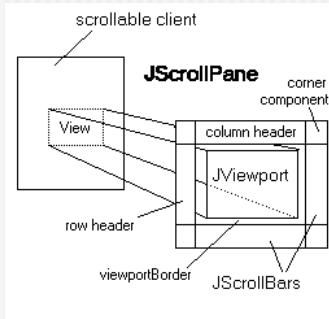
29

2.7. Viewport. [JViewport](#)

- Clase asociada a las clases que permiten desplazamientos (*scrolls*)
- Ventana dentro de la vista actual
 - Muestra una sección de los datos
 - Área visible en cada momento
 - Permite desplazar la vista hacia el resto de datos
- Usa interfaz [Scrollable](#)
 - Permite realizar desplazamientos
- Manejo
 - Construcción del objeto
 - Asignación de un componente ligero (panel)
 - Método `setView`
 - Pasando el componente a visualizar

30

2.7. Viewport. [JViewport](#)



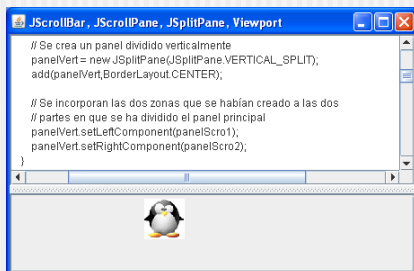
31

2.8. Paneles con desplazamiento. [JScrollPane](#)

- Para colocar barras de desplazamiento a cualquier componente
- Usa interfaz [Scrollable](#)
 - Permite realizar desplazamientos

32

- **JScrollBar**, **JScrollPane**, **JSplitPane**, **Viewport**



33

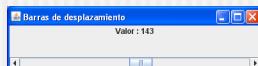
2.9. Barras de desplazamiento. [JScrollBar](#)

- Objetos de barra de desplazamiento
- Suele ser suficiente con usar [JScrollPane](#)
- Sólo suele usarse JScrollBar como tal
 - Para acciones especiales sobre las barras
 - Para modificar las propiedades de las barras
 - Extensión → Tamaño de la guía (*track*)
 - Valor → Valor que la representa actualmente
 - Depende de la posición actual de la guía
 - Mínimo → Mínimo valor que la representa
 - Valor de la barra cuando la guía está al principio
 - Máximo → Máximo valor que la representa
 - Valor de la barra cuando la guía está al final

34

2.9. Barras de desplazamiento. [JScrollBar](#)

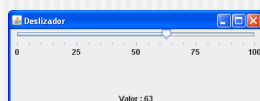
- [AdjustmentEvent](#)
 - Al modificarse el valor de las barras
 - Métodos interesantes
 - `getValue()`
 - Valor de la barra
 - `getAdjustmentType()`
 - Tipo de cambio que se produjo en la barra
 - `AdjustmentEvent.UNIT_INCREMENT`
 - `AdjustmentEvent.UNIT_DECREMENT`
 - `AdjustmentEvent.BLOCK_INCREMENT`
 - `AdjustmentEvent.BLOCK_DECREMENT`
 - `AdjustmentEvent.TRACK`
- [AdjustmentListener](#)
 - `adjustmentValueChanged`



35

2.10. Deslizadores. [JSlider](#)

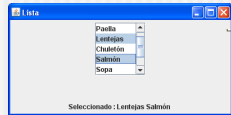
- Similar a la barra de desplazamiento, pero sólo para elegir un valor numérico
- Permiten mostrar marcas
 - Facilitan al usuario la selección de un valor
 - Método `setPaintTicks`
 - Métodos para espacio entre marcas, etc
- Eventos [ChangeEvent](#)
 - Escuchador [ChangeListener](#)
 - Añadido al JSlider con `addChangeListener`
 - Método de captura `stateChanged`



36

2.11. Listas. [JList](#)

- Permite elegir entre un conjunto de alternativas
- Posibilidades
 - Selección de una única opción
 - Selección de varias opciones (tecla Control)
- Es típico añadirle un panel de desplazamiento
 - `JScrollPane panel = new JScrollPane(miLista);`
- Eventos [ListSelectionEvent](#)
 - Escuchador [ListSelectionListener](#)
 - Añadido al `JList` con `addListSelectionListener`
 - Método de captura `valueChanged`



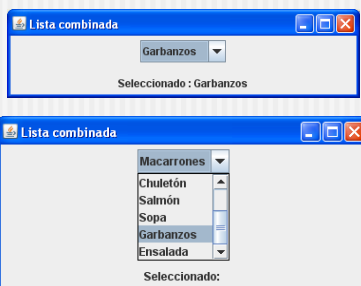
37

2.12. Listas combinadas. [JComboBox](#)

- Listas especiales
 - Capacidades de una lista
 - Capacidades de un cuadro de texto
- Apariencia de cuadros de texto
 - Botón para abrirlo y seleccionar UNA opción
- Uso común
- Eventos
 - [ActionEvent](#)
 - [ActionListener](#)
 - `actionPerformed`
 - [ItemEvent](#)
 - [ItemListener](#)
 - `itemStateChanged`
 - Método `getStateChange()`
 - `ItemEvent.SELECTED` → El cambio fue para seleccionar
 - `ItemEvent.DESELECTED` → El cambio fue para deseleccionar

38

2.12. Listas combinadas. [JComboBox](#)



39

2.13. Cuadros de diálogo

- Ventanas especializadas en operaciones complejas
 - Propósito genérico
 - [JDialog](#)
 - Propósito específico
 - Información, aviso, confirmación, solicitud de datos, ...
 - [JOptionPane](#)
 - Selección colores
 - [JColorChooser](#)
 - Selección archivos
 - [JFileChooser](#)
- Todo diálogo depende de un Frame
 - Se construye, destruye, maximiza, minimiza, ... con él
- Tipos de diálogo
 - Modal → Al ser visible acapara el programa
 - No modal → No acapara el programa

40

2.13.1. Diálogos genéricos. [JDialog](#)

- Cuadros de diálogo de propósito genérico
 - Sin propósito específico como otros diálogos
 - Adecuado para diálogo personalizado
- Una de las clases más importantes
- Tiene un contenedor asignado
 - JFrame
 - JApplet
- Métodos heredados de [JWindow](#). Entre ellos:
 - `show` → Muestra el cuadro de diálogo
 - `dispose` → Cierra el cuadro de diálogo

41

2.13.2. Mensajes al usuario. [JOptionPane](#)

- Cuadro de diálogo para comunicarse con el usuario
 - Uno de los elementos más usados
 - Varios propósitos
 - Textos de aviso
 - Textos de error
 - Confirmaciones
 - Entradas sencillas de datos
 - ...
- Posibilidades
 - Creación de objetos [JOptionPane](#)
 - Posee constructores
 - Uso de métodos estáticos
 - No requiere objeto
 - Muy usado por su sencillez y rapidez

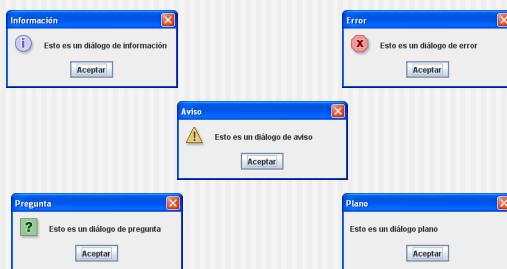
42

2.13.2. Mensajes al usuario. [JOptionPane](#)

- Cuadros de información
 - Informan al usuario de algún hecho
 - Construcción con métodos estáticos
 - `showMessageDialog(Component padre, Object mensaje)`
 - `showMessageDialog(Component padre, Object mensaje, String título, int tipo)`
 - `showMessageDialog(Component padre, Object mensaje, String título, int tipo, Icon icono)`
 - Parámetro tipo
 - `JOptionPane.INFORMATION_MESSAGE`
 - `JOptionPane.ERROR_MESSAGE`
 - `JOptionPane.WARNING_MESSAGE`
 - `JOptionPane.QUESTION_MESSAGE`
 - `JOptionPane.PLAIN_MESSAGE`

43

2.13.2. Mensajes al usuario. [JOptionPane](#)



Cuadros de información

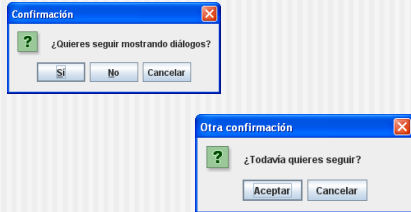
44

2.13.2. Mensajes al usuario. [JOptionPane](#)

- Cuadros de confirmación
 - Se captura la respuesta del mensaje
 - Acepta el mensaje
 - Declina el mensaje
 - Construcción con métodos estáticos
 - `showConfirmDialog(Component padre, Object mensaje)`
 - `showConfirmDialog(Component padre, Object mensaje, String título, int opciones)`
 - `showConfirmDialog(Component padre, Object mensaje, String título, int opciones, int tipo)`
 - `showConfirmDialog(Component padre, Object mensaje, String título, int opciones, int tipo, Icon icono)`
 - Valores devueltos
 - `JOptionPane.NO_OPTION`
 - `JOptionPane.CLOSE_OPTION`
 - `JOptionPane.OK_OPTION`
 - `JOptionPane.YES_OPTION`
 - `JOptionPane.CANCEL_OPTION`
 - Parámetro opciones
 - `JOptionPane.OK_CANCEL_OPTION`
 - `JOptionPane.YES_NO_OPTION`
 - `JOptionPane.YES_NO_CANCEL_OPTION`
 - Parámetro tipo → Igual que para los cuadros de información

45

2.13.2. Mensajes al usuario. JOptionPane



Cuadros de confirmación

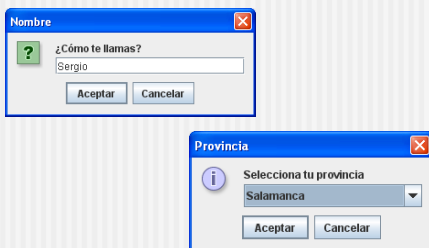
46

2.13.2. Mensajes al usuario. JOptionPane

- **Solicitud de entradas**
 - Introducción de algún valor por el usuario
 - A partir de un cuadro de texto
 - A partir de una lista combinada
 - Construcción con métodos estáticos
 - `showInputDialog(Component padre, Object mensaje)`
 - `showInputDialog(Component padre, Object mensaje, Object valorInicial)`
 - `showInputDialog(Component padre, Object mensaje, String titulo, int tipo)`
 - `showInputDialog(Component padre, Object mensaje, String titulo, int tipo, Icon icono, Object[] listaValores, Object valorInicial)`
 - `showInputDialog(Object mensaje)`
 - `showInputDialog(Object mensaje, Object valorInicial)`
 - **Valores devueltos**
 - `String` → Si el usuario proporciona un valor
 - `null` → Si el usuario cancela

47

2.13.2. Mensajes al usuario. JOptionPane



Solicitud de entradas

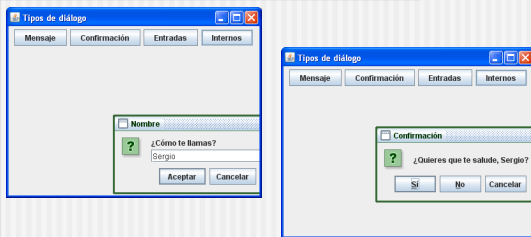
48

2.13.2. Mensajes al usuario. [JOptionPane](#)

- Cuadros de diálogo internos
 - Dentro de un contenedor
 - No pueden salir fuera de él
 - Más ligeros
 - Ocupan menos recursos
 - Mismos métodos y capacidades que los no internos
 - Métodos de creación
 - `showInternalMessageDialog`
 - Como `showMessageDialog`, pero interno
 - `showInternalInputDialog`
 - Como `showInputDialog`, pero interno
 - `showInternalConfirmDialog`
 - Como `showConfirmDialog`, pero interno

49

2.13.2. Mensajes al usuario. [JOptionPane](#)



Diálogos internos

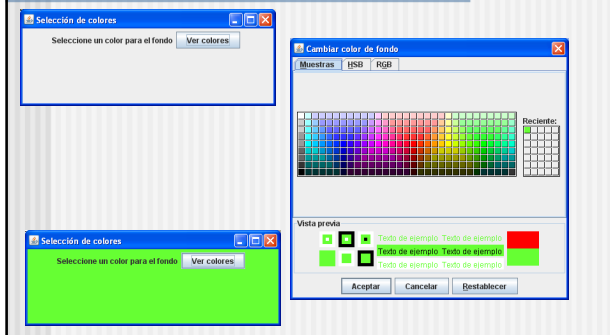
50

2.13.3. Selector de colores. [JColorChooser](#)

- Diálogo para seleccionar colores
 - [JColorChooser](#)
 - `showDialog` → Visualización del diálogo
 - Estático
 - Objeto devuelve
 - `null` → Si no se ha seleccionado un color
 - `Color` → Si se ha seleccionado un color

51

2.13.3. Selector de colores. [JColorChooser](#)



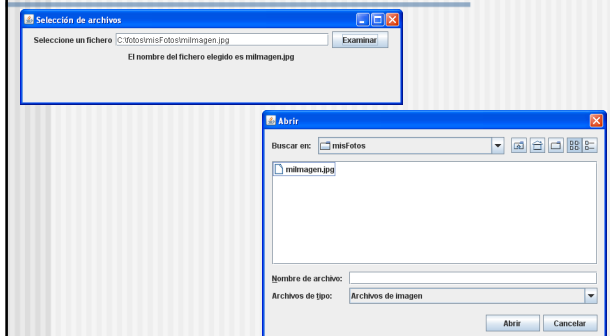
52

2.13.4. Selector de archivos. [JFileChooser](#)

- Diálogo para seleccionar archivos
- [JFileChooser](#)
 - `setFileFilter` → Establecer filtro para el selector
 - `showOpenDialog` → Mostrar el selector para abrir
 - `showSaveDialog` → Mostrar el diálogo para guardar
 - `getSelectedFile` → Fichero seleccionado (File)
 - `getName` → Nombre del fichero seleccionado
 - `getPath` → Path del fichero seleccionado
- [FileFilter](#)
 - Filtro de ficheros a mostrar en el selector
 - `javax.swing.filechooser.FileFilter`
 - Clase abstracta de la que heredar para crear el filtro
 - Métodos abstractos a implementar
 - `accept` → Ficheros que integrarán el filtro
 - `getDescription` → Descripción para el filtro
- También existe la interfaz [FileFilter](#)

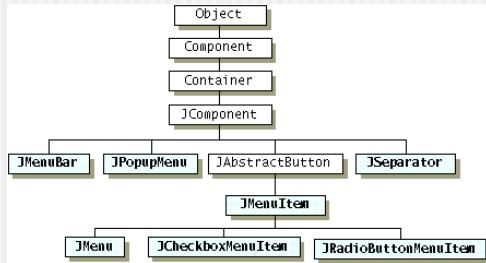
53

2.13.4. Selector de archivos. [JFileChooser](#)



54

2.14. Menús. [JMenu](#), [JMenuItem](#), [JMenuBar](#) [JRadioButtonMenuItem](#), [JCheckBoxMenuItem](#)



55

2.14. Menús. [JMenu](#), [JMenuItem](#), [JMenuBar](#) [JRadioButtonMenuItem](#), [JCheckBoxMenuItem](#)

- **JMenu**
 - Menú que agrupa unos cuantas opciones
 - Contenedor de [JMenuItem](#)
 - add para añadirlos
 - Otros métodos
 - setMnemonic
 - addSeparator
- **JMenuBar**
 - Barra de menú propiamente dicha
 - Contenedor de [JMenu](#)
 - add para añadirlos
 - Normalmente asociado a una ventana ([JFrame](#))
 - Método setJMenuBar del JFrame

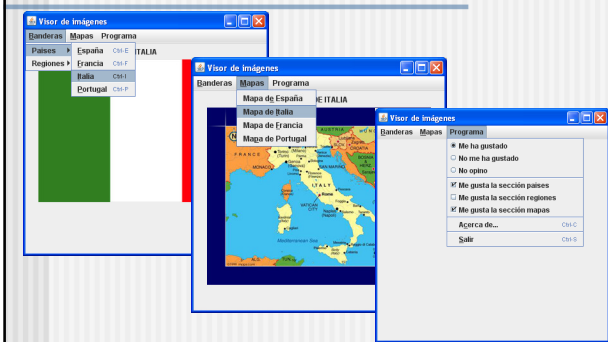
56

2.14. Menús. [JMenu](#), [JMenuItem](#), [JMenuBar](#) [JRadioButtonMenuItem](#), [JCheckBoxMenuItem](#)

- **JMenuItem**
 - Cada opción de un menú ([JMenu](#))
 - Eventos [ActionEvent](#)
 - [ActionListener](#)
 - actionPerformed
 - Métodos
 - addActionListener
 - setAccelerator
 - setMnemonic
- **JRadioButtonMenuItem**
 - Botón de radio para un menú
 - Posibilidad de añadirse a un [ButtonGroup](#)
- **JCheckBoxMenuItem**
 - Casilla de activación para un menú

57

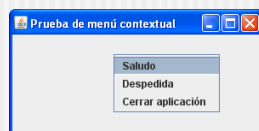
2.14. Menús. [JMenu](#), [JMenuItem](#), [JMenuBar](#) [JRadioButtonMenuItem](#), [JCheckBoxMenuItem](#)



58

2.15. Menús contextuales. [JPopupMenu](#)

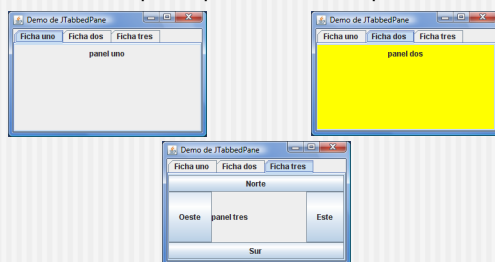
- Menú popup
 - Dependiente de eventos de ratón
 - [MouseEvent](#)
 - [MouseListener](#) - [MouseAdapter](#)
 - Asociados a cada componente en que pueda mostrarse el menú contextual



59

2.16. Paneles con pestañas. [JTabbedPane](#)

- Cada pestaña conlleva un panel independiente
- Cambio de panel por medio de la pestaña



60

3. Contenedores Swing

61

3. Contenedores Swing

■ Jerarquía

- `java.lang.Object`
 - `java.awt.Component`
 - `java.awt.Container`

62

3. Contenedores Swing

- | | | |
|-------------------------------|---|---|
| ■ <code>Box</code> | → | Posiciona hijos usando <code>BoxLayout</code> |
| ■ <code>JApplet</code> | → | Para applets |
| ■ <code>JDesktopPane</code> | → | Desktop que contiene <code>JInternalFrame</code> 's |
| ■ <code>JFrame</code> | → | Ventana |
| ■ <code>JInternalFrame</code> | → | Ventana interna. Normalmente dentro de un <code>JDesktopPane</code> |
| ■ <code>JLayeredPane</code> | → | Contenedores apilados |
| ■ <code>JPanel</code> | → | Agrupación de hijos |
| ■ <code>JRootPane</code> | → | |
| ■ <code>JScrollPane</code> | → | Añade barras de desplazamiento a su hijo |
| ■ <code>JSplitPane</code> | → | Muestra dos hijos pudiendo ajustar sus tamaños relativos |
| ■ <code>JTabbedPane</code> | → | Solapas para mostrar diferentes hijos |
| ■ <code>JViewport</code> | → | Muestra una parte de sus hijos. Suele usarlo <code>JScrollPane</code> |
| ■ <code>JWindow</code> | → | Ventana sin decoración |

63

3. Contenedores Swing

- Componentes [JMenuBar](#) y [JPopupMenu](#) son en realidad contenedores
- Contenedores de alto nivel
 - Extienden sus versiones AWT
 - [JFrame](#)
 - [JDialog](#)
 - [JWindow](#)
 - [JApplet](#)
 - Contienen método `getContentPane()`
 - Da acceso al contenedor al que se añaden sus hijos
 - Por defecto un `JPanel`

64

Unidad 11

Componentes y contenedores Swing

Programación
1º D.A.M.

65
