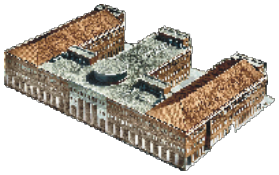
Interacción Persona Computador
Laboratorio - Sesión 2

UVa



Diseño de GUI con Java Swing y NetBeans (*)

C. Hernández / M.Gonzalo / A. Martínez

(*) Alguna parte de este material está basado en C.Vaca (2011).

INTRODUCCIÓN

Introducción

- Contenido en el paquete javax.swing
- Creada a partir de java.awt
- Escrita totalmente en Java
- Permite una interfaz adaptada a cada S.O. sin cambio decódigo
- JFrame, JApplet, JLabel,.....todos comienzan con J

CONTENEDORES

Contenedores y componentes

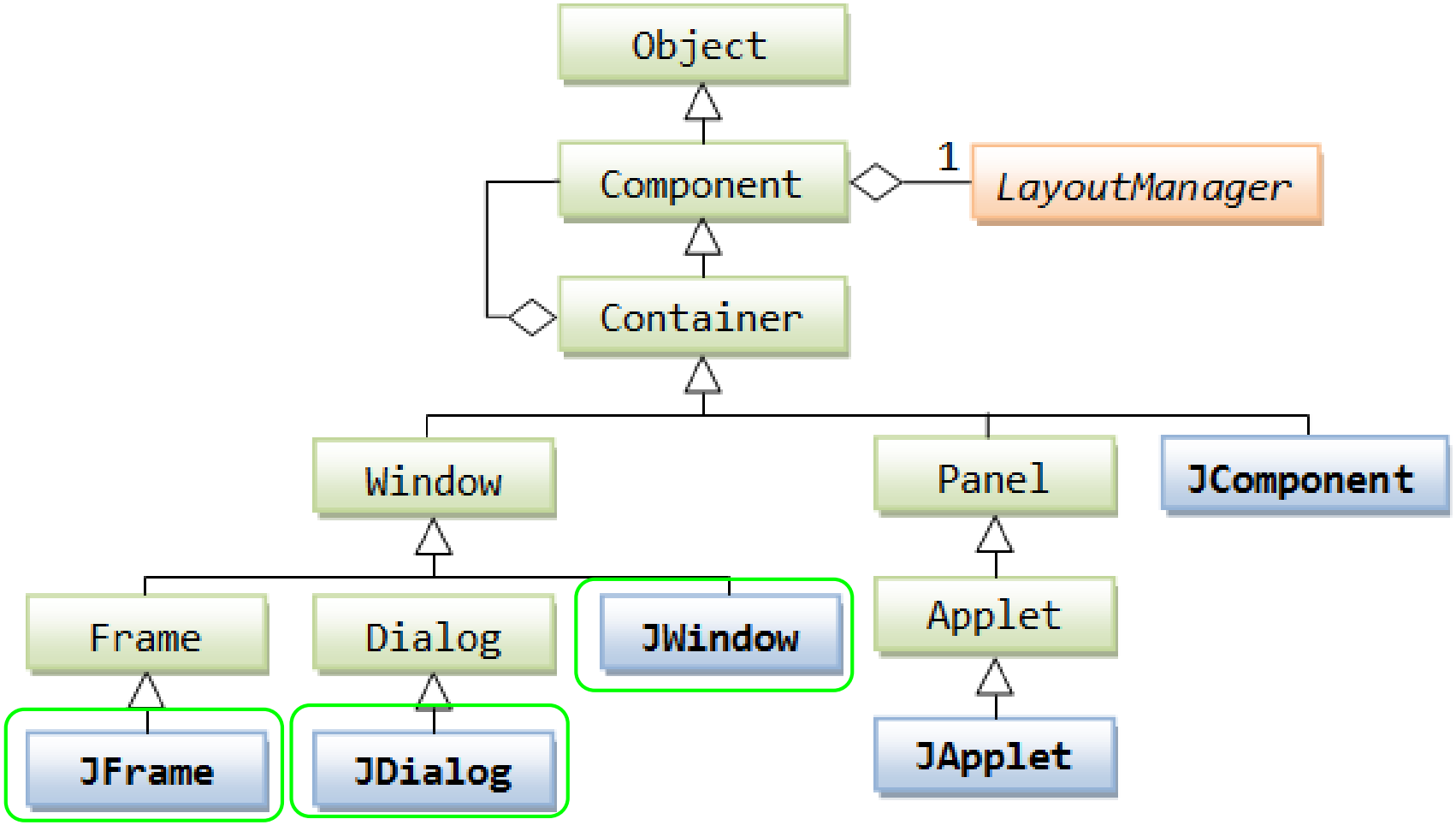
- Contenedores
  - Almacén de componentes
    - Contenedores superiores
    - Contenedores intermedios

- Componentes: sirve para construir el GUI
  - Botones, etiquetas, campos de texto, etc
  - Se sitúan dentro de algún contenedor

Contenedores

- Contienen componentes u otros contenedores
- Pueden ser:
  - Contenedores Superiores
    - JFrame
    - JDialog
    - JApplet
  - Contenedores Intermedios
    - JPanel
    - JScrollPane
    - JSplitPane
    - JTabbedPane
    - JToolBar y otros más especializado

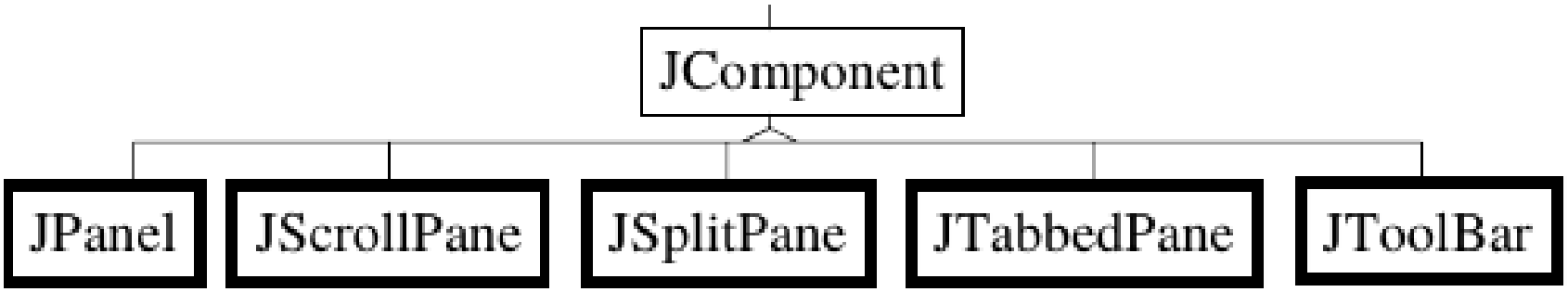
Contenedores superiores



Contenedores superiores en Swing

- JFrame
  - Main window
- JDialog
  - Pop - up para interactuar con los usuarios
- JApplet
  - Ejecuta un programa dentro de un navegador

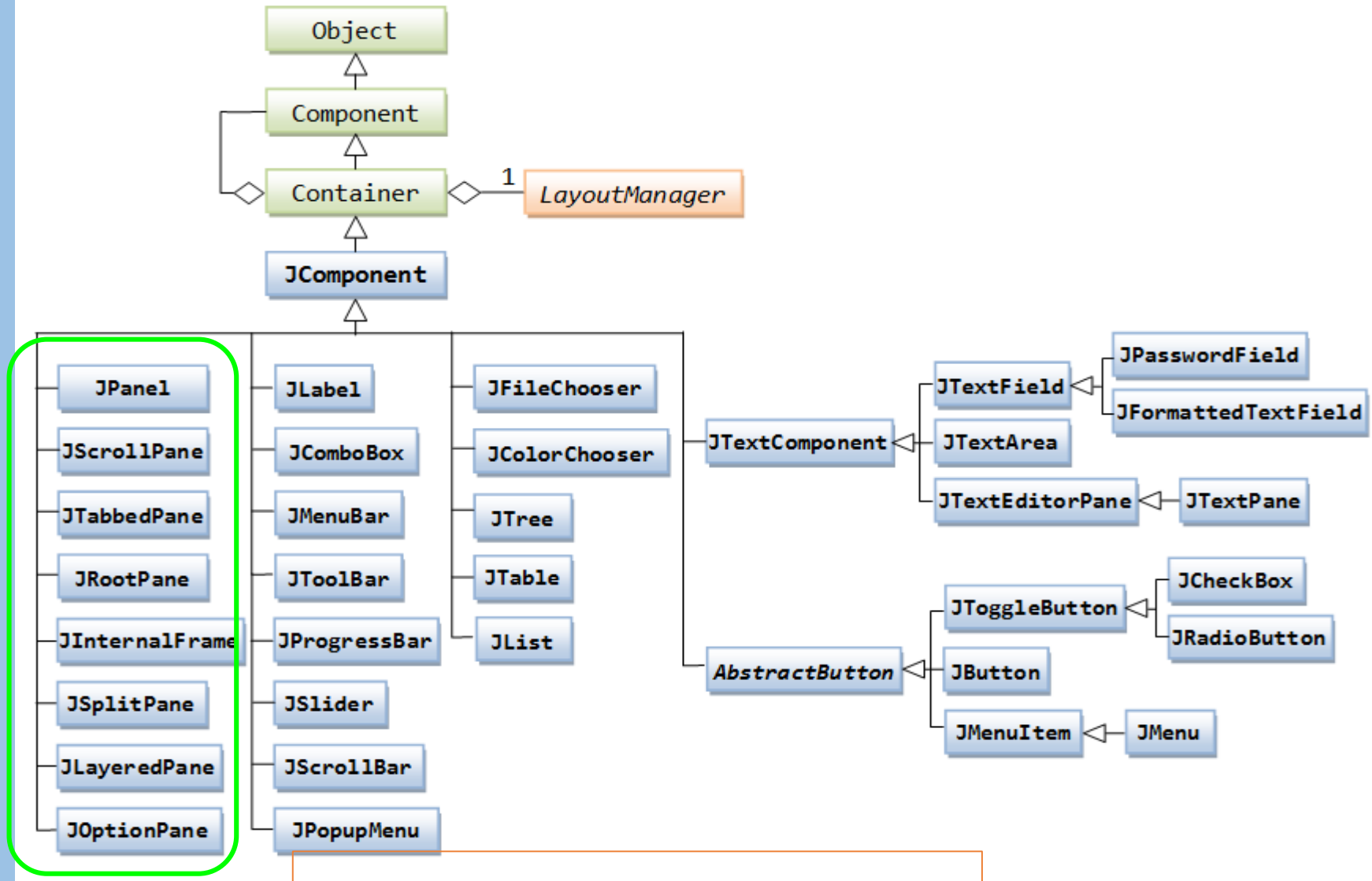
Contenedores intermedios



Contenedores intermedios en Swing

- JPanel
  - contenedor invisible que mantiene componentes de interfaz y que se puede anidar, colocándose en otros paneles o en ventanas.
  - También sirve de lienzo para dibujar.
  - Controla la disposición de otros componentes (layout)
- JScrollPane
  - Desplazamiento automático horizontal y vertical
- JSplitPane, JTabbedPane, JToolBar y otros más especializados

Contenedor intermedio: JPanel



Contenedores intermedios: Layout Manager

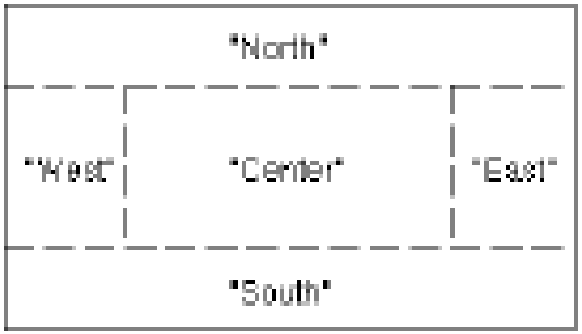
- Todos los contenedores intermedios tienen un Layout Manager
  - Layout Manager
    - Determina cómo se distribuirán los componentes en el contenedor
      - FlowLayout
      - BorderLayout
      - GridLayout
      - GridBagLayout
      - CardLayout (Swing propone alternativa)
      - BoxLayout (nueva en Swing: javax.swing)

LAYOUT MANAGER

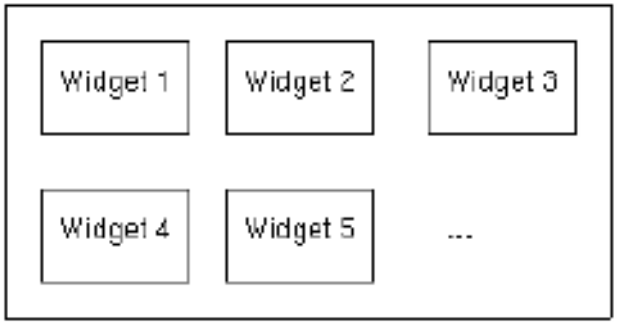
http://zetcode.com/tutorials/javaswingtutorial/swinglayoutmanagement/

Layout Manager

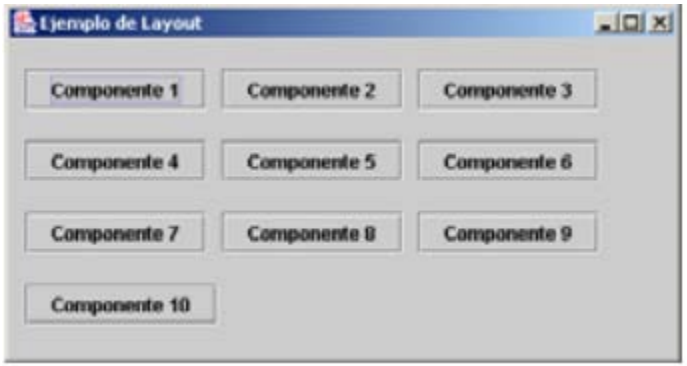
BorderLayout organiza el contenedor en 5 zonas:



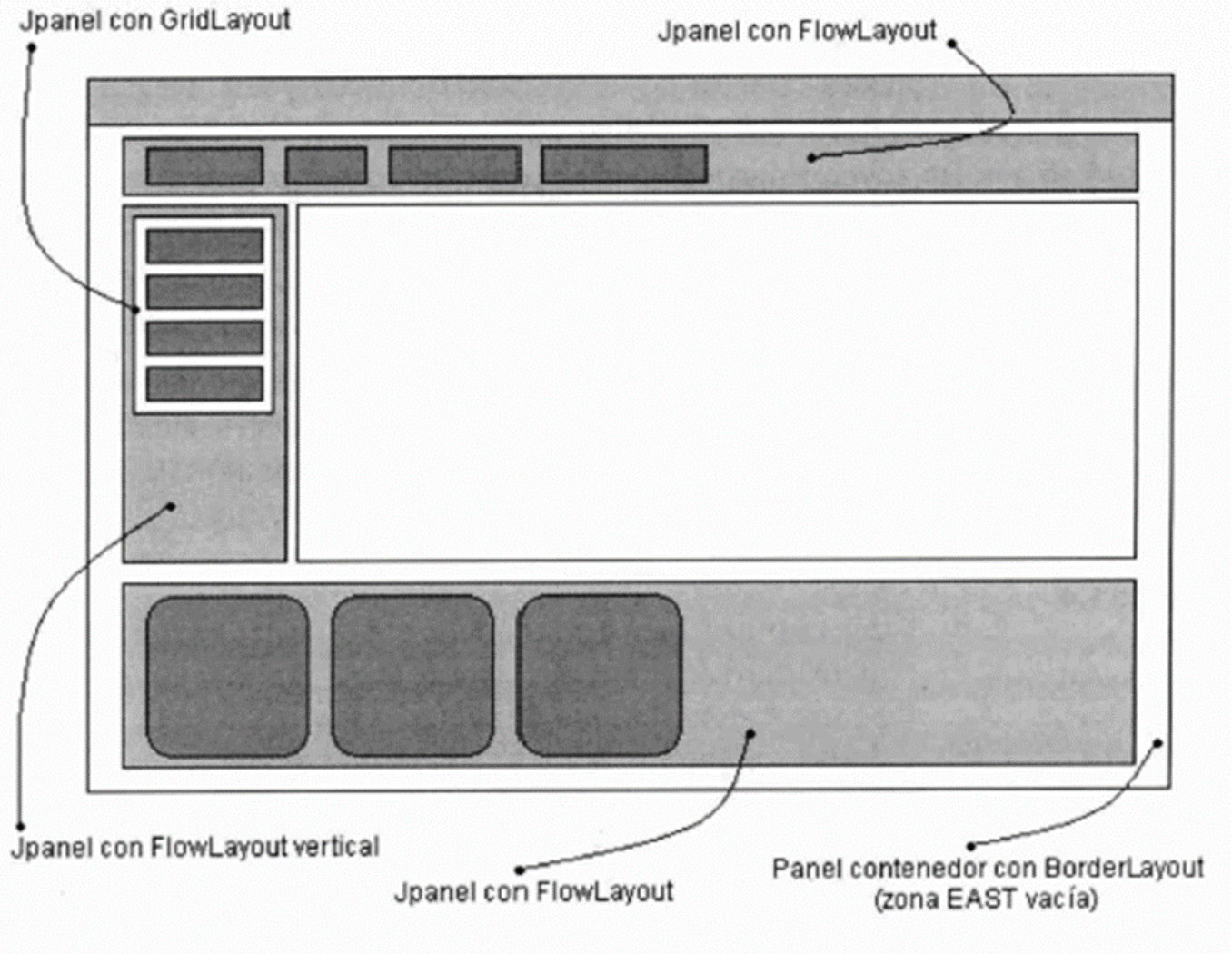
FlowLayout coloca los componentes de izquierda a derecha y de arriba hacia abajo:



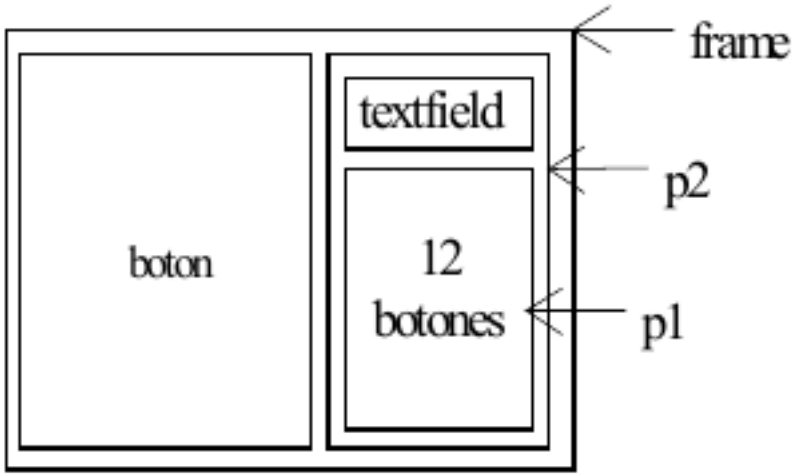
GridLayout



Layout Manager

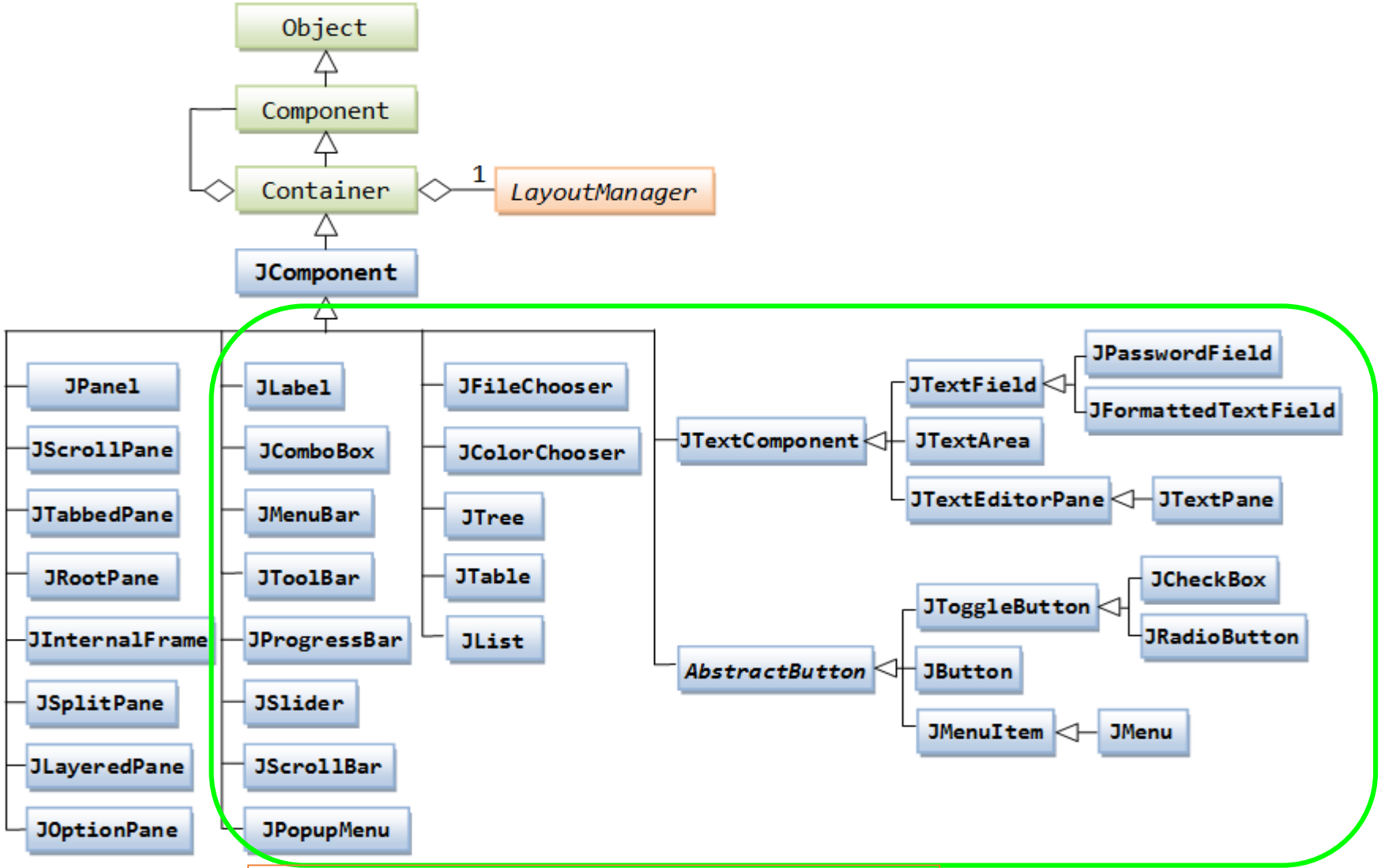


Layout Manager (ejemplo)



COMPONENTES

Componentes



Componente

- JButton
  - Crea botones de pulsación
- JLabel
  - Es una etiqueta con una línea de texto
- JCheckBox
  - Marcador
- JRadioButtons y ButtonGroup
  - Botones circulares
  - Se agrupan de manera que solo uno esté pulsado

Componente (II)

- JTextField
  - Permite editar un texto en una línea
- JTextArea
  - Edita un texto en un área
- JList
  - Muestra una lista de elementos para su selección
    - ❖La selección puede ser única, intervalo o múltiple
    - ❖Es muy interesante la forma de añadir elementos a la lista
- JComboBox
  - Selecciona un elemento pero no está desplegado

GESTIÓN de EVENTOS

Manejo de eventos

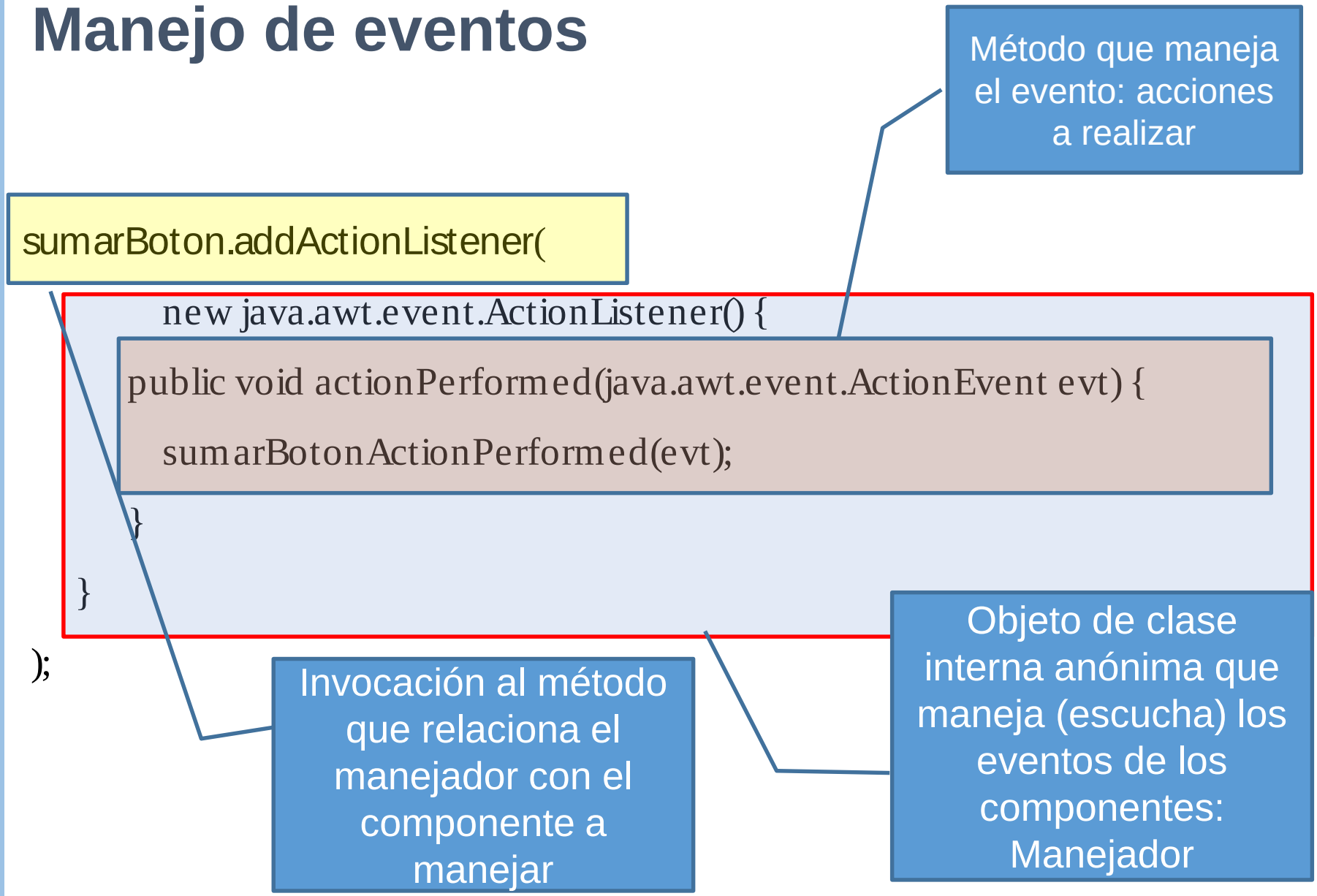
- Una aplicación responde a los eventos ejecutando código adecuado para cada tipo particular de eventos.
- No todos los eventos necesitan ser tenidos en cuenta por una aplicación. Por ejemplo: Una aplicación para dibujar puede estar interesada sólo en movimientos del mouse.
- Como diseñador de una aplicación manejada por eventos, deberá escribir clases/métodos para manejar los eventos relevantes.

Manejo de eventos

- Los componentes de Swing disparan eventos de acción (ActionEvents) cuando son activados por un usuario.

- Los objetos que desean oír estos eventos deben implementar la interfaz ActionListener del paquete java.awt.event

Manejo de eventos



Manejo de eventos

- Hay que invocar al método **addActionListener** para registrar al oyente (manejador).
  - *El componente añade (registra) un objeto oyente*
- *El objeto oyente (manejador) realiza las acciones previstas*.
  - El método **actionPerformed** es el único que se define en ActionListener.
- La clase interna implementa la interfaz **ActionListener**.

Manejo de eventos

```
private void sumarBotonActionPerformed
                                (java.awt.event.ActionEvent evt)
{
    float num1, num2, result;
    num1 = Float.parseFloat(primerNumero.getText());
    num2 = Float.parseFloat(segundoNumero.getText());
    result = num1 + num2;
    resultado.setText(String.valueOf(result));
 }
```

Tipos de eventos

EVENTO	LISTENER TYPE
El usuario clica un botón, presiona Enter, escribe un texto	**ActionListener**
Usuario cierra un frame	**WindowListener**
Pulsa un botón del ratón mientras el cursor está en un componente	**MouseListener**

Tipos de eventos (II)

EVENTO	LISTENER TYPE
El usuario mueve el ratón sobre un componente	MouseMotionListener
Componente se hace visible	ComponentListener
Cambia la selección de una lista o tabla	ListSelectionListener

Tipos de eventos (III)

Interfaces	Métodos
ActionListener	actionPerformed(ActionEvent)
AdjustmentListener	adjustmentValueChanged(AdjustementEvent)
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)

Tipos de eventos (IV)

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)