



Java Swing



Elementos Básicos

Contenido

- Java Swing
- Ventanas (JFrame)
- Componentes de una ventana
- Layout Managers:
 - FlowLayout
 - GridLayout
 - BorderLayout
- Interfaces Complejas (JPanel)
- Manejo de Eventos
- Cuadros de diálogo predefinidos
- Dibujando gráficos
- Constructor de Interfaces en Netbeans



Java Swing

Características:

- Swing, es una biblioteca de interfaces gráficas de usuario (GUI) para Java
- Viene incluida con el entorno de desarrollo de Java (JDK)
- Extiende a otra librería gráfica más antigua (legacy) llamada AWT (Abstract Window Toolkit)
- Paquetes:
 - javax.swing
 - java.awt
 - java.awt.event

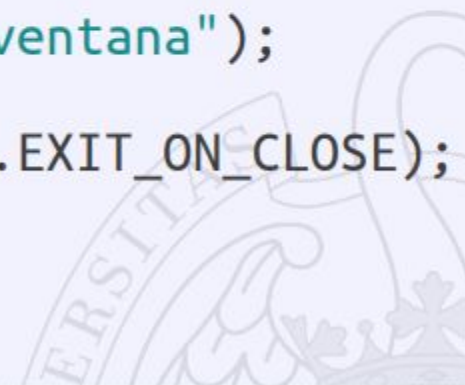
Ventanas (JFrame)

- La clase **JFrame** proporciona operaciones para manipular ventanas
- Constructores:
 - `JFrame()`
 - `JFrame(String titulo)`
- Una vez creado el objeto de la ventana, hay que:
 - Establecer su tamaño
 - Establecer la acción de cierre
 - Hacerla visible
- Acciones de cierre:
 - `JFrame.EXIT_ON_CLOSE`: Abandona aplicación
 - `JFrame.DISPOSE_ON_CLOSE`: Libera los recursos asociados a la ventana
 - `JFrame.DO_NOTHING_ON_CLOSE`: No hace nada
 - `JFrame.HIDE_ON_CLOSE`: Cierra la ventana, sin liberar sus recursos

Ventanas (JFrame)

```
import javax.swing.*;

public class VentanaTest {
    public static void main(String[] args) {
        JFrame f = new JFrame("Titulo de ventana");
        f.setSize(400, 300);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setVisible(true);
    }
}
```



Ventanas (JFrame)

```
public class MiVentana extends JFrame {  
    public MiVentana() {  
        super("Titulo de ventana");  
        setSize(400, 300);  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    }  
}  
  
public class VentanaTest {  
    public static void main(String[] args) {  
        MiVentana v = new MiVentana();  
        v.setVisible(true);  
    }  
}
```

Ejemplo

Componentes de una Ventana



Etiqueta:



JButton

JLabel

TextField

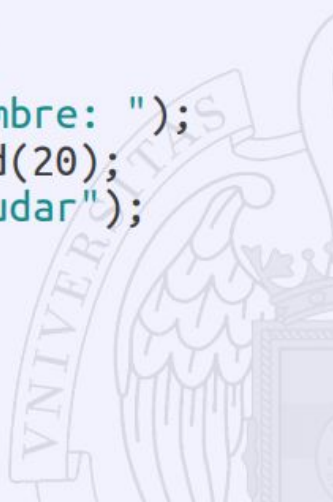
JCheckBox

JRadioButton

Nota: Luego de crear cada uno de estos componentes a través de una instancia, se debe añadir al **ContentPane** de la ventana correspondiente mediante su método **add**.

Componentes de una Ventana

```
public class MiVentana extends JFrame {  
    public MiVentana() {  
        super("Titulo de ventana");  
        setSize(400, 300);  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        Container cp = getContentPane();  
        cp.setLayout(new FlowLayout());  
        JLabel etiqueta = new JLabel("Nombre: ");  
        JTextField texto = new JTextField(20);  
        JButton boton = new JButton("Saludar");  
        cp.add(etiqueta);  
        cp.add(texto);  
        cp.add(boton);  
    }  
}
```



Ejemplo

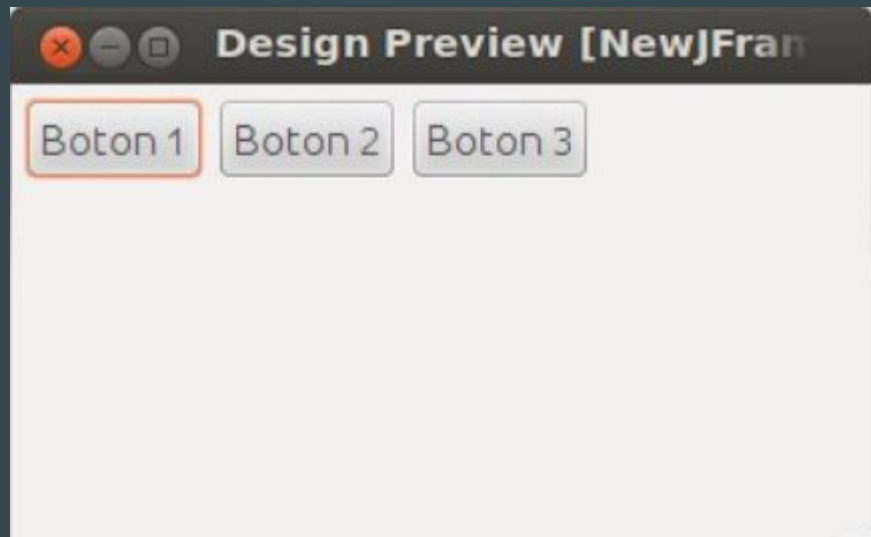
Layout Managers

- En Java no es habitual indicar explícitamente la posición de los componentes de la interfaz dentro de la ventana
- Los layout managers se encargan de colocar los componentes de la interfaz de usuario en la ventana contenedora
- Especifican la **posición** y el **tamaño** de dichos componentes:
 - FlowLayout
 - GridLayout
 - BorderLayout

FlowLayout

Características:

- Coloca los elementos uno a continuación de otro, de manera similar a la colocación de palabras en un procesador de textos
- Métodos:
 - `setAlignment(int alineación)`
 - `setHgap(int separación)`
 - `setVgap(int separación)`



GridLayout

Características:

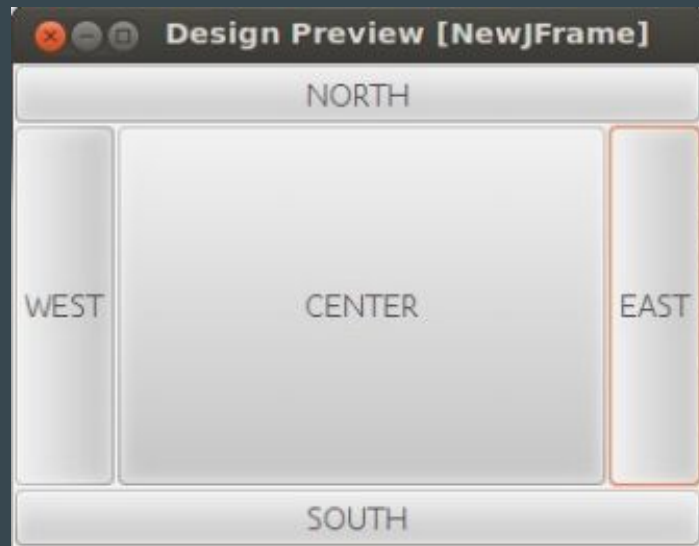
- Coloca los componentes de la interfaz en forma de rejilla
- El orden en que se añaden los componentes determina su posición en la rejilla
- Constructor:
 - `GridLayout(int filas, int columnas)`
- Métodos:
 - `setHgap(int separación)`
 - `setVgap(int separación)`



BorderLayout

Características:

- Coloca y cambia de tamaño sus componentes para que se ajusten a los bordes y parte central de la ventana.
- Métodos:
 - `setHgap(int separación)`
 - `setVgap(int separación)`
- Al añadir un elemento a la ventana, hay que especificar su colocación:



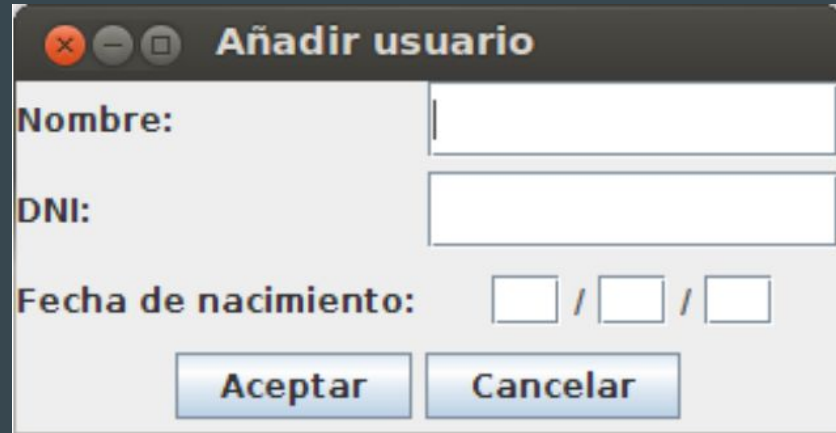
```
JButton b = new JButton(...);  
getContentPane().add(b, BorderLayout.EAST)
```

Ejemplo

Interfaces Complejas: JPanel

Características:

- Un panel es un componente con un Layout Manager propio y que puede contener varios componentes en su interior
- Constructor:
 - `JPanel()`
- Métodos:
 - `void setLayout(LayoutManager lm)`
 - `void add(JComponent componente)`



A screenshot of a Java Swing window titled "Añadir usuario". The window has a standard title bar with a close button (red X), a minimize button (grey dash), and a maximize button (grey square). The main content area is light grey and contains three input fields. The first field is labeled "Nombre:" and is a single-line text box. The second field is labeled "DNI:" and is a single-line text box. The third field is labeled "Fecha de nacimiento:" and consists of three small text boxes separated by slashes (/). At the bottom of the window, there are two buttons: "Aceptar" and "Cancelar", both with a blue gradient and white text.

Interfaces Complejas: JPanel

```
public MiVentana3() {  
    super("Añadir usuario");  
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    // Panel de fecha  
    JPanel panelFecha = new JPanel();  
    panelFecha.setLayout(new FlowLayout());  
    panelFecha.add(new JTextField(2));  
    panelFecha.add(new JLabel("/"));  
    panelFecha.add(new JTextField(2));  
    panelFecha.add(new JLabel("/"));  
    panelFecha.add(new JTextField(2));  
    // Panel de datos  
    JPanel panelDatos = new JPanel();  
    GridLayout gl = new GridLayout(3,2,0,5);  
    panelDatos.setLayout(gl);  
    panelDatos.add(new JLabel("Nombre:"));  
    panelDatos.add(new JTextField(10));  
    panelDatos.add(new JLabel("DNI:"));  
    panelDatos.add(new JTextField(10));  
    panelDatos.add(new JLabel("Fecha de nacimiento: "));  
    panelDatos.add(panelFecha);  
    ...  
}
```

```
...  
// Panel de botones  
JPanel panelBotones = new JPanel();  
panelBotones.setLayout(new FlowLayout());  
panelBotones.add(new JButton("Aceptar"));  
panelBotones.add(new JButton("Cancelar"));  
  
Container cp = getContentPane();  
cp.add(panelDatos, BorderLayout.CENTER);  
cp.add(panelBotones, BorderLayout.SOUTH);  
}
```

Ejemplo

Manejo de Eventos

Características:


- Un evento es un suceso que ocurre como consecuencia de la interacción del usuario con la interfaz gráfica:
 - Pulsación de un botón
 - Cambio del contenido en un cuadro de texto
 - Deslizamiento de una barra
 - Activación de un JCheckBox
 - Movimiento de la ventana
- La clase **JButton** tiene un método:
 - `void addActionListener(ActionListener l)`
- Que especifica el objeto (manejador de evento) que se encargará de tratar el evento de pulsación del botón
- Este objeto ha de interpretar la interfaz **ActionListener**
 - `paquete java.awt.event`

```
public interface ActionListener {  
    void actionPerformed(ActionEvent e)  
}
```

Manejo de Eventos

- Cuando el usuario pulse el botón, se llamará al método `actionPerformed` de todos los manejadores de eventos que se hayan registrado
- Métodos de **ActionEvent**:
 - `public Object getSource()`
 - `public int getModifiers()`

```
public class Manejador implements ActionListener {  
    public void actionPerformed(ActionEvent e) {  
        ...  
    }  
}
```



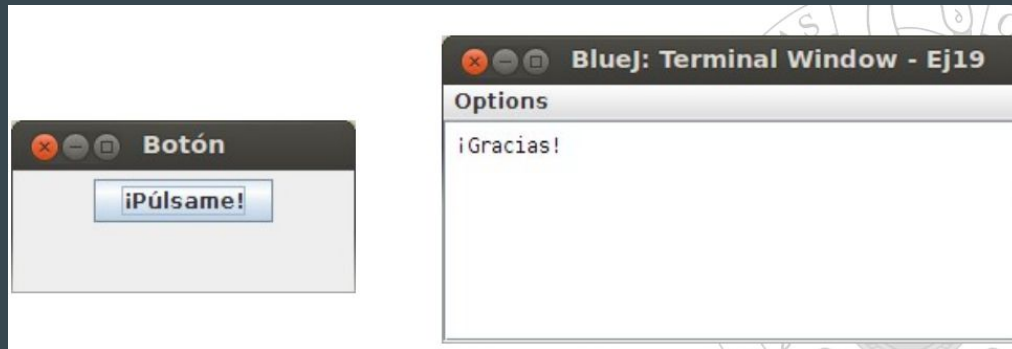
Información sobre el evento

Manejo de Eventos

```
public class BotonVentana extends JFrame {  
    public BotonVentana() {  
        super("Botón");  
        setSize(200,100);  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        Container cp = getContentPane();  
        cp.setLayout(new FlowLayout());  
        JButton boton = new JButton("¡Púlsame!");  
        boton.addActionListener(new EventoBotonPulsado());  
        cp.add(boton);  
    }  
}
```

Manejo de Eventos

```
public class EventoBotonPulsado implements ActionListener {  
    public void actionPerformed(ActionEvent e) {  
        System.out.println("¡Gracias!");  
    }  
}
```



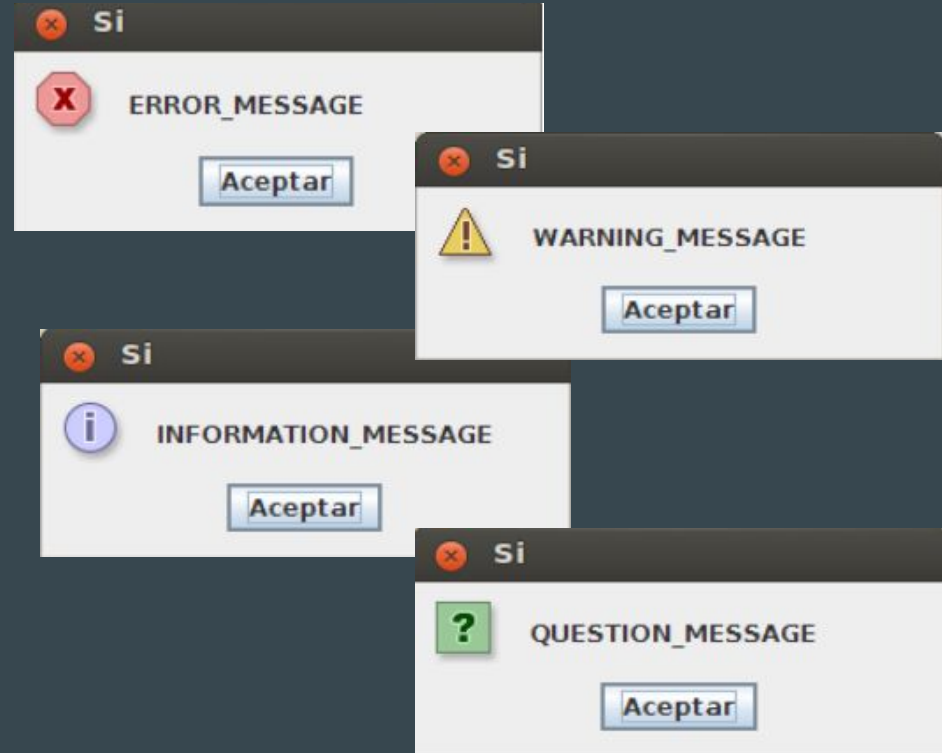
Manejo de Eventos

- `void windowActivated(WindowEvent e)`
- `void windowClosed(WindowEvent e)`
- `void windowClosing(WindowEvent e)`
- `void windowOpened(WindowEvent e)`
- `MouseListener`: Pulsaciones de botón, entradas y salidas del puntero en un componente
- `MouseMotionListener`: Movimientos del ratón dentro de un componente
- `MouseWheelListener`: Movimientos de la rueda central de un ratón
- `KeyListener`: Pulsaciones de teclas
- Más eventos:
 - <https://docs.oracle.com/javase/tutorial/uiswing/events/eventsandcomponents.html>

Ejemplo

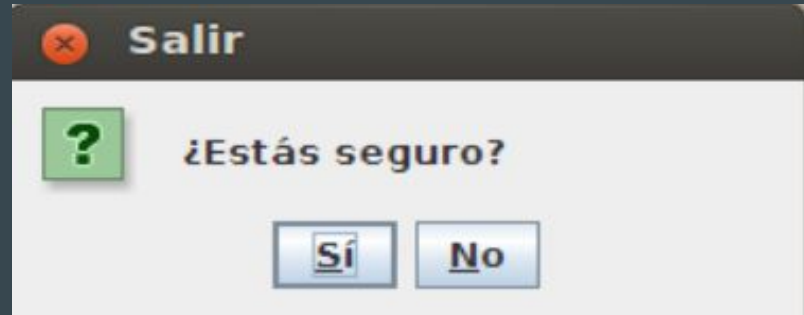
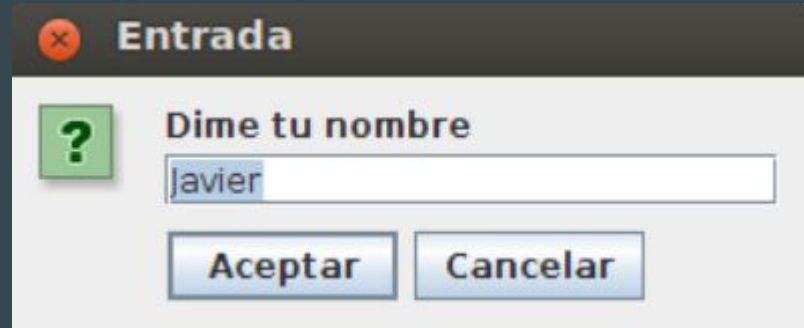
Cuadros de Diálogo Predefinidos: JOptionPane

- Paquete: javax.swing.JOptionPane
- Método: showMessageDialog
 - Component padre
 - Object mensaje
 - String tituloVentana
 - int tipoMensaje
 - ERROR_MESSAGE
 - WARNING_MESSAGE
 - INFORMATION_MESSAGE
 - QUESTION_MESSAGE



Cuadros de Diálogo Predefinidos: Confirmación

- Paquete: javax.swing.JOptionPane
- Método: showInputDialog
 - Devuelve String
 - Component padre
 - Object mensaje
 - Object valorDefecto
- Método: showConfirmDialog
 - Component padre
 - Object mensaje
 - String titulo
 - int tipoOpciones
 - int tipoMensaje



Ejemplo

Dibujando Gráficos

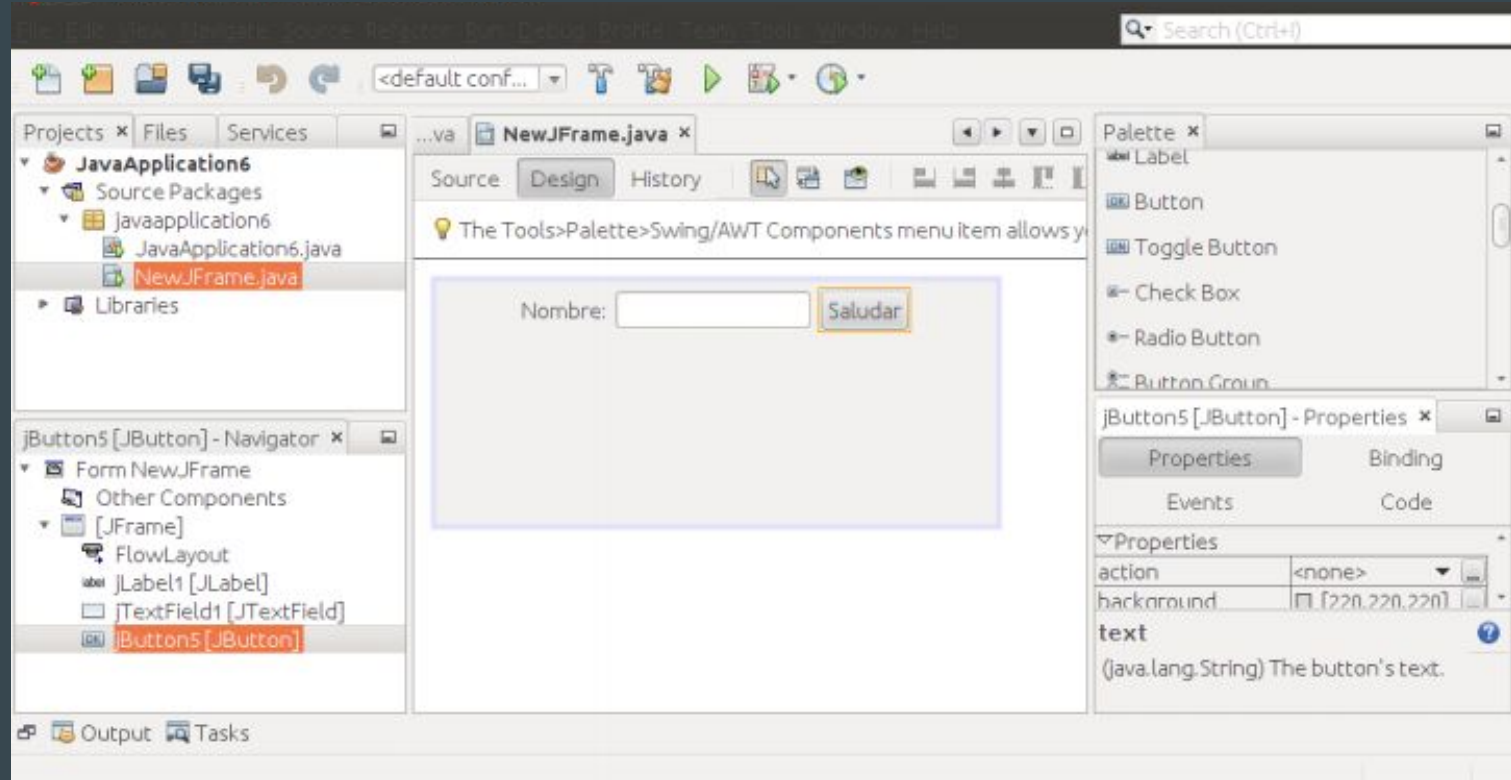
- Cada componente tiene un método llamado **paintComponent**, que se encarga de pintarlo en pantalla
- Para realizar un dibujo definido por nosotros, basta con heredar de un componente (normalmente un **JPanel**), y sobrescribir su método **paintComponent**
- Podemos dibujar:
 - Polígonos
 - Rectángulos
 - Óvalos
 - Líneas

Métodos de Graphics

- `void drawPolygon(int[] x, int[] y, int puntos)`
- `void drawRect(int x, int y, int ancho, int alto)`
- `void fillRect(int x, int y, int ancho, int alto)`
- `void drawOval(int x, int y, int ancho, int alto)`
- `void fillOval(int x, int y, int ancho, int alto)`
- `void drawString(String cad, int x, int y)`
- `void setColor(Color c)`
- `void setFont(Font f)`

Ejemplo

Constructor de Interfaces en Netbeans



Ejemplo