UT1.2. Generación de Interfaces gráficos de usuario

18 horas

Contenidos

1. Introducción al diseño de interfaces

2. **Lenguaje de programación en diseño de interfaces**

3. Generación de interfaces a partir de documentos XML

Resultados de Aprendizaje

RA1: Genera interfaces gráficos de usuario mediante editores visuales utilizando las funcionalidades del editor y adaptando el código generado	**15 %**
a) Se han analizado las herramientas y librerías disponibles para la generación de interfaces gráficos.	10% .
b) Se ha creado un interfaz gráfico utilizando las herramientas de un editor visual.	15%
c) Se han utilizado las funciones del editor para ubicar los componentes del interfaz.	10% .
d) Se han modificado las propiedades de los componentes para adecuarlas a las necesidades de la aplicación.	15% .
e) Se ha analizado el código generado por el editor visual.	10%
f) Se ha modificado el código generado por el editor visual.	10%
g) Se han asociado a los eventos las acciones correspondientes.	10%
h) Se ha desarrollado una aplicación que incluye el interfaz gráfico obtenido.	20%

2. Lenguaje de programación en diseño de interfaces

2.1. Programación de eventos

2.2. Librerías de componentes

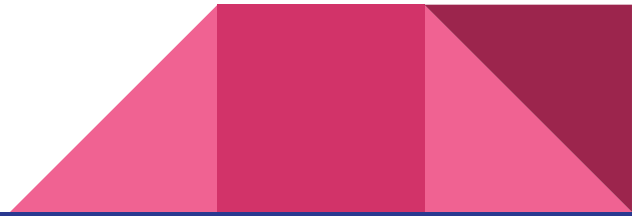
2.3. Herramientas de edición de interfaces

2.4. Contenedores

2.5. Componentes

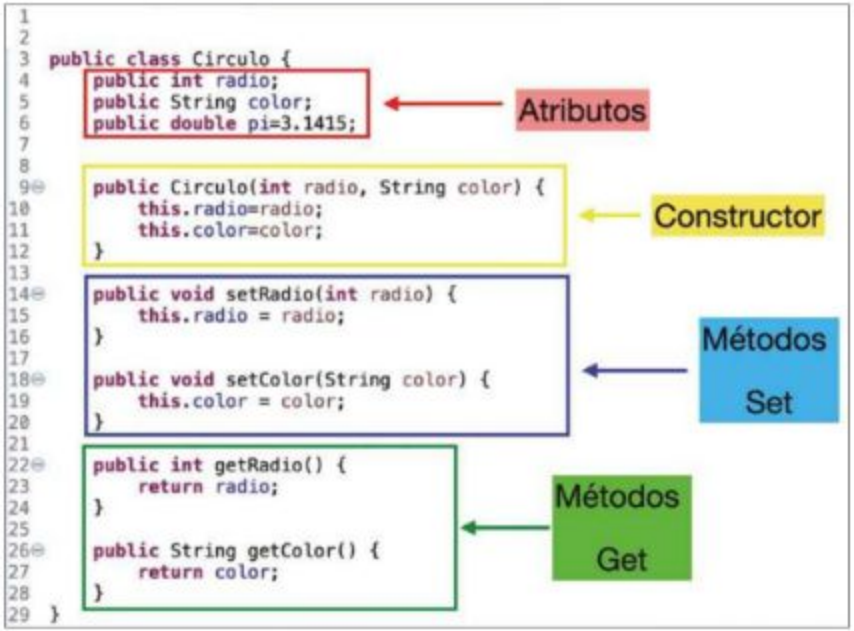
2.6. Asociación de acciones a eventos

2.7. Diálogos



Introducción

Clase java

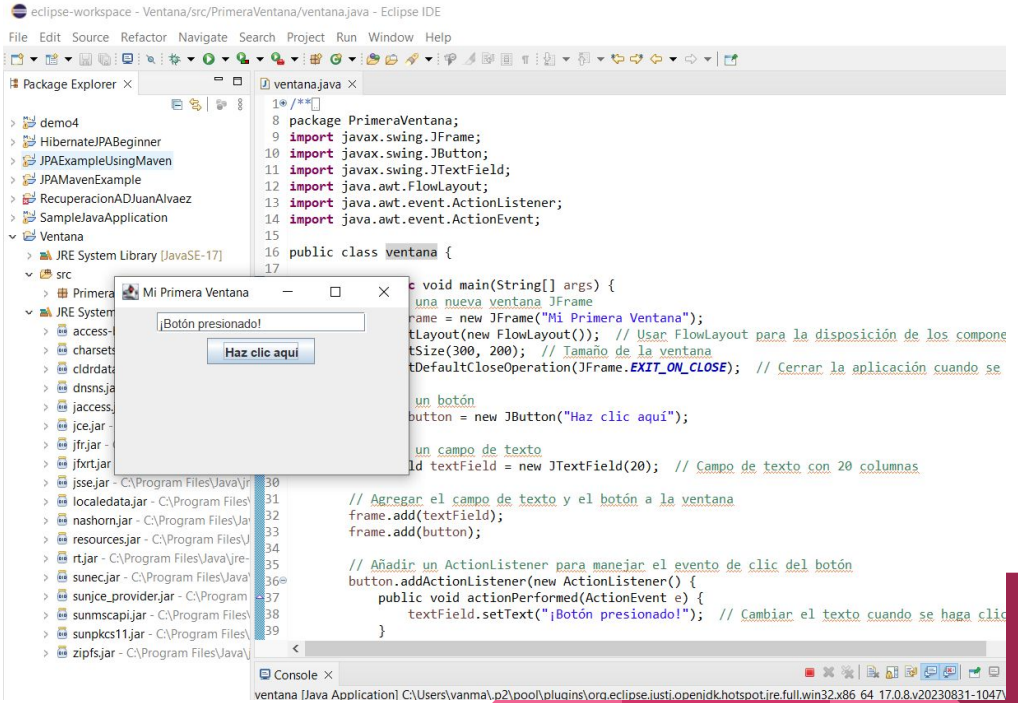


2.1. Programación de eventos

Para conectar dos o más ventanas, debe ocurrir un evento, por ejemplo pulsar un botón.

Vamos a ver un ejemplo sencillo.

Copia el siguiente código JAVA (en el siguiente slide) para crear una ventana.



Ejemplo ventana

```
package PrimeraVentana;
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JTextField;
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class ventana {

    public static void main(String[] args) {
        // Crear una nueva ventana JFrame
        JFrame frame = new JFrame("Mi Primera Ventana");
        frame.setLayout(new FlowLayout());  // Usar FlowLayout para la disposición de los componentes
        frame.setSize(300, 200);  // Tamaño de la ventana
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  // Cerrar la aplicación cuando se cierre la ventana

        // Crear un botón
        JButton button = new JButton("Haz clic aquí");

        // Crear un campo de texto
        JTextField textField = new JTextField(20);  // Campo de texto con 20 columnas

        // Agregar el campo de texto y el botón a la ventana
        frame.add(textField);
        frame.add(button);

        // Añadir un ActionListener para manejar el evento de clic del botón
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                textField.setText("¡Botón presionado!");  // Cambiar el texto cuando se haga clic
            }
        });

        // Hacer la ventana visible
        frame.setVisible(true);
    }
}
```

Configurar Eclipse

IMPORTANTE

Eclipse no incluye por defecto un GUI Builder como el que si trae el otro IDE, pero existe el plugin windowbuilder, para instalarlo ve a Help >> Install New Software, si estas usando Eclipse Neon pega en la caja de texto Work with ésta URL:
http://download.eclipse.org/windowbuilder/WB/integration/4.6/

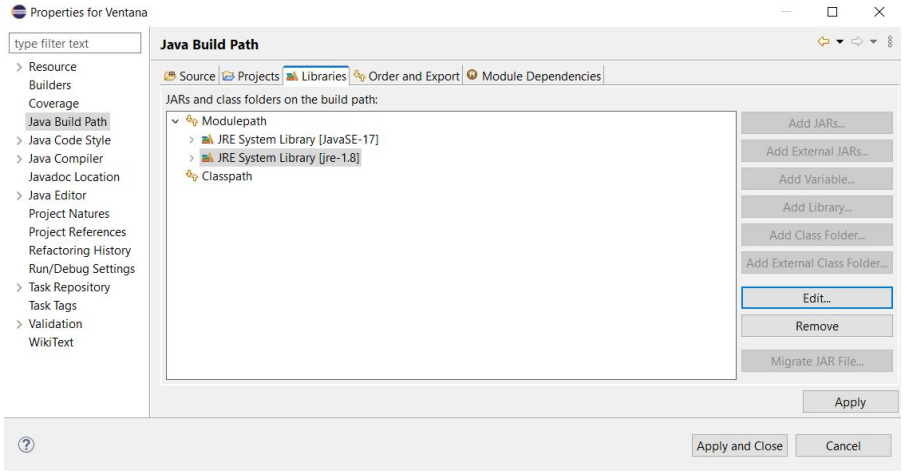
Asegúrate de que tu proyecto esté utilizando el JDK/JRE correcto y esté configurado correctamente en Eclipse:

Haz clic derecho sobre tu proyecto en el explorador de Eclipse.

Selecciona Properties.

Ve a Java Build Path.

En la pestaña Libraries, asegúrate de que el JRE System Library esté configurado correctamente (deberías ver algo como JavaSE-1.8 o superior). Si no está configurado, añade la biblioteca haciendo clic en Add Library y selecciona JRE System Library.



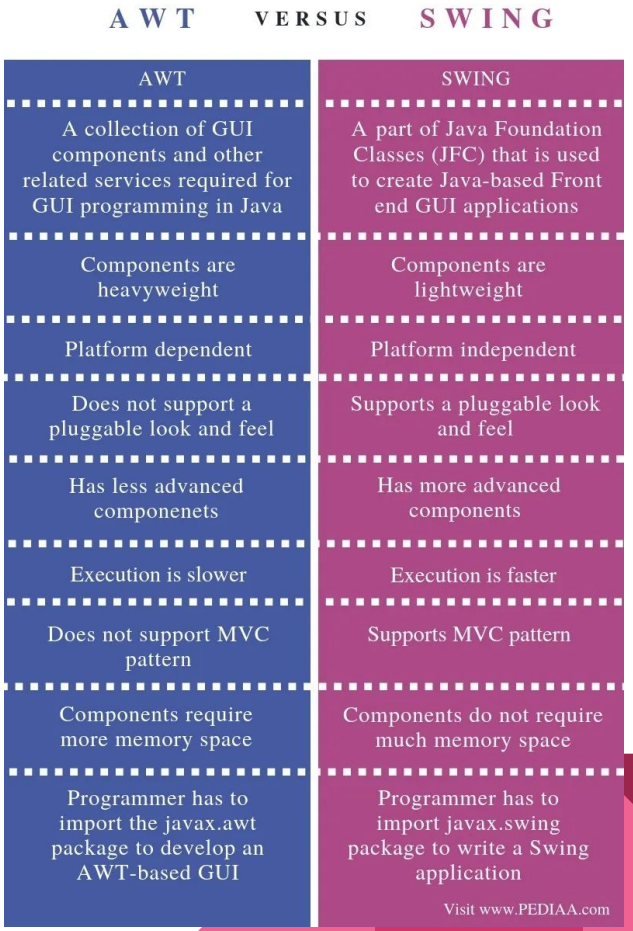
2.2. Librerías de componentes

En Java se utilizan las librerías AWT (Abstract Windows Toolkit) y Swing.

import javax.swing.*;

import java.awt.*; → Clase Component

→ Clase Container



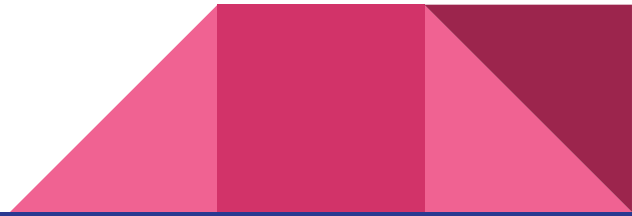
2.3. Herramientas de edición de interfaces

Elige un IDE para programar nuestras aplicaciones en Java:

- IntelliJ IDEA
- Eclipse
- NetBeans
- …

En cada empresa se utiliza un IDE, un Framework, un lenguaje de programación…

Alternativas a Java/Swing



2.4. Contenedores

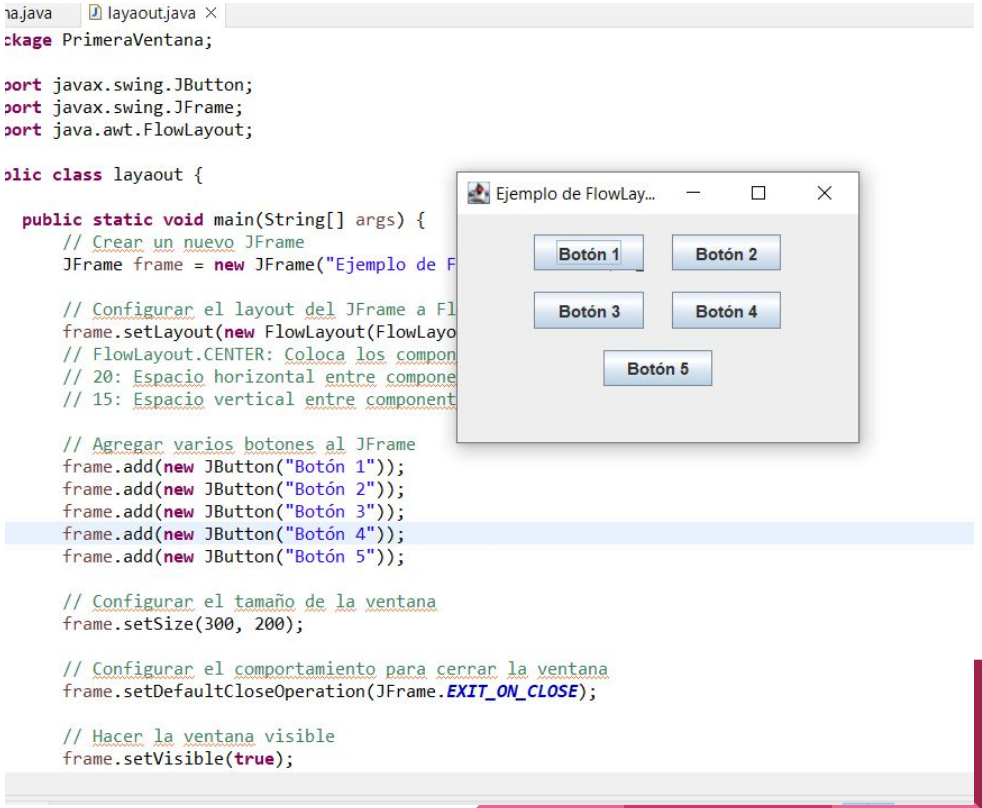
Usar contenedores permite implementar componentes que pueden contener componentes, muy útil para el diseño de interfaces porque se puede determinar la posición exacta de cada uno de los elementos.

Layaout Manager → Permite modificar el tamaño y la posición de los componentes.

> **FlowLayaout** → Sitúa los elementos en la misma fila, se puede elegir el tipo de alineación (setAlignment), la distancia de separación (en vertical, setVgap, en horizontal, set Hgap)

Prueba este código

```
package PrimeraVentana;

import javax.swing.JButton;
import javax.swing.JFrame;
import java.awt.FlowLayout;

public class layaout {

    public static void main(String[] args) {
        // Crear un nuevo JFrame
        JFrame frame = new JFrame("Ejemplo de FlowLayout");

        // Configurar el layout del JFrame a FlowLayout
        frame.setLayout(new FlowLayout(FlowLayout.CENTER, 20, 15));
        // FlowLayout.CENTER: Coloca los componentes centrados
        // 20: Espacio horizontal entre componentes
        // 15: Espacio vertical entre componentes

        // Agregar varios botones al JFrame
        frame.add(new JButton("Botón 1"));
        frame.add(new JButton("Botón 2"));
        frame.add(new JButton("Botón 3"));
        frame.add(new JButton("Botón 4"));
        frame.add(new JButton("Botón 5"));

        // Configurar el tamaño de la ventana
        frame.setSize(300, 200);

        // Configurar el comportamiento para cerrar la ventana
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Hacer la ventana visible
        frame.setVisible(true);
    }
}
```



2.4. Contenedores

GridLayout → Permite colocar los componentes siguiendo un patrón de filas y columnas.

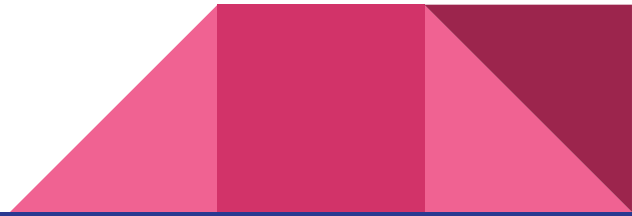
```
frame.setLayout(new GridLayout(3, 2, 10, 10));
```

`3`: Número de filas.

`2`: Número de columnas.

`10`: Espacio horizontal entre los componentes.

`10`: Espacio vertical entre los componentes.



Prueba este código

```
import javax.swing.JButton;
import javax.swing.JFrame;
import java.awt.GridLayout;

public class GridLayoutExample {

    public static void main(String[] args) {
        // Crear un nuevo JFrame
        JFrame frame = new JFrame("Ejemplo de GridLayout");

        // Configurar el GridLayout: 3 filas y 2 columnas
        frame.setLayout(new GridLayout(3, 2, 10, 10));
        // 10 es el espacio horizontal y vertical entre las celdas

        // Agregar 6 botones
        frame.add(new JButton("Botón 1"));
        frame.add(new JButton("Botón 2"));
        frame.add(new JButton("Botón 3"));
        frame.add(new JButton("Botón 4"));
        frame.add(new JButton("Botón 5"));
        frame.add(new JButton("Botón 6"));

        // Configurar el tamaño de la ventana
        frame.setSize(300, 200);

        // Configurar el comportamiento para cerrar la ventana
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Hacer visible la ventana
        frame.setVisible(true);
    }
}
```

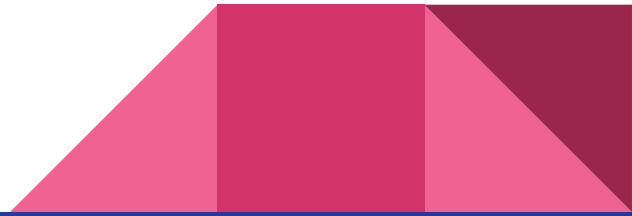
a.java layaout.java gridlayaout.java ×

```
kage PrimeraVentana;

ort javax.swing.JButton;
ort javax.swing.JFrame;
ort java.awt.GridLayout;

lic class gridlayaout {

 public static void main(String[] args) {
     // Crear un nuevo JFrame
     JFrame frame = new JFrame("Ejemplo de GridLayout");

     // Configurar el GridLayout: 3 filas y 2 columnas
     frame.setLayout(new GridLayout(3, 2, 10, 10));
     // 10 es el espacio horizontal y v

     // Agregar 6 botones
     frame.add(new JButton("Botón 1"));
     frame.add(new JButton("Botón 2"));
     frame.add(new JButton("Botón 3"));
     frame.add(new JButton("Botón 4"));
     frame.add(new JButton("Botón 5"));
     frame.add(new JButton("Botón 6"));

     // Configurar el tamaño de la vent
     frame.setSize(300, 200);

     // Configurar el comportamiento para cerrar la ventana
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

     // Hacer visible la ventana
     frame.setVisible(true);
 }
```

2.4. Contenedores

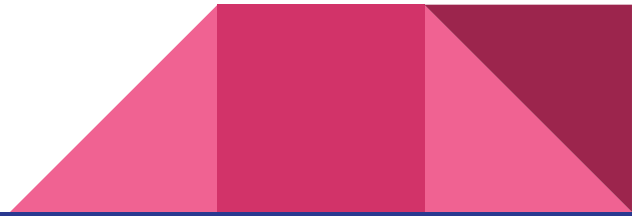
BorderLayout → Permite situar los elementos en el centro, arriba, abajo….

GridBagLayout → Permite ser más preciso en la posición de cada elemento en la rejilla que utilizando GridLayout.



Actividad 1

Haz un código similar a los vistos en clase para probar los contenedores **BorderLayout** y **GridBagLayout**.



2.5. Componentes

JButton → Crea botones.

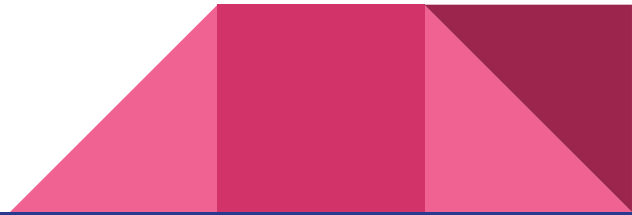
JLabel → Crea etiquetas.

JTextField → Crea caja de texto con un tamaño definido.

JCheckBox → Crea un CheckBox.

JRadioButton → Crea una casilla de selección de varias opciones para que el usuario sólo pueda escoger una.

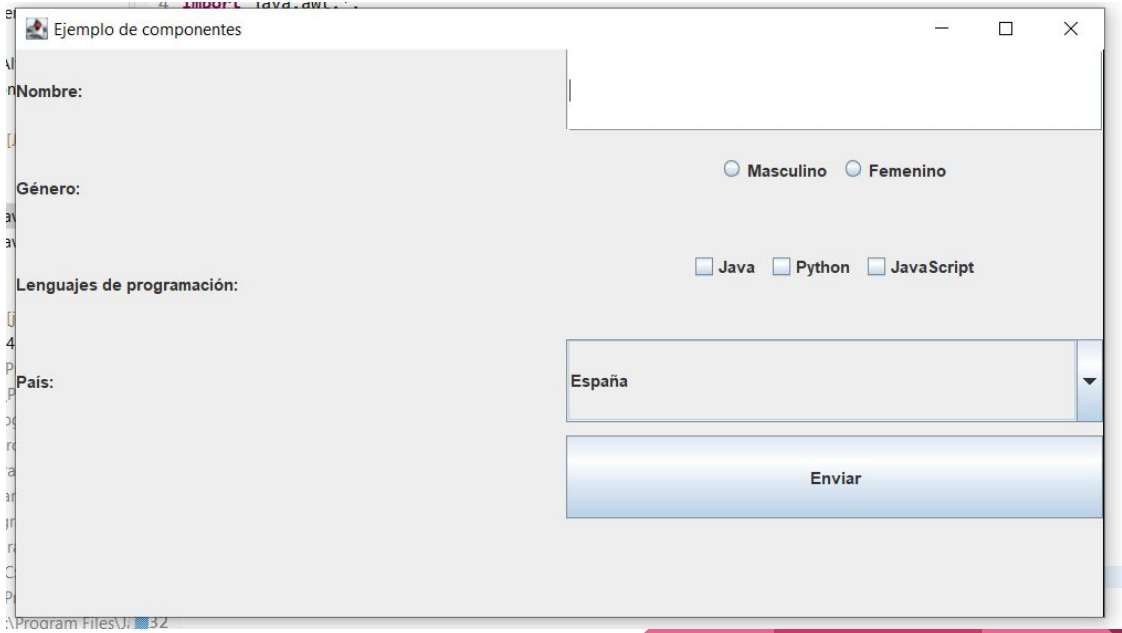
JComboBox → Crea menús desplegables.



Prueba este código

Prueba el código que encontrarás en el aula:

component.java



2.6. Asociación de acciones a eventos

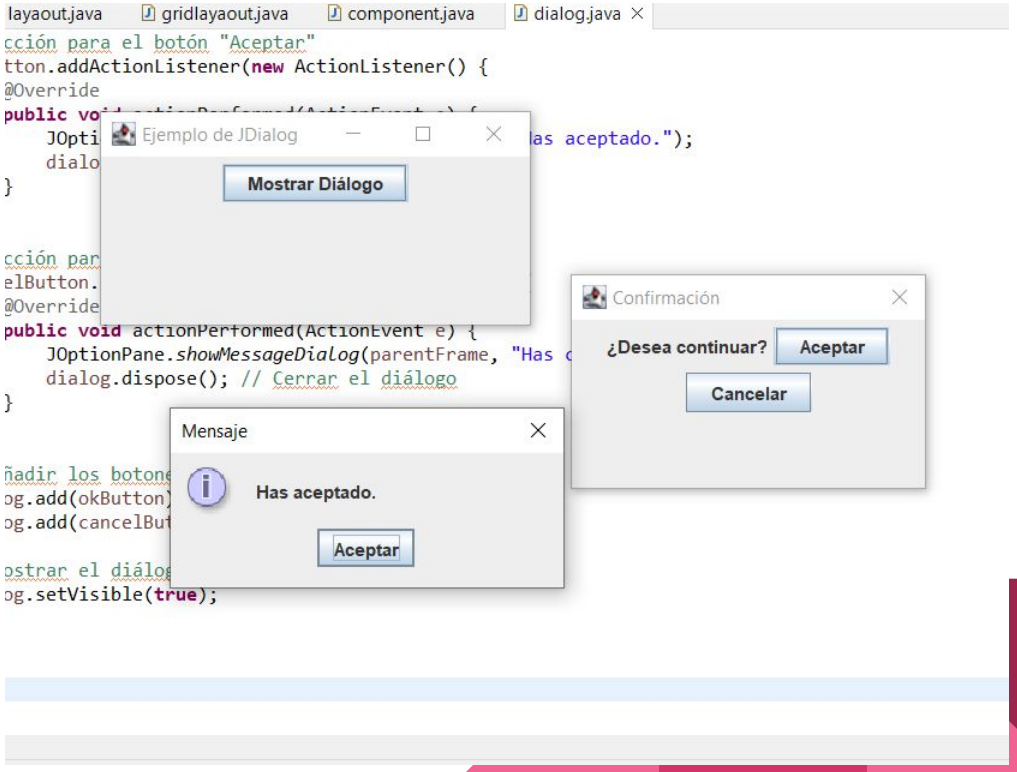
addActionListener → Permite añadir una acción a un componente.

Por ejemplo:

```
okButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            JOptionPane.showMessageDialog(parentFrame, "Has aceptado.");
            dialog.dispose(); // Cerrar el diálogo
        }
    });
```

2.7. Diálogos

JDialog →Permite crear ventanas secundarias.



Actividad 2

Prueba y estudia el código **dialog.java** que encontrarás en el Aula Virtual.

Investiga las acciones y los eventos y modifícalo para añadir una nueva ventana.

Investiga cómo cambiar el tipo de letra, tamaño, negrita, etc. y modifica el programa.