

MÓDULO 07

Desarrollo de interfaces

CFGS Técnico Superior en Desarrollo de Aplicaciones Multiplataforma

UF 01

Diseño e implementación de interfaces

Tema 1. Confección de interfaces de usuario

Tema 1. CONFECCIÓN DE INTERFACES DE USUARIO

- 01.** LIBRERÍAS DE COMPONENTES DISPONIBLES PARA LOS DISTINTOS SSOO Y LENGUAJES DE PROGRAMACIÓN. CARACTERÍSTICAS
- 02.** ENLACES DE COMPONENTES A ORÍGENES DE DATOS
- 03.** COMPONENTES: CARACTERÍSTICAS Y CAMPOS DE APLICACIÓN
- 04.** EVENTOS, ESCUCHADORES Y ACCIONES A EVENTOS
- 05.** EDICIÓN DEL CÓDIGO GENERADO POR LAS HERRAMIENTAS DE DISEÑO
- 06.** CLASES, PROPIEDADES, MÉTODOS

CONFECCIÓN DE INTERFACES DE USUARIO

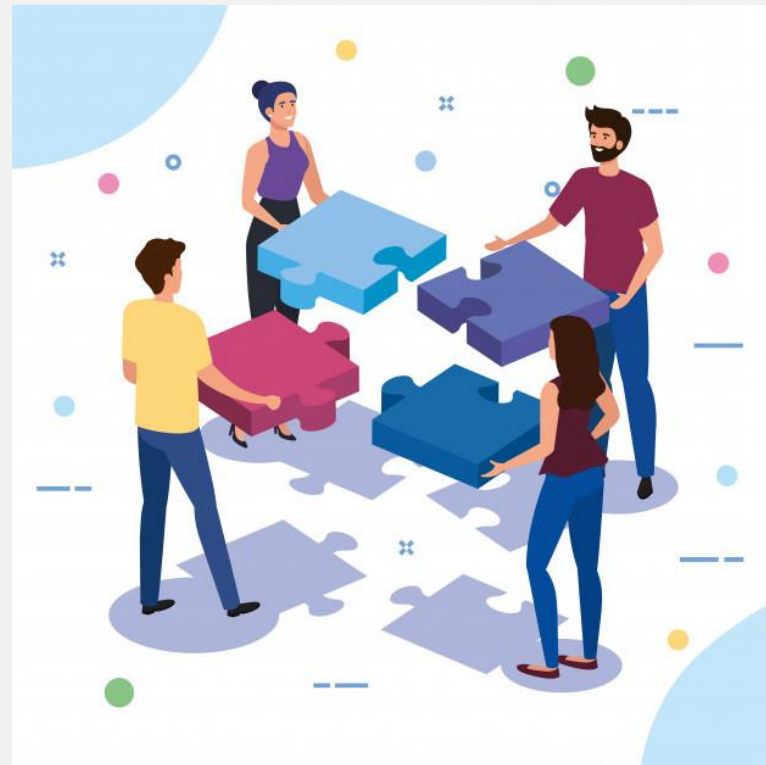
¿Qué es un IDE?

Se llama así a los entornos de programación que han sido empaquetados como un programa de aplicación.

Es una solución software que incluye:

- Editor de código
- Compilador
- Depurador
- Constructor de interfaz

Algunos ejemplos de IDE: Visual Studio, Netbeans, Eclipse



Tipos de IDE

Visual Studio

Es una IDE creada por Microsoft que se basa en centrar su núcleo de desarrollo alrededor de esta solución.



Netbeans

Puede instalarse en cualquier sistema operativo gracias al lenguaje Java necesitando la existencia de una máquina virtual.



Eclipse

Herramienta de código abierto asociada al lenguaje java. Además existen variables plantillas para crear proyectos en otros lenguajes de programación diferentes como php, html5,...



Tipos de IDE

Android Studio

Es una IDE es para la plataforma Android. Fue anunciado que reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para dispositivos con este SO.



**Android
Studio**

IntelliJ IDEA

Puede instalarse en cualquier sistema operativo gracias al lenguaje Java necesitando la existencia de una máquina virtual.



IntelliJ IDEA

PyCharm

Herramienta de código abierto asociada al lenguaje Python. Además existen variables plantillas para crear proyectos en otros lenguajes de programación diferentes como php, html5,...



PyCharm



Ventajas

- Aplicaciones para todas las plataformas
- Son gratuitas
- Libertad de copia, modificación, mejora...
- El usuario no depende del autor del software



Desventajas

- Interfaces menos amigables
- Menos compatibilidad con el hardware



Ventajas

- Compatibilidad con el hardware
- Facilidad de instalación
- Interfaces gráficas mejor diseñada



Desventajas

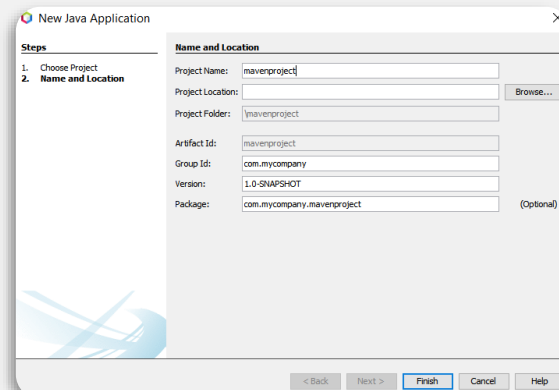
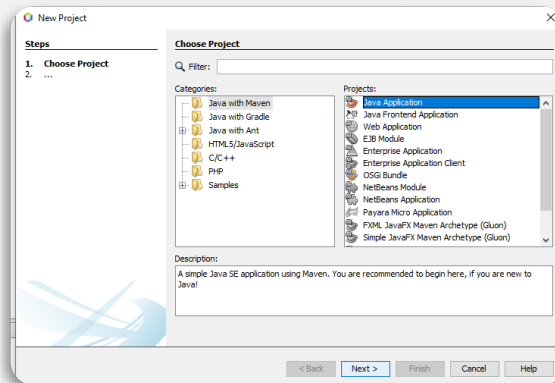
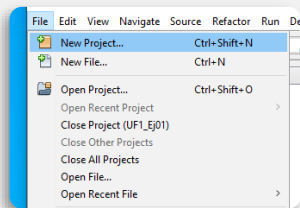
- Desarrollado para Windows
- No puede ser modificado por el usuario final (no código libre)
- Coste de aplicaciones mayor
- Soporte exclusivo del propietario

Crear Proyecto Interfaz Visual en Netbeans

Crear New Project

Java with Maven/
Java Application

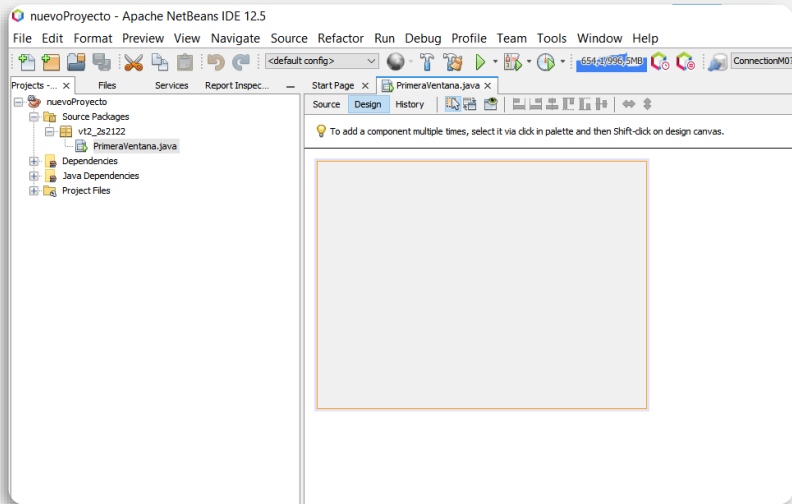
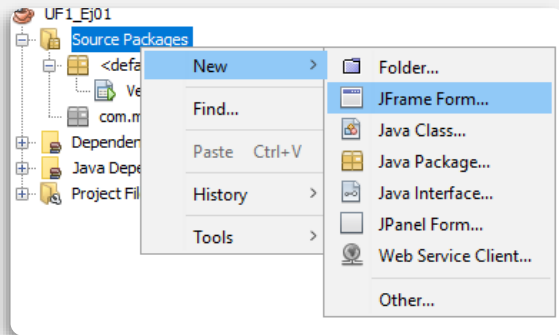
Formulario
Crear Proyecto



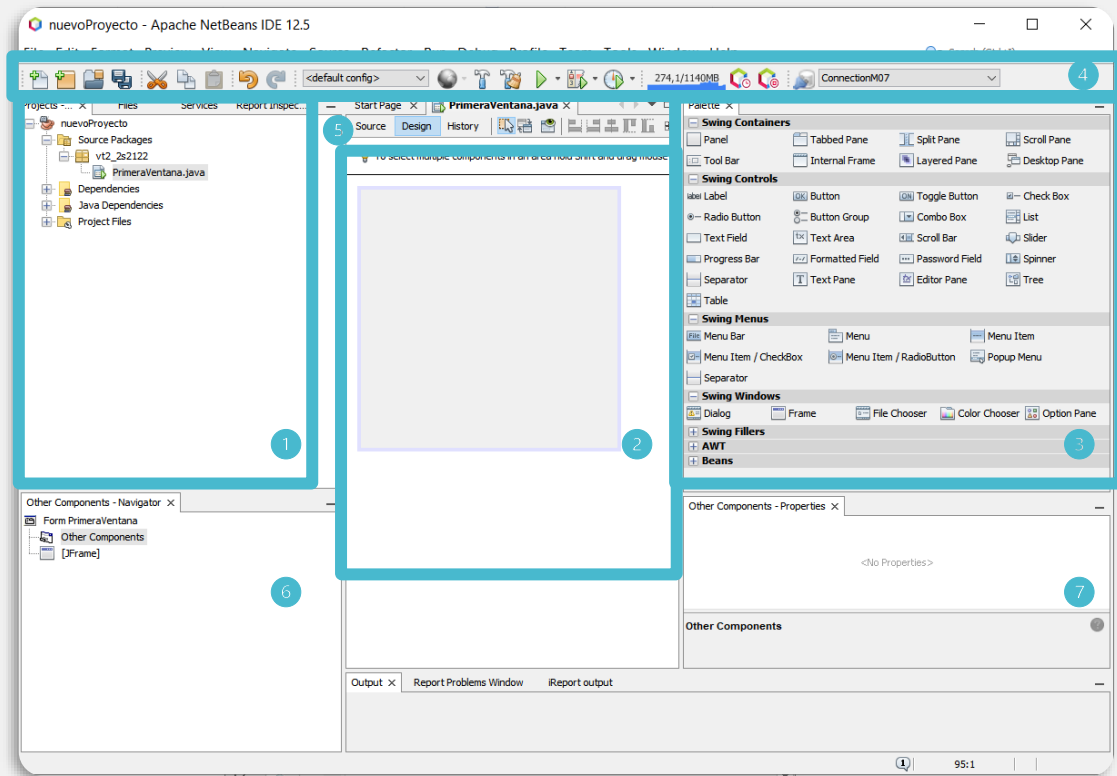
Crear Proyecto Interfaz Visual en Netbeans

Añadir JFrame
Form

NetBeans
Proyecto Creado

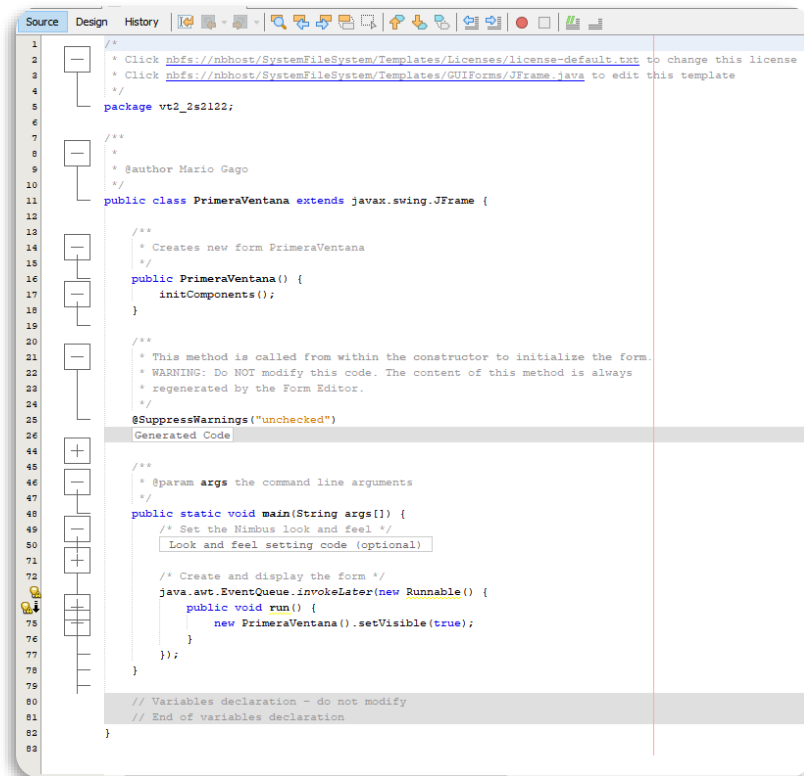


Crear Proyecto Interfaz Visual en Netbeans



1. Panel de Proyectos
2. Area principal de trabajo
3. Paleta de Objetos gráficos
4. Menú Principal y Barra de herramientas
5.
 - Vista Código fuente
 - Vista Diseño
 - Modo Selección
 - Modo conexión
 - Vista Preliminar
 - Barra de herramientas del Editor Gráfico
6. Navegador Componentes
7. Ventana de Propiedades

Pestaña Código en Netbeans



```

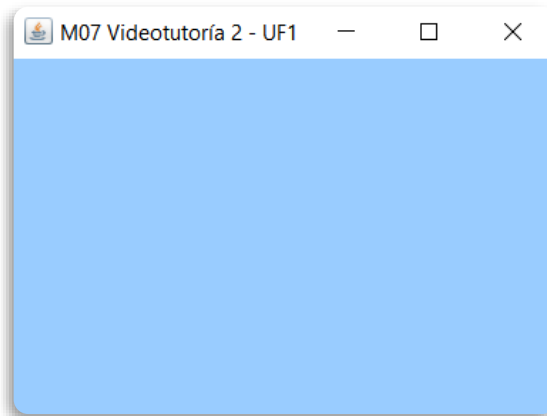
1  /**
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
4   */
5  package vt2_2s2122;
6
7  /**
8   *
9   * @author Mario Gago
10  */
11  public class PrimeraVentana extends javax.swing.JFrame {
12
13      /**
14       * Creates new form PrimeraVentana
15       */
16      public PrimeraVentana() {
17          initComponents();
18      }
19
20      /**
21       * This method is called from within the constructor to initialize the form.
22       * WARNING: Do NOT modify this code. The content of this method is always
23       * regenerated by the Form Editor.
24       */
25      @SuppressWarnings("unchecked")
26      Generated Code
27
28      /**
29       * @param args the command line arguments
30       */
31      public static void main(String args[]) {
32          /* Set the Nimbus look and feel */
33          Look and feel setting code (optional)
34
35          /* Create and display the form */
36          java.awt.EventQueue.invokeLater(new Runnable() {
37              public void run() {
38                  new PrimeraVentana().setVisible(true);
39              }
40          });
41      }
42
43      // Variables declaration - do not modify
44      // End of variables declaration
45  }
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83

```

Propuesta de ejercicio 1

Crea un JFrame con las siguientes características:

- Posición(x,y):
(10,20)
- Tamaño:
400x300px
- Título de la ventana:
M07 Videotutoría 2 – UF1
- Color de fondo:
#99CCFF
- Cerrar al salir



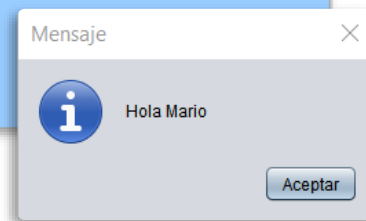
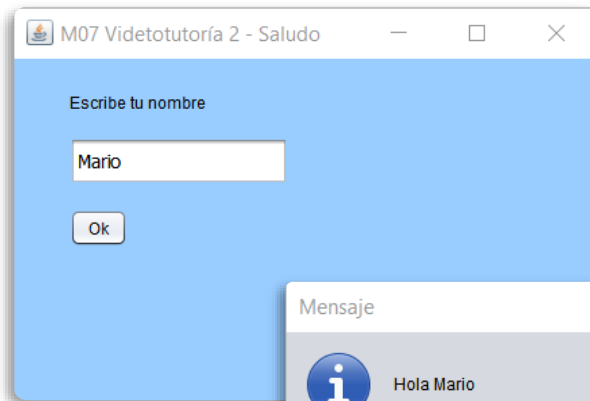
¡TÚ PUEDES!



Propuesta de ejercicio 2

Al JFrame anterior, o crear uno nuevo:

- Tiene que tener al menos:
 - Etiqueta
 - Campo de texto
 - Botón
- Usuario introduce Nombre y al pulsar ok, se muestra una ventana con el saludo



Si quieres practicar más, puedes añadir más componentes, modificar el JOptionPanel ,...



¡TÚ PUEDES!



¿DUDAS?



MÓDULO 07

Desarrollo de interfaces

CFGS Técnico Superior en Desarrollo de Aplicaciones Multiplataforma

Librerías de Componentes

AWT (Abstract Window Toolkit)

- Diseñada en Java puro y es la base de librería SWING
- Buen funcionamiento, a pesar de no tener controles demasiados avanzados
- Se utiliza para desarrollo de diseños prácticos y eficientes
- El aspecto dependerá del S.O. que se utilice
- Hoy en día, considerada obsoleta

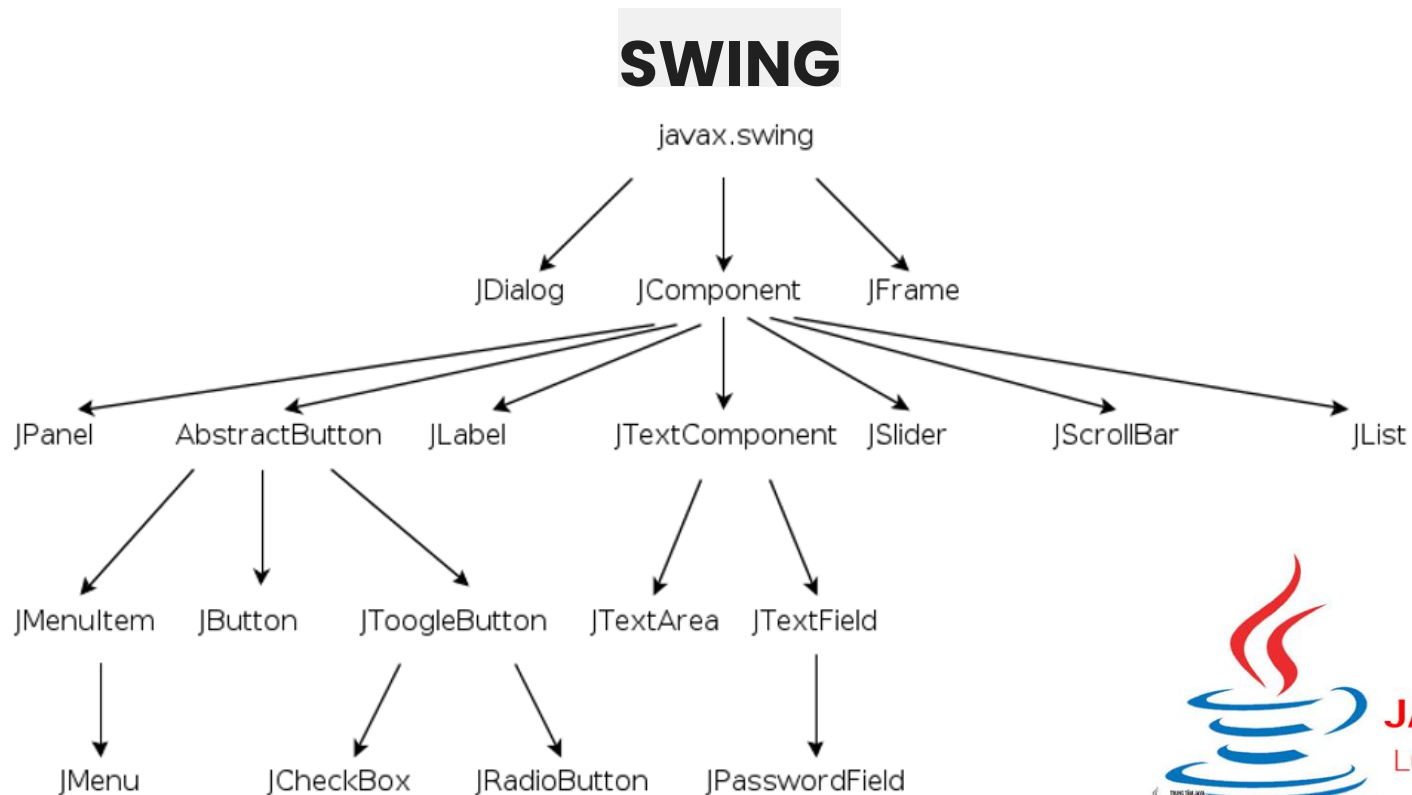


Librerías de Componentes

SWING

- Diseñada en Java puro y creada a partir de la librería AWT
- Mejora y da respuesta a los inconvenientes de AWT
- Conjunto de funcionalidades más completos
- A partir de Java 5 y 6 se convierte en Framework de desarrollo de interfaces. Cuenta con controles de gran funcionalidad.
- Inconvenientes: no cuenta con filtrado y organización de datos en forma de tabla y árbol

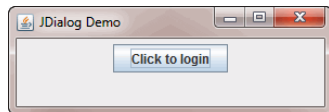




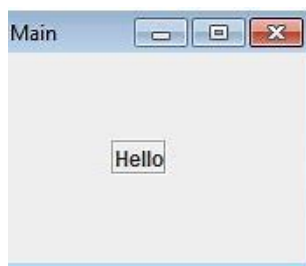
SWING



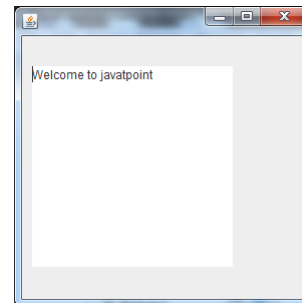
JDialog



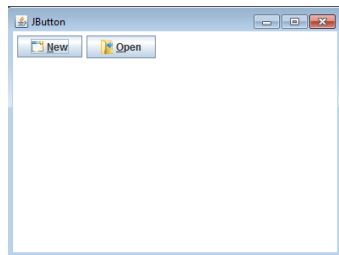
JLabel



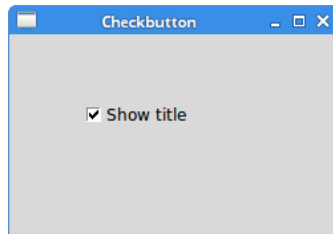
JTextArea



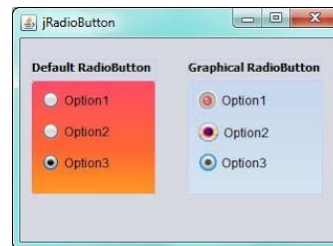
JButton



JCheckBox



JRadioButton





Ventajas

- Dibuja sus propios componentes
- El diseño en Java puro posee menos limitaciones de plataforma.
- El desarrollo de componentes Swing es más activo.
- Los componentes de Swing soportan más características



Desventajas

- Los componentes son más lentos porque están hechos en java puro
- No siempre tienen el mismo aspecto que en el sistema donde fueron diseñados

Librerías de Componentes

SWT

- Creada por IBM para el entorno de desarrollo Eclipse, mejorando la versión Swing que existía en ese momento.
- Estas interfaces no se pueden ejecutar en todas las plataformas.
- Uno de sus inconvenientes es que la API que proporciona la librería es bastante complicada de utilizar y no muy intuitiva.

SwingX

- Librería que está basada en Swing y utilizada para desarrollar aplicaciones RIA (Rich Internet Application)*.
- Gran parte de sus componentes se desarrollan sobre los existentes en la librería Swing.
- Diseñada en Java puro.
- Puede utilizarse en diferentes plataformas.

JavaFX:

- Desarrollada por Java/Oracle, siendo una plataforma open source.
- Basada principalmente en el desarrollo de aplicaciones RIA (rich internet application), permite ser utilizada en diferentes plataformas y dispositivos.
- En esta librería se agrupan las tecnologías conocidas como JavaFX Mobile y JavaFX Script.

¿DUDAS?



MÓDULO 07

Desarrollo de interfaces

CFGS Técnico Superior en Desarrollo de Aplicaciones Multiplataforma

Componentes: características y campos de aplicación

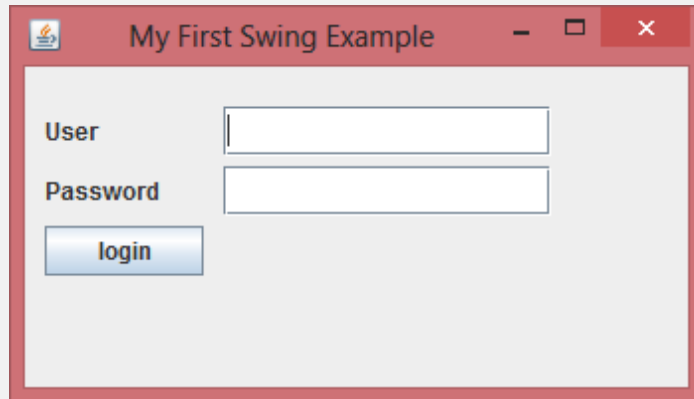
¿Qué es un componente?

- Es un objeto escrito de acuerdo con unas especificaciones, las cuales hacen que el objeto se convierta en componente adquiriendo características como, por ejemplo, la reusabilidad

Objetivo de la POC

- Reutilizar componentes ya diseñados y testados para desarrollar así las aplicaciones de una forma más rápida y robusta.

Diseñar un componente reutilizable conlleva gran esfuerzo y atención (bien documentado, preparado para uso de maneras imprevistas, probado en profundidad, robusto, etc.)



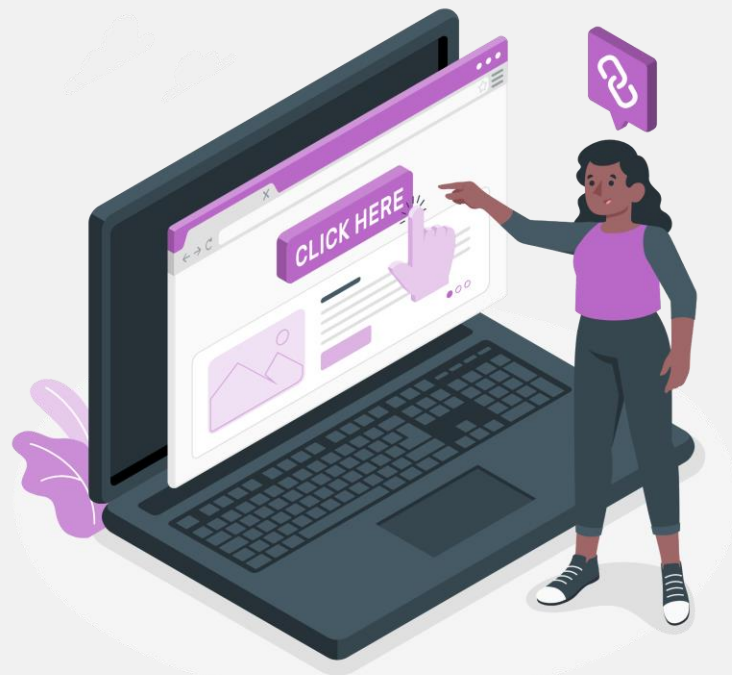
Eventos, escuchadores y acciones a eventos

¿Qué es un evento?



- Acción que puede realizar un usuario
- Está asociada a un componente en concreto
- Cuando se realiza un evento se producen acciones que tiene definidas

Ejemplos: Al pulsar un botón, se produce un sonido; Al salir de un campo de texto en un formulario



EJEMPLO GESTIÓN EVENTOS DEL RATÓN

MÉTODOS	Public void mouseClicked(MouseEvent e)	
	Public void mouseEntered(MouseEvent e)	
	Public void mouseExited(MouseEvent e)	
	Public void mousePressed(MouseEvent e)	
EVENTOS	mouseClicked	Pinchar y soltar
	mouseEntered	Entra en un componente con el puntero
	mouseExited	Sale de un componente con el puntero
	mousePressed	Presiona el botón
	mouseReleased	Suelta el botón
ESCUCHADOR	MOUSELISTENER → Cuando se lleva a cabo alguna acción con el ratón	



Eventos, escuchadores y acciones a eventos

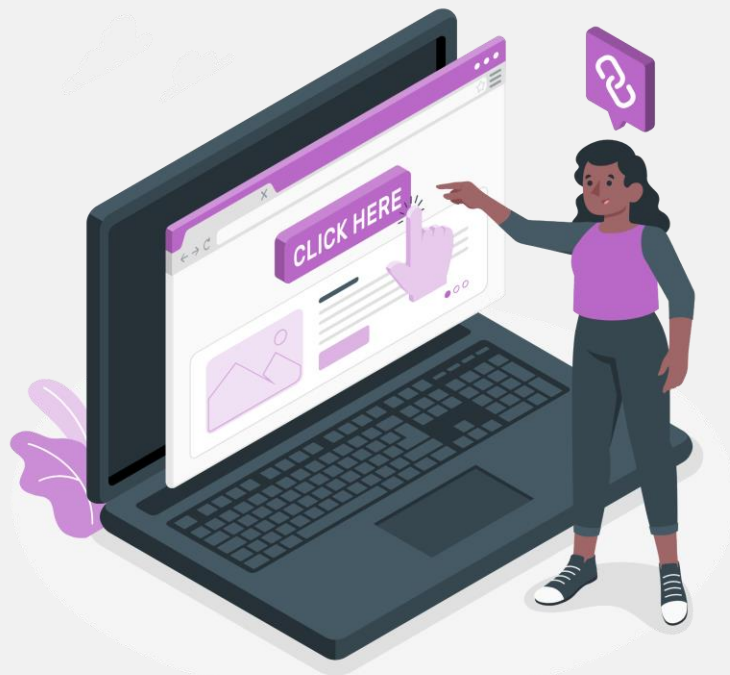
¿Qué es un listener?



- Controlan los eventos
- Esperan a que el evento se produzca (mientras están "escuchando")
- Según el evento necesitaremos un listener
- Cada listener tiene unos métodos que debemos implementar

¿Usar un listener?

- Implementar la interfaz del escuchador.
- Registrar el escuchador en el objeto que genera el evento, indicándole el objeto que los recogerá.
- Implementar los métodos callback correspondientes



EJEMPLO

```

20
21 public class ClassA {
22
23     /**
24      * @param args the command line arguments
25      */
26     public static void main(String[] args) {
27         // TODO code application logic here
28         JFrame miFrame = new JFrame ("Estamos creando código");
29         miFrame.setLayout ( new FlowLayout());
30         miFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
31         miFrame.setSize(300,300);
32         JButton button1 = new JButton("Este es un botón");
33         ClassB miClase = new ClassB();
34         button1.addActionListener(miClase);
35         miFrame.getContentPane().add(button1);
36         miFrame.setVisible(true);
37     }
38 }
39
40
41
42 class ClassB implements ActionListener{
43
44     public void actionPerformed(ActionEvent e){
45         JOptionPane.showMessageDialog(null, "Algo esta pasando");
46     }
47 }

```

Evento

Listener



JAVA SWING
LOOK AND FELL

Propuesta de ejercicio 3

Crea un formulario como el de la imagen.

A screenshot of a Java Swing window titled "M07 - UF1 - VT3 - Formulario". The window has a light blue background and contains two text input fields. The first field is labeled "Nombre:" and the second is labeled "Apellidos:". Below the fields is a button labeled "Aceptar". The window title bar shows "M07 Videotutoría 3 ...".

Propuesta de ejercicio 3.1

Al ejercicio anterior propuesto añadirle funcionalidad al botón aceptar; para que después de rellenar los datos, al pulsar Aceptar, de la bienvenida al usuario donde pone Usuario



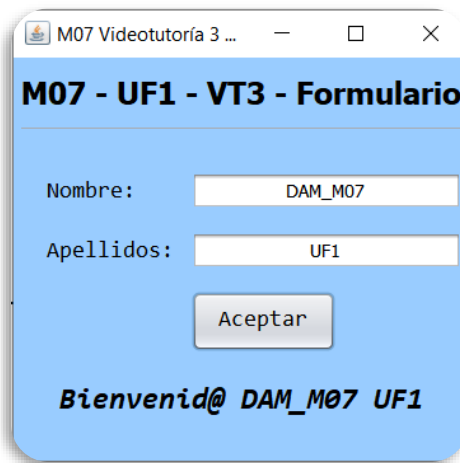
M07 Videotutoría 3 ...

M07 - UF1 - VT3 - Formulario

Nombre:

Apellidos:

Aceptar



M07 Videotutoría 3 ...

M07 - UF1 - VT3 - Formulario

Nombre:

Apellidos:

Aceptar

Bienvenid@ DAM_M07 UF1

¡TÚ PUEDES!



Solución propuesta de ejercicio

Al ejercicio anterior propuesto añadirle funcionalidad al botón aceptar; para que después de rellenar los datos, al pulsar Aceptar, de la bienvenida al usuario donde pone Usuario

```
private void btnAceptarActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    labelSaludo.setText("Bienvenid@ "+txtFieldNombre.getText()+" "  
        + txtFieldApellidos.getText());  
}
```

Pero que para manejar eventos, hemos dicho que era necesario añadir un Listener para que "escuche" los eventos. Entonces?



Solución propuesta de ejercicio

Al trabajar con NetBeans, te autogenera el código necesario para ese componente, dentro del método initComponents()

```
64 btnAceptar.addActionListener(new java.awt.event.ActionListener() {  
65     public void actionPerformed(java.awt.event.ActionEvent evt) {  
66         btnAceptarActionPerformed(evt);  
67     }  
68 });  
69
```

Clases, propiedades, métodos

¿Qué es una clase?

- Es una plantilla que se utiliza para crear objetos de datos. Concepto de POJO.

¿Qué son campos de datos?

- Los datos pueden estar almacenados en variables o estructuras más complejas (como structs).
- Reflejan el estado de la clase

¿Qué son los métodos?

- Implementan la funcionalidad asociada al objeto

¿Qué son las propiedades?

- Son un tipo especial de métodos para poder acceder a los atributos de la clase, que suelen ser privados.

CÓDIGO:

```
public class Coche {
    String marca;
    String modelo;
    Int potencia;

    // Constructor, se llamará cuando se cree la
    // clase
    public Coche () {
        marca = 'Ford';
        modelo = 'Focus';
        potencia = 150;
    }

    // Métodos para insertar valores, conocidos
    // como setters
    public void setMarca(String marca) {
        this.marca = marca;
    }
    public void setPotencia(int potencia) {
        this.potencia = potencia;
    }
    ...
    // Métodos para obtener valores, conocidos como
    // getters
    public String getMarca() {
        return marca;
    }
    public int getPotencia() {
        return potencia;
    }
    ...
}
```


¿DUDAS?



MÓDULO 07

Desarrollo de interfaces

CFGS Técnico Superior en Desarrollo de Aplicaciones Multiplataforma