



Programación de sistemas

Interfaces Gráficas

I. Conceptos básicos

II. Eventos

Julio Villena Román

<jvillena@it.uc3m.es>

MATERIALES BASADOS EN EL TRABAJO DE DIFERENTES AUTORES:
José Jesús García Rueda, Carlos Alario Hoyos

Contenidos

- ❖ ¿Qué son las interfaces gráficas?
- ❖ Elementos de la interfaz gráfica
- ❖ Organización de elementos: Layouts

¿Qué son las interfaces gráficas?

- Parte del programa que interactúa con el usuario, utilizando objetos gráficos e imágenes.
 - Proporcionan un entorno (gráfico) sencillo e intuitivo
 - Evolución de la línea de comandos
 - GUI (*Graphical User Interface*)

¿Cómo construimos interfaces gráficas en Java?

- Utilizando clases incluidas en el paquete

```
package javax.swing;
```

API: <http://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>

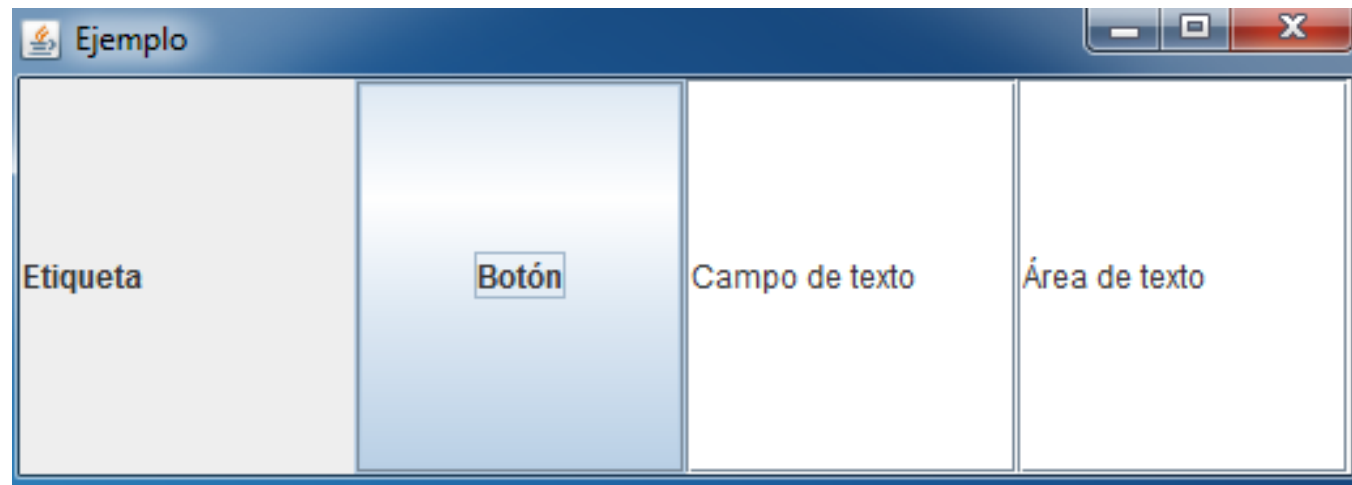
- Necesitamos importar esas clases en nuestro programa

```
import javax.swing.*;
```



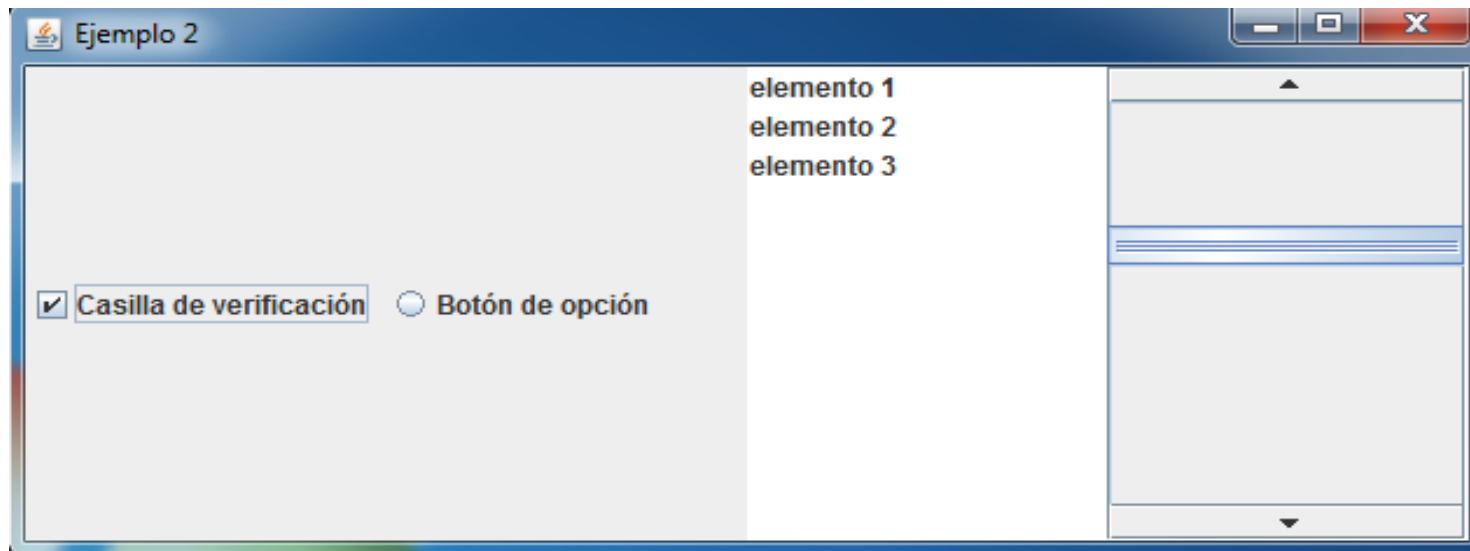
Ejemplos de clases en javax.swing

- **JLabel** Etiquetas
- **JButton** Botones
- **JTextField** Campos de texto
- **TextArea** Áreas de texto

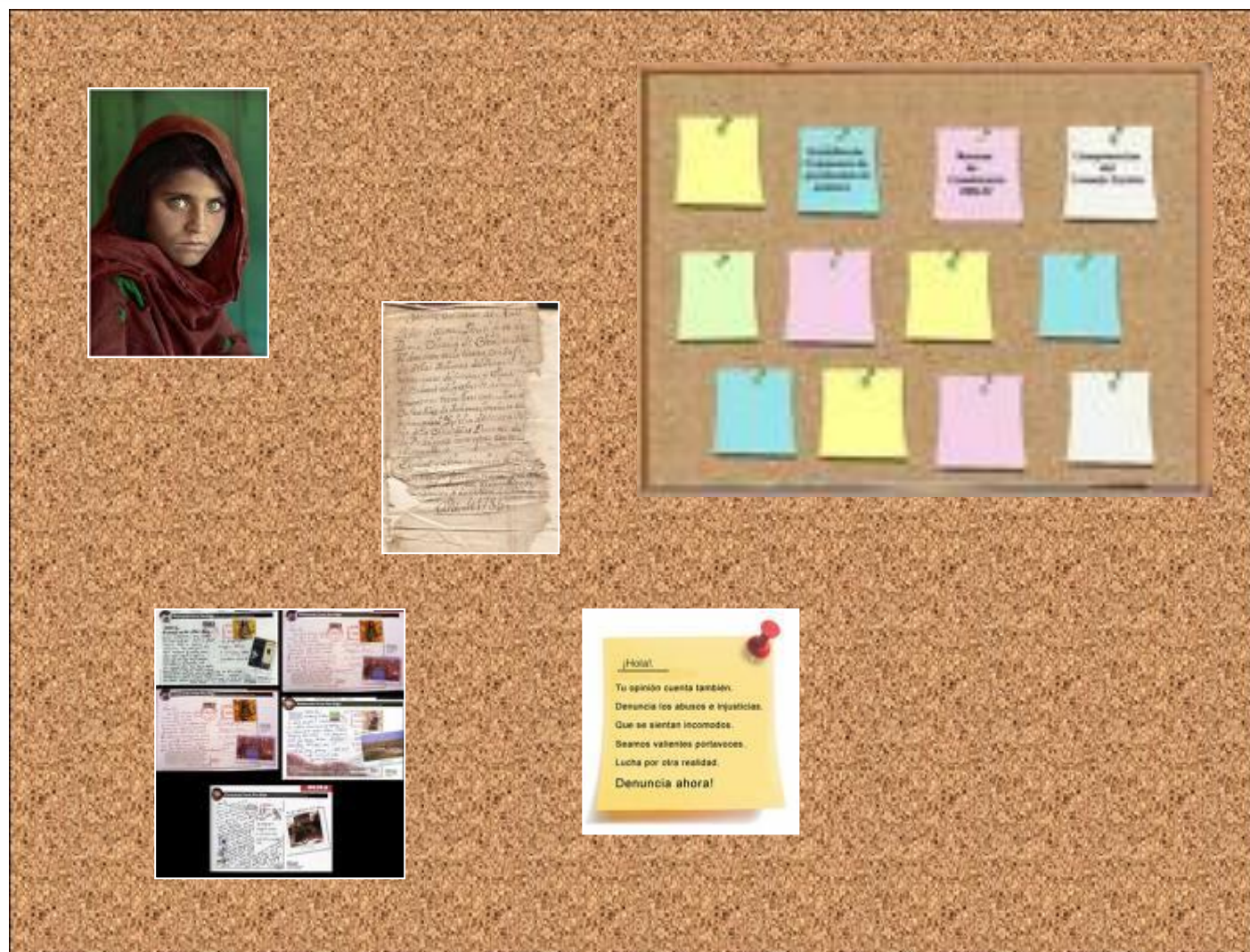


Ejemplos de clases en javax.swing

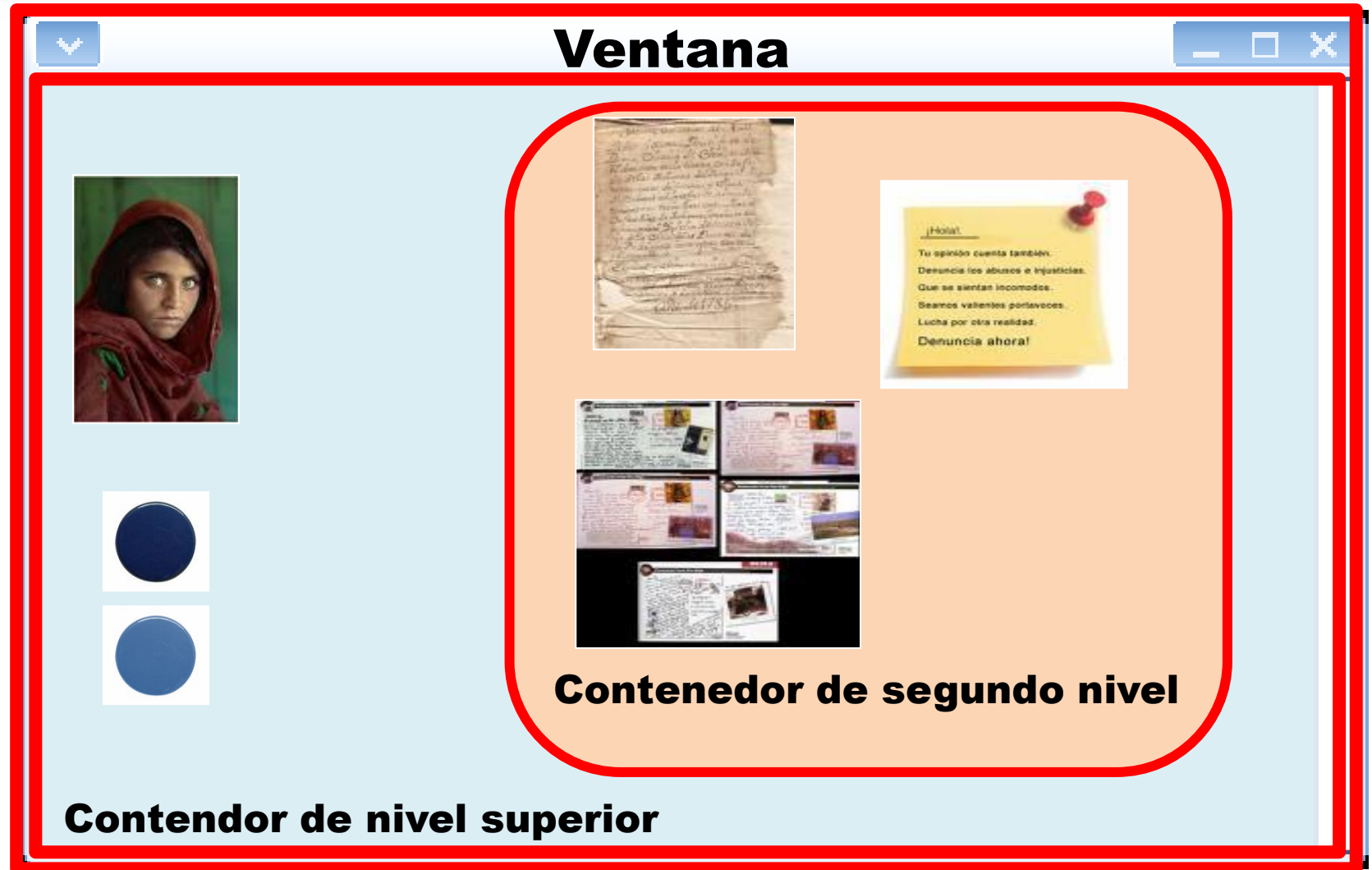
- **JCheckBox** Casillas de verificación
- **JRadioButton** Botones de opción
- **JList** Listas
- **JScrollBar** Barras de desplazamiento
- ...



La metáfora de la pared



La metáfora de la pared



Paso 1: Crear la ventana

```
import javax.swing.JFrame;

public class Ejemplo {
    public Ejemplo() {
        JFrame frame = new JFrame("Ejemplo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400,100);
        frame.setVisible(true);
    }
    public static void main(String args[]) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                Ejemplo gui = new Ejemplo();
            }
        });
    }
}
```

Necesitamos importar la clase **JFrame** (ventana genérica)

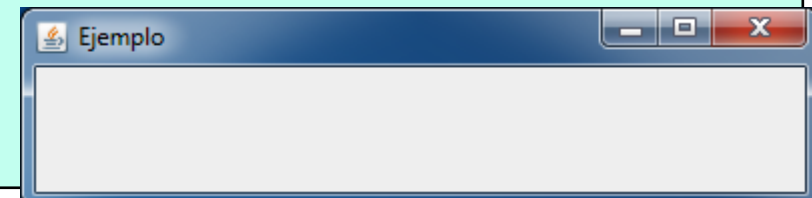
Creamos una nueva ventana

Hay que hacer visible la ventana explícitamente

Siempre igual

Creación de una instancia de nuestra interfaz gráfica

El main() es igual en las siguientes transparencias.



Paso 2: El contenedor de nivel superior

- Todas las ventanas tienen un contenedor de alto nivel sobre el que se colocan el resto de componentes
 - Incluidos otros contenedores

```
import javax.swing.JFrame;  
import javax.swing.JPanel;
```

Necesitamos importar la clase **JPanel** (contenedor genérico)

```
public class Ejemplo {  
    public Ejemplo() {  
        JFrame frame = new JFrame("Ejemplo");  
JPanel contentPane = (JPanel) frame.getContentPane();  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        frame.setSize(400,100);  
        frame.setVisible(true);  
    }  
}
```

Obtenemos la referencia al objeto que representa al contenedor de alto nivel

Alternativamente: **Container contentPane = frame.getContentPane();**

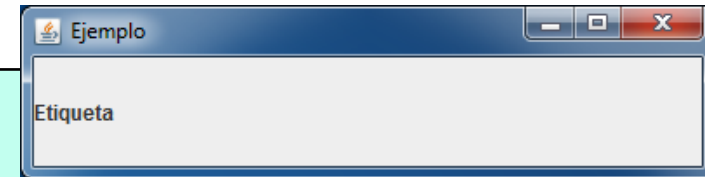
Paso 3: Añadir elementos

```
import javax.swing.JFrame;  
import javax.swing.JPanel;  
import javax.swing.JLabel;
```

Importamos la clase **JLabel**
(para incluir etiquetas)

```
public class Ejemplo {  
    public Ejemplo() {  
        JFrame frame = new JFrame("Ejemplo");  
        JPanel contentPane = (JPanel) frame.getContentPane();  
JLabel label = new JLabel("Etiqueta");  
contentPane.add(label);  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        frame.setSize(400,100);  
        frame.setVisible(true);  
    }  
}
```

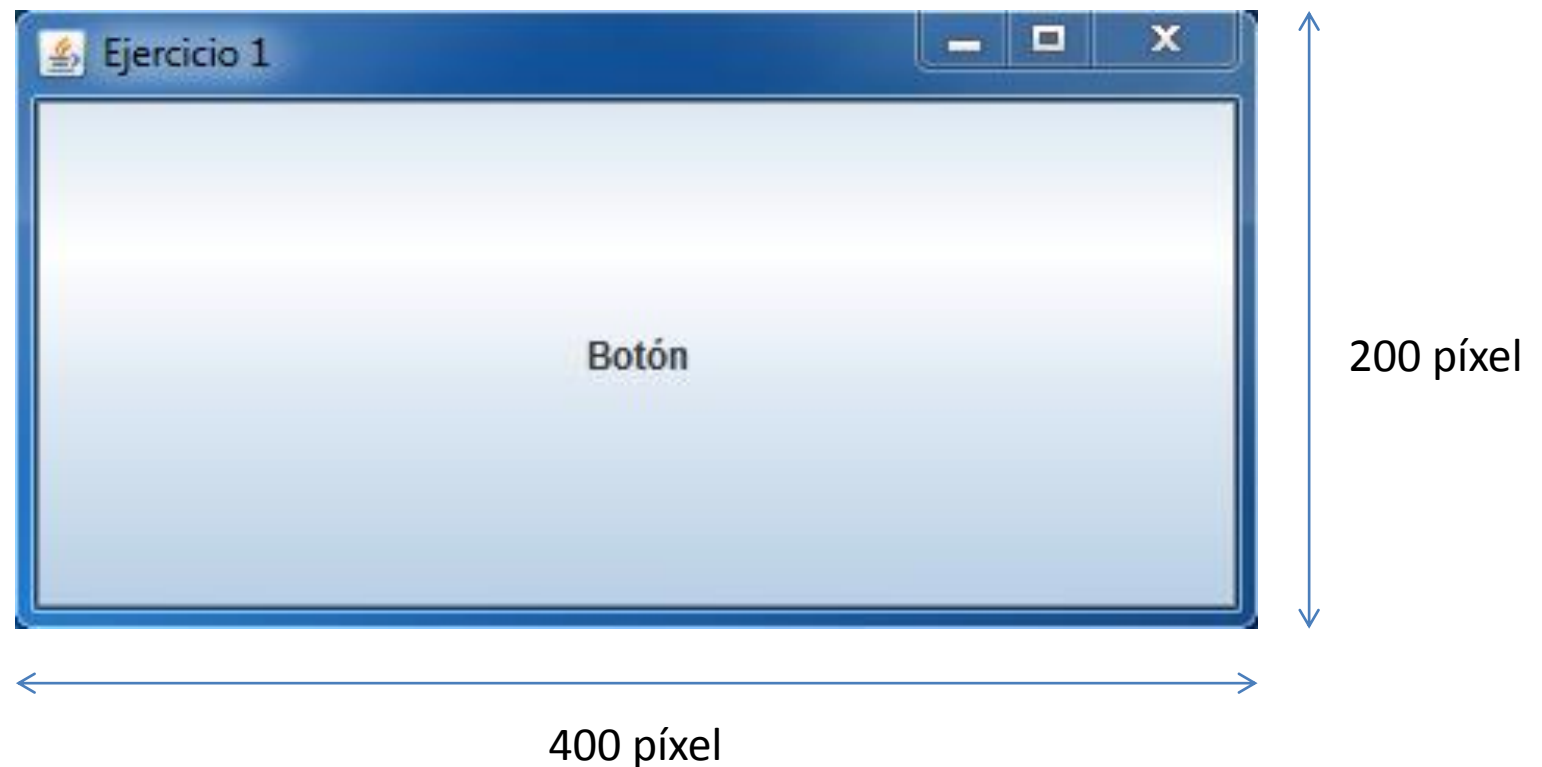
Creamos una etiqueta y la añadimos al panel



Ejercicio 1



- Implementa el código que permite generar la siguiente interfaz gráfica. No olvides importar las clases necesarias y hacer visible la ventana. El programa debe finalizar al cerrar la ventana.



Paso 4: Contenedores de niveles inferiores

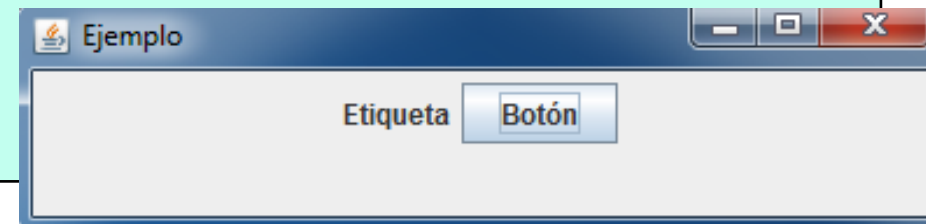
- Un panel puede contener a otros paneles

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JButton;

public class Ejemplo {
    public Ejemplo() {
        JFrame frame = new JFrame("Ejemplo");
        JPanel contentPane = (JPanel) frame.getContentPane();
        JPanel panel = new JPanel();
        JLabel label = new JLabel("Etiqueta");
        JButton button = new JButton("Botón");
        panel.add(label);
        panel.add(button);
        contentPane.add(panel);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400,100);
        frame.setVisible(true);
    }
}
```

Creamos un nuevo panel
y le añadimos los
elementos

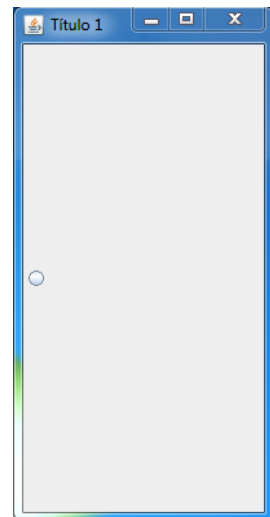
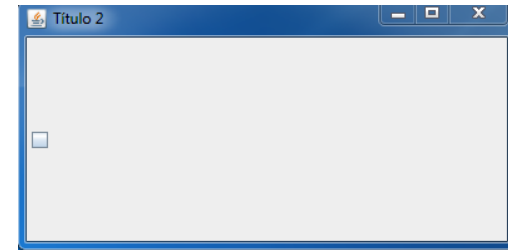
Añadimos el panel al contenedor de nivel superior



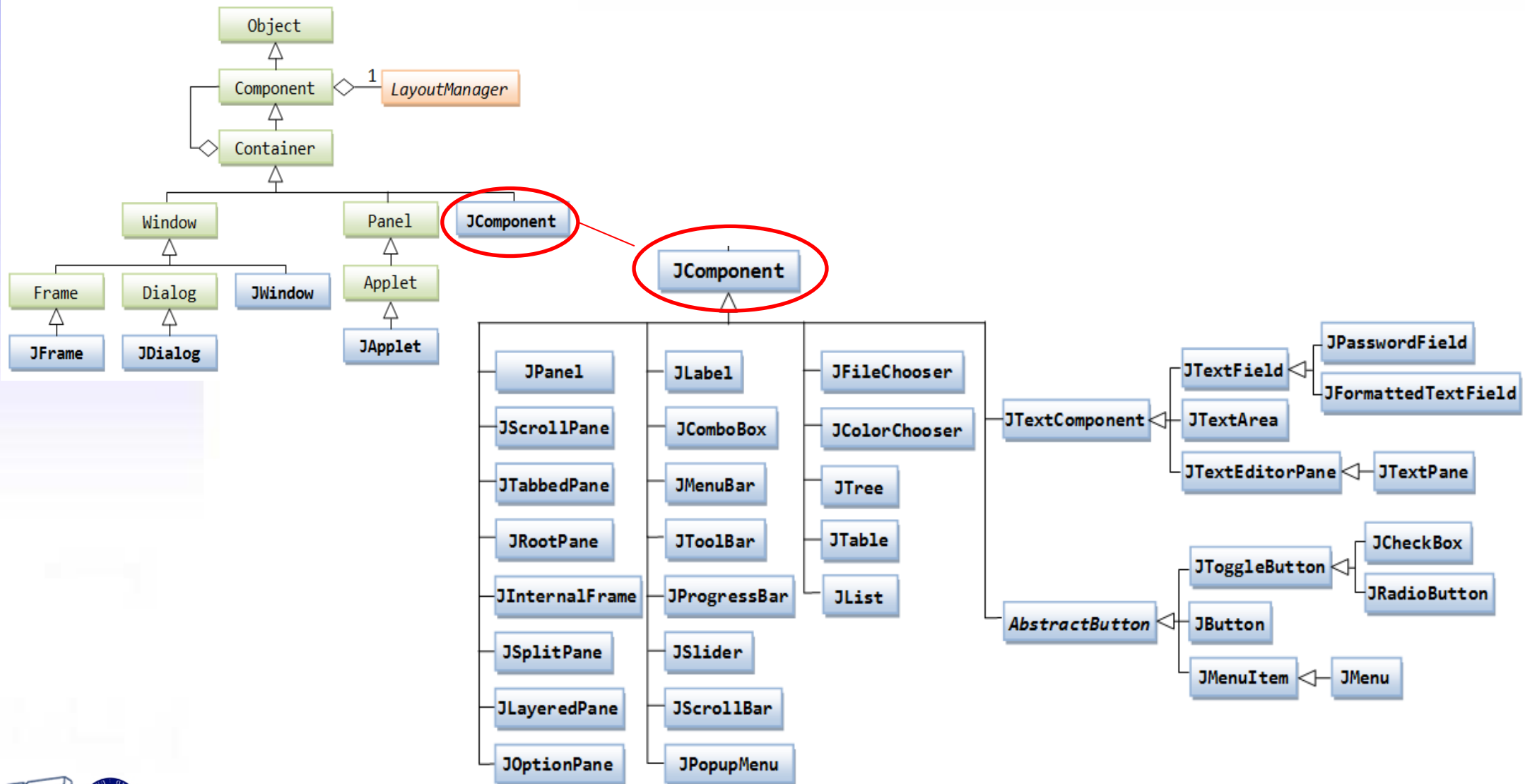
Creando nuestras propias ventanas

- También podemos crear nuestras propias ventanas como subclases de **Jframe**
 - Especialmente cuando se necesitan varias instancias del mismo tipo de ventana (p.ej. ventanas de error)

```
public class MiVentana extends JFrame{
    public MiVentana(String title, int x, int y){
        this.setSize(x,y);
        this.setTitle(title);
        this.setVisible(true);
    }
    public static void main (String[] args){
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                MiVentana ventana1 = new MiVentana("Título 1", 200, 400);
                MiVentana ventana2 = new MiVentana("Título 2", 400, 200);
                JPanel contentPane1 = (JPanel) ventana1.getContentPane();
                contentPane1.add(new JRadioButton(""));
                JPanel contentPane2 = (JPanel) ventana2.getContentPane();
                contentPane2.add(new JCheckBox(""));
            }
        });
    }
}
```



Jerarquía de clases de los elementos de las interfaces gráficas en Java



¿Cómo ordenar los elementos de nuestra interfaz gráfica?

- Opción 1: Uso de **layouts**
 - Plantillas que facilitan la colocación de elementos en la interfaz
 - Se asocian a paneles (cada panel puede tener un layout diferente)
- Opción 2: Por coordenadas
 - Método `setBounds(int x, int y, int width, int height)`
 - Para usar coordenadas hay que anular el layout del panel
 - `setLayout(null)`



Layouts

- Conjunto de clases incluidas en el paquete

```
package java.awt;
```

API: <http://docs.oracle.com/javase/7/docs/api/java/awt/package-summary.html>

- Necesitamos importar esas clases en nuestro programa

```
import java.awt.*;
```



FlowLayout

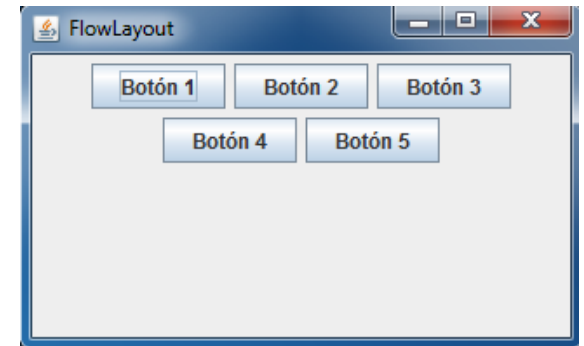
- Coloca los elementos en línea uno detrás de otro
 - Layout por defecto en los paneles

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Flow {

    public Flow(){
        JFrame frame = new JFrame("FlowLayout");
        JPanel contentPane = (JPanel) frame.getContentPane();
        JPanel panel = new JPanel();
        panel.add(new JButton("Botón 1"));
        panel.add(new JButton("Botón 2"));
        panel.add(new JButton("Botón 3"));
        panel.add(new JButton("Botón 4"));
        panel.add(new JButton("Botón 5"));
        contentPane.add(panel);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400,200);
        frame.setVisible(true);
    }

    public static void main(String args[]) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                Flow gui = new Flow();
            }
        });
    }
}
```



GridLayout

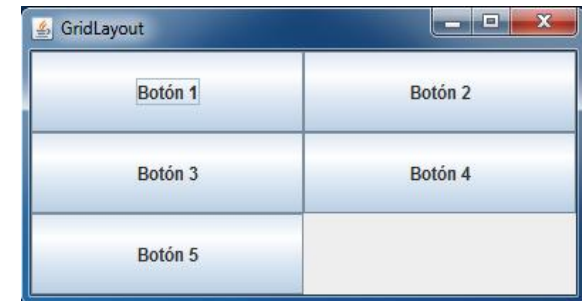
- Coloca los elementos en cuadrícula

Importamos la clase GridLayout

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

public class Grid {
    public Grid(){
        JFrame frame = new JFrame("GridLayout");
        JPanel contentPane = (JPanel) frame.getContentPane();
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(3,2));
        panel.add(new JButton("Botón 1"));
        panel.add(new JButton("Botón 2"));
        panel.add(new JButton("Botón 3"));
        panel.add(new JButton("Botón 4"));
        panel.add(new JButton("Botón 5"));
        contentPane.add(panel);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400,200);
        frame.setVisible(true);
    }

    public static void main(String args[]) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                Grid gui = new Grid();
            }
        });
    }
}
```



Indicamos el tipo de Layout

BorderLayout

- Coloca los elementos en cuadrícula

Importamos la clase BorderLayout

```
import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

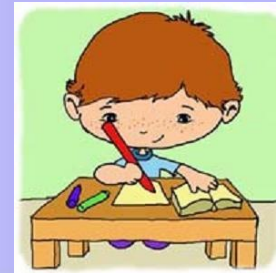
public class Border {
    public Border(){
        JFrame frame = new JFrame("BorderLayout");
        JPanel contentPane = (JPanel) frame.getContentPane();
        JPanel panel = new JPanel();
        panel.setLayout(new BorderLayout());
        panel.add(new JButton("Botón 1"), BorderLayout.NORTH);
        panel.add(new JButton("Botón 2"), BorderLayout.EAST);
        panel.add(new JButton("Botón 3"), BorderLayout.SOUTH);
        panel.add(new JButton("Botón 4"), BorderLayout.WEST);
        panel.add(new JButton("Botón 5"), BorderLayout.CENTER);
        contentPane.add(panel);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400,200);
        frame.setVisible(true);
    }

    public static void main(String args[]) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                Border gui = new Border();
            }
        });
    }
}
```

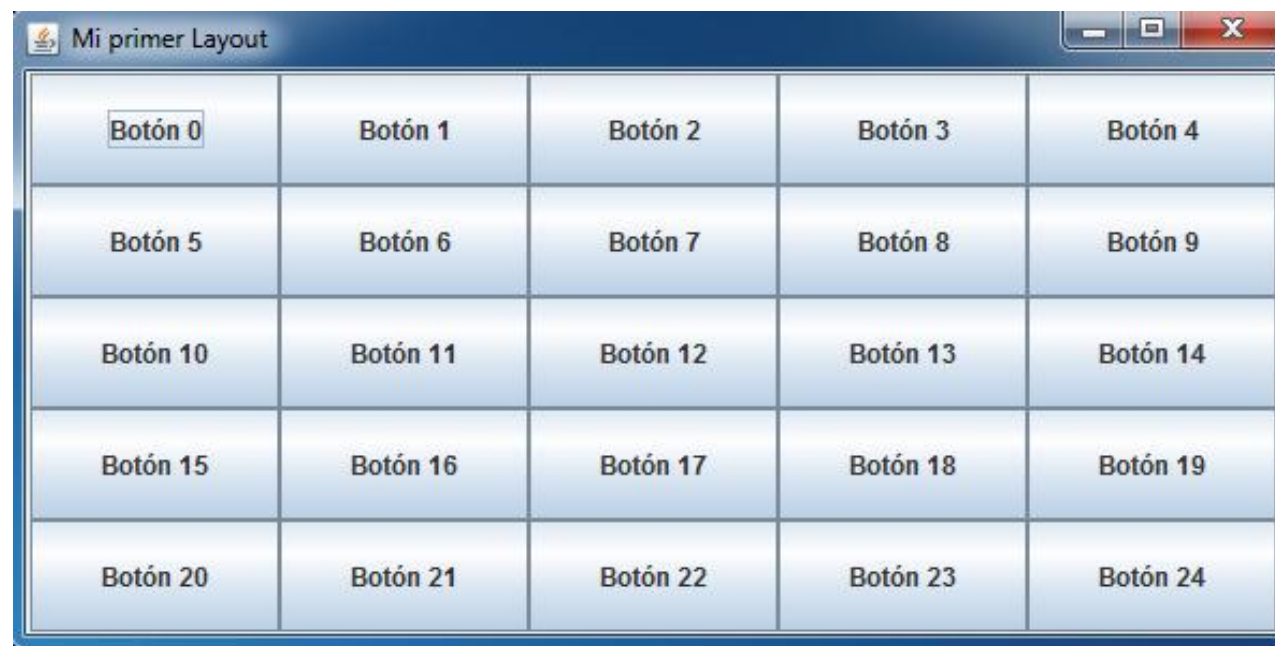


Indicamos el tipo de Layout y la posición de los elementos

Ejercicio 2



- Implementa el código que permite generar la siguiente interfaz gráfica. No olvides importar las clases necesarias y hacer visible la ventana. El programa debe finalizar al cerrar la ventana.

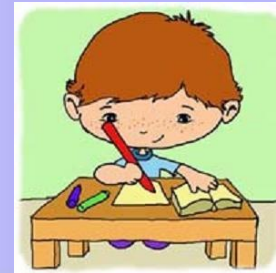


300 píxel

600 píxel



Ejercicio 3



- Implementa el código que permite generar la siguiente interfaz gráfica. No olvides importar las clases necesarias y hacer visible la ventana. El programa debe finalizar al cerrar la ventana.



Ejercicio 4



- Implementa el código que permite generar la siguiente interfaz gráfica. No olvides importar las clases necesarias y hacer visible la ventana. El programa debe finalizar al cerrar la ventana. Utiliza **JPasswordField** para el campo de texto del password

Formulario

Nombre

Apellidos

Password

Género H M

¿Estás de acuerdo?

Enviar



Programación de sistemas

Interfaces Gráficas

I. Conceptos básicos

II. Eventos

Julio Villena Román

<jvillena@it.uc3m.es>

MATERIALES BASADOS EN EL TRABAJO DE DIFERENTES AUTORES:
José Jesús García Rueda, Carlos Alario Hoyos

Contenidos

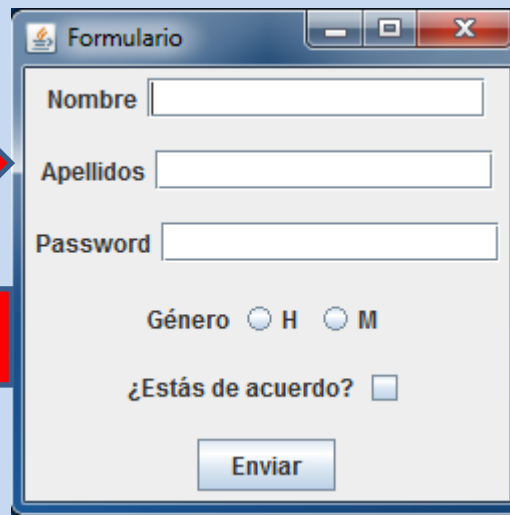
- ❖ ¿Qué son los eventos en una interfaz gráfica?
- ❖ Ejemplos de escuchadores
- ❖ Adaptadores



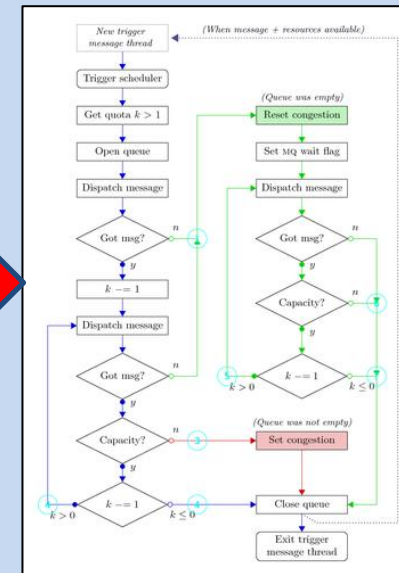
¿Qué son los eventos en una interfaz gráfica?



Usuario



Interfaz Gráfica

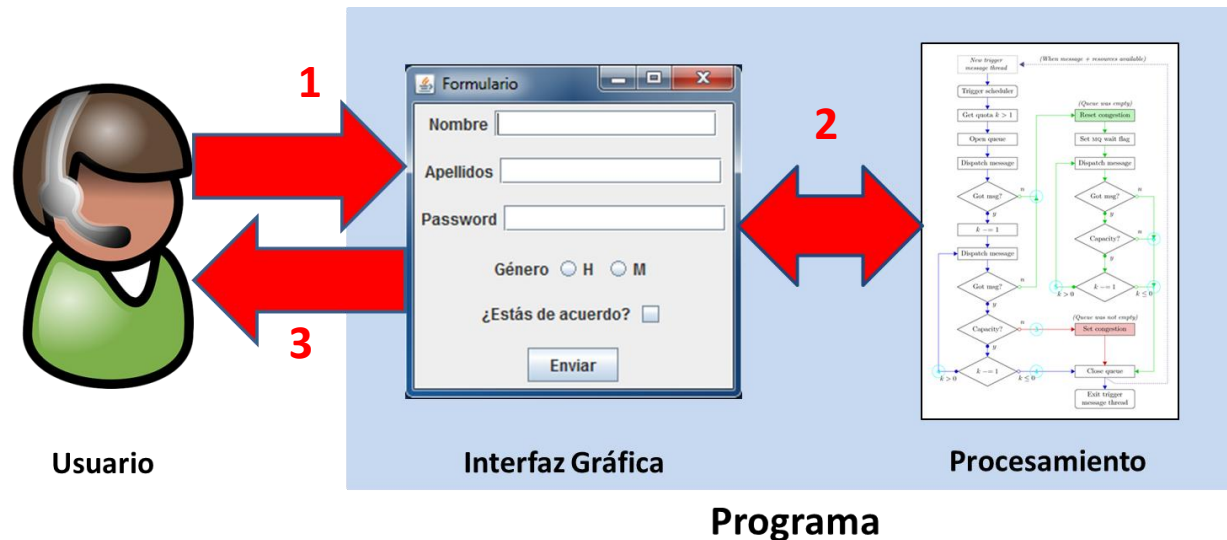


Procesamiento

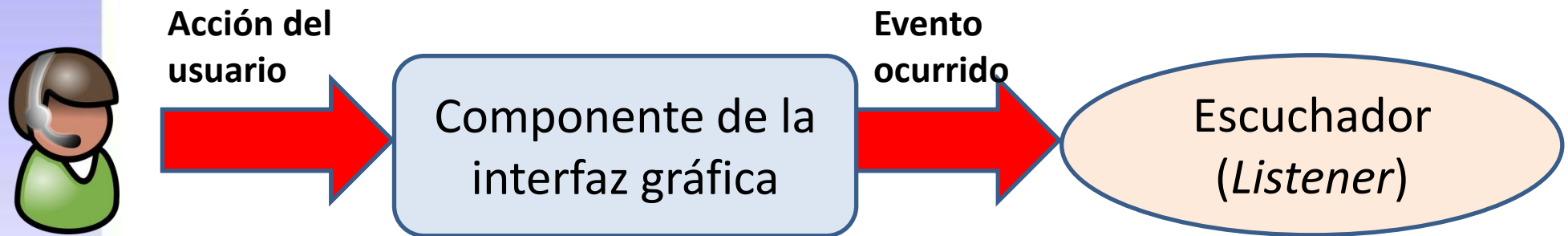
Programa

¿Qué son los eventos en una interfaz gráfica?

1. El usuario realiza acciones (**genera eventos**) sobre componentes de la interfaz gráfica (p.ej. pulsar botón).
 - Los componentes de la interfaz gráfica “**deben estar atentos**” para recibir los eventos generados por el usuario
2. Una vez recibido un evento hay que **procesarlo**
 - Comunicación con la parte del programa encargada del procesamiento
3. (Cuando proceda) modificar la interfaz gráfica para presentar el resultado del procesamiento (ej: mensaje)



Paso 1: Los componentes de la interfaz gráfica deben “estar atentos”



- Los escuchadores son **interfaces Java** incluidos en el paquete

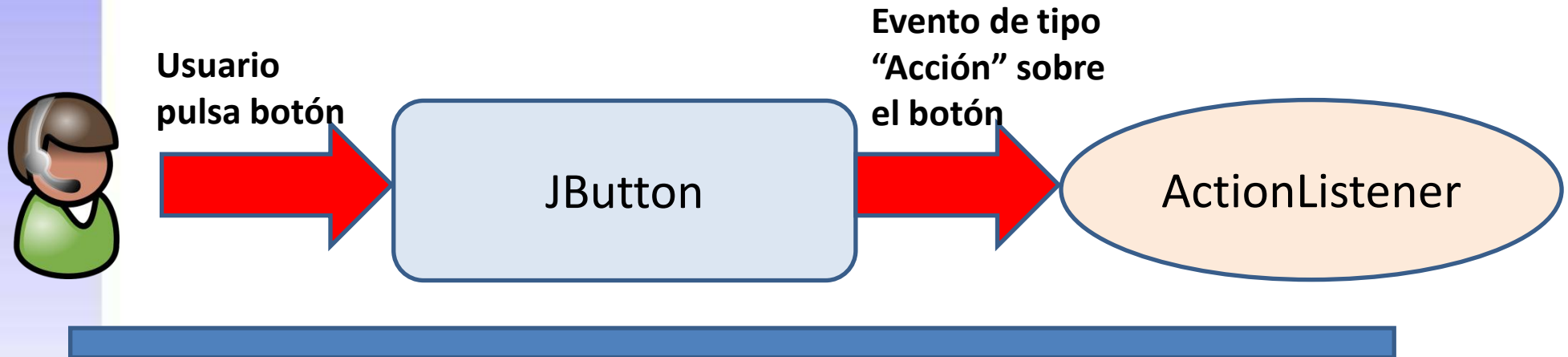
```
package java.awt.event;
```

API: <http://docs.oracle.com/javase/7/docs/api/java/awt/event/package-summary.html>

- Necesitamos importar esas interfaces en nuestro programa

```
import java.awt.event.*;
```

Paso 1: Los componentes de la interfaz gráfica deben “estar atentos”

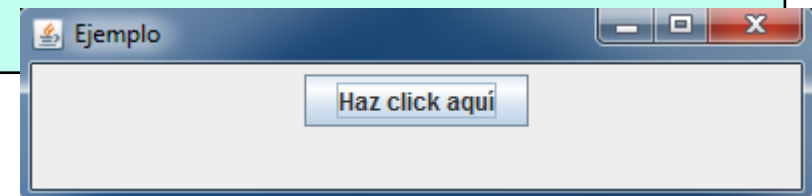


- Interfaz **ActionListener**
 - Incluye el método **actionPerformed(ActionEvent e)**, el cual debe ser implementado

Código de partida (sesión anterior)

```
import javax.swing.JFrame;  
import javax.swing.JPanel;  
import javax.swing.JButton;  
  
public class Ejemplo {  
    public Ejemplo() {  
        JFrame frame = new JFrame("Ejemplo");  
        JPanel contentPane = (JPanel) frame.getContentPane();  
        JPanel panel = new JPanel();  
        JButton button = new JButton("Botón");  
        panel.add(button);  
        contentPane.add(panel);  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        frame.setSize(400,100);  
        frame.setVisible(true);  
    }  
}
```

Alternativamente: **Container contentPane = frame.getContentPane();**

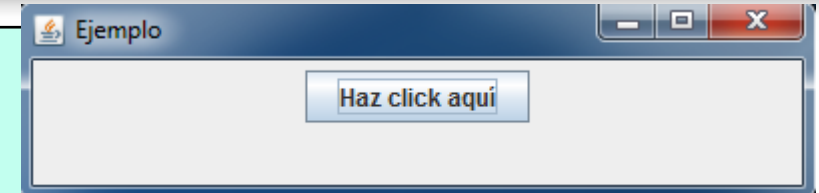


Paso 1: Los componentes de la interfaz gráfica deben “estar atentos”

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
```

```
public class Ejemplo {
    public Ejemplo() {
        JFrame frame = new JFrame("Ejemplo");
        JPanel contentPane = (JPanel) frame.getContentPane();
        JPanel panel = new JPanel();
        JButton button = new JButton("Haz click aquí");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
            }
        });
        panel.add(button);
        contentPane.add(panel);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400,100);
        frame.setVisible(true);
    }
}
```

Importar interfaz **ActionListener** y clase **ActionEvent**



Añadimos el escuchador **ActionListener** al botón. Eso nos obliga a tener que implementar el método **actionPerformed(ActionEvent e)** incluido en esa interfaz

Paso 2: Procesar el evento

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

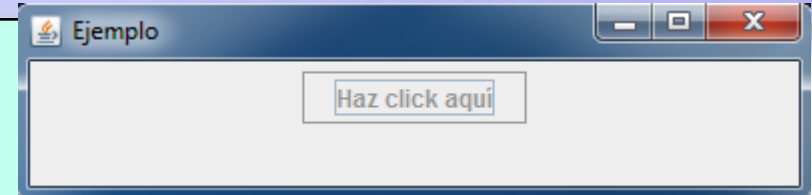
public class Ejemplo {
    public Ejemplo() {
        JFrame frame = new JFrame("Ejemplo");
        JPanel contentPane = (JPanel) frame.getContentPane();
        JPanel panel = new JPanel();
        JButton button = new JButton("Haz click aquí");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Botón pulsado");
            }
        });
        panel.add(button);
        contentPane.add(panel);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400,100);
        frame.setVisible(true);
    }
}
```

Imprimimos por pantalla que el botón ha sido pulsado

Paso 3: Modificar la interfaz gráfica

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class Ejemplo {
    public Ejemplo() {
        JFrame frame = new JFrame("Ejemplo");
        JPanel contentPane = (JPanel) frame.getContentPane();
        JPanel panel = new JPanel();
        JButton button = new JButton("Haz click aquí");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Botón pulsado");
                JButton clickedButton = (JButton) e.getSource();
                clickedButton.setEnabled(false);
            }
        });
        panel.add(button);
        contentPane.add(panel);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400,100);
        frame.setVisible(true);
    }
}
```

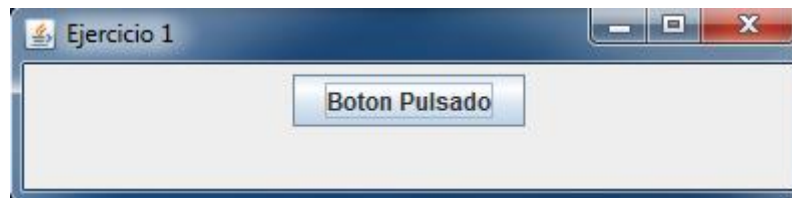
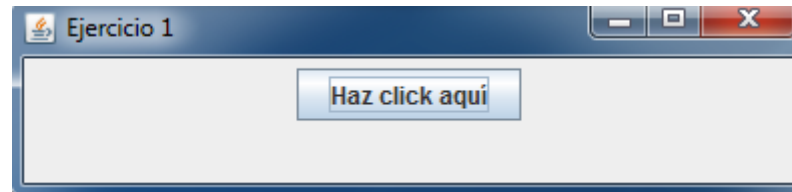


Recuperamos el componente concreto que produjo el evento (button) y lo deshabilitamos

Ejercicio 5



- Implementa el código que permite generar la siguiente interfaz gráfica. En la figura superior el usuario no ha realizado ninguna acción. En la figura inferior el usuario ha pulsado el botón. Además, tu interfaz gráfica debe almacenar en el atributo **clicked** si el botón ha sido pulsado ya o no.



- No olvides importar las clases necesarias y hacer visible la ventana.
- El programa debe finalizar al cerrar la ventana.

¿Y si tengo varios componentes que reaccionan de la misma forma a las acciones del usuario?

- Podemos crear nuestra propia clase que implementa la interfaz del escuchador (p.ej. **ActionListener**)
 - Varios componentes pueden implementan el mismo escuchador
- Separamos las clases que nos permiten “pintar” la interfaz gráfica de las clases que reciben (y procesan) eventos



¿Y si tengo varios componentes que reaccionan de la misma forma a las acciones del usuario?

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;

public class MiEscuchador implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        JButton clickedButton = (JButton) e.getSource();
        clickedButton.setEnabled(false);
    }
}
```

Hay que implementar la interfaz ActionListener

a) Clase que recibe y procesa eventos

```
public class Ejemplo {
    public Ejemplo() {
        ...
        JButton button = new JButton("Botón");
        button.addActionListener(new MiEscuchador());
        ...
    }
}
```

b) Clase que "pinta" la interfaz gráfica

Añadimos una instancia de MiEscuchador a nuestro botón

Ejercicio 6



- Partiendo del Ejercicio 2 de la sesión anterior en el que se implementaba un grid con 25 botones, modifica el código para que cada vez que el usuario pulse uno de ellos, se deshabilite y el texto que contiene muestre “Clicked” (ver figura)



- No olvides importar las clases necesarias y hacer visible la ventana.
- El programa debe finalizar al cerrar la ventana.

Más ejemplos de escuchadores: WindowListener

WindowListener

- `void windowClosing (WindowEvent evt)`
- `void windowOpened (WindowEvent evt)`
- `void windowClosed (WindowEvent evt)`
- `void windowIconified (WindowEvent evt)`
- `void windowDeiconified (WindowEvent evt)`
- `void windowActivated (WindowEvent evt)`
- `void windowDeactivated (WindowEvent evt)`



Más ejemplos de escuchadores: WindowListener

```
import java.awt.event.WindowEvent;  
import java.awt.event.WindowListener;  
import javax.swing.JFrame;  
import javax.swing.JOptionPane;
```

Importar la interfaz **WindowListener**
y la clase **WindowEvent**

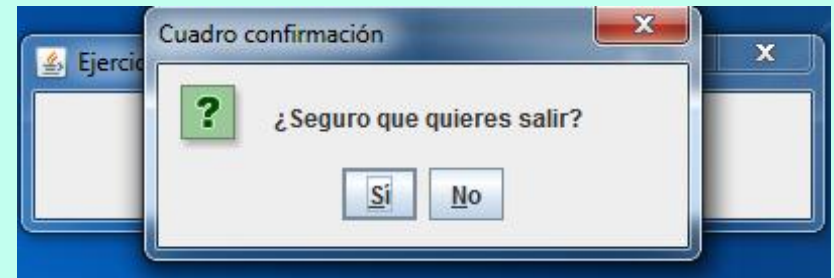
```
public class Ejemplo {  
    public Ejemplo() {  
        JFrame frame = new JFrame("Ejemplo");  
        frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);  
        frame.addWindowListener(new WindowListener() {  
            public void windowClosing(WindowEvent e) {...}  
            public void windowOpened(WindowEvent e) {...}  
            public void windowClosed(WindowEvent e) {...}  
            public void windowIconified(WindowEvent e) {...}  
            public void windowDeiconified(WindowEvent e) {...}  
            public void windowActivated(WindowEvent e) {...}  
            public void windowDeactivated(WindowEvent e) {...}  
        });  
        frame.setSize(400,100);  
        frame.setVisible(true);  
    }  
}
```

Añadimos el escuchador **WindowListener** a la ventana. Eso nos obliga a tener que implementar siete métodos (aunque algunos métodos pueden no hacer nada)

Más ejemplos de escuchadores: WindowListener

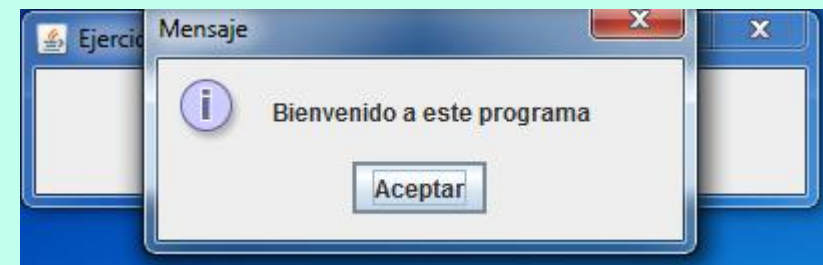
Método que al hacer click sobre la "x" pide al usuario que confirme si quiere salir del programa

```
public void windowClosing(WindowEvent evt) {  
    JFrame frame = (JFrame) e.getSource();  
    int confirm = JOptionPane.showOptionDialog(frame, "¿Seguro que quieres salir?", "Cuadro confirmación", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null, null, null);  
    if (confirm == JOptionPane.YES_OPTION) {  
        System.exit(0);  
    }  
}
```



Método que al visualizar la ventana por primera vez da la bienvenida al usuario

```
public void windowOpened(WindowEvent evt) {  
    JFrame frame = (JFrame) e.getSource();  
    JOptionPane.showMessageDialog(frame, "Bienvenido a este programa");  
}
```



Ejercicio 6



- Implementa una ventana que el usuario no pueda cerrar. Para ello debes crear una ventana básica y añadirle un escuchador **WindowListener**, de tal forma que al ir a cerrar esta ventana (**windowClosing**) se cree una nueva ventana con el mismo comportamiento que la primera.
- No olvides importar las clases necesarias y hacer visible la ventana.
- El programa debe finalizar al cerrar la ventana.

Más ejemplos de escuchadores

FocusListener

- Recoge eventos del tipo recibir o perder el foco mediante acciones realizadas con el teclado

KeyListener

- Recoge eventos del tipo presionar/liberar una tecla del teclado y escribir un carácter

MouseListener

- Eventos del presionar/liberar el ratón sobre un componente

¡Un mismo componente gráfico puede implementar varios escuchadores!



¿Qué parte del escuchador “se despierta”?

- Java invoca automáticamente al método oportuno del escuchador dependiendo del evento recibido.
- El cuerpo de dicho método lo programamos nosotros, pudiendo invocar desde él a otros métodos.
- Cuando el método termina su ejecución el programa vuelve a quedarse a la espera de nuevos eventos.
- Los métodos reciben un objeto de tipo “evento” como argumento (p.ej. **ActionEvent**, **WindowEvent**)
 - Nosotros podemos preguntar a ese objeto para saber qué ocurrió (p.ej. el elemento concreto sobre el que se produjo el evento)



Programación orientada a eventos

- Todo lo visto no es más que un caso particular de una técnica de programación muy importante y extendida: la **Programación Orientada a Eventos**
- En un programa todo está bien planeado: se sabe a priori cuándo va a ocurrir
 - ¿Cómo tener en cuenta entonces aquellos sucesos del mundo exterior que no sabemos con certeza cuándo ocurrirán?
- Los programas tienen mecanismos que les permiten reaccionar (“despertar”) cuando ocurren determinados eventos en el mundo exterior



Adaptadores

- Algunas interfaces de escuchadores tienen muchos métodos y hay que implementar todos (p.ej. **WindowListener**)
- Los adaptadores son clases que implementan todos los métodos de un escuchador determinado.
- Como son clases, basta con heredarlas y reescribir los métodos que necesitemos.
- Por cada interfaz ***Listener**, hay una clase ***Adapter**
 - **WindowListener** => **WindowAdapter**
 - **KeyListener** => **KeyAdapter**
 - **MouseListener** => **MouseAdapter**



Adaptadores: WindowAdapter

```
import java.awt.event.WindowAdapter;  
import java.awt.event.WindowEvent;  
import javax.swing.JFrame;  
import javax.swing.JOptionPane;
```

Importar las clases **WindowAdapter** y **WindowEvent**

```
public class Ejemplo {  
    public Ejemplo() {  
        JFrame frame = new JFrame("Ejemplo");  
        frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);  
        frame.addWindowListener(new WindowAdapter() {  
            public void windowClosing(WindowEvent evt) {  
                JFrame frame = (JFrame) evt.getSource();  
                int confirm = JOptionPane.showOptionDialog(frame, "¿Seguro que  
quieres salir?", "Cuadro confirmación",  
                JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE,  
                null, null, null);  
                if (confirm == JOptionPane.YES_OPTION) {  
                    System.exit(0);  
                }  
            }  
        });  
        frame.setSize(400,100);  
        frame.setVisible(true);  
    }  
}
```

Uso de **WindowAdapter** en
lugar de **WindowListener**



Sobreimplementamos **windowClosing(WindowEvent e)**

Ejercicio 7



- Rehaz el Ejercicio 3 (la ventana del usuario que no se puede cerrar) utilizando **WindowAdapter** en lugar de **WindowListener**.



Ejercicio 8



- Implementa la siguiente interfaz gráfica, la cual al pulsar el botón “Enviar”, imprime por pantalla el texto introducido en los campos de texto de “nombre”, “apellidos” y “password”.

Formulario

Nombre

Apellidos

Password

Enviar