

# Técnicas de Programación Avanzada (Java)

Curso 2021-2021

```
S_4 (Programación Interfaces) {  
    4.1 Interfaces Gráficas de Usuario.  
    4.2 Gestión de eventos.  
}
```

## **Tema 1: Objetos y memoria.**

1.1 Características básicas del lenguaje. Primer programa. Compilación y Ejecución. IDE.

1.2 Sentencias de control. Secuencia, selección e iteración.

1.3 Abstracción. Clases, objetos, métodos y atributos.

1.4 Sobrecarga de métodos y encapsulamiento.

## **Tema 2. Otros conceptos fundamentales de la Programación Orientada a Objetos**

2.1 Herencia. Interfaces y clases abstractas. Agregación.

2.2 Polimorfismo.

2.3 Gestión de Excepciones.

2.4 Genericidad y plantillas.

2.5 Utilidades. Entrada y Salida.

2.6 Anotaciones.

## **Tema 3. Patrones de Diseño.**

3.1 Concepto de Patrones de Diseño.

3.2 Patrones de creación.

3.3 Patrones estructurales.

3.4 Patrones de comportamiento.

## **Tema 4. Programación de Interfaces.**

4.1 Interfaces Gráficas de Usuario.

4.2 Gestión de eventos.

## **Tema 5. Temas Avanzados.**

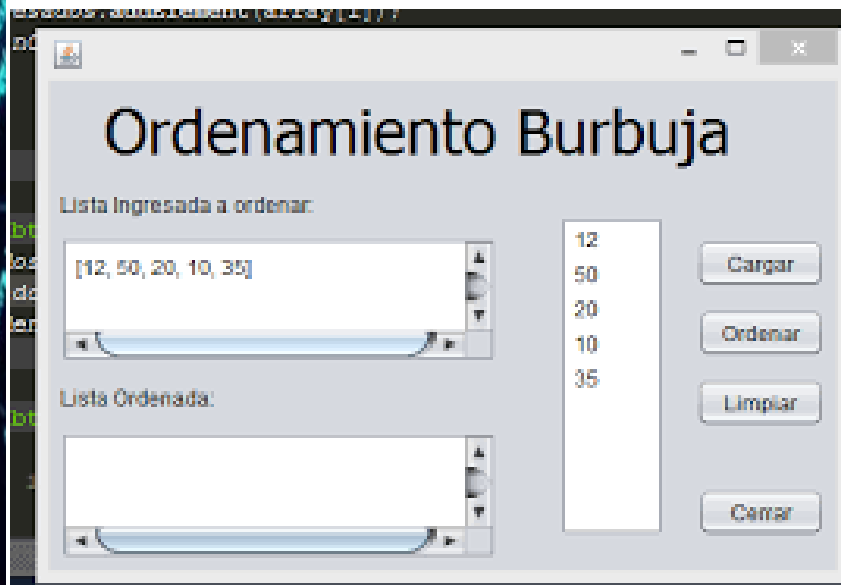
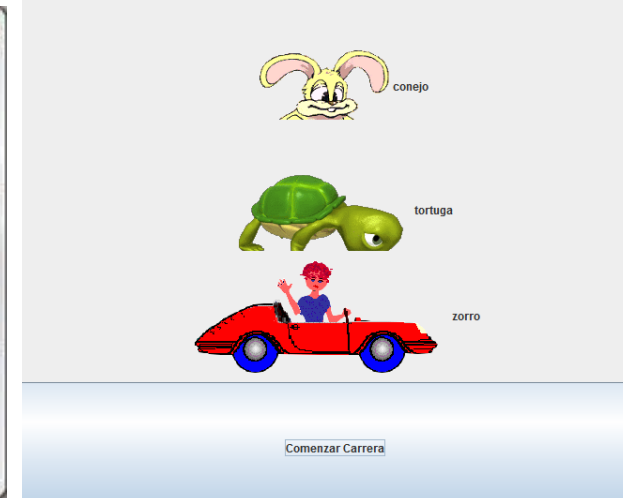
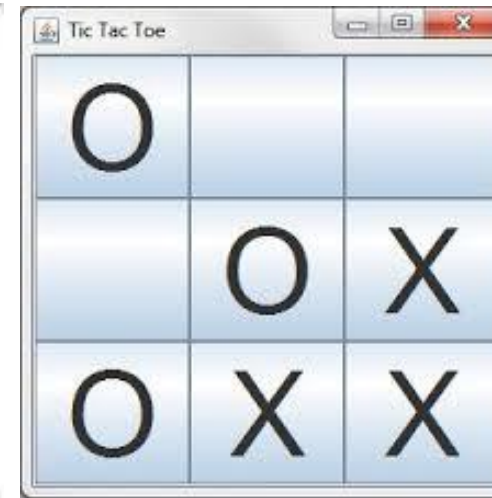
5.1 Concurrencia.

5.2 Inversión de Control. Definición y ejemplos. Inyección de dependencias.

5.3 Expresiones avanzadas del lenguaje.

# 4.1 Interfaces gráficas de Usuario

## Ejemplos





# 4.1 Interfaces gráficas de Usuario

## Java Swing

**Java Swing** es una herramienta de interfaz gráfica de usuario (GUI) ligera que incluye un amplio conjunto de widgets. Incluye paquete que le permite crear componentes de GUI para sus aplicaciones Java, y es independiente de la plataforma.

<https://docs.oracle.com/javase/tutorial/uiswing/components/index.html>

La biblioteca Swing está construida sobre el conjunto de herramientas de widgets abstractos de Java (**AWT**), un kit de herramientas GUI más antiguo que depende de la plataforma. Puede utilizar los componentes de la GUI de Java como el botón, el cuadro de texto, etc. de la biblioteca y no tiene que crear los componentes desde cero.

Podemos crear las GUI introduciendo el código a mano, o a través de IDEs gráficos (compatibles o incluso integrados en Eclipse), como:

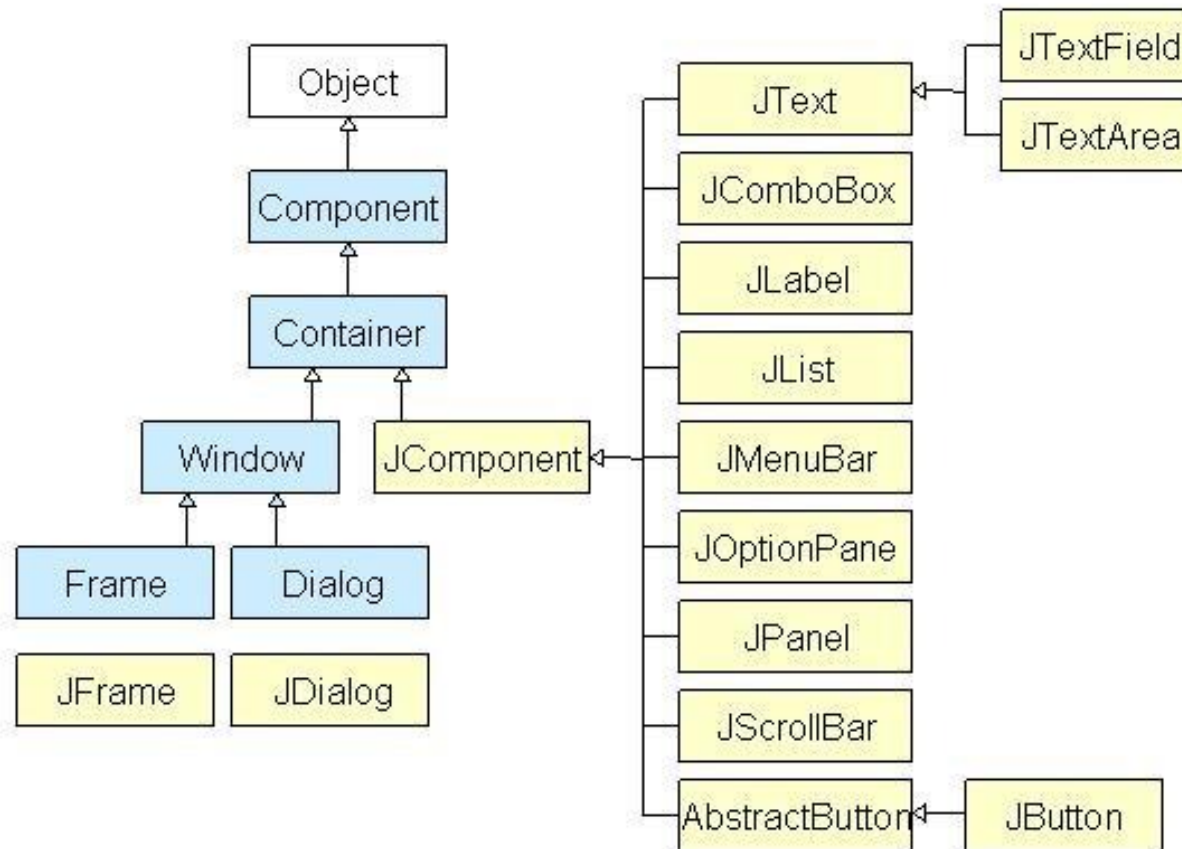
- [WindowBuilder](#)

- [NetBeans IDE](#) (Apache)

<https://guru99.es/java-swing-gui/>

# 4.1 Interfaces gráficas de Usuario

## Clases Swing



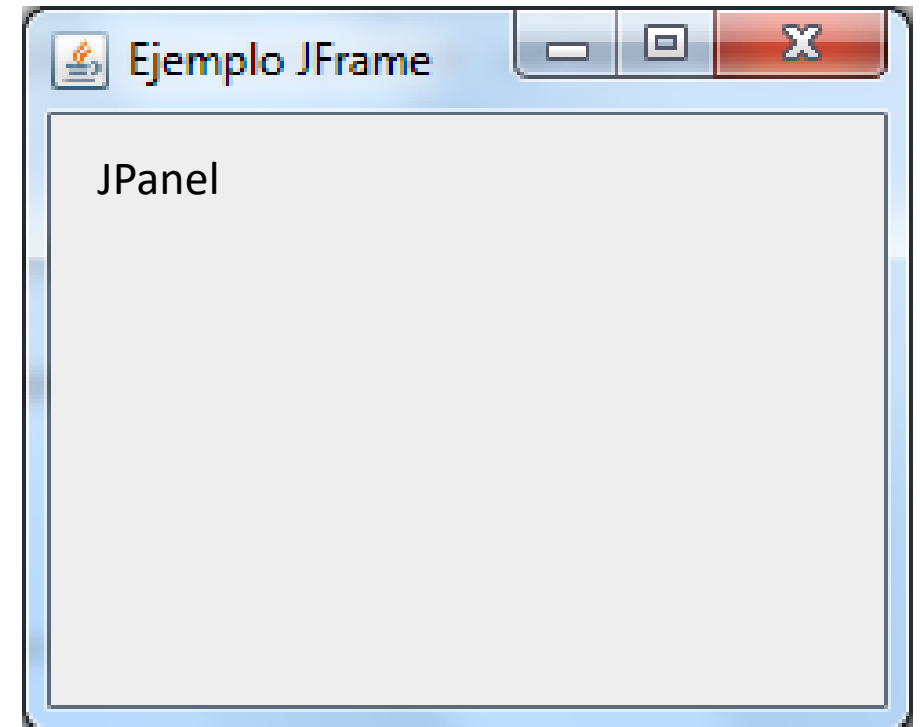
### Top level containers (ventanas):

- `JFrame` (desktop)
- `JApplet` (browser)

### Dentro:

JComponents

<https://docs.oracle.com/javase/tutorial/uiswing/components/rootpane.html>



# 4.1 Interfaces gráficas de Usuario

## Componentes básicos (Swing)

### \*Contenedores:

- [JFrame](#) es la ventana principal del programa, **debe haber sólo uno**. Tiene icono, y sale en la barra de tareas.
- [JDialog](#) pertenece a un *JFrame*, debe tener un padre. No tiene icono ni sale en la barra de tareas. *Se pueden abrir muchos diálogos*. Si se configura como “modal” no deja tocar el resto de ventanas hasta cerrar/terminar esta.
- [JInternalFrame](#) corre dentro de otro panel (+ habitual: [JDesktopPane](#)) *Se pueden abrir muchas ventanas internas*.

### \*Componentes:

[AbstractButton](#), [BasicInternalFrameTitlePane](#), [Box](#), [Box.Filler](#), [JColorChooser](#), [JComboBox](#), [JFileChooser](#), [JInternalFrame](#), [JInternalFrame.JDesktopIcon](#), [JLabel](#), [JLayer](#), [JLayeredPane](#), [JList](#), [JMenuBar](#), [JOptionPane](#), [JPanel](#), [JPopupMenu](#), [JProgressBar](#), [JRootPane](#), [JScrollBar](#), [JScrollPane](#), [JSeparator](#), [JSlider](#), [JSpinner](#), [JSplitPane](#), [JTabbedPane](#), [JTable](#), [JTableHeader](#), [JTextComponent](#), [JToolBar](#), [JToolTip](#), [JTree](#), [JViewport](#)

# 4.1 Interfaces gráficas de Usuario

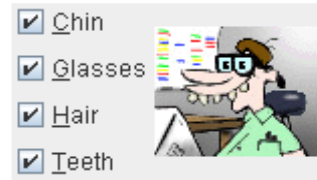
## JComponents

### Basic Controls

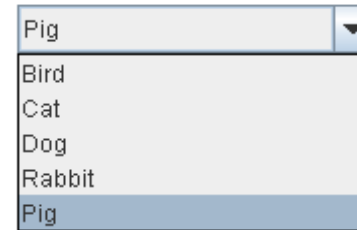
Simple components that are used primarily to get input from the user; they may also show simple state.



[JButton](#)



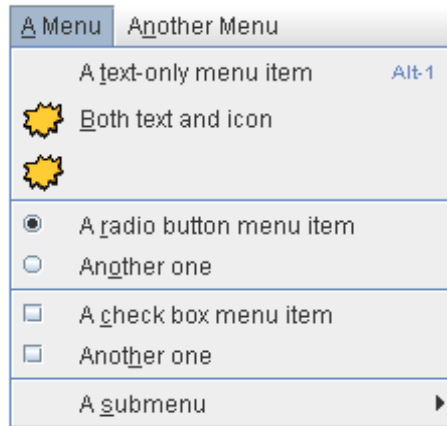
[JCheckBox](#)



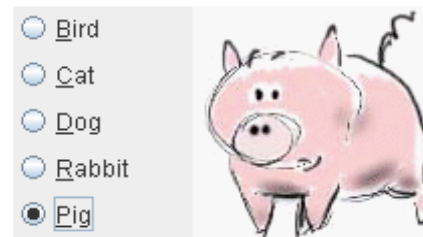
[JComboBox](#)



[JList](#)



[JMenu](#)



[JRadioButton](#)



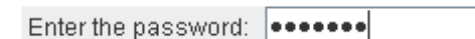
[JSlider](#)



[JSpinner](#)



[JTextField](#)



[JPasswordField](#)

<https://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html>

# 4.1 Interfaces gráficas de Usuario


JComponents – Cómo colocarlos: manualmente

Ejemplo de programa: ASOCIACIÓN  
como **Objeto** dentro de un *método*

```
public static void exFramePane() {  
  
    JFrame frame= new JFrame();  
  
    frame.setTitle("Ejemplo de JFrame con JPanel");  
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    frame.setBounds(100, 100, 500, 500); //(pos_pantalla(x,y), size(x,y))  
    //frame.setSize(500, 300); //Similar set size and default position. pixels  
  
    JPanel contentPane = new JPanel();  
  
    frame.setContentPane(contentPane);  
    contentPane.setLayout(null);  
  
    JButton initButton = new JButton("Presionar");  
    initButton.setBounds(200, 100, 100, 50); //(pos_pantalla(x,y), size(x,y))  
    frame.add(initButton);  
  
    frame.setVisible(true);  
}
```

```
public class GUI {  
    public static void main(String args[]) {  
        exFramePane();  
    }  
}
```

Importante para poder  
usar coordenadas (null)





# 4.1 Interfaces gráficas de Usuario

Jcomponents – Cómo colocarlos: manualmente

Ejemplo de programa: HERENCIA  
como *Clase hija* de *JFrame*

```
public class exFrameClass extends JFrame {

    public exFrameClass() {
        setTitle("Ejemplo de JFrame con JPanel");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 500, 500); //(pos_pantalla(x,y), size(x,y))
        //frame.setSize(500, 300); //Similar set size and default position. pixels

        JPanel contentPane = new JPanel();
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JButton initButton = new JButton("Presionar");
        initButton.setBounds(200, 100, 100, 50);
        contentPane.add(initButton);
        add(contentPane);

        setVisible(true);
    }
}
```

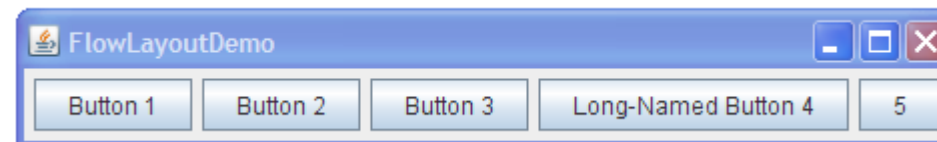
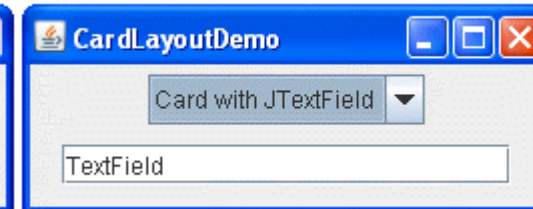
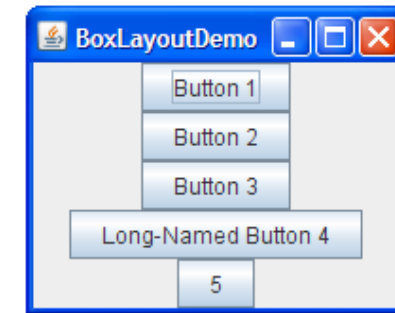
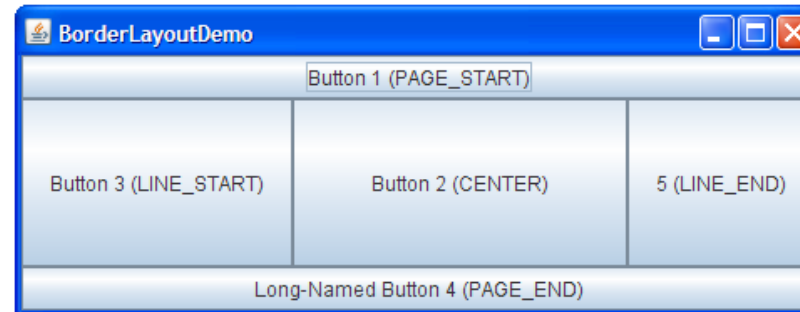
```
public class GUI {
    public static void main(String args[]) {
        new exFrameClass();
    }
}
```

# 4.1 Interfaces gráficas de Usuario

Jcomponents – Cómo colocarlos automáticamente (Java Layout Manager)

Si no desea colocar cada elemento manualmente mediante coordenadas, también se pueden emplear gestores de capas para realizar colocaciones automáticas. Existen diferentes gestores de Layouts de AWT & Swing:

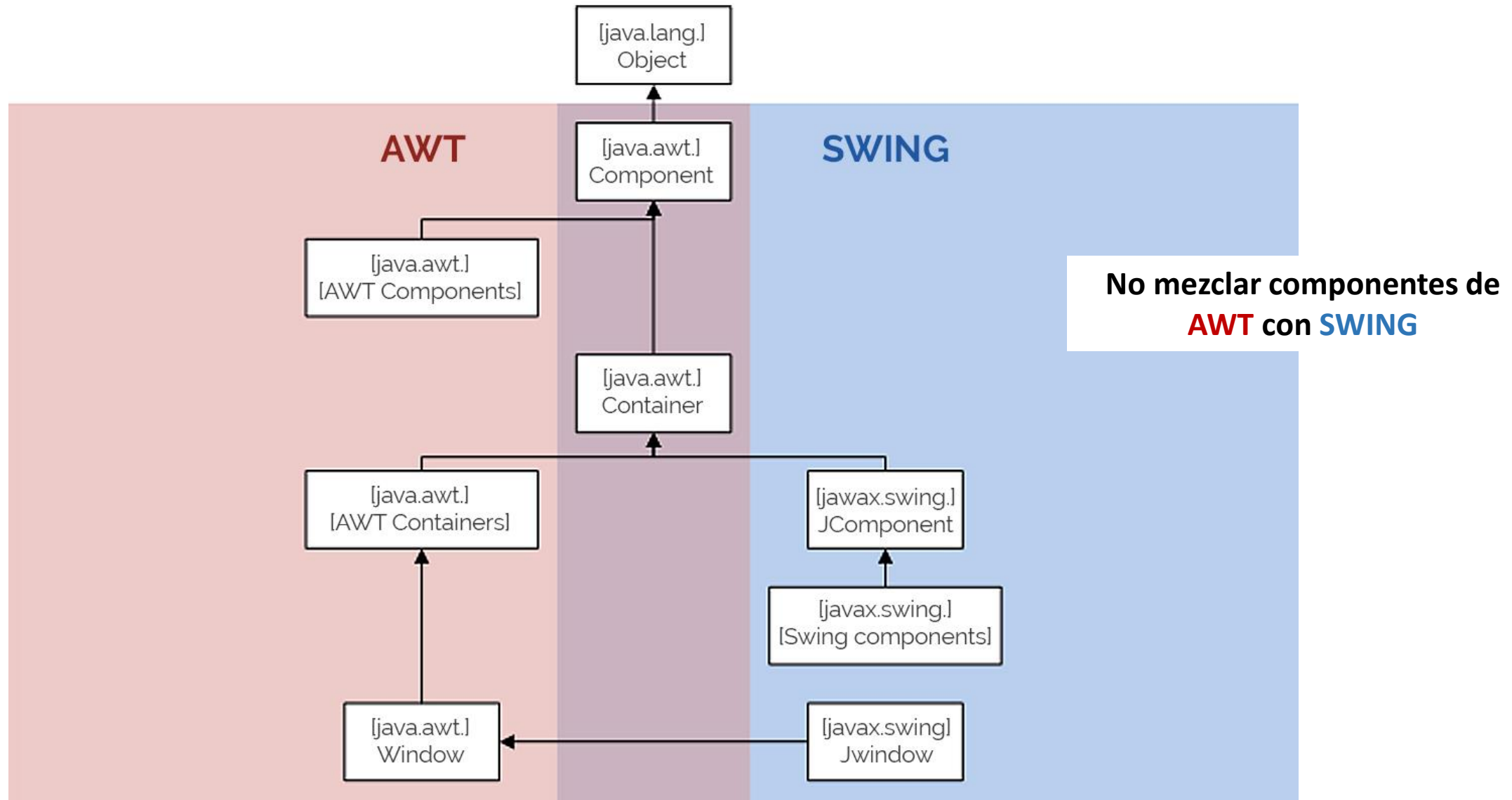
- [BorderLayout](#)
- [BoxLayout](#)
- [CardLayout](#)
- [FlowLayout](#)
- [GridBagLayout](#)
- [GridLayout](#)
- [GroupLayout](#)
- [SpringLayout](#)



[...]

# 4.1 Interfaces gráficas de Usuario

## AWT vs Swing - Components



# 4.1 Interfaces gráficas de Usuario

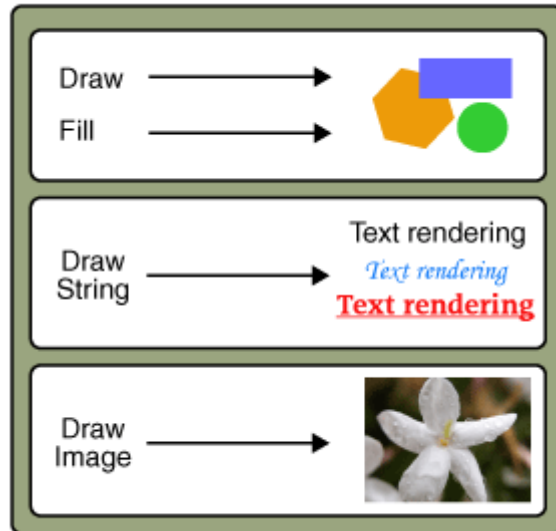
## AWT vs Swing – Advantages

No.	Java AWT	Java Swing
1)	AWT components are <b>platform-dependent</b> .	Java swing components are <b>platform-independent</b> .
2)	AWT components are <b>heavyweight</b> .	Swing components are <b>lightweight</b> .
3)	AWT <b>doesn't support pluggable look and feel</b> .	Swing <b>supports pluggable look and feel</b> .
4)	AWT provides <b>less components</b> than Swing.	Swing provides <b>more powerful components</b> such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
5)	AWT <b>doesn't follows MVC</b> (Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing <b>follows MVC</b> .



# 4.1 Interfaces gráficas de Usuario

## Graphics - Dibujar



**Swing:** Crear clase que extienda JPanel/JComponent y sobreexcribir *paintComponent*.

**AWT:** Crear clase que extienda Canvas y sobreexcribir *paint*. También sobreexcribir update para evitar flickering:

```
@Override
public void update(Graphics g)
{
    paint(g);
}
```

En cualquier caso un contenedor **Top Level** (ej.: JFrame/Frame) es necesario.

```
public void createImage() {
    BufferedImage img = ImageIO.read(image_path);
    int w = img.getWidth(null);
    int h = img.getHeight(null);
    BufferedImage bi = new BufferedImage(w, h, BufferedImage.TYPE_INT_ARGB);
    //ARGB represents an image with 8-bit RGBA color components packed into integer pixels.

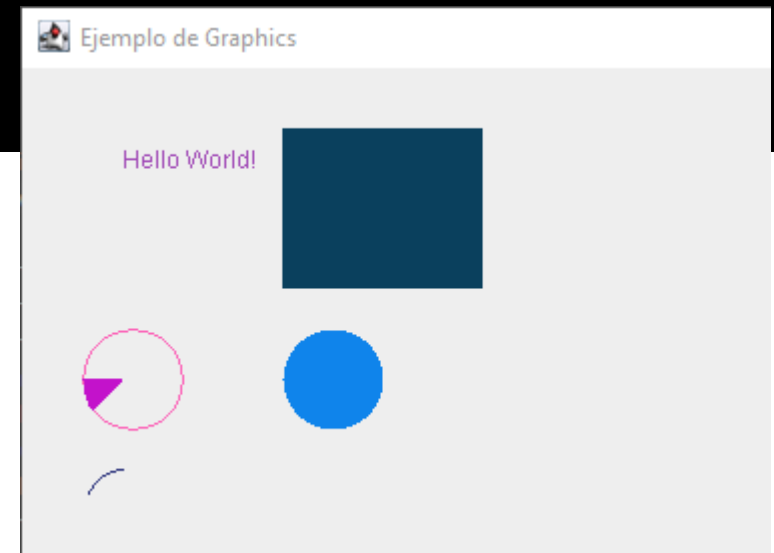
    Graphics g = bi.getGraphics();
    g.drawImage(img, 0, 0, null);
}
```

# 4.1 Interfaces gráficas de Usuario

## Graphics - Dibujar

```
public class drawExample extends JPanel{  
  
    private JFrame frame;  
  
    public drawExample() {  
        frame= new JFrame("Ejemplo de Graphics");  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        frame.setBounds(100, 100, 800, 800); frame.add(this);  
        frame.setVisible(true);  
    }  
  
    @Override  
    public void paint(Graphics g) {  
  
        setBackground(Color.WHITE);  
        g.setColor(getRandomColor());  
        g.drawString("Hello World!", 50, 50);  
        g.fillRect(130, 30,100,80);  
        g.drawOval(30,130,50,50);  
        g.fillOval(130,130,50,50);  
        g.drawArc(30,200,40,50,90,60);  
        g.fillArc(30,130,40,50,180,40);  
    }  
}
```

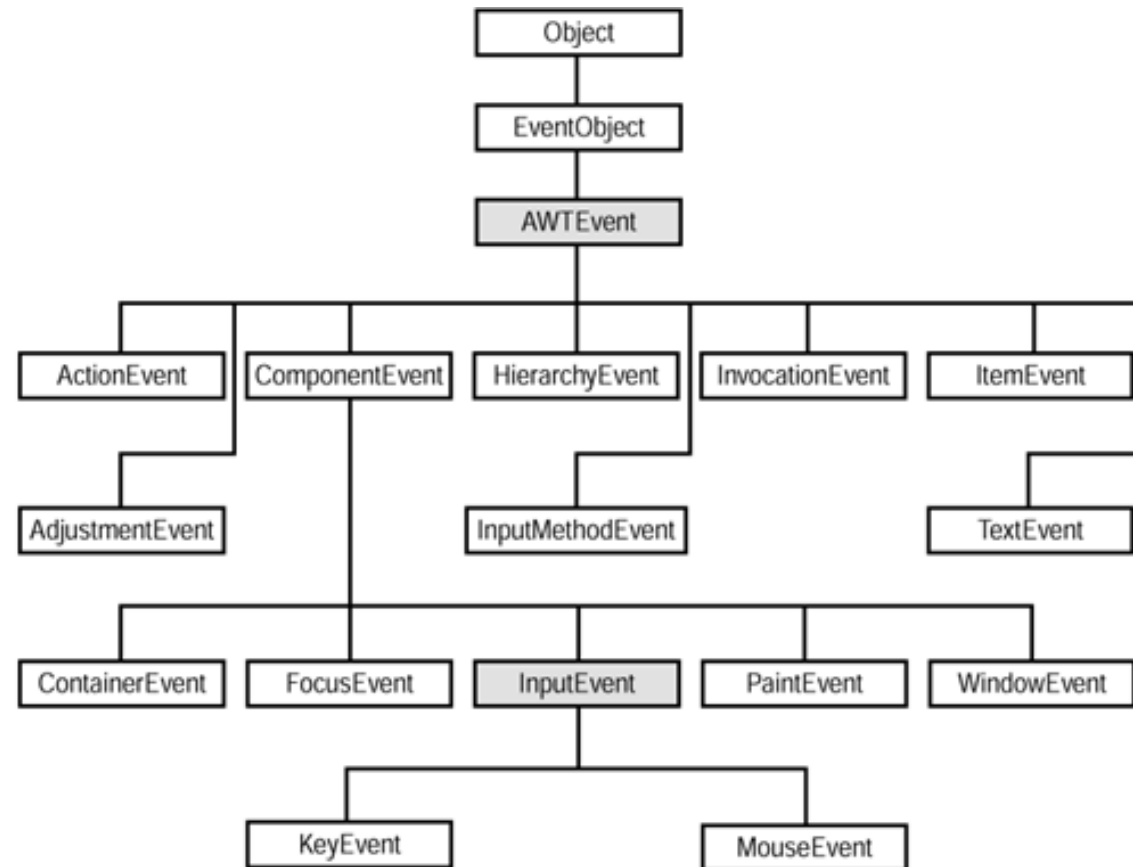
```
public void update(Graphics g)  
{  
    paint(g);  
}  
  
public Color getRandomColor() {  
  
    int R = (int) (Math.random()*256);  
    int G = (int)(Math.random()*256);  
    int B= (int)(Math.random()*256);  
    return new Color(R, G, B);  
}  
  
}
```



## 4.2 Gestión de eventos

Gestionar eventos al interactuar con los objetos

Cada componente puede aplicar un tipo determinado de Listeners ([Tabla](#))



## 4.2 Gestión de eventos

Tipos de Listeners implementables por cada objeto

This table lists Swing components with their specialized listeners

Component	Action Listener	Caret Listener	Change Listener	Document Listener, Undoable Edit Listener	Item Listener	List Selection Listener	Window Listener	Other Types of Listeners
button	✓		✓		✓			
check box	✓		✓		✓			
color chooser			✓					
combo box	✓				✓			
dialog							✓	
editor pane		✓		✓				hyperlink
file chooser	✓							
formatted text field	✓	✓		✓				
frame							✓	
internal frame								internal frame
list						✓		list data
menu								menu
menu item	✓		✓		✓			menu key menu drag mouse
option pane								
password field	✓	✓		✓				
popup menu								popup menu
progress bar			✓					
radio button	✓		✓		✓			
slider			✓					



# 4.2 Gestión de eventos

## Listeners: métodos y eventos

NOMBRE LISTENER	DESCRIPCIÓN	MÉTODOS	EVENTOS
<b>ActionListener</b>	Se produce al hacer click en un componente, también si se pulsa Enter teniendo el foco en el componente.	public void actionPerformed(ActionEvent e)	<ul style="list-style-type: none"><li>•<b>JButton</b>: click o pulsar Enter con el foco activado en él.</li><li>•<b>JList</b>: doble click en un elemento de la lista.</li><li>•<b>JMenuItem</b>: selecciona una opción del menú.</li><li>•<b>TextField</b>: al pulsar Enter con el foco activado.</li></ul>
<b>KeyListener</b>	Se produce al pulsar una tecla. según el método cambiara la forma de pulsar la tecla.	public void keyTyped(KeyEvent e)  public void keyPressed(KeyEvent e)  public void keyReleased(KeyEvent e)	Cuando pulsamos una tecla, según el Listener:  <ul style="list-style-type: none"><li>•<b>keyTyped</b>: al pulsar y soltar la tecla.</li><li>•<b>keyPressed</b>: al pulsar la tecla.</li><li>•<b>keyReleased</b>: al soltar la tecla.</li></ul>
<b>FocusListener</b>	Se produce cuando un componente gana o pierde el foco, es decir, que esta seleccionado.	public void focusGained(FocusEvent e)  public void focusLost(FocusEvent e)	Recibir o perder el foco.
<b>MouseListener</b>	Se produce cuando realizamos una acción con el ratón.	public void mouseClicked(MouseEvent e)  public void mouseEntered(MouseEvent e)  public void mouseExited(MouseEvent e)  public void mousePressed(MouseEvent e)  public void mouseReleased(MouseEvent e)	Según el Listener:  <ul style="list-style-type: none"><li>•<b>mouseClicked</b>: pinchar y soltar.</li><li>•<b>mouseEntered</b>: entrar en un componente con el puntero.</li><li>•<b>mouseExited</b>: salir de un componente con el puntero</li><li>•<b>mousePressed</b>: presionar el botón.</li><li>•<b>mouseReleased</b>: soltar el botón.</li></ul>
<b>MouseMotionListener</b>	Se produce con el movimiento del mouse.	public void mouseDragged(MouseEvent e)  public void mouseMoved(MouseEvent e)	Según el Listener:  <ul style="list-style-type: none"><li>•<b>mouseDragged</b>: click y arrastrar un componente.</li><li>•<b>mouseMoved</b>: al mover el puntero sobre un elemento</li></ul>

## 4.2 Gestión de eventos

ActionListener – Botones y similares

```
public class ExPanelEvents extends JPanel implements ActionListener {
```

```
    JButton botonExit, botonCambiarColor;
```

```
    ExPanelEvents(){
        super(new BorderLayout());
        botonExit=new JButton("Salir");
        add(botonExit, BorderLayout.SOUTH);
        botonExit.addActionListener(this);

        botonCambiarColor=new JButton("Cambiar color");
        add(botonCambiarColor, BorderLayout.WEST);
        botonCambiarColor.addActionListener(this);
    }
```

```
@Override
```

```
public void actionPerformed(ActionEvent e) {
```

```
    if (e.getSource()==botonExit) { System.exit(0);}
    if (e.getSource()==botonCambiarColor) {setBackground(getRandomColor());}
```

```
    }
```

```
    public Color getRandomColor() {

        int R = (int) (Math.random()*256);
        int G = (int) (Math.random()*256);
        int B= (int) (Math.random()*256);

        return new Color(R, G, B);
    }
```

## 4.2 Gestión de eventos

KeyListener – Entradas por teclado

```
public class ExPanelEvents extends JPanel implements KeyListener{

    JLabel result;

    ExPanelEvents(){
        super(new BorderLayout());
        result = new JLabel();
        result.setText("  Presiona una tecla: ");
        add(result, BorderLayout.CENTER);

        addKeyListener(this);
    }
}
```

```
//KeyListener events
@Override
public void keyTyped(KeyEvent e) {
    result.setText("    Presionado: "
                  + e.getKeyChar());
}

@Override
public void keyPressed(KeyEvent e) { }

@Override
public void keyReleased(KeyEvent e) { }
```

```
e.getKeyCode();
```

[https://docs.oracle.com/javase/7/docs/api/constant-values.html#java.awt.event.KeyEvent.VK\\_UP](https://docs.oracle.com/javase/7/docs/api/constant-values.html#java.awt.event.KeyEvent.VK_UP)

## 4.2 Gestión de eventos

MouseListener y MouseMotionListener – Eventos del ratón

```
public class ExPanelMouseListener extends JPanel implements MouseListener, MouseMotionListener{

    Label l;

    ExPanelMouseListener(){
        addMouseListener(this);
        setLayout(null);

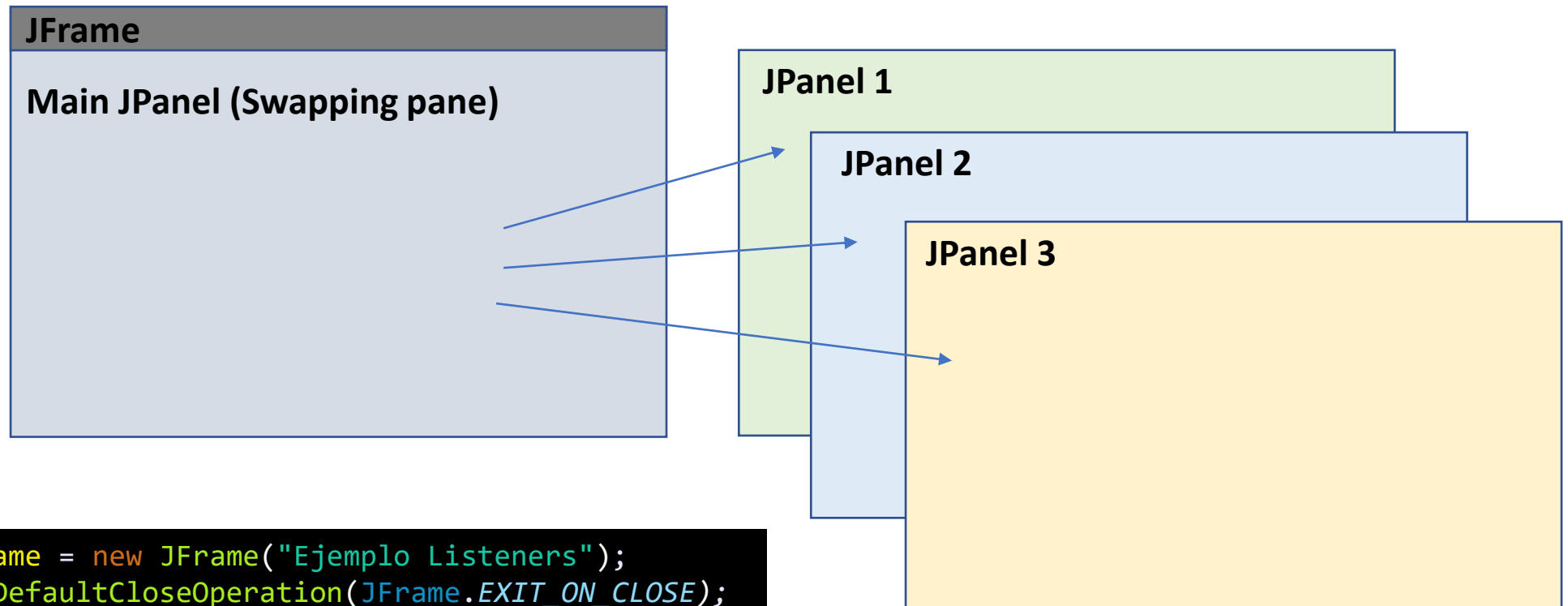
        l=new Label();
        l.setBounds(20,50,100,20);
        add(l);
        setSize(300,300);
        setVisible(true);
    }

    public void mouseClicked(MouseEvent e) {
        l.setText("Mouse Clicked");
        Graphics g=getGraphics();
        g.setColor(Color.BLUE);
        g.fillOval(e.getX()-15,e.getY()-15,30,30);}
    //MouseListener events
    public void mouseEntered(MouseEvent e) {
        l.setText("Mouse Entered");}
    public void mouseExited(MouseEvent e) {
        l.setText("Mouse Exited");}
    public void mousePressed(MouseEvent e) {
        l.setText("Mouse Pressed");}
    public void mouseReleased(MouseEvent e) {
        l.setText("Mouse Released"); }
    //MouseMotionListener events
    public void mouseDragged(MouseEvent e) {}
    public void mouseMoved(MouseEvent e) {}
}
```



## 4.2 Gestión de eventos

Cambio del contenido de la ventana



```
JFrame frame = new JFrame("Ejemplo Listeners");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

frame.setBounds(100, 100, 400, 400);

JComponent newContentPane = new SwapingPanel();
newContentPane.setOpaque(true);
frame.setContentPane(newContentPane);

frame.setVisible(true);
```

```
public class SwapingPanel extends JPanel implements ActionListener{
```

```
    JPanel firstPanel = new ExPanelEvents();  
    JPanel secondPanel = new ExPanelMouseListener();
```

```
    public SwapingPanel() {  
        super(new BorderLayout());  
  
        JButton swap1 = new JButton("Cambiar ventana");  
        swap1.addActionListener(this);  
        firstPanel.add(swap1, BorderLayout.NORTH);  
        add(firstPanel);  
    }
```

```
    public void actionPerformed(ActionEvent e) {  
        for (Component component : getComponents())  
            if (firstPanel == component) {  
                remove(firstPanel);  
                add(secondPanel);  
            } else {  
                remove(secondPanel);  
                add(firstPanel);  
            }  
        repaint();  
        revalidate();  
    }  
}
```

JPanel 1

JPanel 2

Comprueba cuál de los components de Jpanel (añadidos desde aquí) ha lanzado la excepción y actúa en consecuencia

# Bibliografía

- ❖ Deitel, H. M. & Deitel, P. J.(2008). *Java: como programar*. Pearson education. Séptima edición.
- ❖ The Java tutorials. <https://docs.oracle.com/javase/tutorial/>
- ❖ Páginas de bibliotecas, tutoriales, etc.:
  - ❖ <https://docs.oracle.com/javase/tutorial/uiswing/learn/index.html>
  - ❖ <https://help.eclipse.org/latest/index.jsp?topic=%2Forg.eclipse.wb.doc.user%2Fhtml%2Findex.html>
  - ❖ <https://www.w3schools.com/java/default.asp>
  - ❖ <https://docstore.mik.ua/orelly/java-ent/jnut/index.htm>

# Técnicas de Programación Avanzada (Java)

Curso 2021-2021

```
exit(); //Gracias!
```