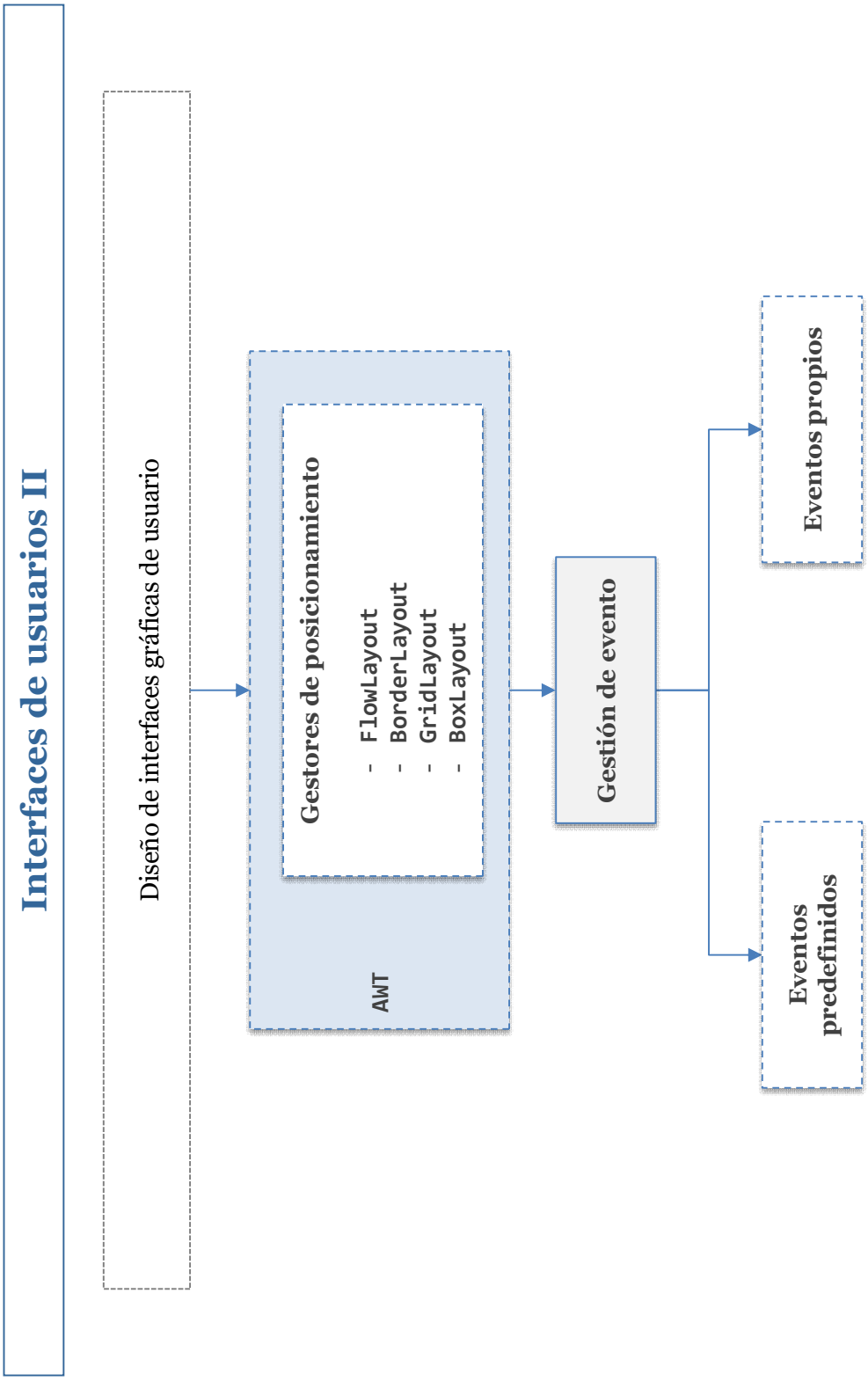
Interfaces de usuarios (II)

TEMA 6

Esquema



Ideas clave

6.1. ¿Cómo estudiar este tema?

Para estudiar este tema lee el capítulo 5, «Interfaz gráfica (GUI)» **(páginas 162-176)** del siguiente libro, disponible en el aula virtual en virtud del artículo 32.4 de la Ley de Propiedad Intelectual:

Sznajdleder, P. A. (2013). *Java a Fondo: estudio del lenguaje y desarrollo de aplicaciones* (2ª ed.). Buenos Aires: Alfaomega Grupo Editor.

Para comprobar si has comprendido los conceptos puedes realizar el test de autoevaluación del tema.

En este tema vamos a ver en profundidad cómo se gestionan los elementos visuales dentro de Java. Veremos:

- Introducción a AWT.
- Organización de los elementos visuales.

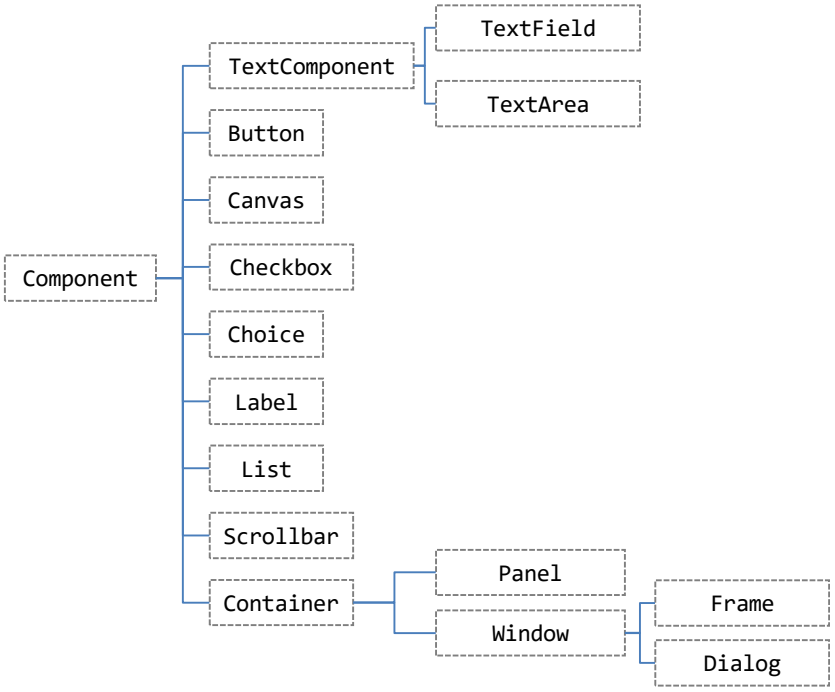
6.2. AWT

Las GUI (Graphical User Interface) se construyen a partir de componentes. Estos componentes se encuentran en el paquete `java.awt` (Abstract Window Toolkit).

En la jerarquía de clases de `awt`, tienen especial importancias las clases `Component` y `Container`.

Toda clase que hereda de la clase `Component` es un componente y toda clase que hereda de la clase `Container` es un contenedor (área en la que se pueden colocar componentes).

Jerarquía de clases `Component`:



Para añadir más componentes y situarlos en una posición exacta, se utiliza un **gestor de posicionamiento o administrador de diseños**, mediante la interfaz `LayoutManager` del paquete `java.awt`.

Las siguientes clases implementan la interfaz `LayoutManager`:

`FlowLayout`	`BorderLayout`
`GridLayout`	`BoxLayout`

El método `setLayout(LayoutManager m)` permite establecer el gestor de posicionamiento. El argumento de este método es un objeto de alguna de las clases que implementan la interfaz `LayoutManager`.

6.3. Gestores de posicionamiento

`FlowLayout`

Es el **administrador de diseños más básico**, en el que los componentes se colocan de izquierda a derecha, dejando, por omisión, un espacio vertical y horizontal de 5 píxeles entre sus componentes.

Constructores:

- `FlowLayout()`: Construye el objeto `FlowLayout` con alineación central.
- `FlowLayout(int alineación)`: construye el objeto `FlowLayout` con la alineación que se le pasa como argumento (RIGHT, LEFT, CENTER).
- `FlowLayout(int alineación, int distancia_h, int distancia_v)`: construye el objeto `FlowLayout` con la alineación indicada en el primer argumento. La distancia entre los componentes se indica en pixeles con los dos últimos argumentos.

Ejemplo:

```
MiFrame {

        setTitle("FlowLayoutprueba");
        getContentPane().setLayout(new FlowLayout());
        getContentPane().add(new JButton("Button 1"));
        getContentPane().add(new JButton("Button 2"));
        getContentPane().add(new JButton("Button 3"));
        getContentPane().add(new JButton("Button 4"));
        getContentPane().add(new JButton("Button 5"));
        setSize(300,200);
}
```

BorderLayout

Este **gestor de posicionamiento** coloca los componentes en cinco zonas (NORTH, SOUTH, EAST, WEST y CENTER). Por omisión se colocan en el centro.

Constructores:

- `BorderLayout()`: Construye el objeto `BorderLayout` sin separación entre los componentes.
- `BorderLayout(int distancia_h, int distancia_v)`: Construye un objeto `BorderLayout` con una separación entre los componentes indicada en píxeles en los dos argumentos del constructor.

Método `add`:

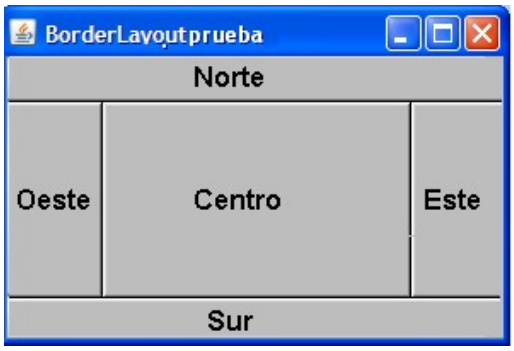
El método `add` permite añadir componentes. Su sintaxis es:

```
add(Component comp, int area)
```

El argumento área indica donde se sitúa el componente. Los valores posibles son: `BorderLayout.CENTER`, `BorderLayout.NORTH`, `BorderLayout.SOUTH`, `BorderLayout.WEST` y `BorderLayout.EAST`.

Ejemplo:

```
setTitle("BorderLayoutprueba");
getContentPane().setLayout(new BorderLayout());
getContentPane().add(new JButton("Centro"), BorderLayout.CENTER);
getContentPane().add(new JButton("Norte"), BorderLayout.NORTH);
getContentPane().add(new JButton("Sur"), BorderLayout.SOUTH);
getContentPane().add(new JButton("Este"), BorderLayout.EAST);
getContentPane().add(new JButton("Oeste"), BorderLayout.WEST);
setSize(300,200);
```

`GridLayout`

El gestor de posicionamiento `GridLayout` **coloca los componentes en filas y columnas**.

Todos los componentes tendrán el mismo tamaño y se comienzan a colocar en la celda superior izquierda, avanzando de izquierda a derecha hasta que se completa la fila. Luego se continúa por la siguiente fila.

Constructores:

- `GridLayout()`: Construye el objeto `GridLayout` con una única fila y una única columna.
- `GridLayout(int filas, int columnas)`: Construye el objeto `GridLayout` el número de filas y columnas indicados en los argumentos.
- `GridLayout(int filas, int columnas, int distancia_h, int distancia_v)`: Construye el objeto `GridLayout` con el número de filas y columnas indicados en los dos primeros argumentos y con una separación horizontal y vertical entre los componentes indicada en pixeles en los dos últimos argumentos.

Ejemplo:

```
setTitle(“GridLayoutprueba”);
getContentPane().setLayout(new GridLayout(3,2,5,5));
getContentPane().add(new JButton("Button 1"));
getContentPane().add(new JButton("Button 2"));
getContentPane().add(new JButton("Button 3"));
getContentPane().add(new JButton("Button 4"));
getContentPane().add(new JButton("Button 5"));
getContentPane().add(new JButton("Button 6"));
```


BoxLayout

El gestor de posicionamiento `BoxLayout` coloca los componentes en una única fila o columna dentro de un objeto de la clase `Container`.

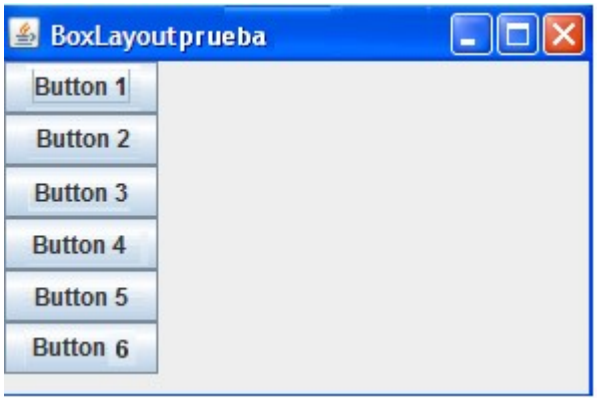
Constructor:

» `BoxLayout(Container destino, int orientación)`

Siendo destino el contenedor a utilizar y orientación indica si los componentes se colocan en fila o en columna. Los valores posibles de orientación son: `BoxLayout.X_AXIS` o `BoxLayout.Y_AXIS`.

Ejemplo:

```
setTitle("BoxLayoutprueba");
JPanel panel1 = new JPanel();
panel1.setLayout(new BoxLayout(panel,BoxLayout.Y_AXIS));
panel1.add(new JButton("Button 1"));
panel1.add(new JButton("Button 2"));
panel1.add(new JButton("Button 3"));
panel1.add(new JButton("Button 4"));
panel1.add(new JButton("Button 5"));
panel1.add(new JButton("Button 6"));
setContentPane(panel1);
```

La clase `Box`

La clase `Box` define contenedores que tienen como gestor de posicionamiento un `BoxLayout`.

Combinar varios gestores

Una GUI se puede construir se puede construir **combinando varios gestores de posicionamiento.** Cuando se utilizan varios gestores de posicionamiento se deben tener en cuenta los siguientes puntos:

- En un contenedor solo se puede utilizar un tipo de gestor.
- Un contenedor de primer nivel puede tener otros contenedores anidados, para ello se utilizan objetos `JPanel`.

La clase `JPanel`

Un objeto `JPanel` es un contenedor con gestor de posicionamiento propio que puede contener varios componentes.

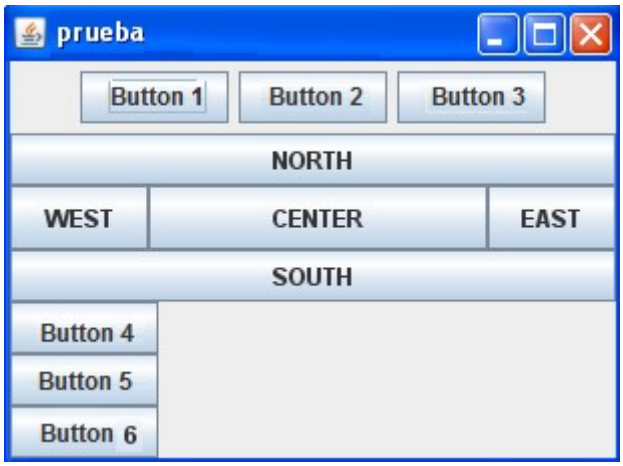
Algunos de los constructores de la clase `JPanel` son:

- `JPanel ()`: crea un objeto `JPanel` con gestor de posicionamiento `FlowLayout`.
- `JPanel (LayoutManager Layout)`: crea un objeto `JPanel` con el gestor de posicionamiento indicado en el argumento.
- `JPanel (LayoutManager Layout, boolean Buffering)`

Ejemplo:

```
JPanel panel1 = new JPanel(new FlowLayout());
JPanel panel2 = new JPanel(new BorderLayout());
JPanel panel3 = new JPanel();
panel1.add(new JButton("Button 1"));
panel1.add(new JButton("Button 2"));
panel1.add(new JButton("Button 3"));


panel2.add(new JButton("CENTER"), BorderLayout.CENTER);
panel2.add(new JButton("NORTH"), BorderLayout.NORTH);
panel2.add(new JButton("SOUTH"), BorderLayout.SOUTH);
panel2.add(new JButton("EAST"), BorderLayout.EAST);
panel2.add(new JButton("WEST"), BorderLayout.WEST);
panel3.setLayout(new BoxLayout(panel3,BoxLayout.Y_AXIS));
panel3.add(new JButton("Button 4"));
panel3.add(new JButton("Button 5"));
panel3.add(new JButton("Button 6”);
getContentPane().setLayout(new BorderLayout());
getContentPane().add(panel1, BorderLayout.NORTH);
getContentPane().add(panel2, BorderLayout.CENTER);
getContentPane().add(panel3, BorderLayout.SOUTH);
```



6.4. Gestión de eventos

Los **eventos** que se generan dentro de las interfaces gráficas deben ser gestionados para poder responder a ellos.

Los eventos **siempre tienen una fuente**, que es el objeto que es capaz de detectar el evento y notificar a los receptores que se ha producido. Por otro lado debe existir un Receptor (`Listener`) que será el encargado de gestionar el evento.

Los siguientes ejemplos han sido extraídos del siguiente enlace bajo la autorización de su autor:

http://www.colimbo.net/documentos/documentacion/Java/JavaTema06B.pdf

Ejemplo de implementación de eventos:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class EjemploEventos{
 public static void main(String args[]){
       FrameBotones frm = new FrameBotones();
       frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       frm.setVisible(true);
 }
}
class FrameBotones extends JFrame{
 JButton btnAzul = new JButton("Azul");
 JButton btnAmarillo = new JButton("Amarillo");
 JButton btnRojo = new JButton("Rojo");


 FrameBotones(){
       setTitle("Ejemplo de eventos");
       getContentPane().setLayout(new FlowLayout());
       getContentPane().add(btnAzul);
       getContentPane().add(btnAmarillo);
       getContentPane().add(btnRojo);
       setVisible(true);
       btnAzul.addActionListener(new MiOyente());
       btnAmarillo.addActionListener(new MiOyente());
       btnRojo.addActionListener(new MiOyente());
       setSize(200,300);
 }

 class MiOyente implements ActionListener{
       public void actionPerformed(ActionEvent event){
             if(event.getSource() == btnAzul)
                   getContentPane().setBackground(Color.BLUE);
             else if(event.getSource() == btnAmarillo)
                   getContentPane().setBackground(Color.YELLOW);
             else if(event.getSource() == btnRojo)
                   getContentPane().setBackground(Color.RED);

       }
 }
}
```


Se pueden crear clases oyentes propias:

```
class HolaMundoFrame extends JFrame{
	JPanel panelBoton = new JPanel();
	JPanel panelEtiqueta = new JPanel();
	JButton btn = new JButton("Pulsa");
	JLabel lbl = new JLabel("¡Hola, mundo!");
	HolaMundoFrame(){
		setTitle("Hola Mundo!");
		panelBoton.add(btn);
		panelEtiqueta.add(lbl);
		setContentPane(panelBoton);
		pack();
		btn.addActionListener(new ActionListener(){
			public void actionPerformed(ActionEvent event){
				setContentPane(panelEtiqueta);
				pack();
			}
		});
	}
}
```

6.5. Adaptadores

El **problema** cuando utilizamos `Listeners` es que **debemos implementar todos los métodos**, para evitarlo algunas interfaces tienen lo que se denominan **adaptadores**.

Adaptadores

Son clases que proporcionan **métodos vacíos** para cada uno de los métodos de la interfaz.

A continuación se detallan algunos adaptadores y la interface que implementan (adaptador – `Listener`):

- `ComponentAdapter - ComponentListener`
- `ContainerAdapter – ContainerListener`
- `FocusAdapter – FocusListener`
- `KeyAdapter – KeyListener`
- `MouseAdapter – MouseListener`
- `MouseMotionAdapter – MouseMotionListener`
- `WindowAdapter – WindowListener`
- `WindowAdapter – WindowFocusListener`

Lo + recomendado

No dejes de leer...

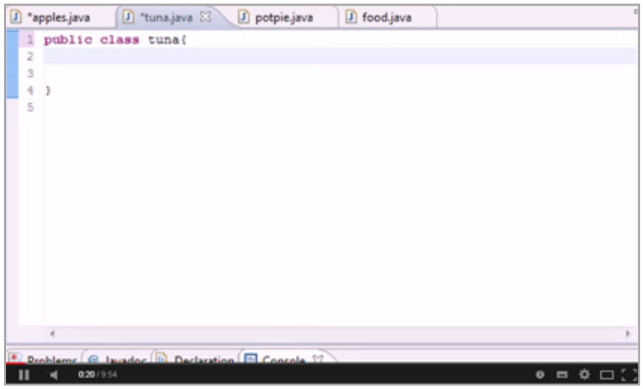
Clase `Box`

En la página de Oracle podrás encontrar información detallada sobre la clase `Box`.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
https://docs.oracle.com/javase/7/docs/api/javax/swing/Box.html

No dejes de ver...

Crear interfaces de usuario

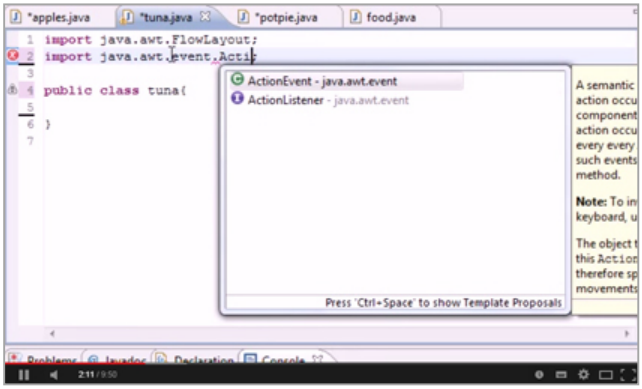


Vídeo en el que se explica cómo crear interfaces de usuario.

Accede al vídeo desde el aula virtual o a través de la siguiente dirección web:
http://www.youtube.com/watch?v=jUdIAgJ7JKo

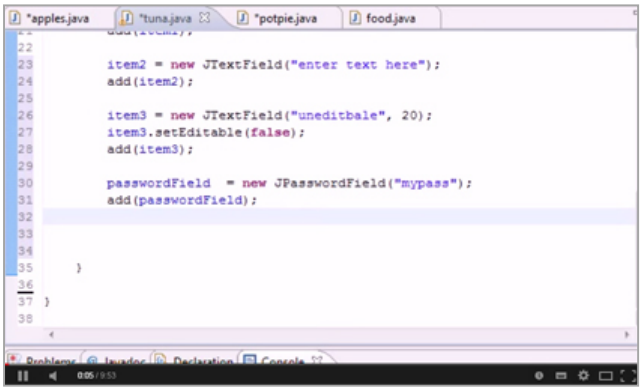
Gestionar eventos en Java

Vídeo en el que se explica cómo gestionar eventos en Java.



Accede al vídeo desde el aula virtual o a través de la siguiente dirección web:
http://www.youtube.com/watch?v=3EE7E3bvfe8

Crear eventos en Java



Vídeo en el que se explica cómo crear eventos en Java.

Accede al vídeo desde el aula virtual o a través de la siguiente dirección web:
http://www.youtube.com/watch?v=qhYook53olE

+ Información

A fondo

Desarrollo y generación de interfaces de usuario

Lozano, M. D., González, P., Ramos, I., Moreno, F., & Molina, J. P. (2002). Desarrollo y generación de interfaces de usuario a partir de técnicas de análisis de tareas y casos de uso. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial,* 6(16).

Paper en el que se explican diferentes técnicas para el desarrollo de interfaces de usuario.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://www.redalyc.org/articulo.oa?id=92561610

Verificar interfaces de usuario

Mehlitz, P., NASA Ames Res., Tkachuk, O, Ujma, M. (2011). JPF-AWT: Model checking GUI applications. *Automated Software Engineering (ASE).* doi: 10.1109/ASE.2011.6100131

Paper en el que se explica cómo verificar interfaces de usuario.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
https://ti.arc.nasa.gov/m/groups/rse/papers/Mehlitz_jpf-awt.pdf

Uso de componentes en SWING

Tutorial sobre la utilización de componentes en SWING.

Accede al tutorial desde el aula virtual o a través de la siguiente dirección web:
http://docs.oracle.com/javase/tutorial/uiswing/components/

Gestores de posicionamiento

Información detallada sobre los distintos gestores de posicionamiento.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
https://docs.oracle.com/javase/tutorial/uiswing/layout/

Recursos externos

Eclipse

Eclipse es una plataforma de desarrollo. Fue concebida desde sus orígenes para convertirse en una plataforma de integración de herramientas de desarrollo. No tiene en mente un lenguaje específico, sino que es un IDE genérico.

Accede a la página desde el aula virtual o a través de la siguiente dirección web:
https://www.eclipse.org/downloads/
A través de la siguiente dirección puedes ver cómo crear un proyecto en Eclipse desde cero:
https://www.youtube.com/watch?v=J9lkAK0L16I

Test

1. Para establecer el gestor de posicionamiento en un contenedor se debe:

A. Utilizar el método `setLayout`.

B. Indicar en el constructor del gestor.

C. Se indica al final de la construcción del objeto.

D. No existen los gestores de posicionamiento.

2. A través del gestor de contenido `FlowLayout`:

A. Los componentes fluyen de arriba abajo.

B. Los componentes fluyes de derecha a izquierda.

C. Los componentes fluyen de izquierda a derecha.

D. Los componentes fluyen de abajo a arriba.

3. El gestor de contenidos `BorderLayout`:

A. Divide el contenedor en cuatro zonas, norte, sur, este y oeste.

B. Divide el contenedor en tantas partes como se quieran.

C. Divide el contendor en cinco partes, norte, sur, este, oeste y centro.

D. Divide el contenedor en dos partes.

4. El gestor de contenidos `GridLayout`:

A. Los componentes se colocan en filas.

B. Los componentes se colocan en orden de aparición.

C. Los componentes se colocan por coordenadas.

D. Los componentes se colocan en una rejilla de celdas iguales.

5. El gestor de contenidos `BoxLayout`:

A. Muestra los componentes en una única fila o columna.

B. Muestra los componentes en tantas filas o columnas como queramos.

C. Muestra los componentes en una fila.

D. Muestra los componentes en una columna.

6. Qué afirmación es verdad:

A. Un contenedor puede tener tantos gestores como se quieran.

B. Un contenedor solo puede tener un gestor.

C. Un contenedor puede usar varios gestores, pero no a la vez.

D. Un contenedor no tiene gestores.

7. Que afirmación es correcta:

A. Los eventos pueden no tener fuente.

B. Los eventos son generados por el `Listener`.

C. Los eventos se crean siempre a través de la instanciación del objeto `Action`.

D. Los eventos siempre tienen una fuente.

8. En Java:

A. Se pueden crear `Listener` propios.

B. Se pueden crear `Listener` propios pero siempre que se definan en el constructor.

C. No se pueden crear `Listener` propios.

D. Todas las clases son `listener`.

9. Los adaptadores se utilizan para:

A. Gestionar objetos de tipo `action`.

B. Para simplificar la creación de `Listener` propios.

C. Para adaptar los programas a las nuevas versiones de eventos.

D. Para hacer compatibles los orígenes de los eventos y los `Listener`.

10. Los adaptadores en Java:

A. Tienen el mismo nombre que el `Listener`.

B. Tienen el mismo nombre que el `Listener` pero con `Adapter` al final en lugar de `Listener`.

C. Tienen un nombre diferente al `Listener`.

D. Los debe crear el usuario.

