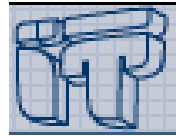


Swing

Andrés Marín López

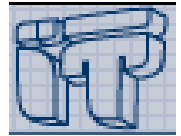
Jesús Arias Fisteus

Laboratorio de Aplicaciones
Telemáticas



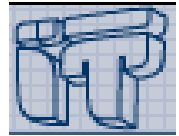
Índice

- Introducción a Swing
- Componentes
- Layouts
- Modelo de eventos
- Accesibilidad



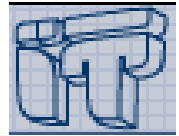
Aplicaciones gráficas

- En Java, hay varias alternativas:
 - AWT (Abstract Window Toolkit)
 - JFC Swing
 - SWT (Standard Widget Toolkit)
- El paradigma de programación de aplicaciones gráficas en Java es similar al de otros lenguajes y bibliotecas



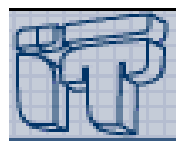
JFC

- Java Foundation Classes incluye paquete swing para crear interfaces gráficas
- Permite al programador elegir la apariencia (Pluggable Look and Feel) entre Java, windows, GTK+, etc.
- Incluye otros APIs adicionales:
 - accesibilidad,
 - 2D,
 - arrastrar y soltar entre aplicaciones
 - internacionalización



Paquetes JFC

- javax.accessibility
- javax.swing.plaf
- javax.swing.text.html
- **javax.swing**
- javax.swing.plaf.basic
- javax.swing.text.parser
- javax.swing.border
- javax.swing.plaf.metal
- javax.swing.text.rtf
- javax.swing.colorchooser
- javax.swing.plaf.multi
- javax.swing.tree
- **javax.swing.event**
- javax.swing.table
- javax.swing.undo
- javax.swing.filechooser
- javax.swing.text



Swing

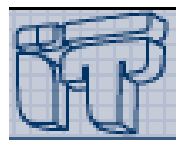
- Se basa en AWT, pero es:
 - Más potente, completo y elegante.
 - Más eficiente.
- Referencias básicas:

“Creating a GUI with JFC/Swing” (The Swing Tutorial)

<http://java.sun.com/docs/books/tutorial/uiswing/>

The Swing Connection

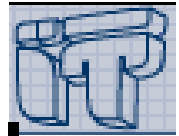
<http://java.sun.com/javase/technologies/desktop/articles.jsp>



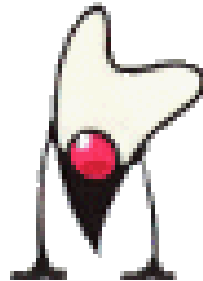
Componentes

- Cada elemento gráfico de GUI es un *componente*
- *Cada componente es una instancia de una clase*
- *Un componente se crea como cualquier otro objeto Java*
- *Algunos componentes pueden contener a otros componentes (son contenedores)*

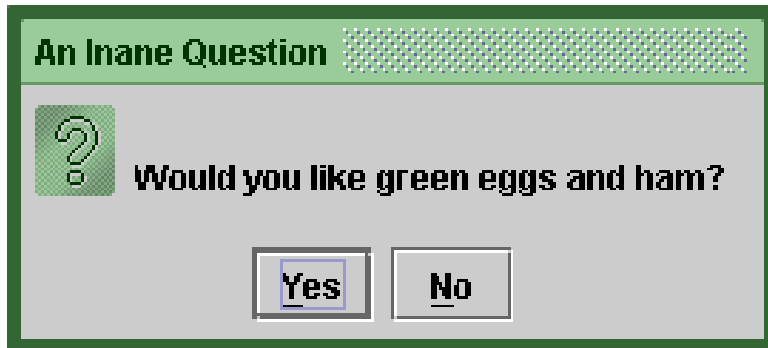
<http://java.sun.com/docs/books/tutorial/uiswing/components/index.html>



Contenedores de alto nivel



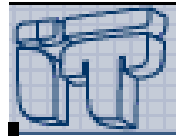
Applets



Diálogos

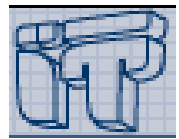


Marcos (ventanas)

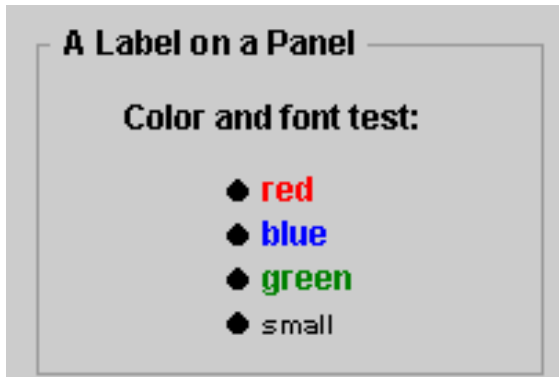


Contenedores de alto nivel

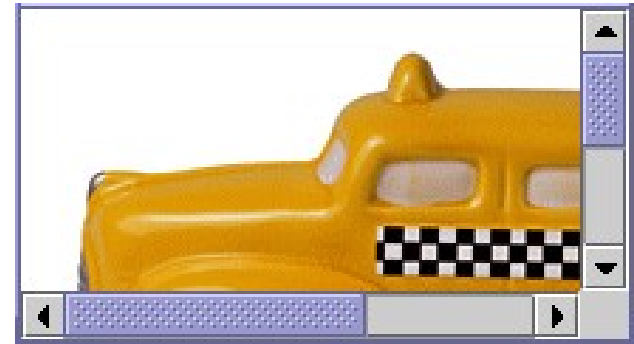
- Cada contenedor de alto nivel tiene un JrootPane que es la raíz de la jerarquía de contenedores.
- Todo componente GUI debe formar parte de la jerarquía de contenedores.
- Cada componente GUI sólo puede aparecer una vez.
- Un contenedor de alto nivel puede opcionalmente tener una barra de menús.



Contenedores intermedios



Panel



Panel deslizante

Panel dividido

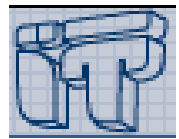


Panel con solapas

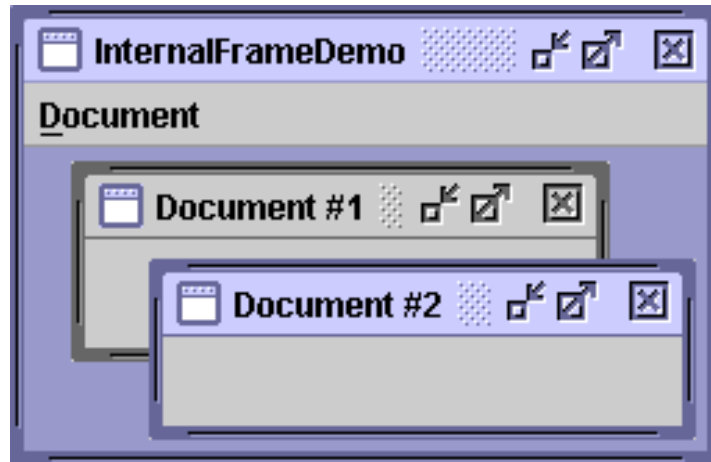


Barra de herramientas

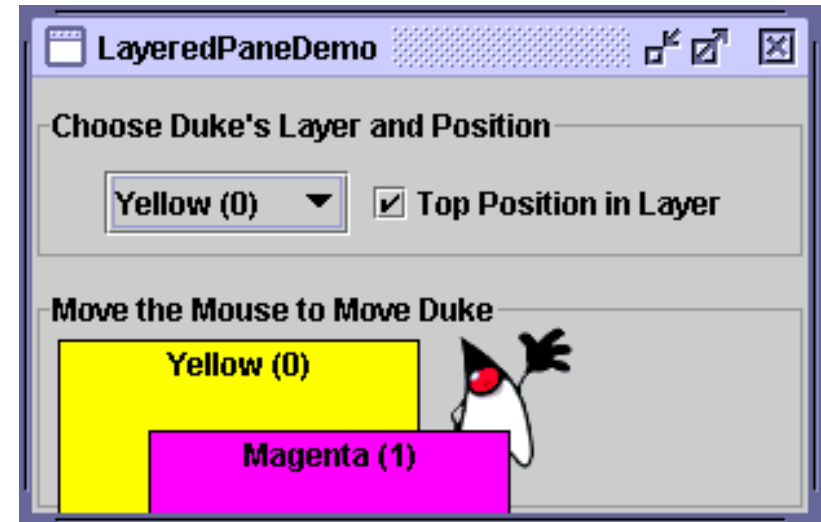




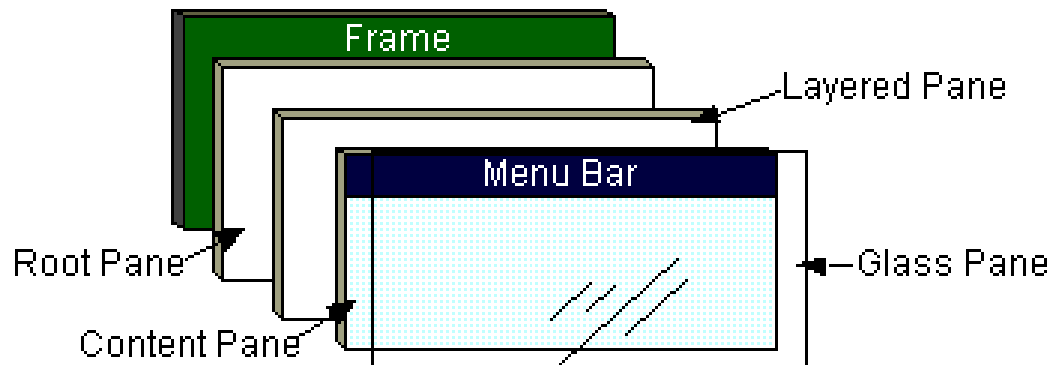
Contenedores específicos

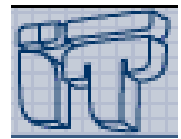


Panel interno



Panel de capas

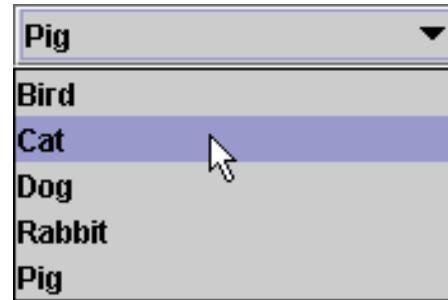




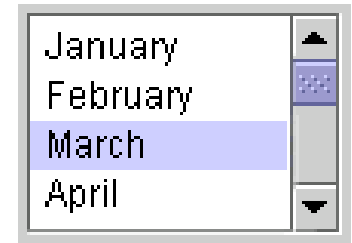
Controles básicos



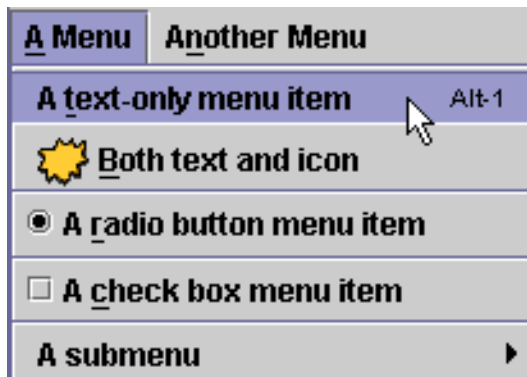
Botones



Cajas combo



Listas



Menús



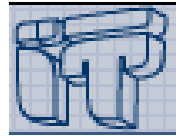
Controles deslizantes



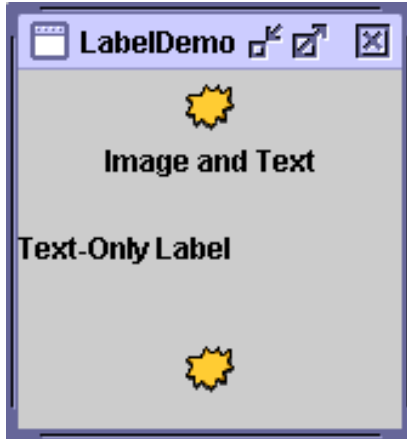
Controles numéricos



Campos de texto
(con/sin formato)



Displays no editables



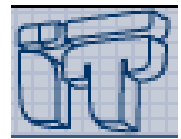
Etiquetas



Barras de progreso



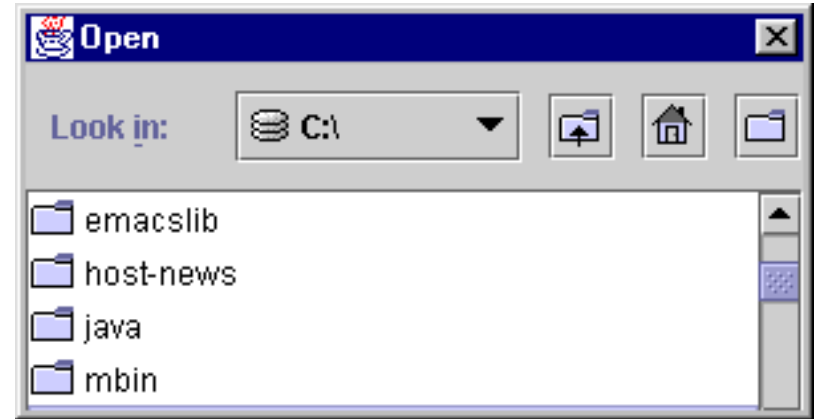
Pistas de herramientas
(tool tips)



Displays interactivos



Selector de colores



Selector de ficheros

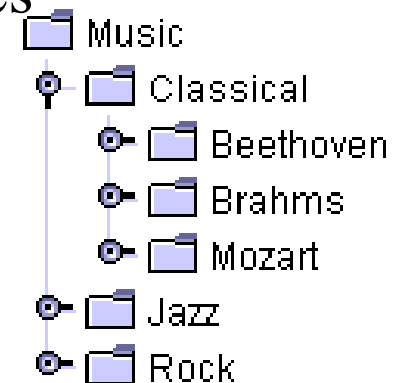
Tabla

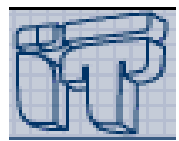
First Name	Last Name	Favorite Food
Jeff	Dinkins	
Ewan	Dinkins	
Amy	Fowler	
Hania	Gajewska	
David	Geary	

Texto



Árboles

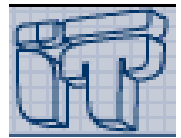




Más sobre componentes

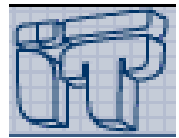
- Cada componente implementa un método *paint()* que contiene el código para *pintarlo*
- *Cuando el entorno necesita volver a pintar un componente, invoca a su método repaint()*

<http://java.sun.com/docs/books/tutorial/uiswing/components/index.html>



Pasos básicos

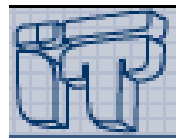
- Importar paquetes `javax.swing.XXX`
- Disponer un contenedor:
 - JFrame
 - JDialog
 - JApplet
- Agregar componentes al contenedor
- Mostrar el contenedor
- Los GUIs deben ser creados en el hilo de atención a eventos



Hola Mundo

```
import javax.swing.*;
public class HelloWorldSwing {
    public void createAndShowGUI() {
        JFrame.setDefaultLookAndFeelDecorated(true);
        JFrame frame = new JFrame("HelloWorldSwing");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel label = new JLabel("Hello World");
        frame.getContentPane().add(label);
        frame.pack();
        frame.setVisible(true);
    }
    public static void main(String[] args) {
        HelloWorldSwing hello = new HelloWorldSwing();
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() { hello.createAndShowGUI(); }
        });
    }
}
```

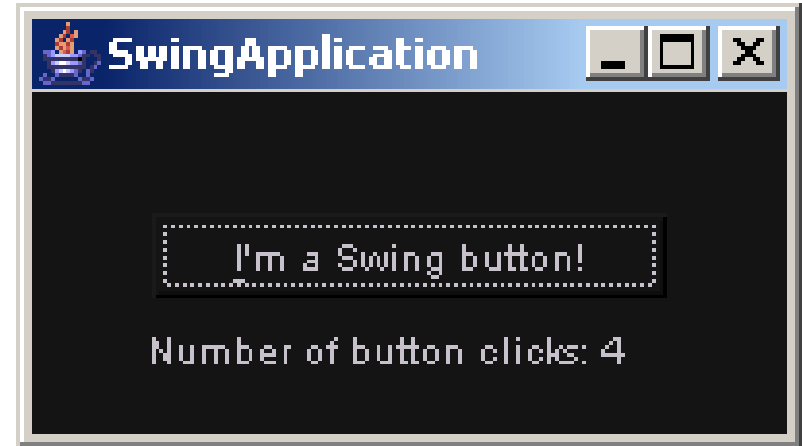




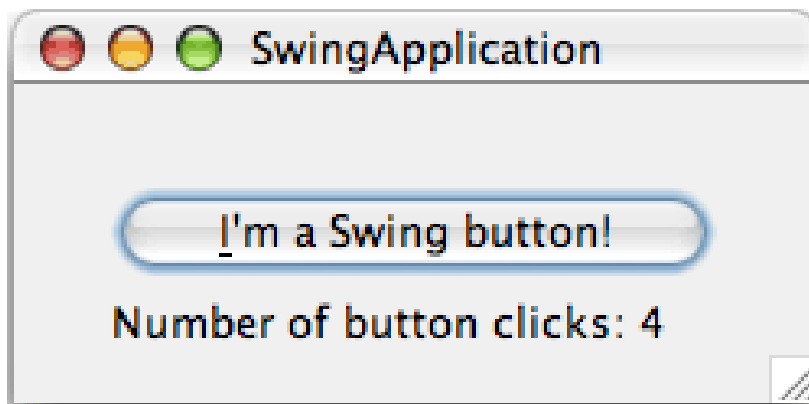
Look And Feel



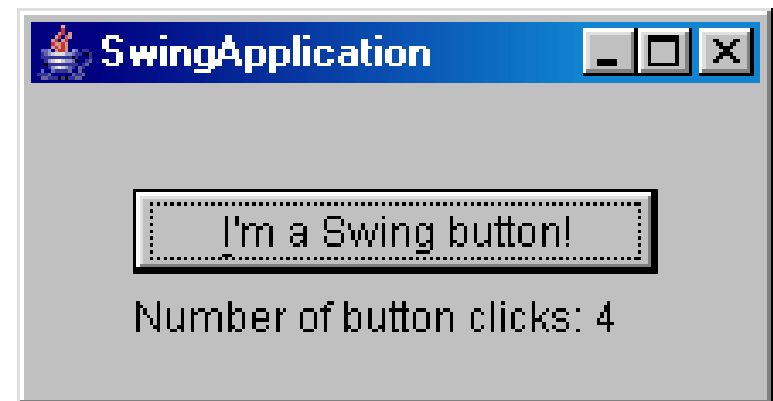
Java



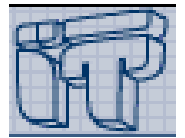
GTK+



MacOS

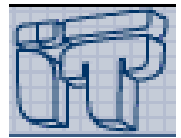


Microsoft Windows



Look And Feel: Ejemplo

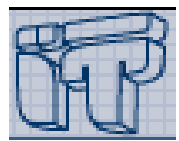
```
String lookAndFeel = null;  
// Pide el LAF de Java  
lookAndFeel =  
    UIManager.getCrossPlatformLookAndFeelClassName();  
...  
try {  
    UIManager.setLookAndFeel(lookAndFeel);  
} catch (Exception e) { }  
...// Crear y mostrar el GUI...
```



Contenedores y layouts

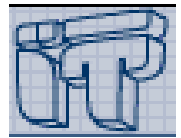
- A cada contenedor se le establece un *layout* asociado
- Un *layout* establece la disposición de los componentes dentro del contenedor
- Los componentes se añaden a un contenedor con el método *add*

<http://java.sun.com/docs/books/tutorial/uiswing/layout/index.html>



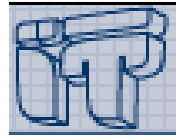
Contenedores: Ejemplo

```
JPanel panel = new JPanel(new GridLayout(0,1));  
panel.add(button);  
panel.getRootPane().addDefaultBotton(button);  
panel.add(label);  
panel.setBorder(BorderFactory.createEmptyBorder(  
    30, //arriba  
    30, //izquierda  
    10, //abajo  
    30) //derecha  
);
```



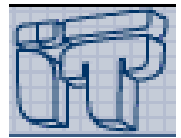
Modelo de eventos

- Cada componente puede generar *eventos*:
 - *Acciones del usuario sobre el componente*
 - *Temporizaciones*
 - *Cambio de estado, etc.*
- En cada componente se pueden registrar *escuchadores de eventos*
- *Cuando el componente genere un evento, invoca a todos sus manejadores de eventos*



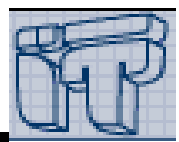
Eventos

- Cada evento es un objeto que hereda de la clase *AWTEvent*
- Ejemplos de eventos:
 - ActionEvent
 - MouseEvent
 - KeyEvent
 - WindowEvent
 - FocusEvent
 - Otros...



Escuchadores de eventos

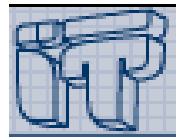
- Cada escuchador es un objeto que implementa la interfaz correspondiente al tipo de evento a escuchar:
 - ActionListener
 - WindowListener
 - MouseListener
 - KeyListener
 - FocusListener
 - Otros...



Etiquetas y botones: ejemplo

```
JButton button = new JButton("I'm a Swing button!");  
button.setMnemonic('i'); /*cuando se pulse Alt-i */  
button.addActionListener(/*...this o creamos una clase  
anónima o...*/);
```

```
private static String labelPrefix = "Number of button clicks: ";  
private int numClicks = 0;  
final JLabel label = new JLabel(labelPrefix + "0  ");  
label.setLabelFor(button);  
label.setText(labelPrefix + numClicks);
```



Manejar eventos

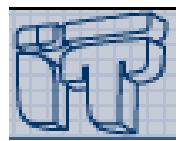
```
public class SwingApplication implements  
    ActionListener {
```

```
    JButton button = new JButton("I'm a Swing  
    button!"); button.addActionListener(this);
```

```
    public void actionPerformed(ActionEvent e) {  
        numClicks++;  
        label.setText(labelPrefix + numClicks);  
    }
```

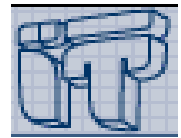


```
}
```



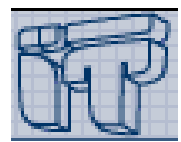
Clases adaptadoras

- Algunas interfaces de escuchadores poseen varios métodos (ej. *MouseListener*)
 - Implementar la interfaz obliga a implementar todos sus métodos: demasiado código si sólo interesa implementar uno de ellos
- Alternativa: la API proporciona implementaciones por defecto (adaptadores; ej. *MouseAdapter*)
 - Basta con heredar de la clase y reescribir el método deseado



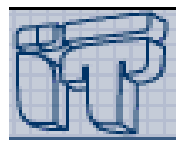
Accesibilidad

- La biblioteca `javax.accessibility` contiene clases e interfaces que facilitan adaptar un interfaz gráfica
 - Para poder navegar e interactuar con el teclado u otros dispositivos
 - Para reconocer el estado del interfaz gráfico mediante interfaces de voz: lectores de pantalla, amplificadores de pantalla, etc.
- En `j2se 1.5.0` hay 16 interfaces y 11 clases



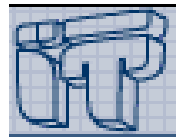
Interfaz Accesible

- Implementada por 105 clases gráficas de `java.awt` y `javax.swing` (`AbstractColorChooserPanel`, `Applet`, ..., `TextField`, `Window`)
- Ofrece un único método: `getAccessibleContext()`
 - Devuelve referencia a un `AccessibleContext` que nos permite otras formas de interactuar con el componente
 - Devuelve `null` si el objeto no es accesible.
 - Ojo! si hacemos una subclase de un componente gráfico:
 - o es accesible
 - o implementamos `getAccessibleContext()` que devuelva `null`



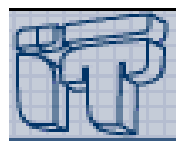
AccessibleContext

- Permite obtener referencias a objetos que implementan otras interfaces accesibles que a su vez ofrecen modos estándar de:
 - `AccessibleAction`: describir las acciones que puede ejecutar el objeto (y ejecutarlas)
 - `AccessibleComponent`: determinar y fijar la representación del objeto en pantalla
 - `AccessibleSelection`: determinar los hijos seleccionados del objeto y cambiar la selección
 - `AccessibleText`: acceder al contenido, atributos y localización del texto
 - `AccessibleValue`: determinar y fijar el valor numérico del objeto, así como obtener el máximo y mínimo valor



AccessibleRelation

- Clase que permite establecer relaciones entre objetos de un GUI
 - **Relaciones:** `CHILD_NODE_OF`, `CONTROLLED_BY`, `EMBEDDED_BY`, `FLOWS_TO`, `LABEL_FOR`, `MEMBER_OF`, `PARENT_WINDOW_OF`, `SUBWINDOW_OF`, ...
 - Las relaciones se deben poder obtener como strings en el locale correspondiente mediante la función `toDisplayString()` que la busca en el correspondiente `AccessibleBundle`.
 - Objeto: Facilitar la navegación y comprender la relación entre componentes



Otras clases

- **AccessibleState** permite obtener el estado de un componente (`BUSY`, `ENABLED`, `ICONIFIED`, ...)
- **AccessibleRole** permite determinar el papel de un componente (`ALERT`, `CANVAS`, `COMBO_BOX`, ...)
- **AccessibleHyperlink** para estandarizar el acceso a los enlaces, es decir conocer las acciones asociadas, su ancla (ej. `javax.swing.ImageIcon`), un objeto que permita ejecutar la acción (ej. `java.net.URL`)