DISEÑO ORIENTADO A OBJETOS

Prof: Esp. Ing. Agustín Fernandez

Introducción AWT

Introducción AWT

- AWT fue la primera solución propuesta por Java al problema frecuente de la programación clásica de como programar interfaces de usuario.
- Este problema no es trivial ya que implica tener que utilizar las API propias del Sistema Operativo y esto provoca que el código no sea transportable a otros sistemas.
- **AWT usa clases gráficas comunes a todos los sistemas operativos gráficos y luego la máquina virtual traduce esa clase a la forma que tenga en el sistema concreto en el que se ejecutó el programa,** sin importar que dicho sistema sea un sistema Linux, Windows, etc.

Introducción AWT

- Desde sus inicios (1995) el entorno Java ya contaba con AWT como biblioteca de componentes gráficos y también estaba concebida como una API estandarizada que permitía utilizar los componentes nativos de cada sistema operativo.
- En la práctica, lo anterior, hizo que esta tecnología no funcionase como se espera.
- Por ello en 1996 se toma como base de JFC (AWT, Swing y Java 2D) la biblioteca gráfica denominada Internet Foundation Classes (IFC), la cual fue desarrollada para Java originalmente por la extinta empresa Netscape Communications Corporation.

Introducción AWT

- Los componentes de IFC eran mostrados y controlados directamente por código Java independiente de la plataforma.
- Además al estar enteramente desarrollada en Java aumentaba su portabilidad lo cual aseguraba un comportamiento idéntico en diferentes plataformas.
- Entonces 1997, Sun Microsystems y Netscape Communications Corporation, entonces, anunciaron su intención de combinar IFC con otras tecnologías de las denominadas JFC (Java Foundation Classes) dando inicio al proyecto Swing.

Introducción AWT

- Concretamente los problemas de AWT fueron:
  - Problemas de compatibilidad en varios sistemas.
  - Falta de algunos componentes avanzados (árboles, tablas, etc).
  - Consumo excesivos recursos del sistema.
  - Swing aporta muchas más clases, consume menos recursos y construye mejor la apariencia de los programas.
  - AWT no desapareció; simplemente se añadió a las nuevas capacidades Swing.

Introducción AWT

- La construcción de interfaces gráficas de usuario en Java (en AWT y Swing) se descompone en tres partes:
  - **El contenedor (container):** es la ventana (o parte de la ventana) donde se situarán los componentes.
  - **Los componentes:** los menús, botones, áreas de texto, etc.
  - **El modelo de eventos:** cada acción producida por el usuario (con el ratón o el teclado), produce un evento que puede ser captado por Java, para permitir ejecutar un código en reacción a ese evento, si es necesario. **Por defecto, Java no capta ninguno evento sobre ningún objeto. Hay que registrar el evento sobre el objeto para que sea captado por Java.**

Introducción AWT

- **El contenedor:** es la ventana que va a contener todos los componentes de la aplicación:
  - Cuando se crea la ventana, se la debe de acompañar de la creación del evento de cierre de ventana. Si se le olvida, solo podrá cerrar la ventana matando al proceso Java. El evento es un escuchador de ventana, que está atento al clic sobre el botón de cierra de ventana.
  - La creación de un contenedor se hace desarrollando una nueva clase que se extienda de la clase Frame del AWT.
- Se necesita también importar el paquete java.awt.* y el
- paquete java.awt.event.*.

Introducción AWT

- **El contenedor (cont.):**
  - Por defecto, la ventana no es visible, así que hay que ponerla visible utilizando el método setVisible(true)
  - A la ventana, se le puede definir el título que aparecerá en la barra de título.
  - También se puede definir muchas propiedades cómo su tamaño por defecto, si está maximizada,etc.

Introducción AWT

- **El contenedor (cont.):**

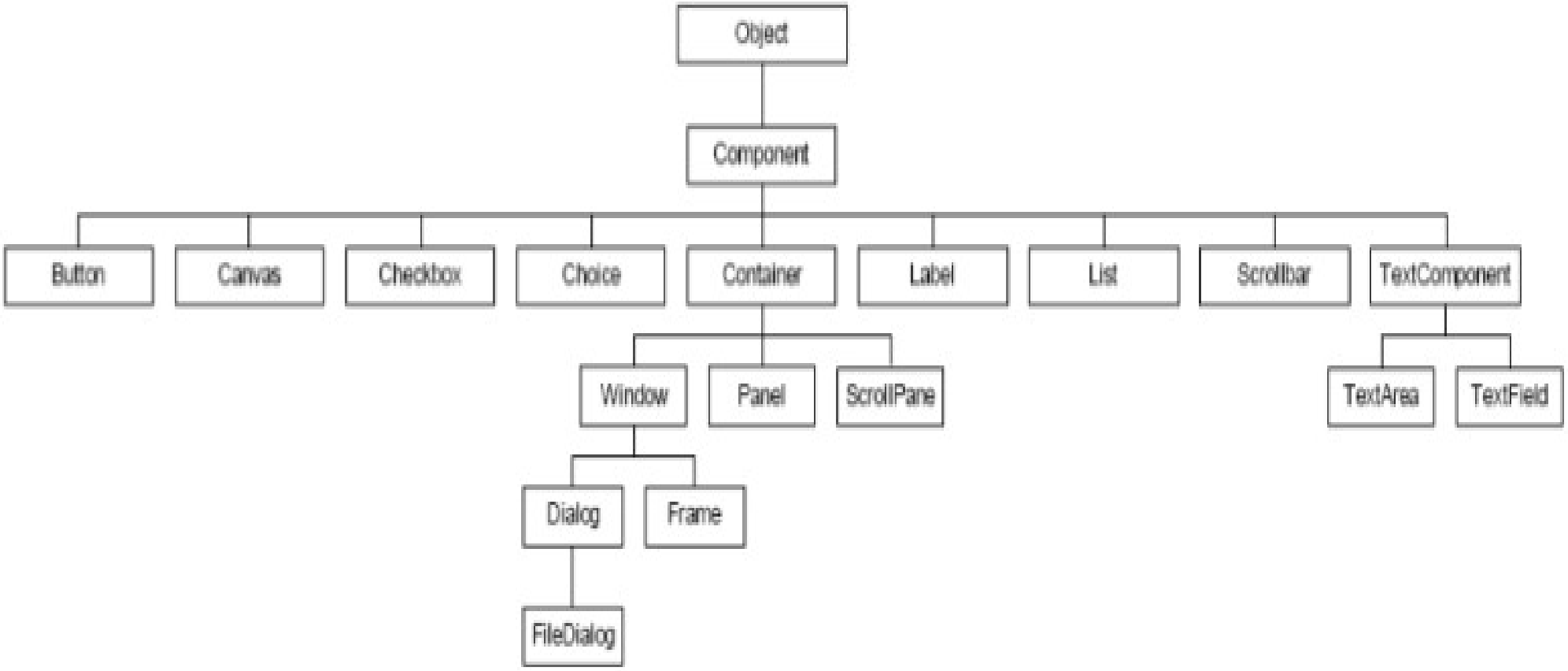
```
import java.awt.*;
import java.awt.event.*;

public class Ventana extends Frame {
    public Ventana() {
        try {
          this.addWindowListener(new WindowAdapter() { // Opens addWindowListener method
                public void windowClosing(WindowEvent e) { // Opens windowClosing method
                    System.exit(0);
                } // Closes windowClosing method
            }); // Closes addWindowListener method
            this.setTitle("Mi primera ventana en Java");
          //  this.setSize(800, 570); // ventana de 800x570 pixeles
          //  this.setExtendedState(this.MAXIMIZED_BOTH); // Maximizar la ventana
            this.setVisible(true);

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
    public static void main(String[] args) {
        new Ventana();
    }
}
```

Introducción AWT

- **Los componentes:**
  - Los componentes son los botones, etiquetas, áreas de textos, etc. que van a llenar la ventana, y permitirnos de enseñar la información y interaccionar con el usuario.
  - Todos los componentes tienen métodos comunes heredados de la clase Component, cómo por ejemplo el tamaño, el color, la fuente, etc.
  - Cada componente tiene también sus propios métodos para sus acciones especificas.
  - Los componentes se pueden incluir en la ventana principal, o en otro componente de tipo Container.

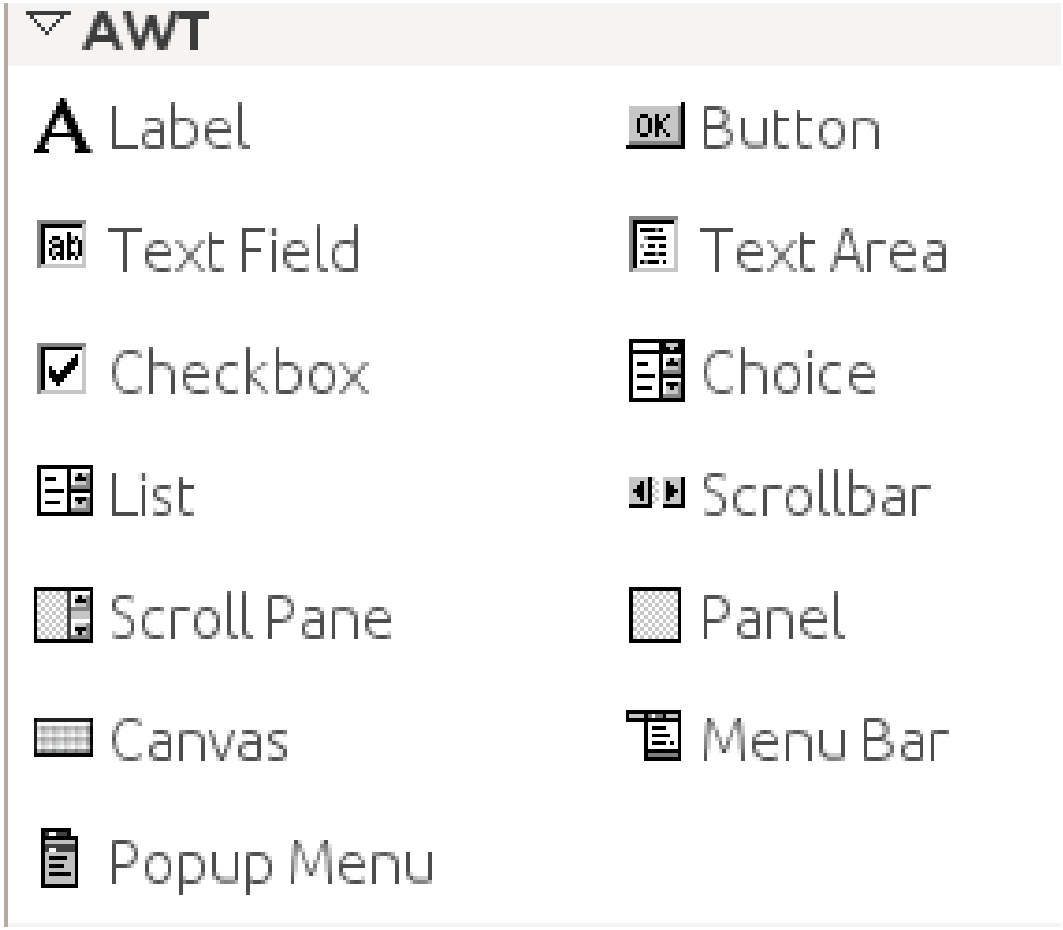
Introducción AWT

- **Los componentes (cont.):**



Introducción AWT

- **Los componentes (cont.):**



Introducción AWT

- **Los componentes (Cont.)**
  - **Etiquetas**: Los Labels son textos que no se pueden cambiar directamente por el usuario (sólo lectura).
  - **Textos**: Existen dos tipos de componentes de texto diferente. Los dos permiten al usuario de modificar su valor. El primero, llamado TextField, se usa para contener textos de una sola línea. El secundo, TextArea, se usa para textos de varias líneas.
  - **Botones**: Los Buttons se utilizan para empezar acciones.
  - **Casillas**: Los CheckBoxs se utilizan para marcar (o no) opciones. La casilla incluye un texto como una etiqueta. Cuando se utiliza un grupo de opciones (CheckBoxGroup), al contrario de los CheckBox, sólo una de las opciones propuestas se puede elegir.

Introducción AWT

- **Los componentes (Cont.)**
  - **Elecciones**: Las Choices permite al usuario elegir un valor entre varias dentro de una lista.
  - **Listas**: La List permite al usuario de elegir una o varias valores dentro de una lista.
  - **Canvas**: Crea un cuadro blanco en el cual se puede dibujar objetos gráficos.
  - **Scrollbar**: Permite elegir valores de un rango de valores moviendo el cursor del ScrollBar.
  - **Barra de menú**: La MenuBar permite crear una barra de menú con menús de varios niveles.

Introducción AWT

- **Los componentes (Cont.)**
  - **Menú contextual**: Permite sacar un menú de acciones desde un componente.
  - **Panel**: Es el contenedor más simple. Provee un espacio a donde se pueden poner otros componentes.
  - **Panel con barra de desplazamiento**: Es un contenedor parecido al Panel que tiene, además, ScrollBar para moverse horizontalmente y verticalmente en el en caso de que el tamaño del contenido sea mas grande que el del contenedor.

Introducción AWT

- **Los eventos:**
  - En Java, para poder interaccionar con el usuario se tiene que "escuchar" a las acciones del usuario.
  - Por eso, hay de declarar escuchadores sobre los objetos (componentes) por los cuales esperamos una acción.
  - Existen diferentes objetos que pueden ser origen de una acción.
  - Cada objeto acepta un cierto tipo de escuchador.
  - Sólo el objeto de tipo Component acepta varios escuchadores diferentes.

Introducción AWT

- **Los eventos (cont.):**

Eventos de bajo nivel	
Objeto	***Escuchador disponible***
Component	ComponentListener FocusListener KeyListener MouseListener MouseMotionListener
Container	ContainerListener
Window	WindowListener

Eventos de alto nivel	
Objeto	***Escuchador disponible***
Button List MenuItem TextField	ActionListener
Choice Checkbox Checkbox CheckboxMenuItem List	ItemListener
Scrollbar	AdjustmentListener
TextArea TextField	TextListener

Introducción AWT

- **Los eventos (cont.):**
  - Cada escuchador (Listener) tiene uno o varios métodos posibles.
  - Los escuchadores son interfaces o clase por los cuales los métodos no están implementadas, así que hay que implementarlas.

Interface	*Method(s)*
ActionListener	`actionPerformed(`ActionEvent `e)`
AdjustmentListener	`adjustmentValueChanged(`AdjustmentEvent `e)`
ComponentListener	`componentHidden(`ComponentEvent `e)`
	`componentMoved(`ComponentEvent `e)`
	`componentResized(`ComponentEvent `e)`
	`componentShown(`ComponentEvent `e)`
ContainerListen er	`componentAdded(`ContainerEvent `e)`
	`componentRemoved(`ContainerEvent `e)`
FocusListener	`focusGained(`FocusEvent `e)`
	`focusLost(`FocusEvent `e)`
ItemListener	`itemStateChanged(`ItemEvent `e)`
KeyListener	`keyPressed(`KeyEvent `e)`
	`keyReleased(`KeyEvent `e)`
	`keyTyped(`KeyEvent `e)`

Introducción AWT

- **Los eventos (cont.):**

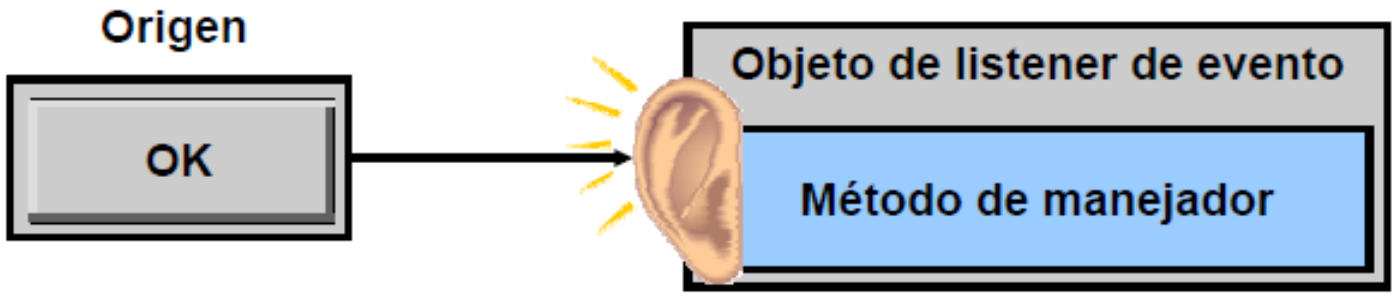
MouseListener	mouseClicked(MouseEvent e)
	mouseEntered(MouseEvent e)
	mouseExited(MouseEvent e)
	mousePressed(MouseEvent e)
	mouseReleased(MouseEvent e)
MouseMotionListener	mouseDragged(MouseEvent e)
	mouseMoved(MouseEvent e)
TextListener	textValueChanged(TextEvent e)
WindowListener	windowActivated(WindowEvent e)
	windowClosed(WindowEvent e)
	windowClosing(WindowEvent e)
	windowDeactivated(WindowEvent e)
	windowDeiconified(WindowEvent e)
	windowIconified(WindowEvent e)
	windowOpened(WindowEvent e)

Introducción AWT

- **Los eventos (cont.):**
  - El listener de evento se registra como "interesado" en un determinado tipo de evento. Por ejemplo, "Clics de botón".
  - Por ejemplo un botón puede generar eventos ActionEvent; por lo tanto, la clase Button proporciona un método denominado addActionListener(). En el ejemplo siguiente, se muestra cómo llamar a este método para registrar el objeto de listener. Este objeto MyListener se agrega a una lista de listeners a los que se informa cuando se hace clic en el botón. El objeto de listener contiene el código para manejar el evento.
  - **Nota:** Existe un método denominado removeActionListener(), que permite eliminar un listener de la lista de listeners registrados.

Introducción AWT

- **Los eventos (cont.):**




```
MyListener actionListenerObj = new MyListener();
public void jbInit() {
  button1.addActionListener(actionListenerObj);
  ...
}
```

Introducción AWT

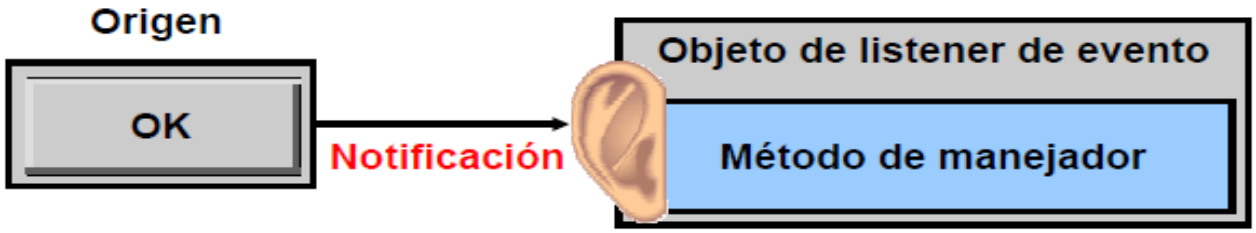
- **Los eventos (cont.):**
  - Las clases siguen las reglas de codificación siguientes:
    - La clase de objeto de evento se denomina XXXEvent.
    - El listener implementa una interfaz denominada XXXXListener.
    - El origen de evento proporciona un método addXXXXListener(), que acepta un argumento de objeto que implementa la interfaz XXXXListener adecuada.

Introducción AWT

- **Los eventos (cont.):**
  - La implementación del escuchador se puede hacer en uno o dos pasos.
  - **En el caso anterior la implementación fue en dos pasos:** el escuchador se podría reutilizar para otros componentes. Se implementa en una clase aparte.

Introducción AWT

- **Los eventos (cont.):**
  - **Implementación en un paso:** el método del escuchador se sobre-define directamente en la declaración del escuchador.




```
public void jbInit() {
  button1.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
    // Your code to handle the ActionEvent
   }
  }); … }
```

Introducción AWT

- **Los eventos (cont.):**
  - **Clases de Adaptador:** Algunas interfaces de listener de evento contienen varios métodos. Para cada una de estas interfaces, hay una clase de adaptador simple que proporciona un cuerpo vacío para cada método de las interfaces.
    - Por ejemplo, MouseListener contiene cinco métodos. Si se implementa directamente MouseListener, se debe implementar sus cinco métodos, aunque sólo esté interesado en uno de ellos.
    - También se puede ampliar la clase MouseAdapter. Si lo hace, se puede sustituir los métodos que se necesite e ignorar los demás.
    - No existe ninguna clase de adaptador para la interfaz ActionListener, porque esta interfaz sólo tiene un método.

Introducción AWT

- **Los eventos (cont.):**

Las clases de adaptador resultan útiles para implementar la interfaz de listener de evento:

- Proporcionan implementaciones de método vacías.
- Se amplían y se puede sustituir el método deseado.

```
interface MouseListener {
 // Declares five methods
}
```

```
class MouseAdapter implements MouseListener {
 // Empty implementations of all five methods
}
```

```
public class MyListener extends MouseAdapter {
 // Override only the methods you need
}
```