

Interfaces Gráficas



Resumen

- Interfaz de Usuario
- Interfaces gráficas
 - Conceptos generales
 - AWT
 - Swing
- Programación basada en eventos:
 - Conceptos generales
 - ActionEvent
 - Ratón
 - Teclado
 - ItemEvent
 - Otros
- Applets



Objetivos

- Conocer y comprender qué es una interfaz de usuario
- Conocer y comprender las características y diferencias de distintas interfaces gráficas, tomando como ejemplo AWT y Swing
- Conocer y comprender qué es un evento, qué es un oyente y cómo programar respuestas a eventos
- Conocer y comprender distintos tipos de eventos utilizándolos para programar respuestas a esos eventos
- Conocer y comprender las características generales y limitaciones de los applets
- Realizar aplicaciones con interfaz gráfica que sean capaces de responder ante distinto tipo de eventos



Interfaz de Usuario

- El concepto de "interfaz de usuario" - IU se refiere a los modos de interacción entre un programa y el usuario
- Java contiene clases para funcionalidades de IU como:
 - Componentes gráficas
 - Información de configuración (argumentos de línea de comandos en aplicaciones)
 - Lectura y escritura por teclado (System.in, System.out)
 - Ficheros
 - Sonido



Interfaces de Usuario Gráficas – AWT – Características

- Paquete java.awt.*
- Componentes:
 - Permiten al usuario interactuar con la aplicación
 - Instancias de la clase Component o subtipos
 - Ejemplos: Botones, barras de desplazamiento, etiquetas, listas, cajas de selección o campos de texto
- Contenedores:
 - Agrupan y organizan componentes
 - Son en sí mismos componentes → pueden agruparse en otros contenedores
 - Instancias de la clase Container o subtipos
 - Distinto tipos de contenedores



Interfaces de Usuario Gráficas – AWT – Tipos de contenedores

- Window:
 - Es una ventana
 - No puede enlazarse o estar dentro de otro contenedor
 - No tiene ni título ni borde
- Frame:
 - Ventana con título y borde
 - Puede tener una barra de menú
- Dialog:
 - Ventana con título y borde
 - Asociada a otra ventana (toma de datos del usuario, mensaje de error, etc)
- Panel:
 - Contenedor genérico de componentes
 - Subclase: Applet



Interfaces de Usuario Gráficas – AWT – Mi primera ventana

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

```
public class MiPrimeraVentana extends Frame
{
    MiPrimeraVentana()
    {
        setSize(300,200);
        setTitle("Mi primera ventana");
    }
    public static void main(String[] args)
    {
        MiPrimeraVentana v = new MiPrimeraVentana();
        v.show();
    }
}
```



- Aunque se cierre la ventana, la aplicación no termina

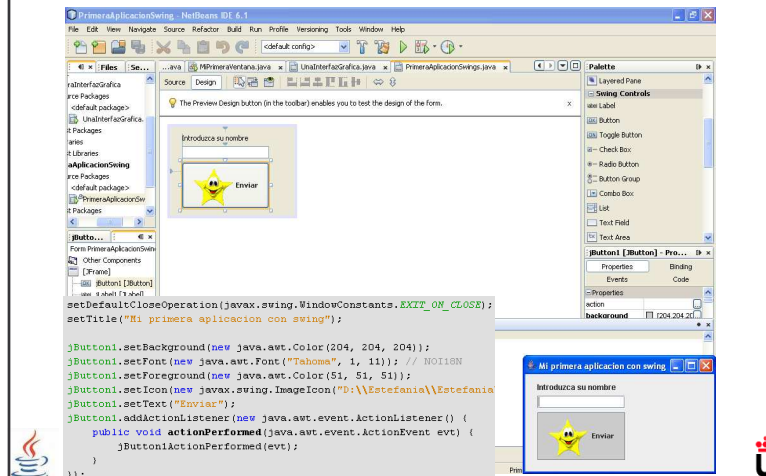


Interfaces de Usuario Gráficas – Swing – Características

- Paquetes javax.swing y javax.swing.event (opcional)
- El nombre de sus componentes empieza por "J"
- Mejoras en la interfaz gráfica y en funcionalidades:
 - Botones y etiquetas pueden mostrar imágenes y/o texto
 - Añadir/modificar fácilmente los bordes de los componentes
 - Modificar comportamiento o apariencia de un componente
 - No tienen porque ser rectangulares (p.e. botones redondos)
- Especificación del aspecto y comportamiento de la IU de nuestro programa (pueden utilizar aspecto Windows)
- API de Accesibilidad → Uso de tecnologías asistivas
- Algunas de estas características se realizaron sin código nativo, tratando sólo con el API de la JDK 1.1



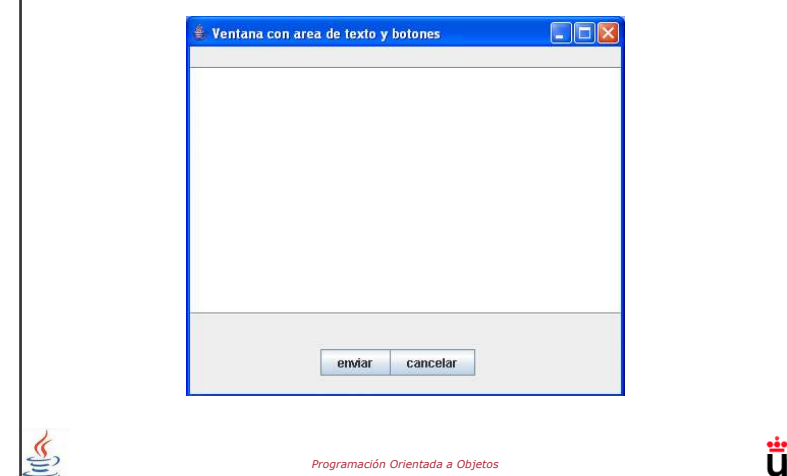
Interfaces de Usuario Gráficas – Swing – Mi primera ventana



```
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("Mi primera aplicacion con swing");

jButton1.setBackground(new java.awt.Color(204, 204, 204));
jButton1.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jButton1.setForeground(new java.awt.Color(51, 51, 51));
jButton1.setIcon(new javax.swing.ImageIcon("D:\\Estefania\\Estefania\\
jButton1.setText("Enviar");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});
```

Interfaces de Usuario Gráficas – Swing – Mi segunda ventana



Programación Orientada a Objetos

Interfaces de Usuario Gráficas – AWT vs. Swing

- AWT:
 - Más antigua
 - Más funcionalidades
- Swing:
 - Más reciente
 - Mejor estética
 - No está soportado directamente por los navegadores Web (paquete JRE en el cliente)
 - Generación de código automática en IDEs como Netbeans (conservación de ficheros *.java y *.form)

Programación Basada en Eventos

Programación basada en eventos – Conceptos generales

- Al interactuar con componentes gráficos se producen eventos a través del ratón y del teclado
- Paquete: `java.awt.event`
 - Interfaces y clases para gestionar los eventos que lanzan las componentes gráficas
- `java.awt.AWTEvent`:
 - Objeto de tipo evento



Programación basada en eventos – Conceptos generales

- Programación de respuestas a los eventos entregando objetos de tipo evento a objetos registrados como oyentes:
 - Fuente del evento: una componente gráfica
 - Un oyente – **listener** es un objeto de alguna clase que actúa de oyente y cuando se produce un evento ejecuta una respuesta
- Pasos:
 - Declaración de clases listeners:
 - `<clase-oyente> implements <tipo-oyente>`
 - Registrar un objeto oyente para un objeto fuente:
 - `<una-componente-fuente>.add<tipo-oyente> (<un-objeto-de-clase-oyente>)`



Programación basada en eventos - *ActionEvent*

- `java.awt.event.ActionEvent` en:
 - "Click" en un `JButton`
 - Presionar "enter" en un `JTextField`
 - ...

```
public class Oyente implements ActionListener
{
    ...
    public void actionPerformed(ActionEvent e)
    { /* Código a ejecutar ante el evento, RESPUESTA*/ }
    ...
}
```
- Oyentes:
 - Interfaz `java.awt.event.ActionListener`:

```
public void actionPerformed (ActionEvent e)
```
 - `<boton>.addActionListener(<instancia-oyente>)`



Programación basada en eventos – *ActionEvent* – Ejemplo

- La clase tiene que implementar una interfaz para escuchar eventos (p.e. *implements ActionListener*)
- Se tiene que añadir un "Listener" al botón para escuchar los eventos (p.e. *b.addActionListener(this);*)
- Implementar los métodos a los que se llama cuando ocurre un evento

```
public void actionPerformed(ActionEvent e)
{
    nclicks++;
    jLabel1.setText("Nº clicks: " + nclicks);
}
```



Ejercicio 1 – Programación basada en eventos

- Tenemos tres componentes dentro de un *JFrame*: un *JLabel*, y dos botones (*JButton*)
- En la etiqueta se irán mostrando el número de veces que el usuario pulsa el botón 'Suma click'. Existe otro botón que sí recoge el evento que se produce pero no actualiza el texto de la etiqueta
- Especificar qué es lo que habría que tener en cuenta sobre gestión de eventos en cada uno de los botones



Programación Orientada a Objetos



Programación basada en eventos - Ratón

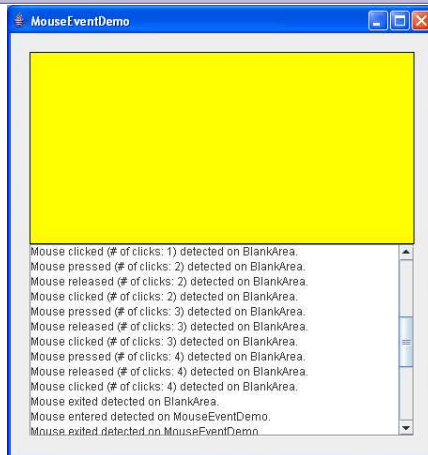
- Se genera un *java.awt.event.MouseEvent* al interactuar con el ratón en un *java.awt.Container*:
- Interfaz *MouseListener* con los métodos:
 - `public void mouse {Clicked | Pressed | Released | Entered | Exited} (MouseEvent e) {}`
- Pasos
 - ... implements *MouseListener* ...
 - Añadir a un componente el oyente → *addMouseListener*
 - Implementar los métodos de la interfaz correspondientes



Programación Orientada a Objetos



Programación basada en eventos – Ratón - Ejemplo



Programación Orientada a Objetos



Programación basada en eventos – Ratón - Ejemplo

```
public void mouseEntered(MouseEvent e)
{ showEvent("Mouse entered", e);}

public void mouseExited(MouseEvent e)
{ showEvent("Mouse exited", e);}

public void mousePressed(MouseEvent e)
{ showEvent("Mouse pressed (# of clicks: "+ e.getClickCount()
+ ")", e);}

public void mouseReleased(MouseEvent e)
{ showEvent("Mouse released (# of clicks: "+ e.getClickCount()
+ ")", e);}

public void mouseClicked(MouseEvent e)
{ showEvent("Mouse clicked (# of clicks: "+ e.getClickCount()
+ ")", e);}
```



Programación Orientada a Objetos



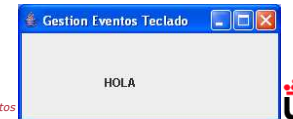
Programación basada en eventos - Teclado

- *KeyEvent* cuando se pulsa o se suelta una tecla
- *KeyListener* es el oyente que escucha eventos de teclado
- Métodos:
 - `keyPressed(KeyEvent e)`: Se presiona una tecla
 - `keyReleased(KeyEvent e)`: Se deja de presionar la tecla
 - `keyTyped(KeyEvent e)`: Ocurre antes de soltar la tecla
- Pasos:
 - ... implements `KeyListener` ...
 - Añadir el oyente → `addKeyListener`
 - Implementar los métodos de la interfaz



Programación basada en eventos - Teclado - Ejemplo

- Ventana que ponga los caracteres que teclea el usuario en una etiqueta que está dentro de la ventana
- Excepciones:
 - Si se pulsa la tecla de suprimir (127), se debe borrar toda la cadena escrita
 - Si se pulsa la tecla `BACKSPACE` - borrar un carácter, se debe borrar el último carácter introducido
- Cosas útiles:
 - `public char evento.getKeyChar()` → Carácter leído
 - `String cadena.substring(origen,fin)` → Subcadena de 'cadena'



Programación basada en eventos - Teclado - Ejemplo

- En la ventana añadir el oyente:

```
...
addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent e){
        formKeyPressed(e);
    }
});
....
```



Programación basada en eventos - Teclado - Ejemplo

```
private void formKeyPressed(java.awt.event.KeyEvent e) {
    char tecla = e.getKeyChar();
    String txt = etq.getText();
    switch(tecla)
    {
        case 127: // Suprimir
            etq.setText("");
            break;

        case 8: // Borrar - Backspace
            if (txt.length() > 0)
                etq.setText(txt.substring(0, txt.length() -1));
            break;

        default: etq.setText(txt+ tecla);
    }
}
```



Programación basada en eventos – ItemEvent

- Se produce en las casillas de verificación y selección: JRadioButton, JCheckBox, JComboBox,...
- `public ItemEvent(ItemSelectable source, int id, Object item, int stateChange)`
- `stateChange` indica si el elemento está seleccionado:
 - `ItemEvent.SELECTED`
 - `ItemEvent.DESELECTED`
- Interfaz `ItemListener`:
 - `public void itemStateChanged (ItemEvent evt)`



Ejercicio 2 - Programación basada en eventos – ItemEvent

- Realizar una aplicación que contenga una etiqueta, un JComboBox y dos JCheckBox
- Si el usuario cambia su selección tanto del JComboBox como de los dos JCheckBox, el texto que está en la etiqueta se debe actualizar apropiadamente



- Necesitas saber:
 - Selección de una casilla:
`checkBox.isSelected()`
 - Elemento seleccionado en un combo:
 - `JComboBox.getSelectedItem()`



Ejercicio 2 - Programación basada en eventos – ItemEvent

- Necesitas saber:
 - Actualización del estilo de la letra:
`etq.setFont(new Font("Tahoma", estilo, 15))`
 - Estilo: `Font.BOLD`, `Font.ITALIC`, `Font.PLAIN`, `Font.BOLD | Font.ITALIC`, ...
 - Color de la letra:
`etq.setForeground(new Color(Rojo,Verde,Azul));`



Programación basada en eventos – Más eventos

- Otros eventos:
 - Generados por `java.awt.Container`:
 - `addFocusListener (java.awt.event.FocusEvent)`
 - `addMouseMotionListener (java.awt.event.MouseEvent)` para eventos del movimiento de ratón como arrastrar y soltar
 - Generados por `JFrame` y por `JDialog`
 - `addWindowListener(java.awt.event.WindowEvent)`
- Hay más tipos de eventos en las interfaces gráficas:
 - Eventos de una ventana (`WindowEvent`) como abrir, cerrar o minimizar
 - ...



Applets



Applets – Características generales

- Un applet es una clase compilada que se envía al cliente incrustado en una página HTML
- Los navegadores llevan una máquina virtual de Java (JVM), que es la que se encarga de cargar el applet de una página web
- Permiten que el cliente pueda interactuar con la aplicación
- Simuladores de applets:
 - Desde línea de comandos → "appletviewer ejemplo.html"
 - ¿Dónde? → En la Web de sun



Applets - Limitaciones

- Las aplicaciones web que contienen applets son lentas
- No pueden leer o escribir en ficheros
- Conexiones imposibles a otra máquina distinta de la que proviene
- No puede arrancar ningún programa en el ordenador donde se está ejecutando, ni tampoco leer sus propiedades



Applets - Implementación

- Una applet debe heredar de la clase applet (java.applet)
- La clase que define un applet debe importar el paquete
import java.applet.*;
- No necesita el método main
- La salida no se realiza con el método System.out.print()
- Para permitir que un applet tenga una interfaz gráfica se debe importar el paquete:
import java.awt.*;
- El applet es una clase que:
 - Implementa los métodos: init, start, paint, ...
 - Se obtiene el fichero xx.class
 - Se incluye en una página HTML con la etiqueta <APPLET>



Applets - Métodos

- Init: Inicializa el applet cada vez que se carga la página
 - Start: Arranca la ejecución del applet (inicialización o recarga de una página)
 - Stop: Detiene la ejecución al salir de la página
 - Destroy: Se ejecuta al cerrar el navegador
-
- Paint: Se llama cuando el applet es modificado
 - Update: Primero genera color por defecto en background y después llama a paint
 - Repaint: Llama al método update



Applets - Parámetros <applet>

- CODE: Nombre de la clase que contiene el applet
- WIDTH: Ancho del applet en pixeles
- HEIGHT: Altura del applet en pixeles
- CODEBASE: Directorio-URL desde donde se carga la clase del applet
- ARCHIVE: Lista de ficheros JAR que el applet utiliza
- ALT: Texto que se mostrará si no se puede cargar



Applets - Mi primer applet

```
import java.applet.Applet;
import java.awt.Graphics;
```

```
public class AppletHolaMundo extends Applet
{
    public void paint(Graphics g)
    {
        g.drawString("Hola Estefania!", 50, 25);
    }
}
```



```
<HTML>
<HEAD>
<TITLE> Hola Mundo</TITLE>
</HEAD>
<BODY>
    Esta es la salida del applet:
    <APPLET CODE="AppletHolaMundo.class" WIDTH=150 HEIGHT=25>
</APPLET>
</BODY>
</HTML>
```



Applets - Ejemplo métodos

```
import java.applet.Applet;
import java.awt.Graphics;
```

```
public class AppletEventos extends Applet
{
    StringBuffer buffer;
    public void paint(Graphics g)
    {
        g.drawRect(0, 0, size().width - 1,
        size().height - 1);
        g.drawString(buffer.toString(), 5, 15);
    }

    public void start() { addItem("Arrancando... "); }
    public void stop() { addItem("Parando... "); }
    public void destroy() { addItem("Cerrando..."); }
    void addItem(String newWord)
    {
        System.out.println(newWord);
        buffer.append(newWord);
        repaint();
    }
}
```



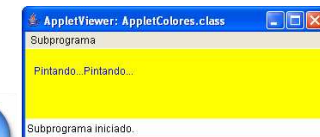
Applets - Algunos métodos gráficos

- `public void drawString(String mensaje, int x, int y)`
 - Permite mostrar cadenas de caracteres en un applet
 - Pertenece a `java.awt.Graphics`
 - Argumentos:
 - `mensaje`: Cadena a mostrar
 - `x,y`: Coordenadas de la posición donde se mostrará
- `public void setBackground(Color c)/public Color getBackground()`
 - Asigna/devuelve color de fondo de la ventana del applet
 - `c`: Objeto de tipo `Color` que define los colores mediante constantes de la clase (`Color.black`, `Color.blue`, ...)
- `public void setForeground(Color nuevocolor) /public Color getForeground()`
 - Asigna/devuelve el color del primer plano



Ejercicio 3 – Applets

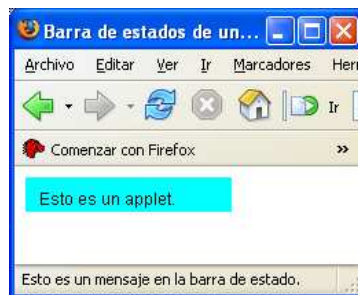
- Realizar un programa que cree un applet teniendo en cuenta que:
 - El color de fondo es amarillo
 - El color que se utilizará para la letra es el azul
 - Cada vez que se cambie el tamaño de la pantalla, se deberá mostrar en el applet el texto "Pintando..."



Applets – Barra de Estado

- Para mostrar un mensaje en la barra de estado de la ventana del navegador se puede utilizar el método

```
public void showStatus (String s)
```



Applets – Barra de Estado - Ejemplo

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Graphics;

public class AppletBarraestados extends Applet {
    public void init() {
        setBackground(Color.cyan);
    }

    public void paint(Graphics g) {
        g.drawString("Esto es un applet", 10, 20);
        showStatus("Esto es un mensaje en la barra de estado");
    }
}
```



Applets – Banner - Ejemplo

- Hacer un applet que sea capaz de mostrar un texto desplazándose horizontalmente - banner



Applets – Banner - Ejemplo

- Necesitas saber:
 - String.charAt(int x): Devuelve el carácter de la posición x de un String
 - String.substring(int x, int y): Devuelve un String con y caracteres cogidos a partir de la posición x del String que llama al método
 - Para producir un retardo en un bucle:
 - Thread.sleep(250)
 - Para poder utilizar la clase Thread la clase debe implementar la interface Runnable.



Applets – Banner - Ejemplo

```
import java.applet.*;
import java.awt.*;
public class AppletBanner extends Applet implements Runnable{
    String msg=" - un cartel en movimiento - ";
    Thread t=null;
    public void destroy(){ if(t !=null){ t.stop(); t=null;}}
    public void init(){
        setBackground(Color.orange);
        setForeground(Color.blue);
        t=new Thread(this);
        t.start();
        t.suspend();
    }
    public void paint (Graphics g){ g.drawString(msg,50,30);}
    ...
```



Applets – Banner - Ejemplo

```
...
public void run(){
    char ch;
    for(int i=1;i<200;i++){
        try{
            repaint();
            t.sleep(250);
            ch=msg.charAt(0);
            msg=msg.substring(1,msg.length());
            msg+=ch;
        }catch(InterruptedException e){}
    }
}
public void start(){ t.resume(); }
public void stop(){ t.suspend();}
}
```

