


Desarrollo de aplicaciones gráficas con swing

Interfaces gráficas

- Se emplean para interactuar de forma amigable con el usuario a través de la capa de presentación
 - Se componen de ventanas, cuadros de diálogo y controles gráficos.
 - La programación de estas aplicaciones es orientada a eventos
- 

Librerías gráficas


➤ La edición Java SE dispone de dos librerías de clases para la construcción de interfaces gráficas:

- AWT. Se trata de la primera librería aparecida en Java para la creación de interfaces gráficas. Existe desde las primeras versiones de Java, y es bastante limitada en cuanto a variedad y capacidades de componentes gráficos

- Swing. Es una extensión de AWT, sus clases heredan a las de AWT, proporcionando mayores capacidades y también más variedad de elementos gráficos.

-

Contenedores y controles

- Los elementos de una interfaz gráfica se dividen en contenedores y controles
 - Contenedores: Su misión es agrupar en su interior otro grupo de elementos gráficos. Ejemplo: Ventana, cuadro de diálogo, panel,...
 - Controles. Permiten la interacción directa con el usuario. Ejemplo: Botones, cajas de texto, listas,...
- 

Creación de una ventana swing

- La clase principal swing para la construcción de ventanas es JFrame.
- Para crear una ventana, extenderemos esta clase y definiremos en el constructor las operaciones para su configuración.

```
public class MyWindow extends JFrame
{
    public MyWindow(){
        //configuración ventana
    }
}
```

Configuración de una ventana

➤ Dentro del constructor, se deberán realizar las siguientes tareas.

- Establecimiento del título de la ventana
- Definir tamaño y posición
- Visualizar la ventana

Botón de cierre de la ventana

- Por defecto, el botón de cierre de la ventana no cierra ésta, sino que la oculta
- Se puede modificar el comportamiento a través del método *setDefaultCloseOperation()*, heredado de JFrame:
 - DO_NOTHING_ON_CLOSE. Al pulsar el botón de cierre no ocurrirá nada.
 - HIDE_ON_CLOSE. La pulsación del botón de cierre provoca que la ventana se oculte. Es el comportamiento por defecto.
 - DISPOSE_ON_CLOSE. Provoca que la ventana se cierre al pulsar el botón de cierre.
 - EXIT_ON_CLOSE . La pulsación del botón de cierre provocará que la aplicación finalice.

Controles gráficos

➤ La librería swing proporciona un amplio conjunto de controles gráficos a través de las clases del paquete javax.swing:

- JButton. Botón de pulsación
- JLabel. Etiqueta
- JTextField. Caja de texto de una línea
- JTextArea. Caja de texto multilínea
- JList y JComboBox. Listas de selección
- JTable. Tabla bidimensional

Proceso de creación de controles


➤ La creación de controles dentro de una interfaz gráfica requiere la realización de las siguientes tareas, normalmente programadas en el constructor de la propia clase ventana:

- Creación del objeto control
- Establecimiento de propiedades del control
- Añadir control al contenedor

➤ Eliminar gestor de organización:

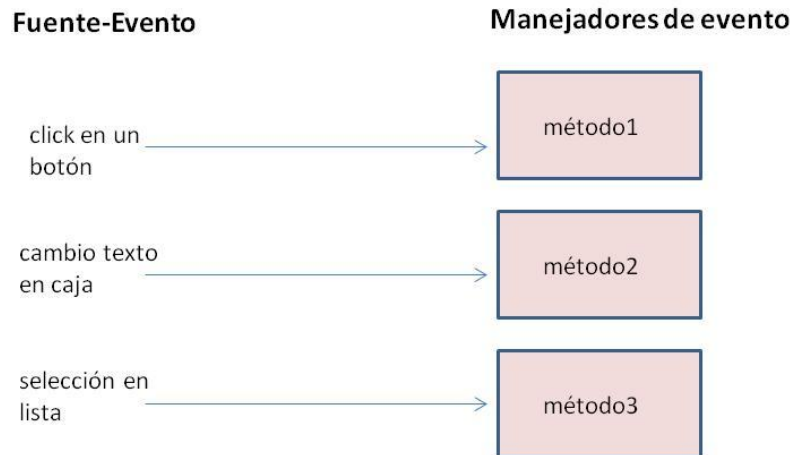
```
this.setLayout(null);
```

Eventos


- Las aplicaciones basadas en interfaz gráfica son aplicaciones conducidas por eventos
 - El programa debe ser capaz de responder a sucesos que puedan producirse sobre los componentes de la interfaz gráfica
 - Para ello, se deberán definir manejadores de evento, que son métodos que se asocian a los eventos de los componentes, ejecutándose cuando estos se producen
- 

Manejadores de evento

- Al objeto gráfico donde puede producirse un evento que queremos capturar se le conoce como objeto fuente de evento:
- Los manejadores de evento son métodos asociados a un evento y objeto fuente concretos.



Interfaces de escucha

- Interfaces Java que contienen la definición de los métodos manejadores.
 - Cada interfaz contiene los métodos para la gestión de un determinado tipo de eventos
 - Se encuentran distribuidas entre los paquetes `java.awt.event` y `javax.swing.event`
 - Todas estas interfaces son fácilmente reconocibles porque su nombre termina en Listener: `MouseListener`, `ActionListener`, `WindowListener`, `FocusListener`, etc.
- 


Proceso de gestión de eventos

- Para responder a los eventos de la interfaz gráfica dentro de una aplicación Java se deben realizar dos tareas:
 - Implementar la clase manejadora del evento
 - Asociar objeto manejador (instancia de la clase anterior) a objeto fuente del evento
- Estas operaciones se realizan, habitualmente, desde el constructor de la ventana o un método específico de inicialización


Cuadros de diálogo

- La clase `JOptionPane` proporciona métodos estáticos para la generación de cuadros de diálogo:
 - `int showConfirmDialog(Component parentComponent, Object message, String title, int optionType)`
 - `String showInputDialog(Component parentComponent, Object message)`
 - `void showMessageDialog(Component parentComponent, Object message)`
- Existen unas constantes en la clase para definir el tipo de opción y comparar con el valor devuelto

Listas

- Se generan con los componentes JList (lista abierta) y JComboBox (lista desplegable).
 - Los datos se obtienen mediante implementaciones de ListModel y ComboBoxModel
 - Existen clases que proporcionan implementaciones por defecto. Son clases de colección
 - Para personalizar la obtención de datos, se heredan estas clases y se sobrescriben los métodos que interesen.
- 

Tablas

- Se genera con la clase `JTable`
 - Los datos se obtienen mediante implementaciones de `TableModel`
 - La clase `DefaultTableModel` proporciona una implementación por defecto
 - Para personalizar la obtención de datos, se hereda esta clase y se sobrescriben los métodos que interesen
- 

Menús

➤ Se gestionan mediante las siguientes clases:

- **JMenuBar.** Representa la barra de menú
- **JMenu.** Representa un submenú
- **JMenuItem.** Representa una opción de menú

```
JMenuBar barra=new JMenuBar();  
JMenu menu=new JMenu("Inicio");  
barra.add(menu);  
JMenuItem it1=new JMenuItem ("Guardar");  
JMenuItem it2=new JMenuItem ("Salir");  
menu.add(it1);  
menu.add(it2);  
//añade barra a la ventana  
this.setJMenuBar(barra);
```

Eventos en menús

➤ Los objetos JMenuItem responden al evento(ActionEvent):

```
JMenuBar barra=new JMenuBar();  
JMenu menu=new JMenu("Inicio");  
barra.add(menu);  
JMenuItem it1=new JMenuItem ("Guardar");  
JMenuItem it2=new JMenuItem ("Salir");  
it1.addActionListener(new ActionListener...);  
it2.addActionListener(new ActionListener...);  
:
```

Menú contextual

- Se crea mediante la clase JPopupMenu
- Cada elemento de un pop up menú es un JMenuItem
- A través del método show(), se indica donde se visualizará el menú. Esta operación se realizará en el mousePressed de algún objeto:

```
JPopupMenu pop=new JPopupMenu();
JMenuItem it1=new JMenuItem ("Guardar");
JMenuItem it2=new JMenuItem ("Salir");
pop.add(it1);
pop.add(it2);
:
public void mouseReleased(MouseEvent e) {
    if (e.isPopupTrigger()) {
        pop.show(e.getComponent(),
                e.getX(), e.getY());
    }
}
```