Capítulo 1

Introducción

Este capítulo pretende dar una visión general de las interfaces gráficas de usuario. Para ello, entre otras cosas, describe de forma breve qué se entiende por *interfaz de usuario* e *interfaz gráfica de usuario*. Estudia los conceptos generales sobre elementos gráficos y eventos, e indica los pasos básicos a seguir en la construcción de una aplicación con interfaz gráfica.

1.1. Interfaces de usuario

Si un ser humano quiere interactuar con una máquina necesita un medio para poder hacerlo. Ese medio es, precisamente, lo que se conoce como interfaz de usuario; la parte del sistema que interactúa con el usuario. La interfaz es un lugar de encuentro entre los bits y las personas [Neg94].

La interfaz es la parte visible de las aplicaciones, siendo lo que se percibe de las mismas; por ello, cada vez se les está dando una importancia mayor y se está poniendo más cuidado en su desarrollo. La creación de interfaces de usuario es un área, dentro del desarrollo de software, que ha evolucionado mucho en los últimos años y lo sigue haciendo a día de hoy.

Se podría decir que la interfaz de usuario es lo primero que se juzga de una aplicación, y si no tiene la calidad adecuada puede producir rechazo por parte del usuario. Una interfaz puede incluso poner limitaciones en la

comunicación de la máquina con el usuario. Todo aquello que no se pueda expresar a través de la interfaz se perderá, por ello, tiene gran importancia dentro del desarrollo de una aplicación el diseño de su interfaz.

Un buen programa con una interfaz pobre y no adaptada a las tareas a realizar, no sirve de mucho, ya que la interfaz será más una barrera que un medio que facilite la interacción. Hay que tener siempre presente que la interfaz de usuario determina la usabilidad de la aplicación.

La usabilidad es una medida de uso de un producto por determinados usuarios, con el fin de alcanzar ciertos objetivos de eficiencia, efectividad y satisfacción, todo ello dentro de un contexto de uso concreto [Abascal01]. Por lo tanto, la interfaz de usuario debe ser usable, lo que implicará que el usuario consiga realizar las tareas por medio de ella de forma rápida y sencilla, y se encuentre cómodo con la aplicación.

Realmente las interfaces son el medio en sí. Por ello, las energías del usuario no deben concentrarse en el uso de la interfaz, sino en su propio trabajo [Nor88]. La interfaz de usuario se encarga de adaptar las complejidades del sistema con las capacidades humanas.

En definitiva, se puede decir que una interfaz de usuario debe ser sencilla, de manera que su uso resulte sencillo y el aprendizaje de la misma sea rápido. Por lo tanto, debe ser intuitiva y directa. También debe ser consistente, manteniendo una uniformidad a lo largo de toda su exposición. En definitiva, el usuario debe sentirse cómodo al manejarla y, también, satisfecho, pudiendo realizar las tareas que desee de manera efectiva.

Una interfaz gráfica de usuario (GUI) es una interfaz de usuario en la que se hace uso de un entorno gráfico. Es decir, permite la interacción del usuario con el ordenador mediante la utilización de imágenes, objetos pictóricos (ventanas, iconos, botones, etcétera),…, además de texto. GUI es un acrónimo del vocablo inglés *Graphical User Interface.*

En la actualidad, la interfaz gráfica de usuario más conocida es el sistema de ventanas.

Para comprender mejor la naturaleza de una aplicación con interfaz gráfica se va a comparar ésta con una aplicación de consola o textual. En una aplicación de consola, donde no existe interfaz gráfica, la interacción con el usuario se realiza de forma textual. Mientras que en una aplicación con interfaz gráfica, la interacción se realiza mediante los elementos gráficos que forman parte de la interfaz. Éstos muestran el estado de la aplicación y permiten su control.

1.1.1. Aplicación de consola vs. Aplicación con Interfaz Gráfica de Usuario

Los elementos que componen la interfaz gráfica son **elementos gráficos,** y a través de ellos el usuario puede interactuar con la aplicación. En esta interacción el usuario introduce datos que el programa necesita para llevar a cabo su funcionalidad y obtiene los resultados de procesar dichos datos. Por ejemplo, las ventanas, los botones, las imágenes, etc. son elementos gráficos.

Una diferencia clara entre una aplicación de consola y una aplicación con interfaz gráfica de usuario, es que la primera no tiene ningún elemento gráfico, mientras que en la segunda éstos si existen.

Por otra parte, un **evento** es la notificación que hace un elemento gráfico cuando el usuario interactúa con él. Por lo tanto, si se realiza alguna acción sobre algún elemento de la interfaz, se dice que se ha generado un evento en dicho elemento.

Otra diferencia destacable entre una aplicación de consola y una con interfaz gráfica de usuario está relacionada con los eventos y su gestión, y afecta notablemente a la hora de programar la aplicación. En una aplicación de consola el programa decide cuándo necesita datos del usuario, y es en ese momento cuando los lee de la cadena de entrada. Sin embargo, una aplicación con interfaz gráfica siempre se encuentra a la espera de una entrada de datos por parte del usuario. Éste, en cualquier momento, puede realizar alguna acción sobre algún elemento de la interfaz (por ejemplo, pulsar con el ratón sobre un botón, introducir un carácter por teclado, etcétera).

Para poder atender las acciones realizadas sobre los elementos de la interfaz gráfica de usuario, es necesario asociar código a los eventos que se puedan generar como consecuencia de dichas acciones. De esta manera, si se asocia código a un evento concreto, éste se ejecutará cuando se realice la acción que genera el evento sobre un elemento de la interfaz. Al código asociado a los eventos se le denomina **código de gestión de eventos.**

A la hora de programar una aplicación, dependiendo si ésta es de consola o con interfaz gráfica, dadas las diferencias existentes entre ellas (existencia o no de elementos gráficos y tratamiento de eventos), los pasos a seguir serán bastante distintos.

1.1.2. Elementos gráficos

En las aplicaciones de consola, básicamente lo que se le puede mostrar al usuario son cadenas de caracteres, pudiéndose definir, en algunos casos, la posición donde se desea que éstas aparezcan. Sin embargo, en las aplicaciones con interfaz gráfica de usuario, se puede hacer uso de un conjunto de elementos gráficos que permiten una mejor interacción del usuario con la aplicación.

El JDK (*Java Development Kit*) de *Sun Microsystems*[1] incluye una aplicación de demostración, llamada **SwingSet2,** en la que se pueden ver distintos elementos gráficos. Muestra diversas configuraciones de cada uno de ellos junto con el código fuente necesario para construirlos. Si la instalación del JDK se realiza para incluir las demostraciones, la aplicación se encontrará en el directorio `<JDK_DIR>\demo\jfc\SwingSet2`. Para ejecutarla basta con hacer doble clic en el fichero `SwingSet2.jar` o invocar al comando `java –jar SwingSet2.jar`. En la figura 1.1 se presenta un ejemplo de dicha aplicación, indicando cuáles son algunos de los elementos que a continuación se definen.

[1] http://java.sun.com/j2se/1.5.0/

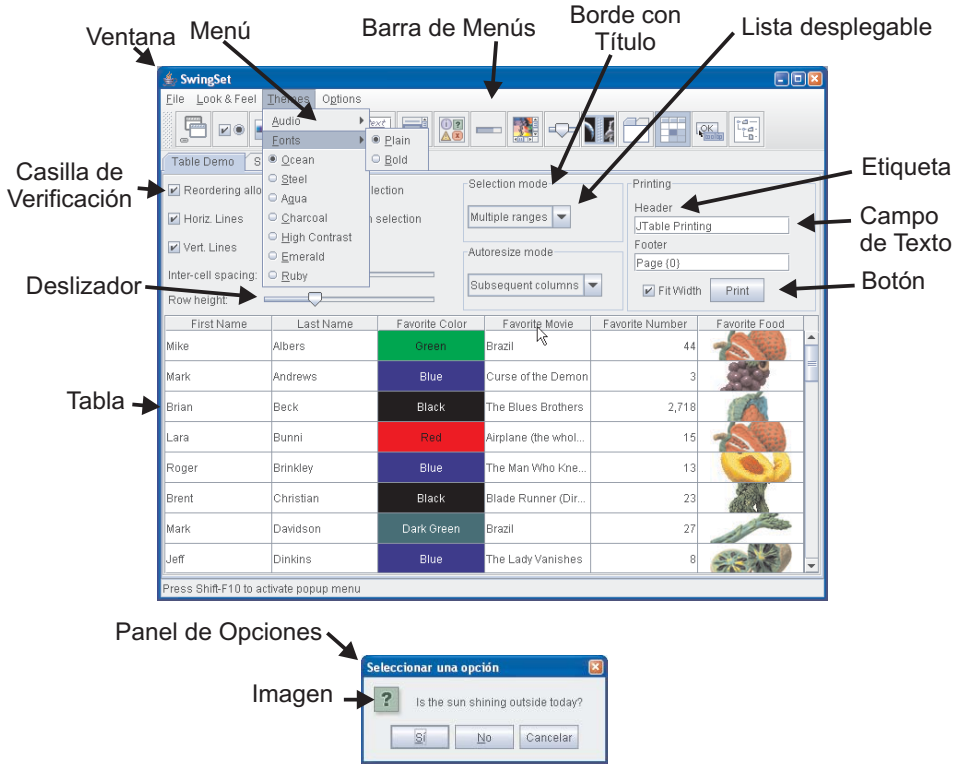

Figura 1.1: Aplicación de ejemplo con interfaz gráfica de usuario

Actualmente, las interfaces gráficas están formadas por ventanas de diferentes tipos que se pueden solapar, mover, cerrar, etc. Dentro de estas ventanas se encuentran otros elementos (botones, etiquetas, campos de texto, imágenes, etc.) que permiten introducir datos y mostrar el estado de la aplicación. El ratón y el teclado permiten manejar los elementos que forman parte de la interfaz. A continuación se describen, de forma general, los distintos elementos que puede tener una interfaz gráfica de usuario.

VENTANAS

Las ventanas son elementos encargados de albergar a otros y que, generalmente, se pueden mover libremente por la pantalla. Existen diferentes tipos en base a su uso y características.

Ventanas de aplicación

Las ventanas de aplicación son aquellas que contienen a todos los elementos de una aplicación.

Cuadros de diálogo

Los cuadros de diálogo son ventanas que, normalmente, se muestran un breve periodo de tiempo en la pantalla. Se suelen utilizar para informar al usuario de alguna situación o pedirle datos en un momento determinado.

De forma habitual, todos los cuadros de diálogo están ligados a una ventana de aplicación, de manera que, cuando la ventana de aplicación se minimiza, éstos también lo hacen.

Al mostrarse en pantalla un cuadro de diálogo, es habitual que se bloqueen todos los elementos de la ventana de aplicación. De esta forma, el usuario sólo puede manejar los elementos del cuadro de diálogo. Por ejemplo, cuando se abre un cuadro de diálogo para seleccionar un fichero de disco, es habitual que no se pueda interactuar con otros elementos de la aplicación hasta que el fichero se haya seleccionado. Si un cuadro de diálogo presenta este funcionamiento se dice que es **modal** (característica que se puede deshabilitar si se considera oportuno).

Es posible que un cuadro de diálogo supere los límites de la ventana de aplicación a la que pertenece; es decir, que si el cuadro de diálogo se mueve o es arrastrado por la pantalla (con el ratón por ejemplo), puede sobrepasar los límites de la ventana.

Ventanas internas

Las ventanas internas son un tipo de ventanas específico. Se suelen utilizar para albergar documentos dentro de la ventana de aplicación o para mostrar cajas de herramientas. Por ejemplo, habitualmente en las aplicaciones de retoque fotográfico, cada una de las imágenes abiertas se muestra dentro de una ventana interna, pudiendo ésta maximizarse o minimizarse.

Una restricción que va unida a este tipo de ventanas es que, a diferencia de los cuadros de diálogo, éstas no pueden sobrepasar los límites de la ventana de aplicación.

Ventanas sin marco

Las ventanas sin marco, como su propio nombre indica, son ventanas que carecen de marco y, por tanto, no tienen ni título ni botones para maximizar, minimizar, etc.

Se suelen mostrar al comienzo de la ejecución de una aplicación cuyo tiempo de inicio sea perceptible. De esta forma, el usuario sabe que la aplicación se está cargando aunque aún no pueda usarla. En algunas ocasiones, en la ventana sin marco se muestra el estado del proceso de inicialización, para ofrecer al usuario una estimación del tiempo restante de dicho proceso.

La funcionalidad aquí descrita para las ventanas es la habitual. Sin embargo, dado que la forma de construir interfaces gráficas se encuentra en constante evolución, pueden producirse cambios al respecto. Un ejemplo de una funcionalidad alternativa son aquellas aplicaciones en las cuales se crea una ventana de aplicación por cada documento abierto, de forma que, mientras no se cierren todos los documentos no se finaliza la ejecución de la aplicación. Otro ejemplo son aplicaciones en las cuales los cuadros de diálogo que muestran herramientas, se ajustan a alguna parte de la ventana de aplicación, dejando de ser cuadros de diálogo.

COMPONENTES

Todos aquellos elementos de una interfaz gráfica con entidad propia y una funcionalidad asociada son componentes. Por ejemplo: botones, barras de desplazamiento, etiquetas, imágenes, listas desplegables, tablas, árboles, etc. No son componentes, por ejemplo, los colores, las líneas, las letras, los píxeles, etc.

Controles

Los controles son aquellos componentes que pueden recibir información del usuario cuando éste interactúa con la aplicación mediante el ratón o el teclado. Los más comunes son: botones, barras de desplazamiento, cuadros de texto, etc.

Contenedores

Un contenedor es un componente que puede mostrar en su interior otros componentes. A los componentes que no son contenedores se les conoce como **componentes atómicos**. Por ejemplo, hay contenedores que muestran todos los componentes a la vez, otros contenedores muestran algunos de los componentes que contienen dependiendo de la pestaña activa, otros muestran los componentes con barras de desplazamiento, etc. A los contenedores se les suele llamar paneles.

MENÚS

Los menús son elementos que contienen botones distribuidos verticalmente. La pulsación de uno de estos botones abrirá un nuevo menú o bien iniciará alguna acción de la aplicación. Los menús pueden aparecer al pulsar el botón secundario del ratón sobre algunos elementos de la interfaz. Si el contenido del menú depende del elemento pulsado, se denomina **menú contextual**.

BARRAS DE MENÚS

Las barras de menús suelen aparecer en la parte superior de las ventanas. Se componen de una barra horizontal con botones, que al ser pulsados despliegan verticalmente un menú.

TOOLTIP

Un tooltip es un mensaje que presenta la descripción de un componente o, simplemente, una ayuda acerca de su utilidad. Aparece cuando el ratón permanece inmóvil sobre dicho componente.

1.1.3. Gestión de los elementos gráficos

Una vez vistos algunos de los elementos que pueden formar parte de una interfaz gráfica de usuario, a continuación, se pasan a ver aspectos relacionados con la gestión de los mismos.

JERARQUÍA DE COMPONENTES

La jerarquía de componentes de una interfaz gráfica muestra las relaciones entre los contenedores y los componentes que contienen. Los contenedores son componentes, por lo tanto, se puede crear una jerarquía de varios niveles. En esta jerarquía también se muestra la relación de la ventana con los componentes que contiene.

DISTRIBUCIÓN DE COMPONENTES

Los componentes dentro de un contenedor han de distribuirse de alguna forma. El desarrollador puede indicar la posición y tamaño de cada componente en píxeles o puede usar algún tipo de lógica o algoritmo. Por ejemplo, puede organizarlos por filas, por columnas, usando una rejilla, con posición fija o variable, dependiendo del tamaño, etc.

FOCO

Se dice que el componente que recibe las pulsaciones del teclado tiene el foco. Por ejemplo, si hay varios campos de texto, sólo el campo que tenga el foco, mostrará lo que el usuario está escribiendo. Normalmente, el foco puede cambiar de componente usando el ratón o mediante la tecla de tabulación del teclado.

1.1.4. Gestión de Eventos

La gestión de los eventos será diferente dependiendo del tipo de aplicación que se desarrolle.

Aplicación de Consola

Una aplicación de consola consiste en una secuencia de instrucciones que se encuentran en un subprograma o método principal, desde donde se hacen llamadas a otros métodos o subprogramas. Al ejecutar una aplicación de este tipo, se van ejecutando cada una de las instrucciones que la componen; algunas de ellas generarán una salida por pantalla y otras pedirán al usuario que introduzca datos. Las instrucciones que piden datos bloquean la ejecución del programa hasta que el usuario los introduzca. Una vez que se obtienen los datos del usuario, se ejecutan las siguientes instrucciones hasta la última, en la que el programa finaliza su ejecución. En la figura 1.2 se muestra gráficamente cómo es una ejecución de una aplicación de consola.

Aplicación con Interfaz Gráfica de Usuario

En este caso, el comportamiento en ejecución es muy diferente con respecto a una aplicación de consola. Al ejecutar una aplicación con interfaz gráfica se pueden distinguir, de forma clara, dos partes:

Construcción de la interfaz inicial

La primera parte se encarga de construir la interfaz gráfica inicial. Se determinan los elementos gráficos que formarán parte de dicha interfaz, su contenido, su organización, etc. Por ejemplo, se indicarán los diferentes botones que se necesiten, la configuración de la ventana de aplicación, los menús, etc.

También se debe establecer la asociación del código de gestión de eventos a los componentes que los puedan producir. De esta manera, se indica, por ejemplo, qué código será ejecutado cuando el usuario pulse sobre un determinado botón, seleccione una determinada opción de un menú, etc. Es decir, se decide qué código se ejecuta para cada evento que se pueda producir.

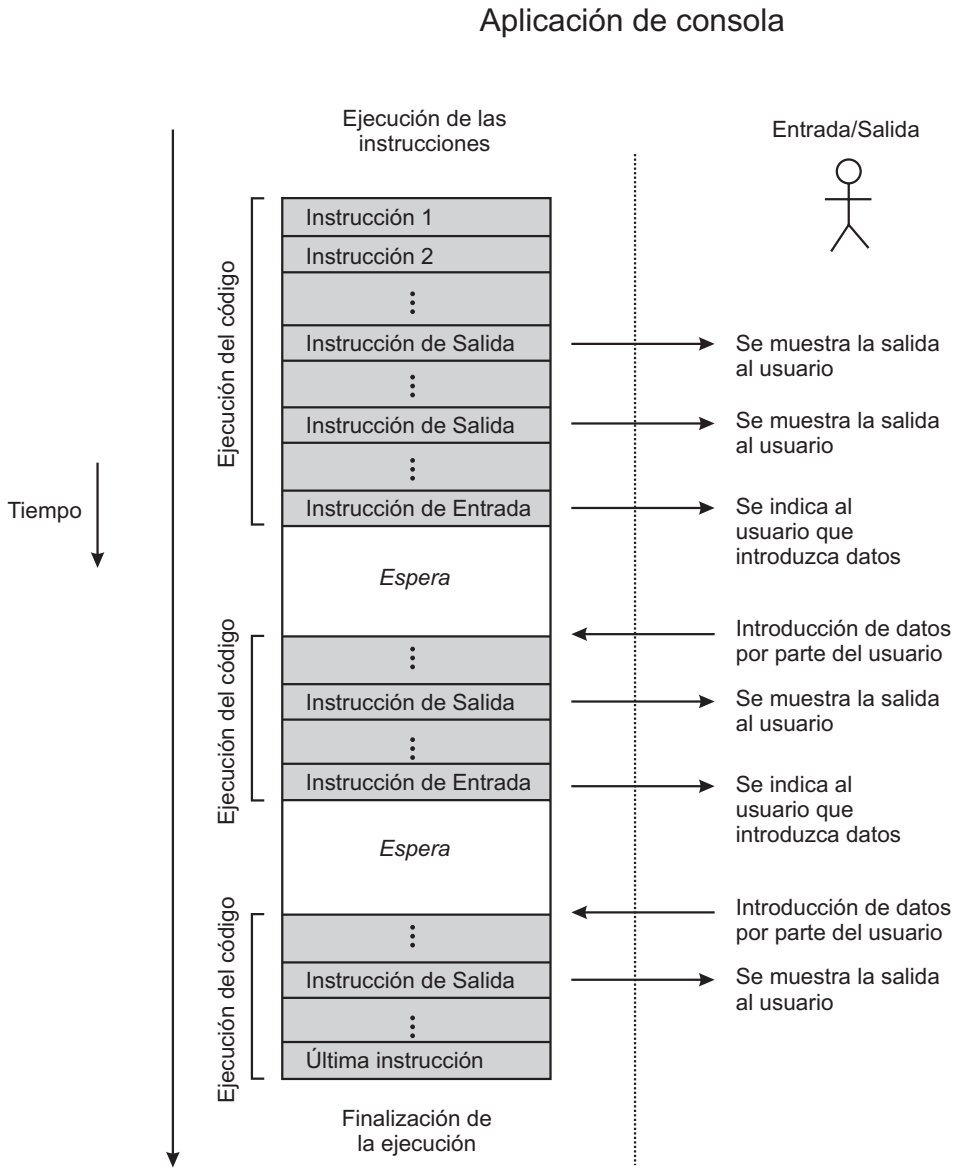

Figura 1.2: Ejecución de una aplicación de consola

Hasta aquí, la aplicación no muestra nada en pantalla. No se presenta ninguna interfaz visible y, por lo tanto, el usuario no podrá interactuar con la aplicación. Cuando la interfaz gráfica de usuario inicial está totalmente construida, se hace visible ejecutando una instrucción especial.

Ejecución del código asociado a los eventos

Una vez que se muestra la interfaz gráfica, la aplicación se queda a la espera de que el usuario interactúe con ella. Cada vez que éste pulse una tecla, mueva el ratón, etc., se generará un evento en el componente correspondiente. En ese momento, se ejecuta el código asociado a dicho evento (asociación realizada en la etapa anterior). La ejecución de este código suele modificar la interfaz gráfica de usuario para reflejar los resultados de la acción realizada.

La aplicación queda a la espera de entradas de usuario y modifica la interfaz cuando se haya finalizado la ejecución del código asociado al evento concreto. Esto implica que todo el código asociado a eventos se ejecuta de forma **secuencial**, uno detrás de otro. Por tanto, un código asociado a un evento, deja bloqueada la interfaz gráfica de usuario hasta que termina su ejecución. Durante este tiempo, la interfaz aparecerá "congelada" y la aplicación tendrá aspecto de haberse "colgado". Más adelante se verán técnicas para evitar esta situación en caso de que sea necesario ejecutar código que tarde un tiempo perceptible en ejecutarse.

Para que finalice la ejecución de una aplicación con interfaz gráfica de usuario es necesario llamar de forma explícita a la instrucción de finalización del programa (normalmente llamada `exit`). También es posible configurar la ventana de aplicación para que, al cerrarse, se finalice la ejecución de la aplicación, algo muy común en este tipo de aplicaciones.

En la figura 1.3 se muestra de forma gráfica la ejecución de una aplicación con interfaz gráfica de usuario.

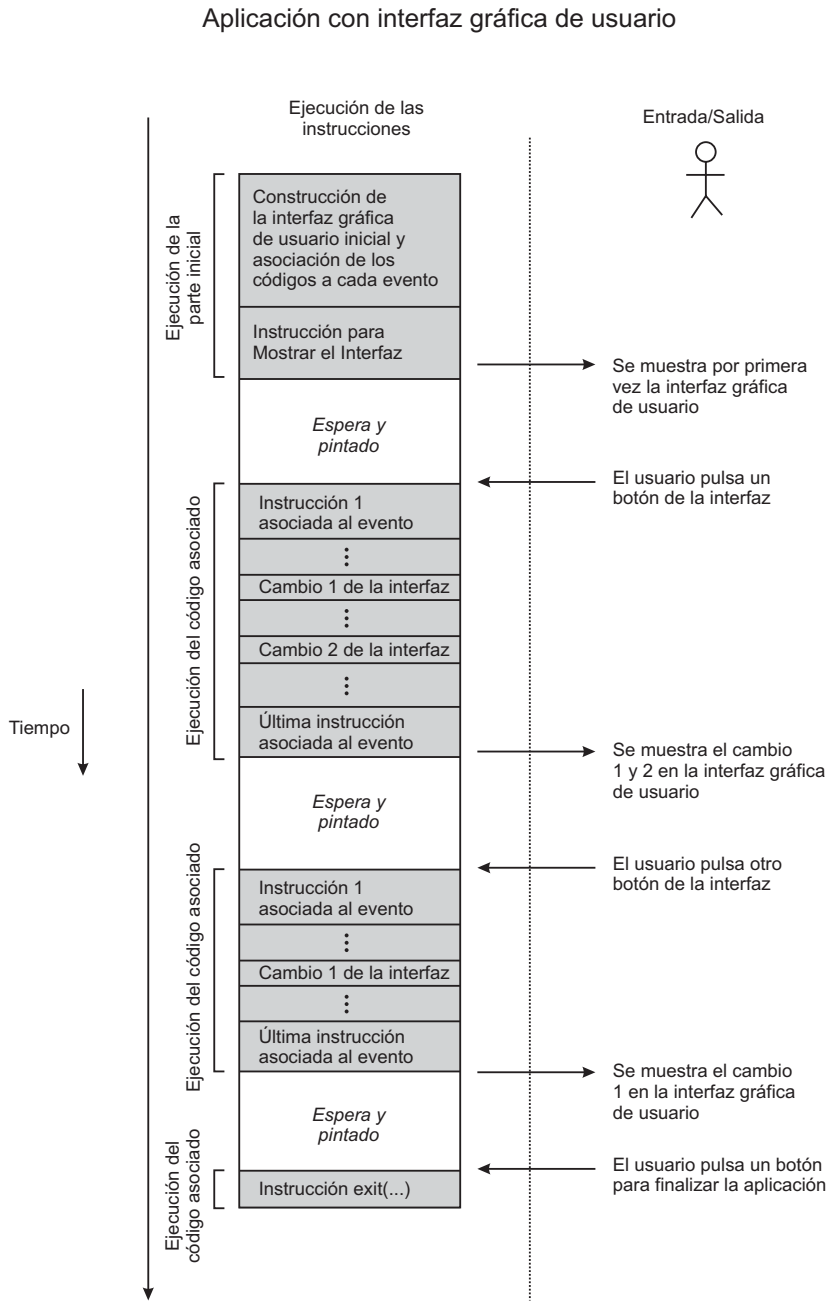

Figura 1.3: Ejecución de una aplicación con interfaz gráfica de usuario

1.2. Interfaces Gráficas de Usuario en Java

La construcción de interfaces gráficas de usuario ha evolucionado mucho desde la primera versión de Java (Java 1.0). En las primeras versiones se utilizaba la librería **AWT** (*Abstract Windows Toolkit*), la cual no era muy potente. Posteriormente, se creó una librería estándar mucho más completa, llamada **Swing**. Por motivos de compatibilidad y reutilización, la librería Swing sigue utilizando, para su funcionamiento, parte de la librería AWT; por lo tanto, será necesario conocer la librería Swing y parte de la librería AWT.

Como se ha mencionado en el prólogo de este libro, lo que se expone aquí no pretende ser una referencia completa de todos y cada uno de los detalles de la librería Swing. Se ha preferido dar una visión general sobre la construcción de aplicaciones con interfaces gráficas de usuario en Java, los conceptos fundamentales y los primeros pasos. Esta decisión se ha tomado teniendo en cuenta que el grado de detalle en el que un programador necesita conocer un componente concreto, dependerá del uso que vaya a hacer de él. También hay que señalar que la construcción de aplicaciones con interfaz gráfica involucra el conocimiento de muchos y variados conceptos, que no se quieren ocultar entre detalles concretos. Para completar la información presentada en este libro se muestra (tabla 1.1) una relación de direcciones de Internet donde se puede encontrar todo tipo de información sobre Swing.

URL	Descripción
http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/package-summary.html	Documentación en formato JavaDoc en línea del paquete javax.swing
http://java.sun.com/j2se/1.5.0/docs/api/java/awt/package-summary.html	Documentación en formato JavaDoc en línea del paquete java.awt
http://www.javadesktop.org	Sitio principal donde poder encontrar información sobre Swing: noticias, APIs, artículos, etc...
http://java.sun.com/products/jfc/tsc/articles/	Artículos oficiales sobre Swing y tecnologías relacionadas

http://java.sun.com/products/jfc/tsc/sightings/	Relación de aplicaciones Java construidas con Swing destacadas por su interfaz gráfica de calidad
http://java.sun.com/docs/books/tutorial/uiswing/	Tutorial oficial de Swing y tecnologías relacionadas
http://www.programacion.com/java/tutorial/swing/	Tutorial oficial de Swing traducido al castellano
http://www.manning.com/sbe/	Libro sobre Swing de la editorial Manning

Tabla 1.1: Direcciones de Internet donde encontrar información sobre Swing

En esta sección se verán, de forma somera, algunas de las posibles aplicaciones con interfaz gráfica de usuario que se pueden crear. Para profundizar más sobre ello se remite al lector al capítulo 8.

1.2.1. Tipos de aplicaciones con interfaz gráfica de usuario

La tecnología Java está muy extendida a día de hoy y permite crear muchos tipos de aplicaciones, con y sin interfaz gráfica. Por ejemplo, aplicaciones de consola, Servlets, Applets, etc.

Las aplicaciones de consola, como ya se ha visto, utilizan la entrada y salida estándar como medio para interactuar con el usuario mediante texto.

Los Servlets son aplicaciones que se ejecutan en un Servidor Web e interactúan con el usuario mediante tecnologías Web (JavaScript, HTML, HTTP, etcétera). Este tipo de aplicaciones no hacen uso de la librería Swing, por ello, no se verán en este libro.

Las aplicaciones con interfaz gráfica de usuario se pueden construir tanto para ordenadores personales como para otro tipo de dispositivos, por ejemplo para los teléfonos móviles, aunque este libro no trata la programación de interfaces gráficas para éstos.

A continuación se presentan, brevemente, los diferentes tipos de aplicaciones que se pueden construir en Java usando Swing: Aplicaciones Autónomas, Applets y Aplicaciones Java Web Start.

Aplicaciones autónomas

Una aplicación autónoma es aquella cuyo funcionamiento es igual a las aplicaciones típicas que se ejecutan en cualquier sistema operativo. Su ejecución se inicia tras pulsar sobre un icono o invocar un comando en la línea de comandos. Habitualmente, cuando se está ejecutando una aplicación de este tipo, aparece en la barra de tareas.

Aplicaciones Java Web Start

Las aplicaciones Java Web Start son aplicaciones muy similares a las aplicaciones autónomas, en lo que se refiere a su funcionamiento durante la ejecución. La diferencia principal es que se pueden cargar desde un servidor Web e instalarse de forma muy cómoda simplemente pulsando un enlace en una página Web.

Applets

Un Applet es una pequeña aplicación Java que se ejecuta dentro de una página Web que está siendo visualizada en un navegador de Internet.

-oOo-

En un primer momento se va a considerar, únicamente, la opción de construir interfaces gráficas de usuario en aplicaciones autónomas, ello se debe a su mayor facilidad de desarrollo y pruebas.

1.3. Pasos básicos para la creación de una interfaz gráfica de usuario

En este apartado se muestran los pasos a seguir en la construcción de una aplicación con interfaz gráfica de usuario. Cada uno de los elementos de una interfaz gráfica de usuario en Java se representa por una instancia de una clase determinada. Por ejemplo, si se quiere que la interfaz de la aplicación tenga un botón, será necesario instanciar un objeto de la clase de ese componente, en este caso: `javax.swing.JButton`.

Para configurar los aspectos gráficos de los componentes se utilizarán los métodos de la clase concreta. Por ejemplo, se podrá cambiar el texto del botón usando el método `void setText(String texto)` de la clase `JButton`.

Todos los componentes de Swing heredan de la clase `javax.swing.JComponent`, la cual hereda de `java.awt.Container` y ésta, a su vez, de `java.awt.Component`. En el capítulo 4 se verán con mayor detalle las relaciones entre las clases de la librería Swing. Por ahora, saber cuales son las clases padre de todos los componentes permitirá ir conociendo algunas de sus características comunes. En la figura 1.4 se muestra la jerarquía de herencia del componente `JLabel`, que representa una etiqueta. Como puede verse, tanto el paquete `java.awt` como el paquete `javax.swing` contienen clases relativas a la interfaz de usuario, lo que hace patente la relación entre Swing y AWT.

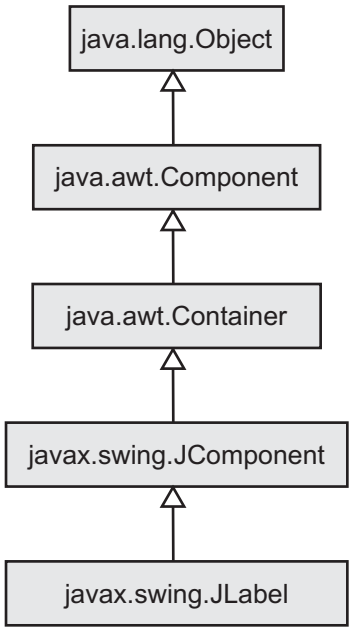

Figura 1.4: Jerarquía de herencia del componente `JLabel`

La interfaz gráfica que se va a construir estará formada por una ventana; dentro de ésta aparecerán: un botón, una etiqueta y un cuadro de texto. Los pasos a seguir son los siguientes:

- Crear una ventana de aplicación.
- Crear los componentes que se van a mostrar en dicha ventana.
- Crear un contenedor.
- Asociar los componentes al contenedor para que, al hacerse visible, muestre en su interior dichos componentes.
- Asociar el contenedor a la ventana de aplicación para que, al hacerse visible, muestre en su interior el contenedor y, por tanto, los componentes asociados.
- Hacer visible la ventana.

La jerarquía de componentes de esta sencilla interfaz gráfica se puede ver en la figura 1.5. En ella se indica que un componente va a ser pintado dentro de un determinado contenedor con una línea terminada en círculo. También se muestra igualmente el contenedor que se pinta dentro de la ventana.

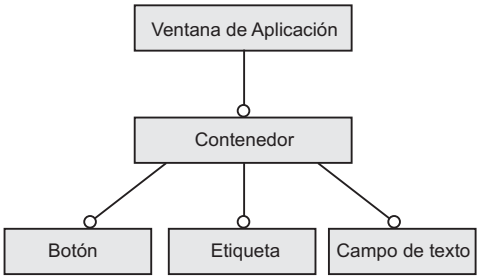

Figura 1.5: Jerarquía de componentes

Para la creación de la interfaz gráfica anterior será necesario introducir algunos elementos gráficos y eventos concretos, en el capítulo 4 se amplía toda la información al respecto.

1.3.1. Creación de una ventana de aplicación

Para crear una ventana de aplicación hay que instanciar un objeto de la clase `javax.swing.JFrame`. Algunos métodos de esta clase relacionados con el aspecto gráfico de la ventana son los siguientes:

- **`public JFrame()`** - Construye una ventana inicialmente invisible.
- **`public JFrame(String title)`** - Construye una ventana inicialmente invisible con el título indicado.
- **`public void setTitle(String title)`** – Establece el título de la ventana.
- **`public void setSize(int width, int height)`** – Establece el tamaño en píxeles de la ventana.
- **`public void setDefaultCloseOperation(int operation)`** – Establece la operación que se ha de realizar cuando el usuario cierra la ventana. Los valores permitidos vienen determinados por las siguientes constantes:
  - **`javax.swing.JFrame.EXIT_ON_CLOSE`** - Finaliza la ejecución de la aplicación.
  - **`javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE`** - No hace nada.
  - **`javax.swing.WindowConstants.HIDE_ON_CLOSE`** - Oculta la ventana (por defecto).
  - **`javax.swing.WindowConstants.DISPOSE_ON_CLOSE`** – Libera los recursos de la ventana, pero no finaliza la ejecución del programa.
- **`public void setResizable(boolean resizable)`** – Establece si el usuario puede cambiar el tamaño de la ventana. Por defecto es `true`.
- **`public void setExtendedState(int state)`** – Establece el estado de la ventana. Puede no funcionar en algunas plataformas. Este método ha de invocarse cuando la ventana sea visible. Los valores permitidos vienen dados por las constantes:
  - **`java.awt.Frame.NORMAL`** – No se encuentra ni minimizada ni maximizada.
  - **`java.awt.Frame.ICONIFIED`** – Minimizada.
  - **`java.awt.Frame.MAXIMIZED_BOTH`** - Maximizada.
  - **`java.awt.Frame.MAXIMIZED_HORIZ`** – Maximizada horizontalmente.

`java.awt.Frame.MAXIMIZED_VERT` – Maximizada verticalmente.

- `public void setLocation(int x, int y)` – Establece la posición de la esquina superior izquierda de la ventana.
- `public void setVisible(boolean b)` – Muestra u oculta la ventana.

Cuando ya se ha creado la ventana de aplicación, se procede a crear los componentes que va a contener.

1.3.2. Creación de los componentes de una ventana de aplicación

Para crear un componente basta con crear una instancia de la clase determinada que represente a dicho componente y configurar ese objeto para que se adapte a las necesidades requeridas para la aplicación particular. Algunos de los métodos relacionados con los aspectos gráficos de algunos componentes muy usados (botones, etiquetas y campos de texto) son los siguientes:

- `JButton`

  `public JButton(String text)` - Crea un botón con el texto indicado.
- `JLabel`

  `public JLabel(String text)` - Crea una etiqueta con el texto indicado.

  `public String getText()` - Devuelve el texto de la etiqueta.

  `public void setText(String text)` - Establece el texto indicado en la etiqueta.
- `JTextField`

  `public JTextField(int columns)` - Crea un campo de texto sencillo con el número de columnas especificado.

  `public String getText()` - Devuelve el texto del campo de texto.

A lo largo del libro se presentarán los componentes más importantes y algunos métodos que permiten configurarlos.

El siguiente paso para crear la interfaz propuesta, una vez se tiene la ventana de aplicación y los componentes que va a contener es crear un contenedor.

1.3.3. Creación de un contenedor

En Swing existen muchos tipos de contenedores; sus diferencias radican en la forma en la que manejan los componentes que tienen dentro. Por ejemplo, el `javax.swing.JTabbedPane` es un contenedor con pestañas donde cada una está asociada a un componente. También existe otro contenedor dividido en dos partes, para dos componentes, donde la separación puede cambiar de posición, es el `javax.swing.JSplitPane`.

En este caso, para construir la interfaz propuesta se va a hacer uso del contenedor de propósito general `javax.swing.JPanel`, que es el más sencillo de todos. Puede contener cualquier número de componentes en su interior, los cuales serán mostrados a la vez. La posición y tamaño de los componentes es configurable.

El método constructor de la clase `JPanel` es:

`public JPanel()` – Crea un contenedor de propósito general.

El siguiente paso consiste en asociar los componentes al contenedor creado.

1.3.4. Asociación de componentes a un contenedor

Para asociar componentes a un contenedor, de forma que se muestren dentro cuando éste sea visible, se usa el siguiente método de la clase `JPanel` (que hereda de `Container`):

`public void add(Component comp)` – Asocia el componente al contenedor, de forma que se muestre el componente al mostrarse el contenedor. Al usar Swing, como parámetro al método `add(...)` se le pasará un objeto de una clase que herede de `JComponent`.

En este punto, se han creado todos los componentes y contenedores necesarios, con las asociaciones pertinentes, lo único que queda es el último paso: asociar el contenedor a la ventana de aplicación.

1.3.5. *Asociación de un contenedor a la ventana de aplicación*

Para asociar un contenedor a la ventana de aplicación, de forma que se muestre dentro cuando ésta sea visible, se usa el siguiente método de la clase `JFrame`:

`public void setContentPane(Container contentPane)` – Establece el componente pasado como parámetro como contenido de la ventana. Al usar Swing, como parámetro al método `setContentPane(...)` se le pasará un objeto de una clase que herede de `JComponent`.

1.3.6. *Hacer visible la ventana*

Por último, lo único que hace falta es hacer visible la ventana para que la interfaz completa se muestre en la pantalla. Para ello, se usa el método de la clase `JFrame`:

`public void setVisible(boolean b)` – Muestra u oculta la ventana dependiendo del parámetro `visible`.

Para ver cómo se relacionan entre sí todos estos pasos en el ejemplo 1.1 se muestra una aplicación que construye una interfaz de usuario simple.

Ejemplo 1.1 – Interfaz de usuario simple

Se pretende construir una aplicación que construya un interfaz de usuario formado por una etiqueta, un campo de texto y un botón. Todo ello en una ventana de 300 x 200 píxeles. La interfaz gráfica de la aplicación se muestra en la figura 1.8.

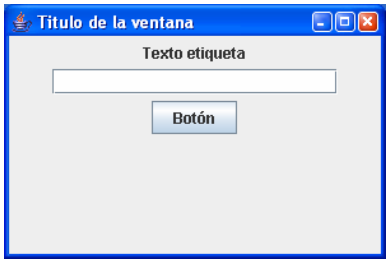

Figura 1.6: Interfaz de usuario simple

El código fuente de la aplicación anterior se muestra a continuación:

libro/ejemplos/ejemplo1.1/InterfazSimple.java

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class InterfazSimple {

    public static void main(String[] args) {

        // Crear la ventana de la aplicacion
        JFrame ventana =
            new JFrame("Titulo de la ventana");
        ventana.setSize(300, 200);
        ventana.setDefaultCloseOperation(
            JFrame.EXIT_ON_CLOSE);

        // Crear los componentes
        JLabel etiqueta1 = new JLabel("Texto etiqueta");
        JTextField campoDeTexto = new JTextField(20);
        JButton boton = new JButton("Botón");

        // Crear un contenedor
        JPanel panelDeContenido = new JPanel();

        // Asociar los componentes al contenedor para
        // que los muestre en su interior
        panelDeContenido.add(etiqueta1);
        panelDeContenido.add(campoDeTexto);
        panelDeContenido.add(boton);

        // Asociar el contenedor a la ventana para
        // que le muestre en su interior
        ventana.setContentPane(panelDeContenido);

        //Hacer visible la ventana
        ventana.setVisible(true);
    }
}
```

1.4. Eventos

Cuando se construye una aplicación con interfaz gráfica de usuario la forma de programar varía con respecto a una aplicación de consola. En una aplicación con interfaz gráfica se asocia código a diferentes eventos que puede producir el usuario cuando interactúa con la aplicación. De manera que, cuando se genera un evento (por ejemplo, pulsar un botón, pulsar una tecla del teclado, pasar el ratón sobre una imagen, etcétera), se ejecuta el código asociado y, habitualmente, se modifica la interfaz gráfica de usuario para mostrar el resultado de esa ejecución. Cuando finaliza la ejecución de ese código asociado, la aplicación espera nuevos eventos por parte del usuario.

Cada componente de la interfaz gráfica de usuario puede generar varios tipos de eventos. Por ejemplo, un componente de árbol genera eventos de un tipo cuando se selecciona alguno de sus nodos, eventos de otro tipo diferente cuando entra o sale el ratón en él, etc.

Se puede asociar un código a cada uno de los eventos que genera un componente. Es importante tener en cuenta que Swing permite asociar **varios códigos distintos** al **mismo tipo de evento** del **mismo componente**. En la figura 1.7 se muestra un esquema indicando esta posibilidad; para ello, se representa mediante una flecha punteada la relación entre un componente que genera eventos y el código asociado.

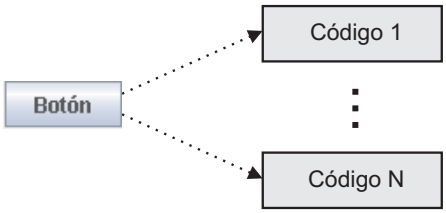

Figura 1.7: Varios códigos asociados a un mismo tipo de evento de un mismo componente

Por otra parte, también está permitido asociar un **mismo código** al **mismo tipo de evento** de **varios componentes distintos** (figura 1.8), de forma que, un mismo código será ejecutado cuando se genere ese tipo de evento en cualquiera de los componentes a los que está asociado.

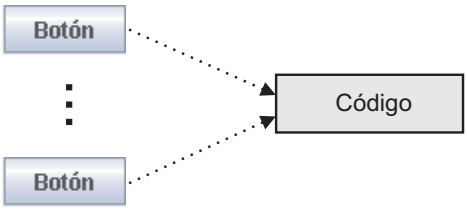

Figura 1.8: Un único código de eventos para varios componentes

Por cada tipo de evento que puede ser generado en cualquier componente, existe una interfaz denominada `XXListener`, siendo `XX` el tipo de evento. Por ejemplo, cuando se interactúa de alguna forma con una ventana, se genera un evento de tipo `Window`. Por tanto, existe una interfaz llamada `java.awt.event.WindowListener`. Todas estas interfaces heredan de la interfaz padre `java.util.EventListener`.

Estas interfaces tienen un método por cada una de las acciones concretas que provocan la generación del evento. Por ejemplo, la interfaz `WindowListener` tiene los métodos `windowActivated(…)`, `windowClosed(…)`, `windowClosing(…)`, `windowDeactivated(…)`, `windowDeiconified(…)`, `windowIconified(…)` y `windowOpened(…)`. Cada uno de estos métodos será ejecutado cuando se produzca dicha acción. Por ejemplo, cuando la ventana se abra, se ejecutará el método `windowOpened(…)`.

Cada uno de los métodos de una interfaz `XXListener` tiene un único parámetro llamado `event` de la clase `XXEvent`. De esta forma, los métodos de la interfaz `WindowListener` tienen un único parámetro de la clase `java.awt.event.WindowEvent`. Todas las clases `XXEvent` disponen de métodos que ofrecen información acerca del evento producido. Por ejemplo, la clase `WindowEvent`, tiene el método `int getNewState()` que informa del estado en el que ha quedado la ventana después de producirse el evento. Las clases `XXEvent` heredan de `java.util.EventObject`. Esta clase tiene el método `Object getSource()` que devuelve una referencia al objeto donde se generó el evento.

Por último, cada componente que genere eventos dispone del método `void addXXListener(XXListener listener)`. Este método se utiliza para asociar el código de gestión de eventos del objeto `listener` al compo-

nente. Cuando en el componente se genere un evento de ese tipo, se ejecutará el método correspondiente en el `listener`. Para saber los eventos que se pueden generar en un componente determinado, basta con ver los métodos de la forma `addXXListener(...)` que tiene, tanto en la propia clase como en las clases padre.

Para crear un código de gestión de eventos y asociar dicho código a los eventos que interesen de un componente determinado se deben seguir los pasos siguientes:

- Crear una clase que implemente una interfaz `XXListener`. Como es lógico, es necesario implementar todos los métodos de la interfaz, no obstante, si no se desea ejecutar nada cuando ocurran determinadas acciones concretas de ese evento, bastará con dejar el cuerpo del método vacío.

Asociar el código anterior al componente o componentes que corresponda. El método void addXXListener(XXListener listener) es el que se utiliza para ello.

Por ejemplo, para asociar un código al evento de tipo `Window`, es necesario hacer que una clase implemente la interfaz `WindowListener`, y por tanto implemente sus métodos. Un objeto de esa clase, deberá asociarse a la ventana (objeto de la clase `JFrame`), para ello deberá invocarse el método `addWindowListener(…)`, pasando como parámetro el objeto de la clase que implemente la interfaz `WindowListener`.

En la figura 1.9 se muestran las clases e interfaces que permiten implementar el evento de tipo `Window`.

Los diferentes tipos de eventos se verán en capítulos posteriores. No obstante, se adelanta un tipo de evento muy utilizado: el evento de tipo `Action`. Este evento se genera cuando se pulsa un botón. Al igual que ocurre con el evento de tipo `Window`, visto anteriormente, existe la interfaz `ActionListener`, que tiene el método `actionPerformed` y cuyo único parámetro se llama `event` y es de tipo `ActionEvent`. Además, la clase `JButton` dispone del método `addActionListener(ActionListener listener)` para asociarle los gestores de eventos.

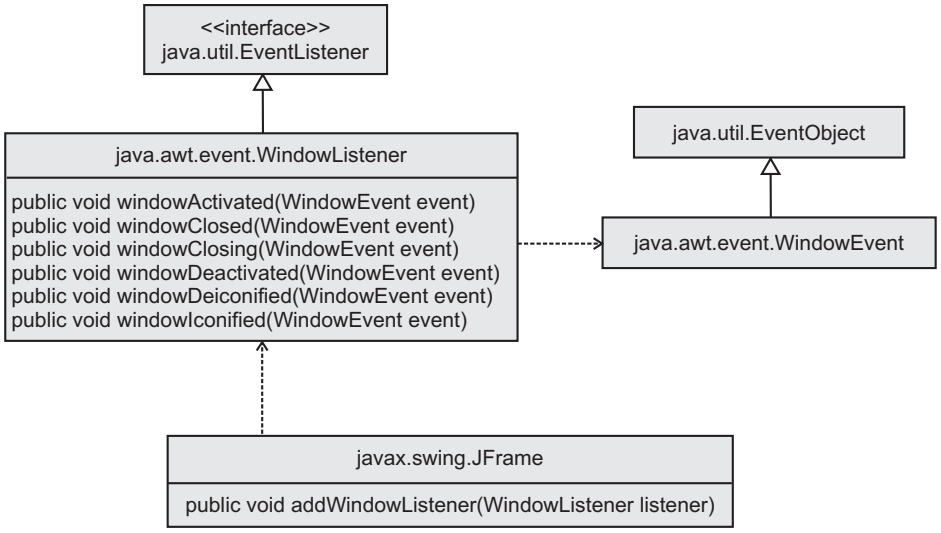

Figura 1.9: Clases relacionadas con el evento de tipo `Window`

1.5. La primera aplicación con interfaz gráfica de usuario

En este punto ya se conocen los mecanismos necesarios para construir una interfaz gráfica de usuario. A continuación, se presenta un primer ejemplo de un programa con interfaz gráfica.

Los pasos básicos para implementar un programa con interfaz gráfica de usuario son:

- Construir la interfaz:
  Construir los elementos gráficos.
  Asociar el código de manejo de eventos a los componentes.
  Hacer visible la ventana de la aplicación.
- Código de manejo de eventos:
  Cuando se produce un evento del usuario, se ejecuta el código de manejo de eventos asociado. En este código, habitualmente, se obtienen datos de la interfaz, se procesan esos datos y se modifica la interfaz en consecuencia.

Existen muchas formas de construir el código para cumplir estas directivas generales. Para aplicaciones pequeñas, la forma en que se construya el código de la interfaz gráfica no será relevante, pero en

aplicaciones medias y grandes, la organización de este código será de vital importancia para que sea más fácil de manejar, mantener, reutilizar, etc. Por este motivo, a medida que se vaya avanzando en este libro, no sólo se verá la librería Swing, sino que se prestará especial atención al diseño de la aplicación y no sólo a su funcionalidad.

Para construir una interfaz gráfica de usuario, el código más sencillo se puede conseguir siguiendo los pasos siguientes:

1. Crear una clase.
2. En el método `main` de esa clase se instancia un objeto de dicha clase.
3. Se ponen como atributos cada uno de los componentes de la interfaz gráfica de usuario.
4. El constructor de la clase se encarga de:
   - Construir los componentes.
   - Asociar el código de manejo de eventos a los mismos.
   - Hacer visible la aplicación.
5. La clase implementa la interfaz `XXListener` e implementa sus métodos. En el cuerpo de esos métodos se obtienen los datos de la interfaz gráfica de usuario: a través del objeto evento, a través de la fuente del evento o directamente de otros componentes de la interfaz gráfica. Posteriormente, se procesan esos datos y se modifica la interfaz para mostrar los resultados.

Ejemplo 1.2 - Interfaz de un "Conversor de Euros a Pesetas"

Se pretende construir una interfaz gráfica de usuario que esté formada por una ventana de 300 x 400 píxeles y cuyo título sea *"Conversor de Euros a Pesetas"*. Dentro de la ventana de aplicación se tiene que mostrar una etiqueta con el texto *"Importe en Euros"* y un cuadro de texto de 20 columnas de longitud. También debe aparecer un botón con el texto *"Convertir"* y una etiqueta con el texto *"Pulse para obtener el importe en pesetas"*.

La aplicación debe presentar la siguiente funcionalidad:

- Al cerrar la ventana de aplicación se finalizará la ejecución del programa.

- Cuando se pulse sobre el botón, se debe obtener el valor del campo de texto, hacer la conversión de euros a pesetas y modificar la interfaz gráfica de usuario con el valor de la conversión.

La interfaz gráfica de la aplicación se muestra en la figura 1.10.

Figura 1.10: Interfaz gráfica de la aplicación de conversión

El código fuente de la aplicación anterior se muestra a continuación.

libro/ejemplos/ejemplo1.2/ConversorEuros.java

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

public class ConversorEuros implements ActionListener {
    private JLabel etiqueta1;
    private JTextField campoDeTexto;
    private JButton boton;
    private JLabel etiqueta2;

    public ConversorEuros() {
        //*********************************************
        //   CREACIÓN DEL INTERFAZ GRÁFICO
        //*********************************************

        // Crear la ventana de la aplicacion
        JFrame ventana = new JFrame(
                "Conversor de Euros a pesetas");
        ventana.setSize(300, 200);
        ventana.setDefaultCloseOperation(
                JFrame.EXIT_ON_CLOSE);

        // Crear los componentes
        etiqueta1 = new JLabel("Importe en Euros");
```

```
        campoDeTexto = new JTextField(20);
        boton = new JButton("Convertir");
        etiqueta2 = new JLabel("Pulse para obtener el "
                + "importe en pesetas");

        // Crear un contenedor
        JPanel panelDeContenido = new JPanel();

        // Configurar el contenedor para mostrar los
        // componentes cuando se muestre.
        panelDeContenido.add(etiqueta1);
        panelDeContenido.add(campoDeTexto);
        panelDeContenido.add(boton);
        panelDeContenido.add(etiqueta2);

        // Configurar la ventana para mostrar el panel
        // cuando se muestre
        ventana.setContentPane(panelDeContenido);

        //*****************************************
        //   ASOCIACIÓN DEL CÓDIGO DE MANEJO DE EVENTOS
        //*****************************************
        boton.addActionListener(this);
        //*****************************************
        //   HACER VISIBLE LA VENTANA
        //*****************************************
        ventana.setVisible(true);
    }

    public static void main(String[] args) {
        new ConversorEuros();
    }

    public void actionPerformed(ActionEvent e) {
        //*********************************************
        //   CÓDIGO DE MANEJO DE EVENTOS
        //*********************************************
        try {
            double euros = Double.parseDouble(
                 campoDeTexto.getText());
            double pesetas = euros * 166.386;
            etiqueta2.setText("Equivale a " + pesetas
                    + " pesetas");
        } catch (NumberFormatException e2) {
            etiqueta2.setText(
                "En el campo de texto no hay un número");
        }
    }
}
```

De acuerdo a los tipos de relaciones que se pueden dar entre clases y objetos, el esquema de la aplicación del ejemplo 1.2 sería el que se muestra en la figura 1.11.

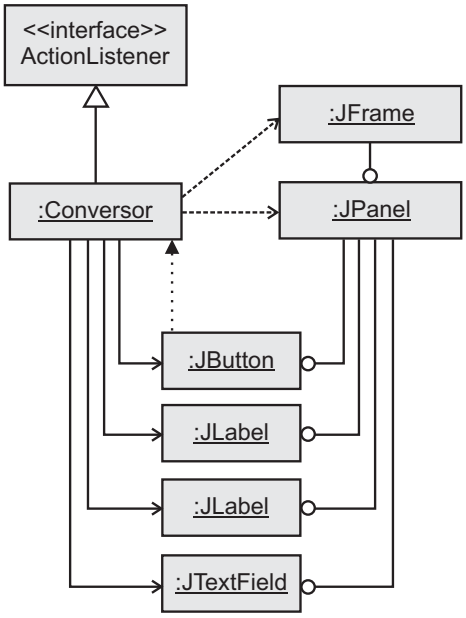

Figura 1.11: Diagrama de clases y objetos de la aplicación de conversión de euros a pesetas (ejemplo 1.2)