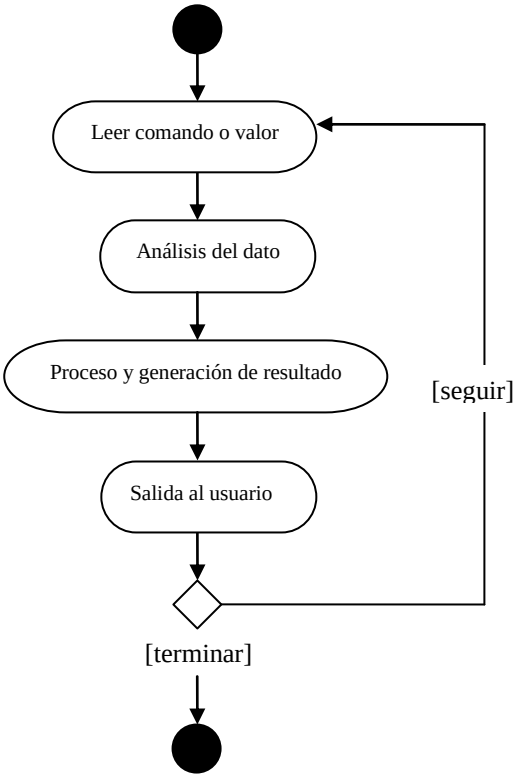
Objetos y Eventos

Programación Secuencial, Interactiva y Orientada a Eventos

Existen distintos tipos de programas. En los primeros tiempos de los ordenadores, antes de la llegada de los ambientes de interfaz de usuario gráfica, la modalidad de programación que se usaba, desde el punto de vista de la interacción del usuario con el programa, era llamada *programación secuencial* (también llamada tipo batch). Un programa secuencial es un programa que se arranca, lee los datos que necesita, realiza los cálculos e imprime o guarda en el disco los resultados. De ordinario, mientras un programa secuencial está ejecutándose no necesita ninguna intervención del usuario. A este tipo de programas se les llama también programas basados u orientados a procedimientos o a algoritmos (procedural languages). Este tipo de programas siguen utilizándose ampliamente en la actualidad, pero la difusión de las PC's favoreció la aparición de otros tipos de programación.

En este paradigma, el programa tiene el control de lo que pasa e interactúa con el usuario pidiéndole datos solamente. Se puede decir que es el usuario quien espera al programa, pues no hay otra forma de que el usuario le diga al programa lo que quiere salvo cuando éste se lo pregunta explícitamente.

La arquitectura típica de esta forma de interacción es un programa que opera según un diagrama de interacción similar al siguiente:



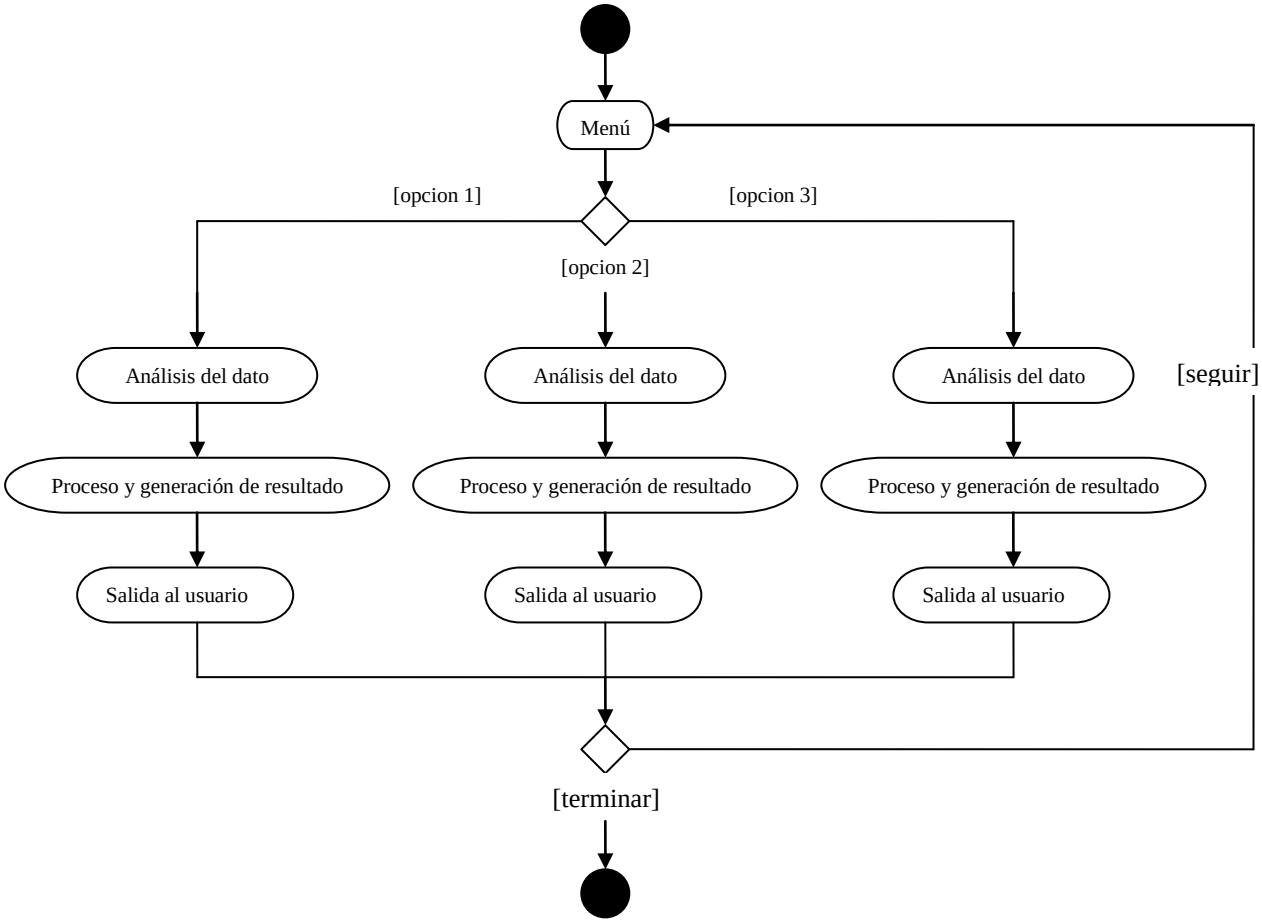
Ejemplos de esta forma de interacción son los sistemas operativos de línea de comandos, como DOS y Unix, y todas las aplicaciones que se desarrol1aban en esos ambientes.

Pero aun en esos ambientes fue surgiendo la necesidad de mejorar la interacción con el usuario, de modo de darle la impresión de que es él quien maneja la secuencia de ejecución.

Así apareció el concepto de menú, en el cual, si bien no se altera la forma de trabajo descrita, se presentan al usuario una serie de opciones de las cuales éste elige una, y la computadora realiza una u otra acción en respuesta.

Los *programas interactivos* exigen la intervención del usuario en tiempo de ejecución, bien para suministrar datos, bien para indicar al programa lo que debe hacer por medio de menús. Los programas interactivos limitan y orientan la acción del usuario.

A continuación se muestra el diagrama de actividades de una interfaz con menús:



Por su parte los *programas orientados a eventos* son los programas típicos de Windows, tales como Word, Excel, PowerPoint y otros. Cuando uno de estos programas arranca, lo único que hace es quedarse a la espera de las acciones del usuario, que en este caso son llamadas eventos.

El usuario indica si quiere abrir y modificar un fichero existente, o bien comenzar a crear un fichero desde el principio. Estos programas pasan la mayor parte de su tiempo esperando las acciones del usuario (eventos) y respondiendo a ellas. Las acciones que el usuario puede realizar en un momento determinado son variadísimas, y exigen un tipo especial de programación: la *programación orientada a eventos*. Este tipo de programación es sensiblemente más complicada que la secuencial y la interactiva, pero con los lenguajes visuales de hoy, se hace accesible y agradable.

Eventos y programación

Evento

Un evento indica la aparición de un estímulo que puede disparar una transición de estados (un elemento pasará de un estado a otro). Es la especificación de un acontecimiento significativo, como una señal recibida, un cambio de estado o el simple paso de un intervalo de tiempo.

En el contexto de la programación, un evento es un vínculo entre una ocurrencia en el sistema y una porción de código que responde a esa ocurrencia. Esa porción de código es lo que denominamos un *manejador de eventos*.

Los eventos pueden ser provocados por el usuario, cuando es éste quien los dispara, o por el sistema o el medio.

Ejemplos de eventos de usuario pueden ser:

- Un *clic* del mouse
- La pulsación de una tecla
- Una selección en un menú.
- Un movimiento del mouse.
- La toma de foco

Ejemplos de eventos del sistema o del medio pueden ser:

- Los intervalos del reloj
- El paso del tiempo
- El cierre del sistema
- Un mensaje que proviene de otra aplicación.
- Un mensaje que proviene de otra computadora

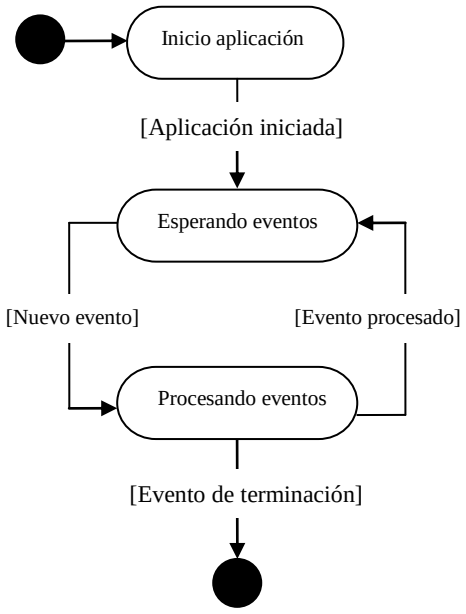
Programación guiada por eventos

La programación guiada por eventos (cuyo nombre viene de la expresión inglesa "event driven programming") es la modalidad de interacción que se impuso con las interfaces de usuario WIMP[1], desde el punto de vista del programador. De todas maneras, no es exclusiva de estas interfaces.

La idea de la programación por eventos, expresada de un modo simple es que, en lugar de que el usuario espere al programa, el programa espera al usuario. En este paradigma el sistema espera la ocurrencia de eventos y actúa en consecuencia. Es decir, el programa debe saber cómo manejar los objetos disponibles y responder a los estímulos que vienen del usuario.

Un evento, en este contexto, es algo significativo que ocurre en el sistema. Podría decirse que un evento es una condición o acción que es observada por el sistema pero proviene desde fuera del control del mismo. Los eventos son previstos, pero no planeados. Esto es, se conoce el hecho de que el evento puede ocurrir, pero no se conocen las circunstancias o el momento en el que ocurrirá.

A continuación se muestra como ejemplo el diagrama de estados de una aplicación simple utilizando programación por eventos:



Estas aplicaciones no tienen un programa principal en el sentido tradicional, pues se inician creando la pantalla principal e iniciando la cola de eventos. El estado casi permanente de la aplicación es estar inactiva, esperando la ocurrencia de eventos.

Los eventos son procesados en orden, obteniéndolos de la *cola de eventos*, y el proceso de cada evento implica despacharlo al componente correcto.

Programación guiada por Eventos y POO

La POO es adecuada para soportar la programación por eventos. Cada componente que recibe un estímulo (mensaje) es un objeto, y el objeto responderá al mensaje con la ejecución de un método definido en su clase. Por ejemplo, cuando un usuario pulsa sobre un botón, está provocando un evento que envía un mensaje al botón, que podría llamarse *OnClick*. A su vez esto provoca que el botón reaccione con un método definido en su clase.

[1] Windows, Icons, Menus and Pointere indicador – Permite al usuario interactuar con una computadora en forma intuitiva, sin tener que aprender comandos

Mensajes y eventos

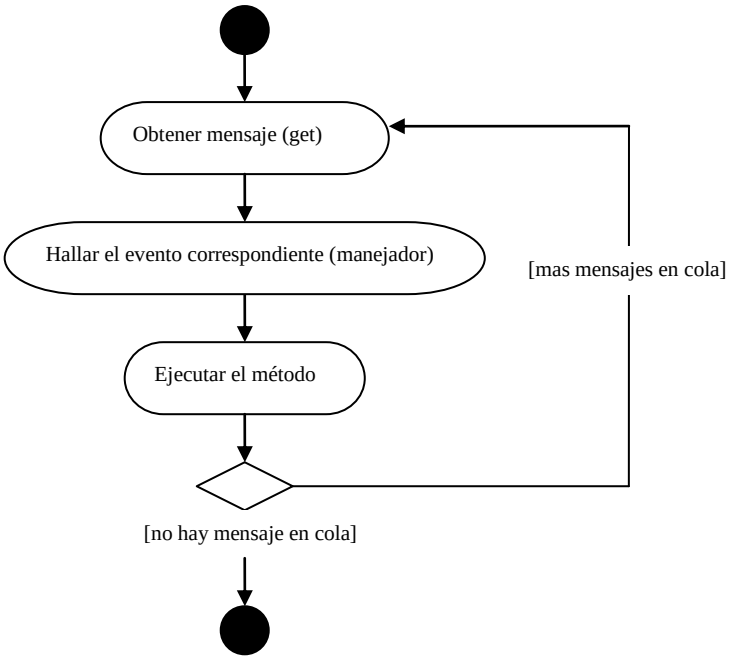
Los mensajes son el medio de comunicación de los objetos.

Los eventos pueden ser creados por el programador o ser propios del sistema o del medio. El programa (código/implementación) se activa cuando le llega un mensaje enviado por la ocurrencia de un determinado evento. Previamente debe definirse **quién** atenderá a esos mensajes.

El sistema operativo es el encargado de realizar una serie de mapeos que transforman los mensajes recibidos en acciones. El programador se encarga de definir en el programa los métodos que atenderán los mensajes generados.

Para procesar los mensajes en orden se implementan colas de mensajes (con la típica estructura FIFO), una para el sistema y una para cada ventana.

Básicamente, hay un ciclo de atención de los mensajes con la siguiente estructura:



Generalización del concepto de eventos en POO

La idea de una generalización del concepto de evento es extenderlo a cualquier sistema, no sólo al entorno visual y la interacción con el usuario (GUI).

De modo que se podría denominar eventos a los siguientes escenarios: "El vuelo 709 ha despegado", "Se ha detectado humo en el sensor número 2".

En este sentido, se puede decir que todos los lenguajes de POO permiten, no sólo trabajar con eventos predefinidos, sino también que el programador implemente sus propios eventos.

En el manejo de eventos la POO es un soporte invaluable, sobre todo por el polimorfismo, que permite que diferentes objetos respondan de forma diferente a un mismo estímulo, basados en el contexto.

El concepto de evento está tan asociado a las interfaces gráficas de usuario (GUI) que todas las implementaciones RAD (Rapid Application Development – Desarrollo Rápido de Aplicaciones) permiten usar eventos predefinidos para los componentes visuales. Sin embargo, como un lenguaje orientado a objetos debiera promover la reutilización y extensión, los auténticos lenguajes de POO deberían poder definir nuevos eventos, incluso sin estar asociados a un componente visual. Esto es lo que ocurre en la biblioteca Swing y la AWT de Java, en Delphi y en todos los productos RAD basados en C++.

Algunas de las herramientas conocidas como IDE's ("Integrated Development Environments") facilitan la generación de programas en Java a través de "Wizards", Menus "Drag & Drop" y otros mecanismos. Aunque un programa también puede ser generado en un editor de textos, para el desarrollo de interfaz gráfica de usuario se recomienda utilizar una herramienta especializada. Por ejemplo para Java:

- Eclipse (http://www.eclipse.org) Open-Source
- NetBeans (http://www.netbeans.org) Open-Source
- JBuilder (http://www.borland.com/jbuilder) de Borland
- VisualAge (http://www-4.ibm.com/software/ad/vajava/) de IBM
- JDeveloper (http://www.oracle.com/index.html) de Oracle

POO e Interfaz Gráfica de Usuario

La amplia y rápida difusión de los lenguajes que implementan este paradigma fue consolidada gracias al auge de las interfaces gráficas de usuario (GUI), para los cuales la programación orientada a objetos está particularmente bien adaptada. Una GUI tiene varias funcionalidades básicas: elementos gráficos (texto, dibujos, imágenes, botones, etc.), los componentes con los cuales interactuará un usuario, y los resultados de las acciones que tienen lugar cuando el usuario interactúa con la GUI (la respuesta al evento).

En Java, el componente que se presenta al usuario no lleva a cabo las acciones realmente, sino que informa a otro objeto que debe llevar a cabo la acción. La administración de la interacción del usuario con una GUI es posible a través de un modelo de delegación de eventos. Esto permite al componente monitorear acciones del usuario mientras otra clase está completando la acción actualmente solicitada.

Java dispone de bibliotecas que permiten desarrollar interfaces WIMP. Las ventanas o modelos de Frames se pueden construir usando cualquiera de las dos galerías de componentes visuales que proporciona java, estas librerías visuales, son:

1.- JAVA AWT: es la librería visual más antigua de java. La versión 1.0 del JDK proveía una biblioteca llamada *Abstract Window Toolkit* (AWT), preparada para correr en cualquier plataforma, pero que generaba interfaces de usuario pobres y ni siquiera era orientada a objetos.

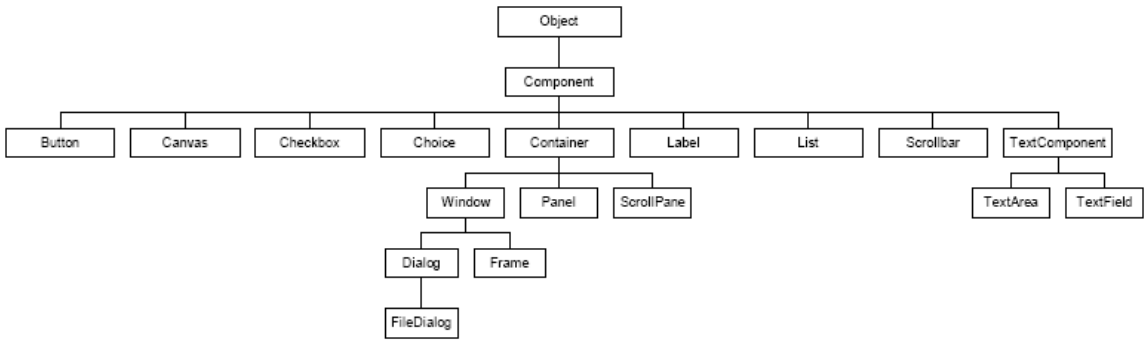
En la versión 1.1 la AWT fue mejorada, con la incorporación de ideas tomadas de Delphi, y con el soporte de objetos. También introdujo un mejor manejo de eventos.

2.- JAVA SWING es la librería de componentes visuales más nueva que proporciona java. Java 2 (JDK 1.2) introdujo un gran cambio con las *Java Foundation Classes* (JFC), una biblioteca que se conoce más comúnmente corno Swing. Swing ha sido diseñada para trabajar correctamente con cualquier producto RAD y sobre cualquier plataforma, al costo de un enorme tamaño.

Una interfaz gráfica está construida en base a elementos gráficos básicos, los *Componentes*. Típicos ejemplos de estos Componentes son los botones, barras de desplazamiento, etiquetas, listas, cajas de selección o campos de texto. Los Componentes permiten al usuario interactuar con la aplicación y proporcionar información desde el programa al usuario sobre el estado del programa. En el AWT, todos los Componentes de la interfaz de usuario son instancias de la clase *Component* o uno de sus subtipos.

Los Componentes no se encuentran aislados, sino agrupados dentro de Contenedores. Los Contenedores contienen y organizan la situación de los Componentes; además, los Contenedores son en sí mismos Componentes y como tales pueden ser situados dentro de otros Contenedores. También contienen el código necesario para el control de eventos, cambiar la forma del cursor o modificar el icono de la aplicación.

Jerarquía de Componentes y eventos de AWT



Jerarquía de clases para los componentes del AWT

Como todas las clases de Java, los componentes utilizados en el AWT pertenecen a una determinada jerarquía de clases, que es muy importante conocer. Según se observa en la figura, todos los componentes descienden de la clase *Component*, de la que pueden heredar algunos métodos interesantes.

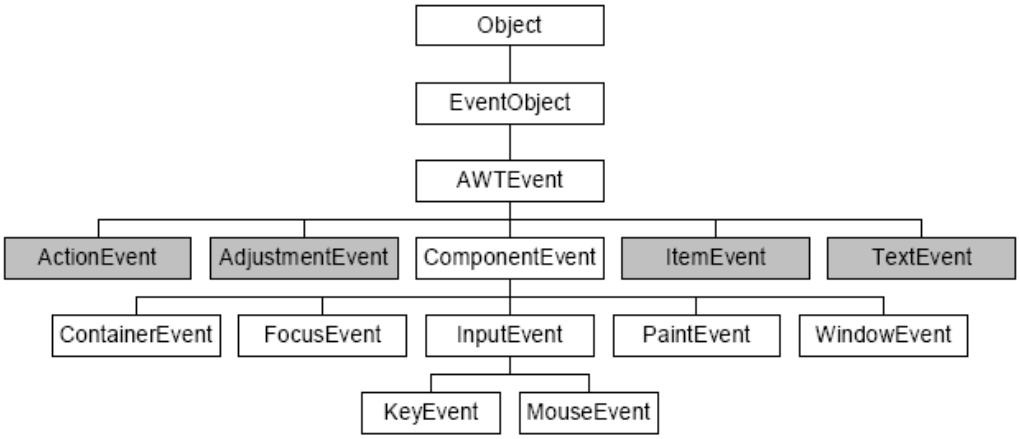
El paquete al que pertenecen estas clases se llama *java.awt.*

Para trabajar con componentes en java, tener en cuenta lo siguiente:

1. Todos los *Components* (excepto *Window* y los que derivan de ella) deben ser añadidos a un *Container*. También un *Container* puede ser añadido a otro *Container*.
2. Para añadir un *Component* a un *Container* se utiliza el método *add()* de la clase *Container*:

```
nombreContenedor.add(nombreComponente);
```

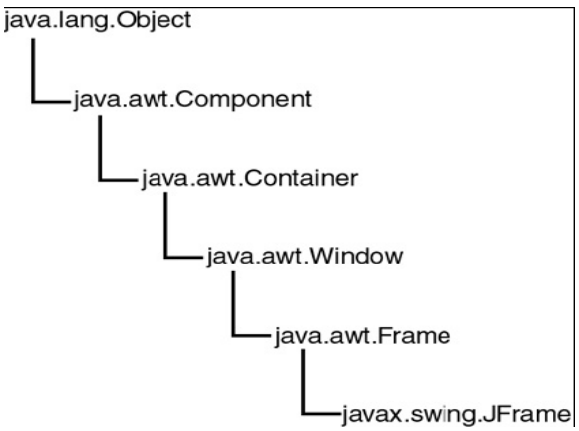
3. Los *Containers* de máximo nivel son las *Windows* (*Frames* y *Dialogs*). Los *Panels* y *ScrollPanes* deben estar siempre dentro de otro *Container*.
4. Un *Component* sólo puede estar en un *Container*. Si está en un *Container* y se añade a otro, deja de estar en el primero.
5. La clase *Component* tiene una serie de funcionalidades básicas comunes (variables y métodos) que son heredadas por todas sus *sub-clases*.



Jerarquía de clases para los eventos del AWT

Todos los eventos de Java 1.1 y Java 1.2 son objetos de clases que pertenecen a una determinada jerarquía de clases. La superclase EventObject pertenece al paquete java.util. De EventObject deriva la clase AWTEvent, de la que dependen todos los eventos de AWT. Estas clases están agrupadas en el paquete java.awt.event.

Jerarquía de Componentes y eventos de SWING



JFrame es el componente principal de esta librería visual o gráfica en java. Es un objeto como cualquier otro, que se deriva de una clase apropiada, tiene sus propiedades y se le pueden asociar eventos. Todo programa visual en java comienza creando un objeto de este tipo, cargando y asignando sus propiedades, y adicionando componentes necesarias (botones, lista, etc).