**EVENTOS**

**Modelo de delegación de eventos:**

* Define mecanismos coherentes y estándares para generar y procesar eventos.
* Una fuente genera un evento y lo envía a uno o más listeners (escuchadores).
* Los listeners esperan hasta que reciben un evento, una vez recibido lo procesa y lo devuelve.
* La lógica que procesa los eventos está separada de la lógica de la interfaz de usuario que genera los eventos.
* Un elemento de interfaz de usuario es capaz de delegar el procesamiento de un evento a una parte separada de código.
* Los listeners atienden a una fuente de la cual reciben la notificación de un evento. Las notificaciones sólo se envían a los listeners que quieren recibirlos.

**Eventos**

* Un evento es un objeto que describe un cambio de estado de una fuente.

**Fuentes de Eventos**

* Una fuente es un objeto que genera un evento, las fuentes pueden generar más de un tipo de eventos.
* Una fuente tiene que **registrar** los listeners para que estos reciban las notificaciones sobre un tipo específico de evento.
* Cada tipo de evento tiene su propio método de registro, por ejemplo.

**public void addTipoListener(TipoListener el)**

**Tipo: nombre del evento**

**el: referencia al listener.**

* Cuando ocurre un evento, se notifica a todos los listeners registrados y reciben una copia del objeto evento – **Multidifusión** de evento.
* Las notificaciones solo se envían a los listeners que quieren recibirlos.
* La fuente que genera los eventos es la que proporciona los métodos para añadir o quitar listeners.

**Listeners – Auditores de eventos**

* Un listener es un objeto que es notificado cuando ocurre un evento.
* 1° Tiene que ser registrado con una o más fuentes para recibir las notificaciones sobre eventos.
* 2° Tiene que implementar métodos para recibir y procesar esas notificaciones.
* Los métodos que reciben y procesan eventos están definidos en un conjunto de interfaces del paquete java.awt.event. Por ejemplo:

|  |  |
| --- | --- |
| **Interface** | **Métodos (para recibir notificaciones)** |
| MouseMotionListener | Cuando se arrastra el ratón |
| Cuando se mueve el ratón |

**Clases de Eventos**

* Las clases que representan eventos son el núcleo del mecanismo de la gestión de eventos en java.
* EventObject es la superclase para todos los eventos, tiene los métodos getSource() y toString().
* AWTEvent es una subclase de EventObject.
* AWTEvent es la superclase de todos los eventos utilizados por el modelo de delegación de eventos.

|  |
| --- |
|  |

**Fuentes de Eventos**

Algunos de los componentes de interfaz de usuario que pueden generar los eventos descritos en el apartado anterior.

|  |
| --- |
|  |

**Las Interfaces de Auditores de Eventos**

* El modelo de delegación de eventos tiene dos partes: fuentes y listeners.
* Los listeners se crean implementando una o más interfaces.
* Cuando se produce un evento, la fuente del evento invoca al método apropiado definido por el listener y proporciona un objeto evento como argumento.

|  |
| --- |
|  |

* Se crea una clase que extienda de la interface deseada y se deben sobre escribir todos los métodos de la interface extendida.

**Clases Adaptadoras**

* Son útiles cuando se quiere recibir y procesar solo alguno de los eventos.
* Se puede definir una nueva clase para que actué como un listener de eventos **extendiendo** una de las clases adaptadoras e implementando solo aquellos eventos (métodos) que se necesiten.
* Extender la clase adaptadora y sobre escribir el método o los métodos que vayamos a utilizar.

|  |
| --- |
|  |

**Clases Internas**

Es una clase definida dentro de otra clase

**Clases Internas Anónimas**

Es aquella a la que no se le asigna un nombre