Manejo de Eventos en Java

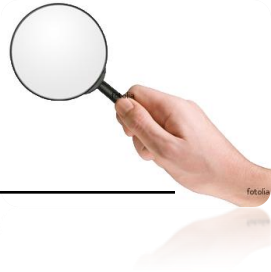
Java.awt.event

Universidad del Valle

CONTENIDO

- Definición
- Clases de: java.awt.Event
- Comportamiento de los eventos
- Tipos de Eventos
- Interfaces Event Listener
- Implementación de Eventos
- Ejemplos
- Ejercicios

EVENTOS

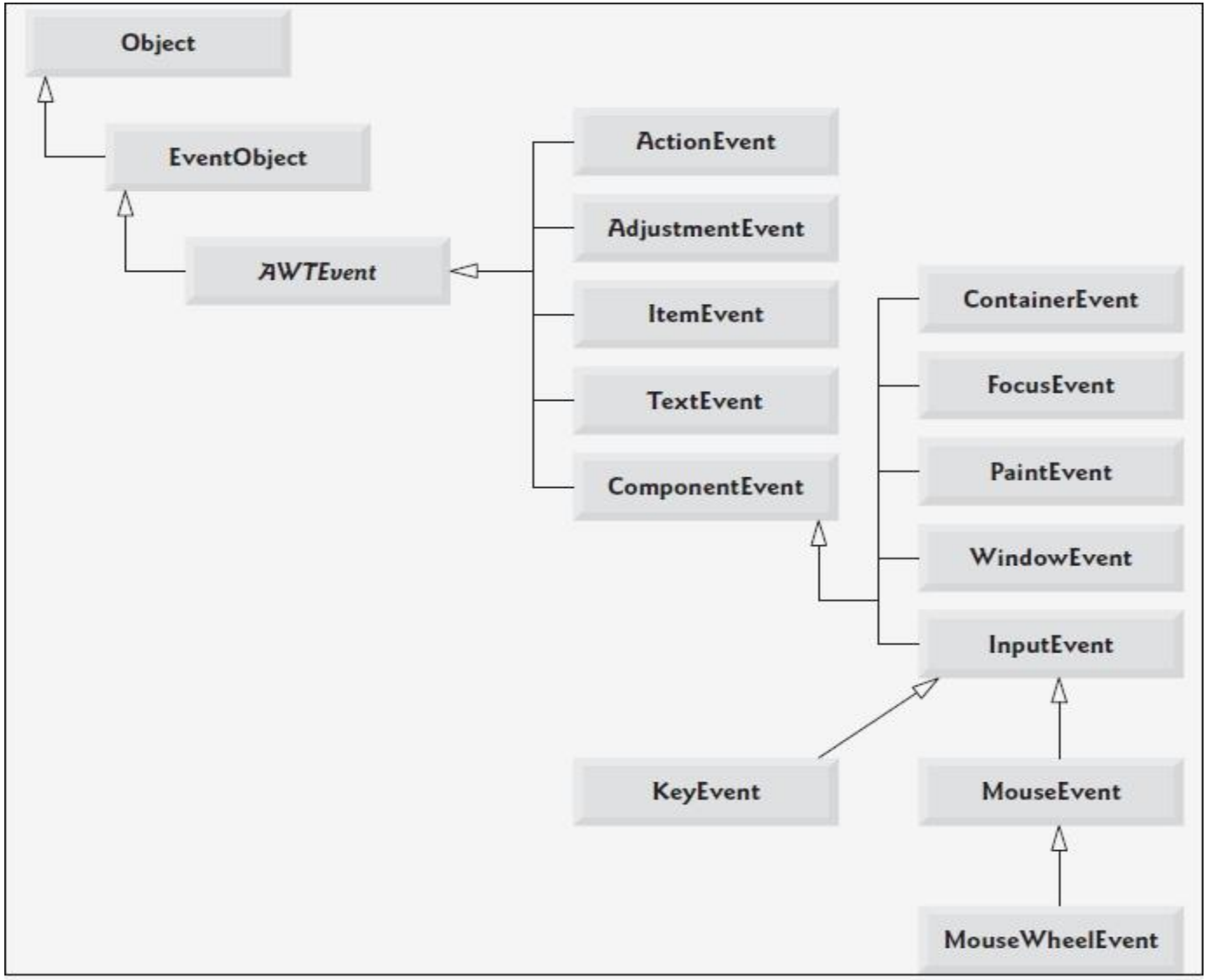


- Un evento es una acción (interacción) iniciada por el usuario en un componente de la GUI. Cada vez que el usuario escribe un carácter, oprime un botón del mouse, hace un movimiento con el cursor del mouse, presiona una combinación de teclas, ocurre un evento.

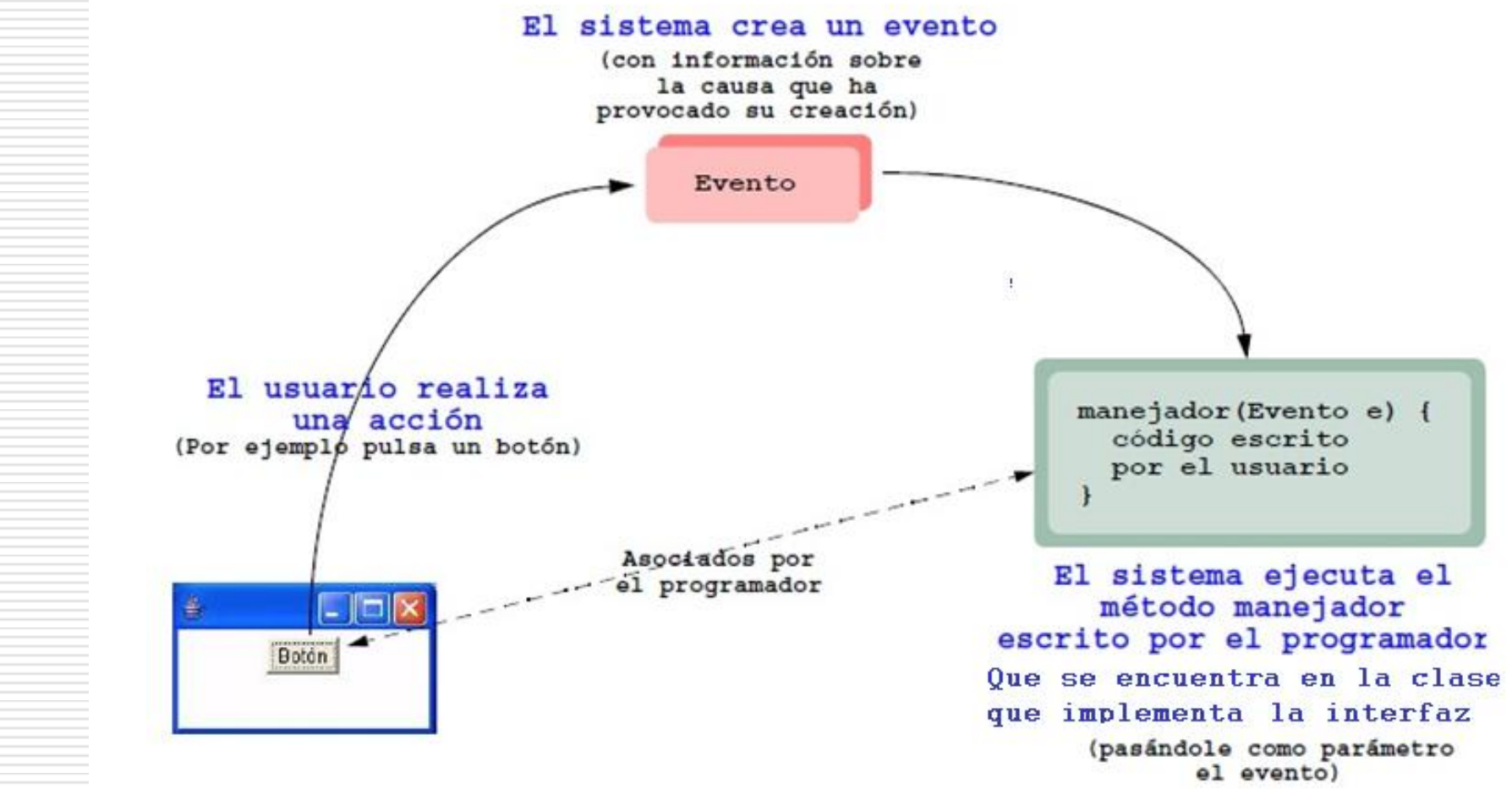
- La clase padre de los eventos es:

java.awt.event

Clases de AWTEvent



Comportamiento de los eventos



Tomado de: http://www.redeszone.net/2011/11/21/curso-java-gui-volumen-x-gestion-de-eventos/

TIPOS DE EVENTOS

- El objeto que recibe el evento (**event source**: un botón, un área de texto, un panel, una lista, entre otros), es notificado en tiempo de ejecución de que recibió el evento, los cuales son de diferentes tipos:

Event Source	Tipo de evento	Se produce cuando:
Button, List, Textfield	ActionEvent	Se efectúe alguna acción sobre el componente, como: la pulsación de un botón.
ScrollBar	AdjustmentEvent	Se ajusta algún valor de un componente.
Component	ComponentEvent	El usuario mueva o redimensione un componente
	FocusEvent	Se cambie el foco de un componente.
	MouseEvent	El usuario efectúe un movimiento con el mouse o haga un click.
	KeyEvent	El usuario pulse una tecla

Tipos de Eventos

Fuente	Tipo de evento	Se produce cuando:
Container	ContainerEvent	se añadan o eliminen componentes en el contenedor
checkBox, Choice, List	ItemEvent	Se ha modificado el estado de algún elemento que pertenece al componente
TextArea, TextField	TextEvent	El contenido de texto de algún componente ha cambiado.
Dialog, Frame	WindowEvent	se realice algún tipo de operación con la ventana como abrirla y cerrarla.

Las Interfaces de escucha (Event Listeners)

- Para poder capturar todos los eventos, Java proporciona las interfaces de escucha (listeners).
- Para cada tipo de evento existe una interfaz de escucha, las cuales están agrupadas de acuerdo a su naturaleza

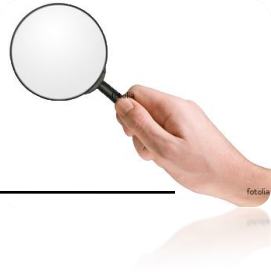
Tipo de Evento	Interfaz de escucha	Descripción
ActionEvent	ActionListener	Acciones sobre componentes
WindowEvent	WindowListener	Cierre o manipulación de ventanas
MouseEvent	MouseListener	Presión de un botón del mouse mientras el cursor está en el componente
MouseMotionEvent	MouseMotionListener	Movimiento del cursor sobre un componente
ComponentEvent	ComponentListener	Visibilidad de los componentes
FocusEvent	FocusListener	Obtención del foco del teclado

LISTENERS

- Para trabajar con eventos se deben tener los siguientes elementos:

1. **La *fuente del evento* (*event source*):** Es el componente que origina el evento.
2. **El *escuchador*: (*event listener*):** Es el encargado de atrapar o escuchar el evento.
3. **El *manejador del evento* (*event handler*):**, es la clase que implementa la interfaz (Event Listener). Esta clase **debe implementar los métodos asociados al evento específico.** Los cuales:
   - **Reciben un objeto evento** (*ActionEvent*) el cuál tiene información sobre el evento que sucedió,
   - **Descifran el evento**, con dicho objeto, (con: getSource()), y
   - **Procesa lo solicitado** por el usuario

IMPLEMENTACIÓN



Para manejar un evento se requiere que:

En la declaración de la clase que maneja el evento (event handler), se indique que **ella implementa la interfaz correspondiente** al evento (**ABC**Listener, donde **ABC** es el tipo de evento a observar o escuchar). La clase puede implementar mas de un ABCListener.

Si la clase no implementa la interfaz puede ser que extienda a una clase que sí la implementa.

```
public class miClase implements ActionListener{...}
```

IMPLEMENTACIÓN

En los componentes que son la fuente del evento (event source) **se registra una instancia del manejador del evento** (event handler), como un observador o escucha del (de los) tipo (s) de eventos que maneja (ABCListener). Es decir, se le dice al componente que va a escuchar los eventos del tipo del manejador.

componente.addActionListener(instancia_de_miClaseListener);

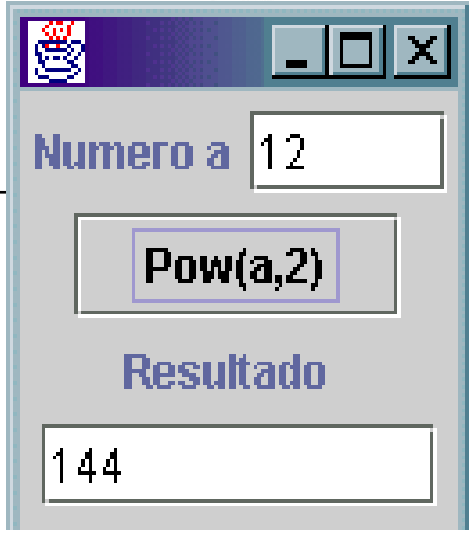
IMPLEMENTACIÓN

En la clase que maneja el evento (event handler) se deben implementar los métodos de la interfaz ABCListener que descifren el evento (ABCEvent) y lo procesen.

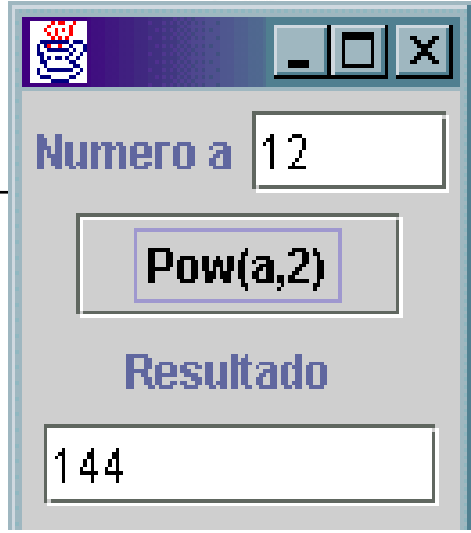
```
public void actionPerformed(ActionEvent e) {

        ...//Código que reaccione a la acción

}
```

Ejemplo - Implementación

```
/*En el constructor de la GUI */

//Se crea instancia de la clase manejadora
ClaseManejadora maneja = new ClaseManejadora();

//Se asocia un escucha al componente
bEncontrarPotencia.addActionListener(maneja);
```



Ejemplo - Implementación




```
/*En el constructor de la GUI */

//Se crea instancia de la clase manejadora
ClaseManejadora maneja = new ClaseManejadora();

//Se asocia un escucha al componente
bEncontrarPotencia.addActionListener(maneja);

//La clase manejadora
Public class ClaseManejadora implements ActionListener{
        //Método que procesa la acción
    public void actionPerformed(ActionEvent e) {
     double valor=Double.parseDouble(tfNumeroA.getText());
     double pw = Math.pow(valor,2);
     tfResultado.setText("”+pw);
    }
  }
```

ACTIONLISTENER

- Cuando un usuario hace clic sobre un botón o presiona la tecla Return después de digitar en un textfield o escoge una opción de un menú, se genera un evento **ActionEvent** y un mensaje **actionPerformed** se envía a todos observadores o escuchas que implementan la interfaz **ActionListener**, registrados en el componente.

- La interfaz ActionListener presenta el método:

```
void actionPerformed(ActionEvent eve)
```

Este método tiene como parámetro el evento generado, que es un instancia de la clase **ActionEvent**, que se genera en tiempo de ejecución cuando el usuario inicia el evento.

COMPONENTLISTENER

- Uno o más eventos se disparan o activan después que un componente es escondido, se hace visible, es movido o se cambia su tamaño.

- La interfaz ComponentListener, presentan los siguientes métodos.

```
void componentHidden(ComponentEvent evento)
void componentMoved(ComponentEvent evento)
void componentResized(ComponentEvent evento)
void componentShown(ComponentEvent evento)
```

ITEMLISTENER

- Los eventos ItemEvent son disparados o activados por componentes que implementan la interfaz **ItemSelectable**. Estos componentes mantienen un estado on/off para uno o más items u opciones.

- La interface ItemListener tiene un solo método

```
void itemStateChanged(ItemEvent eve)
```

- El método se ejecuta justo después de que un componente cambia de estado.

- La clase ItemEvent tiene los siguientes métodos:

Object	getItem()
ItemSelectable	getItemSelectable()
Int	getStateChange()

MouseEvent - MouseListener

- Los eventos MouseEvent le informan cuando el usuario utiliza el mouse para interactuar con un componente.

getClickCount(): Devuelve el número de clic asociados con el evento
getX(), getY(): Devuelve la posición x,y del mouse cuando se genera un evento

- Ocurren cuando el cursor entra o sale del área del componente en la ventana y cuando el usuario oprime o libera el botón del mouse. La interfaz MouseListener contienen los siguientes métodos:

```
void    mouseClicked (MouseEvent eve)
void    mousePressed (MouseEvent eve)
void    mouseReleased (MouseEvent eve)
void    mouseEntered (MouseEvent eve
void    mouseExited (MouseEvent eve)
```

KEYLISTENER

- La Clase **KeyEvent** define los siguientes métodos útiles:

`int getKeyChar()`	`void setKeyChar()`
`int getKeyCode()`	`void setKeyCode(int)`

- La Clase **KeyEvent**, define muchas constantes así:

  KeyEventVK_A especifica la tecla **A**.

  KeyEventVK_ESCAPE especifica la tecla **ESCAPE**.

`void setModifiers(int)`	`String getKeyText()`
`int getKeyModifiersText()`	

Métodos de la clase KeyEvent

getKeyChar(): Devuelve el carácter asociado con la tecla que produjo el evento

getKeyCode(): Devuelve el código de la tecla que produjo el evento

getKeyModifiersText(int): Devuelve una cadena que indica el modificador de la tecla, por ejemplo "Shift"

getKeyText(int): Devuelve una cadena que indica el tipo de tecla pulsada. Ejmp: F1, indicando que es una tecla de función

FocusListener

- Muchos componentes, aún aquellos que inicialmente trabajan con el Mouse (botones), pueden operar con el teclado.
- Para que el presionar una tecla pueda afectar un componente, el componente debe tener el foco del teclado.
- Al menos un componente en la ventana del sistema puede tener el foco del teclado.
- Los eventos de Foco son disparados o activados cuando un componente gana o pierde el Foco
- Típicamente el usuario establece el foco haciendo clic sobre una ventana o un componente, ó haciendo "tab" entre los componentes.

CONTAINERLISTENER

- Los eventos ContainerEvent son disparados o activados después que un componente es agregado o removido del contenedor.

- Estos Eventos son usados solo para notificar que se presentó la adición o remoción del componente, no es necesario el ContainerListener para que los componentes puedan ser agregados o removidos.

- La interfaz ContainerListener presenta 2 métodos:

```
void componentAdded(ContainerEvent eve)
void componentRemoved(ContainerEvent eve)
```

- La clase ContainerEvent, define 2 métodos útiles:

```
Component getChild()
Component getContainer()
```

WINDOWLISTENER

- Los eventos de Ventanas son disparados o activados cuando una ventana es abierta, cerrada, iconificada (iconified), desconificada (deiconified), activada o desactivada.

- Los usos más comunes, son por **ejemplo**, usar un WindowListener, cuando el usuario cierra una ventana, para preguntar si desea salvar los cambios o si es la última ventana cerrada para salir del programa.

  Por defecto cuando un usuario cierra una ventana esta se vuelve invisible

.

MÉTODOS DE WINDOWLISTENER

windowActivated(WindowEvent e)	Es invocado cuando una ventana es seteada como la ventana activa
windowClosed(WindowEvent e)	Es invocado cuando una ventana ha sido cerrada
windowClosing(WindowEvent e)	Es invocado cuando el usuario intenta cerrar la ventana
windowDeactivated(WindowEvent e)	Es invocado cuando la ventana deja de ser la ventana activa
windowDeiconified(WindowEvent e)	Es invocado cuando una ventana pasa de estado minimizado a normal
windowIconified(WindowEvent e):	Es invocado cuando una ventana va estado normal a minimizada
windowOpened(WindowEvent e)	Es invocado la primera vez que la ventana se hace visible.

EJERCICIOS

- Realice la implementación para los eventos de las interfaces trabajadas en las diapositivas de GUI

REFERENCIAS

- JAVA Como Programar séptima edición DEITEL. Pearson, capítulo 11
- JAVA 2 cuarta edición. McGraw Hill 2003, Capítulo 10
- http://docs.oracle.com/javase/7/docs/api/java/awt/event/package-summary.html