Classes for event handling : ActionEvent

An **ActionEvent** is generated when a button is pressed, a list item is double-clicked, or a menu item is selected. The **ActionEvent** class defines four integer constants that can be used to identify any modifiers associated with an action event:

`ALT_MASK, CTRL_MASK, META_MASK, and SHIFT_MASK.`

In addition, there is an integer constant, `ACTION_PERFORMED`, which can be used to identify action events

Constructors
`ActionEvent(Object src, int type, String cmd)`
`ActionEvent(Object src, int type, String cmd, int modifiers)`
`ActionEvent(Object src, int type, String cmd, long when, int modifiers)`

Here, `src` is a reference to the object that generated this event. The `type` of the event is specified by type, and its command string is `cmd`. The argument `modifiers` indicates which modifier keys (ALT, CTRL, META, and/or SHIFT) were pressed when the event was generated. The `when` parameter specifies when the event occurred. The third constructor was added by Java 2, version 1.4

Classes for event handling : AdjustmentEvent

An **AdjustmentEvent** is generated by a *scroll bar*. There are five types of adjustment events. The **AdjustmentEvent** class defines integer constants that can be used to identify them. The constants and their meanings are shown here:

Constants	Definition
BLOCK_DECREMENT	The user clicked inside the scroll bar to decrease its value.
BLOCK_INCREMENT	The user clicked inside the scroll bar to increase its value.
TRACK	The slider was dragged
UNIT_DECREMENT	The button at the end of the scroll bar was clicked to decrease its value.
UNIT_INCREMENT	The button at the end of the scroll bar was clicked to increase its value.

Constructor
AdjustmentEvent(Adjustable src, int id, int type, int data)

Methods	Definition
getAdjustable( )	It returns the object that generated the event.
int getValue( )	The amount of the adjustment can be obtained.

Classes for event handling : ComponentEvent

A **ComponentEvent** is generated when the size, position, or visibility of a component is changed. There are four types of component events. The **ComponentEvent** class defines integer constants that can be used to identify them.

Constants	Definition
COMPONENT_HIDDEN	The component was hidden.
COMPONENT_MOVED	The component was moved.
COMPONENT_RESIZED	The component was resized.
COMPONENT_SHOWN	The component became visible.

Constructor
ComponentEvent(Component src, int type)

Methods	Definition
getComponent( )	It returns the component that generated the event.

Classes for event handling : ContainerEvent

A **ContainerEvent** is generated when a component is added to or removed from a container. There are two types of container events.

Constants	Definition
COMPONENT_ADDED	The component was added.
COMPONENT_REMOVED	The component was removed.

Constructor
ContainerEvent(Component src, int type, Component comp)

Methods	Definition
getChild( )	returns a reference to the component that was added to or removed from the container
getContainer( )	Obtain a reference to the container that ...ed this event

Classes for event handling : FocusEvent

A **FocusEvent** is generated when a component gains or loses input focus.

Constants	Definition
FOCUS_GAINED	The component has been focused.
FOCUS_LOST	The component lost its focus.

Constructors
FocusEvent(Component src, int type)
FocusEvent(Component src, int type, boolean temporaryFlag)
Focus Event(Component src, int type, boolean temporaryFlag, Component other)

Methods	Definition
getOppositeComponent( )	To determine the other component
isTemporary( )	It indicates if this focus change is temporary

Classes for Event Handling : ItemEvent

An **ItemEvent** is generated when a check box or a list item is clicked or when a checkable menu item is selected or deselected.

Constants	Definition
DESELECTED	The user deselected an item.
SELECTED	The user selected an item.

Constructor
ItemEvent(ItemSelectable src, int type, Object entry, int state)

Methods	Definition
getItem( )	It can be used to obtain a reference to the item that generated an event.
getItemSelectable( )	It can be used to obtain a reference to the ItemSelectable object that generated an event
getStateChange( )	It returns the state change (i.e., SELECTED or DESELECTED) for the event.

Classes for event handling : InputEvent

The abstract class **InputEvent** is a subclass of **ComponentEvent** and is the superclass for component input events. Its subclasses are **KeyEvent** and **MouseEvent**.

Constants
ALT_MASK
ALT_GRAPH_MASK
BUTTON1_MASK
BUTTON2_MASK
BUTTON3_MASK
CTRL_MASK
META_MASK
SHIFT_MASK

Methods	Definition
int getModifiers()	To obtain a value that contains all of the original modifier flags

Classes for Event Handling : KeyEvent

A **KeyEvent** is generated when keyboard input occurs.

Constants	Definition
KEY_PRESSED	This event is generated when any key is pressed
KEY_RELEASED	This event is generated when any key is released
KEY_TYPED	This event is generated when a character is generated

Constructor
KeyEvent(Component src, int type, long when, int modifiers, int code, char ch)
KeyEvent(Component src, int type, long when, int modifiers, int code)

Methods	Definition
char getKeyChar( )	It returns the character that was entered.
int getKeyCode( )	It returns the key code.

Classes for event handling : MouseEvent

A `MouseEvent` is generated when mouse input occurs

Constants	Definition
MOUSE_CLICKED	The user clicked the mouse.
MOUSE_DRAGGED	The user dragged the mouse
MOUSE_ENTERED	The mouse entered a component.
MOUSE_EXITED	The mouse exited from a component.
MOUSE_MOVED	The mouse moved.
MOUSE_PRESSED	The mouse was pressed.
MOUSE_RELEASED	The mouse was released.
MOUSE_WHEEL	The mouse wheel was moved (Java 2, v1.4)

Methods	Definition
`char getKeyChar( )`	It returns the character that was entered.
`int getKeyCode( )`	It returns the key code.

Constructor

```
MouseEvent(Component src, int type, long when, int modifiers,
           int x, int y, int clicks, boolean triggersPopup)
```

Classes for Event Handling : TextEvent

Instances of **TextEvent** class describe text events. These are generated by text fields and text areas when characters are entered by a user or program.

Constants	Definition
TEXT_VALUE_CHANGED	When an update in the text is triggered

Constructor
TextEvent(Object src, int type)

Classes for Event Handling : WindowEvent

A **WindowEvent** is generated when window container get some changes.

Constants	Definition
WINDOW_ACTIVATED	This event is generated when any key is pressed
WINDOW_CLOSED	This event is generated when any key is released
WINDOW_CLOSING	This event is generated when a character is generated
WINDOW_DEACTIVATED	The window was deactivated.
WINDOW_DEICONIFIED	The window was deiconified.
WINDOW_GAINED_FOCUS	The window gained input focus
WINDOW_ICONIFIED	The window was iconified.
WINDOW_LOST_FOCUS	The window lost input focus.
WINDOW_OPENED	The window was opened.
WINDOW_STATE_CHANGED	The state of the window changed.

Constructor
WindowEvent(Window src, int type, Window other)
WindowEvent(Window src, int type, int fromState, int toState)
WindowEvent(Window src, int type, Window other, int fromState, int toState)

Methods	Definition
getWindow( )	It returns the Window object that generated the event
getOppositeWindow()	It returns the opposite Window object
getOldState()	It returns the details before modification
getNewState()	It returns the modification details

Interfaces for event handling

Interface	Description
ActionListener	The listener interface for receiving action events.
AdjustmentListener	The listener interface for receiving adjustment events.
AWTEventListener	The listener interface for receiving notification of events dispatched to objects that are instances of Component or MenuComponent or their subclasses.
ComponentListener	The listener interface for receiving component events.
ContainerListener	The listener interface for receiving container events.
FocusListener	The listener interface for receiving keyboard focus events on a component.
HierarchyBoundsListener	The listener interface for receiving ancestor moved and resized events.
HierarchyListener	The listener interface for receiving hierarchy changed events.
InputMethodListener	The listener interface for receiving input method events.

Interfaces for event handling : ActionListener

This interface defines the `actionPerformed( )` method that is invoked when an action event occurs

Methods
`void actionPerformed(ActionEvent ae)`

Interfaces for Event Handling : AdjustmentListener

This interface defines the adjustment `ValueChanged( )` method that is invoked when an adjustment event occurs.

Methods
`void adjustmentValueChanged(AdjustmentEvent ae)`

Interfaces for event handling : ComponentListener

This interface defines four methods that are invoked when a component is resized, moved, shown, or hidden

Methods
void componentResized(ComponentEvent ce)
void componentMoved(ComponentEvent ce)
void componentShown(ComponentEvent ce)
void componentHidden(ComponentEvent ce)

Note: The AWT processes the resize and move events. The componentResized( ) and componentMoved( ) methods are provided for notification purposes only.

Interfaces for Event Handling : ContainerListener

This interface contains two methods. When a component is added to a container, componentAdded( ) is invoked. When a component is removed from a container, componentRemoved( ) is invoked.

Methods
void componentAdded(ContainerEvent ce)
void componentRemoved(ContainerEvent ce)

Interfaces for event handling : FocusListener

This interface defines two methods. When a component obtains keyboard focus, focusGained( ) is invoked. When a component loses keyboard focus, focusLost( ) is called.

Methods
void focusGained(FocusEvent fe)
void focusLost(FocusEvent fe)

Interfaces for event handling : ItemListener

This interface defines the `itemStateChanged( )` method that is invoked when the state of an item changes.

Methods
`void itemStateChanged(ItemEvent ie)`

Interfaces for event handling : KeyListener

This interface defines three methods. The keyPressed( ) and keyReleased( ) methods are invoked when a key is pressed and released, respectively. The keyTyped( ) method is invoked when a character has been entered.

Methods
void keyPressed(KeyEvent ke)
void keyReleased(KeyEvent ke)
void keyTyped(KeyEvent ke)

Interfaces for event handling : KeyListener

This interface defines five methods. If the mouse is pressed and released at the same point, mouseClicked( ) is invoked. When the mouse enters a component, the mouseEntered( ) method is called. When it leaves, mouseExited( ) is called. The mousePressed( ) and mouseReleased( ) methods are invoked when the mouse is pressed and released, respectively.

Methods
void mouseClicked(MouseEvent me)
void mouseEntered(MouseEvent me)
void mouseExited(MouseEvent me)
void mousePressed(MouseEvent me)
void mouseReleased(MouseEvent me)

Interfaces for event handling : MouseWheelListener

This interface defines the mouse WheelMoved( ) method that is invoked when the mouse wheel is moved.

Methods
void mouseWheelMoved(MouseWheelEvent mwe)

Interfaces for Event Handling : TextListener

This interface defines the `textChanged( )` method that is invoked when a change occurs in a text area or text field.

Methods
`void textChanged(TextEvent te)`

Interfaces for event handling : WindowFocusListener

This interface defines two methods: windowGainedFocus( ) and windowLostFocus( ). These are called when a window gains or losses input focus.

Methods
`void windowGainedFocus(WindowEvent we)`
`void windowLostFocus(WindowEvent we)`

Note: `WindowFocusListener` was added by Java 2, version 1.4.

Interfaces for Event Handling : WindowListener

This interface defines seven methods. The windowActivated( ) and windowDeactivated( ) methods are invoked when a window is activated or deactivated, respectively. If a window is iconified, the windowIconified( ) method is called. When a window is deiconified, the windowDeiconified( ) method is called. When a window is opened or closed, the windowOpened( ) or windowClosed( ) methods are called, respectively. The windowClosing( ) method is called when a window is being closed.

Methods
void windowActivated(WindowEvent we)
void windowClosed(WindowEvent we)
void windowClosing(WindowEvent we)
void windowDeactivated(WindowEvent we)
void windowDeiconified(WindowEvent we)
void windowIconified(WindowEvent we)
void windowOpened(WindowEvent we)