Event Handling

- The modern approach to handling events is based on the **delegation event model**, which defines standard and consistent mechanisms to generate and process events.

- Its concept is quite simple: a *source* generates an event and sends it to one or more *listeners*.

- In this scheme, the listener simply waits until it receives an event.

- Once received, the listener processes the event and then returns.

- The advantage of this design is that the application logic that processes events is cleanly separated from the user interface logic that generates those events.

Events

- In the delegation model, an *event* is an object that describes a state change in a source.

- It can be generated as a consequence of a person interacting with the elements in a graphical user interface.

- Some of the activities that cause events to be generated are pressing a button, entering a character via the keyboard, selecting an item in a list, and clicking the mouse, etc.,

- Events may also occur that are not directly caused by interactions with a user interface.

- For example, an event may be generated when a timer expires, a counter exceeds a value, a software or hardware failure occurs, or an operation is completed.

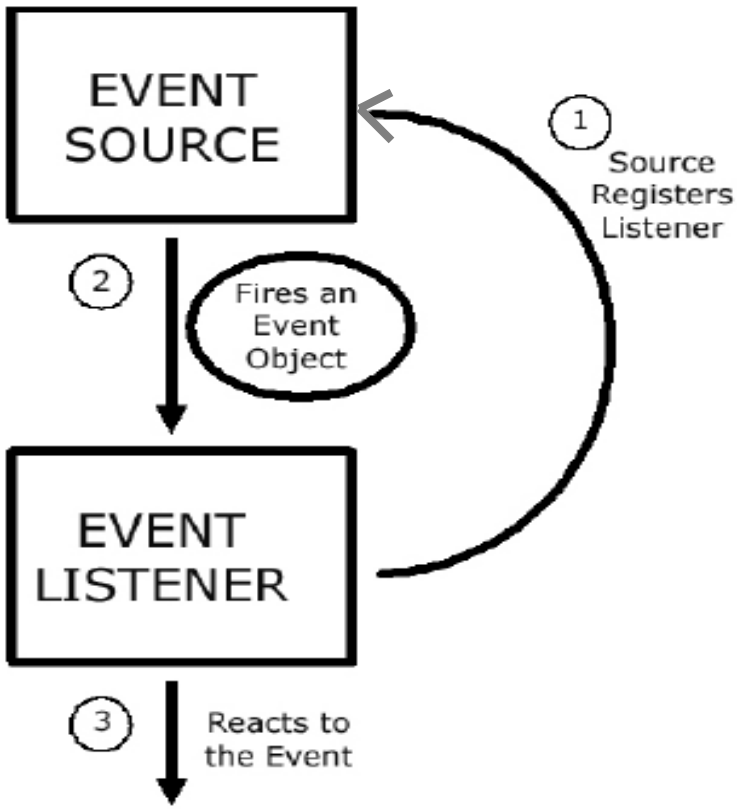
Event Listeners

- A *listener* is an object that is notified when an event occurs.
- It has two major requirements.
- First, it must have been registered with one or more sources to receive notifications about specific types of events.
- Second, it must implement methods to receive and process these notifications.
- The methods that receive and process events are defined in a set of interfaces found in **java.awt.event**

The Event Handling process

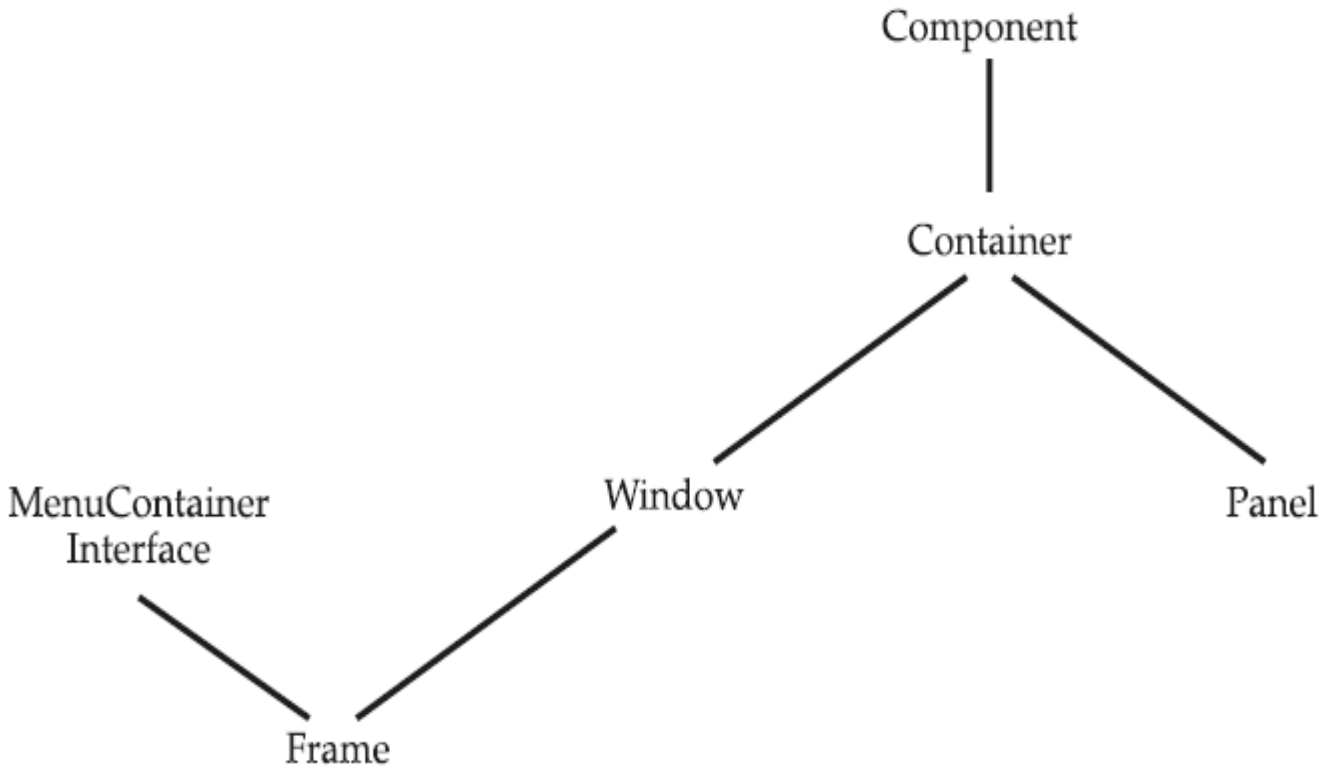
- When an event is triggered, the JAVA runtime first determines its source and type.
- If a listener for this type of event is registered with the source, an event object is created.
- For each listener to this type of an event, the JAVA runtime invokes the appropriate event handling method to the listener and passes the event object as the parameter.

The Event Handling Process (contd..)



Event Classes

- **EventObject** is a superclass of all events.
- **AWTEvent** is a superclass of all AWT events that are handled by the delegation event model. It is a subclass of EventObject.
- EventObject contains two methods: **getSource( ) and toString( ).**
- The getSource( ) method returns the source of the event.
- Its general form is shown here: Object getSource( )
- toString( ) returns the string equivalent of the event.
- The package java.awt.event defines several types of events that are generated by various user interface elements.
- Table enumerates the most important of these event classes and provides a brief description of when they are generated.


Component
Container
MenuContainer
Interface
Window
Panel
Frame

Main Event Classes in java.awt.event

Event Class	Description
ActionEvent	Generated when a button is pressed, a list item is double-clicked, or a menu item is selected.
AdjustmentEvent	Generated when a scroll bar is manipulated.
ComponentEvent	Generated when a component is hidden, moved, resized, or becomes visible.
ContainerEvent	Generated when a component is added to or removed from a container.
FocusEvent	Generated when a component gains or loses keyboard focus.
InputEvent	Abstract super class for all component input event classes.
ItemEvent	Generated when a check box or list item is clicked; also occurs when a choice selection is made or a checkable menu item is selected or deselected.

Main Event Classes in java.awt.event

Event Class	Description
KeyEvent	Generated when input is received from the keyboard.
MouseEvent	Generated when the mouse is dragged, moved, clicked, pressed, or released; also generated when the mouse enters or exits a component.
MouseWheelEvent	Generated when the mouse wheel is moved. (Added by Java 2, version 1.4)
TextEvent	Generated when the value of a text area or text field is changed.
WindowEvent	Generated when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.

The KeyEvent Class

- A KeyEvent is generated when keyboard input occurs.
- There are three types of key events, which are identified by these integer constants:
  - ✓KEY_PRESSED,
  - ✓KEY_RELEASED, and
  - ✓KEY_TYPED.
- The first two events are generated when any key is pressed or released.
- The last event occurs only when a character is generated.
- Remember, not all key presses result in characters.
- For example, pressing the SHIFT key does not generate a character.

- The KeyEvent class defines several methods, but the most commonly used ones are

- getKeyChar( ), which returns the character that was entered, and

- getKeyCode( ), which returns the key code.

- Their general forms are shown here:
  - ✓char getKeyChar( )
  - ✓int getKeyCode( )

- If no valid character is available, then getKeyChar( ) returns CHAR_UNDEFINED.

- When a KEY_TYPED event occurs, getKeyCode( ) returns VK_UNDEFINED.

The MouseEvent Class

There are eight types of mouse events. The MouseEvent class defines the following integer constants that can be used to identify them:

- ✓**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**

➢**The most commonly used methods in this class are getX( ) and getY( ).**

➢**These return the X and Y coordinates of the mouse when the event occurred. Their forms are shown here:**

- ✓**int getX( )**
- ✓**int getY( )**

Sources of Events

Event Source	Description
Button	Generates action events when the button is pressed.
Checkbox	Generates item events when the check box is selected or deselected.
Choice	Generates item events when the choice is changed.
List	Generates action events when an item is double-clicked; generates item events when an item is selected or deselected.
Menu Item	Generates action events when a menu item is selected; generates item events when a checkable menu item is selected or deselected.
Scrollbar	Generates adjustment events when the scroll bar is manipulated.
Text components	Generates text events when the user enters a character.
Window	Generates window events when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.

Event Listener Interfaces

Interface	Description
ActionListener	Defines one method to receive action events.
AdjustmentListener	Defines one method to receive adjustment events.
ComponentListener	Defines four methods to recognize when a component is hidden, moved, resized, or shown.
ContainerListener	Defines two methods to recognize when a component is added to or removed from a container.
FocusListener	Defines two methods to recognize when a component gains or loses keyboard focus.
ItemListener	Defines one method to recognize when the state of an item changes.
KeyListener	Defines three methods to recognize when a key is pressed, released, or typed.
MouseListener	Defines five methods to recognize when the mouse is clicked, enters a component, exits a component, is pressed, or is released.
MouseMotionListener	Defines two methods to recognize when the mouse is dragged or moved.
MouseWheelListener	Defines one method to recognize when the mouse wheel is moved. (Added by Java 2, version 1.4)
TextListener	Defines one method to recognize when a text value changes.
WindowFocusListener	Defines two methods to recognize when a window gains or loses input focus. (Added by Java 2, version 1.4)
WindowListener	Defines seven methods to recognize when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.

The KeyListener Interface

- 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.

- For example, if a user presses and releases the A key, three events are generated in sequence: key pressed, typed, and released.

- If a user presses and releases the HOME key, two key events are generated in sequence: key pressed and released.

- The general forms of these methods are shown here:

  - ✓void keyPressed(KeyEvent *ke*)
  - ✓void keyReleased(KeyEvent *ke*)
  - ✓void keyTyped(KeyEvent *ke*)

The MouseListener Interface

- **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.**

- **The general forms of these methods are shown here:**
  - ✓**void mouseClicked(MouseEvent *me*)**
  - ✓**void mouseEntered(MouseEvent *me*)**
  - ✓**void mouseExited(MouseEvent *me*)**
  - ✓**void mousePressed(MouseEvent *me*)**
  - ✓**void mouseReleased(MouseEvent *me*)**

The MouseMotionListener Interface

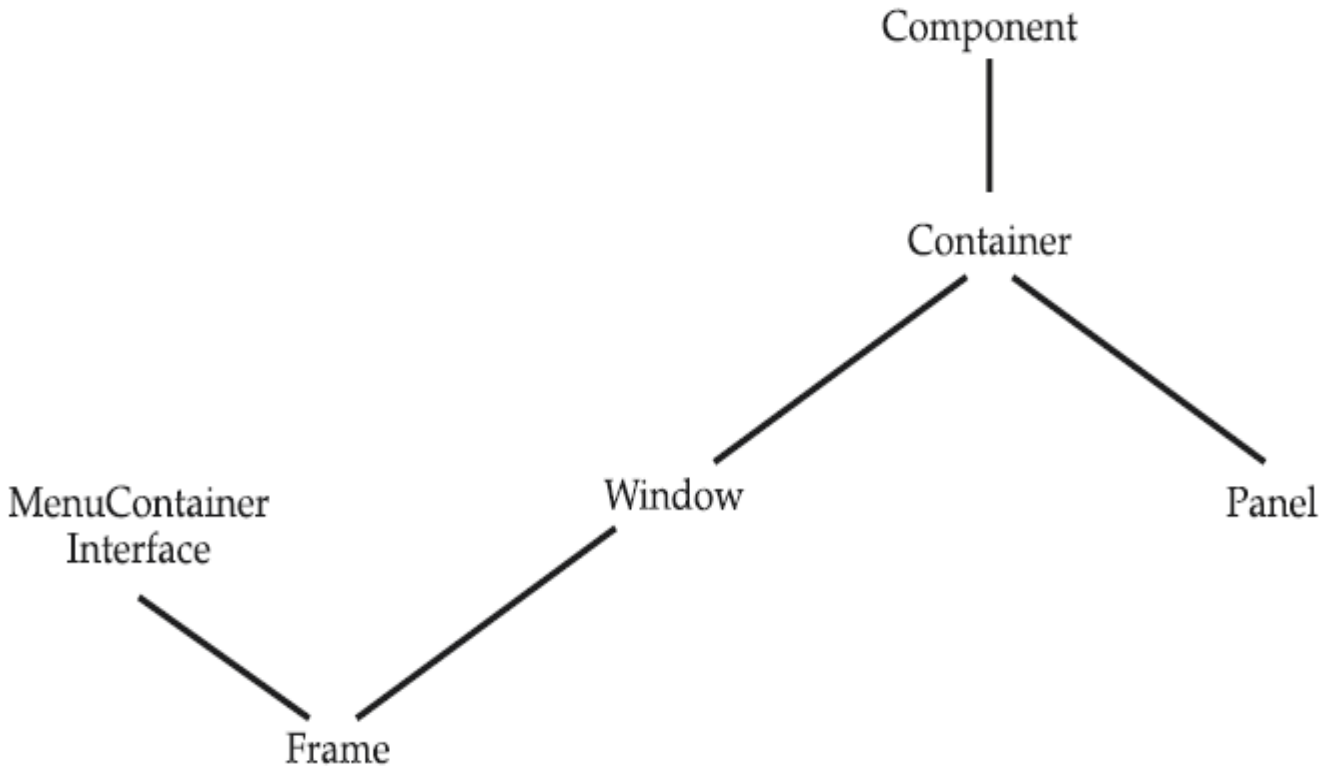
- **This interface defines two methods.**
- **The mouseDragged( ) method is called multiple times as the mouse is dragged.**
- **The mouseMoved( ) method is called multiple times as the mouse is moved.**
- **Their general forms are shown here:**
  - ✓ **void mouseDragged(MouseEvent *me*)**
  - ✓ **void mouseMoved(MouseEvent *me*)**

AWT Control Fundamentals

The AWT supports the following types of controls:

- **Labels**
- **Push buttons**
- **Check boxes**
- **Choice lists**
- **Lists**
- **Scroll bars**
- **Text Box**

These controls are subclasses of Component.


Component
Container
MenuContainer
Interface
Window
Panel
Frame