

GUI Programming with Java



Session 4
Event Handling



GUI Programming with JAVA

Session 4 – Event Handling

- We will look at...
 - Event Handling
 - Listeners





Event Handling

- Every time the user types a character or pushes a mouse button, an event occurs. Any object can be notified of the event.
- All it has to do is implement the appropriate interface and be registered as an *event listener* on the appropriate *event source*.
- Swing components can generate many kinds of events. Lets look at a few examples.



GUI Programming with JAVA

Session 4 – Event Handling

Examples of Event handling

Act that results in the event	Listener type
User clicks a button, presses Return while typing in a text field, or chooses a menu item	ActionListener
User closes a frame (main window)	WindowListener
User presses a mouse button while the cursor is over a component	MouseListener
User moves the mouse over a component	MouseMotionListener
Component becomes visible	ComponentListener
Component gets the keyboard focus	FocusListener
Table or list selection changes	ListSelectionListener

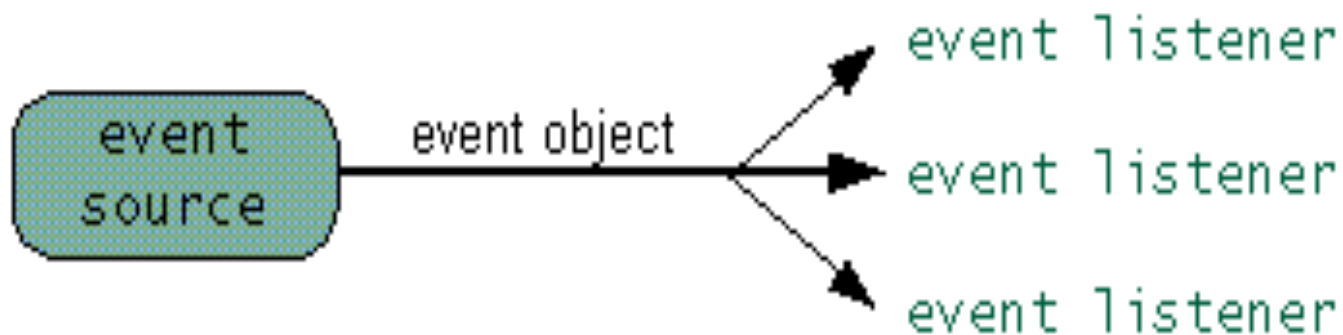


GUI Programming with JAVA

Session 4 – Event Handling

Event Handling

- Each event is represented by an object that gives information about the event and identifies the event source.
- Event sources are typically components, but other kinds of objects can also be event sources.
- As the following figure shows, each event source can have multiple listeners registered on it. Conversely, a single listener can register with multiple event sources.





GUI Programming with JAVA

Session 4 – Event Handling

How to Implement an Event Handler

- Every event handler requires three bits of code:
 1. In the declaration for the event handler class, code that specifies that the class either implements a listener interface or extends a class that implements a listener interface. For example:
`public class MyClass implements ActionListener {`
 2. Code that registers an instance of the event handler class as a listener upon one or more components. For example:
`someComponent.addActionListener(instanceOfMyClass);`
 3. Code that implements the methods in the listener interface. For example:
`public void actionPerformed(ActionEvent e) { ...//code that reacts to the action... }`



GUI Programming with JAVA

Session 4 – Event Handling

How to Implement an Event Handler (2)

- Let's investigate a typical event-handling scenario by looking at how buttons (JButton) handle mouse clicks.
- To detect when the user clicks an on-screen button (or does the keyboard equivalent), a program must have an object that implements the ActionListener interface.
- The program must register this object as an action listener on the button (the event source), using the addActionListener method.
- When the user clicks the on-screen button, the button fires an action event.
- This results in the invocation of the action listener's actionPerformed method (the only method in the ActionListener interface).
- The single argument to the method is an(ActionEvent) object that gives information about the event and its source.



Some simple event handing examples

- Here's the code that implements the event handling for the button.

```
public class Beeper ... implements ActionListener {  
    ...  
    //where initialization occurs:  
    button.addActionListener(this);  
    ...  
    public void actionPerformed(ActionEvent e) {  
        ...//Do Something  
    } //end of actionlistener  
  
    //end of class
```




Some simple event handing examples (2)

- Lets look at the sample program (myAction.java) in the sample 1 folder for this lecture.
- The myAction class implements the ActionListener interface, which contains one method: actionPerformed
- Since myAction implements ActionListener, a myAction object can register as a listener for the action events that buttons fire.
- Once the myAction has been registered using the Button addActionListener method, the myAction actionPerformed method is called every time the button is clicked.



A More Complex Example

- The event model, which you saw at its simplest in the previous example, is quite powerful and flexible.
- Any number of event listener objects can listen for all kinds of events from any number of event source objects.
- For example, a program might create one listener per event source. Or a program might have a single listener for all events from all sources.
- A program can even have more than one listener for a single kind of event from a single event source.



A More Complex Example (2)

- For a more complex sample of event handling please refer to the sample 2 folder - `complexAction.java`
- Any number of event listener objects can listen for all kinds of events from any number of event source objects.
- For example, a program might create one listener per event source. Or a program might have a single listener for all events from all sources.
- A program can even have more than one listener for a single kind of event from a single event source.



GUI Programming with JAVA

Session 4 – Event Handling

User Action	Source Object	Event Type Generated
Click a button	JButton	ActionEvent
Change Text	JTextComponent	TextEvent
Press return on a text field	JTextField	ActionEvent
Select a new item	JComboBox	ItemEvent, ActionEvent
Select Item(s)	JList	ListSelectionEvent
Click a check box	JCheckBox	ItemEvent, ActionEvent
Click a radio button	JRadioButton	ItemEvent, ActionEvent
Select a menu item	JMenuItem	ActionEvent
Move the scroll bar	JScrollBar	AdjustMent Event
Window opened, closed, iconified, deiconified, or closing	Window	WindowEvent
Component Added or removed from the container	Container	ContainerEvent
Component moved, resized, hidden or shown	Component	ComponentEvent
Component gained or lost focus	Component	Focus Event
Mouse Movement	Component	Mouse Event
Key released or pressed	Component	Key EVent



Listeners Supported by Swing Components

- You can tell what kinds of events a component can fire by looking at the kinds of event listeners you can register on it. For example, the Component class defines these listener registration methods:
 - addComponentListener
 - addFocusListener
 - addKeyListener
 - addMouseListener
 - addMouseMotionListener
- Thus, every component supports component, focus, key, mouse, and mouse-motion listeners. See the following link for more:
<http://download.oracle.com/javase/tutorial/uiswing/events/eventsandcomponents.html#many>
- However, a component fires only those events for which listeners have registered on it. For example, if a mouse listener is registered on a particular component, but the component has no other listeners, then the component will fire only mouse events--no component, focus, key, or mouse-motion events.



Common Listeners

- As well as the listeners that are common to all components, there are several listeners that are used by only certain components. These include
 - Action listener (e.g. called when we click on button)
 - Item Listener (e.g. called when we click on a radio button)
 - List Selection Listener (e.g. select an item in a list)
 - Window (e.g. called when we close a window)
- Others such as pop up menu listeners, hyperlink listeners and internal frame listeners will be examined later in this course.



How to Write an Action Listener

- Action listeners are probably the easiest -- and most common -- event handlers to implement.
- When the user clicks a button , chooses a menu item or presses Return in a text field , an action event occurs. The result is that an actionPerformed message is sent to all action listeners that are registered on the relevant component.
- We've already covered the action listener in sample 1. Lets have a quick review.



How to Write an Item Listener

- Item events are fired by components that implement the `ItemSelectable` interface.
- Generally, `ItemSelectable` components maintain on/off state for one or more items.
- The Swing components that fire item events include check boxes , check box menu items and combo boxes .
- For an example of item listening please refer to sample 3 java file `itemAction.java`



How to Write a List Selection Listener

- List selection events occur when the selection in a [list](#) or [table](#) is either changing or has just changed.
- As with our other event handlers we must implement a listener to handle events generated by the list. The listener we implement is `ListSelectionListener`

e.g `public class listSelection implements ListSelectionListener`

- To register the listener we use the `addListSelectionListener` method. This is used to detect when selections on the lists are altered.
- For an example of item listening please refer to sample 4 java file `listSelection.java`



How to Write a Window Listener

- Window events are fired by a window (such as a frame or dialog).
- just after the window is opened, closed, iconified, deiconified, activated, or deactivated.
 - *Opening* a window means showing it for the first time
 - *closing* it means removing the window from the screen
 - *Iconifying* it means substituting a small icon on the desktop for the window
 - *deiconifying* means the opposite
 - A window is *activated* if it or a component it contains has the keyboard focus
 - *deactivation* occurs when the window and all of its contents lose the keyboard focus
- If you want to be notified when a window is made visible or hidden, then you should register a component listener on the window.



How to Write a Window Listener(2)

- The most common use of window listeners is implementing custom window-closing behavior. For example, you might use a window listener to save data before closing the window, or to exit the program when the last window closes.
- You don't necessarily need to implement a window listener to specify what a window should do when the user closes it. By default, when the user closes a window the window becomes invisible.
- Another common use of window listeners is to stop threads and release resources when a window is iconified, and to start up again when the window is deiconified.
- For an example of item listening please refer to sample 5 java file WindowEventDemo.java



Summary

- Every time the user types a character or pushes a mouse button, an event occurs. Any object can be notified of the event.
- All it has to do is implement the appropriate interface and be registered as an *event listener* on the appropriate *event source*.
- Swing components can generate many kinds of events.
- Each event is represented by an object that gives information about the event and identifies the event source.
- Event sources are typically components, but other kinds of objects can also be event sources.



GUI Programming with JAVA

Session 4 – Event Handling

Summary (2)

- Every event handler requires three bits of code:
 1. In the declaration for the event handler class, code that specifies that the class either implements a listener interface or extends a class that implements a listener interface. For example:

```
public class MyClass implements ActionListener {
```
 2. Code that registers an instance of the event handler class as a listener upon one or more components. For example:

```
someComponent.addActionListener(instanceOfMyClass);
```
 1. Code that implements the methods in the listener interface. For example:

```
public void actionPerformed(ActionEvent e) { ...//code that reacts to the action... }
```



Summary (3)

- All components support the following listeners
 - `addComponentListener`
 - `addFocusListener`
 - `addKeyListener`
 - `addMouseListener`
 - `addMouseMotionListener`
- As well as the listeners that are common to all components, there are several listeners that are used by only certain components. These include
 - Action listener (e.g. called when we click on button)
 - Item Listener (e.g. called when we click on a radio button)
 - Window (e.g. called when we close a window)



GUI Programming with JAVA

Session 4 – Event Handling

Labwork 4:

Task 4-1

Create a simple JAVA application that contains three buttons on a window. One of the buttons should have an icon and one of the buttons should have a custom font applied to it. When any of the buttons is clicked a message should be shown on a JLabel showing the name of the button that was clicked (hint: setText() can be used to set the text in a label).

Task 4-2

Create a dropdown list with the following elements – Lion, Monkey, Elephant, Kangaroo, Polar Bear, Penguin, Ostrich. When an item is clicked show in a JLabel the name of the item that was clicked.

Task 4-3

Complete the task set in 3-3 (multiple choice question from last weeks labs) by providing feedback to the user. When the user selects the Dublin option show a label with the words “That’s Correct”. If they get it incorrect show a label “Sorry , the capital of Ireland is Dublin”.



GUI Programming with JAVA

Session 4 – Event Handling

END OF SESSION