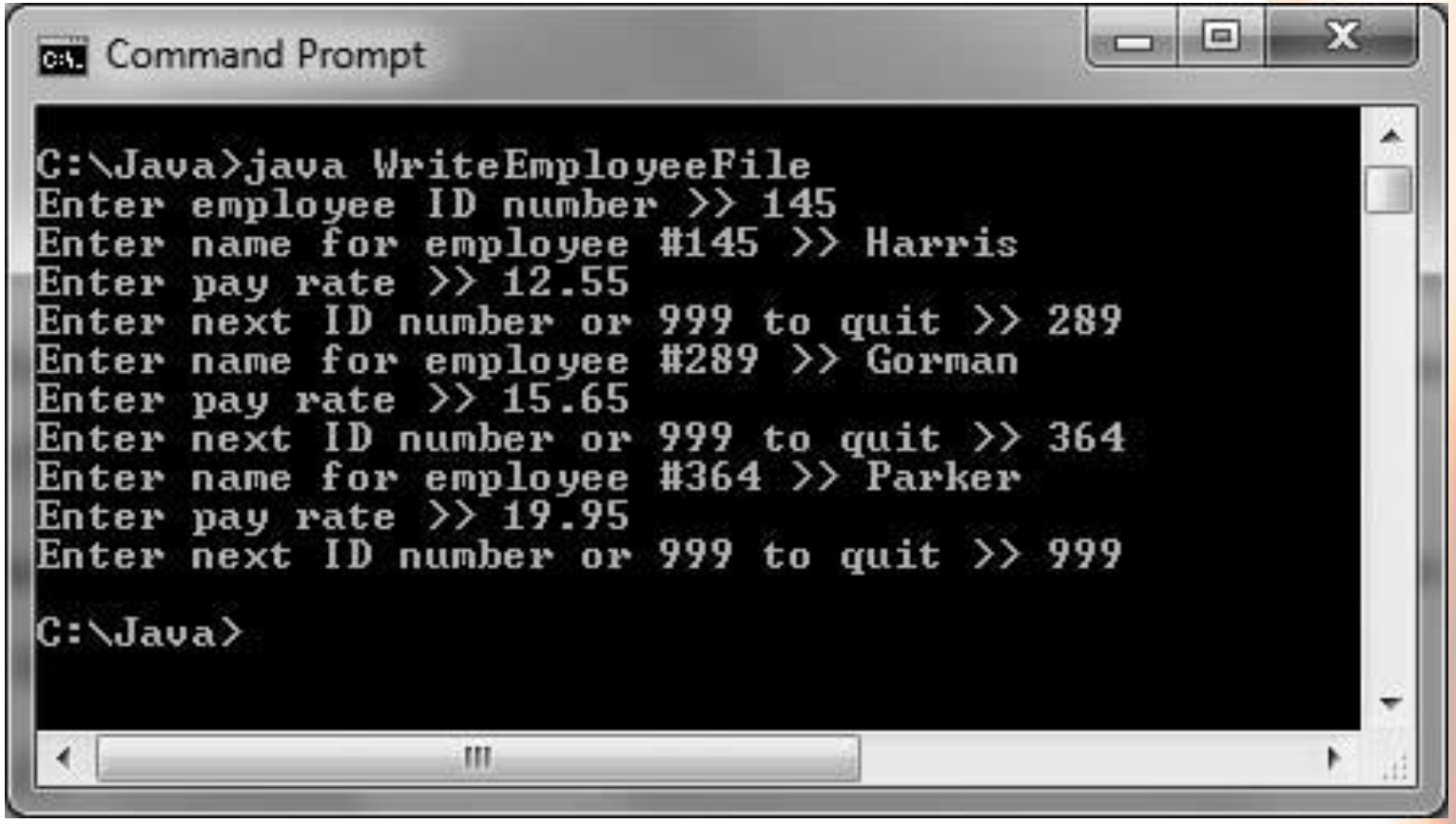
Ch(14): Event-Driven Programming lec (4)

Learning about Event-Driven Programming

- Event-driven program
  - User might initiate any number of events in any order
- Event
  - Occurs when a user takes action on a component, such as clicking the mouse on a `JButton` object
- Source
  - Component on which an event is generated
- Listener
  - Object that is interested in an event

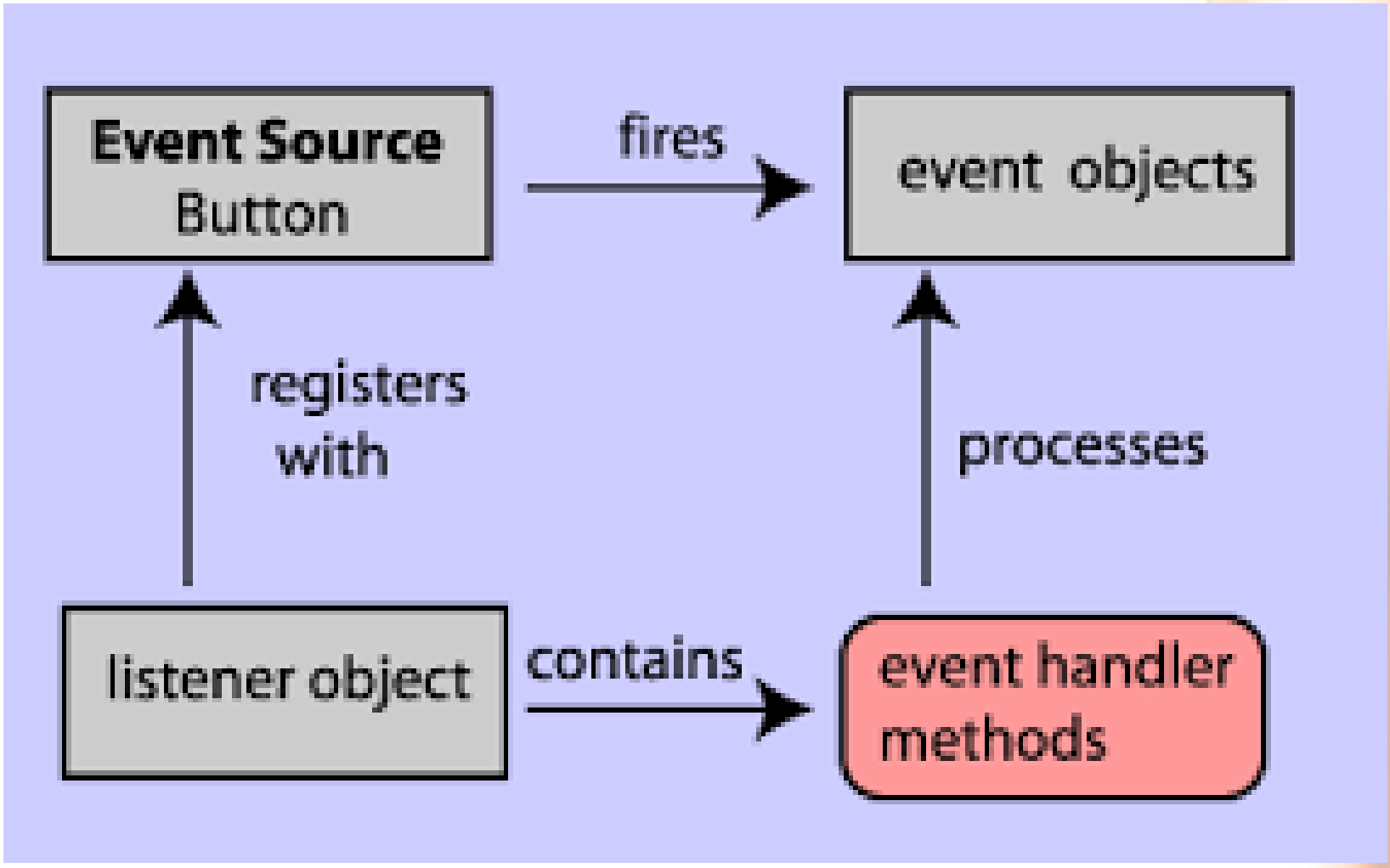
Learning about Event-Driven Programming



Learning about Event-Driven Programming

- If you want an object to be a listener for an event, you must register the object as a listener for the source.
- When the listener “receives the news,” an event-handling method contained in the listener object responds to the event.
- A source object and a listener object can be the same object. For example, you might program a Jbutton to change its own label when a user clicks it.

Learning about Event-Driven Programming



Learning about Event-Driven Programming (cont'd.)

- **Respond to user events within any class you create**
  - Prepare your class to accept event messages
  - Tell your class to expect events to happen
  - Tell your class how to respond to events

Preparing Your Class to Accept Event Messages

- Import the `java.awt.event` package
- Add the phrase `implements ActionListener` to the class header
- `ActionListener`
  - Standard event method specifications that allow your listener to work with `ActionEvents`

Telling Your Class to Expect Events to Happen

- Using `addActionListener()` registration method
  - `aButton.addActionListener(this);`
  - Causes any `ActionEvent` messages (button clicks) that come from `aButton` to be sent to “this current object”

Telling Your Class How to Respond to Events

- `ActionListener` interface
  - `actionPerformed(ActionEvent e)` method specification
    - Body contains any statements that you want to execute when the action occurs

Example demonstrates how to use event-driven

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class JHelloFrame extends JFrame implements ActionListener
{
   JLabel question = new JLabel("What is your name?");
   Font bigFont = new Font("Arial", Font.BOLD, 16);
   JTextField answer = new JTextField(10);
   JButton pressMe = new JButton("Press me");
   JLabel greeting = new JLabel("");
   final int WIDTH = 175;
   final int HEIGHT = 225;
   public JHelloFrame()
   {
      super("Hello Frame");
      setSize(WIDTH, HEIGHT);
      setLayout(new FlowLayout());
      question.setFont(bigFont);
      greeting.setFont(bigFont);
      add(question);
      add(answer);
      add(pressMe);
      add(greeting);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      pressMe.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e)
    {
       String name = answer.getText();
       String greet = "Hello, " + name;
       greeting.setText(greet);
    }
}
```

Example demonstrates how to use event-driven

```
public class JHelloDemo
{
   public static void main(String[] args)
   {
      JHelloFrame frame = new JHelloFrame();
      frame.setVisible(true);
   }
}
```

Determine the source of the event

- When more than one component is added and registered to a `JFrame`
  - Necessary to determine which component was used
  - Find source of the event using `getSource();`

```
void actionPerformed(ActionEvent e)
{
   Object source = e.getSource();
   if(source == option1)
      //execute these statements when user clicks option1
   else
      //execute these statements when user clicks any other option
}
```

Figure 14-27 An `actionPerformed()` method that takes one of two possible actions

Determine the source of the event

- The instanceof keyword is used when it is necessary to know only the component's type.

```
void actionPerformed(ActionEvent e)
{
   Object source = e.getSource();
   if(source instanceof JTextField)
   {
     // execute these statements when any JTextField
     // generates the event
     // but not when a JButton or other Component does
   }
}
```

Figure 14-28 An `actionPerformed()` method that executes a block of statements when a user generates an event from any `JTextField`

Using the `setEnabled()` Method

- Components are enabled by default
- you can use the setEnabled() method to make a component available or unavailable by passing true or false to it, respectively.

Understanding `Swing` Event Listeners

- Classes that respond to user-initiated events
  - Must implement interface that deals with events
  - Called event listeners
- Many types of listeners exist in Java
  - Each can handle specific event type
- Class can implement as many event listeners as it needs
- Event occurs every time user types character or clicks mouse button

Understanding `Swing` Event Listeners (cont'd.)

Listener	Type of Events	Example
`ActionListener`	Action events	Button clicks
`AdjustmentListener`	Adjustment events	Scroll bar moves
`ChangeListener`	Change events	Slider is repositioned
`FocusListener`	Keyboard focus events	Text field gains or loses focus
`ItemListener`	Item events	Check box changes status
`KeyListener`	Keyboard events	Text is entered
`MouseListener`	Mouse events	Mouse clicks
`MouseMotionListener`	Mouse movement events	Mouse rolls
`WindowListener`	Window events	Window closes

Table 14-2 Alphabetical list of some event listeners

Understanding Swing Event Listeners (cont'd.)

- Create relationships between Swing components and classes that react to users' manipulations of them
- JCheckBox responds to user's clicks
  - addItemListener() method
  - Register JCheckBox as type of object that can create ItemEvent
- Format:

```
theSourceOfTheEvent.addListenerMethod
(theClassThatShouldRespond);
```

Understanding `Swing` Event Listeners (cont'd.)

Component(s)	Associated Listener-Registering Method(s)
`JButton`, `JCheckBox`, `JComboBox`, `JTextField`, and `JRadioButton`	`addActionListener()`
`JScrollBar`	`addAdjustmentListener()`
All `Swing` components	`addFocusListener()`, `addKeyListener()`, `addMouseListener()`, and `addMouseMotionListener()`
`JButton`, `JCheckBox`, `JComboBox`, and `JRadioButton`	`addItemListener()`
All `JWindow` and `JFrame` components	`addWindowListener()`
`JSlider` and `JCheckBox`	`addChangeListener()`

Table 14-3 Some `Swing` components and their associated listener-registering methods

Understanding `Swing` Event Listeners (cont'd.)

- Class of object that responds to event
  - Contains method that accepts event object created by user's action
  - Specific methods react to specific event types

Listener	Method
`ActionListener`	`actionPerformed(ActionEvent)`
`AdjustmentListener`	`adjustmentValueChanged(AdjustmentEvent)`
`FocusListener`	`focusGained(FocusEvent)` and `focusLost(FocusEvent)`
`ItemListener`	`itemStateChanged(ItemEvent)`

Table 14-4 Selected methods that respond to events