Events: Mouse, Keyboard

Event-Driven Programming

- *Procedural programming* is executed in procedural order.
- In *event-driven programming*, code is executed upon activation of events.

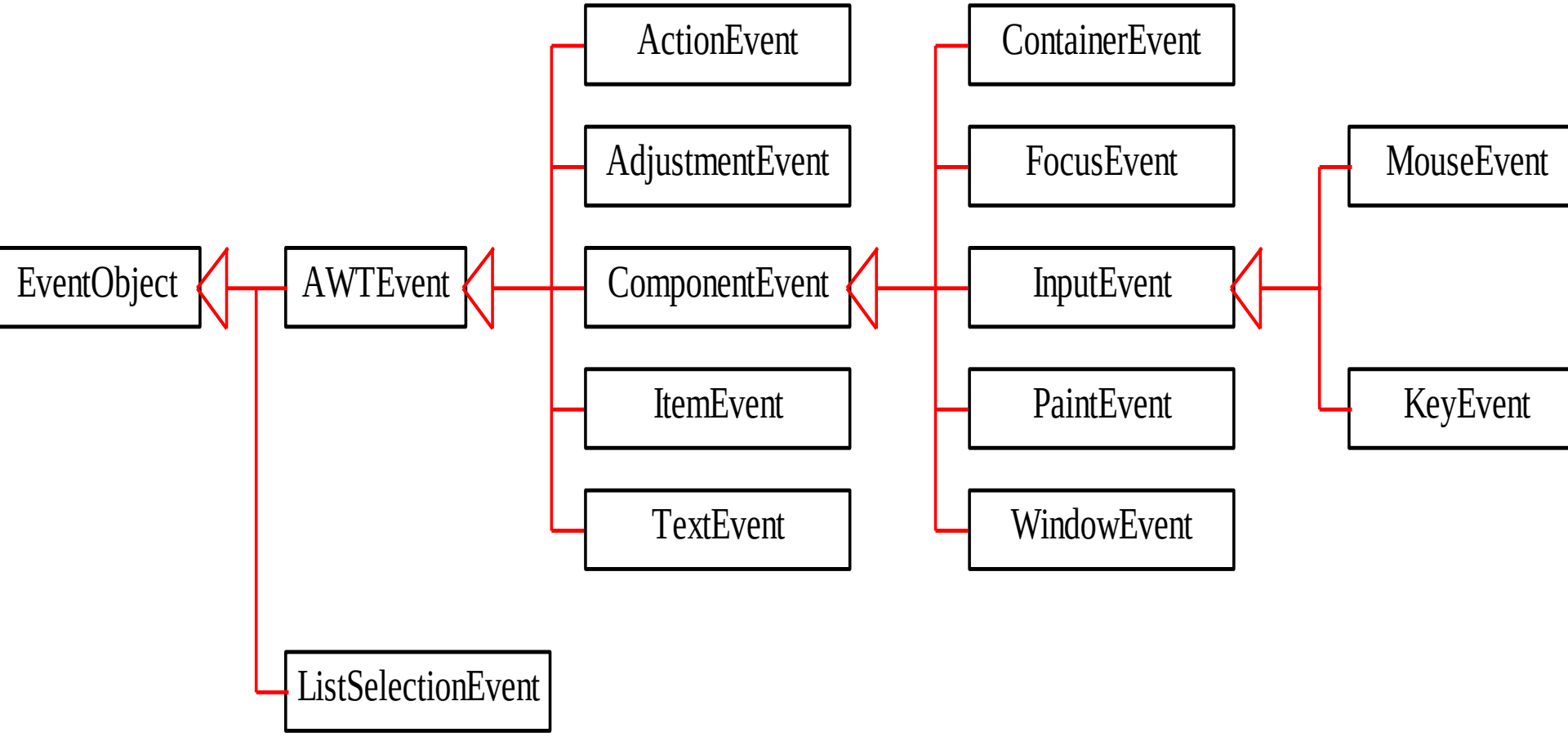
Events

- An *event* can be defined as a type of signal to the program that something has happened.
- The event is generated by external user actions such as mouse movements, mouse button clicks, and keystrokes, or by the operating system, such as a timer.

Event Information

- `id`: A number that identifies the event.
- `target`: The source component upon which the event occurred.
- `arg`: Additional information about the source components.
- `x, y coordinates`: The mouse pointer location when a mouse movement event occurred.
- `clickCount`: The number of consecutive clicks for the mouse events. For other events, it is zero.
- `when`: The time stamp of the event.
- `key`: The key that was pressed or released.

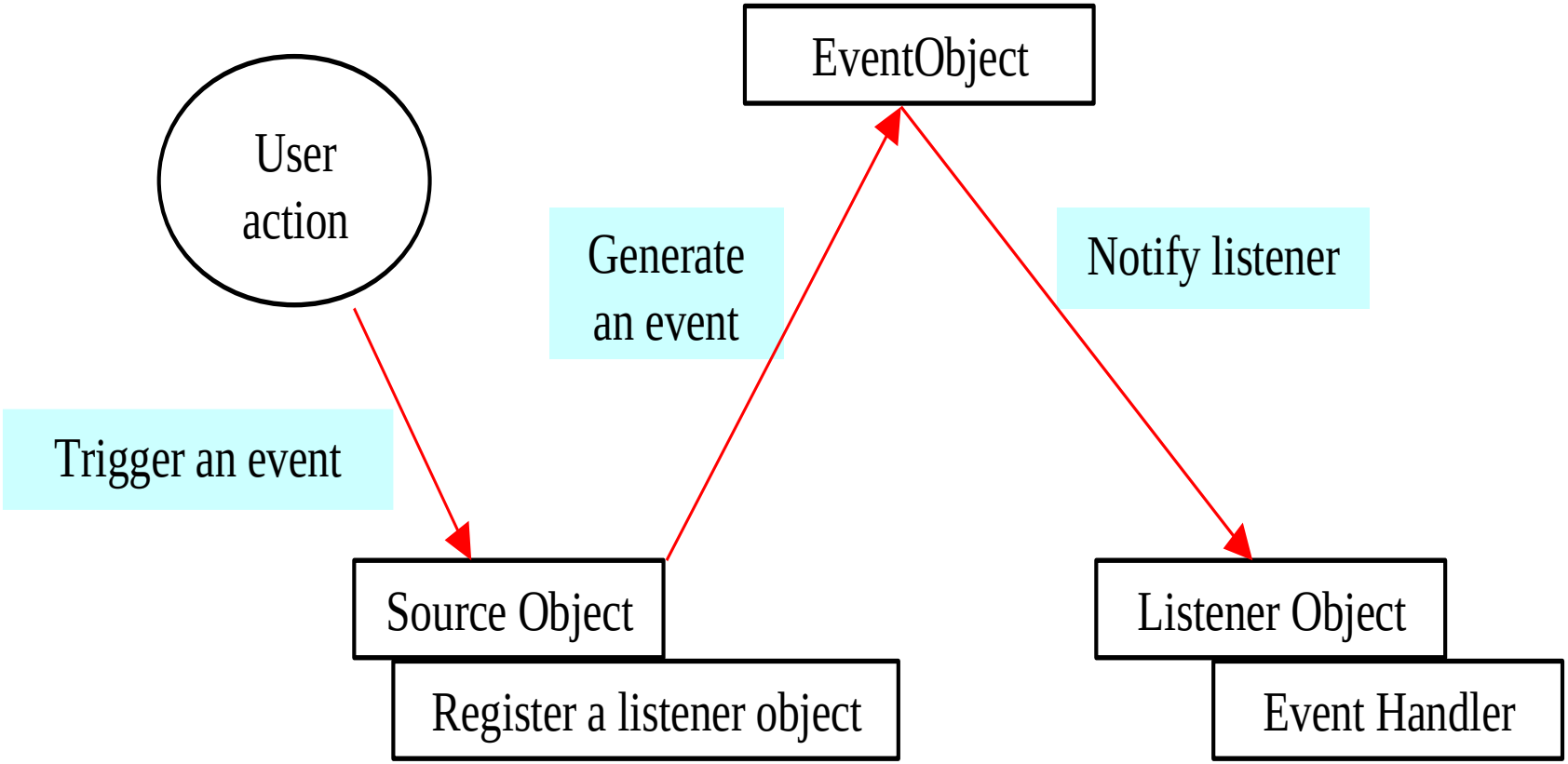
Event Classes



Selected User Actions

User Action	Source Object	Event Type Generated
Clicked on a button	JButton	ActionEvent
Changed text	JTextComponent	TextEvent
Double-clicked on a list item	JList	ActionEvent
Selected or deselected an item with a single click	JList	ItemEvent
Selected or deselected an item	JComboBox	ItemEvent

The Delegation Model



Selected Event Handlers

Event Class	Listener Interface	Listener Methods (Handlers)
ActionEvent	ActionListener	actionPerformed(ActionEvent)
ItemEvent	ItemListener	itemStateChanged(ItemEvent)
WindowEvent	WindowListener	windowClosing(WindowEvent)
		windowOpened(WindowEvent)
		windowIconified(WindowEvent)
		windowDeiconified(WindowEvent)
		windowClosed(WindowEvent)
		windowActivated(WindowEvent)
		windowDeactivated(WindowEvent)
ContainerEvent	ContainerListener	componentAdded(ContainerEvent)
		componentRemoved(ContainerEvent)

Listeners

- Listener methods take a corresponding type of event as an argument.
- Event objects have useful methods. For example, getSource returns the object that produced this event.
- A MouseEvent has methods getX, getY.

Mouse Events

- Mouse events are captured by an object which is a MouseListener and possibly a MouseMotionListener.
- A mouse listener is usually attached to a JPanel component.
- It is not uncommon for a panel to serve as its own mouse listener:

```
addMouseListener(this);
addMouseMotionListener(this); // optional
```

Mouse Events (cont'd)

- Mouse listener methods receive a MouseEvent object as a parameter.
- A mouse event can provide the coordinates of the event and other information:

```
public void mousePressed(MouseEvent e)
{
    int x = e.getX();
    int y = e.getY();
    int clicks = e.getClickCount();
}
```

Mouse Events (cont'd)

- The MouseListener interface defines five methods:

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

Called when the mouse cursor enters/exits component's visible area

- One click and release causes several calls. Using only mouseReleased is usually a safe bet.

Mouse Events (cont'd)

- The MouseMotionListener interface adds two methods:

```
void mouseMoved (MouseEvent e)
void mouseDragged (MouseEvent e)
```

Called when the mouse has moved with a button held down

- These methods are usually used together with MouseListener methods (i.e., a class implements both interfaces).

Event Handling Strategies: Pros and Cons

- Separate Listener
  - Advantages
    - Can extend adapter and thus ignore unused methods
    - Separate class easier to manage
  - Disadvantage
    - Need extra step to call methods in main window
- Main window that implements interface
  - Advantage
    - No extra steps needed to call methods in main window
  - Disadvantage
    - Must implement methods you might not care about

Event Handling Strategies: Pros and Cons, cont.

- Named inner class
  - Advantages
    - Can extend adapter and thus ignore unused methods
    - No extra steps needed to call methods in main window
  - Disadvantage
    - A bit harder to understand
- Anonymous inner class
  - Advantages
    - Same as named inner classes
    - Even shorter
  - Disadvantage
    - Much harder to understand

Event Listeners

Event Listeners

- Listener objects is added to Button
- When the user clicks the button,
- Button object generates an ActionEvent object.
- This calls the listener object's **method** and passes the ActionEvent object generated.

How to Attach an Event Listener to an Event Source

- o is an event source
- h is an event listener of type XXX

o.addXXX(h)

where XXX is one of the following:

- ActionListener
- MouseListener
- MouseMotionListener
- KeyListener
- WindowListener
- ComponentListener
- FocusListener
- TextListener
- AdjustmentListener
- ItemListener

Registering Event Listeners

- To register a listener object with a source object, you use lines of code that follow the model

```
source.addEventListener(eventListenerObject
);
```

The ActionListener Interface

```
interface ActionListener {
    public  void actionPerformed(ActionEvent e);
}
```

version 1

```
JButton hw = new JButton("Hello World!");
    panel.add(hw);

    hw.addActionListener(new ActionListener(){
      public void actionPerformed(ActionEvent e){
        System.exit(0);      }    });
```

version 2

```
class MyFrame extends JFrame implements ActionListener
{  public MyFrame(){
        JButton hw = new JButton("Hello World!");
        panel.add(hw);
        hw.addActionListener(this);


   }

  public void actionPerformed(ActionEvent o){
          System.exit(0);
  }
}
```

version 3

```
class MyFrame extends Jframe  {
   Button hw;
{  public MyFrame(){
       JButton hw = new JButton("Hello World!");
       panel.add(hw);
       hw.addActionListener(new MyActionListener());   }
}
}

class MyActionListener implements ActionListener {
           public void actionPerformed(ActionEvent o){
                System.exit(0);
           }
}
```

Implementing an Event Handler

- Implement a listener interface or extend a class that implements a listener interface.
- Register an instance of the event handler class as a listener upon one or more components.
- Implement the methods in the listener interface to handle the event.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class ButtonListener implements ActionListener {
   public void actionPerformed (ActionEvent e) {
      System.out.println ("Got a button press:” + e);
   }
}

public class Main {
   private static void showGUI() {
      JFrame frame = new JFrame("Swing GUI");
      java.awt.Container content = frame.getContentPane();
      content.setLayout(new FlowLayout());
      content.add(new JLabel ("Yo!"));
      JButton button = new JButton ("Click Me");
        button.addActionListener(new
ButtonListener());
      content.add(button);
      frame.pack();
      frame.setVisible(true);
```

```
class ButtonListener implements ActionListener {
   public void actionPerformed (ActionEvent e) {
      if (e.getActionCommand().equals ("On")) {
         System.out.println("On!");
      } else if (e.getActionCommand().equals("Off")) {
         System.out.println("Off!");
      } else {
         System.out.println("Unrecognized button press!"); } } }
```


```
public class main {
   private static void showGUI() {
         …
         ButtonListener bl = new ButtonListener();
         JButton onButton = new JButton ("On");
         onButton.addActionListener(bl);
         content.add(onButton);
         JButton offButton = new JButton ("Off");
         offButton.addActionListener(bl);
         content.add(offButton);
```

Keyboard Events

- Keyboard events are captured by an object which is a KeyListener.
- A key listener object must first obtain keyboard “focus.” This is done by calling the component’s requestFocus method.
- If keys are used for moving objects (as in a drawing program), the “canvas” panel may serve as its own key listener:

```
addKeyListener(this);
```

Keyboard Events (cont'd)

- The KeyListener interface defines three methods:

```
void keyPressed (KeyEvent e)
void keyReleased (KeyEvent e)
void keyTyped (KeyEvent e)
```

- One key pressed and released causes several calls.

Keyboard Events (cont'd)

- Use keyTyped to capture character keys (i.e., keys that correspond to printable characters).
- e.getKeyChar() returns a char, the typed character:

```
public void keyTyped (KeyEvent  e)
{
   char ch = e.getKeyChar();
   if (ch == 'A')
       ...
}
```

Keyboard Events (cont'd)

- Use keyPressed or keyReleased to handle "action" keys, such as cursor keys, <Enter>, function keys, and so on.
- e.getKeyCode() returns an int, the key's "virtual code."
- The KeyEvent class defines constants for numerous virtual keys. For example:

VK_LEFT, VK_RIGHT, VK_UP, VK_DOWN	Cursor keys
VK_HOME, VK_END, VK_PAGE_UP, ...etc.	Home, etc.

Keyboard Events (cont'd)

- e.isShiftDown(), e.isControlDown(), e.isAltDown() return the status of the respective modifier keys.
- e.getModifiers() returns a bit pattern that represents the status of all modifier keys.
- KeyEvent defines "mask" constants CTRL_MASK, ALT_MASK, SHIFT_MASK, and so on.

Standard AWT Event Listeners (Summary)

Why no adapter class?

Listener	Adapter Class (If Any)	Registration Method
ActionListener		addActionListener
AdjustmentListener		addAdjustmentListener
ComponentListener	ComponentAdapter	addComponentListener
ContainerListener	ContainerAdapter	addContainerListener
FocusListener	FocusAdapter	addFocusListener
ItemListener		addItemListener
KeyListener	KeyAdapter	addKeyListener
MouseListener	MouseAdapter	addMouseListener
MouseMotionListener	MouseMotionAdapter	addMouseMotionListener
TextListener		addTextListener
WindowListener	WindowAdapter	addWindowListener

Standard AWT Event Listeners (Details)

- ActionListener
  - Handles buttons and a few other actions
    - actionPerformed(ActionEvent event)
- AdjustmentListener
  - Applies to scrolling
    - adjustmentValueChanged(AdjustmentEvent event)
- ComponentListener
  - Handles moving/resizing/hiding GUI objects
    - componentResized(ComponentEvent event)
    - componentMoved (ComponentEvent event)
    - componentShown(ComponentEvent event)
    - componentHidden(ComponentEvent event)

Standard AWT Event Listeners (Details Continued)

- ContainerListener
  - Triggered when window adds/removes GUI controls
    - componentAdded(ContainerEvent event)
    - componentRemoved(ContainerEvent event)
- FocusListener
  - Detects when controls get/lose keyboard focus
    - focusGained(FocusEvent event)
    - focusLost(FocusEvent event)

Standard AWT Event Listeners (Details Continued)

- ItemListener
  - Handles selections in lists, checkboxes, etc.
    - itemStateChanged(ItemEvent event)
- KeyListener
  - Detects keyboard events
    - keyPressed(KeyEvent event) -- any key pressed down
    - keyReleased(KeyEvent event) -- any key released
    - keyTyped(KeyEvent event) -- key for printable char released

Standard AWT Event Listeners (Details Continued)

- MouseListener
  - Applies to basic mouse events
    - mouseEntered(MouseEvent event)
    - mouseExited(MouseEvent event)
    - mousePressed(MouseEvent event)
    - mouseReleased(MouseEvent event)
    - mouseClicked(MouseEvent event) -- Release without drag
      - Applies on release if no movement since press
- MouseMotionListener
  - Handles mouse movement
    - mouseMoved(MouseEvent event)
    - mouseDragged(MouseEvent event)