

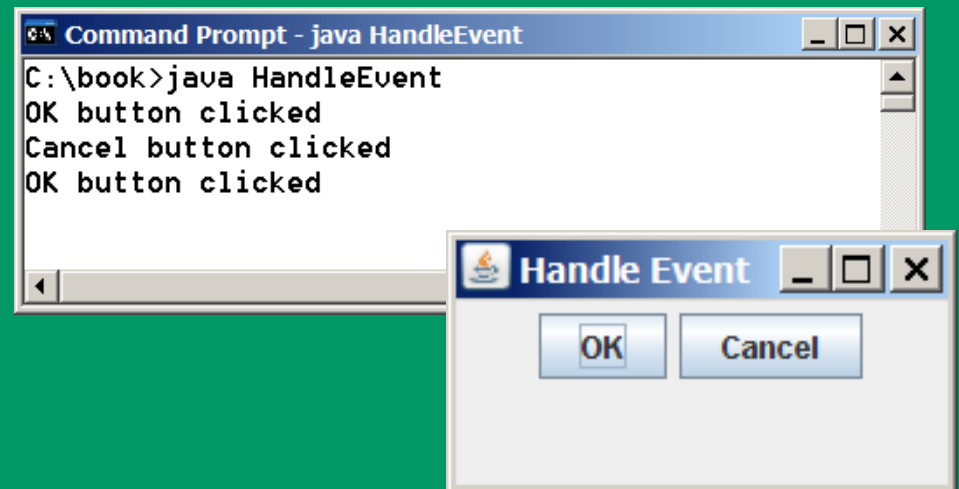
Chapter 16 Event-Driven Programming

Procedural vs. Event-Driven Programming

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

Taste of Event-Driven Programming

- ☞ The example displays a button in the frame. A message is displayed on the console when a button is clicked.



Handling GUI Events

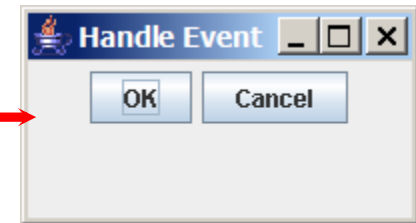
Source object (e.g., button)

Listener object contains a method for processing the event.

Trace Execution

```
public class HandleEvent extends JFrame {  
    public HandleEvent() {  
        ...  
        OKListenerClass listener1 = new OKListenerClass();  
        jbtOK.addActionListener(listener1);  
        ...  
    }  
  
    public static void main(String[] args) {  
        ...  
    }  
}
```

1. Start from the main method to create a window and display it



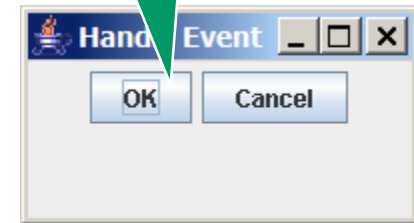
```
class OKListenerClass implements ActionListener {  
    public void actionPerformed(ActionEvent e) {  
        System.out.println("OK button clicked");  
    }  
}
```

Trace Execution

```
public class HandleEvent extends JFrame {  
    public HandleEvent() {  
        ...  
        OKListenerClass listener1 = new OKListenerClass();  
        jbtOK.addActionListener(listener1);  
        ...  
    }  
  
    public static void main(String[] args) {  
        ...  
    }  
}
```

```
class OKListenerClass implements ActionListener {  
    public void actionPerformed(ActionEvent e) {  
        System.out.println("OK button clicked");  
    }  
}
```

2. Click OK

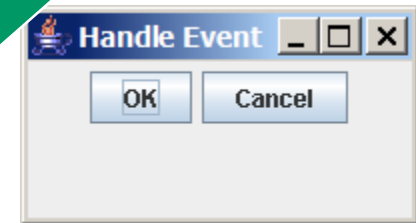


Trace Execution

```
public class HandleEvent extends JFrame {  
    public HandleEvent() {  
        ...  
        OKListenerClass listener1 = new OKListenerClass();  
        jbtOK.addActionListener(listener1);  
        ...  
    }  
  
    public static void main(String[] args) {  
        ...  
    }  
}
```

```
class OKListenerClass implements ActionListener {  
    public void actionPerformed(ActionEvent e) {  
        System.out.println("OK button clicked");  
    }  
}
```

3. Click OK. The JVM invokes the listener's actionPerformed method



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 clicks, and keystrokes, or by the operating system, such as a timer.

Selected User Actions

User Action	Source Object	Event Type Generated
Click a button	JButton	ActionEvent
Click a check box	JCheckBox	ItemEvent, ActionEvent
Click a radio button	JRadioButton	ItemEvent, ActionEvent
Press return on a text field	JTextField	ActionEvent
Select a new item	JComboBox	ItemEvent, ActionEvent
Window opened, closed, etc.	Window	WindowEvent
Mouse pressed, released, etc.	Component	MouseEvent
Key released, pressed, etc.	Component	KeyEvent

The Delegation Model: Example

```
JButton jbt = new JButton("OK");  
ActionListener listener = new OKListener();  
jbt.addActionListener(listener);
```

Selected Event Handlers

Event Class

ActionEvent
ItemEvent
WindowEvent

Listener Interface

ActionListener
ItemListener
WindowListener

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)
MouseEvent	MouseListener	mousePressed (MouseEvent) mouseReleased (MouseEvent) mouseClicked (MouseEvent) mouseExited (MouseEvent) mouseEntered (MouseEvent)
KeyEvent	KeyListener	keyPressed (KeyEvent) keyReleased (KeyEvent) keyTyped (KeyEvent)

MouseEvent

java.awt.event.InputEvent

+getWhen(): long
+isAltDown(): boolean
+isControlDown(): boolean
+isMetaDown(): boolean
+isShiftDown(): boolean

Returns the timestamp when this event occurred.

Returns whether or not the Alt modifier is down on this event.

Returns whether or not the Control modifier is down on this event.

Returns whether or not the Meta modifier is down on this event

Returns whether or not the Shift modifier is down on this event.

java.awt.event.MouseEvent

+getButton(): int
+getClickCount(): int
+getPoint(): java.awt.Point
+getX(): int
+getY(): int

Indicates which mouse button has been clicked.

Returns the number of mouse clicks associated with this event.

Returns a Point object containing the x and y coordinates.

Returns the x-coordinate of the mouse point.

Returns the y-coordinate of the mouse point.

```
private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {  
    System.out.println(evt.isAltDown());  
}
```

Handling Mouse Events

java.awt.event.MouseListener

+mousePressed(e: MouseEvent): void

Invoked when the mouse button has been pressed on the source component.

+mouseReleased(e: MouseEvent): void

Invoked when the mouse button has been released on the source component.

+mouseClicked(e: MouseEvent): void

Invoked when the mouse button has been clicked (pressed and released) on the source component.

+mouseEntered(e: MouseEvent): void

Invoked when the mouse enters the source component.

+mouseExited(e: MouseEvent): void

Invoked when the mouse exits the source component.

java.awt.event.MouseMotionListener

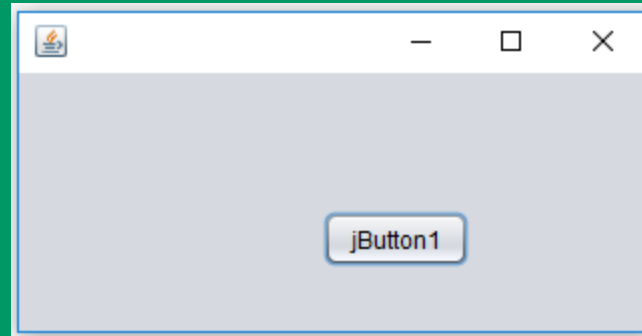
+mouseDragged(e: MouseEvent): void

Invoked when a mouse button is moved with a button pressed.

+mouseMoved(e: MouseEvent): void

Invoked when a mouse button is moved without a button pressed.

Example



```
private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {  
    System.out.println(evt.isAltDown());  
}
```

Handling Keyboard Events

To process a keyboard event, use the following handlers in the `KeyListener` interface:

➔ `keyPressed(KeyEvent e)`

Called when a key is pressed.

➔ `keyReleased(KeyEvent e)`

Called when a key is released.

➔ `keyTyped(KeyEvent e)`

Called when a key is pressed and then released.

The KeyEvent Class

☞ Methods:

`getKeyChar()` method

`getKeyCode()` method

☞ Keys:

Home `VK_HOME`

End `VK_END`

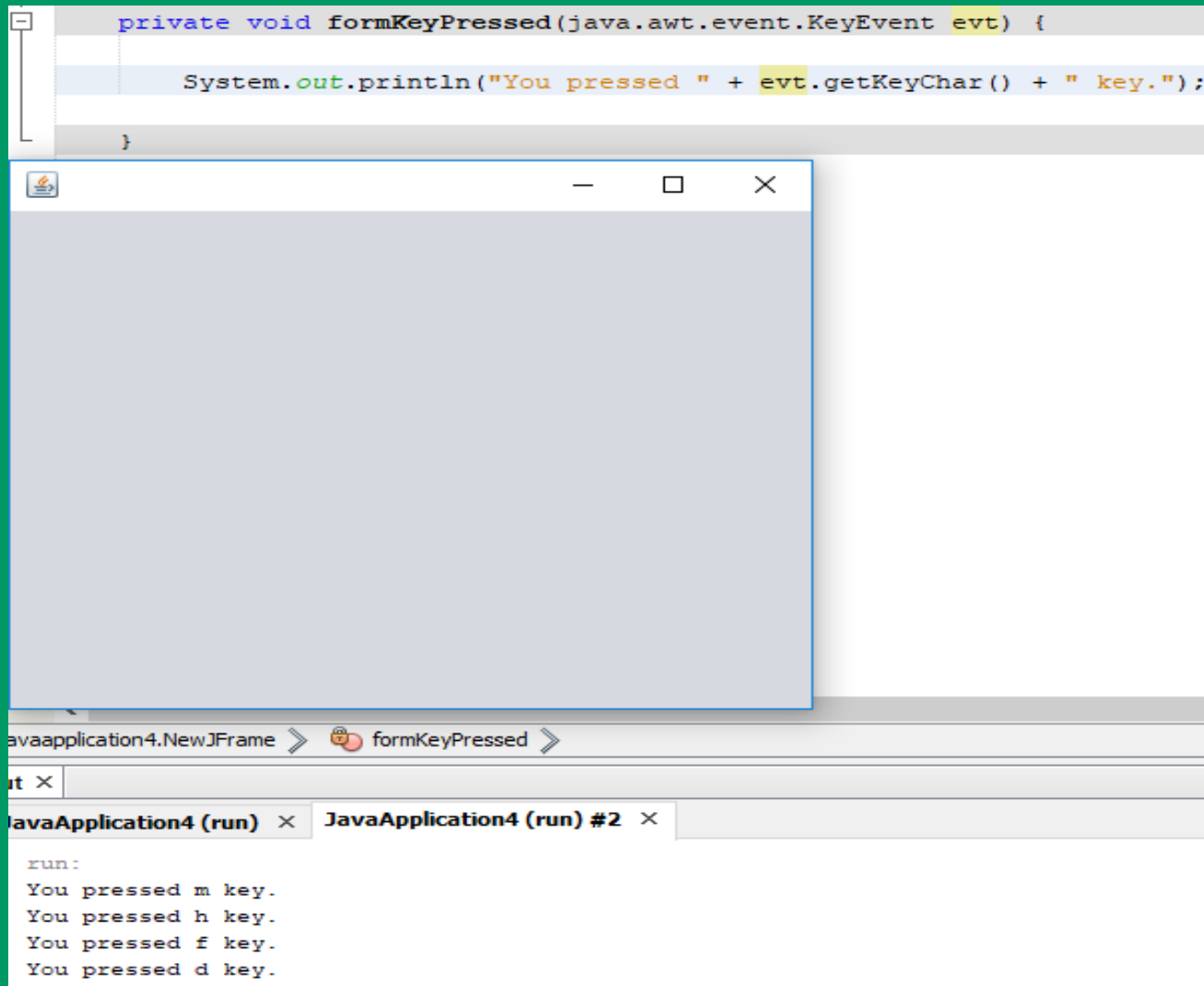
Page Up `VK_PGUP`

Page Down `VK_PGDN`

etc...

Example: Keyboard Events Demo

```
private void formKeyPressed(java.awt.event.KeyEvent evt) {  
    System.out.println("You pressed " + evt.getKeyChar() + " key.");  
}
```



The screenshot displays an IDE environment. At the top, a code editor shows a Java method `formKeyPressed` that prints the character of a pressed key. Below the code, a Java Swing window titled `formKeyPressed` is visible, which is currently empty. At the bottom, the console window shows the output of the program, indicating that the keys 'm', 'h', 'f', and 'd' were pressed in sequence.

```
run:  
You pressed m key.  
You pressed h key.  
You pressed f key.  
You pressed d key.
```