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 the object has to do is implement the appropriate interface and be registered as an event listener on the appropriate event source.

How to Implement an Event Handler

Every event handler requires three pieces of code

• In the declaration for the event handler class, one line of code specifies that the class either implements a listener interface or extends a class that implements a listener interface. For example:

```java
public class MyClass implements ActionListener {
```

• Another line of code registers an instance of the event handler class as a listener on one or more components. For example:

```java
someComponent.addActionListener(instanceOfMyClass);
```

• In the event handler class, a few lines of code implement the methods in the listener interface. For example:

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

Event handlers can be instances of any class. Often an event handler that has only a few lines of code is implemented using an anonymous inner class--an unnamed class defined inside of another class. Anonymous inner classes can be confusing at first, but once you're used to them, they make the code clearer by keeping the implementation of an event handler close to where the event handler is registered.

SwingApplication has two event handlers. One handles window closing (window events); the other handles button clicks (action events). We've already seen the window-closing code. Here is the code that handles button clicks in the SwingApplication:
In general, 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.

Swing components can generate many kinds of events. The following table lists a few examples.
### 1. User action, source object, and event type

<table>
<thead>
<tr>
<th>User Action</th>
<th>Source Object</th>
<th>Event Type Generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click button</td>
<td>JButton</td>
<td>ActionEvent</td>
</tr>
<tr>
<td>Change text</td>
<td>JTextComponent</td>
<td>TextEvent</td>
</tr>
<tr>
<td>Press return on a text field</td>
<td>JTextField</td>
<td>ActionEvent</td>
</tr>
<tr>
<td>Select a new item</td>
<td>JComboBox</td>
<td>ItemEvent, ActionEvent</td>
</tr>
<tr>
<td>Select Item(s)</td>
<td>JList</td>
<td>ListSelectionEvent</td>
</tr>
<tr>
<td>Click a check box</td>
<td>JCheckBox</td>
<td>ItemEvent, ActionEvent</td>
</tr>
<tr>
<td>Click a radio button</td>
<td>JRadioButton</td>
<td>ItemEvent, ActionEvent</td>
</tr>
<tr>
<td>Select a menu item</td>
<td>JMenuItem</td>
<td>ActionEvent</td>
</tr>
<tr>
<td>Move the scroll bar</td>
<td>JScrollBar</td>
<td>AdjustmentEvent</td>
</tr>
<tr>
<td>Window opened, closed</td>
<td>Window</td>
<td>WindowEvent</td>
</tr>
<tr>
<td>Component added or removed from the container</td>
<td>Container</td>
<td>ContainerEvent</td>
</tr>
<tr>
<td>Component moved, resized, hidden, or shown</td>
<td>Component</td>
<td>ComponentEvent</td>
</tr>
<tr>
<td>Component gained or lost focus</td>
<td>Component</td>
<td>FocusEvent</td>
</tr>
<tr>
<td>Key released or pressed</td>
<td>Component</td>
<td>KeyEvent</td>
</tr>
<tr>
<td>Mouse pressed, released, clicked, entered, exited, moved, or dragged</td>
<td>Component</td>
<td>MouseEvent</td>
</tr>
</tbody>
</table>

### 2. Selected event class, listener interface, and handlers
<table>
<thead>
<tr>
<th>Event Class</th>
<th>Listener Interface</th>
<th>Listener Methods (handlers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionEvent</td>
<td>ActionListener</td>
<td>actionPerformed(ActionEvent e)</td>
</tr>
<tr>
<td>ItemEvent</td>
<td>ItemListener</td>
<td>itemStateChanged(ItemEvent e)</td>
</tr>
<tr>
<td>WindowEvent</td>
<td>WindowListener</td>
<td>windowClosing(WindowEvent e)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>windowClosed(WindowEvent e)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>windowActivated(WindowEvent e)</td>
</tr>
<tr>
<td>ContainerEvent</td>
<td>ContainerListener</td>
<td>componentAdded(ContainerEvent e)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>componentRemoved(ContainerEvent e)</td>
</tr>
<tr>
<td>ComponentEvent</td>
<td>ComponentListener</td>
<td>componentMoved(ComponentEvent e)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>componentHidden(ComponentEvent e)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>componentResized(ComponentEvent e)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>componentShown(ComponentEvent e)</td>
</tr>
<tr>
<td>FocusEvent</td>
<td>FocusListener</td>
<td>focusGained(FocusEvent e)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>focusLost(FocusEvent e)</td>
</tr>
<tr>
<td>TextEvent</td>
<td>TextListener</td>
<td>textValueChanged(TextEvent e)</td>
</tr>
<tr>
<td>KeyEvent</td>
<td>KeyListener</td>
<td>keyPressed(KeyEvent e)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>keyReleased(KeyEvent e)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>keyTyped(KeyEvent e)</td>
</tr>
<tr>
<td>MouseEvent</td>
<td>MouseListener</td>
<td>mousePressed(MouseEvent e)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mouseReleased(MouseEvent e)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mouseEntered(MouseEvent e)</td>
</tr>
<tr>
<td></td>
<td>MouseMotionListener</td>
<td>mouseExited(MouseEvent e)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mouseClicked(MouseEvent e)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mouseDragged(MouseEvent e)</td>
</tr>
<tr>
<td>AdjustmentEvent</td>
<td>AdjustmentListener</td>
<td>adjustmentValueChanged(AdjustmentEvent e)</td>
</tr>
</tbody>
</table>