Event Handling – Listening for Events

There are effectively three stages to developing event handling, not necessarily in this order …

The first stage is to create a component which will have an event which needs handling.

Second, a listener is created to watch for the event taking place, this is then associated to the Component.

And lastly, the listener will need a method which will be run when the listener detects an event has taken place.

Event Handling – Listening for Events

There are three main methods of defining listeners in Java

A **public** Listener, shareable by all components within a class and may need to be tested when an event occurs to ascertain which component 'fired' the event. Components needing to use this Listener must be registered to it.

A **private** Listener, created strictly for the use of one instance of a component, useful if only one component needs a listener but rather cumbersome if applied to a process which has ten buttons for example.

An **implemented** Listener, defined within the instance class and its events are tested within it.

Event Handling – Listening for Events

Almost each Object created in Java Swing will need a Listener attached in order for the Developer to detect changes / input etc …

Each Object needs a specific Listener for the event to be detected, the following Appendices list the required Listener for a Component and what events will trigger them …

Appendix J5 lists all Listeners and Components
Appendix J6 lists all Listeners and their events/methods.

Some Listeners have only one event, others like mouse movements have seven, the Developer must create methods for all events for the Listener as all Listeners are effectively Interfaces.