**8.19 Adapters :-**

**Adapter classes**

Many listener interfaces have more than one callback method. An example is java.awt.FocusListener that has two methods; focusGained(java.awt.FocusEvent event) and focusLost(java.awt.FocusEvent event). When creating a listener class that implements the interface the Java compiler insists that all of the interface methods are implemented, which often results in many empty methods being created to satisfy its requirements when only one or some of its methods actually contain code. The following statement shows a FocusListener being used to perform some logic when a Java bean gains focus. However, an empty focusLost method must be provided.

javaBean.addFocusListener(new java.awt.event.FocusListener() {

public void focusGained(java.awt.event.FocusEvent e) {

doFocusGainedCode();

}

public void focusLost(java.awt.event.FocusEvent e) {

}

});

To avoid having many empty listener methods for many listeners, Adapter classes are provided. These implement the listener interface, and provide empty no-op implementation of its methods. The advantage is that the listener can extend these, and only specialize methods of choice without having to provide default implementations for the rest ( these are inherited from the Adapter ).

javaBean.addFocusListener(new java.awt.event.FocusAdapter() {

public void focusGained(java.awt.event.FocusEvent e) {

doFocusGainedCode();

}

});

In java programming language, adapter class is used to implement an interface having a set of dummy methods. The developer can then further subclass the adapter class so that he can override to the methods he requires. Implementing an interface directly, requires to write all the dummy methods. In general an adapter class is used to rapidly construct your own Listener class to field events.  
  
Java programming language also have a design adapter named adapter. The Adapter design pattern is also used to join to two unrelated interfaces so that can work together. The joint between these two interfaces is termed as Adapter. This is something like the conversion of interface of one class into interface. This is done by using an Adapter.

 A simple Java Applet that Illustrates the Java 1.1 Event Handling model  
- It displays a circle at the center of its display.   
- If user clicks and drags the mouse over the circle, the circle moves to follow the mouse  
- When the user presses any of the keyboard arrow keys, the circle moves a few pixels in the appropriate direction.  
- When the user types a function key, F1 thro' F10, the circle color changes

|  |  |
| --- | --- |
| Your Experimental Applet: This Applet is for you to experiment with | This Applet displays the source code that was executed in your experimental applet as a result of the action you performed on that object. |
|  |  |
| Having fun with inter-applet communication - are'nt we? | |
|