The Job of a Window Manager

Event Driven Programming

- Programs respond to events that are generated *outside* the control of the program.
  - User types a key
  - The left mouse button is pressed
  - A CD is removed from the CD drive
- When an event occurs, it is handled by an event handler.

Event Driven Programming

- Event driven programming involves writing the handlers and arranging for the handler to be notified when particular events occur.

Event Handling

- Events are represented by objects that provide information about an event and identifies the event source.
  - Event sources are typically components, but other kinds of objects can also be event sources.
Event Handling

• A *listener* is an object that is to be notified when a particular event occurs.
  – An event source may have multiple listeners registered to it.
  – A single listener can register with multiple event sources.

Swing Listeners

<table>
<thead>
<tr>
<th>Action</th>
<th>Listener Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>User clicks a button, presses return while typing in a text filed, or chooses a menu item</td>
<td>ActionListener</td>
</tr>
<tr>
<td>Users closes a frame (main window)</td>
<td>WindowListener</td>
</tr>
<tr>
<td>User presses a mouse button while the cursor is over a component</td>
<td>MouseListener</td>
</tr>
<tr>
<td>User moves the move over a component</td>
<td>MouseMotionListener</td>
</tr>
</tbody>
</table>

Listeners

• An object that is to be notified when a particular event occurs must:
  – Implement the appropriate *Listener* interface
  – Be registered as an event listener on the appropriate event source.

Swing Listeners

<table>
<thead>
<tr>
<th>Action</th>
<th>Listener Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>A component becomes visible</td>
<td>ComponentListener</td>
</tr>
<tr>
<td>A component gets the keyboard focus</td>
<td>FocusListener</td>
</tr>
<tr>
<td>A table of list selection changes</td>
<td>ListSelectionListener</td>
</tr>
</tbody>
</table>
Window Closing

• A very common event directed towards a window is a close event.
  – The default behavior is to simply hide the JFrame when the user closes the window.
• We prefer that the program terminate when the user closes the main window.
  – Two steps are required to accomplish this:
    • Write an event handler for the close event that will terminate the program.
    • Register the handler with the appropriate event source.

WindowAdapter

• A class that implements WindowListener
  – The methods in this class are empty.
  – The class exists as convenience for creating listener objects.

WindowListener

• The WindowListener interface
  – void windowActivated(WindowEvent e);
  – void windowClosed(WindowEvent e);
  – void windowClosing(WindowEvent e);
  – void windowDeactivated(WindowEvent e);
  – void windowDeiconified(WindowEvent e);
  – void windowIconified(WindowEvent e);
  – void windowOpened(WindowEvent e);
• A class that implements WindowListener must implement all of these methods!

WindowAdapter

• To use the WindowAdapter class:
  – Extend this class to create a WindowEvent listener.
  – Override the methods for the events of interest.
  – Create a listener object using the extended class and then register it with the Window.
• When an event occurs the appropriate method in the listener is invoked.
• Example: SwingFrame.java
Buttons

• Buttons generate **action** events.
• The ActionListener interface
  – void actionPerformed(ActionEvent event);
  – Note that there is no need for an ActionAdapter class

StopWatch.java

Buttons

• Generally one ActionListener will be responsible for handling the events generated by a group of buttons.
  – The getActionCommand() method determines which button was pressed.

• SwingFrame1.java

TimerLabel.java
GUI Program Design

- The User Interface code should be an independent module of the program.
  - Changes in the program do not necessarily change the GUI.
  - Different GUIs can be developed for the same program.
  - Debugging and maintaining both the GUI and the program code is easier.

Model-View-Controller

- The MVC pattern is commonly used to develop applications that have a GUI component.
- Consists of three parts
  - Model
    - The program
  - View
    - The GUI
  - Controller
    - The event handling mechanism

MVC

- The model passes its data to the view for rendering
- The controller updates the model based on the events received
- The view determines which events are passed to the controller

MVC in Swing

- View
  - Program Logic
- Controller
  - The GUI
- The Application
A Simple 4 Function Calculator

```
# A Simple 4 Function Calculator

```