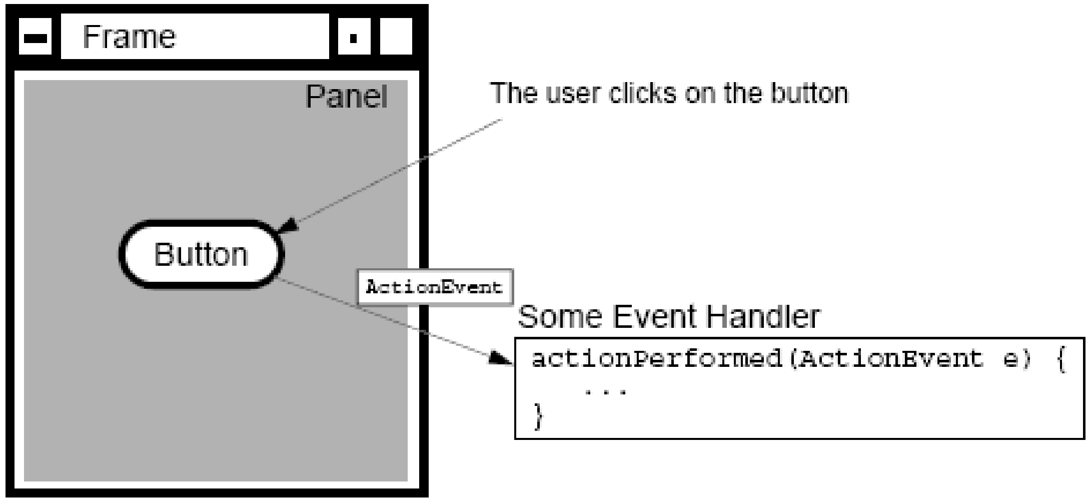
Objectives

- In this session, you will learn to:
  - Define events and event handling
  - Determine the user action that originated the event from the event object details
  - Identify the appropriate listener interface for a variety of event types
  - Create the appropriate event handler methods for a variety of event types
  - Understand the use of inner classes and anonymous classes in event handling
  - Identify the key AWT components and the events they trigger
  - Describe how to create menu, menu bar, menu items and how to control visual aspects

Events

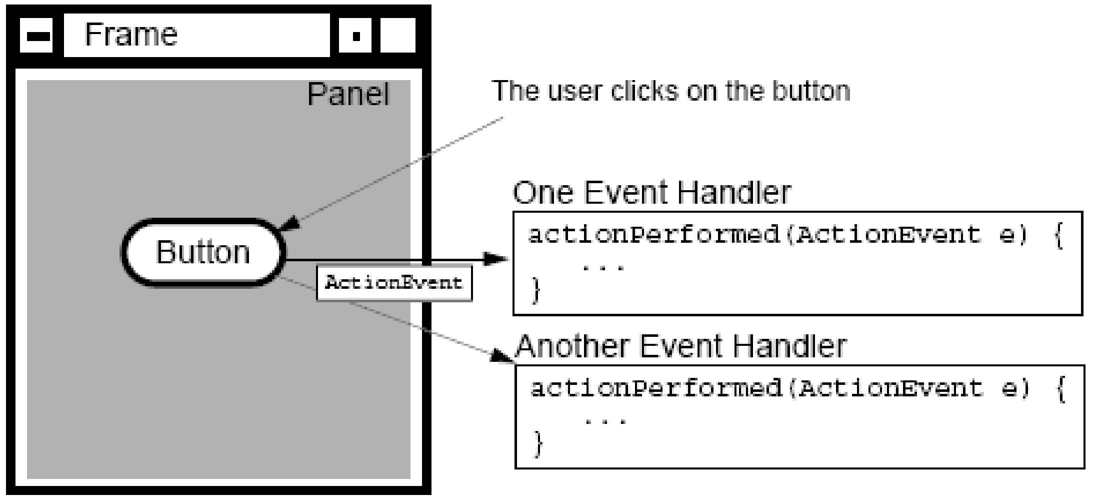
- Events: Objects that describe what happened
- Event sources: The generator of an event
- Event handlers: A method that receives an event object, deciphers it, and processes the user's interaction.



```
actionPerformed(ActionEvent e) {
    ...
}
```

Delegation Model of Event

- An event can be sent to many event handlers.
- Event handlers register with components when they are interested in events generated by that component.



```
actionPerformed(ActionEvent e) {
    ...
}
```

```
actionPerformed(ActionEvent e) {
    ...
}
```

Delegation Model of Event (Contd.)

- Client objects (handlers) register with a GUI component that they want to observe.
- GUI components only trigger the handlers for the type of event that has occurred.
- Most components can trigger more than one type of event.
- The delegation model distributes the work among multiple classes.

A Listener Example

- This code snippet shows a simple Frame with a single button on it, class name is TestButton:

```
public TestButton()
{
  f = new Frame("Test");
  b = new Button("Press Me!");
  b.setActionCommand("ButtonPressed");
}
public void launchFrame()
{
  b.addActionListener(new ButtonHandler());
  f.add(b,BorderLayout.CENTER);
  f.pack();
  f.setVisible(true);
}
```

A Listener Example (Contd.)

- Code for the event listener looks like this:

```
import java.awt.event.*;
public class ButtonHandler implements
ActionListener
{
  public void actionPerformed(ActionEvent e)
  {
    System.out.println("Action occurred");
    System.out.println("Button’s command is:
    "+ e.getActionCommand());
  }
}
```

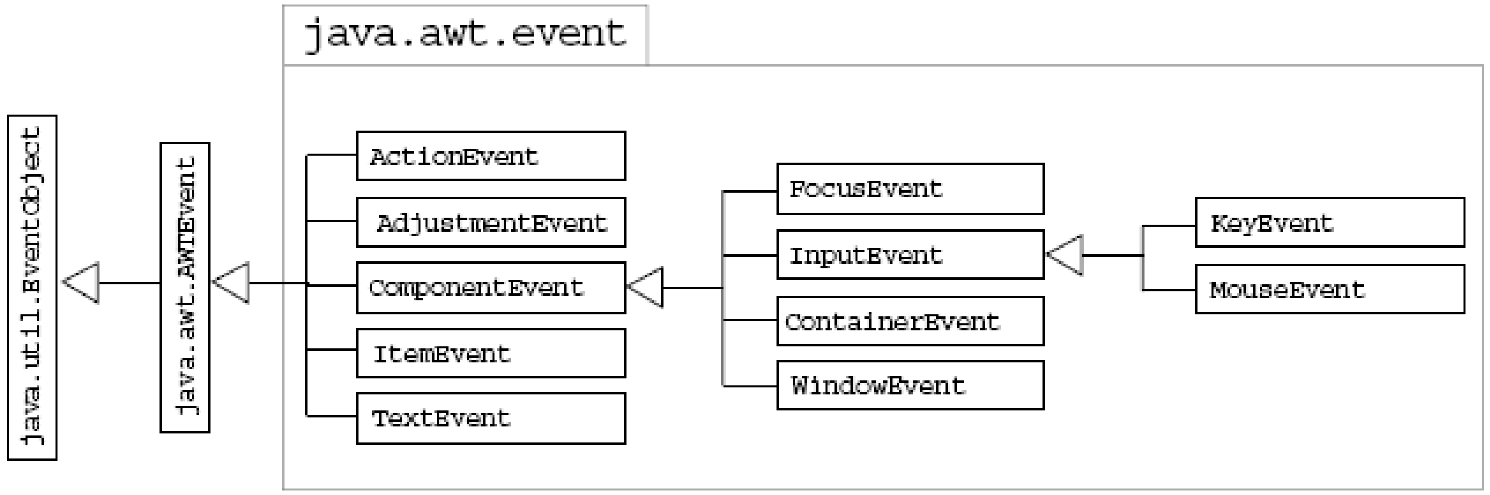
- The event is delegated to ButtonHandler class.

Demonstration

Lets see how to use the Event handling API to handle simple GUI events.

Event Categories

- Class Hierarchy of GUI Events:



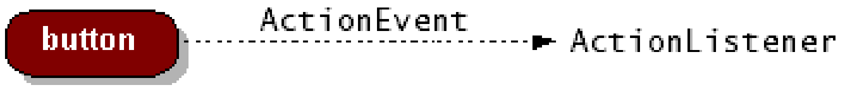
Listener Type

- Some Events and Their Associated Event Listeners:

Act that Results in the Event	Listener Type
User clicks a button, presses Enter while typing in a text field, or chooses a menu item	ActionListener
User closes a frame (main window)	WindowListener
User presses a mouse button while the cursor is over a component	MouseListener
User moves the mouse over a component	MouseMotionListener
Component becomes visible	ComponentListener
Component gets the keyboard focus	FocusListener

Listeners

- ActionListener Interface:
  - Has only one method i.e. `actionPerformed(ActionEvent)`
  - To detect when the user clicks an onscreen 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 onscreen button, the button fires an action event.



Listeners (Contd.)

- MouseListener interface:
  - To detect the mouse clicking, a program must have an object that implements the `MouseListener` interface.
  - This interface includes several events including `mouseEntered`, `mouseExited`, `mousePressed`, `mouseReleased`, and `mouseClicked`.
  - When the user clicks the onscreen button, the button fires an action event.

Listeners (Contd.)

- Implementing Multiple Interfaces:
  - A class can be declared with Multiple Interfaces by using comma separation:
    - Implements MouseListener,MouseMotionListener
- Listening to Multiple Sources:
  - Multiple listeners cause unrelated parts of a program to react to the same event.
  - The handlers of all registered listeners are called when the event occurs.

Event Adapters

- The listener classes that you define can extend adapter classes and override only the methods that you need.
- An example is:

```
import java.awt.*;
import java.awt.event.*;
public class MouseClickHandler extends
MouseAdapter
{
  //We just need the mouseClick handler, so
  //we use an adapter to avoid having to
  //write all the event handler methods
```

Event Adapters (Contd.)

```
    public void mouseClicked(MouseEvent e)
    {
    // Do stuff with the mouse click...
    }
  }
```

Inner Classes

- Event Handling Using Inner Classes:
  - Using inner classes for event handles gives access to the private data of the outer class.

MenuBar

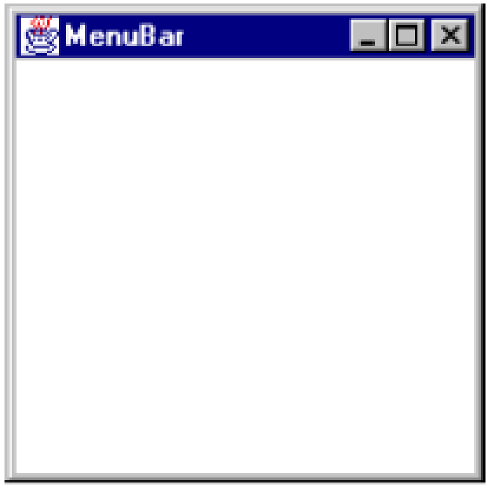
- Frames can contain a menu bar, a menu bar can contain zero or more menus, and menu can contain zero or more menu items (including submenus).

  Let’s see how to do this.

Creating a MenuBar

- Create a MenuBar object, and set it into a menu container, such as a Frame. For example:

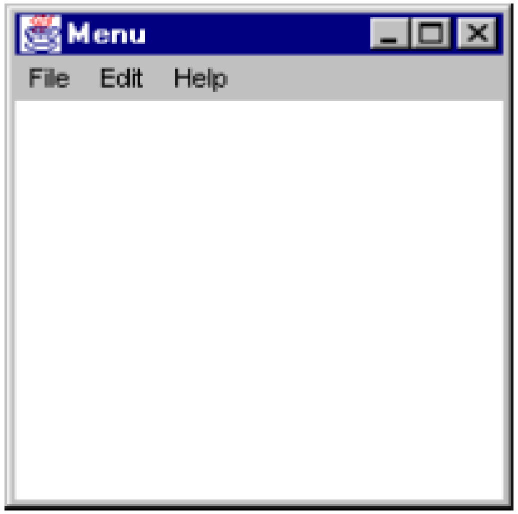
```
Frame f = new Frame("MenuBar");
MenuBar mb = new MenuBar();
f.setMenuBar(mb);
```



Creating a Menu

- Create one or more Menu objects, and add them to the menu bar object. For example:

```
Frame f = new Frame("Menu");
MenuBar mb = new MenuBar();
Menu m1 = new Menu("File");
Menu m2 = new Menu("Edit");
Menu m3 = new Menu("Help");
mb.add(m1);
mb.add(m2);
mb.setHelpMenu(m3);
f.setMenuBar(mb);
```



Creating a MenuItem

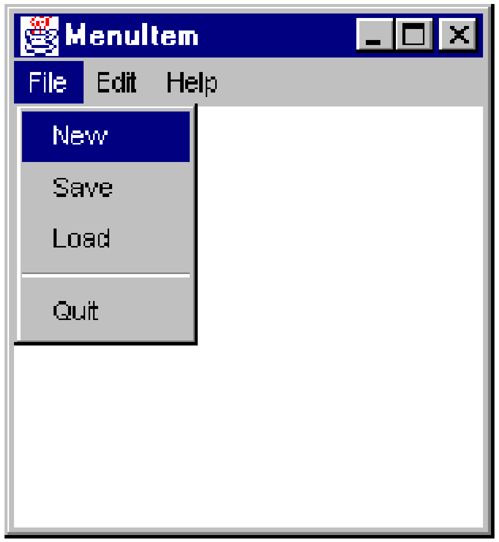
- Create one or more MenuItem objects, and add them to the menu object. For example:

```
MenuItem mi1 = new MenuItem("New");
MenuItem mi2 = new MenuItem("Save");
MenuItem mi3 = new MenuItem("Load");
MenuItem mi4 = new MenuItem("Quit");
mi1.addActionListener(this);
mi2.addActionListener(this);
mi3.addActionListener(this);
mi4.addActionListener(this);
m1.add(mi1);
m1.add(mi2);
```

Creating a MenuItem (Contd.)

```
m1.add(mi3);
m1.addSeparator();
m1.add(mi4);
```

Let' see how MenuItem will look like.



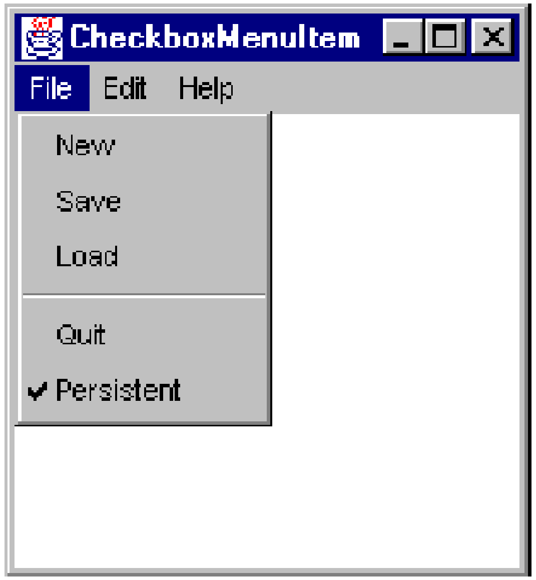
Demonstration

Lets see how to add a menu and other GUI components to a AWT application.

Creating a CheckBoxMenuItem

- Creating a CheckBoxMenuItem:

```
CheckboxMenuItem mi5 =
newCheckboxMenuItem("Persistent");
mi5.addItemListener(this);
m1.add(mi5);
```



Controlling Visual Aspects

- Commands to control visual aspects of the GUI include:
  - Colors:

```
setForeground()
setBackground()
```

Example:

```
Color purple = new Color(255, 0, 255);
Button b = new Button(“Purple”);
b.setBackground(purple);
```

J.F.C./Swing Technology

- Java Foundation Class/Swing (J.F.C./Swing) technology is a second-generation GUI toolkit.
- It builds on top of AWT, but supplants the components with lightweight versions.
- There are many more components, and much more complex components, including `JTable`, `JTree`, and `JComboBox`.

Summary

- In this session, you learned that:
  - When user perform some action, for example, button click or mouse move then the program performs some action which is called event.
  - Events can be handled by implementing appropriate Listener Interface.
  - Most components can trigger more than one type of event.
  - The delegation model distributes the work among multiple classes.
  - ActionListener Interface:
    - When the user clicks an onscreen button (or does the keyboard equivalent), a program must have an object that implements the `ActionListener` interface.
  - MouseListener Interface:
    - To detect the mouse clicking, a program must have an object that implements the `MouseListener` interface.

Summary (Contd.)

- A class can be declared with multiple interfaces by using comma separation.
- Event Adapter classes can be used in place of implementing listener, if you need to implement only one method.
- Manubar can be created by creating a `MenuBar` class object, and set it into a menu container, such as a Frame.
- `Menu` class object is used to create menu, and add them to the `MenuBar` object.
- MenuItems can be created by creating one or more `MenuItem` class objects, and add them to the menu object.
- Checked menuitems can be created by using `CheckboxMenuItem` class object.
- Colors can be set by creating the `Color` class object.