Event handling and examples
Event handling

- An event is a condition or action that is observed by a system but which occurs outside the system's control.
- A system whose primary function is to react to events is termed an event-driven system.
- In the context of the AWT, events correspond to conditions or actions that are related to the user interface.
- Examples of AWT events are conditions such as whether the shift key is down, whether the mouse is inside of a component.
Event handling

• The source object: a Component in which the event "occurs" and which maintains an association to a set of one or more objects "listening" for the event,

• The listener object: an object of a class that implements a prescribed listener interface and which is programmed to react to the occurrence of the event

• An event object: an object that contains information describing a particular event passed from the source to the listener when the event occurs.
A program must arrange for the listener to be made known to, or registered with, the source object. Each source object maintains a list of registered listeners and provides a method for adding a listener object to this list. A listener object will only be notified of those events occurring in a source object only after the listener is registered with the source object.
Event handling

• A Button object generates an ActionEvent when the user presses the user interface button.
• For a user-defined class to listen for the event of a button being pushed, the class must implement the ActionListener interface.
• An object of this user-defined class would register itself as a listener for the object of the Button class using the addActionListener method defined in the Button class.
• When a user-interface button is pushed, an ActionEvent object is passed from the source to the listener. The ActionEvent object contains data that describes in which Button the event occurred and other descriptive information about the event.
Java event types

Event type: ActionEvent
Component: Button, List, MenuItem, TextField

Event type: WindowEvent
Component: Window
Naming convention

• The name of each listener interface is derived from the name of the corresponding event type

```java
public abstract interface xEventListener
{
    // listener methods
    public void methodName(xEvent eventObject);
}
```
Naming convention

A component that generates an xEvent event provides 2 methods:

```java
public void addxEventListener( xEventListener listener);
public void removexEventListener( xEventListener listener);
```
An example: buttons

public abstract interface ActionEventListener
{
    // ActionEvent listener methods
}

public void addActionEventListener(
    ActionEventListener listener);
public void removeActionEventListener(
    ActionEventListener listener);
Listener interfaces

• Each event type has a corresponding listener interface all of whose methods must be implemented by a listener object

• Example:
  – ActionListener: actionPerformed
  – MouseListener: mouseClicked, mouseEntered, mouseExited, mousePressed, mouseReleased
Patterns of Event handling

• What is a pattern?
• “Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem in such a way that you can use this solution a million times over, without ever doing it the same way twice” - C. Alexander
• Three patterns: reactor, monitor, and delegator
A reactor pattern defines a single object with a self-contained, responsive ability: the object itself is programmed to deal with its own events.
Monitor

- The listener object also plays the role of the handler object. The term "monitor" is used in this case because the listener/handler object has the responsibility of monitoring and adjusting to changes in the source object.
Delegator

- listener object uses the services of other objects to respond to the events generated by the source object
In DrawTool

- **DrawFrame** -> reactor
- **DrawCanvas** -> monitor
- **color selection** -> delegator
DrawTool

- WindowEvent: closing the DrawTool window and terminating the application
- MouseEvent: responding to mouse clicking and dragging by drawing and moving a rectangle in the drawing area
- ActionEvent: clearing the drawing area in response to the "Clear Canvas" button being pressed
- ItemEvent: changing the drawing color when the user selects an alternative from the Choice component
• DrawFrame class must implement WindowListener interface
• DrawFrame is both the source and the listener of events
• The DrawCanvas plays the roles of source, listener, and handler for the mouse events and also plays the listener and handler roles for the button events.

• There are two sources of events relevant to the DrawCanvas class. The Button class is a source of ActionEvents, and the Canvas class, from which DrawCanvas inherits, is a source of MouseEvents and MouseMotionEvents.
Menus

- Two types: pulldown and popup
- Multilevel menus
- Shortcut
- Check-box menu
- Separators
Menu

- Pulldown menus can only be used with Frames and not with any other component including applets
- DrawClass implements the ActionListener interface
- private method addItem
Examples
Layout managers

• **Border layout**
  – Places the components at the margins (top, bottom, left, right) and the center. The margins are named by the compass points (North, South, East, West).

• **Flow layout**
  – Places the components in a left-to-right, top-to-bottom fashion
Layout managers

• **Grid layout**
  – Places the components in a two dimensional grid. All grid elements are the same size. Components are placed in successive elements starting in the upper left-hand grid position and proceeding column by column and row by row.

• **Card layout**
  – Presents only one of its components at a time. The choice among components can be made by a name associated with each component or by navigation.
• **GridBag layout**
  - Similar to the GridLayout except that the grid elements are not necessarily of the same size. Each component can have an associated GridBagConstraint to control its configuration.
public class BorderFrame extends Frame implements WindowListener {

    public BorderFrame() {
        ...
        setLayout(new BorderLayout());
        Button button1 = new Button("North");
        Button button2 = new Button("South");
        Button button3 = new Button("East");
        Button button4 = new Button("West");
        Button button5 = new Button("Center");
        add("North", button1);
        add("South", button2);
        add("East", button3);
        add("West", button4);
        add("Center", button5);
    }
}
public class FlowFrame extends Frame implements WindowListener {

    public FlowFrame() {
        //...

       setLayout(new FlowLayout());
        Button button1 = new Button("Button 1");
        Button button2 = new Button("Button 2");
        Button button3 = new Button("Button 3");
        Button button4 = new Button("Button 4");
        Button button5 = new Button("Button 5");
        add(button1);
        add(button2);
        add(button3);
        add(button4);
        add(button5);
    }
}
public class GridFrame extends Frame implements WindowListener
{
    public GridFrame()
    {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new GridLayout(3,2)); // 3 rows, 2 columns
        Label label1 = new Label("First Button");
        Button button1 = new Button("Button 1");
        Label label2 = new Label("Second Button");
        Button button2 = new Button("Button 2");
        Label label3 = new Label("Third Button");
        Button button3 = new Button("Button 3");
        add(label1); add(button1);
        add(label2); add(button2);
        add(label3); add(button3);
    }
    //...
}
Card

Unlike the other layout managers, the programming aspect of this layout require the program to directly interact with the CardLayout.
Card

• The CardLayout treats each component in a container as representing a “card”
• The cards are ordered by the sequence in which they were added to the container
• The CardLayout manager maintains an index of which card in the sequence is currently displayed
GridBay

• The most complex but also the one which offers the most control over the appearance of the GUI.