UNIVERSITY of INFORMATION TECHNOLOGY

CST-204

Java Programming
Learning Java

Fourth Edition

Patrick Niemeyer & Daniel Leuck

O’REILLY

UNIVERSITY of INFORMATION TECHNOLOGY

Graphical User Interface

Chapter 16 – Swing

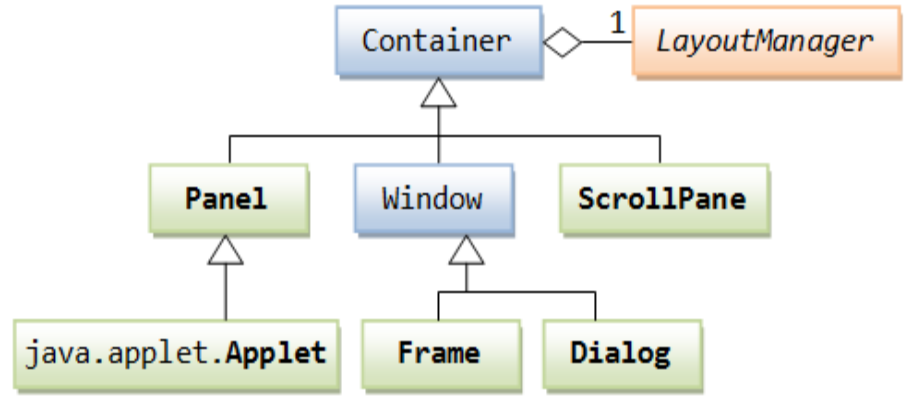
Java's Graphical User Interface Tookit

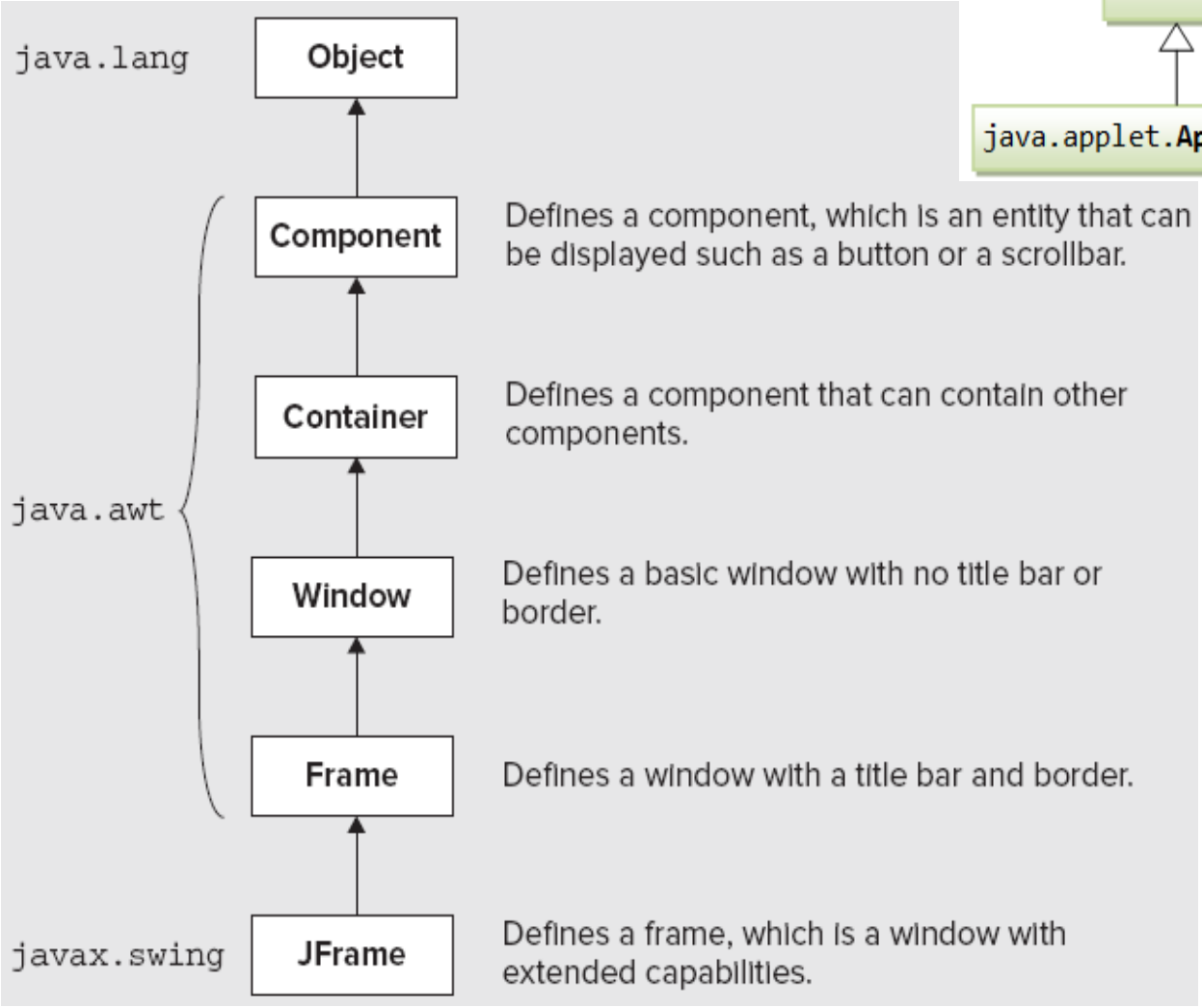
- TWO Packages: **`java.awt`** and **`javax.swing`**
- Abstract Window Toolkit (**AWT**)
  - the primary repository for graphics classes
- Many of AWT classes have been superseded by **`javax.swing`**
- The **Swing** component classes are more flexible than the component classes defined in the java.awt package
- Other than AWT/Swing Graphics APIs provided in JDK, others Graphics APIs that work with Java:
  - Eclipse's Standard Widget Toolkit (SWT), Google Web Toolkit (GWT)
  - 3D Graphics API: Java bindings for OpenGL (JOGL) and Java3D

Swing

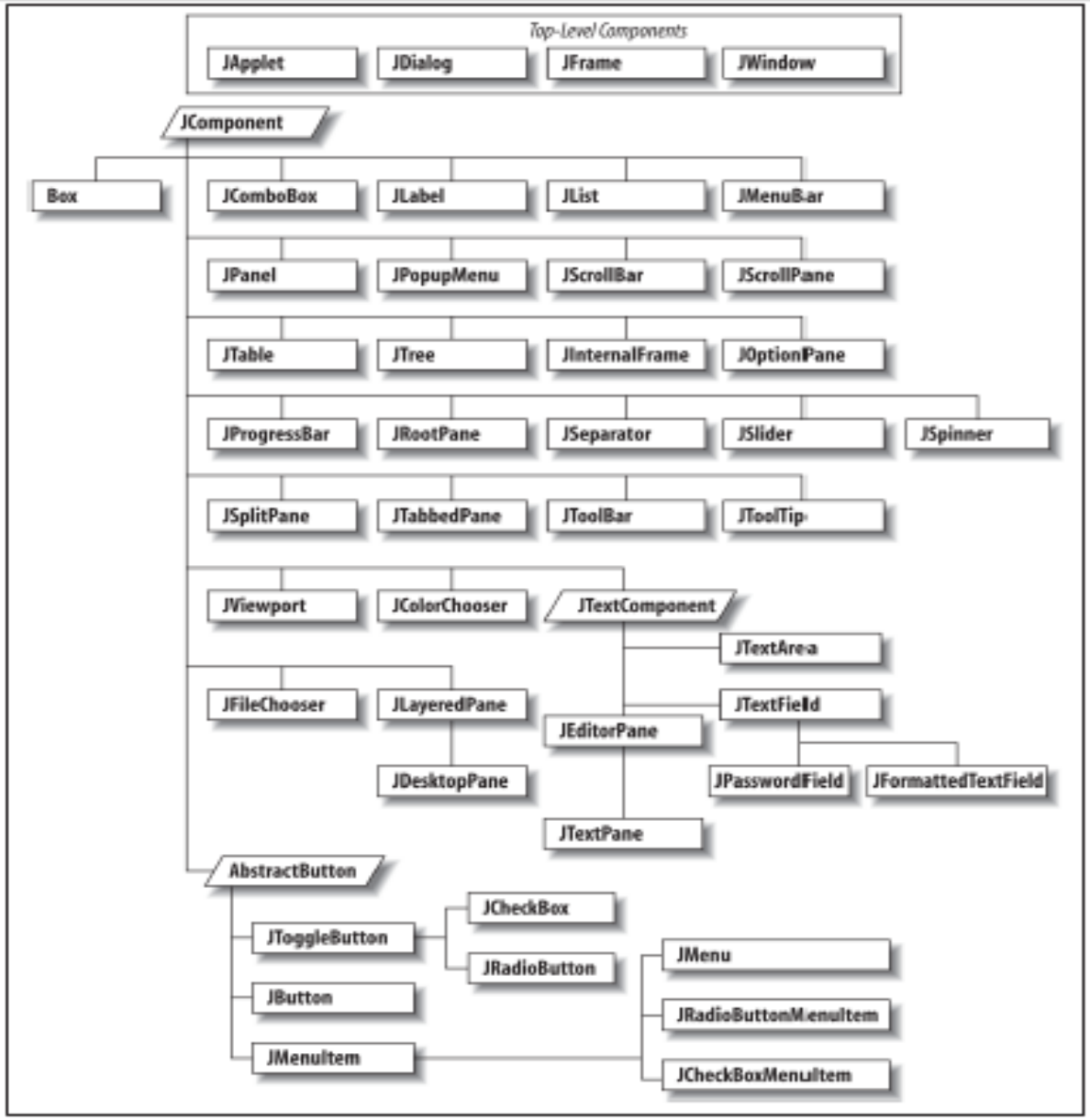
- Package: **`javax.swing`**
  - contains classes representing interface items such as windows, buttons, combo boxes, trees, tables, menus and …
  - Swing is part of a larger collection of software called Java Foundation Classes (JFC)
- JFC includes the following APIs
  - Abstract Window Toolkit (AWT) – the original user interface toolkit and base graphics classes
  - swing - the pure Java user interface toolkit
  - 2D API, a comprehensive set of classes for high-quality drawing
  - Drag and Drop, an API that supports the drag-and-drop metaphor

Components in AWT & swing





User Interface Components in swing



Components

- is the fundamental user interface object in Java
- Everything you see on the display in a Java application is a component.
- must be placed in a container which group components
- are arranged for display using a layout manager
- All Swing components (such as JButton, JLabel, JList, JTextField) are derived from the abstract javax.swing.JComponent class
- the functionality of the JComponent class into two categories: **appearance** and **behavior (event)**

Using a GUI Component

1. Create it
   - Instantiate object: b = new JButton(“press me”);
2. Configure it
   - Properties: b.text = “press me”; [avoided in java]
   - Methods: b.setText(“press me”);
3. Add it
   - panel.add(b);
4. Listen to it
   - Events: Listeners

MVC Framework

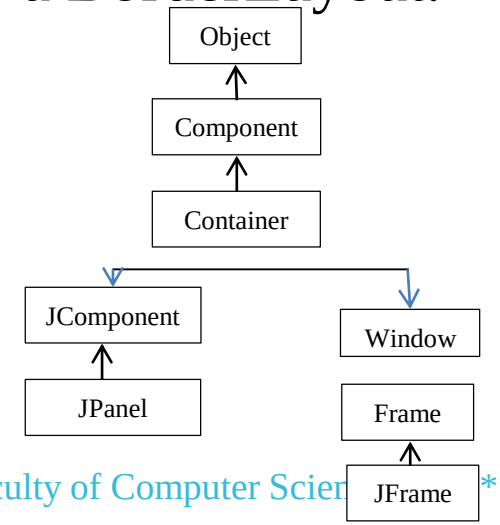
- is a method of building reusable components that logically separates the structure, presentation, and behavior of a component into separate pieces
  - model : data
  - view : presentation
  - controller : defines and governs its behavior(is responsible for processing user requests and building an appropriate model and passes it to the view)
- Swing components explicitly support MVC
  - UI-delegate is responsible for “view” and “controller” roles
  - Data model

Container

- A Container is a kind of component that hold and manages other SWING components
- Three of the most useful general container types are JFrame, JPanel, JApplet.
- JFrame is a top-level window on your display.
- JPanel is a generic container element that groups components inside JFrames and other JPanels.
- JApplet is a kind of container that provides the foundation for applets that run inside web browsers.
- The add() method of the Container class adds a component of the container.
- Can remove a component from a container with the remove() method.

JFrame

- JFrame is the Swing Window, a top-level window
- The class **JFrame** is an extended version of **java.awt.Frame** that adds support for the JFC/Swing component architecture.
- Frame are example of containers. This means that a frame can contain other user interface components such as buttons and text fields.
- The default for a JFrame is a BorderLayout.



Windows and Frames

Example:

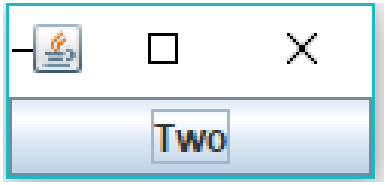
```
    JFrame frame = new JFrame("The Frame");
    frame.setSize(300, 300);//gives the frame a
  fixed size in pixels
    frame.setLocation(100, 100);
      //frame.setResizable(false);
    frame.setVisible(true);   //to make a frame
  appear on the screen after creating it.

    frame.setDefaultCloseOperation(JFrame.EXIT_ON
  _CLOSE);//makes the frame perform the fiven
  action when it closes
/*          JWindow window = new JWindow();
    window.setSize(300, 300);
    window.setLocation(500, 100);
    window.setVisible(true);*/
```

Content Panes

Example:

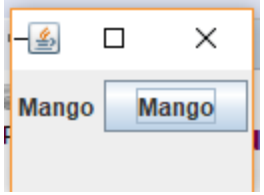
```
    JFrame frame = new JFrame("The Frame");
// The three methods below are delegated to the frame's ContentPane.
   frame.setLayout(new FlowLayout());
   frame.add(new JLabel("Mango"));//places the given component or container inside the frame.
   frame.add(new JButton("Mango"));
   frame.setLocation(100, 100);
   frame.pack();//resizes the frame to fit the components inside it snugly
   frame.setVisible(true);
```

example

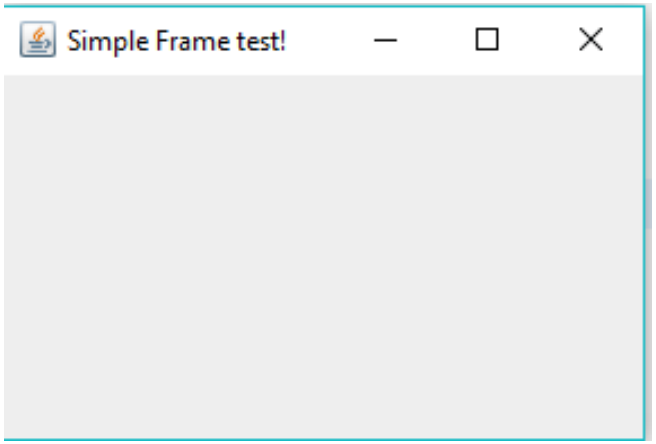



```
import javax.swing.*;

public class GUI {

public static void main(String[] args) {
JButton b1=new JButton("Two");
JFrame frame=new JFrame("The Frame");
//the three methods below are delegated to the frame's contentPane.
frame.add(b1);
frame.setLocation(100, 100);
//frame.setSize(100,100);
frame.pack();
frame.setVisible(true); //frame.show();
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

}

}
```

```java
import javax.swing.*;

public class GUI1 {

public static void main(String[] args) {
    JFrame frame=new JFrame("The Frame");
    frame.setLayout(new FlowLayout());
    frame.add(new JLabel("Mango"));
    frame.add(new JButton("Mango"));
    frame.setLocation(100,100);
    frame.setSize(100, 100);
    frame.setVisible(true);

}

}
```

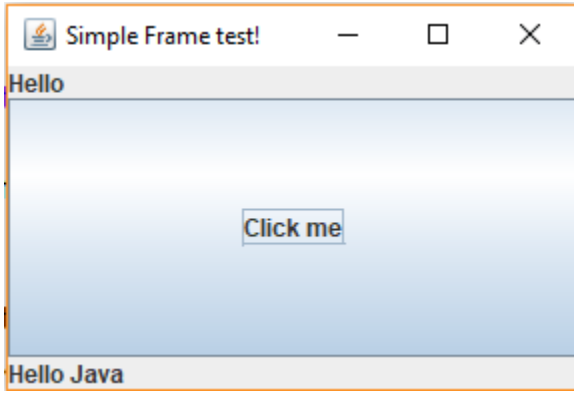


example



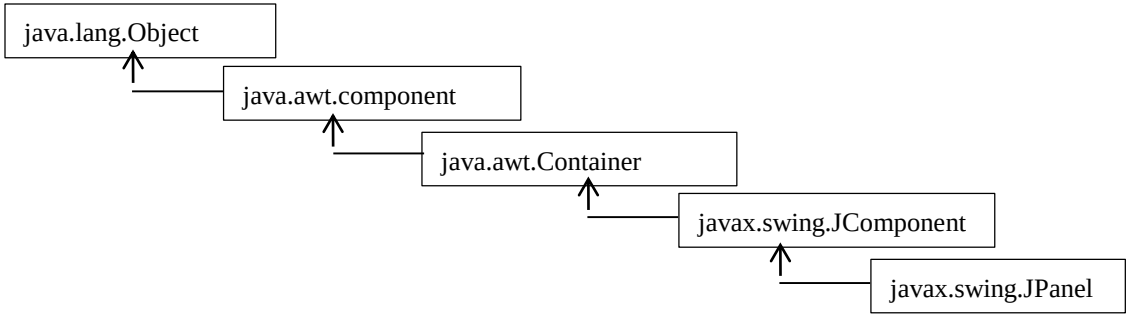

```
import javax.swing.*;

public class SimpleFrame extends JFrame{

public SimpleFrame() {
    setSize(300, 200);
    setTitle("Simple Frame test!");//set frame title
    setLocation(100, 100);//set position on the screen
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setVisible(true);//show window
}

public static void main(String[] args) {
    SimpleFrame sf=new SimpleFrame();

}

}
```

```java
import java.awt.BorderLayout;
import java.awt.Container;
import javax.swing.*;
public class SimpleFrame1 extends JFrame{
    JLabel lbl=new JLabel("Hello");
    JLabel lbl1=new JLabel("Hello Java");
    JButton btn=new JButton("Click me");
public SimpleFrame1() {
    setSize(300, 200);
    setTitle("Simple Frame test!");
    setLocation(100, 100);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setVisible(true);
    Container c=this.getContentPane(); //add components to the content
    pane

    c.add(lbl, BorderLayout.NORTH);
    c.add(btn,BorderLayout.CENTER);
    c.add(lbl1,BorderLayout.SOUTH);
}
public static void main(String[] args) {
    SimpleFrame1 sf=new SimpleFrame1();
}
}
```

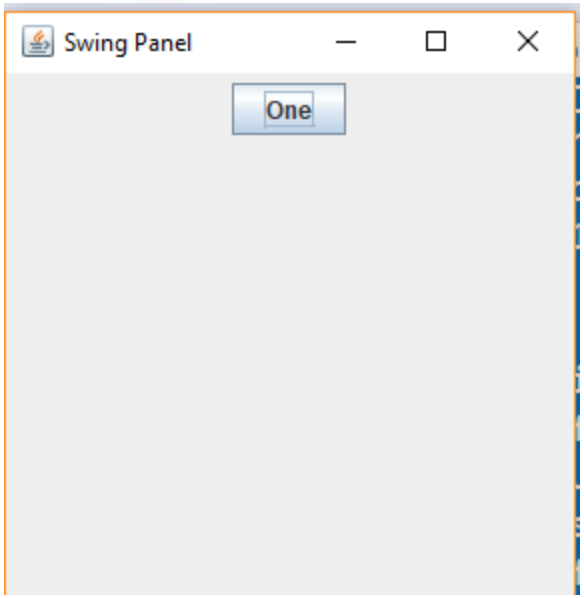


JPanel

- The class **JPanel** is a generic lightweight container.
- To group/organize components
- A custom component that requires embedded components
- The default layout manager for a JPanel is a FlowLayout
- To group organize components



```java
import java.awt.*;
import javax.swing.*;

public class PanelTest extends JFrame{

    JButton b1=new JButton("One");

public PanelTest() {
    super("Swing Panel");
    //Container contentpane=getContentPane();
    JPanel j=new JPanel();
    //contentpane.add(j);
    this.add(j);
    j.add(b1);
}

public static void main(String[] args) {
    JFrame f=new PanelTest();
    f.setBounds(100,100,300,300);
    f.setVisible(true);
    f.setDefaultCloseOperation(EXIT_ON_CLOSE);

}
}
```

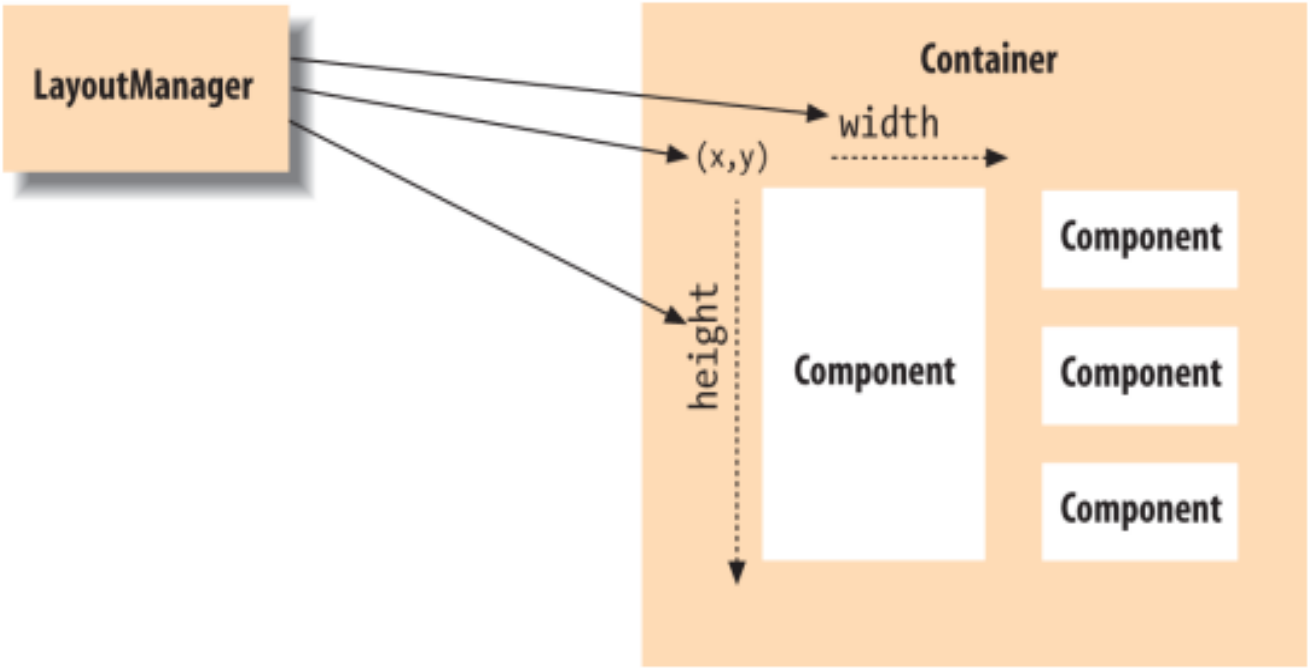


Layout Managers

- Container – is a kind of component that holds and manages other components
  - JFrame, JPanel, JApplet and Jdialog,ScrollPane
- Layout manager - is an object that controls the placement and sizing of components within the display area of a container
- Default layout manager
  - FlowLayout for JPanel
  - BorderLayout for JFrame
  - GridLayout
  - GridBagLayout
  - …
- Insets specify a container's margins; top, bottom, left, and right
- specify the component's exact position in the container's stacking order

Layout Manager

- A layout manager handle how components are displayed in containers and you can install those layout managers into those.



Layout Managers

- FlowLayout
  - The class **FlowLayout** components in a left-to-right flow.
- BorderLayout
  - The **BorderLayout** arranges the components to fit in the five regions: east, west, north, south, and center.
- GridLayout
  - The class **GridLayout** arranges the components in a rectangular grid.
- BoxLayout
- CardLayout
- GridBagLayout
- Other layout Managers- SpringLayout and GroupLayout

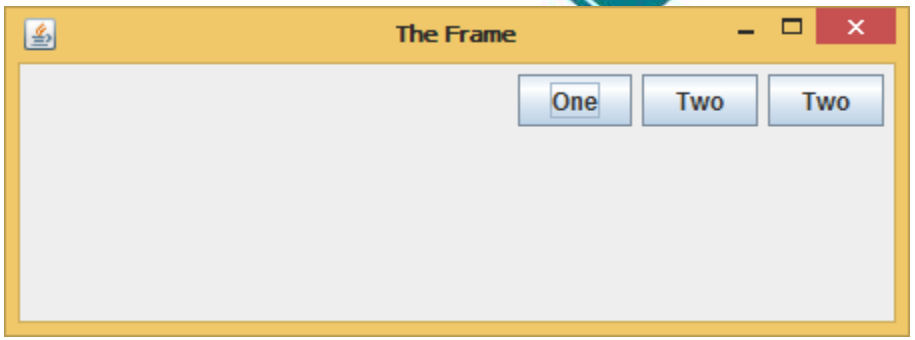
FlowLayout

- is a simple layout manager that tries to arrange components at their preferred sizes, from left to right and top to bottom in the container
- can have a specified row justification of LEFT, CENTER, or RIGHT and a fixed horizontal and vertical padding
- By default, a flow layout uses CENTER justification

  – Flow .java – Page 711 ~ 712

Flow Layout Constructors

new FlowLayout(FlowLayout.*RIGHT)*



- FlowLayout ()

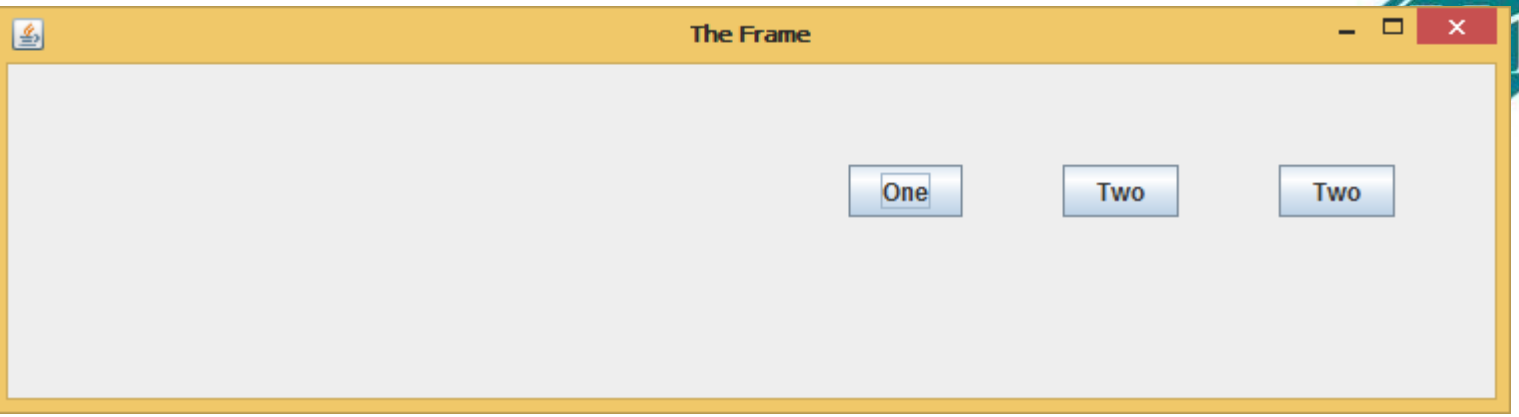
default : CENTER alignment, hgap = 5 and vgap = 5

- FlowLayout (int align)

align ➔ CENTER, LEADING, LEFT, RIGHT, TRAILING

default : hgap = 5 and vgap = 5

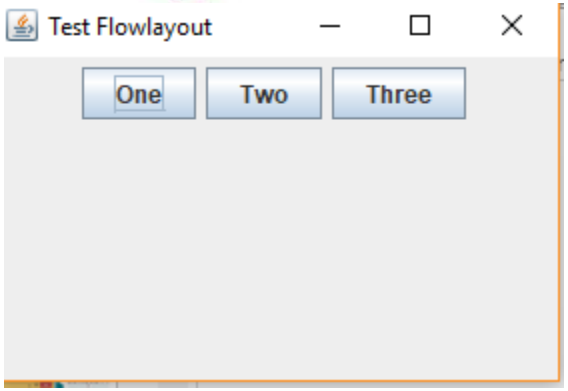
- FlowLayout (int align, int hgap, int vgap)



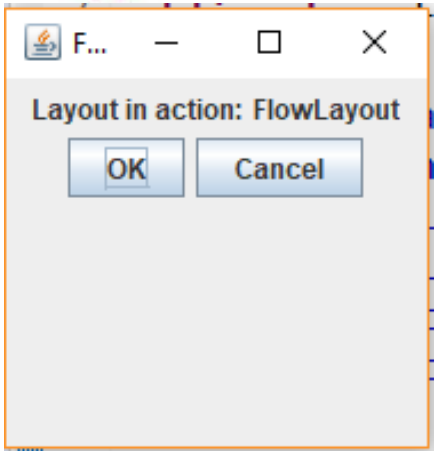
new FlowLayout(2,50,50)


```
setLayout(new FlowLayout());
 add(button);
```

```java
import javax.swing.*;

public class FlowLayoutEg extends JFrame{
    JButton b1=new JButton();
    JButton b2=new JButton("Two");
    JButton b3=new JButton("Three");
public FlowLayoutEg() {
    setTitle("Test Flowlayout");
    setSize(300, 200);
    setLocation(100, 100);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
    b1.setText("One");
    this.setLayout(new FlowLayout());
    this.add(b1);
    this.add(b2);
    this.add(b3);
}
public static void main(String[] args) {
    new FlowLayoutEg();
}
}
```



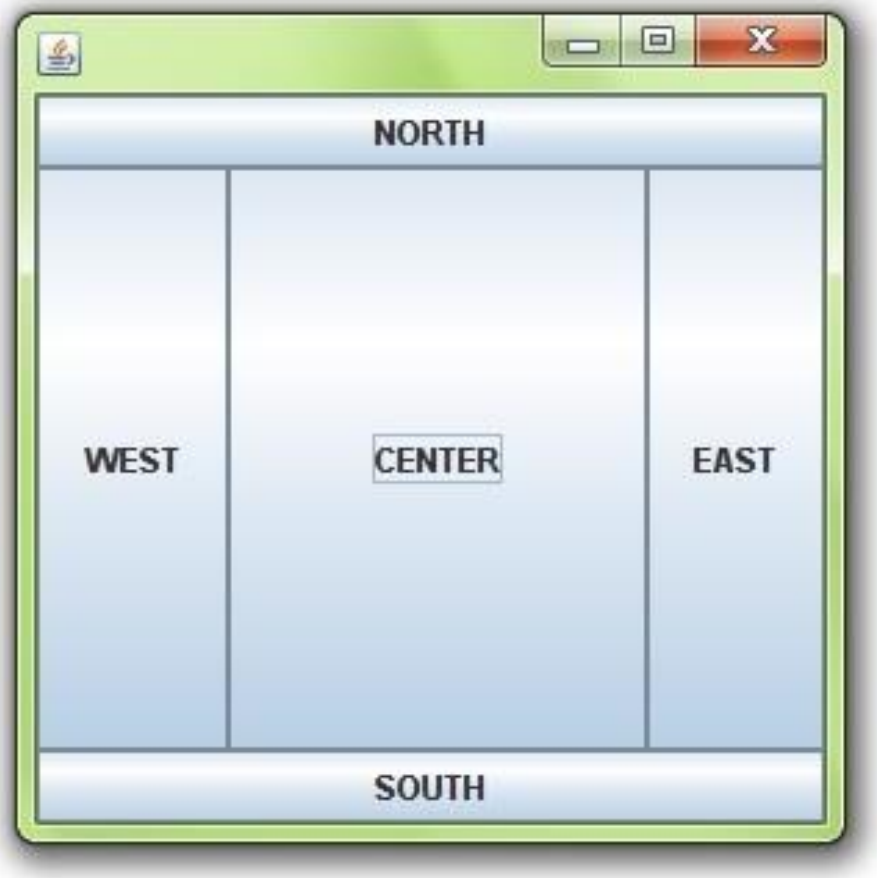
```java
import java.awt.FlowLayout;
import javax.swing.*;
public class FlowLayoutEg {
    JFrame frame;
    JPanel panel;
    JLabel lbl;
    JLabel lbl1;
    JButton btn;
    JButton btn1;
    FlowLayoutEg(){
        frame=new JFrame("FlowLayout Example");
        frame.setSize(200, 200);
        frame.setLocation(200, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
        panel=new JPanel(new FlowLayout());
        lbl=new JLabel("LayoutManager:",JLabel.CENTER);
        lbl1=new JLabel("FlowLayout",JLabel.CENTER);
        btn=new JButton("OK");
        btn1=new JButton("Cancel");
        panel.add(lbl);
        panel.add(lbl1);
        panel.add(btn);
        panel.add(btn1);
        frame.add(panel);}
public static void main(String[] args) {
        new FlowLayoutEg();}
}
```



BorderLayout

- tries to arrange objects in one of five geographical locations, represented by constants in the BorderLayout class: NORTH, SOUTH, EAST, WEST, and CENTER (default)
  - Border1.java – Page 714

- BorderLayout manage sets of components in their own panels. The default layout for a JPanel is FlowLayout
  - Border2.java – Page 715



Usage:

setLayout(new BorderLayout());

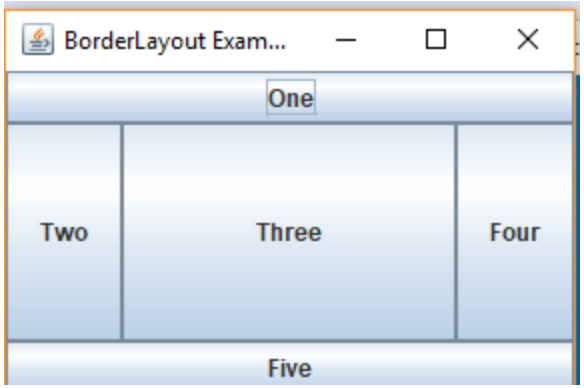
add(button, **BorderLayout.NORTH**);

BorderLayout Constructors

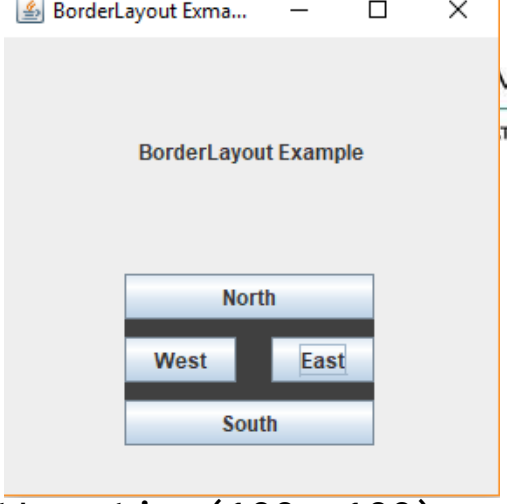
- BorderLayout ()

- BorderLayout (int hgap, int vgap)

```java
import java.awt.BorderLayout;
import javax.swing.*;
public class BorderLayoutEg extends JFrame{
    JButton b1=new JButton();
    JButton b2=new JButton("Two");
    JButton b3=new JButton("Three");
    JButton b4=new JButton("Four");
    JButton b5=new JButton("Five");
public BorderLayoutEg() {
    setTitle("BorderLayout Example");
    setLocation(100,100);
    setSize(300, 200);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
    b1.setText("One");
    this.add(b1,BorderLayout.NORTH);
    this.add(b2,BorderLayout.WEST);
    this.add(b3,BorderLayout.CENTER);
    this.add(b4,BorderLayout.EAST);
    this.add(b5,BorderLayout.SOUTH);
}
public static void main(String[] args) {
    new BorderLayoutEg();
}
}
```

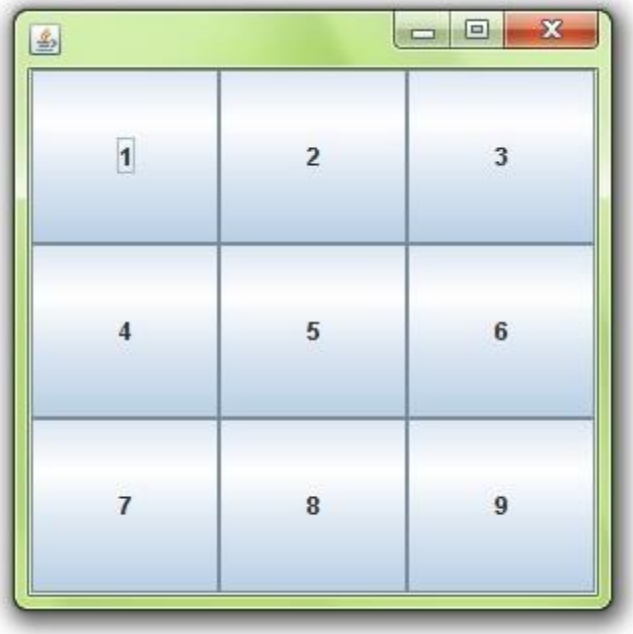


```java
import java.awt.*;
import javax.swing.*;
public class LayoutEg1 {
    JFrame frame;
    JPanel panel, panel1;
    JLabel lbl;
    JButton btn, btn1, btn2, btn3;
LayoutEg1(){
    frame=new JFrame("BorderLayout Exmaple");
    frame.setSize(300, 300);
    frame.setLayout(new GridLayout(2, 1));frame.setLocation(100, 100);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);frame.setVisible(true);
    lbl=new JLabel("",JLabel.CENTER);lbl.setText("BorderLayout Example");
    frame.add(lbl);
    panel=new JPanel();panel.setLayout(new FlowLayout());
    panel1=new
    JPanel();panel1.setBackground(Color.DARK_GRAY);panel1.setSize(300,300);
    BorderLayout layout=new BorderLayout();
    layout.setHgap(10);layout.setVgap(10);
    panel1.setLayout(layout);
    btn=new JButton("East"); btn1=new JButton("West");btn2=new JButton("North");
    btn3=new JButton("South");
    panel1.add(btn,BorderLayout.EAST);panel1.add(btn1,BorderLayout.WEST);
    panel1.add(btn2,BorderLayout.NORTH);panel1.add(btn3,BorderLayout.SOUTH);
    panel.add(panel1);
    frame.add(panel);
}
public static void main(String[] args) {
new LayoutEg1();}}
```



GridLayout

- arranges components into regularly spaced rows and columns, in other words, GridLayout takes the number of rows and columns in its constructor
- is most useful for arranging identically sized objects
- The components are arbitrarily resized to fit the grid; their minimum and preferred sizes are consequently ignored.
- For example,
  - GridLayout(2,0) requests a layout with two rows and an unlimited number of columns
  - Grid.java – Page 712 ~ 713



Usage:

```
setLayout(new GridLayout(row, col));//3,3
add(button);
```

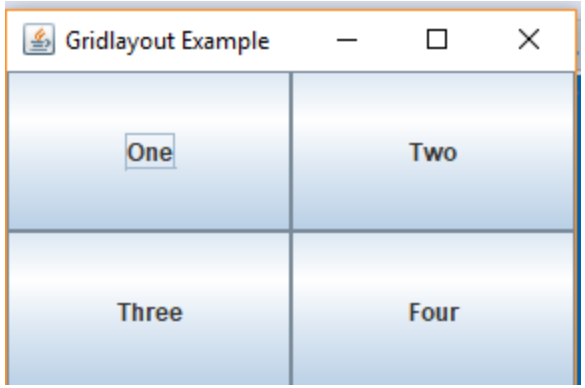
Grid Layout Constructors

- GridLayout ()

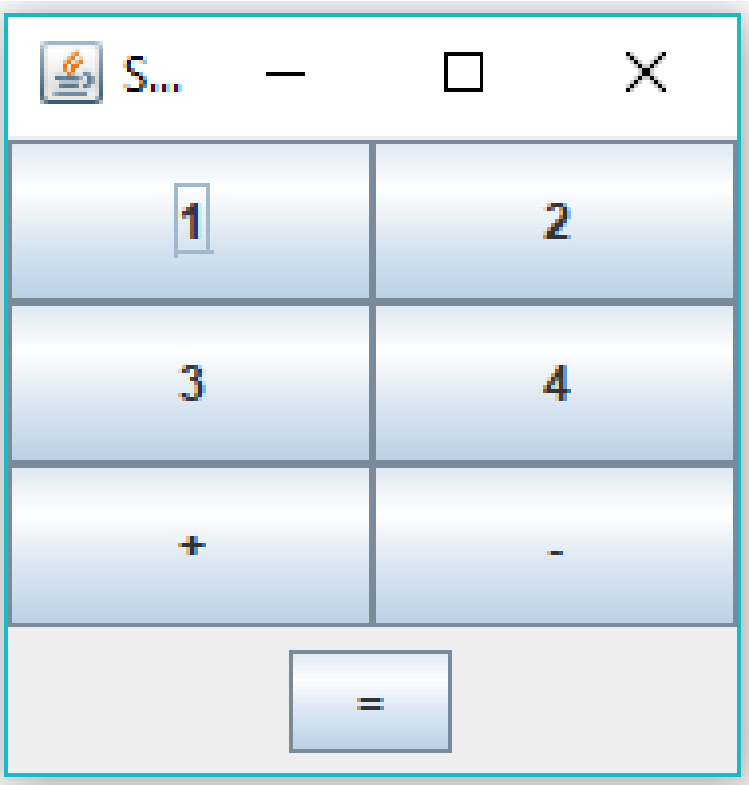
- GridLayout (int rows, int cols)

- GridLayout (int rows, int cols, int hgap, int vgap)

```java
import java.awt.GridLayout;
import javax.swing.*;
public class GridLayoutEg extends JFrame{
    JButton b1=new JButton("One");
    JButton b2=new JButton("Two");
    JButton b3=new JButton("Three");
    JButton b4=new JButton("Four");

public GridLayoutEg() {
    setTitle("Gridlayout Example");
    setLocation(100, 100);
    setSize(300, 200);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
    this.setLayout(new GridLayout(2, 2));
    this.add(b1);
    this.add(b2);
    this.add(b3);
    this.add(b4);
}
public static void main(String[] args) {
    new GridLayoutEg();
}
}
```



exercise

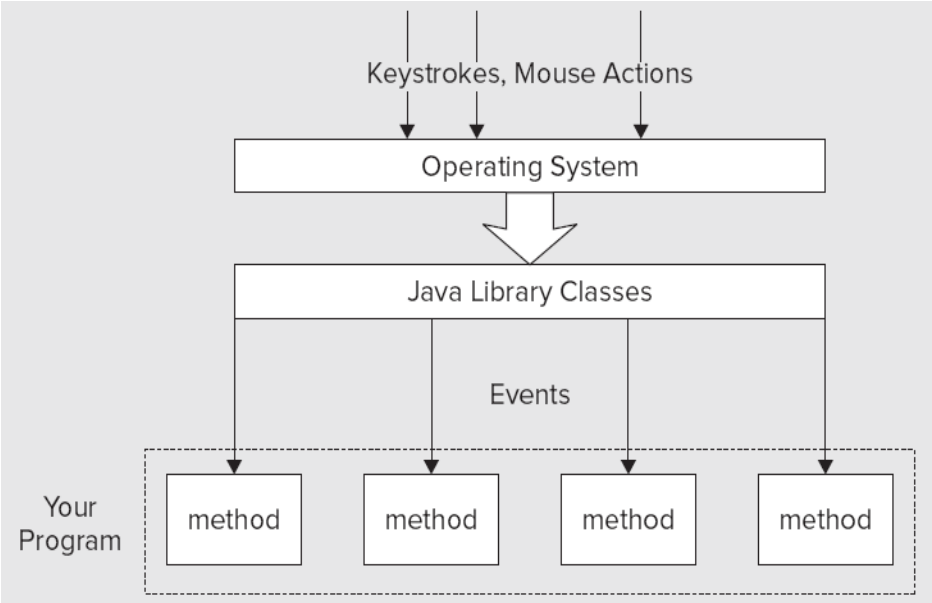


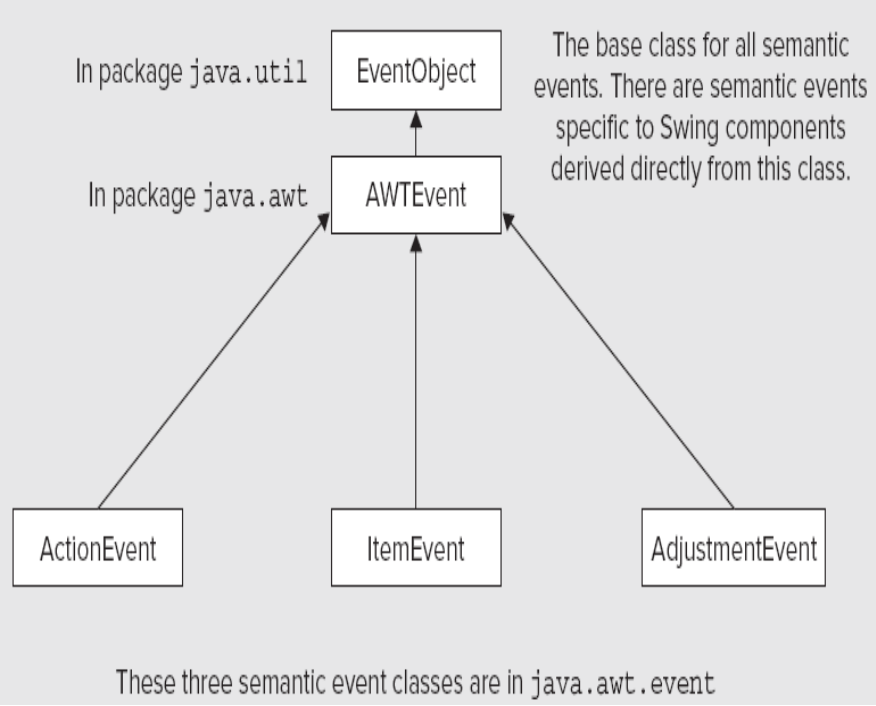
Events

- Change in the state of an object is known as Event, i.e., event describes the change in the state of the source.
- Events are generated as a result of user interaction with the graphical user interface components.
- An event is simply an ordinary Java object that is delivered to its receiver by invoking an ordinary Java method
- Two Categories of Events
  - Low-Level Events - arise from the keyboard or from the mouse, or events associated with operations on a window
  - Semantic Events - specific GUI component-level events

Basic Idea of how actions and events are communicated with Java Program

Semantic Event Classes



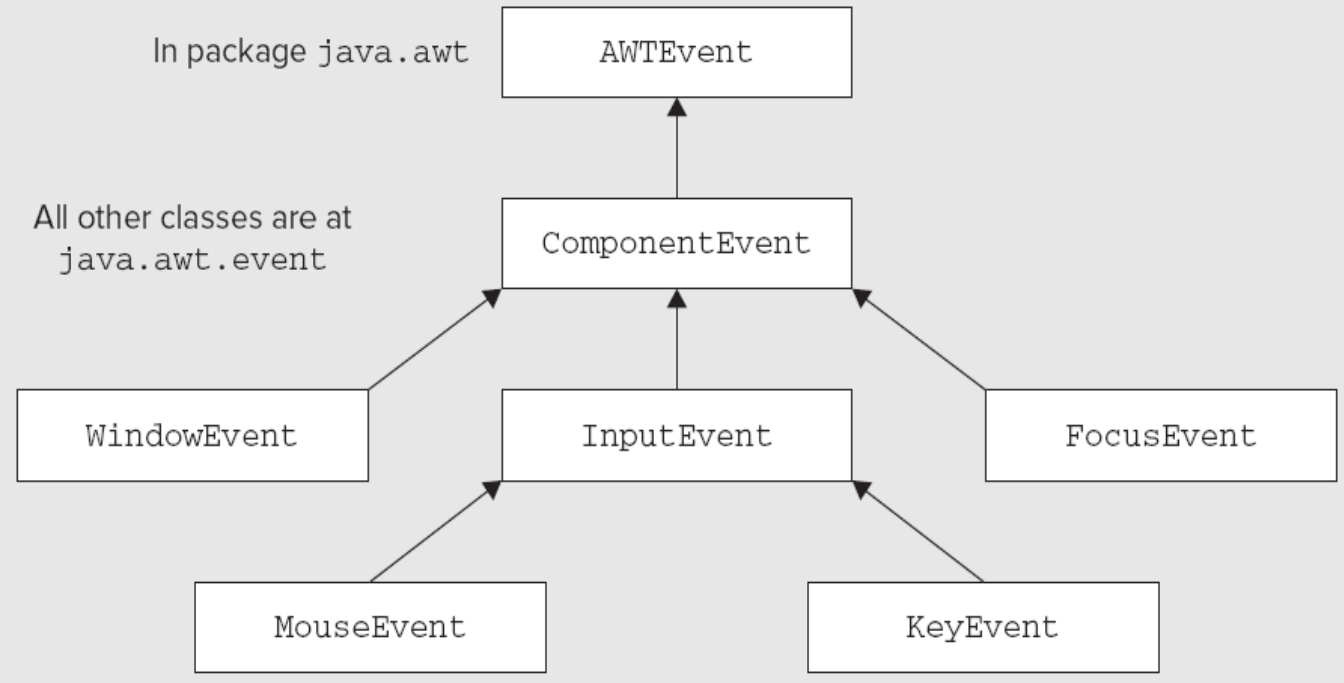


Basic Idea of how actions and events are communicated with Java Program

Event Classes - Relationship

- An event object is an instance of a subclass of **`java.util.EventObject,`** `it holds information about something that's happened to its source.`Components don't normally send or receive EventObjects as such; they work with subclasses that provide more specific information
- AWTEvent is a subclass of java.awt.EventObject; further subclasses of AWTEvent provide information about specific event types.
- Swing has events of its own that descend directly from EventObject.
- ActionEvents correspond to "action" that a user has taken with the component, such as clicking a button or pressing Enter.
- An ActionEvent carries the name of an action to be performed(the action command) by the program.

Event Classes - Relationship



Event Receivers and Listener Interfaces

- An object that waits for events and responds to them
  - To handle an event, attach a listener to a component
  - The listener will be notified when the event occurs(eg. Button click)
- An event is delivered by passing it as an argument to the receiving object's event handler method. ActionEvents, for example, are always delivered to a method called **actionPerformed()** in the receiver:
- Any object that receives ActionEvents must implement the **ActionListener** interface
- To manage its listeners, an ActionEvent source always implements two methods:

```
// ActionEvent source
public void addActionListener(ActionListener listener) {
    ...
}
public void removeActionListener(ActionListener listener) {
    ...
}
```

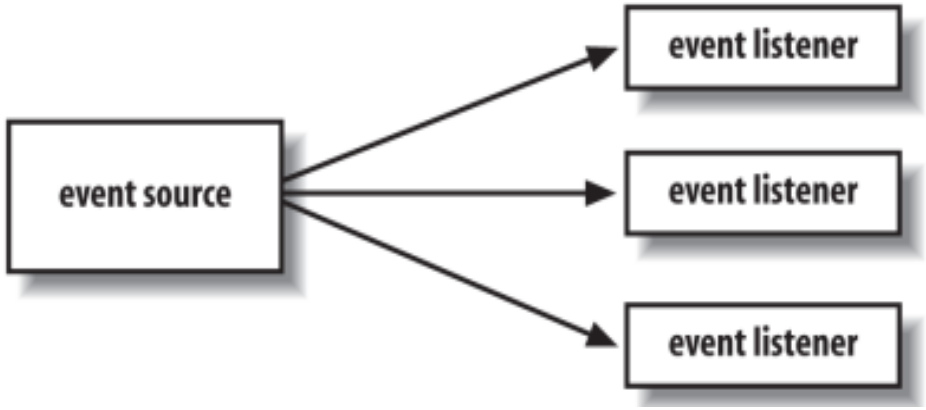
Event Sources

- How a receiver tells an event source to send it events
  - An eligible listener must register itself with an event source
  - Makes a call to "add Listener" method in the event source and passes a reference to itself

```
// receiver of ActionEvents
class TheReceiver implements ActionListener
{
   // source of ActionEvents
   JButton theButton = new JButton("Belly");

   TheReceiver() {
      ...
      theButton.addActionListener( this );
   }

   public void actionPerformed( ActionEvent e ) {
      // Belly Button pushed...
   }
```

Event Delivery

- Swing and AWT events are multicast;
- Every event is associated with a single source but can be delivered to any number of receivers
- No guarantees about the order in which events are delivered



Event Summary

Table 16-1. Swing component and container events

Event	Fired by	Listener interface	Handler methods

`java.awt.event.ComponentEvent`	All components	`ComponentListener`	`componentResized()`
			`componentMoved()`
			`componentShown()`
			`componentHidden()`

`java.awt.event.FocusEvent`	All components	`FocusListener`	`focusGained()` `focusLost()`

`java.awt.event.KeyEvent`	All components	`KeyListener`	`keyTyped()`
			`keyPressed()`
			`keyReleased()`

`java.awt.event.MouseEvent`	All components	`MouseListener`	`mouseClicked()`
			`mousePressed()`
			`mouseReleased()`
			`mouseEntered()`
			`mouseExited()`

		`MouseMotionListener`	`mouseDragged()` `mouseMoved()`
`java.awt.event.ContainerEvent`	All containers	`ContainerListener`	`componentAdded()` `componentRemoved()`

Table 16-2. Component-specific swing events

Event	Fired by	Listener interface	Handler method

`java.awt.event.ActionEvent`	`JButton`
	`JCheckBoxMenuItem`
	`JComboBox`
	`JFileChooser`
	`JList`
	`JRadioButtonMenuItem`
	`JTextField`
	`JToggleButton`
		`ActionListener`	`actionPerformed()`

`java.awt.event.AdjustmentEvent`	`JScrollBar`	`Adjustment-Listener`	`adjustmentValue-Changed()`
`javax.swing.event.CaretEvent`	`JTextComponent`	`CaretListener`	`caretUpdate()`

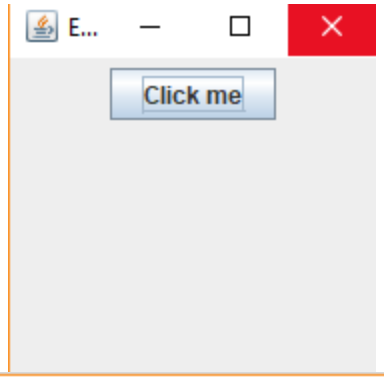
Event	Fired by	Listener interface	Handler method
javax.swing.event.HyperlinkEvent	JEditorPane JTextPane	HyperlinkListener	hyperlinkUpdate()
java.awt.event.InternalFrameEvent	JInternalFrame	InternalFrameListener	internalFrameActivated() internalFrameClosed() internalFrameClosing() internalFrameDeactivated() internalFrameDeiconified() internalFrameIconified() internalFrameOpened()
java.awt.event.ItemEvent	JCheckBoxMenuItem JComboBox JRadioButtonMenuItem JToggleButton	ItemListener	itemStateChanged()

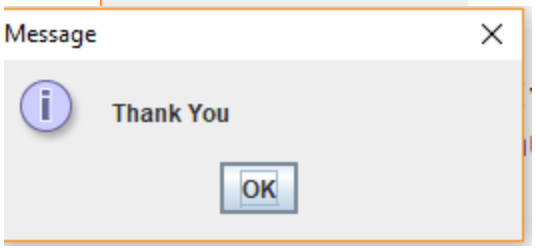
javax.swing.event.ListDataEvent	ListModel	ListDataListener	contentsChanged() intervalAdded() intervalRemoved()
javax.swing.event.ListSelectionEvent	JList ListSelectionModel	ListSelection-Listener	valueChanged()
javax.swing.event.MenuEvent	JMenu	MenuListener	menuCanceled() menuDeselected() menuSelected()
javax.swing.event.PopupMenuEvent	JPopupMenu	PopupMenu-Listener	popupMenuCanceled() popupMenuWill-BecomeInvisible() popupMenuWill-BecomeVisible()
javax.swing.event.MenuKeyEvent	JMenuItem	MenuKeyListener	menuKeyPressed() menuKeyReleased() menuKeyTyped()

Event	Fired by	Listener interface	Handler method
javax.swing.event.MenuDragMouseEvent	JMenuItem	MenuDragMouse-Listener	menuDragMouse-Dragged() menuDragMouse-Entered() menuDragMouse-Exited() menuDragMouse-Released()
javax.swing.event.TableColumnModelEvent	TableColumnModel*	TableColumn-ModelListener	columnAdded() columnMargin-Changed() columnMoved() columnRemoved() columnSelection-Changed()
javax.swing.event.TableModelEvent	TableModel	TableModel-Listener	tableChanged()
javax.swing.event.TreeExpansionEvent	Jtree	TreeExpansion-Listener	treeCollapsed() treeExpanded()
javax.swing.event.TreeModelEvent	TreeModel	TreeModel-Listener	treeNodesChanged() treeNodesInserted() treeNodesRemoved() treeStructure-Changed()
javax.swing.event.TreeSelectionEvent	JTree TreeSelectionModel	TreeSelection-Listener	valueChanged()
javax.swing.event.UndoableEditEvent	jav ax.swing.text.Document	UndoableEdit-Listener	undoableEdit-Happened()
java.awt.event.WindowEvent	JDialog JFrame JWindow	WindowListener	windowOpened() windowClosing() windowClosed() windowIconified() windowDeiconified() windowActivated() windowDeactivated()

UNIVERSITY of INFORMATION TECHNOLOGY

```java
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
public class EventEg extends JFrame implements ActionListener{
    JButton b=new JButton("Click me");
public EventEg() {
    setTitle("Event Example");
    setLocation(100, 100);
    setSize(200, 200);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
    setLayout(new FlowLayout());
    b.addActionListener(this);
    this.add(b);
}
public static void main(String[] args) {
    new EventEg();
}
public void actionPerformed(ActionEvent arg0) {
    JOptionPane.showMessageDialog(null, "Thank You");
}
}
```

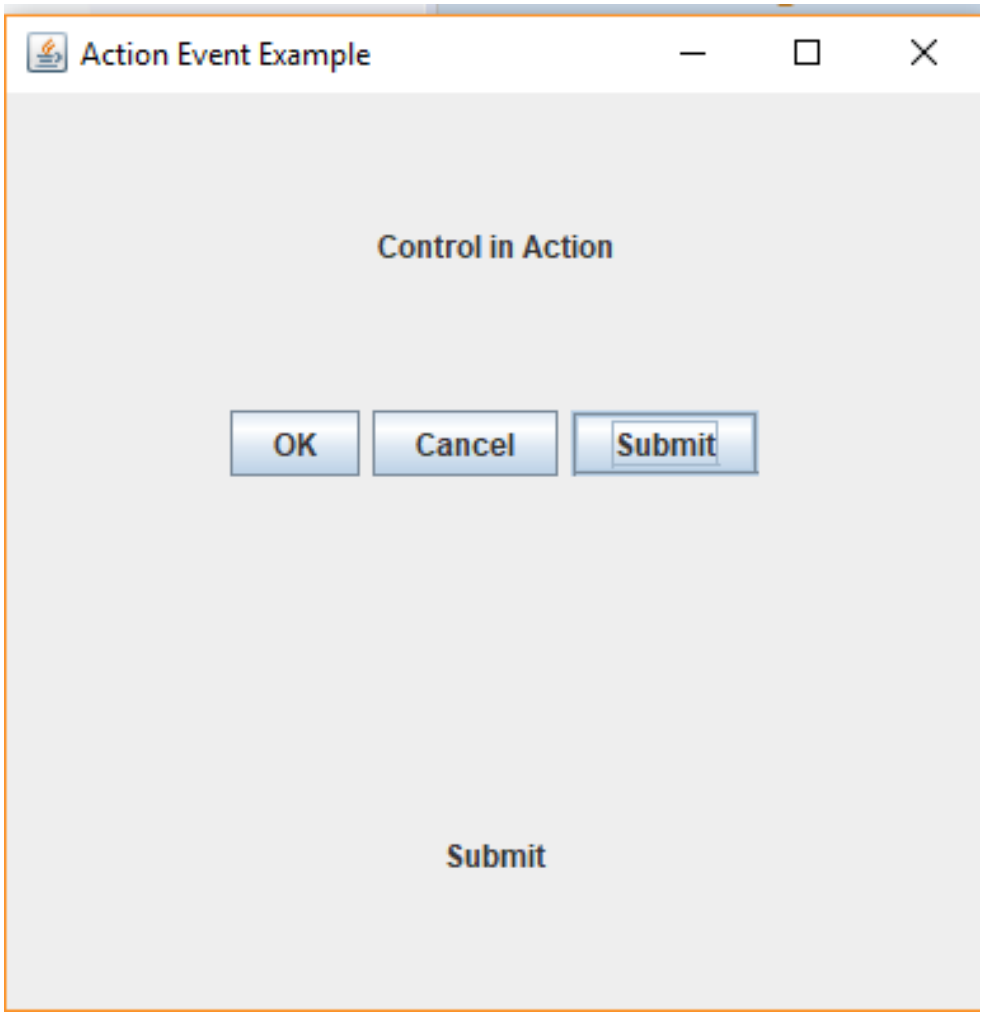




```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class SwingDemo implements ActionListener{
JFrame frame;
JPanel panel, panel1;
JLabel headerlbl,statuslbl;
JButton ok,cancel,submit;

SwingDemo(){
frame=new JFrame("BorderLayout Exmaple");
frame.setSize(400, 400);
frame.setLayout(new GridLayout(3, 1));
frame.setLocation(100, 100);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);

headerlbl=new
JLabel("",JLabel.CENTER);
headerlbl.setText("Control in
Action");

statuslbl=new
JLabel("",JLabel.CENTER);
statuslbl.setSize(350,100);
```

```java
panel=new JPanel();
panel.setLayout(new FlowLayout());

frame.add(headerlbl);
frame.add(panel);
frame.add(statuslbl);

ok=new JButton("OK");
cancel=new JButton("Cancel");
submit=new JButton("Submit");

ok.setActionCommand("OK");
cancel.setActionCommand("Cancel");
submit.setActionCommand("Submit");

ok.addActionListener(this);
cancel.addActionListener(this);
submit.addActionListener(this);

panel.add(ok);
panel.add(cancel);
panel.add(submit);

}
```

```java
public static void main(String[]
args) {
new SwingDemo();
}
public void
actionPerformed(ActionEvent e) {
/*if(ok==e.getSource()) {
statuslbl.setText("OK");
}
else if(cancel==e.getSource()) {
statuslbl.setText("Cancel");
}
else {
statuslbl.setText("Submit");
}*/
String command=e.getActionCommand();
if(command.equals("OK“)) {
statuslbl.setText("OK");}
else if(command.equals("Cancel")) {
statuslbl.setText("Cancel");}
else {
statuslbl.setText("Submit");}
}
}
```


Action Event Example
Control in Action
OK
Cancel
Submit
Submit

AWT Robot

```
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.InputEvent;

public class RobotExample {

public static void main(String[] args) throws AWTException,
                                              InterruptedException {
    Robot r = new Robot();
    r.mouseMove(300,35);
    r.mousePress( InputEvent.BUTTON1_MASK );
    /*r.mouseRelease( InputEvent.BUTTON1_MASK );
    Thread.sleep(50);
    r.mousePress( InputEvent.BUTTON1_MASK );
    r.mouseRelease( InputEvent.BUTTON1_MASK );*/
  }
}
```