ADVANCED JAVA
SWEG2033

Advanced Programming

Code: SWEG2033

Chapter Two

AWT and Swing

Part One

Objective

- Within this chapter you will be introduced to many of the Java components for constructing graphical user interfaces (GUIs).
- With the information in this chapter you will be able to develop your own GUIs.

Introduction

- Graphical User Interface (GUI) offers user interaction via some graphical components. E.g. window, frame, Panel, Button, Textfield, Text-Area, List-box, Combo- box, Label, Checkbox etc. These all are known as components. Using these components we can create an interactive user interface for an application.
- GUI provides result to end user in response to raised events. GUI is entirely based events.
- For example clicking over a button, closing a window, opening a window, typing something in a textarea etc. These activities are known as events.

Introduction

- GUI makes it easier for the end user to use an application. It also makes them interesting.
- A GUI gives an application a distinctive “look” and “feel”.
- There are many sets of visual components and containers for user interface design in JAVA. But the two are basic :
  - AWT (Abstract Window Toolkit) - in package **java.awt** and
  - Swing - in package **javax.swing**

Introduction

Class Activity 1

Discuss about GUI and User Experience (UIX)

What is the Advantage of GUI over Character based system

Advantages of GUI over CUI

- GUI provides graphical icons to interact while the CUI (Character or command User Interface) offers the simple text-based interfaces.
- GUI makes the application more entertaining and interesting on the other hand CUI does not.
- GUI offers click and execute environment while in CUI every time we have to enter the command for a task.
- GUI provides multitasking environment so as the CUI also does but CUI does not provide same ease as the GUI do.

Class Activity 2 :

List and Discuss GUI based Application

GUI based Applications

Following are some of the examples for GUI based applications:

- Automated Teller Machine (ATM)
- Airline Ticketing System
- Information Kiosks at railway stations
- Mobile Applications
- Navigation System

Basic Terminology of GUI

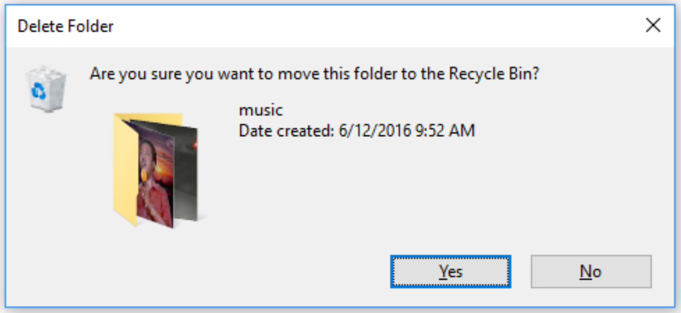
1. **Component :** is an object having a graphical representation that can be displayed on the screen and that can interact with the user. For examples buttons, checkboxes, list and scrollbars of a graphical user interface.
2. **Container :** The Container is a component that can contain another components like buttons, text fields, labels etc. The classes that extends Container class are known as container such as Frame, Dialog and Panel.
3. **Panel :** The Panel is the container that doesn't contain title bar and menu bars. It can have other components like button, text field etc.

ADVANCED JAVA SWEG2033

Basic Terminology of GUI

4. Window :

- Window is a rectangular area which is displayed on the screen.
- In different window we can execute different program and display different data.
- Window provide us with multitasking environment.(e.g., modal window)
- A window must have either a frame, dialog, or another window defined as its owner when it's constructed.



Are you sure you want to move this folder to the Recycle Bin?

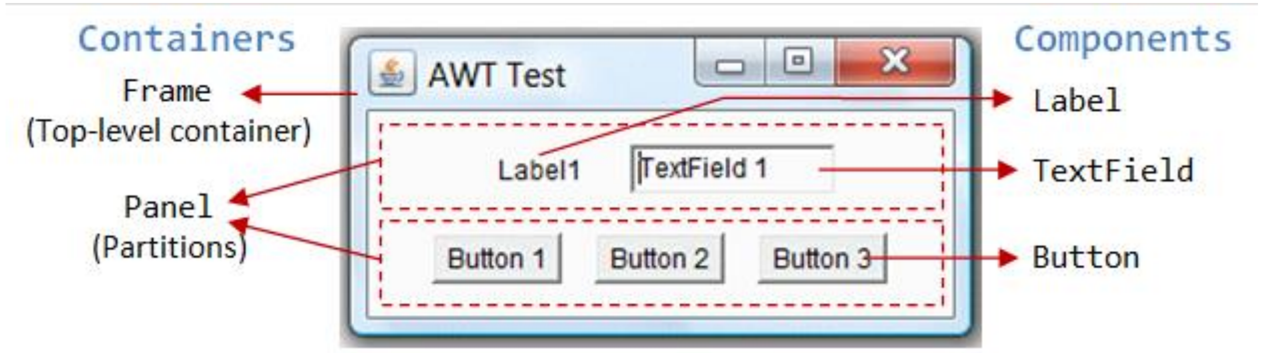
music
Date created: 6/12/2016 9:52 AM

Basic Terminology of GUI

5. **Frame :** A Frame is a top-level window with a title and a border. The size of the frame includes any area designated for the border. Frame encapsulates **window**. It and has a title bar, menu bar, borders, and resizing corners.

6. **Canvas:** Canvas component represents a blank rectangular area of the screen onto which the application can draw.

Basic Terminology of GUI

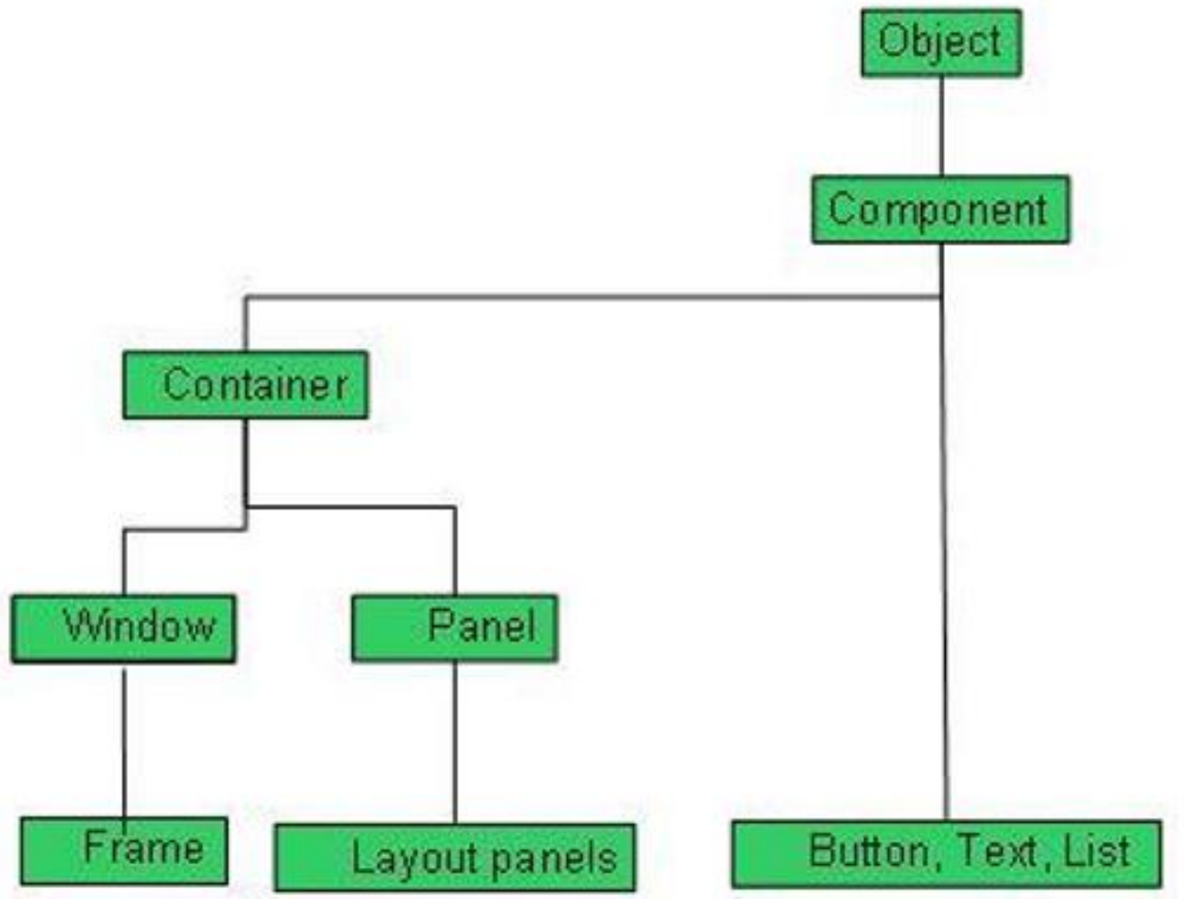


AWT

- The **AWT** (**Abstract Window Toolkit**) package is java package designed for doing windowing interfaces.
- Every user interface considers the following three main aspects:

1. **UI elements:** These are the core visual elements, the user eventually sees and interacts with.
2. **Layouts:** They define how UI elements should be organized on the screen and provide a final look and feel to the GUI (Graphical User Interface).
3. **Behavior:** These are events that occur when the user interacts with UI elements.

AWT


Object
Component
Container
Window
Panel
Frame
Layout panels
Button, Text, List

AWT

- The class Component is the abstract base class for the non-menu user interface controls of AWT. Component represents an object with graphical representation.
- Following is the declaration for java.awt.Component class:

```
public abstract class Component
   extends Object
      implements ImageObserver, MenuContainer, Serializable
```

AWT

Following are the **fields** for java.awt.Component class:

- static float BOTTOM_ALIGNMENT : Ease-of-use constant for getAlignmentY.
- static float CENTER_ALIGNMENT : Ease-of-use constant for getAlignmentY and getAlignmentX.
- static float LEFT_ALIGNMENT : Ease-of-use constant for getAlignmentX.
- static float RIGHT_ALIGNMENT : Ease-of-use constant for getAlignmentX.
- static float TOP_ALIGNMENT : Ease-of-use constant for getAlignmentY().

AWT

Methods of AWT

1. protected Component() – Constructor functions.
2. boolean action(Event evt, Object what): should register this component as ActionListener on component which fires action events.
3. void addComponentListener(ComponentListener l): Adds the specified component listener to receive component events from this component.

N.B: In general in AWT we have almost 221 for deferent purpose.

See the reference “awt tutorial” book from tutorial point site page number 7-19

AWT vs. Swing

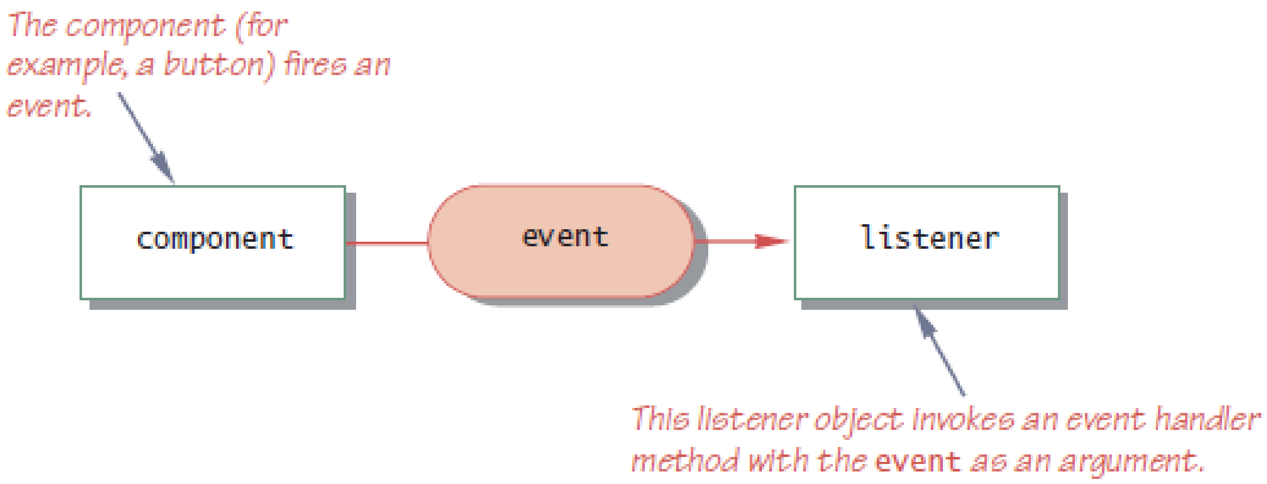
- The **AWT** (**Abstract Window Toolkit**) package is java package designed for doing windowing interfaces. Swing can be viewed as an improved version of the AWT.
- However, Swing did not completely replace the AWT package. Some AWT classes are replaced by Swing classes, but other AWT classes are needed when using Swing. We will use classes from both Swing and the AWT.
- Swing GUIs are designed using a particular form of object-oriented programming that is known as *event-driven programming*.
- **Event-driven programming** is a programming style that uses a signal-and-response approach to programming. Signals to objects are things called events.

AWT vs. Swing

- AWT is fine for developing simple graphical user interfaces, but not for developing comprehensive GUI projects.
- Besides, AWT is prone to platform-specific bugs.
- The AWT user-interface components were replaced by a more robust, versatile, and flexible library known as *Swing components*.
- Swing components depend less on the target platform and use less of the native GUI resource.
- Swing components are painted directly on canvases using Java code.
- For this reason, Swing components that don't rely on native GUI are referred to as *lightweight components*, *and AWT components* are referred to as *heavyweight components*.

AWT vs. Swing

- Swing programs use events and event handlers. An **event** is an object that acts as a signal to another object known as a **listener**.
- The sending of the event is called firing the event.


The component (for example, a button) fires an event.
component
event
listener
This listener object invokes an event handler method with the event as an argument.

AWT vs. Swing (cont'd)

AWT features include:

- A rich set of user interface components.
- A robust event-handling model.
- Graphics and imaging tools, including shape, color, and font classes.

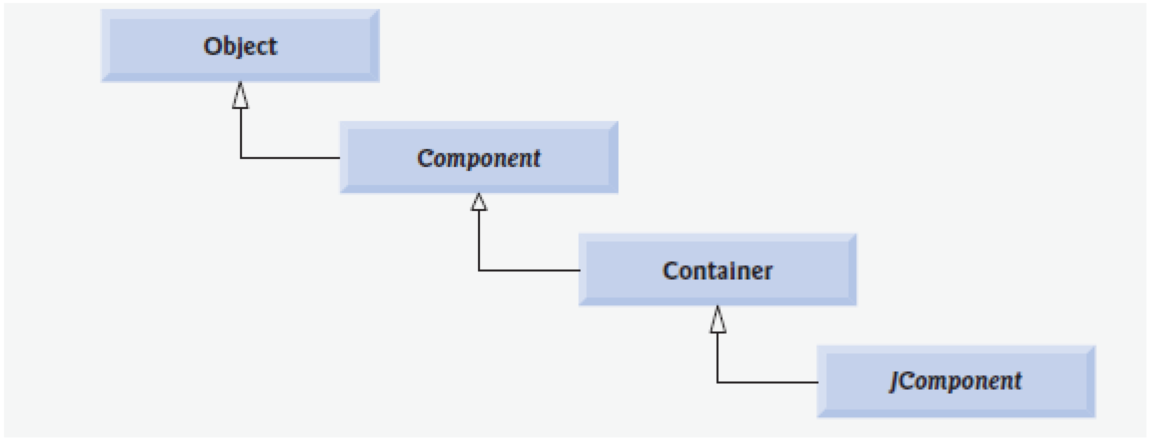
Swing features include:

- All the features of AWT.
- 100% Pure Java certified versions of the existing AWT component set (Button, Scrollbar, Label, etc.).
- A rich set of higher-level components (such as tree view, list box, and tabbed panes).
- Pure Java design, no reliance on peers.
- Pluggable Look and Feel.

Java GUI API

- The GUI API contains classes that can be classified into three groups:
  - *component classes*
  - *container classes*
  - *helper classes*
- The component classes, such as **JButton, JLabel, and JTextField,** are for creating the user interface.
- The container classes, such as **JFrame, JPanel, and JApplet,** are used to contain other components.
- The helper classes, such as **Graphics, Color, Font, FontMetrics, and Dimension,** are used to support GUI components.

Java GUI API (cont'd)

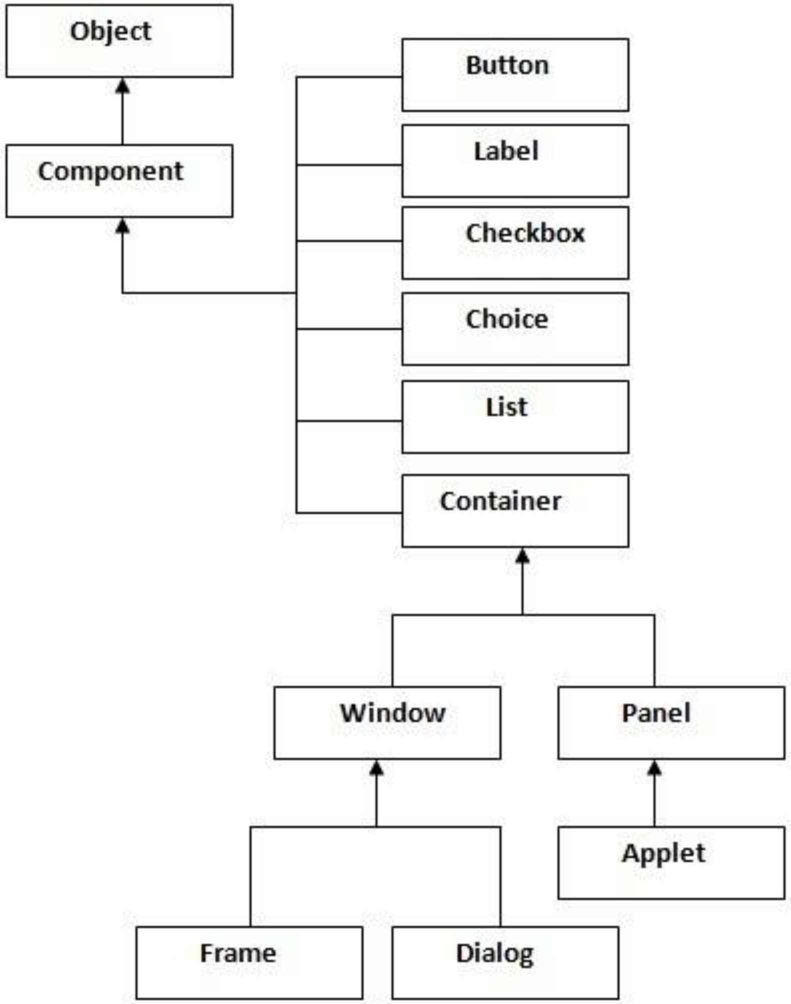


Common super classes of many of the Swing components

Java GUI API (cont'd)

- Class Object is the super class of the Java class hierarchy.
- **Component** is the root class of all the user-interface classes including container classes.
- An instance of Container can hold instances of Component.
- Window, Panel, Applet, Frame, and Dialog are the container classes for AWT components.
- To work with Swing components, use Container, JFrame, JDialog, JApplet, and Jpanel.

AWT Hierarchy



Useful Methods of Component class

Method	Description
public void add(Component c)	inserts a component on this component.
public void setSize(int width,int height)	sets the size (width and height) of the component.
public void setLayout(LayoutManager m)	defines the layout manager for the component.
public void setVisible(boolean status)	changes the visibility of the component, by default false.

Frames

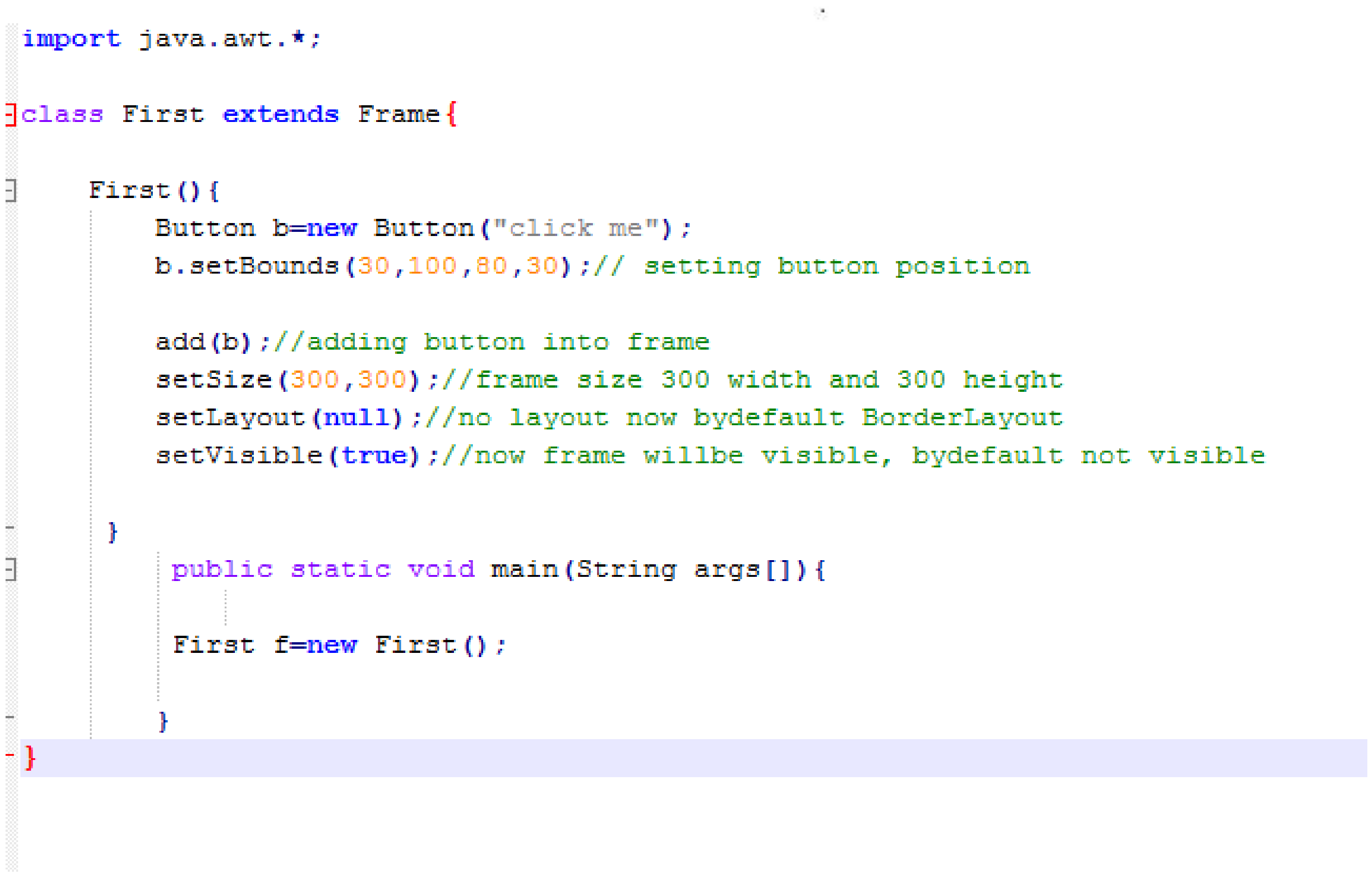
- To create a user interface, you need to create either a frame or an applet to hold the user-interface components.
- A top-level window (that is, a window that is not contained inside another window) is called a frame in Java.
- A Frame has:
  - a title bar (containing an icon, a title, and the minimize/maximize(restore-down)/close buttons),
  - an optional menu bar and
  - the content display area.
- To create a frame, use the **JFrame class**

Frames

There are two ways to create a frame in AWT.

- By extending Frame class (inheritance)
- By creating the object of Frame class (association)

Frames Example by Inheritance



```
import java.awt.*;

class First extends Frame{

        First(){
                Button b=new Button("click me");
                b.setBounds(30,100,80,30);// setting button position

                add(b);//adding button into frame
                setSize(300,300);//frame size 300 width and 300 height
                setLayout(null);//no layout now bydefault BorderLayout
                setVisible(true);//now frame willbe visible, bydefault not visible

          }
                  public static void main(String args[]){

                  First f=new First();

                }
}
```

ADVANCED JAVA
SWEG2033

AWT Example by Association

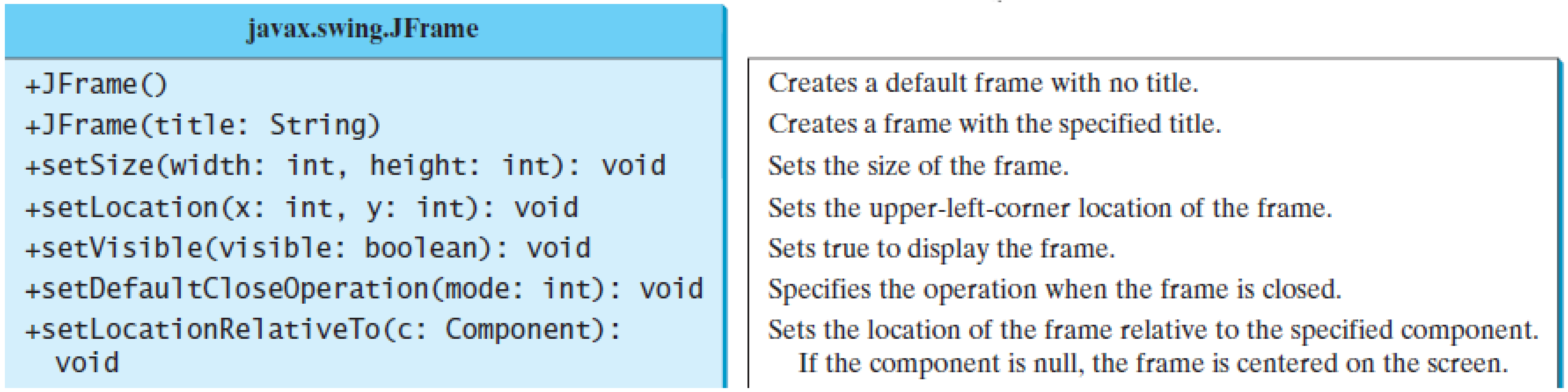
```
import java.awt.*;
class First2{
First2(){
Frame f=new Frame();
Button b=new Button("click me");
b.setBounds(30,50,80,30);
f.add(b);
f.setSize(300,300);
f.setLayout(null);
f.setVisible(true);
}
public static void main(String args[]){
First2 f=new First2();
}}
```

Frame Swing

Example 1

```
import javax.swing.JFrame;
public class MyFrame {
    public static void main(String[] args) {
            // Create a frame
            JFrame frame = new JFrame("MyFrame");
            frame.setSize(400, 300); // Set the frame size
            // Center a frame
            frame.setLocationRelativeTo(null); //or .setLocation(300,200)
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setVisible(true); // Display the frame
   }
 }
```

Frame (cont'd)



javax.swing.JFrame	
`+JFrame()`	Creates a default frame with no title.
`+JFrame(title: String)`	Creates a frame with the specified title.
`+setSize(width: int, height: int): void`	Sets the size of the frame.
`+setLocation(x: int, y: int): void`	Sets the upper-left-corner location of the frame.
`+setVisible(visible: boolean): void`	Sets true to display the frame.
`+setDefaultCloseOperation(mode: int): void`	Specifies the operation when the frame is closed.
`+setLocationRelativeTo(c: Component): void`	Sets the location of the frame relative to the specified component. If the component is null, the frame is centered on the screen.

JFrame is a top-level container to hold GUI components

Frame (cont’d)

Adding Components to a Frame

```
import javax.swing.JFrame;
import javax.swing.JButton;
public class FrameWithButton {
  public static void main(String[] args) {
            JFrame frame = new JFrame("MyFrame");
            // Add a button into the frame
            JButton jbtOK = new JButton("OK");
            frame.add(jbtOK);
            frame.setSize(400, 300);
             frame.setLocation(360,360);
             frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
             frame.setVisible(true);
   }
 }
```

Java Label

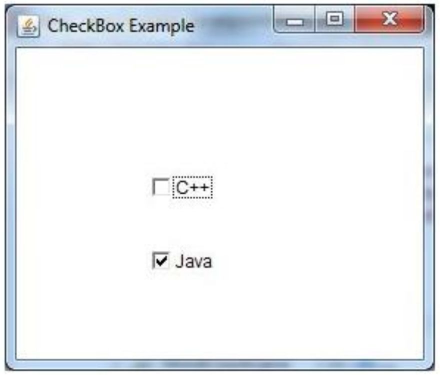
Adding Lablel

```
import javax.swing.JFrame;
import javax.swing.JLabel;
public class FrameWithLabel {
  public static void main(String[] args) {
            JFrame frame = new JFrame("MyFrame");
           // Add a lable into the frame
           JLabel jLblName = new JLabel(“First Name");
           frame.add(jLblName);
           frame.setSize(400, 300);
            frame.setLocation(360,360);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setVisible(true);
   }
 }
```

Java Checkbox

- The Checkbox class is used to create a checkbox. It is used to turn an option on (true) or off (false).
- Clicking on a Checkbox changes its state from "on" to "off" or from "off" to "on".

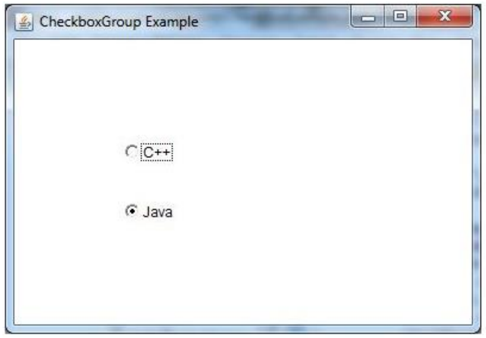
```
Frame f= new Frame("Checkbox Example");
 Checkbox checkbox1 = new Checkbox("C++");
 checkbox1.setBounds(100,100, 50,50);
 Checkbox checkbox2 = new Checkbox("Java", true);
 checkbox2.setBounds(100,150, 50,50);
 f.add(checkbox1);
 f.add(checkbox2);
```



Java CheckboxGroup

- The object of CheckboxGroup class is used to group together a set of Checkbox. At a time only one check box button is allowed to be in "on" state and remaining check box button in "off" state. It inherits the object class.

```
Frame f= new Frame("CheckboxGroup Example");
CheckboxGroup cbg = new CheckboxGroup();
Checkbox checkBox1 = new Checkbox("C++", cbg, false);
checkBox1.setBounds(100,100, 50,50);
Checkbox checkBox2 = new Checkbox("Java", cbg, true);
checkBox2.setBounds(100,150, 50,50);
f.add(checkBox1);
f.add(checkBox2);
```



Java Label

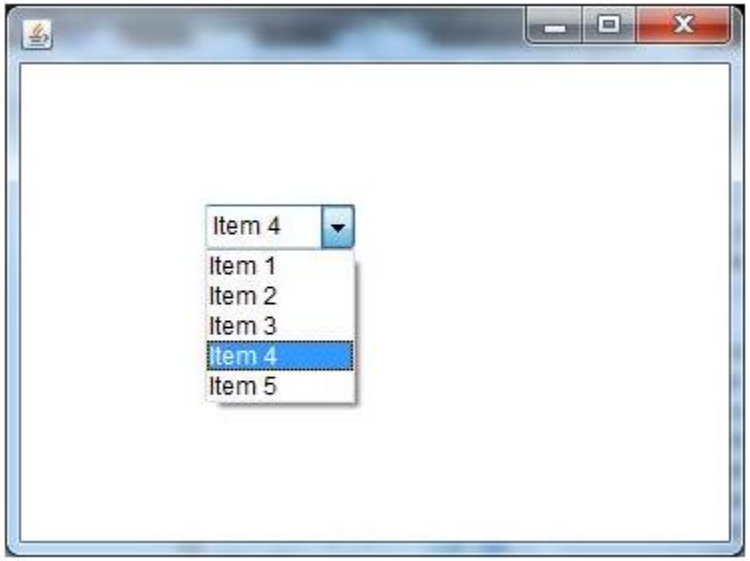
- The object of Label class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly.

```
Frame f= new Frame("Label Example");

Label l1,l2;

l1=new Label("First Label.");

l1.setBounds(50,100, 100,30);

l2=new Label("Second Label.");

l2.setBounds(50,150, 100,30);

f.add(l1); f.add(l2);
```

Java Choice

- The object of Choice class is used to show popup menu of choices. Choice selected by user is shown on the top of a menu. It inherits Component class.

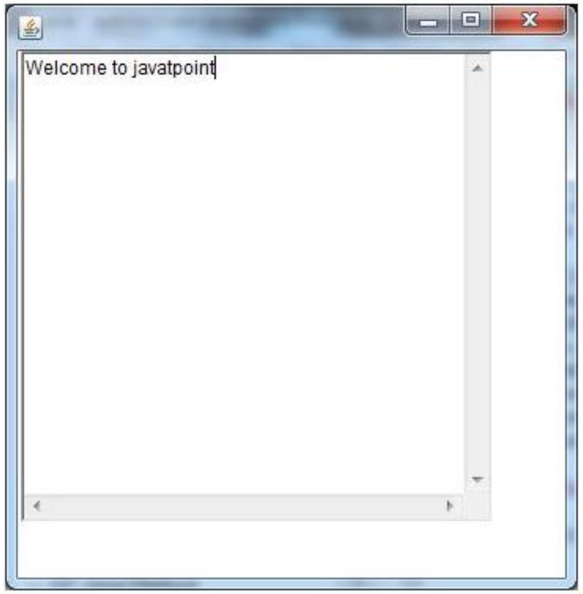
```
Frame f= new Frame();
Choice c=new Choice();
c.setBounds(100,100, 75,75);
c.add("Item 1");
c.add("Item 2");
c.add("Item 3");
c.add("Item 4");
c.add("Item 5");
f.add(c);
f.setSize(400,400);
```



Java TextArea

- The object of a TextArea class is a multi line region that displays text. It allows the editing of multiple line text. It inherits TextComponent class.

```
Frame f= new Frame();
    TextArea area=new TextArea("Welcome to javatpoint");
area.setBounds(10,30, 300,300);
f.add(area);
f.setSize(400,400);
```


Welcome to javatpoint

Java Button

- A button object is created in the same way that any other object is created, but you use the class JButton.

```
JButton endButton = new JButton("Click to end program.");

frameObjectVariable.add(endButton);
```

Redding Assignments

- Assignments 2 write a code for the following
- Java AWT Canvas
- Java AWT list
- Java AWT dialog
- Other GUI Componet

AWT and Swing

Part Two

Layout Management

Layout Managers

- Java’s layout managers provide a level of abstraction that automatically maps your user interface on all window systems.
- The Java GUI components are placed in containers, where they are arranged by the container’s layout manager.
- Layout managers are set in containers using the setLayout(aLayoutManager) method.
- This section introduces three basic layout managers: FlowLayout, GridLayout, and BorderLayout.

FlowLayout

FlowLayout

- Is the simplest layout manager.
- The components are arranged in the container from left to right in the order in which they were added. When one row is filled, a new row is started.
- You can specify the way the components are aligned by using one of three constants:
  - **FlowLayout.RIGHT,**
  - **FlowLayout.CENTER, or**
  - **FlowLayout.LEFT.**
- You can also specify the gap between components in pixels.

FlowLayout (cont’d)

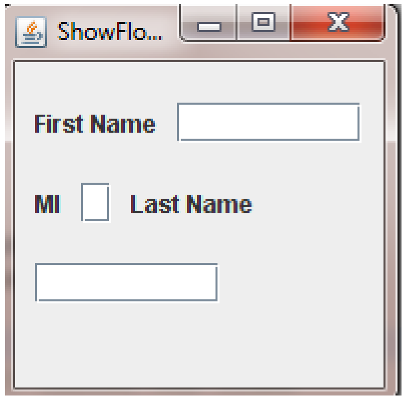
EXAMPLE

```
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JFrame;
import java.awt.FlowLayout;

public class ShowFlowLayout extends JFrame{
      public ShowFlowLayout() {
            // Set FlowLayout, aligned left with horizontal gap 10
            // and vertical gap 20 between components
             setLayout(new FlowLayout(FlowLayout.LEFT, 10, 20) );
            // Add labels and text fields to the frame
             add(new JLabel("First Name"));
             add(new JTextField(8));
             add(new JLabel("MI"));
             add(new JTextField(1));
             add(new JLabel("Last Name"));
             add(new JTextField(8));
       }
```

FlowLayout (cont’d)

```
        /** Main method */
    public static void main(String[] args) {
        ShowFlowLayout  frame = new ShowFlowLayout();
        frame.setTitle("ShowFlowLayout");
        frame.setSize(200, 200);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

Output



GridLayout

- Arranges components in a grid (matrix) formation.
- The components are placed in the grid from left to right, starting with the first row, then the second, and so on, in the order in which they are added.

GridLayout (cont'd)

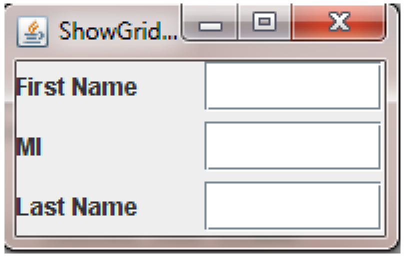
EXAMPLE

```
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JFrame;
import java.awt.GridLayout;

 public class ShowGridLayout extends JFrame {
     public ShowGridLayout() {
         // Set GridLayout, 3 rows, 2 columns, and gaps 5 between
         // components horizontally and vertically
         setLayout(new GridLayout(3, 2, 5, 5));

         // Add labels and text fields to the frame
         add(new JLabel("First Name"));
         add(new JTextField(8));
         add(new JLabel("MI"));
         add(new JTextField(1));
         add(new JLabel("Last Name"));
         add(new JTextField(8));
   }
```

GridLayout (cont'd)

```
        /** Main method */
         public static void main(String[] args) {
               ShowGridLayout frame = new ShowGridLayout();
               frame.setTitle("ShowGridLayout");
               frame.setSize(200, 125);
               frame.setLocationRelativeTo(null); // Center the frame
               frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
               frame.setVisible(true);
         }
}
```

Output



ShowGrid...

First Name

MI

Last Name

BorderLayout

- Divides a container into five areas: East, South, West, North, and Center.
- Components are added to a BorderLayout by using add(Component, index), where index is a constant
  - BorderLayout.EAST,
  - BorderLayout.SOUTH,
  - BorderLayout.WEST,
  - BorderLayout.NORTH, or
  - BorderLayout.CENTER.

BorderLayout

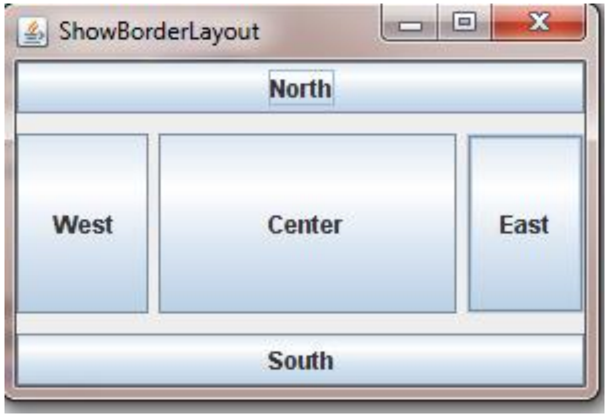
- **EXAMPLE**

```
import javax.swing.JButton;
 import javax.swing.JFrame;
import java.awt.BorderLayout;
public class ShowBorderLayout extends JFrame
 {
      public ShowBorderLayout() {
      // Set BorderLayout with horizontal gap 5 and vertical gap 10
      setLayout( new BorderLayout(5, 10));

      // Add buttons to the frame
      add(new JButton("East"), BorderLayout.EAST);
      add(new JButton("South"), BorderLayout.SOUTH);
      add(new JButton("West"), BorderLayout.WEST);
      add(new JButton("North"), BorderLayout.NORTH);
      add(new JButton("Center"), BorderLayout.CENTER);
 }
```

BorderLayout (cont’d)

```
    /** Main method */
     public static void main(String[] args) {
          ShowBorderLayout frame = new ShowBorderLayout();
          frame.setTitle("ShowBorderLayout");
          frame.setSize(300, 200);
          frame.setLocationRelativeTo(null); // Center the frame
          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          frame.setVisible(true);
     }
}
```

Output



Panels

- With Java GUI programming, you can divide a window into panels.
- Panels act as sub containers to group user-interface components.
- You add the buttons in one panel, then add the panel into the frame.
- You can use new JPanel() to create a panel with a default FlowLayout manager or new Panel(LayoutManager) to create a panel with the specified layout manager.

Panels (cont’d)

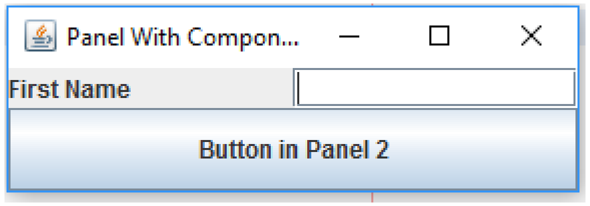
Example 1

```
import java.awt.GridLayout;
import javax.swing.*;
import java.awt.BorderLayout;
 public class SimplePanel1 extends JFrame {
 public SimplePanel1() {
       // Create panel p1 for the buttons and set GridLayout
      JPanel p1 = new JPanel();
      p1.setLayout(new GridLayout(1, 2));
      //Add label and text field to the panel
        p1.add(new JLabel("First Name"));
       p1.add(new JTextField(8));
   // Create panel p2 to hold a p1
       JPanel p2 = new JPanel(new BorderLayout());
       p2.add(p1, BorderLayout.NORTH);
       p2.add (new JButton("Button in Panel 2"));
        // add contents into the frame
       add(p2);
       }
```

Panels (cont’d)

```
/** Main method */
 public static void main(String[] args) {
        SimplePanel1 frame = new SimplePanel1();
        frame.setTitle("Panel With Components");
        frame.setSize(300, 100);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
 }
 }
```

Output


Panel With Compon...
First Name
Button in Panel 2

Panels (cont'd)

- **Example 2**

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

public class TestPanels extends JFrame {
      public TestPanels() {
            // Create panel p1 for the buttons and set GridLayout

            JPanel p1 = new JPanel();

            p1.setLayout(new GridLayout(4, 3));
            //Add buttons to the panel
             for (int i = 1; i <= 9; i++) {
             p1.add(new JButton("" + i));
}
```

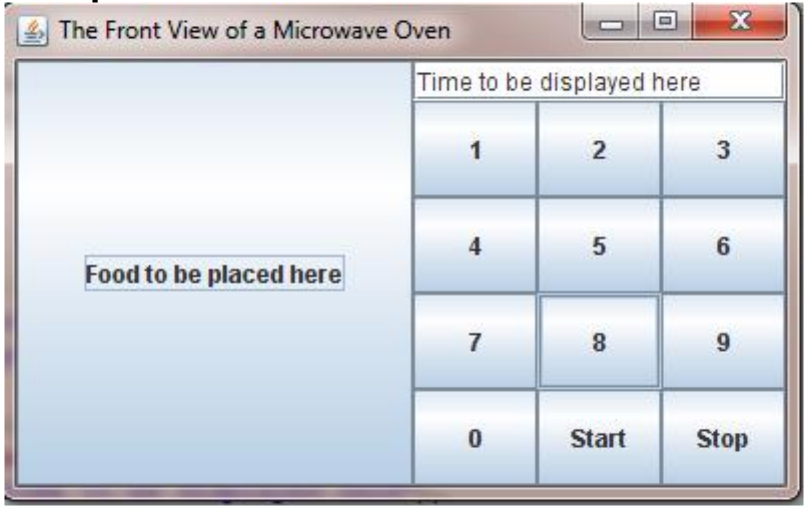
Panels (cont’d)

```
      p1.add(new JButton("" + 0));
      p1.add(new JButton("Start"));
      p1.add(new JButton("Stop"));

      // Create panel p2 to hold a text field and p1
     JPanel p2 = new JPanel(new BorderLayout());
     p2.add (new JTextField("Time to be displayed here"),
      BorderLayout.NORTH);
      p2.add(p1, BorderLayout.CENTER);

      // add contents into the frame
      add(p2, BorderLayout.EAST);
      add(new JButton("Food to be placed here"),
      BorderLayout.CENTER);
  }
```

Panels (cont’d)

```
        /** Main method */
        public static void main(String[] args) {
        TestPanels frame = new TestPanels();
        frame.setTitle("The Front View of a Microwave Oven");
        frame.setSize(400, 250);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true); vowels
        }
    }
```

Output



The Font Class

- You can create a font using the java.awt.Font class and set fonts for the components using the setFont method in the Component class.
- The constructor for Font is:
  - Font(String name, int style, int size);
- You can choose a font name from SansSerif, Serif, Monospaced, Dialog, or DialogInput,
- Choose a style from Font.PLAIN (0), Font.BOLD (1), Font.ITALIC (2), and Font.BOLD Font.ITALIC (3), and specify a font size of any positive integer.

Font (cont’d)

- Example:

```
Font font1 = new Font("SansSerif", Font.BOLD, 16);
Font font2 = new Font("Serif", Font.BOLD + Font.ITALIC, 12);
JButton jbtOK = new JButton("OK");
jbtOK.setFont(font1);
```

The Color Class

- You can set colors for GUI components by using the java.awt.Color class.
- Colors are made of red, green, and blue components, each represented by an int value that describes its intensity, ranging from 0 (darkest shade) to 255 (lightest shade).
- You can create a color using the following constructor:
  - public Color(int r, int g, int b);
- Example:
  - Color color = new Color(128, 100, 100);
- You can use the setBackground(Color c) and setForeground(Color c) methods to set a component’s background and foreground colors.

The Color Class (cont'd)

- Example
  - JButton jbtOK = new JButton("OK");
  - jbtOK.setBackground(color);
  - jbtOK.setForeground(new Color(100, 1, 1));
- Alternatively, you can use one of the 13 standard colors (BLACK, BLUE, CYAN, DARK_GRAY, GRAY, GREEN, LIGHT_GRAY, MAGENTA, ORANGE, PINK, RED, WHITE, and YELLOW) defined as constants in java.awt.Color.
- The following code, for instance, sets the foreground color of a button to red:
  - jbtOK.setForeground(Color.RED);

The Color Class (cont'd)

- Example

```
import java.awt.GridLayout;
import java.awt.Color;
import javax.swing.*;
public class ColorExample extends JFrame{
public ColorExample(){

        JFrame jf = new JFrame("Color Frame");
        setLayout(new GridLayout(1,2));
        Color color = new Color(128, 100, 100);
        JButton bc1 = new JButton("Left Button");
        JButton bc2 = new JButton("Right Button");
        bc1.setBackground(color);
        bc2.setForeground(new Color(250,0, 0));
        bc2.setForeground(Color.BLUE);
        add(bc1);
        add(bc2);
}
```

Paces

The Color Class (cont’d)

```
    public static void main(String[] args){
        ColorExample ce = new ColorExample();
        ce.setSize(300,150);
        ce.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ce.setLocationRelativeTo(null);
        ce.setVisible(true);
    }
}
```

Output

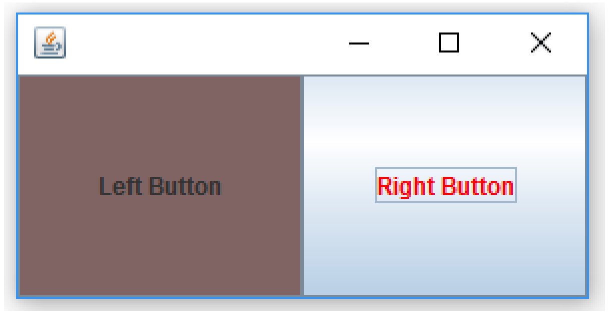


Image Icons

```
import javax.swing.*;
import java.awt.*;
 public class ImageExample extends JFrame {
       public ImageExample() {
       ImageIcon homeIcon = new ImageIcon("src/home.gif");
       ImageIcon birdIcon = new ImageIcon("src/bird.gif");
       ImageIcon mailIcon = new ImageIcon("src/mail.gif");
       setLayout(new GridLayout(1, 3, 5, 5));
       add(new JLabel(homeIcon));
       add(new JButton(birdIcon));
       add(new JLabel(mailIcon));

 }
```

Image Icons (cont’d)

```
/** Main method */
public static void main(String[] args) {
    ImageExample frame = new ImageExample();
    frame.setTitle("TestImageIcon");
    frame.setSize(500, 200);
    frame.setLocationRelativeTo(null);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
 }
}
```

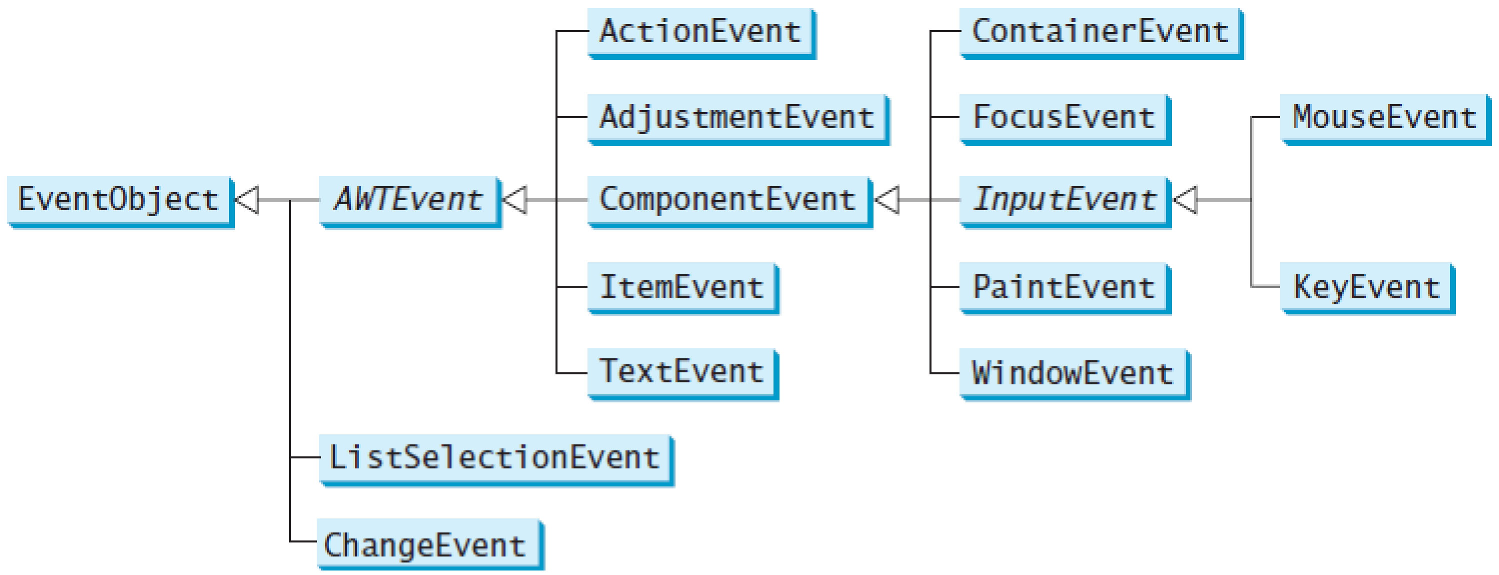
Event Handling

- Event and Event Source
  - When you run a Java GUI program, the program interacts with the user, and the events drive its execution.
  - An event can be defined as a signal to the program that something has happened.
  - Events are triggered either by external user actions, such as mouse movements, button clicks, and keystrokes, or by internal program activities, such as a timer.
  - The program can choose to respond to or ignore an event.
  - The component that creates an event and fires it is called the *source object or source component.*
  - For example, a button is the source object for a button-clicking action event.
  - An event is an instance of an event class.

Event Handling

- Java has a number of events and event handlers.
- Java.awt.event –is the main package
- It is up to the programmer to decide what to be executed, how to handle the generated event, etc
- There are more than 16 event types in java.
- **Eventhandler** and **ActionListener** are the two important aspects in event manipulation
- Adding Listener helps to organize the action(event) to be done after the button or the GUI control is selected
- ActionListner is the main Interface in event handling

Event Handling



An event is an object of the EventObject class.

Event Handling

- Listeners, Registrations, and Handling Events
  - Java uses a delegation-based model for event handling:
    - a source object fires an event, and an object interested in the event handles it. The latter object is called a *listener.*
  - For an object to be a listener for an event on a source object, two things are needed:
    - **The listener object must be an instance of the corresponding event-listener interface** to ensure that the listener has the correct method for processing the event.
    - **The listener object must be registered by the source object.** Registration methods depend on the event type. For **ActionEvent, the method is addActionListener.**

Event Handling

- Example 1

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

 public class HandleEvent extends JFrame{
    public HandleEvent() {
            // Create two buttons
            JButton jbtOK = new JButton("OK");
            JButton jbtCancel = new JButton("Cancel");

            // Create a panel to hold buttons
            JPanel panel = new JPanel();
            panel.add(jbtOK);
            panel.add(jbtCancel);

            add(panel); // Add panel to the frame
```

Event Handling

```
          // Register listeners
           OKListenerClass listener1 = new OKListenerClass();
           CancelListenerClass listener2 = new CancelListenerClass();
           jbtOK.addActionListener(listener1);
           jbtCancel.addActionListener(listener2);
        }

        public static void main(String[] args) {
           JFrame frame = new HandleEvent();
           frame.setTitle("Handle Event");
           frame.setSize(200, 150);
           frame.setLocation(200, 100);
           frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
           frame.setVisible(true);
        }
    }
```

Event Handling

```
class OKListenerClass implements ActionListener{
  public void actionPerformed(ActionEvent e) {
          System.out.println("OK button clicked");
  }
 }

 class CancelListenerClass implements ActionListener{
  public void actionPerformed(ActionEvent e) {
          System.out.println("Cancel button clicked");
   }
 }
```

Event Handling

Example 2

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

public class MyGUI extends Jframe
                        implements ActionListener {
   private JButton clickme = new JButton("ClickMe");
   private JButton tests = new JButton("Test");
   public MyGUI()  {
      // Add clickme to the GUI and assign it a listener
      setLayout(new GridLayout(2, 0, 5, 5));
      add(clickme);
          add(tests);
          clickme.addActionListener(this);
          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          setSize(200,200);
          setVisible(true);
   }
```

Event Handling

```
public void actionPerformed(ActionEvent e) {
        if (e.getSource() == clickme) {
             tests.setText("Clickme button clicked");
        }
    } // actionPerformed()

    public static void main(String args[]) {
        MyGUI gui = new MyGUI();
    }
}  // MyGUI  class
```

Event Handling

Example 3

```
import javax.swing.*;                              // Packages used
import java.awt.*;
import java.awt.event.*;
public class Converter extends JFrame implements ActionListener{
  private JLabel prompt = new JLabel("Distance in miles: ");
  private JTextField input = new JTextField(6);
  private JTextArea display = new JTextArea(10,20);
  private JButton convert = new JButton("Convert!");
  public Converter() {
      setLayout(new FlowLayout());
     add(prompt);
      add(input);
      add(convert);
      add(display);
      display.setLineWrap(true);
      display.setEditable(false);
      convert.addActionListener(this);
  } // Converter()
```

Event Handling

```
public void actionPerformed( ActionEvent e ) {

          //CHECK TO ACCEPT ONLY NUMBERS
       double miles;
      try{
       miles =  Double.valueOf(input.getText()).doubleValue();
      }
      catch ( NumberFormatException ex )
      {
      JOptionPane.showMessageDialog(null, "Please Enter Only Number" );
      return;
      } // end catch

      display.setText("");
          double km = miles/0.62;
          display.append(miles + " miles equals " + km + " kilometers\n");

  } // actionPerformed()
```

Event Handling

```
public static void main(String args[]) {
    Converter f = new Converter();
    f.setSize(400, 300);
    f.setVisible(true);
    f.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        System.exit(0);                          // Quit the application
      }
    });
  } // main()
} // Converter class
```

More Examples

- Checkbox- CheckBoxTest.java
- Text Area – TextArea.java
- Combobox – ComboBoxDemo.java – uses DescriptionPanel.java class
- Lists – ListTest.java
- Tree – SimpleTree.java
- Menu – SimpleTextEditor.java
- **Progress Indicators – ProgressBarDemo.java**

Overview of Java Applets

- Applet is a browser dependent program of java
- In applet, methods like init(),paint() etc are included
- There is no main() method in applet
- Package:- import java.applet.*;
- Java application uses JFrame but java applet uses JApplet
- Java application saved by filename.java and run by **javac** filename
- But, java applets needs another **html file** and save by filename.html and run by **appletviewer** filename.html

Cont..

- **Basic drawing Tools**:
- Some of the known functions for drawing are:
- drawLine()-x,y coordinates and length and height
- drawRect();
- fillRect();
- drawPolygon()
- drawOval();
- drawstring();-gin-bcoordinates, end-coordinates
-

Cont..

- **for irregular shapes-we use drawPolygon()**
- eg:
- int x[]={10,20,30,40,50,60};
- int y[]={60,50,40,30,20,10};
- int p=6;
- drawPolygon(x,y,p);//hexagon

Examples

- **Eg1:** public class eg1 extends Japplet{
- Public void init()
- {setBackground(Color.red);
- }
- Public void paint(Graphics g)
- { g.setColor(Color.blue);
- g.fillRect(0,0,100,100);
- }
- }

Examples

- import java.awt.*;
- import java.applet.*;
- public class Applet1 extends JApplet {
- public void paint(Graphics g)
- {
- g.fillRect(30, 20, 200, 200);
- g.clearRect(50, 30,70,50);
- g.drawRect(60,50,40,20);
- g.drawLine(10,50,250,55);
- }}

Examples(applet)

- **Eg html file for a java class name Eg;**
- **<html>**
- **<head>the result</head>**
- **<body>**
- **<applet code=”Eg.class” width=250   height=250>**
- **</applet></body></html>**

Tip GUI

- Why java need yet another GUI toolkit?
- Why we living AWT and Swing ?
- Did we really need another GUI ?