Object Oriented Programming

Graphical User Interface

Sadaf Anwar
sanwar@numl.edu.pk
Department of Software Engineering

GUI in Java

- There are two main sets of Java APIs for graphics programming:

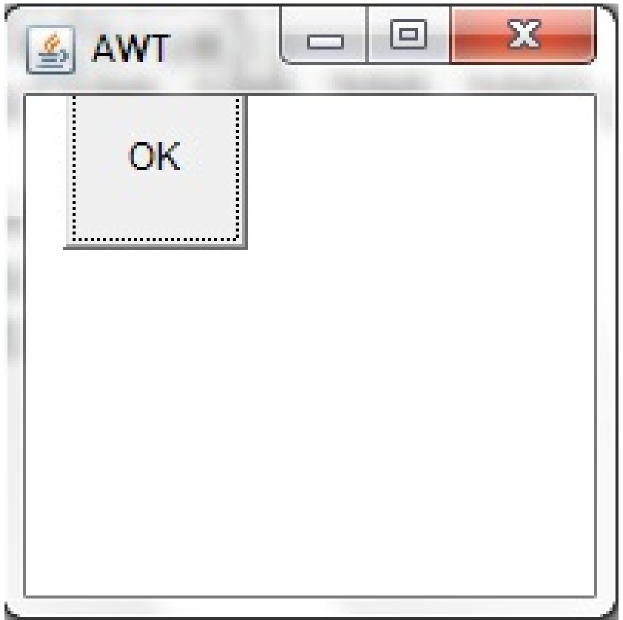
1. **AWT (<u>A</u>bstract <u>W</u>indowing <u>T</u>oolkit),**
2. **Swing**

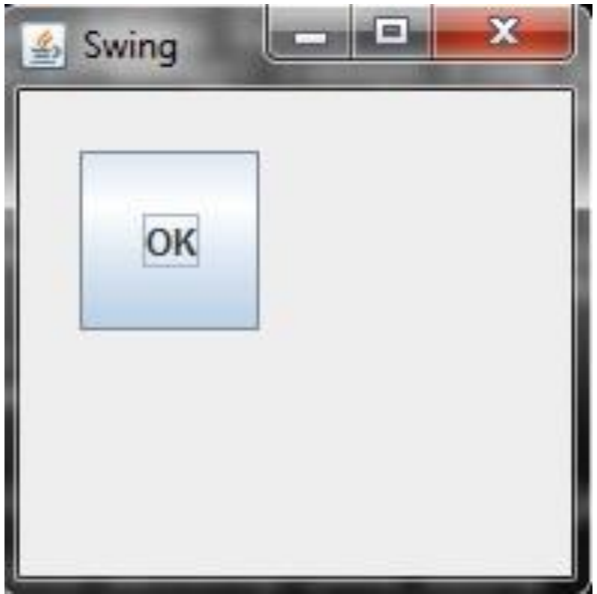
Package Comparison

AWT	• AWT API	Swing	• Swing API
AWT	• 12 Packages	Swing	• 18 Packages
AWT	• 370 Classes	Swing	• 737 Classes

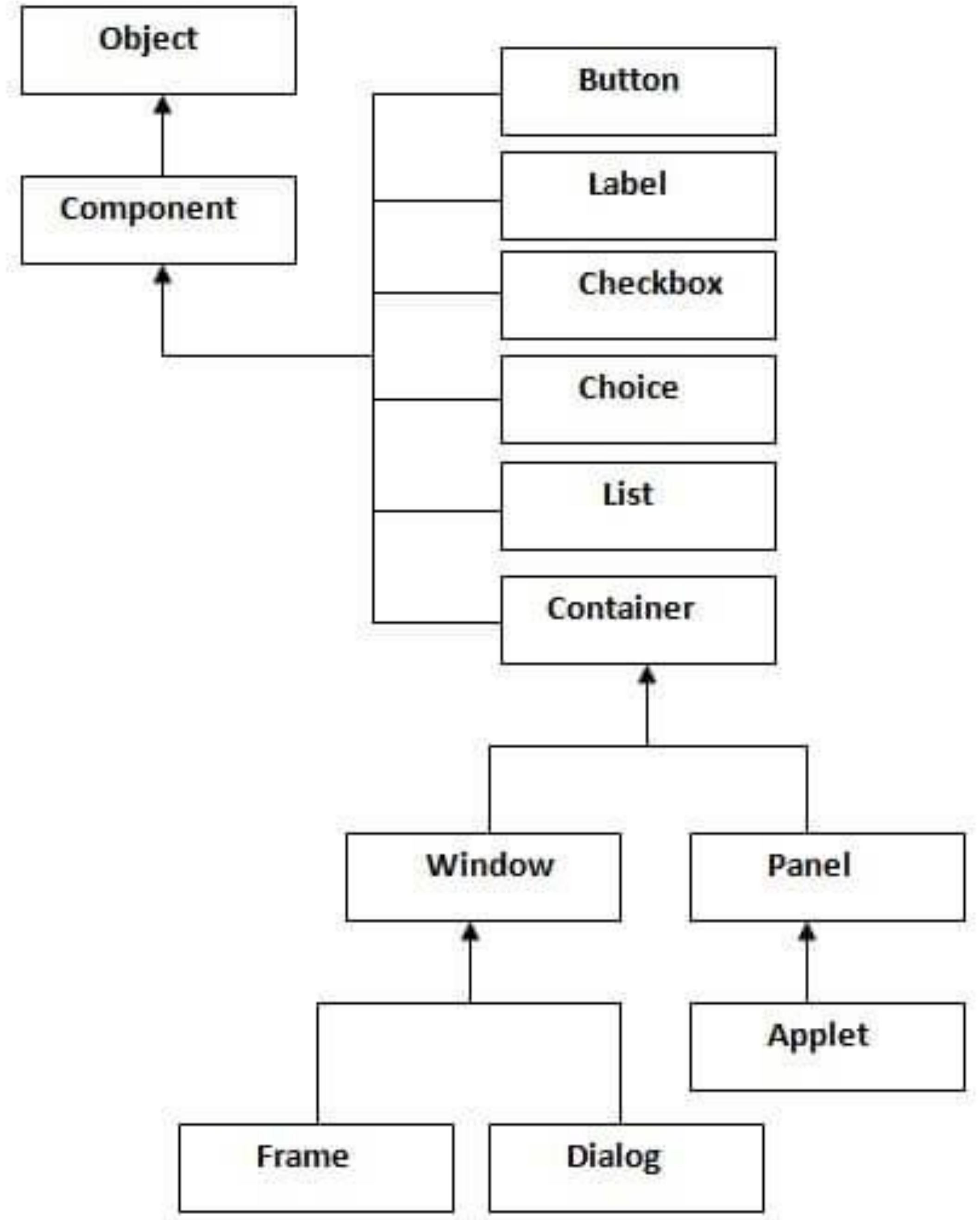
Difference b/w AWT and SWING

AWT	Swing
AWT stands for Abstract Window Toolkit.	Swing is a part of Java Foundation Class (JFC).
AWT components are platform dependent	Swing components are platform independent





AWT Hierarchy



The java.awt package supports

3. Layout managers:

   FlowLayout, BorderLayout and GridLayout.

4. Custom graphics classes:

   Graphics, Color and Font.

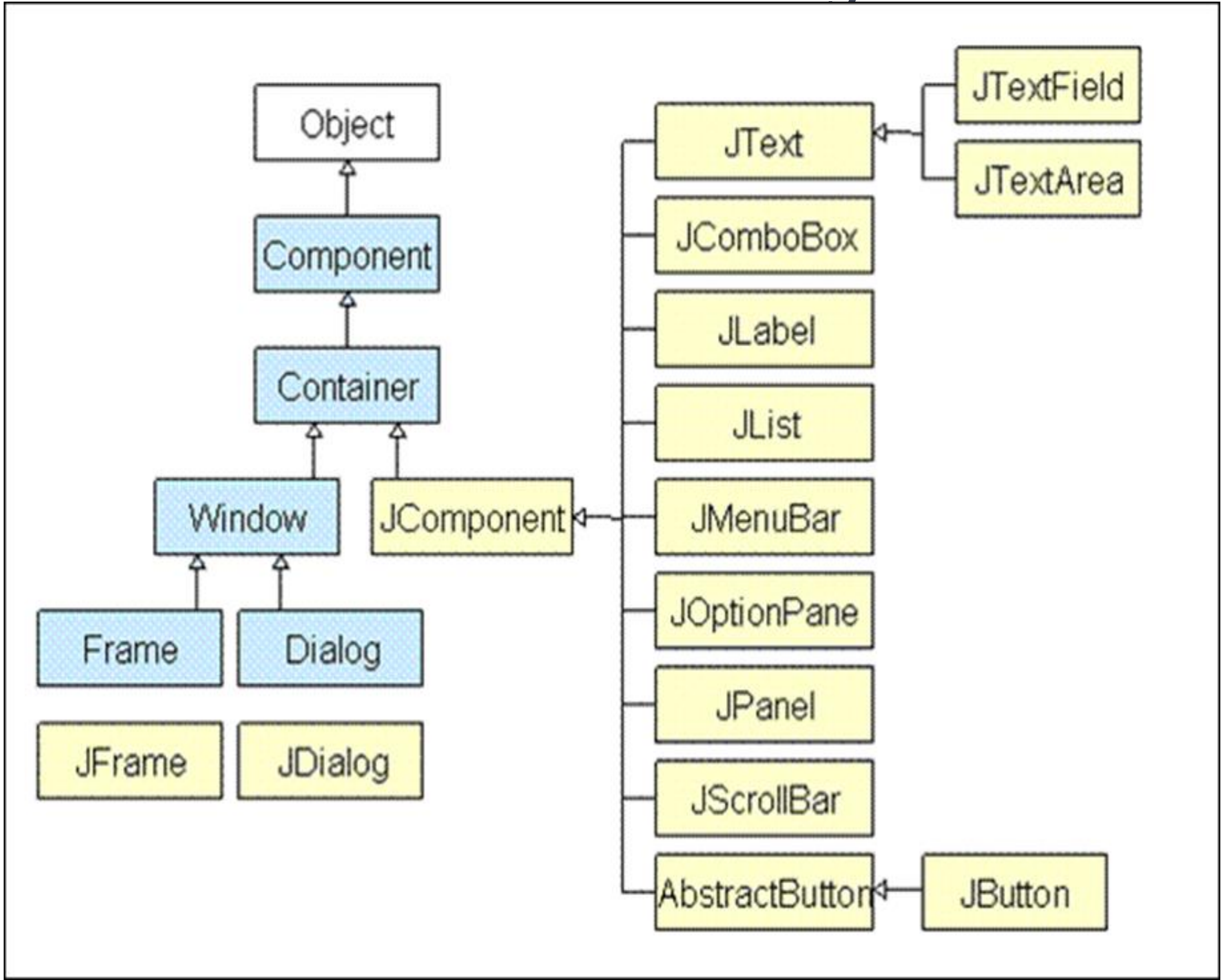
The java.awt package also supports

5. Event package supports event handling:
   a) Event classes:
   ActionEvent, MouseEvent, KeyEvent & Window Event
   b) Event Listener Interfaces:

ActionListener, MouseListener, MouseMotion Listener, KeyListener & WindowListener,

c) Event Listener Adapter classes:
MouseAdapter, KeyAdapter & WindowAdapter.

SWING Hierarchy



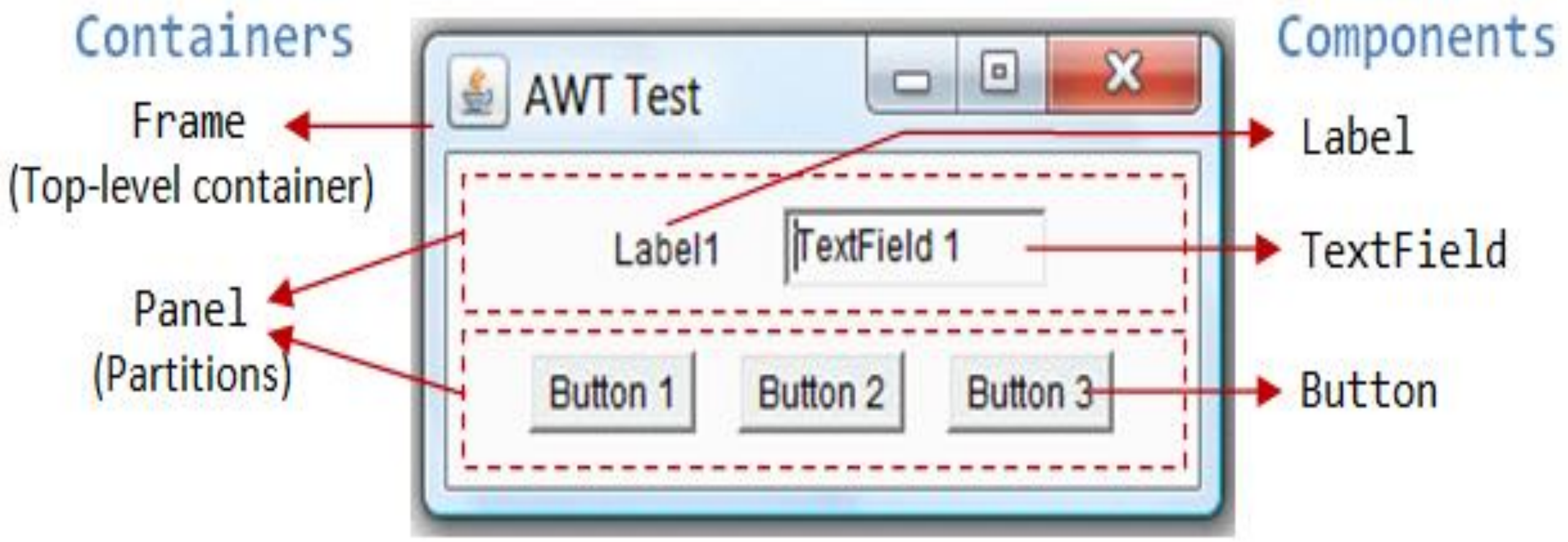
Applets and Applications

- An applet is a Java program that runs on a web page
  - Applets can be run from:
    - Internet Explorer etc.
    - Applet viewer
- An application is a Java program that runs all by itself

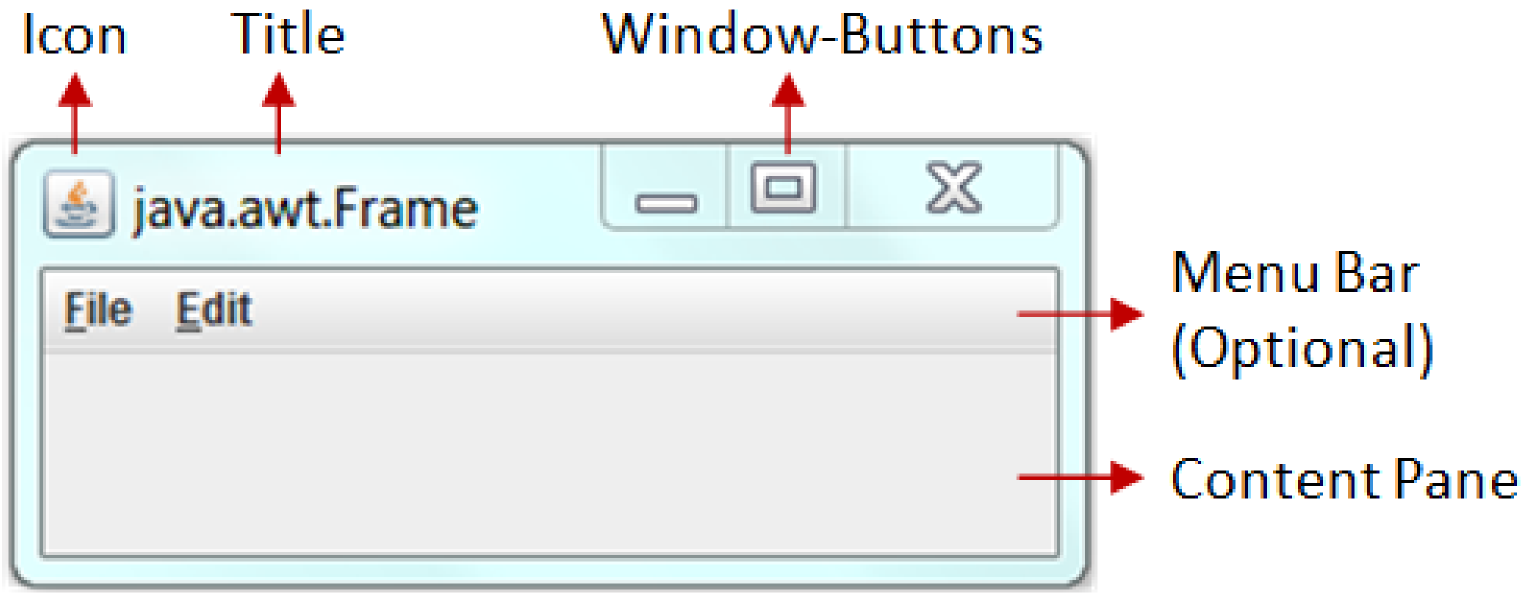
There are two types of main GUI Elements:

- ***Component:*** Components are elementary GUI entities, such as Button, Label, and TextField.
- ***Container:*** Containers, such as Frame and Panel, are used to *hold components in a specific layout* (such as FlowLayout or GridLayout).

A container can also hold sub-containers.

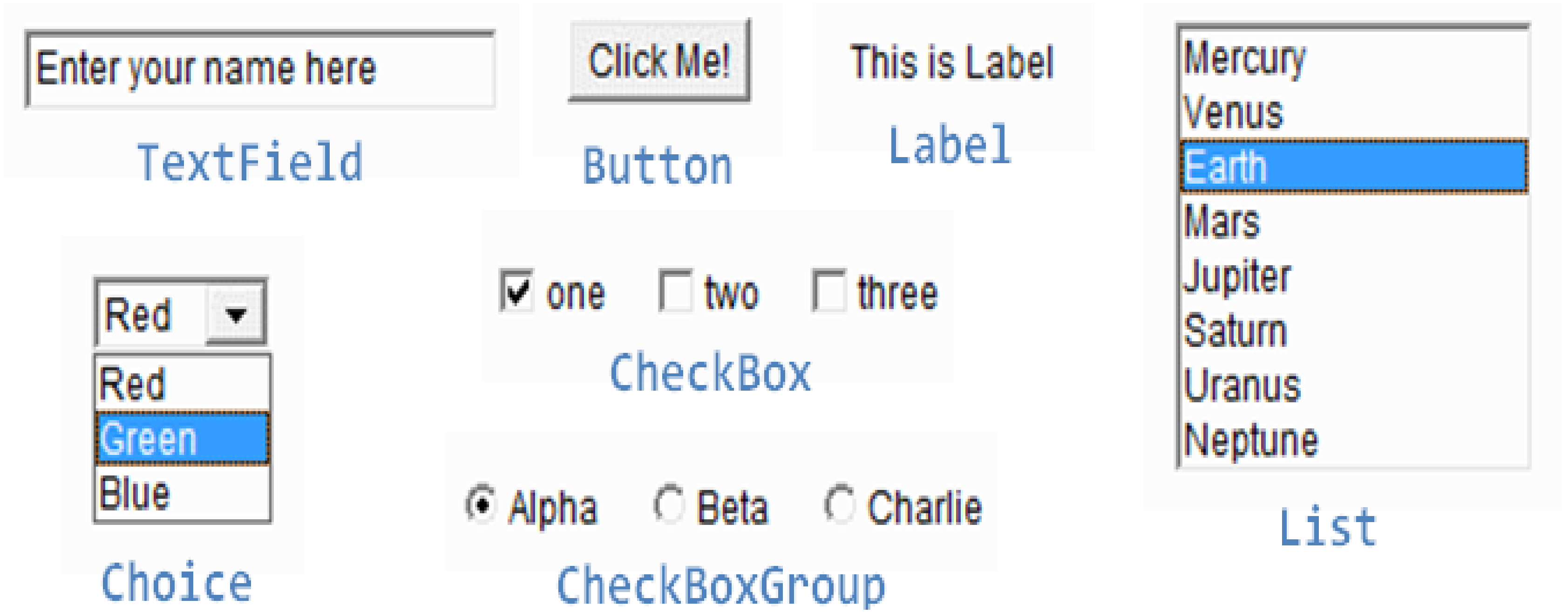

Containers
Frame
(Top-level container)
Panel
(Partitions)
AWT Test
Label1
TextField 1
Button 1
Button 2
Button 3
Components
Label
TextField
Button

- **AWT Container Classes**



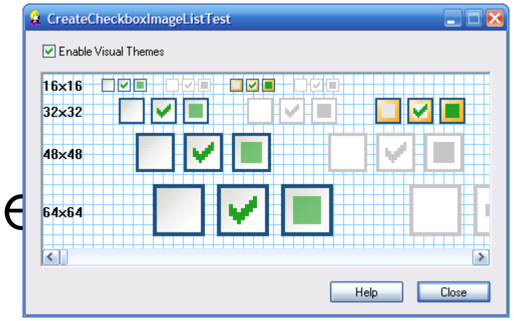
Important Objects

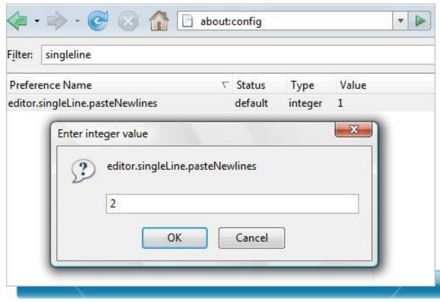
- Panel p = new Panel();
- Button b = new Button("Press");
- p.add(b);


Enter your name here
TextField
Click Me!
Button
This is Label
Label
Mercury
Venus
Earth
Mars
Jupiter
Saturn
Uranus
Neptune
List
Red
Red
Green
Blue
Choice
one
two
three
CheckBox
Alpha
Beta
Charlie
CheckBoxGroup

AWT Components

- AWT supplies the following UI components:
  - Buttons (java.awt.Button)
  - Checkboxes (java.awt.Checkbox)
  - Single-line text fields (java.awt.TextFie
  - Menus (java.awt.MenuItem)
  - Containers (java.awt.Panel)
  - Lists (java.awt.List)



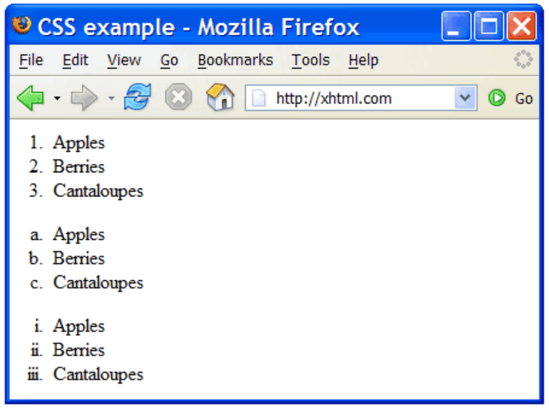


Filter: singleline

Preference Name	Status	Type	Value
editor.singleLine.pasteNewlines	default	integer	1

editor.singleLine.pasteNewlines

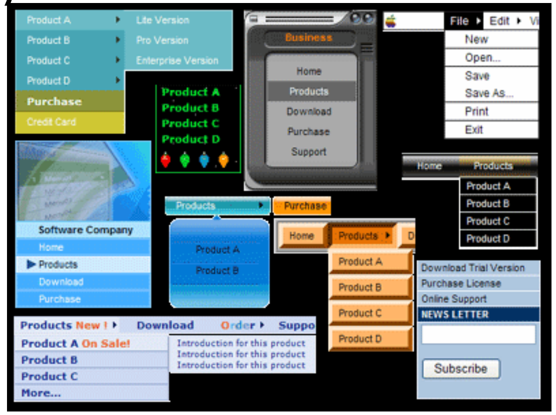
2

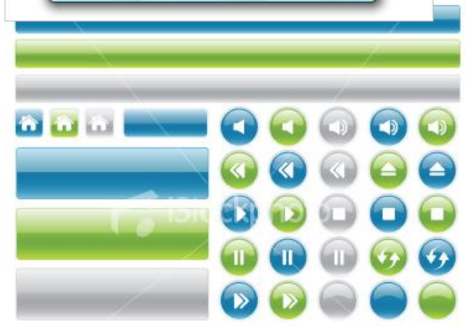

CSS example - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://xhtml.com
Go

1. Apples
2. Berries
3. Cantaloupes

a. Apples
b. Berries
c. Cantaloupes

i. Apples
ii. Berries
iii. Cantaloupes





Some AWT Component Methods

- `void setBackground(Color c)`
- `void setEnabled(boolean b)`
- `void setVisible(boolean b)`
- `void setFont(Font f)`
- `void setSize(Dimension d)`
- `void setLocation(int x, int y)`

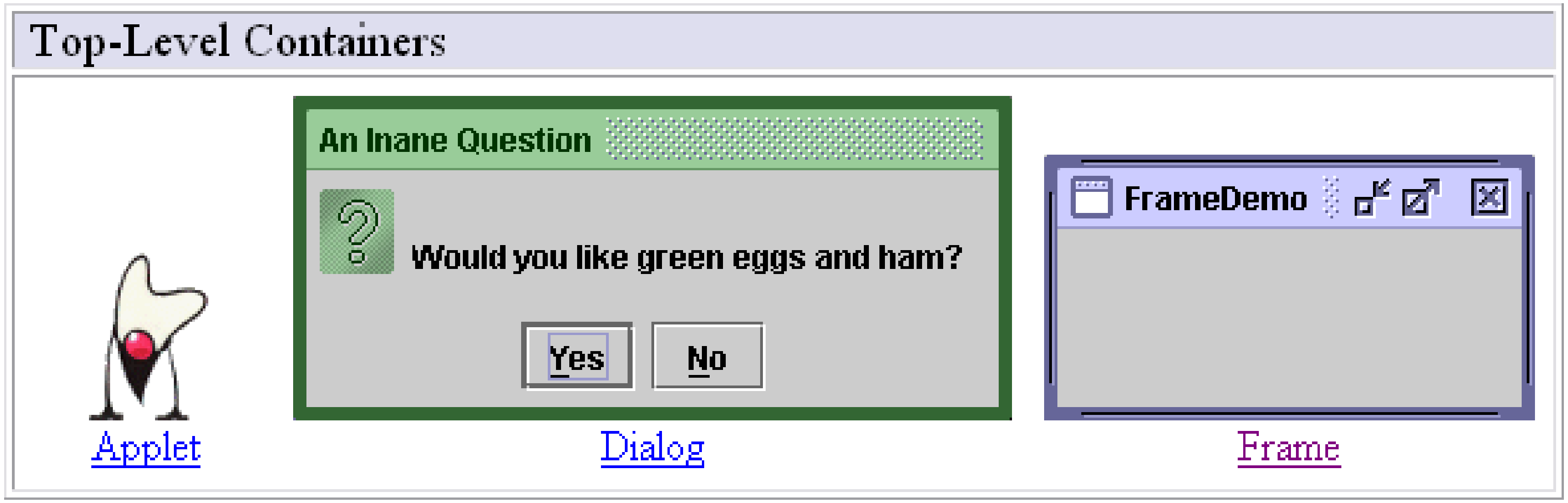
Introduction to Swing Classes

- **JPanel :** JPanel is Swing's version of AWT class Panel and uses the same default layout, FlowLayout. JPanel is descended directly from JComponent.
- **JFrame :** JFrame is Swing's version of Frame and is descended directly from **Frame** class. The component which is added to the **Frame**, is refered as its Content.
- **JWindow :** This is Swing's version of Window and has descended directly from **Window** class. Like **Window** it uses BorderLayout by default.

- **JLabel :** JLabel has descended from JComponent, and is used to create text labels.
- **JButton :** JButton class provides the functioning of push button. JButton allows an icon, string or both associated with a button.
- **JTextField :** JTextFields allow editing of a single line of text.

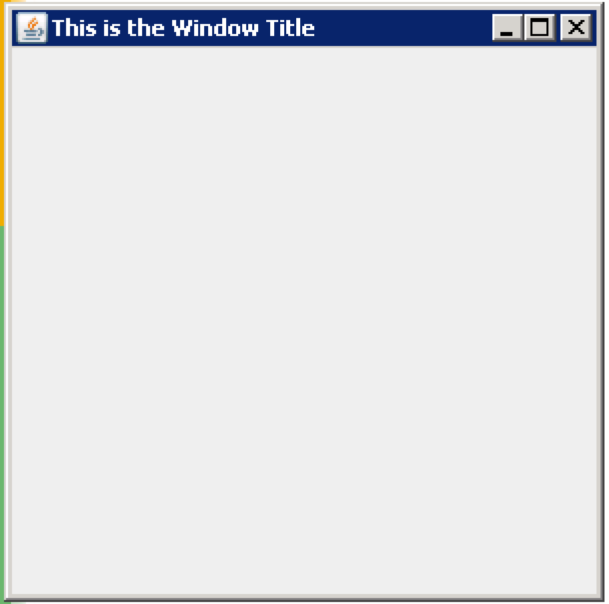
Swing Components

Top Level Containers

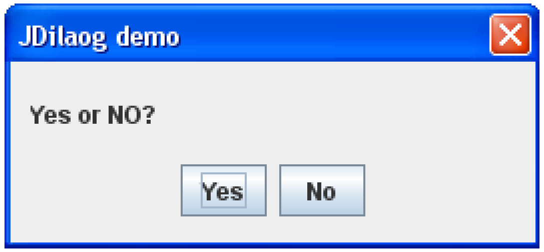


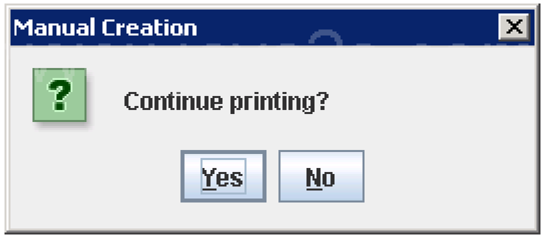
Your application usually extends one of these classes !

Jframe

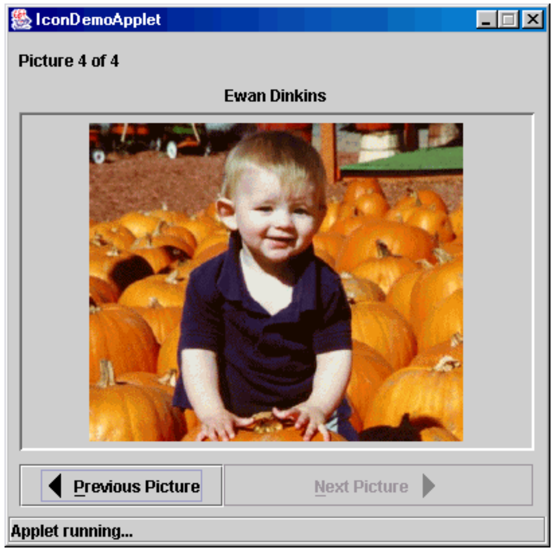


JDialog



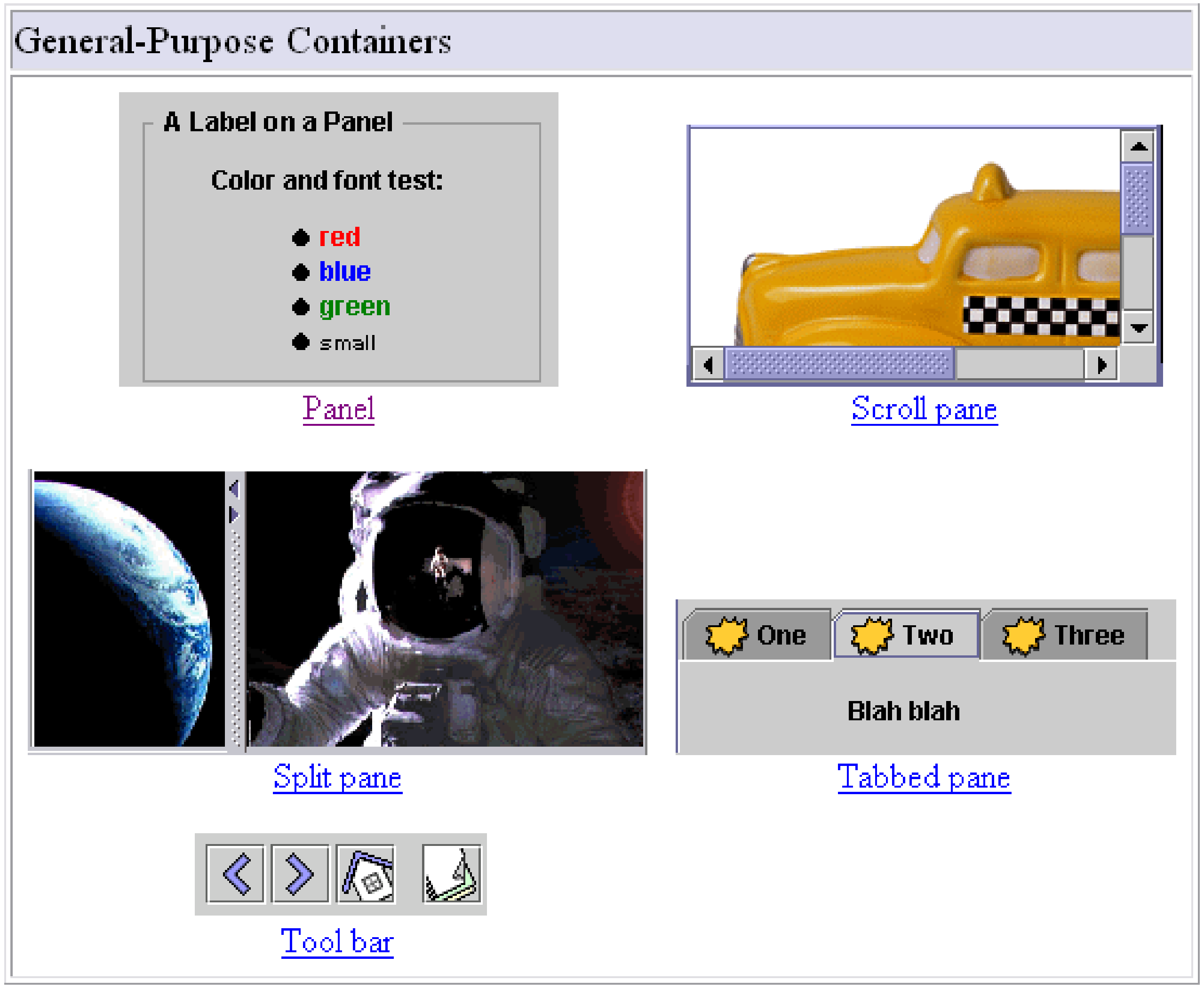


JApplet



Swing Components

- General Purpose Containers


General-Purpose Containers

A Label on a Panel

Color and font test:

- red
- blue
- green
- small

Panel

Scroll pane

Split pane

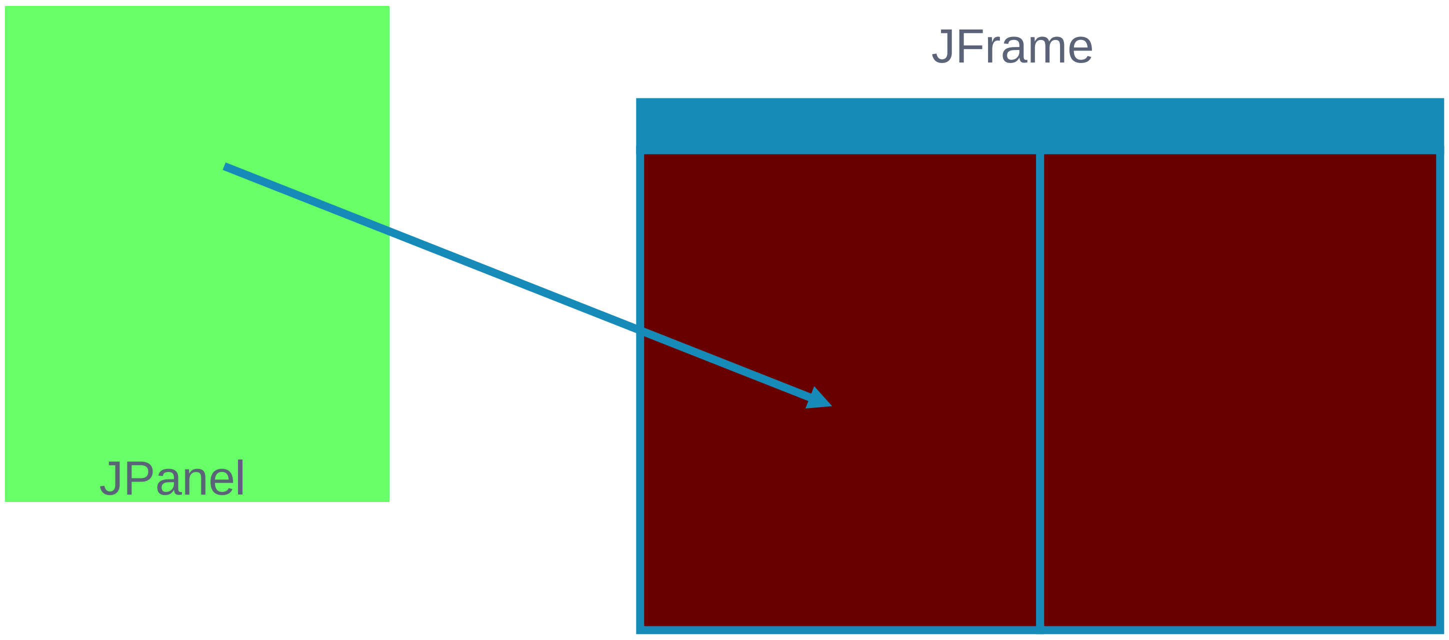
One Two Three

Blah blah

Tabbed pane

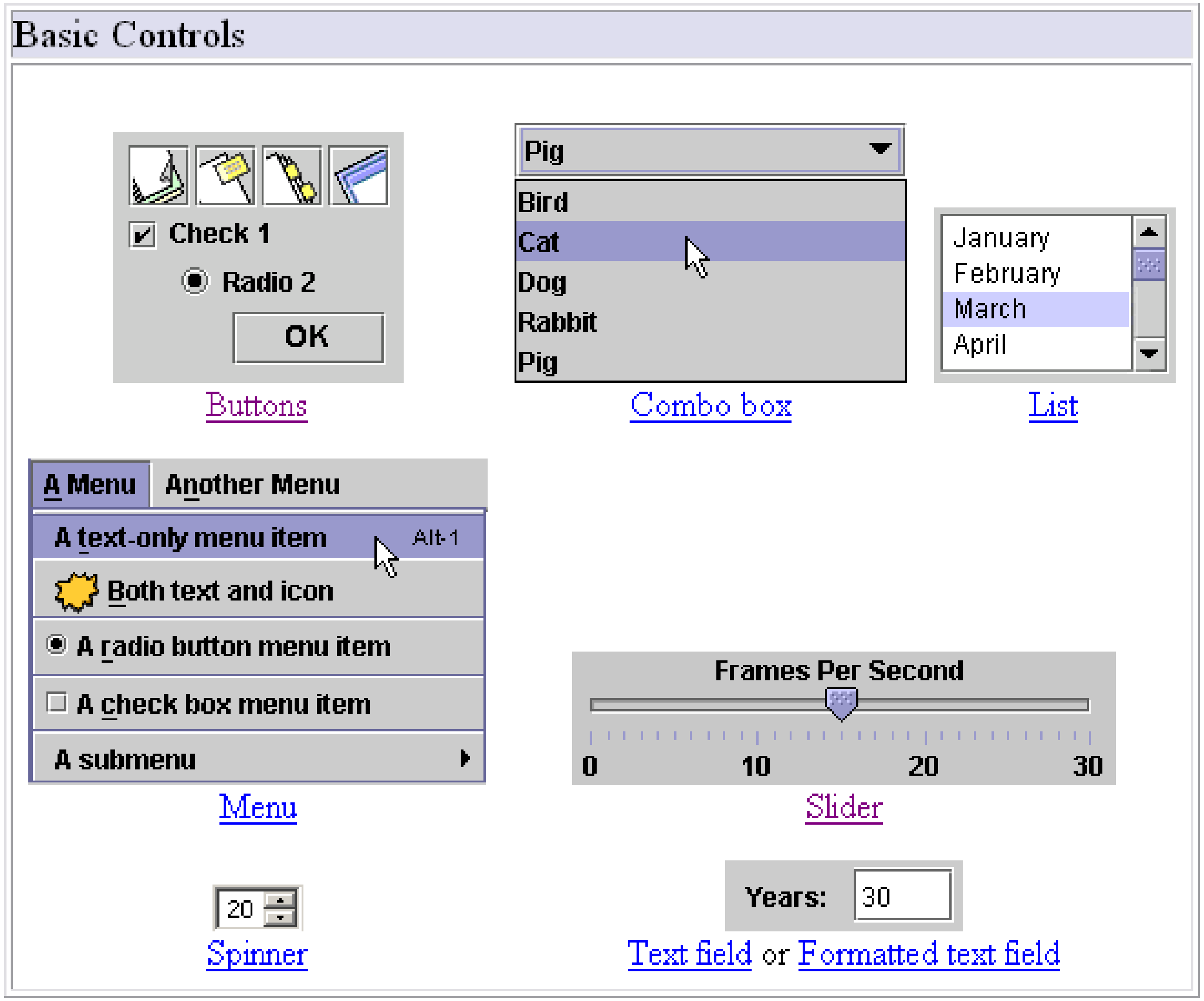
Tool bar

Swing Components



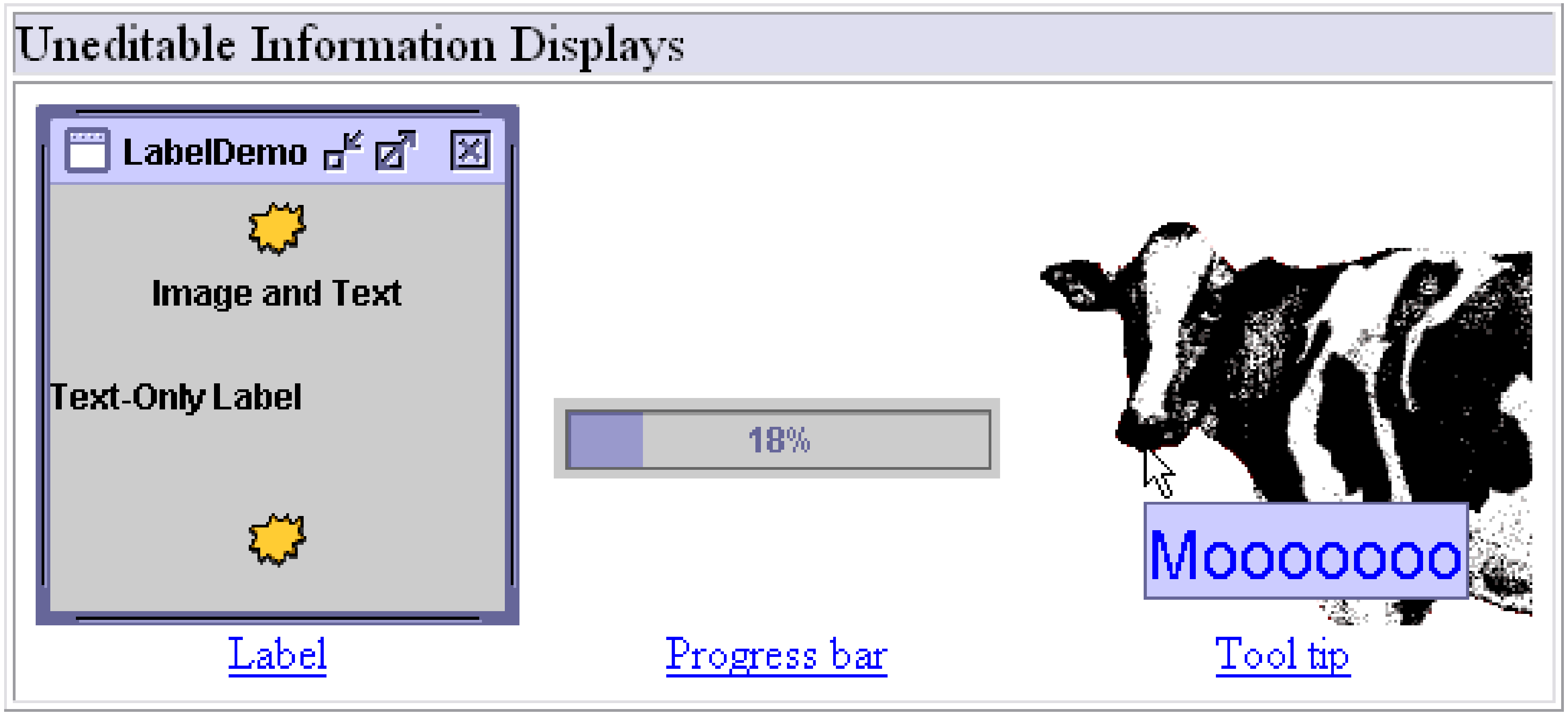
Swing Components

- Basic Controls



Swing Components

- Uneditable Information Displays



Example: A Simple Framed Window

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

public class SwingTest {

    public static void main(String[] args) {
        JFrame frame = new JFrame("Test Frame");
        frame.setSize(new Dimension(300,200));
        frame.setLocation(100,100);
        frame.setVisible(true);
    }
```

Why we use @Override annotation

It has two advantages:

1) If programmer makes any mistake such as wrong method name, wrong parameter types while overriding, you would get a compile time error. As by using this annotation you instruct compiler that you are overriding this method. If you don’t use the annotation then the sub class method would behave as a new method (not the overriding method) in sub class.

2) It improves the readability of the code. So if you change the signature of overridden method then all the sub classes that overrides the particular method would throw a compilation error, which would eventually help you to change the signature in the sub classes. If you have lots of classes in your application then this annotation would really help you to identify the classes that require changes when you change the signature of a method.

Questions