Object Oriented Programming Java: Introduction to GUI and Swing

Bikash Karmokar
Lecturer,
Department of CSE,
Bangladesh University

Overview

- GUI and SWING
  - Container, Components, Layouts

The First Swing Program

- Example:
  - The First Swing Program

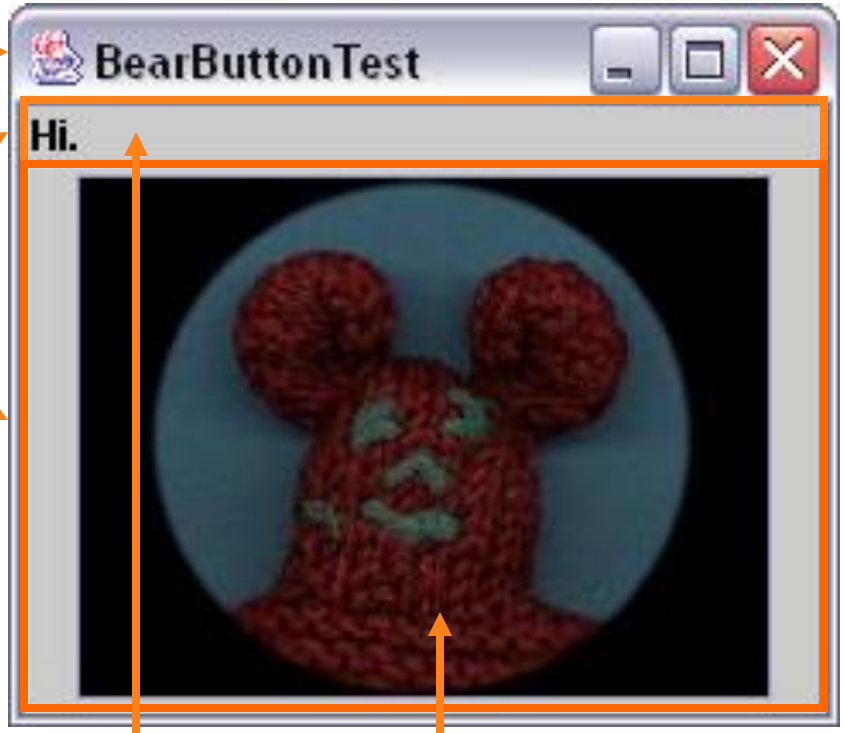
The GUI

Container: JFrame

Layout: BorderLayout

North

Center



Components: JLabel JButton, containing an ImageIcon

Steps to build a GUI

1. import package
2. set up top level container (e.g. JFrame)
3. apply layout

   (e.g. BorderLayout)
4. add components

   (e.g. Label, Button)
5. REGISTER listeners
6. show it to the world !

The Source

1. import package
2. set up top level container(e.g. JFrame)
3. apply layout (e.g. BorderLayout)
4. add components (e.g. Label, Button)
5. REGISTER listeners
6. show it to the world !

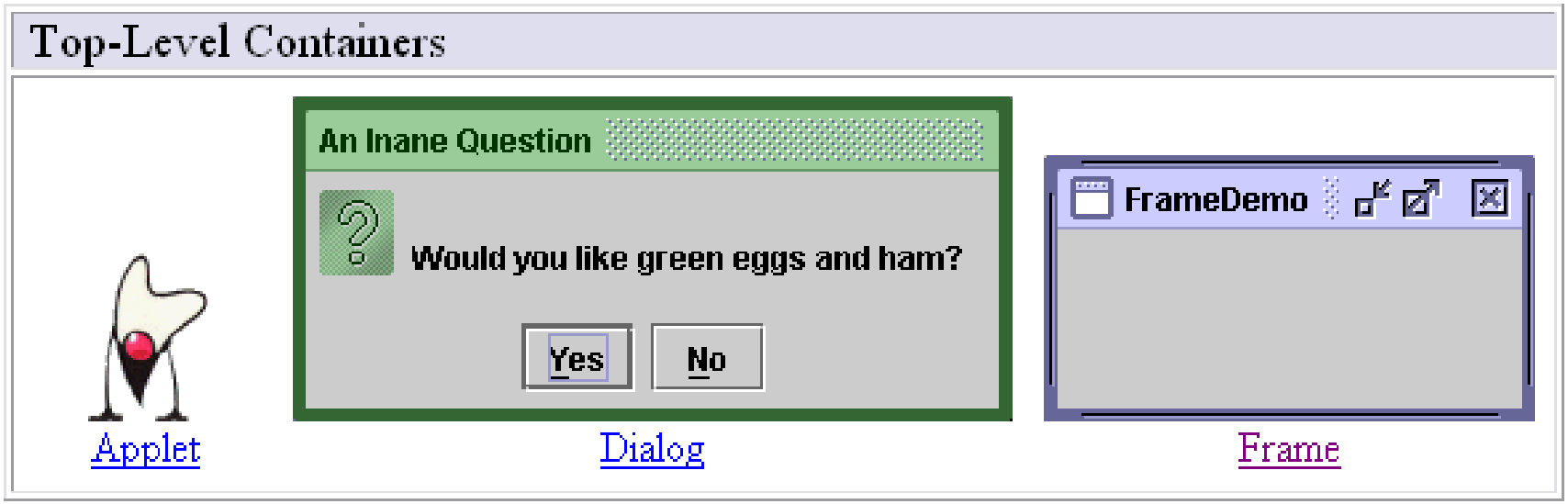
```
package guiusingswing;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
public class BearButtonTest extends JFrame {
        JLabel theLabel;
        //the constructor
        public BearButtonTest(){
                super("BearButtonTest"); // for the window name
                //set the action of x button click
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                // set the size of the frame
                setSize(300,300);
                //create graphical components
                //1. button
                JButton theButton = new JButton();
                theButton.setIcon(new ImageIcon("images/BearButton.jpg"));
                //theButton.setPressedIcon(new ImageIcon("images/friend.jpg"));

                //2. Label
                theLabel = new JLabel("Hi");
                //add components to container
                //specify the area (eg. "Center") if needed or possible

                getContentPane().add("North",theLabel);
                getContentPane().add("Center",theButton);

        }
        public static void main(String[] args) {
                BearButtonTest obj = new BearButtonTest();
                obj.setVisible(true);
        }
}
```

Swing Components

- Top Level Containers
- General Purpose Containers
- Special Purpose Containers
- Basic Controls
- Un editable Information Displays
- Interactive Displays of Highly Formatted Information

Swing Components

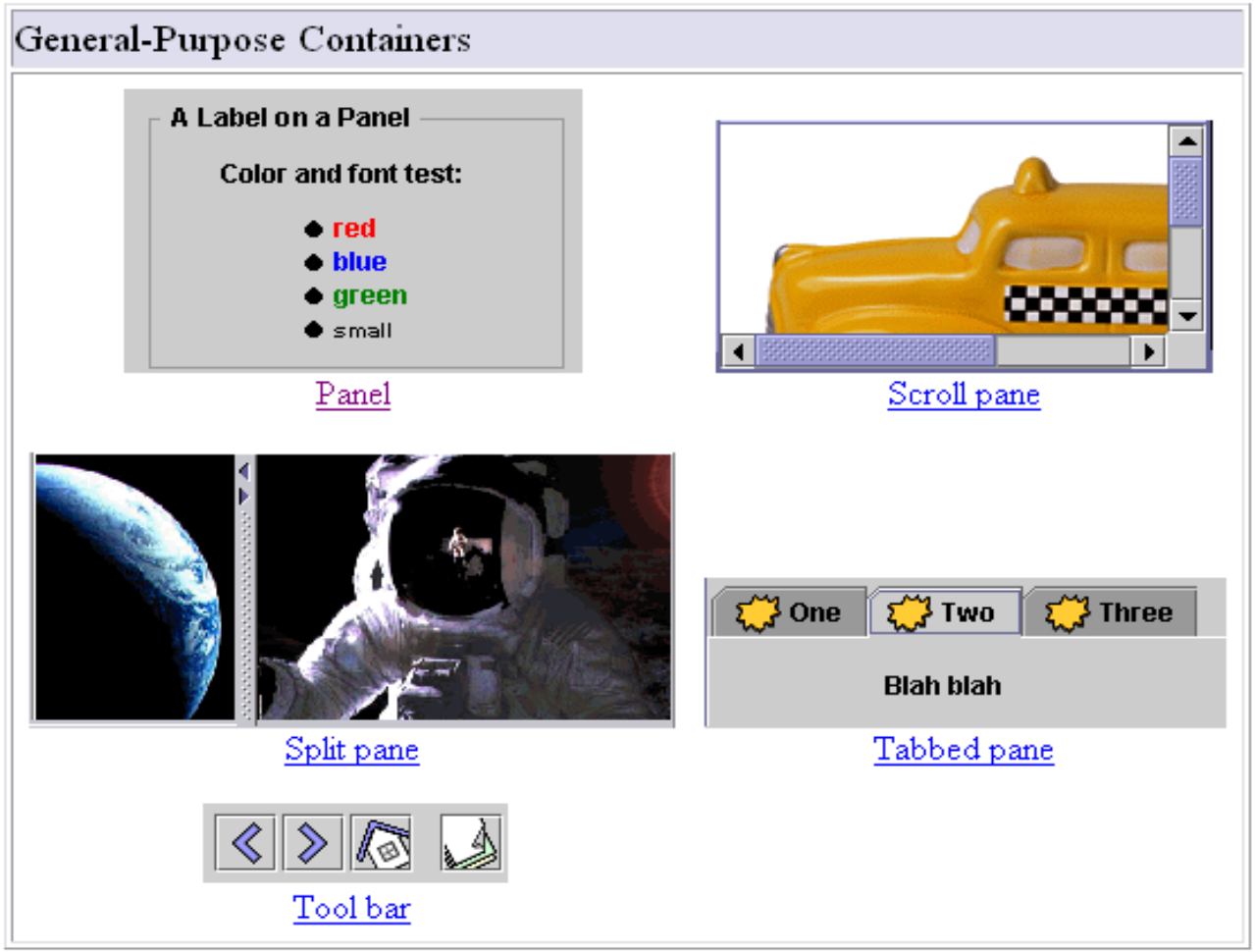
Top Level Containers



Your application usually extends one of these classes !

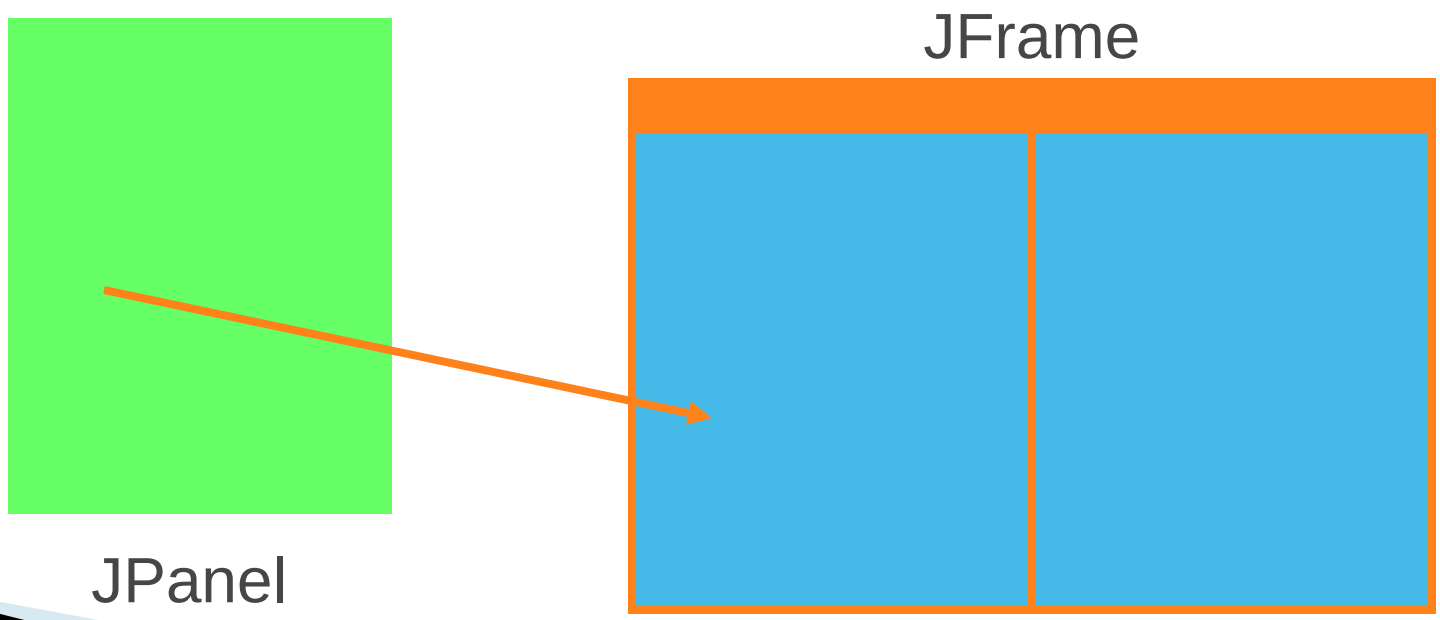
Swing Components

- General Purpose Containers



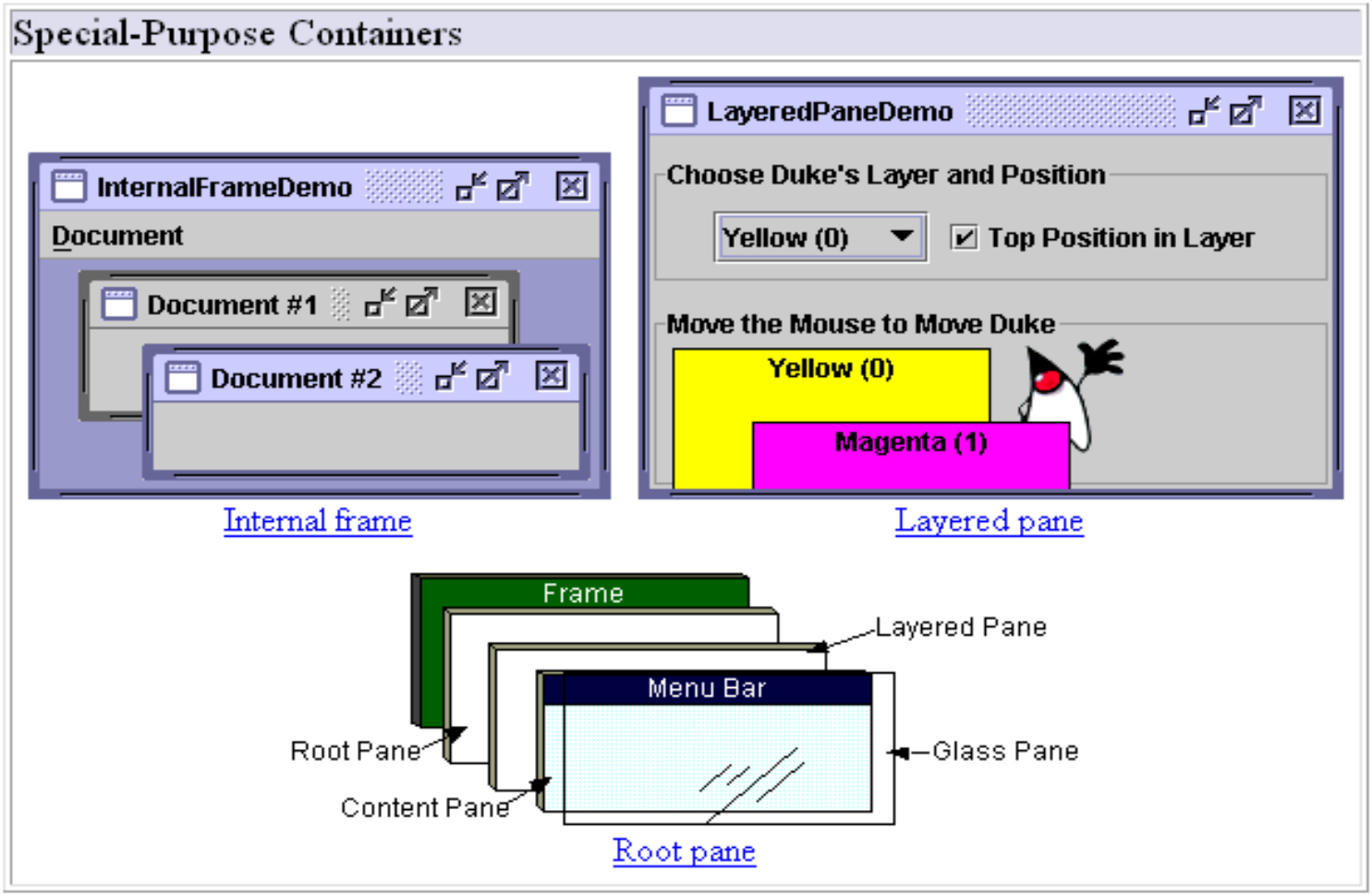
Swing Components

- General Purpose Containers
- typically used to collect Basic Controls (JButton, JChoiceBox…)
- Added to layout of top-level containers



Swing Components

- Special Purpose Containers



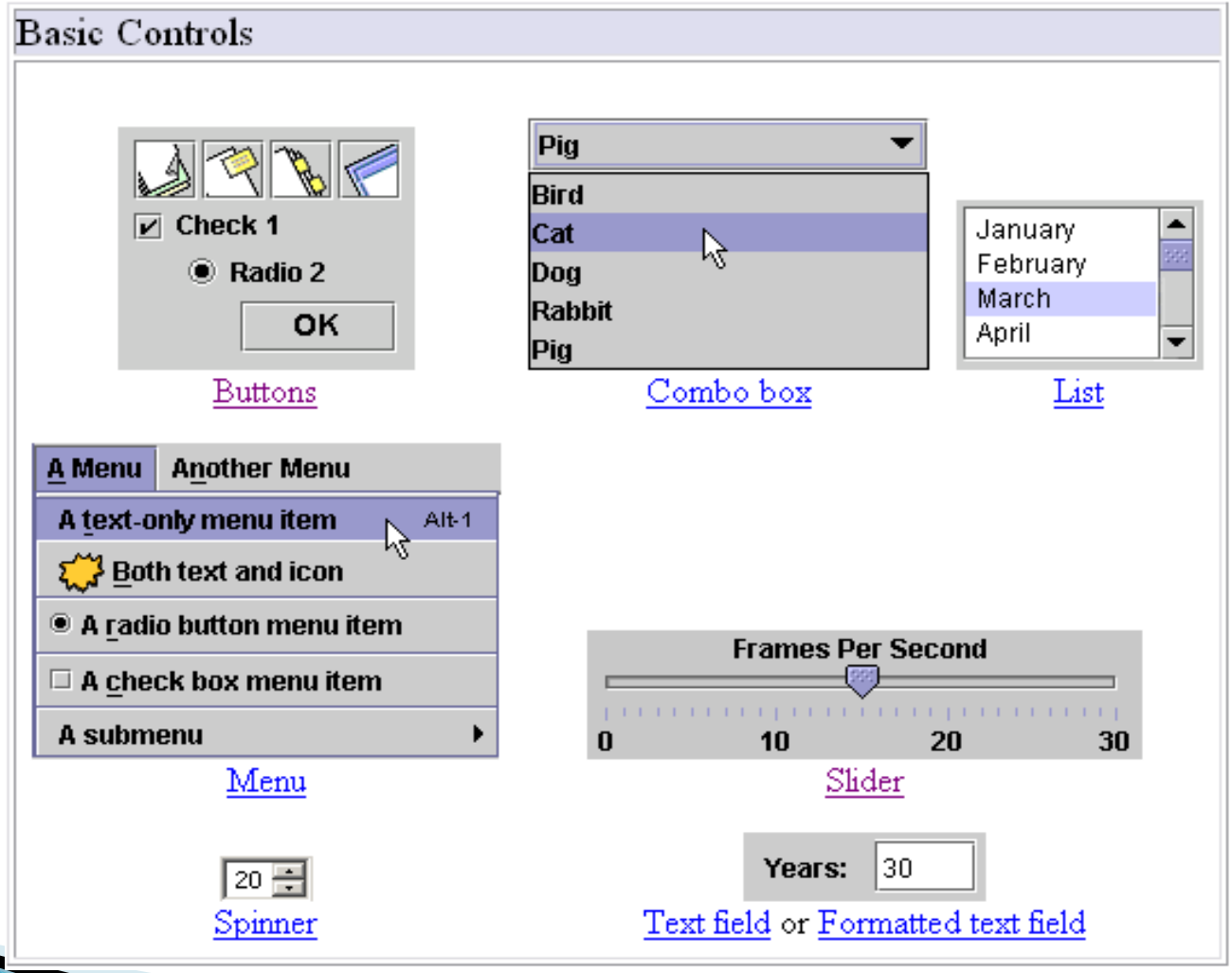
Swing Components

- Special Purpose Containers

- If you want to use them, go to java.sun.com

Swing Components

- Basic Controls

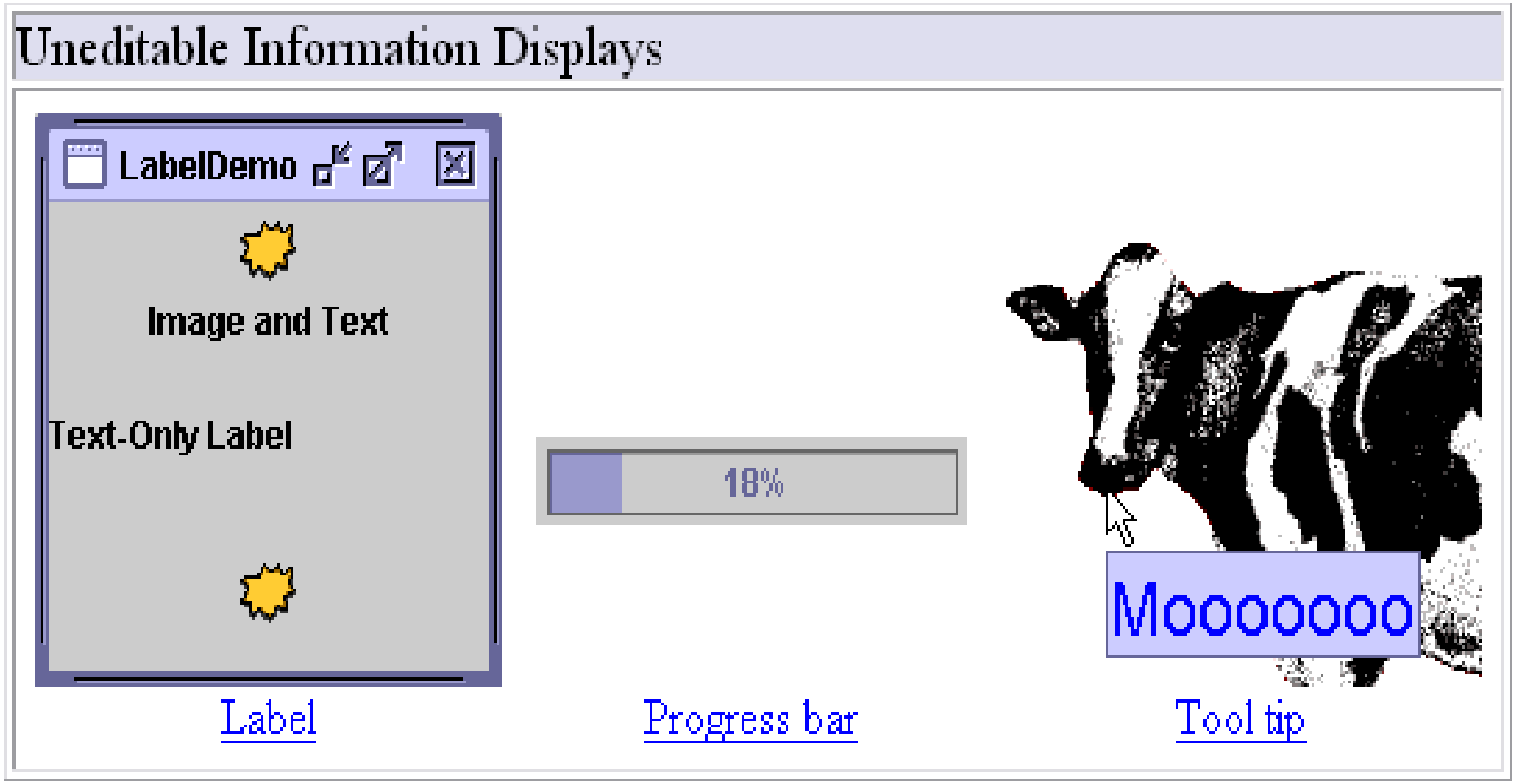


Swing Components

- Basic Controls
- Unlike ‘passive’ containers, controls are the ‘active’ part of your GUI
- Remark: containers aren’t only ‘passive’, they are also ‘active’ sources of events, eg. Mouse-events.
- Being the visible part of your interface, controls bring your application to life
- Controls are event sources !
- Objects of your application register to controls to handle the events

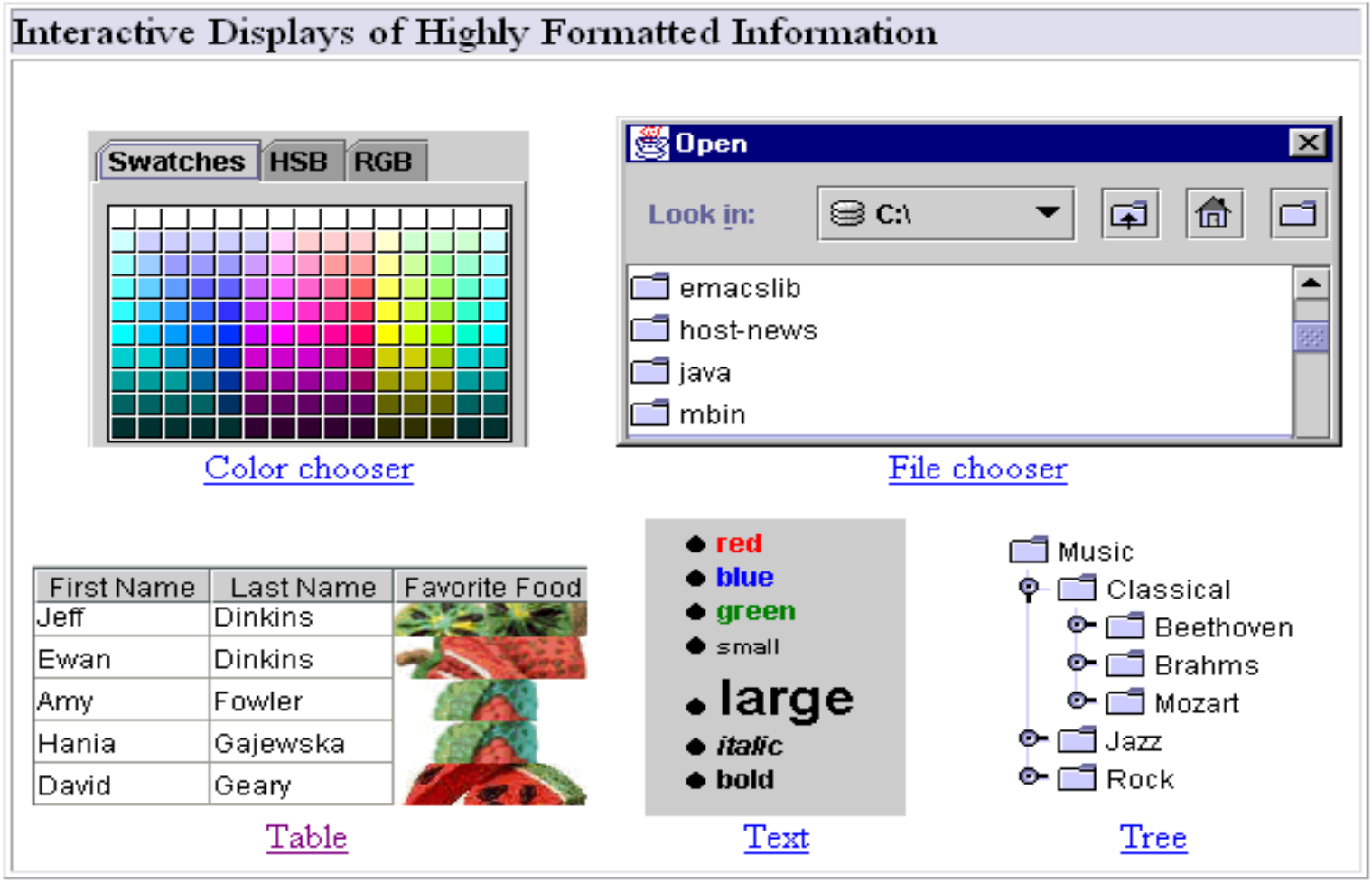
Swing Components

- Un editable Information Displays



Swing Components

- Interactive Displays of Highly Formatted Information



Swing Components

- Interactive Displays of Highly Formatted Information
- Define standard interfaces for frequently needed tasks

... go to java.sun.com for further information ...

Layout Management

Bikash Karmokar, Lecturer, CSE, Bangladesh University

Layout Management

- The process of determining the size and position of components
- A layout manager is an object that performs layout management for the components within the container.
- Layout managers have the final say on the size and position of components added to a container
- Using the add method to put a component in a container, you must ALWAYS take the container's layout manager into account

Layout Management

- Java supplies five commonly used layout managers:

1. BorderLayout
2. BoxLayout
3. FlowLayout
4. GridBagLayout
5. GridLayout

Layouts

- BorderLayout

Position must be specified, e.g. add (“North”, myComponent)

Layouts

- BoxLayout

The BoxLayout class puts components in a single row or column.

It respects the components‘ requested maximum sizes.

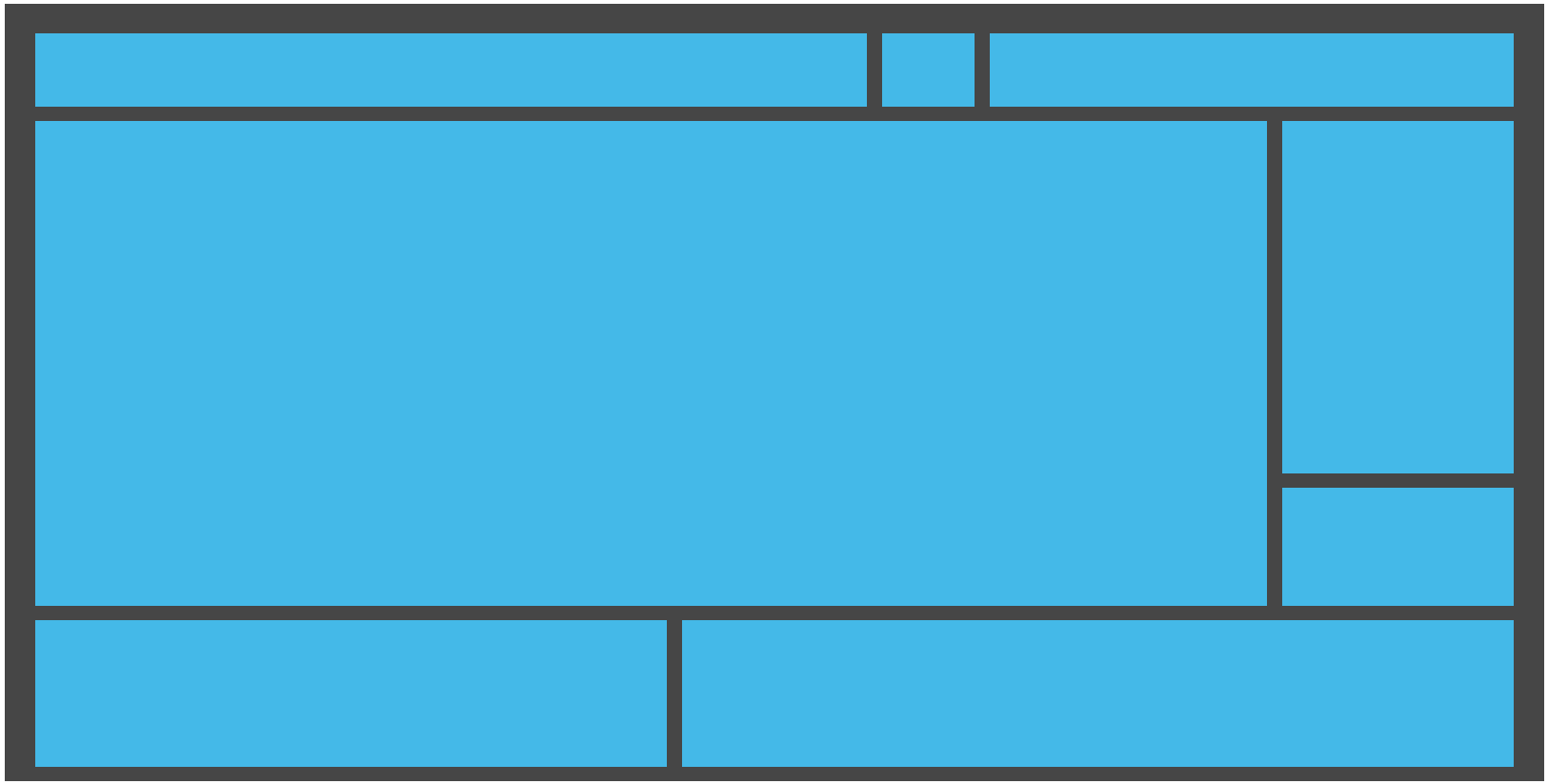
Layouts

- FlowLayout

FlowLayout is the default layout manager for every JPanel. It simply lays out components from left to right, starting new rows if necessary

Layouts

- GridBagLayout



GridBagLayout is the most sophisticated, flexible layout manager the Java platform provides. If you really want to use it, go to java.sun.com …

Layouts

- GridLayout

GridLayout simply makes a bunch of components equal in size and displays them in the requested number of rows and columns .

Using Components

- Examples:

  - Using a JButton
  - Using a JSlider
  - Using a JCheckBox

Using a JButton

- Some Constructors:

JButton()	Creates a button with no text or icon
JButton(Icon icon)	Creates a button with an icon
JButton(String text)	Creates a button with text
JButton(String text, Icon icon)	Creates a button with initial text and an icon

Using a JButton

- Some Methods:

addActionListener(ActionListener **a)**	Registers ActionListener to JButton Inherited from AbstractButton
setFont(Font font)	Specifies Font (Type, Style, Size) Inherited from JComponent
setBackground(Color color)	Sets background color Inherited from JComponent
setActionCommand (String text)	Used to specify button if listener is registered to multiple buttons (see ActionEvent.getActionCommand())

Using a JSlider

- Some Constructors:

JSlider()	Creates a horizontal slider with the range 0 to 100 and an initial value of 50
JSlider(int min, int max, int value)	Creates a horizontal slider using the specified min, max and value.
JSlider(Int orientation int min, int max, int value)	Creates a slider with the specified orientation and the specified minimum, maximum, and initial values.

Using a JSlider

- Some Methods:

addChangeListener(ChangeListener cl)	Registers ChangeListener to slider
int getValue()	Returns the slider's value
setValue(int value)	Sets the slider's value

Using a JCheckBox

- Some Constructors:

JCheckBox()	Creates an initially unselected check box button with no text, no icon.
JCheckBox(String text)	Creates an initially unselected check box with text.
JCheckBox(String text, Icon icon, boolean selected)	Creates a check box with text and icon, and specifies whether or not it is initially selected.

Using a JCheckBox

- Some Methods:

addItemListener(ItemListeneril)	Registers ItemListener to checkbox Inherited from AbstractButton
setSelected(boolean select)	Sets the state of checkbox Inherited from AbstractButton
boolean getSeleted()	Gets the state of checkbox. calling method often saves from registering to the checkbox !

At last...

This was a brief overview and introduction to SWING.

SWING has MUCH more to offer, see:

http://java.sun.com/docs/books/tutorial/uiswing/
http://java.sun.com/j2se/1.4.1/docs/api/

Reference:

Dr. Rolf Lakaemper,

Associate Professor

Department of Computer and Information Sciences

Temple University

Thank You