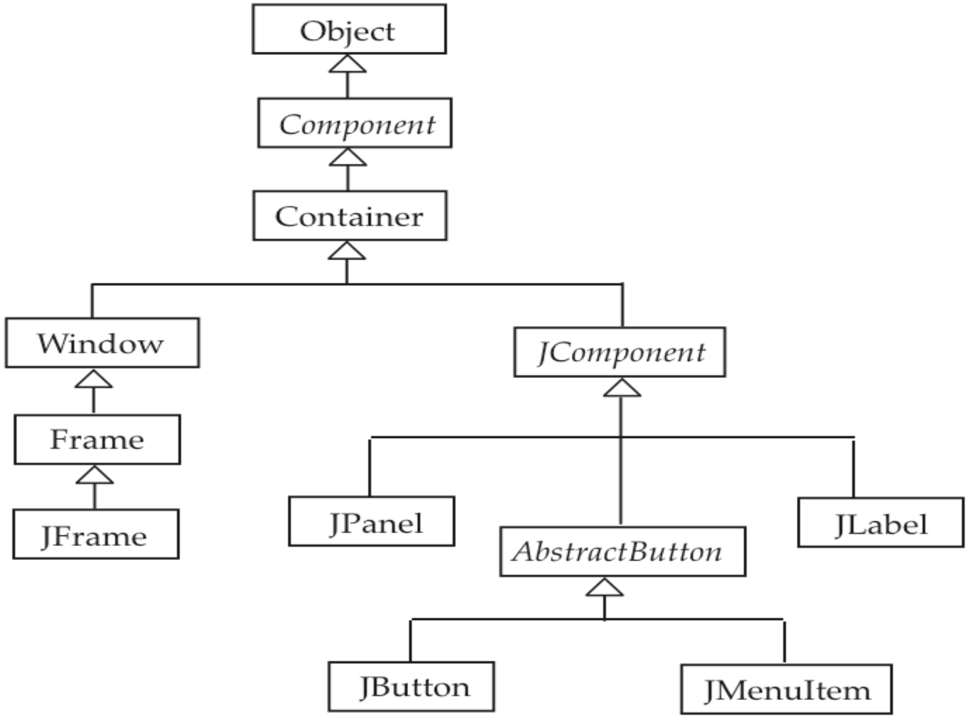
Day 03: GUI Programming

Slide Credits: *Internet / Chetan Arora*

Introduction

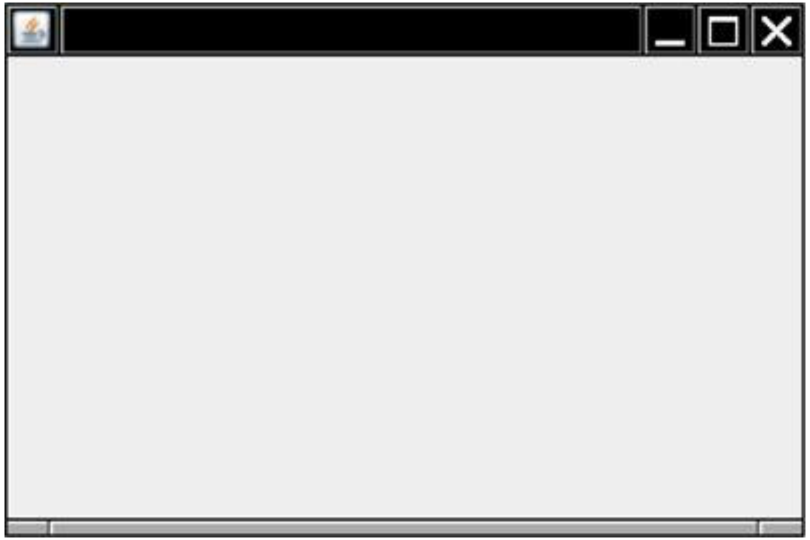
- Java 1.0: Contained AWT (Abstract Window Toolkit) class library for basic GUI programming.
- SWING: GUI Toolkit available from Java 1.1
- Swing has a rich and convenient set of user interface elements.
- Swing has few dependencies on the underlying platform; it is therefore less prone to platform-specific bugs.
- Swing gives a consistent user experience across platforms.

Inheritance Hierarchy for Frame and Component Classes



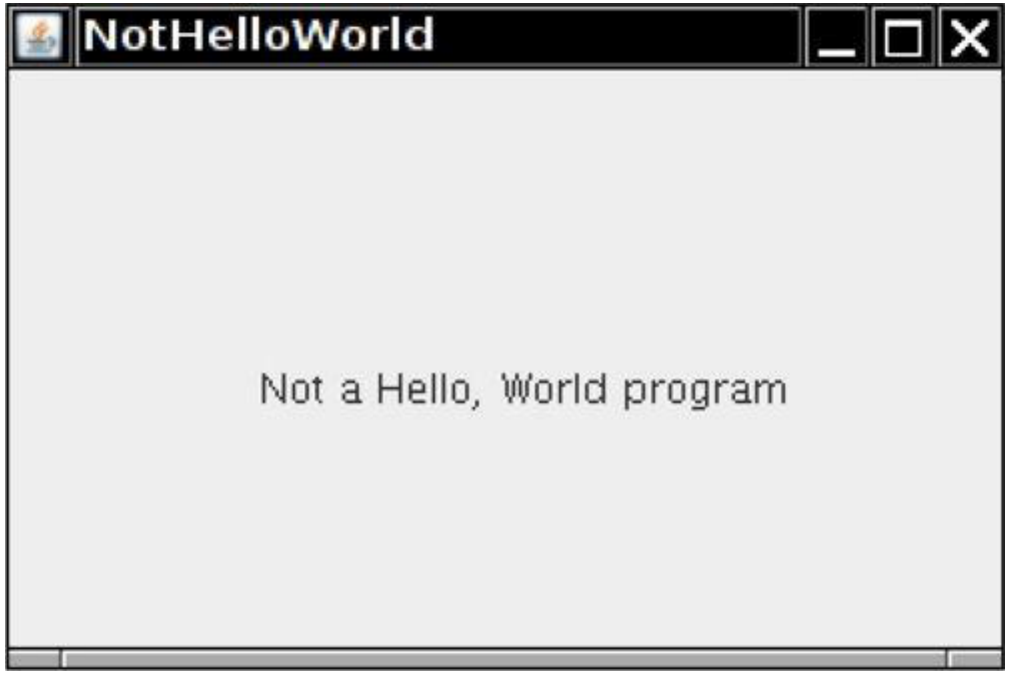
Create a Frame

- Top-level Window
- JFrame Class in Swing (extends Frame Class)
- Not painted on a canvas



```
public class SizedFrameTest
{
    public static void main(String[] args)
    {

            JFrame frame = new JFrame();
            frame.setTitle("BasicFrame");
            frame.setSize(300,400);
            frame.setIconImage(new ImageIcon("smiley.jpg").getImage());
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setVisible(true);

     }
  }
```

Display Information in Component



Not a Hello, World program

Display Information in Component

- Could draw the message string directly onto a frame
  - Bad programming practice
  - Frames are really designed to be containers for components
  - Normally draw on another component added to the frame
  - When designing a frame, you add components into the content pane

```
Container contentPane = frame.getContentPane();
Component c = . . .;
contentPane.add(c);
```

Display Information in Component

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

public class NotHelloWorld
{
    public static void main(String[] args)
    {
       EventQueue.invokeLater(new Runnable()
          {
             public void run()
             {
                JFrame frame = new NotHelloWorldFrame();
                frame.setTitle("NotHelloWorld");
             frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setVisible(true);
             }
          });
     }
  }
```

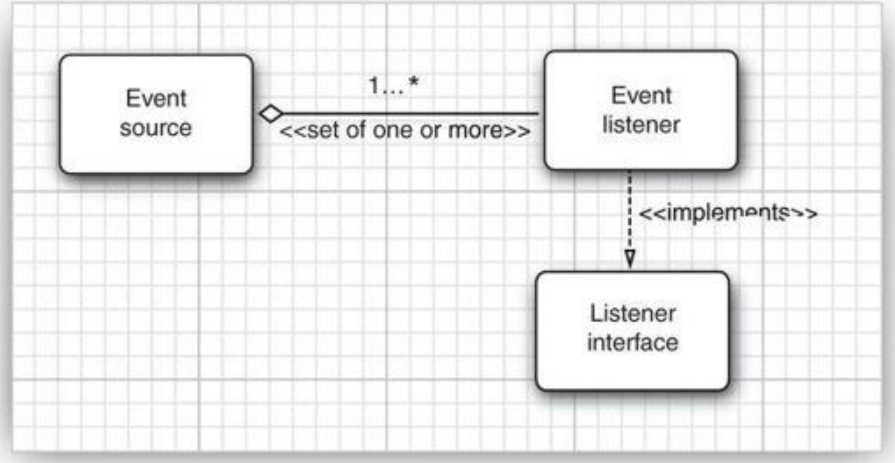
Display Information in Component

```
/**
     * A frame that contains a message
     panel
     */
    class NotHelloWorldFrame extends
     JFrame
    {
       public NotHelloWorldFrame()
       {
          add(new
     NotHelloWorldComponent());
          pack();
       }
    }
```

```
/**
* A component that displays a message.
*/
class NotHelloWorldComponent extends JComponent
  {
    public static final int MESSAGE_X = 75;
    public static final int MESSAGE_Y = 100;
    private static final int DEFAULT_WIDTH =
      300;
    private static final int DEFAULT_HEIGHT
      =200;
    public void paintComponent(Graphics g)
     {
      g.drawString("Not a Hello, World program",
     MESSAGE_X, MESSAGE_Y);
     }
     public Dimension getPreferredSize() {
     return new Dimension(DEFAULT_WIDTH,
      DEFAULT_HEIGHT); }
    }
```

Event Handling

Any operating environment that supports GUIs constantly monitors events such as keystrokes or mouse clicks



Event Handling

- A listener object is an instance of a class that implements a special interface called (naturally enough) a listener interface.
- An event source is an object that can register listener objects and send them event objects.
- The event source sends out event objects to all registered listeners when that event occurs.
- The listener objects will then use the information in the event object to determine their reaction to the event

Event Handling

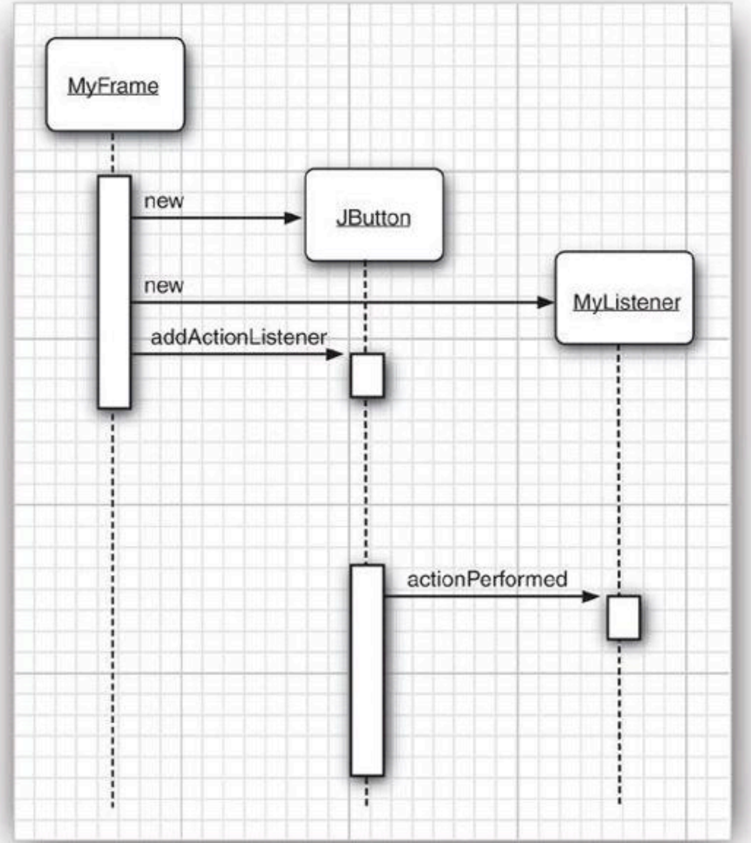
- Specify Listener

```
ActionListener listener = . . .;
JButton button = new JButton("Ok");
button.addActionListener(listener);
```

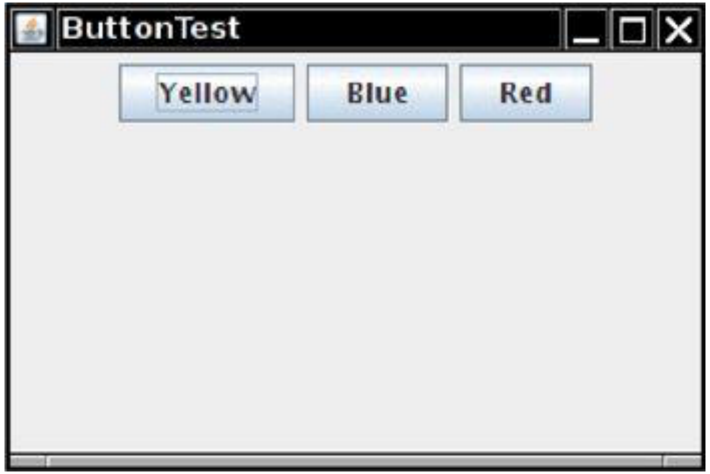
Event Handling

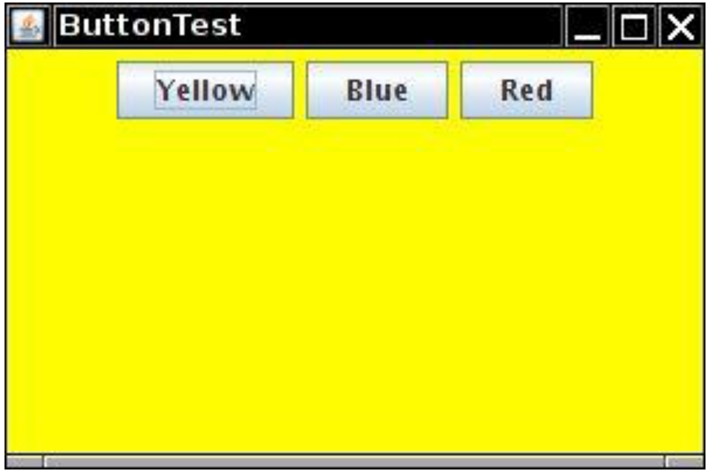
- To implement the ActionListener interface, the listener class must have a method called actionPerformed that receives an ActionEvent object as a parameter.

```
class MyListener implements ActionListener
{
   . . .
   public void actionPerformed(ActionEvent event)
   {
      // reaction to button click goes here
      . . .
   }
}
```


MyFrame
JButton
MyListener
new
new
addActionListener
actionPerformed

Example: Handling a Button Click





```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
 /**
  * A frame with a button panel
  */
 public class ButtonFrame extends JFrame
  {
    private JPanel buttonPanel;
    private static final int DEFAULT_WIDTH = 300;
    private static final int DEFAULT_HEIGHT = 200;

     public ButtonFrame()
     {
        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
        JButton yellowButton = new
JButton("Yellow");
        JButton blueButton = new JButton("Blue");
        JButton redButton = new JButton("Red");

        buttonPanel = new JPanel();
```

```
// add buttons to panel
buttonPanel.add(yellowButton);
buttonPanel.add(blueButton);
buttonPanel.add(redButton);

// add panel to frame
add(buttonPanel);

// create button actions
ColorAction yellowAction = new
ColorAction(Color.YELLOW);

ColorAction blueAction = new
ColorAction(Color.BLUE);

ColorAction redAction = new
ColorAction(Color.RED);

// associate actions with buttons
yellowButton.addActionListener(yellowActi
on);
blueButton.addActionListener(blueAction);
redButton.addActionListener(redAction);
}
```

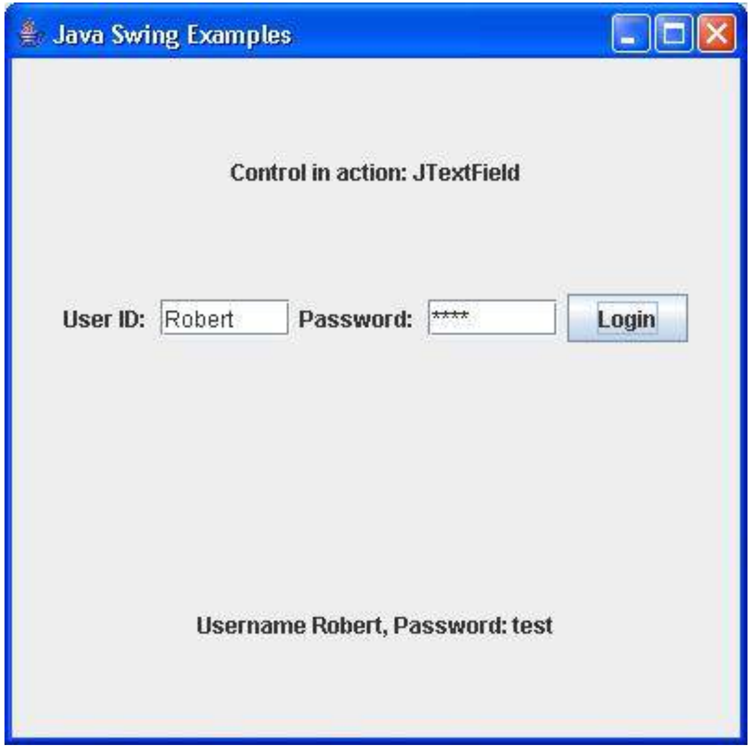
```
    /**
     * An action listener that sets the panel's background color.
     */
    private class ColorAction implements ActionListener
    {
       private Color backgroundColor;

       public ColorAction(Color c)
       {
          backgroundColor = c;
       }

       public void actionPerformed(ActionEvent event)
       {
          buttonPanel.setBackground(backgroundColor);
       }
    }
 }
```

Alternative

```
public void actionPerformed(ActionEvent e) {

       Color color = display.getBackground();

       int red = color.getRed();
       int green = color.getGreen();
       int blue = color.getBlue();

       if (e.getActionCommand().equals("Red")) {
           if (red == 0) {
               red = 255;
           } else {
               red = 0;
           }
       }
```

```
if (e.getActionCommand().equals("Green")) {
           if (green == 0) {
               green = 255;
           } else {
               green = 0;
           }
       }

if (e.getActionCommand().equals("Blue")) {
           if (blue == 0) {
               blue = 255;
           } else {
               blue = 0;
           }
       }
Color setCol = new Color(red, green, blue);
       display.setBackground(setCol);
   }
```

Using TextFields: JTextFields Example



Control in action: JTextField

User ID: Robert Password: **** Login

Username Robert, Password: test

Create Basic GUI

```
private void prepareGUI(){
     mainFrame = new JFrame("Java Swing Examples");
     mainFrame.setSize(400,400);
     mainFrame.setLayout(new GridLayout(3, 1));
     mainFrame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent windowEvent){
           System.exit(0);
        }
       });
```

Create Basic GUI: Add components on the Frame

Continued...

```
 headerLabel = new JLabel("", JLabel.CENTER);
statusLabel = new JLabel("",JLabel.CENTER);

 statusLabel.setSize(350,100);

controlPanel= new JPanel();

controlPanel.setLayout(new FlowLayout());

mainFrame.add(headerLabel);

mainFrame.add(controlPanel);

mainFrame.add(statusLabel);

mainFrame.setVisible(true);
```

Adding TextFields: JTextFields

```
private void showTextFieldDemo(){
     headerLabel.setText("Control in action: JTextField");
     JLabel namelabel= new JLabel("User ID: ", JLabel.RIGHT);
     JLabel passwordLabel = new JLabel("Password:", JLabel.CENTER);
     final JTextField userText = new JTextField(6);
     final JPasswordField passwordText = new JPasswordField(6);
```

Adding Button and Event handling

```
 JButton loginButton = new JButton("Login");
loginButton.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
      String data = "Username " + userText.getText();
      data += ", Password: " + new String(passwordText.getPassword());
      statusLabel.setText(data);
  }
});
controlPanel.add(namelabel);
controlPanel.add(userText);
controlPanel.add(passwordLabel);
controlPanel.add(passwordText);
controlPanel.add(loginButton);
mainFrame.setVisible(true);
}
```

Java Operator Precedence and Associativity

Operators	**Precedence**	**Associativity**
Postfix increment and decrement	++ --	left to right
Prefix increment and decrement, and unary	++ -- + - ~ !	right to left
Multiplicative	* / %	left to right
Additive	+ -	left to right
Shift	<< >> >>>	left to right
Relational	< > <= >= instanceof	left to right
Equality	== !=	left to right
Bitwise AND	&	left to right
Bitwise exclusive OR	^	left to right
Bitwise inclusive OR	|	left to right
Logical AND	&&	left to right
Logical OR	||	left to right
Ternary	? :	right to left
Assignment	= += -= *= /= %= &= ^= | = <<=>>= >>>=	left to right