

© 2003 Walter Savitch

These slides are for the exclusive use of students in
CSE 11 at UCSD, Winter quarter 2003.

They may not be copied or used for any other
purpose without the written permission of Walter
Savitch.

wsavitch@ucsd.edu

**GUI ==
Graphical User Interface ==
Windowing systems that
interact with the user.**

**SWING == Java Library for
GUIs**

**SWING Uses Event-Driven
Programming**

Event-Driven Programming

Windows, mouse, etc. are represented by objects.

When something happens (e.g. a mouse click) an EVENT is fired.

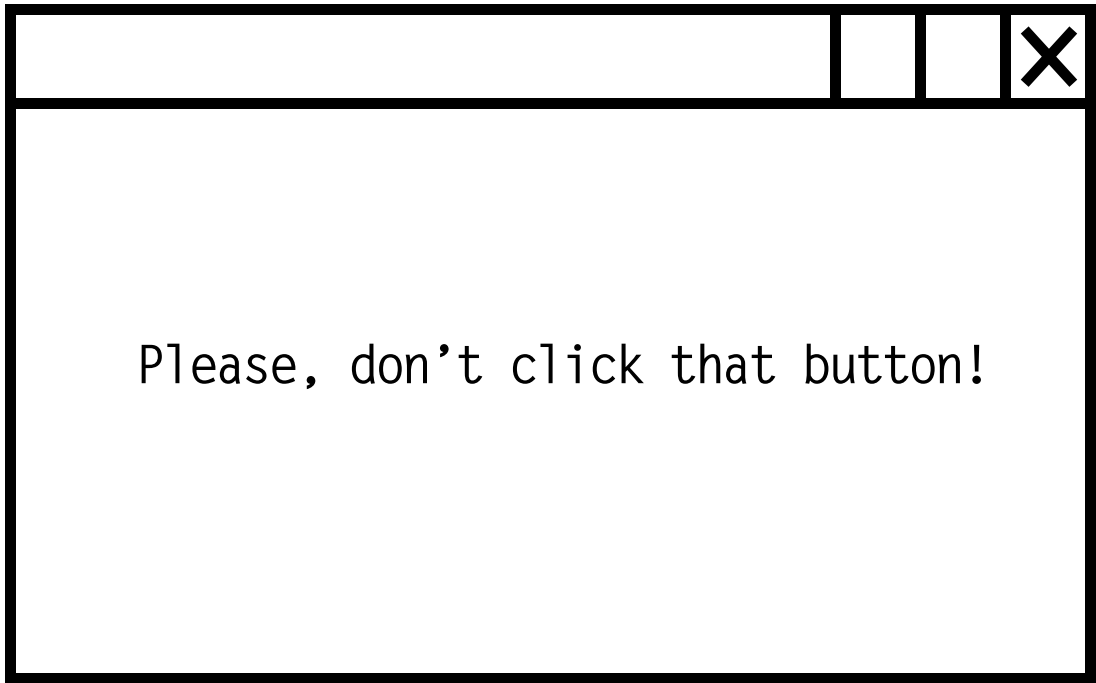
An EVENT is an object.

The event is sent to another object called a LISTENER.

The listener handles the event by invoking a method called an EVENT HANDLER

You the programmer will define (or redefine) these event-handler methods.

First GUI



```
import javax.swing.*;

public class FirstSwingDemo
{
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;

    public static void main(String[] args)
    {
        JFrame myWindow = new JFrame();
        myWindow.setSize(WIDTH, HEIGHT);
        JLabel myLabel = new JLabel(
            "Please don't click that button!");
        myWindow.getContentPane().add(myLabel);

        WindowDestroyer myListener =
            new WindowDestroyer();
        myWindow.addWindowListener(myListener);

        myWindow.setVisible(true);
    }
}
```

Classes:

JFrame for a window

**(Usually use a derived class of
JFrame)**

Methods:

```
JFrame myWindow = new JFrame();  
myWindow.setSize(WIDTH, HEIGHT);  
myWindow.setVisible(true);
```

```
myWindow.getContentPane().add(myLabel);  
myWindow.addWindowListener(myListener);
```

Pixels

A PIXEL is the smallest unit of space on which your screen can write. The more pixels you have on a screen, the greater the screen resolution.

Both size and position of objects on the screen are measured in pixels.

```
public class FirstSwingDemo
{
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;

    public static void main(String[] args)
    {
        JFrame myWindow = new JFrame();
        myWindow.setSize(WIDTH, HEIGHT);
    }
}
```

Classes:

`JLabel` **for a label**

= A string to add to a Window or such

Classes:

WindowDestroyer responds to a click of the close window button

Classes:

```
WindowDestroyer myListener =  
    new WindowDestroyer();  
myWindow.addWindowListener(myListener);
```

Not Part of Java, but short easy definition

```

import java.awt.*;
import java.awt.event.*;

/*****
 *If you register an object of this class as
 *a listener to any object of the class Frame,
 *then if the user clicks the close-window
 *button in the Frame, the object of this class
 *will end the program and close the Frame.
 *****/
public class WindowDestroyer
        extends WindowAdapter
{
    public void windowClosing(WindowEvent e)
    {
        System.exit(0);
    }
}

```

WindowAdapter is a predefined AWT class.

windowClosing is a method redefined by you.

```
public class WindowDestroyer
    extends WindowAdapter
{
    public void windowClosing(WindowEvent e)
    {
        System.exit(0);
    }
}
```

WindowAdapter is a predefined AWT class.

Classes for handling window events are derived from WindowAdapter.

windowClosing is a method redefined by you.

The heading

public void windowClosing(WindowEvent e)
is determined for you.

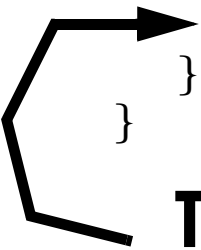
```
public class WindowDestroyer
    extends WindowAdapter
{
    public void windowClosing(WindowEvent e)
    {
        System.exit(0);
    }
}
```

WindowEvent is the class for window events, like clicking the close-window button.

If you click the close-window button that fires an event of the class WindowEvent and evokes the method windowClosing with the event as parameter. (In this case the parameter is not used.)

REVIEW

```
public class WindowDestroyer
    extends WindowAdapter
{
    public void windowClosing(WindowEvent e)
    {
        System.exit(0);
    }
}
```



The body of the method is all you get to define.

The heading of the method is determined for you and you cannot change it.

The Normal Way to Get That First Window

```
import javax.swing.*;

public class FirstWindow extends JFrame
{
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;

    public FirstWindow()
    {
        super();

        setSize(WIDTH, HEIGHT);
        JLabel myLabel = new JLabel(
            "Please don't click that button!");
        getContentPane().add(myLabel);

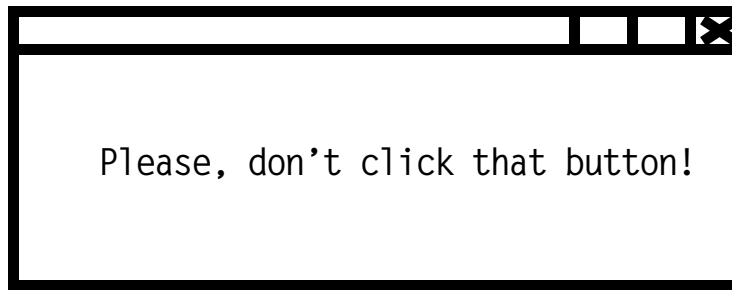
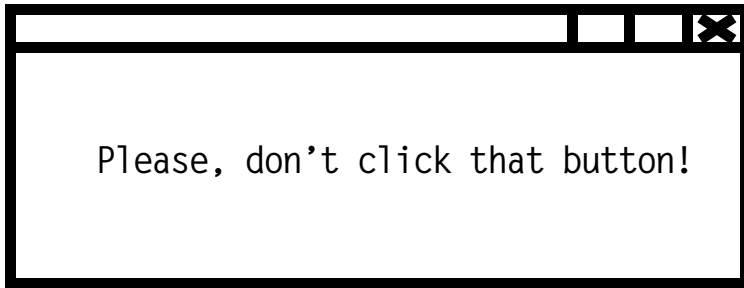
        WindowDestroyer listener =
            new WindowDestroyer();
        addWindowListener(listener);
    }
}
```

```
import javax.swing.*;

public class FirstWindowDemo
{
    public static void main(String[] args)
    {
        FirstWindow window1 =
            new FirstWindow();
        window1.setVisible(true);

        FirstWindow window2 =
            new FirstWindow();
        window2.setVisible(true);
    }
}
```

IGUI



Always need:

```
import javax.swing.*;
```

Sometimes need:

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

When doing a window usually need

```
public class Window_Class_Name extends JFrame  
{  
    public static final int WIDTH = A_Number;  
    public static final int HEIGHT = Other_Num;
```

For example:

```
public class FirstWindow extends JFrame  
{  
    public static final int WIDTH = 300;  
    public static final int HEIGHT = 200;
```

JFrame is the class for windows.

You can add a JLabel to a JFrame as follows:

```
public FirstWindow()  
{  
    JLabel myLabel = new JLabel(  
        "Please don't click that button!");  
    getContentPane().add(myLabel);  
}
```

getContentPane() is an accessor method that gets "the content pane" (inside of frame that holds the stuff you add).

```
getContentPane().add(myLabel);
```

Is equivalent to

```
Container identifierOfYourChoice =  
    getContentPane();  
identifierOfYourChoice.add(myLabel);
```

Color

Almost anything can have a color.

What is a color in Java Swing?

It's actually in the AWT and is the class named `Color`.

Some handy constants:

```
Color.black  
Color.blue  
Color.cyan  
Color.darkGray  
Color.gray  
Color.green  
Color.lightGray  
Color.magenta  
Color.orange  
Color.pink  
Color.red  
Color.white  
Color.yellow
```

Need: `import java.awt.*;`

```

import javax.swing.*;
import java.awt.*; //needed for the Color class

public class SecondWindow extends JFrame
{
    public static final int WIDTH = 500;
    public static final int HEIGHT = 400;

    public SecondWindow()
    {
        setSize(WIDTH, HEIGHT);
        Container contentPane =
            getContentPane();
        JLabel label = new JLabel(
            "Coming to you in living color!");
        contentPane.add(label);

        setTitle("Second Window");
        contentPane.setBackground(Color.blue);

        addWindowListener(
            new WindowDestroyer());
    }
}

```

```

public SecondWindow(Color customColor)
{
    setSize(WIDTH, HEIGHT);
    Container contentPane =
        getContentPane();
    JLabel label = new JLabel(
        "Coming to you in living color!");
    contentPane.add(label);

    setTitle("Second Window");
    contentPane.setBackground(customColor);

    addWindowListener(
        new WindowDestroyer());
}
}

```

Using `new` as Part of a Method Argument

```
addWindowListener(new WindowDestroyer());
```

`new WindowDestroyer()` **creates an object of the class** `WindowDestroyer` **and returns a reference to that object.**

```

import java.awt.*; //for the class Color

public class SecondWindowDemo
{
    /**
     *Creates and displays two windows
     *of the class SecondWindow.
     */
    public static void main(String[] args)
    {
        SecondWindow window1 =
            new SecondWindow();
        window1.setVisible(true);

        SecondWindow window2 =
            new SecondWindow(Color.pink);
        window2.setVisible(true);
    }
}

```

Layout Manger

An object that arranges things in a container, like a JFrame.

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

public class BorderLayoutDemo extends JFrame
{
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;

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

```
public BorderLayoutDemo()
{
    setSize(WIDTH, HEIGHT);
    addWindowListener(
        new WindowDestroyer());
    setTitle("Layout Demonstration");
    Container content =
getContentPane();

    content.setLayout(new
BorderLayout());

    JLabel label1 =
        new JLabel("First label
here.");
    content.add(label1,
BorderLayout.NORTH);

    JLabel label2 =
        new JLabel("Second label
there.");
    content.add(label2,
BorderLayout.SOUTH);

    JLabel label3 =
        new JLabel("Third label
anywhere.");
    content.add(label3,
BorderLayout.CENTER);
}
}
```

Border Layout Manager

```
public BorderLayoutDemo()  
{  
    ...  
    Container content = getContentPane();  
    content.setLayout(new BorderLayout());  
    content.add(label1, BorderLayout.NORTH);  
    content.add(label2, BorderLayout.SOUTH);  
    ...  
    content.add(label3, BorderLayout.CENTER);  
}
```

BorderLayout.NORTH		
BorderLayout. WEST	BorderLayout.CENTER	BorderLayout. EAST
BorderLayout.SOUTH		

Grid Layout

```
setLayout(new GridLayout(2, 3));
```

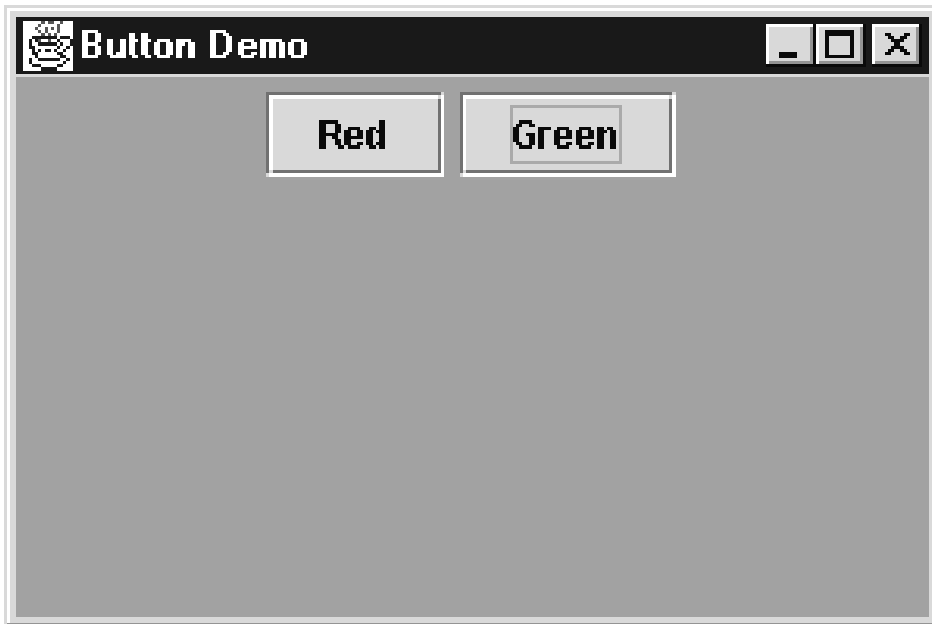

**Must fill all grid elements in order,
row by row.**

Flow Layout

```
setLayout(new FlowLayout());
```

One after the other left to right.

Buttons



```

public class ButtonDemo extends JFrame
                        implements ActionListener
{
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;

    public ButtonDemo()
    {
        setSize(WIDTH, HEIGHT);

        addWindowListener(
            new WindowDestroyer());
        setTitle("Button Demo");
        Container contentPane =
            getContentPane();
        contentPane.setBackground(Color.blue);
        contentPane.setLayout(
            new FlowLayout());

        JButton stopButton =
            new JButton("Red");
        stopButton.addActionListener(this);
        contentPane.add(stopButton);

        JButton goButton =
            new JButton("Green");
        goButton.addActionListener(this);
        contentPane.add(goButton);
    }
}

```

```
public void actionPerformed(ActionEvent e)
{
    Container contentPane =
        getContentPane();
    String ActCmmd = e.getActionCommand();
    if (ActCmmd.equals("Red"))
        contentPane.setBackground(Color.red);
    else if (ActCmmd.equals("Green"))
        contentPane.setBackground(Color.green);
    else
        System.out.println("Error");
}
}
```

A JFrame (or any other class) Can Be It's Own Action Listener

```
public class ButtonDemo extends JFrame
    implements ActionListener
{
    public ButtonDemo()
    {
        Container contentPane = getContentPane();

        JButton stopButton = new JButton("Red");
        stopButton.addActionListener(this);
        contentPane.add(stopButton);
        . . .
    }

    public void actionPerformed(ActionEvent e)
    {
        . . .
    }
}
```

A Listener Can Be Something Else Besides Being A Listener!

actionPerformed Method

```
public void actionPerformed(ActionEvent e)
{
    Container contentPane = getContentPane();
    String actionCommand =
        e.getActionCommand();

    if (actionCommand.equals(...))
        .....
    else if (actionCommand.equals(...))
        .....
    else if (actionCommand.equals(...))
        .....
    .
    .
    .
    else
        System.out.println("Error...");
}
```

Interface

An interface is a property of a class that says what methods it must have. A class that satisfies an interface is said to implement the interface.

An Interface Is A Type.

In order to implement an interface, a class must include the phrase

`implements ActionListener`

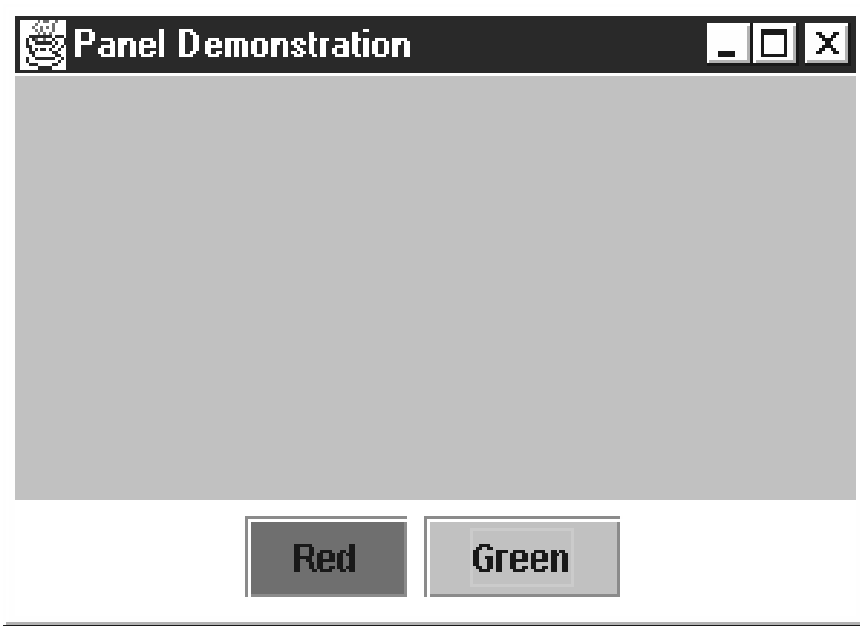
(or whatever the name of the interface is) and it must define all the methods specified in the interface. (The particular interface `ActionListener` specifies only the one method `actionPerformed`.)

setActionCommand

```
JButton stopButton = new JButton("Red");  
stopButton.setActionCommand("Stop");
```

The button stopButton will have "Red" written on it, but it's action command will be "Stop", not "Red".

Panels: The JPanel Class



```

public class PanelDemo extends JFrame
                        implements ActionListener
{
    public PanelDemo()
    {
        setSize(WIDTH, HEIGHT);
        addWindowListener(new WindowDestroyer());
        setTitle("Panel Demonstration");
        Container contentPane = getContentPane();
        contentPane.setBackground(Color.blue);
        contentPane.setLayout(
            new BorderLayout());

        JPanel buttonPanel = new JPanel();
        buttonPanel.setBackground(Color.white);

        buttonPanel.setLayout(new FlowLayout());

        JButton stopButton = new JButton("Red");
        stopButton.setBackground(Color.red);
        stopButton.addActionListener(this);
        buttonPanel.add(stopButton);

        JButton goButton = new JButton("Green");
        goButton.setBackground(Color.green);
        goButton.addActionListener(this);
        buttonPanel.add(goButton);

        contentPane.add(
            buttonPanel, BorderLayout.SOUTH);
    }
}

```

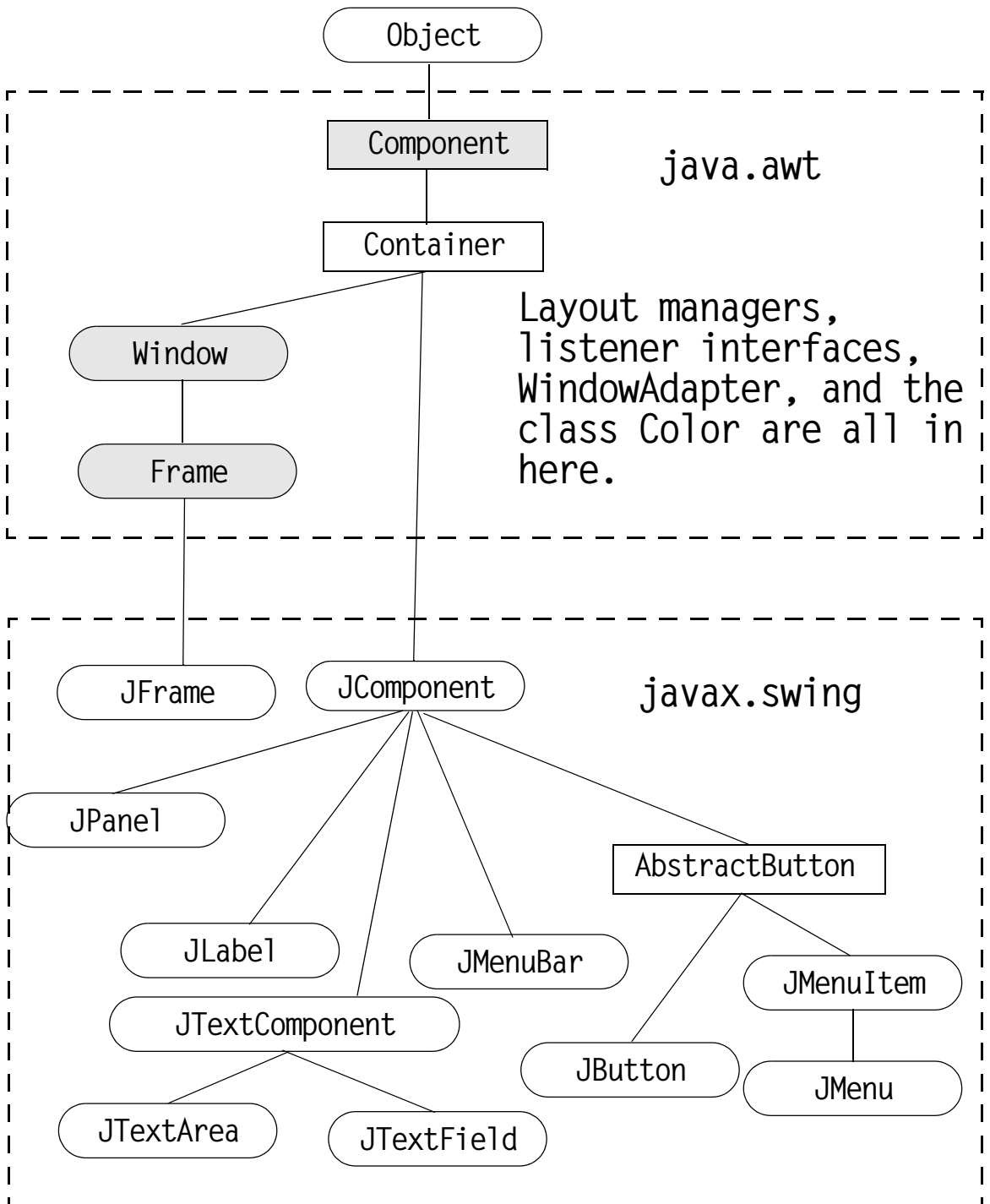
```
public PanelDemo()
{
    Container contentPane = getContentPane();
    contentPane.setBackground(Color.blue);
    contentPane.setLayout(
        new BorderLayout());

    JPanel buttonPanel = new JPanel();
    buttonPanel.setBackground(Color.white);
    buttonPanel.setLayout(new FlowLayout());

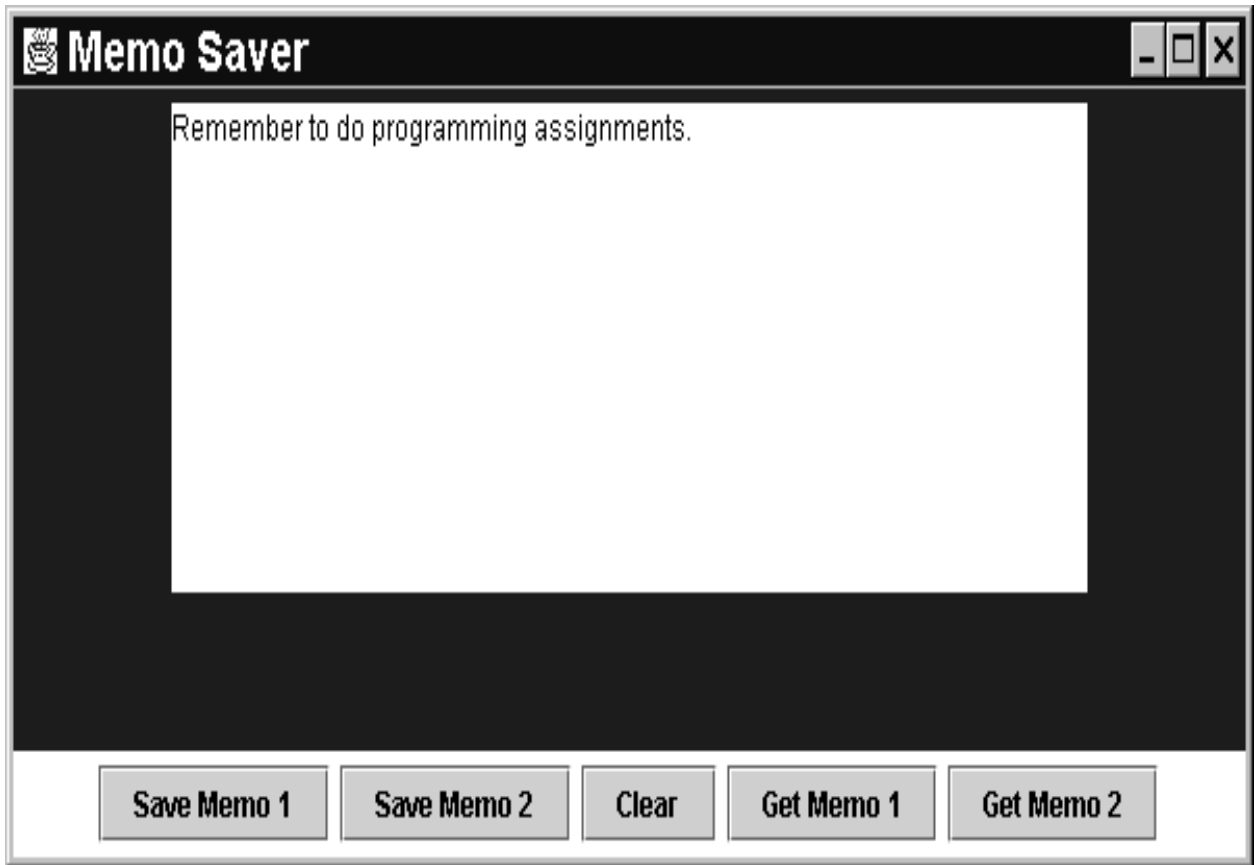
    ...
    stopButton.addActionListener(this);
    buttonPanel.add(stopButton);

    ...
    goButton.addActionListener(this);
    buttonPanel.add(goButton);

    contentPane.add(
        buttonPanel, BorderLayout.SOUTH);
}
```



JTextComponents: JTextArea and JTextField



```

public class TextAreaDemo extends JFrame
                                implements ActionListener
{
    private JTextArea theText;
    private String memo1 = "No Memo 1.";
    private String memo2 = "No Memo 2.";

    public TextAreaDemo()
    {
        ...
        Container contentPane = getContentPane();
        contentPane.setLayout(new BorderLayout());

        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new FlowLayout());
        JButton memo1Button = new JButton("SaveMemo1");
        memo1Button.addActionListener(this);
        buttonPanel.add(memo1Button);

        ...
        JButton clearButton = new JButton("Clear");
        clearButton.addActionListener(this);
        buttonPanel.add(clearButton);
        JButton get1Button = new JButton("Get Memo1");
        get1Button.addActionListener(this);
        buttonPanel.add(get1Button);

        ...
        contentPane.add(
            buttonPanel, BorderLayout.SOUTH);
    }
}

```

```
        JPanel textPanel = new JPanel();
        theText =
            new JTextArea(LINES,
CHAR_PER_LINE);
        theText.setBackground(Color.white);
        textPanel.add(theText);
        contentPane.add(
            textPanel,
BorderLayout.CENTER);
    }
```

JTextArea

```
private JTextArea theText;

public TextAreaDemo()
{
    ...
    JPanel textPanel = new JPanel();
    theText =
        new JTextArea(LINES, CHAR_PER_LINE);
    theText.setBackground(Color.white);
    textPanel.add(theText);
    contentPane.add(
        textPanel, BorderLayout.CENTER);
}
```

```
public void actionPerformed(ActionEvent e)
{
    String actionCommand = e.getActionCommand();
    if (actionCommand.equals("Save Memo 1"))
        memo1 = theText.getText();
    else if (actionCommand.equals("Save Memo 2"))
        memo2 = theText.getText();
    else if (actionCommand.equals("Clear"))
        theText.setText("");
    else if (actionCommand.equals("Get Memo 1"))
        theText.setText(memo1);
    else if (actionCommand.equals("Get Memo 2"))
        theText.setText(memo2);
    else
        theText.setText("Error ...");
}
}
```

JTextArea

```
private JTextArea theText;

public TextAreaDemo()
{
    ...
    JPanel textPanel = new JPanel();
    theText =
        new JTextArea(LINES, CHAR_PER_LINE);
    theText.setBackground(Color.white);
    textPanel.add(theText);
    contentPane.add(
        textPanel, BorderLayout.CENTER);
}

public void actionPerformed(ActionEvent e)
{
    if (...)
        memo1 = theText.getText();

    else if (...)
        theText.setText(memo2);
    else
        ...;
}
```

Inputting Numbers within a GUI

```
double x = Double.parseDouble(  
    inputOutputField.getText().trim());
```

```
private static double stringToDouble(  
    String stringObject)  
{  
    return  
        Double.parseDouble(stringObject.trim());  
}
```

```
double x =  
    stringToDouble(inputOutputField.getText());
```

GUI Adding Machine

