

# 19. A Quick Java Swing Tutorial

# Introduction

- Swing – A set of GUI classes
  - Part of the Java's standard library
  - Much better than the previous library: AWT
    - Abstract Window Toolkit
- Highlights
  - A rich set of widgets
    - Widget: Any GUI element (also called: components)
  - Contents and shape are separated (MVC support)
  - Fine-grained control over the behavior and look and feel
  - Platform independent
    - Isolates the programmer from the operating system's GUI

# Swing Components

- Containers
  - Contain and manage other components.
  - Top Level/Internal
  - Examples: `JFrame` (Top Level), `JScrollPane`, `JPanel`.
- Basic controls
  - Atomic components
  - Used for showing output and/or getting some input
  - Inherits `JComponent`
  - Examples: `JButton`, `JLabel`, `JTextArea`, `JTable`, `JList`
- Usually every Swing class extends the corresponding AWT class
  - For backward-compatibility reasons

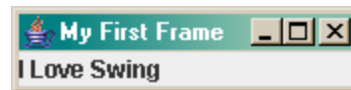
# My First Swing Program

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

public class First {
    public static void main(String[] args) {
        JFrame frame = new JFrame("My First Frame");

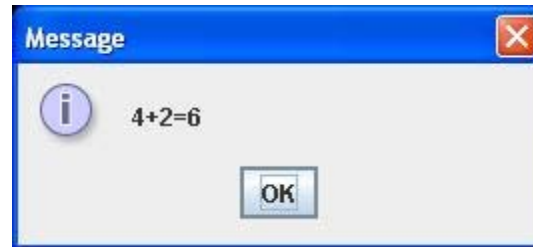
        // operation to do when the window is closed.
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.getContentPane().setLayout(new BorderLayout());
        frame.getContentPane().add(new JLabel("I Love Swing"),
            BorderLayout.CENTER);
        frame.pack();
        frame.setVisible(true);
    }
}
```

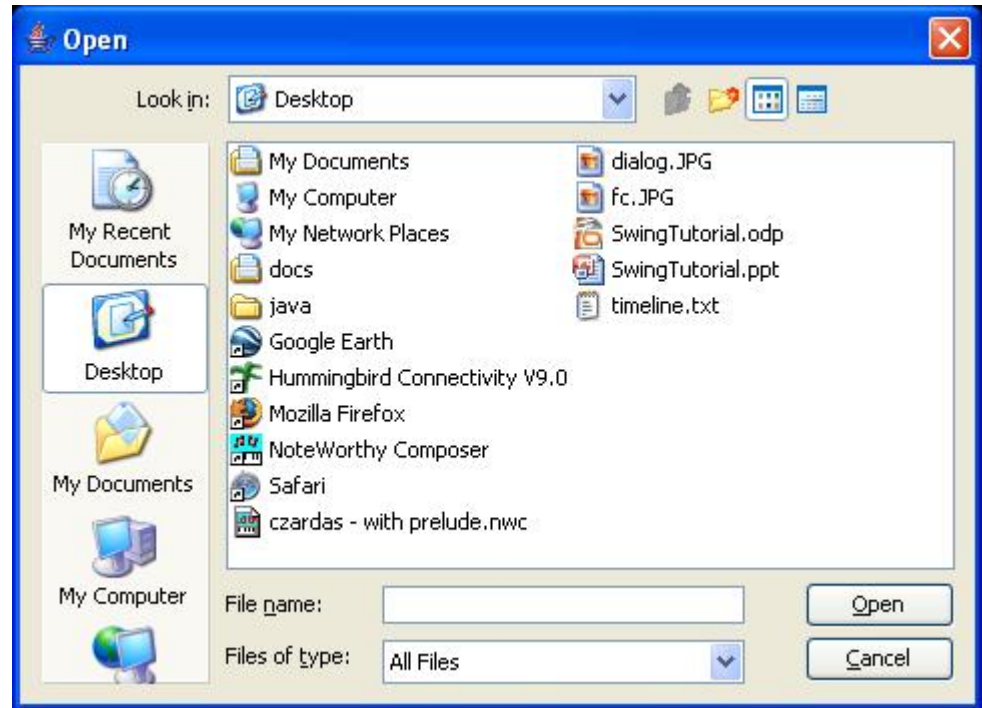


# Top Level Containers: JDialog

- **javax.swing.JDialog:**
  - More simple and limited than frames
  - Typically used for showing a short message on the screen
  - Also has a border and a title bar
  - May have an owner
    - If the owner is invisible the dialog will also be invisible
  - Use the static method of JOptionPane to show standard dialog boxes:  
`JOptionPane.showMessageDialog(null, "4+2=6");`



# Top Level Containers: JFileChooser

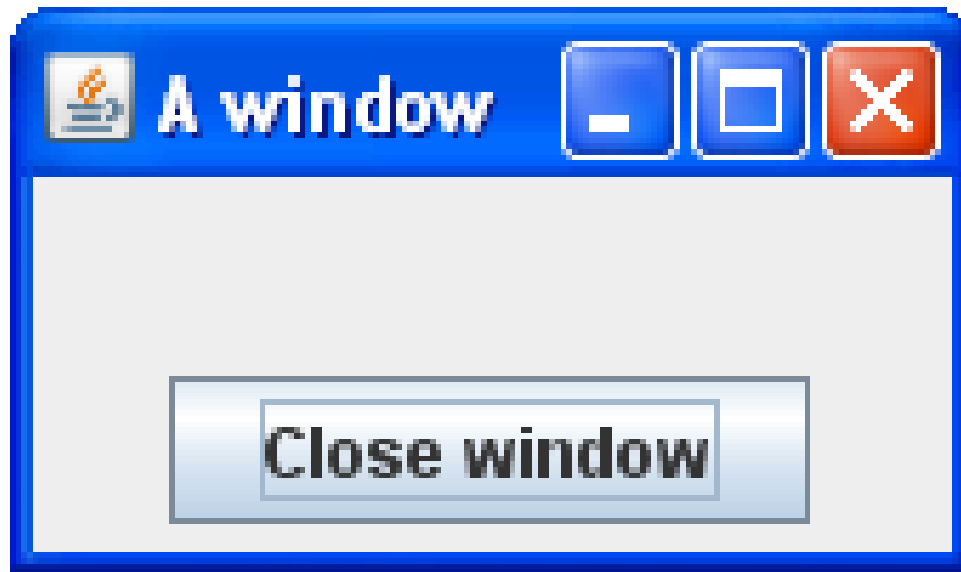


- **javax.swing.JFileChooser.**
  - Allows the the user to choose a file
  - Supports “open” and “save”: `showOpenDialog()` , `showSaveDialog()`

```
JFileChooser fc = new JFileChooser();  
int returnVal = fc.showOpenDialog(null);  
if(returnVal == JFileChooser.APPROVE_OPTION)  
    System.out.println("File: " + fc.getSelectedFile());
```

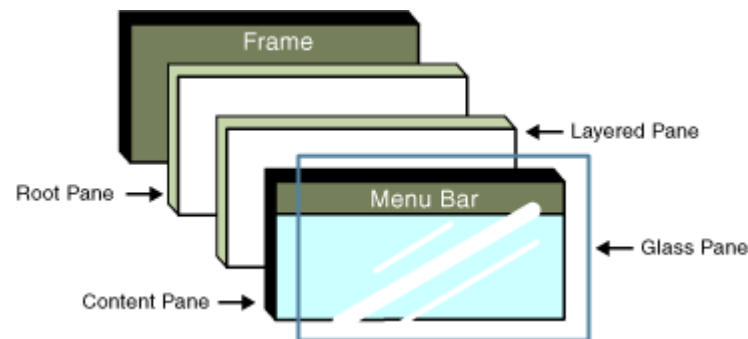
# Top Level Containers: JFrame

- **javax.swing.JFrame:**
  - Top-level window with a title and a border.
  - Usually used as a program's main window



# More on JFrame

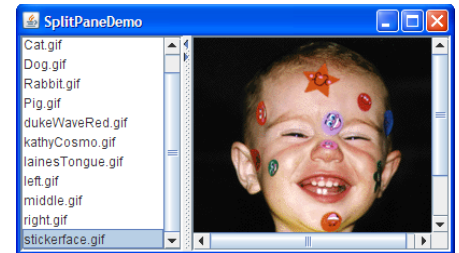
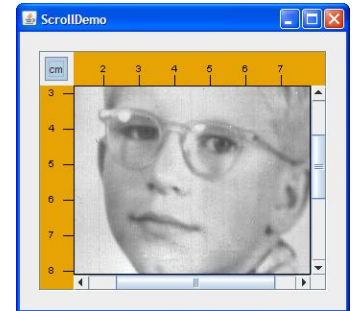
- Made of several layers
- Widgets are added to the Content Pane layer.
  - Use `getContentPane()` to obtain it
- Other layers are used for customizing the window's appearance





# Internal Containers

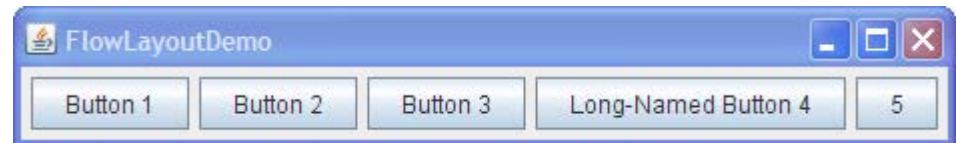
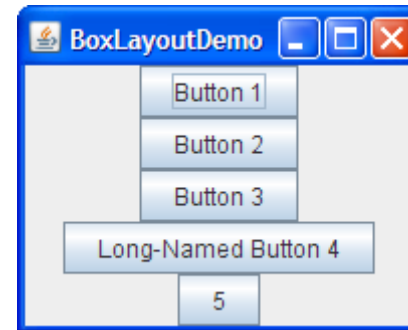
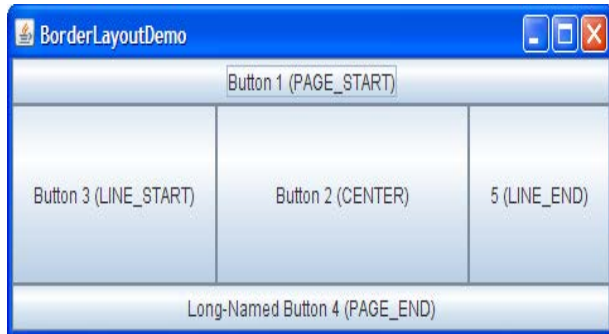
- Not Top level containers
- Can contain other non-top level components
- Examples:
  - JScrollPane: Provides a scrollable view of its components
  - JSplitPane: Separates two components
  - JTabbedPane: User chooses which component to see



# Containers - Layout

- Each container has a layout manager
  - Determines the size, location of contained widgets.
- Setting the current layout of a container:  
*void setLayout(LayoutManager lm)*
- *LayoutManager* implementing classes:
  - BorderLayout
  - BoxLayout
  - FlowLayout
  - GridLayout

# Containers - Layout



# Swing Components

## Basic Controls

Simple components that are used primarily to get input from the user; they may also show simple state.

112kB



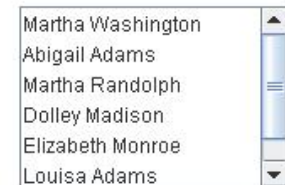
[JButton](#)



[JCheckBox](#)



[JComboBox](#)



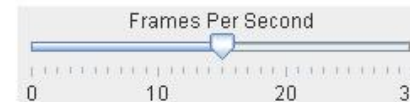
[JList](#)



[JMenu](#)



[JRadioButton](#)



[JSlider](#)



[JSpinner](#)



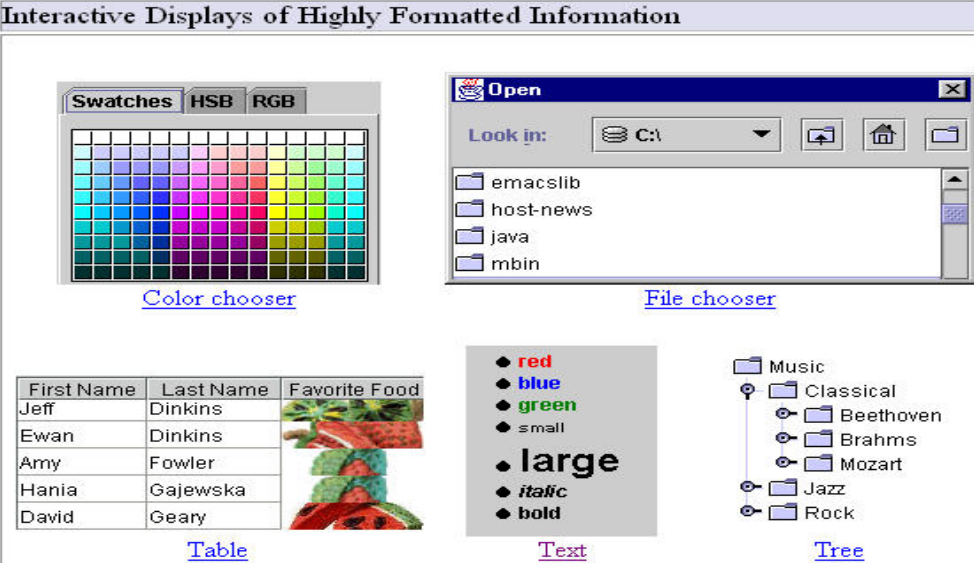
[JTextField](#)



[JPasswordField](#)

# Swing Components

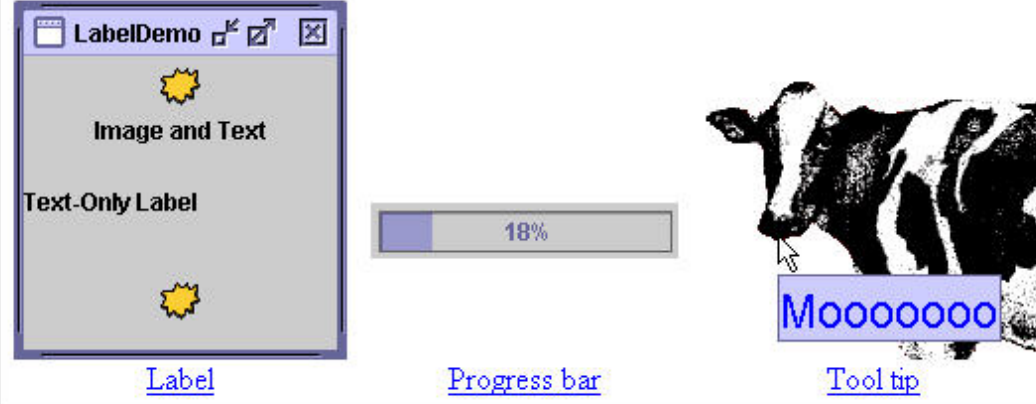
Interactive Displays of Highly Formatted Information



The image displays five Swing components arranged in a grid. Top-left: A 'Color chooser' with 'Swatches', 'HSB', and 'RGB' tabs and a grid of color swatches. Top-right: A 'File chooser' window titled 'Open' showing a file list with 'Look in:' set to 'C:\'. Bottom-left: A 'Table' with columns 'First Name', 'Last Name', and 'Favorite Food'. Bottom-center: A 'Text' area showing a list of text styles like 'red', 'blue', 'green', 'small', 'large', 'italic', and 'bold'. Bottom-right: A 'Tree' view showing a hierarchy of music folders like 'Classical', 'Beethoven', 'Brahms', 'Mozart', 'Jazz', and 'Rock'.

First Name	Last Name	Favorite Food
Jeff	Dinkins	
Ewan	Dinkins	
Amy	Fowler	
Hania	Gajewska	
David	Geary	

## Uneditable Information Displays



The image displays three Swing components. Left: A 'Label' window titled 'LabelDemo' containing an 'Image and Text' label with a sun icon and a 'Text-Only Label' with another sun icon. Center: A 'Progress bar' showing 18% completion. Right: A 'Tool tip' showing a cow image with the text 'Mooooooo'.

# First Swing Program Revisited

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

```
public class First {  
    public static void main(String[] args) {  
        JFrame frame = new JFrame("My First Frame");  
  
        // operation to do when the window is closed  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
        frame.getContentPane().setLayout(new BorderLayout());  
        frame.getContentPane().add(new JLabel("I Love Swing"),  
            BorderLayout.CENTER);  
        frame.pack();  
        frame.setVisible(true);  
    }  
}
```

Create a frame

Choose the border  
layout

Create a text  
label

Specify CENTER  
as the layout  
position

Add the label to  
the content pane

# Input

- So we now know how to present widgets on the screen
- A program also needs to react to the user's actions
- Examples:
  - When the user presses a button we want to save a file
  - When the user closes the program we want to ask “are you sure?”
  - ...
- Swing mechanism: Events and Listeners

# Events, Listeners

- Swing defines all sorts of Listener interfaces
  - E.g.: `ActionListener`, `MouseMotionListener`, *`WindowListener`*, ...

```
public interface ActionListener extends EventListener {  
    public void actionPerformed(ActionEvent e);  
}
```

```
public interface MouseMotionListener extends EventListener {  
    public void mouseDragged(MouseEvent e);  
    public void mouseMoved(MouseEvent e);  
}
```

- There are default (empty) implementations for many of the listeners
  - E.g.: `MouseMotionAdapter`, *`WindowAdapter`*



# Events, Listeners (cont.)

- A listener is an object that implements a listener interface
- If we need to react to an event (on a certain widget) we register a listener object with that widget
- E.g.: `addActionListener()` registers an action listener with its receiver:

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

- When an event occurs, all registered listeners are notified
  - The appropriate listener method (e.g: `actionPerformed()`) is invoked
  - An object describing the event is passed as a parameter

# Event Handling Demo: GUI



# Event Handling Demo: Code

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

public class Events implements ActionListener {
    public Events() {
        JFrame frame = new JFrame("Events");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.getContentPane().setLayout(new FlowLayout());
        JButton b = new JButton("Click me!");
        b.addActionListener(this);
        frame.getContentPane().add(b);

        frame.pack();
        frame.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(null, "Thank you");
    }
    public static void main(String[] args) { new Events(); }
}
```

# Inner Classes

- Nested within another classes
- Instance specific:
  - Has access to methods & fields of the object that created it
  - => An inner class has TWO this variables
- Can be static
  - Can access only static members and methods only
  - A static method cannot create a non-static inner class

# Local Classes

- Same as inner classes but defined inside a method
- Has access to local variables of the enclosing method
  - Only if the variable is defined as final
- Can be anonymous
  - Doesn't have a name.

# Event Handling Demo: Local Class

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

public class Events {
    public Events() {
        JFrame frame = new JFrame("Events");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.getContentPane().setLayout(new FlowLayout());
        JButton b = new JButton("Click me!");
        b.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(null, "Thank you");
            }
        });
        frame.getContentPane().add(b);

        frame.pack();
        frame.setVisible(true);
    }
    public static void main(String[] args) { new Events(); }
}
```

# Accessing Fields of Enclosing Object

```
public class A {  
    int x = 0;  
    public void f() {  
        B b = new B();  
        b.g();  
        System.out.println(x);    // Output: 5  
    }  
}
```

内部类

```
public class B {  
    public void g() { x = 5; }  
}
```

```
public static void main(String[] args) {  
    new A().f();  
}
```

# Using the Second this Variable

```
public class A {  
    public void f() {  
        B b = new B();  
        System.out.println(b.g());    // Output: 1024  
    }  
  
    public int g() { return 512; }  
  
    public class B {  
        public int g() { return A.this.g() * 2; }  
    }  
  
    public static void main(String[] args) {  
        new A().f();  
    }  
}
```