Human Computer Interface Design
Chapter 13.3 - A Quick Tour of a Swing Toolkit Based-Application's Code Structure

Objective

This section takes you through the code for the Swing Application program. It illustrates the structure of GUI program and the event-driven programming principles.

Main Steps for Creating a GUI

- **Importing Swing packages**
- Choosing the look and feel
- **Setting up the top-level container**
- **Setting up Components**
- **Adding components to containers**
- **Handling events**

The example: Swing Application

- Swing Application brings up a Window and each time the user clicks the button, the label is updated.

1- Importing Swing Packages (Java Swing Toolkit)

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

Most Swing programs also need to import the two main AWT packages:

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

2- Choosing the Look and Feel

Swing allows you to specify which look and feel your program uses – Java look and feel, Windows look and feel, CDE/Motif look and feel, and so on.

Windows Look and Feel

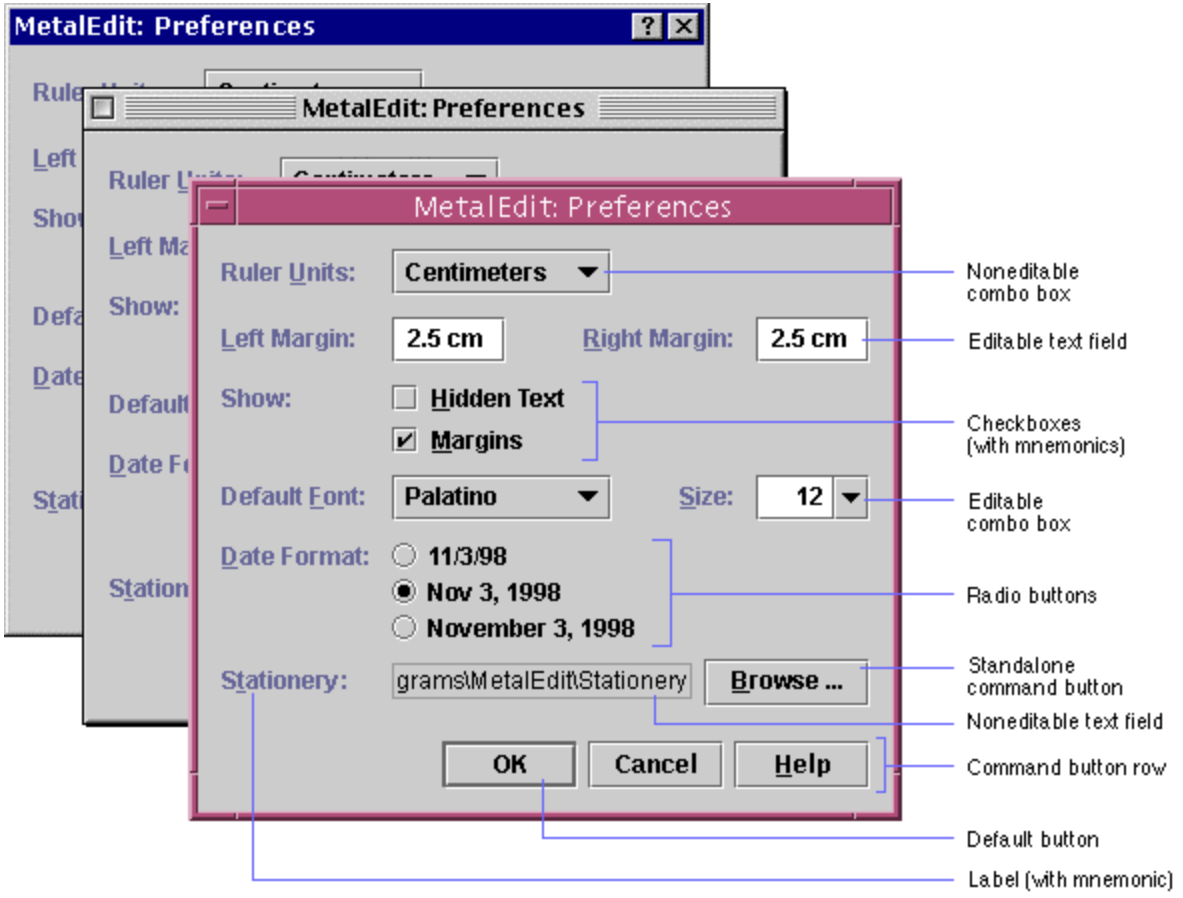
UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");

Java Look and Feel

UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());

Pluggable Look and Feel

- Gives any program that uses Swing components a choice of looks and feels.



3- Setting up the top-level container

- Every program that presents a Swing GUI contains at least one top-level Swing container. For most programs, the top-level Swing containers are instances of:

  1- Jframe: implements a single main window
  2- Jdialog: implements a secondary window
  3- Japplet (for applets): implements an applet's display area within a browser window

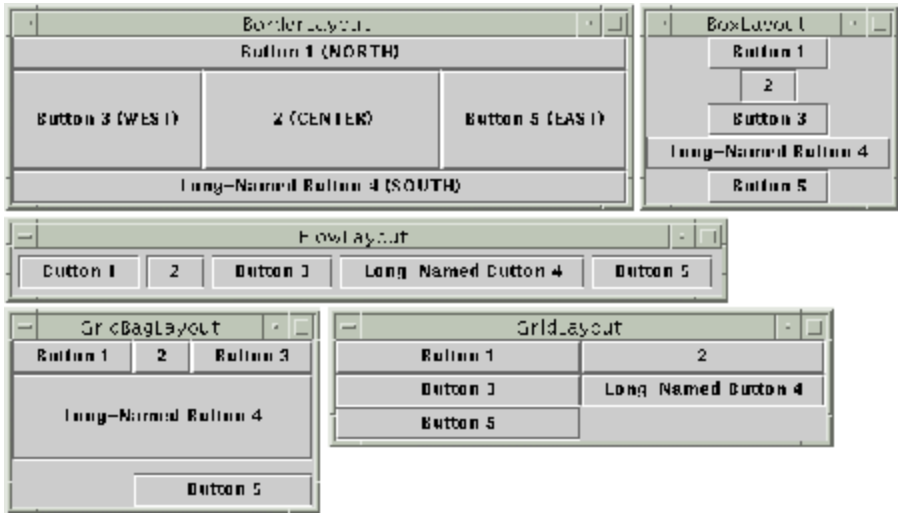
```
//...Create the components to go into the frame...
      JFrame frame = new Jframe ("Swing Application");

//...Stick them in a container named contents...
      frame.getContentPane().add(contents, BorderLayout.CENTER);

//Finish setting up the frame, and show it.
      frame.addWindowListener(...);
      frame.pack();
      frame.setVisible(true);
```

Layout Manager

- A Container always uses **layout managers** to determine the size and position of the components they contain.
- The following figures show the GUIs of five programs, each of which displays five buttons. The buttons are identical, and the code for the programs is almost identical. So why do they look so different? Because they use different layout managers to control the size and position of the buttons.
- Java platform supplies six commonly used layout managers: BorderLayout, BoxLayout, FlowLayout, GridBagLayout, and GridLayout, CardLayout.

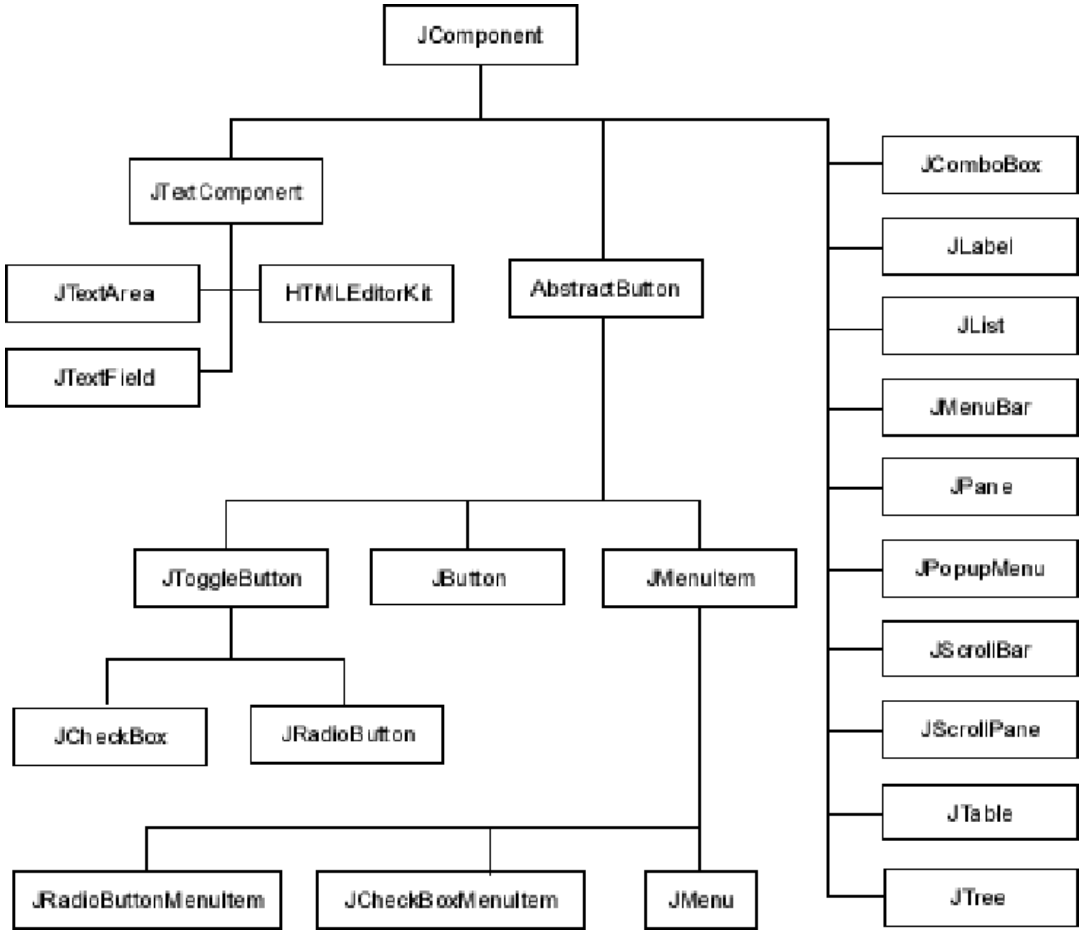


4- Setting up GUI Components

- The Swing Application GUI contains two components a button and a label. The following is the code that initializes the button and label:

```
// Create the Button
JButton button = new JButton ("I'm a Swing button!");
//Sets the M key as the mnemonic that the user can use to simulate a click of the button
button.setMnemonic(KeyEvent.VK_M);
// Create the Button
final JLabel label = new JLabel(labelPrefix + "0    ");
```

Components are instance of the JComponent Class Hierarchy



5- Putting Components in Containers

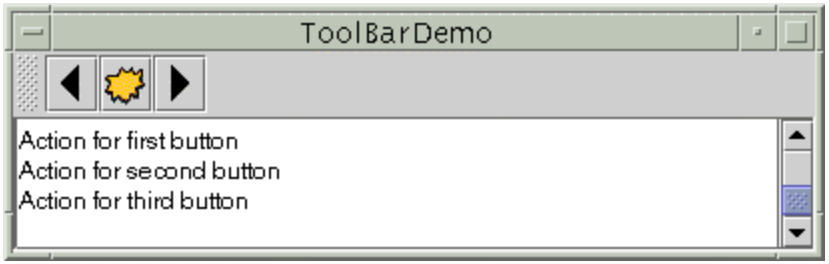
- Containers are used to organize (look and feel) and group (mapping) components
- For example, Swing Application groups the label and button in an **intermediate container** (JPanel) before adding the components to the frame. Here's the code that initializes the panel:

```
// Create a container and define its border (from JPanel)
        JPanel pane = new JPanel();
        pane.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30));
// Creates a layout manager that forces the panel's contents to be displayed in a single column.
        pane.setLayout(new GridLayout(0, 1));
// Add the button and label to the panel.
         pane.add(button);
         pane.add(label);
```

Intermediate Containers

- Swing containers typically used in GUIs are JPanel, JSplitPane, JScrollPane, JTabbedPane, JtoolBar, JDesktopPane, JInternalFrame, JViewport, JTextPane, and JEditorPane.
- You can add other containers and components to these containers, combining components and their containers in various ways to get the interface you want.

Example: A JToolBar is a container that groups several components – usually buttons with icons -- into a row or column.



6- Handling Events

- Every time the user types a character or pushes a mouse button, an event occurs. Any object can be notified of the event.
- All it has to do is implement the appropriate interface and be registered as an event listener on the appropriate event source.

Swing components can generate many kinds of events. Here are a few examples:

Act that results in the event	Listener type
User clicks a button, presses Return while typing in a text field, or chooses a menu item	ActionListene
User closes a frame (main window)	WindowListener
User presses a mouse button while the cursor is over a component	MouseListener

The Swing Application example contains two event handlers.

- One handles button clicks (action events); the other handles window closing (window events).

```
button.addActionListener(new ActionListener()
{
// Increment label and print it
public void actionPerformed(ActionEvent e)
      {numClicks++; label.setText(labelPrefix + numClicks);}
 });

frame.addWindowListener(new WindowAdapter()
{
//close application
public void windowClosing(WindowEvent e) {System.exit(0);}
 });
```