JAVA NOTES

GRAPHICAL USER INTERFACES

Terry Marris 14 June 2001

2 PANELS

2.1 LEARNING OUTCOMES

By the end of this lesson the student should be able to

- create panels with titles and borders
- understand the role of the content pane
- place labels in panels
- place panels in a frame's border regions

2.2 INTRODUCTION

In the last chapter we had our first look at frames. In this chapter we look at panels.

We use panels as containers for GUI components such as buttons and text boxes. Then we place the panels in a frame.

We see how to create panels, complete with a border and a title. We see how to place text, as a label, in a panel.

2.3 CONTENT PANE

In the last chapter we had a first look at frames. No we see how to put GUI components in a frame.

To put GUI components in a *JFrame* we place them in its *content pane*. The content pane is a part of a *JFrame*. A *JFrame* has several layers. The only layer we are interested in at the moment is the content pane.

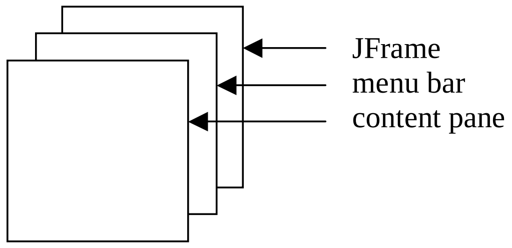


Figure 2.1 *A JFrame has several layers*

A content pane is itself a container. So, to place GUI items in a *JFrame* we place them in its content pane.

The intention is to put some text in a panel, and then put the panel in a frame, so that the result looks like Figure 2.2.

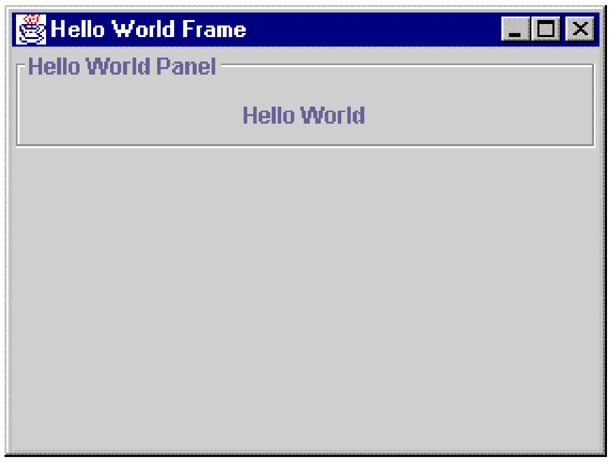


Figure 2.2 *Frame contains a panel with the text Hello World*

2.4 LABELS

A label is a caption; it is just a piece of text like a string. To create a label with the caption *Hello World* is easy; we just supply the quoted text as an argument value to a *JLabel* constructor.

```
     JLabel helloWorldLabel = new JLabel("Hello World");
```

2.5 PANELS

A panel is container for interface elements and can themselves contain other panels.

Just as our frames extend *JFrame*, so our panels extend *JPanel*.

```
class HelloWorldPanel extends JPanel {
  public HelloWorldPanel()
  {
    Border etched = BorderFactory.createEtchedBorder();
    Border titled = BorderFactory.createTitledBorder(
                             etched, "Hello World Panel");
    setBorder(titled);

    JLabel helloWorldLabel = new JLabel("Hello World");
    add(helloWorldLabel);
  }
}
```

We place a border around our panel, and give it the title *Hello World Panel* with

```
     Border etched = BorderFactory.createEtchedBorder();
     Border titled = BorderFactory.createTitledBorder(
                             etched, "Hello World Panel");
     setBorder(titled);
```

First, we create a border with an etched appearance. *Border* is defined in the *javax.swing.border* package. There are other styles of border we can use with panels: *LoweredBevel*, *RaisedBevel* and *Matte* for example.

Then we supply the etched border as an argument value to create a border that is both etched and titled.

Finally, we set the panel's border with *setBorder(titled)*.

2.6 PUTTING PANELS IN FRAMES

To place a panel in a frame is easy. We get the frame's content pane. Then we use the content pane's *add()* method to place the panel in the frame.

```
Container contentPane = getContentPane();

JPanel aPanel = new HelloWorldPanel();
contentPane.add(aPanel, "North");
```

Here, we have specified that the panel is to be placed in the top part of the frame. If we missed *"North"* out, then *"Center"* is assumed (notice American spelling) and the panel then fills all the space in the frame.

2.7 THE HELLO WORLD CLASS

The entire class is shown below. It output is shown in Figure 2.2.

```
/* HelloWorldFrame.java
   Terry Marris  17 June 2001
*/

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

class HelloWorldPanel extends JPanel {
  public HelloWorldPanel()
  {
    Border etched = BorderFactory.createEtchedBorder();
    Border titled = BorderFactory.createTitledBorder(
                             etched, "Hello World Panel");
    setBorder(titled);

    JLabel helloWorldLabel = new JLabel("Hello World");
    add(helloWorldLabel);
  }
}


public class HelloWorldFrame extends JFrame {
  public HelloWorldFrame()
  {
    setTitle("Hello World Frame");

    Toolkit tk = Toolkit.getDefaultToolkit();
    Dimension screen = tk.getScreenSize();
    setLocation(screen.width/4, screen.height/4);
    setSize(screen.width/2, screen.height/2);

    addWindowListener(new WindowAdapter()
      {
        public void windowClosing(WindowEvent e)
        {
          System.exit(0);
        }
     });

    Container contentPane = getContentPane();

    JPanel aPanel = new HelloWorldPanel();
    contentPane.add(aPanel, "North");
  }
  public static void main(String[] s)
  {
    JFrame aFrame = new HelloWorldFrame();
    aFrame.show();
  }
}
```

Exercise: Explain each line of *HelloWorldFrame.java* shown above.

2.8 DOCUMENTATION

JPanel inherits from *JComponent* and *Container*. So some *JPanel* methods may be found defined in *JComponent* and *Container* and in *Container*'s superclass, *Component* (see §1.8).

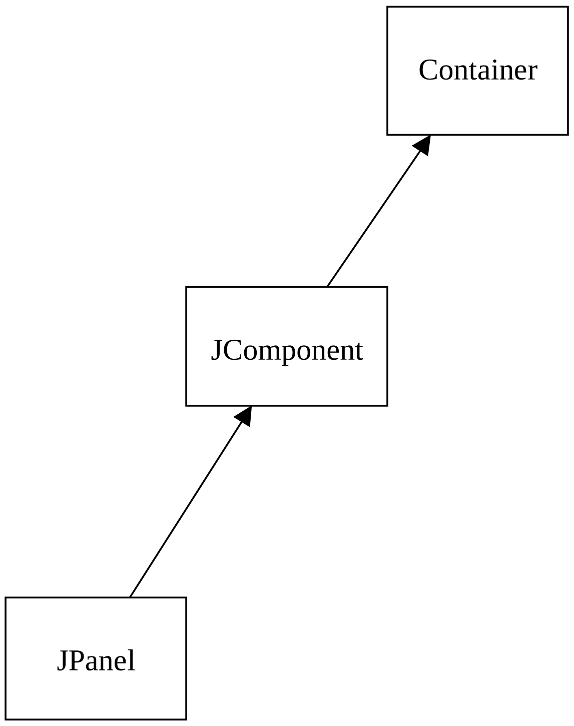


Figure 2.3 *Inheritance Hierarchy for JPanel*

javax.swing.Border

Methods		
Border	JComponent.getBorder()	returns the border for this component or null if no border is currently set.
static Border	BorderFactory.createLineBorder(Color c)	creates a line border with the given colour.
static Border	BorderFactory.createRaisedBevelBorder()	creates a border with a raised bevel edge
static Border	BorderFactory.createLoweredBevelBorder()	creates a border with a lowered bevel edge.
static Border	BorderFactory.createEtchedBorder()	creates a border with an etched look.
static Border	BorderFactory.createEmptyBorder()	creates an empty border that takes up no space.
void	JComponent.setBorder(Border b)	sets the border for this component.
static TitledBorder	BorderFactory.createTitledBorder(Border border)	creates a new titled border with an empty title.
static TitledBorder	BorderFactory.createTitledBorder(Border b, String title)	adds a title to an existing border using the default positioning: top line, left.
static TitledBorder	BorderFactory.createTitledBorder(Border b, String title, int justification, int position)	adds a title to an existing border in the given position.

javax.swing.JLabel

Constructors	
JLabel()	initialises a new label with an empty string for a title.
JLabel(String text)	initialises a new label with the given text for its title. The label is aligned against the left side of its display area, and centre vertically.
JLabel(String text, int horizontalAlignment)	initialises a new label with the given text for its title and the given horizontal alignment. The alignment could is one of the SwingConstants.LEFT, CENTER or RIGHT.

Methods		
String	getText()	returns the text string that this label displays.
void	setHorizontalAlignment(int alignment)	sets the alignment of this label's contents along the x-axis. The alignment could is one of the SwingConstants.LEFT, CENTER or RIGHT.
void	setLabelFor(Component c)	set the component this is labelling.
void	setText(String text)	defines the single line of text this component will display.

javax.swing.JPanel

Constructors	
JPanel()	initialises a new JPanel with flow layout.
JPanel(LayoutManager lm)	initialises a new JPanel with the given layout manager.

javax.swing.JFrame

Constructors	
JFrame()	initialises a new JFrame that is initially invisible.
JFrame(String title)	initialises an invisible frame with the given title.

Methods		
Container	getContentPane()	returns the content pane for this frame.
void	setDefaultCloseOperation(int operation)	sets the operation that will happen by default when the user initiates a close operation on this frame. The possible operations, defined in the WindowsConstants interface, are DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE and DISPOSE_ON_CLOSE.
void	setJMenuBar(JMenuBar mb)	sets the menu bar for this frame.
JMenuBar	getJMenuBar()	returns the menu bar for this frame.

2.9 REVIEW

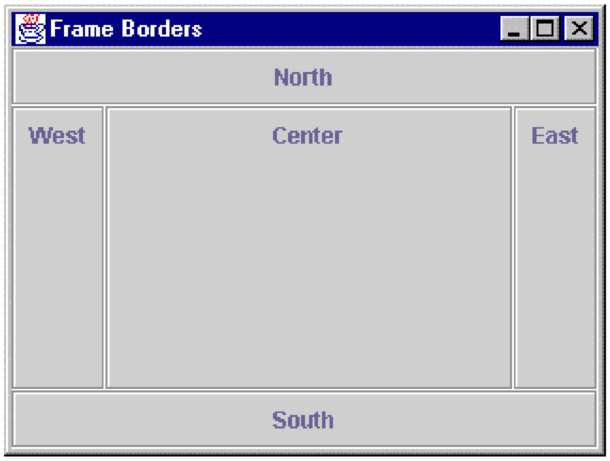
2.10 FURTHER READING

HORSTMANN & CORNELL *Core Java 2 Volume 1*
www.java.sun.com/docs/books/tutorial

2.11 EXERCISES

1 Try out the program shown in §2.7 above.

2 Write a program that produces the following output:


Frame Borders
North
West
Center
East
South