

Object Oriented Programming with Java

COMS M0103
2004

Lecture 14 -- Layout Managers and Swing Text Components

Tim Kovacs
kovacs@cs.bris.ac.uk

Frederik Vercauteren
frederik@cs.bris.ac.uk

Overview

This lecture covers:

- Design patterns and a pattern used with JButtons. Helps demonstrate complexity of Swing classes
- Layout management: how to control where things appear in a window
- Swing components for showing text and letting the user edit text

This material is based on chapter 9 of Core Java 2.

2

1 Design Patterns

Design Pattern = a method of solving a common problem

Defined as:

- a scenario
- a problem
- a solution

Books of software design patterns exist: e.g.

- A system of patterns. Frank Buschmann et al. Wiley & Sons 1996

3

1 Model-View-Controller Design Pattern

Each component has:

- Content (e.g. text data, image data, settings)
- Visual appearance (colour, size etc.)
- Behaviour

Principle: don't make one class do too much

Model-View-Component pattern specifies 3 classes

- Model stores contents
- View displays contents
- Controller handles interactions with user

4

1 Model-View-Controller Examples

Text field example:

- Model: a string: "abcdefghijklmnp"
- View: part of string is visible on-screen: fghijkl

More examples:

- An HTML editor can view document as i) Raw HTML ii) WYSIWYG
- An application can have different Look-and-Feels

Advantage: we can add new views without changing model code.

5

1 MVC and JButton

Design principles like MVC make Swing classes more complex

JButton is really just a wrapper that contains some objects

E.g. for the Metal Look-and-Feel JButton contains:

- a DefaultButtonModel object
- a BasicButtonUI object to implement the view
- a ButtonUIListener object to implement the controller

Usually you can ignore these details and just use JButton
But sometimes you need to look inside Swing components to see how they are implemented

6

2 Introduction to Layout Management

Java vs other languages

- Some others have tools to layout components using mouse (GUI Builders)
- Java does not have a standard tool like this
- Partly because Swing is complex
- But Java layouts can be more robust to look-and-feel changes and window/font resizing

Layout Managers

- Objects that control layout of components
- Adapt layout automatically when e.g. window size changes

7

2 Flow Layout Manager

- Default manager for panels
- Layout: left to right, top to bottom
- By default centers components in each row
- For other alignment make a new object and call `setLayout`:

```
setLayout(new FlowLayout(FlowLayout.LEFT);
```
- Alignment options: LEFT, CENTER, RIGHT
- Horizontal and vertical gaps (in pixels) between components can be added to constructor:

```
FlowLayout(int align, int hgap, int vgap)
```
- Negative gaps force overlaps

8

2 Flow Layout

ButtonTest.java from lecture 13 used the default FlowLayout:



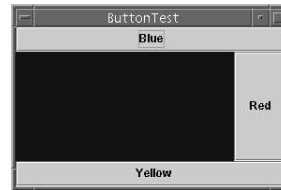
If we add more buttons:



9

2 Border Layout Manager

- Default manager for Frames
- Layout: choose NORTH, SOUTH, EAST, WEST, CENTER



```
BorderLayout()  
BorderLayout(int hgap, vgap)
```

10

2 Border Layout

Modify ButtonTest.java -> ButtonBorderLayoutTest.java

```
class ButtonPanel extends JPanel {  
    public ButtonPanel() {  
        setLayout(new BorderLayout());  
  
        JButton yellowButton = new JButton("Yellow");  
        JButton blueButton = new JButton("Blue");  
        JButton redButton = new JButton("Red");  
  
        // add buttons to panel  
        add(yellowButton, BorderLayout.SOUTH);  
        add(blueButton, BorderLayout.NORTH);  
        add(redButton, BorderLayout.EAST);  
    }  
}
```

11

2 Border Layout Containing a Panel

Notes:

- Only 1 component allowed at each position in BorderLayout
- Button stretches to occupy entire position
- Fix: insert a container to store more components at a position



- Note background of interior JPanel does not change, only background of ButtonPanel

12

2 Border Layout Containing a Panel

Modify ButtonTest.java -> ButtonBorderPanelLayoutTest.java

```
class ButtonPanel extends JPanel {
    public ButtonPanel() {
        setLayout(new BorderLayout());
        JPanel panel = new JPanel();

        // create buttons
        JButton yellowButton = new JButton("Yellow");
        JButton blueButton = new JButton("Blue");
        JButton redButton = new JButton("Red");

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

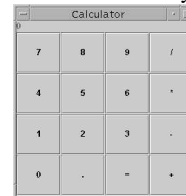
        // add panel to ButtonPanel
        add(panel, BorderLayout.SOUTH);
    }
}
```

13

2 Grid Layout

- Lays out rectangular grid of equally-sized cells
- Specify dimensions in constructor:
`panel.setLayout(new GridLayout(x-size,y-size));`
- Add components left-to-right, top-to-bottom
- Mostly used only for 1-row or 1-column layouts

See Calculator.java



14

2 More Sophisticated Layout

Combining FlowLayout, BorderLayout and Panels is powerful, but sometimes you need more

Box Layout:

- A single row or column of components
- Components stretch as needed
- Each component has:
 - Preferred Size
 - Maximum Size
 - Minimum Size

15

2 More Sophisticated Layout

Grid Bag Layout

- Most flexible manager
- Can be extremely complex
- Basically a grid of components, but:
 - Column and row width can vary
 - Cells can be merged
 - Much more

Spring Layout (Java 1.4)

- Components have springs attached
- Springs pull components into place
- Very flexible, but not as difficult as Grid Bag

16

2 Traversal Order

- User can move between components using tab
- Important for users who cannot use a mouse
- Swing moves left-to-right, top-to-bottom *within each container*

To skip components in traversal:

- Before 1.4
 - override `isFocusTraversable`
- In 1.4
 - `component.setFocusable(false);`

17

3 Text Field

Lets user edit 1 line of text

Adding a Text Field

```
JPanel panel = new JPanel();
JTextField textField = new JTextField("Default input", 20);
panel.add(textField);
```

Text Field Constructors:

```
JTextField(int width);
JTextField(String initialString);
JTextField(String initialString, int width);
```

- Java may adjust width
- Text scrolls automatically -- can irritate user
- Ask for a little more than you want to help avoid scrolling

18

3 Text Field

More API:

- setText(String)
- String getText()

trim() removes spaces from start and end:

```
String text = textField.getText().trim();
```

Numeric values need to be parsed:

```
int year = Integer.parseInt(textField.getText().trim());
```

Validating special text formats (e.g. dates) can be painful

19

3 Text Field Example A

Prints text field to stdout each time contents change

Uses DocumentListener interface from javax.swing.event.*;

```
void insertUpdate(DocumentEvent e)
void removeUpdate(DocumentEvent e)
void changedUpdate(DocumentEvent e)
```

changedUpdate not called for TextFields, but must be implemented.

We register event handler with Document object:

```
myTextField.getDocument().addDocumentListener(listener);
```

not text field object

20

3 Text Field Example A (1/3)

See TextFieldPanel.java

```
import javax.swing.*;
import javax.swing.event.*; // for DocumentEvent
import java.awt.*;

public class TextFieldPanel {
    public static void main(String[] args) {
        myFrame frame = new myFrame();

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.show();
    }
}
```

21

3 Text Field Example A (2/3)

```
/**
 * A frame that contains a panel */

class myFrame extends JFrame {
    public myFrame() {
        setTitle("Frame with Panel with Text Field");
        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

        // add panel to frame
        JPanel panel = new myPanel();
        Container contentPane = getContentPane();
        contentPane.add(panel);
    }

    public static final int DEFAULT_WIDTH = 300;
    public static final int DEFAULT_HEIGHT = 200;
}
```

22

3 Text Field Example A (3/3)

```
// A panel that contains a text field and a listener class.
class myPanel extends JPanel {
    public myPanel() {
        DocumentListener listener = new TextFieldListener();
        myTextField = new JTextField("Initial", 20);
        myTextField.getDocument().addDocumentListener(listener);
        add(myTextField);
    }

    private class TextFieldListener implements DocumentListener {
        public void insertUpdate(DocumentEvent e) {
            System.out.println(myTextField.getText());
        }
        public void removeUpdate(DocumentEvent e) {
            System.out.println(myTextField.getText());
        }
        public void changedUpdate(DocumentEvent e) {}
    }
    private JTextField myTextField;
}
```

23

3 Text Field Example B

When user presses enter/return text field prints to stdout and clears.

See TextFieldPanel2.java

Use ActionEvent interface instead of DocumentEvent interface:

```
import java.awt.event.*; // for ActionEvent
interface
```

instead of

```
import javax.swing.event.*; // for
DocumentEvent
```

Register listener with text field (not document):

```
myTextField.addActionListener(listener);
```

24

3 Text Field Example B

```
...
class myPanel extends JPanel {
    public myPanel() {
        ActionListener listener = new TextFieldListener();
        myTextField = new JTextField("Initial", 20);
        myTextField.addActionListener(listener);
        add(myTextField);
    }

    private class TextFieldListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            System.out.println(myTextField.getText());
            myTextField.setText("");
        }
    }

    private JTextField myTextField;
}
}
```

25

3 Password Fields

Like `TextField`, but all characters display as * (or another symbol)

```
JPasswordField(String initialText, int width)
void setEchoChar(char Echo) // replace * with another char
char[] getPassword()
```

You must overwrite the contents of `char[]` after use

26

3 Formatted Input

SDK 1.4 introduces `JFormattedTextField` for:

- Numbers
- Dates
- Currency
- IP Addresses
- etc.

Use `DocumentFilter` interface to:

- Filter out certain characters (e.g. non-digits)
- Convert all input to upper case

27

3 Text Areas (and Scroll Panes)

Lets user edit many lines

Enter/return key inserts '\n' at end of line

Use `StringTokenizer` to break content into words

Control line clipping with

```
textArea.setLineWrap(true);
```

Add scroll bars by putting text area in a *scroll pane*:

```
textArea = new JTextArea(8,40);
JScrollPane scrollPane = new
JScrollPane(textArea);
contentPane.add(scrollPane,
BorderLayout.CENTER);
```

Scroll bars appear automatically when needed

28

3 TextAreaTest.java (1/4)

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

public class TextAreaTest
{
    public static void main(String[] args)
    {
        TextAreaFrame frame = new TextAreaFrame();

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.show();
    }
}
```



29

3 TextAreaTest.java (2/4)

```
// A frame with a text area and buttons for text editing
class TextAreaFrame extends JFrame {
    public TextAreaFrame() {
        setTitle("TextAreaTest");
        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
        Container contentPane = getContentPane();
        buttonPanel = new JPanel();

        // add button to append text into the text area
        JButton insertButton = new JButton("Insert");
        buttonPanel.add(insertButton);
        insertButton.addActionListener(new

        ActionListener() {
            public void actionPerformed(ActionEvent event){
                textArea.append("The quick brown fox "
                    + "jumps over the lazy dog. ");
            }
        });
    }
}
```

30

3 TextAreaTest.java (3/4)

```
// add button to turn line wrapping on and off
wrapButton = new JButton("Wrap");
buttonPanel.add(wrapButton);
wrapButton.addActionListener(new

    ActionListener() {
        public void actionPerformed(ActionEvent evt){
            boolean wrap = !textArea.getLineWrap();
            textArea.setLineWrap(wrap);
            scrollPane.validate();
            wrapButton.setText(wrap ? "No Wrap" : "Wrap");
        }
    });

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

31

3 TextAreaTest.java (4/4)

```
// add a text area with scroll bars
textArea = new JTextArea(8, 40);
scrollPane = new JScrollPane(textArea);
contentPane.add(scrollPane, BorderLayout.CENTER);
}

public static final int DEFAULT_WIDTH = 300;
public static final int DEFAULT_HEIGHT = 300;

private JTextArea textArea;
private JScrollPane scrollPane;
private JPanel buttonPanel;
private JButton wrapButton;
}
```

32

3 Selecting and Editing Text

API for selected (highlighted) text:

- For both Text field and Text Area
- Inherited from JTextComponent

```
void selectAll()
void select(int startPosition, int endPosition)
int getSelectionStart()
int getSelectionEnd()
String getSelectedText()
void insert(String str, int pos)
void replaceRange(String str, int start, int end)
```

33

3 Labels

- Labels hold text and do not react to user
- Use them to label components with no label (e.g. text field)

```
JLabel(String text)
JLabel(Icon icon)
JLabel(String text, int align)
JLabel(String text, Icon icon, int align)
void setText(String text)
void setIcon(Icon icon)
```

Alignment options are from SwingConstants interface and include: LEFT, RIGHT, CENTER, NORTH ...

```
JLabel label = new JLabel("text", SwingConstants.LEFT);
JLabel label = new JLabel("text", JLabel.LEFT);
```

34

Summary

- The MVC design pattern makes code conceptually cleaner but increases the number of classes involved
- Sometimes you need to look inside Swing classes to make them do what you want
- Flow layout: left to right, top to bottom
- Border layout: North, South, East, West, Center
- Others: Grid, Box, Grid Bag, Spring
- Traversal is left to right, top to bottom within each container
- Text fields: let the user edit one line
- Text Areas: let the user edit many lines
- Labels: show text but do not react to user

35

Test Yourself

- Design Patterns
- JLabel
- Model-View-Controller Design Pattern
- Layout Managers
 - FlowLayout
 - BorderLayout
 - GridLayout
- Traversal Order
- Text Components:
 - JTextField
 - JPasswordField
 - JTextArea
- Wrapper class

36