Swing Containers and Components

Contents

1. Using panes
2. Creating components

Demo project:
DemoSwingContainersComponents

2

1. Using Panes

- Overview of panes
- Using `JPanel`
- Using `JScrollPane`
- Using `JSplitPane`
- Using `JToolBar`
- Using `JTabbedPane`

Overview of Panes

- Panes provide areas of "real estate" on a top level window
  - Each frame has a content pane
- There are several pane classes available
  - `JPanel` is the simplest ☺
- We'll investigate the various panel classes in this section
  - See the `PaneDemo.java` sample code

Using JPanel

- This example shows how to create and use a simple JPanel pane in a frame window

```
public void demoJPanel() {

  // Create a JFrame and a JPanel.
  JFrame  frame  = new JFrame("Frame using JPanel");
  JPanel  pane   = new JPanel();

  // Configure the JPanel as you like.
  pane.setBackground(Color.red);
  pane.setBorder(BorderFactory.createLineBorder(Color.yellow, 3));
  pane.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 10));

  // Add components to the JPanel.
  pane.add(new JTextField(20));
  pane.add(new JButton("A Button"));

  // Add JPanel to "content pane" of Jframe.
  frame.getContentPane().add(pane);

  // Display the frame.
  frame.setSize(300, 200);
  frame.setVisible(true);
}
```

5

Using JScrollPane

- JScrollPane provides a scrollable view of a component
  - Adds scrollable behaviour to components

```
public void demoJScrollPane() {

  JFrame frame = new JFrame("Frame using JScrollPane");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  // Create a JTextArea, and wrap it in a JScrollPane.
  JTextArea   textarea = new JTextArea(10, 30);
  JScrollPane pane     = new JScrollPane(textarea);

  // Add JScrollPane to "content pane" of Jframe.
  frame.getContentPane().add(pane);

  // Display the frame.
  frame.setSize(300, 100);
  frame.setVisible(true);
}
```

Using JSplitPane

- JSplitPane divides two (and only two) components
  - Split horizontally or vertically

```
public void demoJSplitPane() {
  JFrame frame = new JFrame("Frame using JSplitPane");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  // Create two JTextAreas, and wrap each in its own JScrollPane.
  JTextArea   textareaT = new JTextArea(10, 30);
  JTextArea   textareaB = new JTextArea(10, 30);
  JScrollPane paneT     = new JScrollPane(textareaT);
  JScrollPane paneB     = new JScrollPane(textareaB);

  // Add the two JScrollPanes to a JSplitPane.
  JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT, paneT, paneB);

  // Configure the panes in the JSplitPane.
  splitPane.setOneTouchExpandable(true);
  splitPane.setDividerLocation(100);
  paneT.setMinimumSize(new Dimension(100,50));
  paneB.setMinimumSize(new Dimension(100,50));

  // Add JSplitPane to frame, and display.
  frame.getContentPane().add(splitPane);
  frame.setSize(300, 300);
  frame.setVisible(true);
}
```

7

Using JToolBar

- JToolBar groups components into a row or column

```
public void demoJToolBar() {
  JFrame frame = new JFrame("Frame using JToolBar");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  JTextArea   textarea   = new JTextArea(10, 30);
  JScrollPane scrollpane = new JScrollPane(textarea);

  JButton button1 = new JButton("Button1");
  button1.setActionCommand("OPEN");
  button1.setToolTipText("Open a file");

  JButton button2 = new JButton("Button2");
  button2.setActionCommand("CLOSE");
  button2.setToolTipText("Close a file");

  JToolBar toolbar = new JToolBar();   // Create toolbar.
  toolbar.add(button1);                // Add button1.
  toolbar.add(button2);                // And button2.

  JPanel mainpane = new JPanel();
  mainpane.setLayout(new BorderLayout());
  mainpane.add(toolbar, BorderLayout.PAGE_START);
  mainpane.add(scrollpane, BorderLayout.CENTER);

  frame.getContentPane().add(mainpane);
  frame.setSize(300, 300);
  frame.setVisible(true);
}
```

8

Using JTabbedPane

- JTabbedPane displays tabbed panes ☺

```
public void demoJTabbedPane() {
  JFrame frame = new JFrame("Frame using JTabbedPane");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  // Create components for tab1 (wrap in a JPanel).
  JTextField textfield1  = new JTextField(30);
  JPanel panel1 = new JPanel();
  panel1.add(textfield1);

  // Create components for tab2 (wrap in a JPanel).
  JButton button2 = new JButton("A button");
  JPanel panel2 = new JPanel();
  panel2.add(button2);

  // Create a JTabbedPane, and add tabs.
  JTabbedPane tabbedPane = new JTabbedPane();
  tabbedPane.addTab("Tab 1", null, panel1, "Go to tab 1");
  tabbedPane.addTab("Tab 2", null, panel2, "Go to tab 2");

  // Add JTabbedPane to frame, and display.
  frame.getContentPane().add(tabbedPane);
  frame.setSize(300, 300);
  frame.setVisible(true);
}
```

9

2. Creating Components

- Overview of components
- Text fields and text areas
- Push buttons
- Check boxes
- Radio buttons
- Menus
- Other controls

10

Overview of Components

- Swing provides a wide range of components
  - Simple components, such as text boxes and buttons
  - More interesting components, such as progress bars and sliders
- We'll investigate various component classes in this section
  - See the `ComponentDemo.java` sample code

Text Fields and Text Areas

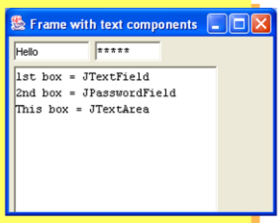
- Useful classes:
  - `JTextField` = Single line of text
  - `JPasswordField` = Single line of text, displays as asterisks
  - `JTextArea` = Multiple lines of text
  - `JEditorPane` = Multiple lines of text in multiple fonts

```
public void demoText() {
  JFrame frame = new JFrame("Frame with text components");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  // Create a JPanel, to hold text components.
  JPanel pane = new JPanel();
  pane.setLayout(new FlowLayout(FlowLayout.LEFT));

  // Add text components to JPanel.
  pane.add(new JTextField(10));
  pane.add(new JPasswordField(10));
  JScrollPane scrollpane = new JScrollPane(new JTextArea(30, 30));
  pane.add(scrollpane);

  // Add JPanel to frame, and display.
  frame.getContentPane().add(pane);
  frame.setSize(300, 200);
  frame.setVisible(true);
}
```



12

Push Buttons (1 of 2)

- Use `JButton` and handle "action" events (see next slide)

```
public void demoPushButtons() {
  JFrame frame = new JFrame("Frame with buttons");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  JButton b1 = new JButton("Hide icons", icon1);
  b1.setVerticalTextPosition(AbstractButton.CENTER);
  b1.setHorizontalTextPosition(AbstractButton.LEADING);
  b1.setMnemonic(KeyEvent.VK_H);
  b1.setActionCommand("hide");
  b1.setToolTipText("Click to hide icons on buttons");

  JButton b2 = new JButton("Show icons", icon2);
  b2.setVerticalTextPosition(AbstractButton.CENTER);
  b2.setHorizontalTextPosition(AbstractButton.TRAILING);
  b2.setMnemonic(KeyEvent.VK_S);
  b2.setActionCommand("show");
  b2.setToolTipText("Click to show icons on buttons");

  PushButtonActionListener listener = new PushButtonActionListener(); // See next slide
  b1.addActionListener(listener);
  b2.addActionListener(listener);

  JPanel pane = new JPanel();
  pane.setLayout(new FlowLayout(FlowLayout.LEFT));
  pane.add(b1);
  pane.add(b2);
  …
```

13

Push Buttons (2 of 2)

- Buttons generate "action" events
  - To handle action events, define a class that implements the `ActionListener` interface
  - Implement `actionPerformed()` to handle the event

```
// Inner class, to handle ActionEvents for the demoPushButtons() method.
class PushButtonActionListener implements ActionListener {

  public void actionPerformed(ActionEvent e) {

    if ("hide".equals(e.getActionCommand())) {
      b1.setIcon(null);
      b2.setIcon(null);

    }
    else if ("show".equals(e.getActionCommand())) {
      b1.setIcon(icon1);
      b2.setIcon(icon2);
    }
  }
}
```

Check Boxes (1 of 2)

- Use `JCheckBox` and listen for "item" events

```
public void demoCheckBoxes() {
  JFrame frame = new JFrame("Frame with check boxes");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  JCheckBox cb1 = new JCheckBox("Ski hire");
  cb1.setMnemonic(KeyEvent.VK_S);
  cb1.setSelected(true);
  cb1.setToolTipText("Do you require ski hire?");

  JCheckBox cb2 = new JCheckBox("Boots hire");
  cb2.setMnemonic(KeyEvent.VK_B);
  cb2.setSelected(true);
  cb2.setToolTipText("Do you require boot hire?");

  CheckBoxItemListener listener = new CheckBoxItemListener();  // See next slide.
  cb1.addItemListener(listener);
  cb2.addItemListener(listener);

  JPanel pane = new JPanel();
  pane.setLayout(new FlowLayout(FlowLayout.LEFT));
  pane.add(cb1);
  pane.add(cb2);
  …
```

Check Boxes (2 of 2)

- Check boxes generate "item" events
  - To handle them, define a class that implements `ItemListener`
  - Implement `itemStateChanged()` to handle the event

```
// Inner class, to handle ItemEvents for the demoCheckBoxes() method.
class CheckBoxItemListener implements ItemListener {

  public void itemStateChanged(ItemEvent e) {

    Object source = e.getItemSelectable();
    String label  = "";

    if (source == cb1) {
      label = cb1.getText();
    }
    else if (source == cb2) {
      label = cb2.getText();
    }

    if (e.getStateChange() == ItemEvent.SELECTED) {
      System.out.println(label + " required");
    }
    else if (e.getStateChange() == ItemEvent.DESELECTED) {
      System.out.println(label + " not required");
    }
  }
}
```

16

Radio Buttons (1 of 2)

- Create group of `JRadioButtons`, await "action" events

```
public void demoRadioButtons() {
  JFrame frame = new JFrame("Frame with radio buttons");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  rb1 = new JRadioButton("Male");
  rb1.setMnemonic(KeyEvent.VK_M);
  rb1.setActionCommand("male");
  rb1.setToolTipText("Are you male?");
  rb1.setSelected(true);

  rb2 = new JRadioButton("Female");
  rb2.setMnemonic(KeyEvent.VK_F);
  rb2.setActionCommand("female");
  rb2.setToolTipText("Are you female?");

  RadioButtonActionListener listener = new RadioButtonActionListener(); // Next slide.
  rb1.addActionListener(listener);
  rb2.addActionListener(listener);

  ButtonGroup group = new ButtonGroup();
  group.add(rb1);
  group.add(rb2);

  JPanel pane = new JPanel();
  pane.setLayout(new FlowLayout(FlowLayout.LEFT));
  pane.add(rb1);
  pane.add(rb2);
  …
```

17

Radio Buttons (2 of 2)

- Handle "action" events:

```
// Inner class, to handle ActionEvents for the demoRadioButtons() method.
class RadioButtonActionListener implements ActionListener {

  public void actionPerformed(ActionEvent e) {

    if ("male".equals(e.getActionCommand())) {
      JOptionPane.showMessageDialog(null,
                                    "Male",
                                    "Item selected",
                                     JOptionPane.PLAIN_MESSAGE);

    } else if ("female".equals(e.getActionCommand())) {
      JOptionPane.showMessageDialog(null,
                                    "Female",
                                    "Item selected",
                                     JOptionPane.PLAIN_MESSAGE);
    }
  }
}
```

Menus

- Use `JMenuBar`, `JMenu`, and `JMenuItem`

```
public void demoMenus() {
  JFrame frame = new JFrame("Frame with menus");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  JMenuBar menuBar = new JMenuBar();
  frame.setJMenuBar(menuBar);

  // Create a "File" menu, and add "Open" and "Close" menu items.
  JMenu file = new JMenu("File");
  file.setMnemonic(KeyEvent.VK_F);
  menuBar.add(file);

  JMenuItem open = new JMenuItem("Open...", KeyEvent.VK_O);
  open.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O, ActionEvent.CTRL_MASK));

  JMenuItem close = new JMenuItem("Close", KeyEvent.VK_C);
  close.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F4, ActionEvent.ALT_MASK));

  file.add(open);
  file.addSeparator();
  file.add(close);

  // Now create an "Edit" menu.
  JMenu edit = new JMenu("Edit");
  edit.setMnemonic(KeyEvent.VK_E);
  menuBar.add(edit);
  …
```

Other Controls

- `JProgressBar`
  - Vertical or horizontal progress indicator
- `ProgressMonitor`
  - Similar to `JProgressBar` (but invisible until a task completes)
- `JSlider`
  - Enables the user to select a numerical value in a specified range
- And more…
  - Lookup `JComponent` in the Swing API documentation, and find the list of known subclasses

20

Any Questions?

21