Swing Layouts and Dialogs

Contents

1. Using layout managers
2. Displaying dialog boxes

Demo project:
DemoSwingLayoutsDialogs

2

1. Using Layout Managers

- Overview of layout managers
- Flow layout
- Border layout
- Grid layout
- Box layout
- Other layouts

Overview of Layout Managers

- Swing uses layout managers to organize the overall arrangement of components in a window
  - You will typically use several layout managers in a window
  - Gives Swing insight into how to "flow" your components
- We'll take a look at the most common layout managers in this section
  - See the `LayoutManagerDemo.java` sample code

4

Flow Layout

- Call `setLayout(new FlowLayout())`

```
public void demoFlowLayout() {
  JFrame frame = new JFrame("FlowLayout Frame");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  JPanel pane = new JPanel();

  // Set the layout manager (and border) for the JPanel.
  pane.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 10));
  pane.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

  // Add components to the JPanel.
  pane.add(new JTextField(20));
  pane.add(new JTextField(20));
  pane.add(new JButton("Button 1"));
  pane.add(new JButton("Button 2"));
  pane.add(new JButton("Button 3"));

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

5

Border Layout

- Call `setLayout(new BorderLayout())`

```
public void demoBorderLayout() {
  JFrame frame = new JFrame("BorderLayout Frame");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  JPanel pane = new JPanel();

  // Set the layout manager (and border) for the JPanel.
  pane.setLayout(new BorderLayout(10, 10));
  pane.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

  // Add components to the JPanel.
  pane.add(new JTextField(20), BorderLayout.NORTH);
  pane.add(new JTextField(20), BorderLayout.SOUTH);
  pane.add(new JButton("Button 1"), BorderLayout.WEST);
  pane.add(new JButton("Button 2"), BorderLayout.CENTER);
  pane.add(new JButton("Button 3"), BorderLayout.EAST);

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

6

Grid Layout

- Call `setLayout(new GridLayout())`

```
public void demoGridLayout() {
  JFrame frame = new JFrame("GridLayout Frame");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  JPanel pane = new JPanel();

  // Set the layout manager (and border) for the JPanel.
  pane.setLayout(new GridLayout(5, 0, 10, 10));  // rows, cols, hgap, vgap.
  pane.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

  // Add components to the JPanel.
  pane.add(new JTextField(20));
  pane.add(new JTextField(20));
  pane.add(new JButton("Button 1"));
  pane.add(new JButton("Button 2"));
  pane.add(new JButton("Button 3"));

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

7

Box Layout

- Very flexible way to organize complex user interfaces
  - Create panels with X-axis or Y-axis `BoxLayout`
  - Then add the panels to a top-level panel

```
public void demoBoxLayout() {
  …
  JPanel textPanel   = new JPanel();     // Panel to hold text components.
  JPanel buttonPanel = new JPanel();     // Panel to hold button components.
  JPanel mainPanel   = new JPanel();     // Contains the 2 other panels.

  textPanel.setLayout(new BoxLayout(textPanel, BoxLayout.Y_AXIS));
  textPanel.add(new JTextField(20));
  textPanel.add(Box.createVerticalStrut(5));
  textPanel.add(new JTextField(20));

  buttonPanel.setLayout(new BoxLayout(buttonPanel, BoxLayout.X_AXIS));
  buttonPanel.add(new JButton("Button 1"));
  buttonPanel.add(Box.createHorizontalStrut(5));
  buttonPanel.add(new JButton("Button 2"));
  buttonPanel.add(Box.createHorizontalStrut(5));
  buttonPanel.add(new JButton("Button 3"));

  mainPanel.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 10));
  mainPanel.add(textPanel);
  mainPanel.add(buttonPanel);
  …
}
```

8

Other Layouts

- `CardLayout`
  - Panels overlap just like in a tabbed pane, but without the tabs
- `DefaultMenuLayout`
  - The layout manager used by menubars and popups
- `ScrollPaneLayout`
  - The default layout manager used by `JScrollPane`
- And more…
  - Lookup `LayoutManager` in the Swing API documentation, and find the list of known subclasses

9

2. Displaying Dialog Boxes

- Overview of dialog boxes
- Creating simple dialog boxes
- Displaying a message dialog box
- Displaying a confirmation dialog box
- Displaying a dialog box with specific options
- Displaying an input dialog box
- Choosing files
- Choosing colours
- Displaying custom dialog boxes

10

Overview of Dialog Boxes

- Most GUI applications need dialog boxes at some point
  - Swing provides standard dialog box classes for common tasks (file open, file save, colour chooser, etc).
  - You can also define custom dialog boxes
- We'll take a look at how to use standard dialogs and custom dialogs in this section
  - See the `DialogBoxDemo.java` sample code

11

Creating Simple Dialog Boxes

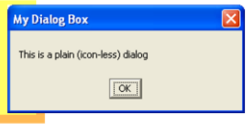
- Use `JOptionPane` to create simple dialog boxes
  - Modal
  - Layout is managed implicitly
  - You can specify text, title, icon, screen location, and parent frame
- Use one of the following class methods in `JOptionPane`
  - `showMessageDialog()` – For a message dialog
  - `showConfirmDialog()` – For a confirmation dialog
  - `showOptionDialog()` – For a dialog with specific options
  - `showInputDialog()` – For a dialog that gets user input

12

Displaying a Message Dialog Box

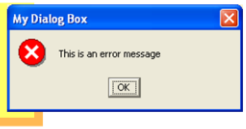
- Call `JOptionPane.showMessageDialog()`
  - See demo function: `displayMessageDialogBoxes()`

```
JOptionPane.showMessageDialog(
        null,  // Parent component
        "This is a plain (icon-less) dialog",
        "My Dialog Box",
        JOptionPane.PLAIN_MESSAGE);
```

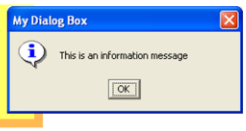



```
JOptionPane.showMessageDialog(
        null,  // Parent component,
        "This is a warning message",
        "My Dialog Box",
        JOptionPane.WARNING_MESSAGE);
```


```
JOptionPane.showMessageDialog(
        null,  // Parent component
        "This is an error message",
        "My Dialog Box",
        JOptionPane.ERROR_MESSAGE);
```




```
JOptionPane.showMessageDialog(
        null,  // Parent component
        "This is an information message",
        "My Dialog Box",
        JOptionPane.INFORMATION_MESSAGE);
```

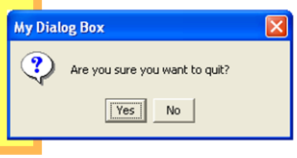


13

Displaying a Confirmation Dialog Box

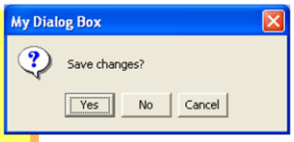
- Call `JOptionPane.showConfirmDialog()`
  - See demo function: `displayConfirmationDialogBoxes()`
  - To display Yes/No buttons:

```
int result;
result = JOptionPane.showConfirmDialog(
           null,  // Parent component
           "Are you sure you want to quit?",
           "My Dialog Box",
           JOptionPane.YES_NO_OPTION);

if (result == JOptionPane.YES_OPTION) { … }
```



  - To display Yes/No/Cancel buttons:

```
int result;
result = JOptionPane.showConfirmDialog(
           null,  // Parent component
           "Save changes?",
           "My Dialog Box",
           JOptionPane.YES_NO_CANCEL_OPTION);

if (result == JOptionPane.YES_OPTION) { … }
```



14

Displaying a Dialog Box with Specific Options

- Call `JOptionPane.showOptionDialog()`
  - Specify what you want for the buttons
  - See demo function: `displayOptionDialogBoxes()`

```
Object[] options = { "Oui", "Non" };

int result;
result = JOptionPane.showOptionDialog(
             null,  // Parent component
             "Parlez-vous français couramment?",
             "Français",
             JOptionPane.YES_NO_OPTION,
             JOptionPane.QUESTION_MESSAGE,
             null,
             options,
             options[0]);

if (result == JOptionPane.YES_OPTION) { … }
```

15

Displaying an Input Dialog Box

- Call `JOptionPane.showInputDialog()`
  - See demo function: `displayInputDialogBoxes()`
  - Allow free-format text entry

```
String s;
s = (String) JOptionPane.showInputDialog(
               null,  // Parent component
               "Please enter your name:",
               "My Dialog Box",
               JOptionPane.PLAIN_MESSAGE);
```

  - Choose from a finite list of allowable options

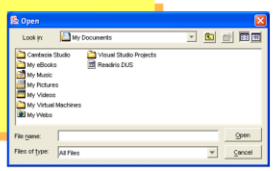
```
String s;
s = (String) JOptionPane.showInputDialog(
               null,  // Parent component,
               "Choose a currency:",
               "My Dialog Box",
               JOptionPane.PLAIN_MESSAGE,
               null,
               new Object[] {"USD", "GBP", "EUR"},
               "USD");
```

16

Choosing Files

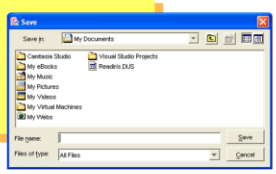
- Create a `JFileChooser` object…
  - See demo function: `displayFileChooserDialogBoxes()`
  - Call `showOpenDialog()` to display a file-open dialog

```
JFileChooser chooser = new JFileChooser();
if (chooser.showOpenDialog(null) ==
                  JFileChooser.APPROVE_OPTION) {
  File file = chooser.getSelectedFile();
  …
}
```



  - Call `showSaveDialog()` to display a file-save dialog

```
JFileChooser chooser = new JFileChooser();
if (chooser.showSaveDialog(null) ==
                  JFileChooser.APPROVE_OPTION) {
  File file = chooser.getSelectedFile();
  …
}
```

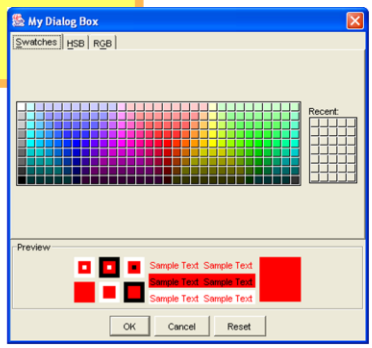


17

Choosing Colours

- Call `JColorChooser.showDialog()`
  - See demo function: `displayColorChooserDialogBoxes()`

```
Color chosenColour;
chosenColour = JColorChooser.showDialog(
                  null,
                  "My Dialog Box",
                  Color.red);
if (chosenColour != null) {
  …
}
```



18

Creating Custom Dialog Boxes

- See demo function: `displayCustomDialogBoxes()`

```
// Create controls to display in the dialog (well, in a JPanel actually).
JLabel      label      = new JLabel("Enter text:");
JTextField  textField  = new JTextField(20);
JButton     button     = new JButton("Done");
JPanel      pane       = new JPanel();

// Add the controls to the JPanel.
pane.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
pane.add(label);
pane.add(textField);
pane.add(button);

// Add the JPanel to a modal JDialog.
JDialog dialog = new JDialog((Frame)null, "My Dialog Box", true);
dialog.getContentPane().add(pane);
dialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
dialog.pack();

// Close the dialog when the user clicks the "Done" button.
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        dialog.setVisible(false);
    }
});

// Display the dialog
dialog.setVisible(true);
```

19

Any Questions?

20