10. GUI Programming :: Event Handling :: An Inner Class Implementation

In Section 9 we implemented a *JButton* event handler class *ButtonListener* as a class independent of the *ButtonDemo3* class.

ActionListener classes tend to be very small so rather than implementing our *JButton* event handler as a standalone class, in this section we will see how to implement it as an inner class. In fact, the code (*ButtonDemo4*) is almost identical to *ButtonDemo3*; all we did was move the *ButtonListener* class from *ButtonListener.java* to the *ButtonDemo4* class.

10. GUI Programming :: Event Handling :: *ButtonDemo4* Example

```
// ButtonDemo4.java
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

/**
 * This application demonstrates how to create an event listener for JButton
 * clicks as an inner class.
 */
public class ButtonDemo4 {
  public static void main(String[] args) { new ButtonDemo4().run(); }

  public void run() {
    // Use the Swing look and feel.
    JFrame.setDefaultLookAndFeelDecorated(true);

    // Create a JLabel which will be added to the JPanel panelBorder later. Set
    // the horizontal alignment of the JLabel so it will be centered in the
    // BorderLayout region. By default, the label displays no text.
    JLabel label = new JLabel("");
    label.setHorizontalAlignment(JLabel.CENTER);
```

10. GUI Programming :: Event Handling :: *ButtonDemo4* Example (continued)

```
// Create a JPanel for the buttons using the horizontal BoxLayout layout
// manager.
JPanel panelButton = new JPanel();
panelButton.setLayout(new BoxLayout(panelButton, BoxLayout.X_AXIS));

// Create a JButton with the text "OK" and make the button 90 pixels wide
// and 30 pixels high.
JButton butOk = newButton("OK", 90, 30);

// Create a ButtonListener object to respond to button clicks on butOk. The
// listener will display the message, "You clicked OK", on the label when
// butOk is clicked.
butOk.addActionListener(new ButtonListener(label, "You clicked OK"));

// Add butOk to the JPanel panelButton, but put some glue before it. The
// goal is to center the two JButtons in the BorderLayout south region.
panelButton.add(Box.createHorizontalGlue());
panelButton.add(butOk);

// Create a rigid area 25 pixels wide between the OK and Cancel buttons.
panelButton.add(Box.createRigidArea(new Dimension(25, 0)));

// Create a JButton with the text "Cancel" and make the button 90 pixels
// wide and 30 pixels high.
JButton butCancel = newButton("Cancel", 90, 30);

// Create a ButtonListener object to respond to button clicks on butCancel.
// The listener will display the message, "You clicked Cancel", on the
// label when butCancel is clicked.
butCancel.addActionListener(new ButtonListener(label, "You clicked Cancel"));
```

10. GUI Programming :: Event Handling :: ButtonDemo4 Example (continued)

```
    // Add butCancel to the JPanel panelButton, but put some glue after it. The
    // glue before butOk and after butCancel will cause these buttons to be
    // centered in the BorderLayout south region.
    panelButton.add(butCancel);
    panelButton.add(Box.createHorizontalGlue());

    // Create a new BorderLayout panel.
    JPanel panelBorder = new JPanel();
    panelBorder.setLayout(new BorderLayout());

    // Add the label to panelBorder in the center region.
    panelBorder.add(label, BorderLayout.CENTER);

    // Add the JPanel panelButton to panelBorder in the south region.
    panelBorder.add(panelButton, BorderLayout.SOUTH);

    // Create the JFrame and add the JPanel panelBorder to it.
    JFrame frame = new JFrame("Button Demo 4");
    frame.setSize(400, 300);
    frame.add(panelBorder);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
}
```

10. GUI Programming :: Event Handling :: ButtonDemo4 Example (continued)

```
  // Create a new JButton displaying the text pText. If pWidth and pHeight are
  // greater than zero, then set the JButton dimensions to pWidth by pHeight.
  private JButton newButton(String pText, int pWidth, int pHeight) {
    JButton button = new JButton(pText);
    button.setPreferredSize(new Dimension(pWidth, pHeight));
    return button;
  }

  private class ButtonListener implements ActionListener {
    private JLabel mLabel;
    private String mText;

    public ButtonListener(JLabel pLabel, String pText) {
      mLabel = pLabel;
      mText = pText;
    }

    // This method is called when the JButton associated with this ActionListener
    // is clicked. We display mText on mLabel.
    public void actionPerformed(ActionEvent pEvent) {
      mLabel.setText(mText);
    }
  }
}
```