9. GUI Programming :: Event Handling :: A *JButton* Event Handler

Users interact with GUI programs by clicking on buttons, entering text in text fields, moving the mouse around the window, and so on. In response, the GUI program must respond to these **events**.

For example, how does the program know when a *JButton* was clicked? In the Java AWT and Swing library, button clicks generate a *java.awt.event.ActionEvent* object. To be notified when a button event occurs, the application must create an **event listener** objects which "listen" for events. This event listener is an instance of a class which implements the *java.awt.event.ActionListener* interface:

```
// ButtonListener.java
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public 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);
  }
}
```

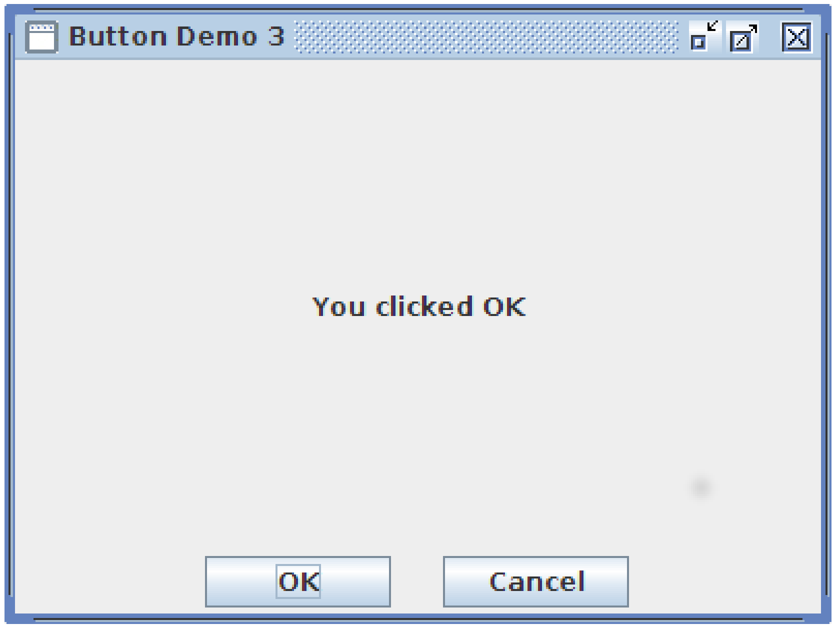
9. GUI Programming :: Event Handling :: A *JButton* Event Handler (continued)

To make a JButton respond to button clicks we next call *addActionListener*() on the button object passing a *ButtonListener* object.

```
// 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"));
...
// 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;
}
```

9. GUI Programming :: Event Handling :: A *JButton* Event Handler (continued)

When the OK button is clicked, the *ButtonListener.actionPerformed*() method is called which will display the text "You clicked OK" on the *JLabel* that was passed to the *ButtonListener* constructor:



You clicked OK

9. GUI Programming :: Event Handling :: *ButtonDemo3* Example

```java
// ButtonDemo3.java
import java.awt.BorderLayout;
import java.awt.Dimension;
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.
 */
public class ButtonDemo3 {
  public static void main(String[] args) { new ButtonDemo3().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);
```

9. GUI Programming :: Event Handling :: *ButtonDemo3* 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"));
```

9. GUI Programming :: Event Handling :: *ButtonDemo3* (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 3");
    frame.setSize(400, 300);
    frame.add(panelBorder);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
```

9. GUI Programming :: Event Handling :: *ButtonDemo3* 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;
  }
}
```

9. GUI Programming :: Event Handling :: *ButtonDemo3* Example (continued)

```
// ButtonListener.java
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JLabel;

/**
 * ButtonListener will listen for JButton clicks. When the button is clicked,
 * the string in mText will be displayed on the JLabel mLabel.
 */
public 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);
  }
}
```