Chapter 9 Sample Programs

```
//********************************************************************
//  GenericWindowListener.java       Author: Lewis and Loftus
//  Represents a generic listener for window components.
//********************************************************************
import java.awt.event.*;
public class GenericWindowListener extends WindowAdapter
{
   //-----------------------------------------------------------------
   //  Terminates the program when the window is closed.
   //-----------------------------------------------------------------
   public void windowClosing (WindowEvent event)
   {
      System.exit(0);
   }
}

//********************************************************************
//  FrameDemo.java       Author: Lewis and Loftus
//  Demonstrates the use of frames.
//********************************************************************
import javax.swing.JFrame;
public class FrameDemo extends JFrame
{
   //-----------------------------------------------------------------
   //  Sets the title and size of an empty frame.
   //-----------------------------------------------------------------
   public FrameDemo ()
   {
      super ("Frame Demonstration");
      setSize (300, 200);
   }
}

//********************************************************************
//  ShowFrames.java       Author: Lewis and Loftus
//  Demonstrates the use of frames.
//********************************************************************
import FrameDemo;
import GenericWindowListener;
import java.awt.event.*;

public class ShowFrames
{
   //-----------------------------------------------------------------
   //  Creates and displays an empty frame.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      FrameDemo frame = new FrameDemo();
      frame.addWindowListener (new GenericWindowListener());
      frame.show();
   }
}
```

```
//********************************************************************
//  LabelDemo.java       Author: Lewis and Loftus
//  Demonstrates the use of labels and image icons.
//********************************************************************
import java.awt.*;
import javax.swing.*;

public class LabelDemo extends JFrame
{
   private ImageIcon icon;
   private JLabel label1, label2, label3;

   //-----------------------------------------------------------------
   //  Sets up the GUI for this frame using three labels.
   //-----------------------------------------------------------------
   public LabelDemo ()
   {
      super ("Label Demonstration");
      setSize (200, 300);

      icon = new ImageIcon ("devil.gif");
      label1 = new JLabel ("Devil Left", icon, SwingConstants.LEFT);
      label2 = new JLabel ("Devil Right", icon, SwingConstants.LEFT);
      label2.setHorizontalTextPosition (SwingConstants.LEFT);
      label3 = new JLabel ("Devil Above", icon, SwingConstants.LEFT);
      label3.setHorizontalTextPosition (SwingConstants.CENTER);
      label3.setVerticalTextPosition (SwingConstants.BOTTOM);

      Container content = getContentPane();
      content.setLayout (new FlowLayout());
      content.add (label1);
      content.add (label2);
      content.add (label3);
   }
}

//********************************************************************
//  Quotes.java       Author: Lewis and Loftus
//  Demonstrates the use of various button types.
//********************************************************************
import QuotesControls;
import GenericWindowListener;
import java.awt.event.*;

public class Quotes
{
   //-----------------------------------------------------------------
   //  Creates and displays the QuotesControls frame.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      QuotesControls frame = new QuotesControls();
      frame.addWindowListener (new GenericWindowListener());
      frame.show();
   }
}
```

```
//********************************************************************
//  QuotesControls.java       Author: Lewis and Loftus
//  Demonstrates the use of various button types.
//********************************************************************
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class QuotesControls extends JFrame
{
   private String[] text;
   private JLabel quote;

   private JButton uppercase, lowercase;
   private JCheckBox bold, italic;
   private JRadioButton comedy, philosophy, carpentry;
   private ButtonGroup topic;
   private JPanel letters, domain, style, app;

   //-----------------------------------------------------------------
   //  Sets up the GUI for the Quotes program.
   //-----------------------------------------------------------------
   public QuotesControls ()
   {
      super ("Quotes");
      setSize (300, 150);

      text = new String[3];
      text[0] = "Take my wife, please.";
      text[1] = "I think, therefore I am.";
      text[2] = "Measure twice, cut once.";

      quote = new JLabel (text[0], SwingConstants.CENTER);
      quote.setFont (new Font ("Serif", Font.PLAIN, 12));

      uppercase = new JButton ("Uppercase");
      lowercase = new JButton ("Lowercase");

      bold = new JCheckBox ("Bold");
      italic = new JCheckBox ("Italic");

      comedy = new JRadioButton ("Comedy", true);
      philosophy = new JRadioButton ("Philosophy", false);
      carpentry = new JRadioButton ("Carpentry", false);

      topic = new ButtonGroup();
      topic.add (comedy);
      topic.add (philosophy);
      topic.add (carpentry);

      QuoteActionListener actionListener = new QuoteActionListener();
      uppercase.addActionListener (actionListener);
      lowercase.addActionListener (actionListener);
      comedy.addActionListener (actionListener);
      philosophy.addActionListener (actionListener);
      carpentry.addActionListener (actionListener);
```

```
    QuoteItemListener itemListener = new QuoteItemListener();
    bold.addItemListener (itemListener);
    italic.addItemListener (itemListener);

    // Organize the components
    letters = new JPanel();
    letters.setLayout (new BoxLayout (letters, BoxLayout.Y_AXIS));
    letters.add (uppercase);
    letters.add (lowercase);

    domain = new JPanel();
    domain.setLayout (new BoxLayout (domain, BoxLayout.Y_AXIS));
    domain.add (comedy);
    domain.add (philosophy);
    domain.add (carpentry);

    style = new JPanel();
    style.setLayout (new BoxLayout (style, BoxLayout.Y_AXIS));
    style.add (bold);
    style.add (italic);

    app = new JPanel();
    app.setLayout (new BorderLayout(15,10));
    app.add (quote, BorderLayout.NORTH);
    app.add (letters, BorderLayout.WEST);
    app.add (domain, BorderLayout.CENTER);
    app.add (style, BorderLayout.EAST);

    setContentPane (app);
  }
  //*****************************************************************
  //  Inner class to handle the uppercase and lowercase buttons, as
  //  well as the topic buttons.
  //*****************************************************************
  private class QuoteActionListener implements ActionListener
  {
    //-----------------------------------------------------------------
    //  Sets the quote to the correct string or the correct case
    //  depending on which button was pressed.
    //-----------------------------------------------------------------
    public void actionPerformed (ActionEvent event)
    {
      Object source = event.getSource();

      if (source == uppercase)
        quote.setText (quote.getText().toUpperCase());
      else if (source == lowercase)
        quote.setText (quote.getText().toLowerCase());
      else if (source == comedy)
        quote.setText (text[0]);
      else if (source == philosophy)
        quote.setText (text[1]);
      else  // must be carpentry
        quote.setText (text[2]);
    }
  }
```

```
   //*****************************************************************
   //  Inner class to handle the check boxes for bold and italic.
   //*****************************************************************
   private class QuoteItemListener implements ItemListener
   {
      //--------------------------------------------------------------
      //  Toggles bold or italic as appropriate.
      //--------------------------------------------------------------
      public void itemStateChanged (ItemEvent event)
      {
         Font font = quote.getFont();
         int style = font.getStyle();

         if (event.getSource() == bold)
         {
            if (event.getStateChange() == ItemEvent.SELECTED)
               style += Font.BOLD;
            else
               style -= Font.BOLD;

            quote.setFont (new Font (font.getName(), style, 12));
         }
         else  // must be italic
         {
            if (event.getStateChange() == ItemEvent.SELECTED)
               style += Font.ITALIC;
            else
               style -= Font.ITALIC;

            quote.setFont (new Font (font.getName(), style, 12));
         }
      }
   }
}

//*****************************************************************
//  JukeBox.java       Author: Lewis and Loftus
//  Demonstrates the use of a combo box.
//*****************************************************************

import JukeBoxControls;
import GenericWindowListener;
import java.awt.event.*;

public class JukeBox
{
   //--------------------------------------------------------------
   //  Creates and displays the JukeBoxControls frame.
   //--------------------------------------------------------------
   public static void main (String[] args)
   {
      JukeBoxControls frame = new JukeBoxControls();
      frame.addWindowListener (new GenericWindowListener());
      frame.show();
   }
}
```

```
//********************************************************************
//  JukeBoxControls.java       Author: Lewis and Loftus
//  Demonstrates the use of a combo box.
//********************************************************************

import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import javax.swing.*;
import java.net.URL;

public class JukeBoxControls extends JFrame
{
   private JLabel titleLabel;
   private JComboBox musicCombo;
   private JButton stopButton, playButton;
   private JPanel appPanel, buttonPanel;
   private AudioClip[] music;
   private AudioClip current;

   //-----------------------------------------------------------------
   //  Sets up the JukeBox GUI.
   //-----------------------------------------------------------------
   public JukeBoxControls()
   {
      super ("Java Juke Box");
      setSize (300, 200);

      try
      {
         music = new AudioClip[7];
         music[0] = null;  // Corresponds to "Make a Selection..."
         music[1] = Applet.newAudioClip (
            new URL("file", "localhost", "westernBeat.wav"));
         music[2] = Applet.newAudioClip (
            new URL("file", "localhost", "classical.wav"));
         music[3] = Applet.newAudioClip (
            new URL("file", "localhost", "jeopardy.au"));
         music[4] = Applet.newAudioClip (
            new URL("file", "localhost", "newAgeRythm.wav"));
         music[5] = Applet.newAudioClip (
            new URL("file", "localhost", "eightiesJam.wav"));
         music[6] = Applet.newAudioClip (
            new URL("file", "localhost", "hitchcock.wav"));
      }
      catch (Exception exception) {}

      titleLabel = new JLabel ("Java Juke Box");
      titleLabel.setAlignmentX (Component.CENTER_ALIGNMENT);

      // Create the list of strings for the combo box options.
      String[] musicNames = {"Make A Selection...", "Western Beat",
            "Classical Melody", "Jeopardy Theme", "New Age Rythm",
            "Eighties Jam", "Alfred Hitchcock's Theme"};
```

```
    musicCombo = new JComboBox (musicNames);
    musicCombo.setAlignmentX (Component.CENTER_ALIGNMENT);

    playButton = new JButton ("Play", new ImageIcon ("play.gif"));
    playButton.setBackground (Color.white);
    stopButton = new JButton ("Stop", new ImageIcon ("stop.gif"));
    stopButton.setBackground (Color.white);

    buttonPanel = new JPanel();
    buttonPanel.setLayout(new BoxLayout(buttonPanel, BoxLayout.X_AXIS));
    buttonPanel.add (playButton);
    buttonPanel.add (Box.createRigidArea (new Dimension(5,0)));
    buttonPanel.add (stopButton);
    buttonPanel.setBackground (Color.cyan);

    appPanel = new JPanel();
    appPanel.setLayout (new BoxLayout(appPanel, BoxLayout.Y_AXIS));
    appPanel.add (titleLabel);
    appPanel.add (Box.createRigidArea (new Dimension(0,5)));
    appPanel.add (musicCombo);
    appPanel.add (Box.createRigidArea (new Dimension(0,5)));
    appPanel.add (buttonPanel);
    appPanel.setBackground (Color.cyan);

    setContentPane (appPanel);

    musicCombo.addActionListener (new JukeBoxListener());
    stopButton.addActionListener (new JukeBoxListener());
    playButton.addActionListener (new JukeBoxListener());
    current = null;
  }
  //*****************************************************************
  //  An inner class to handle the various events of the juke box.
  //*****************************************************************
  private class JukeBoxListener implements ActionListener
  {
    //-----------------------------------------------------------------
    //  Handles the play and stop buttons and the combo box.
    //-----------------------------------------------------------------
    public void actionPerformed (ActionEvent event)
    {
      if (current != null)  // Stop current clip no matter what
        current.stop();

      Object source = event.getSource();
      if (source == playButton)
      {
        if (current != null)
          current.play();
      }
      else if (source == musicCombo)
        current = music[musicCombo.getSelectedIndex()];
    }
  }
}
```