Labs

Lab 9.1: Alternating Two Messages

What is the purpose?

Modify the MoveMessageDemo class on Listing 15.7, pages 499-501 of your book, or write a program to rotate two messages—"Java is fun" and "Java is powerful"—displayed on a panel with a mouse click.

What are the steps?

- Task 1:

  Procedure

  1. Create a MessagePanel class to display a message on a JPanel.
  2. Complete the code shown in Figure 9-1-1 or create your own.

```
// MessagePanel.java: Display a message on a JPanel
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Dimension;
import java.awt.Graphics;
import javax.swing.JPanel;

public class MessagePanel extends JPanel {
  /** The message to be displayed */
  private String message = "Welcome to Java";

  /** The x coordinate where the message is displayed */
  private int xCoordinate = 20;

  /** The y coordinate where the message is displayed */
  private int yCoordinate = 20;

  /** Indicate whether the message is displayed in the center */
  private boolean centered;

  /** The interval for moving the message horizontally and vertically */
  private int interval = 10;

  /** Default constructor */
  public MessagePanel() {
  }

  /** Constructor with a message parameter */
  public MessagePanel(String message) {
    this.message = message;
```

```
    }

    /** Return message */
    public String getMessage() {
      return message;
    }

    /** Set a new message */
    public void setMessage(String message) {
      this.message = message;
      repaint();
    }

    /** Return xCoordinator */
    public int getXCoordinate() {
      return xCoordinate;
    }

    /** Set a new xCoordinator */
    public void setXCoordinate(int x) {
      this.xCoordinate = x;
      repaint();
    }

    /** Return yCoordinator */
    public int getYCoordinate() {
      return yCoordinate;
    }

    /** Set a new yCoordinator */
    public void setYCoordinate(int y) {
      this.yCoordinate = y;
      repaint();
    }

    /** Return centered */
    public boolean isCentered() {
      return centered;
    }

    /** Set a new centered */
    public void setCentered(boolean centered) {
      this.centered = centered;
      repaint();
    }
```

```
/** Return interval */
public int getInterval() {
  return interval;
}

/** Set a new interval */
public void setInterval(int interval) {
  this.interval = interval;
  repaint();
}

/** Paint the message */
protected void paintComponent(Graphics g) {
  super.paintComponent(g);

  if (centered) {
    // Get font metrics for the current font

    // Find the center location to display

    // Get the position of the leftmost character in the baseline

  }

  g.drawString(message, xCoordinate, yCoordinate);
}

/** Move the message left */
public void moveLeft() {
  xCoordinate -= interval;
  repaint();
}

/** Move the message right */
public void moveRight() {
  xCoordinate += interval;
  repaint();
}

/** Move the message up */
public void moveUp() {
  yCoordinate -= interval;
  repaint();
}

/** Move the message down */
```

```
  public void moveDown() {
    yCoordinate -= interval;
    repaint();
  }

  /** Override get method for preferredSize */
  public Dimension getPreferredSize() {
    return new Dimension(200, 30);
  }
}
```

Figure 9-1-1

3. Create a test program called MessageDemo to run the MessagePanel class by an inner class called DisplayPanel.
4. Complete the code shown in Figure 9-1-2 or create your own.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MessageDemo extends JFrame {
  private DisplayPanel panel = new DisplayPanel();

  public MessageDemo() {
    getContentPane().add(panel, BorderLayout.CENTER);
    panel.setFocusable(true);
  }

  /** Main method */
  public static void main(String[] args) {
    JFrame frame = new MessageDemo();
    frame.setTitle("Message Demo");
    frame.setSize(300, 300);
    frame.setVisible(true);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }

  class DisplayPanel extends MessagePanel implements MouseListener {

    public DisplayPanel() {
    }

    public void mousePressed(MouseEvent e) {
    }

    public void mouseReleased(MouseEvent e) {
```

```
        }

        public void mouseClicked(MouseEvent e) {
        }

        public void mouseEntered(MouseEvent e) {
        }

        public void mouseExited(MouseEvent e) {
        }
    }
}
```

Figure 9-1-2

5. Express the messages to a panel in the frame.
6. Compile the java files using the javac command.
7. Execute the MessageDemo class using the java command.
8. Save screenshots of the output similar to Figures 9-1-3 and 9-1-4 and submit them to your instructor.

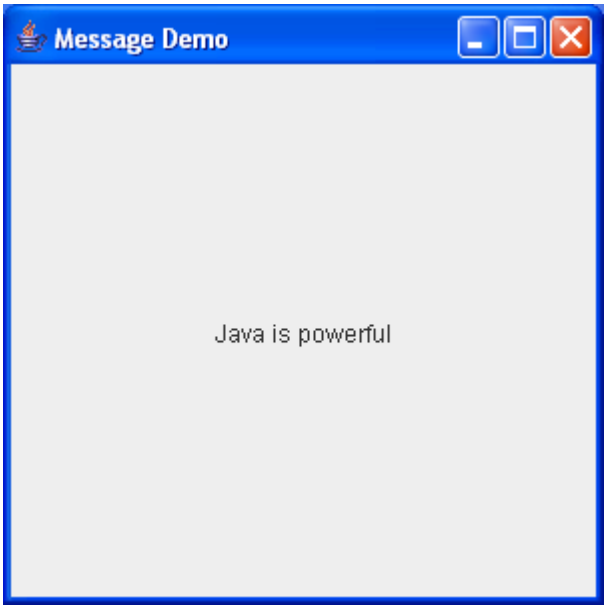

Figure 9-1-3

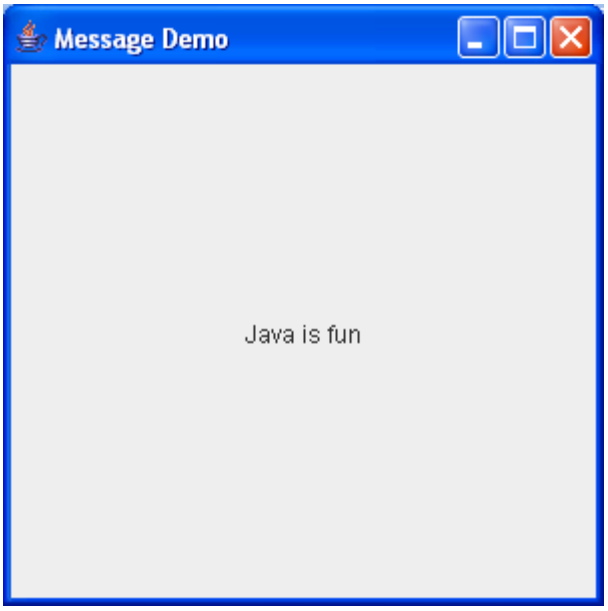

Figure 9-1-4

Did it work?

- Were you able to display two messages—"Java is fun" and "Java is powerful"—on a panel alternately with a mouse click?

Lab 9.2: An Interface that Uses Buttons, Labels, Text Fields, Combo Boxes, Radio Buttons, Check Boxes, and Borders

What is the purpose?

In this lab, you will modify the ButtonDemo class in Listing 16.2, pages 518 and 519 of your book. Improve ButtonDemo program incrementally as follows:

1. Add a text field labeled "Enter a new message" in the same panel with the buttons. When the user types a new message in the text field and presses the ENTER key, the new message is displayed in the message panel.
2. Add a combo box labeled "Select an interval" in the same panel with the buttons. The combo box enables the user to select a new interval for moving the message. The selection values range from 5 to 100 with interval 5. The user can also type a new interval in the combo box.
3. Add three radio buttons that enable the user to select the foreground color for the message as red, green, or blue. Group the radio buttons in a panel, and place the panel in the north of the frame's content pane.
4. Add three check boxes that enable the user to center the message and display it in italic or bold. Place the check boxes in the same panel with the radio buttons.

5. Add a border titled “Message Panel” on the message panel, add a border titled “South Panel” on the panel for buttons, and add a border titled “North Panel” on the panel for radio buttons and check boxes.

What are the steps?

- Task 1:

  Procedure

  1. Modify the ButtonDemo class given on pages 518 and 519.
  2. Complete the code shown in Figure 9-2-1 or create your own.

```
// Demonstrate Simple GUI components and button
import java.awt.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.*;
import javax.swing.border.*;

public class ButtonDemo extends JFrame implements ActionListener {
  // Declare a panel for displaying message
  private MessagePanel messagePanel;

  // Declare two buttons to move the message left and right
  private JButton jbtLeft, jbtRight;

  private JTextField jtfNewMessage = new JTextField(10);
  private JComboBox jcboInterval = new JComboBox();
  private JRadioButton jrbRed = new JRadioButton("Red");
  private JRadioButton jrbGreen = new JRadioButton("Green");
  private JRadioButton jrbBlue = new JRadioButton("Blue");
  private JCheckBox jchkCentered = new JCheckBox("Center");
  private JCheckBox jchkBold = new JCheckBox("Bold");
  private JCheckBox jchkItalic = new JCheckBox("Italic");

  // Font name
  private String fontName = "SansSerif";

  // Font style
  private int fontStyle = Font.PLAIN;

  // Font Size
  private int fontSize = 12;

  /** Main method */
  public static void main(String[] args) {
    ButtonDemo frame = new ButtonDemo();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
    frame.setSize(550, 200);
    frame.setVisible(true);
  }

  /** Default constructor */
  public ButtonDemo() {
    setTitle("User Interface Demo");

    // Create a MessagePanel instance and set colors
    messagePanel = new MessagePanel("Welcome to Java");
    messagePanel.setBackground(Color.white);

    // Create Panel jpButtons to hold two Buttons "<=" and "right =>"
    JPanel jpButtons = new JPanel();
    jpButtons.setLayout(new FlowLayout());
    jpButtons.add(jbtLeft = new JButton());
    jpButtons.add(jbtRight = new JButton());

    // Set button text
    //jbtLeft.setText("<=");
    //jbtRight.setText("=>");

    // Set keyboard mnemonics
    jbtLeft.setMnemonic('L');
    jbtRight.setMnemonic('R');

    // Set icons
    jbtLeft.setIcon(new ImageIcon("image/left.gif"));
    jbtRight.setIcon(new ImageIcon("image/right.gif"));

    // Set toolTipText on the "<=" and "=>" buttons
    jbtLeft.setToolTipText("Move message to left");
    jbtRight.setToolTipText("Move message to right");

    // Place panels in the frame
    getContentPane().setLayout(new BorderLayout());
    getContentPane().add(messagePanel, BorderLayout.CENTER);
    getContentPane().add(jpButtons, BorderLayout.SOUTH);

    // Register listeners with the buttons
    jbtLeft.addActionListener(this);
    jbtRight.addActionListener(this);

    /** 1.Add a text field labeled “New Message.\uFFFD
     *    Upon typing a new message in the text field and pressing the Enter
     *    key, the new message is displayed in the message panel.
```

```
    */


   /** 2. Add a combo box label "Interval\uFFFD that enables the user to select
a
    * new interval for moving the message. The selection values range from
    * 10 to 100 with interval 5. The user can also type a new
    *  interval in the combo box.
    */


   /**
    * 3. Add three radio buttons that enable the user to select the foreground
    * color for the message as Red, Green, and Blue.
    */


   /**
    * 4. Add three check boxes that enable the user to center the message
    * and display it in italic or bold.
    */


   /**
    * 5. Add a border titled "Message Panel\uFFFD on the message panel.
    */

  }

  /** Handle button events */
  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == jbtLeft) {
      messagePanel.moveLeft();
      messagePanel.repaint();
    }
    else if (e.getSource() == jbtRight) {
      messagePanel.moveRight();
      messagePanel.repaint();
    }
    else if (e.getSource() == jtfNewMessage) {
      messagePanel.setMessage(jtfNewMessage.getText());
      messagePanel.repaint();
    }
    else if (e.getSource() == jcboInterval) {
      messagePanel.setInterval(
        Integer.parseInt((String)(jcboInterval.getSelectedItem())));
```

```
            messagePanel.repaint();
          }
          else if (e.getSource() == jrbRed) {
            messagePanel.setForeground(Color.red);
          }
          else if (e.getSource() == jrbGreen) {
            messagePanel.setForeground(Color.green);
          }
          else if (e.getSource() == jrbBlue) {
            messagePanel.setForeground(Color.blue);
          }
          else if (e.getSource() == jchkCentered) {
            if (jchkCentered.isSelected())
              messagePanel.setCentered(true);
            else
              messagePanel.setCentered(false);

            messagePanel.repaint();
          }
          else if ((e.getSource() == jchkBold) ||
                (e.getSource() == jchkItalic)) {
            fontStyle = Font.PLAIN;

            // Determine a font style
            if (jchkBold.isSelected())
              fontStyle = fontStyle + Font.BOLD;
            if (jchkItalic.isSelected())
              fontStyle = fontStyle + Font.ITALIC;

            // Set font for the message
            messagePanel.setFont(new Font(fontName, fontStyle, fontSize));
          }
        }
      }
```

Figure 9-2-1

3. Compile the java files using the javac command.
4. Execute the ButtonDemo class using the java command.
5. Save a screenshots of the output similar to the ones shown in Figures 9-2-2 through 9-2-4 and submit them to your instructor.

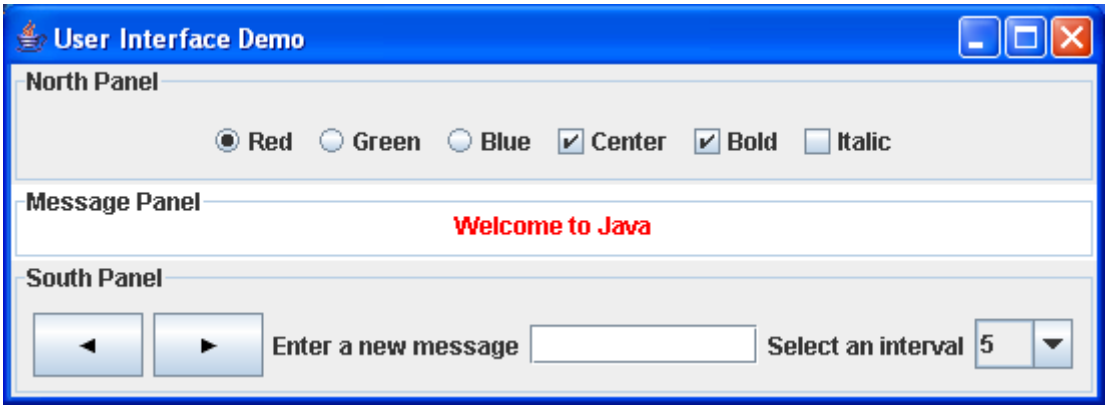

Figure 9-2-2

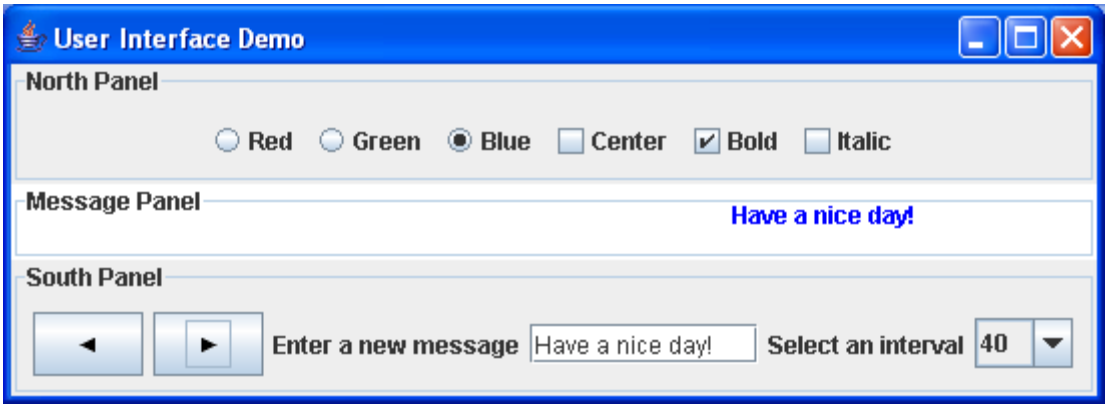

Figure 9-2-3

Figure 9-2-4

Did it work?

Were you able to—

- Display a new message on the message panel with a text field?
- Control the message panel with a tab interval from 5 to 100 with a combo box?
- Control the foreground color of the message panel to be red, green, or blue with a set of radio buttons?

- Control the font of the new message panel as plain, bold or italic with a set of check boxes?
- Display a border title “Message Panel” on the message panel and a border title “South Panel” on the bottom panel?