Lập trình hướng đối tượng

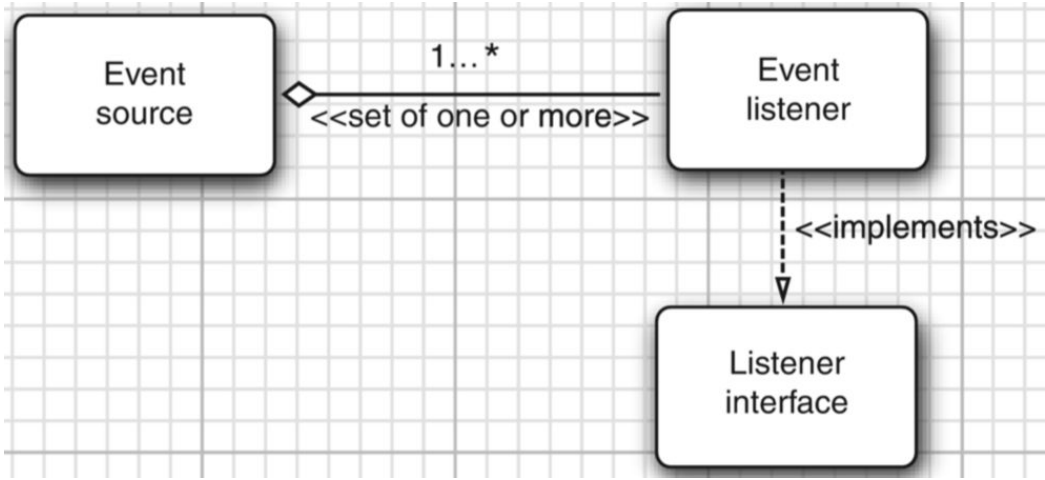
Biên soạn: Nguyễn Thị Tuyết Hải

Email: tuyethai@ptithcm.edu.vn

Chapter 11: Event Handling

11.1 Basics of Event Handling

- An event source is an object that
  - register listener objects
  - sends out event objects to all registered listeners when that event occurs.
- The listener objects decide their reaction to the event.



11.1 Basics of Event Handling

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

public class ButtonFrame extends JFrame{
   private JPanel buttonPanel;
   private static final int DEFAULT_WIDTH = 300;
   private static final int DEFAULT_HEIGHT = 200;

   public ButtonFrame(){
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

      JButton yellowButton = new JButton("Yellow");
      JButton blueButton = new JButton("Blue");
      JButton redButton = new JButton("Red");

      buttonPanel = new JPanel();
      buttonPanel.add(yellowButton); buttonPanel.add(blueButton);
      buttonPanel.add(redButton);
      add(buttonPanel);  // add panel to frame

      // create button actions
      ColorAction yellowAction = new ColorAction(Color.YELLOW);
      ColorAction blueAction = new ColorAction(Color.BLUE);
      ColorAction redAction = new ColorAction(Color.RED);

      // associate actions with buttons
      yellowButton.addActionListener(yellowAction);
      blueButton.addActionListener(blueAction);
      redButton.addActionListener(redAction);
   }
```

```
   /**
    * An action listener that sets the panel's background color
    */
   private class ColorAction implements ActionListener
   {
      private Color backgroundColor;

      public ColorAction(Color c)
      {
         backgroundColor = c;
      }

      public void actionPerformed(ActionEvent event)
      {
         buttonPanel.setBackground(backgroundColor);
      }
   }
}
```

11.1.2 Specifying Listeners Concisely

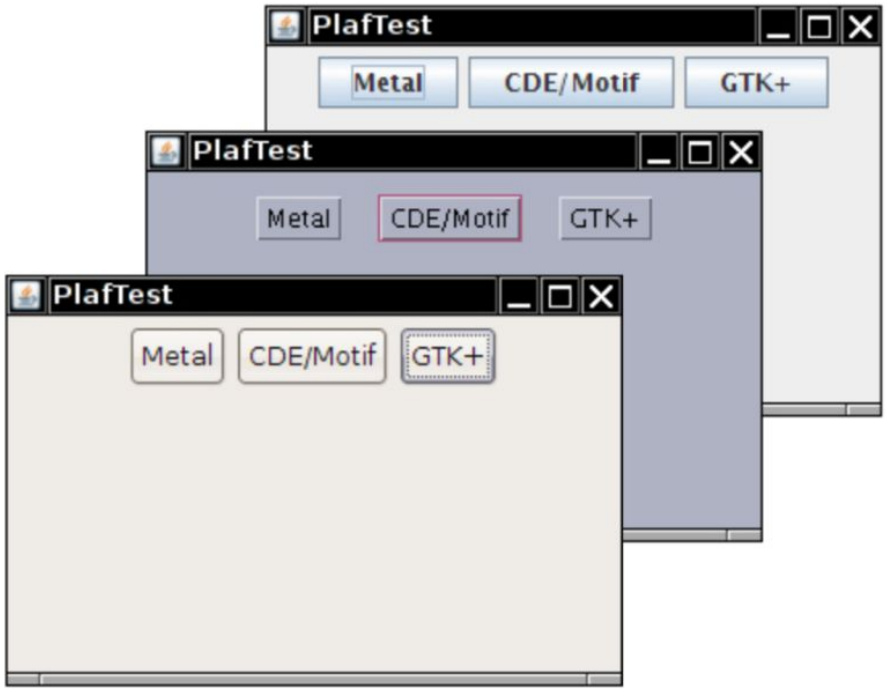
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ButtonFrameLambda extends JFrame{
   private JPanel buttonPanel;
   private static final int DEFAULT_WIDTH = 300;
   private static final int DEFAULT_HEIGHT = 200;

   public ButtonFrameLambda(){
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

      // create buttons
      JButton yellowButton = new JButton("Yellow");
      JButton blueButton = new JButton("Blue");
      JButton redButton = new JButton("Red");

      buttonPanel = new JPanel();
      // add buttons to panel
      buttonPanel.add(yellowButton);
      buttonPanel.add(blueButton);
      buttonPanel.add(redButton);

      // add panel to frame
      add(buttonPanel);

      // associate actions with buttons
      yellowButton.addActionListener(event -> buttonPanel.setBackground(Color.YELLOW));
      blueButton.addActionListener(event -> buttonPanel.setBackground(Color.BLUE));
      redButton.addActionListener(event -> buttonPanel.setBackground(Color.RED));
   }
}
```

11.1.3 Example: Changing the Look-and-Feel

Swing programs use the Metal look-and-feel, two ways to change to a different look-and-feel:

- jre/lib/swing.properties
  swing.defaultlaf=com.sun.java.swing.plaf.motif.MotifLookAndFeel
- UIManager.setLookAndFeel

11.1.3 Example: Changing the Look-and-Feel




```
import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JPanel
import javax.swing.SwingUtilities; import javax.swing.UIManager;

public class PlafFrame extends JFrame{
   private JPanel buttonPanel;

   public PlafFrame(){
      buttonPanel = new JPanel();

      UIManager.LookAndFeelInfo[] infos = UIManager.getInstalledLookAndFeels();
      for (UIManager.LookAndFeelInfo info : infos)
         makeButton(info.getName(), info.getClassName());

      add(buttonPanel);
      pack();
   }

   private void makeButton(String name, String className){
      // add button to panel
      JButton button = new JButton(name);
      buttonPanel.add(button);
      // set button action
      button.addActionListener(event -> {
         try{
            UIManager.setLookAndFeel(className);
            SwingUtilities.updateComponentTreeUI(this);
            pack();
         }catch (Exception e){
            e.printStackTrace();
         }
      });
   }
}
```

11.1.4 Adapter Classes

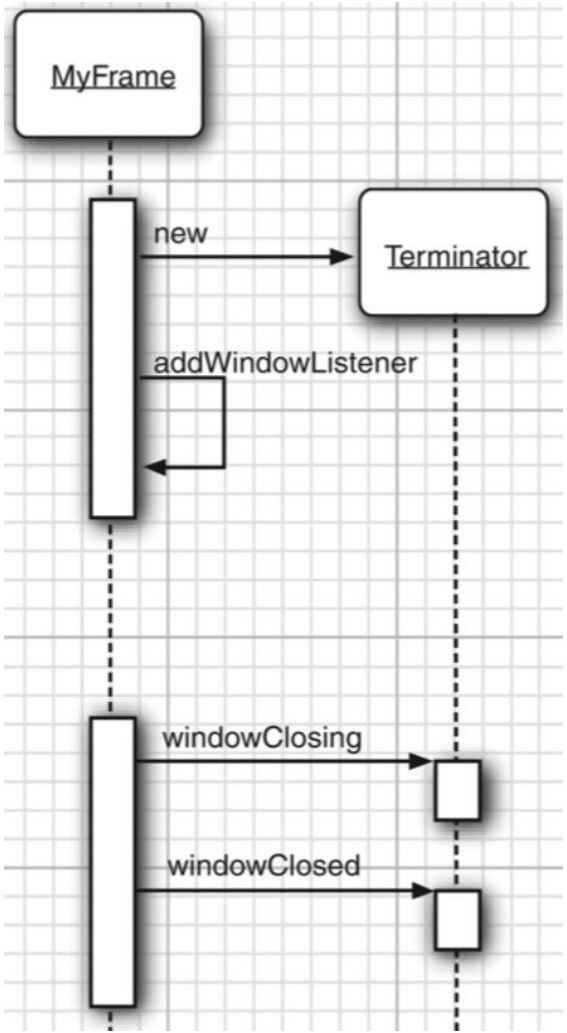
- When the user tries to close a window, the JFrame object is the source of a WindowEvent.
- To catch that event, you must have an appropriate listener object and add it to the frame's list of window listeners.

```
WindowListener listener = . . .;
frame.addWindowListener(listener);

public interface WindowListener {
        void windowOpened(WindowEvent e); // only need one but override all methods in listener
        void windowClosing(WindowEvent e);
        void windowClosed(WindowEvent e);
        void windowIconified(WindowEvent e);
        void windowDeiconified(WindowEvent e);
        void windowActivated(WindowEvent e);
        void windowDeactivated(WindowEvent e);
}
```

11.1.4 Adapter Classes

- each interface that have more than one method comes with a companion adapter class that implements all the methods in the interface but does nothing with them.

```
class Terminator extends WindowAdapter {
        public void windowClosing(WindowEvent e) {
                if(user agrees) System.exit(0);
        }
}

// register listener to frame
WindowListener listener = new Terminator();
frame.addWindowListener(listener);
```



11.2 Actions

An action is an object that encapsulates

- A description of the command
- Parameters that are necessary to carry out the command

The Action interface has the following methods:

- void actionPerformed(ActionEvent event)
- void setEnabled(boolean b)
- boolean isEnabled()
- void putValue(String key, Object value)
- Object getValue(String key)
- void addPropertyChangeListener(PropertyChangeListener listener)
- void removePropertyChangeListener(PropertyChangeListener listener)

11.2 Actions

- Any class implementing this Action interface must implement the seven methods we just discussed.
- AbstractAction class that implements all methods except for actionPerformed.
- What you do to carry out the same action in response to a button, a menu item, or a keystroke:
    a. Implement a class that extends the AbstractAction class.
    b. Construct an object of the action class.
    c. Construct a button or menu item from the action object. The constructor will read the label text and icon from the action object.

11.2 Actions

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

public class ActionFrame extends JFrame{
   private JPanel buttonPanel;
   private static final int DEFAULT_WIDTH = 300;
   private static final int DEFAULT_HEIGHT = 200;

   public ActionFrame(){
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

      buttonPanel = new JPanel();

      // define actions
      Action yellowAction = new ColorAction("Yellow", new ImageIcon("yellow-ball.gif"), Color.YELLOW);
      Action blueAction = new ColorAction("Blue", new ImageIcon("blue-ball.gif"),  Color.BLUE);
      Action redAction = new ColorAction("Red", new ImageIcon("red-ball.gif"), Color.RED);

      // add buttons for these actions
      buttonPanel.add(new JButton(yellowAction));
      buttonPanel.add(new JButton(blueAction));
      buttonPanel.add(new JButton(redAction));

      // add panel to frame
      add(buttonPanel);
      }
```

11.2 Actions

```
public class ColorAction extends AbstractAction{
      public ColorAction(String name, Icon icon, Color c){
         putValue(Action.NAME, name);
         putValue(Action.SMALL_ICON, icon);
         putValue(Action.SHORT_DESCRIPTION, "Set panel color to " + name.toLowerCase());
         putValue("color", c);
      }

      public void actionPerformed(ActionEvent event){
         Color c = (Color) getValue("color");
         buttonPanel.setBackground(c);
      }
   }
}
```

11.4 The AWT Event Hierarchy

The interfaces listen to these common events:

- ActionListener
- AdjustmentListener
- ItemListener

- FocusListener (FocusAdapter)
- WindowListener (WindowAdapter)
- WindowFocusListener
- WindowStateListener

- KeyListener (KeyAdapter)
- MouseListener (MouseAdapter)
- MouseMotionListener (MouseMotionAdapter)
- MouseWheelListener