Object Oriented Programming and Design in Java

Session 8
Instructor: Bert Huang
Announcements

• Example code and slides on homepage in Schedule section
• Homework 1 example posted
• Homework 2 due Mar. 3rd, 11 AM
• Midterm exam Mar. 10th
Review

- Introduction to **programming patterns**
- Patterns in GUI programming and examples
  - Model/View/Controller - GUIs
  - Observer - ActionListener
  - Composite - JPanel
  - Decorator - JScrollPane
  - Strategy - Layout Managers
Today's Plan

• More Swing components
  • JTextArea, JSplitPane
• Listeners in Swing
  • Change listener
  • Focus listener
• Mouse listeners
JTextArea

• Arbitrarily large, editable text field
  • void setEditable(boolean)
    // sets whether the user can edit text
  • void setText(String)
    // sets the text
  • void append(String)
  • void insert(String, int)
JSplitPane

- Not a Composite nor a Decorator
- JSplitPane contains two JPanels, placed horizontally or vertically

![Diagram of JSplitPane with two JPanels and a draggable divider]

- Provides a draggable divider
JSplitPane Methods

- JSplitPane(int Orientation, Component, Component)
  // JSplitPane.VERTICAL_SPLIT, HORIZONTAL_SPLIT

- void setOneTouchExpandable(boolean)
  // Adds one-click expand arrows

- setTopComponent(Component)
  // also Bottom, Right, Left

- setDividerLocation(double)
  // moves divider
JSplitPane Example

```java
JTextArea left = new JTextArea();
for (int i=0; i<20; i++)
    left.append("Hello, World!\n");

JPanel right = new JPanel();
right.setLayout(new GridLayout(0,4));
for (int i=0; i<20; i++)
    right.add(new JButton("Button " +i));

JSplitPane splitPane =
    new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
                   new JScrollPane(left),
                   new JScrollPane(right));
```
Observers in MVC

- Swing provides many Listeners for standard JComponents
- so the View can notify the Controller
Listener Interfaces for Simple Controllers

• Trick if your program is very simple

• You can have your Controller class implement Listener interfaces

• Then you add the Controller directly to your components

• But this doesn't work if your Controller has more than one action
ActionListener

- A single method,
- `void actionPerformed(ActionEvent a)`
- Makes sense for components with clear actions
  - Clicking buttons
  - hitting "enter" in text fields
EventObject

- All Listener methods get a single argument, which must inherit from EventObject
- Object getSource()
  - e.g., getSource() and then getText()
  - Need to cast to (JTextField)
Types of Events

• Events are either *low-level* events or *semantic* events

• Low-level: window-system, raw input
e.g., click, key presses

• Semantic: item events, actions
e.g., button, timer tick, text field entry

• Try to listen for semantic events when possible. Let Swing take care of low-level
ChangeListener

- Provides method
  - stateChanged(ChangeEvent event)
- ChangeEvent just an EventObject
- Object getSource()

- JSlider source = (JSlider)event.getSource();
  Double x = source.getValue();
ChangeListener Example
public static void main(String[] args) {
    JProgressBar myBar = new JProgressBar();
    myBar.setValue(50);

    JSlider mySlider = new JSlider();
    mySlider.addChangeListener(new MyListener(myBar));

    JFrame frame = new JFrame("ChangeListener Tester");
    frame.setLayout(new FlowLayout());
    frame.add(myBar);
    frame.add(mySlider);
    frame.pack();
    frame.setVisible(true);
}
ChangeListener Example

```java
public static class MyListener implements ChangeListener {

    public MyListener(JProgressBar bar) {
        myProgressBar = bar;
    }

    public void stateChanged(ChangeEvent event) {
        JSlider mySlider = (JSlider) event.getSource();
        myProgressBar.setValue(mySlider.getValue());
    }

    private JProgressBar myProgressBar;
}
```
FocusListener

- Triggers events when component gains or loses keyboard focus
  - focusGained(FocusEvent event)
  - focusLost(FocusEvent event)
- FocusEvent provides getComponent() and
  - boolean isTemporary()
- Component getOppositeComponent()
MouseListener

- Listens for any low-level mouse activity
- Provides five methods:
  - mouseClicked(MouseEvent event)
  - mouseEntered(MouseEvent event)
  - mouseExited(MouseEvent event)
  - mousePressed(MouseEvent event)
  - mouseReleased(MouseEvent event)
MouseEvent

- int getX(); int getY(); Point getPoint()
- int getButton()
- int getClickCount()
- String getMouseModifiersText(int)
- boolean isPopupTrigger()
- int getWhen() // inherited
Event Adapters

- Usually, you don't need all five methods in MouseListener
- But to implement interface, you must implement all five (some empty)
- Instead, Swing provides adapter classes that implement all with empty
- Extend adapter class and only implement methods you need
MouseAdapter Example
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class MousePanelTester {
    public static void main(String [] args) {
        MousePanel panel = new MousePanel();
        panel.addMouseListener(new MyMouseListener(panel));

        JFrame frame = new JFrame("MousePanel Tester");
        frame.setLayout(new FlowLayout());
        frame.add(panel);
        frame.pack();
        frame.setVisible(true);
    }

    private static class MyMouseListener extends MouseAdapter {
        private MousePanel myPanel;

        public MyMouseListener(MousePanel panel) {
            super();
            myPanel = panel;
        }

        public void mouseClicked(MouseEvent event) {
            myPanel.setLoc(event.getX(), event.getY());
        }
    }
}
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class MousePanelTester {
    public static void main(String[] args) {
        MousePanel panel = new MousePanel();
        panel.addMouseListener(new MyMouseListener(panel));

        JFrame frame = new JFrame("MousePanel Tester");
        frame.setLayout(new FlowLayout());
        frame.add(panel);
        frame.pack();
        frame.setVisible(true);
    }

    private static class MyMouseListener extends MouseAdapter {
        private MousePanel myPanel;

        public MyMouseListener(MousePanel panel) {
            super();
            myPanel = panel;
        }

        public void mouseClicked(MouseEvent event) {
            myPanel.setLoc(event.getX(), event.getY());
        }
    }
}
import java.awt.*;
import java.awt.geom.*;
import javax.swing.*;

/**
 * MousePanel draws a mouse
 *
 @author 1007
 */

public class MousePanel extends JPanel {

    public MousePanel() {
        super();
        this.setPreferredSize(new Dimension(DEFAULT_SIZE, DEFAULT_SIZE));
        location = new Point(DEFAULT_SIZE / 2, DEFAULT_SIZE / 2);
    }

    public void setLoc(int x, int y) {
        location.setLocation(x, y);
        this.repaint();
    }

    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
        double x = location.getX();
        double y = location.getY();

        Ellipse2D.Double head = new Ellipse2D.Double(x - RADIUS, y - RADIUS, 2 * RADIUS, 2 * RADIUS);
        Ellipse2D.Double rightEar = new Ellipse2D.Double(x - OFFSET, y - OFFSET, RADIUS, RADIUS);
        Ellipse2D.Double leftEar = new Ellipse2D.Double(x + OFFSET - RADIUS, y - OFFSET, RADIUS, RADIUS);

        g2.fill(head);
        g2.fill(leftEar);
        g2.fill(rightEar);
    }

    private int DEFAULT_SIZE = 300;
    private double RADIUS = 15;
    private double OFFSET = 22;
    private Point location;
}
import java.awt.*;
import java.awt.geom.*;
import javax.swing.*;

/**
 * MousePanel draws a mouse
 *
 * @author 1007
 */
public class MousePanel extends JPanel {

    public MousePanel() {
        super();
        this.setPreferredSize(new Dimension(DEFAULT_SIZE, DEFAULT_SIZE));
        location = new Point(DEFAULT_SIZE / 2, DEFAULT_SIZE / 2);
    }

    /**
     * Sets the location of the mouse drawing
     * @param x location of mouse drawing
     * @param y location of mouse drawing
     */
    public void setLoc(int x, int y) {
        location.setLocation(x, y);
        this.repaint();
    }

    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;

        double x = location.getX();
        double y = location.getY();

        Ellipse2D.Double head =
                new Ellipse2D.Double(x - RADIUS, y - RADIUS,
                        2 * RADIUS, 2 * RADIUS);

        g2.fill(head);
        g2.fill(leftEar);
        g2.fill(rightEar);
    }

    private int DEFAULT_SIZE = 300;
    private double RADIUS = 15;
    private double OFFSET = 22;
    private Point location;
}
** MousePanel draws a mouse  

* @author 1007  

```java
public class MousePanel extends JPanel {

    public MousePanel() {
        super();
        this.setPreferredSize(new Dimension(DEFAULT_SIZE, DEFAULT_SIZE));
        location = new Point(DEFAULT_SIZE / 2, DEFAULT_SIZE / 2);
    }

    public void setLoc(int x, int y) {
        location.setLocation(x, y);
        this.repaint();
    }

    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
        double x = location.getX();
        double y = location.getY();
        Ellipse2D.Double head =
            new Ellipse2D.Double(x - RADIUS, y - RADIUS,
                                 2 * RADIUS, 2 * RADIUS);
        Ellipse2D.Double rightEar =
            new Ellipse2D.Double(x - OFFSET, y - OFFSET,
                                 RADIUS, RADIUS);
        Ellipse2D.Double leftEar =
            new Ellipse2D.Double(x + OFFSET - RADIUS,
                                 y - OFFSET, RADIUS, RADIUS);

        g2.fill(head);
        g2.fill(leftEar);
        g2.fill(rightEar);
    }

    private int DEFAULT_SIZE = 300;
    private double RADIUS = 15;
    private double OFFSET = 22;
    private Point location;
```
Reading

- Today's material is loosely based on official Java tutorials
  - [http://java.sun.com/docs/books/tutorial/uiswing/events/index.html](http://java.sun.com/docs/books/tutorial/uiswing/events/index.html)
  - Use these and the API for hw reference