CS324e - Elements of Graphics and Visualization

Java GUIs - Event Handling
Event Driven Programming

• A "Programming Paradigm"
  – others: object-oriented, functional, data-flow, procedural, and more!

• Most early programs we write:
  – get data
  – perform computations
  – output results
  – CRUD programming (Create, Read, Update, Delete)

• That's not how most programs we use actually behave.
Event Driven Programming
GUls and Events

• Most programs sit there and wait for the user to respond
• Flow of control is based on user actions
• User action is an *event* that the program responds to
• Different languages have different levels of support for doing event driven programming
Events Handling

- High level approach:
  - fixes set of events and can attach code to the event: android:onclick

- Low level approach
  - must write code to check if events have occurred and deal with them in other code
  - Big old switch statement
Java Event Handling

• Java is in between the high level and low level approaches

• Built in GUI components in Swing:
  – buttons, check box, combo box, lists, menus, radio buttons, sliders, spinners, text fields, password text fields, labels, trees, color chooser, file chooser, separators, progress bars, trees, tables, and more

http://docs.oracle.com/javase/tutorial/ui/features/components.html
Java Event Handling

• These built in components can be added to top level containers such as frames (menus) and panels
  – Position is handled via a layout manager
  – initially we will use default layout manager FlowLayout
    • components added one after another in a line

• Components are drawn and generate events
New Sample Program

• Program with buttons
  – background color changes when button pressed
• Main program -> frame -> panel
• Panel has an instance variable currentColor
• When paint component called, background set to currentColor
• demo
Add Buttons

• Add Buttons to the panel

class EventExamplePanel extends JPanel {

    private Color currentColor;

    private static String[] buttonNames = {"Red", "Green", "Blue"};

    private JButton[] buttons;

    public EventExamplePanel() {
        currentColor = Color.ORANGE;
        buttons = new JButton[buttonNames.length];

        for (int i = 0; i < buttonNames.length; i++) {
            buttons[i] = new JButton(buttonNames[i]);
            add(buttons[i]);
        }
    }
}
Result of Adding Buttons

• Notice order of buttons
• What happens if change order of names?
• What happens if add more buttons?
• What happens if resize Frame?
• What happens if Button pressed?
Listeners

• When the buttons are pressed events are being generated, but *no one is listening*

• In other words we don't have any code that responds to the events

• We need to create listeners for each button to listen for the event and respond by changing background color
ActionListener

• LOTS of kinds of listeners
• All extend or implement the EventListener interface
  • http://docs.oracle.com/javase/7/docs/api/java/util/EventListener.html
• We will create a class that implements the ActionListener interface

```
java.awt.event

Interface ActionListener

All Superinterfaces:
  EventListener

All Known Subinterfaces:

Method Detail

actionPerfomed

void actionPerformed(ActionEvent e)
Invoked when an action occurs.
```
Try a Separate Class

• Create a ColorAction class
  – instance vars
  – constructor
  – actionPerformed method

• repaint -> request an entire component be repainted. Don't call paintComponent

• array of colors

• build ColorAction and attach to each button
class ColorAction implements ActionListener {
    private EventExamplePanel panel;
    private Color color;

    public ColorAction(EventExamplePanel p, Color c) {
        panel = p;
        color = c;
    }

    public void actionPerformed(ActionEvent e) {
        System.out.println(e);
        panel setColor(color);
        panel repaint();
    }
}
Change Panel Class

• create setColor method
• add array of colors
• change constructor
  • call addActionListener on each button and add an appropriate ColorAction
Changes to EventExamplePanel

```java
private Color currentColor;

private static String[] buttonNames = {"Red", "Green", "Blue"};
private static Color[] colors = {Color.RED, Color.GREEN, Color.BLUE};

private JButton[] buttons;

public EventExamplePanel(){
    currentColor = Color.ORANGE;
    buttons = new JButton[buttonNames.length];

    for(int i = 0; i < buttonNames.length; i++) {
        buttons[i] = new JButton(buttonNames[i]);
        buttons[i].addActionListener(new ColorActionActionListener(this, colors[i]));
        add(buttons[i]);
    }
}
```

- Demo -> Examine output of ActionPerformed
- Add more buttons and colors