GUI and Event-Driven Programming
Recitation – 3/6/2009

CS 180
Department of Computer Science,
Purdue University
Announcement

- Project 5 due Wed, Mar. 11 at 10 pm.
- All questions on the class newsgroup.
- Do not wait till the last minute to start your project!
- Mid-term evaluation of this course.
Inheritance & Interface

- They will be focused later, but we can have a glance at their basic ideas.
- Inheritance models the is-a relationship.
  - Student is a Person.
  - Car is a Vehicle.
- Use extends to state an inheritance.
- Interface is a kind of protocol.
  - Human can Move. (Move is an interface here)
  - Animal can Move.
- Use implements to implement an interface.
A Simple GUI Example

- This simple program produces a window and displays some text.
  - `JFrame`: to create a window
  - `JLabel`: to create a label
  - `getContentPane().add()`: add a component such as a label to the content pane of the window
  - `setTitle()`: set the title of the window
  - `setSize()`: set the size of the window
  - `setVisible()`: permits the programmer to specify when GUI objects should be displayed and when they should not
public class MyWindow {

    public static void main(String args[])
    {
        JFrame myWindow=new JFrame();
        //create the window
        myWindow.setSize(300, 200);
        //set the title of the window
        myWindow.setTitle("this is a window");
        //create the label
        JLabel myLabel=new JLabel("this is a label");
        //add the label to the content pane of the window
        myWindow.getContentPane().add(myLabel);
        //set color of the content pane
        myWindow.getContentPane().setBackground(Color.CYAN);
        //make the window visible
        myWindow.setVisible(true);
    }
}
Event Driven Programming

- An event is an object that represents an action.
  - Mouse clicking
  - Key pressing
  - Window closing
  - …
ActionListener & ActionAdapter

- ...\texttt{implements ActionListener}
- ...\texttt{extends ActionAdapter}
- They can both do the same thing, but what’s their differences?
- Keep that question till you learn inheritance and interface systematically.
- Anyway, remember the most useful method \texttt{actionPerformed}. What’s its argument?
  - \texttt{ActionEvent} (An event is also an object)
Layout Managers

- A layout manager arranges objects within a container.
- After a container has been created, you can set its layout manager using the `setLayout` method.

```java
Container contentPane = frame.getContentPane();
contentPane.setLayout(new FlowLayout());
```

- `FlowLayout`: It simply lays out components in a single row, starting a new row if its container is not sufficiently wide.
Layout Managers

- **BorderLayout**: It places components in up to five areas: top, bottom, left, right, and center.
  - Default layout

- **GridLayout**: It simply arranges a bunch of components in a grid of rows and columns.
Nesting Panels

- How to create any kind of GUI as you wish?
  - The first choice is to draw a picture on the paper, calculate positions for all components and type in one by one.
  - Maybe we can use an IDE to drag some components onto the frame.
  - But what happens when the window is resized?
  - One decent solution is to use nested panels.
Nesting Panels

DIY time – How to get these GUI styles?
Nesting Panels

- What about this?
Nesting Panels

- Very well! This?

Whoops!! Why?
More Components

- **Menus**
  - JMenuBar
  - JMenu
  - JMenuItem

- **Text**
  - JTextField
  - JTextArea

- More... So, how can we remember all of them?!
  - Java API and the Internet
Quiz

Do you think JPanel and JButton are similar components? If not, what’s the main difference between them? In this way, we can classify the components into two categories. Can you give two more examples, one similar to JPanel and the other similar to JButton?