Announcements

- Prelim 2 reminders:
  - Tue 4/19, 7:30-9pm
  - OH 155, 255
  - Which room? Will be posted on website (and in class)
- In the world of DIS:
  - CIS300!

Overview

- Motivation
- JFC
- AWT and Swing
- Creating and Using A GUI
- Containers
- Layout Managers
- Where to go for more info?
  - http://java.sun.com/docs/books/tutorial/uiswing/
  - http://java.sun.com/j2se/1.5.0/docs/relnotes/features.html

Motivation

- **Program Driven:**
  - statements execute in sequential, pre-determined order
  - typically use keyboard/file I/O from console
- **Event Driven:**
  - program waits for user input to activate certain statements
  - typically use graphical I/O
  - GUI: graphical user interface
- Design...Which to pick?
  - program called by another program?
  - program used at command line?
  - program interacts often with user?
  - program used in window environment?
  - “old school” vs “new school”?
- Up next...How does Java do GUIs?
Java Foundation Classes

- **JFC**: Java Foundation Classes
  - API classes for building GUIs
  - Major components:
    * Swing
    * Pluggable Look and Feel Support
    * Accessibility API
    * Java 2D API
    * Drag and Drop Support
    * Internationalization
- **Our main focus**: **Swing**
  - the visual components of the GUI
  - building windows
  - user interactions
  - built upon something called **AWT** (Abstract Window Toolkit)
- What are the other things...?

More Aspects of JFC

- Pluggable Look and Feel Support:
  - define look for particular windowing environment
  - ex) Windows, Motif
- Accessibility API:
  - assistive technologies such as screen readers and Braille
  - displays for non-standard I/O
- **Java 2D**:
  - draw images
  - rectangles, lines, circles, images, ....
- Drag and Drop:
  - drag and drop between Java application and a native application
- Internationalization:
  - other languages

Brief Example

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

public class Intro extends JFrame {
    private int count;
    private JButton b = new JButton("Push Me!");
    private JLabel label = new JLabel(generateLabel());

    public static void main(String[] args) {
        JFrame f = new Intro();
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(200, 100);
        f.setVisible(true);
    }

    public Intro() {
        setLayout(new FlowLayout(FlowLayout.LEFT));
        add(b);
        add(label);
        b.addActionListener( new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                count++;
                label.setText(generateLabel());
            }
        });

        public String generateLabel() {
            return "Count: " + count;
        }
    }
}
```

Creating a GUI

- We'll split your GUI learning into 2 parts:
  - **Statics**: what you draw on the screen
    - **Components**: what you see on the screen
    - **Containers**: special kind of components that contain other components
    - **Layout managers**: objects that control placement and sizing of components
  - **Dynamics**: how user interacts with elements on screen
    - **Events**: an object that represents an occurrence
    - **Listeners**: an object that listens for an event
    - **Helper classes**: AWT classes Graphics, Color, Font, FontMetrics, Dimension
- Start with statics:
  - figure out which components you want
  - pick a top-level container in which to put the components
  - pick layout manager to arrange components
  - place components
- Swing or AWT?
**AWT and Swing**

- **AWT**:
  - use code for windowing system from your computer
  - called *heavyweight*
  - disadvantage: not being able to port to other OS
  - basic API package: `java.awt.*`

- **Swing**:
  - Swing classes have no native code
  - more portable, added functionality
  - called *lightweight* because mostly written in Java
  - essentially supersedes many AWT components
  - basic API package: `javax.swing.*`

Swing replaced AWT? not quite:
- you'll usually use AWT's event model
- still need AWT for each OS
- more info:
- So...where are all these classes?

**Hierarchy for Statics Classes**

- Java API!
- Basic statics hierarchy (AWT, Swing):
  - `Object`
  - Helper Classes
  - Layout Managers
  - `Component`
    - AWT components, like `Button`, `Canvas`, etc.
    - `Container` (in AWT)
      - `Panel`
      - `Applet`
      - `JApplet` (heavyweight)
      - `Window`
      - `Frame` (`JFrame` (heavyweight))
      - `Dialog`
      - `JDialog` (heavyweight)
      - `JWindow`
    - `JComponent` (lightweight)
      - many subclasses that start with `J`

Now, more detail on Components, Containers, Layout Managers...

**Components**

- **Components**...what you paint:
  - visual part of interface
  - represents something with position and size
  - can be painted on screen and receive events
  - buttons, labels, etc.
- See http://java.sun.com/docs/books/tutorial/uiswing/components/index.html
- Some examples in ComponentExamples.java (next slide)

**Component Examples**

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

public class ComponentExamples extends JFrame {
    public static void main(String[] args) {
        ComponentExamples f = new ComponentExamples();
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.pack();
        f.setVisible(true);
    }
    public ComponentExamples() {
        setLayout( new FlowLayout(FlowLayout.LEFT) );
        add(new JButton("Button"));
        add(new JLabel("Label"));
        add(new JComboBox(new String[] { "A", "B", "C" } ));
        add(new JCheckBox("JCheckBox"));
        add(new JSlider(0,100));
        add(new JColorChooser());
        add(new JSlider(0,100));
        add(new JColorChooser());
    }
}
```
Containers

- **Container**: Component that holds other components
  - eg, see JFrame components
- **Top-level container**: special kind of container (eg, JFrame) that holds all components that will appear on screen

JFrame

- commonly-used top-level container
- example)
  - JFrame f = new JFrame("Title!");
- using default layout manager to place components
  - you can set different layout managers
  - descriptions coming up!
- content pane:
  - Old Java: add components to content pane
    - f.getContentPane().add(new JButton("OK"));
  - Java 1.5: you can add components to frame, but Java still puts them on the content pane:
    - f.add(new JButton("Hi"));
- See JRootPane in API and tutorial for more info

JPanel

- Simplest container:
  - opaque container
  - handy for place to draw graphics
  - store components but no borders
- Not top-level:
  - cannot be "stand-alone"
  - must put in other container
- example)
  - JFrame frame = new JFrame("Title!");
  - JPanel panel = new JPanel();
  - p.add(new JButton("OK!"));
  - frame.add(panel);

Example 1

```java
import javax.swing.*;
public class Basic1 {
    public static void main(String[] args) {
        // Create window:
        JFrame f = new JFrame("Basic Test!");
        // Set 500x500 pixels^2:
        f.setSize(500, 500);
        // Show the window:
        f.setVisible(true);
        // Quit Java after closing the window:
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```
Example 2

```java
import javax.swing.*;
public class Basic2 {
    public static void main(String[] args) {
        new MyGUI();
    }
}

class MyGUI {
    JFrame f = new JFrame("Basic Test2!");
    f.setSize(500, 500);
    f.setVisible(true);
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
```

Example 3

```java
import javax.swing.*;
public class Basic3 extends JFrame {
    public static void main(String[] args) {
        new Basic3();
    }
    public Basic3() {
        // Title window:
        setTitle("Basic Test!");
        // Set 500x500 pixels:
        setSize(500, 500);
        // Show the window:
        setVisible(true);
        // Quit Java after closing the window:
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Layout Managers

- What is a layout manager?
  - object that controls placement and sizing of components in container
  - if you do not specify a layout manager, the container will use a default:
    * JPanel—FlowLayout
    * JFrame—BorderLayout

- five common layout managers:
  * BorderLayout, BoxLayout, FlowLayout, GridBagLayout, GridLayout

- General syntax:
  ```
  container.setLayout(new LayoutMan());
  ```

- Examples:
  ```
  JPanel p1 = new JPanel(new BorderLayout());
  JPanel p2 = new JPanel();
  p2.setLayout(new BorderLayout());
  ```

Some Layout Managers

- **FlowLayout**:
  - components arranged in container from left to right in order added
  - new row started each time row ends
  - simple alignment with RIGHT, LEFT, CENTER fields
  - see also BoxLayout

- **GridLayout**:
  - arranges components in rectangular grid (think array)
  - rows, columns defined by constructor
  - components go into grid left-to-right, then top-to-down

- **BorderLayout**:
  - divides window into 5 areas: East, South, West, North, Center
  - add components with `add(Component, index)`
  - indices are BorderLayout.EAST, ...
More Layout Managers

- **CardLayout:**
  - tabbed index card look from Windows
- **GridBagLayout:**
  - most versatile, but most complicated
- Custom:
  - define your own layout manager
  - best to try Java's supplied version first...
- Null Layout
  - don’t use a layout manager
  - programmer has to give absolute locations
  - can be dangerous to application because of platform dependency

---

Another Example

```java
import javax.swing.*;
import java.awt.*;
public class Statics1 {
    public static void main(String[] args) {
        new MyGUI();
    }
}
```

```java
class MyGUI {
    private JFrame f;
    private Container c;
    public MyGUI() {
        f = new JFrame("Statics1");
        f.setSize(500,500);
        f.setLayout(new FlowLayout(FlowLayout.LEFT));
        for (int b = 1; b < 9; b++)
            f.add(new JButton("Button "+b));
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

---

Statics Example

```java
import java.awt.*;
import java.awt.event.*;
public class Calculator {
    public static void main(String[] args) {
        String tmp1; double v1; // 1st user-entered value
        String tmp2; double v2; // 2nd user-entered value
        String op; // desired operation
        double result = 0; // desired result of v1+v2
        // Obtain user input:
        tmp1 = JOptionPane.showInputDialog("Enter 1st value:");
        tmp2 = JOptionPane.showInputDialog("Enter 2nd value:");
        op = JOptionPane.showInputDialog("Enter operation:");
        // Convert input to numbers:
        v1 = Double.parseDouble(tmp1);
        v2 = Double.parseDouble(tmp2);
        // Perform operation:
        if (op.equals("+"))
            result = v1+v2;
        else if (op.equals("-"))
            result = v1-v2;
        else if (op.equals("*"))
            result = v1*v2;
        else if (op.equals("/"))
            result = v1/v2;
        else {
            JOptionPane.showMessageDialog(null, "Unknown operator!");
            System.exit(0);
        }
        // Output result:
        JOptionPane.showMessageDialog(null, "Result: "+result);
    }
}
```

---

Dialog Boxes

```java
import java.awt.*;
import java.awt.event.*;
public class Calculator {
    public static void main(String[] args) {
        String tmp1; double v1; // 1st user-entered value
        String tmp2; double v2; // 2nd user-entered value
        String op; // desired operation
        double result = 0; // desired result of v1+v2
        // Obtain user input:
        tmp1 = JOptionPane.showInputDialog("Enter 1st value:");
        tmp2 = JOptionPane.showInputDialog("Enter 2nd value:");
        op = JOptionPane.showInputDialog("Enter operation:");
        // Convert input to numbers:
        v1 = Double.parseDouble(tmp1);
        v2 = Double.parseDouble(tmp2);
        // Perform operation:
        if (op.equals("+"))
            result = v1+v2;
        else if (op.equals("-"))
            result = v1-v2;
        else if (op.equals("*"))
            result = v1*v2;
        else if (op.equals("/"))
            result = v1/v2;
        else {
            JOptionPane.showMessageDialog(null, "Unknown operator!");
            System.exit(0);
        }
        // Output result:
        JOptionPane.showMessageDialog(null, "Result: "+result);
    }
}
```