FALL 2003 CS211
GUIS: STATICS

❒ Announcements
- The Supplement in consulting office
- The Tutorial:
  - http://java.sun.com/docs/books/tutorial/uiswing/mini/index.html
  - http://java.sun.com/docs/books/tutorial/uiswing/index.html
- Prelim 2 coming up 11/18!

❒ Overview
- Motivation
- JFC
- AWT and Swing
- Creating and Using A GUI
- Containers
- Layout Managers

1. Motivation

1.1 Program-Driven:

- program executes each statement in a sequential, pre-determined order
- typically use keyboard/file I/O from console
- usefulness of keyboard I/O?

1.2 Event-Driven

- program waits for user input to activate certain statements
- typically use graphical I/O
- ***GUI***: graphical user interface

1.3 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"?

2. Java Foundation Classes

2.1 JFC

- API classes for building GUIs
- five major components:
  - Swing
  - Pluggable Look and Feel Support
  - Accessibility API
  - Java 2D API
  - Drag and Drop Support

2.2 Swing

- the visual components of the GUI
- built on AWT (Abstract Window Toolkit)
- AWT was original API for making GUIs
- Swing built upon AWT
  - supersedes some classes
  - uses some others

2.3 Pluggable Look and Feel Support

- ways to define certain look for a particular windowing environment: Motif vs Windows vs …
- http://java.sun.com/docs/books/tutorial/uiswing/misc/plaf.html

2.4 Accessibility API

- tools for assistive technologies such as screen readers and Braille
- displays for non-standard I/O
- http://java.sun.com/docs/books/tutorial/uiswing/misc/access.html

2.5 Drag and Drop

- drag and drop between Java application and a native application
- http://java.sun.com/docs/books/tutorial/uiswing/misc/dnd.html

2.6 Example

- AWT?
- Swing?
- Events?

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

public class Counter3 extends JFrame {

  private int count;
  private JButton b = new JButton("Push Me!");
  private JLabel label = new JLabel(generateLabel());
  private Container c = getContentPane();

  public static void main(String[] args) {
    Counter3 f = new Counter3();
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.setSize(200,100);
    f.setVisible(true);
  }

  public Counter3() {
    c.setLayout(new FlowLayout(FlowLayout.LEFT) );
    c.add(b);
    c.add(label);

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

  }

    private String generateLabel() {
        return "Count: "+Integer.toString(count);
    }

}
```


3. AWT and Swing

3.1 AWT

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

3.2 Swing

- Swing classes have no native code
  - more portable
  - added functionality
  - called ***lightweight*** because mostly written in Java
  - essentially supersedes many AWT components
- uses ***layout managers*** to arrange ***components*** inside ***containers***
- basic API package: **`javax.swing.*`**
- replace AWT? not quite:
  - Swing uses AWT event model
  - still need AWT for each OS

4. Creating and Using A GUI

4.1 Quick Overview

- http://java.sun.com/docs/books/tutorial/uiswing/mini/index.html
- http://java.sun.com/docs/books/tutorial/uiswing/start/swingTour.html

4.2 Overall classification of classes

- ***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
- ***Events***: an object that represents an occurrence
- ***Listeners***: an object that listens for an event
- ***Helper classe***s: AWT classes **`Graphics`**, **`Color`**, **`Font`**, **`FontMetrics`**, **`Dimension`**

4.3 Process of Creating a GUI

- Set up components:
  - figure out which Swing components you want
  - pick a container in which to put the components (special kind of container: call top-level)
  - pick layout manager: you could use default!
  - place components
- Set up listeners:
  - create listener objects and connect them to the components that generate events
  - ensure that user events are handled
  - will discuss this part more in GUI Dynamics

4.4 Basic Object Hierarchy

```
Object
    Helper Classes
    Layout Managers
    Component
        AWT components, like Button, Canvas, etc.
        Container
            Panel
                Applet
                    JApplet (heavyweight)
            Window
                Frame
                    JFrame (heavyweight)
                Dialog
                    JDialog (heavyweight)
                JWindow
            JComponent (lightweight)
                many subclasses that start with "J"
```

4.5 Components

- 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

5. Containers

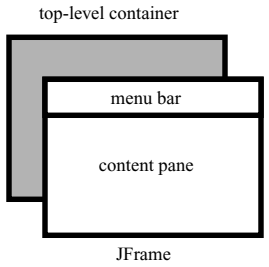
http://java.sun.com/docs/books/tutorial/uiswing/components/toplevel.html

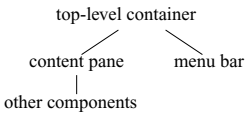
5.1 Container

- special kind of component
- collect other components together

5.2 Top-Level Container

- top-level container:
  - special kind of container
  - holds all components that will appear on screen
  - JFrame,
- containment hierarchy:
  - layer of components
  - bottom layer is top-level container
  - next layer is ***content pane*** that actually holds visual components
  - successive layers are more components
  - forms a tree of containment: containment hierarchy
- optional: add a menu bar to a top-level container, but it's outside of the content pane





5.3 Four Top-level Containers

- **`JFrame`**: window that stores other components in applications has border and can have a **`JMenuBar`**
- **`JDialog`**: pop-up window or message box
- **`JApplet`**: Swing-based Applet
- **`JWindow`**: Swing version of Window, but not very useful! (no border)

5.4 JFrames

- commonly-used top-level container
- example)

```
JFrame f = new JFrame("Title!");
f.getContentPane().add(new JButton("OK"));
```

- currently using default layout manager to place components

5.5 JPanel

- opaque container
  - handy for place to draw graphics
  - store components but no borders
  - thus, simplest container!
- cannot be "stand-alone"
  - it's not a top-level container
  - put inside top-level container or another panel
- example)

```
JFrame frame = new JFrame("Title!");
JPanel panel = new JPanel();
p.add(new JButton("OK!"));
frame.getContentPane().add(panel);
```

- http://java.sun.com/docs/books/tutorial/uiswing/components/panel.html

5.6 Example 1

```
import javax.swing.*;
public class Basic {
  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);

  }
}
```

5.7 Example 2

```
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);
    }
}
```


6. Layout Managers

6.1 Tutorial

- http://java.sun.com/docs/books/tutorial/uiswing/overview/layout.html
- http://java.sun.com/docs/books/tutorial/uiswing/layout/using.html

6.2 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`**

6.3 Setting Layout Manager

- general syntax: **`container.setLayout(new LayoutMan())`**
- examples)

```
JPanel p1 = new JPanel(new BorderLayout());

JPanel p2 = new JPanel();
p2.setLayout(new BorderLayout());
```

6.4 FlowLayout

- http://java.sun.com/docs/books/tutorial/uiswing/layout/flow.html
  - simplest
  - 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`**: http://java.sun.com/docs/books/tutorial/uiswing/layout/box.html

6.5 GridLayout

- http://java.sun.com/docs/books/tutorial/uiswing/layout/grid.html
- arranges components in rectangular grid (think array)
- rows, columns defined by constructor
- components go into grid left-to-right, then top-to-down

6.6 BorderLayout

- http://java.sun.com/docs/books/tutorial/uiswing/layout/border.html
- divides window into 5 areas: East, South, West, North, Center
- add components with **`add(Component,index`**)
- indices are **`BorderLayout.EAST`**, …

6.7 CardLayout

- http://java.sun.com/docs/books/tutorial/uiswing/layout/card.html
- tabbed index card look from Windows

6.8 GridBagLayout

- http://java.sun.com/docs/books/tutorial/uiswing/layout/gridbag.html
- most versatile, but most complicated

6.9 Custom

- http://java.sun.com/docs/books/tutorial/uiswing/layout/custom.html

6.10 Null Layout

- http://java.sun.com/docs/books/tutorial/uiswing/layout/none.html
- don't use a layout manager
- programmer has to give absolute locations
- can be dangerous to application because of platform dependency

7. More Examples

7.1 Example 1

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

public class AddStuff2 {
  public static void main(String[] args) {
    new MyGUI();
  }
}

class MyGUI {

  private JFrame f;
  private Container c;

  public MyGUI() {

    f = new JFrame("AddStuff2");
    f.setSize(500,500);

    c = f.getContentPane();
    c.setLayout(new FlowLayout(FlowLayout.LEFT) );

    for (int b = 1; b < 9; b++)
      c.add(new JButton("Button "+b));

    f.setVisible(true);
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  }
}
```


7.2 Example 2

```
import javax.swing.*;
import java.awt.*;
public class AddStuff3 {
  public static void main(String[] args) { new MyGUI(); }
}

class MyGUI {
  private JFrame f;
  private Container c;
  private LayoutManager l;
  private MyPanel[] p;
  private int dim;

  public MyGUI() {
    makeWindow();
    showWindow();
  }

  private void makeWindow() {
    dim = 4;
    f = new JFrame("AddStuff3");
    p = new MyPanel[dim*dim];
    c = f.getContentPane();
    l = new GridLayout(dim,dim,2,2);
    c.setLayout(l);
    for (int i=0;i<p.length;i++) {
      p[i]=new MyPanel();
      c.add(p[i]);
    }
  }

  private void showWindow() {
    f.setSize(500,500);
    f.setVisible(true);
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```



```
class MyPanel extends JPanel {
    public void paintComponent (Graphics g) {
        super.paintComponent(g); // clear drawing area
        g.setColor(Color.white);
        g.fill3DRect(0,0,getWidth(),getHeight(),true);
    }
}
```