GUI Programming

OUTLINE

- **Assertions and Loop Invariants**
- **Event-Oriented Programming**
- **GUI – first look**

Reasoning about Programs: Assertions and Loop Invariants

- **Assertions: logical statements about a program that are claimed to be true; generally written as a comment**
- **Preconditions and postconditions are assertions**
- **A loop invariant is an assertion**
  - **Helps prove that a loop meets it specification**
  - **True before loop begins, at the beginning of each repetition of the loop body, and just after loop exit**

Assertions and Loop Invariants Example

```
int i = 0;
// invariant:  for all k, such that 0 <= k < i, x[k] != target
while (i < x.length) {
    if (x[i] == target)
        return i;   // target found at i
    i++;            // Test next element
}

// assert: for all k, such that 0 <= k < i, x[k] != target
//         and i >= x.length
return -1;       // target not found
```

Testing

- **Test drivers and stubs are tools used in testing**
  - **Test drivers exercise a method or class**
  - **Stubs stand in for called methods**

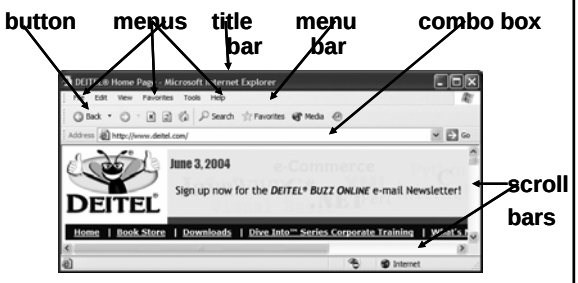
Intro to Event-Oriented Programming

- **Batch-Processing**
- **Request-Response Programming**
- **Event-Oriented Programming**

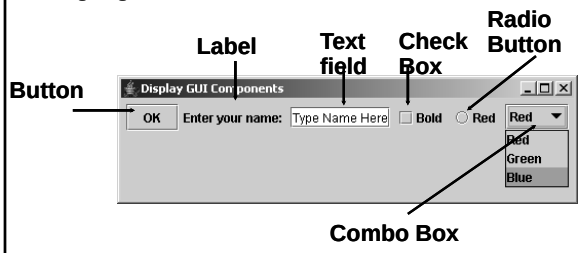
GUI Programming Concepts

- **conventional (request-response) programming:**
  - **sequence of operations is determined by the *program***
  - **what you want to happen, happens when you want it**
- **event-oriented programming:**
  - **sequence of operations is determined by the user's interaction with the application's interface**
  - **anything that can happen, happens at any time**

GUI - first look: Graphical Components



Graphical Components – more...



GUI Programming Concepts in Java

- **Java GUI ("Swing") has components**
- **Windows GUI has controls**
- **Unix GUI has widgets**
- **examples: labels, buttons, check boxes, radio buttons, text input boxes, pull down lists**
- **Java Swing components: JLabel, JButton, JCheckBox, JRadioButton, JTextField, JTextArea, JComboBox**

Java GUI history: the AWT

- **AWT(JDK 1.0, 1.1): Abstract Window Toolkit**
- **package: java.awt, java.awt.event**
- **heavyweight components using native GUI system elements**
- **used for applets until most browsers supported JRE 1.2**

AWT vs. Swing (1)

- **AWT: Abstract Windowing Toolkit**
  - **import java.awt.***
  - **AWT drawing uses "native peers" -- creating an AWT button creates a native peer (Unix, Mac, Win32) button to put on screen, and then tries to keep the AWT button and the peer in sync.**
  - **Advantage: an AWT app has the "native" appearance for buttons etc. since there are in fact native buttons on screen.**
  - **Disadvantage: very hard to implement in a reliable way -- keeping peers consistent on all platforms.**

AWT vs. Swing (2)

- **Swing is a new widget toolkit for Java (came with Java2)**
  - import javax.swing.*
  - **Extends AWT (provides a more sophisticated set of GUI components)**
  - **Standard dialog boxes, tooltips, …**
  - **Look-and-feel, skins**
  - **Event listener**
  - **Swing inherits from AWT**
  - **AWT still used for events, layouts**

Swing Components in Java

- **advanced GUI support. e.g. drag-and-drop**
- **package names: javax.swing, javax.swing.event**
- **components inherit from JComponent**
- **components are added to a top-level container: JFrame, JDialog, or JApplet.**

Sources on the Internet:

- **Application Programming Interface:**
- **http://java.sun.com/javase/6/docs/api/**
- **Swing API documentation**
- **http://java.sun.com/javase/6/docs/technotes/guides/swing/**
- **Swing architecture**
- **http://java.sun.com/products/jfc/tsc/articles/architecture/**

running a Swing application

- **`java -Dswing.aatext=true MySwingClass`**
  - **the option sets the system property "swing.aatext" to "true" to enable anti-aliasing for every JComponent**
- **`javaw` runs a GUI without the console window**

Event-driven Programming

- **Normal (control flow-based) programming**
  - **Approach**
    - Start at main()
    - Continue until end of program or exit()
- **Event-driven programming**
  - **Unable to predict time & occurrence of event**
  - **Approach**
    - Start with main()
    - Build GUI
    - Await events (& perform associated computation)

Event-driven Paradigm in Java

- **During implementation**
  - **Implement event listeners for each event**
  - **Usually at least one event listener class per widget**
- **At run time**
  - **Register listener object with widget object**
  - **Java generates event object when events occur**
  - **Java then passes event object to event listener**

Basic GUI Programming Steps in Java

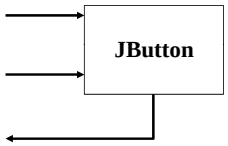
- declare a container and components
- add components to one or more containers using a layout manager
- register event listener(s) with the components
- create event listener method(s)

Basic GUI Programming Concepts in Java

- **Example:** JFrameDemoTM.java, JFrameDemoTM2.java, JFrameDemo.java
- **container:** a screen window/applet window/panel that groups and arranges GUI components
- **GUI component:** an object with visual representation
  - **Swing containers:** JFrame, JApplet, JPanel
  - **AWT containers:** Frame, Applet, Panel

GUI Component API

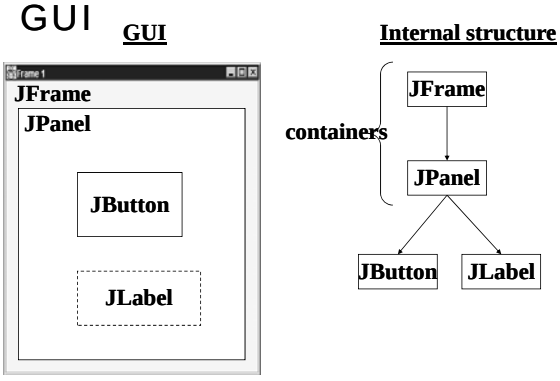
- Java: GUI component = class
- Properties
- Methods
- Events



Using a GUI Component

1. Create it
   - Instantiate object: b = new JButton("press me");
2. Configure it
   - Properties: b.text = "press me";
   - Methods: b.setText("press me");
3. Add it
   - panel.add(b);
4. Listen to it
   - Events: Listeners

Anatomy of an Application GUI



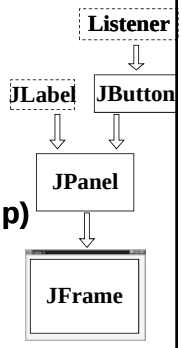
Using a GUI Component (II)

1. Create it
2. Configure it
3. Add children (if container)
4. Add to parent (if not JFrame)
5. Listen to it

the order IS important

Build from bottom up

- **Create (order indifferent):**
  - Frame
  - Panel
  - Components
  - Listeners
- **Build: (aggregate bottom up)**
  - listeners into components
  - components into panel
  - panel into frame



Code

```
JFrame f = new JFrame("title");
JPanel p = new JPanel( );
JButton b = new JButton("press me");

p.add(b);              // add button to panel
f.setContentPane(p);   // add panel to frame

f.setVisible (true);
```

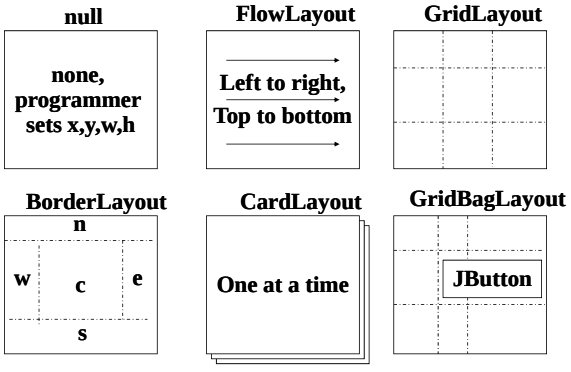
(trivial) Application Code

```
import javax.swing.*;
class hello {
   public static void main(String[] args){
     JFrame f = new JFrame("title");
     JPanel p = new JPanel();
     JButton b = new JButton("press me");

     p.add(b);      // add button to panel
     f.setContentPane(p);// add panel to frame

     f.setVisible (true);
   }
}
```

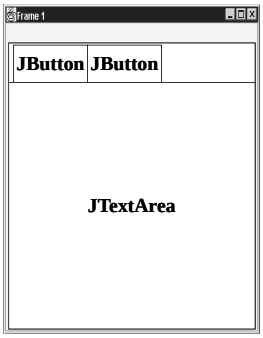
Layout Managers

- **Automatically control placement of components in a panel**
- **Why?**

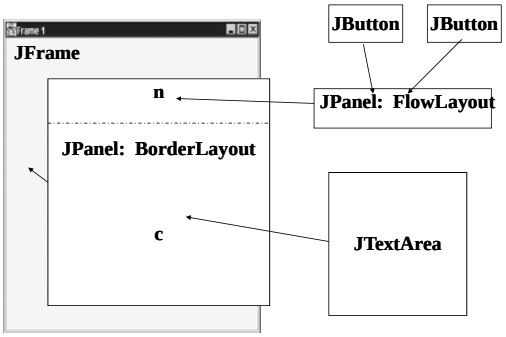
Layout Manager Heuristics



Combinations



Combinations

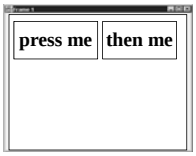


Our code so far: null layout

```
JFrame f = new JFrame("title");
JPanel p = new JPanel( );
JButton b = new JButton("press me");

b.setBounds(new Rectangle(10,10, 100,50));
p.setLayout(null);        // x,y layout
p.add(b);
f.setContentPane(p);
```

Code: FlowLayout

```
JFrame f = new JFrame("title");
JPanel p = new JPanel( );
FlowLayout L = new FlowLayout( );
JButton b1 = new JButton("press me");
JButton b2 = new JButton("then me");

p.setLayout(L);
p.add(b1);
p.add(b2);
f.setContentPane(p);
```



Set layout mgr before adding components

Using a GUI Component (Recap)

1. **Create it**
2. **Configure it**
3. **Add children (if container)**
4. **Add to parent (if not JFrame)**
5. **Listen to it**

the order IS important

Questions?

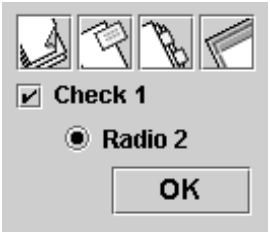
Java GUI Components Examples

- **Examples**
  - **JFrame**
  - **JTextfield**
  - **Jlabel**
  - **Button**
  - **JList**
  - **JComboBox**
  - **Menu**
  - **Combo**
  - **Panes**
  - **Indicators**
  - **Dialog boxes**
  - **JFileChooser**
  - **Color chooser**
  - **JTable**
  - **JTree**

JFrame

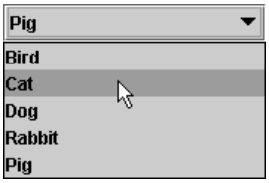
JTextField and JLabel

Buttons

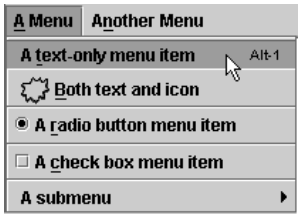


Toolbar

JList and JComboBox



Menu



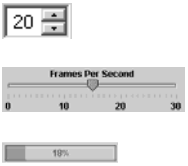
Panels and Panes

Tabbed Pane

Split Pane

Scroll Pane

Various Indicators

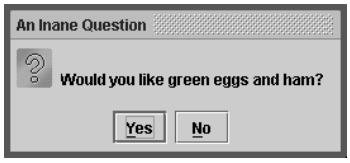


Spinner

Slider

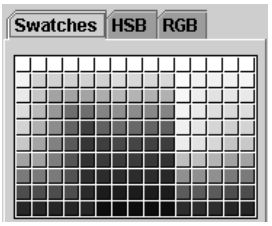
Progress Bar

A Dialog Box



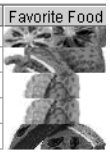
JFileChooser

Color Chooser



Jtable and JTree

First Name	Last Name	Favorite Food
Jeff	Dinkins	
Ewan	Dinkins	
Amy	Fowler	
Hania	Gajewska	
David	Geary	



Text Fields