Java AWT Notes

AWT Class Hierarchy (partial)

Object
  Container
    Component
      Container
        Window
          Frame
            Panel
              Button
                Canvas
          Ex.

Parent class for all AWT objects - lots of common behavior defined here

Container = Component that can hold other Components

Frame - Top-level window
  Useful container for grouping Components into single Component

Component
  Container = Component that can hold other Components

Button
  UI control

Canvas
  Simple drawing surface

Panel
  Useful container for grouping Components into single Component

Window
  Top-level window

Parent class for all AWT objects - lots of common behavior defined here

A Simple Java Application

```
import java.awt.*;
// free-standing application w/Window
public class App extends Frame {
  public void paint(Graphics g) {
    // redraw screen when requested by window manager
  }

  other function declarations
}

// main program -- create window etc.
Public static void main(String args[]){
  App app = new App();
  set up window app
  app.show();
  continue processing
}
```

Java Application Notes

- paint() is called by the window manager as needed, i.e., asynchronously.
- Component can request redrawing by calling repaint()
- Window manager doesn’t call paint() directly - it calls update(). The default implementation inherited from component is (roughly)
  ```
  public void update(Graphics g){
    set window to background color
    paint(g);
  }
  ```
- Override update() if desired (ex. less flicker)

Event Handling

- User interface components generate events
- Objects (often other components) can register themselves to receive events of interest
- When an event happens, an appropriate method is called in all listeners (all interested objects)
- A listener object must implement the interface corresponding to the events, which means implementing all methods declared in the interface
- Need import java.awt.event.*;

Example: Track Mouse

```
Public class TrackMouse
extends Frame
implements MouseMotionListener {
  // instance variables
  int locX = 100; // last mouse
  int locY = 100; // location

  // constructor - register this
  // object to receive mouse move
  public TrackMouse() {
    addMouseMotionListener(this);
  }

  ..."
Example: Track Mouse (cont)

```java
// MouseMotionListener methods
public void MouseMoved() {} //
public void MouseDragged(MouseEvent e){
    locX = e.getX();
    locY = e.getY();
    repaint();
}
// repaint screen
public void paint(Graphics g){
    g.drawString("Here!", locX, locY);
}
```

Example: Button

- Most user-interface components need to be allocated, added to an appropriate container, and interested objects need to register to receive events.

```java
public class WatchButton extends Frame implements ActionListener {
    // instance variables
    Button belly; // the button
    ...
    public WatchButton() {
        belly = new Button("press me");
        add(belly);
        belly.addActionListener(this);
    }
    // react to button press
    public ActionPerformed(ActionEvent e) {
        if (e.getSource()==belly){
            respond to button press
        }
    }
```

Example: Button (concl)

- The test isn’t strictly necessary if we know that belly is the only button that could generate the event.
- Many other UI components (text boxes, dials, …) generate similar events. The ActionEvent contains details of the event (source, kind, data values, locations, …).

Layout Managers

- A Layout Manager is associated with every Container. The layout manager is responsible for positioning components in the container when the container is redrawn.
- Basic layout manager classes
  - FlowLayout - arranges components from left to right, top to bottom. Nothing Fancy
  - GridLayout - regularly spaced rows and columns
  - BorderLayout - Components can be placed in the Center, North, South, East, or West. Useful trick: to place several controls in one of these places, create a Panel containing the controls, then place the Panel in one of the 5 BorderLayout locations.
  - GridBagLayout - General constraint layout. Can create almost any effect, but can take some work to do it. If you’re comfortable with complex HTML tables, you’ll feel at home.

Layout Manager Example

```java
public SomeContainer() extends ... {
    button c = new Button("cold");
    button w = new Button("warm");
    layout(new BorderLayout());
    add(c, "North");
    add(w, "South");
    ...
```

- Also need to add listeners for the buttons, etc.