CSE 142 Summer 2001

Event Driven Programming & Graphical User Interfaces – a Survey

Introduction

- Review:
  - Creating Objects
  - Conventional programs with data
- Today:
  - Event-driven programming
  - User interface objects
  - Applications & Applets
- Disclaimer: Just skimming the details, and giving basic examples using Java's Abstract Window Toolkit. Much more to learn – many more methods, SWING, etc.

Conventional Programs

- Form and order of input is given in program specification
- Program typically reads input, processes somehow, produces some output
  - Weather data examples
  - Simulation

Event Driven Programs

- Program starts up then waits for "events" and reacts to them
- Events
  - Mouse click
  - Keyboard press
  - Menu selection
  - Timer or clock tick
  - Request over a network connection
  - etc. etc. etc.
- Typically, events may arrive in arbitrary order

Java Event Handling Architecture

- Objects indicate they are interested in particular classes of events by implementing an appropriate interface and registering themselves with the objects that generate the events
- When an event happens, an appropriate method is called in all registered objects listening for that event

Example Event Interface - MouseListener

- Specification

```
interface MouseListener {
    public void MouseClicked(MouseEvent e);
    public void MouseEntered(MouseEvent e);
    public void MouseExited(MouseEvent e);
    public void MousePressed(MouseEvent e);
    public void MouseReleased(MouseEvent e);
}
```

- An object that wants to know about mouse events that happen to it can implement this interface & register itself by calling addMouseListener(this) on an appropriate object
  - Must implement all methods, but bodies can be empty for methods that the object wants to ignore

Java Graphical User Interface Components

- Many kinds: buttons, labels, text boxes, drawing areas, …
- A few are "top-level" objects, including
  - Frame (a regular window)
  - Applet (a window that is hosted in another window)
- Some are individual objects that have no substructure
  - Label, Button, …
- Top-level and some other kinds of objects (Panel) can "contain" other objects
  - A Frame may contain text fields, labels, pictures, buttons, etc…

Drawing – Method Paint

- All of the GUI components are responsible for displaying themselves when asked
- Mechanism – implement method paint(…)

```
public void paint(Graphics g) {
    g.setColor(Color.green);
    g.fillRect(0,0,width,height);
    g.setColor(Color.black);
    g.drawRect(0,0,width-1,height-1);
```

  - The *graphics context* g is a reference to the window object where the drawing actually occurs
  - Graphics supports many procedural drawing methods

Object Class Relationships

- Most of the objects in the Java user interface share common behavior
  - Can draw themselves, can move to the front or back of the window
  - Have a size, color, etc., and can change these things
- Would be terribly inconvenient to have to implement this by writing the same methods over and over in each class
- Want to be able to specify generic behavior for GUI components once and reuse

Inheritance

- A key idea in object-oriented programming
- Concept: A new class can be defined as a variation on another one

```
public class Othello extends Panel { … }
```

  - Notion: an Othello object *is a* Panel object and, unless specified otherwise, it works exactly like a generic Panel – we say it *inherits* all of Panel's methods and instance variables
  - Can customize Othello objects by defining new versions of generic Panel methods in class Othello – the new methods in Othello are said to *override* the inherited methods
  - Can also define new instance variables in Othello

Inheritance vs Interfaces

- Key difference: inheritance provides code sharing – we inherit implementations from the base class
- Can only extend one class, but can also implement one or more interfaces

```
public class Othello extends Panel implements MouseListener { … }
```

- If we don't define a method that exists in the base class, we use the inherited one
- If we implement an interface, we must give implementations of all methods defined in that interface

Class Object

- Now we know the story behind Object
  - Every class implicitly extends Object
  - Implication: every object, regardless of its specific class, is also an instance of class Object
  - That's why container classes like ArrayList can hold Strings, Rectangles, etc., and why we need to use a cast to specify that the Object retrieved from a container has a more specific class

Starting a Program

- Native Java supports two ways to run a program
  - Applications – stand alone programs
  - Applets – programs that run in another environment, typically a web browser
- BlueJ allows us to create objects without knowing about such things, by hiding the details behind the scenes

Applications

- Every class can contain a static method main

  public static void main(String[ ] args) { … }
  - Must be declared exactly like that
- Any class with a main can be run as a main program
  - Command line interface for compiling and running a program

    javac Board.java

    java Board
- Some classes are designed as the starting point for a program
  - Create some objects, start things off by calling methods
- Other classes often contain a main method that serves as a test or demo program for the class

Applets

- An applet is any class that extends Applet

```
class Applet {
    // called when applet first loaded – initialize things as in a constructor
    public void init( ) { … }
    // called when applet becomes visible
    public void start( ) { … }
    // called when applet is no longer displayed
    public void stop( ) { … }
    // called when applet is deleted from the host window
    public void destroy( ) { … }
    …
}
```


Applets

- A web browser knows to load and start an applet when it renders html code that contains an applet tag (or something similar)

  <APPLET CODE="OthelloApplet.class" WIDTH=400 HEIGHT=400
  CODEBASE=.> </APPLET>
  - If the applet is loaded over the web, CODEBASE would contain a URL showing where to find it

Applets vs Applications

- Applications
  - Can do anything that a program running on the machine with the current user's permissions can do
- Applets
  - Normally run in a security "sandbox".
  - Restrictions: Can't read/write local files, can't open arbitrary web connections, can't do other nasty things….