CS1004: Intro to CS in Java, Spring 2005

Lecture #14: Java OO cont’d.

Janak J Parekh
janak@cs.columbia.edu

Administrivia

- Homework due Tuesday
- Midterm next Thursday
  - I don’t have a formal midterm review, but I’ll leave a little bit of next class, plus office hours right after class, for questions/discussion
  - Next class isn’t until 1:10pm, so we can hang out in the classroom for a while

Java modifiers, redux

- Actually, you can have private classes, but only if they’re “inner classes”, i.e., inside another class
- Constants frequently use the static keyword as well; what exactly does static mean?
  - public static final int NUM_SIDES = 6;
- You can also create static methods, just like the utility methods in the Math class
Finish circle example

- ... and square example
- We’re not going to worry about the GUI part (yet)

Graphical Applications

- Except for the applets seen in Chapter 2, the example programs we've explored thus far have been text-based
- Let’s examine some Java applications that have graphical components
- These components will serve as a foundation to programs that have true graphical user interfaces (GUIs)
  - Applets can use these, too

GUI Components

- A GUI component is an object that represents a screen element such as a button or a text field
- GUI-related classes are defined primarily in the java.awt and the javax.swing packages
- First major component: a container
  - A GUI container is a component that is used to hold and organize other components
  - A frame is a container that is used to display a GUI-based Java application
Frames and panels
- A frame is displayed as a separate window with a title bar – it can be repositioned and resized on the screen as needed
  - “Heavyweight”: managed by the underlying operating system
- A panel is a container that cannot be displayed on its own but is used to organize other components
  - “Lightweight”: managed by the Java program itself
- A panel must be added to another container to be displayed
  - But you can nest panels to form more sophisticated GUIs

Labels
- A label is a GUI component that displays a line of text
- Labels are usually used to display information or identify other components in the interface
- Let’s look at a simple example
  - This is not like g.drawString(); it’s an object-oriented approach to organizing text

Images
- Images are often used in programs with a graphical interface
- Java can manage images in both JPEG and GIF formats
- As we’ve seen, a JLabel object can be used to display a line of text
- It can also be used to display an image
  - The ImageIcon class is used to represent an image that is stored in a label
  - That is, a label can be composed of text, and image, or both at the same time
So how do we paint()?

- We can still make a paint method in a component, so that we can mix a structured GUI interface along with custom elements
- We extend a JPanel and put a paintComponent(...) method inside it
- Other GUI constructs (like a JLabel) already have useful paintComponent implementations, so you rarely put one explicitly in there
- Note that we can draw on a panel or put stuff in the panel

Events

- An event is an object that represents some activity to which we may want to respond
- For example, we may want our program to perform some action when the following occurs:
  - the mouse is moved or dragged
  - a mouse button is clicked
  - a graphical button is clicked
  - a keyboard key is pressed
  - a timer expires
- Events often correspond to user actions, but not always

Events and Listeners

- The Java standard class library contains several classes that represent typical events
- Components, such as a graphical button, generate (or fire) an event when it occurs
- A listener object "waits" for an event to occur and responds accordingly
- We can design listener objects to take whatever actions are appropriate when an event occurs
GUI Development

- Generally we use components and events that are predefined by classes in the Java class library
- Therefore, to create a Java program that uses a GUI we must:
  - instantiate and set up the necessary components
  - implement listener classes for any events we care about
  - establish the relationship between listeners and components that generate the corresponding events

Buttons

- A push button is a component that allows the user to initiate an action by pressing a graphical button using the mouse
- A push button is defined by the JButton class
- It generates an action event
- Let’s set up a quick example

Flow of Control

- As we discussed earlier, code usually runs linearly
- We can affect this flow of control in one of two ways
  - Conditional operation: decide whether or not to execute a particular statement
  - Iterative operation: execute a statement over and over, repetitively
- These decisions are based on boolean expressions
Conditional Statements

- A conditional statement lets us choose which statement will be executed next.
- The Java conditional statements are:
  - if statement
  - if-else statement
  - ? operator (well, not quite a statement)
  - switch statement
- Less “clumsy” than the assembly equivalents

The if Statement

- The if statement has the following syntax:

  ```java
  if (condition)
  statement;
  ```

  - `if` is a Java reserved word.
  - The condition must be a boolean expression. It must evaluate to either true or false.
  - If the condition is true, the statement is executed.
  - If it is false, the statement is skipped.

Next time

- Continue chapter 5 of L/L
- Midterm review
- Today’s class is the last material for the midterm