Slide 1

Administrivia

- Reminder: Homework 5 design due today, code Tuesday.

Slide 2

Java GUI Libraries

- Java being an evolving language, it has two groups of GUI-related classes:
  - Abstract Window Toolkit (AWT) — older, “look and feel” consistent with platform’s windowing system.
  - Swing — newer, more extensive, look and feel more aimed at being consistent across platforms. Makes use of AWT components.
- Many, many classes to build GUIs:
  - GUI elements —- buttons, labels, text boxes, menus, etc., etc., etc., etc.
  - “Containers” to group elements and arrange them for display.
  - “Listeners” and “events” to allow program to respond to user input.
- Programs are “event-based” or “event-driven”, can seem a little different from traditional text-in/text-out programs.

Slide 3

Some GUI Classes

- `Component` — base class.
- `Container` – component that can contain other components.
- `JFrame` — window with titlebar, etc.; usually the "main" window for an application.
- `JDialog` — popup dialog box.
- `JPanel` – very simple container, useful for grouping things, providing custom graphics.
- `JMenuBar`.
- Etc., etc., etc., etc. — far more than we can cover in this course! Read the API. Some classes have links to online tutorials too.

Slide 4

Using the GUI Classes — Appearance

- When using predefined components, key issue is how they're grouped into container and how things are laid out within each container.
- Preferred method is to use a layout manager — places elements in some reasonable way, does something reasonable if container is resized.
  - Simple layouts include `FlowLayout`, `GridLayout`, `BorderLayout`, `BoxLayout`.
  - `GridBagLayout` provides more control, but is more complex.

  Some of them expand components to fit, others lay them out at their minimum size. See API and tutorials for more info.
- Often makes sense to group elements hierarchically — `JPanel` is useful for that.

Slide 5

Using the GUI Classes — Behavior

- Runtime system (JVM) translates each user action (keyboard or mouse input) into an "event" and then calls method(s) in "event listener" objects.
- So, to tell the runtime system what should happen when, e.g., a `JButton` is clicked, call button's `addActionListener` with an object `listener` that implements `ActionListener` interface. Now when the button is clicked, `listener`'s `actionPerformed` method is called.
- Several approaches to defining listener objects. One is to have "main" class implement required interface. Another is to use anonymous inner classes.
- Example(s) as time permits . . .

Slide 6

Minute Essay

- None — quiz.