Slide 1

Administrivia

- Reminder: Homework 3 design due today, code Tuesday after break.

Slide 2

GUIs and Event-Driven Programming

- In PAD I (and in most previous in-class examples this semester) we usually focus on programs with simple text-based input and output — a basically synchronous interaction with the user.
- Programs with GUIs, though, are typically somewhat different — the main program (which is sometimes hidden in library code) is often just a loop that waits for keyboard/mouse input delivered by the program's environment (operating system, graphical environment, window manager, etc.).
- This leads to an "event-driven" programming model that can seem rather different from what's used for text-based programs.

Slide 3

Java GUI Libraries

- Java has evolved over its short lifetime, and sometimes there seems to be more than one way to do something. One example — resizable arrays (`Vector` versus `ArrayList`). Another — 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.

Slide 4

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 5

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 for laying things out — layout manager, which 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 6

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.

Slide 7

Java GUI Classes and Multithreading

- Currently Java GUI classes are implemented in terms of an “event dispatch thread” (EDT) — something that listens (to some part of the operating system/environment?) for “events” (from keyboard or mouse, e.g.) and “dispatches” them by calling appropriate methods associated with GUI components.
- Not all of what’s under the hood is thread-safe, so Sun recommends that all changes to GUI components be done in the EDT. This happens automatically with listener methods. Accesses from the “main” thread and from other threads should use `SwingUtilities.invokeLater`.

Slide 8

Examples

- (Examples as time permits.)

Multithreading and the Game Framework

Slide 9

- Listener methods run in the EDT. Other methods run in a different thread.
- Problem? Maybe. Concurrent access to simple primitive types (`boolean`, `int`) is pretty safe — the worst that's likely to happen is that changes made by one thread aren't immediately visible to others. But anything involving more complicated data structures is probably a bad idea without explicit synchronization.

Minute Essay

Slide 10

- How did the midterm compare to your expectations? with regard to length, difficulty, topics, or whatever?
- (And best wishes for a good spring break!)