Slide 1

Administrivia

- Reminder: Homework 5 due soon (design today, code next Tuesday).

Slide 2

Java GUI Libraries — Recap

- Many, many classes for GUI components — pre-defined components (e.g., `JButton`), containers (e.g., `JPanel`).
- How things are arranged on screen is controlled by “layout manager”. Can nest containers, giving them different layout managers.
- How things work depends on “event listener” methods. Good place to use anonymous inner classes.
- (Review/rework short example.)

Slide 3

Java GUI Libraries — Design Tips

- Probably better not to mix AWT and Swing unless necessary (e.g., unless you're doing an AWT-only program, prefer `JFrame` to `Frame`).
- To find out how to use components — skim online API, Sun tutorials (follow links from API), look for examples similar to what you want to do.
- For small programs, okay to put GUI and underlying data all in one class. For larger programs, consider separating them — "Model/View/Controller" design pattern.
- GUI components that must be accessed by more than one method — e.g., by listener methods — should be instance variables. Other components can often be declared locally in constructor.

Slide 4

Sidebar: Multithreading

- (We will talk more about multithreading later. A little now, so some things about the GUI classes make more sense.)
- Idea of threads is to have multiple things happening "at the same time" — what operating systems texts call concurrency.

  (Why the quotes? How many things can actually happen at the same time? How can we — or something — fake it?)
- Typically threads share access to the same memory/variables. Convenient but potentially risky. (Why?)

  Methods/functions are called "thread-safe" if they can be called by different threads at the same time without potential problems.

Slide 5

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`. (Rework example to make this true for construction of components.)

Slide 6

Multithreading and the Game Framework

- 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 — probably a bad idea without explicit synchronization (more about that later).

Slide 7

Java GUI Libraries — Other Gotchas

- Local variables used in / passed to anonymous inner classes must be `final`. (Apparently this is because the class is passed a snapshot of these variables, and it's not clear that makes sense if they're not immutable.)

Slide 8

Example(s)

- Revisit example(s) from last time (slightly updated).
- Then let's write a simple calculator program . . . .

Minute Essay

- The game framework will allow you to add panels to any or all four sides of your game. You can display info (text is easiest) or include GUI components for additional user input (e.g., click a button to speed up the player). You can also add to the menu bar.

  How might this be helpful for your game?