Slide 1

Administrivia

- Reminder: Homework 6 due . . . Due date was today, changed to Thursday by request.
- One more homework.
- No Quiz 6.
- Homeworks 1 through 5 graded and grades sent by e-mail. Midterms graded and to be returned next time. (If you want to know your score right away send me mail.)

Slide 2

GUIs and Event-Driven Programming — Recap

- Up to now our programs have all interacted with their environment in a fairly primitive and synchronous way.
- Programs with GUIs, though, are typically somewhat different — interaction is more complex and also asynchronous ("event-driven" programming model). Typically significant parts of control flow are done in library code; you provide setup code and code to respond to events (including user input).
- In writing programs, often useful to think in terms of "what should the program's interface look like?" (appearance) and "how should the program respond to user actions/inputs?" (behavior).

Slide 3

GUIs in Scala — Code

- Code for building GUIs in Scala uses several things that may not make 100% sense right now. More exposure/experience (e.g., in CS2) should help. For now a good strategy may be to start from working example and tinker.
- In particular, libraries make use of:
  - Scala/Java package naming conventions.
  - Object-oriented syntax and ideas, including subtypes.
  - Curried functions and pass-by-name (compare to `Array.fill`).
  - Partial functions (compare to `match`).

Slide 4

GUIs in Scala — Appearance

- Scala provides many many library components programs can use as building blocks (and also provides a way to define your own components).
- Key idea — components are grouped hierarchically using “containers”, and Scala provides different kinds of containers that arrange components differently (e.g., as a rectangular grid, or “flowed” as text is flowed in a paragraph).

Slide 5

GUIs in Scala — Behavior

- For components that can interact with the user (e.g., buttons), Scala provides different mechanisms for programs to say what should happen when the user does something (clicks a button, presses a key, etc.).
- Key ideas: Programs respond to 'events". Scala models this in terms of "publishers" (that generate events — e.g., a button generates button-pressed events) and "reactors" (that respond to events).
- Simplest kind of interaction is that defined for buttons and menu items — one possible kind of event, and Scala provides a simple way to specify what code will be executed when that event happens. For other kinds of events, you specify (based on the publishers/reactors model) what kinds of events you want to be notified about and what you want to do when they occur.

Slide 6

Graphics in Scala — Custom Panels

- Predefined components do a lot, but what if you want something that's not provided? in particular, you want to control the image yourself?
- Make a custom component — in Scala, a `Panel` that contains code that says what should be drawn for this component. Must also explicitly ask the runtime system to redisplay when something changes.
- Possibilities for what to draw are, well, extensive! graphics library provides ways to draw a lot of things, including but not limited to simple shapes and images from files.

Slide 7

GUIs in Scala — Timers

- Last but not least, Scala provides mechanisms for specifying that something should happen repeatedly, at timed intervals.
- This makes animations relatively straightforward.

Slide 8

Examples

- “Hello world”.
- GUI “sampler” program with buttons, menu, list.
- (More examples next time.)

Slide 9

Minute Essay

- None — quiz.