Slide 1

Administrivia

- (None.)

Slide 2

Minute Essay From Last Lecture

- Most people mentioned insertion sort or (a few) selection sort.
- One person mentioned another way: Start by identifying values that occur a lot and group them. Similar ideas behind “bucket sort” and “tally sort”.

Slide 3

GUIs and Event-Driven Programming

- Up to now our programs have all interacted with their environment in a fairly primitive way — getting text input from standard input and files and writing text output to standard output and files, and interacting with the human user in a basically synchronous way.
- Programs with GUIs, though, are typically somewhat different — the main program (which is often 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. Rather than defining the whole interaction in the way we've been doing, you typically write code that defines (often in terms of library components) what appears on the screen and how each component responds to user input.

Slide 4

GUIs in Scala

- Java (which is what's under Scala's hood in some sense) defines not one but two libraries that provide functionality for building GUIS — predefined components such as buttons and checkboxes and menus, plus frameworks/mechanisms for laying things out and defining interaction with users. (Why two? Historical reasons.)
- Scala gives you access to all of that if you want it, and also provides a less verbose syntax for using some of the more-commonly-used things.
- 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).

GUIs in Scala — Appearance

Slide 5

- 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 into “panels”, and Scala provides different kinds of panels that arrange components differently (e.g., as a rectangular grid, or “flowed” as text is flowed in a paragraph).

GUIs in Scala — Behavior

Slide 6

- 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 7

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 8

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 9

Examples

- (Simple examples as time permits.)

Slide 10

Minute Essay

- None — quiz.