

Minute Essay From Last Lecture
(Many answers that seemed to go off on tangents.)
How people sort things, though — most popular choice seemed to be something like "bucket sort", with insertion sort a distant runner-up.
(FYI — code from last time is insertion sort, not selection sort! Corrected on sample programs page.)

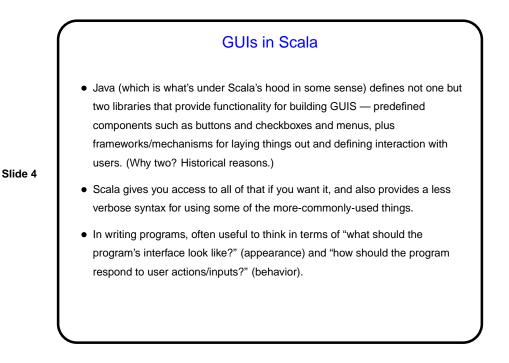
Slide 2

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.

Slide 3

- 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.



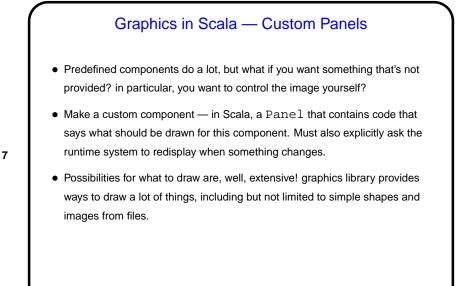
GUIs in Scala — Appearance

- 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 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).

Slide 5

Slide 6

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 7

