Slide 1

Administrivia

- Homework 7 on the Web. Due next Tuesday.

Slide 2

Minute Essay From Last Lecture

- Several people thought “errors” meant programming errors. I meant user errors. For the former — the only thing that helps is practice, as far as I know!

Slide 3

Sidebar: Command-Line Arguments in Scala

- You've noticed that many commands (`vi`, `scala`, etc.) can be followed by additional text? this is one more way of getting input to a program — *command-line arguments*.
- In Scala, you can access these arguments via an array (of `String`s) called `args`.
- Often a good way to specify things such as filenames.

Slide 4

A Little About Errors in Scala (Exceptions)

- You've noticed that some kinds of user-input error result in rather ugly program crashes. If you wonder why, and what to do about it . . .
- The "why": If writing programs meant to teach you about programming, has its advantages, as compared to other languages, which sometimes allow you to continue even when it wouldn't make sense to do so. (Think about what happens if you prompt the human for an integer and he/she enters something else.)
- The "how": Crashes come from Scala's main mechanism ("exceptions") for signalling that something has gone wrong.

  A short discussion of errors in general, then . . .

Slide 5

Errors in Programs

- Some errors in programs are caused by programmer mistakes — e.g., trying to access an element of an array using an index that's out of bounds (in Scala, negative, or more than the array size minus 1).
- Other errors have external causes — e.g., input that's not what was intended, or files that can't be found.
- What to do about errors is something to decide when designing a programming language.
- Different languages use different approaches:

  Some try to detect and warn about programmer mistakes (safer but possibly inefficient); others don't.

  All(?) try to detect and do something about external-causes mistakes, but what they do varies.

Slide 6

Errors in Programs, Continued

- Some example errors:
  - Accessing an array element with an out-of-bounds index.
  - Getting square root of a negative number, with a function that returns a `Double`.
  - Converting text to an `Int`, when the text doesn't represent an integer.
  - Opening a file when there is no such file.
- Some things the programming language could do:
  - Ignore the error (only for programmer mistakes).
  - Return a "didn't work" value (not always possible).
  - Set a global variable somewhere (ugly).
  - Bail out of normal program control flow via *exception*.

Slide 7

A Little (More) About Errors in Scala

- Scala uses exceptions to signal most kinds of errors, including some programmer mistakes. When some kinds of errors are detected, the code that detects them (e.g., `toInt` or `fromFile`) "throws an exception".
- By default, this causes the program to crash, with error messages that are meant to be helpful to the programmer — but probably will baffle or annoy an end user.
- If you want some other behavior, you can "catch the exception".

  A very bad idea for programmer mistakes; a very *good* idea for other errors. (And I usually put a lot more emphasis on this, but this semester with this textbook I haven't.)

  Details in POP II, but for now a short example . . .

Slide 8

Checkbook Example Revisited/Continued

- Let's (finally!) extend checkbook program to work with file of saved transactions.
- Notice that by keeping information in a text file we *can* modify it with a text editor. (What's the downside of this?)

Slide 9

Minute Essay

- None — quiz.