Administrivia

- Homework 5 to be on Web tomorrow; due after holiday.

Slide 1

Files — Overview

- One of the things that's useful about computers is their ability to store large amounts of information in a form that they can process — i.e., the ability to store and work with *files.*

- "File" is a pretty broad and generic term and includes everything from simple text files (such as the ones that contain your Scala programs) to word-processing documents and images and digital representations of music and video and . . .

Slide 2

- Up to now, our ability to work with files has been limited to what I/O redirection provides — which is useful, but very limited since we can only work with one source and one destination. Most programming languages provide something more general.

Sidebar: Packages in Scala

- Before talking about working with files in Scala, useful to know a little about *packages*.

- Basic idea of packages is to provide some way to organize lots and lots of code: Languages may include extensive libraries. Real applications typically involve quite a lot of code. How to organize? One way is to somehow group related functionality. Scala (and Java) does this using packages. Idea is similar to folders/directories for organizing files.

- Packages also provide a nice mechanism for avoiding naming collisions — i.e., names of things (such as `List`) don't have to be unique across everything in the library and your own code, only within a package.

Slide 3

Sidebar: Packages in Scala, Continued

- You may notice that when you type an expression into the interpreter, it tells you its type, and sometimes the type is something simple (e.g., `Int`) but sometimes it's less scrutable — e.g., for a range (such as `0 to 5`) it's
`scala.collection.immutable.Range.Inclusive`
The lower-case parts identify the "package" containing the library code for ranges.
You could use this whole name as the type for a function parameter, but that's unwieldy, so . . .

- `import` gives you a way to tell the Scala compiler/interpreter where to look for things it couldn't otherwise find. (The above isn't the best example because everything in `scala.collection` is automatically imported.)

Slide 4

Files in Scala

Slide 5

- Simplest way to read files in Scala is with `scala.io.Source` (or just `Source` with an `import scala.io.Source`):

  `Source.fromFile("somefile")`

- This gives you back something that the interpreter claims is an "iterator". What's that ...

Sidebar: Iterators

Slide 6

- For arrays and lists you know it's sometimes useful to be able to go through every element of the array/list and do something (print it, or add it to a running total, e.g.). It's useful to be able to do that with other kinds of collections too, (e.g., lines in a file).

- Abstract term for something that lets you "visit" each element of a collection — *iterator*. As used in Scala/Java, it's something with two operations, "is there another element?" and "give me the next element".

- Something to know about iterators — not necessarily reusable (so must be somehow reset or recreated if you want to go through the collection more than once).

Files in Scala, Continued

Slide 7

- What you get back from `fromFile` is an iterator over the characters of the file, and you can apply to it lots of the methods you use on arrays and lists.

- To read a line at a time — `getLines`, which gives you an iterator over `String`s.

- After using a file, good practice to "close" it (free up any resources used to manage it).

Files in Scala, Continued

Slide 8

- Other ways of working with input files, and all ways of working with output files, use the underlying Java libraries.

- Two simple ones — `Scanner` for input, `PrintWriter` for output. Simple example:

```
import java.io.File
import java.io.PrintWriter
val pw = new PrintWriter(new File("out.txt"))
pw.println("hello world")
pw.close
```

(More examples in textbook.)

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 `Strings`) called `args`.

Slide 9

- Often a good way to specify things such as filenames.

Example

- Example, loosely based on the game Scrabble: Given a file containing a list of words and a list of letters, search the file for all words that can be formed with the given letters.

- Basic strategy: Scan through the whole file, looking at each word, and print the ones that can be formed from the given letters.

Slide 10

Slide 11

Minute Essay

- None — sign in.