Files

3-7-2012

Opening Discussion

- Minute essay comments
  - Would things make more sense without the power outage?
  - An Array can be treated as a vector.
  - Are there packages for matrix math? (Scalalab)
  - My reactions to the Slate articles.
  - Send code when missing an IcP.
  - <- is read as “in”
  - How was SIGCSE?
  - Can Scala do 4-D matrices?

More

- More matrices?
- Most common loop?
- Accessing higher dimensional arrays.
- Extra credit on other exam?
- When is the final?
- Why use print instead of println?
- I don't think Japan has an obesity problem.

The API

- To get a sense of the different package in Scala, it is helpful to look at the API.
- There are still lots of things in the API you won't fully understand. That isn't a problem as you aren't expected to get too much from it right now.

scala.io.Source

- Call Source.fromFile(fileName:String) to get a Source object that reads from a file.
- There are other methods in the main Source object that we will learn about later.
- The fromFile method technically gives you BufferedSource. This is for efficiency.

Iterators

- Both Source and BufferedSource are of the type Iterator[Char].
- An Iterator has most of the methods you are used to from List and Array. However, you can only go through it once.
- Fundamentally uses hasNext and next methods.

getLines

- This will give you an Iterator[String] that will go through the file one line at a time instead of a character at a time.
- You will often find this more useful.

java.util.Scanner

- Java Scanner class sometimes easier for input.
  - hasNext(), next()
  - hasNextInt(), nextInt()
  - hasNextDouble(), nextDouble() ...
- Doesn't produce a Scala collection.
- Needs java.io.File: new Scanner(new File(fileName))
- Always make a new object from the Java libraries using new. Scala typically allows you to leave that off.

Closing Files

- Make sure you always close files when you are done using them.
- Source, Scanner, and pretty much anything else that pulls from a file will have a close() method.

java.io.PrintWriter

- To write to files use java.io.PrintWriter.
- Create with new PrintWriter(fileName)
- Has print and println methods just like what you have been using to print to screen.

Flush

- To make certain contents have been written to the file use flush().
- Doing close() will also flush and you should definitely remember to close all files you are writing when done with them.

Command Line Arguments

- In a script, the command line arguments are put in args:Array[String].
- You can do anything with them that you would do with a normal array.

Examples

- Print some random numbers to a file.
- A file copy with word replace.

Minute Essay

- What questions do you have?
- IcP #5 on Friday (note this is moving back a class).