Slide 1

Administrivia

- Reminder: Homework 7 design due today, code Thursday.
- Reminder: Quiz 6 Thursday.

Slide 2

I/O In Java — Overview

- Abstract view — “file” is a collection of data. Java provides methods for sequential and “random” (non-sequential) access.
- Sequential file access is via “streams” — concept that applies to other kinds of sequential I/O (stdin/stdout, sockets, etc.).
- Stream — sequential flow of data.
  - Input streams connect program with an outside “source” (stdin, file, socket, etc.). (If data is characters, use “reader” instead.)
  - Output streams connect program with outside “destination”. (If data is characters, use “writer” instead.)

Slide 3

Stream I/O

- I/O in Java often requires at least two classes:
  - One that connects to the desired source/destination (file, socket, array, string, etc.).
  - One that defines interface for program (character or binary data, byte-by-byte or a line at a time, etc.)
- Short examples:

```
BufferedReader rdr =
  new BufferedReader(new FileReader("in.txt"));
String s = rdr.readLine();

PrintWriter pw =
  new PrintWriter(new FileWriter("out.txt"));
pw.println("hello, world");
```

Slide 4

I/O and Exceptions

- Many I/O methods throw "checked" exceptions — which your code must explicitly do something about. Sensible but sometimes annoying.
- First example from previous page would not compile — either declare that the method it's in throws an `IOException`, or use a "try" block, e.g.,

```
try {
  BufferedReader rdr =
    new BufferedReader(new FileReader("in.txt"));
  String s = rdr.readLine();
}
catch (FileNotFoundException e) {
  System.err.println(e); // or better error message
}
catch (IOException e) {
  System.err.println(e); // or better error message
}
```

Slide 5

Character-Based Stream I/O

- Prior to Java 1.5, typical way to parse input was to read a line at a time and use `String` methods, `Integer.parseInt`, `Double.parseDouble`, etc. `StringTokenizer`, `StreamTokenizer` also sometimes useful.
- Now, `Scanner` class may do what you need. `split()` method of `String` class may also be useful.
- For output, `PrintWriter` methods will likely be useful. Notice that Java also has (as of 1.5) a `printf`!
- (Example — "almost an editor" program(s).)

Slide 6

Binary Stream I/O

- Can also read/write binary data:
  - `DataInputStream`, `DataOutputStream` to write out primitive types.
  - `ObjectInputStream`, `ObjectOutputStream` to write out primitives, `Serializable` objects.
- Object serialization:
  - Object and all referenced objects (except `static` and `transient` variables) are turned into sequential stream of bytes.
  - Can override `readObject`, `writeObject` to control what happens more precisely.
- (Example — "silly class" and saver.)

Slide 7

Minute Essay

- Try writing code to count the lines of a file containing character data. (No need to make a complete class or method.)

Slide 8

Minute Essay Answer

- One way:

```
BufferedReader rdr =
  new BufferedReader(new FileReader("whatever"));
String line;
int lines = 0;
while ((line = rdr.readLine()) != null)
    ++lines;
```