Slide 1

## Administrivia

- Reminder: Homework 6 code due today. Remaining homeworks will be on the Web soon (tomorrow?), due around the time of our final.

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.)
- Also many useful utility classes for working with files — `java.io` package, `javax.swing.JFileChooser`.

Slide 3

## Stream I/O

- Most I/O in Java 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 InputStreamReader(System.in));
String s = rdr.readLine();

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

Slide 4

## Character-Based Stream I/O

- Parsing text input — `String` methods may be useful, also `StringTokenizer`, `StreamTokenizer`, `Integer.parseInt`, `Double.parseDouble`, etc.
- Examples — “almost an editor” (adapted from textbook).

Slide 5

## 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.
- Example . . .

Slide 6

## Serialization

- 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.
- Many uses — for saving objects in a file, for sending them over a network connection, etc.
- Useful, but as you know also a bit risky, since if the class changes, files containing serialized versions of old class don't work any more. `serialVersionUID` can help some.
- Example . . .

**Slide 7**

## Minute Essay

- None — quiz.