

Files and Streams



4-20-2004

Opening Discussion



- Do you have any questions about the quiz?
- What did we talk about last class?
- Do you have any questions about the assignment?

Motivation



- One of the most important things we do on computers is store and access large collections of data. Typically this is done with files.
- File access comes in two flavors, random and sequential. Files of the latter type are often called streams. In a stream the basic operation is to get or put the next byte of data, though more elaborate wrappers can be put around that.

java.io Package



- The normal way of doing I/O in Java is with the classes in the java.io package.
- This package has an elaborate class hierarchy with different classes that play the different roles for almost everything you want to do.
- There are also some special classes that perform specific tasks like the RandomAccessFile class.

InputStreams and OutputStreams



- The most basic classes in java.io are the InputStream and OutputStream classes. These are the base classes for dealing with streams of bytes.
- Let's look in the documentation to see the methods of these classes. The most significant ones are the read and write methods though the others can be important for different tasks.

Streams vs. Readers/Writers



- The stream classes handle reading and writing bytes. For text data it can be easier to read and write character data. This functionality is provided by the Reader and Writer classes.
- Personally, I've never really seen the need for these classes, but if you do text input it could be helpful.

Plentiful Subclasses



- All of these classes have multiple subclasses to give you more specific abilities. We can look at these in the docs.
 - File versions for I/O with files.
 - Piped versions for connecting different streams.
 - Buffered streams for better speed.
 - Data and object streams we will discuss next class.

Basic Text Input?



- One thing that you might notice is missing is the ability to do basic text input. We can do text output with a `PrintWriter`, but there is no equivalent for input in Java.
- This design decision was based on the idea that programs rarely need to do general text file reading. `BufferedReader` allows reading lines of text that can be parsed.

The File Class



- One other helper class in java.io is the File class. This class represents a specific file and allows us to get information about files. It is written in a way to be largely platform independent.
- This class also gives us the basic functionality that we would like to have when interacting with files.

JFileChooser



- For programs that use files, it is often nice to bring up a GUI component to let the user pick a file. This can be quite a pain. Java makes it easy by providing a class that automatically views and selects files.
- By simply creating and “showing” one of these, we can very easily have the user specify a file for our program to work with.

Code



- Let's write a simple little text editor program that uses a GUI and allows us to edit text files.
- We will also use some File objects even though we could avoid them.

Minute Essay



- Why is inheritance used so much in the java.io package? How might having it work that way help you in your programming?
- Remember that design #7 is due on Thursday.