R·I·T

Computer Science II
4003-232-08 (Winter 2007-2008)

Week 4: Files Continued

Richard Zanibbi
Rochester Institute of Technology

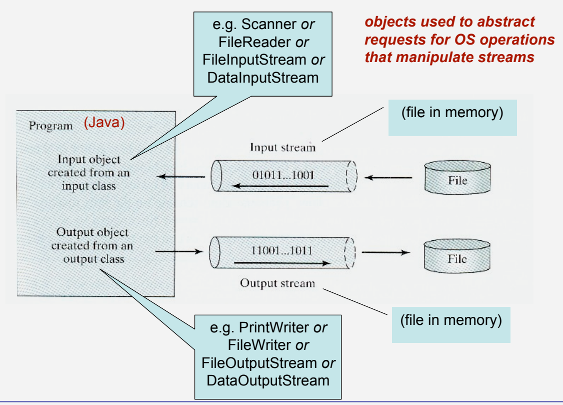
Input and Output Streams

(Byte) Streams

– Represented in memory as a sequence of bytes (low level file); no "real" notion of type of data represented
– Used to store data read from or to be written to system devices (e.g. physical file on disc, monitor, keyboard, pen/stylus input)
– In Java: represented and utilized through objects (most found in java.io package)

Default Streams for Programs in Most OS's

Standard input (System.in) – usually from keyboard.
Standard output (System.out) – usually sent to terminal.
Standard error (System.err) – usually sent to terminal also


e.g. Scanner or FileReader or FileInputStream or DataInputStream
objects used to abstract requests for OS operations that manipulate streams
Program (Java)
Input object created from an input class
Input stream
01011...1001
File
(file in memory)
Output object created from an output class
11001...1011
File
Output stream
(file in memory)
e.g. PrintWriter or FileWriter or FileOutputStream or DataOutputStream

"Low Level Files" = Byte Stream Files

File Pointer

On file open:
(default) pointer set to 0

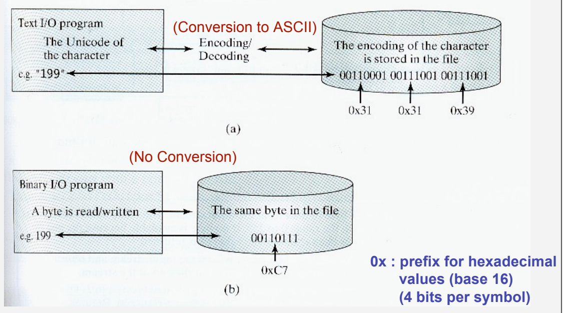
Read/Write K bytes:
Advance pointer by K

Setting File Pointer:
`skip()`,`mark()`,`reset()` (*Java*)
`lseek()` **(POSIX)**
`SetFilePointer()` **(Win)**

No "Type" for Bytes:
Treated as "raw" bytes

Name: Test		(ASCII)
Byte	Value	
0	0100 0001	A
1	0100 0010	B
2	0100 0011	C
3	0100 0100	D
4	0100 0101	E
5	0100 0110	F
6	0100 0111	G

Text vs. Binary I/O



0x : prefix for hexadecimal values (base 16) (4 bits per symbol)

Java Byte Stream Classes (Binary I/O)

See Fig 18.3 (p. 608) for inheritance hierarchy

main (*abstract)* classes:

InputStream, OutputStream

(summary: pp. 608-609)

***** All methods declared to throw java.io.IOException (IOException) or one of its subclasses (e.g. FileNotFoundException)***

FileInputStream, FileOutputStream

FileInputStream (see Fig. 18.6):

Produces an input stream from a file on disk. Constructors take a File object or a string giving the path to the file.

FileOutputStream (see Fig. 18.7):

Produce output stream (file) that will be stored on disk. Constructors also take a File object or path string, but have an alternate form that takes a boolean flag indicating whether to overwrite ('delete') or append data to the end of the file.

'FileStream' Example

TestFileStream.java (p. 610 in text)

Unix: to view file contents, can use *od (octal dump)*

e.g. od –Ad –tx1 temp.dat
- Shows file contents, one byte at a time in hexadecimal (with decimal numbering for bytes in the file)

od –Ad –a temp.dat
- Shows file contents, interpreted as (named) characters (ASCII)

od –Ad –t x1 –a –c temp.dat
- Show file contents in hexadecimal, as named characters, and as ASCII characters (including escape sequences)

"Filtered" Streams (FilterInputStream, FilterOutputStream)

Purpose

Converting bytes to other data types in java (for use with streams)
– Syntax: done through "wrapping" another class around an existing stream

FilterInputStream, FilterOutputStream
– Used to convert primitive types and Strings to and from bytes
– Subclasses of interest: DataInputStream (see Fig 18.9), DataOutputStream (see Fig 18.10)
  - These classes write Java variables directly to a byte stream (quietly making the necessary conversion), or convert byte stream data to Java variable primitive (e.g. int, double) or String types (e.g. to store in a variable).
– Methods for the conversions are defined by the *DataInput* interface.

'DataStream' Example

TestDataStream.java (p. 613, text)

Buffered Streams in Java (also Filtered Streams)

Buffering
– Using additional intermediate storage (a 'buffer') to allow *reading ahead* and *writing behind*
– Reduces the number of (actual) read and write operations needed, improves performance
– Default buffer size is 512 bytes (can be changed using a constructor)

Java Syntax (Example; mod. TestDataStream.java)

```
DataOutputStream output = new DataOutputStream(new
  BufferedOutputStream(new FileOutputStream("temp.dat")));
DataInputStream input = new DataInputStream(new
  BufferedInputStream(new FileInputStream("temp.dat")));
```

Example of Buffering

Copy.java (p. 615)

Standard Streams in Java

System.out

Is a PrintStream reference (subclass of FilterOutputStream)

System.err

Also a PrintStream reference

System.in

Is an InputStream reference

Reader and Writer Classes *(abstract):* Alternate Classes to Read, Write Text Files in the java.io Package *(see Java API)*

PrintWriter

Is a subclass of *Writer*
Has same interface as *PrintStream* (e.g. as used by System.out)
Can be used to write to a

FileReader

Is a subclass of *Reader* used to read text files

BufferedFileReader

– Subclass of *Reader*
– Use a buffer for read/write operations for concrete *Reader* instances (e.g. *FileReader*)
  - BufferedFileReader in = new BufferedReader(new FileReader(“foo.in”))

Exercise: Files

Part A

1. Can the File class be used for I/O? If not, what is it used for?
2. What is the (parent) type of checked exception thrown by classes in the java.io package?
3. How must this be handled within a method using these operations? (Hint: two possibilities)
4. Write java statements to create a PrintWriter object for a file “foo.txt” in the current directory, write “hello” to the file, and then close the file.
5. What sources can a Scanner object read data from? How does this differ for a FileInputStream?
6. What kind of data ‘unit’ are Scanners normally used to recover?

Part B

1. What is stored in a stream? Where do streams exist?
2. What three streams exist for all programs in a modern OS?
3. What type of data do FileInputStream and FileOutputStream support?
4. What types of data do DataInputStream and DataOutputStream support?
5. What is buffering? Why is it used?
6. Write Java statements to construct a buffered FileInputStream for “foo.txt” (current dir.), read a byte, and close the stream.