Redirecting Standard Output, Error, and Input (bash shell)

Redirecting Standard Output

java Streams > output

Redirecting Standard Input

java Streams 2> error

Redirecting Both Together or Separately

java Streams &> output (one file)

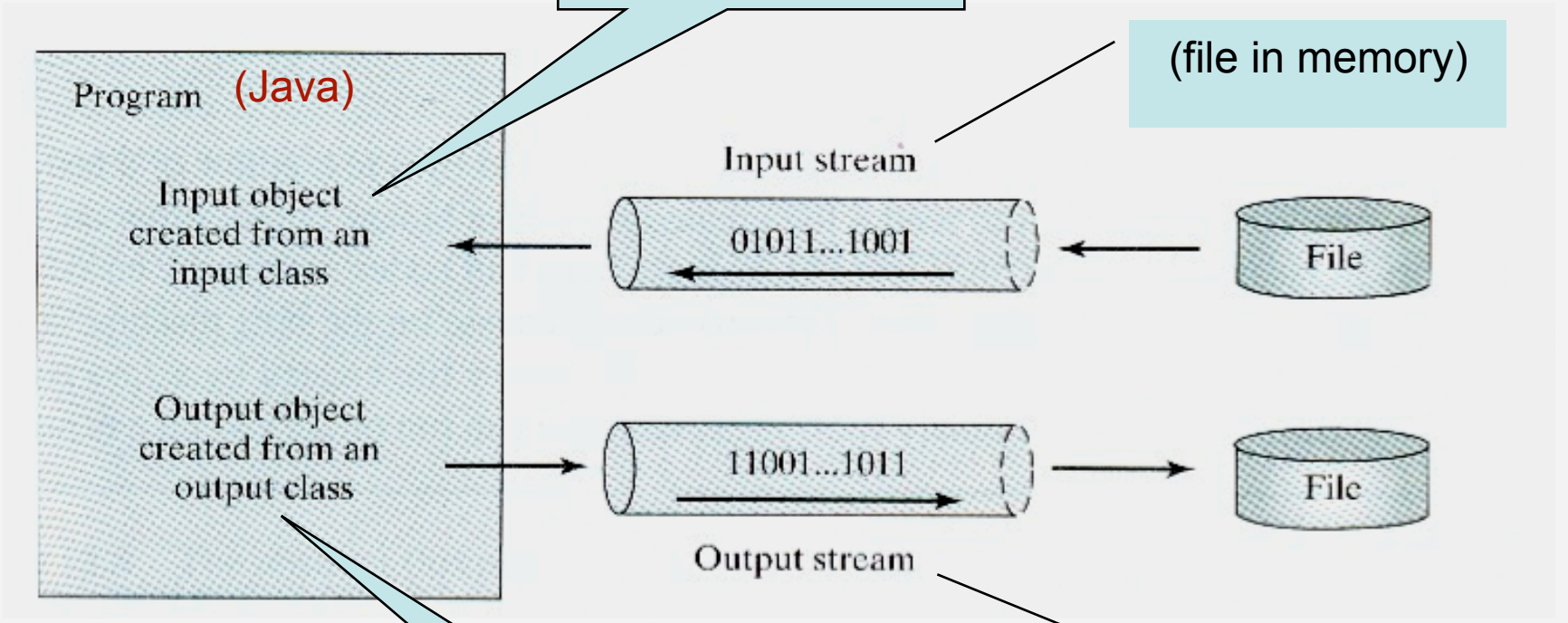
java Streams > output 2> error

Redirecting Standard Input

java Streams < textfile

e.g. Scanner *or*
FileReader *or*
FileInputStream *or*
DataInputStream

objects used to abstract requests for OS operations that manipulate streams

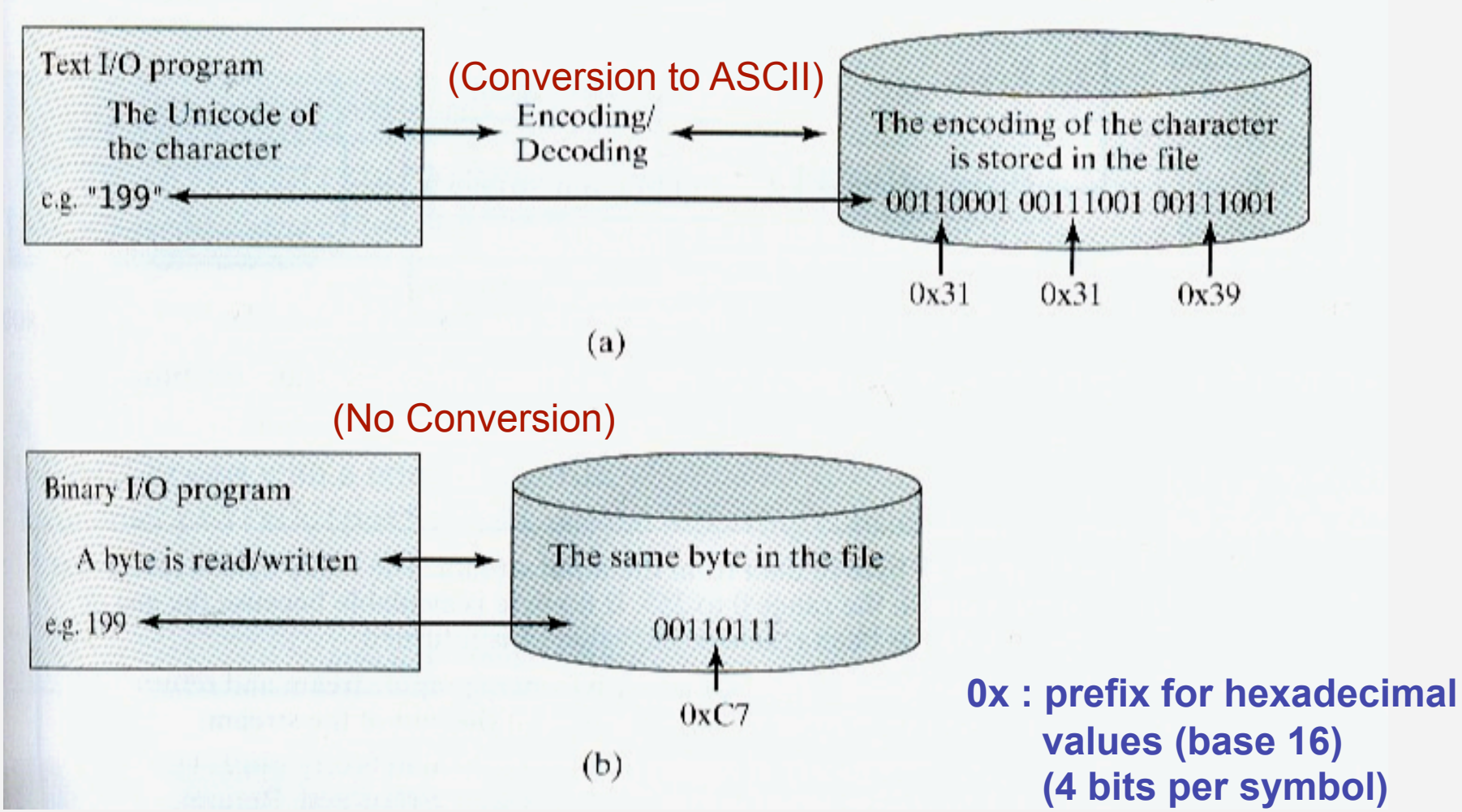


(file in memory)

(file in memory)

e.g. PrintWriter *or*
FileWriter *or*
FileOutputStream *or*
DataOutputStream

Text vs. Binary I/O



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

Viewing File Contents on the Command Line

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

od –Ad –tx1 *file*

- Shows file contents, one byte at a time in hexadecimal (with decimal numbering for bytes in the file)

od –Ad –c *file*

- Shows file contents, interpreted as (ASCII) characters

od –Ad –tx1 –c –a *file*

- Show file contents in three different representations simultaneously:
  - hexadecimal
  - named characters
  - ASCII characters (using escape sequences)

Java Byte Stream Classes (Binary I/O)

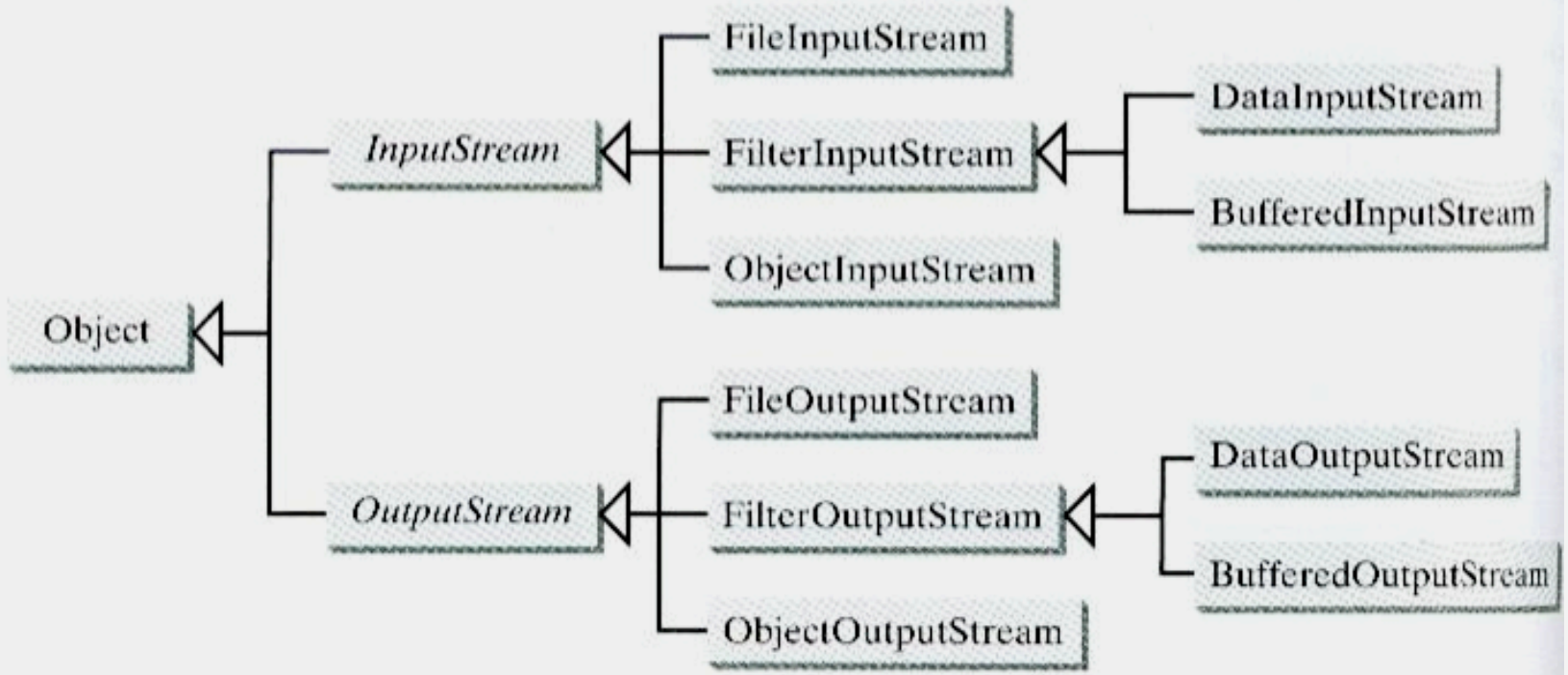
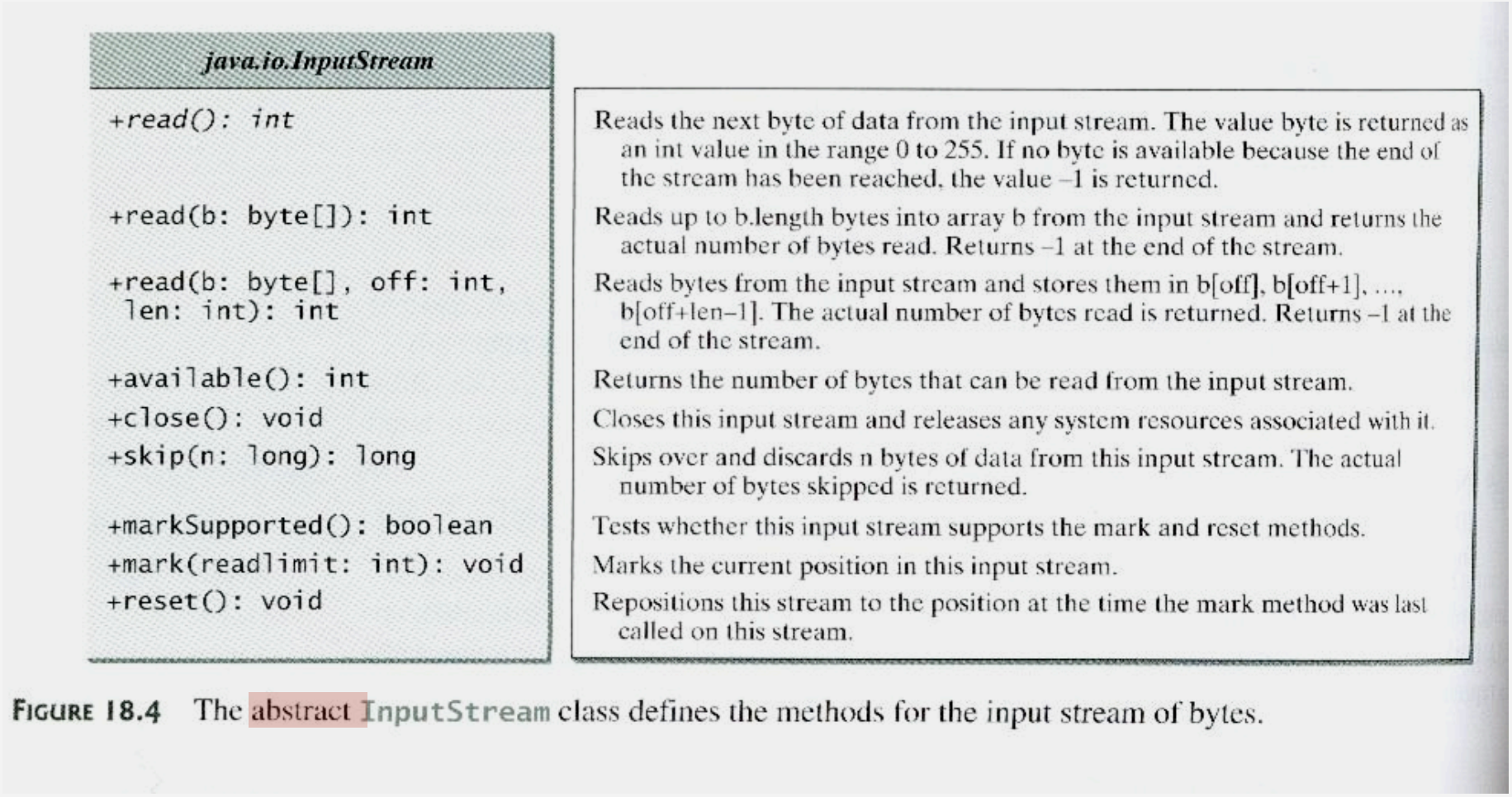


FIGURE 18.3 `InputStream`, `OutputStream`, and their subclasses are for binary I/O.

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

InputStream *(abstract)*: Reading bytes and arrays of bytes



java.io.InputStream	
+read(): int	Reads the next byte of data from the input stream. The value byte is returned as an int value in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value –1 is returned.
+read(b: byte[]): int	Reads up to b.length bytes into array b from the input stream and returns the actual number of bytes read. Returns –1 at the end of the stream.
+read(b: byte[], off: int, len: int): int	Reads bytes from the input stream and stores them in b[off], b[off+1], ..., b[off+len–1]. The actual number of bytes read is returned. Returns –1 at the end of the stream.
+available(): int	Returns the number of bytes that can be read from the input stream.
+close(): void	Closes this input stream and releases any system resources associated with it.
+skip(n: long): long	Skips over and discards n bytes of data from this input stream. The actual number of bytes skipped is returned.
+markSupported(): boolean	Tests whether this input stream supports the mark and reset methods.
+mark(readlimit: int): void	Marks the current position in this input stream.
+reset(): void	Repositions this stream to the position at the time the mark method was last called on this stream.

FIGURE 18.4 The abstract `InputStream` class defines the methods for the input stream of bytes.

OutputStream *(abstract):* Writing bytes and arrays of bytes

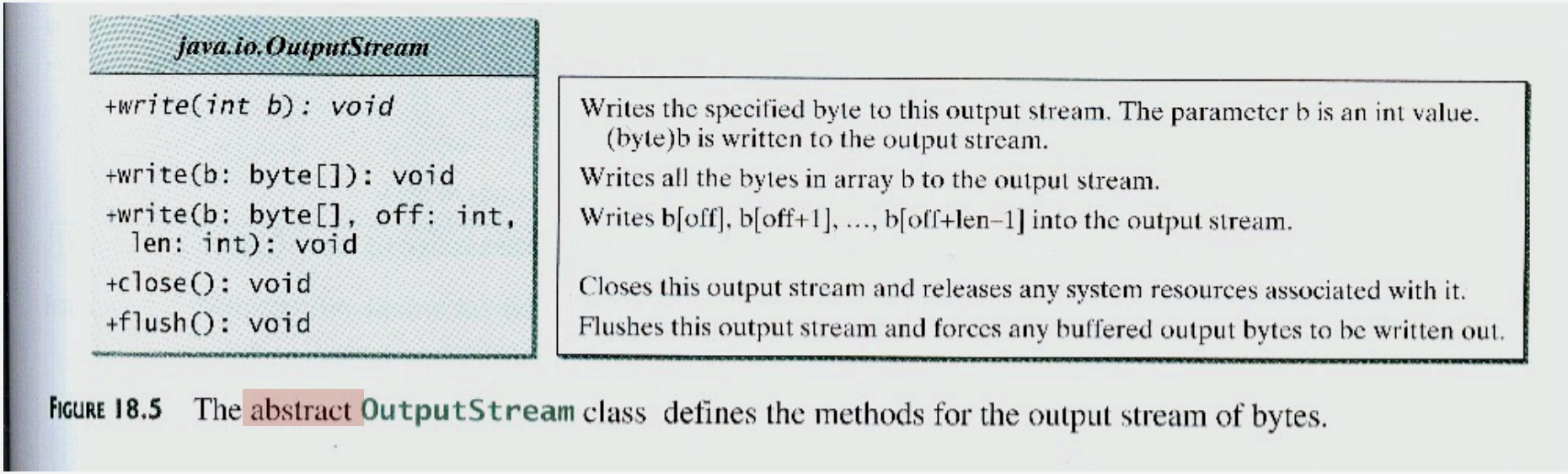


FIGURE 18.5 The abstract `OutputStream` class defines the methods for the output stream of bytes.

FileInputStream, FileOutputStream

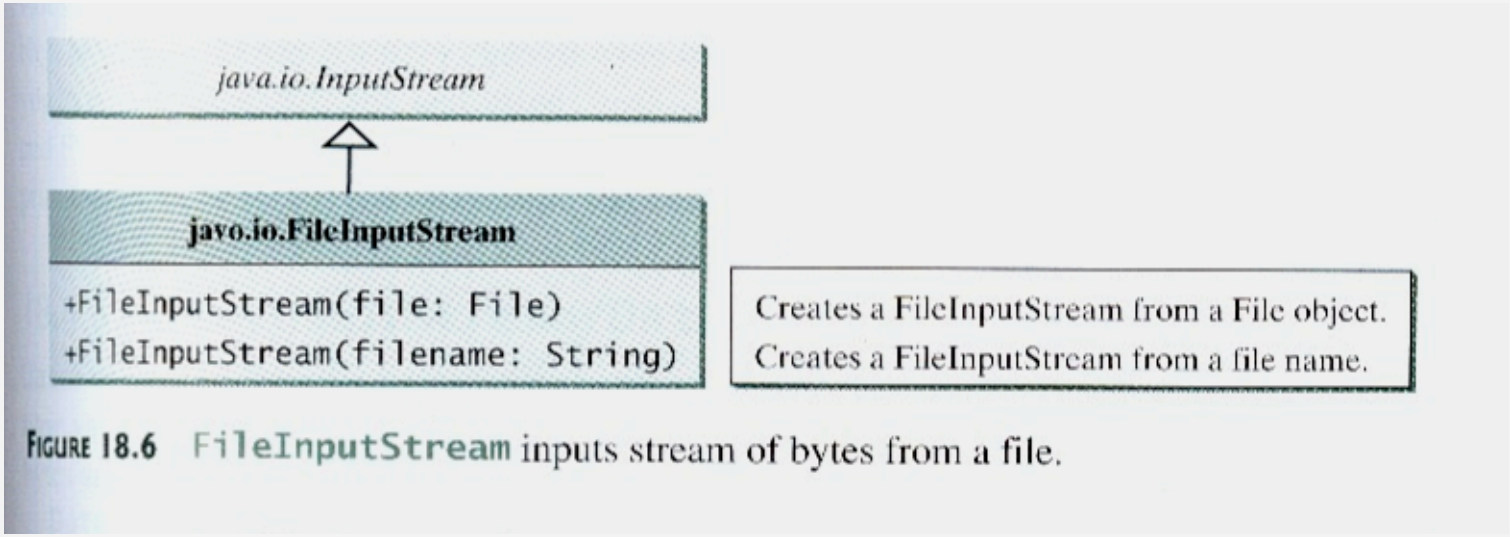


FIGURE 18.6 `FileInputStream` inputs stream of bytes from a file.

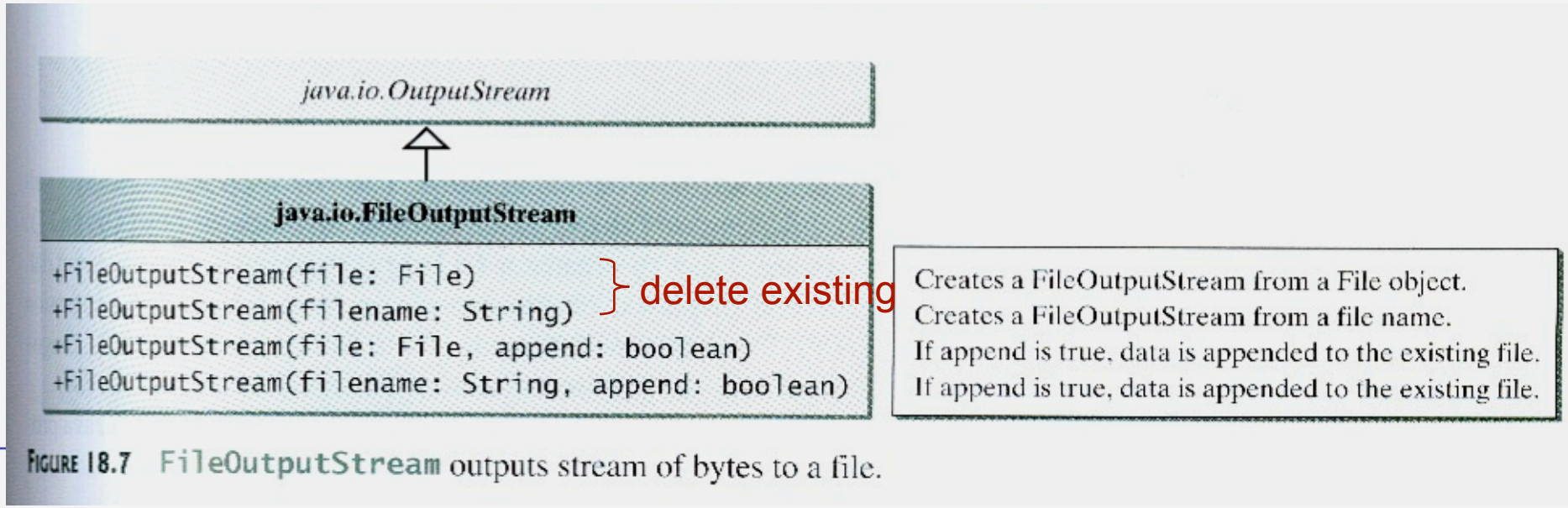


FIGURE 18.7 `FileOutputStream` outputs stream of bytes to a file.

Filtered Streams

Purpose

Converting bytes to other data types

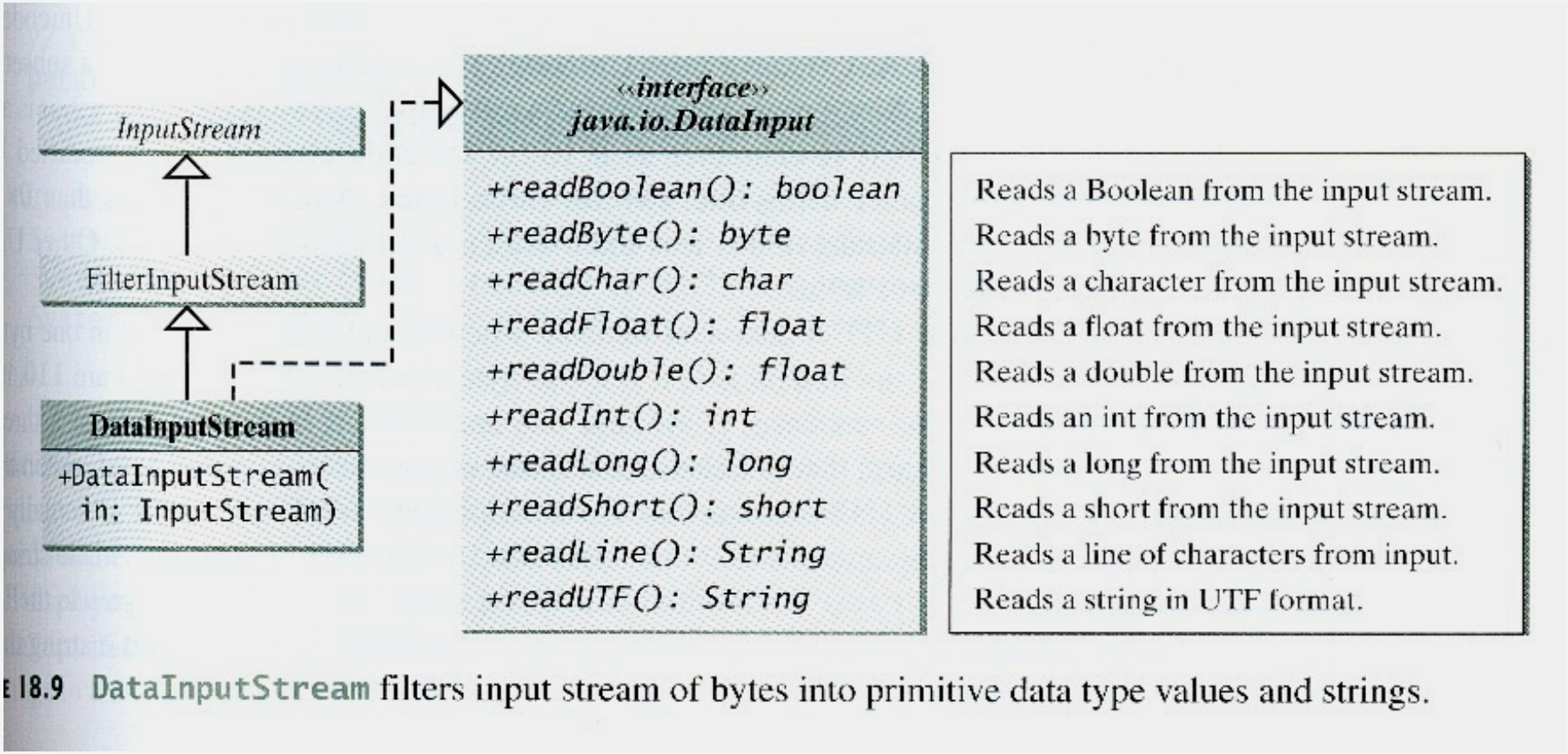
– Syntax: done through “wrapping” another class around an existing stream
  – “Wrapping:” constructor takes another object as input; use methods of “wrapping” object to access “wrapped” object
  – Methods in wrapping class simplify use, add functionality to “wrapped” object

FilterInputStream, FilterOutputStream

– Convert primitive types and Strings to and from bytes

– Subclasses of interest: DataInputStream, DataOutputStream

DataInputStream: convert byte stream to primitive data type values



18.9 DataInputStream filters input stream of bytes into primitive data type values and strings.

DataOutputStream: write primitive data, strings as bytes

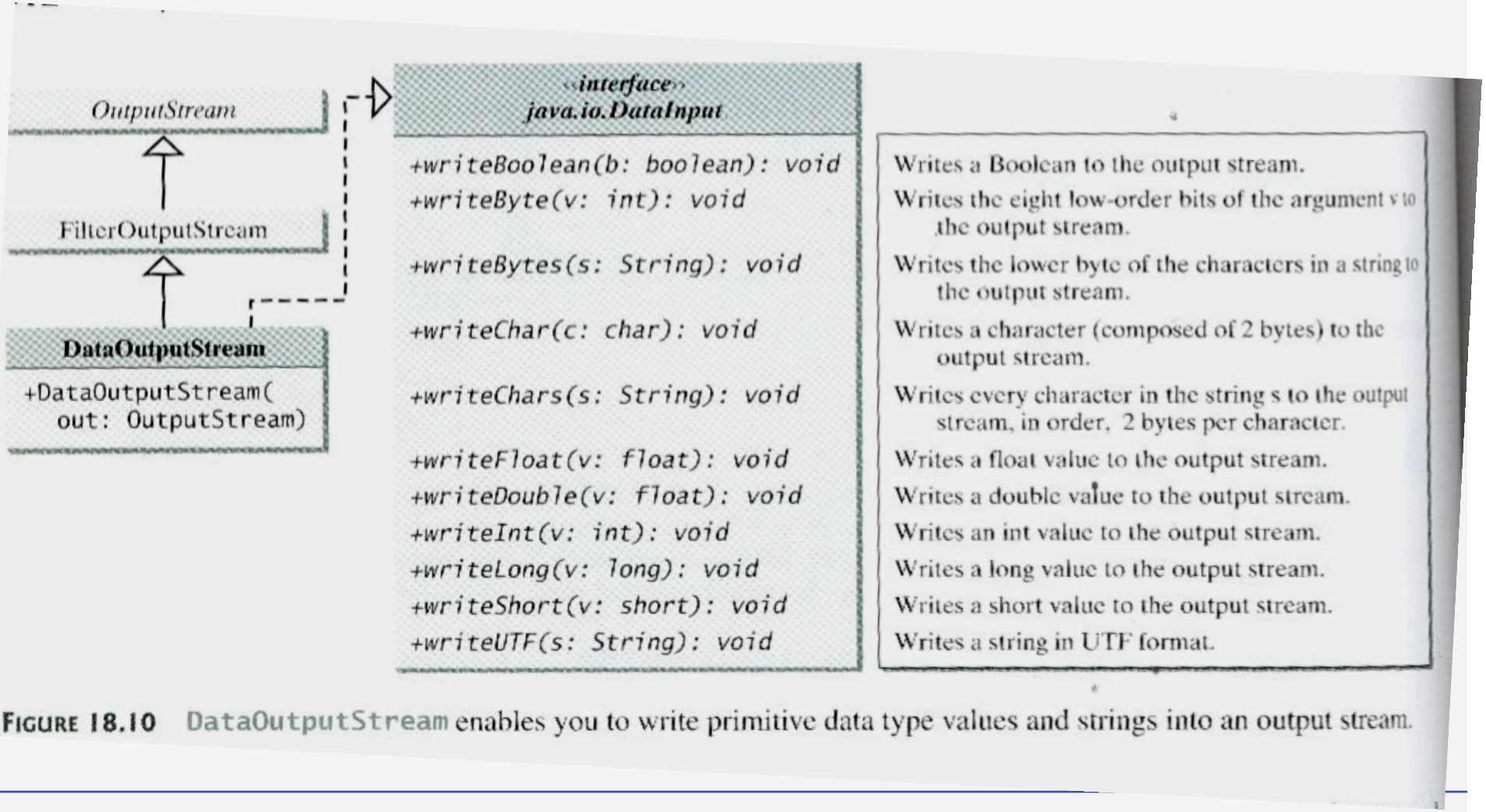


FIGURE 18.10 `DataOutputStream` enables you to write primitive data type values and strings into an output stream.

DataStream Example

TestDataStream.java

Buffered Streams in Java (also Filtered Streams)

Buffering

- Using additional intermediate storage (a 'buffer') to allow *reading ahead* and *writing behind*
- Reduces read and write OS operations needed (faster)
- Java: Default buffer size is 512 bytes
- Java: Wrapper objects define buffered read/write methods for streams (e.g. store data in buffer arrays)

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")));
```

Buffering Example

Copy.java

Design/Style Notes

-Standard error used for error messages

-Separate method for checking arguments defined, to make code easier to read

- return, not System.exit() used to terminate

- Files closed before exiting

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

FileWriter

Is a subclass of *Writer*

Used with write() methods to write data (e.g. Strings) to a text file

FileReader

Is a subclass of *Reader* used to read text files as characters (returning an integer) using read() methods; -1 returned for EOF

BufferedFileReader

- Subclass of *Reader*
- Provides a method to read a line of text and return a String (readLine())
- Use a buffer for read/write operations for concrete *Reader* instances (e.g. *FileReader*)
  - BufferedFileReader in = new BufferedFileReader(new FileReader("foo.in"))