



C06 Files

Java File IO
Streams
Standard IO
Buffering

What is a file?

A file is a collection of data on secondary storage (hard drive, USB key, network file server).

Data in a file is a sequence of bytes (integer $0 \leq b \leq 255$).

- The program reading a file must interpret the data (as text, image, sound, etc).
- Standard libraries provide support for interpreting data as text.

I/O streams

A stream is a standard abstraction used for files:

- A sequence of values are read.
- A sequence of values are written.

The stream reflects the sequential nature of file IO and the physical characteristics of the media on which files traditionally reside (e.g. tape or a spinning disk).

Other I/O (e.g., network, keyboard) is also typically accessed as streams.





I/O in Java: Byte streams

The classes `java.io.InputStream` and `java.io.OutputStream` allow reading and writing bytes to and from streams.

- Subclasses: `FileInputStream` and `FileOutputStream` for files.
 - Open the stream (create stream object)
 - Read or write **bytes** from the stream
 - Wrap operations in a **try** clause
 - Use **finally** to close the streams

I/O in Java: Character streams


To read/write text files, use `java.io.Reader` and `java.io.Writer` which convert between **bytes** and **characters** according to a specified encoding.

- Subclasses: `InputStreamReader` and `OutputStreamWriter`
- Subclasses `FileReader` and `FileWriter` (shortcuts for wrapping a `FileInputStream` / `FileOutputStream` in a `InputStreamReader` / `OutputStreamWriter`).

Text encoding

Each character is assigned a number.

Unicode defines a unique number (“code point”) for > 120,000 characters (space for > 1 million).

Encoding (UTF-8)		Font
Bytes	Code point	Glyph
0100 0101 (69)	69	
1110 0010 (226)	8364	
1000 0010 (130)		
1010 1100 (172)		

Buffering I/O

In traditional storage media, accessing a specific byte (point in a file) is time consuming:

Disk: ~2-10ms **SSD:** ~10-100 μ s **RAM:** ~100ns **Cache:** ~1-15ns

But reading a consecutive “block” at one time is not much more so. Hence, buffering is used to absorb some of the overhead.

- `BufferedReader` and `BufferedWriter` can be wrapped around other reader/writer (e.g., `FileReader` and `FileWriter`) to buffer I/O.
- To flush the buffer, call `flush()`, or close the file.

Terminal I/O

Three standard I/O streams:

- standard input: (usually typed) input to the program
- standard output: normal printed program output
- standard error: program error messages (not buffered)
- Available in Java as `System.in`, and `System.out` and `System.err`.

```
byte b = (byte) System.in.read();  
System.out.write(b);  
System.out.flush();  
System.err.write(b);
```