# The Java™ Tutorials

**Trail:** Essential Classes
**Lesson:** Basic I/O
**Section:** I/O Streams

*The Java Tutorials have been written for JDK 8. Examples and practices described in this page don't take advantage of improvements introduced in later releases.*

## Buffered Streams

Most of the examples we've seen so far use *unbuffered* I/O. This means each read or write request is handled directly by the underlying OS. This can make a program much less efficient, since each such request often triggers disk access, network activity, or some other operation that is relatively expensive.

To reduce this kind of overhead, the Java platform implements *buffered* I/O streams. Buffered input streams read data from a memory area known as a *buffer*; the native input API is called only when the buffer is empty. Similarly, buffered output streams write data to a buffer, and the native output API is called only when the buffer is full.

A program can convert an unbuffered stream into a buffered stream using the wrapping idiom we've used several times now, where the unbuffered stream object is passed to the constructor for a buffered stream class. Here's how you might modify the constructor invocations in the `CopyCharacters` example to use buffered I/O:

```
inputStream = new BufferedReader(new FileReader("xanadu.txt"));
outputStream = new BufferedWriter(new FileWriter("characteroutput.txt"));
```

There are four buffered stream classes used to wrap unbuffered streams: `BufferedInputStream` and `BufferedOutputStream` create buffered byte streams, while `BufferedReader` and `BufferedWriter` create buffered character streams.

### Flushing Buffered Streams

It often makes sense to write out a buffer at critical points, without waiting for it to fill. This is known as *flushing* the buffer.

Some buffered output classes support *autoflush*, specified by an optional constructor argument. When autoflush is enabled, certain key events cause the buffer to be flushed. For example, an autoflush `PrintWriter` object flushes the buffer on every invocation of `println` or `format`. See Formatting for more on these methods.

To flush a stream manually, invoke its `flush` method. The `flush` method is valid on any output stream, but has no effect unless the stream is buffered.

About Oracle | Contact Us | Legal Notices | Terms of Use | Your Privacy Rights

Copyright © 1995, 2017 Oracle and/or its affiliates. All rights reserved.

**Previous page:** Character Streams
**Next page:** Scanning and Formatting