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.

Byte Streams

Programs use *byte streams* to perform input and output of 8-bit bytes. All byte stream classes are descended from `InputStream` and `OutputStream`.

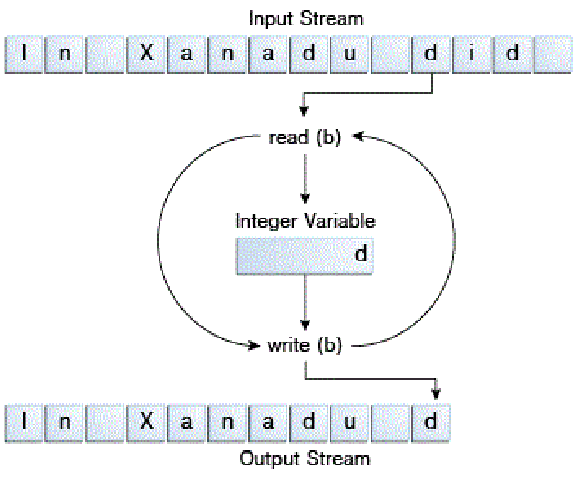
There are many byte stream classes. To demonstrate how byte streams work, we'll focus on the file I/O byte streams, `FileInputStream` and `FileOutputStream`. Other kinds of byte streams are used in much the same way; they differ mainly in the way they are constructed.

Using Byte Streams

We'll explore `FileInputStream` and `FileOutputStream` by examining an example program named `CopyBytes`, which uses byte streams to copy `xanadu.txt`, one byte at a time.

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class CopyBytes {
    public static void main(String[] args) throws IOException {

        FileInputStream in = null;
        FileOutputStream out = null;

        try {
            in = new FileInputStream("xanadu.txt");
            out = new FileOutputStream("outagain.txt");
            int c;

            while ((c = in.read()) != -1) {
                out.write(c);
            }
        } finally {
            if (in != null) {
                in.close();
            }
            if (out != null) {
                out.close();
            }
        }
    }
}
```

`CopyBytes` spends most of its time in a simple loop that reads the input stream and writes the output stream, one byte at a time, as shown in the following figure.



Simple byte stream input and output.

Always Close Streams

Closing a stream when it's no longer needed is very important — so important that `CopyBytes` uses a `finally` block to guarantee that both streams will be closed even if an error occurs. This practice helps avoid serious resource leaks.

One possible error is that `CopyBytes` was unable to open one or both files. When that happens, the stream variable corresponding to the file never changes from its initial `null` value. That's why `CopyBytes` makes sure that each stream variable contains an object reference before invoking `close`.

When Not to Use Byte Streams

`CopyBytes` seems like a normal program, but it actually represents a kind of low-level I/O that you should avoid. Since `xanadu.txt` contains character data, the best approach is to use character streams, as discussed in the next section. There are also streams for more complicated data types. Byte streams should only be used for the most primitive I/O.

So why talk about byte streams? Because all other stream types are built on byte streams.

Previous page: I/O Streams
Next page: Character Streams