Decorating the java.io classes

You can see that this isn’t so different from the Starbuzz design. You should now be in a good position to look over the java.io API docs and compose decorators on the various input streams.

And you’ll see that the output streams have the same design. And you’ve probably already found that the Reader/Writer streams (for character-based data) closely mirror the design of the streams classes (with a few differences and inconsistencies, but close enough to figure out what’s going on).

But Java I/O also points out one of the downsides of the Decorator Pattern: designs using this pattern often result in a large number of small classes that can be overwhelming to a developer trying to use the Decorator-based API. But now that you know how Decorator works, you can keep things in perspective and when you’re using someone else’s Decorator-heavy API, you can work through how their classes are organized so that you can easily use wrapping to get the behavior you’re after.