到这个时候，大家或许会陷入一种困境之中，怀疑是否存在 IO 流的另一种设计方案，并可能要求更大的代码量。还有人能提出一种更古怪的设计吗？事实上， Java 1.1 对 IO 流库进行了一些重大的改进。看到 Reader 和 Writer 类时，大多数人的第一个印象（就象我一样）就是它们用来替换原来的 InputStream 和 OutputStream 类。但实情并非如此。尽管不建议使用原始数据流库的某些功能（如使用它们，会从编译器收到一条警告消息），但原来的数据流依然得到了保留，以便维持向后兼容，而且：

(1) 在老式层次结构里加入了新类，所以 Sun 公司明显不会放弃老式数据流。

(2) 在许多情况下，我们需要与新结构中的类联合使用老结构中的类。为达到这个目的，需要使用一些“桥”类： InputStreamReader 将一个 InputStream 转换成 Reader，OutputStreamWriter 将一个 OutputStream 转换成 Writer。

所以与原来的 IO 流库相比，经常都要对新 IO 流进行层次更多的封装。同样地，这也属于装饰器方案的一个缺点——需要为额外的灵活性付出代价。之所以在 Java 1.1 里添加了 Reader 和 Writer 层次，最重要的原因便是国际化的需求。老式 IO 流层次结构只支持 8 位字节流，不能很好地控制 16 位 Unicode 字符。由于 Unicode 主要面向的是国际化支持（ Java 内含的 char 是 16 位的 Unicode），所以添加了 Reader 和 Writer 层次，以提供对所有 IO 操作中的 Unicode 的支持。除此之外，新库也对速度进行了优化，可比旧库更快地运行。