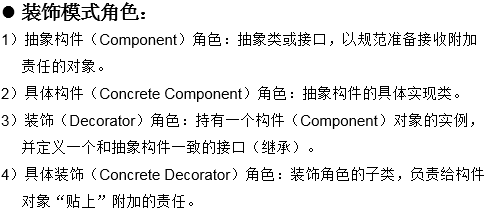
* 设计模式 - 装饰者设计模式（decorator）：

又叫包装器Wrapper模式，增强一个类的功能，而且还可以让这些装饰类互相装饰。



以对客户端透明的方式扩展对象的功能，是继承关系的一个替代方案，提供比继承更多的灵活性。动态给一个对象增加功能，这些功能可以再动态的撤消。增加由一些基本功能的排列组合而产生的非常大量的功能。



以后遇到类需要增强而且增强还不止一个功能的时候，就可以考虑一下使用继承来做 还是使用装饰者设计模式来做。如果有许多增强而且还能够互相装饰，建议使用装饰者设计模式来做。

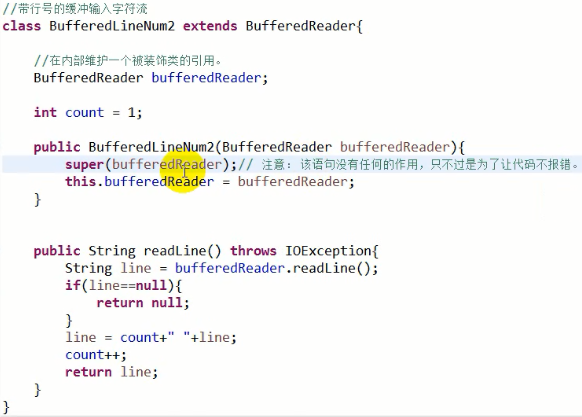


示例：对BufferedReader的readLine()方法增强。





如果使用继承来实现这三个需求，每实现一个需求都要写一个实现类。如果使用装饰者设计模式，只需要写两个类就可以了。

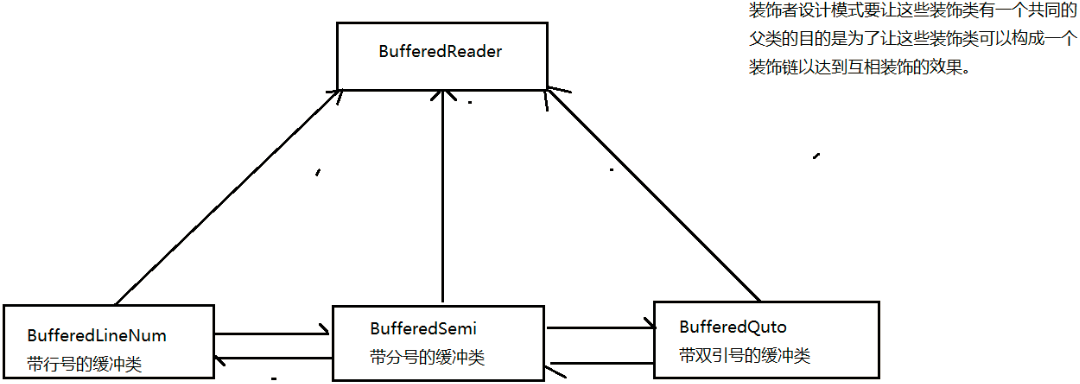


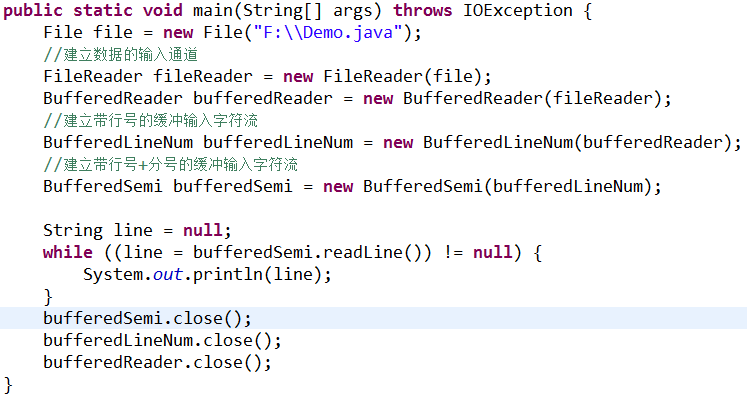


如上，两个类分别实现了readLine()方法的带分号和带行号的增强，并且让装饰类有一个共同的父类或父接口，当使用带分号的增强类时，只要构造方法中传入的共同父类为带有行号的增强子类，就可以实现第三个需求(行号+分号)的增强效果。

利用多态的思想，实现两个子类的对象互相传入，然后就可以调用传入对象的readLine()方法，实现行号+分号都有的结果。







* 为什么使用Decorator (装饰者设计模式)：我们通常可以使用继承来实现功能的拓展,如果这些需要拓展的功能的种类很繁多,那么势必生成很多子类,增加系统的复杂性,同时,使用继承实现功能拓展,我们必须可预见这些拓展功能,这些功能是编译时就确定了,是静态的.