1. 关于多项式的组织

    1. 对多项式本身 polynomial类
        1. 逻辑层次：（ 系数+次数）-> 项 -> 式子
        2. 功能：
            1. 多项式应有的各种操作
            2. 多项式格式化输出
        3. 对多项式的管理：式子+标识符 -> 多项式库，
            1. 以每个多项式对应的标识符作为索引去寻找对应的多项式
            2. map容器的优点：搜索，一对一的关系

2. 关于main函数的设计

    1. 用户交互：设计input函数和output函数，提高其可复用性
    2. 在交互过程即保证输入的合法性，确保以后正确处理

3. 技巧：关于字符串的匹配：正则表达式

缺点：

1. 关于操作封装
    1. 用户层面应该只涉及到string类的处理
        1. 比如说输入输出
    2. 同时在用户层面的交互中需要保证输入的合法性
        1. 多项式的输入格式合法
        2. 多项式的标识符合法且对应多项式存在
2. 处理string对象的polynomialCalculator类
    1. 作为用户与底层多项式沟通的桥梁。用户输入的字符串输入到polynomialCalculator类中，相应的输出也以string的形式输出。
    2. polynomialCalculator类还兼有将字符串转变为多项式类的功能
3. 最底层的Polynomail类，负责运算。

适配器模式

1. 底层核心的多项式类，可复用性高
    1. 数据存储的数据结构与用户输入无关
    2. 通过接入不同的适配器，可直接接入其他的计算器，扩展功能
2. 可拓展底层适配者的功能而不影响计算器，为以后的进一步扩展做准备
3. 目标类和适配者类解耦，增加了类的透明性和复用性，同时系统的灵活性和扩展性都非常好，更换适配器或者增加新的适配器都非常方便，符合“开闭原则”；
4. 最少知道原则

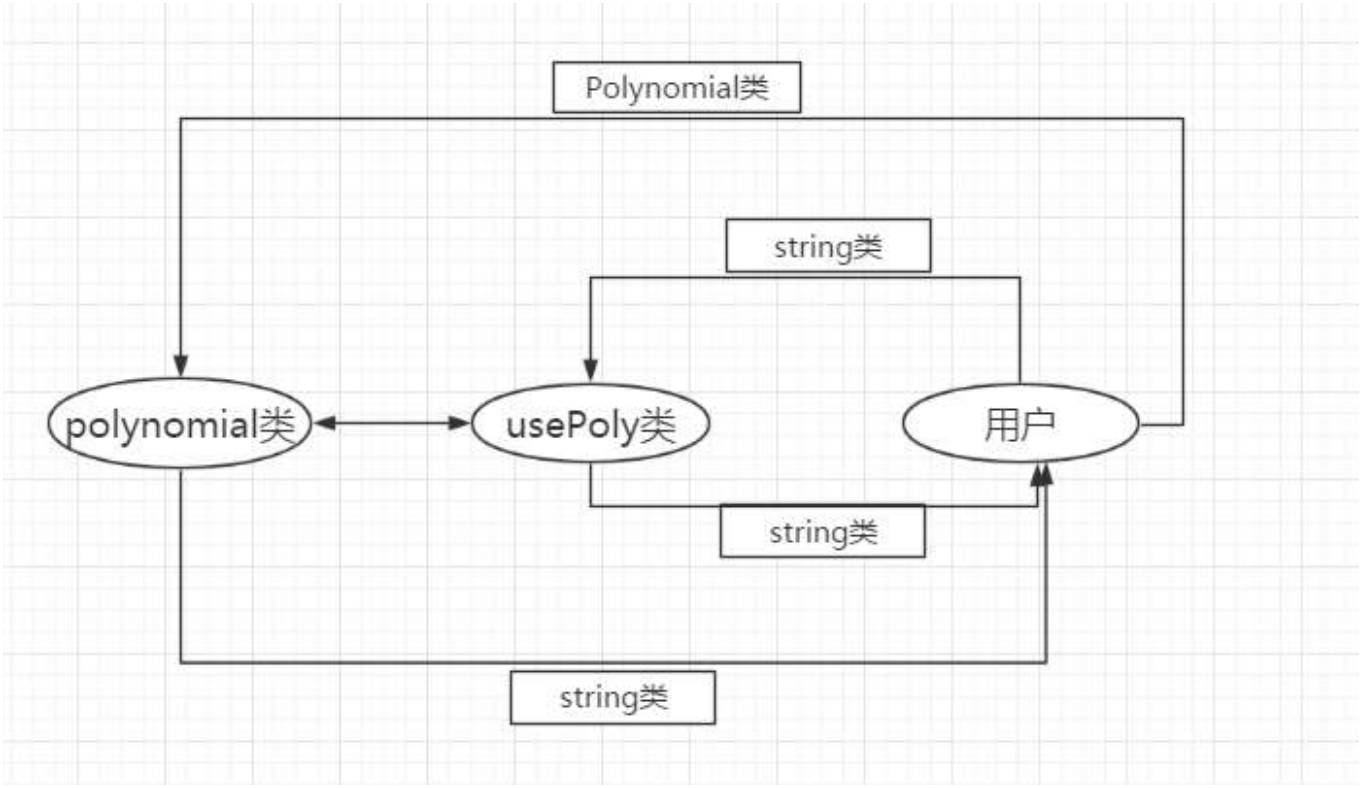
S 单一功能原则：对象应该仅具有一种单一功能。

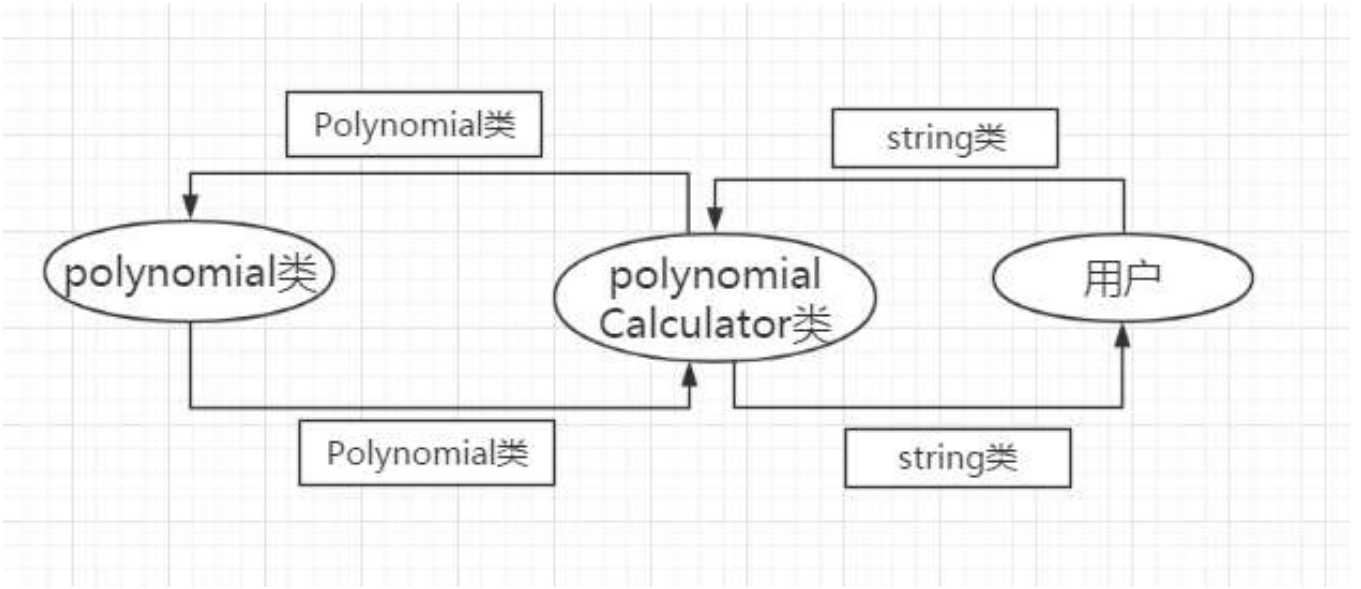
O 开闭原则：软件体应该是对于扩展开放的，但是对于修改封闭的。

L 里氏替换原则：程序中的对象应该是可以在不改变程序正确性的前提下被它的子类所替换的。

I 接口隔离原则：多个特定客户端接口要好于一个宽泛用途的接口。

D 依赖反转原则：一个方法应该遵从“依赖于抽象而不是一个实例”。


Polynomial类
string类
polynomial类
usePoly类
用户
string类
string类


Polynomial类
string类
polynomial类
polynomial
Calculator类
用户
Polynomial类
string类