1. **设计模式**

设计模式是软件开发过程中，经过验证的，解决在特定环境下，重复出现的特定问题的解决方案。适用于实际工程既有设计模式的特征，又具有其独特性，即修改少量代码适应需求变化。

静态多态：函数重载；动态多态：虚函数

1. **设计原则**

依赖倒置：高层不依赖底层，高层和底层都应该依赖于抽象接口，以免一方改动另一方也需改动

开放封闭：对外层扩展开放，对内部修改封闭（封装，多态）

面向接口：（封装）

封装变化点：（封装，多态），将变化点与稳定点分层次隔离处理

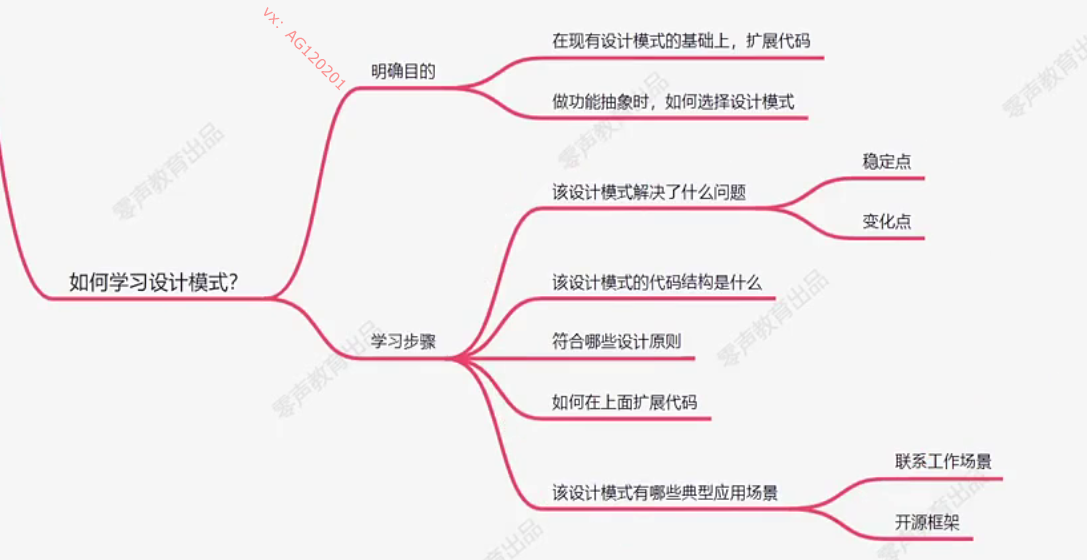
单一职责：（封装）

里氏替换：子类替换父类时，程序仍能正常运行（多态）

接口隔离：类与类之间的关系通过接口处理

组合优于继承：

最小知道原则：客户对对象内部知道的越少越好



友元可以访问友类的私有变量，子类不能访问父类的私有变量

稳定点通过抽象实现，变化点通过扩展代码实现

1. **模板方法**

代码骨架固定，根据实际需要实现子类，多态调用。例如表演流程固定，中间的节目内容可变

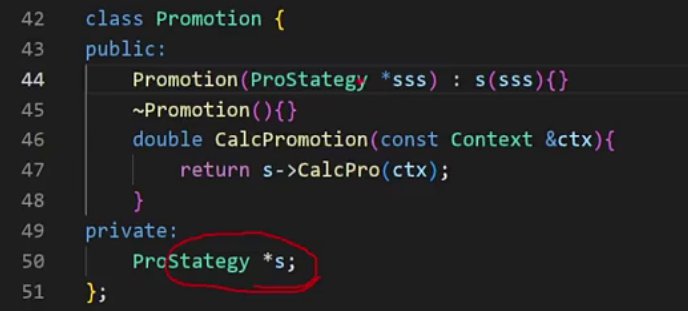
1. **观察者模式**

一对多的依赖关系，一个对象发生改变时，依赖于它的对象都会自动更新。稳定点：“一”和“多”的依赖关系；变化点：“多”的数量的增加或者减少，使用一个容器放置，提供接口增加（attach接口）或减少(detach接口)“多”的数量，例如气象台与各终端的数据更新

1. **策略模式**

定义一系列算法，把它们一个个封装起来，并且使他们可相互替换，使得算法独立于使用它的客户程序。稳定点：程序和算法的调用关系；变化点：算法变化

例如不同假日促销力度不同



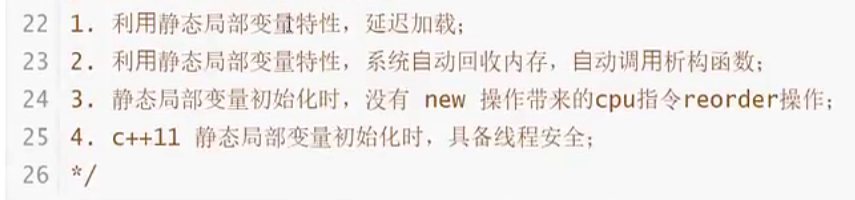
定义一个ProStategy接口，稳定点中注入并调用这个多态抽象接口，其他算法继承并实现这个接口

1. **单例模式**

定义：保证一个类只有一个实例，提供这个实例的全局访问点

需要禁止构造、析构、拷贝构造、移动构造、拷贝赋值和移动赋值函数（移动构造优先于拷贝构造）。

原子操作：load，获取其他线程最新修改过的数据；store，修改数据时保持其他线程可见；memory\_order\_acquire与memory\_order\_release之间的语句无法被CPU指令重排。



1. **工厂模式**

定义：定义一个用于创建对象的接口，让子类决定实例化哪一个类，让类的实例化延迟到子类。解决创建过程比较复杂，对外隐藏实现细节的问题。

稳定点：创建对象的接口，同一对象有一个相同职责，变化点：创建对象的扩展

例如：实现一个导出数据的接口，让客户选择如何导出数据

代码需要分别设计一个功能接口和一个对象创建接口

1. **抽象工厂**

与工厂模式相比，同一对象有多个相同职责

1. **责任链模式**

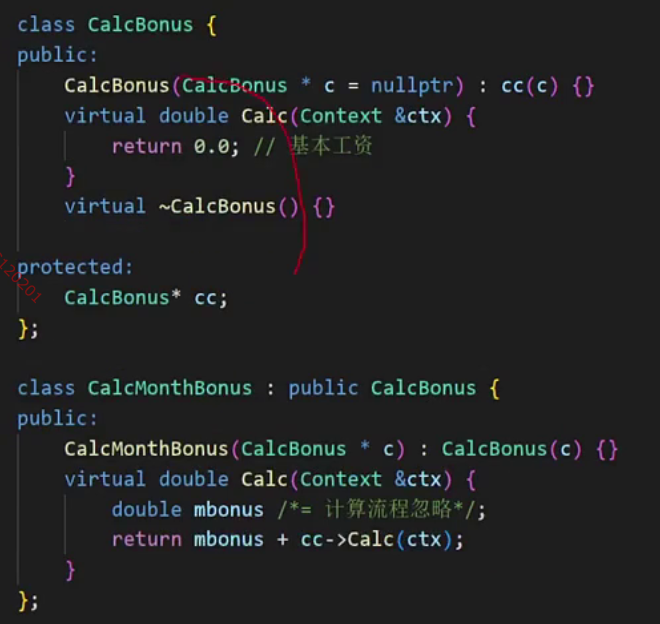
使多个对象都有机会处理清求，从而避免请求的发送者和接收者之间的耦合关系，将这些对象连成一条裤，并沿着这条链传递清求，直到有一个对象外理它为止。Nginx使用

稳定点：请求沿链传递，可被任一链中节点处理

变化点：链条中的节点内容及顺序可变

1. **装饰器模式**

动态地给一个对象增加一些额外的职责，就增加功能而言，装饰器模式比生产子类更为灵活。顺序无关地增加职责。



组合基类指针

1. **组合模式**

将对象组合成树型结构以表示“部分-整体”的层次结构，组合模式使得用户对单个对象和组合对象的使用具有一致性。

