控制反转，依赖注入

解耦合

想用任何方式记录操作日志都不需要去修改 User 类了，只需要通过构造函数参数传递就可以实现--“控制反转”

不需要自己内容修改，改成由外部传递。这种由外部负责其依赖需求的行为，我们可以称其为 “控制反转（IoC）”。

也算是依赖注入，不是由自己内部 new 对象或者实例，通过构造函数，或者方法传入的都属于 依赖注入（DI） 。

对象实例通过方法参数定义就能传递进来，不需要手动传入

单一职责原则，一个类应当只有一个职责

UML统一建模语言

strstr() 函数搜索字符串在另一字符串中是否存在，如果是，返回该字符串及剩余部分，否则返回 FALSE。



Include\_once(‘MobileSniffer.php’)

OOP和设计模式的主要作用之一就是能够改变单个模块而不会破快整个程序。

算法处理的是操作速度，而设计模式解决的是开发速度。

顺序编程-》过程编程（引入函数）-》OOP（引入类、对象）

顺序代码随着编程的复杂，开始互相纠缠，变成所谓“意大利面”式的代码。

有些时候，我们必须放弃原来的老习惯，提升我能的能力。

在某种程度上，所有的类都是对数据的一组操作的抽象。

抽象是用来出来复杂的主要工具，一个问题越复杂，就需要抽象来解决。

在OOP和设计模式中，抽象类可以为项目提供一种组织机制。

抽象类不能实例化，只能由具体类（可以实例化的类）继承抽象类的接口及他的所有具体属性。

如果一个类至少有一个抽象的方法，他必然是一个抽象类，不过，抽象类也可以有具体的方法。

有些将属性和方法都简单称为属性，但多数情况属性指变量和常量，方法指函数。

PHP有抽象类、方法，但没有抽象属性一说。

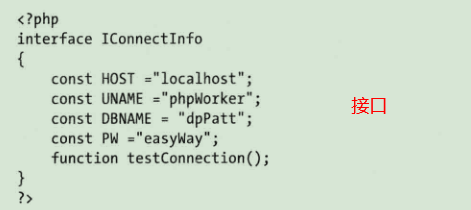
一个抽象类中声明了一个抽象方法，哪继承这个父类的各个子类都必须实现这个方法。

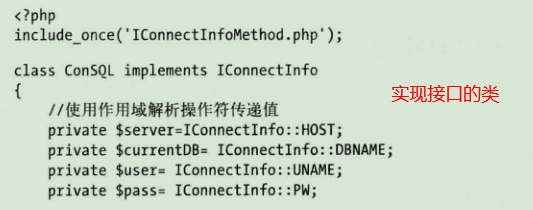
方法签名包括：方法名、参数、返回值类型（PHP没有）

接口区别抽象类，不能包含具体的方法和变量，但可以包含具体常量。

类只要实现接口中所有抽象方法，就可以根据需要增加其他方法和属性。

作用域解析操作符，即双冒号操作符：： 允许在类中以及接口的实现中存取常量



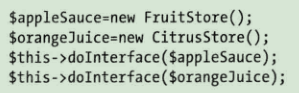


接口中也可以只包含常量而不包括任何方法。通过 接口名：：常量名，可以将常量的值传递给实现类的属性中。

函数参数的类型提示必须是类或者接口的名字，在设计模式中，更倾向于使用抽象类或者接口。

例如，一个接口/抽象类IProduct定义两个方法，有两个子类实现/继承，第三个类中的方法的类型提示就可以是IProduct，就可以传入两个子类的实例化：





实际开发中，强制数据类型为一个接口或者抽象类，好处多多。

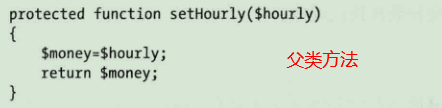
可以使用数组、接口、类做提示，不可以使用标量类型（如string或者int）

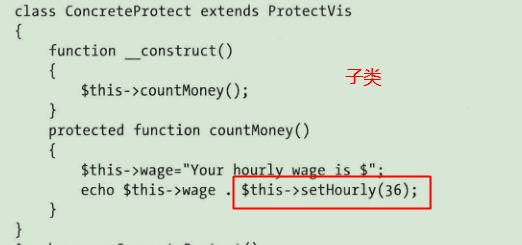
3中类型的可见性：private（私有）、protected（保护）、public（公有）

Private 私有只能在同一个类中访问，只对同一个类的元素可见。

要引用同一个类中的属性或方法，PHP要求对象使用$this语句。

保护protected，则允许这个类以及子类都可以访问该属性，抽象方法和具体方法的可见性都可以设置为保护。同时子类也可以直接用this->调用父类protected属性的方法。



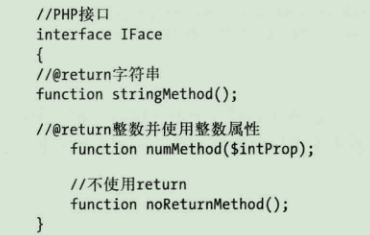


继承的所有属性和方法都需要使用$this->调用

所有构造函数都是公共的。

开发中，为了保证类之间的松绑定，通常会继承抽象类，而且是浅继承，只有一层子类，否则深层次继承到一个具体类，修改父类会带来很大麻烦。

接口的注释中一定要声明是否使用return，以及返回的数据类型，因为有利接口的多态实现时保证统计格式。模仿强类型语言



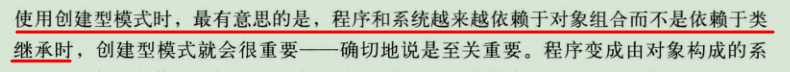
MVC

模型负责数据部分，也称为企业或应用逻辑。

一个对象实例的数据类型不仅仅是他实例化的对象类型，该对象的父类也将作为他的数据类型。



UserAlpha的数据类型就是AlphaA，这是一耳光具体类，同时AlphaA的父类也可以作为UserAlpha的数据类型。



UML

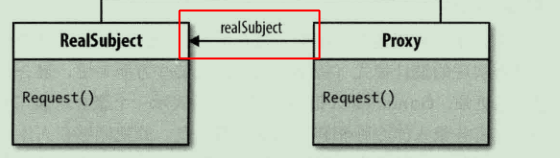
接口（接口和抽象类）显示为斜体

三角形表示继承和实现

抽象方法使用斜体

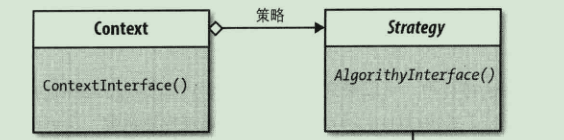
具体方法使用常规字体

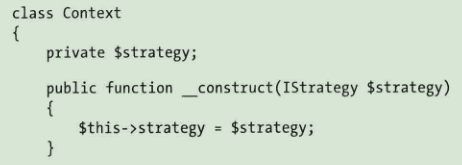
相识关系：一个参与者包含另一个参与者的引用，用一个简单的黑箭头表示





聚合关系：一个聚合对象与他的所有者有相关生命周期。一个接口/抽象类的类型作为另一个类方法（构造方法）的参数类型

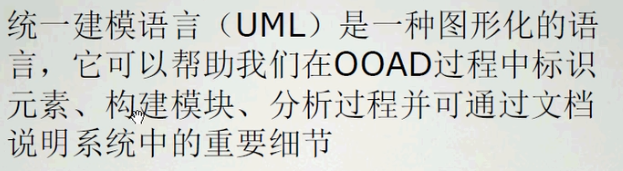


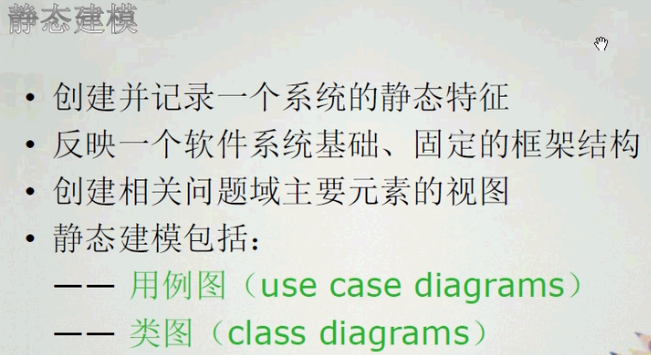


内聚：度量一个类独立完成某项工作的能力

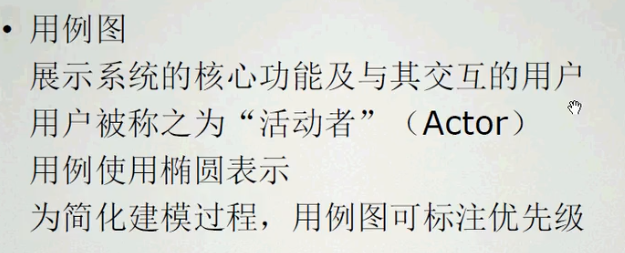
耦合：度量系统内或者系统间依赖关系的复杂度

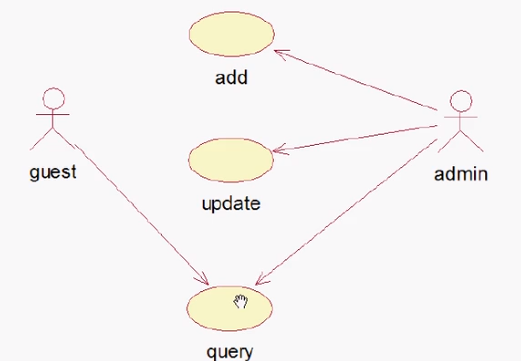
设计原则：高内聚，低耦合



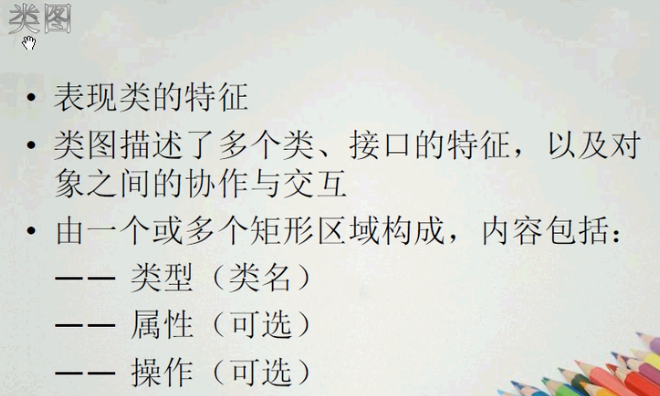








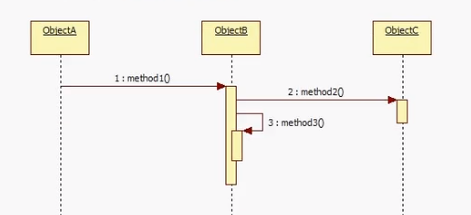




jiantou

箭头指向的是父类

时序图



A方法里引用了B B引用了C，B还调用了自己的方法

