设计模式

六大原则：

• 单一职责原则（SingleResponsibilityPrinciple）；

类的职责应该单一，一个方法只做一件事。职责划分清晰了，每次改动到最小单位的方法或类。◦使用建议：两个完全不一样的功能不应该放一个类中，一个类中应该是一组相关性很高的函数、数据的封装◦用例：网络聊天：网络通信&聊天，应该分割成为网络通信类&聊天类

- 开闭原则（OpenClosedPrinciple）

◦对扩展开放，对修改封闭

◦使用建议：对软件实体的改动，最好用扩展而非修改的方式。

◦用例：超时卖货：商品价格---不是修改商品的原来价格，而是新增促销价格。

- 里氏替换原则（Liskov Substitution Principle）

通俗点讲，就是只要父类能出现的地方，子类就可以出现，而且替换为子类也不会产生任何错误或异常。

◦在继承类时，务必重写父类中所有的方法，尤其需要注意父类的protected方法，子类尽量不要暴露自己的public方法供外界调用。 ◦使用建议：子类必须完全实现父类的方法，孩子类可以有自己的个性。覆盖或实现父类的方法时，输入参数可以被放大，输出可以缩小

◦用例：跑步运动员类-会跑步，子类长跑运动员-会跑步且擅长长跑， 子类短跑运动员-会跑步且擅长短跑

- 依赖倒置原则（Dependence Inversion Principle）

高层模块不应该依赖低层模块，两者都应该依赖其抽象.不可分割的原子逻辑就是低层模式，原子逻辑组装成的就是高层模块。

◦模块间依赖通过抽象（接口）发生，具体类之间不直接依赖

◦使用建议：每个类都尽量有抽象类，任何类都不应该从具体类派生。尽量不要重写基类的方法。结合里氏替换原则使用。

◦用例：奔驰车司机类--只能开奔驰；司机类--给什么车，就开什么车；开车的人：司机--依赖于抽象

- 迪米特法则（Law of Demeter），又叫“最少知道法则”；

◦尽量减少对象之间的交互，从而减小类之间的耦合。一个对象应该对其他对象有最少的了解。对类的低耦合提出了明确的要求：

只和直接的朋友交流，朋友之间也是有距离的。自己的就是自己的（如果一个方法放在本类中，既不增加类间关系，也对本类不产生负面影响，那就放置在本类中）。

◦用例：老师让班长点名--老师给班长一个名单，班长完成点名勾选，返回结果，而不是班长点名，老师勾选

- 接口隔离原则（Interface Segregation Principle）；
- ◦客戶端不应该依赖它不需要的接口，类间的依赖关系应该建立在最小的接口上
- ◦使用建议：接口设计尽量精简单一，但是不要对外暴露没有实际意义的接口。◦用例：修改密码，不应该提供修改用戶信息接口，而就是单一的最小修改密码接，更不要暴露数据库操作