软件设计原则

开闭原则

对扩展开放，对修改关闭。在程序需要进行扩展的时候，不能去修改原有的代码，实现一个热插拔的效果。简言之，是为了使程序的扩展性好，易于维护和升级。（接口和抽象类）

抽象灵活性好，应用性广，只要抽象的合理，可以基本保持软件架构的稳定。而软件中易变的细节可以从抽象派生来的实现类来进行扩张，当软件需要变化时，只需要根据需求重新派生一个实现类来扩展。

里氏代换原则

里氏代换原则是面对对象设计的基本原则之一

里氏代换原则：任何基类可以出现的地方，子类一定可以出现。子类可以扩展父类的功能，但不能改变父类原有的功能。即：除了新的方法新增功能外，尽量不要重写父类的方法。

如果通过父类的方法来完成新的功能，这样虽然简单，但是整个继承体系的可复用性会比较差，特别是多态比较频繁时，程序运行出错的概率非常大🤔。

依赖倒转原则

高层模块不应该依赖底层模块，两者都应该依赖其抽象；抽象不应该依赖细节（子类），细节应该依赖于抽象。简单来说就是要求对抽象进行编程，不要对其实现编程，这样就降低了实现模块之间的耦合。

接口隔离原则

客户端不应该被迫实现依赖于它不使用的方法；一个类对另一个类的依赖应该建立在最小的接口上。

迪米特法则

迪米特法则又叫最少知识原则

只和你的直接朋友交谈，不和陌生人说话

其含义是；如果两个实体无序直接通信，那么久不应当发生直接的互相调用，可以通过第三方转发改调用。其目的是降低类之间的耦合度，提交模块的相对独立性。

迪米特法则中的“朋友”是指：当前对象本身、当前对象的成员对象、当前对象所创建对象，当前对象的方法参数等，这些对象同当前对象存在关联、聚合或组合的关系，可以直接访问这些对象的方法。

合成复用原则

合成复用原则是指：尽量先使用组合或者聚合等关联关系来实现，其次才考虑使用继承关系来实现

继承复合简单是易实现，但是存在一下缺点：

- 继承复用破坏了类的封装性。因为继承将父类的实现细节暴露给子类，父类对子类是透明的，所以这种复用又称为“白箱”复用
- 子类与父类的耦合度高。父类实现的任何改变都会影响子类，这不利于类的扩展和维护
- 它限制了复用的灵活性。从父类继承而来的实现是静态的，在编译时已经定义，所以运行是不能发生变化。

采用组合或者聚合复用时，可以将已有的对象纳入新对象，是指成为新对象的一部分，新对象可以调用已有对象的功能，他有以下优点：

- 它维持了类的封装性。因为对象的内部细节是新对象看不见的，所以称为“黑箱”复用
- 对象间的耦合度低。在类的成员位置申明对象
- 复用的灵活性高。这种复用可以在运行时动态进行，新对象可以动态的引用与成分对象类型相同的对象