### 单一职责原则

定义：不要存在多于一个导致类变更的原因，一个类只负责一项职责。

职责扩散：职责被分化为粒度更细的职责，在职责扩散到我们无法控制的程度之前，立刻对代码重构。

只有逻辑足够简单，才可以在代码级别上违反单一职责原则，只有类中方法数量足够少，才可以在方法级别上违反单一职责原则。

单一职责的优点：降低类的复杂度，只负责一项职责，会导致逻辑比较简单、提高类的可读性、变更引起的风险降低。

### 里式替换原则

定义1：如果每一个类型为T1的对象O1，都有类型为T2的对象O2，使得以T1定义的所有程序P在所有的对象O1都替换成O2时，程序P的行为没有发生变化，那么类型T2是类型T1的子类型。

定义2：所有引用基类的地方必须能透明的使用其子类的对象。

尽量不要重写父类A的方法，也尽量不要重载父类A的方法。这是因为，父类中凡是已经实现好的方法，实际上是在设定一系列的规范与契约，虽然不强制要求所有的子类都要遵从这些契约，但是如果子类对这些非抽象方法任意修改，就会对整个继承体系造成破坏。

继承会给程序带来侵入性，程序的可移植性降低，增加了对象间的耦合性；如果非要重写父类的方法，比较通用的做法是：原来的父类和子类都继承一个更通俗的基类，原有的继承关系去掉，采用依赖、聚合，组合等关系代替。里式替换原则的思想就是：子类可以扩展父类的功能，但不能改变父类原有的功能。

### 依赖倒置原则

定义：高层模块不应该依赖底层模块，二者都应该依赖抽象，抽象不应该依赖细节，细节应该依赖抽象。

相对于细节的多变性，抽象的东西要稳定的多，以抽象为基础搭建的架构比以细节为基础搭建起来的架构稳定，抽象是指接口或者抽象类，细节就是具体的实现类，使用接口或者抽象类的目的是指定好规范与契约，而不去涉及任何具体的操作，把展现细节的任务交给他们的实现类来完成。遵循依赖倒置原则可以降低类之间的耦合性，提高系统的稳定性，降低修改程序造成的风险。

传递依赖关系的三种方式：接口传递、构造方法传递、setter方法传递。

底层模块尽量要有抽象类或者接口；变量的声明类型尽量是抽象类或者接口；使用继承时遵循里氏替换原则。

### 接口隔离原则

客户端不应该依赖它不需要的接口，一个类对另一个类的依赖应该建立在最小的接口上，建立单一接口，不要建立庞大臃肿的接口，尽量细化接口；接口中的方法尽量少；接口设计是对外部设定的契约，通过分散定义多个接口，可以预防外来变更的扩散，提高系统的灵活性和可维护性。

单一职责注重的是职责，接口隔离原则注重对接口依赖的隔离；单一职责原则主要约束类，其次是接口和方法，它针对的是程序中的实现和细节。接口隔离原则主要约束接口，主要针对抽象，针对程序整体框架的构建。

### 迪米特法则

一个对象应该对其他对象保持最少的了解，又叫最少知道法则，一个类对自己依赖的类知道的越少越好，对于被依赖的类来说，无论逻辑多么复杂，都尽量的将逻辑封装在类的内部，除了对外提供public的方法，不对外泄漏任何消息；更简单的定义：只与直接的朋友通信，直接的朋友：2个对象有耦合关系：组合依赖、聚合、关联等；成员变量、方法参数、方法的返回值为类的直接的朋友，局部变量中的类不是直接的朋友。

### 6．开闭原则

一个软件实体如模块、类、函数应该对扩展开放，对修改关闭；当软件需要变化时，尽量通过扩展软件实体的行为来实现变化，不是通过修改已有的代码来实现变化。

开闭原则的思想：用抽象构建框架，用实现扩展细节；抽象灵活性好，适应广，软件中易变的细节，需要从抽象派生的实现类来扩展；当软件需要发生变化时。