1，开闭原则

软件实体应该对扩展开放，对修改关闭

当应用的需求改变时，在不改变软件实体源代码时，可以扩展模块的功能，使其满足新的需求

2，里氏替换原则

继承必须确保超类的所拥有的性质在子类中仍然成立

什么时候该继承，什么时候不应该继承

子类可以扩展父类中的功能，但是不能改变原有父类的功能

3，依赖倒置原则

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，细节应该依赖抽象

要面向接口编程，不要面向实现编程

用抽象层的接口定义基础架构，然后细节是具体实现

1.每个类尽量提供接口或抽象类，或者两者兼备

2.变量的声明类型尽量是接口或者抽象类

3.任何类都不应该从具体类派生

4.使用继承时应该尽量遵循里氏替换原则

4，单一责任原则

一个类应该有仅有一个引起它变化的原因，否则类应该被拆分

对象不应该承担太多的责任，否则或出现功能削弱或者冗余代码

需要设计人员发现类的不同职责，并且将它们分离出来，在封装到不同的类和模块里面

5，接口隔离原则

要求程序员将尽量臃肿庞大的接口拆分成更小，更具体的接口，让接口只包含客户感兴趣的方法

客户端不应该依赖不使用的方法

一个类对另一个类的依赖应该建立在最小的接口上

定尽可能多的接口，然后一个类都implement 他们，然后再在分别返回

6，迪米特原则 最少知识原则

只与你的朋友交谈，不跟陌生人说话，，如果两个软件实体无须直接通信，那么就不应该发生直接的相互

7，合成复用原则

尽量使用组合或者聚合等关联关系来实现，其次再考虑继承关系

如果使用继承关系，则必须严格遵循里氏替代原则