设计模式的六大原则：

**开放-封闭原则：**

**定义:**

软件实体（类，模块，函数等）应该可以扩展，但是不可修改。

对于扩展是开放的，对于更改是封闭的。

**方法：**

设计人员应先猜测出最有可能发生的变化种类，然后构造抽象来隔离那些变化。当变化发生时，立即采取行动，创建抽象来隔离以后发生的同类变化。面对需求，对程序的改动是通过增加新代码进行的，而不是更改现有的代码。

**单一职责原则：**

**定义：**

就一个类而言，应该仅有一个因其他变化的原因。

**依赖倒转原则：**

**定义：**

抽象不应该依赖细节，细节应该依赖于抽象。

高层模块不应该依赖底层模块，两个都应该依赖抽象。

抽象不应该依赖细节，细节应该依赖抽象。

**里氏代换原则：**

**定义：**

子类型必须能够替换掉它们的父类型。

一个软件实体如果使用的是一个父类的话，那么一定适用于其子类，而且它觉察不出父类对象和子类对象的区别。也就是说，在软件里，把父类都替换成它的子类，程序的行为没有变化。

**迪米特法则：**

**定义：**

如果两个类不必彼此直接通信，那么这两个类就不应当发生直接的相互作用。如果其中一个类需要调用另一个类的某一个方法的话，可以通过第三者转发这个调用。

每一个类都应尽量降低成员的访问权限，包装好自己的私有成员，不需要让别的类知道的字段或行为就不要公开。

强调了类之间的松耦合。

**接口隔离原则：**

**定义：**

客户端不应该依赖它不需用的接口，类间的依赖关系应该建立在最小的接口上。

接口尽量细化，同时接口中的方法尽量的少