面向对象的三大特征和六大原则

一.面相对象

面向对象的思想已经涉及到软件开发的各个方面。如，面向对象的分析（OOA，Object Oriented Analysis），面向对象的设计（OOD，Object Oriented Design）、以及我们经常说的面向对象的编程实现（OOP，Object Oriented Programming）。

二.面向对象的三大原则

1.封装

用一个类把多个变量包装起来.

核心设计思想: **分**而治之,变则疏之.

利用抽象数据类型将数据和基于数据的操作封装在一起，使其构成一个不可分割的独立实体。数据被保护在抽象数据类型的内部，尽可能地隐藏内部的细节，只保留一些对外接口使之与外部发生联系。用户无需知道对象内部的细节，但可以通过对象对外提供的接口来访问该对象。

优点:

减少耦合：可以独立地开发、测试、优化、使用、理解和修改

减轻维护的负担：可以更容易被程序员理解，并且在调试的时候可以不影响其他模块

有效地调节性能：可以通过剖析确定哪些模块影响了系统的性能

提高软件的可重用性

降低了构建大型系统的风险：即使整个系统不可用，但是这些独立的模块却有可能是可用的

封装两个方面的含义：一是将有关数据和操作代码封装在对象当中，形成一个基本单位，各个对象之间相对独立互不干扰。

封装的意义?

把一部分或全部属性和部分功能（函数）对外界屏蔽，就是从外界（类的大括号之外）看不到，不可知，这就是封装的意义。

二是将对象中某些属性和操作私有化，已达到数据和操作信息隐蔽，有利于数据安全，防止无关人员修改。

2.继承

核心设计思想: 抽取共性,统一概念,**隔**离变化.

面向对象编程 (OOP) 语言的一个主要功能就是“继承”。继承是指这样一种能力：它可以使用现有类的所有功能，并在无需重新编写原来的类的情况下对这些功能进行扩展。

通过继承创建的新类称为“子类”或“派生类”。

被继承的类称为“基类”、“父类”或“超类”。

继承的过程，就是从一般到特殊的过程。

继承的优越性：通过使用继承，程序员可以在不同的子类中多次重新使用父类中的代码，使程序结构清晰，易于维护和修改，而子类又可以提供一些特殊的行为，这些特殊的行为在父类中是没有的.

3.多态

核心设计思想: 干

定义:基类的一种方法或行为,在不同的派生类中用有不同的表现形式.

没有继承就没有多态，继承是多态的前提。

快捷键: Ctrl + o

举例来说明多态

猫、狗、鸟都是动物

动物不同于植物，猫、狗、鸟都可以叫、都可以吃

这些功能可以同时继承自动物类，猫类、狗类、鸟类都是动物类的子类

但是猫的叫声是“喵喵喵”，狗的叫声是“汪汪汪”，鸟的叫声是“吱吱支”

猫爱吃鱼，狗爱吃骨头，鸟爱吃虫子

这就是虽然继承自同一父类，但是相应的操作却各不相同，这叫多态。

由继承而产生的不同的派生类，其对象对同一消息会做出不同的响应。

三.面向对象的六大原则

1.开闭原则

开闭原则是面向对象的可复用设计的第一块基石，它是最重要的面向对象设计原则。

一个软件实体应当对扩展开放，对修改关闭。即软件实体应尽量在不修改原有代码的情况下进行扩展。

> 开闭原则的英文全称是`Open Close Principle`缩写即`OCP`。

开闭原则的定义是：软件中的对象(类、模块、函数等)应该对于扩展是开放的，但是对于修改是封闭的。在软件的生命周期内，因为变化、升级和维护等原因需要对软件的原有代码进行修改时，可能会将错误的代码引入，从而破坏原有系统。因此当软件需求发生变化时，我们应该尽量通过扩展的方式来实现变化，而不是通过修改已有的代码。

2.单一职责

一个类应该仅有一个引起它变化的原因.

简单来说单一职责就是一个类只负责一个功能。更加具体的说就是对一个类而言，应该是一组相关性很高的函数、数据的封装，是高内聚低耦合的，对外界而言应该仅有一个引起它变化的原因。

3.依赖倒置

抽象不应依赖于细节，细节应该依赖于抽象。

依赖倒置原则有以下几个关键点:

1.高层模块不应该依赖于低层模块，两者都应该依赖其抽象

2.抽象不应该依赖于细节

3.细节应该依赖于抽象

4.组合复用

如果仅仅为了代码的复用优先选择组合复用,而非继承复用.

组合的耦合性相对继承要低.

5.里氏替换

子类的方法来替换父类的方法,实现代码的透明度,灵活性,从而不影响功能的实现.

里氏代换原则告诉我们，在软件中将一个基类对象替换成它的子类对象，程序将不会产生任何错误和异常，反过来则不成立，如果一个软件实体使用的是一个子类对象的话，那么它不一定能够使用基类对象。例如：我喜欢动物，那我一定喜欢狗，因为狗是动物的子类；但是我喜欢狗，不能据此断定我喜欢动物，因为我并不喜欢老鼠，虽然它也是动物。

里氏替换原则的核心是抽象，而抽象又依赖于继承这个特性，在OOP当中，继承的优缺点都相当明显。

优点:

1.代码重用，减少创建类的成本，每个子类都拥有父类的方法和属性

2.子类与父类基本相似，但又与父类有所区别

3.提高代码的可扩展性

缺点:

1.继承是侵入性的，只要继承就必须拥有父类的方法和属性

2.可能造成子类代码冗余，灵活性降低，因为子类必须拥有父类的属性和方法.

6.迪米特法则

迪米特原则：一个对象应该对其他对象有最少的了解，通俗的讲，一个类应该对自己需要耦合或调用的类知道的最少，类的内部如何实现与调用者或者依赖者没有关系，调用者或者依赖者只需要知道他需要的方法即可，其他的一概不管。类与类之间的关系越密切，耦合度越大，当一个类发生改变时，对另一个类的影响也越大。

7.小结

在应用开发过程中，最难的不是完成应用的开发工作，而是在后续的升级、维护过程中让应用系统能够拥抱变化。拥抱变化也意味着在满足需求而且不破坏系统稳定的前提下保持高可扩展性、高内聚、低耦合，在经历了各版本的变更之后依然保持着清晰、灵活、稳定的系统架构。