关于面向对象编程

面向对象是相对于面向过程而言的。面向过程语言是一种基于**功能分析**的、以**算法**为中心的程序设计方法；而面向对象是一种基于**结构分析**的、以**数据**为中心的程序设计思想。早在面向过程语言时代，有一句话说：程序=算法+数据结构。而现在在面向对象语言时代，这句话变为：程序= 对象+消息。对象：万物皆对象； 消息：指对象之间的相互通信。在面向对象语言中有一个有很重要东西，叫做类。从面向过程的角度看，类就是一个特殊的数据结构，它就好像是我们 C 语言中的结构体;从面向对象的角度看，类就是具有相同属性和方法的对象的集合。

面向对象有三大特性：封装、继承、多态。

所谓封装,就是指隐藏对象的实现细节，给外界提供公共的方法来访问。这一点，我个人认为和面向过程语言有本质的区别。在 C 语言中，我们必须在乎每一个实现细节，去关注每一个过程； 而自从从在面向对象语言中提出了封装这个概念后，我们就可以不必要去关心每一个对象的实现细节， 我们只要关注我们所要实现的功能就行，然后根据给我们提供好的接口，我们去面向接口编程就行了。面向对象的封装思想，我认为应用的最好、最成功的地方，就是在微软的.NET 技术上.微软把很多经常用到的功能都封装在一个控件里，作为我们用户不必去在意到底这个控件是用什么实现的，它内部到底是怎么样的？我们只需要关心我们需要实现的功能就行， 然后根据控件给我们提供的属性和方法去操作这些控件，实现我们想要的功能就行了。

面向对象第二个特征，继承。我认为面向对象的继承和生物学的继承很相似。子类可以继承父类的公共属性和方法，子类永远没法继承到父类的私有属性和方法。这一点还区别于生物学的继承，生物学中子类可以同时继承父亲和母亲。但是在 java|C#|C++等面向对象语言中，是不允许多重继承的，但可以多层继承。为了弥补不能多重继承这点，在 java 和 c#语言中都提出了接口这一概念。**接口就是一种规范**。它同样不会有实现细节，而只是给那些要实现这个接口的类一个规范和约束，约束那些实现这些接口的类，要实现我提供的功能，就必须实现我的所有方法， 要不你就声明为抽象类。

面向对象第三大特征，多态。多态，就是同一个实现接口，对不同的实例而执行不同的操作。这一点，我记得在上高中生物的时候学过遗传学，我觉得这里的多态就是

遗传学的变异。　同一个物种的后代由于基因突变或自然环境等影响，而造成不同的个体差异。而我们这里的多态也一样，同属一个基类的不同派生类也可以有自己不同于其他类的属性和方法。除了这封装、继承、多态这三点基本特征外，面向对象还有一个很重要的概念，叫**抽象**。抽象就是把提取事物的本质东西，而忽视非本质的东西。对应于抽象这一概念，java 和 c#中都有一个类叫做抽象类。抽象类中可以给出方法的实现细节，同接口一样如果你要实现我这个抽象类就必须实现我的所有方法，要不你就声明你为抽象类。如果不允许抽象类中有方法的实现细节，这就变成了接口。

总之，面向对象就是万物皆对象，把客观事物当成一个对象来处理的程序设计思想。是一种区别于 POP、SOA、面向组件

等其他程序设计思想，是一种基于结构分析的，以数据为中心的程序设计思想。

面向对象包括三个特征：继承、封装、多态。

类的继承性是指从已有的一个类来extends子类，子类具有了父类的所有特征，同时，子类也可以有新的特性。比如：人是一个类，男人具有了人类的所有的特性，比如思考，比如劳动。同时，男人也有新的特性，比如男人会长胡子。

类的封装是指类把所有的操作都封闭起来，仅仅提供接口出来让其他人使用，使用的人没有必要懂得类里面的操作，只需要懂得操作接口就可以了。比如开汽车，我们只需要懂得方向盘、刹车、油门等等的使用就可以了，没有必要知道刹车是如何构成的封闭好的操作。同时，封装也提高了程序的可替换性。比如两个汽车公司实现刹车的方式不一样，但是，只要我们学会了使用刹车，那么，开两个汽车公司的车就都没有问题了，我们不需要考虑他们怎样实现刹车这门技术的。

类的多态性是一个对象的某项功能可以处理不同类型的问题，或者一个子类可以将父类的某个功能替换成新的功能，具体到编程的方面来说实现类的多态有override和overload，也就是覆写和重载。

同时面向对象还有一个很重要的概念：抽象。抽象就是一个事物的本质。当然对于本质，不同的角度却会有不同的理解。男人，在和女人一起的时候，它的本质是人。在和所有的生物一起的时候，它的本质是动物。（虽然哲学家都说事物的本质只有一个，但是，我仍然坚持我的观点）

面向对象是基于万物皆对象这个哲学观点. 把一个对象抽象成类, 具体上就是把一个对象的静态特征和动态特征抽象成属性和方法, 也就是把一类事物的算法和数据结构封装在一个类之中, 程序就是多个对象和互相之间的通信组成的.

面向对象具有封装性, 继承性, 多态性. 封装隐蔽了对象内部不需要暴露的细节, 使得内部细节的变动跟外界脱离, 只依靠接口进行通信. 封装性降低了编程的复杂性. 通过继承, 使得新建一个类变得容易, 一个类从派生类那里获得其非私有的方法和公用属性的繁琐工作交给了编译器. 而 继承和实现接口和运行时的类型绑定机制所产生的多态, 使得不同的类所产生的对象能够对相同的消息作出不同的反应, 极大地提高了代码的通用性.

总之, 面向对象的特性提高了大型程序的重用性和可维护性.