从北京陈总身上学习到了很多面向对象方面的知识，我自己也在工作中得到了一点点儿经验，本文档仅是对目前自己对面向对象思考的一点儿个人见解，仅作为面向对象学习过程中的学习笔记。有任何不正确的地方，欢迎大家指正。

面向对象：按照字面意思就是说按照对象来思考问题，这点儿相信从事软件方面的人员都能够讲出来。但是根据我自己的个人经历以及周边其他人员的情况来看，至少80%甚至90%以上的从业人员对【面向对象】没有清晰的认知。大家其实根本不知道封装、继承、接口、多态、类、对象这些概念的深层含义以及所要表述的思考过程。

面向对象概念是相对于面向过程来说的，其实只是代表的人们对周遭事物以及发生的各种现象的一种更贴近人们理解认知方式的一种思考方式。

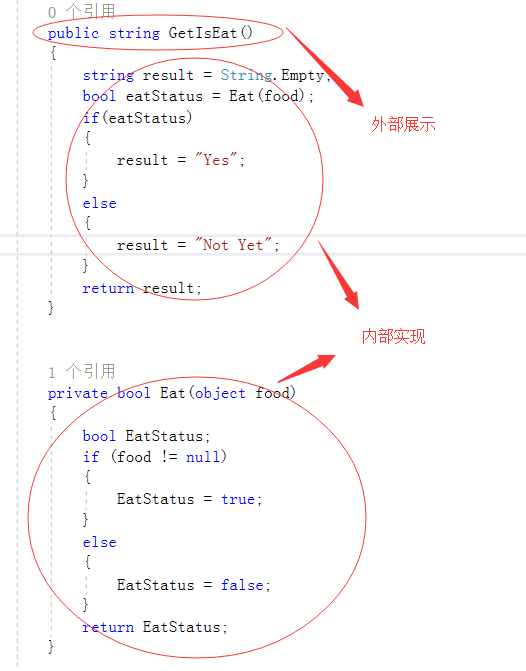
下面解释下我个人对以上几种概念的目前的认知。

对象：对象代表一个具体的事物，这个具体的事务有可能是真实世界的一个具体，比如我，也有可能是一个抽象的具体，比如一条交易记录（可能举例不贴切）。对象都能清楚的知道自己，也就是说能够跟别的对象有很明显的区分，通俗的将就是对象都具有唯一标识，也就是我们代码里面经常用的ID。另外对象都有一些特征，包含特征值以及特征动作。比如一个具体的人：张三。有名称、性别、年龄等特征值，有会吃饭、会睡觉、会开车、会开飞机等特征动作。另外一点儿最重要的事情就是，对象能够处理好自己的事情，也就是说自己的事情自己解决。

说完对象我们来说类。类这个概念其实我认为是从归纳里面的出来的概念。离开对象，就不会有类。类是按照某一观察角度从相关具体对象上抽象/归纳出来的各对象都有的一般/共同点。要想出来类的特征必须先设定一个观察点，另外设定一个对象组（问题域）。举例来说：我想建一个类：人类，这就设定一个观察点（人的共同点）和对象组（所有的人），在这里，我们不按照人种区分黄种人、白种人、黑人等，也不区分高个子人、矮个人、男人、女人、老人、小孩等。就仅仅是人。我们分下发现：人一般都有一个名称，有性别、有出生日期等特征，人都有呼吸、吃饭、喝水、睡觉等动作。这样我们就实现了一个简单的人类。

接着我们讲讲接口。接口在面向对象概念和各面向对象开发语言里面的含义是不一样的。例如C#里面的interface表示的接口是一种开发技巧。虽然也涵盖了面向对象的接口的部分含义，但是没有那么清晰。给很多开发人员造成了接口=interface的假象。重新重点说明下：面向对象里面的接口指的是各交互方之间的一种约定/协议。举例来说：华为目前所有的充电线接口都是用的Type-C规格的接口，这是一种接口。所有的Andriod手机系统应用都要符合Andriod语言的开发规范，Andriod语言的开发规范也是一种接口。所以，再次重申以下：面向对象里面的接口指的是一种规范、约定、协议。

接着我们说说封装。我问过周边一些开发人员，他们对封装的看法大都是停留在Private和Public上，就我目前的认知来说，我没法否认这一点儿跟封装有那么一点儿关系，但是这种认知太浅显和牵强。Private和Public这种仅是个别开发语言的访问权限/级别控制，语言设计师设计这些关键字的时候可能是出于实现封装的概念而制定的，但是跟我们说的面向对象的封装是差别很大的。封装：指的是对象内部的实现与外部的展示相隔离。举个不恰当的例子：两个人见面，一人问到，你午饭吃了么？另一人答：吃了。这个是外部展示，而答的人的吃饭如何拿筷子、如何喝水、如何咀嚼等等属于内部实现，不需要告知外部。在编程上更容易理解：我们写了一个可供外部调用的方法，然后方法里面调用了一个仅供自身调用的内部方法。参考下面的例子理解下。通俗的概括封装就是：外部不需要知道我们是怎么干活的，只需要知道我们干活的结果就行了。这个倒是跟现在流行的管理学上的一些理论很像：老板不关心你是如何工作的，老板只关心你工作的成功。目前比较流行的这种结果为导向的管理方法论。



上面解释了一下什么是封装，另外一个值得深思的问题是我们要封装什么。就是说针对我们的问题域，我们在解决特定需求场景的问题时到底应该把什么封装起来，把什么开放出来。这是一个很值得深思的问题。一般建议的原则是非必须不开放，或者说是最小开放。

继承：这个倒是大家基本上都理解的没有太大的偏差。唯一的问题就是什么时候该用继承。继承是便捷创建类并且解决代码重复的一种方式。而与之对应的另一种便捷创建类的方式是组合。继承和组合最大的区别是：继承解决的是is-a的关系，组合解决的是has-a的关系。所谓的is-a白话就是说：是什么，如：狗是动物，苹果是水果这种。而has-a是说：有什么，如：汽车有发动机，有轮胎。人，有头，有手，有脚。还有一点儿区别是继承是强耦合。有一种说法是说继承破坏了封装，这方面我学习的不深入就不表述我的观点了。但是有一点儿需要强调的是，除非必要，少用继承，多用组合【这句话是基于组合相对继承来说更灵活来说的】。原因是一旦使用了继承并且父类发生的变化，那么所有的子类都会跟着发生变化，因为有时候特定事件的时候子类不想跟着变化。所以到底使用继承还是组合，要根据问题域深入的思考。以便做好各种平衡。世界上没有真正的完美解决方案，我们只是在各种解决方案中尽量选择相对最好的一种。

多态：多态说的是不同类的相同操作具有不同表现。这个也很好理解。比如我们有一个Print方法，在发票类里面我打印出来是发票，在请假单类里面我打印出来的是请假单，在消费记录类里面我打印出来是交易流水。这里面我都是执行的同样的Print操作，但是得到的结果却不一样，这就是多态。

额外说一下设计模式，模式一词来源于建筑学，是为了解决相似的建筑而总结出来的一些通用解决方案，设计模式所表述的思想也基本上类同。是从一些相似的案例上总结出来的一套解决方案，该解决方案能够在其他相似案例上通用并解决特定问题。设计模式是为了解决代码灵活性、可扩展、低耦合、高内聚而总结出来的。但是设计模式一般会造成代码复杂度增加的问题。具体使不使用设计模式，以及使用何种设计模式，同设计模式一样重要。