Java 抽象类和接口

面向对象的三大特征：抽象、继承、多态

本文要介绍的是java三大特征之一的抽象。

为了定义java抽象这个特征，java里面提供了两种定义机制：抽象类和接口。（个人觉得java中提供abstract这个关键词对抽象这个特征的理解会造成误会，可能这也正是好多人觉得这块知识晦涩、难懂的原因之一吧）

本文将从语法定义、编程角度、设计角度三个方面来探讨抽象特征的两种定义方式的区别和联系

注：Java中面向接口编程并不是特指使用interface。这里的接口指的是interface和abstract这两种抽象机制

在面向对象的概念中，我们知道所有的对象都是通过类来描绘的，但是反过来却不是这样。并不是所有的类都是用来描绘对象的，如果一个类中没有包含足够的信息来描绘一个具体的对象，这样的类就是抽象类。抽象类往往用来表征我们在对问题领域进行分析、设计中得出的抽象概念，是对一系列看上去不同，但是本质上相同的具体概念的抽象。比如：如果我们进行一个图形编辑软件的开发，就会发现问题领域存在着圆、三角形这样一些具体概念，它们是不同的，但是它们又都属于形状这样一个概念，形状这个概念在问题领域是不存在的，它就是一个抽象概念。正是因为抽象的概念在问题领域没有对应的具体概念，所以用以表征抽象概念的抽象类是不能够实例化的。

在面向对象领域，抽象类主要用来进行类型隐藏。我们可以构造出一个固定的一组行为的抽象描述，但是这组行为却能够有任意个可能的具体实现方式。这个抽象描述就是抽象类，而这一组任意个可能的具体实现则表现为所有可能的派生类。模块可以操作一个抽象体。由于模块依赖于一个固定的抽象体，因此它可以是不允许修改的；同时，通过从这个抽象体派生，也可扩展此模块的行为功能。熟悉OCP的读者一定知道，为了能够实现面向对象设计的一个最核心的原则OCP(Open-Closed Principle)，抽象类是其中的关键所在。

**从语法定义层面**

在语法层面，Java语言对于abstract class和interface给出了不同的定义方式，下面以定义一个名为Demo的抽象类为例来说明这种不同。

使用abstract class的方式定义Demo抽象类的方式如下：

abstract class Demo ｛

abstract void method1();

abstract void method2();

…

｝

使用interface的方式定义Demo抽象类的方式如下：

interface Demo {

void method1();

void method2();

…

}

在abstract class方式中，Demo可以有自己的数据成员，也可以有非abstarct的成员方法，而在interface方式的实现中，Demo只能够有静态的不能被修改的数据成员（也就是必须是static final的，不过在interface中一般不定义数据成员），所有的成员方法都是abstract的。从某种意义上说，interface是一种特殊形式的abstract class。

**从编程的角度来看**

首先，abstract class在Java语言中表示的是一种继承关系，一个类只能使用一次继承关系。但是，一个类却可以实现多个interface。也许，这是Java语言的设计者在考虑Java对于多重继承的支持方面的一种折中考虑吧。

其次，在abstract class的定义中，我们可以赋予方法的默认行为。但是在interface的定义中，方法却不能拥有默认行为，为了绕过这个限制，必须使用委托，但是这会 增加一些复杂性，有时会造成很大的麻烦。

**从设计理念层面看**

上面主要从语法定义和编程的角度论述了abstract class和interface的区别，这些层面的区别是比较低层次的、非本质的。本小节将从另一个层面：abstract class和interface所反映出的设计理念，来分析一下二者的区别。作者认为，从这个层面进行分析才能理解二者概念的本质所在。

前面已经提到过，abstarct class在Java语言中体现了一种继承关系，要想使得继承关系合理，父类和派生类之间必须存在"is a"关系，即父类和派生类在概念本质上应该是相同的（参考文献〔3〕中有关于"is a"关系的大篇幅深入的论述，有兴趣的读者可以参考）。对于interface 来说则不然，并不要求interface的实现者和interface定义在概念本质上是一致的，仅仅是实现了interface定义的契约而已。

当我们系统需要的是一个类型的时候(即我们的派生类和抽象类均是用来表示现实世界的某个类型)我们使用抽象类，因为抽象类及其实现类是is-a的关系，或者是当我们多个子类中都有相同的方法实现的时候可以考虑使用抽象类，把这些公共的方法提到抽象类中，接口无法做到这一点。其它情况使用接口即可，接口变相的支持多重集成。