**深入理解Java的接口和抽象类**

接口和内部类为我们提供了一种将接口与实现分离的更加结构化的方法。抽象类与接口是Java语言中对抽象概念进行定义的两种机制。

我们都知道在面向对象的领域一切都是对象，同时所有的对象都是通过类来描述的，但是并不是所有的类都是来描述对象的。如果一个类没有足够的信息来描述一个具体的对象，而需要其他具体的类来支撑它，那么这样的类我们称它为抽象类。比如new Animal()，我们都知道这个是产生一个动物Animal对象，但是这个Animal具体长成什么样子我们并不知道，它没有一个具体动物的概念，所以他就是一个抽象类，需要一个具体的动物，如狗、猫来对它进行特定的描述，我们才知道它长成啥样。

      在面向对象领域由于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实例化的。

      同时，抽象类体现了数据抽象的思想，是实现多态的一种机制。它定义了一组抽象的方法，至于这组抽象方法的具体表现形式有派生类来实现。同时抽象类提供了继承的概念，它的出发点就是为了继承，否则它没有存在的任何意义。所以说定义的抽象类一定是用来继承的，同时在一个以抽象类为节点的继承关系等级链中，叶子节点一定是具体的实现类。（不知这样理解是否有错!!!高手指点….）

      在使用抽象类时需要注意几点：

         1、抽象类不能被实例化，实例化的工作应该交由它的子类来完成，它只需要有一个引用即可。

         2、抽象方法必须由子类来进行重写。

         3、只要包含一个抽象方法的抽象类，该方法必须要定义成抽象类，不管是否还包含有其他方法。

         4、抽象类中可以包含具体的方法，当然也可以不包含抽象方法。

         5、子类中的抽象方法不能与父类的抽象方法同名。

         6、abstract不能与final并列修饰同一个类。

         7、abstract 不能与private、static、final或native并列修饰同一个方法。、

在了解抽象类之前，先来了解一下抽象方法。抽象方法是一种特殊的方法：它只有声明，而没有具体的实现。抽象方法的声明格式为：

|  |  |
| --- | --- |
| 1 | abstract void fun(); |

　　抽象方法必须用abstract关键字进行修饰。如果一个类含有抽象方法，则称这个类为抽象类，抽象类必须在类前用abstract关键字修饰。因为抽象类中含有无具体实现的方法，所以不能用抽象类创建对象

　　下面要注意一个问题：在《JAVA编程思想》一书中，将抽象类定义为“包含抽象方法的类”，但是后面发现如果一个类不包含抽象方法，只是用abstract修饰的话也是抽象类。也就是说抽象类不一定必须含有抽象方法。个人觉得这个属于钻牛角尖的问题吧，因为如果一个抽象类不包含任何抽象方法，为何还要设计为抽象类？所以暂且记住这个概念吧，不必去深究为什么。

|  |  |
| --- | --- |
| 1  2  3 | [public] abstract class ClassName {      abstract void fun();  } |

　　从这里可以看出，抽象类就是为了继承而存在的，如果你定义了一个抽象类，却不去继承它，那么等于白白创建了这个抽象类，因为你不能用它来做任何事情。对于一个父类，如果它的某个方法在父类中实现出来没有任何意义，必须根据子类的实际需求来进行不同的实现，那么就可以将这个方法声明为abstract方法，此时这个类也就成为abstract类了。

　　包含抽象方法的类称为抽象类，但并不意味着抽象类中只能有抽象方法，它和普通类一样，同样可以拥有成员变量和普通的成员方法。注意，抽象类和普通类的主要有三点区别：

　　1）抽象方法必须为public或者protected（因为如果为private，则不能被子类继承，子类便无法实现该方法），缺省情况下默认为public。

　　2）抽象类不能用来创建对象；

　　3）如果一个类继承于一个抽象类，则子类必须实现父类的抽象方法。如果子类没有实现父类的抽象方法，则必须将子类也定义为为abstract类。

　　在其他方面，抽象类和普通的类并没有区别。