抽象类的定义及使用

# 引言

1. 抽象类的基本定义；
2. 抽象类的使用原则。

# 具体内容

不会抽象类与接口，Java等于没学。

## 抽象类的基本概念

如果说在一个类之中需要定义一个没有方法体的方法，那么可以利用abstract关键字来进行抽象方法的定义，而包含有抽象方法的类就可以使用abstract来定义成为抽象类。

类的核心组成：属性、方法，但是在学习继承操作之后，会发现子类存在有一种覆写父类方法的机制，而且这一机制直接与对象的多态性有关。于是这样就会出现一个问题：假设现在使用的是普通类，并且在这个类里面有一个print()方法。

但是这个类在设计之初有一个要求，希望继承它的子类一定要覆写这个print()方法。但事实上这个时候的子类完全可以灵活的选择是否需要覆写方法。但是由于只是一个普通方法，所以对于子类是否覆写没有任何的要求，于是这样就会出现一个漏洞，父类无法强制要求子类覆写方法。

如果只依靠普通类的继承，那么根本就不能够对子类产生限制，所以就可以利用抽象类和抽象方法来解决此类问题。

抽象方法的特点：一个是使用了abstract关键字定义，另外一个是方法的后面没有{}，表示没有方法体。

抽象类是不能够直接对象实例化操作的。因为一旦类的对象实例化了，就意味着可以调用类中的所有方法了，但是抽象方法只是一个声明，并没有具体的方法体。所以在实际的开发之中，对于抽象类的使用原则如下：

* 抽象类必须有子类，子类利用extends关键字来继承抽象类，一个子类只能够继承一个父类。
* 抽象类的子类（如果不是抽象类），那么必须要覆写抽象类中的全部抽象方法；
* 抽象类可以利用对象的向上转型机制，通过子类对象进行实例化操作。

抽象类与普通类相比最大的好处是强制定义了子类的实现要求。

本质上讲抽象类就是比普通类多了一些抽象方法的定义而已。

在实际的设计之中，父类的设计师最重要的，普通类与抽象类相比，明显抽象类的约束更加的严格，所以在实际的开发之中，几乎不会出现普通来定义子类的情况，大多数都是继承抽象类。

## 抽象类的相关说明

整个的设计结构里面多了抽象类的定义，那么多了一个定义之后，就需要与原始结构有一些对比。

1. 抽象类不能使用final关键字来定义，因为抽象类必须有子类，而final不能有子类；
2. 抽象类就是比普通类多了抽象方法而已，但是普通类中的所有结构抽象类都可以定义，包括普通方法、构造方法、属性、常量等内容，而且子类对象也符合与对象实例化过程，默认先调用父类中无参构造，而后再执行子类自己的构造方法。
3. 抽象类中可以没有抽象方法，但是依然不可以使用关键字new进行对象的实例化操作。
4. 外部抽象类不允许使用static声明，但是内部抽象类中可以使用static声明，这样表明的是一个外部抽象类。
5. 抽象类中可以存在有static方法，而且static方法不受实例化对象的控制。

## 抽象类实际应用——模板设计模式

清楚了抽象类产生动机以及抽象类的使用之后，下面必须搞清楚一个问题，抽象类与普通类到底有哪些区别？

# 总结

1. 抽象类的设计是在普通类之上的抽象类；
2. 抽象类关键的问题就是约定了子类必须要覆写的抽象方法；
3. 抽象类的使用原则。