学JAVA必须知道:抽象类与接口的区别计算机等级考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/646/2021_2022__E5_AD_A6JAVA_E5_BF_85_c97_646166.htm abstract class和interface是Java语言中对于抽象类定义进行支持的两种机制，正是由于这两种机制的存在，才赋予了Java强大的面向对象能力。 abstract class和interface之间在对于抽象类定义的支持方面具有很大的相似性，甚至可以相互替换，因此很多开发者在进行抽象类定义时对于 abstract class和interface的选择显得比较随意。 其实，两者之间还是有很大的区别的，对于它们的选择甚至反映出对于问题领域本质的理解、对于设计意图的理解是否正确、合理。本文将对它们之间的区别进行一番剖析，试图给开发者提供一个在二者之间进行选择的依据。 一、理解抽象类 abstract class和interface在Java语言中都是用来进行抽象类（本文中的抽象类并非从abstract class翻译而来，它表示的是一个抽象体，而abstract class为Java语言中用于定义抽象类的一种方法，请读者注意区分）定义的，那么什么是抽象类，使用抽象类能为我们带来什么好处呢？ 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。

二、从语法定义层面看abstract class和interface 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(). … } 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com