从一道测试题分析java中的方法重载（overload） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/138/2021_2022__E4_BB_8E_E4_B8_80_E9_81_93_E6_c97_138186.htm 本文旨在通过一道测试题目分析java语言中方法重载的机制，帮助读者更好的掌握java语言的基础知识。 首先我们先看一道测试题目，源代码如下所示,你觉得程序能否通过编译呢，如果可以通过编译输出的结果会是什么呢？ //TestOverLoad.javapublic class TestOverLoad{ public static void main(String[] args) { Test test = new Test(). test.print(null). } }class Test{ public void print(String some) { System.out.println("String version print"). } public void print(Object some) { System.out.println("Object version print"). }} 答案是可以通过编译，输出的结果是String version print。不知道你猜测的是否准确是否知道其中的原理，这个题目明显是考察方法重载的，重载使得java的类可以有具有多个相同方法名的方法。编译器可以通过方法的参数的类型和个数来区分他们。而返回值和异常是不能作为区别标志的。上面的程序输出了String version print是遵循了方法重载中准确性的原则，null是作为一个很特别的参数传给了方法print()，因为你可以认为null是String，也可以认为null是Object。但是从层次上看Object处在更上层，String是从Object继承过来的，调用print(String some)将更准确。 如果在TestOverLoad类中再添加一个方法如下所示，这样会如何呢？ public class TestOverLoad{ public static void main(String[] args) { Test test = new Test(). test.print(null). } }class Test{ public void print(String some) { System.out.println("String version print"). } public void

print(Object some) { System.out.println("Object version print"). } public void print(StringBuffer some) { System.out.println("StringBuffer version print"). }}答案是不能通过编译，为什么呢？由于StringBuffer和String并没有继承上的关系，因此编译器感觉StringBuffer和String作为参数的方法都很准确，它就不知道到时候会运行哪个方法了，因此会出现编译错误，这是方法重载中唯一性的原则。如果我们把参数null修改为"hello world"，那么就可以通过编译并运行输出String version print了。 曾经在java.sun.com上读文章看到一篇文章说方法的返回值也是区别方法的标志，其实这是错误的。看看下面的程序public class A{ public int aMethod(String s) { System.out.println(s). return 1. } public void aMethod(String s) { System.out.println(s). }}编译的时候会提示aMethod(String s)方法已经定义过的错误。实践证明一切！ 