Java中static、this、super、final用法 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022_Java_E4_B8_ADsta_c104_145646.htm 一、static 请先看下面这段程序：public class Hello{public static void main(String[] args){ //(1)System.out.println("Hello,world!"). //(2)}} 看过这段程序，对于大多数学过Java 的从来说，都不陌生。即使没有学过Java，而学过其它的高级语言，例如C，那你也应该能看懂这段代码的意思。它只是简单的输出“ Hello,world ”，一点别的用处都没有，然而，它却展示了static关键字的主要用法。 在1处，我们定义了一个静态的方法名为main，这就意味着告诉Java编译器，我这个方法不需要创建一个此类的对象即可使用。你还得你是怎么运行这个程序吗？一般，我们都是在命令行下，打入如下的命令(加下划线为手动输入)：javac Hello.javajava HelloHello,world! 这就是你运行的过程，第一行用来编译Hello.java这个文件，执行完后，如果你查看当前，会发现多了一个Hello.class文件，那就是第一行产生的Java二进制字节码。第二行就是执行一个Java程序的最普遍做法。执行结果如你所料。在2中，你可能会想，为什么要这样才能输出。好，我们来分解一下这条语句。（如果没有安装Java文档，请到Sun的官方网站浏览J2SE API）首先，System是位于java.lang包中的一个核心类，如果你查看它的定义，你会发现有这样一行：public static final PrintStream out.接着在进一步，点击PrintStream这个超链接，在METHOD页面，你会看到大量定义的方法，查找println，会有这样一行：public void println(String x)。 好了，现在你应该明白为什么我们要那样调

用了，out是System的一个静态变量，所以可以直接使用，而out所属的类有一个println方法。静态方法 通常，在一个类中定义一个方法为static，那就是说，无需本类的对象即可调用此方法。如下所示：class Simple{static void go(){System.out.println("Go...").}}public class Cal{public static void main(String[] args){Simple.go().}} 调用一个静态方法就是“ 类名.方法名 ”,静态方法的使用很简单如上所示。一般来说，静态方法常常为应用程序中的其它类提供一些实用工具所用，在Java的类库中大量的静态方法正是出于此目的而定义的。静态变量 静态变量与静态方法类似。所有此类实例共享此静态变量，也就是说在类装载时，只分配一块存储空间，所有此类的对象都可以操控此块存储空间，当然对于final则另当别论了。看下面这段代码：class Value{static int c=0.static void inc(){c .}}class Count{public static void prt(String s){System.out.println(s).}public static void main(String[] args){Value v1,v2.v1=new Value().v2=new Value().prt("v1.c=" v1.c " v2.c=" v2.c).v1.inc().prt("v1.c=" v1.c " v2.c=" v2.c). }} 结果如下：v1.c=0 v2.c=0v1.c=1 v2.c=1 由此可以证明它们共享一块存储区。static变量有点类似于C中的全局变量的概念。值得探讨的是静态变量的初始化问题。我们修改上面的程序：class Value{static int c=0.Value(){c=15.}Value(int i){c=i.}static void inc(){c .}}class Count{public static void prt(String s){System.out.println(s).}Value v=new Value(10).static Value v1,v2.static{prt("v1.c=" v1.c " v2.c=" v2.c).v1=new Value(27).prt("v1.c=" v1.c " v2.c=" v2.c).v2=new Value(15).prt("v1.c=" v1.c " v2.c=" v2.c).}public static void

main(String[] args){Count ct=new Count().prt("ct.c=" ct.v.c).prt("v1.c=" v1.c " v2.c=" v2.c).v1.inc().prt("v1.c=" v1.c " v2.c=" v2.c).prt("ct.c=" ct.v.c).}}运行结果如下： v1.c=0 v2.c=0v1.c=27 v2.c=27v1.c=15 v2.c=15ct.c=10v1.c=10 v2.c=10v1.c=11 v2.c=11ct.c=11 这个程序展示了静态初始化的各种特性。如果你初次接触Java，结果可能令你吃惊。可能会对static后加大括号感到困惑。首先要告诉你的是，static定义的变量会优先于任何其它非static变量，不论其出现的顺序如何。正如在程序中所表现的，虽然v出现在v1和v2的前面，但是结果却是v1和v2的初始化在v的前面。在static{后面跟着一段代码，这是用来进行显式的静态变量初始化，这段代码只会初始化一次，且在类被第一次装载时。如果你能读懂并理解这段代码，会帮助你对static关键字的认识。在涉及到继承的时候，会先初始化父类的static变量，然后是子类的，依次类推。非静态变量不是本文的主题，在此不做详细讨论，请参考Think in Java中的讲解。 