在Java中使用方法重载 PDF转换可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/252/2021_2022__E5_9C_A8Java_E4_B8_AD_c104_252473.htm 命名转换是任何程序开发工程的重要部分，但当处理特别的名称的时候命名转换也会变得相当繁琐。简化这一过程的其中一个方法是通过重载而重新使用方法的名称。重载能够使具有相同名称但不相同数目和类型参数的类传递给方法。 名称中包含的内容 当将名称分配到类、方法、变量时，使用能够容易理解的名称相当重要。例如，建立一个定义一个人的类，此时应该建立一个合适的名为Person的类。这一个类命名为一个随便的名称dkjfb也无可厚非，但对于开发这一软件的任何程序员是没有什么意义，因为它不能代表本身的含义。Person类应该具有以下的内容： class Person { private String firstName. private String lastName. } 列举的代码声明了一个名为Person的类，其带有两个成员变量参数以存储姓和名。分配给成员变量参数的名称就符合它们本身的含义，这样就很容易地识别存储在变量中的内容。当调用一个Person类时，我们可以很直观地使用这些成员变量参数。 对象构建 建立一个新的对象的实例会触发类的构造函数的方法。以下代码使用了一个基本的构造函数，这一构造函数无需接收任何变量： class Person { private String firstName. private String lastName. Person() { this.firstName = "". this.lastName = "". } } 这一基本的构造函数使用空字符串的变量参数。在以后的程序中可以看到，在很多时候，对象建立时都附带已知的名字。你可以使用方法重载来建立多个方法，但每一个方法都有自己的方法记号。记号指定了被方法

接受的参数。例如，这里是前一构造函数的方法记号：Person() 这一方法可以重载以接收姓和名或者只是名： class Person { private String firstName. private String lastName. Person() { this.firstName = "". this.lastName = "". } Person(String lname) { this.firstName = "". this.lastName = lname. } Person(String fname, String lname) { this.firstName = fname. this.lastName = lname. } } 在一个类中任何具有相同名称的两个方法必须有不同的参数类型或者不同的参数数目，否则编译器拒绝它们。现在类可以声明如下： Person p1 = new Person(). Person p2 = new Person("Patton"). Person p3 = new Person("Patton", "Tony"). 一个Java的特性 重载能够用于标准的Java类中。System.out.println 方法接收多参数列表。相关范例代码可以见以下：System.out.println("Builder.com"). 以及： int test = 2. System.out.println(test). 这两个代码片段编译与执行时都没有任何错误。Println方法已经被设计为接收不同的变量，所以重载超乎构造函数的程序。为了更进一步的说明这一点，我们可以通过添加一个print方法来输出姓和名以加深我们的范例程序： class Person { private String firstName. private String lastName. Person() { this.firstName = "". this.lastName = "". } Person(String lname) { this.firstName = "". this.lastName = lname. } Person(String fname, String lname) { this.firstName = fname. this.lastName = lname. } public void Print() { System.out.println(firstName " " lastName). } public void Print(String pout) { System.out.println(pout " " firstName " " lastName). } } 这两个print方法输出成员变量，其中一个方法接收文本而输出，而另一方法没有采用这样的方式。 当使用重

载的时候 重载是一个功能强大的特性，但你只能在需要的时候使用它。当你确实需要不同变量的多种方法，但这些方法都可以做相同的任务，此时就可以采用重载方式。也就是说，如果多种方法执行不同的任务，此时不能采用重载方式。否则，这一方法只能导致你的程序显得很混乱，特别是其他程序员阅读你的代码的时候。 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