java参数是如何传递的 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/253/2021_2022_java_E5_8F_82_E6_95_B0_c97_253065.htm 总的来说，计算机语言给子程序传递参数的方法有两种。第一种方法是按值传递（call-by-value）。这种方法将一个参数值（value）复制成为子程序的正式参数。这样，对子程序的参数的改变不影响调用它的参数。第二种传递参数的方法是引用调用（call-by-reference）。在这种方法中，参数的引用（而不是参数值）被传递给子程序参数。在子程序中，该引用用来访问调用中指定的实际参数。这样，对子程序参数的改变将会影响调用子程序的参数。你将看到，根据传递的对象不同，Java 将使用这两种不同的方法。 在Java 中，当你给方法传递一个简单类型时，它是按值传递的。因此，接收参数的子程序参数的改变不会影响到该方法之外。例如，看下面的程序： // Simple types are passed by value. class Test { void meth(int i，int j) { i *= 2.j /= 2. } } class CallByValue { public static void main(String args[]) { Test ob = new Test(). int a = 15，b = 20. System.out.println("a and b before call: " a " " b). ob.meth(a，b). System.out.println("a and b after call: " a " " b). } } 该程序的输出如下所示： a and b before call: 15 20 a and b after call: 15 20 可以看出，在meth( ) 内部发生的操作不影响调用中a和b的值。它们的值没在本例中没有变为30和10。 当你给方法传递一个对象时，这种情形就会发生戏剧性的变化，因为对象是通过引用传递的。记住，当你创建一个类类型的变量时，你仅仅创建了一个类的引用。因此，当你将这个引用传递给一个方法时

，接收它的参数将会指向该参数指向的同一个对象。这有力地证明了对象是通过引用调用传递给方法的。该方法中对象的改变确实影响了作为参数的对象。例如，考虑下面的程序：// Objects are passed by reference. class Test { int a，b. Test(int i，int j) {a = i.b = j. } // pass an object void meth(Test o) { o.a *= 2. o.b /= 2. } } class CallByRef {public static void main(String args[]) { Test ob = new Test(15，20). System.out.println("ob.a and ob.b before call: " ob.a " " ob.b). ob.meth(ob). System.out.println("ob.a and ob.b after call: " ob.a " " ob.b). } } 该程序产生下面的输出：ob.a and ob.b before call: 15 20 ob.a and ob.b after call: 30 10 正如你所看到的，在这个例子中，在meth ( ) 中的操作影响了作为参数的对象。 有趣的一点是，当一个对象引用被传递给方法时，引用本身使用按值调用被传递。但是，因为被传递的值指向一个对象，该值的拷贝仍然指向它相应的参数所指向的同一个对象。 注意：当一个简单类型传递给一个方法时，使用按值传递。对象传递则按引用传递。