<https://mp.weixin.qq.com/s/d34YDBPrv-KGzfqOso-HWg>

# 一文彻底搞懂Java中的值传递和引用传递！

关于Java中方法间的参数传递到底是怎样的、为什么很多人说Java只有值传递等问题，一直困惑着很多人，甚至我在面试的时候问过很多有丰富经验的开发者，他们也很难解释的很清楚。

我很久也写过一篇文章，我当时认为我把这件事说清楚了，但是，最近在整理这部分知识点的时候，我发现我当时理解的还不够透彻，于是我想着通过Google看看其他人怎么理解的，但是遗憾的是没有找到很好的资料可以说的很清楚。

于是，我决定尝试着把这个话题总结一下，重新理解一下这个问题。

**辟谣时间**

关于这个问题，在StackOverflow上也引发过广泛的讨论，看来很多程序员对于这个问题的理解都不尽相同，甚至很多人理解的是错误的。还有的人可能知道Java中的参数传递是值传递，但是说不出来为什么。

在开始深入讲解之前，有必要纠正一下大家以前的那些错误看法了。如果你有以下想法，那么你有必要好好阅读本文。

错误理解一：值传递和引用传递，区分的条件是传递的内容，如果是个值，就是值传递。如果是个引用，就是引用传递。

错误理解二：Java是引用传递。

错误理解三：传递的参数如果是普通类型，那就是值传递，如果是对象，那就是引用传递。

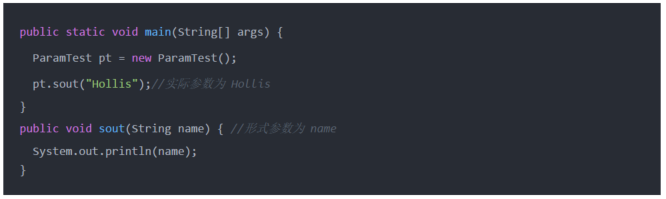
## 实参与形参

我们都知道，在Java中定义方法的时候是可以定义参数的。比如Java中的main方法，public static void main(String[] args)，这里面的args就是参数。参数在程序语言中分为形式参数和实际参数。

形式参数：是在定义函数名和函数体的时候使用的参数,目的是用来接收调用该函数时传入的参数。

实际参数：在调用有参函数时，主调函数和被调函数之间有数据传递关系。在主调函数中调用一个函数时，函数名后面括号中的参数称为“实际参数”。

简单举个例子：



实际参数是调用有参方法的时候真正传递的内容，而形式参数是用于接收实参内容的参数。

## 求值策略

我们说当进行方法调用的时候，需要把实际参数传递给形式参数，那么传递的过程中到底传递的是什么东西呢？

这其实是程序设计中**求值策略（Evaluation strategies）**的概念。

在计算机科学中，**求值策略是确定编程语言中表达式的求值的一组（通常确定性的）规则**。求值策略定义何时和以何种顺序求值给函数的实际参数、什么时候把它们代换入函数、和代换以何种形式发生。

求值策略分为两大基本类，基于如何处理给函数的实际参数，分为严格的和非严格的。

### 严格求值

在“严格求值”中，函数调用过程中，给函数的实际参数总是在应用这个函数之前求值。多数现存编程语言对函数都使用严格求值。所以，我们本文只关注严格求值。

在严格求值中有几个关键的求值策略是我们比较关心的，那就是**传值调用**（Call by value）、**传引用调用**（Call by reference）以及**传共享对象调用**（Call by sharing）。

#### 传值调用（值传递）

 在传值调用中，实际参数先被求值，然后其值通过复制，被传递给被调函数的形式参数。因为形式参数拿到的只是一个"局部拷贝"，所以如果在被调函数中改变了形式参数的值，并不会改变实际参数的值。

#### 传引用调用（引用传递）

在传引用调用中，传递给函数的是它的实际参数的隐式引用而不是实参的拷贝。因为传递的是引用，所以，如果在被调函数中改变了形式参数的值，改变对于调用者来说是可见的。

#### 传共享对象调用（共享对象传递）

传共享对象调用中，先获取到实际参数的地址，然后将其复制，并把该地址的拷贝传递给被调函数的形式参数。因为参数的地址都指向同一个对象，所以我们称也之为"传共享对象"，所以，如果在被调函数中改变了形式参数的值，调用者是可以看到这种变化的。

不知道大家有没有发现，其实传共享对象调用和传值调用的过程几乎是一样的，都是进行"求值"、"拷贝"、"传递"。你品，你细品。

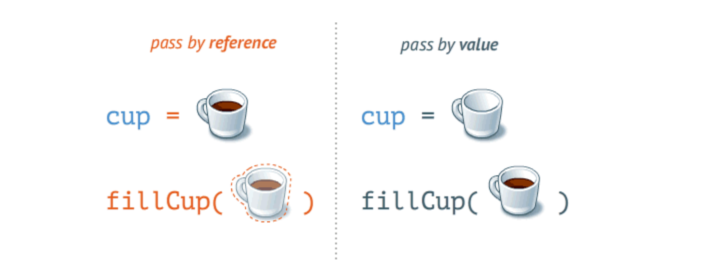
但是，传共享对象调用和内传引用调用的结果又是一样的，都是在被调函数中如果改变参数的内容，那么这种改变也会对调用者有影响。你再品，你再细品。

那么，共享对象传递和值传递以及引用传递之间到底有什么关系呢？

对于这个问题，我们应该关注过程，而不是结果，**因为传共享对象调用的过程和传值调用的过程是一样的，而且都有一步关键的操作，那就是"复制"，所以，通常我们认为传共享对象调用是传值调用的特例**

我们先把传共享对象调用放在一边，我们再来回顾下传值调用和传引用调用的主要区别：

**传值调用是指在调用函数时将实际参数`复制`一份传递到函数中，传引用调用是指在调用函数时将实际参数的引用`直接`传递到函数中。**



所以，两者的最主要区别就是是直接传递的，还是传递的是一个副本。

这里我们来举一个形象的例子。再来深入理解一下传值调用和传引用调用：

你有一把钥匙，当你的朋友想要去你家的时候，如果你直接把你的钥匙给他了，这就是引用传递。

这种情况下，如果他对这把钥匙做了什么事情，比如他在钥匙上刻下了自己名字，那么这把钥匙还给你的时候，你自己的钥匙上也会多出他刻的名字。

你有一把钥匙，当你的朋友想要去你家的时候，你复刻了一把新钥匙给他，自己的还在自己手里，这就是值传递。

这种情况下，他对这把钥匙做什么都不会影响你手里的这把钥匙。

**Java的求值策略**

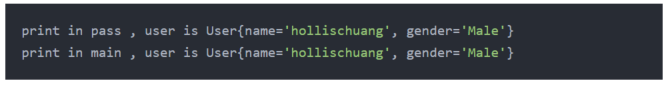
前面我们介绍过了传值调用、传引用调用以及传值调用的特例传共享对象调用，那么，Java中是采用的哪种求值策略呢？

很多人说Java中的基本数据类型是值传递的，这个基本没有什么可以讨论的，普遍都是这样认为的。

但是，有很多人却误认为Java中的对象传递是引用传递。之所以会有这个误区，主要是因为Java中的变量和对象之间是有引用关系的。Java语言中是通过对象的引用来操纵对象的。所以，很多人会认为对象的传递是引用的传递。

而且很多人还可以举出以下的代码示例：

  
输出结果：



可以看到，对象类型在被传递到pass方法后，在方法内改变了其内容，最终调用方main方法中的对象也变了。

所以，很多人说，这和引用传递的现象是一样的，就是在方法内改变参数的值，会影响到调用方。

但是，其实这是走进了一个误区。

**Java中的对象传递**

很多人通过代码示例的现象说明Java对象是引用传递，那么我们就从现象入手，先来反驳下这个观点。

我们前面说过，无论是值传递，还是引用传递，只不过是求值策略的一种，那求值策略还有很多，比如前面提到的共享对象传递的现象和引用传递也是一样的。那凭什么就说Java中的参数传递就一定是引用传递而不是共享对象传递呢？

那么，Java中的对象传递，到底是哪种形式呢？其实，还真的就是共享对象传递。

其实在 《The Java™ Tutorials》中，是有关于这部分内容的说明的。首先是关于基本类型描述如下：

Primitive arguments, such as an int or a double, are passed into methods by value. This means that any changes to the values of the parameters exist only within the scope of the method. When the method returns, the parameters are gone and any changes to them are lost.

**即，原始参数通过值传递给方法。这意味着对参数值的任何更改都只存在于方法的范围内。当方法返回时，参数将消失，对它们的任何更改都将丢失。**

关于对象传递的描述如下：

Reference data type parameters, such as objects, are also passed into methods by value. This means that when the method returns, the passed-in reference still references the same object as before. However, the values of the object’s fields can be changed in the method, if they have the proper access level.

**也就是说，引用数据类型参数(如对象)也按值传递给方法。这意味着，当方法返回时，传入的引用仍然引用与以前相同的对象。但是，如果对象字段具有适当的访问级别，则可以在方法中更改这些字段的值。**

这一点官方文档已经很明确的指出了，Java就是值传递，只不过是把对象的引用当做值传递给方法。你细品，这不就是共享对象传递么？

**其实Java中使用的求值策略就是传共享对象调用，也就是说，Java会将对象的地址的拷贝传递给被调函数的形式参数。**只不过"传共享对象调用"这个词并不常用，所以Java社区的人通常说"Java是传值调用"，这么说也没错，因为传共享对象调用其实是传值调用的一个特例。

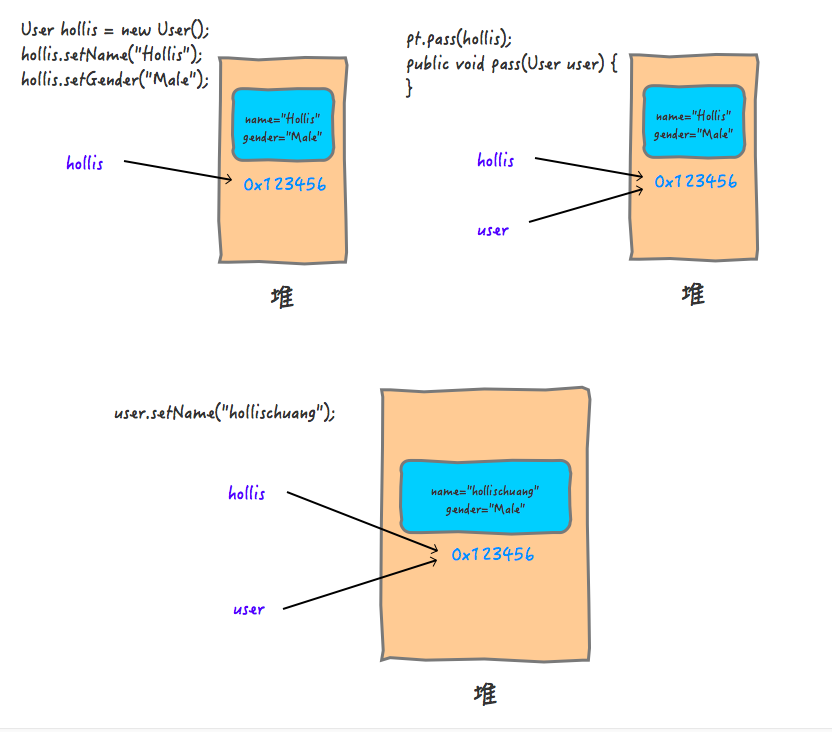
**值传递和共享对象传递的现象冲突吗？**

看到这里很多人可能会有一个疑问，既然共享对象传递是值传递的一个特例，那么为什么他们的现象是完全不同的呢？

难道值传递过程中，如果在被调方法中改变了值，也有可能会对调用者有影响吗？那到底什么时候会影响什么时候不会影响呢？

其实是不冲突的，之所以会有这种疑惑，是因为大家对于到底是什么是"改变值"有误解。

我们先回到上面的例子中来，看一下调用过程中实际上发生了什么？



在参数传递的过程中，实际参数的地址0X1213456被拷贝给了形参。这个过程其实就是值传递，只不过传递的值得内容是对象的应用。

那为什么我们改了user中的属性的值，却对原来的user产生了影响呢？

其实，这个过程就好像是：你复制了一把你家里的钥匙给到你的朋友，他拿到钥匙以后，并没有在这把钥匙上做任何改动，而是通过钥匙打开了你家里的房门，进到屋里，把你家的电视给砸了。

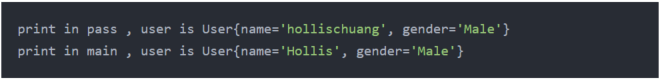
这个过程，对你手里的钥匙来说，是没有影响的，但是你的钥匙对应的房子里面的内容却是被人改动了。

也就是说，**Java对象的传递，是通过复制的方式把引用关系传递了，如果我们没有改引用关系，而是找到引用的地址，把里面的内容改了，是会对调用方有影响的，因为大家指向的是同一个共享对象。**

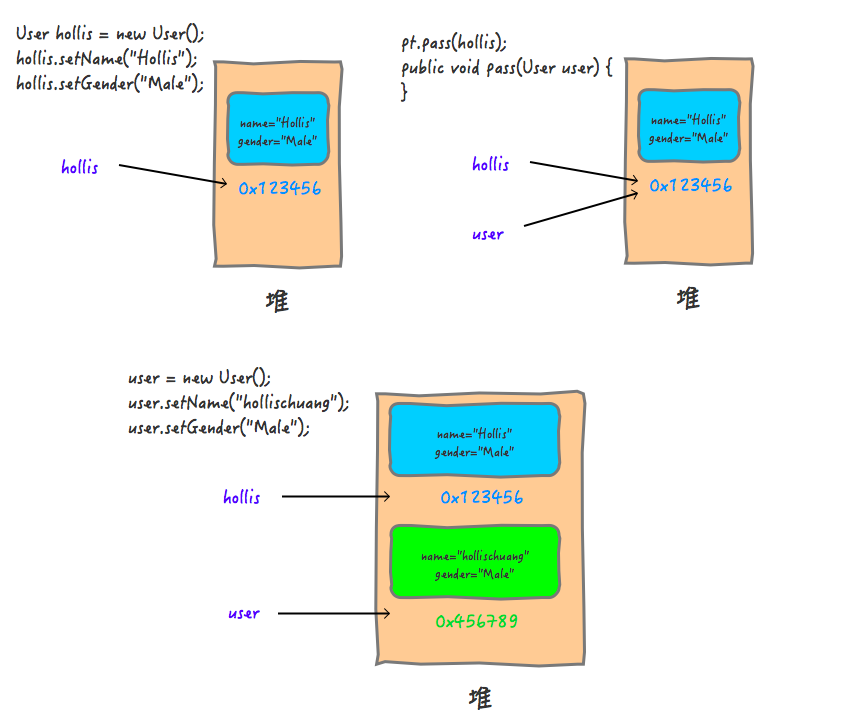
那么，如果我们改动一下pass方法的内容：



上面的代码中，我们在pass方法中，重新new了一个user对象，并改变了他的值，输出结果如下：



再看一下整个过程中发生了什么：



这个过程，就好像你复制了一把钥匙给到你的朋友，你的朋友拿到你给他的钥匙之后，找个锁匠把他修改了一下，他手里的那把钥匙变成了开他家锁的钥匙。这时候，他打开自己家，就算是把房子点了，对你手里的钥匙，和你家的房子来说都是没有任何影响的。

**所以，Java中的对象传递，如果是修改引用，是不会对原来的对象有任何影响的，但是如果直接修改共享对象的属性的值，是会对原来的对象有影响的。**

**总结**

我们知道，编程语言中需要进行方法间的参数传递，这个传递的策略叫做求值策略。

在程序设计中，求值策略有很多种，比较常见的就是值传递和引用传递。还有一种值传递的特例——共享对象传递。

值传递和引用传递最大的区别是传递的过程中有没有复制出一个副本来，如果是传递副本，那就是值传递，否则就是引用传递。

在Java中，其实是通过值传递实现的参数传递，只不过对于Java对象的传递，传递的内容是对象的引用。

**我们可以总结说，Java中的求值策略是共享对象传递，这是完全正确的。**

但是，为了让大家都能理解你说的，**我们说Java中只有值传递，只不过传递的内容是对象的引用。这也是没毛病的。**

但是，绝对不能认为Java中有引用传递。