java参数传递（超经典）

http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif (2009-02-20 14:47:22)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

|  |  |
| --- | --- |
| 标签：  [杂谈](http://search.sina.com.cn/?c=blog&q=%D4%D3%CC%B8&by=tag) | 分类： [Java](http://blog.sina.com.cn/s/articlelist_1264724622_1_1.html) |

Java中的参数传递机制一直以来大家都争论不休，究竟是“传值”还是“传址（传引用）”，争论的双方各执一词，互不相让。不但“菜鸟”们一头雾水，一些“老鸟”也只知道结果却说不出所以然来。我相信看过下面的内容后，你就会明白一些。

先看**基本类型作为参数传递**的例子：

public class Test1 {

       public static void main(String[] args) {

        int n = 3;

        System.out.println("Before change, n = " + n);

        changeData(n);

        System.out.println("After changeData(n), n = " + n);

    }

       public static void changeData(int nn) {

        n = 10;

    }

}

我想这个例子大家都明白，**基本类型作为参数传递时，是传递值的拷贝，无论你怎么改变这个拷贝，原值是不会改变的**，输出的结果证明了这一点：

Before change, n = 3

After changeData(n), n = 3

那么，我们现在来看看**对象作为参数传递**的例子，这也是大家争论的地方。

public class Test2 {

       public static void main(String[] args) {

        StringBuffer sb = new StringBuffer("Hello ");

        System.out.println("Before change, sb = " + sb);

        changeData(sb);

        System.out.println("After changeData(n), sb = " + sb);

    }

       public static void changeData(StringBuffer strBuf) {

        strBuf.append("World!");

    }

}

先看输出结果：

Before change, sb = Hello

After changeData(n), sb = Hello World!

从结果来看，sb的值被改变了，那么是不是可以说：对象作为参数传递时，是把对象的引用传递过去，如果引用在方法内被改变了，那么原对象也跟着改变。从上面例子的输出结果来看，这样解释是合理。

现在我们对上面的例子稍加改动一下：

public class Test3 {

       public static void main(String[] args) {

        StringBuffer sb = new StringBuffer("Hello ");

        System.out.println("Before change, sb = " + sb);

        changeData(sb);

        System.out.println("After changeData(n), sb = " + sb);

    }

       public static void changeData(StringBuffer strBuf) {

           strBuf = new StringBuffer("Hi ");

           strBuf.append("World!");

    }

}

按照上面例子的经验：对象作为参数传递时，是把对象的引用传递过去，如果引用在方法内被改变了，那么原对象也跟着改变。你会认为应该输出：

Before change, sb = Hello

After changeData(n), sb = Hi World!

但运行一下这个程序，你会发现结果是这样的：

Before change, sb = Hello

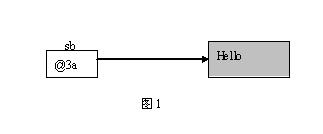
After changeData(n), sb = Hello

这就是让人迷惑的地方，对象作为参数传递时，同样是在方法内改变了对象的值，为什么有的是改变了原对象的值，而有的并没有改变原对象的值呢？这时候究竟是“传值”还是“传引用”呢？

下面就让我们仔细分析一下，来揭开这中间的奥秘吧。

先看Test2这个程序：

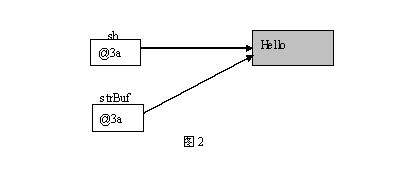
StringBuffer sb = new StringBuffer("Hello ");

这一句执行完后，就会在[内存](http://hw.rdxx.com/Memory/)的堆里生成一个sb对象，请看图1：[](http://photo.blog.sina.com.cn/showpic.html#blogid=4b622a8e0100c1bo&url=http://s4.sinaimg.cn/orignal/4b622a8eh635416d81893)

如图1所示，sb是一个引用，里面存放的是一个地址“@3a”（这个“@3a”是我举的代表内存地址的例子，你只需知道是个内存地址就行了），而这个地址正是“Hello ”这个字符串在内存中的地址。

changeData(sb);

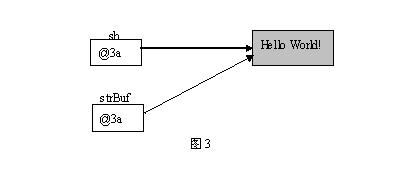
执行这一句后，就把sb传给了changeData方法中的StringBuffer strBuf，由于sb中存放的是地址，所以，strBuf中也将存放相同的地址，请看图2：

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4b622a8e0100c1bo&url=http://s4.sinaimg.cn/orignal/4b622a8eh63541cfeadf3)

此时，sb和strBuf中由于存放的内存地址相同，因此都指向了“Hello”。

strBuf.append("World!");

执行changeData方法中的这一句后，改变了strBuf指向的内存中的值，如下图3所示：

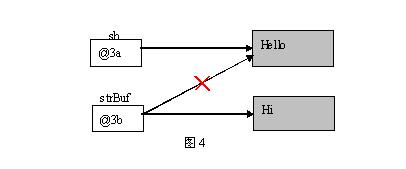
[](http://photo.blog.sina.com.cn/showpic.html#blogid=4b622a8e0100c1bo&url=http://s1.sinaimg.cn/orignal/4b622a8eh635422cd77a0)

所以，Test2 这个程序最后会输出：

After changeData(n), sb = Hello World!

再看看Test3这个程序。

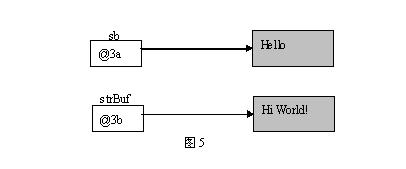
在没有执行到changeData方法的strBuf = new StringBuffer(“Hi “);之前，对象在内存中的图和上例中“图2”是一样的，而执行了strBuf = new StringBuffer(“Hi “);之后，则变成了：

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4b622a8e0100c1bo&url=http://s12.sinaimg.cn/orignal/4b622a8eh6354268f5d5b)

此时，strBuf中存放的不再是指向“Hello”的地址，而是指向“Hi ”的地址“@3b” （同样“@3b”是个例子）了，new操作符操作成功后总会在内存中新开辟一块存储区域。

       strBuf.append("World!");

       而执行完这句后，

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4b622a8e0100c1bo&url=http://s11.sinaimg.cn/orignal/4b622a8eh635429f43b2a)

通过上图可以看到，由于sb和strBuf中存放地址不一样了，所以虽然strBuf指向的内存中的值改变了，但sb指向的内存中值并不会变，因此也就输出了下面的结果：

After changeData(n), sb = Hello

String类是个特殊的类，对它的一些操作符是重载的，如：

String str = “Hello”; 等价于String str = new String(“Hello”);

String str = “Hello”;

str = str + “ world!”;等价于str = new String((new StringBuffer(str)).append(“ world!”));

因此，你只要按上面的方法去分析，就会发现String对象和基本类型一样，一般情况下作为参数传递，在方法内改变了值，而原对象是不会被改变的。

综上所述，我们就会明白，**在Java中对象作为参数传递时，是把对象在内存中的地址拷贝了一份传给了参数。**

你可以试着按上面的画图法分析一下下面例子的结果，看看运行结果与你分析的结果是否一样：

public class Test4 {

       public static void main(String[] args) {

        StringBuffer sb = new StringBuffer("Hello ");

        System.out.println("Before change, sb = " + sb);

        changeData(sb);

        System.out.println("After changeData(n), sb = " + sb);

    }

       public static void changeData(StringBuffer strBuf) {

           StringBuffer sb2 = new StringBuffer("Hi ");

           strBuf = sb2;

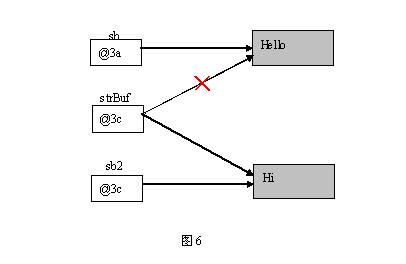
           sb2.append("World!");

    }

}

    提示：

         执行完strBuf = sb2；后：

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4b622a8e0100c1bo&url=http://s13.sinaimg.cn/orignal/4b622a8eh63542d9d175c)