关于Java对象作为参数传递是传值还是传引用的问题

原创 Alan_Xiang 最后发布于2016-09-06 22:37:20 阅读数 14135 ☆ 收藏

分类专栏： Java基础

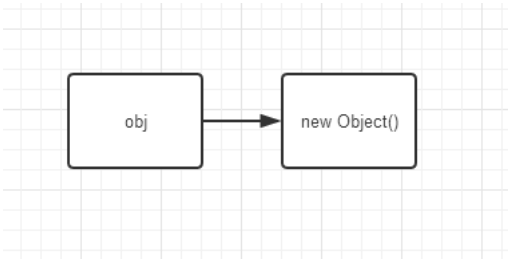
前言

在Java中，当对象作为参数传递时，究竟传递的是对象的值，还是对象的引用，这是一个饱受争议的话题。若传的是值，那么函数接收的只是实参本，函数对形参的操作并不会对实参产生影响；若传的是引用，那么此时对形参的操作则会影响到实参。

首先我们来看一句代码：

```
Object obj = new Object();
```

这句话的意思是：创建一个Object对象，再创建一个名为obj的引用，让这个引用指向这个对象，如下图所示：



在有了上面的基础之后，我们便来看下面这组在网上很流行的例子：

基本数据类型作为参数传递：

例1：

```
public class test {
    public static void main(String[] args) {
        int i = 1;
        System.out.println("before change, i = "+i);
        change(i);
        System.out.println("after change, i = "+i);
    }
    public static void change(int i){
        i = 5;
    }
}
```

这个例子不难理解，**当基本数据类型作为参数传递时，传递的是实参值的副本，即传的是值，无论在函数中怎么操作这个副本，实参的值是不会被**以上代码执行的结果是：

before change, i = 1
after change, i = 1

对象作为参数传递：

在下面的例 2 中，我们把StringBuffer对象作为参数传递到change函数。

例2：

```
public class test {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Hello ");
        System.out.println("before change, sb is "+sb.toString());
        change(sb);
        System.out.println("after change, sb is "+sb.toString());
    }
    public static void change(StringBuffer stringBuffer){
        stringBuffer.append("world !");
    }
}
```

为了方便推理出结论，我们先直接看程序的运行结果：

before change, sb is Hello
after change, sb is Hello world !

从输出结果中我们可以发现，sb所指向的对象的值被改变了，那么是否我们可以推论出，在Java中，当对象作为参数传递时，传递的 对象的引
再来看下面这个例子：

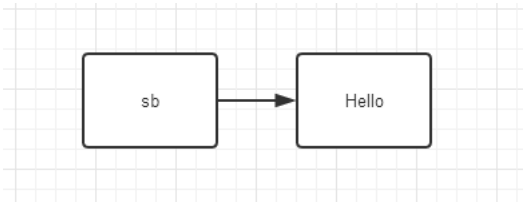
例3：

```
public class test {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Hello ");
        System.out.println("before change, sb is "+sb.toString());
        change(sb);
        System.out.println("after change, sb is "+sb.toString());
    }
    public static void change(StringBuffer stringBuffer){
        stringBuffer = new StringBuffer("Hi ");
        stringBuffer.append("world !");
    }
}
```

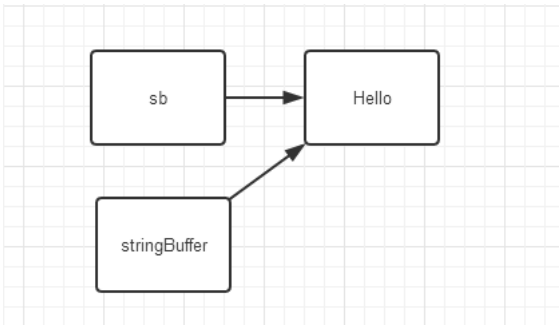
如果上面的推论是正确的，即Java中对象作为参数传递，实际传递的是该对象的引用，那么在调用change函数之后，原对象的值应该是会改变的，
world！”，但是，当我们运行程序后，结果却是如下所示：

before change, sb is Hello
after change, sb is Hello

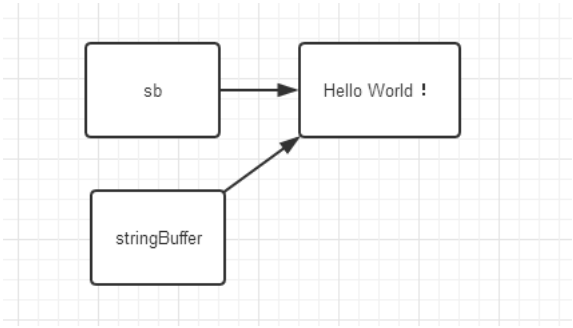
原对象的值并没有被改变，这与上面的推论相矛盾！为什么在Java中，当对象作为参数传递时，有的时候实参被改变了，而有的时候实参并未被改
让我们来分析一下其中的原因：
从文章的开头我们知道，当执行StringBuffer sb = new StringBuffer(“Hello ”)时，我们创建了一个指向新建对象“new StringBuffer(“Hello
“sb”，如下图所示：



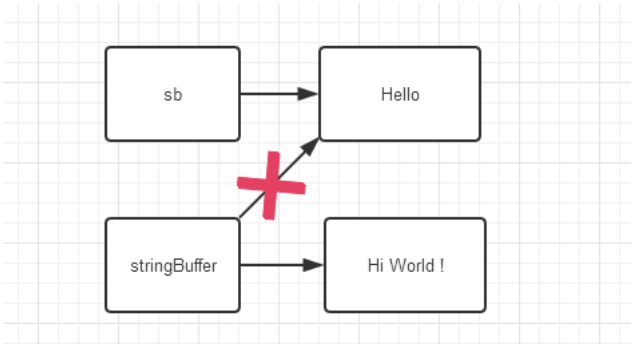
在例2中，当我们调用change函数后，实际上，形参stringBuffer也指向了实参sb所指向的对象，即：



那么当我们执行stringBuffer.append(“world !”)后，便通过对象的引用“stringBuffer”修改了对象的值，使之变成了“Hello w ”，即



但是，在例3中的change函数中，我们又新建了一个对象“new StringBuffer(“Hi ”)” （这实际上在内存中开辟了一块在原对象地址之外的新区形参stringBuffer实际指向了这个新建的对象，并将新对象的值设置为“Hi world ! ”，即：



那么我们就不难理解，为何在执行完change函数之后，实参的值仍为“Hello”了。

结论

综上所述，我们可以得出结论：**在Java中，当对象作为参数传递时，实际上传递的是一份“引用的拷贝”。**