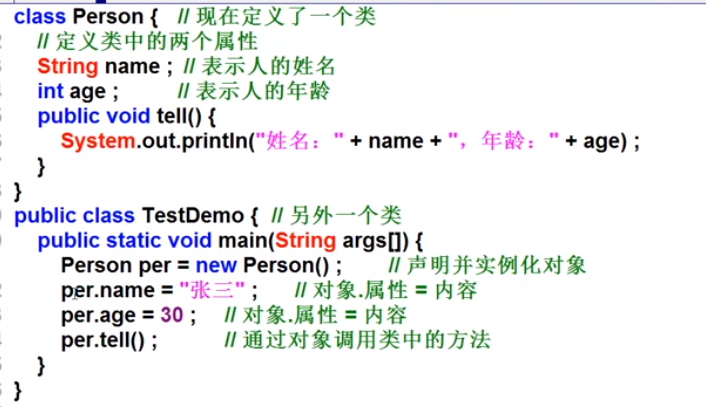
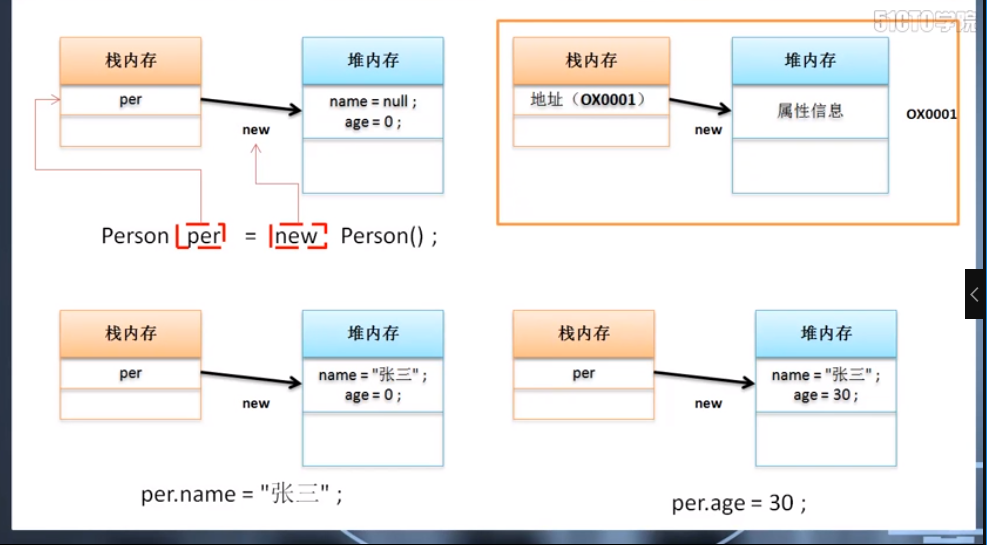
# java内存模型

1. 对象内存分析

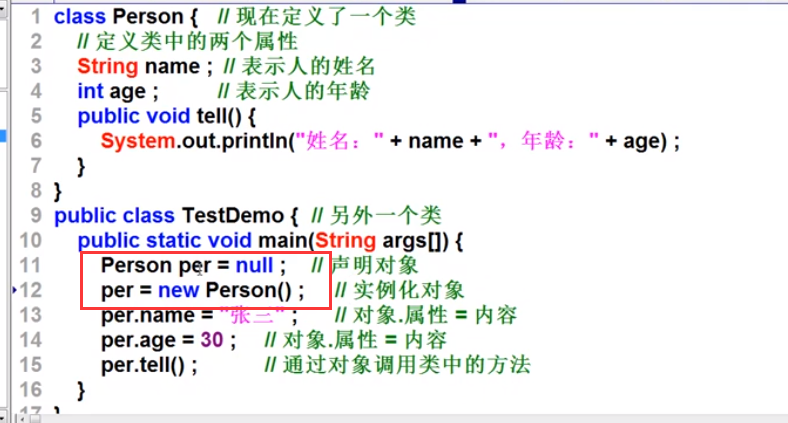
类本身属于引用数据类型，所以对于引用数据类型就必须进行内存分析，首先是两块内存空间的定义：

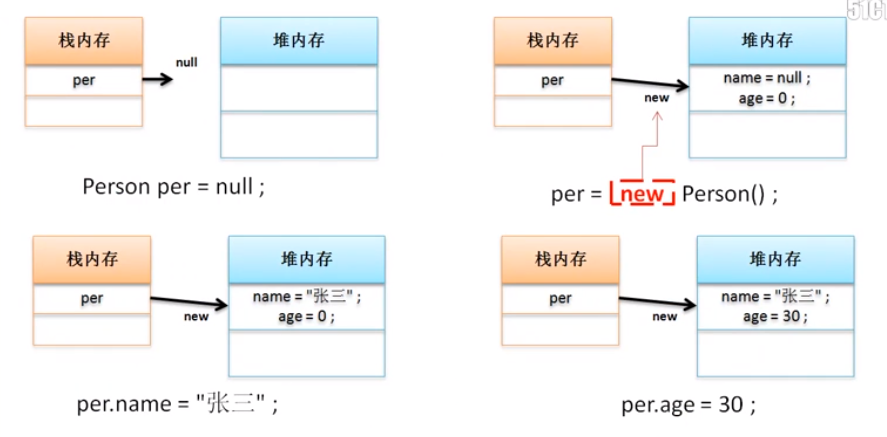
* 堆内存空间（Heap）:保存具体的对象属性信息；
* 栈内存空间（Stack）:保存的是堆内存地址数值；假设栈内存中保存的是对象名称。





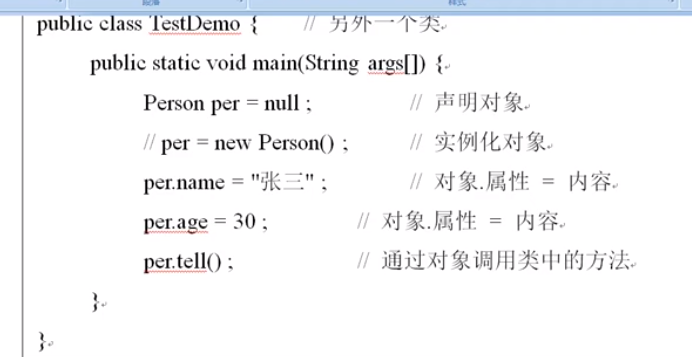
实际上对象的产生本次只使用了一种，还有一种分步的方式完成。





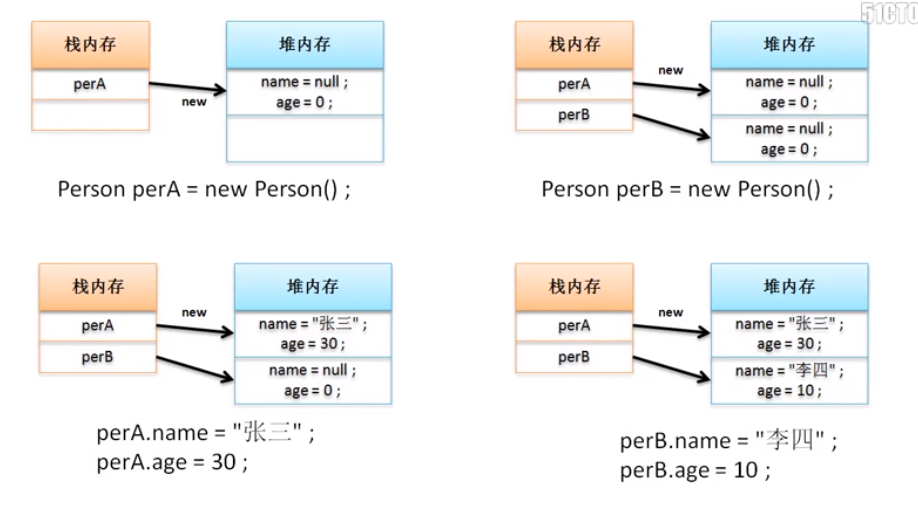
注意：关于引用数据类型操作存在的主要问题。

理论上当对象开辟堆内存（实例化对象），那么属性才会进行内存的分配，如果使用了没有实例化的对象。



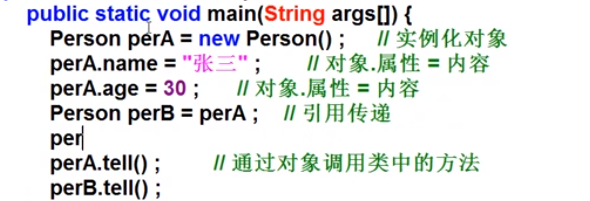
相当于此时只是声明了对象，但是并没有进行实例化，于是编译没有问题，而运行则会报空指针。

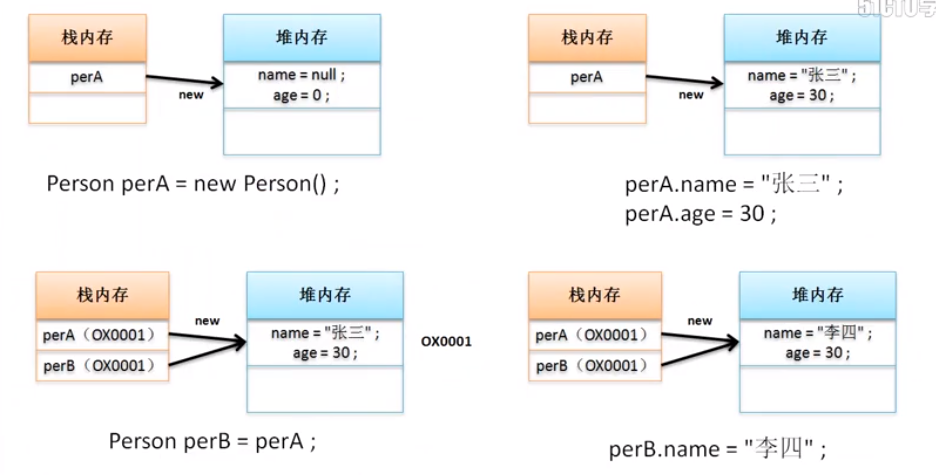
现在既然已经开辟了一个对象了，那么也就可以开辟两个对象。



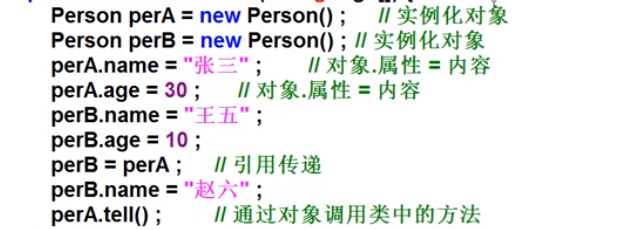
1. 初步分析引用传递

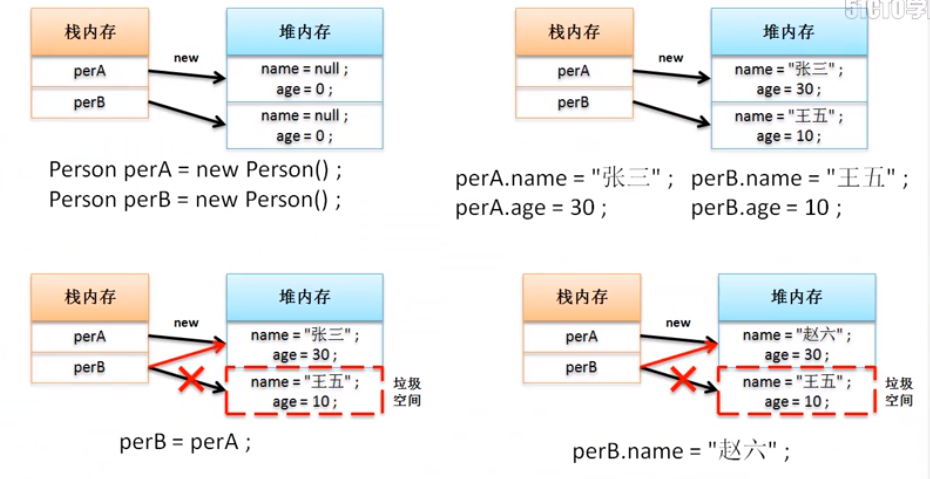
引用传递是在引用数据类型上所用的一个操作定义，其是java的精髓。





以上是采用了声明对象的方式进行了引用数据类型的接收，那么如果说此时两个对象都已经各自分别实例化并设置内容呢？

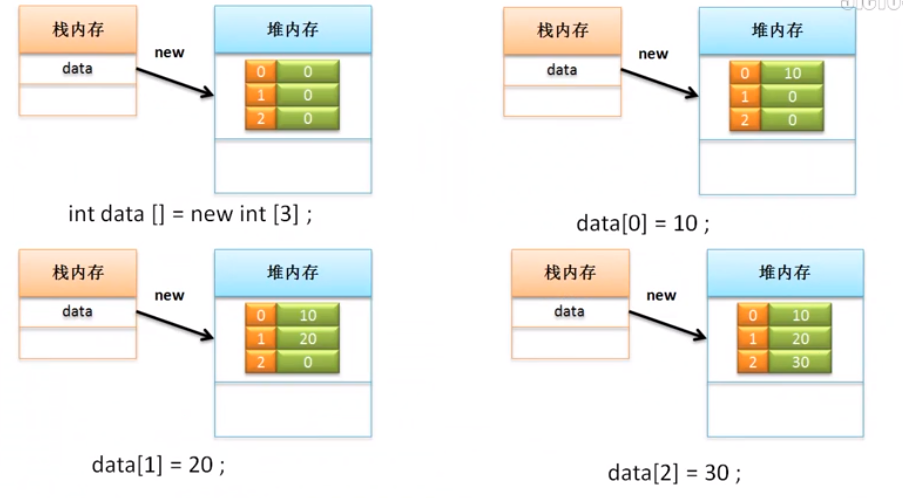




通过以上的分析可以发现几点：

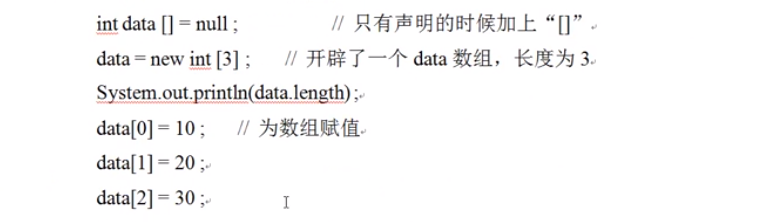
* 使用关键字new永恒可以开辟新的堆内存空间，堆内存保存的就是对象属性信息。
* 栈内存只能够保存一块对内存的使用地址；
* 引用传递的本质在于同一块堆内存可以被不同的栈内存所指向；
* 在发生引用传递时，如果操作的栈内存原本有堆内存指向，那么堆内存就意味着改变内存指向；
* 如果某一块堆内存没有任何的栈内存所指向，那么此空间将成为垃圾空间，会被GC回收。

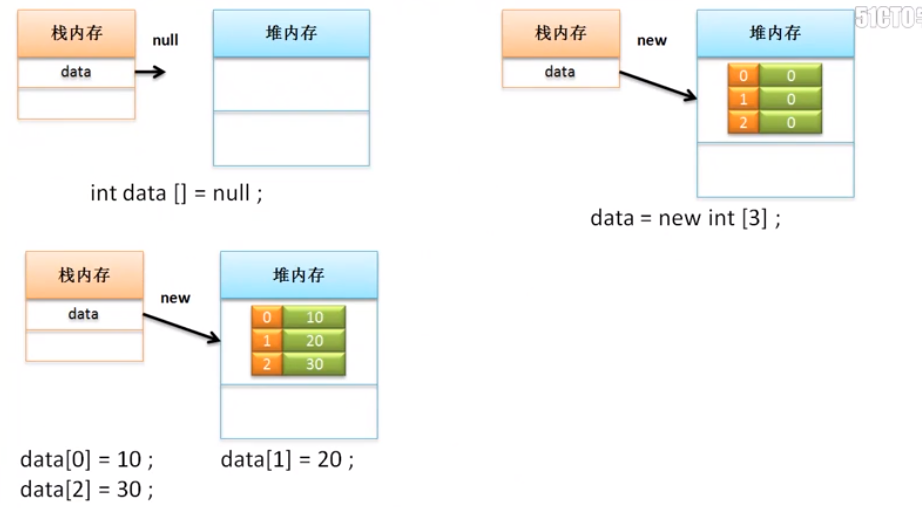
1. 数组引用分析



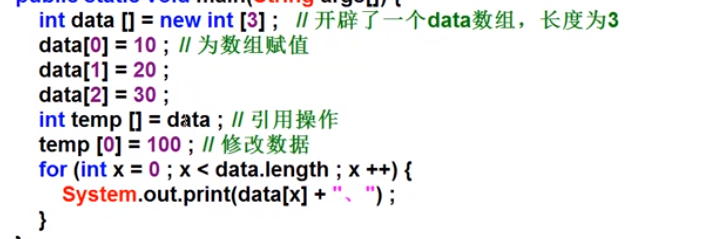
以上的流程与普通对象本质上是没有什么区别的，唯一的区别是在于普通的类对象时保存属性，利用属性名称来操作，但是数组保存的是一组内容，利用索引来操作。

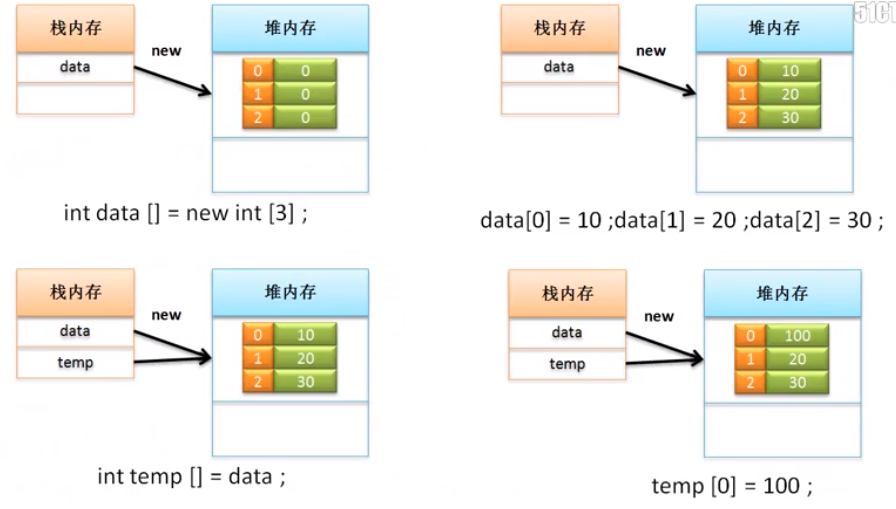
那么既然此关系可以弄明白，所谓的分步操作也就是一样的过程。





引用的本质是：同一块堆内存被不同的栈内存所指向。下面分析数组的引用操作：

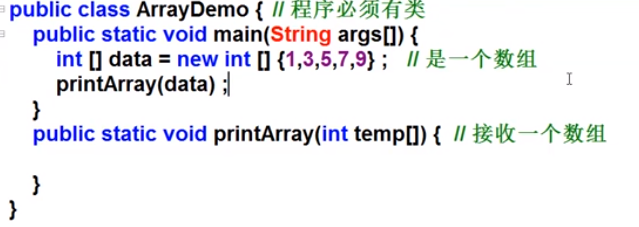


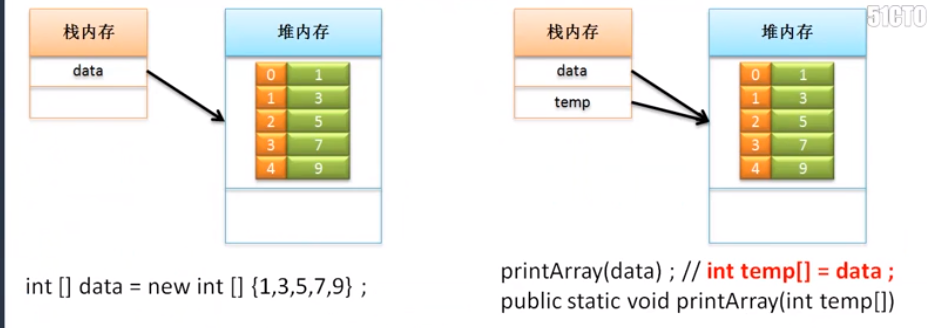


### 数组与方法的引用操作

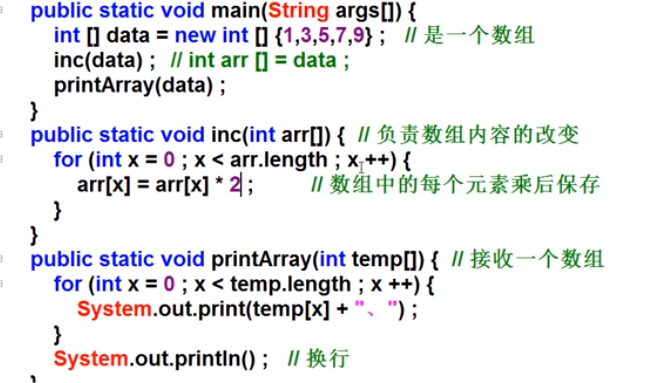
在数组与方法进行传递操作过程中，实际上就属于引用传递。在数组与方法间的操作中，主要考虑两种形式：方法接收数组、方法返回数组。不管是接收还是返回，最终操作的一定是一块堆内存的使用。

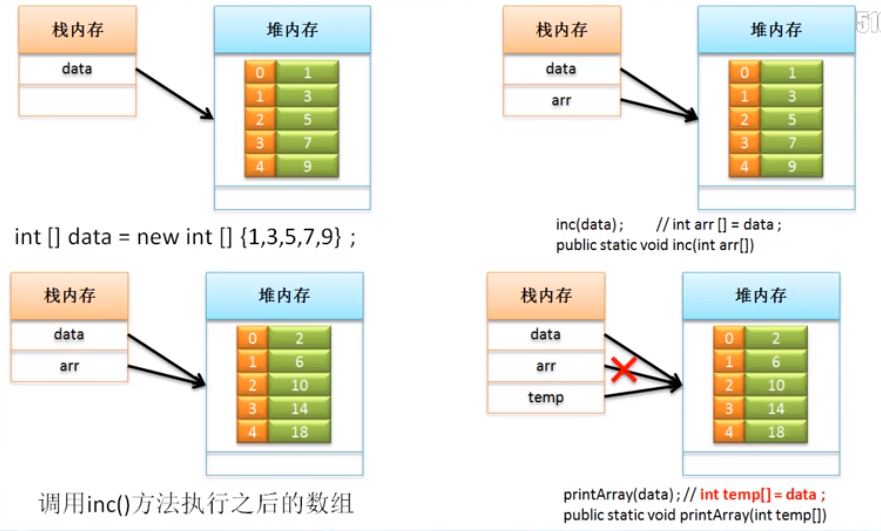
1. 利用方法接收数组



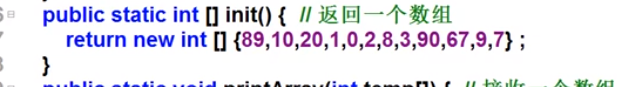


整个的操作过程与数组的引用接收完全是一样的。

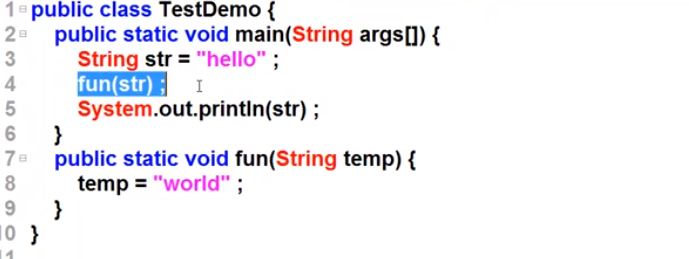


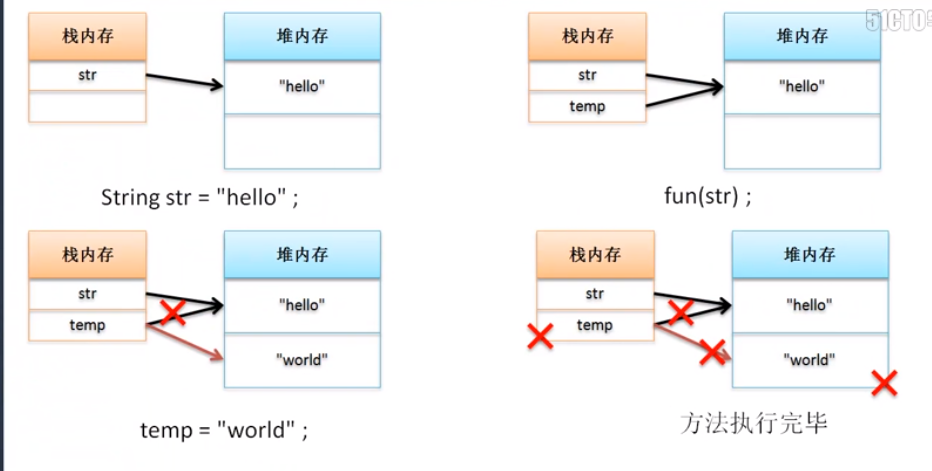


在大部分情况下，如果某一方法可以接收的数据类型是引用数据类型，那么往往都不需要设置返回值。



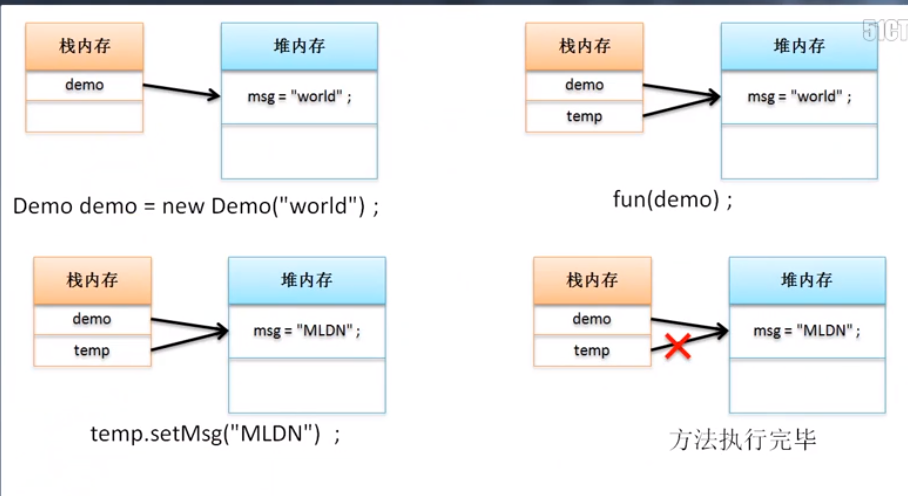
1. 引用传递进阶分析





String字符串堆空间是不可变的，temp=”world”是一个新对象。基本类型和String方法内的引用结果一致。

类对象在方法间的引用：



但是以上的流程并不是正规的操作，正确的应该如下：

