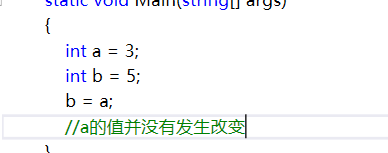
**值类型**

Int为值类型，那么我们现在写下如下代码：



（这里可以看到 a的值赋给了b，这里a的值是不会发生改变的）

**详细原理：**

**这里 int a = 3；其实就是a这个变量里的内存的空间存了一个为3的值，并且在栈中有属于a的地址**

**这里 int b = 5；其实就是b这个变量里的内存的空间存了一个为5的值，并且在栈中有属于b的地址**

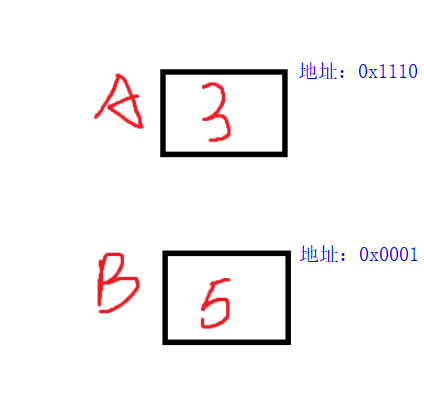
**当我们进行赋值的时候，其实就是a在内存空间里的值，复制了一份，然后给b的变量，两个变量在栈中的地址不变**

**通俗原理：**

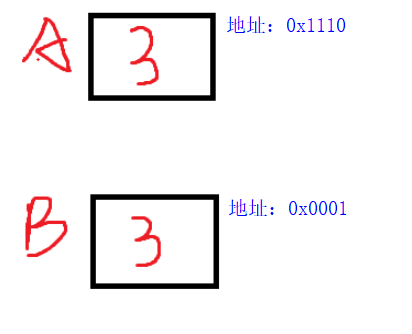
**使用的时候则当成是复印别人的笔记，变成自己的笔记，当我们对自己的笔记进行修改的时候，别人的笔记也不会给修改**

**具体原理图：**

1. 我们申明了int a 和int b的变量 分别为3 和 5 ，这两个空间存放了申明的值

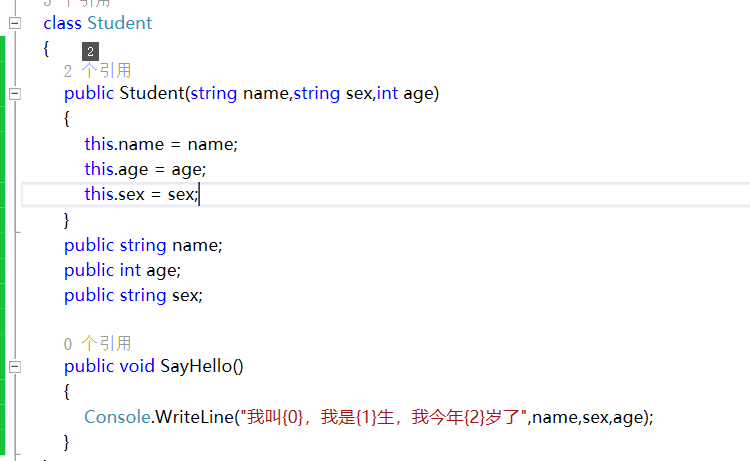


1. b = a，这时候a的值会复制一份出来，并赋值给了b的变量，所以这时候a的值是没改变

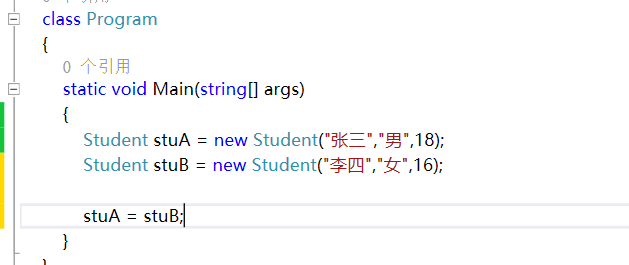


**引用类型**

类为引用类型，下面我们定义了一个Student的类，并且再Program类中实例化并调用



(这个是Student类)



(这个是Program类)

**由代码图可以看出，stuB的内存地址的值赋给了stuA，所以当stuA的值发生了改变stuB的值也会给改变**

**详细原理：**

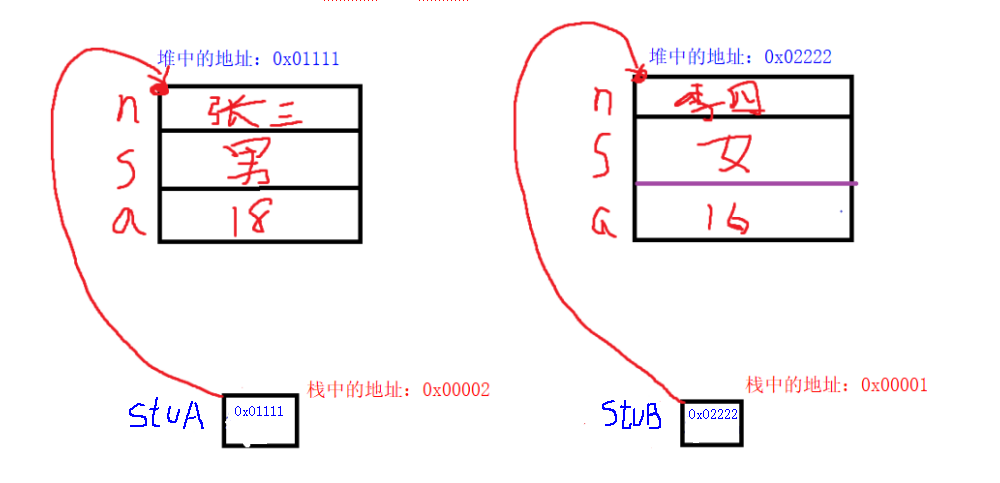
1. **在** Student stuA = new Student("张三","男",18); **的时候，便是在内存的堆中新开辟了一块内存空间用来存放Student类中的对象，并且变量 stuA在内存的栈中开辟了一块新的内存空间并且有属于stuA的栈地址，stuA里面存放了堆中的地址，并把指针指向堆中对象的首地址**
2. **在** Student stuB = new Student("李四","女",16);**的时候，便是在内存的堆中新开辟了一块内存空间用来存放Student类中的对象，并且变量 stuB在内存的栈中开辟了一块新的内存空间并且有属于stuB的栈地址，stuB里面存放了堆中的地址，并把指针指向堆中对象的首地址**

**通俗原理：**

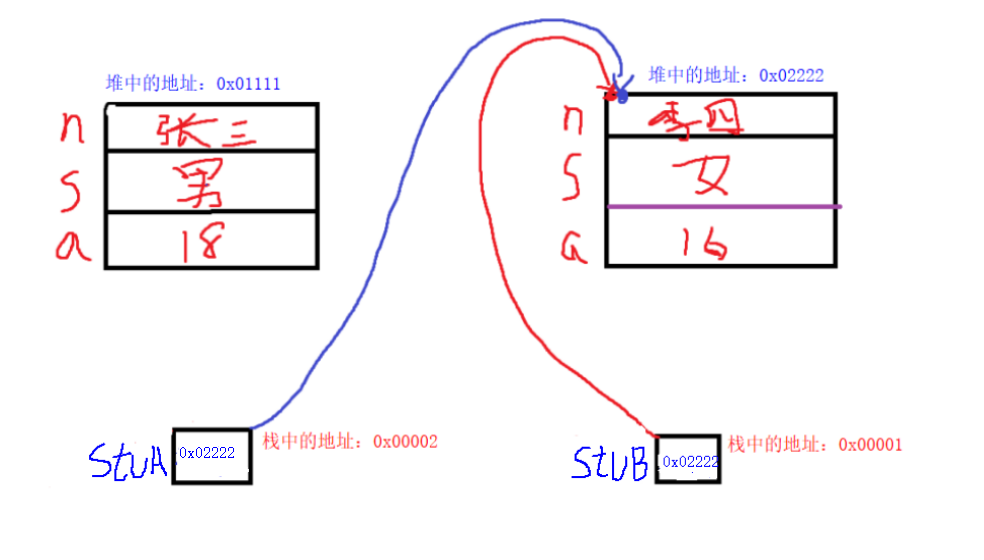
**当成是借别人的笔记，当我们在别人的笔记进行修改的时候，别人的笔记肯定也会给需改**

**具体原理如图：**

1. **当我们实例化一个类的时候，并且对他进行赋值，系统则是在堆中开辟了一块新的内存空间，并且里面存放该类中的属性 方法等等，实例名则在内存的栈中开辟了一块新的内存空间，并且有属于这个实例名的地址，实例名里存放了所指向的堆中的地址，如上图代码图，我们是进行了两次类的实例化，变量为：stuA 和stuB**



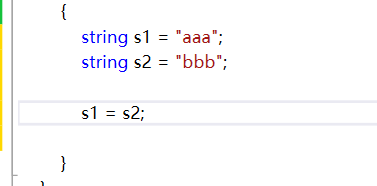
1. **当我们在进行**stuA = stuB;**赋值的时候其实是把stuB所存放的地址赋值给了stuA，那么指向就发生了变化了**



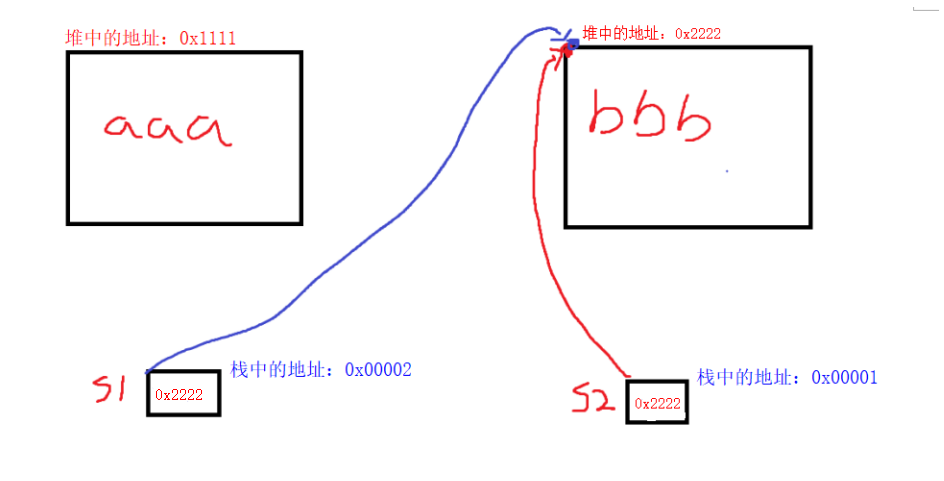
1. **当stuA的指向stuB所指向的内存空间的地址的时候，这时候我们改变该内存空间的值，那么stuB也会发生改变，因为他们指向的内存空间的地址是一样的！**

**注意：string类型虽然为引用类型但是却很特殊**

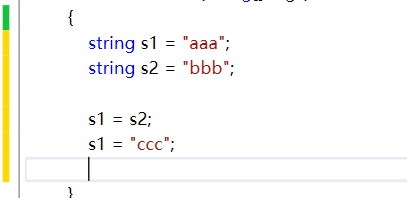
**如下代码图:**



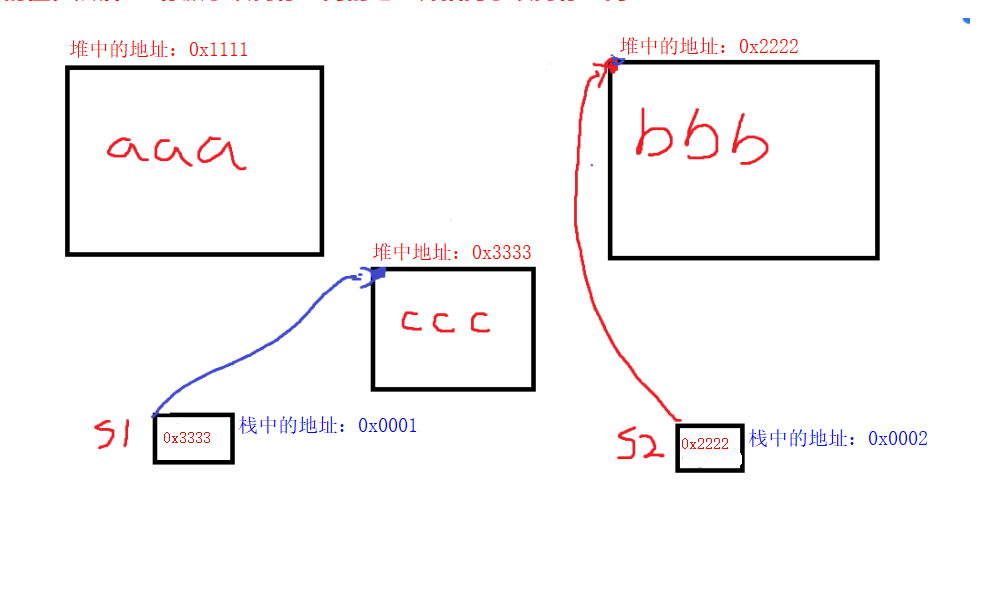
1. **通过刚刚的讲解，这时候s2的地址赋给了s1，也就是说这时候s1指向的是s2所指向的内存空间**

****

1. **这时候我们对s1的值改变：**

****

**当 s1 = “ccc”的时候，并没有改变s2所指向的内存空间的值，而是新开辟了一块存放”ccc”的值，然后s1存放了该内存空间的地址并指向了该内存空间**

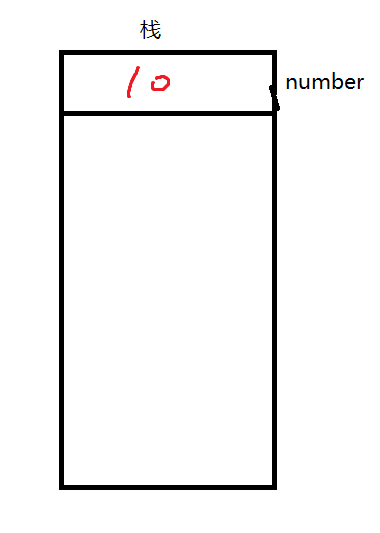


**栈和堆**

**栈：**

**值类型的变量和值都是存储在栈中**，比如说：int number = 10;

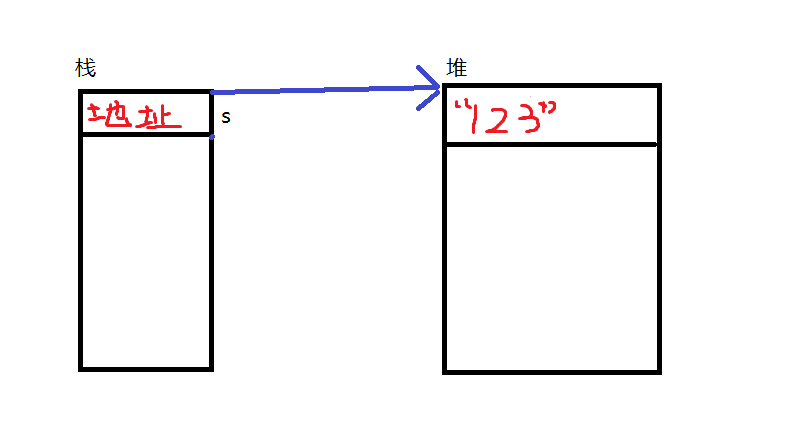
内存空间图：



**堆：**

**引用类型的值存储在堆当中，而变量名和地址存储在栈当中**，比如：string s = “123”;

内存空间图：



**值传递：传递值**

**引用传递：传递地址**

**值传递变成引用传递通过ref和out：**

**值类型均派生自：ValueType，而ValueType派生与Object**

**引用类型均派生自：Object**