**在Java中类型可分为两大类：值类型与引用类型。**值类型就是基本数据类型（如int ,double 等），而引用类型,是指除了基本的变量类型之外的所有类型（如通过 class 定义的类型）。所有的类型在内存中都会分配一定的存储空间(形参在使用的时候也会分配存储空间,方法调用完成之后,这块存储空间自动消失), **基本的变量类型只有一块存储空间(分配在stack中), 而引用类型有两块存储空间(一块在stack中,一块在heap中)**，在函数调用时Java是传值还是传引用，这个估计很多人至今都很糊涂，下面用图形与代码来解释：

　　在上图中引用类型在传参时不是在heap中再分配一块内存来存变量c 所指向的A(),而是让a 指向同一个A 的实例，这就与C++ 中的指针一样，先声明指针变量a,b,c,d 在传参的时候让a 指向c所指向的内存，让 d 指向 b 所指向的内存。很明显Java中的引用与C++中的指针在原理上是相类似的，但记住Java没有指针，只有引用。下面再通过一些具体的代码来讨论引用：

**1. 简单类型是按值传递的**

Java 方法的参数是简单类型的时候，是按值传递的 (pass by value)。这一点我们可以通过一个简单的例子来说明：

**package** test;

**public** **class** Test {

   //交换两个变量的值

**public** **static** **void** Swap(**int** a,**int** b){

**int** c=a;

      a=b;

      b=c;

      System.**out**.println("a: "+a);

      System.**out**.println("b: "+b);

   }

**public** **static** **void** main(String[] args){

**int** c=10;

**int** d=20;

      Swap(c,d);

      System.**out**.println("After Swap:");

      System.**out**.println("c: "+d);

      System.**out**.println("d: "+c);

   }

}

运行结果：

a: 20

b: 10

After Swap:

c: 20

d: 10

　　不难看出，虽然在 Swap (a,b) 方法中改变了传进来的参数的值，但对这个参数源变量本身并没有影响，即对 main(String[]) 方法里的 a,b 变量没有影响。那说明，参数类型是简单类型的时候，是按值传递的。以参数形式传递简单类型的变量时，实际上是将参数的值作了一个拷贝传进方法函数的，那么在方法函数里再怎么改变其值，其结果都是只改变了拷贝的值，而不是源值。

**2. 什么是引用**

　　Java 是传值还是传引用，问题主要出在对象的传递上，因为 Java 中简单类型没有引用。既然争论中提到了引用这个东西，为了搞清楚这个问题，我们必须要知道引用是什么。

　　简单的说，**引用其实就像是一个对象的名字或者别名 (alias)**，一个对象在内存中会请求一块空间来保存数据，根据对象的大小，它可能需要占用的空间大小也不等。访问对象的时候，我们不会直接是访问对象在内存中的数据，而是通过引用去访问。引用也是一种数据类型，我们可以把它想象为类似 C++ 语言中指针的东西，它指示了对象在内存中的地址——只不过我们不能够观察到这个地址究竟是什么。

如果我们定义了不止一个引用指向同一个对象，那么这些引用是不相同的，因为引用也是一种数据类型，需要一定的内存空间（stack，栈空间）来保存。但是它们的值是相同的，都指示同一个对象在内存（heap，堆空间）的中位置。比如：

      String a="This is a Text!";

      String b=a;

　　通过上面的代码和图形示例不难看出，a 和 b 是不同的两个引用，我们使用了两个定义语句来定义它们。但它们的值是一样的，都指向同一个对象 "This is a Text!"。**但要注意String 对象的值本身是不可更改的 (像 b = "World"; b = a; 这种情况不是改变了 "World" 这一对象的值，而是改变了它的引用 b 的值使之指向了另一个 String 对象 a)**。

　　如图，开始b 的值为绿线所指向的“Word Two”，然后 b=a; 使 b 指向了红线所指向的”Word“.

**这里我描述了两个要点：**

**（1） 引用是一种数据类型（保存在stack中），保存了对象在内存（heap，堆空间）中的地址，这种类型即不是我们平时所说的简单数据类型也不是类实例(对象)；**

**（2） 不同的引用可能指向同一个对象，换句话说，一个对象可以有多个引用，即该类类型的变量。**

**3. 对象是如何传递的呢**

　　随着学习的深入，你也许会对对象的传递方式产生疑问，即对象究竟是“按值传递”还是“按引用传递”?

（1）认为是“按值传递”的：

**package** test;

**public** **class** Test {

**public** **static** **void** Sample(**int** a){

         a+=20;

         System.**out**.println("a: "+a);

      }

**public** **static** **void** main(String[] args){

**int** b=10;

         Sample(b);

         System.**out**.println("b: "+b);

   }

}

运行结果：

a: 30

b: 10

    在这段代码里，修改变量 a 的值，不改变变量 b 的值，所以它是“值传递”。

（2）认为是“按引用传递”的：

**package** test;

**public** **class** Test {

**public** **static** **void** Sample(StringBuffer a){

         a.append(" Changed ");

         System.**out**.println("a: "+a);

      }

**public** **static** **void** main(String[] args){

         StringBuffer b=**new** StringBuffer("This is a test!");

         Sample(b);

         System.**out**.println("b: "+b);

   }

}

运行结果：

a: This is a test! Changed

b: This is a test! Changed

　　在Sample(StringBuffer)这个函数中，修改了引用 a 的值，同时 b 的值也变化了，所以它是“按引用传递”的！

　　那么对象（记住在Java中一切皆对象，无论是int a;还是String a;，这两个变量a都是对象）在传递的时候究竟是按什么方式传递的呢？其答案就只能是：**即是按值传递也是按引用传递，但通常基本数据类型（如int,double等）我们认为其是“值传递”，而自定义数据类型（class）我们认为其是“引用传递”。**

**4. 正确看待传值还是传引用的问题**

　　要正确的看待这个问题必须要搞清楚为什么会有这样一个问题。

　　实际上，问题来源于 C，而不是 Java。

**C 语言中有一种数据类型叫做指针，于是将一个数据作为参数传递给某个函数的时候，就有两种方式：传值，或是传指针。 在值传递时，修改函数中的变量值不会改变原有变量的值，但是通过指针却会改变。**

    void Swap(int a,int b){ int c=a;a=b;b=c;}

    void Swap(int \*a,int \*b){ int c=\*a;\*a=\*b;\*b=c; }

    int c=10;

    int d=20;

    Swap(c,d);    //不改变 c , d 的值

    Swap(&c,&d);  //改变 c , d 的值

　　许多的 C 程序员开始转向学习 Java，他们发现，**使用类似 SwapValue(T,T)(当T 为值类型时) 的方法仍然不能改变通过参数传递进来的简单数据类型的值，但是如果T时一个引用类型时，则可能将其成员随意更改。**于是他们觉得这很像是 C 语言中传值/传指针的问题。但是 Java 中没有指针，那么这个问题就演变成了传值/传引用的问题。可惜将这个问题放在 Java 中进行讨论并不恰当。

**讨论这样一个问题的最终目的只是为了搞清楚何种情况才能在方法函数中方便的更改参数的值并使之长期有效。**

**5. 如何实现类似 swap 的方法**

　　传值还是传引用的问题，到此已经算是解决了，但是我们仍然不能解决这样一个问题：如果我有两个 int型的变量 a 和 b，我想写一个方法来交换它们的值，应该怎么办？有很多方法，这里介绍一种简单的方法：

**package** test;

**public** **class** Test {

**public** **static** **void** Swap(**int**[] a){

**int** c=a[0];

         a[0]=a[1];

         a[1]=c;

      }

**public** **static** **void** main(String[] args){

**int**[] a=**new** **int**[2];

         a[0]=10;

         a[1]=20;

         Swap(a);

         System.**out**.println(a[0]);

         System.**out**.println(a[1]);

   }

}

　　通过数组可以方便的实现值类型的数据源的交换，不过还有一种方法是将所有变量封装到一个类里面去，通过引用类型来实现。