Java中的形参和实参传递问题：值传递

# Java中参数是单向的、值传递

## 首先明确的是形参和实参的值传递方向是单向传递，即实参传向形参。

### 基本数据类型：值传递的方式，实参变量把值传递给了形参变量，形参变量做运算，实参变量不改变，只可以通过return语句返回值。

### 引用数据类型：传递的是地址，即实参引用变量把地址传给了形参引用变量，此时形参和实参引用变量都是指向同一个对象，故在不改变形参引用所指对象时，形参的操作会引起实参对象的改变。但是如果形参改变了指向的对象，则形参操作和实参对象就没有了任何关系。(地址也是一个值，也属于值传递)

## 示例：

Student s = new Student(4);

xiugai(s);

System.out.println(s.age);

判断输出值？

第一种情况：

private static void xiugai(Student s) {

s.setAge(19);

}

输出结果：19。传递的是对象s的地址，s通过setAge方法修改了age的大小。

第二种情况：

private static void xiugai(Student s) {

s = new Student(18);//改变了形参的指向对象，相当于对实参对象没有操作

s.setAge(19);

}

输出结果：4，因为形参s引用变量在操作之前，指向了新的对象，地址发生改变。

# String类

常考点：**对于String类，它们是存放的是常量，内容不可以改变，即无法修改，利用replace方法如果修改了，则是创建了新的String对象，如果没有修改返回仍然是原来的那个对象，str = “dd”；就相当于创建了新的对象，然后引用指向了新的对象。**

**实现原理：String中的value属性是final修饰的，不可被修改。当然String类也是final修饰的，不可被继承。**

**如：**

**String str = "dongxi"; change(str);**

**System.out.println(str);//输出dongxi**

**private static void change(String str) {**

**str = "beijing";//str引用指向了新的String对象（beijing）**

}

# 基本数据类型的包装类

**Integer等八种基本数据类型的包装类：和String一样，存放的是常量**。

例如:

Integer i = 9;

change(i);

System.out.println(i); //输出为9

private static void change(Integer i) {

i = 4;// 有自动拆箱和自动装箱功能

}