Java中的形参和实参传递问题

1. **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。**
2. ***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。**
3. ***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。**

**例如：Student s = new Student(4);**

**xiugai(s);**

**System.out.println(s.age);**

**第一种情况：输出为19，因为传递的是引用，实参引用和形参引用指向的是同一个对象**

**private static void xiugai(Student s) {**

**s.setAge(19);**

**}**

**第二种情况：输出为4，以为形参引用变量在操作之前，指向了新的对象。**

**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);**

**}**

1. **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。**

**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）。**

**}**

1. **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**

1. **注意：8中原生数据类型是值传递的方式，而对于引用都是地址传递。只不过String类比较特殊而已，因为字符串对象一旦创建就不可以改变了，即使是改变，也是建立了一个新的字符串对象（利用new的方式），引用变量的指向已经发生了变化。如果利用StringBuilder可以进行修改字符串。如：**

**StringBuilder sb = new StringBuilder("nihaoma");**

**ssss(sb);**

**System.out.println(sb);//nihaoma你猜啊;改动了**

**String str = "nihaoma";**

**System.out.println(str);//输出为nihaoma，没有改动**

**}**

**private static void ssss(StringBuilder sb) {**

**sb.append("你猜啊"); }**

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

**str.replace("hao","好"); }**