Java03

Task1

- 1
  - byte:整数 1byte 范围-128到127
  - short:整数 2byte 范围-32,768到32767。
  - int:整数 (跟编译环境有关 这里我以32位为例 32bite 也就是 4byte) 范围-2^{31}到2^{31}-1
  - long:整数 8byte 范围-2^{63}到2^{63}-1
  - char:字符
  - float:浮点
  - double:浮点
  - boolean:布尔值

2. 类型转换 举个简单例子

```
public class Hello{
    public static void main(String[] args){
        int a = 1/2;
        //1与2均为整数 故得到的a也是整数 即a = 0

        double b = 1.0/2;
        //1.0是浮点数 浮点类型可以用来表示整数 比int更广泛 所以浮点数与整数做运算时整数就会自动转化

        int c = (int)1.0/2;
        //这里1.0/2得到0.5 把浮点数直接赋给int型的c就会报错 java是一种强类型语言 因此需要用(int)来
    }
}
```

3.

```
int a=4
char c='0';
int b=a+c;

//请回答这个过程涉及到的是自动类型转换还是强制类型转换，b的值是多少，为什么会是这个值。
```

自动类型转换 字符在参与算数运算时会自动转换为数 Java使用的是Unicode编码 每一个数字都对应一个字符 在编码表对应下'0'对应48 故b = 52;

4.

```
Integer x = new Integer(18);
Integer y = new Integer(18);
System.out.println(x == y);

Integer z = Integer.valueOf(18);
Integer k = Integer.valueOf(18);
System.out.println(z == k);

Integer m = Integer.valueOf(300);
Integer p = Integer.valueOf(300);
System.out.println(m == p);
```

false ture false

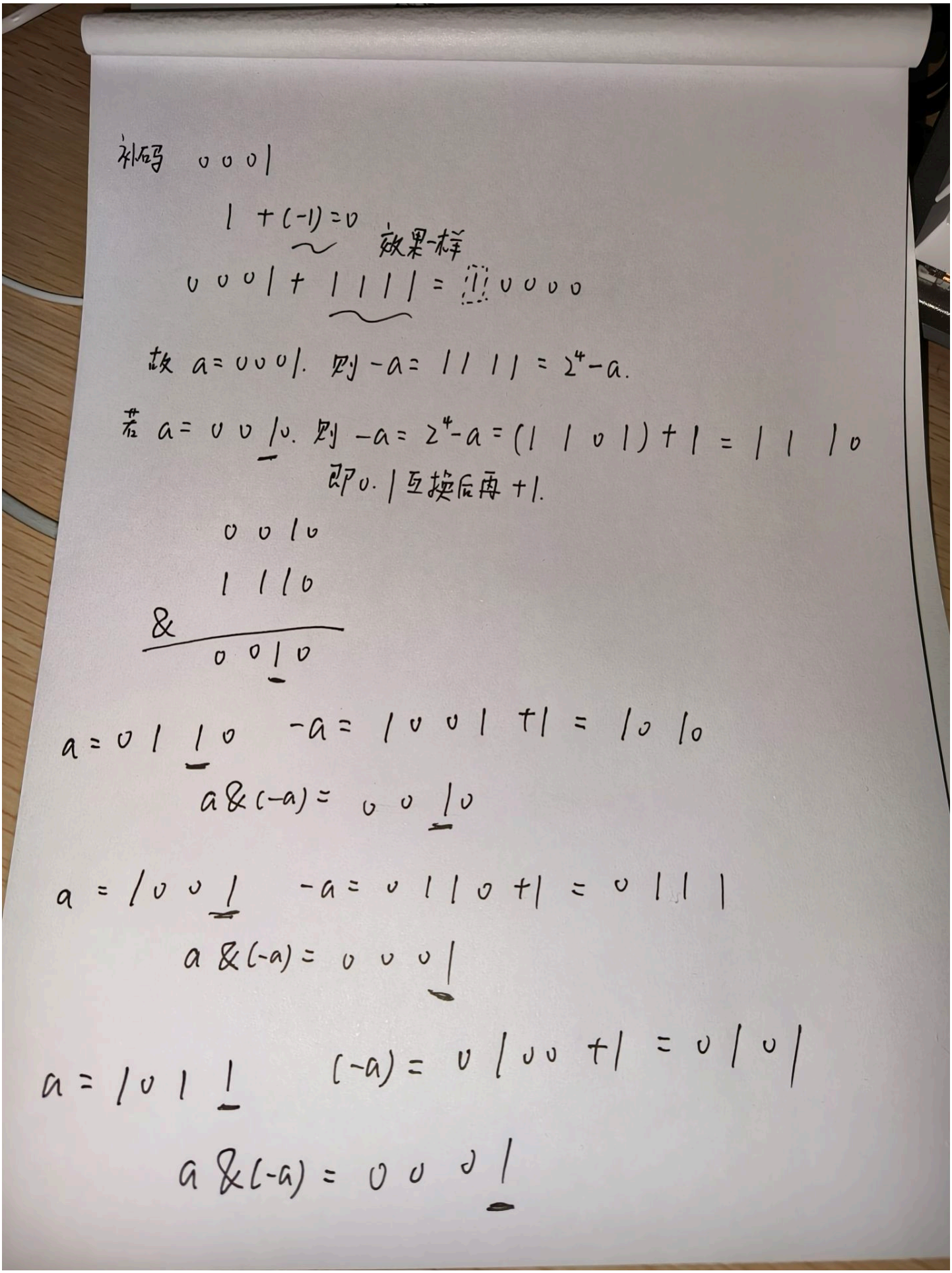
- 第一个我是最新版本java 已经取消了这种用法 不过可以类比数组 字符串 都需要new一个数值出来交给这个对象 对于他们 ==比较的不是这两个对象的数值是否相同 而在比较他们是否管理同一个数 尽管18和18一样 但他们是两个18而非同一个18 只有当两个对象同时管理一个数据时才会输出false
- 第二个和第三个放在一起说 Integer默认缓存了-128到127的对象 如果赋值在这个范围内就会直接调用 如上面所说 两个对象管理的是同一个数值 但如果超出了这个范围就会创建独立的新的Integer对象

Task2

5.

```
int a = 5 ;
int b = 7 ;
int c= (++a) + (b++)
System.out.println( c );
System.out.println(a+" "+b);
```

c = 13 6 8 a++即a = a + 1 并返回a自增前的结果 而++a 返回a自增后的结果
第二个输出的+用于连接字符串和数字 不是数学运算的加法
6.

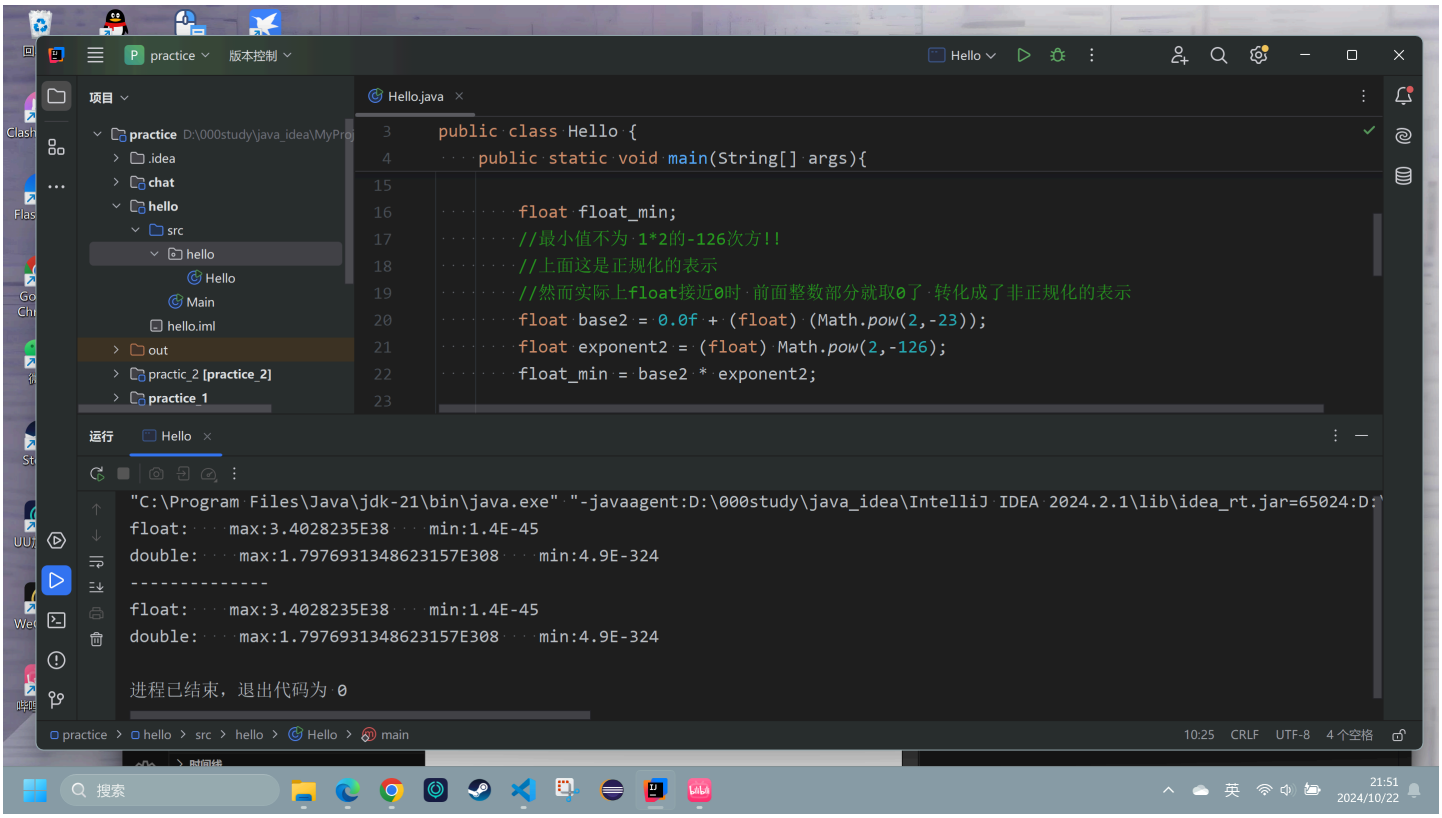


应该取得是a最低位上的1 -a要0.1对换再加1最后那肯定-a和a最低位上都是1 其他位因为对换 最低位1之前的都不一样 之后的都是0 所以他们都取0

Plus Content

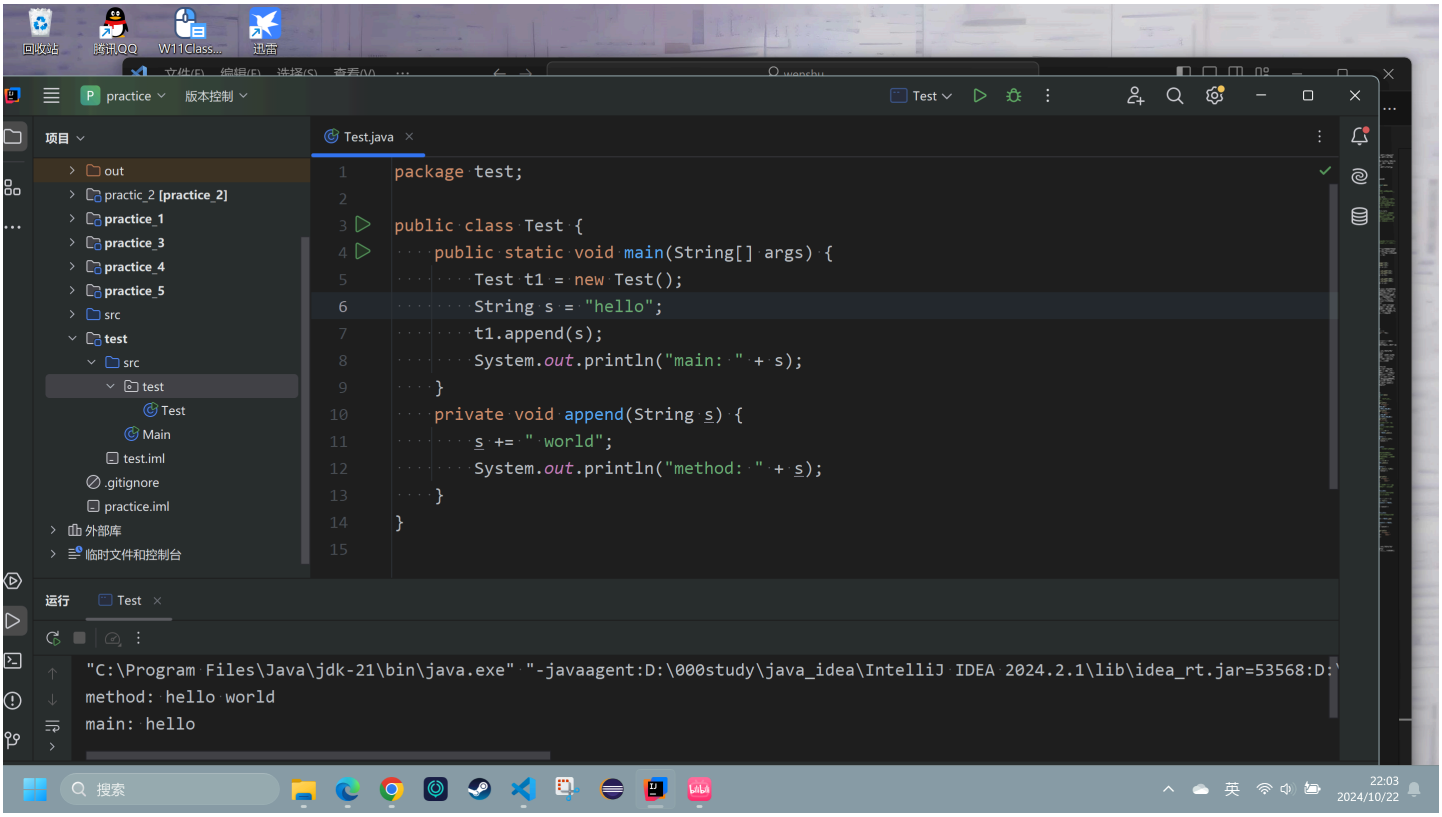
7. 先简要阐述一下计算思想
   以float为例 float用32位比特储存 1位符号位(0为正 1为负) 23位尾数位 即把浮点数化为二进制表示1.xxxx 小数点后面有23位 然后是指数位 因为是二进制 有8位用于表示2的n次幂的指数 考虑到指数可以取负 所以有127的偏移量 并且0和255端点取了特殊值(无穷 不存在) 所以n的范围是-126到127

```java
package hello;

public class Hello {
    public static void main(String[] args){
        //这里先查看一下float和double的最大值
        System.out.println("float:    "+"max:"+Float.MAX_VALUE+"    min:"+Float.MIN_VALUE);
        System.out.println("double:    "+"max:"+Double.MAX_VALUE+"    min:"+Double.MIN_VALUE);
        System.out.println("--------------");
        //下面我们自己来计算一下
        float float_max;
        //最大值为 (1+(1-2的-23次方))*2的127次方
        float base1 = 1.0f + (float) (1 - Math.pow(2,-23));
        float exponent1 = (float) Math.pow(2,127);
        float_max = base1 * exponent1;

        float float_min;
        //最小值不为 1*2的-126次方!!
        //上面这是正规化的表示
        //然而实际上float接近0时 前面整数部分就取0了 转化成了非正规化的表示
        float base2 = 0.0f + (float) (Math.pow(2,-23));
        float exponent2 = (float) Math.pow(2,-126);
        float_min = base2 * exponent2;

        System.out.println("float:    "+"max:"+float_max+"    min:"+float_min);
        //下面double 符号位1位 指数位11位 尾数52位 偏移量1023
        double double_max;
        //最大值为 (1+(1-2的-52次方))*2的(2^11-2-1023)=1023次方
        double base3 = 1.0 + (1 - Math.pow(2,-52));
        double exponent3 = Math.pow(2,1023);
        double_max = base3 * exponent3;

        double double_min;
        //最小值位 2的-52次方*2的-1022次方
        double base4 = Math.pow(2,-52);
        double exponent4 = Math.pow(2,-1022);
        double_min = base4 * exponent4;
        System.out.println("double:    "+"max:"+double_max+"    min:"+double_min);

    }
}
```



可以看到计算输出结果和直接调用结果是一样的

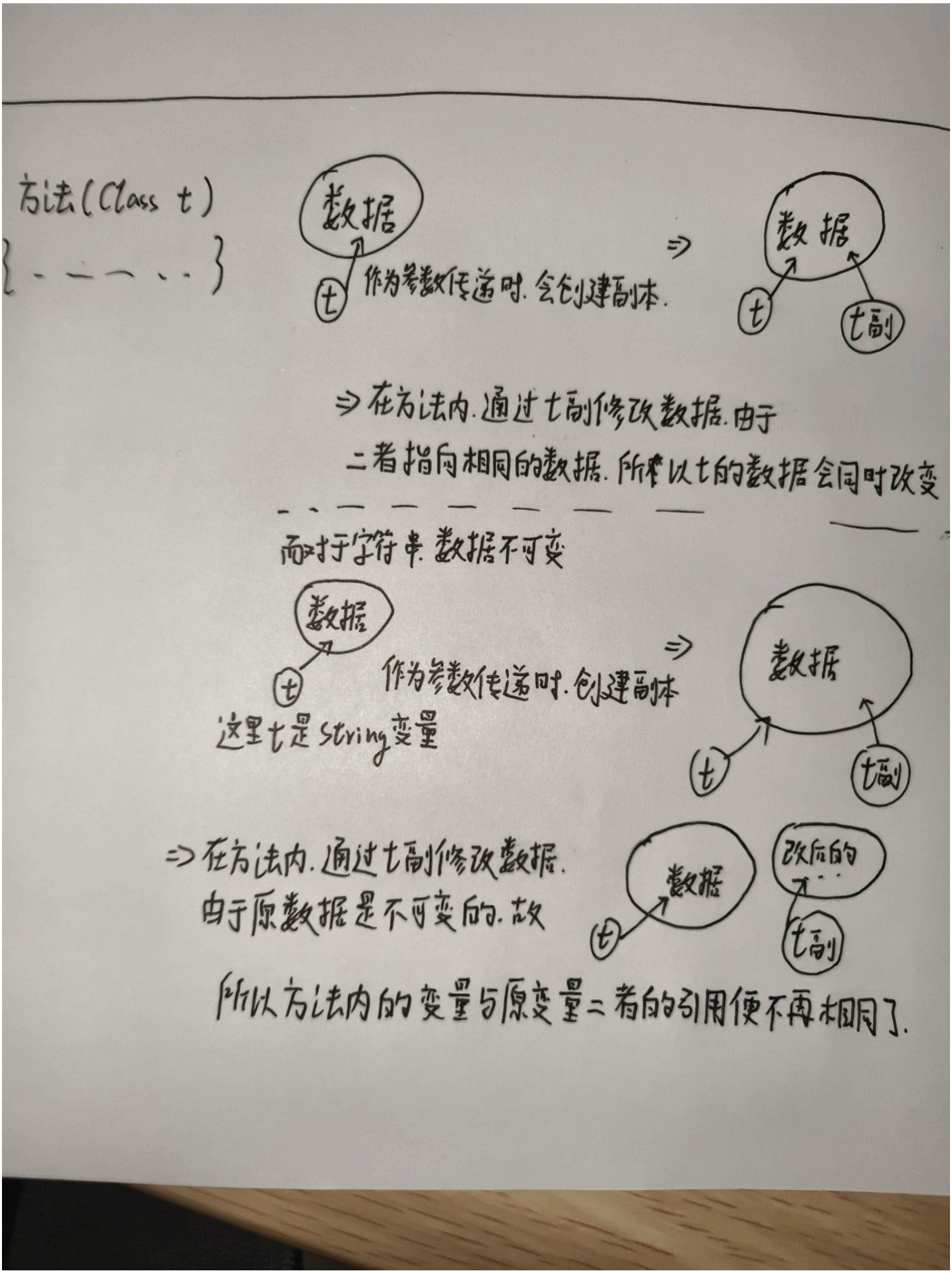
8.



这个结果看起来好像在说String是值传递 传进append函数的只是一个值 所以才会有上面的输出结果 其实这种说法是不对的 String s = "Hello"的完整写法应该是String s = new String("Hello") s是指向这个内容为"Hello"的String变量的 所以String应该是引用传递 这里输出结果是由String的不可变性造成的 例如下面这个代码

```
package test;
public class Test {
    public static void main(String[] args){
        String s = new String("hello");
        s += " world";
        System.out.println(s);
    }
}
```

输出结果是hello world 并不是字符串s的后面加上了world 而是原本s指向"hello"这个字符串 现在s指向了"hello world"这个字符串 "hello world"是一个独立于原来的"hello"的一个新创建的字符串 也就是s += "world"实际上是新创建了一个"hello world"字符串 然后让s指向它 原来的"hello"字符串仍然存在 只是没有变量引用它而已了
对于任何对象的传递都是引用传递 并且传递的是该对象引用的副本 什么意思呢?只用文字叙述不太说的清楚 这里我画了一张图解 比较字符串和其他类传递的差异



这里可以清楚地解释为什么main里面的s仍然指向"hello" 而append方法里面的s指向"hello world"