**实验报告：**

1.以下表达式是否能够通过编译? 如果能够通过, 表达式的类型是什么? 表达式的值是什么? 请尝试解释原因.

1 == True

1 == true

0 == false

2 + "ab"

2.3 + "ab"

2 + 'a'

2 \* "ab"

2 \* 'a'

1 + 1.0

1/3

1.0/3

1.1

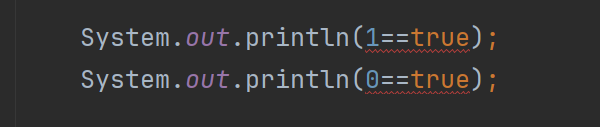
因为java 是语法严格的语言我们写错一个keyword 时编译不会通过的；

1.2

在java 语言数据类型要严格区分在这里 1==true 1 是整形类型而true 是boolea 类型字面变量所以编译不会通过；

1.3

0==false 也是跟上面一样的例子整型变量和boolean字面变量不匹配的会报错；



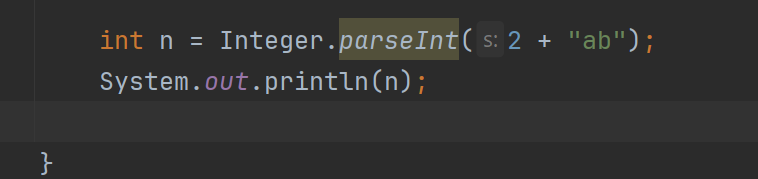
1.4

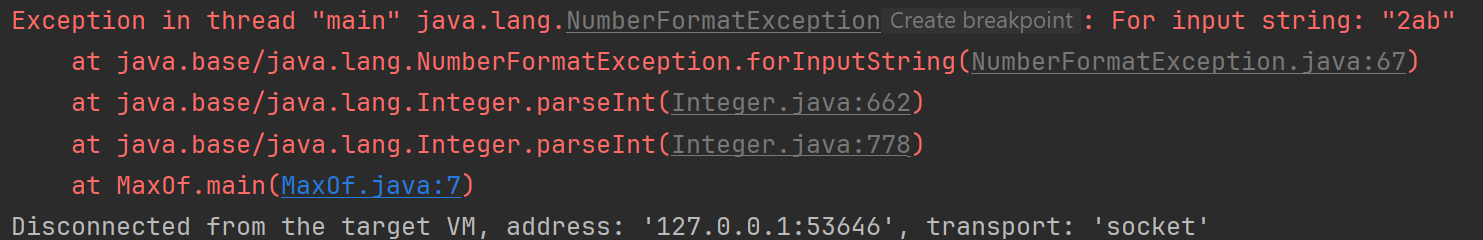
2+“ab” 如果我们把结果给我一个整形类型会把报错 因为整形类型不会接受字符串类型

如果是int型的包装类其实可以把字符串转换对应的数字但是在上面的例子是不行的

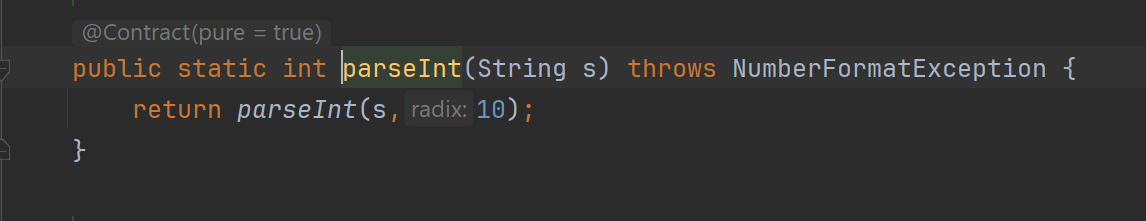
因为Integer.parsInt 只能接受0到9的数字；

如果用Integer.pasrInt 这个方法先会检查是否是字面字符如果是会捕捉：





因为parsint 的方法是如下：



所以当遇到字面字符会捕捉；

如果把结果给一个字符串型那会把前面的整形编译的时候自动转换字符串再合并他们呢

因为+在字符串操作时会合并两个字符串的。

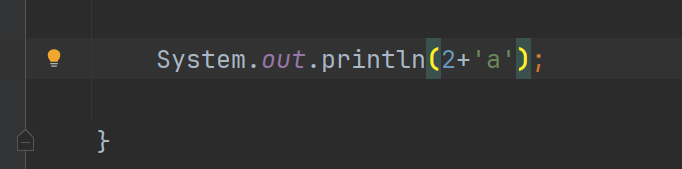
2.5+“ab” 也是一样的；

1.5

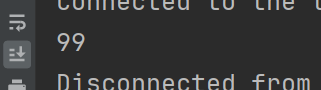
2+‘a’

我们知道根据unicode a的十进制code等于97 那么上面的表达式意味着 97+2 = 99

所以如果把结果给一个char型会输出 c 字符；



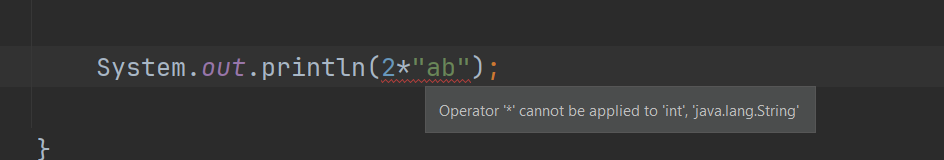
否则会输出对应unicode 十进制：



1.6

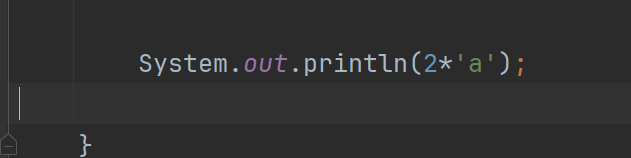
1.7

2\*“ab” 编译不会通过因为 \* 数学操作符不能对字符串进行操作；



1.8

2\* ‘a’ 像 2+‘a’ 一样的 a 的代码等于 97再程2 = 194

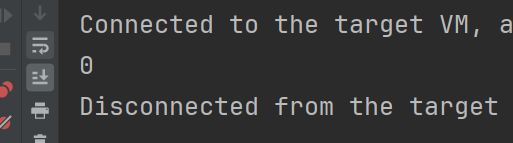


1.9

1+1.0 在编程语言有一个类型升级的概念 每当我们把类型小的数据加减乘除一个比自己大的数据类型是会自动的转换那个大的类型 再上面的例子1 是整形 1.0 是double 所以结果是2.0 类型为 double；

1.10

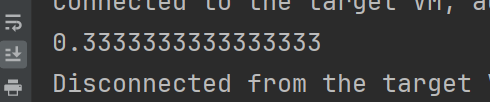
1/3 输出的结果可能不是我们希望得到的一个结果 因为两个数字都是整形所以结果也会是整形 则答案是 0



1.11

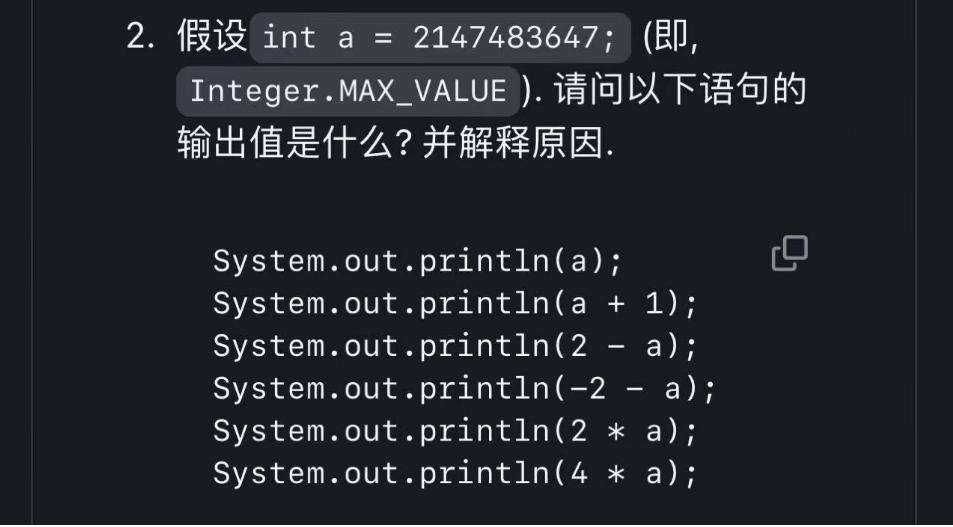
1.0/3

在这个例子 1.0 是double 所以结果也是一个double 则等于：

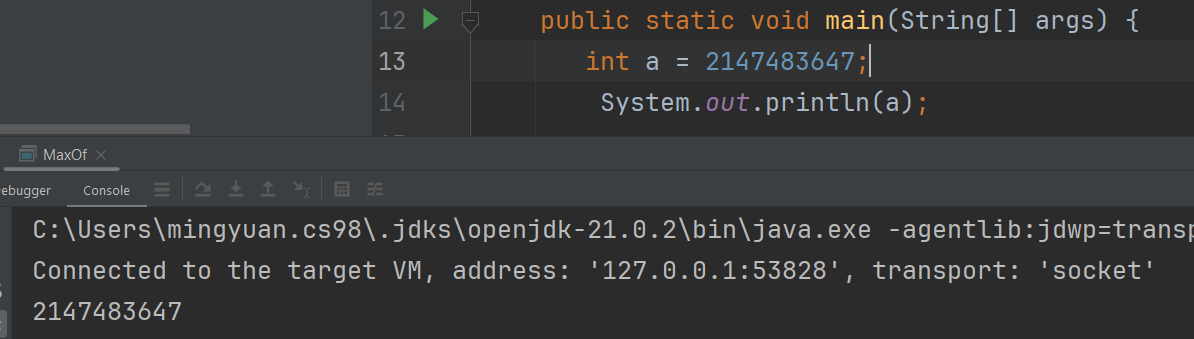


虽然3是整形但是因为一个是double 又因为int 小于double 所以结果也是为double；

**2**

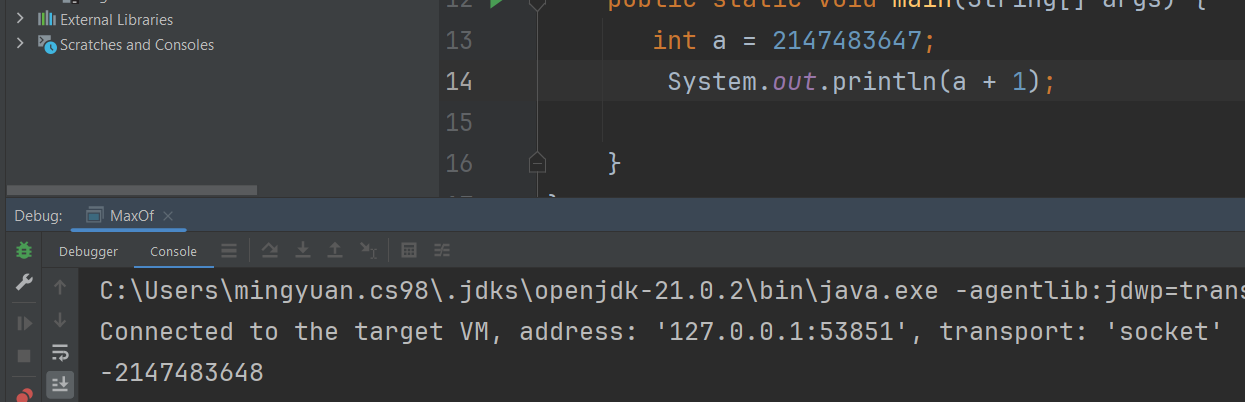


2.1



第一个会输出准确数字因为int 的最大数是2147483647 所以可以容纳这么大的一个数，

2.2



这个例子也很合理因为int最大值二进制表示法是：01111111111111111111111111111111

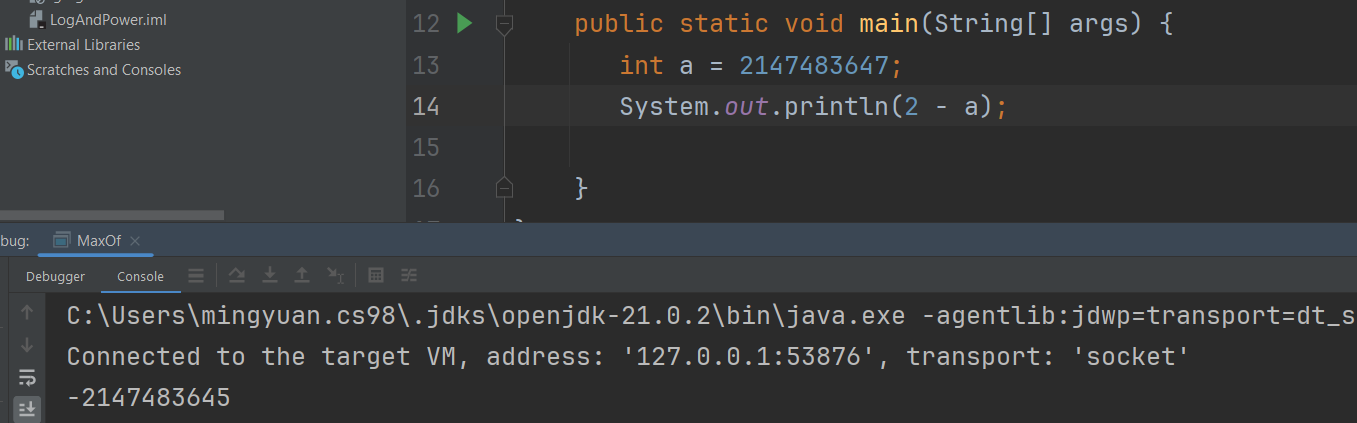
如果我们在加已会溢出所以会跳到int的另一个极端：

数字会变为：10000000000000000000000000000000

在表示二进制法是最高位也就是说第一位是该数字的符号 整形的话是0 负数的话是1

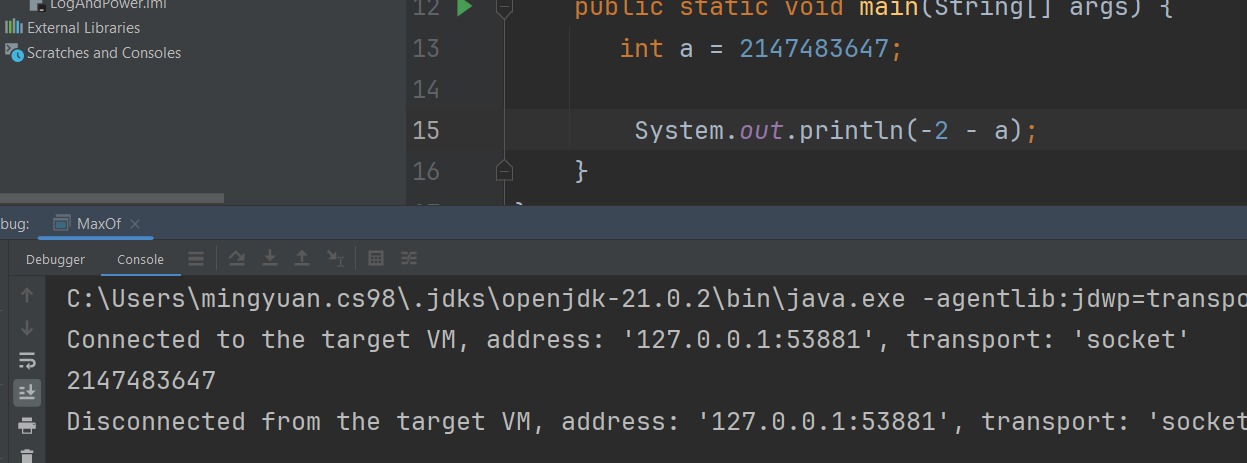
所以会输出int 的最大负数；

2.3



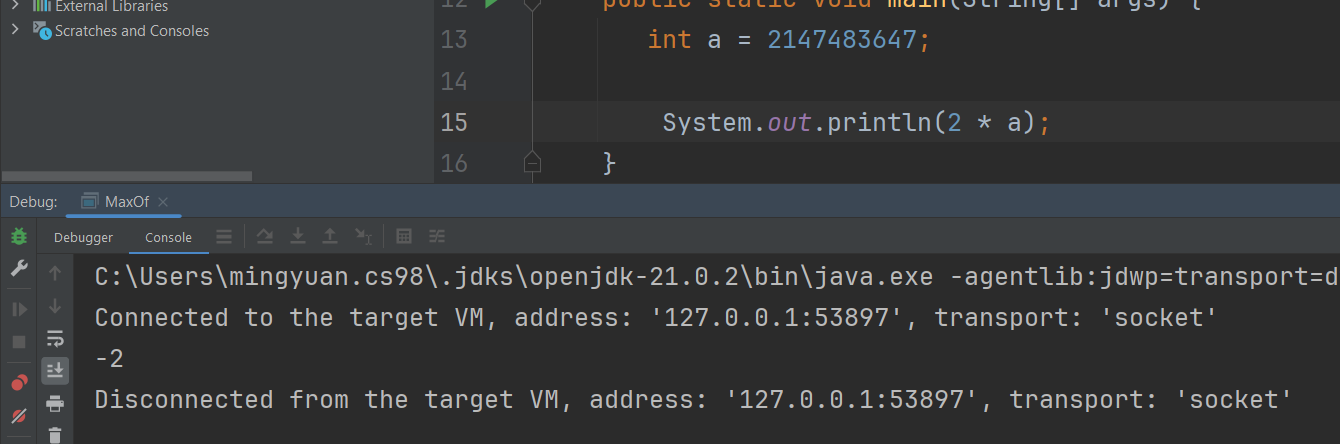
因为-2147483645 是在整形负数的范围内所以我们可以得到准确数字；

2.4



因为 --2147483649 是小于int能够表示的最小数所以会导致溢出，则跑到int的另一端然后输出int 整形能够表示的最大数;

2.5



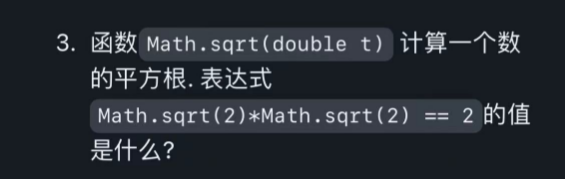
因为2\*int整形最大数所以它会跳到整形的另一端，而且数字\*2是左移一位所以

2\*011111111111111111111111111111111 = 111111111111111111111111111111110 = -2

最后一个\*4 也是一样的左移 2位

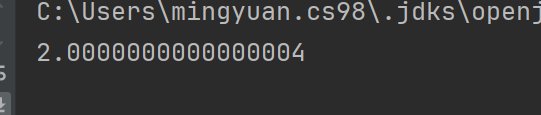
4\*011111111111111111111111111111111 = 111111111111111111111111111111100 = -4

3



在编程语言一个浮点数不能够完美的表示，因为数据类型像float 和double 分别能够比较准确的小点数是6位和15位 所输出的结果不会是2.0 则会输出false

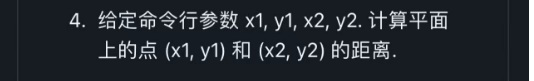
我们来测试这个：

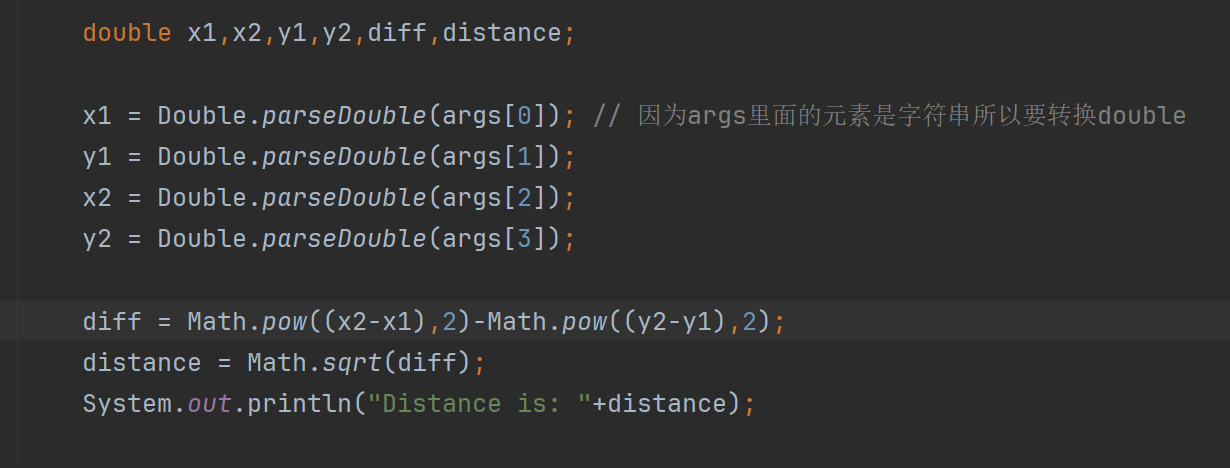




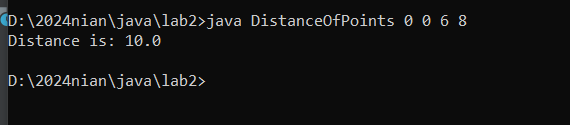
果然这个表达式不等于2 所以他们是不等价的；

4

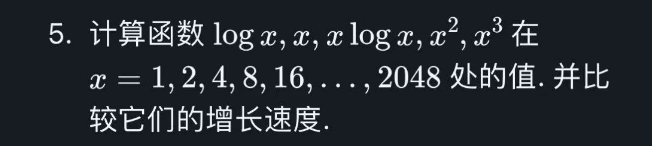




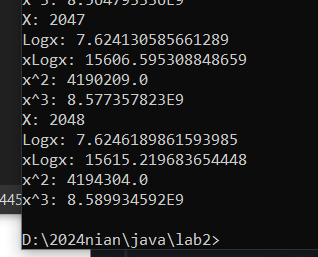
输出：

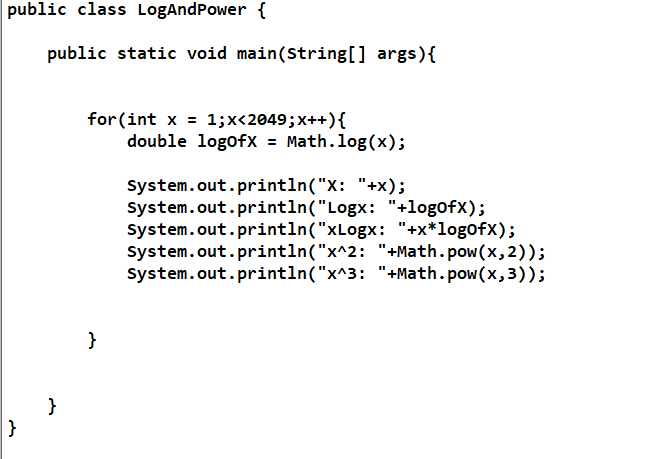


5

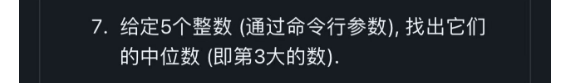


根据输出的结果我们可以的判断如果从小到大排序则：logx，x，xlogx，x^2,x^3;

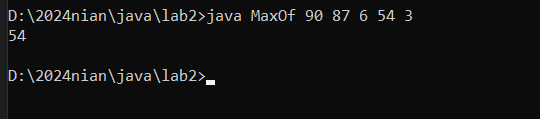




7



考虑到代码有点长的问题就没截图上传到这 但是会解释我的思路：



1.我的代码首先把命令行的字符串转换int 整形然后存储到一个新数组；

2.然后通过findMax 自定义的一个方法计算最大值；

3.在findMax方法另有一个max方法这个方法的作用是每次从给定小标开始找到最大值：

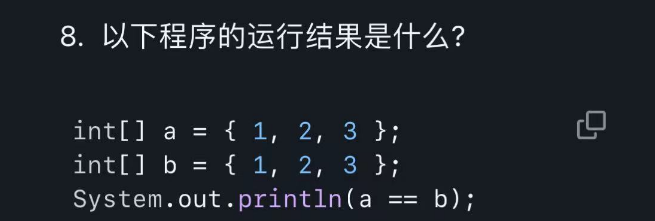
4.找到最大值把它跟前面的下表换位置；

每次找到最大值放到前面然后从下一个小标开始找最大值；

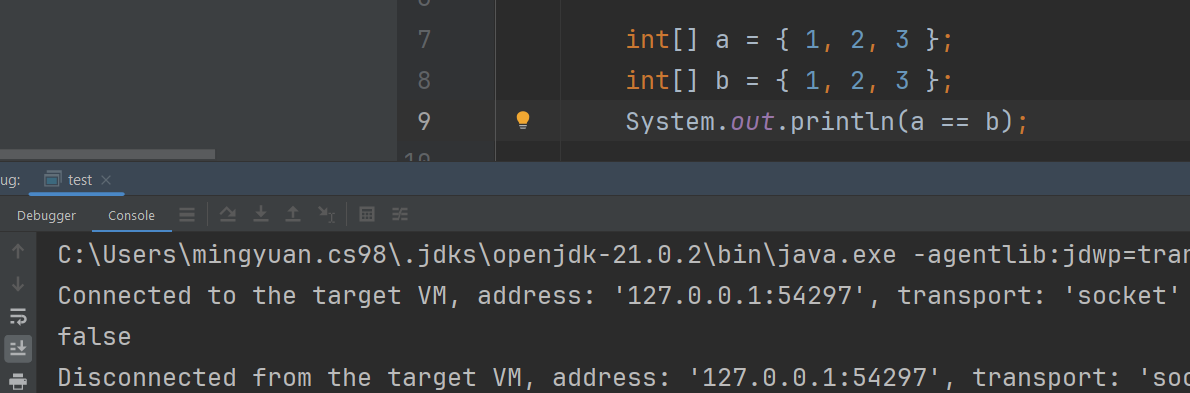
这个是选择排序的思想；

然后在每次找到最大值时有一个cnt 变量会++ 这个变量等于3时是我们返回arr[2] 这样可以找到第三个最大值；

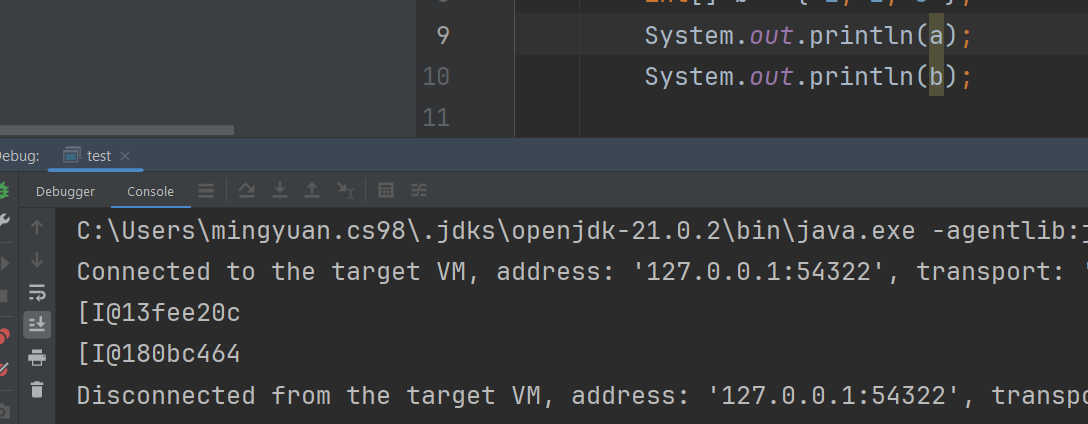
8



他们当然是不等于的虽然他们的value是一样的 因为arr 是包装类的每次创建哪怕他们的值一样但是他们会指向不同的memory地址：

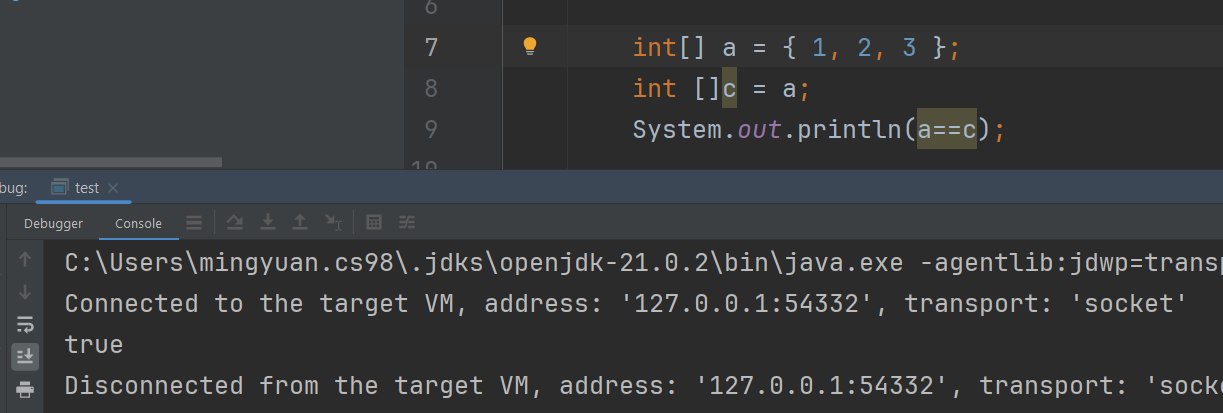


我们现在验证上面说的话：



我们会发现他们指向不一样的地址所以== 会比较他们的地址这样会输出不一样的；

现在我们再次验证指向同一个地址的是等价如下代码：



我们把c 也指向a 这样就会输出他们是等价的；

1. **8皇后问题. 我们可以用排列代表国际象棋棋盘上皇后的位置. 例如排列 （52431） 可以代表如下棋盘的布局**

\* \* \* Q \*

\* Q \* \* \*

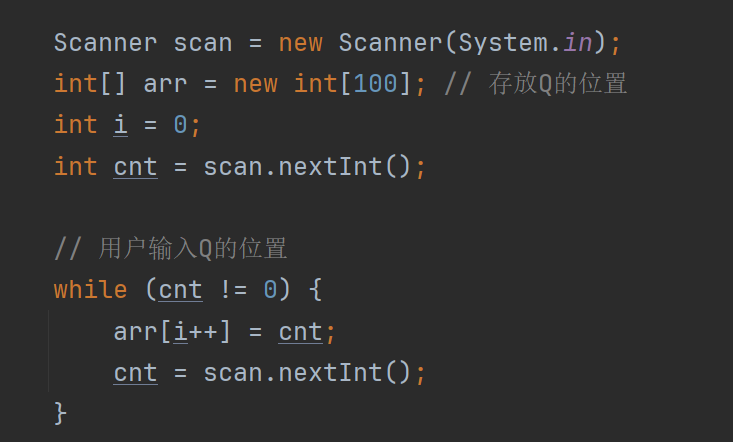
\* \* \* \* Q

\* \* Q \* \*

Q \* \* \* \*

解法：

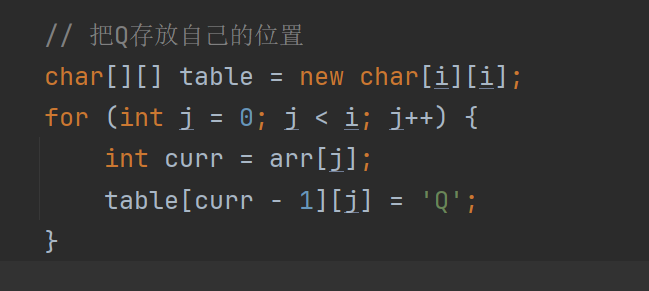
13.1 用一个数组来存储用户输入的皇后位置，如上面例子数组会存储 **5 2 4 3 1**



**13.2 我们用一个2为字符串数组来把每个皇后存放到自己位置。**

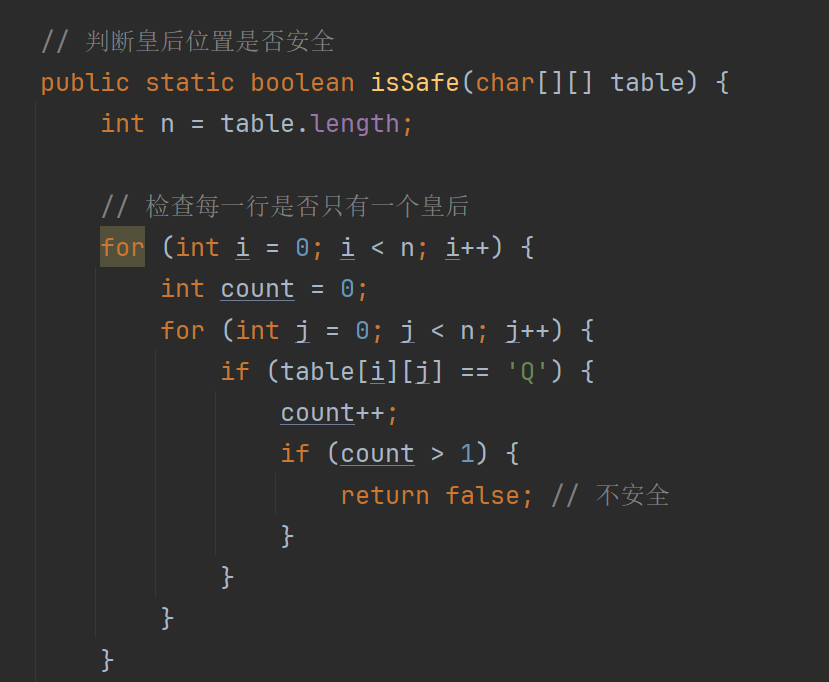
**皇后位置是数组每个元素代表行，而j代表相应的列**

**比如 5 代表皇后位置在5行 1列，因为5是第一个元素所以 1列，其他元素以此类推都存放到自己的位置**

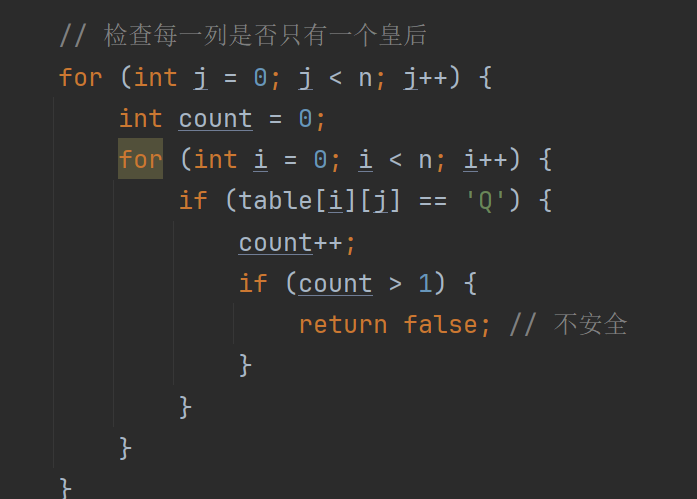


**13.3 写一个函数来判断皇后位置是否安全：**

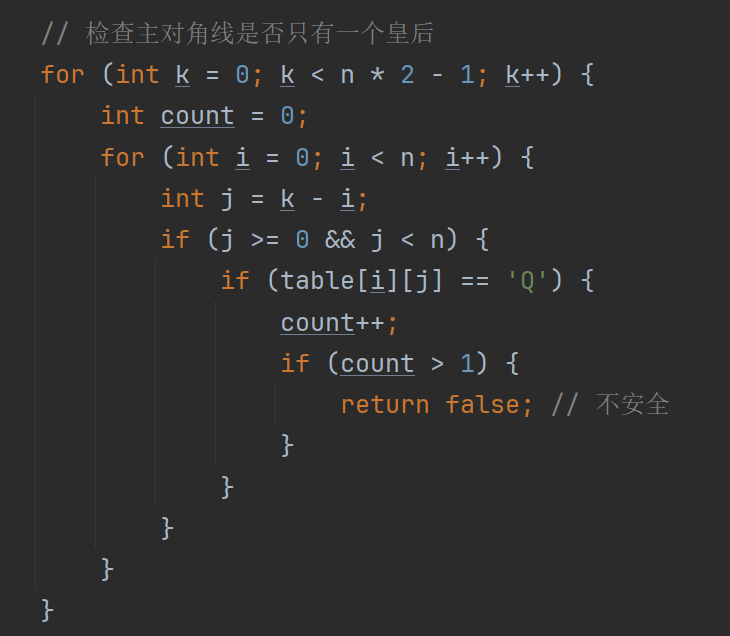
我分成4个情况来判断皇后位置是否准确，第一种是每行不能有两个皇后：



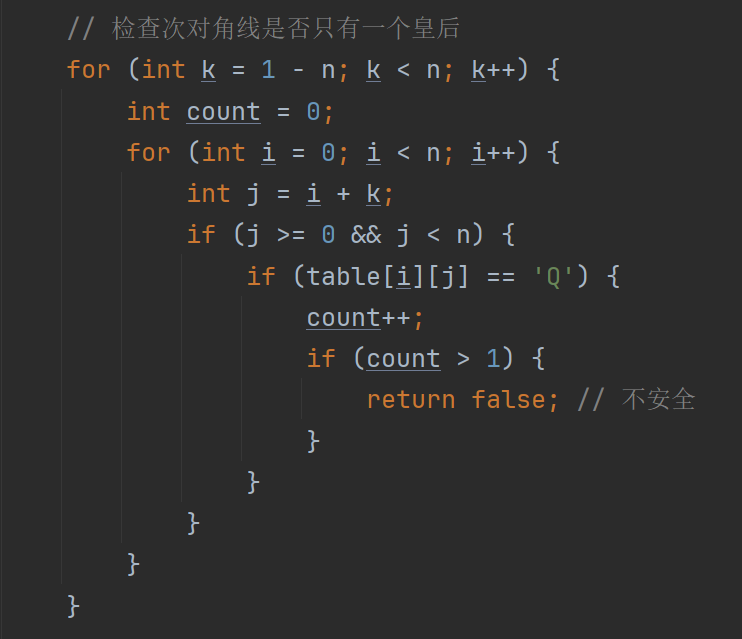
13.4 检查每一列是否只有一个皇后：



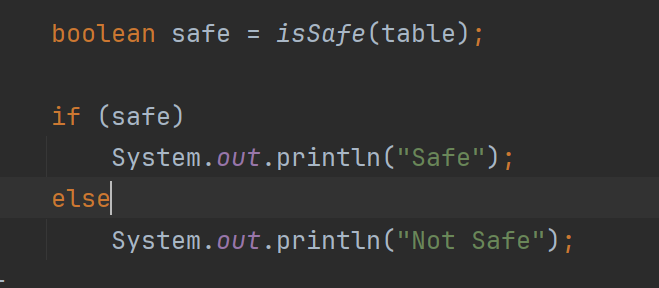
13.5 检查主对角线：



13.6 检查次对角线：

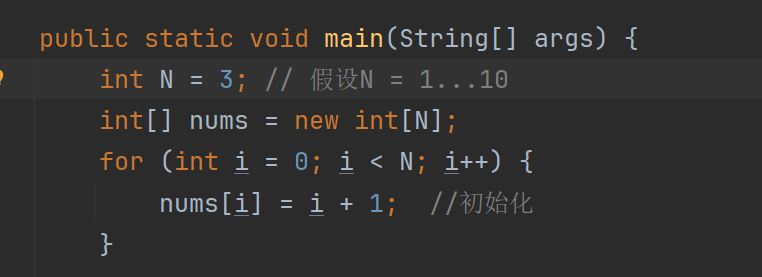


每一种情况分你别检查如果一种情况是假的会返回假，否子返回真

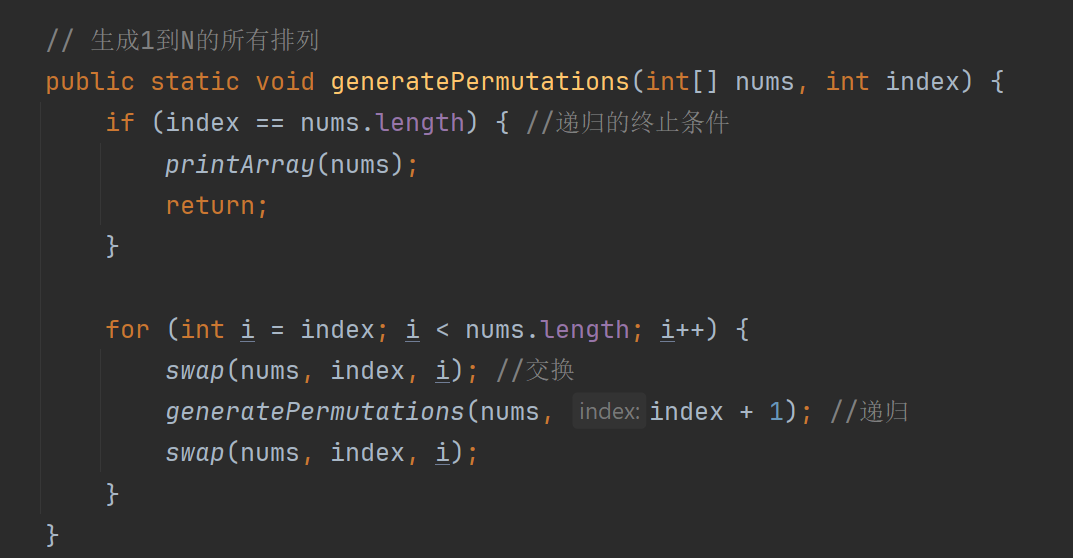


1. **给定整数 N , 输出 1 到 N 的所有排列. 利用本题测试第7题中寻找中位数的算法是否正确.**

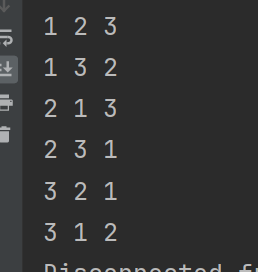
假设我们的数组只有1....3的数字，通过下面代码我们生命，且初始话数组：



* 1. 通过递归来实现1到N的排列



输出结果：



1. **给定N, 输出N阶蛇型矩阵 分别是s（1）.s（3）**

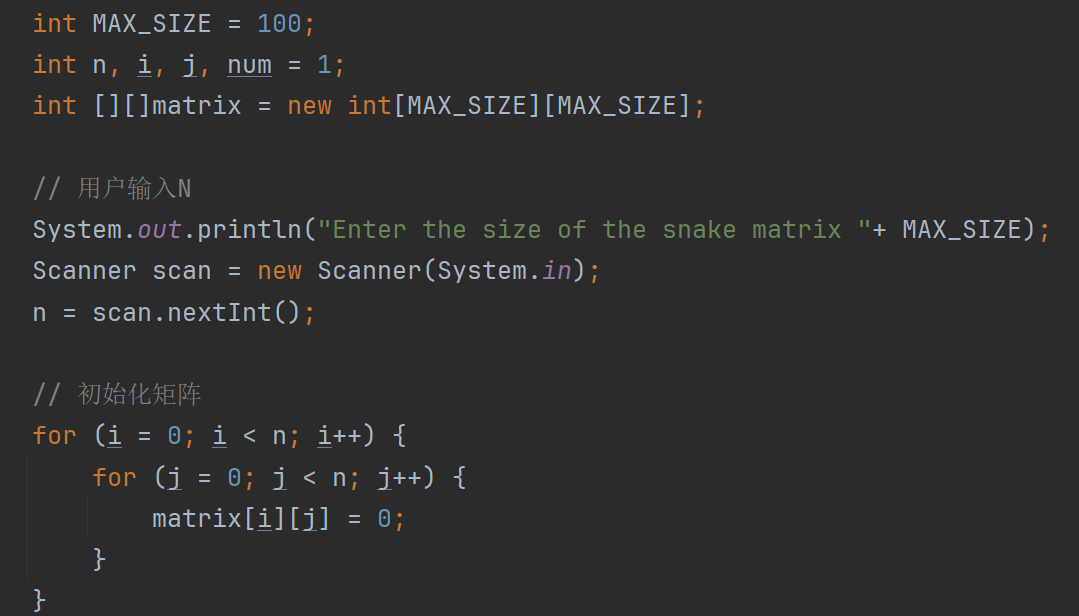
s（1） = [1] s(3) = 1 2 3

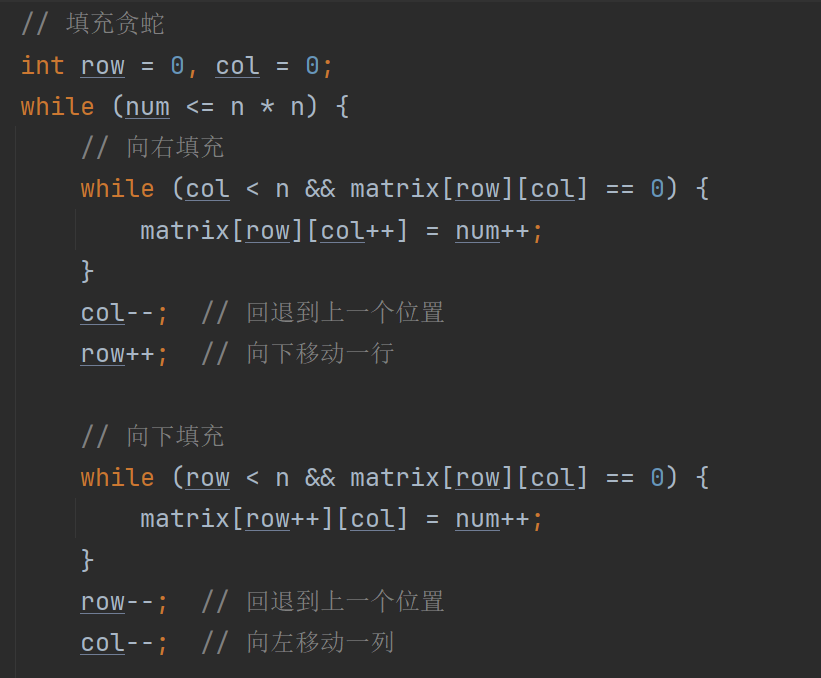
8 9 4

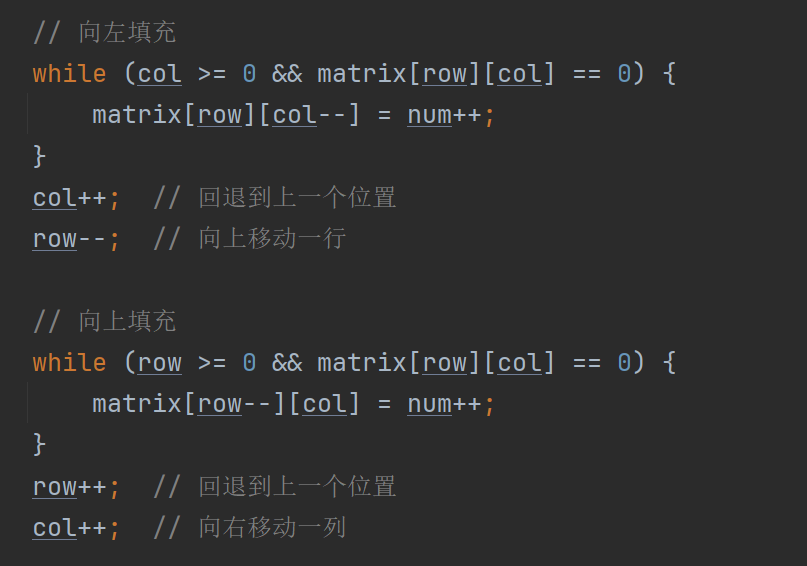
7 6 5

分析首先创建一个NxN的矩阵，并用0初始化。然后按照贪蛇的填充规则，从左上角开始填充数字，依次向右、向下、向左、向上填充，直到矩阵填满。最后打印出填充完成的贪蛇矩阵。

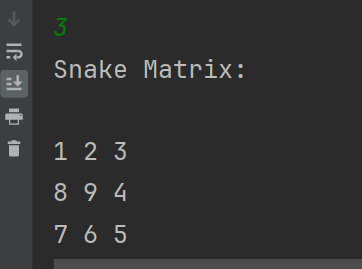
初始化：



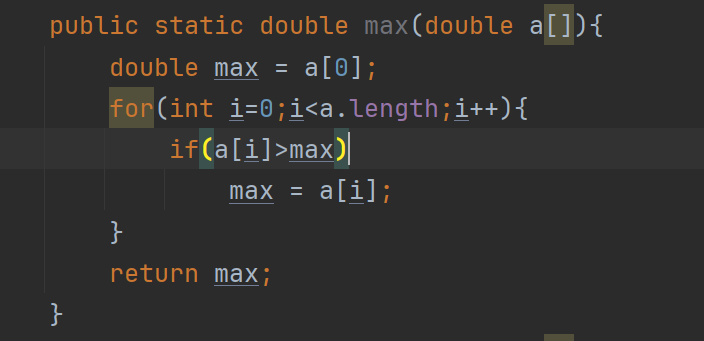




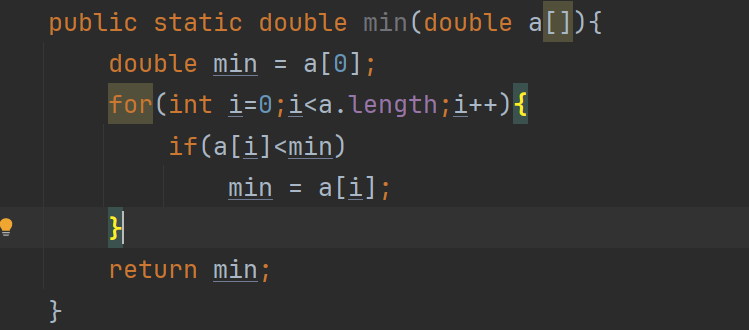
输出结果：



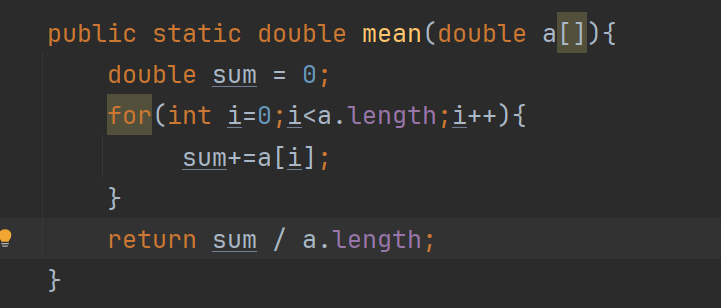
1. **编写库文件 Statistic.java, 包含如下静态方法:**
   1. **max 返回数组a中最大值；**



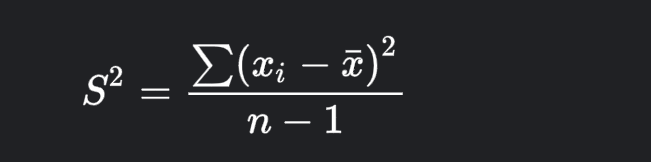
* 1. **min 返回数组a中最小值：**

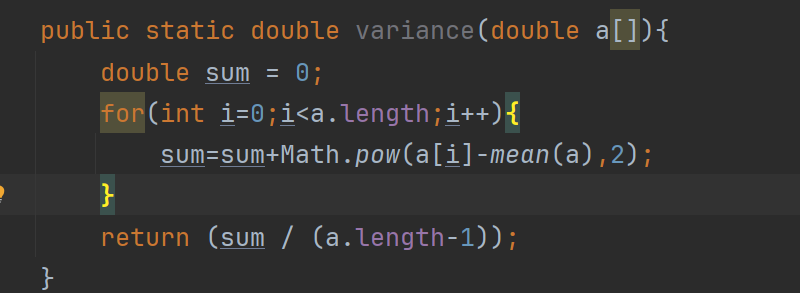


* 1. **返回a数组均值：**



* 1. 返回a数组方差公式如下；





* 1. **返回a数组中第k大解释在第7道题解说思路（即选择排序的思路）：**

