## Java 中的运算符和流程控制 + 面试题

### 算术运算符

Java 中的算法运算符，包括以下几种：

|  |  |  |
| --- | --- | --- |
| **算术运算符** | **名称** | **举例** |
| + | 加法 | 1+2=3 |
| - | 减法 | 2-1=1 |
| \* | 乘法 | 2\*3=6 |
| / | 除法 | 24/8=3 |
| % | 求余 | 24%7=3 |
| ++ | 自增1 | int i=1;i++ |
| -- | 自减1 | int i=1;i– |

我们本讲要重点讲的是 “++” 和 “–”，其他的算术运算符相对比较简单直观，本讲就不花精力去讲解了，之所以要把 “++” 和 “–” 单独拿出来讲，是因为在使用他们的时候有很多坑需要开发者注意，最重要的是 “++” 和 “–” 也是面试中高频出现的面试题。

先来看 “++” 的基本使用：

int i = 1;  
int i2 = ++i; // ++i 相当于 i = 1+i;  
System.out.println(i); // 2  
System.out.println(i2); // 2

**++i 和 i++ 的区别**

* ++i 先自加再赋值
* i++ 先赋值再自加

比如：

int i = 0;  
int i2 = i++;  
int j = 0;  
int j2 = ++j;  
System.out.println("i2=" + i2);  
System.out.println("j2=" + j2);

输出的结果：

i2=0  
j2=1

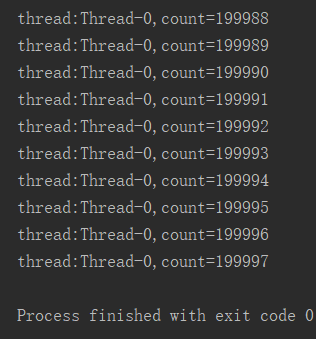
代码解析：i++ 是先给 i2 赋值再自身 +1 ，所以 i2 等于0，而 ++j 是先自加等于 1 之后，再赋值给 j2，所以 j2 等于 1。

**注意事项**

++/– 是非线程安全的，也就是说 ++/– 操作在多线程下可能会引发混乱，例如下面代码：

new Thread() {  
 @Override  
 public void run() {  
 for (int i = 0; i < 100000; i++) {  
 System.out.println("thread:" + this.getName() + ",count=" + (++count));  
 }  
 }  
}.start();  
new Thread() {  
 @Override  
 public void run() {  
 for (int i = 0; i < 100000; i++) {  
 System.out.println("thread:" + this.getName() + ",count=" + (++count));  
 }  
 }  
}.start();

执行的结果，如下图：



执行结果

如上图所示，每台机器的执行可能略有差距，但大多数情况下并不能给我们想要的真实值 200000。

**原理分析**

“++” 操作在多线程下引发混乱的原因：因为 ++ 操作对于底层操作系统来说，并不是一条 CPU 操作指令，而是三条 CPU 操作指令——取值、累加、存储，因此无法保证原子性，就会出现上面代码执行后的误差。

**如何避免 ++/– 操作在多线程下的“误差”？**

* 方法一：++/– 操作放在同步块 synchronized 中。
* 方法二：自己申明锁，把 ++/– 操作放入其中。
* 方法三：使用 AtomicInteger 类型替代 int 类型。

最后，因为 – 的语法和 ++ 完全一致，所以 – 的操作，请参照上面的 ++ 语法。

### 条件运算符（三元运算符）

条件运算符（?:）也叫“三元运算符”。

语法：

布尔表达式 ? 表达式1 ：表达式2

运算过程：如果布尔表达式的值为 true，则返回 **表达式 1 的值** ，否则返回 **表达式 2 的值** 。

例如：

String s = 3 > 1 ? "三大于一" : "三小于一";  
System.out.println(s);

执行结果：三大于一。

### 流程控制

在 Java 语言中使用条件语句和循环结构来实现流程控制。

#### 1 条件语句

条件语句的语法格式：

if(……) ……

其中的条件判断必须使用括号括起来不能省略。

基础用法使用：

int i = 1;  
if (i > 1) {  
 System.out.println("i大于一");  
} else if (i == 1) {  
 System.out.println("i等于一");  
} else {  
 System.out.println("其他");  
}

#### 2 循环

while 当条件成立的时候执行下一条语句。

while 语法格式：

while(……) ……

基本语法使用：

int i = 0;  
 while (i < 3) {  
 System.out.println(++i);  
}

while 是先判断再决定是否执行，有可能一次也不执行，如果希望至少执行一次，可以使用 do/while。

do/while 语法格式：

do{……}while(……);

基本语法使用：

int i = 0;  
do {  
 System.out.println(++i);  
} while (i < 3);

#### 3 确定循环

for 循环是程序中最长使用的循环之一，它是利用每次迭代之后更新计数器来控制循环的次数。

for 语法格式：

for(int i=0;i<n;i++){ …… }

基础语法使用：

for (int i = 0; i < 10; i++) {  
 System.out.println("i=" + i);  
}

for 循环中可使用关键字 continue，跳过后续操作，继续下一次迭代。

例如：

for (int i = 1; i < 4; i++) {  
 if (i == 2) continue;  
 System.out.println("i=" + i);  
}

执行结果：

i=1  
i=3

如结果所示，第二次循环就会跳过，执行下一次循环。

**for 注意事项**

在循环中检查两个浮点数是否相等要格外小心，例如下面代码：

public static void main(String[] args) {  
 for (float i = 0; i != 1; i += 0.1) {  
 System.out.println(i);  
 }  
}

循环永远不会停下来，由于舍入误差，因为 0.1 无法精确的用二级制表示，所以上面代码到 0.9000001 之后，会直接跳到 1.0000001，不会等于 1，所以循环就永远不会停下来。

#### 4 多重选择

switch 的特点是可以判断多个条件，if 的特点是执行少量判断，它们两个刚好形成互补的关系。

switch 语法格式：

switch(……){ case 1: …… break; …… default: …… break; }

switch 基础使用：

int i = 3;  
switch (i) {  
 case 1:  
 System.out.println("等于1");  
 break;  
 case 2:  
 System.out.println("等于2");  
 break;  
 case 3:  
 System.out.println("等于3");  
 break;  
 default:  
 System.out.println("等于其他");  
 break;  
}

可用于 case 的类型有：

* byte、char、short、int
* 枚举
* 字符串（Java SE 7 新加入）

**switch 注意事项**

switch 使用时，每个选项最末尾一定不要忘记加 break 关键字，否则会执行多个条件。

案例：

int i = 1;  
switch (i) {  
 case 1:  
 System.out.println("等于1");  
 case 2:  
 System.out.println("等于2");  
 case 3:  
 System.out.println("等于3");  
 default:  
 System.out.println("等于其他");  
}

程序执行的结果：

等于1  
等于2  
等于3  
等于其他

所以使用 switch 时，每个选项的末尾一定得加 break 关键字。

### 相关面试题

#### 1. Java 中 i++ 和 ++i 有什么区别？

答：i 先赋值再运算；i 先运算再赋值。

示例代码：

int i = 0;  
int i2 = i++;  
int j = 0;  
int j2 = ++j;  
System.out.println("i2=" + i2);  
System.out.println("j2=" + j2);

输出结果：i2=0，j2=1

#### 2. 以下代码 i 的值是多少？

int i = 0;  
i = i++;  
System.out.println(i);

答：i=0

题目解析：因为 Java 虚拟机在执行 i++ 时，把这个值有赋值给了 i，而 i++ 是先赋值再相加，所以这个时候 i 接收到的结果自然是 0 了。

#### 3. 以下代码 i2 和 i3 的值分别为多少？

int i = 0;  
int i2 = i++;  
int i3 = ++i;

答：i2=0，i3=2

#### 4. 以下代码能不能正常执行？

if (true) System.out.println("laowang");

答：可以正常执行，其中判断条件的括号不能省略，大括号是可以省略的（作者并不建议为了省代码的而牺牲代码的可读性）。

#### 5. 以下 switch 执行的结果是什么？

int num = 1;  
switch (num) {  
 case 0:  
 System.out.print("0");  
 case 1:  
 System.out.print("1");  
 case 2:  
 System.out.print("2");  
 case 3:  
 System.out.print("3");  
 default:  
 System.out.print("default");  
}

答：123default

#### 6. switch 能否用于 byte 类型的判断上？能否用于 long 类型的判断上？

答：switch 支持 byte 类型的判断，不支持 long 类型的判断。

题目解析：switch 支持的全部类型（JDK 8）：char、byte、short、int、Charachter、Byte、Short、Integer、String、enum。

#### 7. while 必须配合 break 一起使用的说法正确吗？

答：错误，while 可以单独使用。

例如：

int i = 0;  
while (i < 3) {  
 System.out.println(++i);  
}

#### 8. 以下代码可以正常运行吗？为什么？

int i = 0;  
while (i < 3) {  
 if (i == 2) {  
 return;  
 }  
 System.out.println(++i);  
}

答：可以正常运行，这里的 return 和 break 的效果是一致的，while 可以配合 return 或 break 一起使用。

#### 9. 以下的程序执行结果什么？

int i = 0;  
do {  
 System.out.println(++i);  
} while (i < 3)

答：编译器报错，do/while 之后必须使用分号 ; 结尾。

#### 10. 以下程序输出的结果是？

String s = new String("laowang");  
String s2 = new String("laowang");  
System.out.println(s == s2);  
switch (s) {  
 case "laowang":  
 System.out.println("laowang");  
 break;  
 default:  
 System.out.println("default");  
 break;  
}

A：true,default  
B：false,default  
C：false,laowang  
D：true,laowang

答：C

#### 11. 以下代码循环执行了几次？

for (float i = 0; i != 10; i += 0.1) {  
 System.out.println("hi");  
}

答：无数次，循环永远不会停下来。由于舍入误差，因为 0.1 无法精确的用二级制表示，所以上面代码到 0.9000001 之后，会直接跳到 1.0000001，不会等于 1，所以循环就永远不会停下来。

#### 12. 以下代码输出的结果是？

int num = -4;  
System.out.println(num % 2 == 1 || num % 2 == -1);

A：1  
B：-1  
C：true  
D：false

答：D

题目解析：-4 % 2 = 0 既不等于 1 也不等于 -1，所以结果为 false。

#### 13. 以下代码输出的结果是？

int num = 4;  
num = ((num & 1) == 1);  
System.out.println(num);

A：4  
B：1  
C：以上都不是

答：C

题目解析：== 运算返回的是 boolean 类型，不能使用 int 接收，所以程序会报错。

方便读者更有针对性地讨论专栏相关问题，以及分享Java 技术和面试心得，GitChat 编辑团队组织了一个《Java 面试全解析》读者交流群，添加编辑小姐姐微信：「GitChatty6」，回复关键字「234」给编辑小姐姐获取入群资格。

## 更多资源下载交流请加微信：Morstrong,加入永久会员,网盘更新更快捷！

# 本资源由微信公众号：光明顶一号，提供支持