

微光招新

Java03

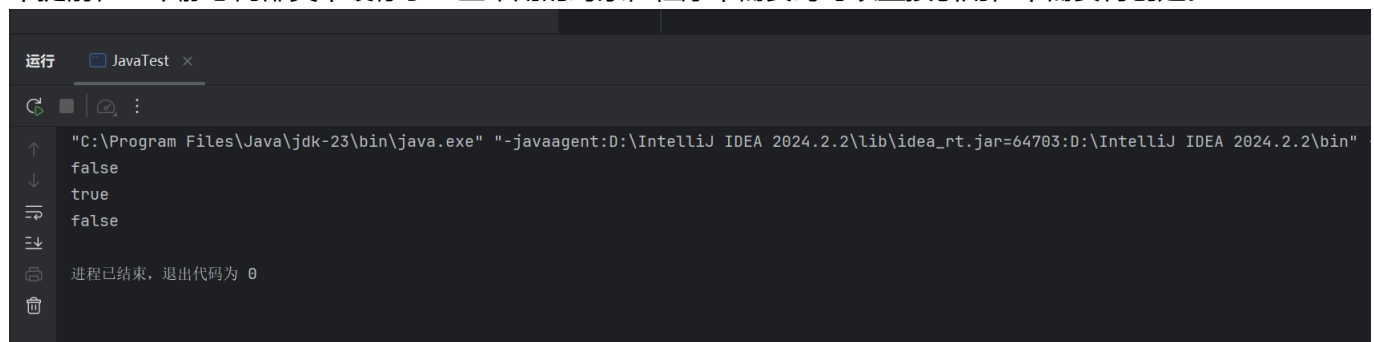
task1

- (1)整形: byet (字节), short (短整型), int (整数), long (长整型) (2)浮点型: float (单精度 (在数字尾加f否则默认double)), double (双精度) (3)字符型: char (4)布尔值: Boolean (true, false)
- (1) byet: 1字节 (8位), 取值范围为: $(-2^7) \sim (2^7-1)$ (原因: 有符号整数, 原码一共8位, 符号位有1位, 剩下7位即数字转化所得二进制数, 所以正数最大为 $1111111 = 2^7-1$, 但是负数之所以到 (-2^7) 而不是 $(-(2^7-1))$, 是因为0会分正负 (符号位), 用正0表示0, -128原码11000000多出一位存储到byet里时会截断最左边一位, 此时负0的补码与原码和-128的补码与原码一模一样, 因此可以用-0表示-128, 所以最小是-128) (下列整型取值范围同理) (2)short: 2字节 (16位), 取值范围: $-2^{15} \sim 2^{15}-1$ (3)int: 4字节 (32位), 取值范围: $-2^{31} \sim 2^{31}-1$ (4)long: 8字节 (64位), 取值范围: $-2^{63} \sim 2^{63}-1$

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

- 自动转化类型 (char字节长度小于int, 可以直接在左边加0成32位变成int类型); b为52, 因为c变量里面存的是字符0 (单引号), c自动转换为int型时是将字符0变成其对应ASCII表的值进行接下来的运算的, 0的ASCII值是48, 所以 $b=48+4=52$
-

包装类: 用来将基本数据类型包装成对象 (引用类型的数据), (装箱和拆箱) 引用类型: (1) 用于引用对象的一种变量类型, 我们创建对象时, 是在堆内存中分配了一块内存空间, 并返回了一个指向该内存空间的引用。 (2) 引用类型主要分4种: 强引用 (指向对象并可以阻止垃圾回收器对该对象的回收), 软引用 (稍弱于强引用, 在内存不足时可能其指向的对象会被垃圾回收器回收), 弱引用 (更弱, 内存不足优先回收), 虚引用 (最弱, 用来帮助对象在被回收时做一些处理); (3) 作用: 便于管理内存。基本数据类型缓存池: 包装类中提前在一个静态内部类中缓存了一些常用的对象, 程序中需要时可以直接引用, 不需要再创建。



```
运行 JavaTest x
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\IntelliJ IDEA 2024.2.2\lib\idea_rt.jar=64703:D:\IntelliJ IDEA 2024.2.2\bin"
false
true
false
进程已结束, 退出代码为 0
```

```

Microsoft Windows [版本 10.0.22621.4317]
(c) Microsoft Corporation. 保留所有权利。

D:\weiguangzxt>javac JavaTest.java
JavaTest.java:3: 警告: [removal] Integer 中的 Integer(int) 已过时, 且标记为待删除
    Integer x = new Integer(18);
                  ^
JavaTest.java:4: 警告: [removal] Integer 中的 Integer(int) 已过时, 且标记为待删除
    Integer y = new Integer(18);
                  ^
2 个警告

D:\weiguangzxt>java JavaTest
false
true
false

```

前提：关系运算符(==)会判断2个数是否相等（地址，数值），相等返回true，不相等返回false；原因：；第一个输出为false，因为`Integer x = new Integer(18);`表示将新建一个对象，2个新建对象的地址会不同，所以用(==)比较时会输出false；用第二个输出为true，第三个输出为false，因为其都使用了Integer中的valueOf方法去将该int值包装成对象，这个方法中缓存了经常使用的一些Integer数（从（-128）到（127））在这个范围内的数会直接引用静态数组cache里缓存的对象（满足地址相同），如果超过这个范围的数会新建一个对象，新建对象的地址不相同（若要比对值，需要用equals方法），第二个int数值为18，第三个int数值是300，所以使用(==)判断：第二个是true，第三个是false，

task2

```

class JavaAlu {
    public static void main(String[] args) {
        int a = 5;
        int b = 7;
        int c = (++a) + (b++); // ++a是先加后用，所以会先赋值a变成5+1=6，再将6用来运算
        // b++是先加后用，所以先将7使用，再+1变成8，所以c=6+7=13
        System.out.println(c);
        System.out.println(a); // 中间是空格隔开
    }
}

```

```

运行
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\IntelliJ IDEA 2024.2.2\lib\idea_rt.jar=65027:D:\IntelliJ IDEA 2024.2.2\bin" -Dfile.encoding=UTF-8
13
6 8
进程已结束，退出代码为 0

```

5.

(解释过程在注释)

6. (1) 答案：0010；计算机以补码进行计算， $a=0010$ ，说明是个正数，其补码就是其本身，而 $(-a)$ 的补码则是将其相反数的原码取反加一得到1110，&是位运算符，我们将 a 与 $(-a)$ 补码的位一一对应，进行（位于）运算，2数对应位若都为1则为1，否则为0，0010与1110只有第三位都是1，所以结果是0010即为2；（2）得到 a 的最低有效位，因为相当于在计算机中 $(-a)$ 的补码经过取反加一后会在 a 最低的1的位置同样是1，再向左就都是原码的反码，向右一定是0或者没有，所以在经历了&操作后，只有 a 的最低有效位处的会是1，相当于求 a 的最低有效位。

