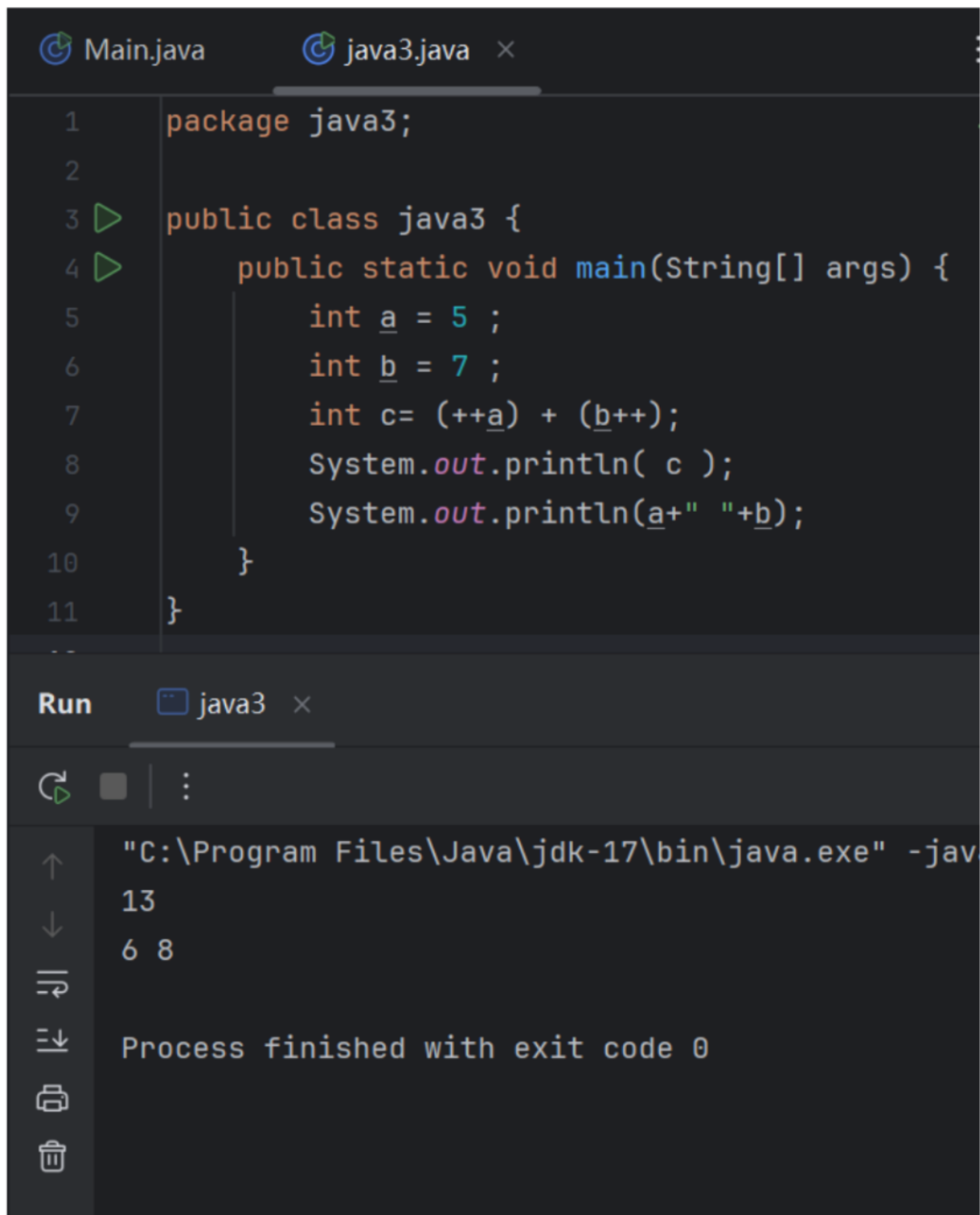


Task1

- (1) 整形有 byte,short,int,long
字符型有 char
浮点型有 float,double
布尔型有 boolean
- (2) byte 占用 1 个字节, 表示-128~127
short 占用 2 个字节, 表示 $-32768 \sim 32767[-2^{15} \sim (2^{15}-1)]$
int 占用 4 个字节, 表示 $-2^{31} \sim (2^{31}-1)$
long 占用 8 个字节, 表示 $-2^{63} \sim (2^{63}-1)$
- (3)涉及的是自动类型转换, b 值为 52;
在表达式中 char 类型是直接转换成 int 类型参与运算, 因此运行到 int b = a + c;时实际上是 a = 4 与 c 在 ASCII 码表中对应的 48 进行 int 类型与 int 类型的加法运算,由此计算出 b 的值为 52, 且为 int 类型。
- (4)第一部分输出结果为 false。第一部分使用 new Integer()创建对象, 虽然其在堆内存中存储的数据都是 18, 但是由于是分别创建的两个对象, 其堆内存的位置不同, 则内存地址不同; 对于引用类型的变量, ==比较的是内存地址; 二者内存地址不同, 则输出 false。
第二部分输出结果为 true。第二部分使用 Integer.valueOf(), 由于此时 Java 会对在-128 到 127 这一小范围的整数进行缓存, 使得 z 和 k 指向了缓存中的同一个对象。则 z 和 k 保存了指向由缓存机制提供的特定整数值 (18) 对应的 Integer 对象在内存中的地址。二者内存地址相同, 则输出 true。
第三部分输出结果为 false。由于数值 300 超过了缓存允许的整数范围, 因此每次调用 Integer.valueOf(300)都会创建一个新的 Integer 对象。即 m 和 p 分别指向两个不同的对象, 其内存地址不同, 所以输出结果为 false。

Task2

- (5) ++a 相当于 a = a+1, 此时 a 直接进行自增运算, 结果为 6; b++则是先使用当前值进行运算, 该行代码运算结束之后再进行自增运算, b 值变为 8; 因此对于 int c = (++a) + (++b) 实际上是 int c = 6+7, 则 c 输出值为 13, 且运算后 b 再进行自增运算, 值为 8; a 的值为 6



The screenshot shows an IDE with two tabs: 'Main.java' and 'java3.java'. The 'java3.java' tab is active, displaying the following code:

```
1 package java3;
2
3 public class java3 {
4     public static void main(String[] args) {
5         int a = 5;
6         int b = 7;
7         int c = (++a) + (b++);
8         System.out.println(c);
9         System.out.println(a + " " + b);
10    }
11 }
```

Below the code editor, there is a 'Run' button and a terminal window titled 'java3'. The terminal shows the execution command and output:

```
"C:\Program Files\Java\jdk-17\bin\java.exe" -jav
13
6 8
Process finished with exit code 0
```

(6) $a \& (-a)$ 的二进制形式是 0010。

表示的数是 a 的二进制表示中最右边的 1 及其右所有 0 组成的数字

证明如下：

我们假设 a 共有 n 位，其中第 i 位为 a 最右侧的 1

第 n 位为 1 时，此时 a 取反后第 n 位为 0，再加 1 得到 $-a$ 且不进位，发现 a 和 $-a$ 第 n 位均为 1，第 1 到 $n-1$ 位互为反码，根据位与运算可知结论正确；

第 n 位为 0 时， a 取反后 i 位为 0， $i+1$ 位到 n 位均为 1，再加 1 且进位到 i 位结束进位，得到 $-a$ ，此时 $-a$ 的 1 到 $i-1$ 位互为反码， i 位是 1， $i+1$ 位到第 n 位均为 0，且 a 的 i 位也为

1, 由位与运算可知结论正确。

综上, 对于任意的非负整数 a , 式子 $a \& (-a)$ 表示的数是 a 的二进制表示中最右边的 1 及其右所有 0 组成的数字。