

Task1

一、八种数据类型

整型: `byte` (位), `short` (短整数), `long` (长整数), `int` (整数)

字符型: `char` (字符)

浮点型: `float` (单精度浮点数), `double` (双精度浮点数)

布尔型: `boolean` (布尔值)

二、四种整型字节数和范围

1.byte:

顾名思义就是 1 字节

取值范围: -128 到 127

2.short:

2 字节

取值范围: -32,768 到 32,767

3.int:

4 字节

取值范围: -2^{31} 到 $2^{31}-1$

4.long:

8 字节

取值范围: -2^{63} 到 $2^{63}-1$

三、类型转换

题中所给代码为自动类型转换 (`char`—`int`)，不过由于 `c` 本来为字符型，所以会根据 ASCII 表的对应关系被赋为 48，所以最终 `b` 的结果为 52

四、输出结果及解释

注：在我的 IDEA 中显示 `new Integer` 已被弃用，但居然能正常运行（QAQ！）

代码从上至下结果分别为

`false`

`true`

`false`

下给出解释

1.由于使用了 `new` 创建了两个值都为 18 的对象 `x` 和 `y`，但是 `new` 决定了他们本质上是两个不同的对象（即同时创造出两个新对象）而在使用“`==`”进行比较时，实际上是比较 `x` 和 `y` 的内存地址，两个不同对象的地址不同所以结果为 `false`

补充一下我在一次运行中使用 `System.out.println("HashCode of x/y: " + System.identityHashCode(x/y))` 显示出的作为对象独一无二标识的哈希码

HashCode of x: 284720968

HashCode of y: 1534030866

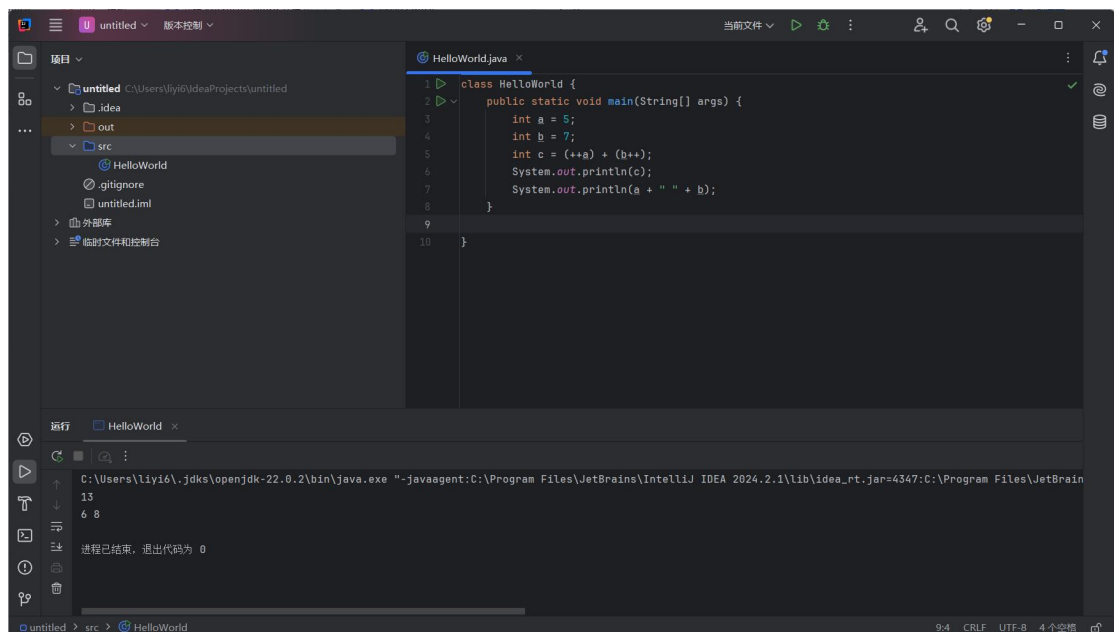
再次证明他们不是一个对象

2&3.

`Integer.valueOf()`可以将基本类型 `int` 转换为包装类型 `Integer`，在 `int` 在`[-128,127]`之间的时候会直接拿缓存（查资料据说是 `java` 为节省内存所致），而不会 `new`，所以第二段代码中 `z` 和 `k` 的地址一样所以为 `true`。而在第三段代码中它的区间不在`[-128,127]`内，故效果同第一段代码（会 `new`），为 `false`

Task2

五、运行结果



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

运行结果：

```
C:\Users\liy16\.jdk\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.1\lib\idea_rt.jar=4347:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.1\bin" -Dfile.encoding=UTF-8
13
6 8
进程已结束，退出代码为 0
```

解释：

前两行代码初始化了 `a` 为 5，`b` 为 7。而前缀 `++a` 表示 `a` 先自增 1 变为 6，再参与计算，后缀 `b++` 先用 `b=7` 参加计算，后自增 1 变为 8，故结果为 13，6，8

六、关于补码

$a \& (-a)$ 的二进制形式为 0010

正整数和 0 的补码是它本身，负整数的补码是它的原码取反再加一按位与运算（&）后，同 1 为 1，不同取 0，所以 $a \& (-a)$ 表示的数就是 a 的最低位的 1 和所有更低的位