JVM中一个字节以下的整型数据会在JVM启动的时候加载进内存，除非用new Integer()显式的创建对象，否则都是同一个对象

所有只有i04是一个新对象，其他都是同一个对象

Integer i01=59 的时候，会调用 Integer 的 valueOf 方法，

|  |  |
| --- | --- |
| 1  2  3  4  5 | public static Integer valueOf(int i) {       assert IntegerCache.high>= 127;       if (i >= IntegerCache.low&& i <= IntegerCache.high)       return IntegerCache.cache[i+ (-IntegerCache.low)];       return new Integer(i); } |

这个方法就是返回一个 Integer 对象，只是在返回之前，看作了一个判断，判断当前 i 的值是否在 [-128,127] 区别，且 IntegerCache 中是否存在此对象，如果存在，则直接返回引用，否则，创建一个新的对象。

在这里的话，因为程序初次运行，没有 59 ，所以，直接创建了一个新的对象。

int i02=59 ，这是一个基本类型，存储在栈中。

Integer i03 =Integer.*valueOf*(59); 因为 IntegerCache 中已经存在此对象，所以，直接返回引用。

Integer i04 = **new**Integer(59) ；直接创建一个新的对象。

System. *out*.println(i01== i02); i01 是 Integer 对象， i02 是 int ，这里比较的不是地址，而是值。 Integer 会自动拆箱成 int ，然后进行值的比较。所以，为真。

System. *out*.println(i01== i03); 因为 i03 返回的是 i01 的引用，所以，为真。

System. *out*.println(i03==i04); 因为 i04 是重新创建的对象，所以 i03,i04 是指向不同的对象，因此比较结果为假。

System. *out*.println(i02== i04); 因为 i02 是基本类型，所以此时 i04 会自动拆箱，进行值比较，所以，结果为真。