

53从零开始学Java53之Integer底层原理探究

前言

配套开源项目资料

一. Integer底层原理探究

1. int和Integer的区别
2. 被final修饰的Integer类
3. IntegerCache缓冲区
4. 几个核心实验
 - 4.1 比较new出的两个Integer对象
 - 4.2 Integer对象和int变量进行比较
 - 4.3 非new的Integer变量和new出的Integer变量进行比较
 - 4.4 两个非new生成的Integer对象进行比较
 - 4.5 ==和equals的区别

5. 结论

二. 结语

三. 今日作业

作者：孙玉昌，昵称【**一一哥**】，另外【**壹壹哥**】也是我哦

千锋教育高级教研员、CSDN博客专家、万粉博主、阿里云专家博主、掘金优质作者

前言


在之前的两篇文章中，壹哥给大家介绍了Java中的包装类及其特点、用法，但是这些内容主要是停留在”怎么用“的层面，没有太多涉及”为什么“，所以接下来壹哥会给大家讲一讲Integer这个包装类的底层原理。在现在的就业环境下，我们需要知其然，还要知其所以然，才能更好地满足就业需求。

-----前戏已做完，精彩即开始-----

全文大约【3200】字，不说废话，只讲可以让你学到技术、明白原理的纯干货！本文带有丰富的案例及配图视频，让你更好地理解 and 运用文中的技术概念，并可以给你带来具有足够启迪的思考.....

配套开源项目资料

Github:



GitHub – SunLtd/LearnJava

Contribute to SunLtd/LearnJava development by creating an account on GitHub.

GitHub

Gitee:



一一哥/从零开始学Java

从零开始学Java系列 稀土掘金专栏地址: <https://juejin.cn/column/7175082165548351546> CSDN专...

Gitee

一. Integer底层原理探究

1. int和Integer的区别

在前面的内容介绍中，壹哥给大家讲过Integer这个类，现在大家对它的用法应该都比较清楚了。但是除了要掌握Integer的用法之外，我们还要了解它的一些底层内容，因为在面试时，关于Integer的底层考察的比较多。比如一个常见的面试题是这样的：**请问int和Integer的区别有哪些？**

面对这样的一道题目，你该怎么回答？常规的答案其实很容易答出来，比如：

- int是基本数据类型，代表整型数据，默认值是0；
- Integer是 int的包装类，属于引用类型，默认值为null；
- int 和 Integer 都可以表示某一个整型数值；
- Integer变量实际是对象的引用，当new一个Integer时，实际上是生成一个指针指向此对象；而int则是直接存储数据值；
- Integer可以区分出未赋值和值为 0 的区别，而int 则无法表达出未赋值的情况；

- `int` 和 `Integer` 不能够互用，因为他们是两种不同的数据类型；
- `int`在初始化时，可以直接写成 `int=1` 的形式；
- 因为`Integer`是包装类型，使用时可以采用 `Integer i = new Integer(1)` 的形式，但因为Java中的自动装箱和拆箱机制，使得对`Integer`类的赋值也可以使用 `Integer i=1` 的形式；
- 如果我们只是进行一些加减乘除的运算 或者 作为参数进行传递，那么就可以直接使用`int`这样的基本数据类型；但如果想按照对象来进行操作处理，那么就要使用`Integer`来声明一个对象。

但是如果你只能回答出这样的答案，你在面试官的眼里只能算合格，还算不上优秀，我们需要对`Integer`了解地更多一些。

2. 被final修饰的Integer类

为了搞清楚`Integer`的底层，我们就不得不研究一下它的源码，我们来追踪一下`Integer`源码，如下图所示：

```
@jdk.internal.ValueBased
public final class Integer extends Number
    implements Comparable<Integer>, Constable, ConstantDesc {
```

从源码中可以看出，`Integer`是`Number`的一个子类，且被`final`所修饰！请大家回顾一些壹哥之前讲过的`final`知识点。我们知道，被`final`修饰的类是常量类，该类不能被继承，里面的方法不能被重写，创建出的对象也不能被修改！总之，`Integer`符合`final`类的特征。

3. IntegerCache缓冲区

我们还记得，在`Integer`中有一个`valueOf()`方法，该方法可以将`int`值转为`Integer`对象。接下来我们来看看该方法的实现源码，如下图所示：

```
@IntrinsicCandidate
public static Integer valueOf(int i) {
    if (i >= IntegerCache.Low && i <= IntegerCache.high)
        return IntegerCache.cache[i + (-IntegerCache.Low)];
    return new Integer(i);
}
```

图中有红色标注：-128 和 127，分别指向 `IntegerCache.Low` 和 `IntegerCache.high`。还有一个红色箭头指向 `IntegerCache.cache`。

从上图的源码截图中我们可以看到，Integer中有一个缓冲区叫做IntegerCache，这是Integer中的一个内部类，如下图所示：

```
private static class IntegerCache {
    static final int low = -128;
    static final int high;
    static final Integer[] cache;
    static Integer[] archivedCache;

    static {
        // high value may be configured by property
        int h = 127;
        String integerCacheHighPropValue =
            VM.getSavedProperty("java.lang.Integer.IntegerCache.high");
        if (integerCacheHighPropValue != null) {
            try {
                h = Math.max(parseInt(integerCacheHighPropValue), 127);
                // Maximum array size is Integer.MAX_VALUE
                h = Math.min(h, Integer.MAX_VALUE - (-low) - 1);
            } catch (NumberFormatException nfe) {
                // If the property cannot be parsed into an int, ignore
            }
        }
        high = h;
    }
}
```

千锋教育-孙玉昌

我们可以看到，low就是-128，high等于127，这是缓冲区的最低和最高边界。那么这个缓冲区的存在到底有什么用呢？大家别着急，我们先做几个核心实验。

4. 几个核心实验

为了能够讲清楚Integer的底层逻辑，壹哥给大家设计了如下代码，用于验证Integer的底层设计。

4.1 比较new出的两个Integer对象

我们通过new对象的方式，来创建两个Integer对象i和j，并比较这两个对象。

```
1 //通过new生成的两个Integer变量进行比较，结果为false
2 Integer i = new Integer(100);
3 Integer j = new Integer(100);
4 System.out.print(i == j); //false
```

从运行的结果中可以看出，**通过new生成的两个Integer对象永远是不会相等的**。这是因为new生成的是两个对象，Integer变量实际上是对Integer对象的引用，这两个对象的内存地址是不同的。

4.2 Integer对象和int变量进行比较

接下来我们在把一个Integer对象和int变量进行比较，如下：

```
1 Integer i = new Integer(100);
2 int j = 100;
3 System.out.print(i == j); //true
```

Integer变量和int变量进行比较时，只要两个变量的值是相等的，结果就为true。这是因为Integer包装类和int基本类型进行比较时，Java会进行自动拆箱操作，将Integer转为了int，然后再进行比较，实际上就变为了两个int变量的比较。本案例中两者的值都是100，所以用“==”等号进行比较时自然就是相等的。

4.3 非new的Integer变量和new出的Integer变量进行比较

然后我们再把一个非new的Integer变量和new出的Integer变量进行比较，如下所示：

```
1 //非new生成的Integer变量和new Integer()生成的变量进行比较
2 Integer i = new Integer(100);
3 //自动装箱
4 Integer j = 100;
5 System.out.print(i == j); //false
```

在这段代码中，非new生成的Integer变量和new Integer()生成的变量进行比较时，结果却为false！这是因为非new生成Integer变量时，内部会调用valueOf()方法，进行自动装箱操作，此时会把Integer变量的值指向Java常量池中的数据。而new Integer()生成的变量，则指向的是堆中新建的对象，两者在内存中的地址是不同的。

4.4 两个非new生成的Integer对象进行比较

接着我们再对两个非new生成的Integer对象进行比较，如下所示：

```

1  //两个非new生成的Integer对象进行比较
2  //i与j的取值范围是在 -128~127 之间!
3  Integer i = 100;
4  Integer j = 100;
5  System.out.print(i == j); //true
6
7  //x与y的取值范围不在 -128~127 之间!
8  Integer x = 200;
9  Integer y = 200;
10 System.out.print(x == y); //false

```

这段代码中，两个非new生成的Integer对象进行比较时，**如果两个变量的取值在 -128到127 之间，则比较结果为true；如果两个变量的值不在此区间，则比较结果为false。**这又是为什么呢？其实要想弄明白这个原因，我们只需要看看Integer类的valueOf()方法是怎么写的就可以了。

valueOf()方法源码如下：

```

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

```

我们知道，valueOf(int i)方法可以**将int值自动装箱变成对应的Integer实例**。并且从这段源码中我们可以看到其内部有一个if判断，根据判断结果的不同，**会有2种不同的方式得到Integer对象：当arg大于等于-128且小于等于127时，则直接从缓存中返回一个已经存在的对象；如果参数的值不在这个范围内，则new一个Integer对象返回，即要么new Integer，要么从int常量池中获取！**

之前我们构建Integer对象的传统方式是直接 new 一个Integer对象，内部会调用构造器。但是根据实践，我们发现大部分的数据操作都是集中在有限的、较小的数值范围内。因而在JDK 1.5中，新增了一个静态工厂方法valueOf(int i)。当我们进行Integer i=xxx 赋值操作时，Java内部会调用执行这个valueOf()实现自动装箱。而在调用valueOf()方法时，其内部会利用缓存机制，对取值在-128~127之间的int值进行缓存操作，这是在JDK 1.5 之后进行的一个可以明显改善性能的提升。而按照Javadoc文档，该缓存机制默认会缓存在 -128 到 127 之间的值，不在该区间的值并不会进行缓存。所以，给 Integer i 赋值的大小不同，比较的结果也可能会不同。

4.5 ==和equals的区别

最后我们再做一个实验，来看看==与equals比较两个Integer对象时有什么不同。

Java | 复制代码

```
1 Integer x = 127;
2 Integer y = 127;
3
4 Integer m = 100000;
5 Integer n = 100000;
6
7 System.out.println("x == y: " + (x==y)); // true
8 System.out.println("m == n: " + (m==n)); // false
9
10 System.out.println("x.equals(y): " + x.equals(y)); // true
11 System.out.println("m.equals(n): " + m.equals(n)); // true
```

从该实验中可以看出，==比较时，较小的两个相同的Integer会返回true，较大的两个相同的Integer会返回false。结合上面壹哥给大家的讲解，你思考一下这是为什么？

5. 结论

通过以上的几个核心实验，壹哥可以给大家梳理出一个结论：

当我们利用“==”等号比较两个 `Integer i` 和 `Integer j` 的值时，如果取值范围是在-128~127之间，两个相同的Integer值会返回true；如果不在该区间，两个相同的Integer值会返回false。这是因为Integer是final类，编译器把 `Integer i = 100;` 自动变为 `Integer i = Integer.valueOf(100);`。为了节省内存，`Integer.valueOf()` 对于较小的数，始终会返回相同的实例对象，因此，==比较的结果就是true。

那么如果我们只是为了比较两个Integer对象的值是否相等，而不是为了比较两个对象的地址是否相同，在开发时请尽量使用equals()方法，而不是==！

并且我们现在还知道，在Java中有3种方式可以构造出一个Integer对象，代码如下：

```
1 //方法1:
2 Integer i = new Integer(100);
3
4 //方法2:
5 Integer i = Integer.valueOf(100);
6
7 //方法3:
8 Integer i = 100;
```

实际上，方法2和方法3的本质是一样的，所以开发时为了简洁，我们一般是通过方法3来得到一个Integer对象。**但是尽量不要使用方法1来构建Integer对象，这是因为方法1总是会创建一个新的Integer实例，而方法2和方法3则会尽可能地返回缓存的实例对象，以节省内存。**

所以最终关于“int和Integer的区别有哪些”这道面试题的答案，如果你想拿到高分，就需要把Integer的底层原理也回答出来才行！如果你可以把以上内容都回答清楚，我相信单凭这一道题目，就足以让面试官对你刮目相看！

壹哥在自己的面试题精讲专栏中，对此题目有着非常细致地讲解，这里不再赘述，大家可以参考如下链接：

[高薪程序员&面试题精讲系列04之说说int与Integer的区别及底层原理](#)

-----正片已结束，来根事后烟-----

二. 结语

这样 壹哥 就给大家分析了“int和Integer的区别有哪些”这个面试题，猛一看很简单，实际涉及的内容很多！**最后我再梳理一下该问题的回答要点：**

1. 先简单回顾Java中的数据类型及取值范围；
2. 然后简介基本类型与包装类，最后还能说明为什么需要有包装类；
3. 接着说一下int与Integer的基本区别；
4. 最后再说int与Integer的深入区别，即底层的源码和原理。

如果你可以把我总结的这4点都能回答好，就这一个问题，面试官就会对你留下深刻的影响，他就会认为你的基础知识足够扎实，因为大多数人只会回答int和Integer的基本区别，很少有人去回答

底层的内容！而通过这个问题，面试官也会了解到，你对Java的内存分配是很熟悉的！

另外如果你独自学习觉得有很多困难，可以加入壹哥的学习互助群，大家一起交流学习。

三. 今日作业

请回答”int和Integer有哪些区别？“，评论区给出你的答案。