

# 彻底理解 为什么重写equals()方法为什么要重写hashCode()方法



TUCJVCB [关注](#)

0.127 2019.07.07 14:13:20 字数 724 阅读 935

重写equals()方法为什么要重写hashCode()方法，是面试中一个经常会出现的一个问题。看完这篇文章，你一定对这个问题有更深入的理解。

equals方法和hashCode方法都是Object类中的方法，我们来看看他们的源码：

```
1 | public boolean equals(Object obj) {
2 |     return (this == obj);
3 | }
```

```
1 | public native int hashCode();
```

可知，equals方法在其内部是调用了"=="，所以说不重写equals方法的情况下，equals方法是比较两个对象是否具有相同的引用，即是否指向了同一个内存地址。

而hashCode是一个本地方法，他返回的是这个对象的内存地址。

知道了这些之后我们接着往下看。

hashCode的通用规定：

- 在应用程序的执行期间，只要对象的equals方法的比较操作所用到的信息没有被修改，那么对同一个对象的多次调用，hashCode方法都必须始终返回同一个值。在一个应用程序与另一个应用程序的执行过程中，执行hashCode方法所返回的值可以不一致。
- 如果两个对象根据equals(Object)方法比较是相等的，那么调用这两个对象中的hashCode方法都必须产生同样的整数结果
- 如果两个对象根据equals(Object)方法比较是不相等的，那么调用这两个对象中的hashCode方法，则不一定要求hashCode方法必须产生不同的结果。但是程序员应该知道，给不相等的对象产生截然不同的整数结果，有可能提高散列表的性能。

由上面三条规定可知，如果重写了equals方法而没有重写hashCode方法的话，就违反了第二条规定。**相等的对象必须拥有相等的hash code。**

接下来，我用一个程序来演示一下不重写hashCode方法所带来的严重后果：

```
1 | public class Test {
2 |
3 |     public static void main(String[] args) {
4 |         Person person1 = new Person("TUCJVCB");
5 |         Person person2 = new Person("TUCJVCB");
6 |
7 |
8 |         Map<Person, Integer> hashMap = new HashMap<>();
9 |         hashMap.put(person1, 1);
10 |
11 |
12 |         System.out.println(person1.equals(person2));
13 |         System.out.println(hashMap.containsKey(person2));
14 |     }
15 |
16 |     static class Person {
17 |         private String name;
18 |
19 |         public Person(String name) {
20 |             this.name = name;
21 |         }
22 |
23 |         @Override
24 |         public boolean equals(Object obj) {
25 |             if (obj instanceof Person) {
26 |                 Person person = (Person) obj;
27 |
28 |                 return name.equals(person.name);
29 |             }
30 |             return false;
31 |         }
32 |     }
33 | }
```

以下是输入结果

```
1 | true
```

```
2 | false
```

对于第一个输出true我们很容易知道，因为我们重写了equals方法，只要两个对象的名字属性相同就会返回ture。但是为什么第二个为什么输出的是false呢？就是因为我们没有重写hashCode方法。所以我们得到一个结论：**如果一个类重写了equals方法但是没有重写hashCode方法，那么该类无法结合所有基于散列的集合（HashMap，HashSet）一起正常运作。**

那么我们如何解决这个问题，很简单，重写hashCode方法就行了。

```
1 | @Override
2 | public int hashCode() {
3 |     return name.hashCode();
4 | }
```

经过修改后，输入如下：

```
1 | true
2 | true
```

**以上就是重写equals方法后必须要重写hashCode方法的原因了。**

 1人点赞 >



 Java基础知识



“小礼物走一走，来简书关注我”


赞赏支持

还没有人赞赏，支持一下




TUCJVXCB ACM-ICPC退役选手，现学习后端开发，水平有限，欢迎交流~~  
总资产0.132 (约0.01元) 共写了5773字 获得1个赞 共1个粉丝






写下你的评论...

全部评论 0  只看作者

 按时间倒序  按时间正序

被以下专题收入，发现更多相似内容

 随笔-生活工作点滴

推荐阅读 更多精彩内容 >

### equals()与hashCode()方法详解

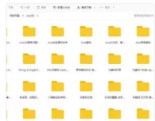
java.lang.Object类中有两个非常重要的方法：1publicbooleanequals(Object...

 小陈阿飞 阅读 119 评论 1 赞 1

### 学习Java编程equals()和hashCode()方法

equals()和hashCode()区别？ equals()：反映的是对象或变量具体的值，即两个对象里面包含的值...

 墨雨轩夏 阅读 344 评论 0 赞 17



### Java对象中equals()和hashCode()区别？

#equals()：反映的是对象或变量具体的值，即两个对象里面包含的值--可能是对象的引用，也可能是值类型的值。...

 liuzx32 阅读 414 评论 1 赞 9

### equals和hashcode区别？

equals()和hashCode()区别？ equals()：反映的是对象或变量具体的值，即两个对象里面包含的值...

 半路和尚怎么出家 阅读 91 评论 0 赞 2

### Java中的equals和hashCode方法详解



Java中的equals方法和hashCode方法是Object中的，所以每个对象都是有这两个方法的，有时候我们需--

 差不多先生\_t1 阅读 762 评论 2 赞 6

