

简书

首页

下载APP

搜索



Aa



登录

注册

# hashCode() 和equals() 区别和作用

骏骏的简书 [关注](#)

0.151 2016.07.21 03:20:04 字数 2,254 阅读 2,257

本章的 要解决下面几个问题:

11赞

- 1 equals() 的作用是什么?
  - 2 equals() 与 == 的区别是什么?
  - 3 hashCode() 的作用是什么?
  - 4 hashCode() 和 equals() 之间有什么联系?
- 赞赏

## 第1部分 equals() 的作用

equals() 的作用是 用来判断两个对象是否相等。

equals() 定义在JDK的Object.java中。通过判断两个对象的地址是否相等(即, 是否是同一个对象)来区分它们是否相等。源码如下:

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

既然Object.java中定义了equals()方法, 这就意味着所有的Java类都实现了equals()方法, 所有的类都可以通过equals()去比较两个对象是否相等。但是, 我们已经说过, 使用默认的“equals()”方法, 等价于“==”方法。因此, 我们通常会重写equals()方法: 若两个对象的内容相等, 则equals()方法返回true; 否则, 返回false。

下面根据“类是否覆盖equals()方法”, 将它分为2类。

(01) 若某个类没有覆盖equals()方法, 当它的通过equals()比较两个对象时, 实际上是比较两个对象是不是同一个对象。这时, 等价于通过“==”去比较这两个对象。

(02) 我们可以覆盖类的equals()方法, 来让equals()通过其它方式比较两个对象是否相等。通常的做法是: 若两个对象的内容相等, 则equals()方法返回true; 否则, 返回false。

下面, 举例对上面的2种情况进行说明。

1. “没有覆盖equals()方法”的情况

代码如下 (EqualsTest1.java):

```
1 import java.util.*;
```

写下你的评论...



评论1



赞11



骏骏的简书

总资产1 (约0.17元)

[关注](#)

Java中的sleep()和wait()的区别

阅读 209

Android 内存泄漏分析心得

阅读 96

### 推荐阅读

Go协程、并发、信道

阅读 47

抢红包算法

阅读 341

你以为你真的了解final吗?

阅读 848

MS-关于锁 (乐观锁, 悲观锁, 行锁、表锁, 共享锁, 排他锁)

阅读 2,188

记近三个月面试的经历 微软 阿里巴巴 字节跳动 依图科技 拼多多 (上海)

阅读 314

```
5  * @desc equals()的测试程序。
6  *
7  * @author skywang
8  * @email kuiwu-wang@163.com
9  */
10 public class EqualsTest1{
11
12     public static void main(String[] args) {
13         // 新建2个相同内容的Person对象,
14         // 再用equals比较它们是否相等
15         Person p1 = new Person("eee", 100);
16         Person p2 = new Person("eee", 100);
17         System.out.printf("%s\n", p1.equals(p2));
18     }
19
20     /**
21     * @desc Person类。
22     */
23     private static class Person {
24         int age;
25         String name;
26
27         public Person(String name, int age) {
28             this.name = name;
29             this.age = age;
30         }
31
32         public String toString() {
33             return name + " - " + age;
34         }
35     }
36 }
```

1 | 运行结果: false

## 结果分析：

1 | 我们通过 `p1.equals(p2)` 来“比较p1和p2是否相等时”。实际上，调用的Object.java的`equals()`方法，即调用  
2 | 而由 `p1` 和 `p2` 的定义可知，它们虽然内容相同；但它们是两个不同的对象！因此，返回结果是`false`。

## 2. "覆盖equals()方法"的情况

我们修改上面的EqualsTest1.java：覆盖equals()方法。

代码如下 (EqualsTest2.java)：

```
1 import java.util.*;
2 import java.lang.Comparable;
3
4 /**
5  * @desc equals()的测试程序。
6  *
7  */
8 public class EqualsTest2 {
9
10     public static void main(String[] args) {
11         // 新建2个相同内容的Person对象,
12         // 再用equals比较它们是否相等
```

```
13     Person p1 = new Person("eee", 100);
14     Person p2 = new Person("eee", 100);
15     System.out.printf("%s\n", p1.equals(p2));
16 }
17
18 /**
19  * @desc Person类。
20  */
21 private static class Person {
22     int age;
23     String name;
24
25     public Person(String name, int age) {
26         this.name = name;
27         this.age = age;
28     }
29
30     public String toString() {
31         return name + " - " + age;
32     }
33
34     /**
35      * @desc 覆盖equals方法
36      */
37     @Override
38     public boolean equals(Object obj) {
39         if (obj == null) {
40             return false;
41         }
42
43         // 如果是同一个对象返回true, 反之返回false
44         if (this == obj) {
45             return true;
46         }
47
48         // 判断是否类型相同
49         if (this.getClass() != obj.getClass()) {
50             return false;
51         }
52
53         Person person = (Person) obj;
54         return name.equals(person.name) && age == person.age;
55     }
56 }
57 }
```

1 | 运行结果: true

### 结果分析:

我们在EqualsTest2.java 中重写了Person的equals()函数: 当两个Person对象的 name 和 age 都相等, 则返回true。

因此, 运行结果返回true。

讲到这里, 顺便说一下java对equals()的要求。有以下几点:

1. 对称性: 如果x.equals(y)返回是"true", 那么y.equals(x)也应该返回是"true"。
2. 反射性: x.equals(x)必须返回是"true"。
3. 类推性: 如果x.equals(y)返回是"true", 而且y.equals(z)返回是"true", 那么z.equals(x)也应该返回是"true"。
4. 一致性: 如果x.equals(y)返回是"true", 只要x和y内容一直不变, 不管你重复x.equals(y)多少次, 返回都是"true"。

5 | 5. 非空性, `x.equals(null)`, 永远返回是"`false`"; `x.equals`(和`x`不同类型的对象)永远返回是"`false`".

现在, 再回顾一下`equals()`的作用: 判断两个对象是否相等。当我们重写`equals()`的时候, 可千万不好将它的作用给改变了!

## 第2部分 equals() 与 == 的区别是什么?

`==`: 它的作用是判断两个对象的地址是不是相等。即, 判断两个对象是不是同一个对象。

`equals()`: 它的作用也是判断两个对象是否相等。但它一般有两种使用情况(前面第1部分已详细介绍过):

情况1, 类没有覆盖`equals()`方法。则通过`equals()`比较该类的两个对象时, 等价于通过"`==`"比较这两个对象。

情况2, 类覆盖了`equals()`方法。一般, 我们都覆盖`equals()`方法来两个对象的内容相等; 若它们的内容相等, 则返回`true`(即, 认为这两个对象相等)。

下面, 通过示例比较它们的区别。

代码如下:

```
1 import java.util.*;
2 import java.lang.Comparable;
3
4 /**
5  * @desc equals()的测试程序。
6  *
7  * @author skywang
8  * @email kuiwu-wang@163.com
9  */
10 public class EqualsTest3{
11
12     public static void main(String[] args) {
13         // 新建2个相同内容的Person对象,
14         // 再用equals比较它们是否相等
15         Person p1 = new Person("eee", 100);
16         Person p2 = new Person("eee", 100);
17         System.out.printf("p1.equals(p2) : %s\n", p1.equals(p2));
18         System.out.printf("p1==p2 : %s\n", p1==p2);
19     }
20
21     /**
22     * @desc Person类。
23     */
24     private static class Person {
25         int age;
26         String name;
27
28         public Person(String name, int age) {
29             this.name = name;
30             this.age = age;
31         }
32
33         public String toString() {
34             return name + " - " + age;
35         }
36
37     }
38 }
```

```
38         * @desc 覆盖equals方法
39         */
40         @Override
41         public boolean equals(Object obj){
42             if(obj == null){
43                 return false;
44             }
45
46             //如果是同一个对象返回true, 反之返回false
47             if(this == obj){
48                 return true;
49             }
50
51             //判断是否类型相同
52             if(this.getClass() != obj.getClass()){
53                 return false;
54             }
55
56             Person person = (Person)obj;
57             return name.equals(person.name) && age==person.age;
58         }
59     }
60 }
```

```
1  运行结果:
2  p1.equals(p2) : true
3  p1==p2 : false
```

结果分析：

在EqualsTest3.java 中：

(01) p1.equals(p2)

这是判断p1和p2的内容是否相等。因为Person覆盖equals()方法，而这个equals()是用来判断p1和p2的内容是否相等，恰恰p1和p2的内容又相等；因此，返回true。

(02) p1==p2

这是判断p1和p2是否是同一个对象。由于它们是各自新建的两个Person对象；因此，返回false。

## 第3部分 hashCode() 的作用

hashCode() 的作用是获取哈希码，也称为散列码；它实际上是返回一个int整数。这个哈希码的作用是确定该对象在哈希表中的索引位置。

hashCode() 定义在JDK的Object.java中，这就意味着Java中的任何类都包含有hashCode() 函数。

虽然，每个Java类都包含hashCode() 函数。但是，仅仅当创建并某个“类的散列表”(关于“散列表”见下面说明)时，该类的hashCode() 才有用(作用是：确定该类的每一个对象在散列表中的位置；其它情况下(例如，创建类的单个对象，或者创建类的对象数组等等)，类的hashCode() 没有作用。

上面的散列表，指的是：Java集合中本质是散列表的类，如HashMap，Hashtable，HashSet。

1 | 也就是说：hashCode() 在散列表中才有用，在其它情况下没用。在散列表中hashCode() 的作用是获取对象的散

OK! 至此，我们搞清楚了：hashCode()的作用是获取散列码。但是，散列码是用来干什么的呢？为什么散列表需要散列码呢？要解决这些问题，就需要理解散列表！关于散列表的内容，非三言两语道的明白；

为了能理解后面的内容，这里简单的介绍一下散列码的作用。

的“值”。这其中就利用到了散列码！

某个位置的元素。而数组的位置，就是通过“键”来获取的；更进一步说，数组的位置，是通过“键”对应的散列码计算得到的。

下面，我们以HashSet为例，来深入说明hashCode()的作用。

假设，HashSet中已经有1000个元素。当插入第1001个元素时，需要怎么处理？因为HashSet是Set集合，它允许有重复元素。

“将第1001个元素逐个的和前面1000个元素进行比较”？显然，这个效率是相等低下的。散列表很好的解决了这个问题，它根据元素的散列码计算出元素在散列表中的位置，然后将元素插入该位置即可。对于相同的元素，自然是只保存了一个。

由此可知，若两个元素相等，它们的散列码一定相等；但反过来确不一定。在散列表中，

1、如果两个对象相等，那么它们的hashCode()值一定要相同；

2、如果两个对象hashCode()相等，它们并不一定相等。

注意：这是在散列表中的情况。在非散列表中一定如此！

对“hashCode()的作用”就谈这么多。

## 第4部分 hashCode() 和 equals() 的关系

接下面，我们讨论另外一个话题。网上很多文章将 hashCode() 和 equals 关联起来，有的讲的不透彻，有误导读者的嫌疑。在这里，我自己梳理了一下 “hashCode() 和 equals()的关系”。

我们以“类的用途”来将“hashCode() 和 equals()的关系”分2种情况来说明。

### 1. 第一种 不会创建“类对应的散列表”

```
1 | 这里所说的“不会创建类对应的散列表”是说：我们不会在HashSet, Hashtable, HashMap等等这些本质是散列
2 |
3 | 在这种情况下，该类的“hashCode() 和 equals()”没有半毛钱关系的！
4 | 这种情况下，equals() 用来比较该类的两个对象是否相等。而hashCode() 则根本没有任何作用，所以，不用理
```

下面，我们通过示例查看类的两个对象相等 以及 不等时hashCode()的取值。

源码如下 (NormalHashCodeTest.java)：

```
1 | import java.util.*;
2 | import java.lang.Comparable;
```

```

3
4 /**
5  * @desc 比较equals() 返回true 以及 返回false时, hashCode()的值。
6  *
7  * @author skywang
8  * @email kuiwu-wang@163.com
9  */
10 public class NormalHashCodeTest{
11
12     public static void main(String[] args) {
13         // 新建2个相同内容的Person对象,
14         // 再用equals比较它们是否相等
15         Person p1 = new Person("eee", 100);
16         Person p2 = new Person("eee", 100);
17         Person p3 = new Person("aaa", 200);
18         System.out.printf("p1.equals(p2) : %s; p1(%d) p2(%d)\n", p1.equals(p2), p1.hashCode(), p2.hashCode());
19         System.out.printf("p1.equals(p3) : %s; p1(%d) p3(%d)\n", p1.equals(p3), p1.hashCode(), p3.hashCode());
20     }
21
22     /**
23      * @desc Person类。
24      */
25     private static class Person {
26         int age;
27         String name;
28
29         public Person(String name, int age) {
30             this.name = name;
31             this.age = age;
32         }
33
34         public String toString() {
35             return name + " - " + age;
36         }
37
38         /**
39          * @desc 覆盖equals方法
40          */
41         public boolean equals(Object obj){
42             if(obj == null){
43                 return false;
44             }
45
46             //如果是同一个对象返回true, 反之返回false
47             if(this == obj){
48                 return true;
49             }
50
51             //判断是否类型相同
52             if(this.getClass() != obj.getClass()){
53                 return false;
54             }
55
56             Person person = (Person)obj;
57             return name.equals(person.name) && age==person.age;
58         }
59     }
60 }

```

```

1 运行结果:
2 p1.equals(p2) : true; p1(1169863946) p2(1901116749)
3 p1.equals(p3) : false; p1(1169863946) p3(2131949076)

```

从结果也可以看出：p1和p2相等的情况下，hashCode()也不一定相等。

## 2. 第二种 会创建“类对应的散列表”

```

1  这里所说的“会创建类对应的散列表”是说：我们会在HashSet，Hashtable，HashMap等等这些本质是散列表的类
2
3  在这种情况下，该类的“hashCode() 和 equals() ”是有关系的：
4  1)、如果两个对象相等，那么它们的hashCode()值一定相同。
5      这里的相等是指，通过equals()比较两个对象时返回true。
6  2)、如果两个对象hashCode()相等，它们并不一定相等。
7      因为在散列表中，hashCode()相等，即两个键值对的哈希值相等。然而哈希值相等，并不一定能得出键值
8
9  此外，在这种情况下。若要判断两个对象是否相等，除了要覆盖equals()之外，也要覆盖hashCode()函数。否则

```

例如，创建Person类的HashSet集合，必须同时覆盖Person类的equals() 和 hashCode()方法。

如果单单只是覆盖equals()方法。我们会发现，equals()方法没有达到我们想要的效果。

参考代码 (ConflictHashCodeTest1.java)：

```

1  import java.util.*;
2  import java.lang.Comparable;
3
4  /**
5   * @desc 比较equals() 返回true 以及 返回false时， hashCode()的值。
6   *
7   * @author skywang
8   * @email kuiwu-wang@163.com
9   */
10 public class ConflictHashCodeTest1{
11
12     public static void main(String[] args) {
13         // 新建Person对象，
14         Person p1 = new Person("eee", 100);
15         Person p2 = new Person("eee", 100);
16         Person p3 = new Person("aaa", 200);
17
18         // 新建HashSet对象
19         HashSet set = new HashSet();
20         set.add(p1);
21         set.add(p2);
22         set.add(p3);
23
24         // 比较p1 和 p2， 并打印它们的hashCode()
25         System.out.printf("p1.equals(p2) : %s; p1(%d) p2(%d)\n", p1.equals(p2), p1.hashCode(), p2.hashCode());
26         // 打印set
27         System.out.printf("set:%s\n", set);
28     }
29
30     /**
31      * @desc Person类。
32      */
33     private static class Person {
34         int age;
35         String name;
36
37         public Person(String name, int age) {
38             this.name = name;
39             this.age = age;
40         }
41     }

```



```

42     public String toString() {
43         return "("+name + ", " +age+")";
44     }
45
46     /**
47      * @desc 覆盖equals方法
48      */
49     @Override
50     public boolean equals(Object obj){
51         if(obj == null){
52             return false;
53         }
54
55         //如果是同一个对象返回true, 反之返回false
56         if(this == obj){
57             return true;
58         }
59
60         //判断是否类型相同
61         if(this.getClass() != obj.getClass()){
62             return false;
63         }
64
65         Person person = (Person)obj;
66         return name.equals(person.name) && age==person.age;
67     }
68 }
69 }

```

```

1  运行结果:
2  p1.equals(p2) : true; p1(1169863946) p2(1690552137)
3  set:[(eee, 100), (eee, 100), (aaa, 200)]

```

### 结果分析:

```

1  我们重写了Person的equals()。但是, 很奇怪的发现: HashSet中仍然有重复元素: p1 和 p2。为什么会这样?
2
3  这是因为虽然p1 和 p2的内容相等, 但是它们的hashCode()不等; 所以, HashSet在添加p1和p2的时候, 认为它们是不同的元素。

```

下面, 我们同时覆盖equals() 和 hashCode()方法。

参考代码 (ConflictHashCodeTest2.java):

```

1  import java.util.*;
2  import java.lang.Comparable;
3
4  /**
5   * @desc 比较equals() 返回true 以及 返回false时, hashCode()的值。
6   *
7   * @author skywang
8   * @email kuiwu-wang@163.com
9   */
10 public class ConflictHashCodeTest2{
11
12     public static void main(String[] args) {
13         // 新建Person对象,
14         Person p1 = new Person("eee", 100);
15         Person p2 = new Person("eee", 100);
16         Person p3 = new Person("aaa", 200);

```

```
17         Person p4 = new Person("EEE", 100);
18
19         // 新建HashSet对象
20         HashSet set = new HashSet();
21         set.add(p1);
22         set.add(p2);
23         set.add(p3);
24
25         // 比较p1 和 p2, 并打印它们的hashCode()
26         System.out.printf("p1.equals(p2) : %s; p1(%d) p2(%d)\n", p1.equals(p2), p1.hashCode(), p2.hashCode());
27         // 比较p1 和 p4, 并打印它们的hashCode()
28         System.out.printf("p1.equals(p4) : %s; p1(%d) p4(%d)\n", p1.equals(p4), p1.hashCode(), p4.hashCode());
29         // 打印set
30         System.out.printf("set:%s\n", set);
31     }
32
33     /**
34      * @desc Person类。
35      */
36     private static class Person {
37         int age;
38         String name;
39
40         public Person(String name, int age) {
41             this.name = name;
42             this.age = age;
43         }
44
45         public String toString() {
46             return name + " - " + age;
47         }
48
49         /**
50          * @desc 重写hashCode
51          */
52         @Override
53         public int hashCode(){
54             int nameHash = name.toUpperCase().hashCode();
55             return nameHash ^ age;
56         }
57
58         /**
59          * @desc 覆盖equals方法
60          */
61         @Override
62         public boolean equals(Object obj){
63             if(obj == null){
64                 return false;
65             }
66
67             //如果是同一个对象返回true, 反之返回false
68             if(this == obj){
69                 return true;
70             }
71
72             //判断是否类型相同
73             if(this.getClass() != obj.getClass()){
74                 return false;
75             }
76
77             Person person = (Person)obj;
78             return name.equals(person.name) && age==person.age;
79         }
80     }
81 }
```

```
1 运行结果:
2 p1.equals(p2) : true; p1(68545) p2(68545)
3 p1.equals(p4) : false; p1(68545) p4(68545)
4 set:[aaa - 200, eee - 100]
```

```
1 结果分析:
2 这下, equals()生效了, HashSet中没有重复元素。
3 比较p1和p2, 我们发现: 它们的hashCode()相等, 通过equals()比较它们也返回true。所以, p1和p2被视为相
4 比较p1和p4, 我们发现: 虽然它们的hashCode()相等; 但是, 通过equals()比较它们返回false。所以, p1和p
```

 11人点赞 > Android 面试要点

"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



骏骏的简书

总资产1 (约0.17元) 共写了3.8W字 获得141个赞 共64个粉丝

关注

英国大学前十名

美国地图

开服时间表

分手复合的方法

丰台孕妇摄影

美国前30名大学

哈佛大学录取线

巴厘岛举办婚礼

53度茅台酒报价

三亚过冬短租房

排名美国大学

北海现在房价

北海房价

上海房价下降

美国高中哪些学校好

写下你的评论...

全部评论 1 只看作者

按时间倒序 按时间正序



中原1991

2楼 2017.02.14 01:06

"假设, HashSet中已经有1000个元素。当插入第1001个元素时, 需要怎么处理? 因为HashSet是Set集合, 它允许有重复元素。"有点问题, 不允许有重复元素才对。

👍 2 回复

被以下专题收入, 发现更多相似内容



抽象类



Java干货



JAVA学习之路

## 推荐阅读

[更多精彩内容 >](#)

### HashCode() 和 equals()的若干问题

参考:Java hashCode() 和 equals()的若干问题解答 本章的内容主要解决下面几个问题: 1 eq...



jacky123 阅读 183 评论 0 赞 4

### Android 面试准备之「equals 和 ==」

本文为原创文章，如需转载请注明出处，谢谢! 概述 本文将围绕以下五点进行说明 1.equals() 和 == 的作...



Android\_ZzT 阅读 928 评论 0 赞 8

### Java 基础复习实践 --- Hashcode Equals

虽然很多知识点书籍都有整理，但是记性总是不好，所以决定将一些细小容易混淆的概念，通过简单的 Demo 实践，加深复...



谢三弟 阅读 197 评论 1 赞 3

### 面试知识点1

Java8张图 11、字符串不变性 12、equals()方法、hashCode()方法的区别 13、...



Miley\_MOJIE 阅读 1,733 评论 0 赞 6

### 鞋合不合适，只有脚知道

2017.7.15 鞋合不合适，只有脚知道。别人夸的再天花乱坠，鞋柜里看起来再光鲜亮丽，也只有上脚了，才知道适不适...



热闹且孤独 阅读 225 评论 0 赞 0