<https://www.cnblogs.com/dolphin0520/p/3681042.html>

# [浅谈Java中的hashcode方法](https://www.cnblogs.com/dolphin0520/p/3681042.html)

　浅谈Java中的hashcode方法

　　哈希表这个数据结构想必大多数人都不陌生，而且在很多地方都会利用到hash表来提高查找效率。在Java的Object类中有一个方法:

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

　　根据这个方法的声明可知，该方法返回一个int类型的数值，并且是本地方法，因此在Object类中并没有给出具体的实现。

　　为何Object类需要这样一个方法？它有什么作用呢？今天我们就来具体探讨一下hashCode方法。

## 一.hashCode方法的作用

　　对于包含容器类型的程序设计语言来说，基本上都会涉及到hashCode。在Java中也一样，hashCode方法的主要作用是为了配合基于散列的集合一起正常运行，这样的散列集合包括HashSet、HashMap以及HashTable。

　　为什么这么说呢？考虑一种情况，当向集合中插入对象时，如何判别在集合中是否已经存在该对象了？（注意：集合中不允许重复的元素存在）

　　也许大多数人都会想到调用equals方法来逐个进行比较，这个方法确实可行。但是如果集合中已经存在一万条数据或者更多的数据，如果采用equals方法去逐一比较，效率必然是一个问题。此时hashCode方法的作用就体现出来了，当集合要添加新的对象时，先调用这个对象的hashCode方法，得到对应的hashcode值，实际上在HashMap的具体实现中会用一个table保存已经存进去的对象的hashcode值，如果table中没有该hashcode值，它就可以直接存进去，不用再进行任何比较了；如果存在该hashcode值， 就调用它的equals方法与新元素进行比较，相同的话就不存了，不相同就散列其它的地址，所以这里存在一个冲突解决的问题，这样一来实际调用equals方法的次数就大大降低了，说通俗一点：Java中的hashCode方法就是根据一定的规则将与对象相关的信息（比如对象的存储地址，对象的字段等）映射成一个数值，这个数值称作为散列值。下面这段代码是java.util.HashMap的中put方法的具体实现：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | public V put(K key, V value) {          if (key == null)              return putForNullKey(value);          int hash = hash(key.hashCode());          int i = indexFor(hash, table.length);          for (Entry<K,V> e = table[i]; e != null; e = e.next) {              Object k;              if (e.hash == hash && ((k = e.key) == key || key.equals(k))) {                  V oldValue = e.value;                  e.value = value;                  e.recordAccess(this);                  return oldValue;              }          }            modCount++;          addEntry(hash, key, value, i);          return null;      } |

　　put方法是用来向HashMap中添加新的元素，从put方法的具体实现可知，会先调用hashCode方法得到该元素的hashCode值，然后查看table中是否存在该hashCode值，如果存在则调用equals方法重新确定是否存在该元素，如果存在，则更新value值，否则将新的元素添加到HashMap中。从这里可以看出，hashCode方法的存在是为了减少equals方法的调用次数，从而提高程序效率。

　　如果对于hash表这个数据结构的朋友不清楚，可以参考这几篇博文;

<http://www.cnblogs.com/jiewei915/archive/2010/08/09/1796042.html>

<http://www.cnblogs.com/dolphin0520/archive/2012/09/28/2700000.html>

<http://www.java3z.com/cwbwebhome/article/article8/83560.html?id=4649>

　　有些朋友误以为默认情况下，hashCode返回的就是对象的存储地址，事实上这种看法是不全面的，确实有些JVM在实现时是直接返回对象的存储地址，但是大多时候并不是这样，只能说可能存储地址有一定关联。下面是HotSpot JVM中生成hash散列值的实现：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | static inline intptr\_t get\_next\_hash(Thread \* Self, oop obj) {    intptr\_t value = 0 ;    if (hashCode == 0) {       // This form uses an unguarded global Park-Miller RNG,       // so it's possible for two threads to race and generate the same RNG.       // On MP system we'll have lots of RW access to a global, so the       // mechanism induces lots of coherency traffic.       value = os::random() ;    } else    if (hashCode == 1) {       // This variation has the property of being stable (idempotent)       // between STW operations.  This can be useful in some of the 1-0       // synchronization schemes.       intptr\_t addrBits = intptr\_t(obj) >> 3 ;       value = addrBits ^ (addrBits >> 5) ^ GVars.stwRandom ;    } else    if (hashCode == 2) {       value = 1 ;            // for sensitivity testing    } else    if (hashCode == 3) {       value = ++GVars.hcSequence ;    } else    if (hashCode == 4) {       value = intptr\_t(obj) ;    } else {       // Marsaglia's xor-shift scheme with thread-specific state       // This is probably the best overall implementation -- we'll       // likely make this the default in future releases.       unsigned t = Self->\_hashStateX ;       t ^= (t << 11) ;       Self->\_hashStateX = Self->\_hashStateY ;       Self->\_hashStateY = Self->\_hashStateZ ;       Self->\_hashStateZ = Self->\_hashStateW ;       unsigned v = Self->\_hashStateW ;       v = (v ^ (v >> 19)) ^ (t ^ (t >> 8)) ;       Self->\_hashStateW = v ;       value = v ;    }      value &= markOopDesc::hash\_mask;    if (value == 0) value = 0xBAD ;    assert (value != markOopDesc::no\_hash, "invariant") ;    TEVENT (hashCode: GENERATE) ;    return value;  } |

　　该实现位于hotspot/src/share/vm/runtime/synchronizer.cpp文件下。

　　因此有人会说，可以直接根据hashcode值判断两个对象是否相等吗？肯定是不可以的，因为不同的对象可能会生成相同的hashcode值。虽然不能根据hashcode值判断两个对象是否相等，但是可以直接根据hashcode值判断两个对象不等，如果两个对象的hashcode值不等，则必定是两个不同的对象。如果要判断两个对象是否真正相等，必须通过equals方法。

　　也就是说对于两个对象，如果调用equals方法得到的结果为true，则两个对象的hashcode值必定相等；

　　如果equals方法得到的结果为false，则两个对象的hashcode值不一定不同；

　　如果两个对象的hashcode值不等，则equals方法得到的结果必定为false；

　　如果两个对象的hashcode值相等，则equals方法得到的结果未知。

## 二.equals方法和hashCode方法

　　在有些情况下，程序设计者在设计一个类的时候为需要重写equals方法，比如String类，但是千万要注意，在重写equals方法的同时，必须重写hashCode方法。为什么这么说呢？

　　下面看一个例子：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | package com.cxh.test1;    import java.util.HashMap;  import java.util.HashSet;  import java.util.Set;      class People{      private String name;      private int age;        public People(String name,int age) {          this.name = name;          this.age = age;      }        public void setAge(int age){          this.age = age;      }        @Override      public boolean equals(Object obj) {          // TODO Auto-generated method stub          return this.name.equals(((People)obj).name) && this.age== ((People)obj).age;      }  }    public class Main {        public static void main(String[] args) {            People p1 = new People("Jack", 12);          System.out.println(p1.hashCode());            HashMap<People, Integer> hashMap = new HashMap<People, Integer>();          hashMap.put(p1, 1);            System.out.println(hashMap.get(new People("Jack", 12)));      }  } |

　　在这里我只重写了equals方法，也就说如果两个People对象，如果它的姓名和年龄相等，则认为是同一个人。

　　这段代码本来的意愿是想这段代码输出结果为“1”，但是事实上它输出的是“null”。为什么呢？原因就在于重写equals方法的同时忘记重写hashCode方法。

　　虽然通过重写equals方法使得逻辑上姓名和年龄相同的两个对象被判定为相等的对象（跟String类类似），但是要知道默认情况下，hashCode方法是将对象的存储地址进行映射。那么上述代码的输出结果为“null”就不足为奇了。原因很简单，p1指向的对象和

　　System.out.println(hashMap.get(new People("Jack", 12)));这句中的new People("Jack", 12)生成的是两个对象，它们的存储地址肯定不同。下面是HashMap的get方法的具体实现：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | public V get(Object key) {          if (key == null)              return getForNullKey();          int hash = hash(key.hashCode());          for (Entry<K,V> e = table[indexFor(hash, table.length)];               e != null;               e = e.next) {              Object k;              if (e.hash == hash && ((k = e.key) == key || key.equals(k)))                  return e.value;          }          return null;      } |

　　所以在hashmap进行get操作时，因为得到的hashcdoe值不同（注意，上述代码也许在某些情况下会得到相同的hashcode值，不过这种概率比较小，因为虽然两个对象的存储地址不同也有可能得到相同的hashcode值），所以导致在get方法中for循环不会执行，直接返回null。

　　因此如果想上述代码输出结果为“1”，很简单，只需要重写hashCode方法，让equals方法和hashCode方法始终在逻辑上保持一致性。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 | package com.cxh.test1;    import java.util.HashMap;  import java.util.HashSet;  import java.util.Set;      class People{      private String name;      private int age;        public People(String name,int age) {          this.name = name;          this.age = age;      }        public void setAge(int age){          this.age = age;      }        @Override      public int hashCode() {          // TODO Auto-generated method stub          return name.hashCode()\*37+age;      }        @Override      public boolean equals(Object obj) {          // TODO Auto-generated method stub          return this.name.equals(((People)obj).name) && this.age== ((People)obj).age;      }  }    public class Main {        public static void main(String[] args) {            People p1 = new People("Jack", 12);          System.out.println(p1.hashCode());            HashMap<People, Integer> hashMap = new HashMap<People, Integer>();          hashMap.put(p1, 1);            System.out.println(hashMap.get(new People("Jack", 12)));      }  } |

　　这样一来的话，输出结果就为“1”了。

　　下面这段话摘自Effective Java一书：

* 在程序执行期间，只要equals方法的比较操作用到的信息没有被修改，那么对这同一个对象调用多次，hashCode方法必须始终如一地返回同一个整数。
* 如果两个对象根据equals方法比较是相等的，那么调用两个对象的hashCode方法必须返回相同的整数结果。
* 如果两个对象根据equals方法比较是不等的，则hashCode方法不一定得返回不同的整数。

　　对于第二条和第三条很好理解，但是第一条，很多时候就会忽略。在《Java编程思想》一书中的P495页也有同第一条类似的一段话：

　　“设计hashCode()时最重要的因素就是：无论何时，对同一个对象调用hashCode()都应该产生同样的值。如果在讲一个对象用put()添加进HashMap时产生一个hashCdoe值，而用get()取出时却产生了另一个hashCode值，那么就无法获取该对象了。所以如果你的hashCode方法依赖于对象中易变的数据，用户就要当心了，因为此数据发生变化时，hashCode()方法就会生成一个不同的散列码”。

　　下面举个例子：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | package com.cxh.test1;    import java.util.HashMap;  import java.util.HashSet;  import java.util.Set;      class People{      private String name;      private int age;        public People(String name,int age) {          this.name = name;          this.age = age;      }        public void setAge(int age){          this.age = age;      }        @Override      public int hashCode() {          // TODO Auto-generated method stub          return name.hashCode()\*37+age;      }        @Override      public boolean equals(Object obj) {          // TODO Auto-generated method stub          return this.name.equals(((People)obj).name) && this.age== ((People)obj).age;      }  }    public class Main {        public static void main(String[] args) {            People p1 = new People("Jack", 12);          System.out.println(p1.hashCode());            HashMap<People, Integer> hashMap = new HashMap<People, Integer>();          hashMap.put(p1, 1);            p1.setAge(13);            System.out.println(hashMap.get(p1));      }  } |

　　这段代码输出的结果为“null”，想必其中的原因大家应该都清楚了。

　　因此，在设计hashCode方法和equals方法的时候，如果对象中的数据易变，则最好在equals方法和hashCode方法中不要依赖于该字段。

　　以上属个人理解，如有不正之处，欢迎批评指正。

作者：[海子](http://www.cnblogs.com/dolphin0520/)

出处：<http://www.cnblogs.com/dolphin0520/>

本博客中未标明转载的文章归作者[海子](http://www.cnblogs.com/dolphin0520/)和博客园共有，欢迎转载，但未经作者同意必须保留此段声明，且在文章页面明显位置给出原文连接，否则保留追究法律责任的权利。

分类: [Java基础](https://www.cnblogs.com/dolphin0520/category/361055.html)

标签: [Java](https://www.cnblogs.com/dolphin0520/tag/Java/)

[**好文要顶**](javascript:void(0);) [**关注我**](javascript:void(0);) [**收藏该文**](javascript:void(0);) **[https://common.cnblogs.com/images/icon_weibo_24.png](javascript:void(0);)** **[https://common.cnblogs.com/images/wechat.png](javascript:void(0);)**

[https://pic.cnblogs.com/face/u288799.jpg?id=05221746](http://home.cnblogs.com/u/dolphin0520/)

[海 子](http://home.cnblogs.com/u/dolphin0520/)  
[关注 - 6](http://home.cnblogs.com/u/dolphin0520/followees)  
[粉丝 - 6330](http://home.cnblogs.com/u/dolphin0520/followers)

[+加关注](javascript:void(0);)

77

0

[«](https://www.cnblogs.com/dolphin0520/p/3752492.html)上一篇：[GNU C中的零长度数组（转载）](https://www.cnblogs.com/dolphin0520/p/3752492.html)  
[»](https://www.cnblogs.com/dolphin0520/p/3613043.html)下一篇：[JVM的内存区域划分](https://www.cnblogs.com/dolphin0520/p/3613043.html)

posted @ 2014-05-27 11:46 [海 子](https://www.cnblogs.com/dolphin0520/) 阅读(172028) 评论(35) [编辑](https://i.cnblogs.com/EditPosts.aspx?postid=3681042) [收藏](https://www.cnblogs.com/dolphin0520/p/3681042.html)

**评论列表**

[#1楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#2950549) 2014-05-27 14:18 [打一个情](https://www.cnblogs.com/houguiqiang/) 

好文章

[支持(0)反对(0)](javascript:void(0);)

[#2楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#2950667) 2014-05-27 15:42 [ImaginationOf左撇子](http://home.cnblogs.com/u/576696/) 

温故而知新。看了贴对hashCode、equals有了更深入的理解。

[支持(6)反对(0)](javascript:void(0);)

[#3楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3123103) 2015-02-04 10:47 [星星亮了](http://home.cnblogs.com/u/709312/) 

好

[支持(0)反对(0)](javascript:void(0);)

[#4楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3131883) 2015-02-26 15:48 [BeCoded](https://www.cnblogs.com/D-Key/) 

我有一个疑问，就是（如果存在该hashcode值， 就调用它的equals方法与新元素进行比较，相同的话就不存了，不相同就散列其它的地址）这个地方，为什么table中有了那个hash值还要调用equals方法来重新验证？我的猜想是：  
1、可能不同的key也能产生相同的hash值(在网上找了一下散列函数的定义，好像这种可能可以排除)  
2、防止人为重写了hashcode方法(这个我觉得可能性是最大的)  
不知道我的想法对不对，希望楼主指点指点，谢谢

[支持(3)反对(0)](javascript:void(0);)

[#5楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3195344) 2015-05-28 14:50 [G.hang](https://www.cnblogs.com/Ghang/) 

私以为，你写的equals方法并不是很标准，core java上的建议写法是：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | @Override      public boolean equals(Object obj) {          if (this == obj) {              return true;          }          if (obj == null) {              return false;          }          if (this.getClass() != obj.getClass()) {              return false;          }            People p = (People)obj;          return this.name.equals(p.name)              && this.age == p.age;      } |

[支持(20)反对(2)](javascript:void(0);)

[#6楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3200116) 2015-06-04 11:06 [繁华未至](http://home.cnblogs.com/u/669327/) 

好文章。。。

[支持(0)反对(0)](javascript:void(0);)

[#7楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3220863) 2015-07-03 13:34 [云中烛火](https://www.cnblogs.com/yunzhongzhuhuo/) 

不错

[支持(0)反对(0)](javascript:void(0);)

[#8楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3239803) 2015-08-02 16:02 [张杰克](http://home.cnblogs.com/u/772688/) 

好文！

[支持(0)反对(0)](javascript:void(0);)

[#9楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3249262) 2015-08-17 17:12 [想做大哥很多年](https://www.cnblogs.com/cauyahui/) 

写的很透彻

[支持(0)反对(0)](javascript:void(0);)

[#10楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3250853) 2015-08-20 13:31 [情报员](http://home.cnblogs.com/u/766027/) 

好文章，力顶

[支持(0)反对(0)](javascript:void(0);)

[#11楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3352713) 2016-01-22 17:59 [红莲如梦](http://home.cnblogs.com/u/762460/) 

谢谢分享~~~

[支持(0)反对(0)](javascript:void(0);)

[#12楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3367843) 2016-02-27 18:59 [Ansonwoo](http://home.cnblogs.com/u/739453/) 

Hi  
  
重写的hashcode() 怎么知道要这么写:  
  
return name.hashCode()\*37+age;  
  
谢谢

[支持(0)反对(0)](javascript:void(0);)

[#13楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3369756) 2016-03-01 22:11 [nice-gh](https://www.cnblogs.com/nice-gh/) 

写的挺好的，很通俗、透彻

[支持(0)反对(0)](javascript:void(0);)

[#14楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3377839) 2016-03-11 12:04 [马哈](http://home.cnblogs.com/u/911007/) 

看不懂O(∩\_∩)O

[支持(1)反对(0)](javascript:void(0);)

[#15楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3379497) 2016-03-13 11:40 [maxiaoge](https://www.cnblogs.com/xiaomacdut/) 

如果两个对象的hashcode值不等，则equals方法得到的结果必定为false；  
  
这句话我感觉不对，因为hashcode()和equals()方法是可以重写的，如果  
hashcode（）返回的是一个随机数，两个对象的hashcode不同，但是equals用了属性字段来计算，equals方法得到的可能是true，可能是false。

[支持(1)反对(6)](javascript:void(0);)

[#16楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3383997) 2016-03-17 17:19 [zryjasan](https://www.cnblogs.com/MarvinRA/) 

好文章， 谢谢分享

[支持(0)反对(0)](javascript:void(0);)

[#17楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3419551) 2016-04-28 16:40 [lszy24](http://home.cnblogs.com/u/573960/) 

我觉得hashcode可以看作是人的名字，人名相同的不一定是同一个人  
但是如果人名不同的话肯定不是同一个人了  
  
你说的这种情况是存在，但是没什么意义

[支持(3)反对(0)](javascript:void(0);)

[#18楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3476295) 2016-07-25 17:05 [Minho\_yu](http://home.cnblogs.com/u/998242/) 

[@](https://www.cnblogs.com/dolphin0520/p/3681042.html#3131883) BeCoded  
因为在hashmap中，当两个不同对象的hashcode相同(或者两个对象在调用hashmap的hash方法得到的值相同)时，会把这两个对象存在hashmap中同一索引的位置下的链表上。两个不同的对象分别存在同一索引下的链表的不同位置上(java8中加入了红黑树)，也就是说hashmap中同一个索引位置是可能存着多个对象的。代码中e.next也是对链表遍历的操作。所以即使hash值相同，还是要equals来比较是否为同一个对象。

[支持(6)反对(0)](javascript:void(0);)

[#19楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3489992) 2016-08-16 09:07 [orzangleli](https://www.cnblogs.com/orzangleli/) 

[@](https://www.cnblogs.com/dolphin0520/p/3681042.html#3131883) BeCoded  
第一条是很可能的，并不是不可能，因为有些hashCode方式只是对参数的一个映射，可以这么说：两个类，如果参数完全相同，hashCode应该是相同的，但是hashCode相同，不一定字参数相同。举个简单例子：

|  |  |
| --- | --- |
| 1  2  3  4  5 | @Override      public int hashCode() {          // TODO Auto-generated method stub          return age;      } |

hashCode里只对年龄进行判断，实际上是对所有的对象进行按年龄分类了，如果hashCode相同再去判断equals，因为hashCode的执行速度比equals快，所以在数据量大时，hashCode进行筛选排除绝对不同的对象，对剩下进行equals判断，能够大大提高效率。

[支持(2)反对(0)](javascript:void(0);)

[#20楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3507703) 2016-09-10 23:13 [32ddd](https://www.cnblogs.com/masterlibin/) 

[@](https://www.cnblogs.com/dolphin0520/p/3681042.html#3195344) G.hang  
确实，要先看看object是不是String类。

[支持(0)反对(0)](javascript:void(0);)

[#21楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3507704) 2016-09-10 23:14 [32ddd](https://www.cnblogs.com/masterlibin/) 

好文章，看了不少作者的文章。谢谢作者。

[支持(0)反对(0)](javascript:void(0);)

[#22楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3536971) 2016-10-20 15:35 [天旋逸风](http://home.cnblogs.com/u/1026304/) 

[@](https://www.cnblogs.com/dolphin0520/p/3681042.html#3379497) maxiaoge  
前提是没被重写，否则随便怎么重写，都能搞成相等的，结论也就不成立了

[支持(0)反对(0)](javascript:void(0);)

[#23楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3587261) 2016-12-22 16:35 [登陆成功](http://home.cnblogs.com/u/1084931/) 

|  |  |
| --- | --- |
| 1  2  3  4 | public int hashCode() {          // TODO Auto-generated method stub          return name.hashCode()\*37+age;  } |

这个重写的方法根据字符串name的hashCode来生成，可能不一定得到想要的结果。因为楼主说过跟存储地址有关，如果一个name的值来自初始化赋值的“jack”，另一个name的值从某个字符串赋值过来，两个值的地址就不同了，导致的结果是，name的值都是jack，但是hashCode就不一样了。

[支持(0)反对(0)](javascript:void(0);)

[#24楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3637149) 2017-03-10 16:19 [傍晚的羊羔](http://home.cnblogs.com/u/1048273/) 

[@](https://www.cnblogs.com/dolphin0520/p/3681042.html#3379497) maxiaoge  
但是，我认为不会有人故意这样写的，如果这样写，那就是故意留漏洞。

[支持(0)反对(0)](javascript:void(0);)

[#25楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3686456) 2017-05-08 10:59 [七友DyL](http://home.cnblogs.com/u/1029224/) 

@登录成功  
只能说可能存储地址有间接关联，String类的hashCode()值是通过字符串的长度以及内容计算获得的。  
public int hashCode() {  
int h = hash;  
if (h == 0 && value.length > 0) {  
char val[] = value;  
  
for (int i = 0; i < value.length; i++) {  
h = 31 \* h + val[i];  
}  
hash = h;  
}  
return h;  
}  
所以如果字符串的内容相同，获得的hashCode值也一样。

[支持(0)反对(0)](javascript:void(0);)

[#26楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3833882) 2017-11-07 18:59 [kevinliukaiwen](http://home.cnblogs.com/u/1274484/) 

每个博文写的都好详细

[支持(0)反对(0)](javascript:void(0);)

[#27楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3842869) 2017-11-17 10:00 [风雪山神庙](https://www.cnblogs.com/maenqiong/) 

引用经典，有理有据，  
目录清晰，文字表达清楚，  
图文并茂，字体分明，颜色有区分；  
博客文章的好范文。

[支持(0)反对(0)](javascript:void(0);)

[#28楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3865864) 2017-12-14 15:29 [睡你的椒](http://home.cnblogs.com/u/1299058/) 

MARK

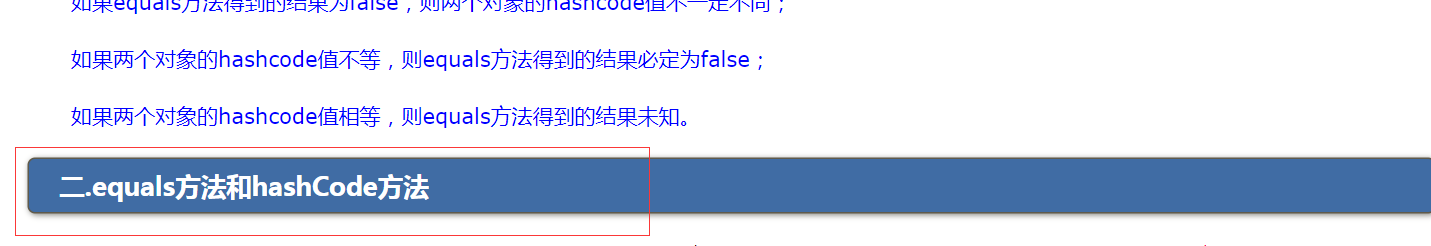
[支持(0)反对(0)](javascript:void(0);)

[#29楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3923044) 2018-03-14 21:33 [N3verL4nd](https://www.cnblogs.com/lgh1992314/) 

[@](https://www.cnblogs.com/dolphin0520/p/3681042.html#3379497) maxiaoge  
这种只是约定，人家API写的只是建议采用。  
hashcode()相等的两个对象，equals()不一定相等；而equals()相等的两个对象，hashcode()一定相等

[支持(0)反对(0)](javascript:void(0);)

[#30楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#3939511) 2018-04-01 18:07 [飘云粟](https://www.cnblogs.com/cafebabe-yun/) 

  
请问海子前辈，标题的蓝色框框怎么添加

[支持(0)反对(0)](javascript:void(0);)

[#31楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#4012382) 2018-07-04 15:11 [dashuai的博客](https://www.cnblogs.com/kubixuesheng/) 

[@](https://www.cnblogs.com/dolphin0520/p/3681042.html#3195344) G.hang  
[引用](https://www.cnblogs.com/dolphin0520/p/3681042.html#3195344)私以为，你写的equals方法并不是很标准，core java上的建议写法是：  
  
  
你这个写法也不好啊,不需要判断null,直接使用 instanceof 运算符即可,getClass都不用写.这也是第三版 effective Java中的推荐写法

[支持(0)反对(0)](javascript:void(0);)

[#32楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#4016407) 2018-07-09 18:16 [dongsir2017](https://www.cnblogs.com/dongsir2018/) 

[@](https://www.cnblogs.com/dolphin0520/p/3681042.html#3637149) 傍晚的羊羔  
不会的，字符串的hashCode方法是通过计算通过每个字符进行相关计算的到的，所以跟地址没有关系，只和字符串的内容有关系。  
如下字符串hashCode源码：  
public int hashCode() {  
int h = hash;  
if (h == 0 && value.length > 0) {  
char val[] = value;  
  
for (int i = 0; i < value.length; i++) {  
h = 31 \* h + val[i];  
}  
hash = h;  
}  
return h;  
}

[支持(0)反对(0)](javascript:void(0);)

[#33楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#4016410) 2018-07-09 18:20 [dongsir2017](https://www.cnblogs.com/dongsir2018/) 

[@](https://www.cnblogs.com/dolphin0520/p/3681042.html#3383997) zryjasan  
"**如果两个对象的hashcode值不等，则equals方法得到的结果必定为false**；  
"这句话准确来说是一个契约，是规定。如果不按照规定来就会出现博主所举示例中的问题了。

[支持(0)反对(0)](javascript:void(0);)

[#34楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#4079427) 2018-09-29 00:18 [ytZhang](https://www.cnblogs.com/ZYTZ/) 

讲的太好了，感谢

[支持(0)反对(0)](javascript:void(0);)

[#35楼](https://www.cnblogs.com/dolphin0520/p/3681042.html#4094158) 2018-10-19 16:01 [沙漠冥王星](http://home.cnblogs.com/u/1241140/) 

good!!