正确重写hashCode的办法

http://simg.sinajs.cn/blog7style/images/common/sg_trans.gifhttp://simg.sinajs.cn/blog7style/images/common/sg_trans.gif (2013-11-25 14:51:31)

|  |  |
| --- | --- |
|  |  |

**转自：<http://blog.csdn.net/benjaminzhang666/article/details/9468605>**

**——————————————————————————————————————————**

**正确重写hashCode的办法**

[1]. hashCode重写成**相同的值**的**缺点**

将所有对象的hashCode都返回一样的值是不科学的。比如a1和a3这两个根本不同的对象，就没有必要去比较equals，**增加无谓的计算量**。**所以应该对象本身的内容 (属性)来重写hashCode。**

一旦两个**对象内部**不一样，就**直接判定出hashCode不一样**，**不用再调用equals进行比较**。

[2]. 正确书写hashCode的办法：

【原则】按照**equals( )**中**比较两个对象是否一致**的**条件用到的属性**来**重写hashCode()**。

{1}. 常用的办法就是利用**涉及到的的属性**进行**线性组合**。

{2}. **线性组合**过程中涉及到的**组合系数**自定义即可。

注意，拼接之后的数值不能超过整形的表达范围。

{3}. 公式：**属性1的int形式**+ C1\***属性2的int形式**+  C2\***属性3的int形式**+ …

【技巧】当属性是**引用**类型的时候，如果**已经重写过hashCode()**，那么这个引用属性的int形式就是直接调用属性已有的hashCode值。

**最典型**的就是这个属性是**字符串类型**的，String类型已经重写了hashCode()方法，所以直接拿来使用即可。

**——————————————————————————————————————————**

**(1). 主要代码**

**[java]** [view plaincopy](http://blog.csdn.net/benjaminzhang666/article/details/9468605)

1. **class** Person{
2. **private** String name;
3. **private** **int** age;
5. **public** **static** **void** sop(Object o){
6. System.out.println(o);
7. }
9. **public** Person(String name, **int** age){
10. **this**.name =name;
11. **this**.age =age;
12. }
14. **public** **void** setName(String name){
15. **this**.name =name;
16. }
18. **public** String getName(){
19. **return** **this**.name;
20. }
22. **public** **void** setAge(**int** age){
23. **this**.age =age;
24. }
26. **public** **int** getAge(){
27. **return** **this**.age;
28. }
30. **public** String toString(){
31. **return** **this**.name+"::"+**this**.age;
32. }
34. //equals已经重写
35. **public** **boolean** equals(Object obj){
36. **if**(!(obj **instanceof** Person)){
37. **return** **false**;
38. }
40. Person p =(Person)obj;
42. //用来查看equals方法是否被调用
43. sop(**this**.name +".......equals......."+ p.name);
44. //认为名字相同并且年龄一样大的两个对象是一个
45. **return** **this**.name.equals(p.name) && **this**.age == p.age;
46. }
47. }

(2). 将hashCode()重写成相同的值-----解决HashSet中重复添加

[1]. 问题：**内容相同**但是**地址不同**的**自定义对象如何避免重复的内容**添加到HashSet中？

【解决办法】**必须重写hashCode**和**equals**这**两个**方法

{1}. 此时根据**底层哈希表**的存储方式：哈希表会将具有相同哈希值的元素依次顺延。

{2}. hashCode值相同，HashSet在存储对象的时候，**equals方法就会起作用**

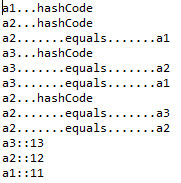
{3}. 示例代码：为Person类重写如下的hashCode代码

**[java]** [view plaincopy](http://blog.csdn.net/benjaminzhang666/article/details/9468605)

1. **public** **int** hashCode(){
2. System.out.println(**this**.name +"...hashCode");
3. **return** 60;
4. }

这样，每一个Person对象都具有相同的哈希值。

打印结果：

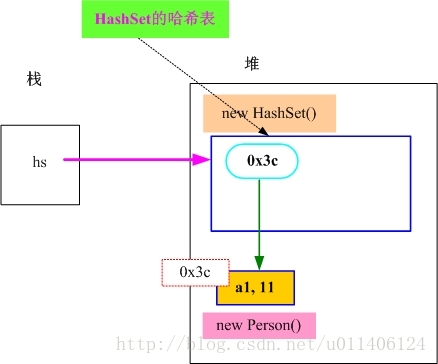


【**HashSet添加过程**分析】

【1】a1率先存入HashSet中

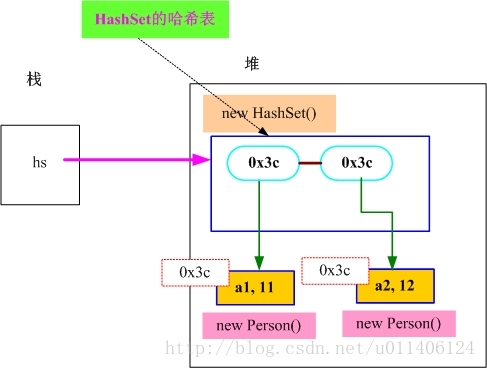
添加的时候，调用一次a1的hashCode方法，打印一次a1...hashCode。由于开始HashSet中没有内容，所以没有调用a1的equals方法。

内存图如下：



【2】当a2要存入HashSet的时候，HashSet首先调用a2的hashCode方法，此时打印出a2...hashCode。发现a2的hashCode也是0x3c和a1的**地址值都是一样的**，此时要和a1进行内容比较，打印出a2.......equals.......a1。但是发现name和age都不一样，所以equals返回false。此时HashSet就将这个a2存到集合中来。

内存图如下：



【3】当a3要存入HashSet的时候，HashSet首先调用a3的hashCode方法，查看有没有地址相同的元素，此时打印出a3...hashCode。此时集合中已经存在两个元素a1和a2，发现a3的hashCode也是0x3c和a1、a2的**地址值都是一样的**，此时要一一和这些对象的内容进行比较。

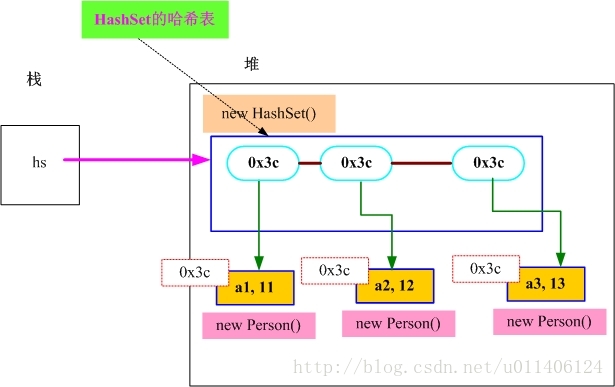
       当a3和a2的进行比较时，a3的equals方法被调用，打印出a3.......equals.......a2。 比较发现a3和a2是地址相同但是内容不同的元素。

    a3再和a1进行比较，a3的equals方法再次被调用，打印出a3.......equals.......a1。

比较发现a3和a1仍然是地址相同但是内容不同的元素。

a3就被认为是集合中以前并不存在的元素，所以仍然被添加进来。

内存图如下：



【4】当运行到第二个a2要添加进来的时候，先调用hashCode，所以马上打印a2...hashCode。

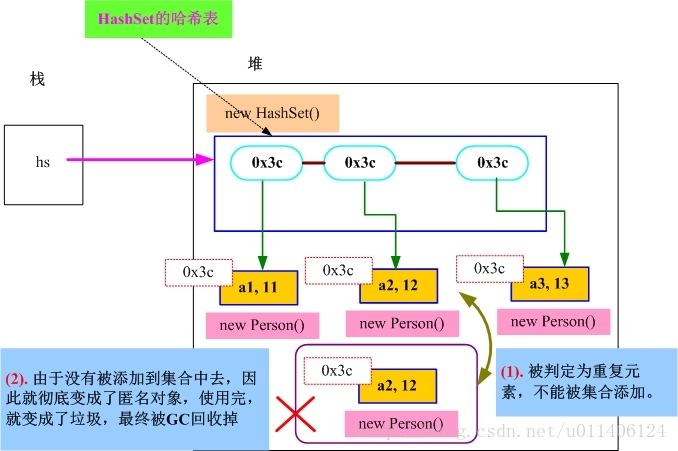
但是发现，和a2 hashCode相同的元素有a1、a2和a3。这个a2要和集合中的a1、a2和a3都做内容上是否相同的比较。

    a2先比较集合中的a3，调用a2的equals方法，打印出a2.......equals.......a3。

位置相同，但是内容不同。

    a2再比较集合中的a2，调用a2的equals方法，打印出a2.......equals.......a2。

但是发现**两者内容、地址均相同**，是**重复的元素**，**不能加到集合中来**，所以没有必要再把这个a2和集合中的a1进行比较。所以**没有打印**a2.......equals.......a1



(3). 正确重写hashCode的办法

[1]. hashCode重写成**相同的值**的**缺点**

将所有对象的hashCode都返回一样的值是不科学的。比如a1和a3这两个根本不同的对象，就没有必要去比较equals，**增加无谓的计算量**。**所以应该对象本身的内容 (属性)来重写hashCode。**

一旦两个**对象内部**不一样，就**直接判定出hashCode不一样**，**不用再调用equals进行比较**。

[2]. 正确书写hashCode的办法：

【原则】按照**equals( )**中**比较两个对象是否一致**的**条件用到的属性**来**重写hashCode()**。

{1}. 常用的办法就是利用**涉及到的的属性**进行**线性组合**。

{2}. **线性组合**过程中涉及到的**组合系数**自定义即可。

注意，拼接之后的数值不能超过整形的表达范围。

{3}. 公式：**属性1的int形式**+ C1\***属性2的int形式**+  C2\***属性3的int形式**+ …

【技巧】当属性是**引用**类型的时候，如果**已经重写过hashCode()**，那么这个引用属性的int形式就是直接调用属性已有的hashCode值。

**最典型**的就是这个属性是**字符串类型**的，String类型已经重写了hashCode()方法，所以直接拿来使用即可。

e.g. 分析案例

这个例子中，重写的equals方法中是通过name和age来判定两个对象是否一致的。所以，就通过Person的这两个属性name和age的线性组合来获取这个Person的hashCode值。注意到name是String类型的，所以，可以调用name的hashCode()来直接获取name对应的int值。

这里重写的hashCode方法是：

**public int** hashCode(){

*sop*(**this**.name +"......hashCode");

**return this**.name.hashCode() + 29\*age;

}

打印结果：



分享：