

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

Take the 2-minute tour ✕

Example of ==, equals and hashCode in java

Given this:

```
String s1= new String("abc");
String s2= new String("abc");
String s3 ="abc";
System.out.println(s1==s3);
System.out.println(s1==s2);
System.out.println(s1.equals(s2));
System.out.println(s1.equals(s3));
System.out.println(s1.hashCode());
System.out.println(s2.hashCode());
System.out.println(s3.hashCode());
```

Output is:

```
false
false
true
true
96354
96354
96354
```

Here == is giving false for each object but the hashCode for each String object is same.
Why is it so?

java equals hashCode

edited Aug 13 '14 at 1:19

 MichaelT
1,489 3 12 29

asked Apr 28 '10 at 17:47

 Abhishek Jain
1,088 4 13 27

4 Answers

== does compare real equality of **objects** (I mean - both references point to the same object), not their content, whereas .equals compares content (at least for String).

```
String a = new String("aa");
String b = new String("aa");
```

a and b are pointing to different objects.

Notice also that if objects are equal then their hashcodes must be the same, but if hashcodes are the same, it doesn't mean that objects are equal.

edited Aug 13 '14 at 1:26

 MichaelT
1,489 3 12 29

answered Apr 28 '10 at 17:50

 Mirek Pluta
3,601 19 18

The equals contract says that if `o1.equals(o2)`, then `o1.hashCode() == o2.hashCode()`. It doesn't specify anything about the hash codes of *unequal* objects. You could have a method like

```
public int hashCode()
{
    return 42;
}
```

and it'd fulfill the contract. It's just expected that the hash code be related to the value of the object, in order to make hash tables work more efficiently.


Now, as for why your == doesn't work, two objects will always be compared by reference.

That is, if `o1 == o2`, then `o1` and `o2` are the exact same object. That's rarely what you want; you usually want to see if `o1.equals(o2)` instead.

edited Aug 13 '14 at 2:33

answered Apr 28 '10 at 17:53

 cHao

 44.8k 8 57 103

+1 for actually answering the question that was asked – Sean Owen Apr 28 '10 at 18:19

When you use `==`, you are comparing if two variables hold reference to the same Object. In other words `s1 == s2` is like asking: are the `s1` and `s2` variables referring to the same String object? And that's not true, even when both String objects have the same "abc" value.

When you use `equals()`, you are comparing the value of both objects. Both objects may not be the same, but their value (in this case "abc") is the same, so it returns `true`.

How do you define whether an object is equal to another? That's up to you. In this case the String object already defines this for you, but for example if you define a `Person` object, how do you know if a person P1 is equal to P2? You do that by overriding `equals()` and `hashCode()`.

answered Apr 28 '10 at 17:56

 Cesar

4,699 2 11 28

`==` tells you whether the two variable references point at the same object in memory, nothing more. `equals()` and `hashCode()` both look at the contents of the object and each uses its own algorithm for calculation.

answered Apr 28 '10 at 17:52

 Jim Kiley

2,772 3 15 30