## Overriding the equals method vs creating a new method

Asked 16 years, 4 months ago Modified 7 years, 7 months ago Viewed 2k times



21





I have always thought that the .equals() method in java should be overridden to be made specific to the class you have created. In other words to look for equivalence of two different instances rather than two references to the same instance. However I have encountered other programmers who seem to think that the default object behavior should be left alone and a new method created for testing equivalence of two objects of the same class.

What are the argument for and against overriding the equals method?

java oop

Share

Improve this question

Follow

edited Apr 27, 2017 at 18:00



asked Aug 19, 2008 at 17:05



## 7 Answers

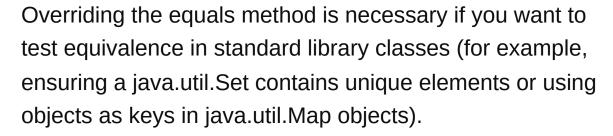
Sorted by:

Highest score (default)



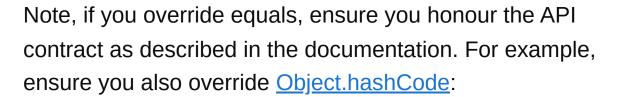


19











If two objects are equal according to the equals(Object) method, then calling the hashCode method on each of the two objects must produce the same integer result.

EDIT: I didn't post this as a complete answer on the subject, so I'll echo Fredrik Kalseth's statement that overriding equals works best for <u>immutable objects</u>. To quote the API for <u>Map</u>:

Note: great care must be exercised if mutable objects are used as map keys. The behavior of a map is not specified if the value of an object is changed in a manner that affects equals comparisons while the object is a key in the map.



The Effective Java (Josh Bloch; 1st ed.) chapter on java.lang.Object methods:

java.sun.com/developer/Books/effectivejava/Chapter3.pdf

- McDowell Apr 30, 2009 at 16:51



8



1

I would highly recommend picking up a copy of Effective Java and reading through item 7 obeying the <u>equals</u> <u>contract</u>. You need to be careful if you are overriding equals for mutable objects, as many of the collections such as Maps and Sets use equals to determine equivalence, and mutating an object contained in a collection could lead to unexpected results. Brian Goetz also has a pretty good <u>overview of implementing equals and hashCode</u>.

Share Improve this answer

edited Aug 19, 2008 at 17:26

Follow

answered Aug 19, 2008 at 17:21



David Schlosnagle
4.323 • 2 • 24 • 16



You should "never" override equals & getHashCode for mutable objects - this goes for .net and Java both. If you do, and use such an object as the key in f.ex a dictionary

4

and then *change* that object, you'll be in trouble because the dictionary relies on the hashcode to find the object.



Here's a good article on the topic:

http://weblogs.asp.net/bleroy/archive/2004/12/15/316601.



<u>aspx</u>

Share Improve this answer Follow

answered Aug 19, 2008 at 17:28





@David Schlosnagle <u>mentions</u> mentions Josh Bloch's <u>Effective Java</u> -- this is a **must-read** for any Java developer.



There is a related issue: for immutable value objects, you should also consider overriding <code>compare\_to</code>. The standard wording for if they differ is in the <code>Comparable</code> API:



**1** 

It is generally the case, but not strictly required that (compare(x, y)==0) == (x.equals(y)). Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

Share Improve this answer Follow

edited May 23, 2017 at 11:46



answered Aug 19, 2008 at 20:46





The Equals method is intended to compare references. So it should not be overriden to change its behaviour.





You should create a new method to test for equivalence in different instances if you need to (or use the CompareTo method in some .NET classes)





Share Improve this answer **Follow** 

answered Aug 19, 2008 at 17:08



juan

**81.8k** • 52 • 164 • 198



To be honest, in Java there is not really an argument against overriding *equals*. If you need to compare instances for equality, then that is what you do.









As mentioned above, you need to be aware of the contract with hashCode, and similarly, watch out for the gotchas around the <u>Comparable</u> interface - in almost all situations you want the *natural ordering* as defined by Comparable to be consistent with equals (see the BigDecimal api doc for the canonical counter example)

Creating a new method for deciding equality, quite apart from not working with the existing library classes, flies in the face of Java convention somewhat.

Share Improve this answer Follow

edited Aug 19, 2008 at 21:12

answered Aug 19, 2008 at 19:50



serg10

**32.6k** • 16 • 75 • 94



You should only need to override the equals() method if you want specific behaviour when adding objects to sorted data structures (SortedSet etc.)



When you do that you should also override hashcode().



See <u>here</u> for a complete explanation.



Share Improve this answer Follow

edited Dec 15, 2014 at 17:04



naXa stands with Ukraine

**37.7k** • 24 • 202 • 272

answered Aug 19, 2008 at 17:15



Michael Sharek

**5,069** • 2 • 31 • 33