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

Cœur

38.6k ● 26 ● 202 ● 276

asked Aug 19, 2008 at 17:05

N8g

636 ● 2 ● 8 ● 19

7 Answers

Sorted by: Highest score (default)

19

Overriding the equals method is necessary if you want to test equivalence in standard library classes (for example, ensuring a java.util.Set contains unique elements or using objects as keys in java.util.Map objects).

Note, if you override equals, ensure you honour the API contract as described in the documentation. For example, ensure you also override Object.hashCode:

> 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 immutable objects. To quote the API for Map:

> 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.

Share Improve this answer Follow

edited Aug 19, 2008 at 17:39

answered Aug 19, 2008 at 17:14

McDowell
109k ● 31 ● 206 ● 270

2 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

I would highly recommend picking up a copy of Effective Java and reading through item 7 obeying the equals contract. 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 overview of implementing equals and hashCode.

Share Improve this answer Follow

edited Aug 19, 2008 at 17:26

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.aspx

Share Improve this answer Follow

answered Aug 19, 2008 at 17:28

Fredrik Kalseth
14.2k ● 4 ● 26 ● 18

2

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

There is a related issue: for immutable value objects, you should also consider overriding `compare_to`. The standard wording for if they differ is in the Comparable API:

> 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

Community Bot
1 ● 1

answered Aug 19, 2008 at 20:46
James A. Rosen
65.2k ● 62 ● 184 ● 263

0

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

0

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 Comparable 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

0

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 here 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