hashCode And equals Methods Override

In this Java tutorial, we will discuss about hashCode, equals methods and what role they play in an object. Through this article we will find answer for the following questions,

- **What** is the purpose of hashCode and equals methods?
- **How** hashCode and equals are implemented?
- **Why** hashCode should also be overridden when equals is overridden?

equals Method

equals() method is used to determine the equality of two objects. To bring in little bit of Mathematics flavor lets see the properties of equality. When we say equality, it should adhere by the following properties,

- Reflexive: Always, a = a. In Java, *a.equals(a)* should always be true.
- Symmetric: If a = b, then b = a. In Java, if *a.equals(b)* is true, then *b.equals(a)* should be true.
- Transitive: If a = b and b = c, then a = c. In Java, if *a.equals(b)* and *b.equals(c)* is true, then *a.equals(c)* should be true.

Above three points are some theory and do not worry about it much.

For a primitive type, determining the equality is simple. We all know an int value of 10 is always equal to 10. But this equals() method is about equality of two objects. When we say object, it will have properties. To decide about equality those properties are considered. It is not necessary that all

properties must be taken into account to determine the equality and with respect to the class definition and context it can be decided. Then the equals() method can be overridden.

Override equals Method

Let us take an example scenario to understand equals() method and study how to override it in our custom implementation. We have a Tiger class and two instances of Tiger are equal if they have the same color and stripePattern is our definition of equality. In our example, though there is an additional property 'height', it is excluded from the equals definition to denote that it is entirely our choice.

```java
package com.javapapers.java;

public class Tiger {
    private String color;
    private String stripePattern;
    private int height;

    @Override
    public boolean equals(Object object) {
        boolean result = false;
        if (object == null || object.getClass() != getClass()) {
            result = false;
        } else {
            Tiger tiger = (Tiger) object;
            if (this.color == tiger.getColor()
                    && this.stripePattern == tiger.getStripePattern()) {
                result = true;
            }
        }
        return result;
    }

    // just omitted null checks
    @Override
    public int hashCode() {
        int hash = 3;
        hash = 7 * hash + this.color.hashCode();
        hash = 7 * hash + this.stripePattern.hashCode();
        return hash;
    }
```

Example Code Output:

bengalTiger1 and bengalTiger2: true

bengalTiger1 and siberianTiger: false

bengalTiger1 hashCode: 1398212510

bengalTiger2 hashCode: 1398212510

siberianTiger hashCode: –1227465966

hashCode Method

In the above example did you notice we have overridden another method hashCode() along with equals()? There is an importance to it. We must override hashCode() when we override equals(). Why hashCode should also be overridden when equals is overridden? Because, they both serve the same purpose but in different contexts.

hashCode() method is used in hashtables to determine the equality of keys. If you want more details about hashCode, their default implementation, buckets, different hashing techniques you should go through my earliest tutorial on Java Hashtable. I guarantee you will enjoy, just read it :-).

When an application is executed, the hashcode (an integer) returned for an object should be same till another execution of that application. Now coming to the important point which is **the contract between hashCode and equals method**,

- if two objects are equal, that is obj1.equals(obj2) is true then, obj1.hashCode() and obj2.hashCode() must return same integer.

Override hashCode Method

To honor the above contract we should always override hashCode() method whenever we override equals() method. If not, what will happen? If we use hashtables in our application, it will not behave

as expected. As the hashCode is used in determining the equality of values stored, it will not return the right corresponding value for a key.

Default implementation given is hashCode() method in Object class uses the internal address of the object and converts it into integer and returns it. This is the lowest form of equality implementation and provides guaranteed results for hastables implementation. When we override equals() and change the meaning of equality for an object then the same should be reflected by overriding the hashCode method.

May be you should also go through String equals also at this juncture as String equality is one of the celebrated topic in Java.

This Core Java tutorial was added on 16/06/2013.

PREVIOUS
Show Map in Android

NEXT
Iterator Design Pattern

Comments on "hashCode And equals Methods Override" Tutorial:

Shivang Gupta says: *17/06/2013 at 2:56 am*

I think there's a typo here. Shouldn't it be this.color.equals(tiger.getColor()) and this.stripePattern.equals(tiger.getStripePattern()) ?

deepak says: *17/06/2013 at 11:50 am*

Hi, I have 1 question, if equals method says two object are equal so if in any class I implement equals method and hashcode method to always return same value, will that class be singleton?

Please confirm.

Thanks
Deepak

Rajkumar says: *17/06/2013 at 12:00 pm*

very nice. Keep posting.

Mohamed Galala says: *17/06/2013 at 12:24 pm*

Yes, exactly in the implementation example of overriding the Equals method.

Prasad says: *17/06/2013 at 1:35 pm*

Good article. Thank You!

Sushil says: *17/06/2013 at 2:08 pm*

I don't think so…

as contract says if two objects are equal(), their hashcode() should also be same, but reverse is not true.

Their could be 2 objects having same hashcode but they are not equal.

Please anybody can further explain this.

Danny says: *17/06/2013 at 2:16 pm*

Override…..

Muralidhar N says: *17/06/2013 at 2:34 pm*

Hi Joe, nice explanation with Bengal tigers. found a typo mistake here:"guaranteed results for hastables implementation."

Thank you.
Muralidhar N

Karthik says: *17/06/2013 at 8:19 pm*

Nice :-)

---

sachin anthwal says: *20/06/2013 at 11:41 pm*

Please elaborate this equals() and hashCode()

Tarun says: *21/06/2013 at 9:38 am*

thanks very nice post it clear all my doubt

---

pius ndugo says: *21/06/2013 at 1:43 pm*

great stuff !

Ragav says: *26/06/2013 at 6:02 pm*

Very Nice… :) .. Keep rocking..

Back To Java says: *29/06/2013 at 6:35 pm*

Hello Joe,

Thank you very much I understood very clearly, few of my interviews I have been asked the same question but could not able to answer, Now I am clear on this. Thanks again, hoping more posts from yourend.

---

sasi says: *30/06/2013 at 11:56 am*

very nice…

Vish says: *03/07/2013 at 5:16 pm*

Nice article

Nikhil says: *12/07/2013 at 10:15 am*

Hi Joe,
If i override hashcode(), is it mandatory to override equals() and why or why not?

Ankit Tripathi says: *15/07/2013 at 3:26 pm*

Whenever a null is passed as argument to equals, it must return false. Ex : bengalTiger1.equals(null) must return false.
Though you mentioned this in your code.
I think, you could have added this line while mentioning the properties of equals method.

Shradha says: *18/07/2013 at 12:16 pm*

Hey Joe..
very nice article ..it clears all my doubts

---

javalearner says: *20/08/2013 at 7:09 pm*

Thank you so much for a simple write up which tells a lot!!

Nitya says: *24/08/2013 at 3:00 pm*

What will happen if we will override hashcode and wont override equals?

Bala says: *31/08/2013 at 8:47 pm*

Dear Joe,

I tried above example program its working properly.
i have query on that.
i removed the hashcode() now also got same result
then what is the purpose of override hashCode() above program
please help me out on this doubt.

Avinash Y N says: *03/09/2013 at 12:30 pm*

Hi Bala,
If you really want to experience the power of hashcode() method then, the best example would be to store the Tiger Objects as a key in hashMap Collection.

Eg: map.put(<>, “Bengal Tiger 1”);
map.put(<>, “Bengal Tiger 2”);

Now try to retrieve the value of <> from map. It will return null if you are not overriding hashCode. Then try the same exercise by implementing hashCode method. You will get the proper value.

Basically hashCode is a slot through which the compiler understands where to search for a particular object. If there is no hashCode then the Compiler will not know where to look for the object and hence returns null.

Avinash Y N says: *03/09/2013 at 12:33 pm*

In above comment means tiger Object.

Avinash Y N says: *03/09/2013 at 12:41 pm*

Hi Sushil,
You might have got answer to this by now.

Consider a simple example of 2 names say MALA and LAMA.

Let's define hashCode() as A=1, B=2 etc.
So here both MALA and LAMA have hashCode of 27.
Here hashCode is equal for both Names but actually they are not same names. Hence it is not necessary that even if hashCode are equal, Objects should be equal.

Sandeep Swaminathan says: *04/09/2013 at 8:19 pm*

Nice article. Thanks. On a lighter side the example given here is logically wrong. Stripes on each tiger are unique. So your equals method will never be true :P Think like a programmer. Always logic :D

Rajasekhar says: *20/09/2013 at 10:17 am*

Scenario 1: If two objects are equal by equals(), then those two objects of hash code must be same.

Scenario 2: If two objects are not equal by equals(),then those two objects of hash code may or may not same.

Scenario 3: If two objects of hash code is same then those two objects are may or may not equal.

Scenario 4: If two objects of hash code is not same,then those two objects must be different.

Note: If am wrong in above case,please correct me.Thanks

---

kuppumani says: *05/10/2013 at 10:01 pm*

good

kumar says: *09/10/2013 at 5:28 pm*

can you please explain the logic in hashCode

why the 3 and 7 values
int hash = 3;
hash = 7 * hash + this.color.hashCode();

Joe says: *09/10/2013 at 6:06 pm*

No special meaning to numerals 3 and 7. They are there just to show that, we can add our own logic there to calculate hash.

Mohamed says: *09/10/2013 at 7:36 pm*

Hats off man….!!!.. well taght :)

Lathish says: *09/10/2013 at 9:26 pm*

very well explained…found the right place for java doubts.. :)

Jaywant Patil says: *11/10/2013 at 3:03 pm*

JOE,
Thanks, all concepts are very well explained…
I always refer this site for any java related problems.

Regards,
Jaywant PATIL(Software Analyst)

Arvind says: *12/10/2013 at 1:10 pm*

bengalTiger1.equals(null) will not return false. it will throw NullpointerException.

Anonymous says: *15/10/2013 at 4:53 pm*

hii i just wanted to know the contact's name pls let me know ,

thanks
subha

Prem Kumar says: *16/10/2013 at 2:14 pm*

Good article. Why are you keeping the tiger images on the blog. It is funny.

Joe says: *16/10/2013 at 2:25 pm*

Thanks Prem.

I love animals. It has no significance with the technology and topic. But the code example has reference to siberian and bengal tiger. Just added this for fun, nothing official about it :-)

Joe says: *16/10/2013 at 2:26 pm*

Thanks Jaywant. Enjoy!

Ashwani says: *17/10/2013 at 10:33 pm*

Nice JOE, very nice example. Its easy to understand….Thanks.

Ashwani says: *17/10/2013 at 10:47 pm*

JOE, I have doubt – Logic made by us in overridden Hashcode() method can also generate similar hashcode value for 2 or more different objects. What will we do in that case??

---

chander says: *18/10/2013 at 10:42 pm*

Easy to Understand, Thanks JOE.

---

kailashdas says: *12/11/2013 at 2:41 pm*

if u create two object of usedefined class. implement hashcode() . u will get two object whose equla() gives false , but hashcode() gives true.

---

skynafo says: *21/11/2013 at 6:33 pm*

Thank you so much dude :)

Shivdas Yeske says: *25/12/2013 at 7:11 am*

Tried above example with hashmap as you suggested but getting value from hashmap whether I implement hashcode() or not.

---

chandan mishra says: *09/01/2014 at 2:15 pm*

how to write overridden hashcode method

Rahul Kamuni says: *10/01/2014 at 2:35 pm*

Superb explanation. Now I got it why we need to overide() equals and hashcode() method.

---

raekwon says: *20/01/2014 at 1:10 am*

Great explanation on this topic. Thanks for this awesome site and Keep up the good work!

Moiz says: *13/02/2014 at 12:16 am*

Hi deepak,

According to the singleton concept, all the object references returned by a singleton class points to one and the same object i.e. there hashcode must be same and all there reference comparison(by == operator) should result in TRUE boolean value.

For ex: A and B are the reference to the object returned by a singleton class then it should satisfy the following conditions:
1) a.hashCode() == b.hashCode() or a.equals(b)
2) a == b

---

allen.ngorora@gmail.com says: *16/02/2014 at 9:17 pm*

Thanks it helps

Yuvaraj Mani says: *02/03/2014 at 12:21 pm*

Came across your blog while writing some code. Very nice collection of java helps. Thanks for putting this blog up.

---

prasune john says: *22/03/2014 at 5:30 pm*

Guys, if you understand the reason behind having the contract for hashcode to be same for two objects which satisfies equals, it would be pretty clear. A collection like HashMap which uses hashing algoritm stores the object in a specific fashion. i.e, all objects with same hashcode is stored in the same index in a linkedlist. So, for retreiving the object, the hashcode is used to find the index. If the contract fails (hash code is not equal for objects that are equal) then search method of HashMap fails.

---

madhusudhan says: *25/03/2014 at 1:55 pm*

```
public int hashCode() {
int hash = 3;
hash = 7 * hash + this.color.hashCode();
hash = 7 * hash + this.stripePattern.hashCode();
return hash;
}
```

why hash value is assigned to times(color.hashcodes and stripepattern.hashcode) anyway the resultnat hash value(return hash) will return only one hash value..

please explain

Anonymous says: *05/04/2014 at 2:03 am*

Very good explaination.

Sandip says: *09/04/2014 at 12:32 pm*

suppose if we return same value each time from hascode() then what will happen?

---

amarshi says: *24/04/2014 at 5:35 pm*

Map map = new HashMap();
map.put(bengalTiger1, “Tiger-1″);
map.put(bengalTiger1, “Tiger-2″);
System.out.println(“bengalTiger1 and bengalTiger2: “+ bengalTiger1.equals(bengalTiger2));

this still returns true??????

Sri Harrsha says: *27/04/2014 at 4:47 pm*

It does return only one value as you know, but the function is just trying to generate a unique value using both of the field’s Hash Code value,Instead of depending on only one’s hashCode Value .

Shiv says: *03/06/2014 at 8:05 pm*

Hi,
Consider the below code.
//Created Tiger class with only Height as member variable. Overridden only equals().
public class Tiger
{
int height = 1;
public Tiger(int height)
{
this.height = height;
}

```
@Override
public boolean equals(Object obj)
{
Tiger other = (Tiger) obj;
if (height != other.height)
return false;
return true;
}
}
```

```
//In the main class: Created 2 tiger objects and added them to HashMap.
Tiger t1 = new Tiger(1);
Tiger t2 = new Tiger(2);

Map tigerMap = new HashMap();
tigerMap.put(t1, "1st Tiger");
tigerMap.put(t2, "2nd Tiger");
```

In the above code, both Tiger1 and Tiger2 will return the same hashcode, but have different values.
So for the same hashcode, both tigers will be stored.
Retrieving these objects will also work fine.

In this scenario, I don't have to override hashcode().. right??

Vinita says: *15/06/2014 at 5:47 pm*

Hi Joe,
I have read many articles about equals & hashCode and this is the best. Simple and Excellent description.
(though some more points could have been added, like is it mandatory to override equals if hashCode is overridden, etc)
The tiger pictures is the best part. I will always remember these tigers whenever this concepts comes into my mind :)

Thanks!

---

srikanth says: *22/06/2014 at 1:50 pm*

My question is in which scenario both the methods will ovverride in strings or collections or some other area ??

Anonymous says: *25/06/2014 at 5:08 pm*

superbbbb……

Anonymous says: *25/06/2014 at 5:09 pm*

no yeah, if u delete all the lines from the code. it will work… otherwise shutdown the system and watch movie on TV…

Anonymous says: *25/06/2014 at 5:11 pm*

ArrayList list=new ArrayList();
list.add(“Tiger”);
System.out.println(list);

Output will be print as Tiger. Is it correct Bala….

MICHELLE says: *25/06/2014 at 8:02 pm*

hi all

in the article it seems like even though we not override a hashcode method for the class then while

adding its object in to any collection
use the class Object hashcode method.

so Object hashcode method also give unique value and unlike the hashcode which we override to generate hashcode based on the alphabetial order of string for ex:MAY=hashcode'42' and AMY=hashcod:'42'.

it is fair to use Object's hashcode() to avoid collision rather than overriding it in our own way?

---

tribhuvan says: *17/07/2014 at 6:19 pm*

@Arvind It will give as false not NPE

Anonymous says: *13/08/2014 at 12:29 am*

how come Tiger1 and Tiger2 returns same hashcode?

Md Farooq says: *18/08/2014 at 11:00 pm*

awesome explanation, i am very much thankfull for this article, keep posting

---

Alwin Jose says: *27/08/2014 at 3:30 pm*

hascode() method will return different values.

Anonymous says: *27/08/2014 at 4:32 pm*

Are you meant the map's key values as the reference of the Tiger objects

---

Alwin Jose M says: *27/08/2014 at 4:38 pm*

3 and 7 represent prime numbers. Instead of that, we can use any prime numbers like 2, 3, 5, 7, 11, 13, 17 etc.,,

---

Alwin Jose M says: *27/08/2014 at 4:42 pm*

I got good idea about equals and hashCode. Thanks all

Why Object is Super Class in Java? says: *04/09/2014 at 11:27 pm*

[…] for all classes and have some list of functions same among them. I am referring to methods like hashCode(), clone(), toString() and methods for threading which is defined in Object […]

Manjunath says: *12/09/2014 at 6:09 am*

In a singleton scenario we have no reason to override equals.. This will return the same instance as u can't create a new from ur class. U should just access the instance created.

Giridhar says: *18/09/2014 at 2:42 pm*

Nice explanation, I got good idea about equals() and hashcode() method. Thanks !

Pankaj says: *21/10/2014 at 10:32 am*

Well Joe, I have been following your articles and they are excellent ones no doubt. But for this example, it is not giving the desired output when I don't override hashcode() method.

So, please can you add a few lines in your above code and show how hashmap would return null if I

don’t override hashcode() method.

Thanks Pankaj

Comments are closed for this "hashCode And equals Methods Override" tutorial.

↑ Go to top

Java

Android

Design Patterns

Hibernate

Spring

Web Services

Servlet

*JavaPapers is a tutorials site and **Joe** runs it passionately.*

Say hi: joe@javapapers.com

Recommended Tutorials

- Java History
- Overloading Vs Overriding In Java
- Eclipse Shortcuts
- Java Serialization
- Difference Between Interface And Abstract Class
- Java Hashtable
- Difference Between Forward And SendRedirect
- Differentiate JVM JRE JDK JIT
- Java (JVM) Memory Types
- Why Multiple Inheritance Is Not Supported In Java