

JavaMadeSoEasy.com (JMSE)

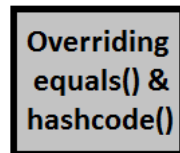
[Home](#)
[Core Java Tutorials](#)
[Interview Questions](#)
[Interview Programs](#)
[Custom](#)
[Quiz/OCJP](#)
[MS EXCEL](#)
[Database](#)
[I](#)

- [Algorithm](#)
- [Apache Tomcat Server](#)
- [Collection Framework](#)
- [Collection programs](#)
- [Collections implementation](#)
- [Core Java](#)
- [Core Java Differences](#)
- [Core Java Tutorials](#)
- [Custom implementation of Data Structures](#)
- [Data Structure](#)
- [Date tutorials](#)
- [eclipse](#)
- [Exceptions](#)
- [File IO/File handling](#)
- [Garbage collection in java](#)
- [Generics](#)
- [Hashing](#)
- [Interview questions](#)
- [jText Pdf tutorial](#)
- [Java 7](#)
- [Java 8](#)
- [Java 9](#)
- [Java QUIZ](#)
- [JavaScript](#)
- [JDBC](#)
- [JSON](#)
- [JUNIT](#)
- [JVM](#)
- [Level1 programs \(beginners\)](#)
- [Level2 programs \(intermediate\)](#)
- [Level3 programs \(advanced\)](#)
- [Linked List](#)
- [Linux](#)
- [Matrix programs](#)
- [Maven](#)
- [MongoDB](#)
- [MongoDB Java](#)
- [MS Excel](#)

Search This Blog

Overriding equals and hashCode method - Top 18 Interview questions and answers in java for experienced and freshers

You are here : [Home](#) / [Core Java Tutorials](#) / [Java Interview Questions and answers](#)



Interviewers always have great focus on checking interviewees in depth knowledge about overriding equals and hashCode() related question. So, I have tried to cover most of the possible question which could be framed related to this topic.

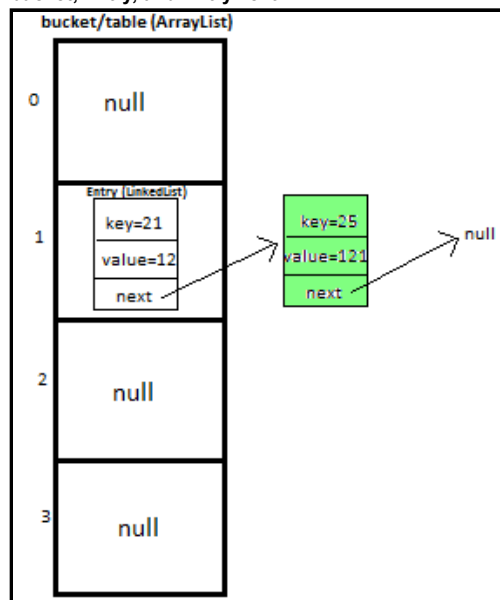
Question1. Why do we need to override equals and hashCode method?

Answer. Equals() and hashCode() are methods of java.lang.Object

It's important to override equals equals() and hashCode() method of class if we are want to use class as key in HashMap.

If we don't override equals() and hashCode() method we will be able to put object, but we won't be able to retrieve object.

Before understanding the concept of overriding equals() and hashCode() method, we must understand what is **bucket, Entry, and Entry.next**



Bucket is [ArrayList](#) of Entry.

- [MsSql](#)
- [MultiThreading](#)
- [MySql](#)
- [Notepad++](#)
- [Oracle](#)
- [OutOfMemoryError](#)
- [Overriding equals and hashCode](#)
- [PostgreSQL](#)
- [Producer Consumer problem/pattern](#)
- [Pyramid generation](#)
- [Regex](#)
- [Serialization](#)
- [Thread Concurrency](#)
- [Threads](#)
- [TUTORIALS](#)
- [UNIX](#)
- [Windows](#)

Join our Groups>

- [FACEBOOK](#)
- [LINKED IN](#)



Entry is [LinkedList](#) which contains information about key, value and next.

Entry.next points to next **Entry** in **LinkedList**.

>Why to override hashCode method?

It helps in finding bucket location, where entry(with key-value pair) will be stored .

Entry (of type **LinkedList**) is stored in **bucket (ArrayList)**.

If, **hashCode()** method is overridden properly, we will find bucket location using **hashCode()** method, we will obtain **Entry** on that bucket location, then iterate over each and every **Entry** (by calling **Entry.next**) and check whether new and existing keys are equal or not. If keys are equal replace key-value in **Entry** with new one, else call **Entry.next** But, now the question comes how to check whether two keys are equal or not. So, it's time to implement **equals()** method.

If, **hashCode** method is not overridden for same key every time **hashCode()** method is called it might produce different **hashCode**, there might happen **2 cases** i.e. when **put** and **get** method are called.

Case 1 : when put() method is called-

There might be possibility that same **Entry** (with key-value pair) will get stored at multiple locations in bucket.

Conclusion> key- value pair may get stored multiple times in **HashMap**.

Case 2 : when get() method is called-

As there is possibility that **hashCode()** method might return different **hashCode** & rather than searching on bucket location where **Entry**(with key) exists we might be searching for key on some other bucket location.

Conclusion> key existed in **HashMap**, but still we were not able to locate the bucket location in which it was stored.

>Why to override equals method?

Once we have located bucket location in which our **Entry** (with key-value pair) will be stored, **Equals** method helps us in finding whether **new and existing keys are equal or not**.

If we **equals** method is not overridden - though we will be able to find out correct bucket location if **hashCode()** method is overridden correctly, but still if **equals** method is not overridden, there might happen **2 cases** i.e. when **put** and **get** method are called.

Case 1 : when put() method is called-

we might end up storing new **Entry** (with new key-value pair) multiple times on same bucket location (because of absence of **equals** method, we don't have any way of comparing key's),

In this case, even if keys are equal, we will keep on calling **Entry.next** until we reach last **Entry** on that bucket location and ultimately we will end up storing new **Entry** (with new key) again in same bucket location.

Conclusion> key- value pair stored multiple times in **HashMap**.

Case 2 : when get() method is called-

we won't be able to compare two keys (new key with existing **Entry**.key) and we will call **Entry.next** and again we won't be able to compare two keys and ultimately when **Entry.next** is null - we will return **false**.

Conclusion> key existed in **HashMap**, but still we were not able to retrieve it.

So, it's important to override **equals** method to check equality of two keys.



See : [HashMap Custom implementation for detailed implementation of equals and hashCode method and for understanding bucket and Entry.](#)

See here for detailed understanding of how to **put, get, remove** **Employee** object in [custom HashMap](#).

Question2. How do we override equals and hashCode method, write a code to use Employee as key in HashMap? (Important)

Answer. We will override **equals()** and **hashCode()** like this -

By overriding **equals()** and **hashCode()** method we could use custom object as key in **HashMap**.

- 1) Check whether obj is null or not.

All Labels

[Algorithm](#)(34) [Apache Tomcat](#)
[Java](#)(30) [Autoboxing](#)(5) [Basics](#)
[beginners](#)(15) [Binary Trees](#)(6)
[Framework](#)(68) [Collections implementation](#)
[Comparator program](#)(22) [C](#)
[Java](#)(1035) [core java](#)
[Conversions](#)(21) [Core Java Tutorials](#)(12) [CRUD MongoDB operations](#) [MongoDB](#)(3) [Custom](#)
[Data Structures](#)(11) [Data Structures](#)
[Database Tutorials](#)(3) [Date tutorial](#)
[DeSerialization](#)(19) [eclipse](#)(
[Exceptions](#)(71) [File IO/File collection in java](#)(43) [General](#)
[Hibernate](#)(1) [Interview question tutorial](#)(45) [Java 4](#)(1) [java 5](#)(3)
[8](#)(80) [Java 9](#)(12) [Java keywords](#)
[MCQ\(Multiple choice questions\)](#)
[Java QUIZ](#)(24) [JavaScript](#)(2
[JSON](#)(27) [JUnit](#)(7) [JVM](#)(22)
[Level1 programs \(beginners programs \(intermediate\)](#)(23'
[\(advanced\)](#)(13) [LinkedList](#)(1
[programs](#)(10) [Maven](#)(6) [MongoDB Java](#)(20) [MS Excel](#)
[MultiThreading](#)(97) [MySQL](#)
[JavaScript \(js\) Engine in Java](#)
[Oracle](#)(18) [OutOfMemoryError and hashCode](#)(11) [Pattern generation](#)(14) [Recursion](#)
[SCJP](#)(26) [Serialization](#)(20) [Interview questions and answers](#)
[Thread Concurrency](#)(35) [Tutorials](#)
[TUTORIALS](#)(22) [UNIX](#)(1) [V](#)

```
if(obj==null) //If obj is null, return without comparing obj & Employee class.
```

2) check whether obj is instance of Employee class or not.

```
if(this.getClass()!=obj.getClass()) //identifies whether obj is instance of Employee class or not.
```

3) Then, type cast obj into employee instance.

```
Employee emp=(Employee)obj; //type cast obj into employee instance.
```

```
@Override
public boolean equals(Object obj){

    if(obj==null)
        return false;

    if(this.getClass()!=obj.getClass())
        return false;

    Employee emp=(Employee)obj;
    return (emp.id==this.id || emp.id.equals(this.id))
        && (emp.name==this.name || emp.name.equals(this.name));

}

@Override
public int hashCode(){
    int hash=(this.id==null ? 0: this.id.hashCode() ) +
        (this.name==null ? 0: this.name.hashCode() );

    return hash;
}
```

Let's say in an organisation there exists an employee with **id=1** and **name='sam'** and **some data** is stored corresponding to him, but if modifications have to be made in data, **previous data must be overridden**.

[DETAILED DESCRIPTION : Override equals\(\) and hashCode\(\) method.](#)



Question3. What classes should i prefer to use a key in HashMap? (Important)

Answer. This question will check your in depth knowledge of Java's Collection Api's. we should prefer **String, Integer, Long, Double, Float, Short and any other wrapper class**. Reason behind using them as a key is that they override equals() and hashCode() method, we need not to write any explicit code for overriding equals() and hashCode() method.

Let's use Integer class as key in HashMap.

```
import java.util.HashMap;
import java.util.Map;

public class StringInMap {
    public static void main(String...a){

        //HashMap's key=Integer class (Integer's api has already overridden hashCode(). and
        equals() method for us )
        Map<Integer, String> hm=new HashMap<Integer, String>();
        hm.put(1, "data");
        hm.put(1, "data OVERRIDDEN");

        System.out.println(hm.get(1));

    }
}
```

Popular Posts of JavaMadeSoEasy

- [Collection Quiz in Java - choice questions](#)
- [Java 8 quiz - MCQ - Multiple choice questions](#)
- [Solve \[ERROR\] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.1:compile \(default-compile\) on project Maven: Compilation failure: diagonal slash supported in -source 1.5 or higher to enable diamond notation](#)
- [How to resolve ERROR: Access denied for user 'root'@'localhost' \(using password: NO\) / \(using java: solution\)](#)
- [Exception handling Quiz](#)
- [Thread/multi threading C++ Multiple choice question](#)
- [vi error - E37: No write mode allowed for ! to override\)](#)
- [error: Class names are not allowed as annotation processing is enabled in java: solution](#)
- [sort Employee on basis of joining Date](#)
- [CORE JAVA - Top 120 important interview questions and answers in core java](#)

```

}
/*OUTPUT

data OVERRIDDEN

*/

```

If, we note above program, what we will see is we didn't override equals() and hashCode() method, but still we were able to store data in HashMap, override data and retrieve data using get method.

>Let's check in **Integer's API**, how Integer class has overridden equals() and hashCode() method :

```

public int hashCode() {
    return value;
}

public boolean equals(Object obj) {
    if (obj instanceof Integer) {
        return value == ((Integer)obj).intValue();
    }
    return false;
}

```

Question4. If two objects have same hashCode, are they always equal?

Answer. No. It's not necessary that object's having same hashCode are always equal. Because same hashCode means object are stored on same bucket location, as key/object in bucket is stored in Entry([Linked List](#)), key/object's might be stored on Entry.next (i.e. on some different entry)

Question5. If two objects equals() method return true, do objects always have same hashCode?

Answer. Yes, two objects can return true only if they are stored on same bucket location. First, hashCode() method must have returned same hashCode for both objects, then on that bucket location's Entry key.equals() is called, which returns true to confirm objects/keys are equal. So, if object's equals return true, they always have same hashCode.

Question6. Can two unequal objects have same hashCode?

Answer. Yes, two unequal objects can have same hashCode.

Question7. What is difference between using instanceof operator and getClass() in equals method?

Answer. If we use instanceof it will return true for comparing current class with its subclass as well, but getClass() will return true only if exactly same class is compared. Comparison with any subclass will return false.

Question8. How we can implement [HashMap](#) for using Integer as key? (Important)

Answer. Here Interviewer tends to check your knowledge of overriding equals and hashCode method, and also how Api's use Integer internally as key in HashMap.

Let's understand in detail how Integer is used as key in [Hashmap](#).

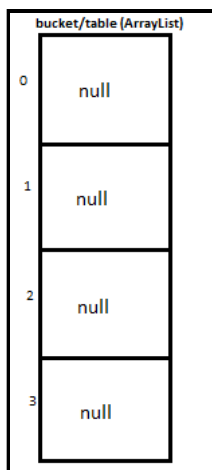
Initially, we have bucket of **capacity=4**. (all indexes of bucket i.e. 0,1,2,3 are pointing to null).

Note: Initial capacity provided by Java Api is different.

JavaMadeSoeasy

- ▶ [2020](#) (1)
- ▶ [2018](#) (114)
- ▶ [2017](#) (211)
- ▶ [2016](#) (240)
- ▶ [2015](#) (722)
 - ▶ [December](#) (61)
 - ▶ [November](#) (96)
 - ▶ [October](#) (14)
 - ▶ [September](#) (13)
 - ▶ [August](#) (73)
 - ▶ [July](#) (54)
 - ▶ [June](#) (29)
 - ▶ [May](#) (53)
 - ▶ [April](#) (109)
 - ▶ [March](#) (93)
 - ▶ [February](#) (99)
 - ▶ [January](#) (28)

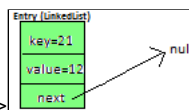




Let's put first key-value pair in HashMap-

Key=21, value=12

newEntry Object will be formed like this >



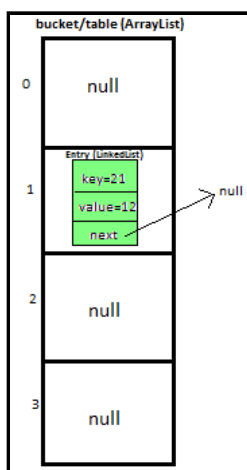
We will calculate hash by using our **hash(K key)** method - in this case it returns

key/capacity= 21%4= 1.

So, 1 will be the **index of bucket** on which **newEntry object** will be stored.

We will go to **1st** index as it is pointing to null we will **put our newEntry object there**.

At completion of this step, our HashMap will look like this-



[DETAILED DESCRIPTION : How Integer is used as key in HashMap](#)

Question9. Implement custom HashMap to put, get, remove Employee object

Answer. See [HashMap Custom implementation - put, get, remove Employee object](#)
[dri bolgEdit](#)

Interviewers sometimes tend to confuse interviewees by framing different question by **overriding either of equals() or hashCode()** method. Interviewers aims to check in **depth knowledge of interviewees** that how many **buckets** are formed, what is **sizeOf HashMap** after each experiment and **get()** method returns object or not (**Important**).

Question10. How many **buckets** will be there and what will be **size of HashMap**?

```
package p1;

import java.util.HashMap;

class Employee {

    private String name;

    public Employee(String name) { // constructor
        this.name = name;
    }

    //no hashCode() method

    //no equals() method

    @Override
    public String toString() {
        return "Employee[ name=" + name + "]" ;
    }

}

/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */
public class Program1 {
    public static void main(String...a){

        HashMap<Employee, String> hm=new HashMap<Employee, String>();
        hm.put(new Employee("a"), "emp1");
        hm.put(new Employee("b"), "emp2");
        hm.put(new Employee("a"), "emp1 OVERRIDDEN");

        System.out.println("HashMap's data> "+hm);
        System.out.println("HashMap's size> "+hm.size());
        System.out.println(hm.get(new Employee("a")));

    }
}
```

Answer.

/*OUTPUT

```
HashMap's data> {Employee[ name=a] =emp1 OVERRIDDEN, Employee[ name=a] =emp1, Employee[ name=b]
=emp2}
HashMap's size> 3
null
```

*/

Buckets= As hashCode() method is not there, hashcode generated for 3 objects will be different and we will end up using **3** buckets.

Size= As equals() method is not their, size will be **3**.

get()=we won't be able to get object.

Question11. How many **buckets** will be there and what will be **size of HashMap**?

```
package p2;
```

```

import java.util.HashMap;

class Employee {

    private String name;

    public Employee(String name) { // constructor
        this.name = name;
    }

    @Override
    public int hashCode(){
        return (this.name==null ? 0: this.name.hashCode() );
    }

    @Override
    public boolean equals(Object obj){
        Employee emp=(Employee)obj;
        return (emp.name==this.name || emp.name.equals(this.name));
    }

    @Override
    public String toString() {
        return "Employee[ name=" + name + " ] ";
    }
}

/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */
public class Program2 {
    public static void main(String...a){

        HashMap<Employee, String> hm=new HashMap<Employee, String>();
        hm.put(new Employee("a"), "emp1");
        hm.put(new Employee("b"), "emp2");
        hm.put(new Employee("a"), "emp1 OVERRIDDEN");

        System.out.println("HashMap's data> "+hm);
        System.out.println("HashMap's size> "+hm.size());
        System.out.println(hm.get(new Employee("a")));

    }
}

```

Answer.

/*OUTPUT

```

HashMap's data> {Employee[ name=b] =emp2, Employee[ name=a] =emp1 OVERRIDDEN}
HashMap's size> 2
emp1 OVERRIDDEN

```

*/

Buckets= As hashCode() method is overridden perfectly, **2** bucket locations will be used.

Size= As equals() method is their, size will be **2**,

value corresponding to Employee with id=1 and name='sam' was employee1 data____
 & was overridden by value employee1 data OVERRIDDEN_

get()=we will be able to get object.

Question12. How many buckets will be there and what will be size of HashMap?

```

package p3;

import java.util.HashMap;

class Employee {

    private String name;

    public Employee(String name) { // constructor
        this.name = name;
    }
}

```

```

    }

    @Override
    public int hashCode(){
        return 1;
    }

    @Override
    public boolean equals(Object obj){
        return true;
    }

    @Override
    public String toString() {
        return "Employee[ name=" + name + "]" ";
    }
}

/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */
public class Program3 {
    public static void main(String...a){

        HashMap<Employee, String> hm=new HashMap<Employee, String>();
        hm.put(new Employee("a"), "emp1");
        hm.put(new Employee("b"), "emp2");
        hm.put(new Employee("a"), "emp1 OVERRIDDEN");

        System.out.println("HashMap's data> "+hm);
        System.out.println("HashMap's size> "+hm.size());
        System.out.println(hm.get(new Employee("a")));

    }
}

```

Answer.

/*OUTPUT

```

HashMap's data> {Employee[ name=a] =emp1 OVERRIDDEN}
HashMap's size> 1
emp1 OVERRIDDEN

```

*/

Buckets= As hashCode() method returns 1, only 1 bucket location will be used.

Size= As equals() method always returns true, size will be 1, all three employees will be stored on same bucket location in one Entry (new Entry will keep on overriding previous Entry). We will always get last stored key-value pair only.

get()=we will be able to get object.

Question13. How many buckets will be there and what will be size of HashMap?

```

package p4;

import java.util.HashMap;

class Employee {

    private String name;

    public Employee(String name) { // constructor
        this.name = name;
    }

    @Override
    public int hashCode(){
        return 1;
    }

    //no equals() method

```



```

@Override
public String toString() {
    return "Employee[ name=" + name + "]" ;
}
}

/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */
public class Program4 {
    public static void main(String...a){

        HashMap<Employee, String> hm=new HashMap<Employee, String>();
        hm.put(new Employee("a"), "emp1");
        hm.put(new Employee("b"), "emp2");
        hm.put(new Employee("a"), "emp1 OVERRIDDEN");

        System.out.println("HashMap's data> "+hm);
        System.out.println("HashMap's size> "+hm.size());
        System.out.println(hm.get(new Employee("a")));

    }
}

```

Answer.

```

/*OUTPUT

HashMap's data> {Employee[ name=a] =emp1 OVERRIDDEN, Employee[ name=b] =emp2, Employee[ name=a]
=emp1}
HashMap's size> 3
null

```

*/

Buckets= As hashCode() method returns 1, only 1 bucket location will be used.

Size= As equals() method doesn't exist, size will be 3, all three employees will be stored on same bucket location but in different Entry.

get()=we won't be able to get object.

Question14. How many buckets will be there and what will be size of HashMap?

```

package p5;

import java.util.HashMap;

class Employee {

    private String name;

    public Employee(String name) { // constructor
        this.name = name;
    }

    //no hashCode() method

    @Override
    public boolean equals(Object obj){
        return true;
    }

    @Override
    public String toString() {
        return "Employee[ name=" + name + "]" ;
    }

}

/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */
public class Program5 {
    public static void main(String...a){

```

```

HashMap<Employee, String> hm=new HashMap<Employee, String>();
hm.put(new Employee("a"), "emp1");
hm.put(new Employee("b"), "emp2");
hm.put(new Employee("a"), "emp1 OVERRIDDEN");

System.out.println("HashMap's data> "+hm);
System.out.println("HashMap's size> "+hm.size());
System.out.println(hm.get(new Employee("a")));

    }
}

```

Answer.

/*OUTPUT

```

HashMap's data> {Employee[ name=b] =emp2, Employee[ name=a] =emp1 OVERRIDDEN, Employee[ name=a]
=emp1}
HashMap's size> 3
null

```

*/

Buckets= As hashCode() method is not there, hashcode generated for 3 objects will be different and we will end up using **3** buckets.

Size= Though equals() method is there (but because of hashCode() method's absence) **which always returns true**, we won't be able to locate correct bucket location for calling equals() method, so, size will be **3**.

get()=we won't be able to get object.

Question15. How many buckets will be there and what will be size of HashMap?

```

package p6;

import java.util.HashMap;

class Employee {

    private String name;

    public Employee(String name) { // constructor
        this.name = name;
    }

    @Override
    public int hashCode(){
        return (this.name==null ? 0: this.name.hashCode() );
    }

    //no equals() method

    @Override
    public String toString() {
        return "Employee[ name=" + name + " ] ";
    }

}

/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */
public class Program6 {

    public static void main(String...a){

        HashMap<Employee, String> hm=new HashMap<Employee, String>();
        hm.put(new Employee("a"), "emp1");
        hm.put(new Employee("b"), "emp2");
        hm.put(new Employee("a"), "emp1 OVERRIDDEN");

        System.out.println("HashMap's data> "+hm);
        System.out.println("HashMap's size> "+hm.size());
        System.out.println(hm.get(new Employee("a")));
    }
}

```



```
    }
}
```

Answer.

/*OUTPUT

```
HashMap's data> {Employee[ name=b] =emp2, Employee[ name=a] =emp1 OVERRIDDEN, Employee[ name=a]
=emp1}
HashMap's size> 3
null
```

*/

Buckets= As hashCode() method is overridden perfectly, **2** bucket locations will be used.**Size=** As equals() method is not their, size will be **3**,**get()**=we won't be able to get object.**Question16. How many buckets will be there and what will be size of HashMap?**

```
package p7;

import java.util.HashMap;

class Employee {

    private String name;

    public Employee(String name) { // constructor
        this.name = name;
    }

    //no hashCode() method

    @Override
    public boolean equals(Object obj){
        Employee emp=(Employee)obj;
        return (emp.name==this.name || emp.name.equals(this.name));
    }

    @Override
    public String toString() {
        return "Employee[ name=" + name + "] ";
    }
}

/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */
public class Program7 {
    public static void main(String...a){

        HashMap<Employee, String> hm=new HashMap<Employee, String>();
        hm.put(new Employee("a"), "emp1");
        hm.put(new Employee("b"), "emp2");
        hm.put(new Employee("a"), "emp1 OVERRIDDEN");

        System.out.println("HashMap's data> "+hm);
        System.out.println("HashMap's size> "+hm.size());
        System.out.println(hm.get(new Employee("a")));

    }
}
```

Answer.

/*OUTPUT

```
HashMap's data> {Employee[ name=a] =emp1, Employee[ name=a] =emp1 OVERRIDDEN, Employee[ name=b]
=emp2}
HashMap's size> 3
null
```

*/

Buckets= As hashCode() method is not there, hashCode generated for 3 objects will be different and we will end up using **3** buckets.

Size= Though equals() method is there (but because of hashCode() method's absence), we won't be able to locate correct bucket location for calling equals() method, so, size will be **3**.

get()=we won't be able to get object.



Must Read : [HashMap Custom implementation](#) for detailed implementation of equals and hashCode method and for understanding bucket and Entry)

See here for detailed understanding of how to [put, get, remove Employee object](#) in [custom HashMap](#).

Labels: [Core Java Interview questions](#) [Overriding equals and hashCode](#)

❏

Must read for you :

❏



2 Comments

 Login ▼

G

Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name

 1

Share

[Best](#) [Newest](#) [Oldest](#)

AM

Ankit Mittal

9 years ago

Thanks Mr. Mrinal Pandey.

  Reply  Share ›

MP

Mrinal Pandey

9 years ago

What a stuff. Huge thanks..

  Reply  Share ›[Subscribe](#)[Privacy](#)[Do Not Sell My Data](#)

E

[Newer Post](#)[Home](#)[Older Post](#)