**[How HashMap works in Java](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html" \o "How HashMap works in Java)**

HashMap in Java works on hashing principle. It is a data structure which allows us to store object and retrieve it in constant timeO (1) provided we know the key. In hashing, hash functions are used to link key and value in HashMap. Objects are stored by calling put (key, value) method of HashMap and retrieved by calling get (key) method. When we call put method, hashcode() method of the key object is called so that hash function of the map can find a bucket location to store value object, which is actually an index of the internal array, known as the table. HashMap internally stores mapping in the form of **Map.Entry** object which contains both key and value object. When you want to retrieve the object, you call the get () method and again pass the key object. This time again key object generate same hash code (it's mandatory for it to do so to retrieve the object and that's why HashMap keys are immutable e.g. String) and we end up at same bucket location. If there is only one object then it is returned and that's your value object which you have stored earlier. Things get little [tricky](http://java67.blogspot.com/2012/09/top-10-tricky-java-interview-questions-answers.html) when collisions occur. It's easy to answer this question if you have read good books on data structure and algorithms like [this](http://www.amazon.com/Data-Structures-Algorithm-Analysis-Edition/dp/0132576279?tag=javamysqlanta-20)one. If you know how hash table data structure works then this is a piece of cake   
  
Since the internal array of HashMap is of fixed size, and if you keep storing objects, at some point of time hash function will return same bucket location for two different keys, this is called collision in HashMap. In this case, a linked list is formed at that bucket location and a new entry is stored as next node.  
  
If we try to retrieve an object from this linked list, we need an extra check to search correct value, this is done by equals () method. Since each node contains an entry, HashMap keeps comparing entry's key object with the passed key using equals () and when it return true, Map returns the corresponding value.

Since searching inlined list is O (n) operation, in worst case hash collision reduce a map to linked list. This issue is recently addressed in Java 8 by replacing linked list to the tree to search in O (logN) time. By the way, you can easily verify how HashMap works by looking at the code of HashMap.java in your Eclipse IDE if you know [how to attach source code of JDK in Eclipse](http://javarevisited.blogspot.com/2012/12/how-to-attach-source-in-eclipse-Jar-JDK-debugging.html).  
  
  
How HashMap works in Java or sometimes how does get method work in HashMap is a very common question on Java interviews nowadays. Almost everybody who worked in Java knows about HashMap, where to use HashMap and difference between Hashtable and HashMap then why this interview question becomes so special? Because of the depth it offers.  
  
It has become very [popular Java interview question](http://javarevisited.blogspot.com/2015/10/133-java-interview-questions-answers-from-last-5-years.html) in almost any senior or mid-senior level Java interviews. Investment banks mostly prefer to ask this question and sometimes even ask you to implement your own HashMap based upon your coding aptitude. The introduction of [ConcurrentHashMap](http://javarevisited.blogspot.co.uk/2013/02/concurrenthashmap-in-java-example-tutorial-working.html) and other concurrent collections has also made this questions as starting point to delve into a more advanced feature. let's start the journey.

**How HashMap Internally Works in Java**

Questions start with simple statement:

**Have you used HashMap before**or**What is HashMap? Why do you use it**

Almost everybody answers this with yes and then interviewee keep talking about common facts about HashMap like HashMap accept null while Hashtable doesn't, [HashMap is not synchronized](http://javarevisited.blogspot.com/2010/10/difference-between-hashmap-and.html), HashMap is fast and so on along with basics like its stores key and value pairs etc. This shows that person has used HashMap and quite familiar with the functionality it offers, but interview takes a sharp turn from here and next set of follow-up questions gets more detailed about fundamentals involved with HashMap in Java. Interviewer strike back with questions like:

**Do you Know how HashMap works in Java** or **How does get () method of HashMap works in Java**

And then you get answers like,  I don't bother its standard Java API, you better look code on Java source or Open JDK; I can find it out in Google at any time etc. But some interviewee definitely answers this and will say **HashMap works on the principle of hashing**, we haveput(key, value) and get(key) method for storing and retrieving Objects fromHashMap. When we pass Key and Value object  to put() method on Java HashMap, HashMap implementation calls [hashCode method](http://javarevisited.blogspot.sg/2011/10/override-hashcode-in-java-example.html)on Key object and applies returned hashcode into its own hashing function to find a bucket location for storing Entry object, important point to mention is that HashMap in Java stores both key and value object asMap.Entry in a bucket which is essential to understand the retrieving logic.   
  
If people fail to recognize this and say it only stores Value in the bucket they will fail to explain the retrieving logic of any object stored in Java HashMap. This answer is very much acceptable and does make sense that interviewee has a fair bit of knowledge on how hashing works and how HashMap  works in Java. But this is just start of story and confusion increases when you put interviewee on scenarios faced by Java developers on day by day basis. Next question could be about collision detection and collision resolution in Java HashMap  e.g.

**What will happen if two different objects have the same hashcode?**

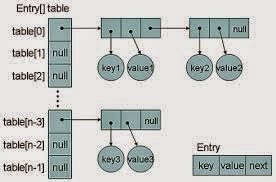
Now from here onwards real confusion starts, sometime candidate will say that since hashcode is equal, both objects are equal and HashMap  will throw exception or not store them again etc, Then you might want to remind them about [equals() and hashCode() contract](http://javarevisited.blogspot.sg/2011/02/how-to-write-equals-method-in-java.html)that two unequal objects in Java can have same hashcode. Some will give up at this point and few will move ahead and say "Since hashcode is same, bucket location would be same and collision will occur in HashMap Since HashMap uses LinkedList to store object, this entry (object of Map.Entry comprise key and value )  will be stored in [LinkedList](http://javarevisited.blogspot.sg/2012/02/difference-between-linkedlist-vs.html). Great this answer make sense though there are many collision resolution methods available  like linear probing and chaining, this is simplest and HashMap in Java does follow this. But story does not end here and interviewer asks

**How will you retrieve Value object  if two Keys will have the same hashcode?**

Interviewee will say we will call get() method and then HashMap uses Key Object's hashcode to find out bucket location and retrieves Value object but then you need to remind him that there are two Value objects are stored in same bucket , so they will say about [traversal in LinkedList](http://javarevisited.blogspot.sg/2010/10/how-do-you-find-length-of-singly-linked.html)until we find the value object , then you ask *how do you identify value object because you don't  have value object to compare* ,Until they know that HashMap  stores both Key and Value inLinkedList node or as Map.Entry they won't be able to resolve this issue and will try and fail.

But those bunch of people who remember this key information will say that after finding bucket location, we will **call keys.equals() method** to identify a correct node in LinkedList and return associated value object for that key in Java HashMap. Perfect this is the correct answer.

In many cases interviewee fails at this stage because they get confused between[hashCode()](http://javarevisited.blogspot.sg/2011/10/override-hashcode-in-java-example.html)and equals(**)** or keys and values object in Java HashMap  which is pretty obvious because they are dealing with the hashcode() in all previous questions and equals() come in picture only in case of retrieving value object from HashMap in Java. Some good developer point out here that using immutable, [final object](http://javarevisited.blogspot.sg/2011/12/final-variable-method-class-java.html) with proper equals() and hashcode()implementation would act as perfect Java HashMap  keys and**improve the performance of Java HashMap  by reducing collision**. Immutability*also allows caching their hashcode of different keys* which makes overall retrieval process very fast and suggest that [String](http://javarevisited.blogspot.sg/2011/07/string-vs-stringbuffer-vs-stringbuilder.html)and various wrapper classes e.g. Integer very good keys in Java HashMap.

[](http://4.bp.blogspot.com/-adRczhctozE/VD_eimhTQbI/AAAAAAAACCg/lfA1G5GZXyM/s1600/How%2BHashMap%2Bworks%2Bin%2BJava%2B(1).jpg)

Now if you clear this entire Java HashMap interview,  You will be surprised by this very interesting question "**What happens On HashMap in Java if the size of the HashMap  exceeds a given threshold defined by load factor ?"**. Until you know how HashMap  works exactly you won't be able to answer this question. If the size of the Map exceeds a given threshold defined by load-factor e.g. if the load factor is .75 it will act to re-size the map once it filled 75%. Similar to other collection classes like [ArrayList](http://javarevisited.blogspot.sg/2011/05/example-of-arraylist-in-java-tutorial.html),  Java HashMap re-size itself by creating a new bucket array of size twice of the previous size of HashMap and then start putting every old [elementhttp://images.intellitxt.com/ast/adTypes/icon1.png](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html) into that new bucket array. This process is calledrehashing because it also applies the hash function to find new bucket location.

If you manage to answer this question on HashMap in Java you will be greeted by **"do you see any problem with resizing of HashMap  in Java"**, you might not be able to pick the context and then he will try to give you hint about multiple thread accessing the JavaHashMap and potentially looking for **race condition on HashMap  in Java**.

So the answer is Yes there is potential [race condition](http://javarevisited.blogspot.sg/2012/02/what-is-race-condition-in.html) exists while resizing HashMap in Java, if two [thread](http://javarevisited.blogspot.sg/2011/02/how-to-implement-thread-in-java.html)at the same time found that now HashMap needs resizing and they both try to resizing. on the process of resizing of HashMap in Java, the element in the bucket which is stored in linked list get reversed in order during their migration to new bucket because Java HashMap  doesn't append the [new elementhttp://images.intellitxt.com/ast/adTypes/icon1.png](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html) at tail instead it append new element at the head *to avoid tail traversing*. If race condition happens then you will end up with an infinite loop. Though this point, you can potentially argue that what the hell makes you think to use HashMap  in multi-threaded environment to interviewer :)

**Some more Hashtable and HashMap Questions**

Few more question on HashMap in Java which is contributed by readers of Javarevisited blog:

**1) Why String, Integer and other wrapper classes are considered good keys?**

String, Integer and other wrapper classes are natural candidates of HashMap key, andString is most frequently used key as well because [String is immutable and final](http://javarevisited.blogspot.sg/2010/10/why-string-is-immutable-in-java.html), and overrides equals and hashcode() method. Other wrapper class also shares similar property. Immutability is required, in order to prevent changes on fields used to calculatehashCode() because if key object returns different hashCode during insertion and retrieval than it won't be possible to get an object from HashMap.   
  
Immutability is best as it offers other advantages as well like [thread-safety](http://javarevisited.blogspot.sg/2012/01/how-to-write-thread-safe-code-in-java.html), If you can  keep your hashCode same by only making certain fields final, then you go for that as well. Sinceequals() and hashCode() method is used during retrieval of value object fromHashMap, it's important that key object correctly override these methods and follow contact. If unequal object returns different hashcode than chances of collision will be less which subsequently improve the performance of HashMap.

**2) Can we use any custom object as a key in HashMap?**

This is an extension of previous questions. Of course you can use any Object as key in JavaHashMap provided it follows equals and hashCode contract and its hashCode should not vary once the object is inserted into [Map](http://javarevisited.blogspot.sg/2011/12/how-to-traverse-or-loop-hashmap-in-java.html). If the custom object is Immutable than this will be already taken care because you can not change it once created.

**3) Can we use ConcurrentHashMap in place of Hashtable?**

This is another question which getting popular due to increasing popularity ofConcurrentHashMap. Since we know Hashtable is synchronized but ConcurrentHashMapprovides better concurrency by only locking portion of map determined by concurrency level. ConcurrentHashMap is certainly introduced as Hashtable and can be used in place of it, but Hashtable provides stronger thread-safety than ConcurrentHashMap. See my post[difference between Hashtable and ConcurrentHashMap](http://javarevisited.blogspot.sg/2011/04/difference-between-concurrenthashmap.html) for more details.

Personally, I like this question because of its depth and number of concept it touches indirectly if you look at questions asked during interview this HashMap  questions has verified

* The concept of hashing
* Collision resolution in HashMap
* Use of equals () and hashCode () and their importance in HashMap?
* The benefit of the immutable object?
* Race condition on HashMap  in Java
* Resizing of Java HashMap

Just to summarize here are the answers which do make sense for above questions

**How HashMap  works in Java**

HashMap  works on the principle of hashing, we have put() and get() method for storing and retrieving object from HashMap.When we pass both key and value to put() method to store on HashMap, it uses key object hashcode() method to calculate hashcode and them by applying hashing on that hashcode it identifies bucket location for storing value object. While retrieving it uses key object equals method to find out correct key value pair and return value object associated with that key. HashMap  uses linked list in case of collision and object will be stored in next node of linked list. Also, [HashMap stores both key and value tuple](http://java67.blogspot.com/2013/02/10-examples-of-hashmap-in-java-programming-tutorial.html) in every node of linked list in the form of Map.Entry object.

**What will happen if two different HashMap  key objects have the same hashcode?**

They will be stored in the same bucket but no next node of linked list. And keys equals () method will be used to identify correct key value pair in HashMap.

**How null key is handled in HashMap? Since equals() and hashCode() are used to store and retrieve values, how does it work in case of the null key?**  
The null key is handled specially in HashMap, there are two separate methods for thatputForNullKey(V value) and getForNullKey(). Later is offloaded version of get() to look up null keys.  Null keys always map to index 0.  This null case is split out into separate methods for the sake of performance in the two most commonly used operations (get and put), but incorporated with conditionals in others. In short, equals() andhashcode() method are not used in case of null keys in HashMap.  
  
here is how nulls are retrieved from HashMap  
  
   **private** V **getForNullKey**() {

**if** (size == **0**) {

**return** **null**;

}

**for** (Entry<K,V> e = table[**0**]; e != **null**; e = e.next) {

**if** (e.key == **null**)

**return** e.value;

}

**return** **null**;

}

In terms of usage, Java HashMap is very versatile and I have mostly used HashMap as cache in an electronic trading application I have worked. Since finance domain used Java heavily and due to performance reason we need caching HashMap and ConcurrentHashMap  comes as very handy there. You can also check following articles from Javarevisited to learn more about HashMap and Hashtable in Java:

**HashMap Changes in JDK 1.7 and JDK 1.8**

There is some [performance improvement done on HashMap and ArrayList from JDK 1.7](http://javarevisited.blogspot.com/2014/07/java-optimization-empty-arraylist-and-Hashmap-cost-less-memory-jdk-17040-update.html), which reduce memory consumption. Due to this empty Map are lazily initialized and will cost you less memory. Earlier, when you create HashMap e.g. new HashMap() it automatically creates an array of default length e.g. 16. After some research, Java team found that most of this Map are temporary and never use that many elements, and only end up wasting memory. Also, From JDK 1.8 onwards HashMap has introduced an improved strategy to [dealhttp://images.intellitxt.com/ast/adTypes/icon1.png](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html) with high collision rate. Since a poor hash function e.g. which always return location of same bucket, can turn a HashMap into linked list, i.e. converting get() method to perform in O(n) instead of O(1) and someone can take advantage of this fact, Java now internally replace linked list to a binary true once certain threshold is breached. This ensures performance or order O(log(n)) even in the worst case where a hash function is not distributing keys properly.

Posted by Javin Paul [[http://img1.blogblog.com/img/icon18_email.gif](https://www.blogger.com/email-post.g?blogID=8712770457197348465&postID=5388962816690361023)](https://www.blogger.com/email-post.g?blogID=8712770457197348465&postID=5388962816690361023)

[Email This](https://www.blogger.com/share-post.g?blogID=8712770457197348465&postID=5388962816690361023&target=email) [BlogThis!](https://www.blogger.com/share-post.g?blogID=8712770457197348465&postID=5388962816690361023&target=blog" \o "BlogThis!" \t "_blank) [Share to Twitter](https://www.blogger.com/share-post.g?blogID=8712770457197348465&postID=5388962816690361023&target=twitter) [Share to Facebook](https://www.blogger.com/share-post.g?blogID=8712770457197348465&postID=5388962816690361023&target=facebook)

Labels: [collections interview questions](http://javarevisited.blogspot.com/search/label/collections%20interview%20questions?max-results=3), [core java](http://javarevisited.blogspot.com/search/label/core%20java?max-results=3), [core java interview question](http://javarevisited.blogspot.com/search/label/core%20java%20interview%20question?max-results=3), [java collection tutorial](http://javarevisited.blogspot.com/search/label/java%20collection%20tutorial?max-results=3)

Location: [North America](https://maps.google.com/maps?q=North+America@37.996162679728116,-101.25&z=10)

**126 comments :**

**Anonymous said...**

This same question was asked to me during my interview with a Investment bank for Java FIX developer position. I like follow up question though :)

[February 3, 2011 at 4:01 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1296734507289#c7665346046109527808)

**[Javin @ Tibco RV Tutorial](http://javarevisited.blogspot.com/)said...**

Hi Sandeep,  
  
Nice to see you here back. yes this is common core java interview question but when you ask cross question then half of developer fails. in most cases they forgot that hashmap stores both key and value in bucket.I have seen interviews where they think that only value is stored in bucket which leads to confusion when asking about duplicate hashcode.  
  
Thanks  
JAvin

[February 4, 2011 at 5:21 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1296825665356#c8993760477571300532)

**Anonymous said...**

Hi Dude Can we use hashMap in multithreaded scenario ? is there any issue with that ?

[February 7, 2011 at 3:42 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1297078977250#c6706920634719087335)

**[Gautam](http://fast-code.sourceforge.net/)said...**

Good explanation. Some pictures and flowcharts will be good.

[February 7, 2011 at 9:55 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1297101333475#c8603422175127835887)

**[Foudres](https://www.blogger.com/profile/05977228915328575616)said...**

Well it is obvious that the only mandatory things to store in a dictionnary/map is the key and the value.  
  
Without any more hypothesis, this will resort to O(n) when retriving from the key.  
  
But O(n) is very very bad.  
  
So you start to think at a way to have better performance. You can think about an index. You use ordered type and construct a tree.  
  
Problem is that not all types can be ordered. The hash allow you to treat any type like an ordered one with the exception of collisions. Thus you have O(log n), but could have O(n) in worst case if all entry have the same hash.  
  
But really I found your interview question to be confusing. I can describe what is a dictionnary, what strategy is basically used for it, but I can't say how JAVA implement it. This is a complex question that need basically that you have read the source code and that would depend of the JVM implementation you use.  
  
And if you are really interrested for this matter you now that depending of the exact algorythm you use you have very different performance depending of your exact needs :  
  
map with lot of entry  
map with few entries.  
  
how to optimise the partitioning what ever the hash you are given...  
  
For hash and map keys : of course you should use only immuable key. A dictionnary/map is a nonscence if your key is mutable as it is likely that the hash change too. This is bad pratice.

[February 7, 2011 at 10:16 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1297102607249#c6945051710205172189)

**[Rod](https://www.blogger.com/profile/01080301889766047685)said...**

I think I'm going to steal this as an interview question. Thank you for the idea :)  
  
I took it upon myself to provide source for how to build your own (for educational purposes only) @ http://www.dzone.com/links/write\_your\_own\_java\_hashmap.html. I would expect a candidate to be able to pseudo-code something similar in an interview - or to admit they have no idea how it all actually works.

[February 7, 2011 at 4:11 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1297123879090#c2822482582727439190)

**[Javin @ Tibco RV Tutorial](http://javarevisited.blogspot.com/)said...**

Hi Anonymous, Since HashMap is not synchronized there are issue while using it on multithreaded scenario e..g some times get() method of hashMap goes into infinite loop if rehashing and insertion occur in same time. there are other alternative available for multithreaded scenario e.g. ConcurrentHashMAp or Hashtable you can also see my article [What is the difference between Synchronized Collection classes and Concurrent Collection Classes ? When to use what ?](http://javarevisited.blogspot.com/2010/10/what-is-difference-between-synchronized.html)

[February 8, 2011 at 4:33 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1297168416858#c3523010341749996138)

**[Pallavi](https://www.blogger.com/profile/08582748639969772118)said...**

This is really helpful.  
  
I want to know how hashset is implemented in java. Do we need to override equals method in addition to hashcode?

[February 13, 2011 at 8:56 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1297616180107#c7178393328354729671)

**[0x1e](https://www.blogger.com/profile/13780658737498694489)said...**

Hi Pallavi,  
  
Internally HashSet is backed by HashMap. And generally yes, in case you have custom hashCode, you may want to override equals so they are consistent with one another.  
  
Cheers  
Oleksiy

[March 8, 2011 at 5:43 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1299591789920#c7211657419619360680)

**Anonymous said...**

This is unrealistic, why did you focus on collisions? Java handles that problem as you pointed out, but quite frankly if you're worried about collisions maybe you should be more worried about your perception of hashing. If collisions would be the problem, then you're doing it wrong, you should not use an hashmap in the first place.  
I'm sorry but discussing this sounds a bit stupid to me. If you would ask when an hashmap should be used and why, you'll be selecting better candidates.

[April 30, 2011 at 4:13 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1304162008885#c9164801665337597729)

**[Javin @ FIX Protocol Tutorials](http://javarevisited.blogspot.com/2011/02/fix-protocol-session-or-admin-messages.html)said...**

Hi Anonymous, I appreciate your thoughts but purpose here is to check whether interviewee is familiar with some common programming concepts or not. would you like to hire a guy who is familiar with how hashmap internally works or the one who just simply used hashmap for years but never bothered to think aabout it ? on a separate note yes your question "when an hashmap should be used and why" is also a starter on this topic.  
  
Thanks for your comment.

[May 3, 2011 at 6:02 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1304427764733#c7845636239501814465)

**[Ramakrishna](http://in.linkedin.com/in/sreeramakrishna)said...**

Put code snippets and add few points why you have used HashMap over other data structure.  
Also mention some of the improvements that can be made to your HashMap code snippets. Improvement w.r.t. performance,accuracy of retrieval.  
  
it would be nice to mention overriding equals() method?

[May 5, 2011 at 9:04 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1304611496802#c6294803192048985649)

**[Javin @ FIX Protocol and Electroinc Trading](http://javarevisited.blogspot.com/2011/02/fix-protocol-session-or-admin-messages.html)said...**

@Ramakrishna , those are actually discussed but yes as a part of actual interview , It does make sense to ask interviewee to write equals() and hashcode() implementation for any class along with those followup.

[May 14, 2011 at 8:23 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1305429790250#c3346772922557052183)

**[Tanito](https://www.blogger.com/profile/03011037374329246346)said...**

Nice article.

[June 30, 2011 at 9:56 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1309452987026#c5855449822354788576)

**Anonymous said...**

Can you explain bit more about put() operation in hashmap and get() operation in hashmap. Also what happens if one thread is iterating over hashmap and other thread is put element into hashmap ? or one thread iterates over hashmap and other thread get() elements from hashmap , any code example will be good. anyway nice java hashmap interview question.

[July 7, 2011 at 6:33 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1310088789515#c128301306072779418)

**Anonymous said...**

I was looking for some information on how hashmap works internally and found your site, you covered almost everything related to what is hashmap in Java, How HashMap works internally in java and more importantly how get method of hashmap works in java. kudos to you buddy. thanks a lot

[July 9, 2011 at 2:46 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1310204787252#c5289099726389010409)

**Javin said...**

Hi Anonymous 1 , put operation in hashmap is used to insert object into hashmap, put method of hashmap takes two argument one is key object and other is value object. always make sure your key object in hashmap is immutable. how get method of hashmap works is clearly explain in the article itself.

[July 9, 2011 at 8:48 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1310269718678#c7439808466285743577)

**[SHASHIDHAR Y](https://www.blogger.com/profile/06243301927000469256)said...**

Hi,  
Nice article. This was the exact sequence I was asked in an interview and I totally messed it up due to lack of knowledge about equals and hashcode methods. Hope, After reading this, I will never ever repeat the same mistake.

[July 13, 2011 at 11:53 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1310626381048#c4320633550226714521)

**Roger said...**

Hi Javin, I am working on electronic trading system which we are going to design for foreign exchange and currency trading. I have some question related to FIX Protocol and how we can use FIX Protocol for currency trading. Can you please help me. since I don't have any prior experience on writing any electronic trading system, any suggestion will be appreciated

[July 17, 2011 at 1:56 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1310892989552#c1221406904835974962)

**Anonymous said...**

Nice Article. I understood how the insertion & retrieval work in HashMap. However, I have a very fundamental question:  
  
As per java,  
If two objects are equal by equals() method then their hashcode must be same.  
  
Now, what is the rule is not followed.   
Lets say we have 2 objects, obj1 & obj2. Lets assume equals method returns two on these 2 objects however the hashCode returns 2 different numbers.  
If we try to add these 2 objects (as key) into a hashmap, we will end up having these 2 objects(& associated values) stored in 2 different buckets. (Had the hash code returned the same numbers, the second put() method would just have updated value of the first entry). So no issue with put() method.  
Now when you try to retrieve values from hashmap using those two objects, you will successfully get 2 distinct entries created above. So, what's that I am missing?  
  
Thanks.

[July 24, 2011 at 12:32 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1311492739684#c6379548706207770809)

**Anonymous said...**

Hi, Is it possible to replace HashMap in Java with ConcurrentHashMap ? we have a project where hashmap has used a lot and now we want to replace that with ConcurrentHashMap.

[July 27, 2011 at 6:07 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1311815247902#c3662008859200675263)

**[NaiveGeek](https://www.blogger.com/profile/13611266100311159917)said...**

Hi ,   
Thanks for presenting this article very clearly. I have been working for more than 5 yrs in java until now but never got the proper reason behind how hashmap works.  
thanks for such a information  
Best  
NaiveGeek

[July 31, 2011 at 8:48 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1312170501958#c1178285363181140467)

**Anonymous said...**

@Anonymous you can't just replace HashMap with ConcurrentHashMap because one is not synchronized at all while other provides synchronization. you probably wanted to ask replacing Hashtable with ConcurrentHashMap but even on that case you need to make sure you are not entering into race condition by using putIfAbsent() method instead of put.

[August 4, 2011 at 6:51 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1312509119439#c3048625420224000788)

**Anonymous said...**

I think your May 3 comment reveals that your approach is unreasonable. You say, "...I appreciate your thoughts but purpose here is to check whether interviewee is familiar with some common programming concepts or not. would you like to hire a guy who is familiar with how hashmap internally works or the one who just simply used hashmap for years but never bothered to think aabout it ?..."  
  
Do you understand everything you've used for years? How does an internal combustion engine work? How about your microwave? How are your clothes made? You've used your body for years, so you should understand anatomy and physiology? Getting closer to programming, how is a keystroke sent to your text editor? What happens if you hit two keys simultaneously? How is the file system implemented? How does the compiler transform Java code to byte code? In Java, do you understand type erasure, generics, wildcards? By your logic, you should since you use it all the time.  
  
Would you agree that we could find an everyday topic that you're not an expert at? If so, I think your interview should be more reasonable. Joel Spolsky is known for hiring some of the industry's truly best people; you can see how he does it here: http://www.joelonsoftware.com/articles/fog0000000073.html

[August 29, 2011 at 7:34 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1314671686566#c1701508706807346059)

**[Javin @ enum in java](http://javarevisited.blogspot.com/2011/08/enum-in-java-example-tutorial.html)said...**

@Anonymous,Thanks for your comment. I see you have a valid point but don't you agree that familiarity with common programming techniques and concept is essential especially if you are a professional programmer, A doctor definitely knows about anatomy of body , a mechanic definitely knows how internal combustion engine work. Also I have not said that don't hire a person because it just doesn't understand how hashmap works in java or what is difference between hashtable and hashmap, this question is more to see his appetite and attitude about its works and technology. Thanks for the link I been following Joel and benefited from his blog.

[August 29, 2011 at 8:29 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1314674949111#c3862064693354176350)

**Anonymous said...**

Awesome post. You've demonstrated your knowledge and desire to learn--you're one of the good guys, no doubt.  
  
"... this question is more to see his appetite and attitude about its works and technology." Awesome! I agree is a fantastic thing to test an interviewee on, but I wonder if one could ask it more directly? Perhaps something like, "Can you tell me about the last time you really dug deep to understand a programming concept?" Maybe more abstractly as in "What are you passionate about?" or "What do you read?"

[September 6, 2011 at 11:08 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1315332491632#c2190587764252835648)

**[Javin @ String and Stringbuffer in java](http://javarevisited.blogspot.com/2011/07/string-vs-stringbuffer-vs-stringbuilder.html)said...**

Thanks Anonymous for your kind comment. Yes those direct questions can be another good alternative.looking forward to see you again.

[September 7, 2011 at 7:03 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1315404235695#c9135502884698086602)

**Anonymous said...**

Amazing article. I wish i could have found this blog before my interview, where i was asked the very questions that you have answered. I was asked to write a HashMap implementation of my own and after reading this article it looks like i could have given a decent try.

[September 29, 2011 at 1:23 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1317284591980#c2812030704483473886)

**[Gokul](https://www.blogger.com/profile/03599243290536794688)said...**

Nice article Javin! Would be nice if you can talk more about how this discussion leads onto ConcurrentHashMap...

[October 9, 2011 at 2:04 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1318194253829#c4523537033992489721)

**[Javin @ OutOfMemoryError in Java](http://javarevisited.blogspot.com/2011/09/javalangoutofmemoryerror-permgen-space.html)said...**

You have a point Gokul , ConcurrentHashMap is very popular data-structure for high frequency trading platform and it has lots of details which can be tested like Can we replace hashtable with ConcurrentHashMap, may be I will write on that sometime. Thanks for your comments.

[October 10, 2011 at 5:27 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1318249679547#c5734561420658236838)

**[Javin @ Generics in Java](http://javarevisited.blogspot.com/2011/09/generics-java-example-tutorial.html)said...**

Thanks for your comment Swapna, Sample priciple applies on hashtable as well. you may also like my article [How substring method works in Java](http://javarevisited.blogspot.com/2011/10/how-substring-in-java-works.html)

[October 14, 2011 at 6:14 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1318641297479#c6749087230609684464)

**[Javin @ Date To String in Java](http://javarevisited.blogspot.com/2011/09/convert-date-to-string-simpledateformat.html)said...**

@Viraj, @ Anonymous and @Bhaskara , Thanks for your comments guys and good to know that you like my hashmap interview experience. you may also like [How Substring works in Java](http://javarevisited.blogspot.com/2011/10/how-substring-in-java-works.html)

[October 29, 2011 at 7:13 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1319897600219#c1719324491609071808)

**Anonymous said...**

This tutorial is simple and clear, very good.  
May I ask a question that is about equal and hashcode?  
  
I can't understand why the following code yield "False" ?  
if( new Boolean("True") == new Boolean("True"))  
System.out.println("True");  
else  
System.out.println("False");  
  
I conduct 2 tests and still don't understand why the above is "False".   
Boolean b1 = new Boolean("True");  
Boolean b2 = new Boolean("True");  
  
b1 and b2 is the same by the equal method  
System.out.println(b1.equals(b2));  
  
Hash code are the same  
System.out.println(b1.hashCode());  
System.out.println(b2.hashCode());  
  
Thanks

[November 15, 2011 at 8:30 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1321374657589#c1284590181578744577)

**[Rosario](https://www.blogger.com/profile/07549223396482670557)said...**

because the operator "==" is used to verify the matching of objects reference, not the value. In other word you have created two distinct object using "new Boolean);" so you have different reference for two object allocated in different area of memory. I f you want compare object of same class and you want know if they have same hashcode you should use the method Equals(, it works well in this case in other case for you need to use the method compareTo() and compare().  
  
Bye!

[November 20, 2011 at 8:31 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1321806686178#c337102736570372161)

**Surya said...**

Hi ,  
Anonymous brought up a valid point. Both equals() and hashCode() on an object should confirm the object equality. "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." .

[November 20, 2011 at 5:46 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1321839978473#c8986209558828555847)

**Anonymous said...**

Hi,  
  
Sorry but I think that asking these kind of questions in a job interview proves nothing. Today you can find anything you need on the web so there's no point in memorizing implementations of HashMap, HashTable etc.  
  
I believe that your questions could be good if you would ask about hashing in general (two objects has the same hashcode etc).  
  
The most important thing for a developer, TMHO, is that she could THINK, not memorize...  
  
Kind regards  
N

[November 27, 2011 at 8:08 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1322453326985#c681689758562192313)

**Anonymous said...**

Isn't other collection classes which is based on hashing e.g. HashSet or LinkedHashMap also works in Same principle ? Does HashSet also use equals and hashcode to insert and retrieve objects ?

[December 19, 2011 at 7:42 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1324352574031#c6172417583531420240)

**Anonymous said...**

does it stores key and value object in different node in the linked list or in same node?

[January 1, 2012 at 10:54 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1325487291118#c29352158132892660)

**[Javin @ jsp interview questions answers](http://javarevisited.blogspot.com/2011/10/jsp-interview-questions-answers-for.html)said...**

Hi Anonymous, keys and values of One Entry in HashMap is stored at one node and that's how after finding bucket location it finds correct object in hashmap.

[January 2, 2012 at 12:03 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1325491429579#c552139790175029829)

**[shivu](https://www.blogger.com/profile/06150320292250167625)said...**

Good article, but created some confusion in my mind...   
  
If the computed hashcode is same for two objects, hashMap will override the existing value for that key.   
  
My question is what is the scenario the value would be overriden and added to same bucket in the HashMap.  
  
The piece of code placed here for overriding  
  
String s1 = new String("abc");  
String s2 = new String("abc");  
System.out.println("s1 hashcode: "+s1.hashCode());  
System.out.println("s2 hashcode: "+s2.hashCode());  
  
Map map = new HashMap();  
map.put(s1, s1);  
map.put(s2, new String("cdf"));  
System.out.println(map);

[January 4, 2012 at 3:08 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1325675326615#c1530332986807437381)

**[Javin @ spring interview questions answers](http://javarevisited.blogspot.com/2011/09/spring-interview-questions-answers-j2ee.html)said...**

Hi Shivu, if key is same i.e. equals by [equals method in Java](http://javarevisited.blogspot.com/2011/02/how-to-write-equals-method-in-java.html) than [hashcode](http://javarevisited.blogspot.com/2011/10/override-hashcode-in-java-example.html) would be same and value will be replaced but if key is not same i.e. not equal but hashcode is same (Which is possible in java) than bucked location would be same and collision would happen and second object will be stored on second node of linked list structure on Map.

[January 4, 2012 at 4:59 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1325681941224#c5397444480033717005)

**Gaurav said...**

isn't it all other hashing related data-structure like Hashtable, HashSet, LinkedHashMap is implemented in this way ? At least implementation of put() and get() should be same with some addition specific details related to individual collection.

[January 5, 2012 at 4:59 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1325811547340#c3102300176741339908)

**[Mandakini Mishra](https://www.blogger.com/profile/09033003987998443222)said...**

You are just awesome.Have no words to praise you.You have explained in such difficult thing in such a simple way.Fantastic, Great work.  
  
Mandakini

[January 17, 2012 at 2:49 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1326797372145#c1103043468999213816)

**Anonymous said...**

hi,  
still i have little confusion over this  
  
can you specify an example how can two different objects has same hash code(say for two custom objects)

[January 28, 2012 at 7:32 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1327807979681#c5520829821132926235)

**[MJB](https://www.blogger.com/profile/06321599173782751450)said...**

In fact Hashset in java is simply a delegated hashmap: here's a snippet of the source to prove the point  
  
private transient HashMap map;  
  
// Dummy value to associate with an Object in the backing Map  
private static final Object PRESENT = new Object();  
  
public boolean add(E e) {  
return map.put(e, PRESENT)==null;  
}  
  
// similar methods that delegate follow

[February 1, 2012 at 7:53 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1328154818398#c7746658159798614324)

**Joe said...**

Thanks for such nice article. Will be expecting such articles in future also.

[February 1, 2012 at 9:45 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1328161503157#c5770220490973460975)

**[Javin @ HashMap vs HashSet](http://javarevisited.blogspot.com/2011/09/difference-hashmap-vs-hashset-java.html)said...**

@MJB, Indeed HashSet uses HashMap for hashing support and thank for putting the code snippet it makes the point much clearer.

[February 3, 2012 at 10:32 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1328337140037#c4401117989023228025)

**[Javin @ hashtable example java](http://javarevisited.blogspot.com/2012/01/java-hashtable-example-tutorial-code.html)said...**

@Mandakini Mishra , Thanks for your kind words, glad to hear you like this Java hashmap question answer article.

[February 3, 2012 at 10:33 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1328337236314#c724357769629390115)

**Anonymous said...**

This is a great article, but if i have one complaint, it seems to be written poorly in terms of english (almost as thought a translator was used). i am finding it hard to understand some of the answers because of this.

[February 12, 2012 at 4:24 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1329092642598#c1120535381690968376)

**Anonymous said...**

I have one comment regarding your explanation: key.equals() is used also when you use put() method, because if hashcode() of an existing key is the same with the one which you want to insert, yes, the new key will be the next one in the bucket, ONLY IF THE KEY IS UNIQUE. So in order to do that, it is used also method key.equals(), to prevent duplicates. (You said that we are talking about equal() method only when we are retriving data from HashMap)  
Regards

[February 19, 2012 at 9:58 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1329674315996#c2201342560731599604)

**[Javin @ collection interview question](http://javarevisited.blogspot.com/2011/11/collection-interview-questions-answers.html)said...**

@Anonymous, your observation is correct key.equals() is used on both put() and get() method call if object already exits in bucked location on hashmap and that's why tuples contains both key and value in each node. thanks for pointing it and making it more clear.

[February 20, 2012 at 5:35 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1329744920465#c5479898272491200557)

**Anonymous said...**

Hi,  
Overriding equals and hashcode we can make the object as key in hashmap.While putting the the reference to the object as Key in hashmap or creating the object and setting it in put method , what difference does it make in retreiving the object?

[February 21, 2012 at 11:02 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1329850932255#c1530543260960042897)

**[Javin @ producer consumer problem java](http://javarevisited.blogspot.com/2012/02/producer-consumer-design-pattern-with.html)said...**

@Tofeeq thanks for your comment. Glad to hear that you like interview question on java hashmap.

[February 24, 2012 at 7:23 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1330096990358#c1796149842047336455)

**[P-H](https://www.blogger.com/profile/15963955701683788257)said...**

Thanks Javin, great post again.  
  
This is true that Hashmap are common questions someone can get in interview from a Java & Java EE perspective. I interview new members myself at my current company (CGI Inc.) and the typical questions that I ask, in a Java EE context are:  
  
- What are best & known risks when using Hashmap data structure?  
- What JDK Map data structure is more suitable to handle concurrent read & write operations in a Java EE environment?  
  
The answer for the first one is all about adding proper synchronization when Hashmap is used in a concurrent Thread access scenario e.g. same Hashmap instance shared across multiple Threads.  
Primary known risks are Hashmap internal index corruption and infinite looping, which can bring your JVM to its knees.  
  
The answer about the second one is to use the ConcurrentHashMap, introduced in JDK 1.5, which provides Thread safe operations but no blocking get() operation; which is key for proper performance. This is the typical data structure used these days in modern Java EE container implementations.  
  
If your readers are interested, I have a case study of a recent [Hashmap infinite looping problem](http://javaeesupportpatterns.blogspot.com/2011/12/hashmap-infinite-loop-problem-case.html), actual Weblogic 11g defect (non Thread safe Hashmap)!  
  
Thanks again for all that good work.  
P-H

[February 26, 2012 at 6:29 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1330266585836#c7851685986113485696)

**[ADNAN ALI](https://www.blogger.com/profile/16991779029961059536)said...**

I am new to Java World! So I have this problem stated as under.....  
  
I have sortedHashMap and hashMap2  
  
Now i have to get hashMap3 based on sortedHashMap sorted order.  
  
Secondly; I have also to limit hashMap3 as of first sorted 100 values.....rest map I dont need.  
Please suggest me coding example or junk of code.  
  
Thanks!

[February 28, 2012 at 1:44 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1330422247150#c4228518952356608075)

**Neeru said...**

I was always curious on How HashMap stores object in Java or How HashMap is implemented in Java. I know about bucket and collision but What I missed was Map.Entry which is used to store both key and value object. HashMap stores Entry object which contains both key and value which then used in case of collision when more than one object is stored inside bucket to find out correct mapping. Correct me if this is not how hashmap stores objects in Java.

[February 29, 2012 at 8:20 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1330575640256#c8437249794114678500)

**Ski said...**

Isn't it Hashtable also works in similar way. Please let us know How Hashtable works in Java. I know about HashMap but now I am keen to know How Hashtable works internally.

[February 29, 2012 at 8:24 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1330575867915#c9146956774105255281)

**Anonymous said...**

Hi Javin,  
Nice artcile..nicely explained..it wud have been great it u cud explain the rehashing of Hashmap and the concurrent thread access in Hashmap.

[March 4, 2012 at 5:02 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1330909358218#c2985186614861377950)

**Anonymous said...**

Excellent article....  
But I would correct on one point.  
>What will happen if two different HashMap key objects have same hashcode?  
>They will be stored in same bucket but no next node of linked list. And keys equals () method will be used to identify correct key value pair in HashMap.  
  
The next node of the linked list will point to the next object in the same bucket.

[March 6, 2012 at 2:08 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1331028504173#c2625513590838142646)

**Anonymous said...**

I liked a post a lot .   
  
I am undertaking interviews and i know that these questions are really difficult to ans , inspite of working on hash map , we generally forget these details .  
Thanks

[April 3, 2012 at 2:38 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1333445886800#c4566818770965940957)

**[Ratnakar Reddy Katipally](https://www.blogger.com/profile/14837196487103031599)said...**

Thanks a lot. Pretty clear :) :)

[April 9, 2012 at 5:42 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1333975335238#c1917081882359544810)

**[Sriram](https://www.blogger.com/profile/11952691783244698157)said...**

Thanks for the post.Its really useful.  
  
1)Can you be more clear on using the final immutable objects as keys?   
  
2) what are the advantages of having final immutable objects as keys?   
  
3)Does they have any performance impact while getting the hashcode or while performing the equals method?  
  
I feel Collision should actually happen to improve the performance.Only then there will be a performance gain while searching for an object.If 1000 objects have 1000 different hashcodes, we will end up with 1000 hash buckets and each containing only one element.In that case, there wont be any use of hashmap and it's same as Arraylist.  
  
4)So, How do we make sure that multiple objects have the same hash code?  
  
5) Hash map should have unique keys right.So if we try to insert an object with the key that was already present in the HashMap, why does it not throw any exception.Instead it will replace that value object in the HashMap.Is there any reason for this?  
  
6) Why did they implement this key value pair? I mean Why cant it be just the value object thereby making the HashMap calculating the hascode of value and use the same while retrieving?If we dont want a key value pair and just want to improve search performance, then is it not this key value pair adding overhead to performance? If I want the same fucntionality as hashmap but dont need the key value pair, which collection do you suggest?  
  
Sorry for more questions on this.. I just thought to be very much clear on this HashMap concept.  
  
Please let me know if am not clear.

[May 9, 2012 at 10:33 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1336628039408#c9194277354420649031)

**[Yadu](https://www.blogger.com/profile/18388446465037858805)said...**

Nice explanation of internal logic of HashMap. But I think this focuses on pre Java 5.  
  
It uses HashMap.Entry for storing key-value not linked list. Very nice and clever use of Entry object.  
  
for (Entry e = table[i]; e != null; e = e.next) {  
Object k;  
if (e.hash == hash && ((k = e.key) == key || key.equals(k))) {  
V oldValue = e.value;  
e.value = value;  
e.recordAccess(this);  
return oldValue;  
}  
}

[May 24, 2012 at 9:45 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1337877903085#c6040040909935976394)

**Anonymous said...**

Excellent article and explanation. Got all the questions cleared. Thanks

[May 29, 2012 at 11:25 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1338359139115#c3618253319681673814)

**Anonymous said...**

Can you please explain "if race condition happens then you will end up with an infinite loop" bit more.  
  
Thank you for nice article

[May 31, 2012 at 2:42 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1338457328057#c3271885553837532937)

**Anonymous said...**

good writeup,, useful  
  
thanks-BIGB

[July 29, 2012 at 9:03 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1343577781102#c5125020977946563750)

**[Priyanka](https://www.blogger.com/profile/16410267249685393189)said...**

Awesome work, u have covered almost everything in hashmaps -one of the most difficult part in collection to understand. Keep up the good work!! I wish if you could include some examples in coding, that would make an interviewee like me more confident to express it in front of interviewer.

[July 30, 2012 at 9:08 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1343664491286#c837329709839876085)

**Prithvi said...**

is there any change in HashMap in Java 5 and Java 6, which affects internal working of HashMap in Java or HashMap in Java 5 is exactly same in HashMap in Java 6 or JDK 4 ?

[July 30, 2012 at 8:45 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1343706331547#c2252183590737002660)

**Anonymous said...**

please explain "if race condition happens then you will end up with an infinite loop" bit more.  
  
it is better if you provide simple example!:)

[August 1, 2012 at 7:38 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1343831891447#c7774458949511352968)

**Prateek said...**

One more question on Java HashMap:  
  
Question: why hashmap keys needs to be immutable ?  
Answer : HashMap keys needs to be immutable so that no one can modify key once created leaving its hashCode unchanged which is used to put and get object from HashMap. If HashMap key is not immutable than key can return hashCode 100 while putting object and after some modification hashCode changes to 102, means you can not get object back from HashMap.That's why hashmap keys needs to be immutable in Java

[August 10, 2012 at 7:19 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1344608364403#c7422527790092880992)

**Anonymous said...**

Diff between HashMap & Hashtable interms of algo for the key index will be  
  
HashMap  
Index = hashcode & (bucketsize) // bitwise   
  
Hashtable  
  
Index = (hashcode & 0x7FFFFFFF) % bucketsize // MOD op e.g 100(hashcode) % 10 (bucket size) = 0  
Index = 0

[August 21, 2012 at 8:03 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1345561412624#c8369049098555850554)

**[P-H](https://www.blogger.com/profile/15963955701683788257)said...**

Hi Javin,  
  
Your readers may be interested in a recent post on [HashMap vs. ConcurrentHashMap](http://javaeesupportpatterns.blogspot.com/2012/08/java-7-hashmap-vs-concurrenthashmap.html) from my Blog.  
  
Article was created following a recent production problem from one of my client Java EE environment. It compliments well your great post.  
  
A sample Java program is also provided on how to replicate this infinite loop condition for get() and put() operations.  
  
Enjoy.  
Regards,  
P-H

[August 29, 2012 at 4:07 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1346281657615#c7774531239650946081)

**[ketaki\_w](https://www.blogger.com/profile/05663152259937854260)said...**

Good one. But I have a question. How does an entry with null key is handled in Hashmap? I mean, one can't calculate hashcode on null so how could an entry with null key get a bucket?  
  
Suppose it is placed at 0 th bucket, but then if 0 th bucket has multiple entries in the linked list (the entries whose keys got hashcode 0 when calculated.) Then how is it resolved, as we can't even call equals method on null!

[August 30, 2012 at 12:15 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1346310931504#c3015570833929672931)

**[Sanjeev](https://www.blogger.com/profile/15497712476946085125)said...**

I case of hashCode collision, equals method is applied in same bucket with existing elements, if equals method return true, new value is applied on the old key value combination, if equals return false new entry is made with new key value in same bucket.  
--Sanjeev  
http://all-technicals.blogspot.in

[October 12, 2012 at 12:07 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1350068861393#c2203465069152863286)

**Anonymous said...**

do you see any problem with resizing of HashMap in Java  
I do not understand how the race condition will occur can you elaborate?

[December 5, 2012 at 12:22 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1354695769500#c5352295964095180166)

**Anonymous said...**

I really like all your posts, very informative!  
  
I did not understand how race condition will cause infinite loop in hashmap resizing...could you please explain?

[January 11, 2013 at 10:23 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1357971836399#c6836759974631138835)

**Anonymous said...**

We create two diff hashmap.  
while inseting records is there any chances of having same hashcode.

[February 13, 2013 at 3:45 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1360755935833#c9136541570886461814)

**Anonymous said...**

Hi I am being a big fan of you now. Just a little bit of problem.  
If storing hashmap in java can be explained in more detail. if some diagrams can be included.

[February 13, 2013 at 9:07 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1360818456829#c4800536166969429921)

**[Javin @ ClassLoader in Java](http://javarevisited.blogspot.com/2012/12/how-classloader-works-in-java.html)said...**

@Anonymous, Thanks. I would definitely try to include a diagram to explain internal working of HashMap in more details.

[February 14, 2013 at 4:35 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1360845319036#c8271883513087473570)

**Anonymous said...**

Hi  
  
You say that put() method does not use equals() method on key objects, is that right?   
  
See below example:  
  
HashMap hm = new HashMap();  
hm.put("a", "A");  
hm.put("a", "NOT A");  
  
In this example how does HashMap only returns "Not A" value.  
  
Thanks,

[March 8, 2013 at 11:45 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1362815139654#c5353651333688428932)

**Pattision said...**

@Anonymous, This is how I see,  
  
1) When you first put "a" as key and "A" as value, HashMap calls "a".hashCode(), which returns a bucket location inside HashMap. if there is no object there than Java will store an entry object which contains both key and value there. If there is collision than, a linked list will be created and an entry corresponding to "a" and "A" will be stored there.   
  
2. Now when you again insert same key, same bucket location will be found, and Java will search for an entry, whose value matches with "A". If found then it would be replaced.  
  
But, YES, Java HashMap uses equals method of key to find correct entry.  
  
Here is the code from java.util.HashMap which clarifies the doubt  
  
int hash = hash(key.hashCode());  
int i = indexFor(hash, table.length);  
for (Entry e = table[i]; e != null; e = e.next) {  
Object k;  
if (e.hash == hash && ((k = e.key) == key || key.equals(k))) {  
V oldValue = e.value;  
e.value = value;  
e.recordAccess(this);  
return oldValue;  
}  
}

[March 28, 2013 at 2:56 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1364464602129#c1839270309598678725)

**Anonymous said...**

I am currently in an entry-level Java class and we were asked to implement a Map interface using an ArrayList of LinkedLists without any prior experience with this topic. This article has been beyond helpful. Thank you, thank you, thank you!

[April 25, 2013 at 7:00 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1366941623794#c2405343861618625683)

**Anonymous said...**

Just to add here are few more interview question asked on java  
How does load factor and. Initial capacity affect performance of hashmap  
Why hashmap always use capacity as power of two?

[April 26, 2013 at 10:07 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1367039232830#c2172756342268578394)

**Anonymous said...**

@ketaki\_w,   
Hashmap allows only one entry with null key. It maintains a separate bucket for null key. When we try to insert any data, put method checks if the key is null. If the key is null, it will insert the entry into the separate bucket maintained for null key. Else, continues with normal logic. Below is the code snippet from the Hashmap source code.  
  
public V put(K key, V value) {  
if (key == null) {  
return putForNullKey(value);  
}  
... //Non-null key data insertion logic here.  
...  
}  
  
  
/\*\*  
\* Offloaded version of put for null keys  
\*/  
private V putForNullKey(V value) {  
for (Entry e = table[0]; e != null; e = e.next) {  
if (e.key == null) {  
V oldValue = e.value;  
e.value = value;  
e.recordAccess(this);  
return oldValue;  
}  
}  
modCount++;  
addEntry(0, null, value, 0);  
return null;  
}

[May 17, 2013 at 11:57 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1368860234227#c3681829473926566559)

**Prateek said...**

as per ur explanation of load factor , loadfactor is sumthing by which the collection is scaled up.

[June 12, 2013 at 11:22 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1371104537869#c6712948479710552623)

**[Ken](https://www.blogger.com/profile/08028651449316484899)said...**

@Anonymous  
HashMap always uses a power of two for its length to be able to calculate the index really fast (much faster than mod operation), doing a bitwise AND operation between the hashcode and length-1, which just returns the lesser bits. But for this, length has to be a power of 2.

[June 19, 2013 at 8:06 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1371654391949#c4674345545994382261)

**Anonymous said...**

Regarding this statement:  
  
important point to mention is that HashMap in Java stores both key and value object as Map.Entry in bucket which is essential to understand the retrieving logic  
  
Actually, each Map.Entry contains 4 items:  
- key  
- value  
- hashcode  
- next pointer to next Map.Entry in bucket list  
  
As the linked list of Map.Entry items in the "bucket slot" is traversed, the hashcode must match and either the address of the key or its hashcode must match:  
  
// e is the pointer to a Map.Entry  
if (e.hash == hash && ((k = e.key) == key || key.equals(k)))  
// found it  
return e.value  
  
So if two different keys happen to have the same hashcode, then the 2nd test (for key being at same (==) memory location or being equals()) will correctly find the correct key in the linked list.  
  
Finally, since the equals() test might take some finite computational time, the code in get() method is optimized to avoid calling equals() unless absolutely necessary. If the e.hash is wrong, there is no need to test for equality. But if the e.hash is correct, and if the address of the search object is the same as the address of the stored object in the Map.Entry, then the call to equals() also is avoided.  
  
The only time the equals() method would be called is if your key inserted with put() is different memory address from the key being looked up:  
  
// HashMap  
Key key = new Integer(1);  
map.put (key, "one");  
String value = get (new Integer(1) ); // differet address  
  
-dbednar

[June 28, 2013 at 3:43 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1372459383391#c2113527849444706521)

**Anonymous said...**

Maybe another good interview question MIGHT BE as follows:  
  
What problems could arise with HashMap if the key were mutable?  
  
Answer: If the key is immutable, then no problems will arise (since the hashcode cannot change). However, if the key is mutable, and is able to change the key in such a manner as to change the hashcode, then you might insert the key, but be unable to retrieve it later.  
  
The reason this can happen is that since the hashcode changed, the index into the bucket array used for the insertion (put(K,V)) might be different than the index for the lookup (get(K)). Since different linked lists are used, and the lookup will search the wrong linked list, and so will return null to mean not found (or null if null value is permitted).  
  
-dbednar

[June 28, 2013 at 3:55 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1372460126957#c149679772999200145)

**Anonymous said...**

Anonomous asked:  
> Hi Dude Can we use hashMap in multithreaded scenario ? is there any issue with that ?   
  
  
Yes, big problems. For thread-safety, its best to use ConcurrentHashMap, introduced in JDK 1.5.  
  
-dbednar

[June 28, 2013 at 4:00 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1372460401924#c8619080706679224748)

**[Javin @ get method of HashMap working](http://java67.blogspot.sg/2013/06/how-get-method-of-hashmap-or-hashtable-works-internally.html)said...**

Hi Volodymyr, You have raised a good point, HashMap implementation has some issue with String keys, because of it's hashCode implementation. Since HashMap uses linked list in case of collision, this can result in performance of O(n) instead of O(1). A Crafted set of String keys can be used in Denial of Service (DOS) attack, where an application server like Tomcat or JBOSS build Hashtable or HashMap with untrusted input retrieved from HTTP request parameter. Thankfully, Java 8 is fixing this HashMap bug, by replacing linked list with balanced tree, once number of element in linked list crossed a threshold. This fix can improve performance from O(n) to O(logN) in case of such collisions. This can also reduce such denial of service attack, which can hamper your server badly.

[July 27, 2013 at 2:35 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1374917728070#c2384738869976429232)

**[Basavaraj Balte](https://www.blogger.com/profile/12214426021728337995)said...**

Hi,  
This is Wonderful blog...,i am addicted to it.  
  
Thanks

[August 13, 2013 at 12:22 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1376378549261#c8194491109207212937)

**[Rama Raju Vatsavai](https://www.blogger.com/profile/14721308226265276883)said...**

Hey Javin,  
  
After reading this article for several times, I have a question. In case of conflicts i.e when two hashcodes are equal, key's equal method is used in the bucket to identify the exact key/value pair. However, doesn't key.equal again use the hashcode() method to compare, which would result in same hashcode() again?

[August 22, 2013 at 3:04 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1377209097214#c1255066605514299337)

**[Tao Chen](https://www.blogger.com/profile/05246414664017962427)said...**

Hi,Javin,  
I am confused by this paragraph  
"What will happen if two different HashMap key objects have same hashcode?  
They will be stored in same bucket but no next node of linked list. And keys equals () method will be used to identify correct key value pair in HashMap ."  
From above, I conclude if two keys have same hashcode, they would be put in same bucket and chained in a linked list. And even two different keys have different hashcode, they still might be put in same bucket and chained in a linked list.  
Is there something wrong in my understanding?

[October 3, 2013 at 11:36 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1380825400586#c8266601465926698622)

**[Nitin Jain](https://www.blogger.com/profile/16938280756747352676)said...**

HI http://java67.blogspot.sg/2013/06/how-get-method-of-hashmap-or-hashtable-works-internally.html,  
  
I am adding some few question on Hashmap asked by interviewer. Please post these ans in your blog it will be helpful to others.  
  
1. Can hashMap store 2 Diff value with same key?  
2. Hashcode and Equals method overriding is compulsory for HasMap and why?  
3. What is importance for overriding hascode and equals method, Key or value or both and how?

[January 15, 2014 at 4:53 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1389790419757#c8549791747193929033)

**[Ankit Tyagi](https://www.blogger.com/profile/14491645590262855297)said...**

Hi All,   
  
I have one small doubt, In case of collision while storing data in to the memory. Is one bucket location contains multiply Entry Object for same bucket location. I am really not able to visualize this.

[January 24, 2014 at 12:44 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1390553064699#c4100084898945667632)

**Anonymous said...**

This is really good. I am preparing for an investment bank. Hope they ask me and I ll burst it out! :D

[April 17, 2014 at 7:10 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1397743826593#c531367493741605503)

**[Ganesh Rashinker](https://www.blogger.com/profile/12908235787232547188)said...**

On February 7, 2011 at 3:42 AM , Anonymous asked   
Can we use hashMap in multithreaded scenario ? is there any issue with that ?  
I have explained the same using a practical example on my blog.  
Please check this blog for a demo of what I have mentioned above. Please correct me if my understanding is wrong somewhere.  
  
http://ganeshrashinker.blogspot.in/

[April 23, 2014 at 7:58 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1398308325814#c527585792615709252)

**Anonymous said...**

hi,  
from this articale i understood that hashmap allow null value,so when i store a value which has null as   
key then how hashing will takes place and what hashcode generated for null values.  
2.can any one please explain in which case that hashing of two key generate the same hashcode?  
3.does it appropriate hashmapobj.get(null) when i store a value that associate null as key?

[May 31, 2014 at 10:47 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1401601630634#c1179036681002495953)

**[Dheeraj Sinha](https://www.blogger.com/profile/18006157117430568013)said...**

Hi,  
  
As per the article "HashMap doesn't append the new element at tail instead it append new element at head to avoid tail traversing" . But anyway it get traversed to check same key , so what do you mean by tail traversing here?

[June 24, 2014 at 3:54 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1403607255479#c4255118553680096204)

**[Ramesh Kumar](https://www.blogger.com/profile/01390467032024724628)said...**

Hi   
How can you create own hash map with out using java api

[June 30, 2014 at 6:11 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1404133880067#c4609008074804508005)

**Anonymous said...**

Does hashmap always uses LinkedList for storing??   
Why to use HashMap?

[July 15, 2014 at 1:22 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1405455775506#c1989257767911088075)

**Anonymous said...**

HashMap in java 1.7 does not handle Collisions. It u try to add a duplicate key, it will get replaced by the new key. Can somebody confirm ?

[July 22, 2014 at 12:19 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1406013577875#c8300106416252877249)

**[Ajay Mehra](https://www.blogger.com/profile/04245212606929139681)said...**

Thanks for explaining in simple language..  
My question is regarding size of HashMap.  
1.When will rehashing be done?.   
suppose 16 is the initial size and i have added 12 ie. 16\*0.75 element into hashmap. is it irrespective of element being added to same or different bucket?  
2. cane we set default size? if yes than how?.

[August 5, 2014 at 7:55 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1407293705366#c516569193802166619)

**[Alen Jain](https://www.blogger.com/profile/15211567207099447748)said...**

While rehashing, if two thread at the same time found that now HashMap needs resizing and they both try to resizing. on the process of resizing of HashMap, the element in bucket which is stored in linked list get reversed in order during there migration to new bucket because HashMap doesn't append the new element at tail instead it append new element at head to avoid tail traversing. If race condition happens then you will end up with an infinite loop.   
  
Please explain above paragraph. I failed to understand below points -  
1. Why list is getting reversed while rehashing. I mean what advantage we got ?  
2. Why elements are getting added at front end of list ? Why not tail end ?

[August 17, 2014 at 10:41 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1408340507951#c5568151091834103150)

**Anonymous said...**

do you know whether HashMap is safe from DOS attack or not? What happen if hash function return same value for multiple keys? If yes, how do you fix that, how do you make sure that it doesn't perform on O(n) level?

[August 26, 2014 at 3:01 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1409047281995#c1761604039362301099)

**[THE CENTAUR](https://www.blogger.com/profile/09663141030617383221)said...**

Just to add my 2 cents,  
HashMap now(i.e. JDK-8) uses balanced tree during collision. http://openjdk.java.net/jeps/180

[September 28, 2014 at 8:56 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1411919796872#c6494731234363766603)

**[aman agarwal](https://www.blogger.com/profile/17814275974972262333)said...**

What a brilliant collection...  
Well i dont know if people will really have patience to come down here and read this comment, and i also dont know if this question has already been covered, here's what was asked from me:  
When you create a hashmap object using Hasmap h = new hashmap(), the constructor here can accept arguments.These can be upto 2. what are these arguments meant for?  
  
I pretty much didnt know the answer then. The answer is:  
HashMap()  
Constructs an empty HashMap with the default initial capacity (16) and the default load factor (0.75).  
HashMap(int initialCapacity)  
Constructs an empty HashMap with the specified initial capacity and the default load factor (0.75).  
HashMap(int initialCapacity, float loadFactor)  
Constructs an empty HashMap with the specified initial capacity and load factor.  
  
Sometimes interviews do make you explore.. Thanks to the interviewer!!! :)

[October 18, 2014 at 8:59 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1413647944773#c5637613139581036508)

**Anonymous said...**

@Solutions,  
while calling get()method  
  
1) hashCode of key object is used to find bucket location in HashMap.  
2) if in bucked location only one object is there than its value object. else if bucket location contains LinkedList than traverse through each node and compare key object stored as tuple along-with value object to key object passed to get() method by using equals() method. if matches than return the value object. point to note was that instead of just storing value object, LinkedList node on HashMap also stores Key object.

[December 17, 2014 at 7:32 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1418830373792#c977415449685343451)

**Anonymous said...**

@Sriram,  
HashMap key object must be immutable in order to be used as key because hashCode of key is used to find the bucket location inside HashMap and retrieve object. if Key object is mutable than any one can change its value after putting object in HashMap and than calling hashCode() on modified key object would have produce different hashcode and consequently different bucket location and you will never be able to retrieve object from HashMap. hope this would be clear

[December 17, 2014 at 7:33 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1418830424319#c4784736628655556828)

**Anonymous said...**

Hello!  
  
I also wanted to point out of Integer as a key.Integer's hashCode is value you provide instantiating Integer. Therefore If you try to put two mappings with keys as Integers and those Integers hold the same value the second mapping will replace the first one because keys will be considered equal with identical hashCode

[December 17, 2014 at 7:38 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1418830736607#c7924042527970062104)

**Anonymous said...**

String is not considered good key for HashMap. Because of weak hash-code calculation of String !!!  
If you want to know more about internal HashMap read my tutorial Internal life of HashMap in Java.

[December 18, 2014 at 5:27 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1418909268026#c7333482842544213702)

**Anonymous said...**

When I put null into map as map.put(null,"demo") and later do operation for getting value "demo" by passing null as key, How does it work? Since on hashmap firstly key of hashcode is used to find bucket and later equals() is used to find the value but what in case of null? how equals and hashcode methode work in this case ?

[January 4, 2015 at 8:21 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1420431688444#c4025550530072547037)

**[Javin Paul](https://www.blogger.com/profile/15028902221295732276)said...**

Hello Anonymous, I have added another section to explain how null key is stored and retrieved in HashMap. When you call map.put(null, "demo") then corresponding entry is stored in first location in bucket array i.e. index 0. when you call get(null) value is retrieved from that location. Two special method putForNullKey() and getForNullKey() are used to put and get null key from HashMap.

[January 4, 2015 at 8:35 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1420432557087#c4022240987770951438)

**Anonymous said...**

Do you know Why HashMap stores both keys and values in Entry? Why not just store values only?

[January 5, 2015 at 8:46 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1420519581806#c3280847271117462259)

**Saurabh said...**

There are multiple ways to resolve collision in hash table. HahsMap in Java i.e.java.util.HashMap uses chaining to resolve collision. Another way to resolve hash collision is by using "Open addressing". If you are going for Java interview, you better know both of them

[January 5, 2015 at 8:48 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1420519699471#c7753854577496580001)

**Saurabh said...**

@Anonymous, HahsMap stores both key and value inside Entry object because it's possible to store more than one value in one bucket location, in that case in order to find the correct value corresponding to key, you also need the key object there. this is exactly where equals() method comes into play.

[January 5, 2015 at 8:50 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1420519816739#c6732899279121938884)

**Anonymous said...**

Please explain how remove(key) method internally works on HashMap??  
  
This was one of my interview questions.

[January 20, 2015 at 3:02 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1421751724332#c4893374104209590508)

**Anonymous said...**

Can you write a similar article on "How treemap works using Red Black ago"?

[March 29, 2015 at 10:41 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1427650864130#c7081453150430149965)

**Anonymous said...**

Hi Javin,  
  
This is a nice article. But I have one question here   
  
Why there is only one null key allowed in Hashmap.Why can't we insert many

[April 13, 2015 at 7:55 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1428936941609#c6895727770752529478)

**[Javin Paul](https://www.blogger.com/profile/15028902221295732276)said...**

@Anonymous, null handling is very special, I would suggest to check the section about how HashMap handles null keys. To give you answer, multiple null is not an option because HashMap doesn't allow duplicate keys.

[April 13, 2015 at 8:53 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1428940424872#c7867697451174201056)

**Thennam said...**

Hi Javin,  
  
Could you please explain how hava 8 uses balanced tree to solve the collision issue.  
How they will create the balanced tree, as keys has same hash code

[May 10, 2015 at 11:34 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1431326058777#c7174103512625222034)

**[vikas mishra](https://www.blogger.com/profile/14156666665112079587)said...**

race condition in hashmap with picture   
http://mailinator.blogspot.in/2009/06/beautiful-race-condition.html

[June 13, 2015 at 1:47 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1434228430611#c6506115732554711968)

**Anonymous said...**

Hi,  
  
As per HashMap implementation, HashMap resizes its internal table size when number of elements are >= Load factor \* Initial Capacity i.e, 12  
  
To test this I debugged below code:  
public class HashMapTest {  
  
public static void main(final String[] args) {  
Map test = new HashMap();  
System.out.println("Initial Default HashMap size : -" + test.size());  
for (int i = 0; i < 16; i++) {  
test.put(i + "", i + "");  
}  
System.out.println("HashMap size After adding 11 Elements: -" + test.size());  
test.put("11", "11");  
System.out.println("HashMap size After adding 11 Elements: -" + test.size());  
test.put("12", "12"); //13th Element  
System.out.println("HashMap size After adding 11 Elements: -" + test.size());  
}  
}  
  
In above code it should resized the Table size while adding 13th element but I found table size to be 16 instead of 32.  
  
Can anyone explain me please ?? Thanx

[July 2, 2015 at 5:37 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1435840672290#c871686703552733538)

**[Atul Pendse](https://www.blogger.com/profile/06249680252862615440)said...**

@Anonymous -   
test.size() always returns the actual count of keys held by the HashMap, no matter what is size of the internal table. Size of internal table is always higher than number of entries in the HashMap, and is expanded once number of entries reach certain threshold, which is determined by the load factor. Higher size of internal table provides capacity to accommodate addition of newer entries without having to rehash the entire table. Without any additional capacity in the table, hash-values of each entry has to be recalculated every time a new entry is added.

[July 16, 2015 at 2:07 AM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1437037662650#c6571937332079959982)

**Anonymous said...**

I like your post. You explained it very well.  
For those who are not clear on race condition which may cause infinite loop in Hashmap: http://javabypatel.blogspot.in/2016/01/infinite-loop-in-hashmap.html

[January 3, 2016 at 8:07 PM](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html?showComment=1451880446900#c8891673898699767839)