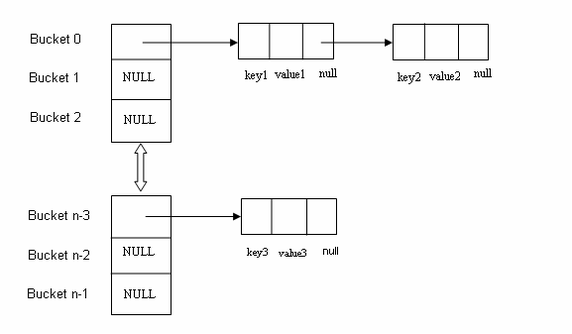
# 1)What is the internal implementation of HashMap?

There are four things we should know about before going into internals of how HashMap works -

* **HashMap** works on the principal of hashing.
* **Map.Entry interface** - This interface gives a map entry (key-value pair). HashMap in Java stores both key and value object, in bucket, as Entry object which implements this nested interface Map.Entry.
* **hashCode()** -HashMap provides put(key, value) for **storing** and get(key) method for**retrieving** Values from HashMap. When put() method is used to store (Key, Value) pair, HashMap implementation **calls hashcode** on Key object to calculate a hash that is used to find a bucket where Entry object will be stored. When get() method is used to retrieve value, again key object is used to calculate a hash which is used then to find a bucket where that particular key is stored.
* **equals()** - equals() method is used to **compare objects for equality**. In case of HashMap key object is used for comparison, also using equals() method Map knows how to handle **hashing collision** (hashing collision means more than one key having the same hash value, thus assigned to the same bucket. In that case objects are stored in a linked list.  
  Where hashCode method helps in finding the bucket where that key is stored, equals method helps in finding the right key as there may be more than one key-value pair stored in a single bucket.



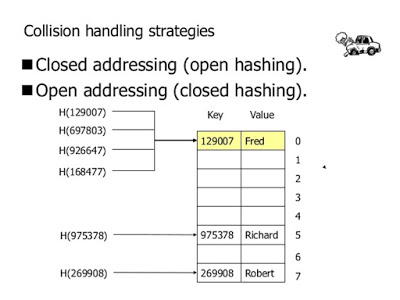
# 2) How does get(Key key) method works internally in HashMap, and Hashtable in Java?

Here are steps, which happens, when you call get() method with key object to retrieve corresponding value from hash based collection

a) Key.hashCode() method is used to find the bucket location in backing array. (Remember HashMap is backed by array in Java) Though hashcode() is not used directly, but they are passed to internal hash() function.

b) In backing array or better known as the bucket, key and values are stored in the form of a nested class called Entry.  If there is only one Entry at bucket location, then the value from that entry is returned. Pretty straightforward right?

Things get little tricky, when Interviewer ask the second question, **What happens if two keys have the same hashCode?** If multiple keys have the same hashCode, then during put() operation collision had occurred, which means multiple Entry objects stored in a bucket location. Each Entry keeps track of another Entry, forming a [linked list data structure](http://javarevisited.blogspot.com/2013/05/find-if-linked-list-contains-loops-cycle-cyclic-circular-check.html) there.

[](https://3.bp.blogspot.com/-D87sspTS5EI/V0R01S0iP4I/AAAAAAAAF_8/jZtoF1219Iw4ByOyDcO8-qoFFNqrkeJ4QCLcB/s1600/Collision+handling+strategies+in+HashMap+Java.jpg)

This has also changed from Java 8, where after a threshold is crossed then a binary tree is used instead of linked list to lift the worst case performance from O(n) to O(logN). You can see [how HashMap and LinkedHashMap handle collision in Java](http://javarevisited.blogspot.com/2016/01/how-does-java-hashmap-or-linkedhahsmap-handles.html) to learn more about this change.   
  
Now, if we need to retrieve value object in this situation, following steps will be followed :

1) Call hashCode() method of the key to finding bucket location.

2) Traverse thought linked list, comparing keys in each entries using keys.equals() until it returns true.

So, we use equals() method of a key object to find correct entry and then return value from that. Remember key.equals() method, and this is what Interviewer want to know. I have seen many programmers mentioning value.equals(), which may be due to interview nervousness, but that’s incorrect. Since you don't have value object passed to get()method, there is no question of calling equals and hashCode method on value object.

That's all on these two HashMap questions guys. Remember to mention about key.hashCode() and key.equals(), whenever someone asks **how to get method of HashMap or Hashtable works in Java**. A value object is not used, it just exists in Entry so that get can return it.

# 3)How HashSet Internally Works in Java

Not many programmer know that HashSet is internally implemented using HashMap in Java, so if you know How HashMap works internally in Java, more likely you can figure out how HashSet works in Java. But, now a curious Java developer can question that, how come HashSet uses HashMap, because you need a key value pair to use with Map, while in HashSet we only store one object. Good question, isn't it? If you remember some functionality of earlier class, then you know that HashMap allows duplicate values and this property is exploited while implementing HashSet in Java. Since HashSet implements Set interface it needs to guarantee uniqueness and this is achieved by storing elements as keys with same value always. Things gets clear by checking HashSet.java from JDK source code. All you need to look at is, how elements are stored in HashSet and how they are retrieved from HashSet. Since HashSet doesn't provide any direct method for retrieving object e.g. get(Key key) from HashMap or get(int index) from List, only way to get object from HashSet is via Iterator. See here for code example of iterating over HashSet in Java. When you create an object of HashSet in Java, it internally create instance of backup Map with default initial capacity 16 and default load factor 0.75 as shown below :

/\*\*

\* Constructs a new, empty set; the backing <tt>HashMap</tt> instance has

\* default initial capacity (16) and load factor (0.75).

\*/

public HashSet() {

map = new HashMap<>();

}

Now let's see the code for add() and iterate() method from java.util.HashSet in Java to understand how HashSet works internally in Java.

How Object is stored in HashSet

As you can see below, a call to add(Object) is delegate to put(Key, Value) internally, where Key is the object you have passed and value is another object, called PRESENT, which is a constant in java.util.HashSet as shown below :

private transient HashMap<E,Object> 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;

}

Since PRESENT is a constant, for all keys we have same value in backup HashMap called map.

How Object is retrieved from HashSet

Now let's see the code for getting iterator for traversing over HashSet in Java. iterator() method from java.util.HashSet class returns iterator for backup Map returned by map.keySet().iterator() method. /\*\*

\* Returns an iterator over the elements in this set. The elements

\* are returned in no particular order.

\*

\* @return an Iterator over the elements in this set

\* @see ConcurrentModificationException

\*/

public Iterator<E> iterator() {

return map.keySet().iterator();

}

How to use HashSet in Java

Using HashSet in Java is very simple, don't think it is Map but think more like Collection i.e. add elements by using add() method, check its return value to see if object already existed in HashSet or not. Similarly use iterator for retrieving element from HashSet in Java. You can also use contains() method to check if any object already exists in HashSet or not. This method use equals() method for comparing object for matching. You can also use remove() method to remove object from HashSet. Since element of HashSet is used as key in backup HashMap, they must implement equals() and hashCode() method. Immutability is not requirement but if its immutable then you can assume that object will not be changed during its stay on set. Following example demonstrate basic usage of HashSet in Java, for more advanced example, you can check this tutorial.

import java.util.HashSet;

import java.util.Iterator;

/\*\*

\* Java Program to demonstrate How HashSet works internally in Java.

\* @author http://java67.blogspot.com

\*/

public class HashSetDemo{

public static void main(String args[]) {

HashSet<String> supportedCurrencies = new HashSet<String>();

// adding object into HashSet, this will be translated to put() calls

supportedCurrencies.add("USD");

supportedCurrencies.add("EUR");

supportedCurrencies.add("JPY");

supportedCurrencies.add("GBP");

supportedCurrencies.add("INR");

supportedCurrencies.add("CAD");

// retrieving object from HashSet

Iterator<String> itr = supportedCurrencies.iterator();

while(itr.hasNext()){

System.out.println(itr.next());

}

}

}

# Output

JPY

EUR

INR

USD

CAD

GBP

# 4.How to read a file from one location and print characters from that file?

public class ListFiles {

static File mainFolder = new File("D:\\Movies");

public static void main(String[] args)

{

ListFiles lf = new ListFiles();

lf.getFiles(lf.mainFolder);

long fileSize = mainFolder.length();

System.out.println("mainFolder size in bytes is: " + fileSize);

System.out.println("File size in KB is : " + (double)fileSize/1024);

System.out.println("File size in MB is :" + (double)fileSize/(1024\*1024));

}

public void getFiles(File f){

File files[];

if(f.isFile())

System.out.println(f.getAbsolutePath());

else{

files = f.listFiles();

for (int i = 0; i < files.length; i++) {

getFiles(files[i]);

}

}

}

}

