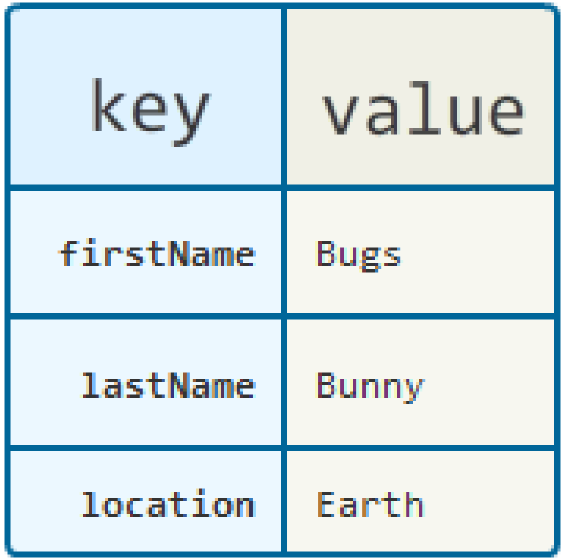
Hashmap Internal Working in Java

Introduction

HashMap is a part of Java’s collection since Java 1.2. It provides the basic implementation of Map interface of Java. It stores the data in (Key, Value) pairs.



key	value
firstName	Bugs
lastName	Bunny
location	Earth

Equals and hashcode methods

In order to understand the internal working of HashMap, we must be aware of **hashcode** and **equals** method.

equals(Object otherObject) – As method name suggests, it is **used to simply verify the equality of two objects.**

hashcode() – is a integer code generated for any variable/object after applying a formula/algorithm on its properties. The hash code for a String object is computed as

`s[0]*31^(n-1) + s[1]*31^(n-2) + ... + s[n-1],` where

`s[i]` **is the ith character** of the string, **n is the length** of the string.

What’s the output?

```
public static void main(String[] args) {
        String s1 = "Compassites";
        String s2 = "Software";
        System.out.println( s1.equals(s2) );
        System.out.println("Compassites  " + s1.hashCode() );
        System.out.println("Ea  " + s2.hashCode() );
}
```

It Produces the following output.

```
false
Compassites hashCode: 1297712907
Software hasCode: 1383974343
```

What about this?

```
public static void main(String[] args) {
        String s1 = "FB";
        String s2 = "Ea";
        System.out.println( s1.equals(s2) );
        System.out.println("FB hashcode  " + s1.hashCode() );
        System.out.println("Ea hashcode  " + s2.hashCode() );
}
```

What about this?

```
public static void main(String[] args) {
        String s1 = "FB";
        String s2 = "Ea";
        System.out.println( s1.equals(s2) );
        System.out.println("Compassites  " + s1.hashCode() );
        System.out.println("Ea  " + s2.hashCode() );
}
```

It Produces the following output.

```
false
FB hashcode 2236
Ea hashcode 2236
```

You can see that the hashcodes are same for “FB” and “Ea”. So we can conclude that...

Two Rules:

1. If two objects are equal then they must have the same hashcode. (hascode of “aa” and “aa” are same)
2. If two objects have the same hashcode, they may or may not be equal. (string “FB” has hashcode 2236 and “Ea” has hashcode 2236 bu they are not equal)

To make it 2nd point clear, we can say that

“Birthday as HashCode”

Now that we are aware of Hashcode and Equals method, let’s learn the internal working of Hashmap

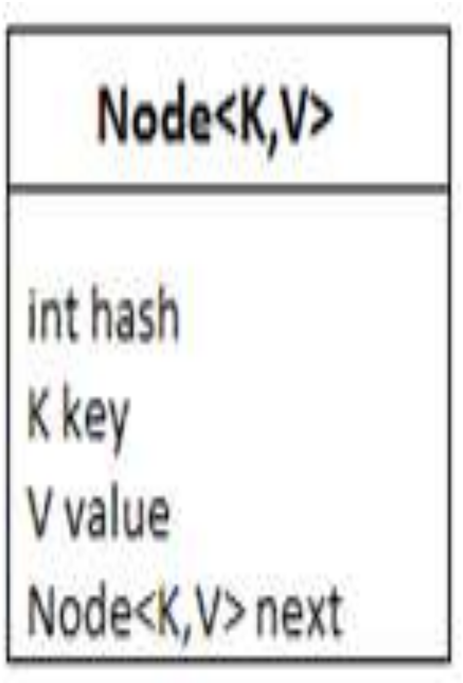
Hashing?

HashMap is known as HashMap because it uses a technique called Hashing.

Hashing is a technique of **converting a large String to small String** that represents same String.

A shorter value helps in indexing and faster searches.

Internally HashMap contains an array of Node and a **node is represented as a class** which contains 4 fields :

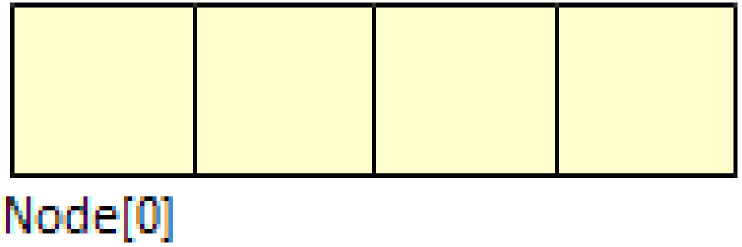


Node<K,V>

int hash
K key
V value
Node<K,V> next

It can be seen that **node is containing a reference of its own object. So it's a linked list.**

Hashmap:



Few Terms...

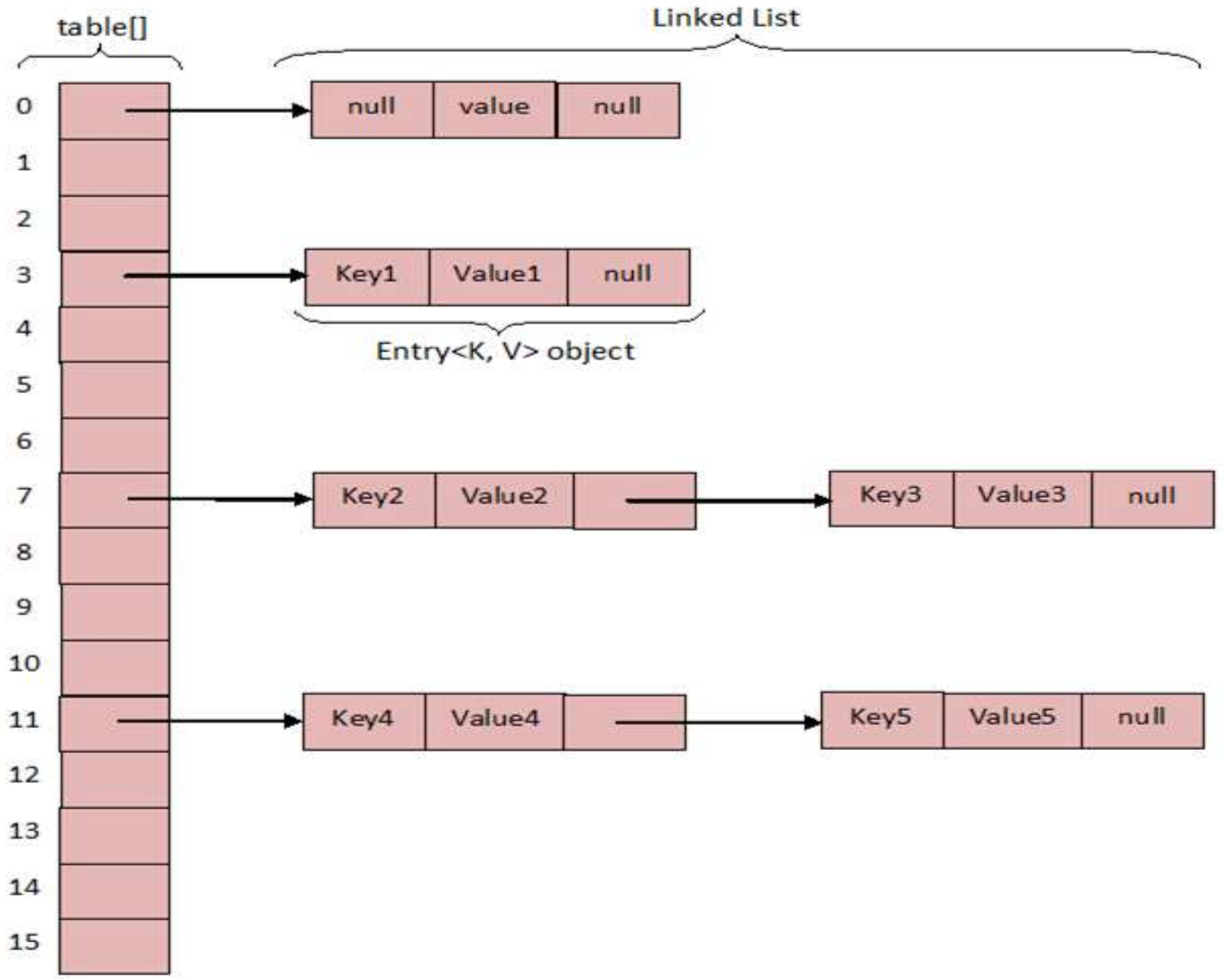
Buckets : A bucket is one element of HashMap array. It is used to store nodes. Two or more nodes can have the same bucket. In that case link list structure is used to connect the nodes.

Load Factor is a measure, which decides when exactly to increase the hashmap capacity(buckets).

HashMap() : It is the default constructor which creates an instance of HashMap with **initial capacity 16 and load factor of 0.75. (meaning capacity is doubled when 75% of hashmap is filled)**

HashMap(int initialCapacity, float loadFactor) : It creates a HashMap instance with specified initial capacity and specified load factor.

.


table[]
Linked List
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
null
value
null
Key1
Value1
null
Entry<K, V> object
Key2
Value2
Key3
Value3
null
Key4
Value4
Key5
Value5
null

***Put Operation in HashMap*:**

Say our program is something like this

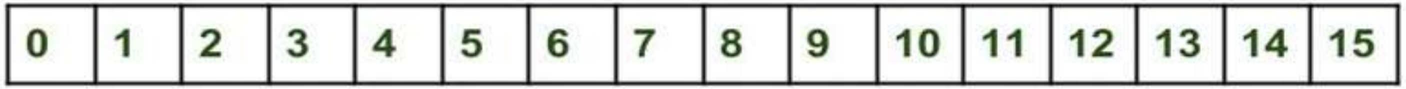
```
HashMap<String, Integer> map = new HashMap<>();
          scores.put (“Rohit”, 140);
          scores.put (“Dinesh”, 70);
          scores.put (“Dhoni”, 90);
          scores.put (“Kholi”, 100);
          scores.put (“Sachin”, 150);
          scores.put (“Dravid”, 130);
```

Let’s see what is happening internally...

Initially Empty hashMap:

Here, the hashmap’s size is 16.(default hashmapsize)

```
HashMap map = new HashMap();
```



Inserting Key-Value Pair:

Let us understand at which location below key value pair will be saved into HashMap.

scores.put (“Rohit”, 140);

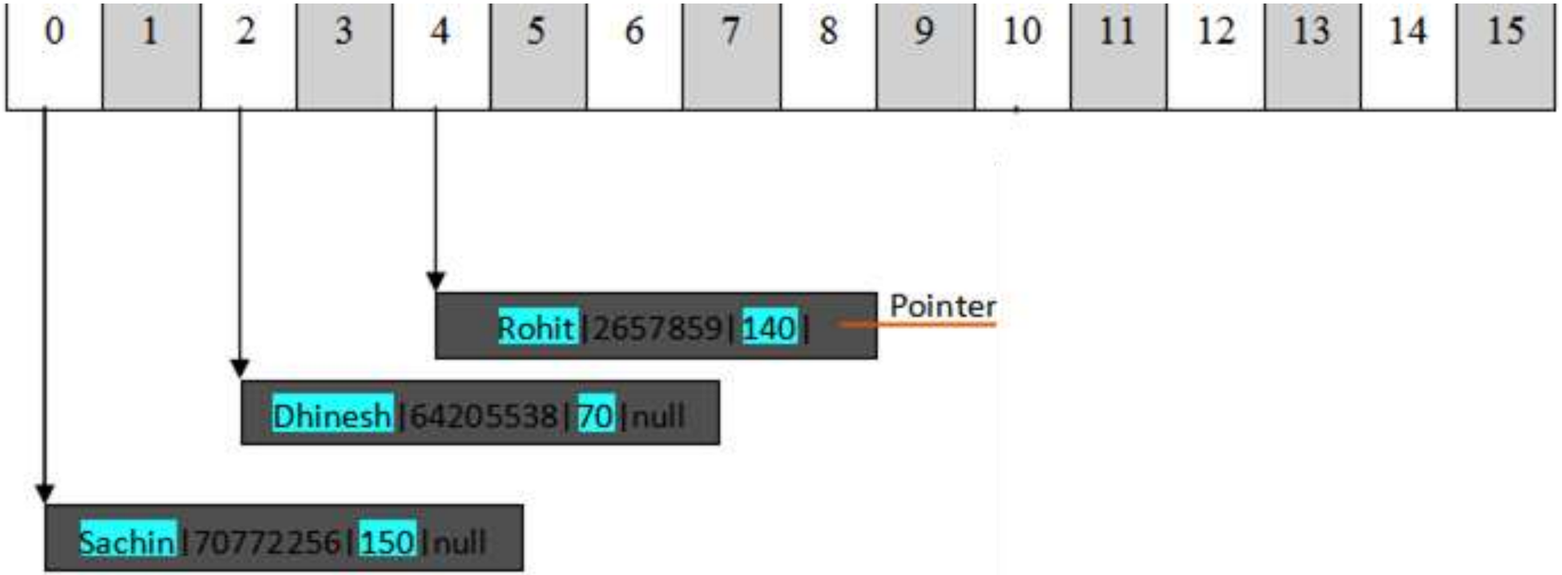
When you call the put function then it computes the hash code of the Key.

Lets suppose the Hash code of (“Rohit”) is **2657860**.

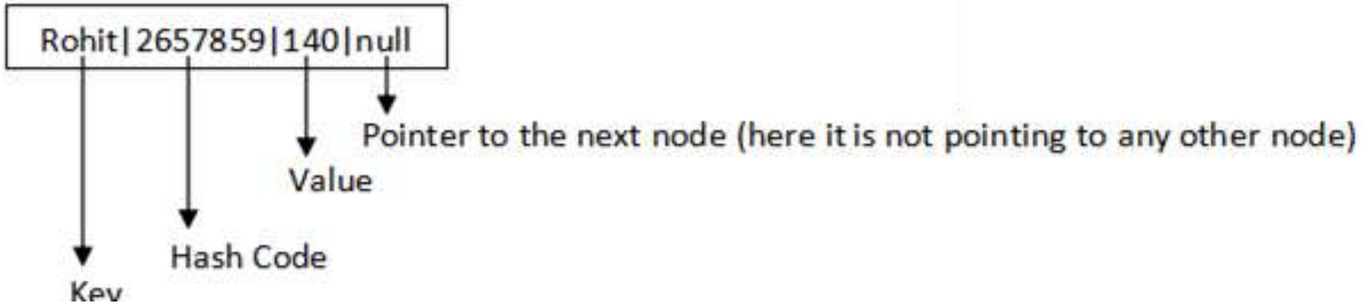
Our array has an index till 15, so in order to store “Rohit”, we have to calculate index using a modular operation

Index = 2657860 % 16 => 4

Index = 4, So 4 is the computed bucket index value where the entry will sit as a node in the HashMap.



What does a Node in a HashTable consists of?



Rohit|2657859|140|null
Pointer to the next node (here it is not pointing to any other node)
Value
Hash Code
Key

Hash Collision

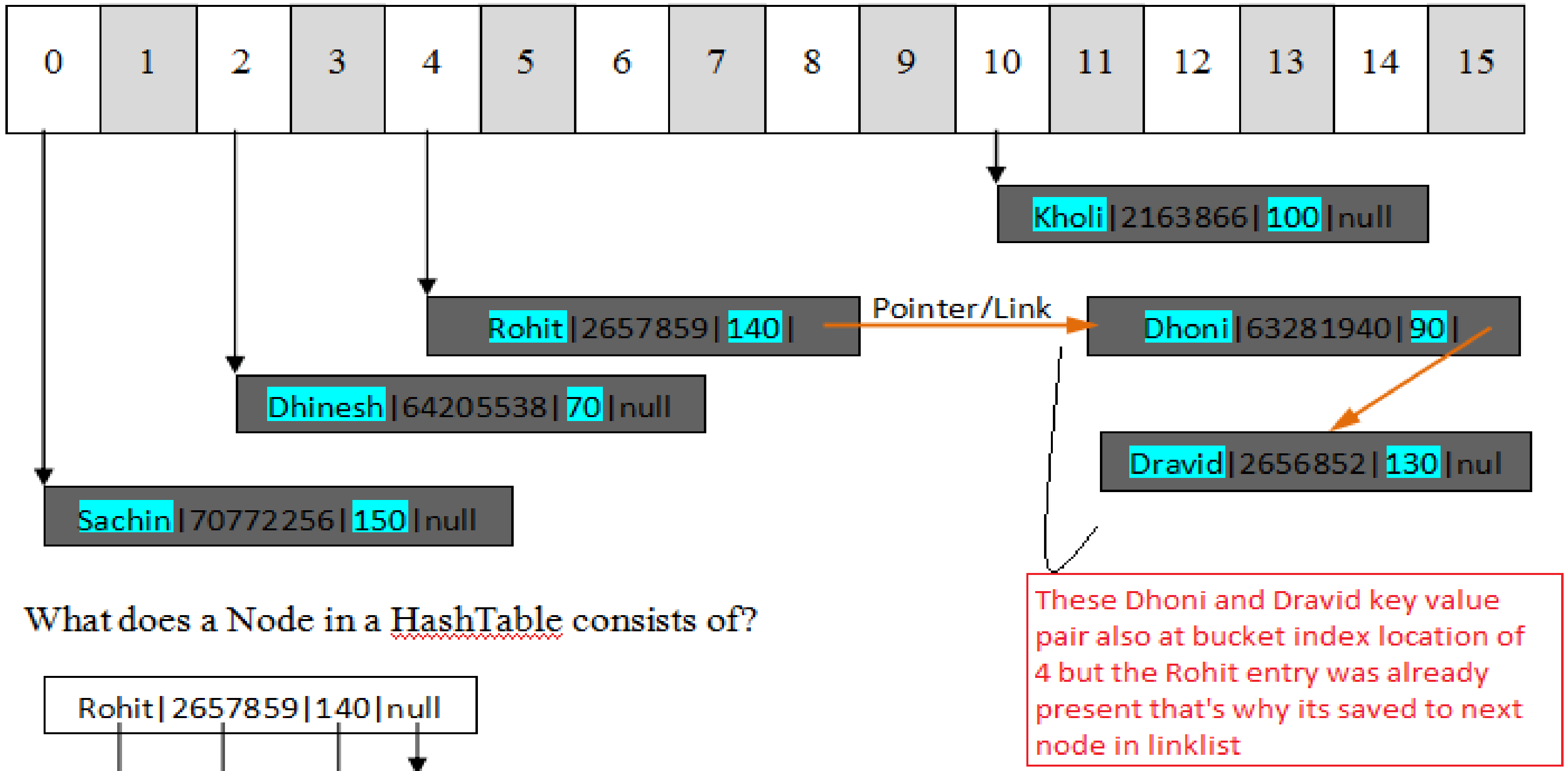
Let us understand at which location below key value pair will be saved into HashMap.

scores.put (“Dhoni”, 90);

When you call the put function then it computes the hash code of the Key.
Lets suppose the **Hash code of (“Dhoni”) is 63281940**.

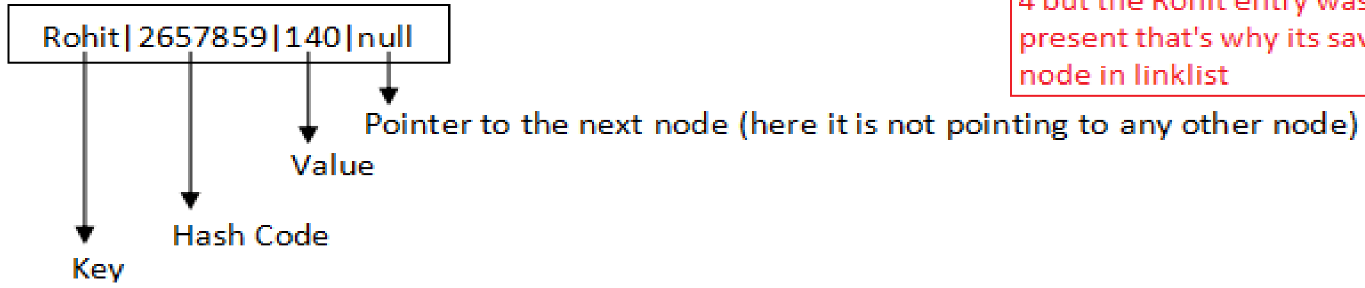
Our array has an index till 15, so in order to store “Rohit”, we have to calculate index using a modular operation

Index = **63281940** % 16 => 4



These Dhoni and Dravid key value pair also at bucket index location of 4 but the Rohit entry was already present that's why its saved to next node in linklist

What does a Node in a HashTable consists of?


Rohit|2657859|140|null
Pointer to the next node (here it is not pointing to any other node)
Value
Hash Code
Key

Get Operation in HashMap:

int rohitScore = **scores.get (“Rohit”);**

So get operation does the same as that of put operation. When the get function is called it basically gets the hash code of the Key.
Lets suppose Hash of (“Rohit”) is 2657860

Index = **2657860** % **16** => 4

Now hashMap lookups at bucket index 4 for the hash code of the key “2657860”.
Hash code “2657860” found then it lookup for the Key “Rohit” itself in that node or linked list.

Thank you :)