Hashmap And Hashset

layout: post title: "Java HashMap "
Hashing is a technique that is used to uniquely identify a specific object from a group of similar objects. Some examples of how hashing is used in our lives include: In universities, each student is assigned a unique roll number that can be used to retrieve information about them. In libraries, each book is assigned a unique number that can be used to determine information about the book, such as its exact position in the library or the users it has been issued to etc. In both these examples the students and books were hashed to a unique number.
Assume that you have an object and you want to assign a key to it to make searching easy. To store the key/value pair, you can use a simple array like a data structure where keys (integers) can be used directly as an index to store values. However, in cases where the keys are large and cannot be used directly as an index, you should use hashing.
In hashing, large keys are converted into small keys by using hash functions. The values are then stored in a data structure called hash table. The idea of hashing is to distribute entries (key/value pairs) uniformly across an array. Each element is assigned a key (converted key). By using that key you can access the element in O(1) time. Using the key, the algorithm (hash function) computes an index that suggests where an entry can be found or inserted.
Hashing is implemented in two steps:

1. An element is converted into an integer by using a hash function. This element can be used as an index to store the original element, which falls into the hash table.
2. The element is stored in the hash table where it can be quickly retrieved using hashed key.

```
hash = hashfunc(key)
index = hash % array_size
```

In second method, the hash is independent of the array size and it is then reduced to an index (a number between 0 and array_size − 1) by using the modulo operator (%). We will focus on second mathod.

Hashing

Hashing is a function which generates a unique value using some algorithms and functions. The first and foremost rule of hashing function is it has to generate the same hash always for the same value.The inbuild function in java for hashing is hashCode().
Example of hashCode() :

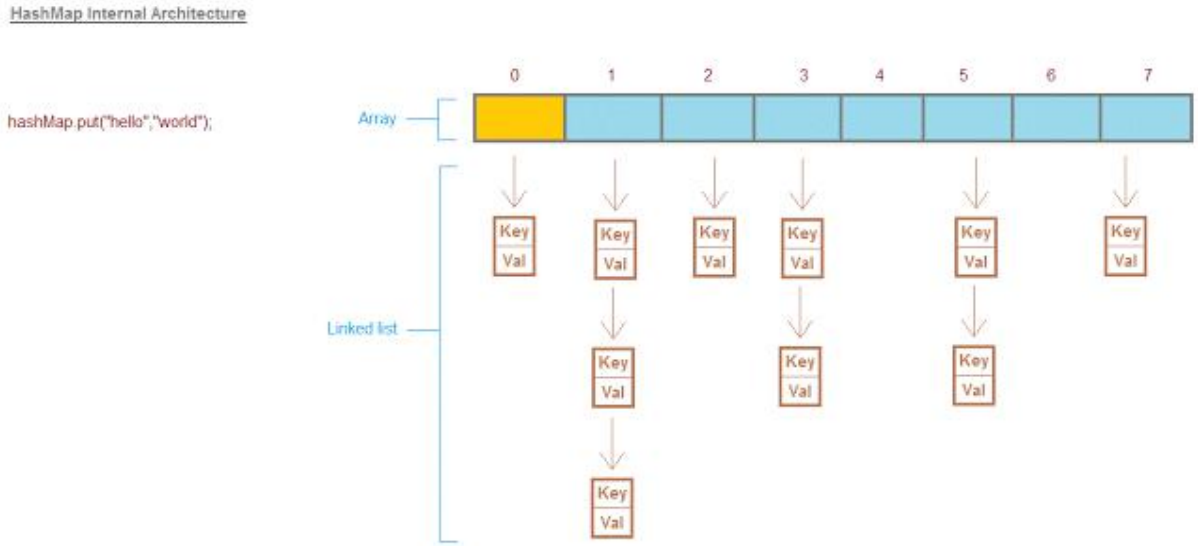
```
Integer Int = new Integer(1234567);
Int.hashCode(); // will return same number
String Str = new String("abc");
Str.hashCode(); // s[0]*31^(n-1) + s[1]*31^(n-2) + ... + s[n-1] = 96354
```

HashMap

1. Works on Hashing.
2. It uses a hash table for storing the values in key-value pair.
3. Hash table is a linked list of key value pair.
4. It has two main methods get and put for storing and retrieving the values.

Hashing in HashMap

To generate the hash value, we have to implement hashCode method for the key. Get the index value from hashCode.



Custom HashMap class :

Methods :

void clear() : It is used to remove all of the mappings from this map.
boolean containsKey(Object key) : It is used to return true if this map contains a mapping for the specified key. boolean containsValue(Object value) It is used to return true if this map maps one or more keys to the specified value.
boolean isEmpty() : It is used to return true if this map contains no key-value mappings. Object clone() It is used to return a shallow copy of this HashMap instance: the keys and values themselves are not cloned.
Set entrySet() : It is used to return a collection view of the mappings contained in this map. Set keySet() It is used to return a set view of the keys contained in this map.
Object put(Object key, Object value) : It is used to associate the specified value with the specified key in this map.
int size() : It is used to return the number of key-value mappings in this map.
Collection values() : It is used to return a collection view of the values contained in this map.

Example :

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

```
public class hashing {

        //defualt initial capacity of HashMap = 16 , load factor = 0.75
    //count frequency of each word
    static void freq(String[] str,HashMap<String,Integer> hm){
        for(int i =0 ;i<str.length;i++){
            Integer freq = hm.get(str[i]);
            if(freq == null){
                freq = 1;
            }else
                freq++;
            hm.put(str[i],freq);
            System.out.println("i : " + i +" , str[i] : "+str[i]
                    + " , currSize of HashMap : " + hm.size());
        }
        return ;
    }

    public static void main(String[] args) {
        String[] str = "Aa BB world hello hi world hi".split(" ");

        //HashMap<key,value>
        HashMap<String,Integer> hm = new HashMap<String,Integer>();

        freq(str,hm);
        System.out.println();

        //method 1 to display key value pair
        for(Map.Entry m:hm.entrySet()){
             System.out.println("key : "+ m.getKey()+" , Value : "+m.getValue());
        }
        System.out.println();

        //method 2
        System.out.println(hm + "\n");

        //method 3
        System.out.println(hm.keySet() + "\n");
        System.out.println(hm.values() + "\n");
    }

}
```

OUTPUT:

```
i : 0 , str[i] : Aa , currSize of HashMap : 1
i : 1 , str[i] : BB , currSize of HashMap : 2
```

```
i : 2 , str[i] : world , currSize of HashMap : 3
i : 3 , str[i] : hello , currSize of HashMap : 4
i : 4 , str[i] : hi , currSize of HashMap : 5
i : 5 , str[i] : world , currSize of HashMap : 5
i : 6 , str[i] : hi , currSize of HashMap : 5

key : Aa , Value : 1
key : BB , Value : 1
key : hi , Value : 2
key : world , Value : 2
key : hello , Value : 1
{Aa=1, BB=1, hi=2, world=2, hello=1}

[Aa, BB, hi, world, hello]

[1, 1, 2, 2, 1]
```

Key points :

1. If two objects are same i.e. Obj1.equals(Obj2) is true then Obj1 and Obj2 **will have same hashcode**.
2. It is not necessary for two different objects to have two different hashcode. Example "Aa" and "BB" will have same hashcode.
3. Hashcode of the key is used to locate the bucket and then equals() is used to check if key matches the required key.
4. hm.put("AA",10) and now if we write hm.put("AA",20) then the **value will be updated to 20**. (This is not a collision as "AA" is ponting to same bucket)
5. Collision happens when multiple keys hash to same bucket.
6. The object hashcode determines which bucket it goes into. (**Object.hashCode()%N** where N is number of buckets)
7. get and put complexity of hashmap is O(1) but worst case complexity is O(N).

```
//HashMap<Object,Object>
HashMap h = new HashMap();
h.put("nshu", "pra");
h.put(10, 123.321);
System.out.println(h.get("nshu") +" "+h.get(10));
```

OUTPUT : pra 123.321

HashSet

HashMap works behind HashSet. Here Object instance is never used (Instead of pair it stores only keys). HashSet custom class implements instance set.

Methods

void clear() It is used to remove all of the elements from this set.

boolean contains(Object o) : It is used to return true if this set contains the specified element.

boolean add(Object o) : It is used to adds the specified element to this set if it is not already present.

boolean isEmpty() : It is used to return true if this set contains no elements.

boolean remove(Object o) : It is used to remove the specified element from this set if it is present.

Iterator iterator() : It is used to return an iterator over the elements in this set.

int size() : It is used to return the number of elements in this set.

Example :

```
import java.util.HashSet;
import java.util.Iterator;

public class Hashset {

    public static void main(String[] args) {
        String str = "hello hi coding ninjas hello ninjas";
        String strarr[] = str.split(" ");
        HashSet<String> hs = new HashSet<>();
        for(String temp:strarr){
            hs.add(temp);
        }
        System.out.println("hashset size : " + hs.size());
        if(hs.contains("coding")){
            System.out.println("yes 'coding' is present");
        }
        System.out.println();
        //Traversing
        Iterator<String> itr = hs.iterator();
        while(itr.hasNext()){
            System.out.println(itr.next());
        }
    }

}
```

OUTPUT:

```
hashset size : 4
yes 'coding' is present
```

hi
coding
ninjas
hello

Example of HashSet

```
import java.util.HashSet;
import java.util.Iterator;

public class Hashset {

    public static void main(String[] args) {
        //defualt initial capacity of HashSet = 16 , load factor = 0.75
        //case1
        int arr1[] = {15,2,9,10};
        HashSet<Integer> hs1 = new HashSet<>(4);
        for(int temp:arr1){
            hs1.add(temp);
        }
        System.out.println("hashset1 size : " + hs1.size());
        //Traversing
        Iterator<Integer> itr1 = hs1.iterator();
        while(itr1.hasNext()){
            System.out.println(itr1.next());
        }

        //case2
        int arr2[] = {4,3,2,1,0,10};
        HashSet<Integer> hs2 = new HashSet<>(4);
        for(int temp:arr2){
            hs2.add(temp);
        }
        System.out.println("\nhashset2 size : " + hs2.size());
        //Traversing
        Iterator<Integer> itr2 = hs2.iterator();
        while(itr2.hasNext()){
            System.out.println(itr2.next());
        }

        //case3
        int arr3[] = {4,3,2,1,0,10};
        HashSet<Integer> hs3 = new HashSet<>(10);
        for(int temp:arr3){
            hs3.add(temp);
        }
        System.out.println("\nhashset3 size : " + hs3.size());
        //Traversing
        Iterator<Integer> itr3 = hs3.iterator();
        while(itr3.hasNext()){
```

```
            System.out.println(itr3.next());
        }
    }
}
```

OUTPUT :

```
hashset1 size : 4
9
2
10
15

hashset2 size : 6
0
1
2
10
3
4

hashset3 size : 6
0
1
2
3
4
10
```