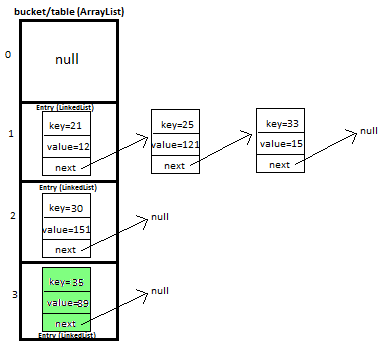
//Hash Map custom implementation in Java



One of the implementations of Hash Map is

ArrayList of linkedlists:

i.e., bucket is implemented by arraylist and entries by linkedlist

bucket-index

we use hashCode to find bucket-index ( i.e., index at which the entry is stored in the bucket)

we use equals to check equality of entries

What is entry :

It is a key, value pair **Entry<K,V>**

create an entry object /class

Entry{

Private int key;

Private int value;

Entry(int k , int v){

This.key = k;

This.value = v;

}

}

Entries are stored as linkedlist , so we also need a next { which points to next entry}

So modify your entry class

Entry{

Private int key;

Private int value;

Private Entry next;

Entry(int k , int v , Entry next ){

This.key = k;

This.value = v;

This.next = next;

}

}

Using template

**static** **class** Entry<K, V> {

       K key;

       V value;

       Entry<K,V> next;

**public** Entry(K key, V value, Entry<K,V> next){

**this**.key = key;

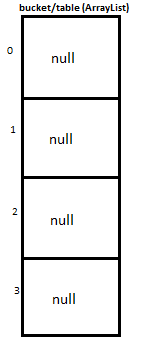
**this**.value = value;

**this**.next = next;

       }

   }

Now try to insert entries into array list

For example 

To decide which bucket ,use hash function

**hash(K key)**

lets use a simple hash function for now = key % capacity

this simple hash function will lead in collisions -🡪 since we are using linkedlist to store the entries . It does something like below



Now lets see methods used in simple hashmap

private int **hash**(K key) – to compute bucket-index using key

public void **display**() – display all key, value pairs

insertion order is not reflected , can use linkedHashmap for this purpose

public boolean **remove**(K deleteKey) – to remove key, value pair

public V **get**(K key) – returns value of corresponding key

public void **put**(K newKey, V data) –

1)inserts key value pair ( calls hash (k) internally)

2) if map alreay contains key , overrides the value

To check quality use equals and hashCode methods .

|  |  |
| --- | --- |
| Method | Description |
| public V **get**(K key) | Returns corresponding value if present , else null |
| public void **put**(K newKey, V data) | Inserts K,V pair – calls hash(key K)  It K alreay present – overrides V  Overrides equals , hashCode method |
| public boolean **remove**(K deleteKey) | to remove key, value pair |
| private int **hash**(K key) | to compute bucket-index using key |
| public void **display**() | display all key, value pairs  insertion order is not reflected , can use linkedHashmap for this purpose |

References : http://www.javamadesoeasy.com/2015/02/hashmap-custom-implementation.html