Hashmap->

Is underlying data structure is {

1.hash table.

}

2.hashcode of keys{ Insertion order not preserved }.

3.DUPLICates keys not allowed.

4.D- values not allowed.

5.hetrogenous object allowed.

6.NULL->is allowed for key.for only once.is for values it allowes infinite;

7.ser.and clonable and random implements.

8.best for SEARCH OPERATIOn.

Properties->

1. HashMap m= new HashMAp();

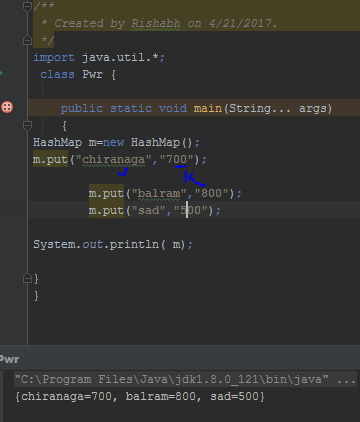
->16,->0.75 load factor/fill ratio.

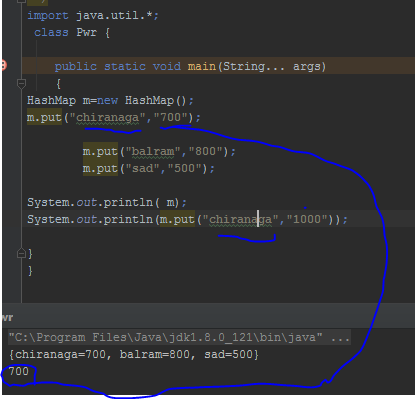
2. HashMap m= new HashMAp(int initailcapactiy);

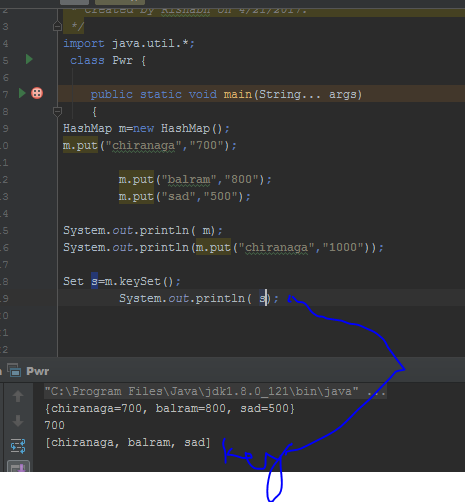
3.

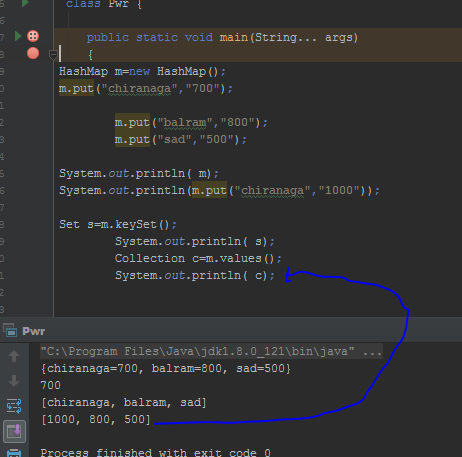
HashMap m= new HashMAp(int initi,double fillratio);

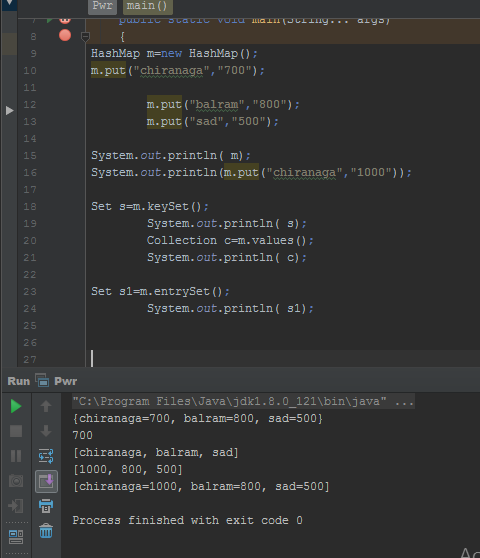
4.HashMap m= new HashMap(Map m);











Iterators for get one by on object

Set s1=m.entrySet();

Iterator itr =r iterator();

While ( itr.hashnext)

{

Map.entry m1=(map.entry)itr.next();//type casting

Sop(m1.getKey()+” ”+getvalue() );

If(m1.gtekey().equals(“sad”))

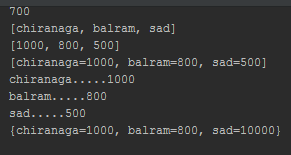
{m1.setValue(10000);

}

}

Program->   
import java.util.\*;  
 class Pwr {  
  
 public static void main(String... args)  
 {  
HashMap m=new HashMap();  
m.put(**"chiranaga"**,**"700"**);  
 m.put(**"balram"**,**"800"**);  
 m.put(**"sad"**,**"500"**);  
System.*out*.println( m);  
System.*out*.println(m.put(**"chiranaga"**,**"1000"**));  
  
Set s=m.keySet();  
 System.*out*.println( s);  
 Collection c=m.values();  
 System.*out*.println( c);  
  
Set s1=m.entrySet();  
 System.*out*.println( s1);  
 Iterator itr = s1.iterator();  
 while(itr.hasNext())  
 {  
 Map.Entry m1=(Map.Entry)itr.next();//type casting  
 System.*out*.println(m1.getKey()+**"....."**+m1.getValue());  
 if(m1.getKey().equals(**"sad"**)) {  
 m1.setValue(10000);  
 }}  
 System.*out*.println( m);  
 }  
}

Output->



|  |  |
| --- | --- |
| HashMap | Hashtable |
| 1.not syncronised. | 2.syncronised |
| 2.mutiple threds are allowed not thread safe | 2.only one thread are allowed. |
| 3.Performance is high | 3.Performance is low (relativity) |
| 4.null-= key value | Null not allowed |
| 5.not 1.2 version | 5.1.0 ver legacy element |

**Q.**How to get syncronised version of hash map object?

->by default hashmap is non-syncronised but we can get Syncronised version of hashmap by using syncronised map method of collections class,

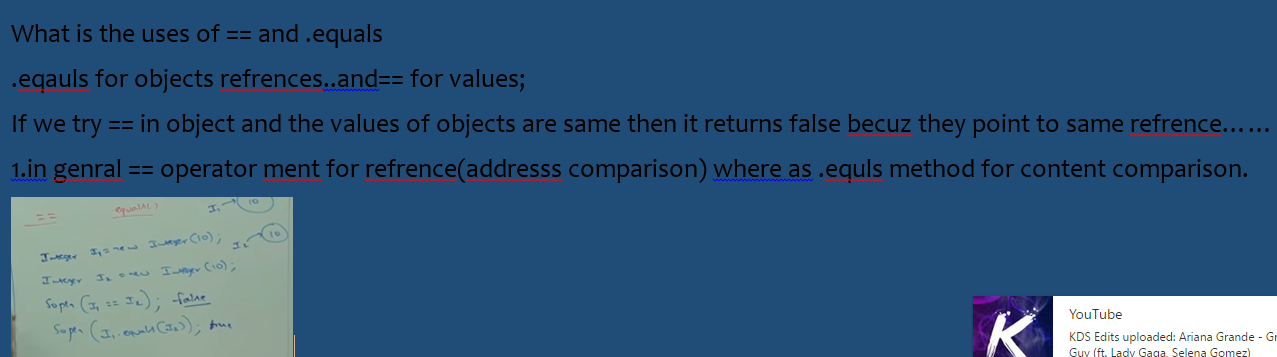
HashMap m=new HashMap();

Map m1=Collection.syncronisedMap(m);

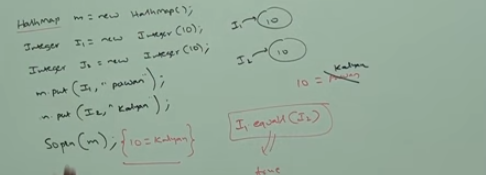
Linked hash map->its thew child class of hashmap.including method and constructor exept the following diff.

|  |  |
| --- | --- |
| Hash map | Linked hash map |
| 1.the underlying ds is hash table . | DS linked DS. |
| 2.insertion order not preserved.based on hash code of keyset. | Insertion order is preserved |
| 3.introduced in 1.0 | 1.4v |
|  |  |
|  |  |
|  |  |

Linked hash set and a linked hashmap are commonly used for developing cache based application



Identity hashmap->

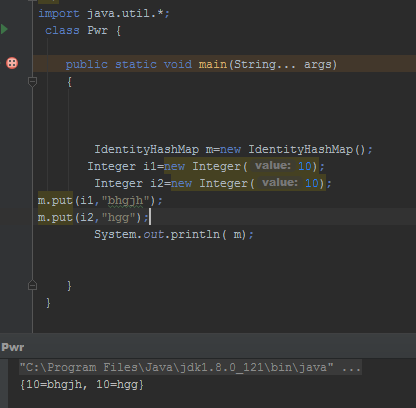


Identitiy uses “== “instead of .eqauls.

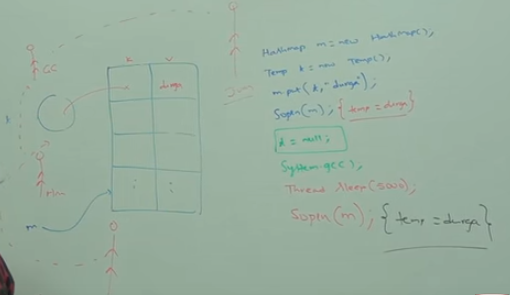
Its exactly same as hash map.including{diff method and constu}exept the following diff.

{

In the case of normal hash map jvm will use .equlas method to identifiy dublicates keys which is ment of for content comparision.,

In the case of identity hash map jvm use “==” method to identifiy dublicates ..which is ment of for address comparision. }

WeakHashMap->



This story cant run on weak hashmap

HASHTABLE->

