Key and Value both are Objects.

Key needs to be Unique

Value can be duplicate

<K,V> ---Entry

Refer PassportMapExample.java

For Calling Garbage Collector we need to use System.gc()

System.gc will internally make a call to finalize() method to remove the unused Objects.

HashMap:

import java.util.\*;

class Employee

{

private int empid;

private String name;

private String empAddr;

*@Override*

public String toString()

{

return "Chinmay";

}

*@Override*

public void finalize()

{

System.***out***.println("Garbage Collector collected the object");

}

//setter

//getters

}

public class HashMapGarbageCollector

{

public static void main(String[] args)throws Exception

{

Employee e=new Employee();

HashMap hm=new HashMap();

hm.put(e, "Chinmay Sai");

e=null; //Eligible for Garbage Collection as it is not pointing to null

//Even though we have made reference of e point to null ,but HashMap will be pointing to Employee

//Due to which garbage Collector will not be able remove the Object.

//So this case HashMap is dominating the Garbage Collector as the Object is part of HashMap

System.*gc*();//call to GC

Thread.*sleep*(4000);

System.***out***.println(hm);

}

}

**WeakHashMap:**

import java.util.\*;

class NewEmployee

{

private int empid;

private String name;

private String empAddr;

*@Override*

public String toString()

{

return "Chinmay";

}

*@Override*

public void finalize()

{

System.***out***.println("Garbage Collector collected the object");

}

//setter

//getters

}

public class WeakHashMapGarbageCollector

{

public static void main(String[] args)throws Exception

{

NewEmployee e=new NewEmployee();

WeakHashMap hm=new WeakHashMap();

hm.put(e, "Chinmay Sai");

e=null; //Eligible for Garbage Collection as it is not pointing to null

//Even though we have made reference of e point to null ,but WeakHashMap will be pointing to Employee

//Even though WeakHashMap is pointing to the Object Garbage Collector will remove the Object

//So this case Garbage Collector is dominating HashMap.

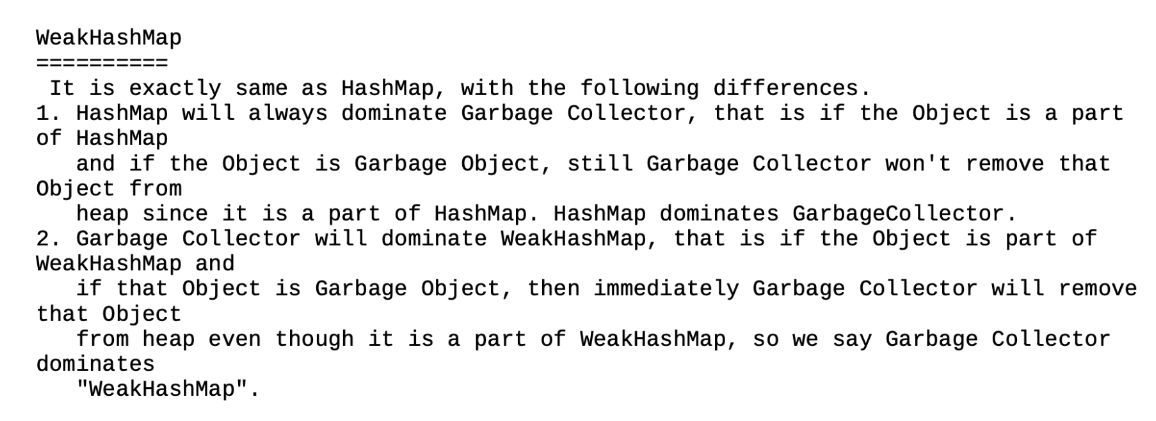
System.*gc*();//call to GC

Thread.*sleep*(4000);

System.***out***.println("Weak HashMap Data: "+hm);

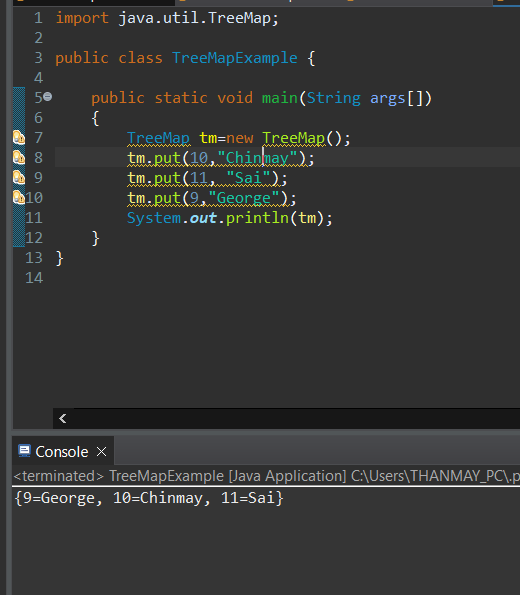
}

}



TreeMap:

TreeMap sorts the elements by Key.



Hashtable:

import java.io.\*;

import java.util.Hashtable;

import java.util.Iterator;

import java.util.Map;

import java.util.Map.Entry;

import java.util.Set;

/\*\*

\* Hash table is a Legacy Class(1.0v)

\* Hashtable methods is Synchronized

\* Hashtable is Thread Safe

\* Hashtable has Less Performance when compared with HashMap

\*

\*/

class Chinmay

{

private int i;

public Chinmay(int i)

{

this.i=i;

}

*@Override*

public String toString() {

return "Chinmay [i=" + i + "]";

}

}

public class HashTableExample {

public static void main(String[] args) {

Hashtable ht=new Hashtable();

ht.put(10,"Chinmay");

Set se=ht.entrySet();

Iterator itr=se.iterator();

while(itr.hasNext())

{

Map.Entry me=(Entry)itr.next();

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

System.***out***.println(me.getKey()+" "+me.getValue());

}

Hashtable ht1=new Hashtable();

ht1.put(new Chinmay(10),"Chinmay");

ht1.put(new Chinmay(11),"Sai");

ht1.put(new Chinmay(12),"George");

System.***out***.println(ht1);

}

}

IdentityHashMap does not Override equals () method. So for Comparision we will use ==

Whereas HashMap Overrides the equals() method so we can use equals() method for Comparison.

