***Map***

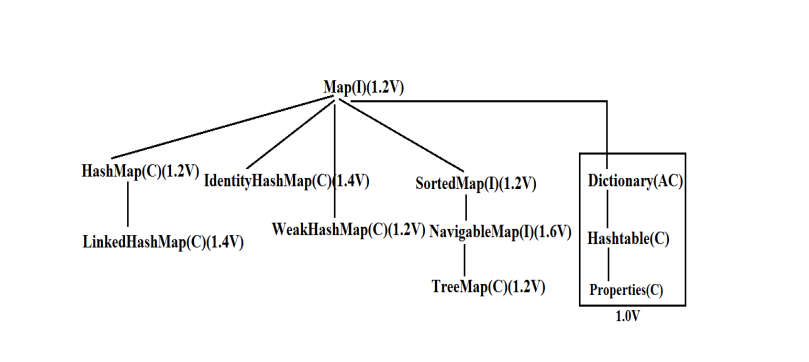
1. Map is not child interface of collection.
2. If we want to represent group of individual objects as key value pair then should go for map.
3. Example- Roll no Name

101 Sanjay

102 Ram

103 Shyam

1. Both key and values are objects, duplicated keys are not allowed, but values may be duplicated.



HashMap

A HashMap is class which implements the Map interface

It stores values based on key.

It has 16 size and internally it will increase the size by double, so new size will be 32,64,128.

It is unordered, which means that the key must be unique

It may have null key-null value

For adding elements in HashMap we use the put method

Return type of put method is Object.

**Constructors:**

1) HashMap m = new HashMap();

Creates an Empty HashMap Object with Default Initial Capacity 16 and Default Fill Ratio 0.75

2) HashMap m = new HashMap(intinitialcapacity);

3) HashMap m = new HashMap(intinitialcapacity, float fillRatio);

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

Example-

**package** com.test;

**import** java.util.HashMap;

**import** java.util.TreeSet;

**public** **class** A {

**public** **static** **void** main(String[] args) {

HashMap hm = **new** HashMap();

hm.put(10, "ashok");

hm.put(11, "ram");

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

}

}

LinkedHashMap-

* A LinkedHashMap is a ‘hashtable and linked list implementation of the map interface with a predictable iteration order.
* It is the same as HashMap except it maintains an insertion order i.e. ordered

Example-

**package** com.test;

**import** java.util.HashMap;

**import** java.util.LinkedHashMap;

**import** java.util.TreeSet;

**public** **class** A {

**public** **static** **void** main(String[] args) {

LinkedHashMap hm = **new** LinkedHashMap();

hm.put(10, "ajay");

hm.put(11, "ram");

hm.put(12, "shyam");

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

}

}

Output-

{10=ajay, 11=ram, 12=shyam}

TreeMap-

* The TreeMap is a class which implements NavigableMap interface which is the sub- interface of SortedMap.
* It stores values based on key
* It is ordered but in an Ascending manner
* Keys should be unique
* It cannot have null key at run time but can have null values because the interpreter will not understand how to sort null with other values.

1) TreeMap t = new TreeMap(); For Default Natural Sorting Order.

2) TreeMap t = new TreeMap(Comparator c); For Customized Sorting Order.

3) TreeMap t = new TreeMap(SortedMap m); Inter Conversion between Map Objects.

4) TreeMap t = new TreeMap(Map m);

Example-

**package** com.test;

**import** java.util.HashMap;

**import** java.util.LinkedHashMap;

**import** java.util.TreeMap;

**import** java.util.TreeSet;

**public** **class** A {

**public** **static** **void** main(String[] args) {

TreeMap hm = **new** TreeMap();

hm.put(10, "Ajay");

hm.put(11, "ram");

hm.put(12, "shyam");

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

}

}

Output-

{10=Ajay, 11=ram, 12=shyam}

Hashtable-

* Hashtable is a class which implements Map interface and extends Dictionary class.
* It stores values based on key
* It is unordered and the key should be unique
* It cannot have null keys or null values. It gives runtime error if we try to add any null keys or values but will not show an error at compile time.
* It has synchronised methods and slower than hashmap.

Example-

**package** com.test;

**import** java.util.Hashtable;

**public** **class** A {

**public** **static** **void** main(String[] args) {

Hashtable ht = **new** Hashtable();

ht.put(10, "ram");

ht.put(11, "sohan");

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

}

}

Output-

{10=ram, 11=sohan}

Example- 1

**package** com.hashmap;

**import** java.util.\*;

**public** **class** HashMapDemo {

**public** **static** **void** main(String[] args) {

HashMap<Integer, String> map = **new** HashMap<Integer, String>();

map.put(10, "Ram");

map.put(20, "yogesh");

map.put(30, "sohan");

Set<Integer> s = map.keySet(); // s contain all the keys only.

**for** (**int** i : s) {

System.***out***.println("Key==" + i);

System.***out***.println("value=" + map.get(i));

/\*

\* get method used to get the respective value of key.

\*/

}

}

}

Comparison between HashMap, LinkedHashMap, TreeMap and HashTable:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Topic | HashMap | LinkedHashMap | TreeMap | HashTable |
| Duplicate Key | Not Allowed | Not Allowed | Not Allowed | Not Allowed |
| Ordering | Unordered | Maintains insertion order | Maintains in Accessing order | Unordered |
| Null (Key Value) | Allow | Allow | key Not allowed but value is Iterator | Not Allowed |
| Accessing Elements | Iterator | Iterator | Iterator | Iterator |
| Thread Safety | No | No | No | Yes |

Example-2

**package** com.hashmap;

**import** java.util.\*;

**public** **class** HashMapDemo2 {

**public** **static** **void** main(String[] args) {

HashMap<Integer, String> map = **new** HashMap<Integer, String>();

map.put(10, "Ram");

map.put(20, "yogesh");

map.put(30, "sohan");

Set<Integer> s = map.keySet();

Iterator<Integer> itr = s.iterator();

**while** (itr.hasNext()) {

**int** i = itr.next();

System.***out***.println("key=" + i);

System.***out***.println("value=" + map.get(i));

}

}

}

Example-3

**package** com.hashmap;

**import** java.util.\*;

**public** **class** LinkedHashMapDemo {

**public** **static** **void** main(String[] args) {

LinkedHashMap<Integer, String> map = **new** LinkedHashMap<Integer, String>();

map.put(10, "Ram");

map.put(10, "yogesh"); //override

Set<Integer> s = map.keySet();

Iterator<Integer> itr = s.iterator();

**while** (itr.hasNext()) {

**int** i = itr.next();

System.***out***.println("key=" + i);

System.***out***.println("value=" + map.get(i));

}

}

}

Example-4

**package** com.setdemo;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Map;

**public** **class** HashMapDemo {

**public** **static** **void** main(String[] args) {

Map<String, String> map = **new** HashMap<String, String>();

map.put("ram", "patil");

map.put("ajay", "deshmukh");

// using iterators

Iterator<Map.Entry<String, String>> itr = map.entrySet().iterator();

**while** (itr.hasNext()) {

Map.Entry<String, String> entry = itr.next();

System.***out***.println("Key = " + entry.getKey() + ", Value = " + entry.getValue());

}

}

}

Example-5

**package** com.setdemo;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Map;

**public** **class** HashMapDemo {

**public** **static** **void** main(String[] args) {

Map<String,String> map = **new** HashMap<String,String>();

// enter name/url pair

map.put("ram", "patil");

map.put("shyam", "deshmukh");

// forEach(action) method to iterate map

map.forEach((k,v) -> System.***out***.println("Key = "

+ k + ", Value = " + v));

}

}

Example- 4

**package** com.hashmap;

**import** java.util.\*;

**public** **class** TreeMapDemo {

**public** **static** **void** main(String[] args) {

TreeMap<String,String> treeMap= **new** TreeMap<String,String>(); //sorted elements based on key

treeMap.put("20","velocity");

treeMap.put("50","pune");

treeMap.put("10","software");

//System.out.println(treeMap);

Set<String> s=treeMap.keySet();

**for**(String i: s) {

System.***out***.println("key="+i);

System.***out***.println("value="+treeMap.get(i));

}

}

}