List of Concepts Involved:

- Introduction to Map in Java
- Map Hierarchy
- HashMap
- Other In-Built classes and Inbuilt methods under Map Hierarchy
- Need of Generics and Basics of Generics
- More on Generics in Java
- Collections class and it's inbuilt methods in Java
- Comparator vs Comparable Interface

Introduction to Map in Java

Map

To represent a group of individual objects as a key value pair then we need to opt for Map(I).

- It is not a child interface of Collection.
- If we want to represent a group of Objects as a key-value pair then we need to go for Map.
- Both keys and values are Objects only
- Duplicate keys are not allowed but values are allowed.
- Key-value pair is called "Entry".

Map interface

It contains 12 methods which is common for all the implementation Map Objects
a. Object put(Object key,Object value)
b. void putAll(Map m)
c. Object get(Object key)
d. Object remove(Object key)
e. boolean containsKey(Object key)
f. boolean containsValue(Object value)
g. boolean isEmpty()
h. int size()
i. void clear()

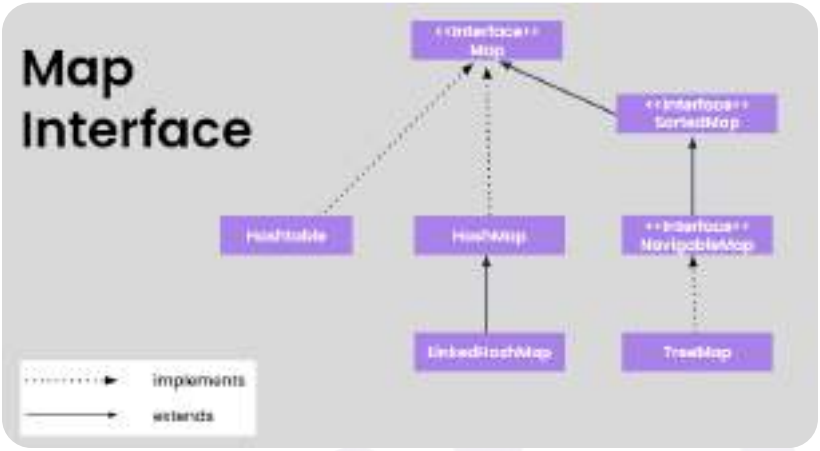
views of a Map

j.Set keySet()
k.Collection values()
l.Set entrySet()

Entry(I)

1. Each key-value pair is called Entry.
2. Without the existence of a Map,there can't be the existence of an Entry Object.
3. Interface entry is defined inside the Map interface.

```
interface Map{
 interface Entry{
  Object getKey();
  Object getValue();
  Object setValue(Object newValue);
 }
}
```



HashMap

- Underlying DataStructure : Hashtable
- insertion order : not preserved
- duplicate keys : not allowed
- duplicate values : allowed
- Heterogenous objects : allowed
- null insertion : for keys allowed only once,but for values can be any no.
- implementation interface : Serializable,Cloneable.

Difference b/w HashMap and Hashtable

HashMap => All the methods are not synchronized.
Hashtable => All the methods are synchronised.

HashMap => At a time multiple threads can operate on an Object, so it is ThreadSafe.
Hashtable => At a time only one Thread can operate on an Object, so it is not ThreadSafe.

HashMap => Performance is high.
Hashtable => performance is low.

HashMap => null is allowed for both keys and values.
Hashtable => null is not allowed for both keys and values,it would result in NullPointerException.

HashMap => Introduced in 1.2v
Hashtable => Introduced in 1.0v

Note: By default HashMap is nonSynchronized,to get the synchronized version of HashMap we need to use synchronizedMap() of Collection class.

Constructors

1. HashMap hm=new HashMap()
   //default capacity => 16, loadfactor => 0.75
2. Hashmap hm=new HashMap(int capacity);
3. HashMap hm=new HashMap(int capacity,flot filtration);
4. HashMap hm=new HashMap(Map m);

LinkedHashMap

=> It is the child class of HashMap.
=> It is same as HashMap,but with the following difference

HashMap => underlying data structure is hashtable.
LinkedHashMap => underlying data structure is LinkedList + hashtable.

HashMap => insertion order not preserved.
LinkedHashMap => insertion order preserved.

HashMap => introduced in 1.2v
LinkedHashMap => introduced in 1.4v

SortedMap

1. It is the child interface of Map
2. If we want an Entry object to be sorted and stored inside the map,we need to use "SortedMap".

SortedMap defines few specific method like
a. Object firstKey()
b. Object lastKey()
c. SortedMap headMap(Object key)
d. SortedMap tailMap(Object key)
e. SortedMap subMap(Object obj1,Object obj2)
f. Comparator comparator()

NavigableMap (I):

=> It is the Child Interface of SortedMap.
=> It Defines Several Methods for Navigation Purposes.

TreeMap

- Underlying data structure is "redblacktree".
- Duplicate keys are not allowed,whereas values are allowed.
- Insertion order is not preserved and it is based on some sorting order.

- If we are depending on natural sorting order,then those keys should be homogenous and it should be Comparable otherwise ClassCastException.
- If we are working on customisation through Comparator,then those keys can be heterogeneous and it can be NonComparable.
- No restrictions on values, it can be heterogeneous or NonComparable also.
- If we try to add null Entry into TreeMap, it would result in "NullPointerException".

Hashtable:

- The Underlying Data Structure for Hashtable is Hashtable Only.
- Duplicate Keys are Not Allowed. But Values can be Duplicated.
- Insertion Order is Not Preserved and it is Based on Hashcode of the Keys.
- Heterogeneous Objects are Allowed for Both Keys and Values.
- null Insertion is Not Possible for Both Key and Values. Otherwise we will get Runtime Exception Saying NullPointerException.
- It implements Serializable and Cloneable,but not RandomAccess.
- Every Method Present in Hashtable is Synchronized and Hence Hashtable Object is Thread Safe,so best suited when we work with Search Operation.

Need of Generics and Basics of Generics Generics

The purpose of Generics is

1. To provide TypeSafety.
2. To resolve TypeCasting problems.

Case1:

TypeSafety

- A guarantee can be provided based on the type of elements.
- If our programming requirement is to hold only String type of Objects,we can choose String Array.
- By mistake if we are trying to add any another type of Objects, we will get "CompileTimeError".

eg#1.

```
String[] s=new String[10000];
     s[0] = "dhoni";
     s[1] = "sachin";
     s[2] = new Integer(10); //CE:
incompatible types found :
java.lang.Integer

                  required:
java.lang.String
```

W.r.t Arrays we can guarantee that what type of elements is present inside Array, hence Arrays are safe to use w.r.t type,so Arrays as TypeSafety.