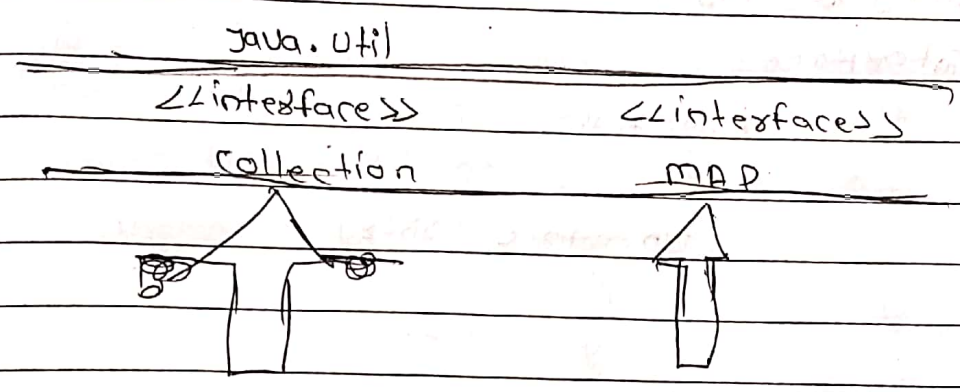
Page No.
Date 18/12/20

MAP

→ A map is nothing but collection of entries ie map can store xple entries.

→ MAP is seperate Vertical or Pillar which is part of collection framework present in java.util package

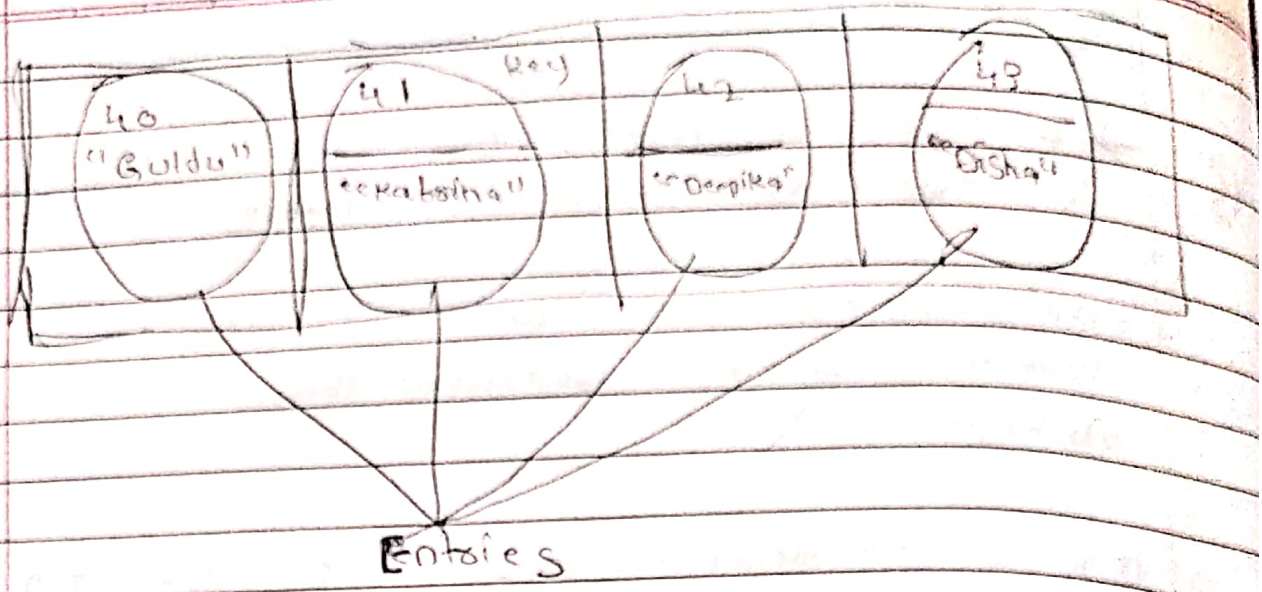
→ MAP is an interface present since JDK 1.2



entry → data
↳ associate data

Note:- Since MAP is a seperate Vertical, it has it's own methods. MAP is nowhere related to Collection interface

entry → is nothing but a data in a associated form. each entry is a pair of key and value data. Key must always be Unique and value can be duplicated or can be null also



* entry is nested interface presente within map interface

```
+ interface Map
{
    tinterface Entry
    {
    }
}
```

2 ways of accessing

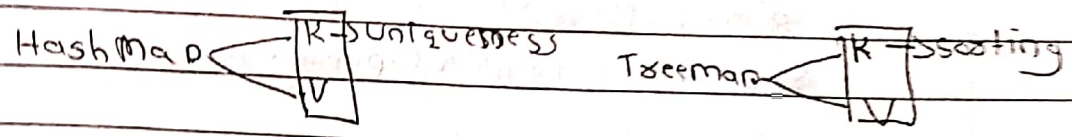
1. import java.util.Map.*; -> Map.Entry
2. import java.util.Map.Entry; -> Entry ->

```
public interface Map
{
    Public interface Entry
    {
        Public K getKey();
        Public V getValue();
    }
}
```

Till today mock

Page No.
Date

In case of map all the operations like Uniqueness, Sorting ~~works on Key~~ but not on the Value (imp)



Sr:- import java.util.Hashmap;
import java.util.Map;

Public class MapMethodsDemo{
Psum (String[] args){

Map<Integer, String> CSKmap = new Hashmap <Integer, String>();
CSKmap.put (17, "Dhoni");
CSKmap.put (10, "Raina");
CSKmap.put (1, "Bravo");

Map<Integer, String> mimap = new Hashmap<Integer, String>();
mimap.put (45, "Rohit");
mimap.put (92, "Pandya");
mimap.put (96, "Bhumra");

Map<Integer, String> iplmap = new Hashmap<Integer, String>();
iplmap.putAll (CSKmap);
iplmap.putAll (mimap);

System.out.println (iplmap);

System.out.Println (iplmap.containsKey (17));
(iplmap.Contains Value ("ABD"));

Page No.
Date

```
System.out.println(iplmap.size());
                   (iplmap.get(92)); // Pandya
iplmap.clear();
System.out.println(iplmap.size());
}
}
```

o/p :- {17=Dhoni, 1=Bravo, 10=Raina, 90=Bhumra, 92=Pandya, 45=Rohit}
true
false
6
Pandya
Bhumra
{17=Dhoni, 1=Bravo, 10=Raina, 92=Pandya, 45=Rohit}
0

methods of map

Set<K> allK = map.keySet(); -> Set because Key is Unique & Set stores only Unique Values
(datatype)
Set<Integer>

17	92	8

Collection<V> allV = map.values();
Collection<String>

Coll2 because value can be same

"Dhoni"	"Pandya"	"Rohit"

{Integer} K →

{String} V →

17	92	8
Dhoni	Pandya	Rohit

Iterator of the Keys & Index;

```
Set<Integer> allk = inlmap.keySet();
for (Integer i : allk) {
    System.out.println(i);
}

Iterator<Iterator> itr = allK.iterator();
while (itr.hasNext()) {
    Integer key = itr.next();
    System.out.println(key);
}

Collection<String> allv = inlmap.values();
for (String val : allv) {
    System.out.println(val);
}
}
}
```

HashMAP : (Points from hash tbl.)

* Hash map is one of the subclass of map Interface Present since JDK 1.2.
* Data structure is hash-table
* Hash map does not maintain insertion order where as insertion order is based on hash-code of key
* Can store single null key

Page No.
Date

* Initial Capacity is 16 and the fill ratio as 0.75.

* Hash map ~~Interface~~ implements marker interface like serialisable & cloenable, but not Random -Access. (?)

Linked Hash map :-

→ is one of the Implementation class of map Interface Present since JDK 1.4

→ LHM maintains insertion ~~order~~.

Difference bln hashmap & hashtable

~~Hash~~table : is one of the implementation Class of map Interface Present since JDK 1.0

* Hashtable is single threaded i.e the methods are synchronised.

* Hash table cannot even store a single null key.

* Hashtable initial Capacity is 11.

* Hash table is Comparatively slow than the Hashmap

Insertion order is based on key values

Page No.
Date

Difference between Hashmap & Hashtable

Hashmap	Hashtable
Hashmap is multithreaded	Hashtable is singlethreaded
Hashmap methods are not synchronized(?)	Hashtable methods are synchronized
Hashmap initial capacity is 16	Initial capacity is 11
Hashmap can store single nullKey	cannot even store single null Key
Hashmap is faster,	Hashtable is slower
Present since JDK 1.2	Present since JDK 1.0

Tree Map : is one of the subclass of Map Interface Present since JDK 1.2

Tree map is mainly used for sorting and sorting happens based on Key

Sorting works based on Comparable in default natural sorting order

In Tree Map Key must be homogeneous

Tree Map cannot even store a single null Key

Initial Capacity cannot be applicable

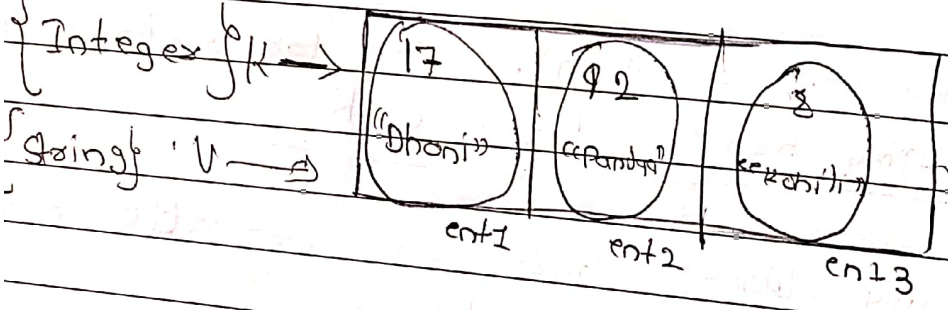
```
import java.util.TreeMap;

Public class TmDemo{
    psvm (String[] args){
TreeMap<Integer, String> map = new TreeMap
    <Integer, String>();

    map.put(17, "Dhoni");
    map.put(10, "Raina");
    map.put(1, "Bravo");
    map.put(45, "Rohit");
    map.put(92, "Pandya");
    map.put(40, "Bhumra");
    S.O.P(map);
}
```

O/P :- {1=Bravo, 10=Raina, 17=Dhoni, 45=Rohit, 40=Bhumra, 92=Pandya}

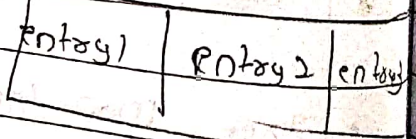
Entries :- (fetching the values from an entry datatype)



Set<Entry> alle = map.entrySet();

Generics of Set

Set<Entry<Integer, String>>



Page No.
Date

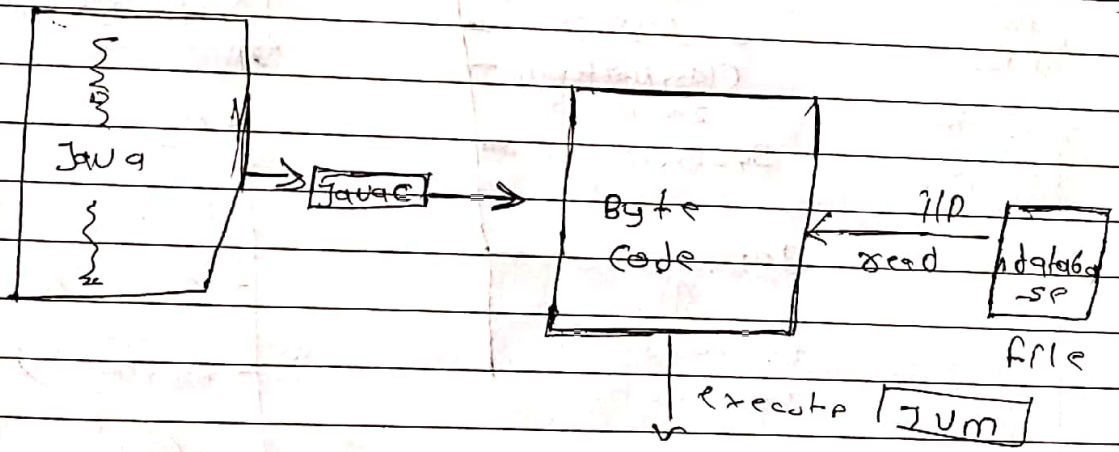
entry data type

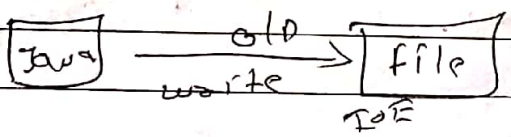
```
for (Entry<Integer, String> ent : allE)
{
    Integer key = ent.getKey();
    String val = ent.getValue();
}
```

Exception Handling :- (14/12/19)

→ Exception :- Exception is a runtime interruption which stops a Program execution.

Exception occurs only at runtime and exception can be handled or suppressed at exception raise.





There are 2 categories of exception

1) Checked exception
2) Unchecked ,,

→ exception can by handled by using