#Hashmaps And Hashsets

M T W T F S S | Page No.: | Date: | YOUVA

Introduction to HashSets

- HashSets → Interface / Data Structure
- size, insert, remove, search → O(1)
- ** In HashSet, Occurence of every element is 1.

STL And Important Methods in Hashsets.

```
HashSet<Integer> set = new HashSet<>();
```

// Insertion :- TC => O(1)

```
set.add(20);
set.add(100);
```

// Size :- TC => O(1)

```
set.size();
```

```
set.add(50);
set.add(50);
```

Added Only Once.

Only Insert One time.
Duplicate Can not be inserted. Not Give Any Error

// Search :- true/False : TC => O(1)

```
set.contains(20);   → True
Set.contains(200);  → False.
```

// Remove :- TC => O(1)

```
set.remove(100);
set.remove(1000);
```

Does Not Give Any Error. if Value is Not in set.

// ~~Error~~ Display

```
System.out.println(set);
```

→ Display in Random Sequence.

Convert set to Array :-

```
Object[] arr = set.toArray();
```

→ It is Interface so we need to use Object.

How To Iterate in Hashset :-

→ For each loop.

// Clear.

```
Set.clear();
```

Delete All the ele from Set.

Why not for/while loop? → Because there is No Concept of Index.

```
for(int ele : set){
    sout(ele + " ");
}
```

```
for(int ele : set){
    set.remove(ele);
}
```

→ It will Give error. Iterative Remove Not Possible

Introduction to HashMaps :

↳ Data Structure.
↳ insert, remove, search ⟶ O(1).

Pairs ⟶ key, Value.

* In a HashMaps, there can be 2 or more keys with same values.

⟶ But keys are unique.

STL & Important Methods in maps :-

↳ HashMap<String, Integer> map = new HashMap<>();
(key, Value)

Insertion / Update :-
↳ Insertion of (key, Value) pairs

↳ map.put("Raghav", 76);
key Value.
↳ map.put("Raghav", 86);

* ⟶ keys must be unique.

↳ map.put("Ram", 86);

⟶ Values can be ununique

key	Value
~~"Ram"~~	~~86~~
"Raghav"	86. ~~76.~~

map

Size :-
↳ map.size();
↳ 2.

Search :- True / False
↳ search for key
↳ map.containsKey("Raghav"); ⟶ True
↳ map.containsKey("Gthagan"); ⟶ False

↳ map.containsValue(86); ⟶ True
↳ map.containsValue(100); ⟶ False
↳ Search for value

Get :- ↳ map.get(key);
↳ Gives value of the key
↳ map.get("Raghav");
↳ (86)

Remove :- ↳ Removes the pair.
↳ map.remove(key);
↳ map.remove("Ram");
↳ It will delete the pair

↳ map.remove("Manahar");
↳ map.remove(100);
} There will be not Error.

How to Iterate in Hashmap :-

↳ For each loop, [map.keySet()].

Raghav, 76
Ram, 20
Om, 34
Gagan, 18

Map.

Raghav
Ram
Om
Gagan

keySet.

76.
20
34
18

Values.

=> Output :-

76.
34
20
18.

=> Iterate using keySet :-

```
for (String key : map.keySet()){
    int val = map.get(key);
    Sout(key + " " + val);
}
```

=> If you have key you can get Values of that key.

Output :- Raghav 76.
Om 34.
Ram 20
Gagan 18.

=> Iterate on/using Values :-

```
for (Integer val : map.values()){
    Sout(val);
}
```

=> Iterate on Pair :-

```
for (Object pair : map.entrySet()){
    Sout(pair);
}
```

=> Output :- Raghav = 76.
Om = 34
Ram = 20.
Gagan = 18.

TreeSet And TreeMaps

→ Insert, remove, search → O(log n)

→ Sorted

TreeSet / TreeMap -> Ordered Set / Ordered Map.

HashSet / HashMap => UnOrdered

Ex :-
```
TreeSet<Integer> set = new TreeSet<>();
Set.add(100);
Set.add(58);
set.add(-8);
Set.add(76);
Set.add(89);

for(int ele : set){
    Sout(ele + " ");
}
```

Output :-
-8, 58, 76, 89, 100.

Ex:- TreeMap<String, Integer> map = new TreeMap<>();

```
map.put("raghav", 76);
map.put("Himanshu", 83);
map.put("amritanshu", 13);
map.put("Sunil", 2);
map.put("Ayushi", 82);
map.put("prachi", 88);
```

```
for(String key : map.keySet()){
{   Sout(Key+" "+map.get(key));
}.
```

=> Output :-

Ayushi 82.

Himanshu 83.

Sunil 2

amritanshu 13.

prachi 88.

raghav 76.