→ Set is child interface of collection.
→ If we want to represent a group of individual object as a single entity where duplicates are not allowed and insertion order not preserved. then we will go for Set.
→ Set interface does not contain any new methods and we have to use only collection interface methods.

⇒ **Hash Set** :→

* The underlined data structure for Hash Set is Hash Table
→ Duplicates are not allowed.
→ Insertion order is not preserved and it is based on hashcode of Objects.
→ NULL insertion is possible (only once)
→ Heterogeneous Objects are allowed.
→ Hash Set implements Serializable and cloneable. but Not Random Access interface.
→ Hash Set is best choice if our frequent opⁿ is Search Oprⁿ.

Note :- In Hash Set, duplicates are not allowed If we are trying to insert duplicates then we won't get any C.T.E or R.T.E and add() method simply returns false.

Ex :→

```
HashSet h = new HashSet();
S.O.p(h.add("A"); → true
S.O.p(h.add("A"); → false.
```

⇒ **Constructors** :→

```
HashSet h = new HashSet();
```

It creates an empty hashset Object with default initial capacity 16 and default fill Ratio. 0.75 (load Factor)

② HashSet h = new HashSet(int initialCapacity);

↳

It creates an empty HashSet object with specified initial capacity and default fill ratio (load factor) is 0.75.

③ HashSet h = new HashSet(int initialCapacity, float fillRatio)

↳ 0.9

It creates an empty hash set object with specified initial capacity and fill Ratio.

④ HashSet h = new HashSet(Collection c);

↳

Creates an equivalent HashSet Object for the given collection.

→ This constructor maint for interconvertion b/w Collection Objects.

⇒ Fill Ratio or Load factor :→

→ After filling how much ratio, a new HashSet Object will be created, this ratio is called Fill Ratio or LoadFactor.

Ex:-

```
public class HashSetDemo
{
    P S V m (String[] args)
    {
        HashSet h = new HashSet();
        h.add("B"); h.add("C");
        h.add("D"); h.add("Z");
        h.add(null); h.add(10);
        S.O.p(h.add("Z")); → false.
        S.O.p(h); // [null, B, Z, D, C, 10] → on hashcode
```