=> Set :-

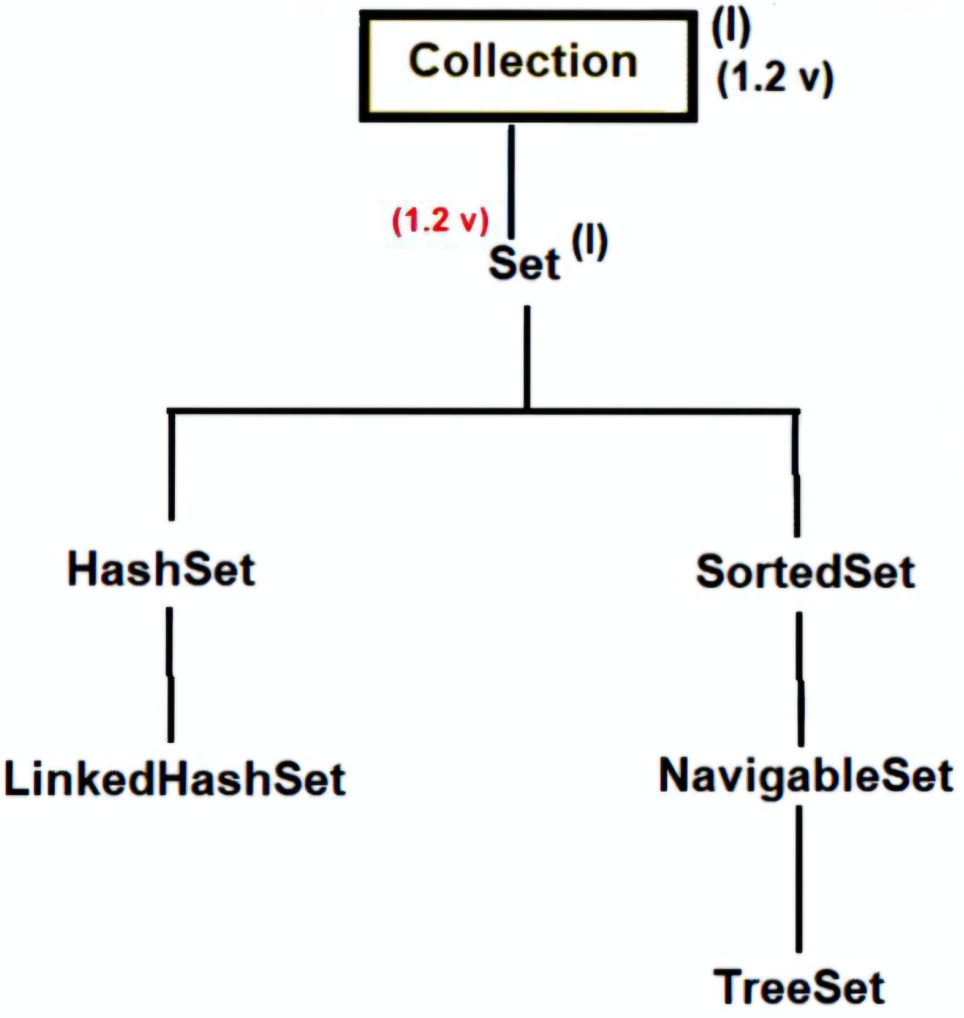
-Set is an interface which is present in java.util package

-Set is the child interface of Collection interface

-Syntax : public interface Set extends Collection { - }

-Set was introduced in JDK 1.2 version

Hierarchy of Set interface :-



-> Properties of Set interface :-

1. Set is not an index based data structure, it stores the elements as per elements "hashcode" values
2. Set does not follow the insertion order (except LinkedHashSet)
3. Set does not follow the sorting order (except SortedSet, NavigableSet & TreeSet)
4. Set can store different data types or hetrogeneous elements (except SortedSet, NavigableSet, TreeSet)
5. We cannot store duplicate elements in Set
6. We can store only one null value in Set

-> Methods of Set interface :-

= approx same methods as that of Collection interface

=> Difference between List & Set :-

1. List is index based data structure which means in list data is stored by using index position
   Set is not an index based data structure, it stores the data according to hashcode values of the elements
2. List allows duplicate elements
   Set does not allow duplicate elements

3. List can store any number of null values
   Set can store only one null value

4. List follows the insertion order
   Set does not follows the insertion order

5. In case of List we can use Iterator & ListIterator cursor
   In case of Set we can use only Iterator cursor

6. List is used in case of retrieveing the elements
   Set is used when we does not want to allow duplicacy

=> HashSet :-

-HashSet is an implemented class of Set interface which is present in java.util package

-Syntax : public class HashSet extends AbstractSet implements Set, Cloneable, Serializable { - }

-The underline data structure of HashSet is Hashtable (HashSet is backed up by Map)

-HashSet was introduced in JDK 1.2 version

Properties of HashSet :-

1. HashSet is not an index based data structure, it stores the elements according to elements hashcode values
2. HashSet can store different data types of hetrogeneous elements
3. HashSet cannot store the duplicate elements
4. HashSet can store maximum only one null value
5. HashSet does not follows the insertion order
6. HashSet does not follow the sorting order
   (same properties as Set interface)
7. HashSet is non-synchronized collection because HashSet does not contain any synchronized methods
8. HashSet allows more than one thread at one time
9. HashSet allows the parallel execution
10. HashSet reduces the execution time which in turn makes our application fast
11. HashSet is not threadsafe
12. HashSet does not gurantee for data consistency

-> Working of HashSet :-

1. HashSet internally works on the basis of Hashtable (it internally backed up by Map.)
2. When we insert any element in HashSet, it stores as a key inside the Map and at value position PRESENT reference variable is stored which is the reference variable of Object class
3. Initial capacity of HashSet is 16 elements
4. Its load factor or fill ratio is 75%
5. After load is filled then a new Map is created with its double capacity

-> Constructors of HashSet :-

1. public HashSet()
2. public HashSet(Object obj)
3. public HashSet(int initialCapacity)
4. public HashSet(int initialCapacity, float loadFactor)

-> Methods of HashSet :-

= approx same methods as that of Collection or Set interface

-> When we should use HashSet ?

= HashSet is good for searching or retriving operations

Q. Find the no. of distinct Elements

```
public class Test {
    public static void main(String[] args) {
        int [] A = {2,6,3,8,2,8,2,3,8} ;
        HashSet<Integer> set = new HashSet<>() ;
        for(int i=0  ; i<A.length ; i++){
            set.add(A[i]) ;
        }
        System.out.println(set);
    }
}
```

Q. Check if all the element are distinct or not

ex: 2 4 5 9. --> True

2 9 8 2 6. --> False

Q. Given 2 array if size N and M, Find the common elements in both array

A = [1, 2, 2, 1] ;

B = [2, 3, 1, 2] ;

```
public class Test {
    public static void main(String[] args) {
        int [] A = {1,2,2,1} ;
        int [] B = {2,3,1,2} ;
        ArrayList<Integer> list = new ArrayList<>() ;

        HashMap<Integer, Integer> map = new HashMap<>() ;
        for(int i=0 ; i<A.length ; i++){
            if(map.containsKey(A[i])){
                map.put(A[i], map.get(A[i])+1) ;
            }
            else{
                map.put(A[i], 1) ;
            }
        }
        for(int i=0 ; i<B.length ; i++){
            if(map.containsKey(B[i]) && map.get(B[i])>0){
                list.add(B[i]) ;
                map.put(B[i], map.get(A[i])-1) ;
            }
        }
        System.out.println(list);
    }
}
```