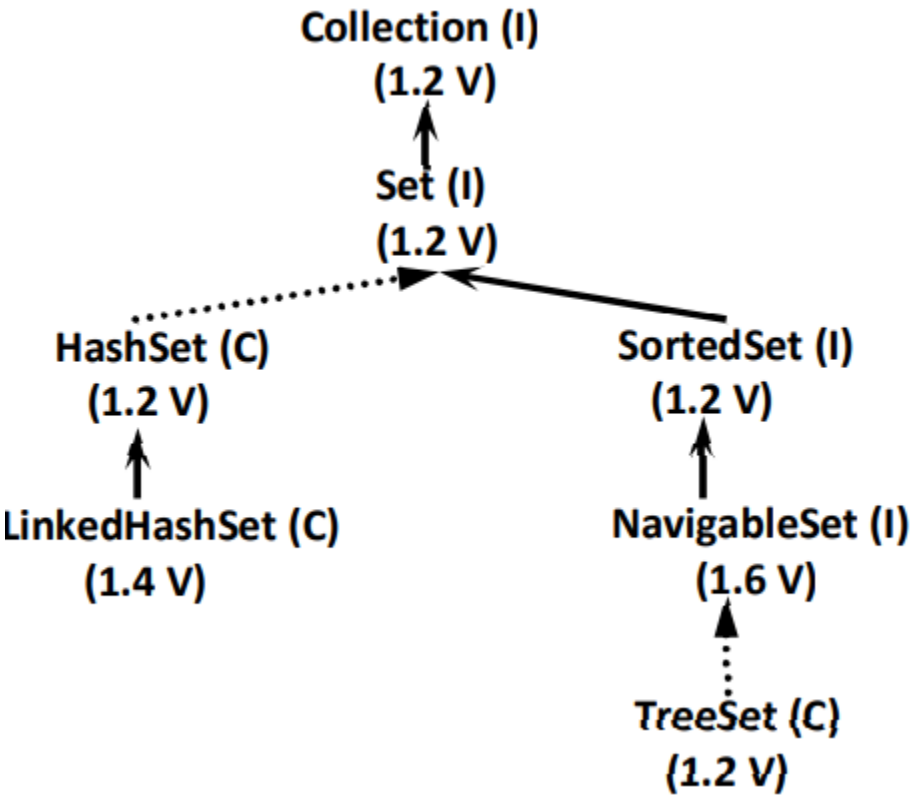
Set

Overview

set

- It is the **Child Interface of Collection**.
- If we want to Represent a Group of Individual Objects as a Single Entity where **Duplicates are Not allowed** and **Insertion Order is Not Preserved** then we should go for Set.
- **Set Interface doesn't contain any new Methods and Hence we have to Use only Collection Interface Methods.**



- Internally, Set uses a Map to store the elements.
- Map is an interface that is used to represent key-value pairs.
- To simplify, each key in a map is associated with a corresponding value.

Hashset

- The Underlying Data Structure is **Hashtable**.
- **Insertion Order is Not Preserved** and it is **Based on hashCode of the Objects**.
- **Duplicate Objects are Not Allowed**. If we are trying to Insert Duplicate Objects then we won't get any Compile Time OR Runtime Error. add() Simply Returns false
- **Null Insertion is Possible.**
- **Heterogeneous objects** are **allowed**.
- **HashSet implements Serializable** and **Cloneable** Interfaces but **Not RandomAccess**.
- If Our Frequent Operation is Search Operation, then HashSet is the Best Choice.

Constructor

1. HashSet<Integer> hashset = new HashSet<>();
   Creates an Empty HashSet Object with Default Initial Capacity 16 and
   Default Fill Ratio : 0.75

2. HashSet h = new HashSet(intinitialCapacity);
   Creates an Empty HashSet Object with specified Initial Capacity and Default Fill Ratio : 0.75
3. HashSet h = new HashSet(intinitialCapacity, float fillRatio);
4. HashSet h = new HashSet(Collection c);

Load Factor:
Fill Ratio 0.75 Means After Filling 75% Automatically a New HashSet Object will be Created.
This Factor is Called Fill RatioORLoad Factor.

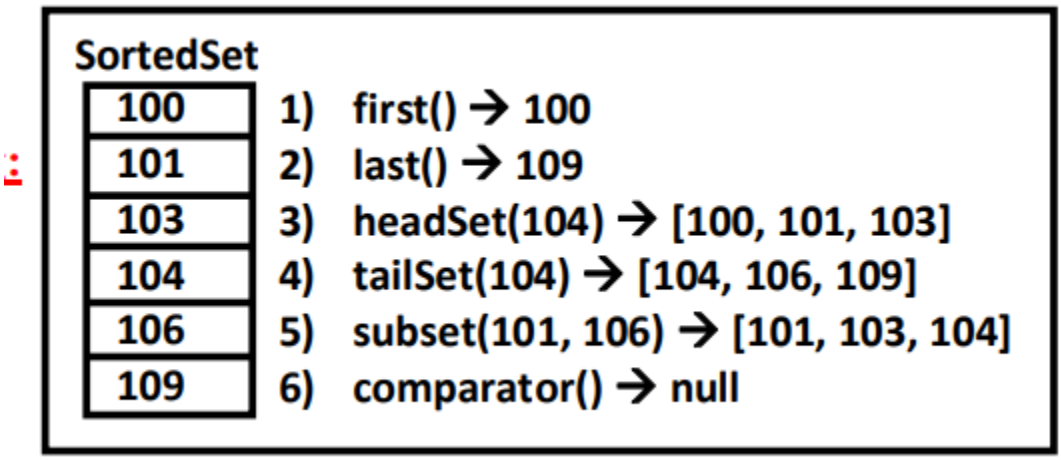
SortedSet

- It is a child interface of set.
- Duplicates are not allowed as it is a child interface of Set.
- Insert elements according to some Sorting Order.
- The Sorting can be Either Default Natural Sorting OR Customized Sorting Order.

  SortedSet<String> sortedSet = new TreeSet<>();

Methods

Sr No	Method	Description
1	Object first();	Returns 1 st Element of the SortedSet.
2	Object last()	Returns the last Element of the SortedSet.
3	SortedSet headSet(Object obj)	Returns SortedSet whose Elements are < Object
4	SortedSet tailSet(Object obj);	Returns SortedSet whose Elements are >= Object.
5	SortedSet subSet(Object obj1, Object obj2)	Returns SortedSet whose Elements are >= obj1 and <obj2.
6	Comparator comparator();	Returns Comparator Object that Describes Underlying SortingTechnique. if we are using Default Natural Sorting Order then we will get null.



TreeSet

- The Underlying Data Structure is a Balanced Tree.
- Insertion Order is Not Preserved.
- It is Based on Some Sorting Order.
- Heterogeneous Objects are Not Allowed. If we are trying to Insert we will get Runtime Exception Saying ClassCastException.
- Duplicate Objects are Not allowed.
- null Insertion is Possible (Only Once).
- Implements Serializable and Cloneable Interfaces but Not RandomAccess Interface.

Constructor

- TreeSet t = new TreeSet();
  Creates an Empty TreeSet Object where all Elements will be Inserted According to Default Natural Sorting Order
- TreeSet t = new TreeSet(Comparator c);
  Creates an Empty TreeSet Object where all Elements will be Inserted According to Customized Sorting Order which is described by Comparator Object.
- TreeSet t = new TreeSet(Collection c);
- TreeSet t = new TreeSet(SortedSet s);

null Acceptance:
For Empty TreeSet as the 1st Element null Insertion is Possible.
But after inserting that null if we are trying to Insert any Element we will get a NullPointerException.
For Non-Empty TreeSet if we are trying to Insert null we will get NullPointerException.

Default Sorting Order

- The Object should be Homogenous and Comparable. i.e., it should implement a Comparable interface. Otherwise we will get a ClassCastException.
- An object is said to be Comparable if and only if corresponding class implements Comparable interface.
- All Wrapper Classes, String Class Already Implements Comparable Interface. But StringBuffer Class doesn't Implement Comparable Interface.

Comparison Table of Set implemented Classes:

Property	HashSet	LinkedHashSet	TreeSet
Underlying Data Structure	Hashtable	Hashtable and LinkedList	Balanced Tree
Insertion Order	Not Preserved	Preserved	Not Preserved
Sorting Order	Not Applicable	Not Applicable	Applicable
Heterogeneous Objects	Allowed	Allowed	Not Allowed
Duplicate Objects	Not Allowed	Not Allowed	Not Allowed
null Acceptance	Allowed (Only Once)	Allowed (Only Once)	For Empty TreeSet as the 1st Element null Insertion is Possible. In all Other Cases we will get NullPointerException.

Comparable

- Present in the j**ava.lang** package.
- Contain only one method compareTo() [it is a functional interface]

  obj1.**compareTo**(obj2)

  Returns –ve if and Only if obj1 has to Come Before obj2
  Returns +ve if and Only if obj1 has to Come After obj2
  Returns 0 if and Only if obj1 and obj2are Equal

Comparator

- Present in the **java.util** package.
- We can define our own Sorting Order by using **Comparator** Object.
- Contains 2 methods
  - compare(Object o1, Object o2)
  - equals(Object o)
- Whenever we are implementing Comparator Interface Compulsory we should Provide Implementation for compare().
- Implementing equals() is Optional because it is Already Available to Our Class from Object Class through Inheritance.

  public int compare(Object obj1, Object obj2);
  Returns –ve if and Only if obj1 has to Come Before obj2.
  Returns +ve if and Only if obj1 has to Come After obj2.
  Returns 0 if and Only if obj1 and obj2 are Equal.