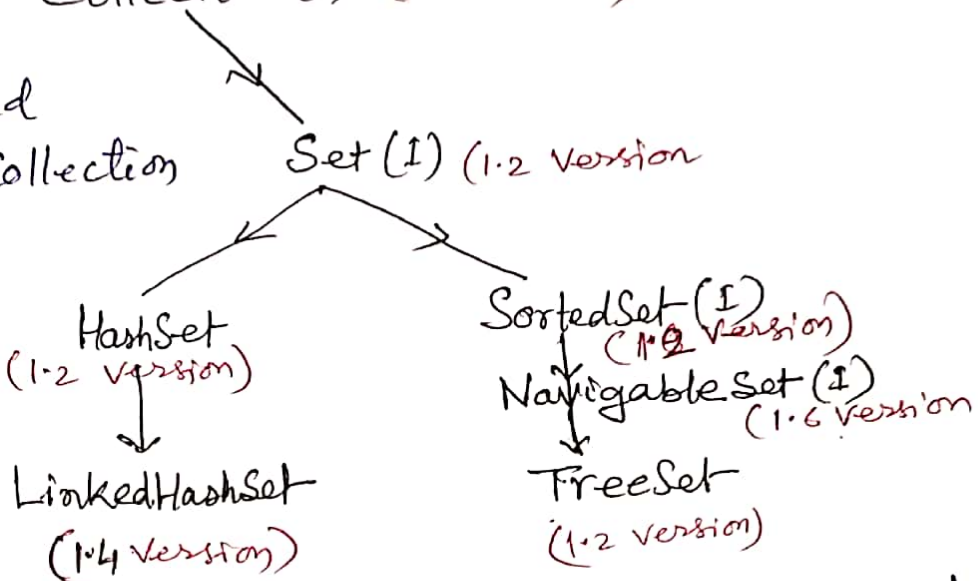
22-05-2020

Today we will going to discuss about the second half of Collection framework - **Set** interface.



→ Set is the child interface of Collection interface.

→ Set interface does not contain any new methods, so we have only Collection interface methods for use with Set interface.

→ 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 have to use Set.

HashSet — It is first class implementing the Set interface. Duplicates are not allowed. If we by mistake enter any duplicates by add()-method then — false is returned. (no error of Compile time)

→ The underlying data structure is Hashtable.

→ Insertion order is not preserved and all objects will be inserted based on hash-code of objects.

→ Heterogeneous objects are allowed. That means different type objects can be inserted.

→ null insertion is possible.

→ implements Serializable, Clonable interfaces but not RandomAccess.

→ HashSet is the best choice, if our frequent operation is search operation.

Constructors of HashSet —

① HashSet h = new HashSet();

where default initial capacity = 16

Fill Ratio = 0.75

or

Load Factor

② HashSet h = new HashSet(int initialCapacity);

→ creates an empty HashSet object with specified initial capacity & fill ratio 0.75

③ HashSet h = new HashSet(int initialCapacity, float loadFactor);

④ HashSet h = new HashSet(Collection c);

→ 3rd Constructor creates an empty HashSet object with specified initial capacity and specified Load Factor or Fill Ratio.

→ For inter conversion between Collection objects.

Load Factor / Fill Ratio — After loading the how much factor, a new HashSet object will be created, that factor is called Load Factor.

Let us observe the above concepts with a simple example :— HashSetDemo.java

```
import java.util.*;
class HashSetDemo{
 public static void main(String args[])
 {
    HashSet h = new HashSet();
    h.add("Shershah");
    h.add("College");
    h.add("Sasaram");
    h.add(50);
    h.add(52.57);
    h.add(null);
  System.out.println(h.add(50)); //false
  System.out.println(h);
 }
}
```

- Output will be not according to insertion order.
- Duplicate entry is also not accepted here (50).
- Heterogeneous elements are also allowed — String, Integer, Float, null etc

Page 1 of 1

LinkedHashSet

It is child class of HashSet, came in version 1.4.

Insertion order preserved and doesnot allows duplicate entry. It is same as HashSet except following differences:

HashSet	LinkedHashSet
① The underlying data structure is Hashtable	① The underlying data structure is Hashtable + LinkedList (Hybrid data structure)
② Insertion order is not preserved.	② Insertion order is preserved.
③ Introduced in Version 1.2	③ Introduced in version 1.4

Let us take an example: LinkedHashSetDemo.java

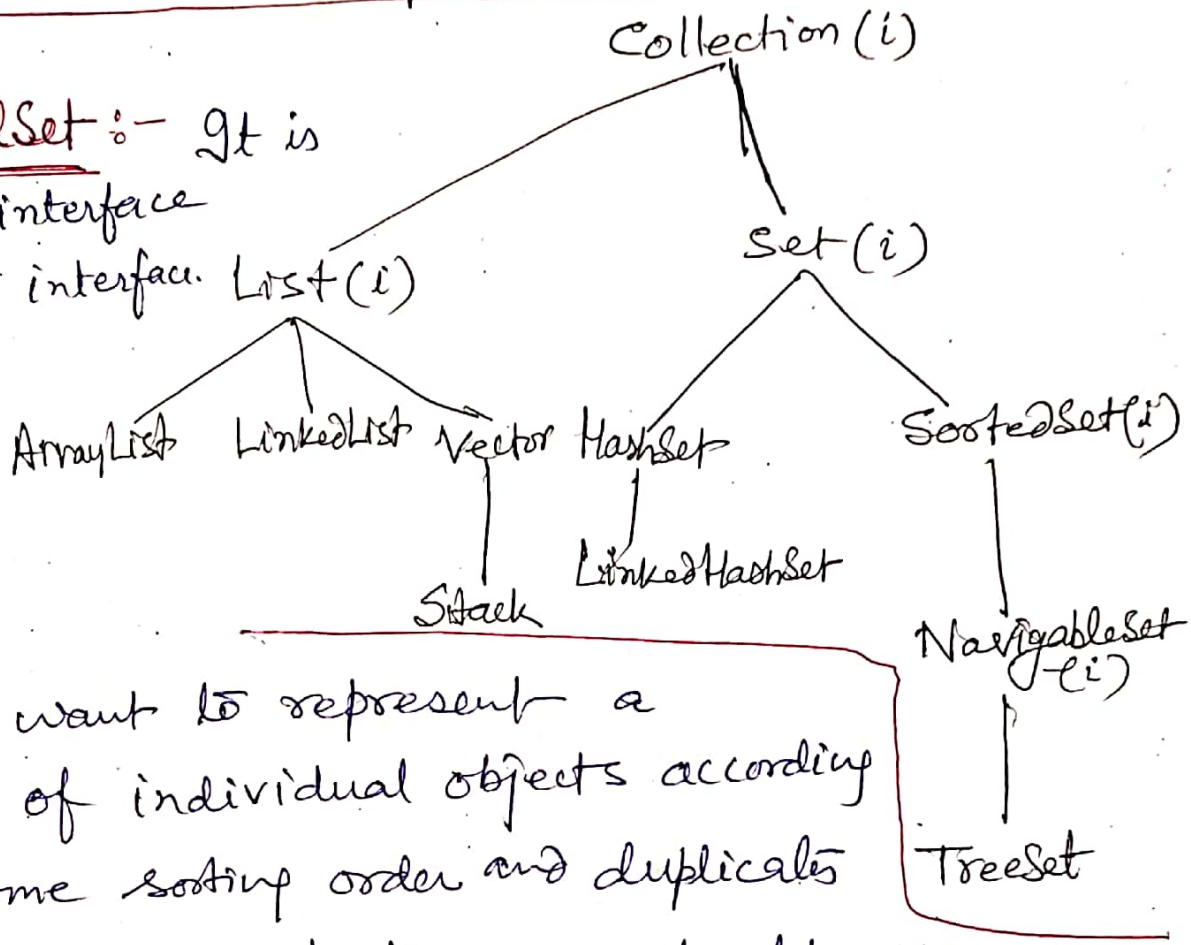
```
import java.util.*;
class LinkedHashSetDemo {
    public static void main(String args[])
    {
        LinkedHashSet h = new LinkedHashSet();
h.add("shershah"); h.add("college");
h.add("sasaram"); h.add(50); h.add(25.5);
h.add(null);
System.out.println(h.add(25.5)); //false
System.out.println(h);
    }
}
```

output:- [shershah, college, sasaram, 50, 25.5, h]

Page 1 of 1

LinkedHashSet —

Note:- LinkedHashSet is the best choice to develop cache based applications, where duplicates are not allowed and insertion order must be preserved.



SortedSet :- It is child interface of Set interface.

If we want to represent a group of individual objects according to some sorting order and duplicates are not allowed then we should use SortedSet.

→ Since Set interface does not have any new method so methods of Collection are used.

There are total 6 methods in SortedSet interface:-

① Object first() — returns first element of the SortedSet

② object last() — returns last element of the SortedSet.

Page 1 of 1

③ SortedSet headSet(Object ob) — returns the SortedSet whose elements are less than ob.

④ SortedSet tailSet(Object ob) — returns the SortedSet whose elements are >= ob

⑤ SortedSet subSet(Object ob1, Object ob2) — returns the SortedSet whose elements are >= ob1 and < ob2

⑥ Comparator comparator() — returns Comparator object that describes underlying sorting technique. If we are using default natural sorting order then we will get null.

Example :- {5, 7, 8, 11, 24, 26, 29}

1. first() → 5
2. last() → 29
3. headSet(24) → [5 7 8 11]
4. tailSet(24) → [24, 26, 29]
5. subSet(7, 26) → [7, 8, 11, 24]
6. comparator() → null

Note :- We can apply the above methods only on SortedSet implemented class objects — i.e; TreeSet class. (as seen before in diagram)

Page 1 of 1

TreeSet :—

1. Underlying data structure of TreeSet is Balanced Tree
2. Duplicates are not allowed.
3. Insertion order is not preserved but all objects will be inserted according to some sorting order.
4. Sorting order
5. Heterogeneous objects are not allowed. If someone tries — then will get — ClassCastException.
6. null acceptance is only once allowed.

Constructors of TreeSet :—

① TreeSet t = new TreeSet();
creates object where elements are inserted according to default sorting order (natural).

② TreeSet t = new TreeSet(Comparator c);
creates object where elements are inserted in customized sorting order — described by Comparator object.

③ TreeSet t = new TreeSet(Collection c);
creates object of Collection object.

④ TreeSet t = new TreeSet(SortedSet ob);
creates object of SortedSet object.

Let us consider a small example :— TreeSetDemo.java

```
import java.util.*;
class TreeSetDemo{
    public static void main(String args[])
    {
```


Page 1 of 1

```
TreeSet t = new TreeSet();
    t.add("sasaram");
    t.add("shershah");
    t.add("College");
    t.add("B+");
  // t.add(new Integer(10)); ClassCastException
  // t.add(null); Null pointer exception
System.out.println(t);
    }
}
```

output :-

[B+, College, sasaram, shershah]

Null Acceptance :-

① For empty TreeSet — as the first element null insertion is possible. But after inserting that null if we are trying to insert any another element we will get NullPointerException

② For non empty TreeSet if we are trying to insert Null then we will get NULLPointerException.

— END —

Page 1 of 1