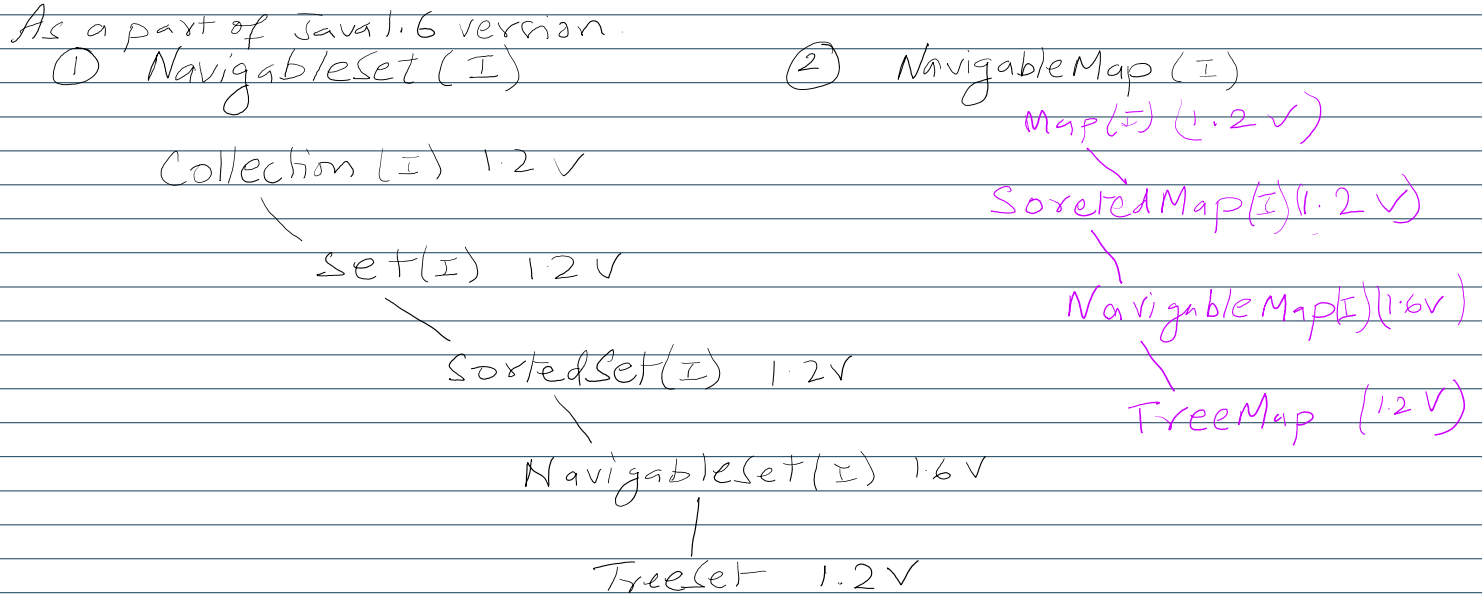
Enhancement in Collection 1.6

26 June 2025 18:33



NavigableSet interface: It is a child interface of Sorted Set and it defines several methods for navigation purpose.

NavigableSet defines the following methods:

floor(<E>) ==> It returns highest element which is <= E.
lower(<E>) ==> It returns highest element which is < E.
ceiling(<E>) ==> It returns lowest element which is >= E.
higher(<E>) ==> It returns lowest element which is > E.

pollFirst() ==> Remove and return first element.
pollLast() ==> Remove and return last element.

descendingSet() ==> It returns navigable set in reverse order.

Methods from SortedSet: first(), last(), headSet(), tailSet(), subSet(),…

TreeSet <E> t = new TreeSet<E>();

```
package EnhansementInJavaCollection;
import java.util.TreeSet;
public class NavigableSetDemo {
    public static void main(String[] args) {
        TreeSet<Integer> t = new TreeSet<Integer>();
        t.add(5000);
        t.add(4000);
        t.add(1000);
        t.add(6000);
        t.add(2000);
        t.add(3000);
        System.out.println(t); //[1000, 2000, 3000, 4000, 5000, 6000]
        System.out.println(t.ceiling(2000)); // 2000
        System.out.println(t.higher(2000)); // 3000
        System.out.println(t.floor(3000)); // 3000
        System.out.println(t.lower(3000)); // 2000
        System.out.println(t.pollFirst()); //1000
```

```
        System.out.println(t.pollFirst()); //1000
        System.out.println(t); //[2000, 3000, 4000, 5000, 6000]
        System.out.println(t.pollLast()); //6000
        System.out.println(t);//[2000, 3000, 4000, 5000]
        System.out.println(t.descendingSet()); //[5000, 4000, 3000, 2000]
        System.out.println("Original object: "+ t); //[2000, 3000, 4000, 5000]
    }
}
```

Output:

```
[1000, 2000, 3000, 4000, 5000, 6000]
2000
3000
3000
2000
1000
[2000, 3000, 4000, 5000, 6000]
6000
[2000, 3000, 4000, 5000]
[5000, 4000, 3000, 2000]
Original object: [2000, 3000, 4000, 5000]
```

NavigableMap (I)

NavigableMap (I)
Map (I) (1.2 V)
SortedMap (I) (1.2 V)
NavigableMap (I) (1.6 V)
TreeMap (1.2 V)

Dictionary (Java 1.0)
(key, value) are called Entry.
key = value

Python:
key: value
Dictionary (dict)

It is a child interface of SortedMap (I)
NavigableMap (I) defines the following methods

floorKey (e)
lowerKey (e)
ceilingKey (e)
higherKey ()
pollFirstEntry ()
pollLastEntry ()
descendingMap () → returns NavigableMap object

```
package EnhansementInJavaCollection;
```

```java
import java.util.TreeMap;
public class NavigableMapDemo {
    public static void main(String[] args) {
        TreeMap<String, String> t = new TreeMap<String, String>();
        t.put("C", "Cat");
        t.put("D", "Dog");
        t.put("B", "Boy");
        t.put("A", "Apple");
        t.put("G", "Gun");
        t.put("F", "Fish");
        t.put("E", "Elephant");
        System.out.println(t);
        System.out.println(t.ceilingKey("C")); //C
        System.out.println(t.higherKey("C")); //D
        System.out.println(t.floorKey("E")); //E
        System.out.println(t.lowerKey("E")); //D
        System.out.println(t.pollFirstEntry()); //A=Apple
        System.out.println(t.pollLastEntry()); //G=Gun
        System.out.println(t.descendingMap());//{F=Fish, E=Elephant, D=Dog, C=Cat, B=Boy}
        System.out.println(t); //{B=Boy, C=Cat, D=Dog, E=Elephant, F= Fish}
    }
}
```

Output:

```
{A=Apple, B=Boy, C=Cat, D=Dog, E=Elephant, F=Fish, G=Gun}
C
D
E
D
A=Apple
G=Gun
{F=Fish, E=Elephant, D=Dog, C=Cat, B=Boy}
{B=Boy, C=Cat, D=Dog, E=Elephant, F=Fish}
```

Official Documentation:

public interface NavigableMap<K,V>
extends SortedMap<K,V>

A SortedMap extended with navigation methods returning the closest matches for given search targets. Methods lowerEntry(K), floorEntry(K), ceilingEntry(K), and higherEntry(K) return Map.Entry objects associated with keys respectively less than, less than or equal, greater than or equal, and greater than a given key, returning null if there is no such key. Similarly, methods lowerKey(K), floorKey(K), ceilingKey(K), and higherKey(K) return only the associated keys. All of these methods are designed for locating, not traversing entries.

A NavigableMap may be accessed and traversed in either ascending or descending key order. The descendingMap() method returns a view of the map with the senses of all relational and directional methods inverted. The performance of ascending operations and views is likely to be faster than that of descending ones. Methods subMap(K, boolean, K, boolean), headMap(K, boolean), and tailMap(K, boolean) differ from the like-named SortedMap methods in accepting additional arguments describing whether lower and upper bounds are inclusive versus exclusive. Submaps of any NavigableMap must implement the NavigableMap interface.

This interface additionally defines methods firstEntry(), pollFirstEntry(), lastEntry(), and pollLastEntry() that return and/or remove the least and greatest mappings, if any exist, else returning null.

The methods ceilingEntry(K), firstEntry(), floorEntry(K), higherEntry(K), lastEntry(), lowerEntry(K), pollFirstEntry(), and pollLastEntry() return Map.Entry instances that represent snapshots of mappings as of the time of the call. They do not support mutation of the underlying map via the optional setValue method.

Methods subMap(K, K), headMap(K), and tailMap(K) are specified to return SortedMap to allow existing implementations of SortedMap to be compatibly retrofitted to implement NavigableMap, but extensions and implementations of this interface are encouraged to override these methods to return NavigableMap. Similarly, SortedMap.keySet() can be overridden to return NavigableSet.

This interface is a member of the Java Collections Framework.

Since:
1.6

https://docs.oracle.com/en/java/javase/24/docs/api/java.base/java/util/NavigableMap.html

Click the link to explore more.

Official Documentation of NavigableSet

public interface NavigableSet<E>
extends SortedSet<E>

A SortedSet extended with navigation methods reporting closest matches for given search targets. Methods lower(E), floor(E), ceiling(E), and higher(E) return elements respectively less than, less than or equal, greater than or equal, and greater than a given element, returning null if there is no such element.

A NavigableSet may be accessed and traversed in either ascending or descending order. The descendingSet() method returns a view of the set with the senses of all relational and directional methods inverted. The performance of ascending operations and views is likely to be faster than that of descending ones. This interface additionally defines methods pollFirst() and pollLast() that return and remove the lowest and highest element, if one exists, else returning null. Methods subSet(E, boolean, E, boolean), headSet(E, boolean), and tailSet(E, boolean) differ from the like-named SortedSet methods in accepting additional arguments describing whether lower and upper bounds are inclusive versus exclusive. Subsets of any NavigableSet must implement the NavigableSet interface.

The return values of navigation methods may be ambiguous in implementations that permit null elements. However, even in this case the result can be disambiguated by checking contains(null). To avoid such issues, implementations of this interface are encouraged to not permit insertion of null elements. (Note that sorted sets of Comparable elements intrinsically do not permit null.)

Methods subSet(E, E), headSet(E), and tailSet(E) are specified to return SortedSet to allow existing implementations of SortedSet to be compatibly retrofitted to implement NavigableSet, but extensions and implementations of this interface are encouraged to override these methods to return NavigableSet.

This interface is a member of the Java Collections Framework.

Since:
1.6

https://docs.oracle.com/en/java/javase/24/docs/api//java.base/java/util/NavigableSet.html

Click this link to know more about NavigableSet.