**Java NavigableMap Interface**

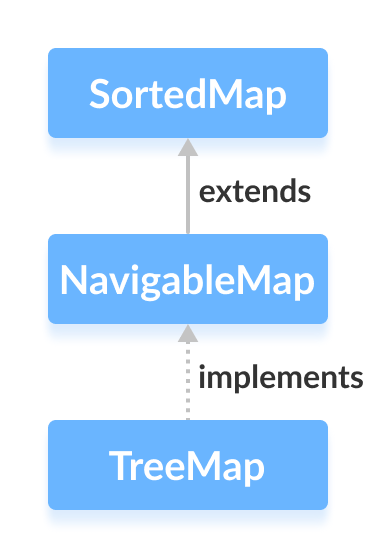
The NavigableMap interface of the Java collections framework provides the features to navigate among the map entries.

It is considered as a type of SortedMap.

## Class that implements NavigableMap

Since NavigableMap is an interface, we cannot create objects from it.

In order to use the functionalities of the NavigableMap interface, we need to use the TreeMap class that implements NavigableMap.



## How to use NavigableMap?

In Java, we must import the java.util.NavigableMap package to use NavigableMap. Once we import the package, here's how we can create a navigable map.

// NavigableMap implementation by TreeMap class

NavigableMap<Key, Value> numbers = new TreeMap<>();

In the above code, we have created a navigable map named numbers of the TreeMap class.

Here,

* Key - a unique identifier used to associate each element (value) in a map
* Value - elements associated by keys in a map

## Methods of NavigableMap

The NavigableMap is considered as a type of SortedMap. It is because NavigableMap extends the SortedMap interface.

Hence, all SortedMap methods are also available in NavigableMap. To learn how these methods are defined in SortedMap, visit Java SortedMap.

However, some of the methods of SortedMap (headMap(), tailMap(), and subMap() ) are defined differently in NavigableMap.

Let's see how these methods are defined in NavigableMap.

### headMap(key, booleanValue)

The headMap() method returns all the entries of a navigable map associated with all those keys before the specified key (which is passed as an argument).

The booleanValue is an optional parameter. Its default value is false.

If true is passed as booleanValue, the method returns all the entries associated with all those keys before the specified key, including the entry associated with the specified key.

### tailMap(key, booleanValue)

The tailMap() method returns all the entries of a navigable map associated with all those keys after the specified key (which is passed as an argument) including the entry associated with the specified key.

The booleanValue is an optional parameter. Its default value is true.

If false is passed as booleanValue, the method returns all the entries associated with those keys after the specified key, without including the entry associated with the specified key.

### subMap(k1, bv1, k2, bv2)

The subMap() method returns all the entries associated with keys between k1 and k2 including the entry associated with k1.

The bv1 and bv2 are optional parameters. The default value of bv1 is true and the default value of bv2 is false.

If false is passed as bv1, the method returns all the entries associated with keys between k1 and k2, without including the entry associated with k1.

If true is passed as bv2, the method returns all the entries associated with keys between k1 and k2, including the entry associated with k1.

## Other Methods

The NavigableMap provides various methods that can be used to locate the entries of maps.

* **descendingMap()** - reverse the order of entries in a map
* **descendingKeyMap()** - reverses the order of keys in a map
* **ceilingEntry()** - returns an entry with the lowest key among all those entries whose keys are greater than or equal to the specified key
* **ceilingKey()** - returns the lowest key among those keys that are greater than or equal to the specified key
* **floorEntry()** - returns an entry with the highest key among all those entries whose keys are less than or equal to the specified key
* **floorKey()** - returns the highest key among those keys that are less than or equal to the specified key
* **higherEntry()** - returns an entry with the lowest key among all those entries whose keys are greater than the specified key
* **higherKey()** - returns the lowest key among those keys that are greater than the specified key
* **lowerEntry()** - returns an entry with the highest key among all those entries whose keys are less than the specified key
* **lowerKey()** - returns the highest key among those keys that are less than the specified key
* **firstEntry()** - returns the first entry (the entry with the lowest key) of the map
* **lastEntry()** - returns the last entry (the entry with the highest key) of the map
* **pollFirstEntry()** - returns and removes the first entry of the map
* **pollLastEntry()** - returns and removes the last entry of the map

To learn more, visit Java NavigableMap official documentation.

## Implementation of NavigableMap in TreeMap Class

import java.util.NavigableMap;

import java.util.TreeMap;

class Main {

public static void main(String[] args) {

// Creating NavigableMap using TreeMap

NavigableMap<String, Integer> numbers = new TreeMap<>();

// Insert elements to map

numbers.put("Two", 2);

numbers.put("One", 1);

numbers.put("Three", 3);

System.out.println("NavigableMap: " + numbers);

// Access the first entry of the map

System.out.println("First Entry: " + numbers.firstEntry());

// Access the last entry of the map

System.out.println("Last Entry: " + numbers.lastEntry());

// Remove the first entry from the map

System.out.println("Removed First Entry: " + numbers.pollFirstEntry());

// Remove the last entry from the map

System.out.println("Removed Last Entry: " + numbers.pollLastEntry());

}

}

**Output**

NavigableMap: {One=1, Three=3, Two=2}

First Entry: One=1

Last Entry: Two=2

Removed First Entry: One=1

Removed Last Entry: Two=2

To learn more about TreeMap, visit Java TreeMap.

Now we know about the NavigableMap interface, we will learn about its implementation using the TreeMap class in detail in the next tutorial.