NavigableSet in Java with Examples

NavigableSet represents a navigable set in Java Collection Framework. The NavigableSet interface inherits from the SortedSet interface. It behaves like a SortedSet with the exception that we have navigation methods available in addition to the sorting mechanisms of the SortedSet. For example, NavigableSet interface can navigate the set in reverse order compared to the order defined in SortedSet.

The classes that implement this interface are, TreeSet and ConcurrentSkipListSet

Methods of NavigableSet (Not in SortedSet):

1. Lower(E e) : Returns the greatest element in this set which is less than the given element or NULL if there is no such element.
2. Floor(E e) : Returns the greatest element in this set which is less than or equal to given element or NULL if there is no such element.
3. Ceiling(E e) : Returns the least element in this set which is greater than or equal to given element or NULL if there is no such element.
4. Higher(E e) : Returns the least element in this set which is greater than the given element or NULL if there is no such element.
5. pollFirst() : Retrieve and remove the first least element. Or return null if there is no such element.
6. pollLast() : Retrieve and remove the last highest element. Or return null if there is no such element.

```
// A Java program to demonstrate working of SortedSet
import java.util.NavigableSet;
import java.util.TreeSet;

public class hashset
{
    public static void main(String[] args)
    {
        NavigableSet<Integer> ns = new TreeSet<>();
        ns.add(0);
        ns.add(1);
```

```
        ns.add(2);
        ns.add(3);
        ns.add(4);
        ns.add(5);
        ns.add(6);

        // Get a reverse view of the navigable set
        NavigableSet<Integer> reverseNs = ns.descendingSet();

        // Print the normal and reverse views
        System.out.println("Normal order: " + ns);
        System.out.println("Reverse order: " + reverseNs);

        NavigableSet<Integer> threeOrMore = ns.tailSet(3, true);
        System.out.println("3 or  more:  " + threeOrMore);
        System.out.println("lower(3): " + ns.lower(3));
        System.out.println("floor(3): " + ns.floor(3));
        System.out.println("higher(3): " + ns.higher(3));
        System.out.println("ceiling(3): " + ns.ceiling(3));

        System.out.println("pollFirst(): " + ns.pollFirst());
        System.out.println("Navigable Set:  " + ns);

        System.out.println("pollLast(): " + ns.pollLast());
        System.out.println("Navigable Set:  " + ns);

        System.out.println("pollFirst(): " + ns.pollFirst());
        System.out.println("Navigable Set:  " + ns);

        System.out.println("pollFirst(): " + ns.pollFirst());
        System.out.println("Navigable Set:  " + ns);

        System.out.println("pollFirst(): " + ns.pollFirst());
        System.out.println("Navigable Set:  " + ns);

        System.out.println("pollFirst(): " + ns.pollFirst());
        System.out.println("pollLast(): " + ns.pollLast());
    }
}
```

Run on IDE

Output:

```
Normal order: [0, 1, 2, 3, 4, 5, 6]
Reverse order: [6, 5, 4, 3, 2, 1, 0]
3 or  more:  [3, 4, 5, 6]
lower(3): 2
floor(3): 3
higher(3): 4
ceiling(3): 3
pollFirst(): 0
Navigable Set:  [1, 2, 3, 4, 5, 6]
pollLast(): 6
```