Java - The SortedSet Interface

The SortedSet interface extends Set and declares the behavior of a set sorted in an ascending order. In addition to those methods defined by Set, the SortedSet interface declares the methods summarized in the following table −

Several methods throw a NoSuchElementException when no items are contained in the invoking set. A ClassCastException is thrown when an object is incompatible with the elements in a set.

A NullPointerException is thrown if an attempt is made to use a null object and null is not allowed in the set.

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **Comparator comparator( )**  Returns the invoking sorted set's comparator. If the natural ordering is used for this set, null is returned. |
| 2 | **Object first( )**  Returns the first element in the invoking sorted set. |
| 3 | **SortedSet headSet(Object end)**  Returns a SortedSet containing those elements less than end that are contained in the invoking sorted set. Elements in the returned sorted set are also referenced by the invoking sorted set. |
| 4 | **Object last( )**  Returns the last element in the invoking sorted set. |
| 5 | **SortedSet subSet(Object start, Object end)**  Returns a SortedSet that includes those elements between start and end.1. Elements in the returned collection are also referenced by the invoking object. |
| 6 | **SortedSet tailSet(Object start)**  Returns a SortedSet that contains those elements greater than or equal to start that are contained in the sorted set. Elements in the returned set are also referenced by the invoking object. |

**Comparator comparator( )**

|  |  |
| --- | --- |
| **package** com.company; **import** java.util.\*;  **public class** Main {   **public static void** main(String[] args) {  *// Create the sorted set* SortedSet set = **new** TreeSet();   *// Add elements to the set* set.add(**"b"**);  set.add(**"c"**);  set.add(**"a"**);   *// Iterating over the elements in the set* Iterator it = set.iterator();   **while** (it.hasNext()) {  *// Get element* Object element = it.next();  System.***out***.println(element.toString());  }   *//Comparator -- return type Comparator* Comparator x = set.comparator();  **if** (x == **null**)  System.***out***.println(**"Natural comparation"**);  **else** System.***out***.println(**"Not natural"**);  } } | a  b  c  Natural comparation |
| // Java program to demonstrate working of Comparator  // interface  import java.util.\*;  import java.lang.\*;  import java.io.\*;    // A class to represent a student.  class Student  {      int rollno;      String name, address;        // Constructor      public Student(int rollno, String name,                                 String address)      {          this.rollno = rollno;          this.name = name;          this.address = address;      }        // Used to print student details in main()      public String toString()      {          return this.rollno + " " + this.name + " " + this.address;        }  }    class Sortbyroll implements Comparator<Student>  {      // Used for sorting in ascending order of      // roll number      public int compare(Student a, Student b)      {          return a.rollno - b.rollno;      }  }    class Sortbyname implements Comparator<Student>  {      // Used for sorting in ascending order of      // roll name      public int compare(Student a, Student b)      {          return a.name.compareTo(b.name);      }  }    // Driver class  class Main  {      public static void main (String[] args)      {          ArrayList<Student> ar = new ArrayList<Student>();          ar.add(new Student(111, "bbbb", "london"));          ar.add(new Student(131, "aaaa", "nyc"));          ar.add(new Student(121, "cccc", "jaipur"));            System.out.println("Unsorted");          for (int i=0; i<ar.size(); i++)              System.out.println(ar.get(i));            Collections.sort(ar, new Sortbyroll());            System.out.println("\nSorted by rollno");          for (int i=0; i<ar.size(); i++)              System.out.println(ar.get(i));            Collections.sort(ar, new Sortbyname());            System.out.println("\nSorted by name");          for (int i=0; i<ar.size(); i++)              System.out.println(ar.get(i));      }  } | Unsorted  111 bbbb london  131 aaaa nyc  121 cccc jaipur  Sorted by rollno  111 bbbb london  121 cccc jaipur  131 aaaa nyc  Sorted by name  131 aaaa nyc  111 bbbb london  121 cccc jaipu |

**Object first( )**

|  |  |
| --- | --- |
| **package** com.company; **import** java.util.\*;  **public class** Main {   **public static void** main(String[] args) {  *// Create the sorted set* SortedSet set = **new** TreeSet();   *// Add elements to the set* set.add(**"b"**);  set.add(**"c"**);  set.add(**"a"**);   Object ob = set.first();  System.***out***.println(**"first element is "** + ob);  } } | first element is a |

**SortedSet headSet(Object end) & Object last( ) & SortedSet subSet(Object start, Object end) & SortedSet tailSet(Object start)**

|  |  |
| --- | --- |
| **package** com.company; **import** java.util.\*;  **public class** Main {   **public static void** main(String[] args) {  String elements[] = { **"A"**, **"C"**, **"D"**, **"G"**, **"F"** };  TreeSet set = **new** TreeSet(Arrays.*asList*(elements));   System.***out***.println(set.tailSet(**"C"**));  System.***out***.println(set.headSet(**"C"**));  System.***out***.println(set.headSet(**"C\0"**));  System.***out***.println(set.tailSet(**"C\0"**));  System.***out***.println(set.subSet(**"C"**, **"F\0"**));  System.***out***.println(set.subSet(**"C"**, **"C\0"**));  System.***out***.println(set.subSet(**"C"**, **"C"**));  System.***out***.println(set.last());  System.***out***.println(set.pollLast());  } } | [C, D, F, G]  [A]  [A, C]  [D, F, G]  [C, D, F]  [C]  []  G  G |