| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SortedSet.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/util/SortedMap.html)   [**NEXT CLASS**](http://docs.google.com/java/util/Stack.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/SortedSet.html)    [**NO FRAMES**](http://docs.google.com/SortedSet.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

## **java.util**

Interface SortedSet<E>

**Type Parameters:**E - the type of elements maintained by this set **All Superinterfaces:** [Collection](http://docs.google.com/java/util/Collection.html)<E>, [Iterable](http://docs.google.com/java/lang/Iterable.html)<E>, [Set](http://docs.google.com/java/util/Set.html)<E> **All Known Subinterfaces:** [NavigableSet](http://docs.google.com/java/util/NavigableSet.html)<E> **All Known Implementing Classes:** [ConcurrentSkipListSet](http://docs.google.com/java/util/concurrent/ConcurrentSkipListSet.html), [TreeSet](http://docs.google.com/java/util/TreeSet.html)

public interface **SortedSet<E>**extends [Set](http://docs.google.com/java/util/Set.html)<E>

A [Set](http://docs.google.com/java/util/Set.html) that further provides a *total ordering* on its elements. The elements are ordered using their [natural ordering](http://docs.google.com/java/lang/Comparable.html), or by a [Comparator](http://docs.google.com/java/util/Comparator.html) typically provided at sorted set creation time. The set's iterator will traverse the set in ascending element order. Several additional operations are provided to take advantage of the ordering. (This interface is the set analogue of [SortedMap](http://docs.google.com/java/util/SortedMap.html).)

All elements inserted into a sorted set must implement the Comparable interface (or be accepted by the specified comparator). Furthermore, all such elements must be *mutually comparable*: e1.compareTo(e2) (or comparator.compare(e1, e2)) must not throw a ClassCastException for any elements e1 and e2 in the sorted set. Attempts to violate this restriction will cause the offending method or constructor invocation to throw a ClassCastException.

Note that the ordering maintained by a sorted set (whether or not an explicit comparator is provided) must be *consistent with equals* if the sorted set is to correctly implement the Set interface. (See the Comparable interface or Comparator interface for a precise definition of *consistent with equals*.) This is so because the Set interface is defined in terms of the equals operation, but a sorted set performs all element comparisons using its compareTo (or compare) method, so two elements that are deemed equal by this method are, from the standpoint of the sorted set, equal. The behavior of a sorted set *is* well-defined even if its ordering is inconsistent with equals; it just fails to obey the general contract of the Set interface.

All general-purpose sorted set implementation classes should provide four "standard" constructors: 1) A void (no arguments) constructor, which creates an empty sorted set sorted according to the natural ordering of its elements. 2) A constructor with a single argument of type Comparator, which creates an empty sorted set sorted according to the specified comparator. 3) A constructor with a single argument of type Collection, which creates a new sorted set with the same elements as its argument, sorted according to the natural ordering of the elements. 4) A constructor with a single argument of type SortedSet, which creates a new sorted set with the same elements and the same ordering as the input sorted set. There is no way to enforce this recommendation, as interfaces cannot contain constructors.

Note: several methods return subsets with restricted ranges. Such ranges are *half-open*, that is, they include their low endpoint but not their high endpoint (where applicable). If you need a *closed range* (which includes both endpoints), and the element type allows for calculation of the successor of a given value, merely request the subrange from lowEndpoint to successor(highEndpoint). For example, suppose that s is a sorted set of strings. The following idiom obtains a view containing all of the strings in s from low to high, inclusive:

SortedSet<String> sub = s.subSet(low, high+"\0");

A similar technique can be used to generate an *open range* (which contains neither endpoint). The following idiom obtains a view containing all of the Strings in s from low to high, exclusive:

SortedSet<String> sub = s.subSet(low+"\0", high);

This interface is a member of the  [Java Collections Framework](http://docs.google.com/technotes/guides/collections/index.html).

**Since:** 1.2 **See Also:**[Set](http://docs.google.com/java/util/Set.html), [TreeSet](http://docs.google.com/java/util/TreeSet.html), [SortedMap](http://docs.google.com/java/util/SortedMap.html), [Collection](http://docs.google.com/java/util/Collection.html), [Comparable](http://docs.google.com/java/lang/Comparable.html), [Comparator](http://docs.google.com/java/util/Comparator.html), [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html)

| **Method Summary** | |
| --- | --- |
| [Comparator](http://docs.google.com/java/util/Comparator.html)<? super [E](http://docs.google.com/java/util/SortedSet.html)> | [**comparator**](http://docs.google.com/java/util/SortedSet.html#comparator())()            Returns the comparator used to order the elements in this set, or null if this set uses the [natural ordering](http://docs.google.com/java/lang/Comparable.html) of its elements. |
| [E](http://docs.google.com/java/util/SortedSet.html) | [**first**](http://docs.google.com/java/util/SortedSet.html#first())()            Returns the first (lowest) element currently in this set. |
| [SortedSet](http://docs.google.com/java/util/SortedSet.html)<[E](http://docs.google.com/java/util/SortedSet.html)> | [**headSet**](http://docs.google.com/java/util/SortedSet.html#headSet(E))([E](http://docs.google.com/java/util/SortedSet.html) toElement)            Returns a view of the portion of this set whose elements are strictly less than toElement. |
| [E](http://docs.google.com/java/util/SortedSet.html) | [**last**](http://docs.google.com/java/util/SortedSet.html#last())()            Returns the last (highest) element currently in this set. |
| [SortedSet](http://docs.google.com/java/util/SortedSet.html)<[E](http://docs.google.com/java/util/SortedSet.html)> | [**subSet**](http://docs.google.com/java/util/SortedSet.html#subSet(E,%20E))([E](http://docs.google.com/java/util/SortedSet.html) fromElement, [E](http://docs.google.com/java/util/SortedSet.html) toElement)            Returns a view of the portion of this set whose elements range from fromElement, inclusive, to toElement, exclusive. |
| [SortedSet](http://docs.google.com/java/util/SortedSet.html)<[E](http://docs.google.com/java/util/SortedSet.html)> | [**tailSet**](http://docs.google.com/java/util/SortedSet.html#tailSet(E))([E](http://docs.google.com/java/util/SortedSet.html) fromElement)            Returns a view of the portion of this set whose elements are greater than or equal to fromElement. |

| **Methods inherited from interface java.util.**[**Set**](http://docs.google.com/java/util/Set.html) |
| --- |
| [add](http://docs.google.com/java/util/Set.html#add(E)), [addAll](http://docs.google.com/java/util/Set.html#addAll(java.util.Collection)), [clear](http://docs.google.com/java/util/Set.html#clear()), [contains](http://docs.google.com/java/util/Set.html#contains(java.lang.Object)), [containsAll](http://docs.google.com/java/util/Set.html#containsAll(java.util.Collection)), [equals](http://docs.google.com/java/util/Set.html#equals(java.lang.Object)), [hashCode](http://docs.google.com/java/util/Set.html#hashCode()), [isEmpty](http://docs.google.com/java/util/Set.html#isEmpty()), [iterator](http://docs.google.com/java/util/Set.html#iterator()), [remove](http://docs.google.com/java/util/Set.html#remove(java.lang.Object)), [removeAll](http://docs.google.com/java/util/Set.html#removeAll(java.util.Collection)), [retainAll](http://docs.google.com/java/util/Set.html#retainAll(java.util.Collection)), [size](http://docs.google.com/java/util/Set.html#size()), [toArray](http://docs.google.com/java/util/Set.html#toArray()), [toArray](http://docs.google.com/java/util/Set.html#toArray(T%5B%5D)) |

| **Method Detail** |
| --- |

### comparator

[Comparator](http://docs.google.com/java/util/Comparator.html)<? super [E](http://docs.google.com/java/util/SortedSet.html)> **comparator**()

Returns the comparator used to order the elements in this set, or null if this set uses the [natural ordering](http://docs.google.com/java/lang/Comparable.html) of its elements.

**Returns:**the comparator used to order the elements in this set, or null if this set uses the natural ordering of its elements

### subSet

[SortedSet](http://docs.google.com/java/util/SortedSet.html)<[E](http://docs.google.com/java/util/SortedSet.html)> **subSet**([E](http://docs.google.com/java/util/SortedSet.html) fromElement,  
 [E](http://docs.google.com/java/util/SortedSet.html) toElement)

Returns a view of the portion of this set whose elements range from fromElement, inclusive, to toElement, exclusive. (If fromElement and toElement are equal, the returned set is empty.) The returned set is backed by this set, so changes in the returned set are reflected in this set, and vice-versa. The returned set supports all optional set operations that this set supports.

The returned set will throw an IllegalArgumentException on an attempt to insert an element outside its range.

**Parameters:**fromElement - low endpoint (inclusive) of the returned settoElement - high endpoint (exclusive) of the returned set **Returns:**a view of the portion of this set whose elements range from fromElement, inclusive, to toElement, exclusive **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if fromElement and toElement cannot be compared to one another using this set's comparator (or, if the set has no comparator, using natural ordering). Implementations may, but are not required to, throw this exception if fromElement or toElement cannot be compared to elements currently in the set. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if fromElement or toElement is null and this set does not permit null elements [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if fromElement is greater than toElement; or if this set itself has a restricted range, and fromElement or toElement lies outside the bounds of the range

### headSet

[SortedSet](http://docs.google.com/java/util/SortedSet.html)<[E](http://docs.google.com/java/util/SortedSet.html)> **headSet**([E](http://docs.google.com/java/util/SortedSet.html) toElement)

Returns a view of the portion of this set whose elements are strictly less than toElement. The returned set is backed by this set, so changes in the returned set are reflected in this set, and vice-versa. The returned set supports all optional set operations that this set supports.

The returned set will throw an IllegalArgumentException on an attempt to insert an element outside its range.

**Parameters:**toElement - high endpoint (exclusive) of the returned set **Returns:**a view of the portion of this set whose elements are strictly less than toElement **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if toElement is not compatible with this set's comparator (or, if the set has no comparator, if toElement does not implement [Comparable](http://docs.google.com/java/lang/Comparable.html)). Implementations may, but are not required to, throw this exception if toElement cannot be compared to elements currently in the set. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if toElement is null and this set does not permit null elements [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if this set itself has a restricted range, and toElement lies outside the bounds of the range

### tailSet

[SortedSet](http://docs.google.com/java/util/SortedSet.html)<[E](http://docs.google.com/java/util/SortedSet.html)> **tailSet**([E](http://docs.google.com/java/util/SortedSet.html) fromElement)

Returns a view of the portion of this set whose elements are greater than or equal to fromElement. The returned set is backed by this set, so changes in the returned set are reflected in this set, and vice-versa. The returned set supports all optional set operations that this set supports.

The returned set will throw an IllegalArgumentException on an attempt to insert an element outside its range.

**Parameters:**fromElement - low endpoint (inclusive) of the returned set **Returns:**a view of the portion of this set whose elements are greater than or equal to fromElement **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if fromElement is not compatible with this set's comparator (or, if the set has no comparator, if fromElement does not implement [Comparable](http://docs.google.com/java/lang/Comparable.html)). Implementations may, but are not required to, throw this exception if fromElement cannot be compared to elements currently in the set. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if fromElement is null and this set does not permit null elements [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if this set itself has a restricted range, and fromElement lies outside the bounds of the range

### first

[E](http://docs.google.com/java/util/SortedSet.html) **first**()

Returns the first (lowest) element currently in this set.

**Returns:**the first (lowest) element currently in this set **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this set is empty

### last

[E](http://docs.google.com/java/util/SortedSet.html) **last**()

Returns the last (highest) element currently in this set.

**Returns:**the last (highest) element currently in this set **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this set is empty

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SortedSet.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/util/SortedMap.html)   [**NEXT CLASS**](http://docs.google.com/java/util/Stack.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/SortedSet.html)    [**NO FRAMES**](http://docs.google.com/SortedSet.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).