compact1, compact2, compact3

java.util

Class TreeMap<K,V>

java.lang.Object
    java.util.AbstractMap<K,V>
        java.util.TreeMap<K,V>

Type Parameters:

-

-

All Implemented Interfaces:

, , , , , , ,

TreeMap<K,V>

,

, , ,

A Red-Black tree based implementation. The map is sorted according to the natural ordering of its keys, or by a provided at map creation time, depending on which constructor is used.

This implementation provides guaranteed log(n) time cost for the , , and operations. Algorithms are adaptations of those in Cormen, Leiserson, and Rivest's *Introduction to Algorithms*.

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

Note that this implementation is not synchronized. If multiple threads access a map concurrently, and at least one of the threads modifies the map structurally, it *must* be synchronized externally. (A structural modification is any operation that adds or deletes one or more mappings; merely changing the value associated with an existing key is not a structural modification.) This is typically accomplished by synchronizing on some object that naturally encapsulates the map. If no such object exists, the map should be "wrapped" using the . method. This is best done at creation time, to prevent accidental unsynchronized access to the map:

. ((...))

The iterators returned by the method of the collections returned by all of this

class's "collection view methods" are *fail-fast*: if the map is structurally modified at any time after the iterator is created, in any way except through the iterator's own method, the iterator will throw a . Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.

Note that the fail-fast behavior of an iterator cannot be guaranteed as it is, generally speaking, impossible to make any hard guarantees in the presence of unsynchronized concurrent modification. Fail-fast iterators throw on a best-effort basis. Therefore, it would be wrong to write a program that depended on this exception for its correctness: *the fail-fast behavior of iterators should be used only to detect bugs.*

All . pairs returned by methods in this class and its views represent snapshots of mappings at the time they were produced. They do **not** support the . method. (Note however that it is possible to change mappings in the associated map using .)

This class is a member of the Java Collections Framework.

Since:

1.

See Also:

, , , , , ,

Nested Class Summary

Nested classes/interfaces inherited from class java.util.AbstractMap

. , , . ,

Constructor Summary

Constructors

Constructor and Description
TreeMap() Constructs a new, empty tree map, using the natural ordering of its keys.
TreeMap(Comparator K) Constructs a new, empty tree map, ordered according to the given comparator.
TreeMap(Map K, V) Constructs a new tree map containing the same mappings as the given map, ordered according to the *natural ordering* of its keys.
TreeMap(SortedMap K, V) Constructs a new tree map containing the same mappings and using the same ordering as the specified sorted map.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`Map.Entry K,V`	`ceilingEntry(K )` Returns a key-value mapping associated with the least key greater than or equal to the given key, or if there is no such key.
`K`	`ceilingKey(K )` Returns the least key greater than or equal to the given key, or if there is no such key.
	`clear()` Removes all of the mappings from this map.
`Object`	`clone()` Returns a shallow copy of this instance.
`Comparator K`	`comparator()` Returns the comparator used to order the keys in this map, or if this map uses the **natural ordering** of its keys.
	`containsKey(Object )` Returns if this map contains a mapping for the specified key.
	`containsValue(Object )` Returns if this map maps one or more keys to the specified value.
`NavigableSet K`	`descendingKeySet()` Returns a reverse order `NavigableSet` view of the keys contained in this map.
`NavigableMap K,V`	`descendingMap()` Returns a reverse order view of the mappings contained in this map.
`Set Map.Entry K,V`	`entrySet()` Returns a `Set` view of the mappings contained in this map.
`Map.Entry K,V`	`firstEntry()` Returns a key-value mapping associated with the least key in this map, or if the map is empty.
`K`	`firstKey()` Returns the first (lowest) key currently in this map.
`Map.Entry K,V`	`floorEntry(K )` Returns a key-value mapping associated with the greatest key less than or equal to the given key, or if there is no such key.
`K`	`floorKey(K )` Returns the greatest key less than or equal to the given key, or if there is no such key.

or if there is no such key.

	forEach(**BiConsumer** **K**, **V**) Performs the given action for each entry in this map until all entries have been processed or the action throws an exception.
V	**get**(**Object**) Returns the value to which the specified key is mapped, or if this map contains no mapping for the key.
SortedMap K,V	**headMap**(**K**) Returns a view of the portion of this map whose keys are strictly less than .
NavigableMap K,V	**headMap**(**K** ,) Returns a view of the portion of this map whose keys are less than (or equal to, if is true) .
Map.Entry K,V	**higherEntry**(**K**) Returns a key-value mapping associated with the least key strictly greater than the given key, or if there is no such key.
K	**higherKey**(**K**) Returns the least key strictly greater than the given key, or if there is no such key.
Set K	**keySet**() Returns a **Set** view of the keys contained in this map.
Map.Entry K,V	**lastEntry**() Returns a key-value mapping associated with the greatest key in this map, or if the map is empty.
K	**lastKey**() Returns the last (highest) key currently in this map.
Map.Entry K,V	**lowerEntry**(**K**) Returns a key-value mapping associated with the greatest key strictly less than the given key, or if there is no such key.
K	**lowerKey**(**K**) Returns the greatest key strictly less than the given key, or if there is no such key.
NavigableSet K	**navigableKeySet**() Returns a **NavigableSet** view of the keys contained in this map.
Map.Entry K,V	**pollFirstEntry**() Removes and returns a key-value mapping associated with the least key in this map, or if the map is empty.
Map.Entry K,V	**pollLastEntry**() Removes and returns a key-value mapping associated with the greatest key in this map, or if the map is empty.

the greatest key in this map, or if the map is empty.

V	**put**(K , V) Associates the specified value with the specified key in this map.
	putAll(Map K, V) Copies all of the mappings from the specified map to this map.
V	**remove**(Object) Removes the mapping for this key from this TreeMap if present.
V	**replace**(K , V) Replaces the entry for the specified key only if it is currently mapped to some value.
	replace(K , V , V) Replaces the entry for the specified key only if currently mapped to the specified value.
	replaceAll(BiFunction K, V, V) Replaces each entry's value with the result of invoking the given function on that entry until all entries have been processed or the function throws an exception.
	size() Returns the number of key-value mappings in this map.
NavigableMap K,V	**subMap**(K , , K ,) Returns a view of the portion of this map whose keys range from to .
SortedMap K,V	**subMap**(K , K) Returns a view of the portion of this map whose keys range from , inclusive, to , exclusive.
SortedMap K,V	**tailMap**(K) Returns a view of the portion of this map whose keys are greater than or equal to .
NavigableMap K,V	**tailMap**(K ,) Returns a view of the portion of this map whose keys are greater than (or equal to, if is true) .
Collection V	**values**() Returns a **Collection** view of the values contained in this map.

Methods inherited from class java.util.AbstractMap

, , ,