Map [Interface]: TreeMap

Definition

The TreeMap is used to implement Map interface and Navigable Map alon with the AbtstractMap Class. This is sorted according to the natural ordering of its keys, or by a comparator provided at map creation time, depending on which constructor is used. The storing order maintained by the TreeMap must be consistent with equals just like any other sorted map. This list is not synchronized in the sense that if a map is accessed by multiple threads, concurrently and at least one of the threads modifies the map structurally.

To prevent accidental unsynchronized access use Collections.synchronizedSortedMap method.

The methods in a TreeMap while getting keyset and values, return an iterator that is fail-fast in nature. Any concurrent modification will throw ConcurrentModificationException.

Each node in the tree has:

- 3 Variables (Key, value, color)
- 3 References (Entry left = left, Entry right = right, Entry parent = parent)
-

Features of a TreeMap

- TreeMap does not allow null keys like Map and thus a NullPointerException is thrown.
- Multiple null values can be associated with differente keys.
- Entry pairs returned by these methods represents snapshots of mappings at the time they were produced.

Constructor summary

TreeMap ()	Constructs an empty treemap that will be stored in default natural sorting order.
TreeMap(Comparator comp)	Constructs an empty TreeMap Object in which elements will need an external specification of the sorting order.
TreeMap(Map M)	Constructs TreeMap object with the entries from the given map M wich will be sorted by using natural order of the keys.
TreeMap(SortedMap sm)	Constructs an Treemap Object with the entries from the given sorted map which will be sorted in the same order as the given sorted map.

Methods summary

Modifier and Type	**Method and Description**
void	**Clear**()

	Removes all mappings from this map.
Comparator<? super K>	**comparator**() Returns the comparator used to order the keys in this map, or null if this map uses the natural ordering of its keys.
boolean	**containsKey**(Object key) Returns true if this map contains a mapping for the specified key.
V	**get**(Object key) Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
Set<K>	**keySet**() Returns a Set view of the keys contained in this map.
V	**put**(K key, V value) Associates the specified value with the specified key in this map.
V	**remove**(Object key) Removes the mapping for this key from this TreeMap if present.
Collection<V>	**values**() Returns a Collection view of the values contained in this map.

When is its use recommended?

- When needs to map Objects and want periodical output objects in alphabetical order by name.
- When it wants to offer a way to given a name, output the next 10 objects in alphabetical order. Likely a “more” function.
- When it want to get keys back in their true/natural order.
- TreeMap is used for sales data. As they capture relative size of data categories, allowing for quick perception of the items that have more contribution to each category.