Class TreeMap

java.util
Class TreeMap<K,V>

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

Parametre tipleri:

K - the type of keys maintained by this map
V - the type of mapped values

Kılgıladığı arayüzler:

Serializable, Cloneable, Map<K,V>, NavigableMap<K,V>, SortedMap<K,V>

Bildirimi:

public class **TreeMap<K,V>**
extends AbstractMap<K,V>
implements NavigableMap<K,V>, Cloneable, Serializable

TreeMap sınıfı Java Collections Framework 'un bir üyesidir.

`NavigableMap` 'e dayalı bir `kırmızı-siyah` (red-black) ağaç yapısıdır. Gönderim anahtarların *doğal sırasına* göre ya da kullanılan kurucuya bağlı olarak `Comparator`'un belirlediği sıraya göre sıralanır.

`NavigableMap` yapısında *containsKey(), get(), put()* ve *remove()* metotları için zaman karmaşası *log(n)* dir.

Bu yapıda oluşturulacak sıranın equals() ile tutarlı olması gerekir. Ayrıca bu yapının senkronize olmadığını anımsamak gerekir. Bu demektir ki, iki thread aynı zamanda erişim sağlıyor ve onlardan birisi map yapısını değiştiriyorsa, yapıda olmayan senkronize işleminin dışarıdan yapılması gerekir.

İlgili konular:

Map, HashMap, Hashtable, Comparable, Comparator, Collection, Serialized Form

---

Nested Class Summary

java.util.AbstractMap sınıfından kalıtsal alınanlar
AbstractMap.SimpleEntry<K,V>, AbstractMap.SimpleImmutableEntry<K,V>

`TreeMap` **Sınıfının Kurucuları**
TreeMap()

Anahtarların doğal sırasında boş bir ağaç gönderimi (tree map) yaratır.
TreeMap(Comparator<? super K> comparator) Comparator'un belirlediği sırada boş bir ağaç gönderimi (tree map) yaratır.
TreeMap(Map<? extends K,? extends V> m) Verilen map'i içeren, anahtarların doğal sırasında bir ağaç gönderimi (tree map) yaratır.
TreeMap(SortedMap<K,? extends V> m) Verilen map'i içeren ve verilen map'in belirlediği sırada bir ağaç gönderimi (tree map) yaratır.

`TreeMap` **Sınıfının Metotları**	
Map.Entry<K,V>	**ceilingEntry**(K key) Verilen anahtara eşit ya da ondan büyük olan en küçük anahtarla eşleşmiç *anahtar-değer* çiftini verir. Böyle bir anahtar yoksa `null` verir.
K	**ceilingKey**(K key) Verilen anahtara eşit ya da ondan büyük olan en küçük anahtarı verir. Böyle bir anahtar yoksa `null` verir.
void	**clear**() Gönderimin bütün öğelerini siler.
Object	**clone**() TreeMap nesnesinin bir kopyasını yapar.
Comparator<? super K>	**comparator**() Bu gönderimde anahtarları sıralamak için kullanılan comparator'u verir. Eğer anahtarlar doğal sıralı iseler `null` değer verir.
boolean	**containsKey**(Object key) Verilen anahtara eşleşen bir gönderim varsa `true` değerini verir.
boolean	**containsValue**(Object value) Verilen öğeye eşleşen bir ya da birden çok anahtar varsa `true` değeri verir.
NavigableSet<K>	**descendingKeySet**() Map içindeki anahtarları ters sırada verir.
NavigableMap<K,V>	**descendingMap**() Yapıdaki gönderimleri ters sırada verir.
Set<Map.Entry<K,V>>	**entrySet**() Yapıdaki gönderimleri bir küme olarak verir.
Map.Entry<K,V>	**firstEntry**() Yapıdaki en küçük anahtarla eşleşen anahtar-değer çiftini verir. Gönderim boş ise null değer verir.
K	**firstKey**() Yapıdaki en küçük (ilk) anahtarı verir.
Map.Entry<K,V>	**floorEntry**(K key) Verilen anahtardan küçük olmayan anahtarla eşleşen ilk *anahtar-değer* çiftini verir. Böyle bir anahtar yoksa `null` değer verir.

K	**floorKey**(K key) Verilen anahtardan küçük olmayan ilk anahtarı verir. Böyle bir anahtar yoksa `null` değer verir.
V	**get**(Object key) Verilen anahtara eşleşen değeri verir. Eğer verilen anahtara eşleşen öğe yoksa `null` verir.
SortedMap<K,V>	**headMap**(K toKey) Verilen anahtardan daha küçük anahtarlarla eşleşen gönderimleri verir.
NavigableMap<K,V>	**headMap**(K toKey, boolean inclusive) `inclusive true` ise verilen anahtardan büyük olmayan anahtarlarla eşleşen gönderimleri verir.
Map.Entry<K,V>	**higherEntry**(K key) Verilen anahtardan büyük olan anahtarların en küçüğü ile eşleşen *anahtar-değer* çiftini verir. Böyle biri yoksa `null` verir.
K	**higherKey**(K key) Verilen anahtardan büyük olan anahtarların en küçüğünü verir. Böyle biri yoksa `null` verir.
Set<K>	**keySet**() Bu gönderimdeki anahtarları bir `küme (Set)` olarak gösterir.
Map.Entry<K,V>	**lastEntry**() En büyük anahtarla eşleşen *anahtar-değer* çiftini verir. Map boş ise `null` verir.
K	**lastKey**() Map içindeki en büyük (son) anahtarı verir.
Map.Entry<K,V>	**lowerEntry**(K key) Verilen anahtardan küçük olan anahtarların en büyüğü ile eşleşen *anahtar-değer* çiftini verir. Böyle biri yoksa `null` verir.
K	**lowerKey**(K key) Verilen anahtardan küçük olan anahtarların en büyüğünü verir. Böyle biri yoksa `null` verir.
NavigableSet<K>	**navigableKeySet**() Map içindeki anahtarların bir Navigable kümesini verir.
Map.Entry<K,V>	**pollFirstEntry**() Yapı içinde en küçük (ilk) anahtarla eşleşen *anahtar-değer* çiftini verir ve onu yapıdan siler. Yapı boş ise `null` verir.
Map.Entry<K,V>	**pollLastEntry**() Yapı içinde en büyük (son) anahtarla eşleşen *anahtar-değer* çiftini verir ve onu yapıdan siler. Yapı boş ise `null` verir.
V	**put**(K key, V value) Verilen anahtarrı verilen öüeye eşler.
void	**putAll**(Map<? extends K,? extends V> map) Verilen mapi belirtilen mape kopyalar.
V	**remove**(Object key)

	Verilen anahtara ile eşleşen anahtar-değer çiftini TreeMap yapısından siler.
int	**size**() Map içindeki anahtar-değer çiftlerinin sayısını verir.
NavigableMap<K,V>	**subMap**(K fromKey, boolean fromInclusive, K toKey, boolean toInclusive) *toInclusive* değeri `true` ise verilen anahtarlar ile belirlenen aralıkta kalan gönderimleri verir. Aralığın uçları dahildir.
SortedMap<K,V>	**subMap**(K fromKey, K toKey) Verilen anahtarlar ile belirlenen aralıkta kalan gönderimleri verir. Aralığın alt ucu dahil, üst ucu hariçtir.
SortedMap<K,V>	**tailMap**(K fromKey) Verilen anahtardan küçük olmayanlarla eşleşen *anahtar-değer* çitlerini verir.
NavigableMap<K,V>	**tailMap**(K fromKey, boolean inclusive) Verilen anahtardan büyük olanlarla eşleşen *anahtar-değer* çitlerini verir. İnclusive true değer alıyorsa, verilen anahtara eşleşen çift de dahil olur.
Collection<V>	**values**() Map içindeki değerleri bir *Collection* olarak gösterir.

java.util.AbstractMap sınıfından kalıtsal gelen metotlar
equals, hashCode, isEmpty, toString

java.lang.Object sınıfından kalıtsal gelen metotlar
finalize, getClass, notify, notifyAll, wait, wait, wait

java.util.Map arayüzünden alınan metotlar
equals, hashCode, isEmpty

Örnek:

Aşağıdaki program illerin trafik kodlarını anahtar, adlarını değer olarak alıp anahtar-değer eşleşmesi yapan bir TreeMap yapısı kuruyor. Sonra TreeMap içindeki değerleri yazdırıyor.

```
import java.util.Collection;
import java.util.TreeMap;
import java.util.Iterator;

public class TreeMapDemo {

	public static void main(String[] args) {

		// TreeMap nesnesi yarat
		TreeMap treeMap = new TreeMap();

		// TreeMap nesnesine anahtar-değer öğelerini ekle
		treeMap.put("01", "Adana");
```

```
            treeMap.put("10", "Balıkesir");
            treeMap.put("17", "Çanakkale");
            treeMap.put("20", "Denizli");
            treeMap.put("22", "Edirne");

 //Collection içindeki values() metodu ile TreeMap içindeki değerleri al.

            Collection c = treeMap.values();

            // Collection için bir Iterator tanımla
            Iterator itr = c.iterator();

            // TreeMap 'in değerlerini iteratör ile tara
            while (itr.hasNext())
                  System.out.println(itr.next());
      }
}

     /*
     Çıktı:
      Adana
      Balıkesir
      Çanakkale
      Denizli
      Edirne
     */
```

Örnek:

Aşağıdaki program bir TreeMap yapısı kuruyor, yapıya veriler giriyor ve sınıfın bazı metotlarını uyguluyor.

```
import java.util.*;

public class TreeMapDemo {
      public static void main(String[] args) {
            System.out.println("TreeMap Örneği!\n");
            TreeMap<Integer, String> tMap = new TreeMap<Integer, String>();
            // TreeMap yapısına öğe ekleme
            tMap.put(1, "Pazartesi");
            tMap.put(2, "Salı");
            tMap.put(3, "Çarşamba");
            tMap.put(4, "Perşembe");
            tMap.put(5, "Cuma");
            tMap.put(6, "Cumartesi");
            tMap.put(7, "Pazar");
            // Yapıdaki anahtarları göster
            System.out.println("TreeMap 'in anahtarları: " + tMap.keySet());
            // Yapıdaki değerleri göster
            System.out.println("TreeMap'in değerleri   : " + tMap.values());
            // Anahtar 5 ile eşleşen değeri ver
            System.out.println("Anahtar 5 için değer: " + tMap.get(5) + "\n");
            // İlk anahtarı ve onunla eşleşen değeri göster
            System.out.println("İlk anahtar: " + tMap.firstKey() + " Value: "
                        + tMap.get(tMap.firstKey()) + "\n");
            // Son anahtarı ve onunla eşleşen değeri göster
            System.out.println("Son anahtar: " + tMap.lastKey() + " Value: "
                        + tMap.get(tMap.lastKey()) + "\n");
            // İlk anahtarla eşleşen veriyi sil
            System.out.println("Silinen ilk veri     : "
                        + tMap.remove(tMap.firstKey()));
            System.out.println("Geri kalan anahtarlar: " + tMap.keySet());
```

```
            System.out.println("Geri kalan değerler  : " + tMap.values() + "\n");
            // Son anahtarla eşleşen veriyi sil
            System.out.println("Silinen son veri     : "
                        + tMap.remove(tMap.lastKey()));
            System.out.println("Geri kalan anahtarlar: " + tMap.keySet());
            System.out.println("Geri kalan değerler  : " + tMap.values());
        }
}
```

```
/*
Çıktı:
TreeMap Örneği!

TreeMap 'in anahtarları: [1, 2, 3, 4, 5, 6, 7]
TreeMap'in değerleri   : [Pazartesi, Salı, Çarşamba, Perşembe, Cuma,
Cumartesi, Pazar]
Anahtar 5 için değer: Cuma

İlk anahtar: 1 Value: Pazartesi

Son anahtar: 7 Value: Pazar

Silinen ilk veri     : Pazartesi
Geri kalan anahtarlar: [2, 3, 4, 5, 6, 7]
Geri kalan değerler  : [Salı, Çarşamba, Perşembe, Cuma, Cumartesi, Pazar]

Silinen son veri     : Pazar
Geri kalan anahtarlar: [2, 3, 4, 5, 6]
Geri kalan değerler  : [Salı, Çarşamba, Perşembe, Cuma, Cumartesi]
*/
```