**简介**

TreeMap是一个有序的key-value集合，它是通过红黑树实现的。

TreeMap继承于AbstractMap，所以是个map。

TreeMap实现了NavigableMap接口，意味着它支持一系列的导航方法。

例如返回有序的key集合。

TreeMap是实现了Cloneable接口，意味着它能被克隆。

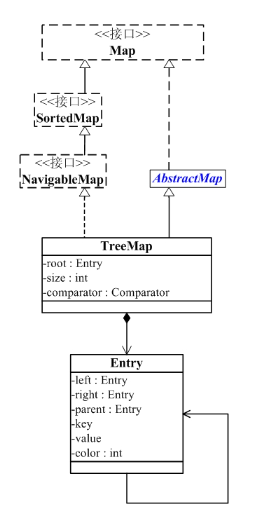
TreeMap实现了java.io.Serializable接口，意味着它支持序列化。

TreeMap是基于红黑树实现的。该映射是根据键的自然顺序排列的，或者根据创建映射时候提供的Comparator进行排序，具体取决它的构造方法。

TreeMap的基本操作containsKey，get，put，remove的时间复杂度都是log(n).

TreeMap是非同步的。它的Iterator方法返回的迭代器是fail-fast的。

**内部结构**



TreeMap继承了AbstractMap，实现了NavigableMap接口。

TreeMap本质是红黑树，它内部包含了root、size、comparator。

root：Entry类型的根节点

size：红黑树中节点个数

comparator：比较器，对key排序使用，默认为自然顺序

Entry内部包含了left、right、parent、key、value、color

left：左孩子域

right：右孩子域

parent：双亲域

key：键

value：值

color：节点颜色(B or R)

HashMap、HashTable内部包含的Entry类型的数组table[]，size，加载因子，阀il-fast比较参数，阈值,Entry内部结构为单链表，包含了下一个链节的指针next，hash值，key，value。

**构造函数**

1.TreeMap() //默认构造函数，数据元素按自然序排列

2.TreeMap(Map<? extends K, ? extends V> copyFrom) //创建的TreeMap包含无序Map

3.TreeMap(Comparator<? super K> comparator) //指定比较器

4. TreeMap(SortedMap<K, ? extends V> copyFrom) //有序的Map

**方法清单**

返回值 方法

Entry<K, V> ceilingEntry(K key)

K ceilingKey(K key)

void clear()

Object clone()

Comparator<? super K> comparator()

boolean containsKey(Object key)

NavigableSet<K> descendingKeySet()

NavigableMap<K, V> descendingMap()

Set<Entry<K, V>> entrySet()

Entry<K, V> firstEntry()

K firstKey()

Entry<K, V> floorEntry(K key)

K floorKey(K key)

V get(Object key)

NavigableMap<K, V> headMap(K to, boolean inclusive)

SortedMap<K, V> headMap(K toExclusive)

Entry<K, V> higherEntry(K key)

K higherKey(K key)

boolean isEmpty()

Set<K> keySet()

Entry<K, V> lastEntry()

K lastKey()

Entry<K, V> lowerEntry(K key)

K lowerKey(K key)

NavigableSet<K> navigableKeySet()

Entry<K, V> pollFirstEntry()

Entry<K, V> pollLastEntry()

V put(K key, V value)

V remove(Object key)

int size()

SortedMap<K, V> subMap(K fromInclusive, K toExclusive)

NavigableMap<K, V> subMap(K from, boolean fromInclusive, K to, boolean toInclusive)

NavigableMap<K, V> tailMap(K from, boolean inclusive)

SortedMap<K, V> tailMap(K fromInclusive)

Entry的相关方法

1.firstEntry() //获取第一个节点，也就是最左下的那个节点，key最小的节点

firstEntry是通过getfirstEntry方法来实现的。

FirstEntry是对getfirstEntry方法的一个封装，只提供了get方法，不可以调用getfirstEntry().set改变节点的值

2.lastEntry() //获取最后一个节点，也就是最右下的那个节点，key最大的节点

3. lowerEntry(key) //返回小于key的最大的节点

4. higherEntry(key) //返回大于key的最小的节点

5. floorEntry() //返回小于等于key的最大的节点

6. ceilingEntry() //返回大于等于key的最小的节点

6. pollFirstEntry() //移除并返回key最小的节点

7. pollLastEntry() //移除并返回key最大的节点

Key的相关方法

1.firstKey() //获取key最小的key

2.lastKey() //获取key最大的key

3.lowerKey() //返回小于key的最大的key

4.higherKey() //返回大于key的最小的key

5.floorKey() //返回小于等于key的最大的key

6.ceilingKey() //返回大于等于key的最小的key

Navigable接口其他方法（上述也都属于Navigable方法）

1. descendingMap() //返回当前tree的反向tree

//将比较器方向

2. descendingKeySet() //返回逆序key的set

3. headMap(toKey) //返回键值小于key的部分视图，也就是key的左子树

4. subMap(K fromKey, boolean fromInclusive, K toKey, boolean toInclusive)

//返回从fromkey到tokey的部分视图

5. subMap(K fromKey, K toKey) //返回从fromkey到tokey的部分视图，包含边界

6. tailMap(K fromKey)//返回大于等于fromkey的部分视图，右子树

4. navigableKeySet() //返回key的navigable类型的set

**遍历**

同HashMap