# TreeMap（jdk1.60\_13）

treeMap的实现是基于红黑树的，所以在阅读本文章之前需要对红黑树的原理及实现有一定的了解。

TreeMap有几个比较重要的属性：

# 属性



比较器，如果该比较器不为空，TreeMap将用这个比较器来维持内部的顺序。否则依赖key的自然顺序。



红黑树的根节点。

# 构造器

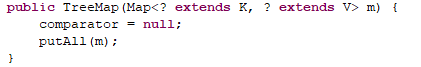


TreeMap默认的构造器初始化了comparator为null，所以当使用默认构造器创建TreeMap时，TreeMap内部元素的顺序是依赖key的自然顺序的。

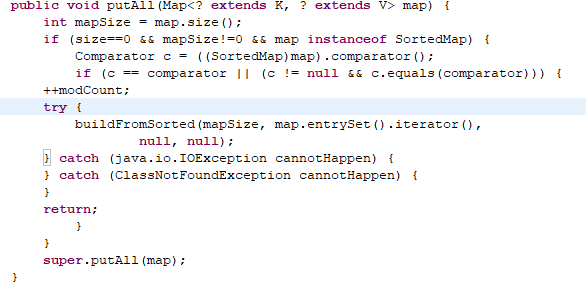


这个是指定比较器的构造器。

## 2.1TreeMap(Map)

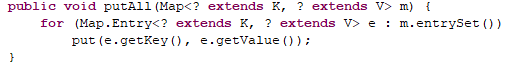


这个构造器的目的是将一个已经存在的Map中的所有元素放入新创建的TreeMap中，这里调用了putAll方法，我们来看一下：

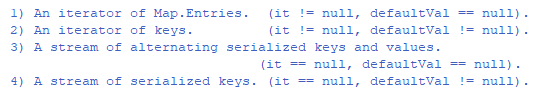


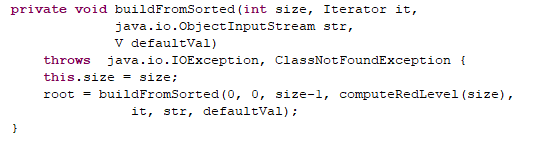
这里分为两种情况，第一种情况传入的Map本来就是一个有序的Map，并且内部的顺序是元素的自然顺序或者传入的Map与新创建的TreeMap使用相同的比较器，第二种情况传入的Map不是有序的或者传入的Map与新创建的TreeMap使用不同的比较器。

第二种情况比较简单，这里通过委托给父类AbstractMap的putAll来做处理。遍历整个Map，将每个元素put进去。



接下来看一下第一种情况，这里传入的Map是有序的，并且新创建的TreeMap要么使用和传入Map相同的比较器要么都没有使用比较器，在把元素put到新TreeMap中后，元素的顺序与以前在Map中的顺序肯定是相同的。先来看一下buildFromSorted这个方法。



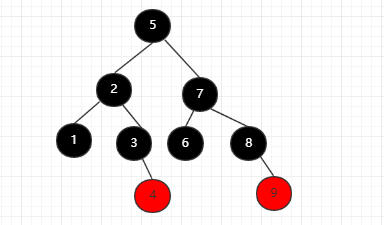


传入的参数有上面四种情况

1. it!=null && defaultVal == null，使用迭代器遍历元素。
2. it!=null && defaultVal!=null，使用迭代器遍历key，并用defaultVal替换原来的值。
3. it==null && defaultVal==null，使用key-value交替连续的流来遍历元素。
4. it==null&&defaultVal!=null，使用流来遍历key并用defaultVal替换原来的值。

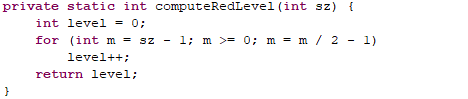
这里是第一种情况。computeRedLeval这个方法是用来计算生成的红黑树的层级的也可以说红黑树中红色节点所在的层级。

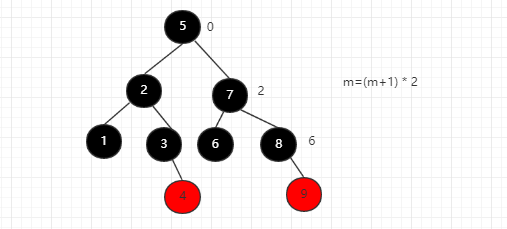
例如：有这样一个有序序列1、2、3、4、5、6、7、8、9，要生成一颗红黑树。可以这样组织一棵树，满足红黑树的特征。



将最后一层节点设置为红色，其他设置为黑色。

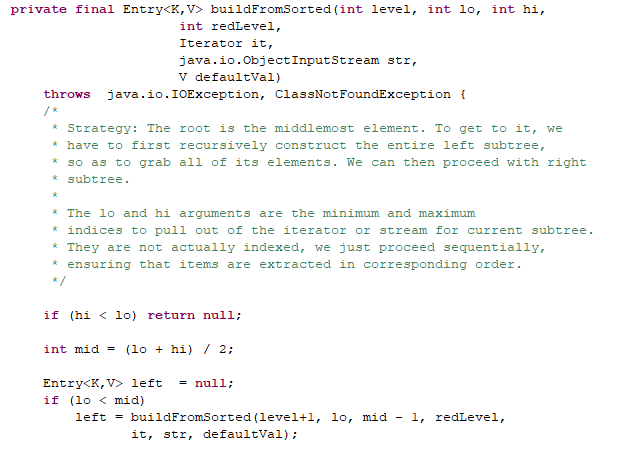
看一下computeRedLevel是怎样计算二叉树高度的。

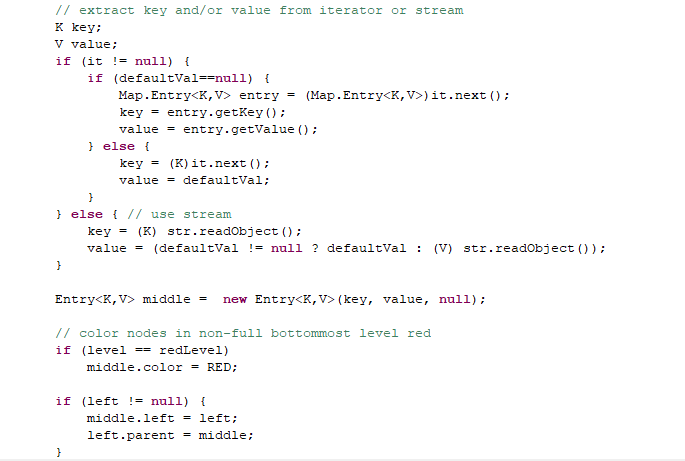


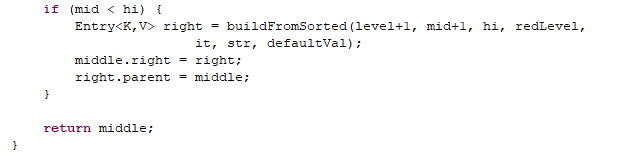


可以看到，如果以根节点作为0号索引，第二层最右边为2号索引...正好是上一层+1再\*2。

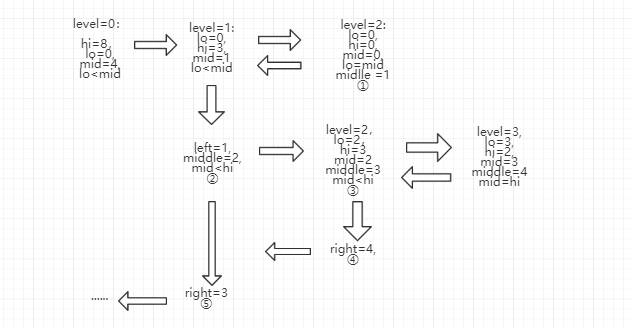
在回到buildFromSorted方法，调用了重载的builFromSorted的方法。



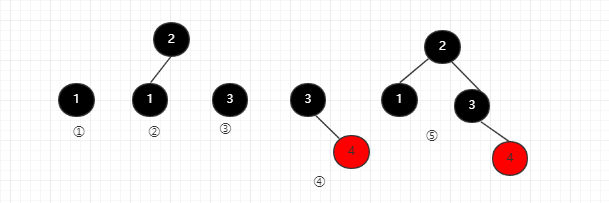




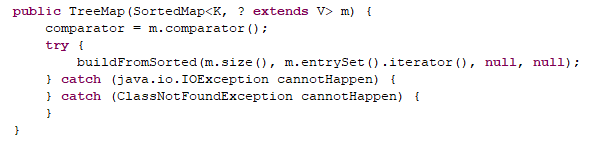
最中间的节点为根节点，为了获取根节点必须递归的构造左子树，然后构造右子树。例如：1、2、3、4、5、6、7、8、9这样一个有序的key的构造过程如下所示。



其中①~⑤树的结构图如下所示：



## 2.2TreeMap(SortedMap)

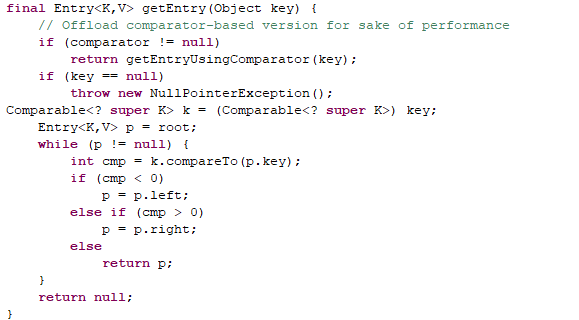
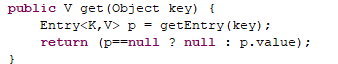


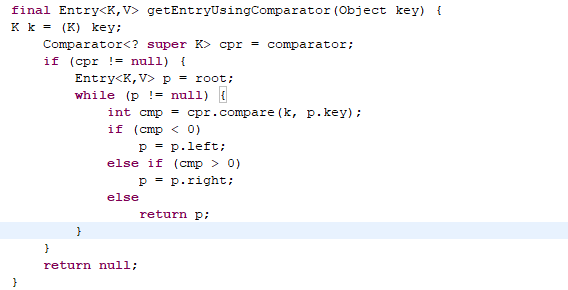
buildFromSorted方法见2.1章节所述。

# 基本方法

TreeMap的put跟remove方法就是红黑树的插入删除操作，详情请见红黑树.docx，这里主要看一下其他的方法。

## 3.1get





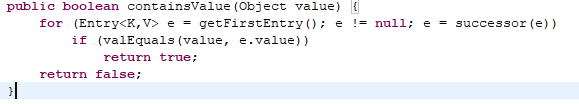
getEntry方法首先判断比较器是否为空，如果不为空，则通过比较器比较并获取元素，否则使用元素的自然顺序，主要这里是将key强转为Comparable类型，如果在创建TreeMap时，key的类型没有实现Comparable接口，则会报ClassCastException异常。

## 3.2containsKey

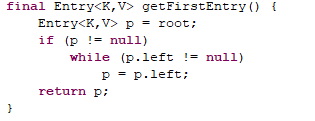


调用了getEntry获取元素。

## 3.3containsValue

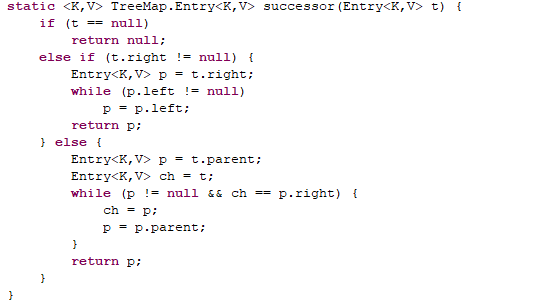


来看一下getFirstEntry这个方法。



获取了最左边的叶子节点。

然后看一下successor这个方法。



这个方法是获取当前节点的中序遍历的后继节点。如果当前节点有右子树，那么它的后继节点是右子树的最左边的节点；如果当前节点没有右子树，（1）当前节点是父节点的左子树，那么后继节点就是它的父节点，（2）当前节点是父节点的右子树，把父节点作为新的当前节点。直到找到当前节点作为左子树的父节点为止。