<https://mp.weixin.qq.com/s?__biz=MzIyNDU2ODA4OQ==&mid=2247483909&idx=1&sn=65fd18c4e83218ac70d18bf0f02d5b13&chksm=e80db473df7a3d6541c8c8330c1896ca26c9b99ab976b8171bb8defd2b16574fe3e1d0a1bbe7&scene=21#wechat_redirect>

# 如何决定使用 HashMap 还是 TreeMap？

**问：如何决定使用 HashMap 还是 TreeMap？**

### 介绍

TreeMap<K,V>的Key值是要求实现java.lang.Comparable，所以迭代的时候TreeMap默认是按照Key值升序排序的；TreeMap的实现是基于红黑树结构。适用于按自然顺序或自定义顺序遍历键（key）。

HashMap<K,V>的Key值实现散列hashCode()，分布是散列的、均匀的，不支持排序；数据结构主要是桶(数组)，链表或红黑树。适用于在Map中插入、删除和定位元素。

### 结论

如果你需要得到一个有序的结果时就应该使用TreeMap（因为HashMap中元素的排列顺序是不固定的）。除此之外，由于HashMap有更好的性能，所以大多不需要排序的时候我们会使用HashMap。

### 拓展

**1、HashMap 和 TreeMap 的实现**

**HashMap：**基于哈希表实现。使用HashMap要求添加的键类明确定义了hashCode()和equals()[可以重写hashCode()和equals()]，为了优化HashMap空间的使用，您可以调优初始容量和负载因子。

* HashMap(): 构建一个空的哈希映像
* HashMap(Map m): 构建一个哈希映像，并且添加映像m的所有映射
* HashMap(int initialCapacity): 构建一个拥有特定容量的空的哈希映像
* HashMap(int initialCapacity, float loadFactor): 构建一个拥有特定容量和加载因子的空的哈希映像

**TreeMap：**基于红黑树实现。TreeMap没有调优选项，因为该树总处于平衡状态。

* TreeMap()：构建一个空的映像树
* TreeMap(Map m): 构建一个映像树，并且添加映像m中所有元素
* TreeMap(Comparator c): 构建一个映像树，并且使用特定的比较器对关键字进行排序
* TreeMap(SortedMap s): 构建一个映像树，添加映像树s中所有映射，并且使用与有序映像s相同的比较器排序

**2、HashMap 和 TreeMap 都是非线程安全**

HashMap继承AbstractMap抽象类，TreeMap继承自SortedMap接口。

**AbstractMap抽象类：**覆盖了equals()和hashCode()方法以确保两个相等映射返回相同的哈希码。如果两个映射大小相等、包含同样的键且每个键在这两个映射中对应的值都相同，则这两个映射相等。映射的哈希码是映射元素哈希码的总和，其中每个元素是Map.Entry接口的一个实现。因此，不论映射内部顺序如何，两个相等映射会报告相同的哈希码。

**SortedMap接口：**它用来保持键的有序顺序。SortedMap接口为映像的视图(子集)，包括两个端点提供了访问方法。除了排序是作用于映射的键以外，处理SortedMap和处理SortedSet一样。添加到SortedMap实现类的元素必须实现Comparable接口，否则您必须给它的构造函数提供一个Comparator接口的实现。TreeMap类是它的唯一一个实现。

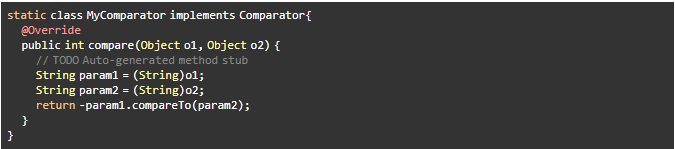
**3、TreeMap中默认是按照升序进行排序的，如何让他降序**

通过自定义的比较器来实现

定义一个比较器类，实现Comparator接口，重写compare方法，有两个参数，这两个参数通过调用compareTo进行比较，而compareTo默认规则是：

* 如果参数字符串等于此字符串，则返回 0 值；
* 如果此字符串小于字符串参数，则返回一个小于 0 的值；
* 如果此字符串大于字符串参数，则返回一个大于 0 的值。

自定义比较器时，在返回时多添加了个负号，就将比较的结果以相反的形式返回，代码如下：



之后，通过MyComparator类初始化一个比较器实例，将其作为参数传进TreeMap的构造方法中：



这样，我们就可以使用自定义的比较器实现降序了

