

目录	
第1章 基础	
01 开篇词：为什么学习本专栏	
02 String、Long 源码解析和面试题	
03 Java 常用关键字理解	
04 Arrays、Collections、Objects 常用方法源码解析	
第2章 集合	
05 ArrayList 源码解析和设计思路	
06 LinkedList 源码解析	
07 List 源码会问哪些面试题	
08 HashMap 源码解析	
09 TreeMap 和 LinkedHashMap 核心源码解析	
10 Map源码会问哪些面试题	
11 HashSet、TreeSet 源码解析	最近阅读
12 彰显细节：看集合源码对我们实际工作的帮助和应用	
13 差异对比：集合在 Java 7 和 8 有何不同和改进	
14 简化工作：Guava Lists Maps 实际工作运用和源码	
第3章 并发集合类	
15 CopyOnWriteArrayList 源码解析和设计思路	
16 ConcurrentHashMap 源码解析和设计思路	
17 并发 List、Map源码面试题	
18 场景集合：并发 List、Map的应用	

11 HashSet、TreeSet 源码解析

更新时间：2019-09-16 19:37:35



“成功的奥秘在于目标的坚定。”
——迪斯雷利

果断更，请联系QQ/微信64260066

HashSet、TreeSet 两个类是在 Map 的基础上组装起来的类，我们学习的侧重点，主要在于 Set 是如何利用 Map 现有的功能，来达成自己的目标的，也就是说如何基于现有的功能进行创新，然后再看看一些改变的小细节是否值得我们学习。

1 HashSet

1.1 类注释

看源码先看类注释上，我们可以得到的信息有：

1. 底层实现基于 HashMap，所以迭代时不能保证按照插入顺序，或者其它顺序进行迭代；
2. add、remove、contains、size 等方法的耗时性能，是不会随着数据量的增加而增加的，这个主要跟 HashMap 底层的数组数据结构有关，不管数据量多大，不考虑 hash 冲突的情况下，时间复杂度都是 O (1)；
3. 线程不安全的，如果需要安全请自行加锁，或者使用 Collections.synchronizedSet；
4. 迭代过程中，如果数据结构被改变，会快速失败的，会抛出 ConcurrentModificationException 异常。

我们之前也看过 List、Map 的类注释，我们发现 2、3、4 点信息在类注释中都有提到，所以如果有人问 List、Map、Set 三者的共同点，那么就可以说 2、3、4 三点。

1.2 HashSet 是如何组合 HashMap 的

目录

- 继承基础类，覆写基础类的方法，比如说继承 HashMap，覆写其 add 的方法；
- 组合基础类，通过调用基础类的方法，来复用基础类的能力。

HashSet 使用的就是组合 HashMap，其优点如下：

1. 继承表示父子类是同一个事物，而 Set 和 Map 本来就是想表达两种事物，所以继承不妥，而且 Java 语法限制，子类只能继承一个父类，后续难以扩展。
2. 组合更加灵活，可以任意的组合现有的基础类，并且可以在基础类方法的基础上进行扩展、编排等，而且方法命名可以任意命名，无需和基础类的方法名称保持一致。

我们在工作中，如果碰到类似问题，我们的原则也是尽量多用组合，少用继承。

组合就是把 HashMap 当作自己的一个局部变量，以下是 HashSet 的组合实现：

```
// 把 HashMap 组合进来，key 是 Hashset 的 key，value 是下面的 PRESENT
private transient HashMap<E,Object> map;
// HashMap 中的 value
private static final Object PRESENT = new Object();
```

从这两行代码中，我们可以看出两点：

1. 我们在使用 HashSet 时，比如 add 方法，只有一个入参，但组合的 Map 的 add 方法却有 key、value 两个入参，相对应上 Map 的 key 就是我们 add 的入参，value 就是第二行代码中的 PRESENT，此处设计非常巧妙，用一个默认值 PRESENT 来代替 Map 的 Value；
2. 如果 HashSet 是被共享的，当多个线程访问的时候，就会有线程安全问题，因为在后续的所有操作中，并没有加锁。

HashSet 在以 HashMap 为基础进行实现的时候，首先选择组合的方式，接着使用默认值来代替了 Map 中的 Value 值，设计得非常巧妙，给使用者的体验很好，使用起来简单方便，我们在工作中也可以借鉴这种思想，可以把底层复杂实现包装一下，一些默认实现可以自己吃掉，使吐出去的接口尽量简单好用。

1.2.1 初始化

HashSet 的初始化比较简单，直接 new HashMap 即可，比较有意思的是，当有原始集合数据进行初始化的情况下，会对 HashMap 的初始容量进行计算，源码如下：

```
// 对 HashMap 的容量进行了计算
public HashSet(Collection<? extends E> c) {
    map = new HashMap<>(Math.max((int) (c.size()/.75f) + 1, 16));
    addAll(c);
}
```

上述代码中：Math.max ((int) (c.size ()/.75f) + 1, 16)，就是对 HashMap 的容量进行了计算，翻译成中文就是 取括号中两个数的最大值（期望的值 / 0.75+1，默认值 16），从计算中，我们可以看出 HashSet 的实现者对 HashMap 的底层实现是非常清楚的，主要体现在两个方面：

1. 和 16 比较大小的意思是说，如果给定 HashMap 初始容量小于 16，就按照 HashMap 默认的 16 初始化好了，如果大于 16，就按照给定值初始化。

目录

还大 1，就不会扩容，符合 HashMap 扩容的公式。

从简单的构造器中，我们就可以看出要很好的组合 api 接口，并没有那么简单，我们可能需要去了解一下被组合的 api 底层的实现，这样才能用好 api。

同时这种写法，也提供了一种思路给我们，如果有人问你，往 HashMap 拷贝大集合时，如何给 HashMap 初始化大小时，完全可以借鉴这种写法：取最大值（期望的值 / 0.75 + 1，默认值 16）。

至于 HashSet 的其他方法就比较简单了，就是对 Map 的 api 进行了一些包装，如下的 add 方法实现：

```
public boolean add(E e) {
    // 直接使用 HashMap 的 put 方法，进行一些简单的逻辑判断
    return map.put(e, PRESENT)!=null;
}
```

从 add 方法中，我们就可以看到组合的好处，方法的入参、名称、返回值都可以自定义，如果是继承的话就不行了。

1.2.2 小结

HashSet 具体实现值得我们借鉴的地方主要有如下地方，我们平时写代码的时候，完全可以参

- 果断更，请联系QQ/微信6426006
- 考参考
1. 对组合还是继承的分析和把握；
 2. 对复杂逻辑进行一些包装，使吐出去的接口尽量简单好用；
 3. 组合其他 api 时，尽量多对组合的 api 多些了解，这样才能更好的使用 api；
 4. HashMap 初始化大小值的模版公式：取括号内两者的最大值（期望的值 / 0.75+1，默认值 16）。

2 TreeSet

TreeSet 大致的结构和 HashSet 相似，底层组合的是 TreeMap，所以继承了 TreeMap key 能够排序的功能，迭代的时候，也可以按照 key 的排序顺序进行迭代，我们主要来看复用 TreeMap 时，复用的两种思路：

2.1 复用 TreeMap 的思路一

场景一：TreeSet 的 add 方法，我们来看下其源码：

```
public boolean add(E e) {
    return m.put(e, PRESENT)!=null;
}
```

可以看到，底层直接使用的是 HashMap 的 put 的能力，直接拿来用就好了。

2.2 复用 TreeMap 的思路二

场景二：需要迭代 TreeSet 中的元素，那应该也是像 add 那样，直接使用 HashMap 已有的迭代能力，比如像下面这样：

目录	// 直接使用 HashMap.keySet 的迭代能力 return m.keySet().iterator(); }
----	--

这种是思路一的实现方式，TreeSet 组合 TreeMap，直接选择 TreeMap 的底层能力进行包装，但 TreeSet 实际执行的思路却完全相反，我们看源码：

```
// NavigableSet 接口，定义了迭代的一些规范，和一些取值的特殊方法
// TreeSet 实现了该方法，也就是说 TreeSet 本身已经定义了迭代的规范
public interface NavigableSet<E> extends SortedSet<E> {
    Iterator<E> iterator();
    E lower(E e);
}

// m.navigableKeySet() 是 TreeMap 写了一个子类实现了 NavigableSet
// 接口，实现了 TreeSet 定义的迭代规范
public Iterator<E> iterator() {
    return m.navigableKeySet().iterator();
}
```

TreeMap 中对 NavigableSet 接口的实现源码截图如下：



从截图中（截图是在 TreeMap 中），我们可以看出 TreeMap 实现了 TreeSet 定义的各种特殊方法。

我们可以看到，这种思路是 TreeSet 定义了接口的规范，TreeMap 负责去实现，实现思路和思路一是相反的。

我们总结下 TreeSet 组合 TreeMap 实现的两种思路：

- 1. TreeSet 直接使用 TreeMap 的某些功能，自己包装成新的 api。
- 2. TreeSet 定义自己想要的 api，自己定义接口规范，让 TreeMap 去实现。

目录	
----	--

义出来后，让 TreeMap 去实现内部逻辑，TreeSet 负责接口定义，TreeMap 负责具体实现，这样的话因为接口是 TreeSet 定义的，所以实现一定是 TreeSet 最想要的，TreeSet 甚至都不用包装，可以直接把返回值吐出去都行。

我们思考下这两种复用思路的原因：

1. 像 add 这些简单的方法，我们直接使用思路 1，主要是 add 这些方法实现比较简单，没有复杂逻辑，所以 TreeSet 自己实现起来比较简单；
2. 思路 2 主要适用于复杂场景，比如说迭代场景，TreeSet 的场景复杂，比如要能从头开始迭代，比如要能取第一个值，比如要能取最后一个值，再加上 TreeMap 底层结构比较复杂，TreeSet 可能并不清楚 TreeMap 底层的复杂逻辑，这时候让 TreeSet 来实现如此复杂的场景逻辑，TreeSet 就搞不定了，不如接口让 TreeSet 来定义，让 TreeMap 去负责实现，TreeMap 对底层的复杂结构非常清楚，实现起来既准确又简单。

2.3 小结

TreeSet 对 TreeMap 的两种不同复用思路，很重要，在工作中经常会遇到，特别是思路二，比如说 dubbo 的泛化调用，DDD 中的依赖倒置等等，原理都是 TreeSet 第二种的复用思想。

3 面试题

HashSet 和 TreeSet 的面试概率比不上 List 和 Map，但只要有机会，并把本文的内容表达出来，绝对是加分项，因为现在 List 和 Map 面试题太多，面试官认为你能答的出来是应该的，但只要你有机会对 HashSet 和 TreeSet 说出本文见解，并且说自己是看源码时领悟到的，绝对肯定是加分项，这些就是你超过面试官预期的惊喜。以下是一些常用的题目：

3.1 TreeSet 有用过么，平时都在什么场景下使用？

答：有木有用过如实回答就好了，我们一般都是在需要把元素进行排序的时候使用 TreeSet，使用时需要我们注意元素最好实现 Comparable 接口，这样方便底层的 TreeMap 根据 key 进行排序。

3.2 追问，如果我想实现根据 key 的新增顺序进行遍历怎么办？

答：要按照 key 的新增顺序进行遍历，首先想到的应该就是 LinkedHashMap，而 LinkedHashSet 正好是基于 LinkedHashMap 实现的，所以我们可以选择使用 LinkedHashSet。

3.3 追问，如果我想对 key 进行去重，有什么好的办法么？

答：我们首先想到的是 TreeSet，TreeSet 底层使用的是 TreeMap，TreeMap 在 put 的时候，如果发现 key 是相同的，会把 value 值进行覆盖，所有不会产生重复的 key，利用这一特性，使用 TreeSet 正好可以去重。

3.4 说说 TreeSet 和 HashSet 两个 Set 的内部实现结构和原理？

答：HashSet 底层对 HashMap 的能力进行封装，比如说 add 方法，是直接使用 HashMap 的 put 方法，比较简单，但在初始化的时候，我看源码有一些感悟：说一下 HashSet 小结的四小点。

目录

总结

本小节主要说了 Set 源码中两处亮点：

1. HashSet 对组合的 HashMap 类扩容的门阈值的深入了解和设计，值得我们借鉴；
2. TreeSet 对 TreeMap 两种复用思路，值得我们学习，特别是第二种复用思路。

HashSet 和 TreeSet 不会是面试的重点，但通过以上两点，可以让我们给面试官一种精益求精的感觉，成为加分项。

精选留言 10

欢迎在这里发表留言，作者筛选后可公开显示

所相虚妄

这个不是TreeSet负责来定义吧，不是NavigableSet来负责定义的么。

0 回复

2019-12-07

weixin_慕工程5089940

老师我想问一下，关于HashSet的add方法，如果我add一个本身map中存在的键，但是值是null的，根据map的put返回规则，会返回null，根据add方法的逻辑，set会判断添加成功，但实际上map中只是修改了一个键的值，会有这样的情况吗？还是我的理解有问题？

0 回复

2019-10-28

文贺 回复 weixin_慕工程5089940

HashSet 的 value 是一个固定的值，所以按照你说的场景，Map 里面会找出存在的 key，不会更新 key。

回复

2019-10-31 11:15:55

慕粉1835158847

追问3.3，HashSet不是也能去重吗

0 回复

2019-10-14

文贺 回复 慕粉1835158847

是的，也可以哈，没说不可以哈。

回复

2019-10-15 13:30:19

qq_Ezio_1

目录

文贺 回复 qq_Ezio_1

知乎上可以搜索到大厂面试题，比如搜索阿里面试题，就会出现最新的面试题，另外说一句，刷面试题全靠运气，大厂每年的面试题都和往年不一样，面试的几个关键要素：1. 运气，问到的都是你会的；2. 有所准备，事先准备几个知识点，是自己深入研究过的，这样可以突出亮点；3. 注意知识点之间的关联，大多人学习的知识都是单点，想孤岛一样，如果你能把知识点关联起来，无疑比大多数人更深入了一步。最后，相信自己，我也相信你可以的。

回复 2019-09-18 21:02:31

qq_Ezio_1 回复 文贺

感谢老师的鼓励！！！！

回复 2019-09-19 19:36:28

monkeyzi

问题3.2有误吧？

👍 0 回复 2019-09-16

weixin_慕哥6366500

老师，这什么时候能更新完呀

👍 0 回复 2019-09-14

文贺 回复 weixin_慕哥6366500

近期计划每周2更哈~

回复 2019-09-16 19:36:41

bb111323

追问的3.2 HashSet不是基于HashMap实现的吗？HashMap输出无序的，HashSet输出好像也不是按照添加的顺序啊。

👍 0 回复 2019-09-14

XsYoung

老师，3.2是不是想说的是 LinkedHashMap

👍 0 回复 2019-09-14

文贺 回复 XsYoung

嗯嗯，收到，感谢指正，订正中了。

回复 2019-09-16 19:36:19

shuangyueliao

不对呀，hashmap的数组大小不是永远是2的倍数吗，所以hashmap数组大小的模板公式不是那么适用

👍 0 回复 2019-09-13

果断更，请联系QQ/微信6426006

<div><div>←</div><div>慕课专栏</div></div>	<div><div>⋮</div><div>面试官系统精讲Java源码及大厂真题 / 11 HashSet、TreeSet 源码解析</div></div>
<div>目录</div>	<div><div>hMap 源码中 tableSizeFor 方法。</div><div>回复2019-09-15 09:39:21</div></div>
	<div><div>Elylic</div><div>非常实用!! 面试必备!!</div><div><div>👍 0</div><div>回复</div></div><div>2019-09-12</div></div>
	<div><div>文贺 回复 Elylic</div><div>嗯嗯, 学习! 学习! 学习! , 进步! 进步! 进步!</div><div>回复2019-09-12 19:00:40</div></div>

千学不如一看, 千看不如一练

果断更，请联系QQ/微信6426006。