

现在这篇主要讲Set集合的三个子类：

- HashSet集合
 - A:底层数据结构是哈希表(是一个元素为链表的数组) + 红黑树
- TreeSet集合
 - A:底层数据结构是红黑树(是一个自平衡的二叉树)
 - B:保证元素的排序方式
- LinkedHashSet集合
 - A: 底层数据结构由哈希表(是一个元素为链表的数组)和双向链表组成。

这篇主要来看看它们比较重要的方法是如何实现的，需要注意些什么，最后比较一下哪个时候用哪个～

强调：**在学习本文之前，最好是看过Map系列的文章**

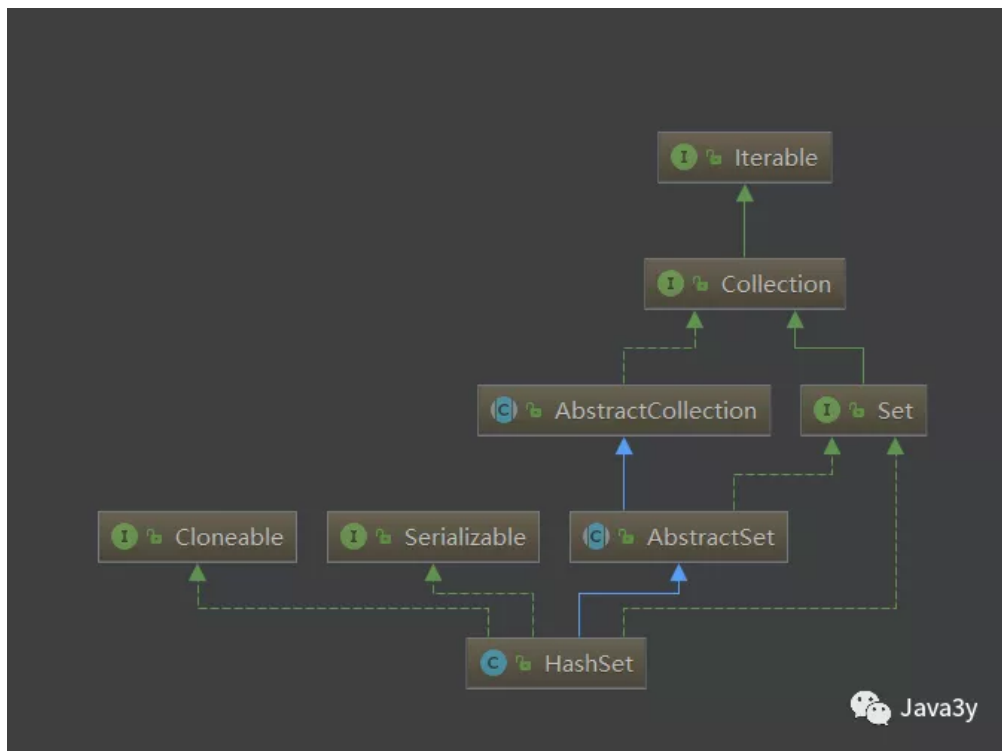
看这篇文章之前最好是有点数据结构的基础：

- [Java实现单向链表](#)
- [栈和队列就是这么简单](#)
- [二叉树就这么简单](#)

当然了，如果讲得有错的地方还请大家多多包涵并不吝在评论去指正～

一、HashSet剖析

首先，我们来看一下HashSet的继承结构图：



按照惯例，我们来看看HashSet顶部注释：

从顶部注释来看，我们就可以归纳HashSet的要点了：

- 实现Set接口
- 不保证迭代顺序

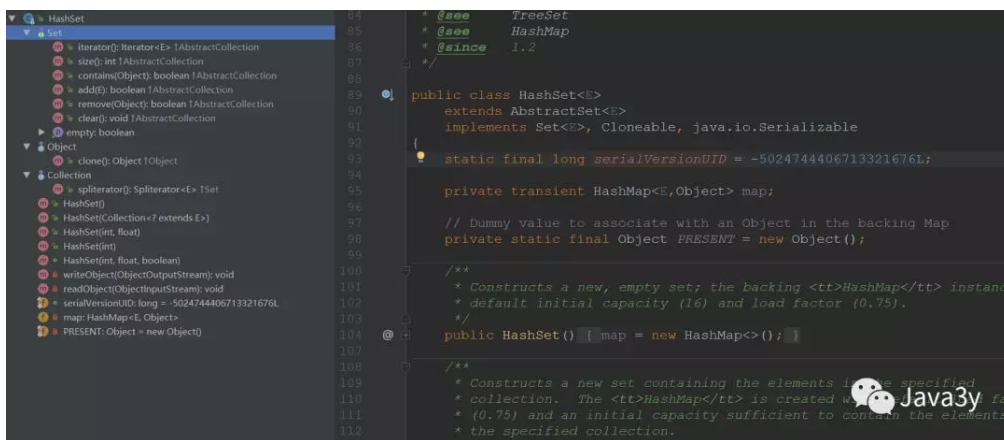
- 允许元素为null
- **底层实际上是一个HashMap实例**
- 非同步
- 初始容量非常影响迭代性能

我本来也是想在写完List集合就转到Set集合的了，可是：看到底层实际上是一个HashMap实例时，我就去学习Map集合先了~

顶部注释说底层实际上是一个HashMap实例，那证据呢？

```
/**
 * Constructs a new, empty set; the backing <tt>HashMap</tt> instance has
 * default initial capacity (16) and load factor (0.75).
 */
public HashSet() { map = new HashMap<>(); }
```

我们再来看一下HashSet整个类的方法和属性：



```
/**
 * Constructs a new, empty set; the backing <tt>HashMap</tt> instance has
 * default initial capacity (16) and load factor (0.75).
 */
public HashSet() { map = new HashMap<>(); }
```

对于学习过HashMap的人来说，简直简单得让人开心，哈哈~

我们知道Map是一个映射，有key有value，**既然HashSet底层用的是HashMap，那么value在哪里呢？？**

```
// Dummy value to associate with an Object in the backing Map
private static final Object PRESENT = new Object();
```

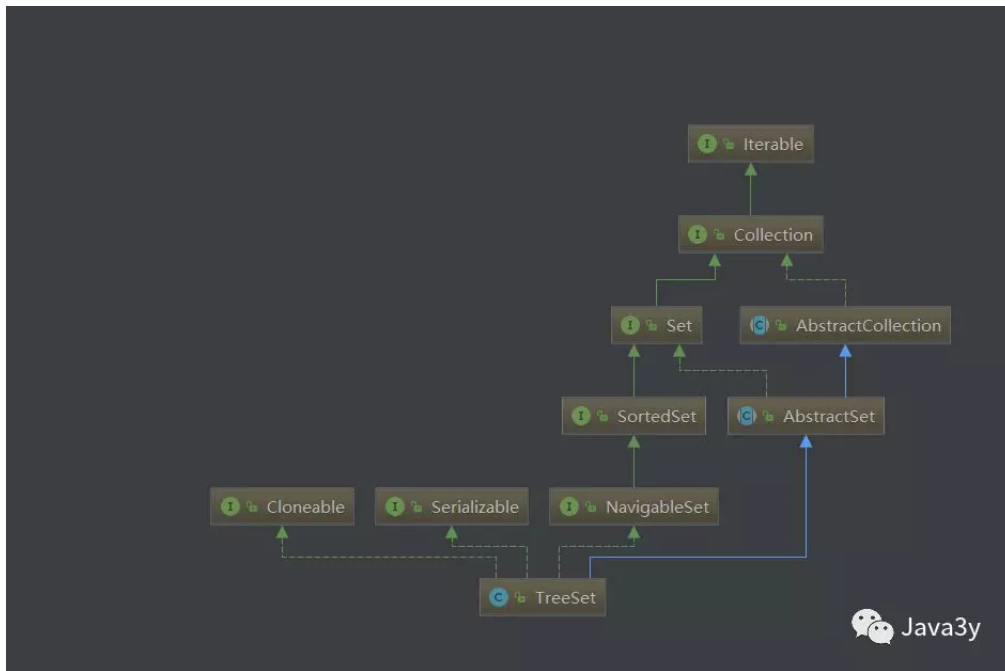
value是一个Object，**所有的value都是它**

所以可以直接总结出：HashSet实际上就是封装了HashMap，**操作HashSet元素实际上就是操作HashMap**。这也是面向对象的一种体现，**重用性贼高！**

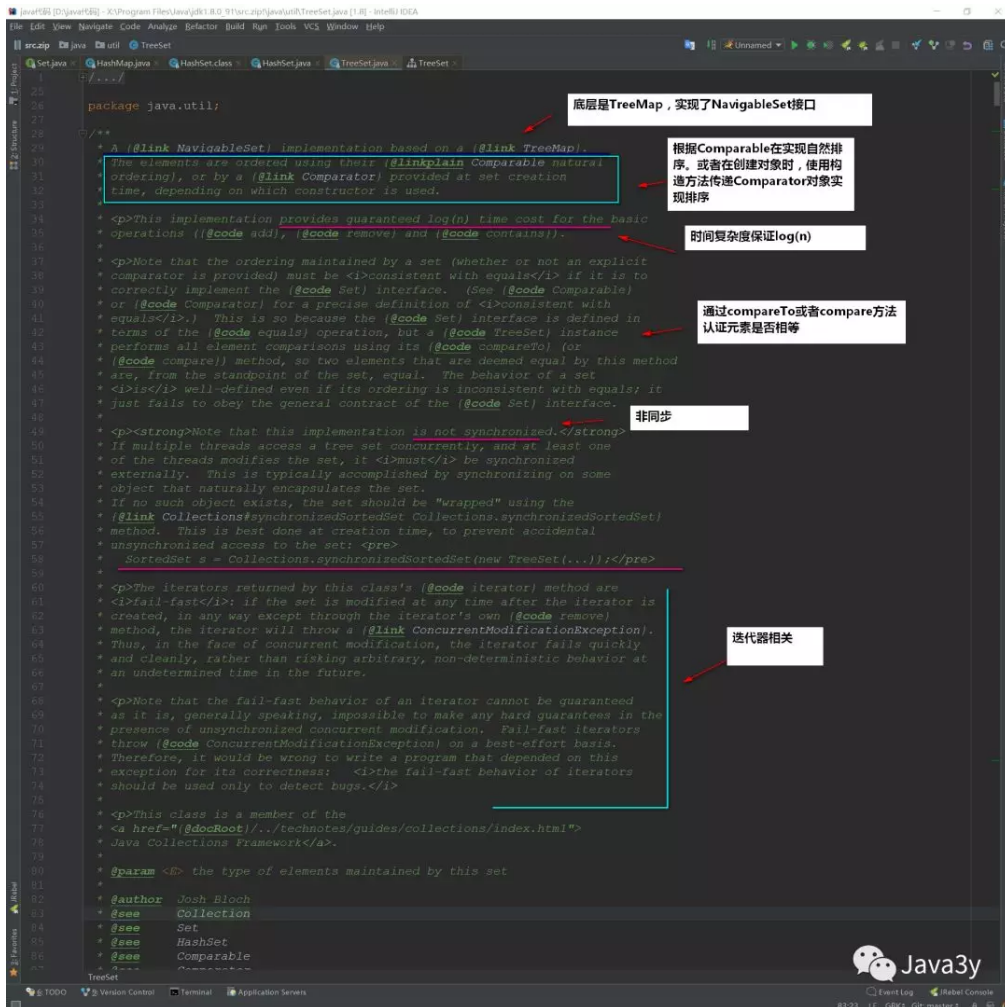
建议：先去阅读[HashMap就是这么简单【源码剖析】](#)

二、TreeSet剖析

首先，我们也来看看TreeSet的类继承结构图：



按照惯例，我们来看看TreeSet顶部注释：



从顶部注释来看，我们就可以归纳TreeSet的要点了：

- 实现NavigableSet接口
- 可以实现排序功能
- 底层实际上是一个TreeMap实例
- 非同步

```
// Dummy value to associate with an Object in the backing Map
private static final Object PRESENT = new Object();

/**
 * Constructs a set backed by the specified navigable map.
 */
TreeSet(NavigableMap<E, Object> m) { this.m = m; }

/**
 * Constructs a new, empty tree set, sorted according to the
 * natural ordering of its elements. All elements inserted into
 * the set must implement the {@link Comparable} interface.
 * Furthermore, all such elements must be <i>mutually
 * comparable</i>: {@code e1.compareTo(e2)} must not throw a
 * {@code ClassCastException} for any elements {@code e1} and
 * {@code e2} in the set. If the user attempts to add an element
 * to the set that violates this constraint (for example, the user
 * attempts to add a string element to a set whose elements are
 * integers), the {@code add} call will throw a
 * {@code ClassCastException}.
 */
public TreeSet() {
    this(new TreeMap<E, Object>());
}
```

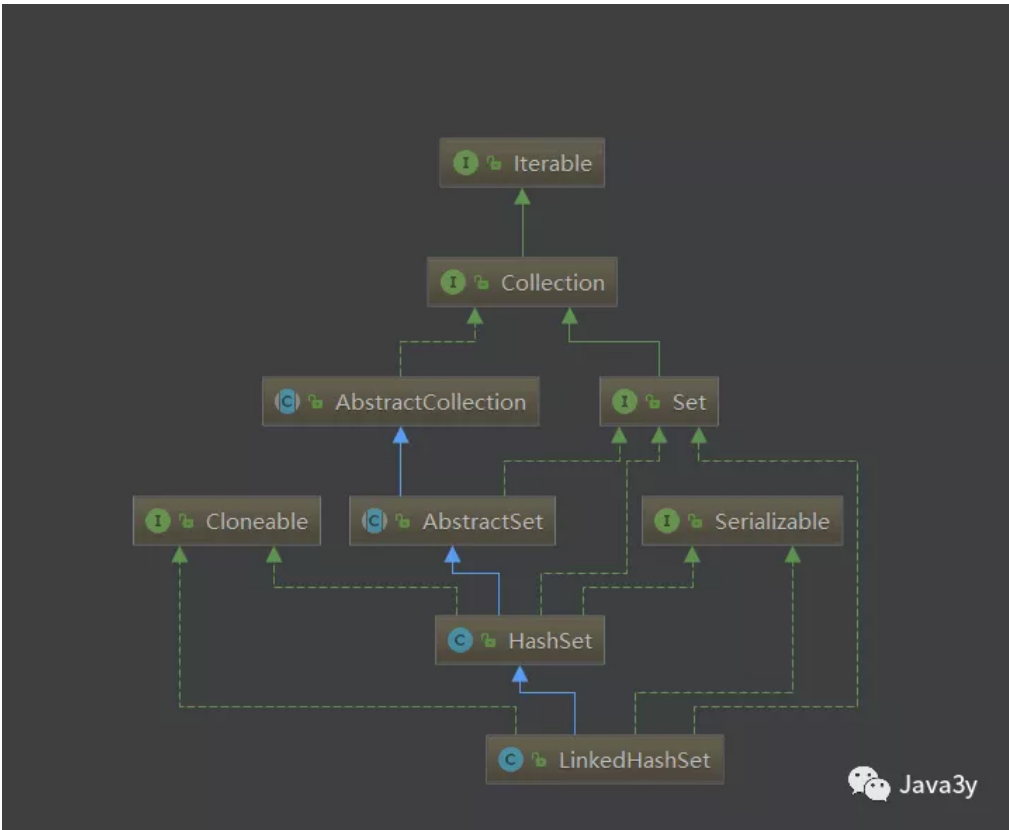
Map的value

实例是TreeMap

Java3y

三、LinkedHashSet剖析

首先，我们也来看看TreeSet的类继承结构图：



按照惯例，我们来看看LinkedHashSet顶部注释：



从顶部注释来看，我们就可以归纳LinkedHashSet的要点了：

- 迭代是有序的
- 允许为null
- 底层实际上是一个HashMap + 双向链表实例(其实就是LinkedHashMap)...
- 非同步
- 性能比HashSet差一丢丢，因为要维护一个双向链表
- 初始容量与迭代无关，LinkedHashSet迭代的是双向链表

四、总结

可以很明显地看到，Set集合的底层就是Map，所以我都没有做太多的分析在上面，也没什么好分析的了。

下面总结一下Set集合常用的三个子类吧：

HashSet:

- 无序，允许为null，底层是HashMap(散列表+红黑树)，非线程同步

TreeSet:

- 有序，不允许为null，底层是TreeMap(红黑树),非线程同步

LinkedHashSet:

- 迭代有序，允许为null，底层是HashMap+双向链表，非线程同步

从结论而言我们就可以根据自己的实际情况来使用了。

参考资料：

- <https://zhuanlan.zhihu.com/p/29021276>
- <https://blog.csdn.net/panweiwei1994/article/details/76555359>

如果文章有错的地方欢迎指正，大家互相交流。习惯在微信看技术文章，想要获取更多的Java资源的同学，可以[关注微信公众号:Java3y](#)。谢谢支持了！希望能多介绍给其他有需要的朋友

文章的目录导航：<https://zhongfucheng.bitcron.com/post/shou-ji/wen-zhang-dao-hang>

目前初步打算写多线程，**你们觉得怎么样呢**？可以在评论区留言~

喜欢此内容的人还喜欢

被面试官乱杀，问了我线程同步工具，太惨了

Java3y

毁掉一个孩子，就是管他，使劲管他

女儿派

这玩意儿不比P站香??!

猿大侠