　　　　　　序言

　　　　　　　　　在写了HashMap文章后，隔了几天才继续这一系列的文章，因为要学的东西实在是太多了，写一篇要花费的时间很多，所以导致隔了几天才来写。不过希望自己坚持下去。终有一天会拨开云雾见青天的。学HashSet的话，要先懂HashMap，所以如果大家如果还不懂HashMap可以先往前看一下我写的HashMap那篇文章。[http://www.cnblogs.com/whgk/p/6091316.html](http://www.cnblogs.com/whgk/p/6091316.html" \o "http://www.cnblogs.com/whgk/p/6091316.html" \t "http://www.cnblogs.com/whgk/p/_blank)

　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　--WH

一、HashSetAPI文档

　　　　　　　　个人感觉看英文的api对自己会有帮助，实在琢磨不透在拿出中文版的来对比一下，看看人家是如何翻译的，这样一来，对自己阅读英文文档有帮助，也让自己对该知识点更加映像深刻，你要去翻译它，就必须先要理解他讲的是什么东西，而直接看中文文档的话，直接一笔带过了。

　　　　　　看了api之后，可以知道三点重要的东西，快速失败这种不算陌生了，

　　　　　　　　1、底层是用HashMap实现

　　　　　　　　2、无序的

　　　　　　　　3、非线程安全的

[IMG_256](http://www.cnblogs.com/whgk/p/javascript:void(0);)

//这一段就告诉了我们hashset底层就是用hashMap来存储的，那我们对HashMap就挺熟悉了，能够存储null值，不是按顺序存储也可以理解，因为HashMap中元素的位置就会变动。  
这里hashSet的第一个特性就是无序，现在就差不多知道了为什么无序了把。

This class implements the Set interface, backed by a hash table (actually a HashMap instance). It makes no guarantees as to the iteration   
order of the set; in particular, it does not guarantee   
that the order will remain constant over time. This class permits the null element.

//讲解一些特点，在add、remove等操作上性能好，查询元素时的效率跟hashset中存放的元素个数还有map的bucket数量成比例，所以不要讲inital capacity设置的太高，也不能讲加载印子设置的太低了

This class offers constant time performance for the basic operations (add, remove, contains and size), assuming the hash function disperses the elements properly among  
 the buckets. Iterating over this set requires time proportional to the sum of the HashSet instance's size (the number of elements) plus the "capacity" of the backing HashMap instance (the number of buckets). Thus, it's very  
 important not to set the initial capacity too high (or the load factor too low) if iteration performance is important.

//非线程安全的，如果多个线程访问他并改变了内部结构，比如add、delete等操作，会发生快速失败，所以解决的方案就是set对象上加同步锁。或者在创建的时候，用Collections.SynchronizedSet这个方法修饰它。还有更下面那些就是解释快速失败是干什么的，等一些东西。

Note that this implementation is not synchronized. If multiple threads access a hash set concurrently, and at least one of the threads modifies the set, it must be   
  
synchronized externally. This is typically accomplished by synchronizing on some object that naturally encapsulates the set. If no such object exists, the set should   
be "wrapped" using the Collections.synchronizedSet method. This is best done at creation time, to prevent accidental unsynchronized access to the set:

Set s = Collections.synchronizedSet(new HashSet(...));The iterators returned by this class's iterator method are fail-fast: if the set is modified at any time after the iterator is created, in any way except through the iterator's own remove method, the Iterator throws a ConcurrentModificationException. Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.

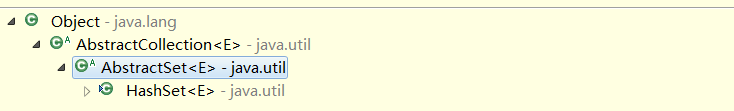
Note that the fail-fast behavior of an iterator cannot be guaranteed as it is, generally speaking, impossible to make any hard guarantees in the presence of unsynchronized concurrent modification. Fail-fast iterators throw ConcurrentModificationException on a best-effort basis. Therefore, it would be wrong to write a program that depended on this exception for its correctness: the fail-fast behavior of iterators should be used only to detect bugs.

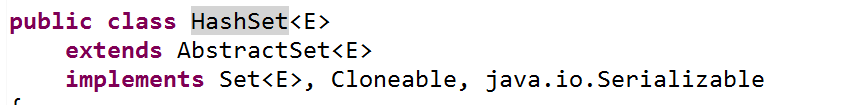
This class is a member of the Java Collections Framework.

Since:

[IMG_257](http://www.cnblogs.com/whgk/p/javascript:void(0);)

二、HashSet继承结构





　　　　　　没有什么特别的东西，很常用的继承结构，一步一步慢慢分解下来，中间用抽象类来缓冲下面实现集合的工作量

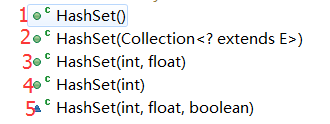
　　　　　　实现的Set接口，这个是多余的东西，可以不实现它，因为在AbstractSet中已经实现了set接口了，继承了AbstractSet，就相当于也实现了Set接口

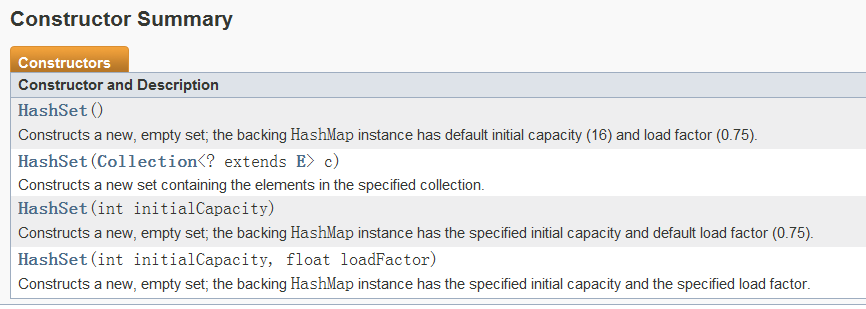
　　　　　　Cloneable：能够使用clone方法，目的是为了在数组扩增大小的时候，能用到此方法

　　　　　　Serializable：能够使之序列化

三、HashSet构造方法

　　　　　　　　　有五个构造方法，其中API只写出四个来，因为最后一个构造方法的访问权限是default。只能在本包内可见，所以API中没有显示出来





　　　　　　　　看看构造方法中度干了些什么

[IMG_262](http://www.cnblogs.com/whgk/p/javascript:void(0);)

/\*\*

\* Constructs a new, empty set; the backing <tt>HashMap</tt> instance has

\* default initial capacity (16) and load factor (0.75).

\*/  
//一个空的构造函数，new一个hashMap对象实例出来。默认初始容量是16，加载因子是0.75，这些都市HashMap的知识

public HashSet() {

map = new HashMap<>();

}//讲一个新的collection转换为一个HashSet集合。这个构造和空构造函数都市规定需要写的。 public HashSet(Collection<? extends E> c) {

map = new HashMap<>(Math.max((int) (c.size()/.75f) + 1, 16));

addAll(c);

}//自己初始化容量和加载因子的大小 public HashSet(int initialCapacity, float loadFactor) {

map = new HashMap<>(initialCapacity, loadFactor);

}//初始化容量大小，加载因子用默认的 public HashSet(int initialCapacity) {

map = new HashMap<>(initialCapacity);

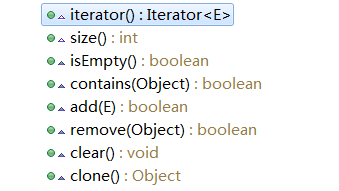
}//这个就是那个default权限的构造方法，只能在包内可见，这个作用是给HashSet的子类 LinkedHashSet用的，LinkedHashSet中并没有什么别的方法，就四个构造方法，并且构造方法也是调用HashSet中的构造方法，也就是  
调用这个构造方法，来使其底层的实现原理为LinkedHashMap，而不是HashMap。所以该方法的作用就在此处。 HashSet(int initialCapacity, float loadFactor, boolean dummy) {

map = new LinkedHashMap<>(initialCapacity, loadFactor);

}

[IMG_263](http://www.cnblogs.com/whgk/p/javascript:void(0);)

四、HashSet常用方法



　　　　　　add(E)

[IMG_265](http://www.cnblogs.com/whgk/p/javascript:void(0);)

//这个方法就可以得知HashSet添加的元素是不能够重复的，原因是什么呢，set将每次添加的元素度是通过map中的key来保存，当有相同的key时，也就是添加了相同的元素，那么map会讲value给覆盖掉，而key还是原来的key  
，所以，这就是set不能够重复的原因。这个方法的PRESENT可以看下面的注释，  
//返回值的式子的意思很好理解，map中增加元素是先通过key查找有没有相同的key值，如果有，则覆盖value，返回oldValue。没有，则创建一个新的entry，并且返回null值。如果key等于null，也会返回null值。  
　　所以return会有一个==null的判断

public boolean add(E e) {

return map.put(e, PRESENT)==null;

}

//PERSENT:通过注释，可以知道这个一个假的值，为了符合map的key，value的方式才用到这个值。

// Dummy value to associate with an Object in the backing Map

private static final Object PRESENT = new Object();

[IMG_266](http://www.cnblogs.com/whgk/p/javascript:void(0);)

　　　　　　remove(Object)

[IMG_267](http://www.cnblogs.com/whgk/p/javascript:void(0);)

//map中通过key来移除对应的元素，如果有该key，会返回其value值，没有，则返回null

public boolean remove(Object o) {

return map.remove(o)==PRESENT;

}

//HashMap中的remove，看一眼就懂了。

public V remove(Object key) {

Entry<K,V> e = removeEntryForKey(key);

return (e == null ? null : e.value);

}

[IMG_268](http://www.cnblogs.com/whgk/p/javascript:void(0);)

　　　　　　　contains(Object)

[IMG_269](http://www.cnblogs.com/whgk/p/javascript:void(0);)

//是否包含某值，也就是判断HashMap中是否有这个Key值。

public boolean contains(Object o) {

return map.containsKey(o);

}//HashMap中的containsKey方法

public boolean containsKey(Object key) {//也就是通过key能不能找得到其entry，找得到就说明有。找不到就没有

return getEntry(key) != null;

}

[IMG_270](http://www.cnblogs.com/whgk/p/javascript:void(0);)

　　　　　　isEmpty()

//非常简单，都市通过调用map的方法

public boolean isEmpty() {

return map.isEmpty();

}

　　　　　size()

//看set中有多少个元素，也就是看map中有多少个元素

public int size() {

return map.size();

}

　　　　iterator()

[IMG_271](http://www.cnblogs.com/whgk/p/javascript:void(0);)

//可以重点看一下这个迭代器，这个迭代器在HashMap中就已经构建好了。

public Iterator<E> iterator() {

return map.keySet().iterator();

}//keySet这个方法也就是为了new一个KeySet这个类出来，来看看KeySet这个类

public Set<K> keySet() {

Set<K> ks = keySet;

return (ks != null ? ks : (keySet = new KeySet()));

}//KeySet类是HashMap中的一个内部类，继承了AbstractSet，所以他也是一个set，其中就是一个写普通的操作，看一下这个newKeyIterator是什么

private final class KeySet extends AbstractSet<K> {

public Iterator<K> iterator() {

return newKeyIterator();

}

public int size() {

return size;

}

public boolean contains(Object o) {

return containsKey(o);

}

public boolean remove(Object o) {

return HashMap.this.removeEntryForKey(o) != null;

}

public void clear() {

HashMap.this.clear();

}

}//newKeyIterator

Iterator<K> newKeyIterator() {

return new KeyIterator();

}//KeyIterator，重点来了，KeyIterator。看名字就知道，就对key进行迭代的迭代器，也就是为set量身打造的，因为set存放的元素就是存放在HashMap中的key，所以为了能够迭代set，HashMap  
就实现了这个专门遍历key的迭代器，通过继承HashIterator，能够每次拿到nextEntry。也就能拿到对应的可以了，可以看下面的图，看下HashIterator有哪些方法就明白了

private final class KeyIterator extends HashIterator<K> {

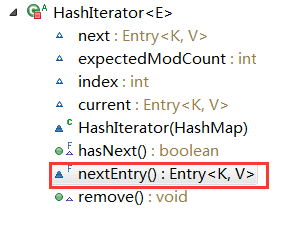
public K next() {

return nextEntry().getKey();

}

}

[IMG_272](http://www.cnblogs.com/whgk/p/javascript:void(0);)



　　　　　　clone()

[IMG_274](http://www.cnblogs.com/whgk/p/javascript:void(0);)

//复制一个HashSet，但是只是复制原HashSet的浅表副本，

public Object clone() {

try {

HashSet<E> newSet = (HashSet<E>) super.clone();

newSet.map = (HashMap<E, Object>) map.clone();

return newSet;

} catch (CloneNotSupportedException e) {

throw new InternalError();

}

}

[IMG_275](http://www.cnblogs.com/whgk/p/javascript:void(0);)

　　　　什么是浅表副本？就是只复制其中元素的值，不复制值所对应的引用，也就是说，一个newHashSet、一个oldHashSet。他们中存放的元素是引用类型的元素，那么浅表副本就是改变newHashSet中引用的值，不会影响大oldHashSet，简单的说，就是两者毫无关系，仅仅是他们元素的值相同，这里要明白的是 值传递和引用传递。

五、总结

　　　　　　到这里，HashSet的源码就已经结束呢，如果你对HashMap很熟悉，那么看HashSet的源码将会毫无阻力，现在来说说HashSet的特点把

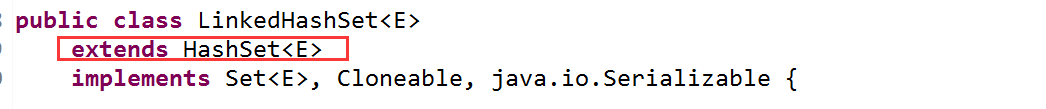
　　　　　　　　1、元素没有顺序(现在知道为什么没有顺序了把，因为底层用的是HashMap，HashMap本身中的元素度没有顺序，那么HashSet更不可能有了)

　　　　　　　　2、元素不能重复(这个也很好理解了，因为HashSet中存放的元素，度是当作HashMap的key来使用，HashMap中key是不能相同的，所以HashSet的元素也不能重复了)

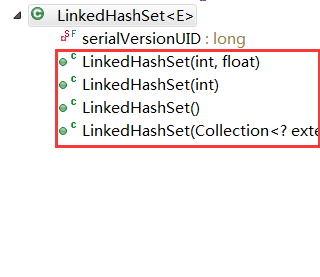
　　　　　　　　3、非线程安全。

　　　　　　linkedHashSet和hashSet的区别

　　　　　　　　　　来看一下linkedHashSet的继承结构



　　　　　　　　　　他是继承自HashSet的，那么就有HashSet的所有功能，在来看一下linkedHashSet其中的方法



　　　　　　　　　　其中只有四个构造方法，每一个构造方法都市调用其父类中，也就是HashSet中的带有三个参数的构造方法，上面我们已经分析过了。其底层实现就是用LinkedHashMap。所以想要知道LinkedHashSet的原理，就要先知道LinkedHashMap的实现原理。这里我就只总结一下其特点，具体的分析，请看LinkedHashMap的源码分析。

　　　　　　　　　　1、LinkedHashSet能按元素插入的顺序进行迭代，也就是说，跟HashSet唯一的区别就是能够保证集合中元素的顺序，注意，是顺序，其位置是变化的，比如：往LinkedHashSet中插入A,B,C,D， 这四个元素其保存的位置可能是变化的，但是在输出的时候能够确定第一个插入的元素是A，A后面是B，说的顺序就是这个意思。 HashSet就不能保证其集合中元素的顺序，应为HashMap每次将元素插入的位置度不确定，并且元素度是放在桶中的，每次遍历度是从第一个桶开始遍历，但是第一个桶中的元素也是不固定的，所以说HashSet不能保证其集合中元素的顺序。