java.util

类 HashMap<K,V>

java.lang.Object

继承者 java.util.AbstractMap<K,V>

继承者 java.util.HashMap<K,V>

类型参数：

K - 此映射所维护的键的类型

V - 所映射值的类型

所有已实现的接口：

Serializable, Cloneable, Map<K,V>

直接已知子类：

LinkedHashMap, PrinterStateReasons

--------------------------------------------------------------------------------

public class HashMap<K,V>extends AbstractMap<K,V>implements Map<K,V>, Cloneable, Serializable基于哈希表的 Map 接口的实现。此实现提供所有可选的映射操作，并允许使用 null 值和 null 键。（除了非同步和允许使用 null 之外，HashMap 类与 Hashtable 大致相同。）此类不保证映射的顺序，特别是它不保证该顺序恒久不变。

此实现假定哈希函数将元素适当地分布在各桶之间，可为基本操作（get 和 put）提供稳定的性能。迭代 collection 视图所需的时间与 HashMap 实例的“容量”（桶的数量）及其大小（键-值映射关系数）成比例。所以，如果迭代性能很重要，则不要将初始容量设置得太高（或将加载因子设置得太低）。

HashMap 的实例有两个参数影响其性能：初始容量 和加载因子。容量 是哈希表中桶的数量，初始容量只是哈希表在创建时的容量。加载因子 是哈希表在其容量自动增加之前可以达到多满的一种尺度。当哈希表中的条目数超出了加载因子与当前容量的乘积时，则要对该哈希表进行 rehash 操作（即重建内部数据结构），从而哈希表将具有大约两倍的桶数。

通常，默认加载因子 (.75) 在时间和空间成本上寻求一种折衷。加载因子过高虽然减少了空间开销，但同时也增加了查询成本（在大多数 HashMap 类的操作中，包括 get 和 put 操作，都反映了这一点）。在设置初始容量时应该考虑到映射中所需的条目数及其加载因子，以便最大限度地减少 rehash 操作次数。如果初始容量大于最大条目数除以加载因子，则不会发生 rehash 操作。

如果很多映射关系要存储在 HashMap 实例中，则相对于按需执行自动的 rehash 操作以增大表的容量来说，使用足够大的初始容量创建它将使得映射关系能更有效地存储。

注意，此实现不是同步的。如果多个线程同时访问一个哈希映射，而其中至少一个线程从结构上修改了该映射，则它必须 保持外部同步。（结构上的修改是指添加或删除一个或多个映射关系的任何操作；仅改变与实例已经包含的键关联的值不是结构上的修改。）这一般通过对自然封装该映射的对象进行同步操作来完成。如果不存在这样的对象，则应该使用 Collections.synchronizedMap 方法来“包装”该映射。最好在创建时完成这一操作，以防止对映射进行意外的非同步访问，如下所示：

Map m = Collections.synchronizedMap(new HashMap(...));由所有此类的“collection 视图方法”所返回的迭代器都是快速失败 的：在迭代器创建之后，如果从结构上对映射进行修改，除非通过迭代器本身的 remove 方法，其他任何时间任何方式的修改，迭代器都将抛出 ConcurrentModificationException。因此，面对并发的修改，迭代器很快就会完全失败，而不冒在将来不确定的时间发生任意不确定行为的风险。

注意，迭代器的快速失败行为不能得到保证，一般来说，存在非同步的并发修改时，不可能作出任何坚决的保证。快速失败迭代器尽最大努力抛出 ConcurrentModificationException。因此，编写依赖于此异常的程序的做法是错误的，正确做法是：迭代器的快速失败行为应该仅用于检测程序错误。

此类是 Java Collections Framework 的成员。

从以下版本开始：

1.2

另请参见：

Object.hashCode(), Collection, Map, TreeMap, Hashtable, 序列化表格

--------------------------------------------------------------------------------

嵌套类摘要

从类 java.util.AbstractMap 继承的嵌套类/接口

AbstractMap.SimpleEntry<K,V>, AbstractMap.SimpleImmutableEntry<K,V>

构造方法摘要

HashMap()

构造一个具有默认初始容量 (16) 和默认加载因子 (0.75) 的空 HashMap。

HashMap(int initialCapacity)

构造一个带指定初始容量和默认加载因子 (0.75) 的空 HashMap。

HashMap(int initialCapacity, float loadFactor)

构造一个带指定初始容量和加载因子的空 HashMap。

HashMap(Map<? extends K,? extends V> m)

构造一个映射关系与指定 Map 相同的新 HashMap。

方法摘要

void clear()

从此映射中移除所有映射关系。

Object clone()

返回此 HashMap 实例的浅表副本：并不复制键和值本身。

boolean containsKey(Object key)

如果此映射包含对于指定键的映射关系，则返回 true。

boolean containsValue(Object value)

如果此映射将一个或多个键映射到指定值，则返回 true。

Set<Map.Entry<K,V>> entrySet()

返回此映射所包含的映射关系的 Set 视图。

V get(Object key)

返回指定键所映射的值；如果对于该键来说，此映射不包含任何映射关系，则返回 null。

boolean isEmpty()

如果此映射不包含键-值映射关系，则返回 true。

Set<K> keySet()

返回此映射中所包含的键的 Set 视图。

V put(K key, V value)

在此映射中关联指定值与指定键。

void putAll(Map<? extends K,? extends V> m)

将指定映射的所有映射关系复制到此映射中，这些映射关系将替换此映射目前针对指定映射中所有键的所有映射关系。

V remove(Object key)

从此映射中移除指定键的映射关系（如果存在）。

int size()

返回此映射中的键-值映射关系数。

Collection<V> values()

返回此映射所包含的值的 Collection 视图。

从类 java.util.AbstractMap 继承的方法

equals, hashCode, toString

从类 java.lang.Object 继承的方法

finalize, getClass, notify, notifyAll, wait, wait, wait

从接口 java.util.Map 继承的方法

equals, hashCode

java.util

类 TreeMap<K,V>

java.lang.Object

继承者 java.util.AbstractMap<K,V>

继承者 java.util.TreeMap<K,V>

类型参数：

K - 此映射维护的键的类型

V - 映射值的类型

所有已实现的接口：

Serializable, Cloneable, Map<K,V>, NavigableMap<K,V>, SortedMap<K,V>

--------------------------------------------------------------------------------

public class TreeMap<K,V>extends AbstractMap<K,V>implements NavigableMap<K,V>, Cloneable, Serializable基于红黑树（Red-Black tree）的 NavigableMap 实现。该映射根据其键的自然顺序进行排序，或者根据创建映射时提供的 Comparator 进行排序，具体取决于使用的构造方法。

此实现为 containsKey、get、put 和 remove 操作提供受保证的 log(n) 时间开销。这些算法是 Cormen、Leiserson 和 Rivest 的 Introduction to Algorithms 中的算法的改编。

注意，如果要正确实现 Map 接口，则有序映射所保持的顺序（无论是否明确提供了比较器）都必须与 equals 一致。（关于与 equals 一致 的精确定义，请参阅 Comparable 或 Comparator）。这是因为 Map 接口是按照 equals 操作定义的，但有序映射使用它的 compareTo（或 compare）方法对所有键进行比较，因此从有序映射的观点来看，此方法认为相等的两个键就是相等的。即使排序与 equals 不一致，有序映射的行为仍然是 定义良好的，只不过没有遵守 Map 接口的常规协定。

注意，此实现不是同步的。如果多个线程同时访问一个映射，并且其中至少一个线程从结构上修改了该映射，则其必须 外部同步。（结构上的修改是指添加或删除一个或多个映射关系的操作；仅改变与现有键关联的值不是结构上的修改。）这一般是通过对自然封装该映射的对象执行同步操作来完成的。如果不存在这样的对象，则应该使用 Collections.synchronizedSortedMap 方法来“包装”该映射。最好在创建时完成这一操作，以防止对映射进行意外的不同步访问，如下所示：

SortedMap m = Collections.synchronizedSortedMap(new TreeMap(...));collection（由此类所有的“collection 视图方法”返回）的 iterator 方法返回的迭代器都是快速失败 的：在迭代器创建之后，如果从结构上对映射进行修改，除非通过迭代器自身的 remove 方法，否则在其他任何时间以任何方式进行修改都将导致迭代器抛出 ConcurrentModificationException。因此，对于并发的修改，迭代器很快就完全失败，而不会冒着在将来不确定的时间发生不确定行为的风险。

注意，迭代器的快速失败行为无法得到保证，一般来说，当存在不同步的并发修改时，不可能作出任何肯定的保证。快速失败迭代器尽最大努力抛出 ConcurrentModificationException。因此，编写依赖于此异常的程序的做法是错误的，正确做法是：迭代器的快速失败行为应该仅用于检测 bug。

此类及其视图中的方法返回的所有 Map.Entry 对都表示生成它们时的映射关系的快照。它们不 支持 Entry.setValue 方法。（不过要注意的是，使用 put 更改相关映射中的映射关系是有可能的。）

此类是 Java Collections Framework 的成员。

从以下版本开始：

1.2

另请参见：

Map, HashMap, Hashtable, Comparable, Comparator, Collection, 序列化表格

--------------------------------------------------------------------------------

嵌套类摘要

从类 java.util.AbstractMap 继承的嵌套类/接口

AbstractMap.SimpleEntry<K,V>, AbstractMap.SimpleImmutableEntry<K,V>

构造方法摘要

TreeMap()

使用键的自然顺序构造一个新的、空的树映射。

TreeMap(Comparator<? super K> comparator)

构造一个新的、空的树映射，该映射根据给定比较器进行排序。

TreeMap(Map<? extends K,? extends V> m)

构造一个与给定映射具有相同映射关系的新的树映射，该映射根据其键的自然顺序 进行排序。

TreeMap(SortedMap<K,? extends V> m)

构造一个与指定有序映射具有相同映射关系和相同排序顺序的新的树映射。

方法摘要

Map.Entry<K,V> ceilingEntry(K key)

返回一个键-值映射关系，它与大于等于给定键的最小键关联；如果不存在这样的键，则返回 null。

K ceilingKey(K key)

返回大于等于给定键的最小键；如果不存在这样的键，则返回 null。

void clear()

从此映射中移除所有映射关系。

Object clone()

返回此 TreeMap 实例的浅表副本。

Comparator<? super K> comparator()

返回对此映射中的键进行排序的比较器；如果此映射使用键的自然顺序，则返回 null。

boolean containsKey(Object key)

如果此映射包含指定键的映射关系，则返回 true。

boolean containsValue(Object value)

如果此映射为指定值映射一个或多个键，则返回 true。

NavigableSet<K> descendingKeySet()

返回此映射中所包含键的逆序 NavigableSet 视图。

NavigableMap<K,V> descendingMap()

返回此映射中所包含映射关系的逆序视图。

Set<Map.Entry<K,V>> entrySet()

返回此映射中包含的映射关系的 Set 视图。

Map.Entry<K,V> firstEntry()

返回一个与此映射中的最小键关联的键-值映射关系；如果映射为空，则返回 null。

K firstKey()

返回此映射中当前第一个（最低）键。

Map.Entry<K,V> floorEntry(K key)

返回一个键-值映射关系，它与小于等于给定键的最大键关联；如果不存在这样的键，则返回 null。

K floorKey(K key)

返回小于等于给定键的最大键；如果不存在这样的键，则返回 null。

V get(Object key)

返回指定键所映射的值，如果对于该键而言，此映射不包含任何映射关系，则返回 null。

SortedMap<K,V> headMap(K toKey)

返回此映射的部分视图，其键值严格小于 toKey。

NavigableMap<K,V> headMap(K toKey, boolean inclusive)

返回此映射的部分视图，其键小于（或等于，如果 inclusive 为 true）toKey。

Map.Entry<K,V> higherEntry(K key)

返回一个键-值映射关系，它与严格大于给定键的最小键关联；如果不存在这样的键，则返回 null。

K higherKey(K key)

返回严格大于给定键的最小键；如果不存在这样的键，则返回 null。

Set<K> keySet()

返回此映射包含的键的 Set 视图。

Map.Entry<K,V> lastEntry()

返回与此映射中的最大键关联的键-值映射关系；如果映射为空，则返回 null。

K lastKey()

返回映射中当前最后一个（最高）键。

Map.Entry<K,V> lowerEntry(K key)

返回一个键-值映射关系，它与严格小于给定键的最大键关联；如果不存在这样的键，则返回 null。

K lowerKey(K key)

返回严格小于给定键的最大键；如果不存在这样的键，则返回 null。

NavigableSet<K> navigableKeySet()

返回此映射中所包含键的 NavigableSet 视图。

Map.Entry<K,V> pollFirstEntry()

移除并返回与此映射中的最小键关联的键-值映射关系；如果映射为空，则返回 null。

Map.Entry<K,V> pollLastEntry()

移除并返回与此映射中的最大键关联的键-值映射关系；如果映射为空，则返回 null。

V put(K key, V value)

将指定值与此映射中的指定键进行关联。

void putAll(Map<? extends K,? extends V> map)

将指定映射中的所有映射关系复制到此映射中。

V remove(Object key)

如果此 TreeMap 中存在该键的映射关系，则将其删除。

int size()

返回此映射中的键-值映射关系数。

NavigableMap<K,V> subMap(K fromKey, boolean fromInclusive, K toKey, boolean toInclusive)

返回此映射的部分视图，其键的范围从 fromKey 到 toKey。

SortedMap<K,V> subMap(K fromKey, K toKey)

返回此映射的部分视图，其键值的范围从 fromKey（包括）到 toKey（不包括）。

SortedMap<K,V> tailMap(K fromKey)

返回此映射的部分视图，其键大于等于 fromKey。

NavigableMap<K,V> tailMap(K fromKey, boolean inclusive)

返回此映射的部分视图，其键大于（或等于，如果 inclusive 为 true）fromKey。

Collection<V> values()

返回此映射包含的值的 Collection 视图。

从类 java.util.AbstractMap 继承的方法

equals, hashCode, isEmpty, toString

从类 java.lang.Object 继承的方法

finalize, getClass, notify, notifyAll, wait, wait, wait

从接口 java.util.Map 继承的方法

equals, hashCode, isEmpty

java.util.concurrent

类 ConcurrentHashMap<K,V>

java.lang.Object

继承者 java.util.AbstractMap<K,V>

继承者 java.util.concurrent.ConcurrentHashMap<K,V>

类型参数：

K - 此映射维护的键的类型

V - 映射值的类

所有已实现的接口：

Serializable, ConcurrentMap<K,V>, Map<K,V>

--------------------------------------------------------------------------------

public class ConcurrentHashMap<K,V>extends AbstractMap<K,V>implements ConcurrentMap<K,V>, Serializable支持获取的完全并发和更新的所期望可调整并发的哈希表。此类遵守与 Hashtable 相同的功能规范，并且包括对应于 Hashtable 的每个方法的方法版本。不过，尽管所有操作都是线程安全的，但获取操作不 必锁定，并且不 支持以某种防止所有访问的方式锁定整个表。此类可以通过程序完全与 Hashtable 进行互操作，这取决于其线程安全，而与其同步细节无关。

获取操作（包括 get）通常不会受阻塞，因此，可能与更新操作交迭（包括 put 和 remove）。获取会影响最近完成的 更新操作的结果。对于一些聚合操作，比如 putAll 和 clear，并发获取可能只影响某些条目的插入和移除。类似地，在创建迭代器/枚举时或自此之后，Iterators 和 Enumerations 返回在某一时间点上影响哈希表状态的元素。它们不会 抛出 ConcurrentModificationException。不过，迭代器被设计成每次仅由一个线程使用。

这允许通过可选的 concurrencyLevel 构造方法参数（默认值为 16）来引导更新操作之间的并发，该参数用作内部调整大小的一个提示。表是在内部进行分区的，试图允许指示无争用并发更新的数量。因为哈希表中的位置基本上是随意的，所以实际的并发将各不相同。理想情况下，应该选择一个尽可能多地容纳并发修改该表的线程的值。使用一个比所需要的值高很多的值可能会浪费空间和时间，而使用一个显然低很多的值可能导致线程争用。对数量级估计过高或估计过低通常都会带来非常显著的影响。当仅有一个线程将执行修改操作，而其他所有线程都只是执行读取操作时，才认为某个值是合适的。此外，重新调整此类或其他任何种类哈希表的大小都是一个相对较慢的操作，因此，在可能的时候，提供构造方法中期望表大小的估计值是一个好主意。

此类及其视图和迭代器实现了 Map 和 Iterator 接口的所有可选 方法。

此类与 Hashtable 相似，但与 HashMap 不同，它不 允许将 null 用作键或值。

此类是 Java Collections Framework 的成员。

从以下版本开始：

1.5

另请参见：

序列化表格

--------------------------------------------------------------------------------

嵌套类摘要

从类 java.util.AbstractMap 继承的嵌套类/接口

AbstractMap.SimpleEntry<K,V>, AbstractMap.SimpleImmutableEntry<K,V>

从接口 java.util.Map 继承的嵌套类/接口

Map.Entry<K,V>

构造方法摘要

ConcurrentHashMap()

创建一个带有默认初始容量 (16)、加载因子 (0.75) 和 concurrencyLevel (16) 的新的空映射。

ConcurrentHashMap(int initialCapacity)

创建一个带有指定初始容量、默认加载因子 (0.75) 和 concurrencyLevel (16) 的新的空映射。

ConcurrentHashMap(int initialCapacity, float loadFactor)

创建一个带有指定初始容量、加载因子和默认 concurrencyLevel (16) 的新的空映射。

ConcurrentHashMap(int initialCapacity, float loadFactor, int concurrencyLevel)

创建一个带有指定初始容量、加载因子和并发级别的新的空映射。

ConcurrentHashMap(Map<? extends K,? extends V> m)

构造一个与给定映射具有相同映射关系的新映射。

方法摘要

void clear()

从该映射中移除所有映射关系

boolean contains(Object value)

一种遗留方法，测试此表中是否有一些与指定值存在映射关系的键。

boolean containsKey(Object key)

测试指定对象是否为此表中的键。

boolean containsValue(Object value)

如果此映射将一个或多个键映射到指定值，则返回 true。

Enumeration<V> elements()

返回此表中值的枚举。

Set<Map.Entry<K,V>> entrySet()

返回此映射所包含的映射关系的 Set 视图。

V get(Object key)

返回指定键所映射到的值，如果此映射不包含该键的映射关系，则返回 null。

boolean isEmpty()

如果此映射不包含键-值映射关系，则返回 true。

Enumeration<K> keys()

返回此表中键的枚举。

Set<K> keySet()

返回此映射中包含的键的 Set 视图。

V put(K key, V value)

将指定键映射到此表中的指定值。

void putAll(Map<? extends K,? extends V> m)

将指定映射中所有映射关系复制到此映射中。

V putIfAbsent(K key, V value)

如果指定键已经不再与某个值相关联，则将它与给定值关联。

V remove(Object key)

从此映射中移除键（及其相应的值）。

boolean remove(Object key, Object value)

只有目前将键的条目映射到给定值时，才移除该键的条目。

V replace(K key, V value)

只有目前将键的条目映射到某一值时，才替换该键的条目。

boolean replace(K key, V oldValue, V newValue)

只有目前将键的条目映射到给定值时，才替换该键的条目。

int size()

返回此映射中的键-值映射关系数。

Collection<V> values()

返回此映射中包含的值的 Collection 视图。

从类 java.util.AbstractMap 继承的方法

clone, equals, hashCode, toString

从类 java.lang.Object 继承的方法

finalize, getClass, notify, notifyAll, wait, wait, wait

从接口 java.util.Map 继承的方法

equals, hashCode