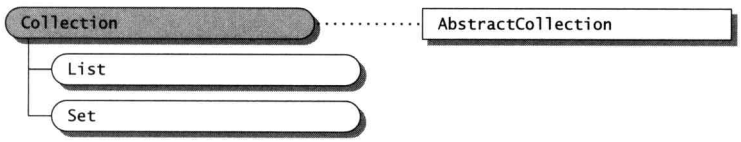
java.util

Collection



语法

public interface Collection

描述

一个收集(collection)是一个对象容器。几种容器用于不同的场合。如果可用的容器集中的容器种类不能满足需要，可以定义自己的定制容器。Collection类代表一个一般的对象容器。

有几种接口和类与收集相关联。它们组成了所谓的收集框架，收集框架是以刻画了收集的一般类型的接口为基础，例如，集合和映射表。框架提供了这些接口的实现，包括哈希 表、链表、平衡树，框架还提供了帮助实现自定义收集的抽象类。最后，框架还提供了各种工具以帮助使用收集，例如，用于排序和翻转链表的工具。

接口

收集框架有六个收集接口。在框架中的每一个实现只实现其中的一个接口，表 5是对六个接口的概述。为了创建一个相容收集，需要实现其中的某个接口。

表5 收集框架

接 口	描 述
List	一个有序的元素表。可以完全了解元素在表的位置。表中的元素通过一个整数索引来访问。一个表中可以有相同的元素
Set	一个可能未被排序的集合。集合中不能有相同的元素
Collection	一个元素收集，它是List和Set的根接口。如果正在创建一个新的收集，但并不是List或Set，那么还是应该实现这个接口，因为它可以利用由收集框架提供的许多服务
Map	一个未排序的集合，其中的元素是关键字 /值对。一个映射表不能包含相同的关键字
SortedSet	一个已排序的集合。同一个集合一样，它不能包含相同的元素
SortedMap	一个已排序的元素为关键字 /值对的集合，通过关键字排序。与一个映射表一样，它不能包含相同的关键字

抽象类

对于在收集框架中的大多数接口都有一个相应的抽象类与之相对应，以帮助实现相对应的接口。例如，为了实现Set应该考虑拥有收集子类AbstractSet，因为它提供了大多数方法的实现。只需提供两个方法 —— 这将使实现Set变得简单许多。

表6中有五个抽象类的简要说明。

表6 抽象类

类	描 述
AbstractList	实现List接口。特别是用于对数组这样的数据结构的实现
AbstractSequentialList	实现List接口。特别是用于对链表这样的数据结构的实现
AbstractSet	实现Set和SortedSet接口
AbstractCollection	实现Collection接口。AbstractList和AbstractSet都是从这个类中扩展出去的
AbstractMap	实现Map和SortedMap接口

当需要实现自定义的收集，必须提供一个无参构造函数；用它构造一个空收集。你还必须提供一个接受Collection对象的构造函数。该构造函数构造一个新的收集，它使用其他收集提供的元素进行初始化，这使它很容易拷贝任何一个收集。

实现

在收集中有七个实现是可用的。它们以ImplementationInterface这样的形式命名。例如，类HashSet是一个集合由一个哈希表实现，TreeSet是一个由平衡树实现的集合。

表7提供了所有实现的简要概述。

表7 实现类

类	描 述
ArrayList	一个用数组实现的列表，用于进行快速随机访问
LinkedList	一个用双向链表实现的列表，用于快速插入和删除
HashSet	一个用哈希表实现的集合
TreeSet	一个用红黑树实现的集合，其中的元素可以自动的进行排序
HashMap	一个关键字保存在哈希 表中映射表
TreeMap	一个关键字存储在红黑树中的映射表，其中的元素可以自动进行排序
WeakHashMap	一个关键字存储在哈希表中的映射表，当一个关键字的所有外部引用被删除，关键字/值对会自动从映射表中删除

除此以外，两个实现通常作为Java编程语言的一部分：Vector和Hashtable。这两个类已经被修改以适应收集框架。Vector类实现了List接口；Hashtable类实现了Map接口。

有序收集

一些收集如TreeSet和TreeMap要对它们中的元素进行排序，所以需要一种方法对元素进行排序。有两种方法可用：

1) 提供一个比较器，它是用于比较两个对象的一个基本方法。排序收集在决定任意两个元素的顺序时使用这个比较器。详见Comparator。

2) 所有的元素都实现接口Comparable。这个接口声明一个单一的方法，该方法允许一个元素与其他元素进行比较。详见Comparable。

迭代器

迭代器被用于以一种顺序的方式访问收集中的元素。例如，为了检查一个集合中的每一个元素，可以在该集合上创建一个迭代器用它对集合中的每一个元素进行访问。通过连续地调用迭代器的next()方法，一定能看到该收集中的每一个元素。

通过迭代器所看到的元素的顺序依赖于收集类型。例如，对于 List返回元素的顺序与它们在列表中的顺序相同。而对于象集合这样的收集来说，返回元素的顺序可以是任意的。

工具类

Collections类包含了许多有用的方法用于对收集进行处理。例如，有一组方法，每一个对应一种收集的类型，它们为对应的收集创建一个固定视图。其他的方法用于对列表进行排序和重组。详见Collections。

收集框架还提供了一个类 —— Arrays，用于对数组进行操作。 Arrays提供排序和用于查找的方法。详见Arrays。

未予支持的方法

一个收集并没有对所有的操作予以支持。例如，一个大小固定的收集就不支持用于添加或删除元素的方法。当一个操作没有得到支持，使用所对应的方法将抛出一个 UnsupportedOperationException异常。

一些实现可能对元素（或者是映射表、关键字和值）的存储加以限制。可能的限制包括要求元素：

- 是一种特殊的类型。
- 在收集中可与其他元素进行比较。
- 非空。
- 遵守一些任意的声明。

如果试图加入一个元素与某个实现的限制相冲突，将会导致一个场时的异常，典型的情况下是这些异常有ClassCastException、IllegalArgumentException或nullPointerException。如果试图删除或测试这样一个元素也可能会导致异常，尽管一些“限制收集”允许这样的使用。

成员概述	
修改方法	
add()	把一个元素加入到该收集中。
addAll()	把某一个收集中的所有元素加入到该收集中。
clear()	把所有的元素从该收集中删除。
remove()	把一个元素从该收集中删除。
removeAll()	把与一个收集中的元素相同的所有元素从该收集中删 除 。
retainAll()	把与一个收集中的元素不同的所有元素从该收集中删除。
迭代器方法	
iterator()	为在该收集中的元素创建一个迭代器。
查询方法	
contains()	用于确定该收集中是否包含某一特定的元素。
containsAll()	用于确定该收集中是否包含另一个收集中的所有元素。
isEmpty()	用于确定该收集是否为空。
size()	返回该收集中元素的个数。
转换方法	
toArray()	把该收集中的所有元素返回到一个数组中。

(续)

成员概述	
对象方法	
equals()	确定该收集与另一个对象是否相等。
hashCode()	为该收集计算哈希码。

参见

ArrayList, Arrays, AbstractCollection, AbstractList, AbstractMap, AbstractSequentialList, AbstractSet, Collection, Collections, Comparator, Comparable, HashSet, HashMap, Iterator, LinkedList, List, ListIterator, Map, Set, SortedSet, SortedMap, TreeSet, TreeMap, WeakHashMap.

add()

目的 把一个元素加入到该收集中。

语法 public boolean add(Object e)

描述 该方法把元素 e加入到收集中。当该调用返回时，收集将包含该元素。对于一些收集，例如集合（见 Set），不允许包含相同的元素。对于这样的收集，只有在该收集中并不包含 e时才将e加入到该收集中。见 contains()方法。一些收集不接受 null。如果 null加入到收集将导致 IllegalArgumentException抛出。一些收集可能只接受一种特定类型的元素。如果一种不匹配的元素加入到收集中，一个ClassCastException将被抛出。
该方法可能未被支持。

参数

e 一个将被加入到该收集中的可能为null的元素。

返回 如果该收集已经修改了，那么将返回 true；如果该收集不允许有重复元素并且已经包含该元素，那么将返回false。

异常

ClassCastException
如果e的类型与该收集不匹配。

IllegalArgumentException
如果e由于某些方面的原因被阻止其加入到收集。

NullPointerException
如果元素e为null并且该收集不接受null元素。

UnsupportedOperationException
如果该收集不支持该方法。

参见 addAll()。

示例 见Set.add()。

addAll()

目的 把在另一收集中的所有元素加入到该收集中。

语法 public boolean addAll(Collection c)

描述 把在收集c中的所有元素加入到该收集中。它同为 c的每一个元素调用 add()方法是等价的。当该方法正在被调用时，该收集和收集 c都不能进行修改；否则，结果可能不确定。
该方法可能不被支持。

参数

c 一个将被加入到该收集中的非空元素。

返回 如果收集可修改将返回true。

异常

ClassCastException
如果c的类型与该收集不匹配。

ConcurrentModificationException
如果c的迭代器是错误快速反应迭代器，并且在该方法调用期间c正在被修改。

IllegalArgumentException
如果c的某些方面阻止其加入到收集。

NullPointerException
如果元素c为null并且该收集不接受null元素。

UnsupportedOperationException
如果该收集不支持该方法。

参见 add()。

示例 见Set.addAll()。

clear()

目的 删除该收集中所有元素。

语法 public void clear()

描述 该方法用于删除该收集中所有元素。如果它被调用，收集将变空。
该方法可能不被支持。

异常

UnsupportedOperationException
如果该收集不支持该方法。

示例 见Set.clear()。

contains()

目的 判断收集中是否包含指定的元素。

语法 public boolean contains(Object e)

描述 该方法用于判断收集中是否包含指定元素 e。如果收集中包含有与 e相等的一些元素则返回true。一个元素与e都等于null或equals()方法返回true时，二者相等。

参数

e 一个可能为null的元素。

返回 如果收集中包含有与e相等的一些元素则返回true。

异常

ClassCastException
如果e的类型与该收集不匹配。
NullPointerException
如果元素e为null并且该收集不接受null元素。
参见 containsAll()、equals()。
示例 见Set.contains()。

containsAll()

目的 用于确定该收集是否包含另一个收集中的所有元素。
语法 public boolean containsAll(Collection c)
描述 该方法用于确定该收集中是否包含另一个收集 c中的所有元素。如果c的所有元素都包含在该收集中，那么该方法返回 true。两个元素当它们都等于null或equals()方法返回true时，二者相等。
参数
c 一个非空的收集。
返回 如果c的所有元素都包含在该收集中，那么该方法返回 true。
异常
NullPointerException
如果c中的一些元素为null并且该收集不接受null元素。
参见 contains()、equals()。
示例 见Set.containsAll()。

equals()

目的 确定该收集与另一个对象是否相等。
语法 public boolean equals(Object c)
描述 该方法用于确定该收集与另一个对象 c是否相等。如果收集与 c相等返回true。判断收集之间是否相等的条件与收集的类型有关。一些收集如，集合，就要求两个收集必须都是集合。另一些收集则有其他的要求；例如，一个列表只与它自己相等。
如果equals(c) 为true，那么c.equals(this)也必须为true 。通常如果equals（c）为true，hashCode()必须等于c.hashCode()。
参数
c 一个可能为null的对象。
返回 如果收集与c相等则返回true。
重载 java.lang.Object.equals()。
参见 hash Code()。
示例 见Set.equals()。

hashCode()

目的 为收集计算哈希码。
语法 public int hashCode()
描述 该方法为收集计算哈希码。该接口并没有定义该收集的哈希码如何产生。

唯一的限制就是如果两个收集相等，那么着两个收集的哈希码也必须相等。在大多数情况下，该收集的哈希码是通过使用该收集的所有元素的哈希码得到的，但这并不是必须的。

返回 返回代表该收集的哈希码的一个整数。

重载 java.lang.Object.hashCode()。

参见 equals()、java.lang.Object.equals()。

示例 见Set.hashCode()。

isEmpty()

目的 判断该收集是否为空。

语法 public boolean isEmpty()

返回 如果收集为空则返回true。

参见 size()。

示例 见Set.isEmpty()。

iterator()

目的 为该收集中的元素创建一个迭代器。

语法 public Iterator iterator()

描述 该方法为该收集的元素创建并返回一个迭代器（详见 Iterator）。被迭代器表示的收集中的元素的顺序是由创建迭代器的类决定的。有些类保证它们是按顺序被迭代的，而有一些则并不保证。

在大多数情况下，在所返回的迭代器正在被使用时，收集是不允许被修改的。否则，一个ConcurrentModificationException被抛出。但是一些收集却允许这样做。详见有关收集的文档。

返回 一个非null的迭代器。

参见 ConcurrentModificationException，Iterator。

示例 见Set.iterator()。

remove()

目的 把一个元素从收集中删除。

语法 public boolean remove(Object e)

描述 该方法用于把元素 e从收集中删除。如果收集中有一个或多个与 e相等的元素（通过使用 equals()方法判定），那么其中的一个元素将被删除并且返回true。否则，该方法将不对收集进行修改，并返回 false。

该方法可能没有得到支持。

参数

e 一个要被从收集中删除的元素，可能为 null。

返回 如果该收集被修改，则返回true。

异常

UnsupportedOperationException

如果该方法没有得到支持。

参见 removeAll()， retainAll()。
示例 见Set.remove()。

removeAll()

目的 把与另一个收集中的元素相同的所有元素从该收集中删除 。
语法 public boolean removeAll(Collection c)
描述 该方法用于把与收集 c中的元素相同的所有元素从该收集中删除。 equals()方法用于判断两个元素是否相等。在该方法返回后，该收集将不会包含与 c相同的元素。该操作可以被认为是一个删除操作，好像 c被从该收集中删除掉了。
该方法可能未被支持。
参数
c 一个非空的收集，其中的元素将被从拥有该元素的另一个收集中删除。
返回 如果收集被修改该方法将返回true。
异常
ConcurrentModificationException
如果c的迭代器是错误快速反应迭代器，并且在该方法调用期间 c正在被修改。
UnsupportedOperationException
如果该方法没有得到支持。
参见 remove()， retainAll()。
示例 Set.removeAll()。

retainAll()

目的 把与另一个收集中的元素不相同的所有元素从该收集中删除。
语法 public boolean retainAll(Collection c)
描述 该方法用于把与收集 c中的元素不相同的所有元素从该收集中删除 。equals()方法用于判断两个元素是否相等。该操作可被认为是一个相交操作。
该方法可能未被支持。
参数
c 一个非空收集，该收集中元素将被保留。
返回 如果收集被修改，该方法将返回true。
异常
ConcurrentModificationException
如果c的迭代器是错误快速反应迭代器，并且在该方法调用期间 c正在被修改。
UnsupportedOperationException
如果该方法没有得到支持。
参见 remove()、removeAll()。
示例 Set.retainAll()。

size()

目的	返回收集中元素的数量。
语法	public int size()
返回	返回收集中元素的个数，一个非负整数。
参见	isEmpty()。
示例	见Set.size()。

toArray()

目的	把收集中的所有元素返回到一个数组中。
语法	public Object[] toArray() public Object[] toArray(Object arr[])
描述	此方法用于把收集中的所有元素返回到数组 arr中并且返回 arr。如果arr不够大，一个新的数组将被创建、填充，并返回。新数组的大小刚好装下收集中的元素，而且，它的类型与数组arr的类型相同。 如果arr足够大，那么可把arr[size()]设为null。这个null值用于确定元素的数量。但是如果收集中包含null元素，这样做就没有用了。返回到数组中的元素的顺序与从在收集上的迭代器返回的顺序是相同的。 如果arr没有被说明，那么缺省为 new Object[0]。
参数	
arr	一个非空的数组，收集中的元素将被拷贝进该数组中。
返回	数组arr或一个新数组它的长度等于size()。
异常	
ArrayStoreException	如果收集中的某个元素不能被分配到 arr中的元素上，该异常将会抛出。但如果arr是一个Object数组，那么该异常将不会被抛出。
示例	见Set.toArray()。