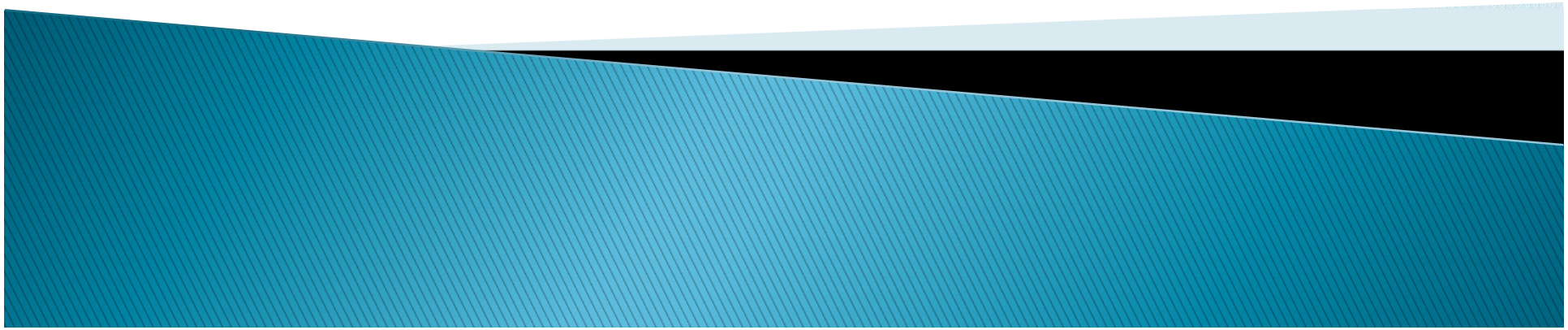
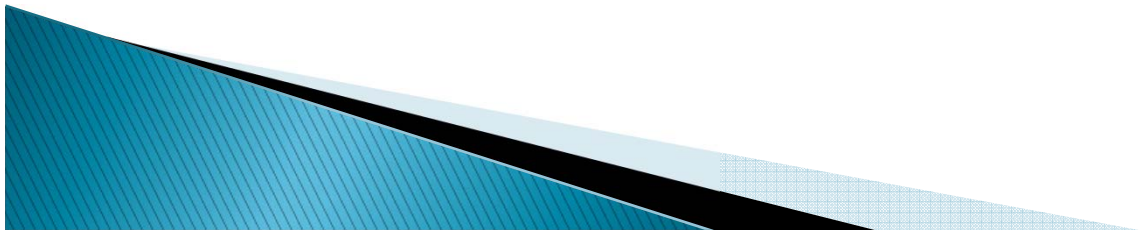
Java集合框架



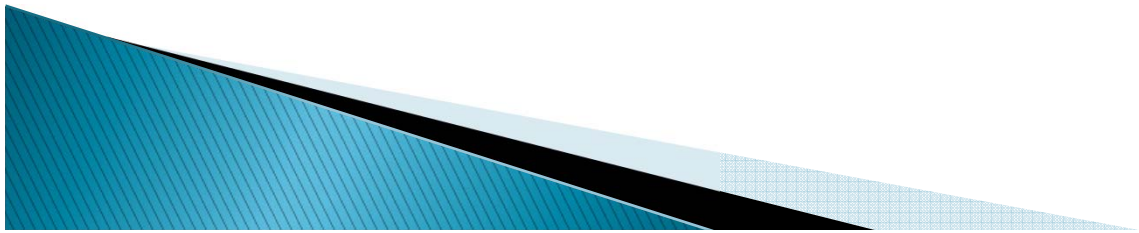
本章教学内容

- Java集合框架概述
- Collection 接口
- 迭代器
- List接口
- Set接口
- Map接口

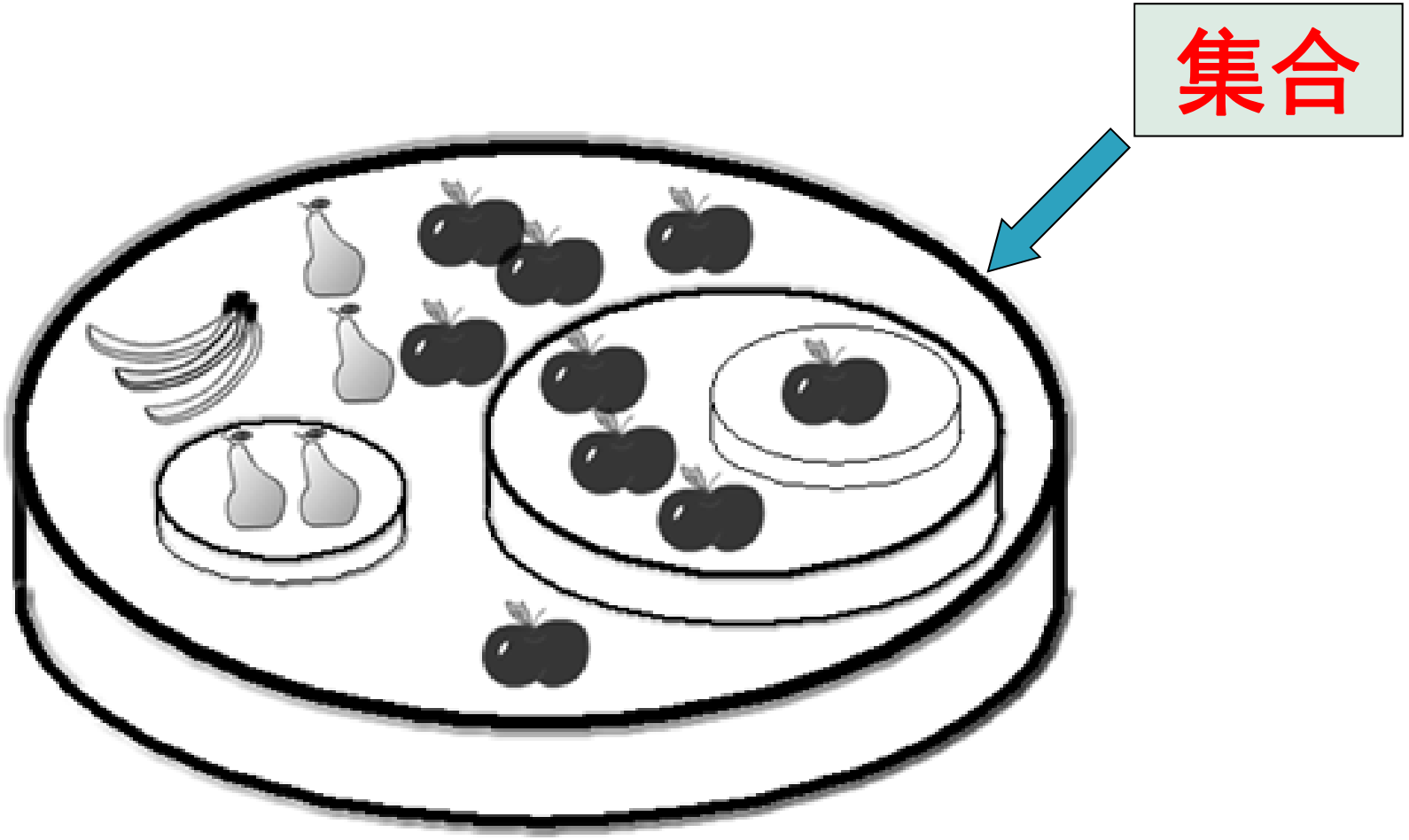


Java集合框架概述

- 在Java语言中可以将多个对象存储在一个集合中，这些集合对象又称为容器对象。
- Java SE设计者将这些集合对象根据其差异抽象出来，所抽象出来的数据结构和操作（算法）统称为Java集合框架（Java Collection Framework）。
- 集合框架，顾名思义，它提供了一系列可以作为集合类型的类，这些类封装了对集合中元素的各种操作。

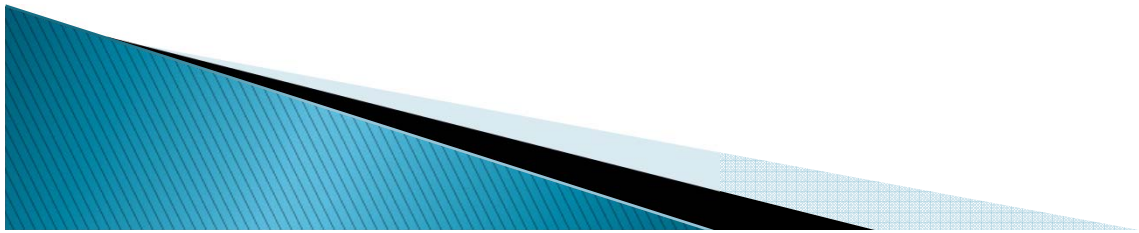


Java集合框架概述



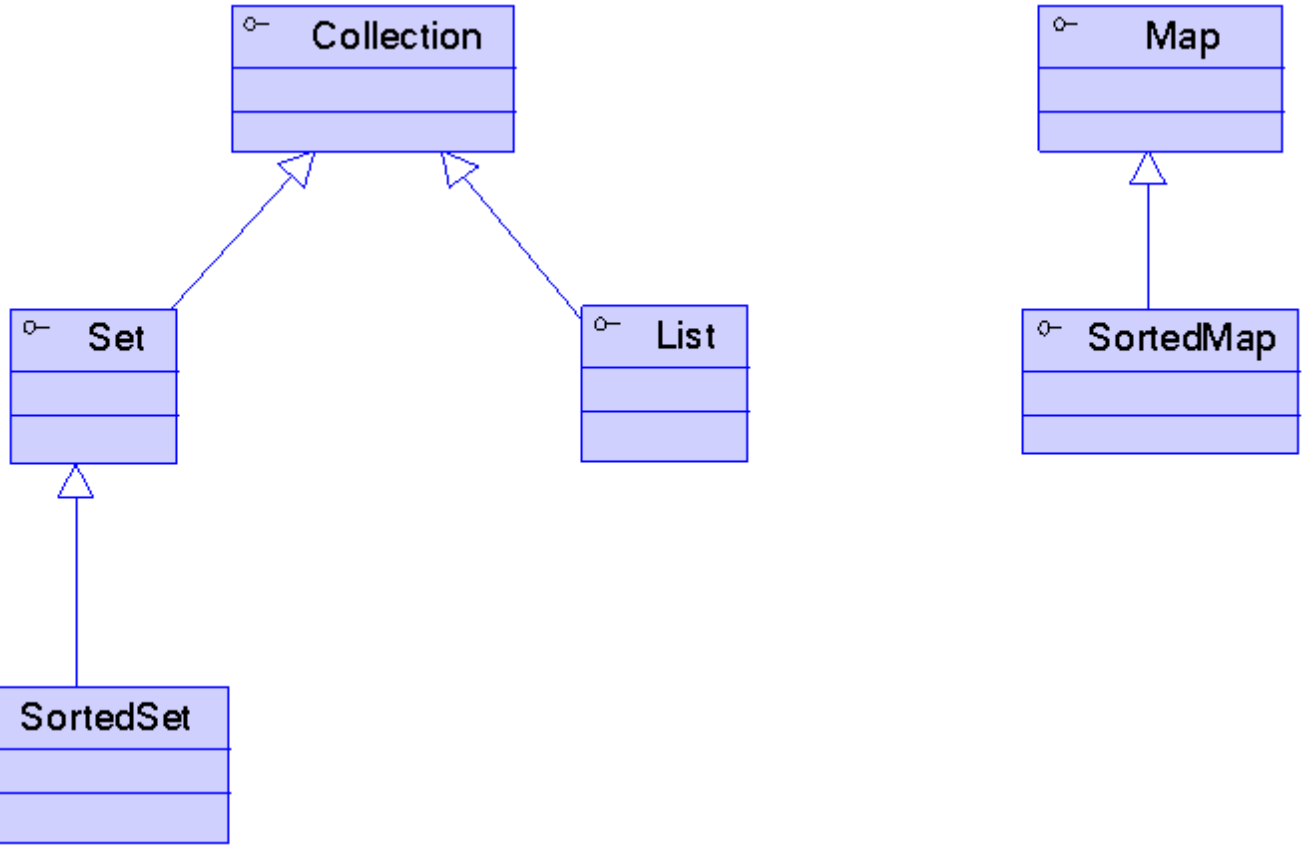
Java集合框架概述

- Java程序员在具体使用集合框架时，不必关心数据结构和算法实现细节，只需要通过这些类创建出来相应的对象，即可直接使用，一方面大大提高了编程效率，另一方面由于这些类被反复使用并加以修正，从而提升了代码的可靠性。



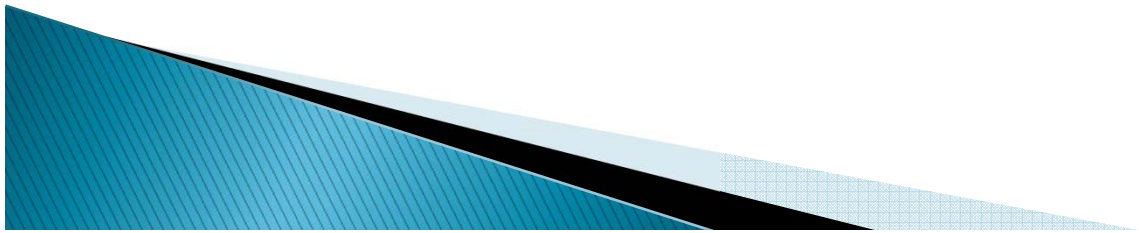
Java集合框架概述

- Java集合框架基本接口层次结构：



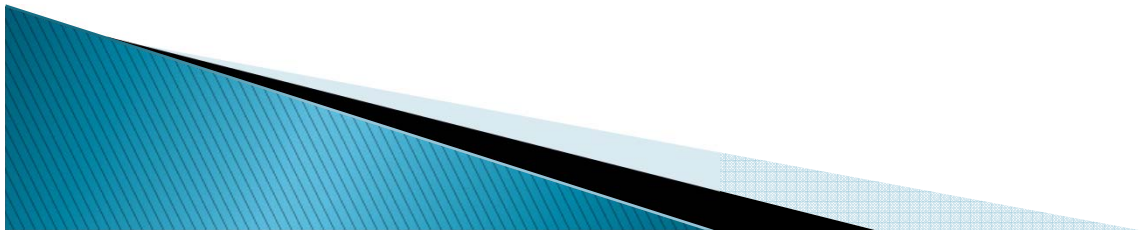
Java集合框架概述

- Java集合框架基本接口
  - Java集合框架的抽象层由一组接口组成，不同的接口用于定义不同类型的集合组。
  - 在Java SE中，所有与Java集合框架相关的接口和类都定义在java.util包中。



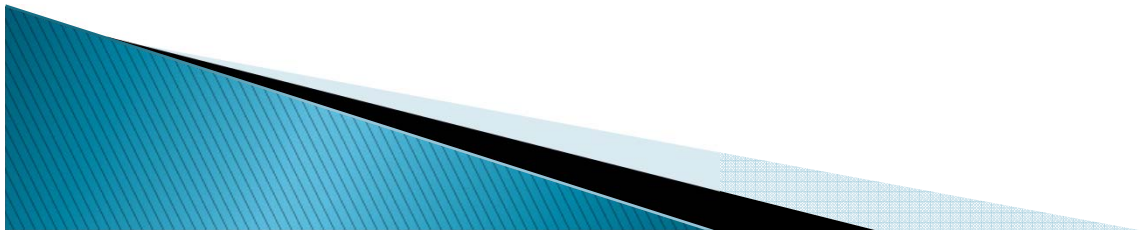
Java集合框架概述

- Java集合框架基本接口
  - Java集合框架中常用的接口如下：
    - **Collection接口**：Collection接口是Collection层次结构的根接口，它包括两个主要的子接口，分别是List接口和Set接口。
    - **List接口**：List接口继承了Collection接口，它是一种有序的Collection，也称为序列或列表。在List接口中，允许有重复的元素出现，而且以元素插入的先后次序来放置元素，不会进行重新排列。
    - **Set接口**：Set接口继承了Collection接口，它与List接口的区别在于不允许出现重复元素，而且它是一种无序的集合，在对Set接口进行遍历时可以发现，元素的排列次序与元素的插入次序无关。



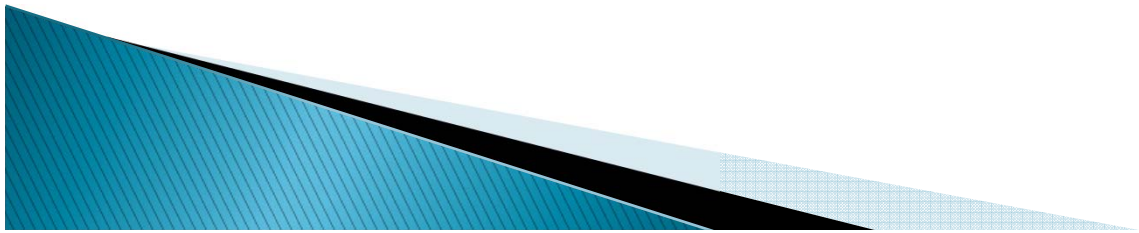
Java集合框架概述

- Java集合框架基本接口
  - Java集合框架中常用的接口如下：
    - **SortedSet接口**：SortedSet接口扩展了Set接口，它是一种有序的Set，其内部元素按照某种特定的排序机制来进行排列。
    - **Map接口**：Map接口是一组成对的键值对象，它将每一个键(Key)映射到一个特定的值(Value)。一个映射不能包含重复的键，每个键最多只能映射到一个值。
    - **SortedMap接口**：SortedMap接口扩展了Map接口，它是一种按照某种特定的排序机制来对键(Key)进行排序的Map。



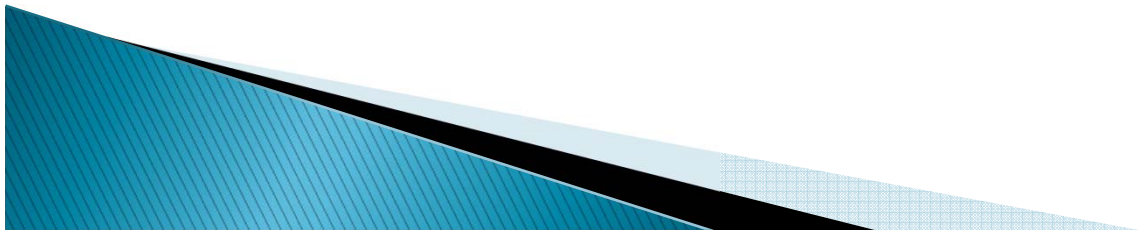
Java集合框架概述

- Java集合框架常用实现类
  - 在java.util包中除了定义一系列接口外，还定义了一系列实现了这些接口的具体类。在Java集合框架中最常用的集合实现类包括：
    - ArrayList类
    - LinkedList类
    - HashSet类
    - TreeSet类
    - HashMap类
    - TreeMap类
    - Vector、Hashtable和Properties



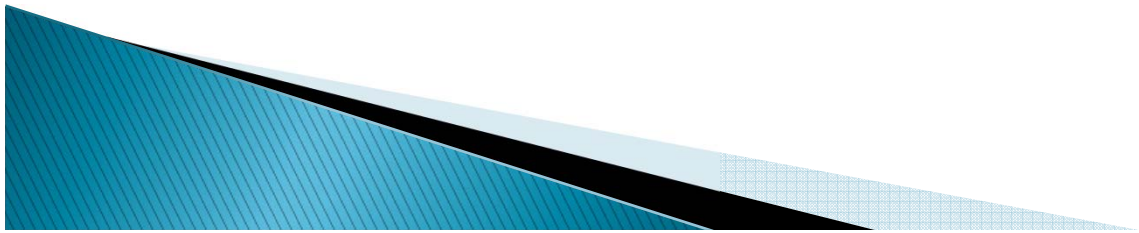
Java集合框架概述

- 集合框架的好处
  - 减轻编码的辛苦程度
  - 提高程序运行速度和质量
  - 不用再学习和使用新的集合API
  - 不用再设计新的集合API
  - 鼓励软件的复用和扩展



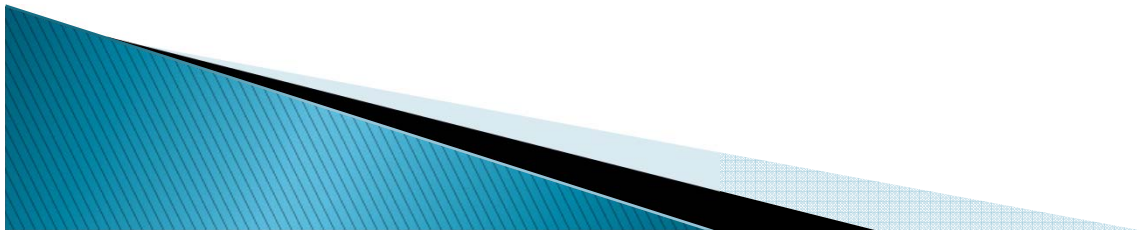
Collection接口

- Collection接口是Collection层次结构中的根接口。Collection接口用于表示任何对象或元素组，想要尽可能以常规方式处理一组元素时，就可以使用该接口。
- 在有些Collection接口中允许有重复的元素（如List），而另一些则不允许（如Set）。有些Collection接口是有序的（SortedSet），而另一些则是无序的（HashSet）。



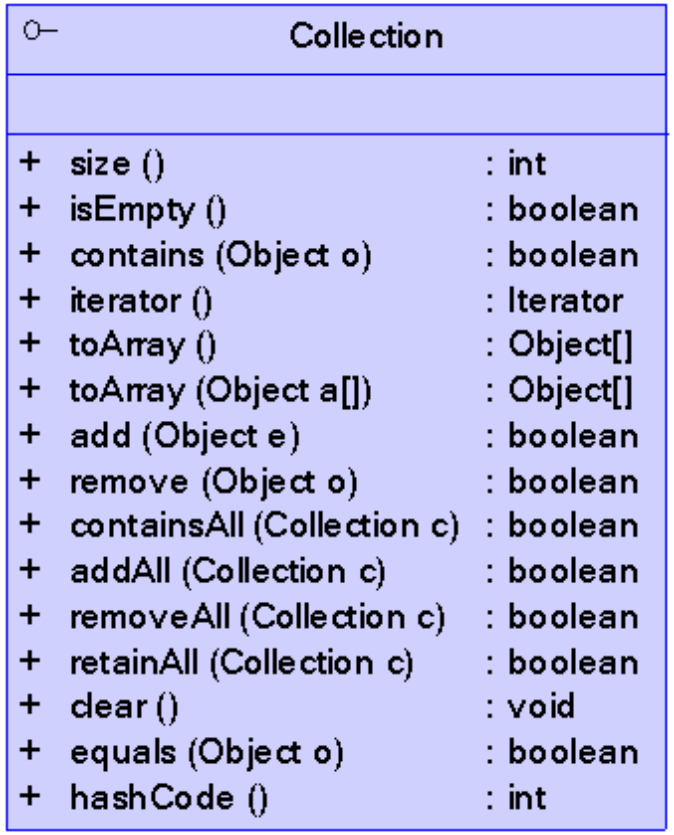
Collection接口

- 所有通用的Collection实现类应该提供两个构造函数：一个是无参构造函数，用于创建空的Collection；另一个是带有Collection类型的单参数构造函数，用于创建一个与其参数具有相同元素的新的Collection。



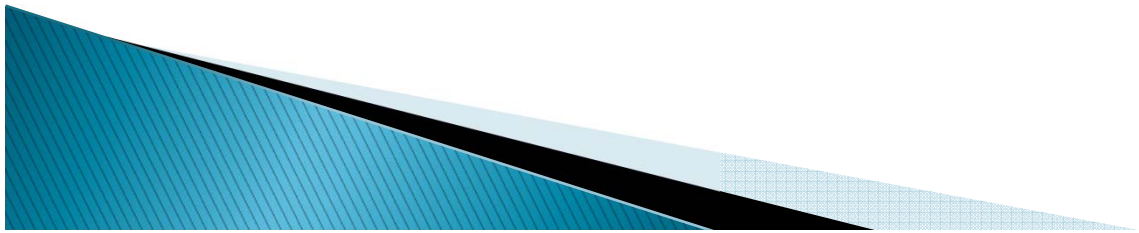
Collection接口

- Collection接口的UML类图表示如图所示：



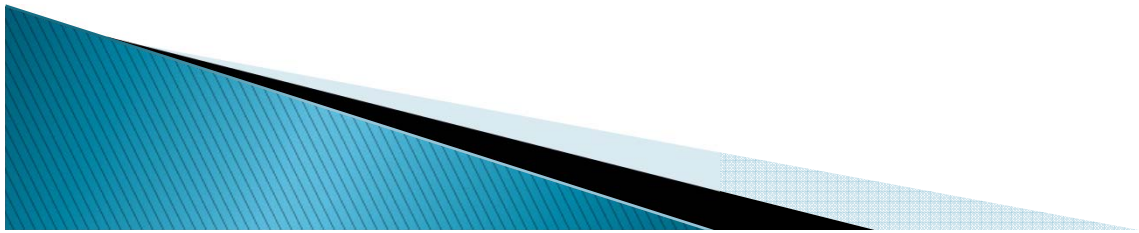
Collection接口

- Collection接口的常用方法如下：
  - 1. 单元素的添加和删除操作
    - boolean add(Object obj)：将对象obj添加到集合中。
    - boolean remove(Object obj)：如果集合中有与obj相匹配的对象，则删除对象obj。
  - 2. 元素查询操作
    - int size()：返回当前集合中元素的数量。
    - boolean isEmpty()：判断集合中是否有任何元素，如果有则返回true，否则返回false。
    - boolean contains(Object obj)：查找集合中是否包含对象obj。
    - Iterator iterator()：返回一个迭代器，用来访问集合中的各个元素。



Collection接口

- Collection接口的常用方法如下：
  - 3. 组操作
    - boolean containsAll(Collection c)：查找集合中是否含有集合c中所有元素。
    - boolean addAll(Collection c)：将集合c中所有元素添加给该集合。
    - void clear()：删除集合中所有元素。
    - void removeAll(Collection c)：从集合中删除集合c中的所有元素。
    - void retainAll(Collection c)：从集合中删除集合c中不包含的元素。
  - 4. 转换操作
    - Object[] toArray()：返回一个包含集合所有元素的数组。
    - Object[] toArray(Object[] a)：返回一个包含集合所有元素的数组，运行期返回的数组和参数a的类型必须相同，因此需要转换为正确类型。



迭代器

遍历频道

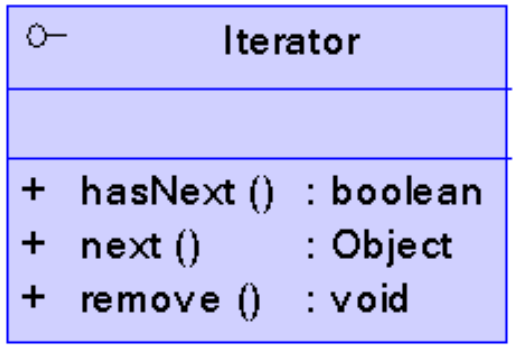
电视机遥控器

电视机

（电视频道的集合）

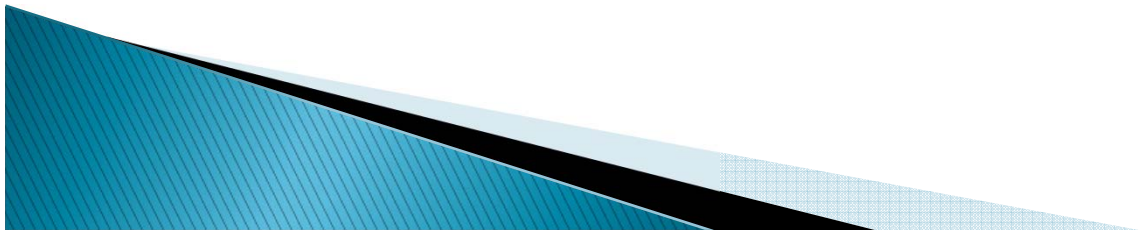
迭代器

- 迭代器概述
  - 使用遥控器可以方便人们对电视频道进行操作，而且不需要关心这些频道如何存储在电视机中。在这里，电视机对应于Collection集合，而遥控器对应于Iterator迭代器。
  - Collection接口的iterator()方法可以返回一个Iterator对象。使用Iterator接口的相应方法，可以从头至尾遍历集合，并安全地从Collection中删除元素。
  - Iterator接口的UML类图表示如图所示：



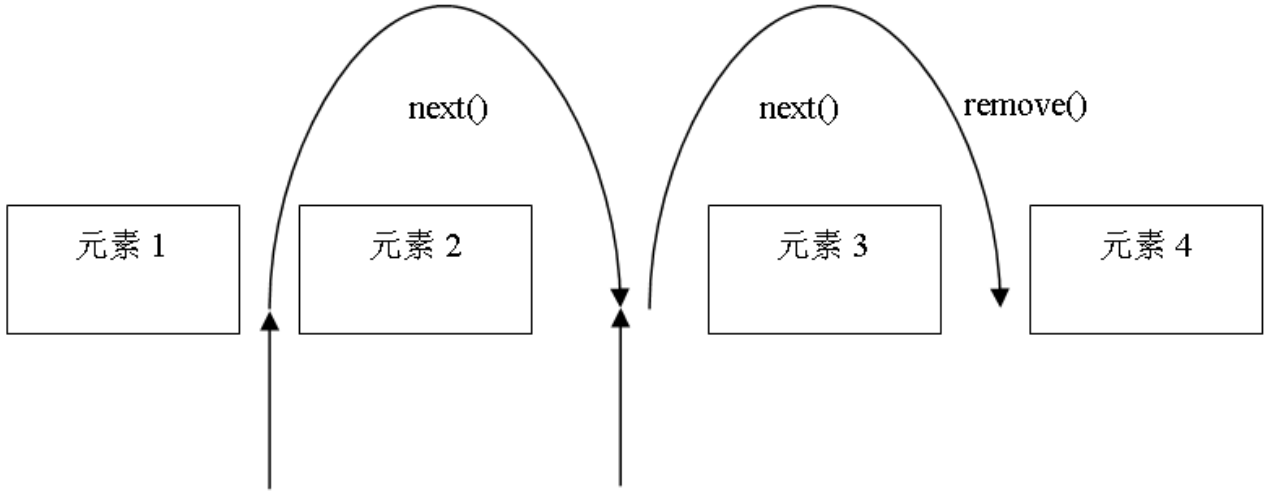
迭代器

- 迭代器概述
  - Iterator接口具有如下3个基本方法：
    - Object next()：通过反复调用next()方法可以逐个访问集合中的元素。
    - boolean hasNext()：为了不抛出异常，必须在调用next()之前先调用hasNext()。
    - void remove()：用于删除上次调用next()时所返回的元素。



迭代器

- 迭代器的特点
  - Java迭代器可以理解为它位于各个元素之间，每调用一次next()方法，迭代器便越过下个元素，并且返回它刚越过的那个元素的地址引用。但是，它也有一些限制，如某些迭代器只能单向移动。在使用迭代器时，访问某个元素的惟一方法就是调用next()。

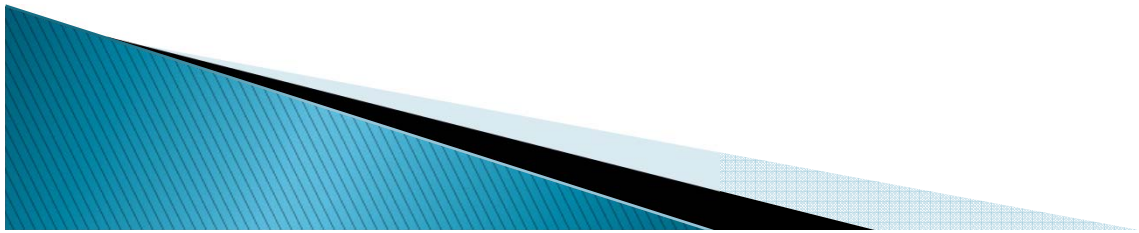


迭代器

- 迭代器的应用

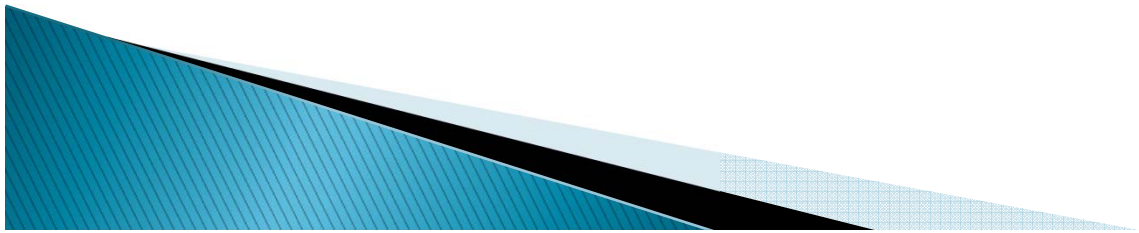
```
Iterator iterator1 = collection.iterator();//创建一个迭代器对象

    //通过迭代器遍历集合
    while (iterator1.hasNext())
    {
        Object element = iterator1.next();
        System.out.println("元素为：" + element);
    }
```



List接口

- List 接口继承了Collection接口，它是一种允许出现重复元素的有序集合。List又称为列表，它除了继承Collection接口的方法外，还增加了一系列方法用于对列表元素进行定位（索引）访问，即面向位置的操作。

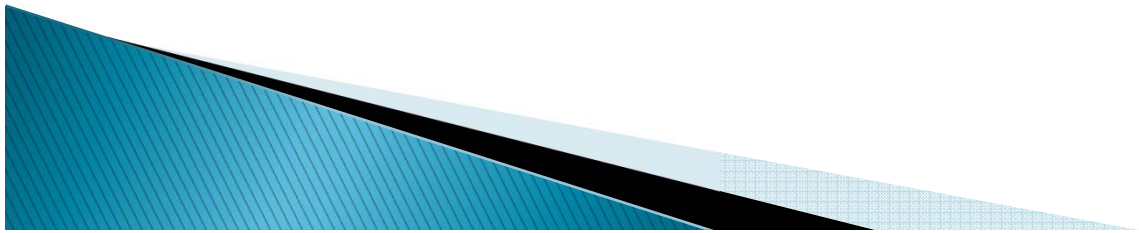


List接口

- List接口概述
  - List接口的UML类图表示如图所示：

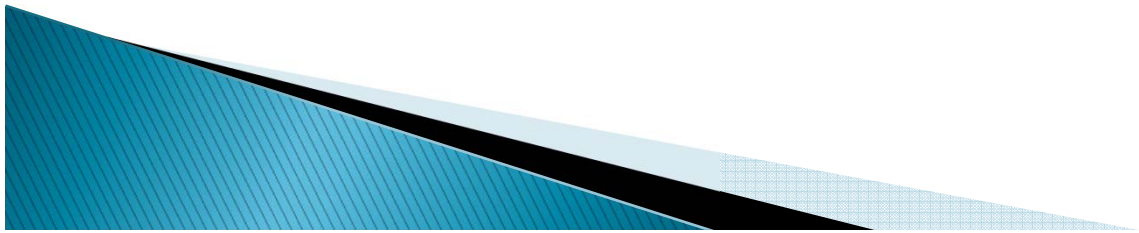
List接口

- List接口概述
  - 在List列表中，增加了一系列面向位置的操作，这些面向位置的操作既包括增加元素或集合的功能，还包括获取、删除和修改元素的功能。在List中搜索元素可以从列表的头部或尾部开始，如果找到元素，还可以报告元素所在的位置。



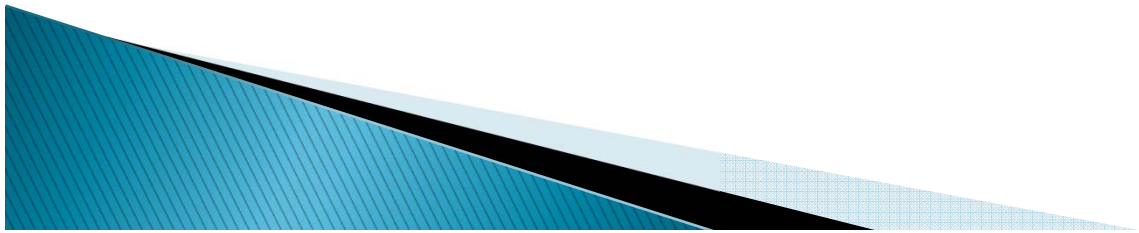
List接口

- List接口概述
  - List中常用的面向位置的操作如下：
    - void add(int index, Object element)：在指定位置index上添加元素element。
    - boolean addAll(int index, Collection c)：将集合c的所有元素添加到指定位置index。
    - Object get(int index)：返回List中指定位置index上的元素。
    - int indexOf(Object obj)：返回第一个出现元素obj的位置，如果不存在则返回-1。
    - int lastIndexOf(Object obj)：返回最后一个出现元素obj的位置，如果不存在则返回-1。
    - Object remove(int index)：删除指定位置index上的元素。
    - Object set(int index, Object element)：用元素element取代位置index上的元素，并且返回旧的元素。



List接口

- ArrayList类
  - ArrayList是用大小可变的数组实现的List。它实现了所有List列表的操作，并允许包括 null在内的所有元素。可以将ArrayList看成是能够自动增长容量的数组，也就是说当数组不够用的时候，再定义更大的数组，然后将数组元素拷贝到新的数组。

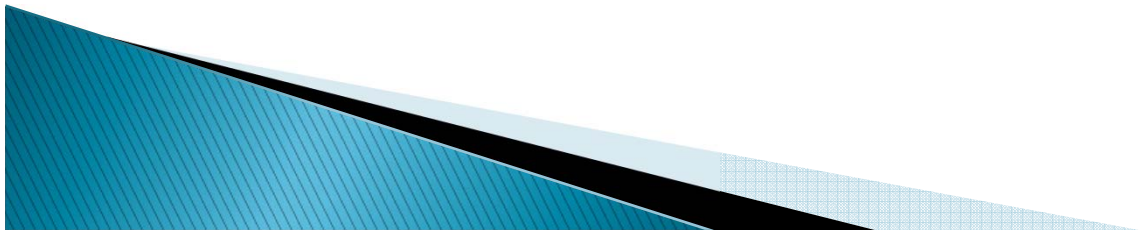


List接口

- 思考
  - 创建一个ArrayList，将“Changsha”、“上海”、“北京”、“深圳”、“武汉”依次增加到列表中，要求将第一个元素“Changsha”修改为“长沙”，并打印出ArrayList的第1个、第3个和第5个元素。

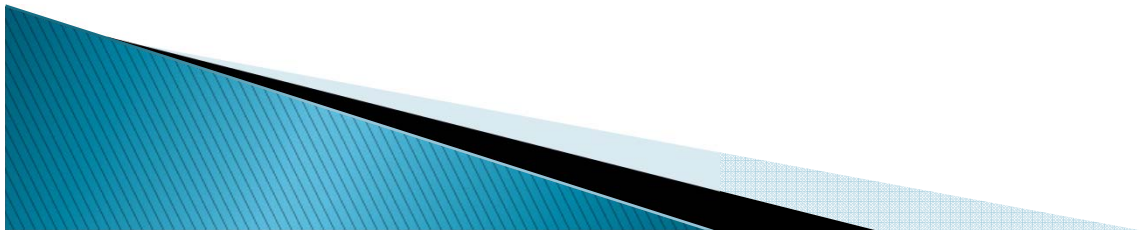
List接口

- LinkedList类
  - LinkedList是采用双向循环链表实现的List，其中每个元素对象除了包含数据本身外，还包含两个引用，分别指向前一个元素和后一个元素。



List接口

- LinkedList类
  - LinkedList类添加了一些处理列表两端元素的方法，其常用方法如下：
    - LinkedList(): 构造一个空的链接列表。
    - LinkedList(Collection c): 构造一个链接列表，并且添加集合c的所有元素。
    - void addFirst(Object obj): 将对象obj添加到列表的开头。
    - void addLast(Object obj)：将对象obj添加到列表的结尾。
    - Object getFirst(): 返回列表开头的元素。
    - Object getLast(): 返回列表结尾的元素。
    - Object removeFirst(): 删除并且返回列表开头的元素。
    - Object removeLast(): 删除并且返回列表结尾的元素。



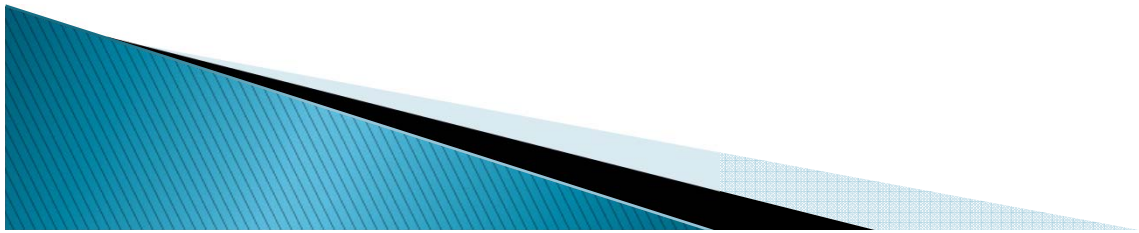
List接口

- 思考
  - 编写如下代码并观察输出结果。

```
import java.util.*;
public class LinkedListTest {
    public static void main(String args[])
    {
            LinkedList nameList;
            nameList=new LinkedList();
             nameList.addFirst("张三");
             nameList.addFirst("李四");
             for(int i=0;i<nameList.size();i++)
            {
                        System.out.println(nameList.get(i).toString());
            }
}
}
```

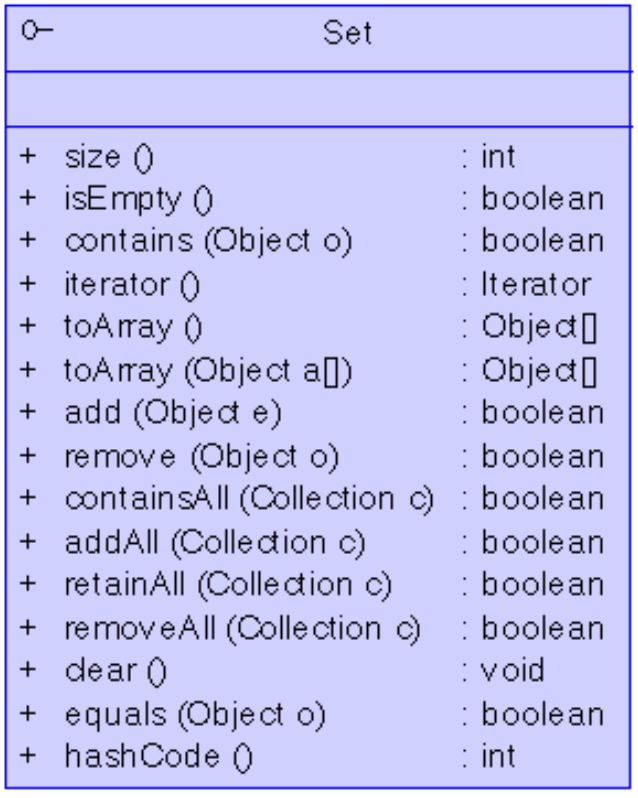
Set接口

- Set接口也继承了Collection接口，它是不允许出现重复项的集合。



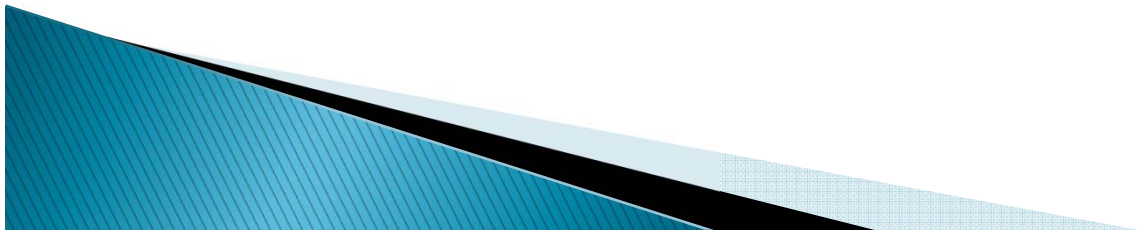
Set接口

- Set概述
  - Set接口的UML类图表示如图所示：



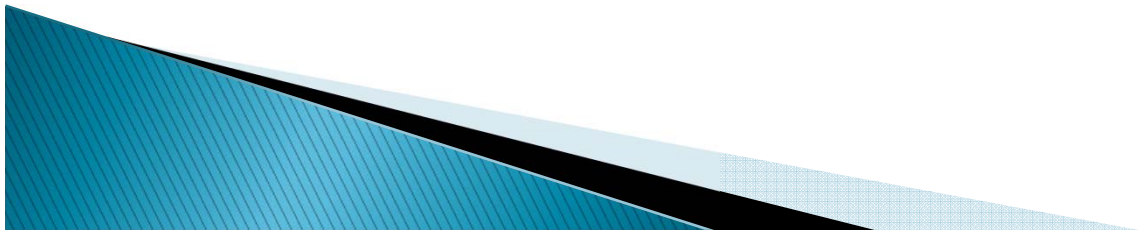
Set接口

- Set概述
  - 在Java集合框架中提供了两种Set实现，分别是HashSet和TreeSet（TreeSet实现了SortedSet接口）。



Set接口

- HashSet类
  - HashSet类使用散列表进行存储。在散列（hashing）中，可用键来确定惟一的一个值，称为散列码（hashcode），而散列码被用来当作与键相连的数据的存储下标。存储在HashSet中的元素所对应的类必须覆盖java.lang.Object中定义的hashCode()方法。
  - HashSet类的构造函数如下：
    - HashSet(): 构造一个空的哈希集。
    - HashSet(Collection c): 构造一个哈希集，并且添加集合c中所有元素。
    - HashSet(int initialCapacity): 构造一个拥有特定容量的空哈希集。
    - HashSet(int initialCapacity, float loadFactor): 构造一个拥有特定容量和加载因子的空哈希集。



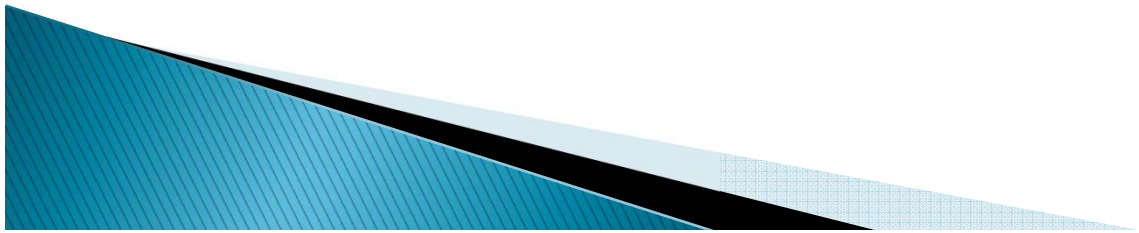
Set接口

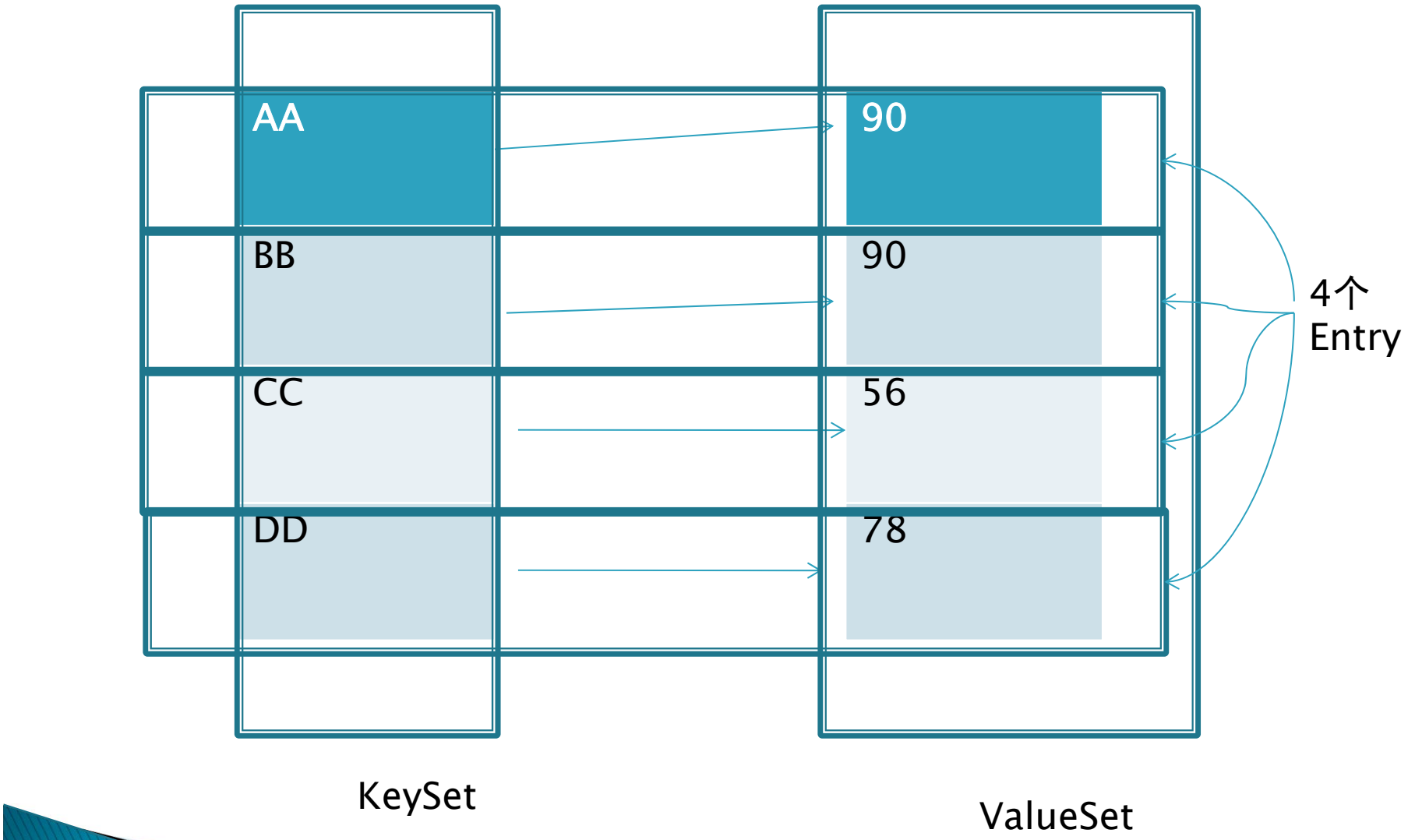
- 思考
  - 编写如下代码并观察输出结果。

```
import java.util.*;
public class HashSetTest {
    public static void main(String args[])
    {
                Set nameSet;
                nameSet=new HashSet();
                  nameSet.add("Tom");
                  nameSet.add("Sunny");
                  nameSet.add("Tom");
                  nameSet.add("Jim");
                  nameSet.add("Smith");
                  nameSet.add("Jack");
                  Iterator i=nameSet.iterator();
                while(i.hasNext())
                {
                                System.out.println(i.next().toString());
                }
                } }
```

Map接口

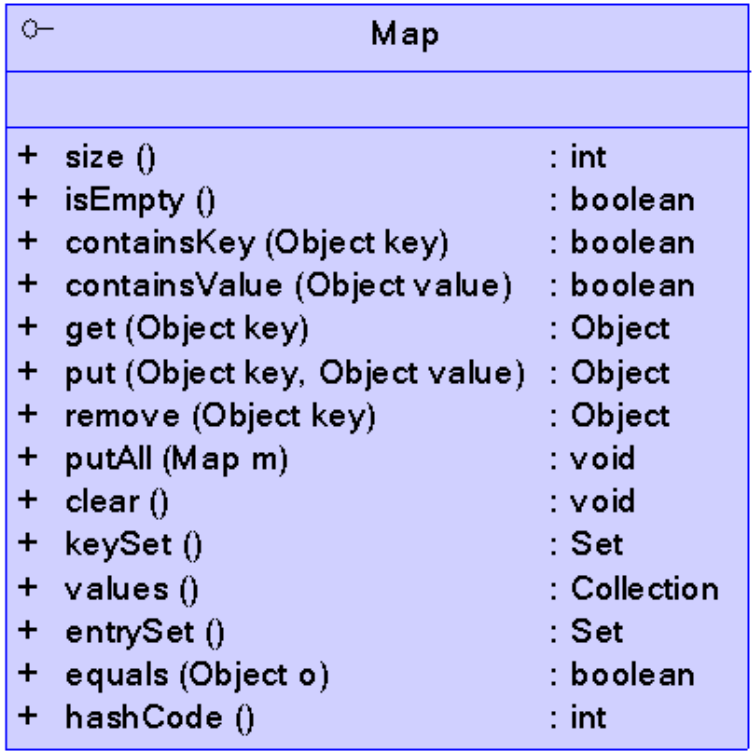
- Map接口不是Collection接口的子接口。Map接口用于存储键值对(key-value pairs)，它描述了从不重复的键到值的映射，是一种将键映射到值的对象。一个映射不能包含重复的键，且每个键最多只能映射到一个值。




AA
BB
CC
DD
90
90
56
78
4个
Entry
KeySet
ValueSet

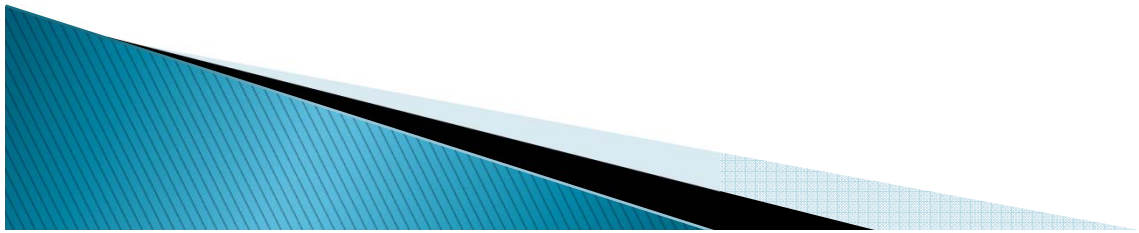
Map接口

- Map接口概述
  - Map接口的UML类图表示如图所示：



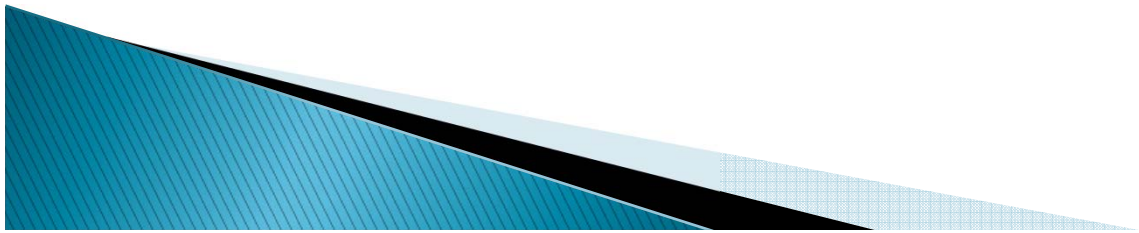
Map接口

- Map接口概述
  - Map接口的常用方法如下：
  - 1. 更改操作
    - (1) Object put(Object key, Object value)：用来添加一个键值对到Map中。
    - (2) Object remove(Object key)：根据key（键），删除一个键值对，并将值返回。
    - (3) void putAll(Map mapping)：将另一个Map中的元素存入当前的Map中。
    - (4) void clear()：清空当前Map中的所有元素。



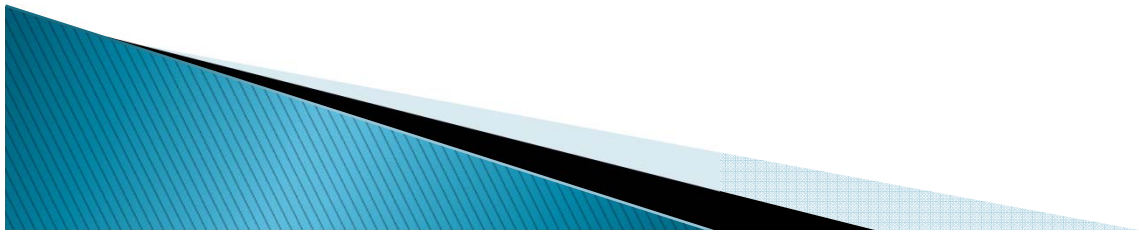
Map接口

- Map接口概述
  - Map接口的常用方法如下：
  - 2. 查询操作
    - (1) Object get(Object key)：根据key(键)获取对应的值。
    - (2) boolean containsKey(Object key)：判断Map中是否存在某个key（键）。
    - (3) boolean containsValue(Object value): 判断Map中是否存在某个value（值）。
    - (4) int size(): 返回Map中键值对的个数。
    - (5) boolean isEmpty()：判断当前Map是否为空。



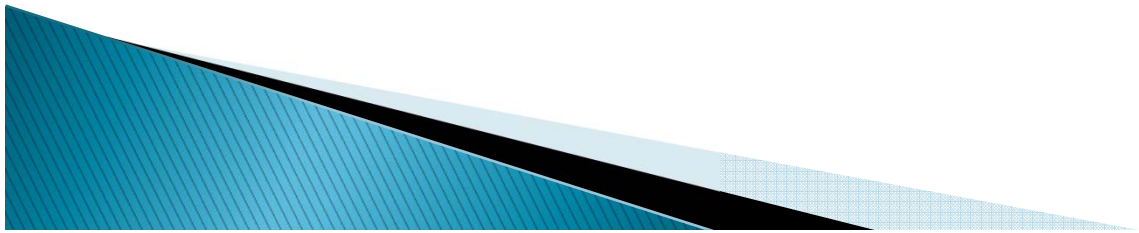
Map接口

- Map接口概述
  - Map接口的常用方法如下：
  - 3. 转换操作
    - (1) public Set keySet()：返回所有的key（键），并使用Set对象存放。
    - (2) public Collection values()：返回所有的Value（值），并使用Collection对象存放。
    - (3) public Set entrySet()：返回一个实现Map.Entry接口的Set对象。



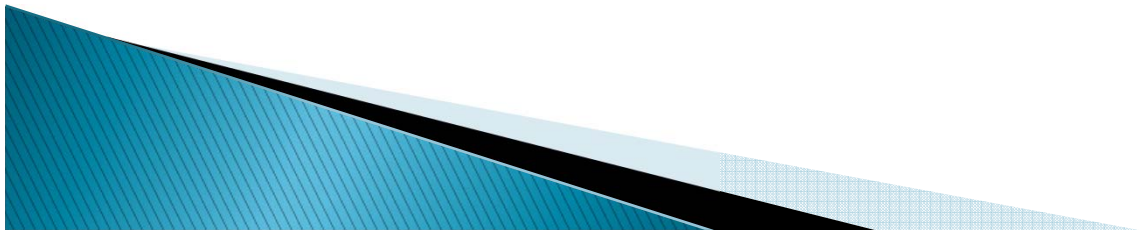
Map接口

- Map接口概述
  - 在Java集合框架中提供了两种Map实现，分别是HashMap和TreeMap（TreeMap实现了SortedMap接口）。



Map接口

- HashMap类
  - HashMap（哈希映像）是最常用的Map接口的子类，它能满足用户对Map的一般需求，键成员可为任意Object子类的对象，但如果覆盖了equals()方法，需要同时覆盖hashCode()方法。
  - HashMap构造函数如下：
    - HashMap(): 构造一个空的哈希映像。
    - HashMap(Map m): 构造一个哈希映像，并且添加映像m的所有元素到映像。
    - HashMap(int initialCapacity): 构造一个拥有特定容量的空的哈希映像。
    - HashMap(int initialCapacity, float loadFactor): 构造一个拥有特定容量和加载因子的空的哈希映像。



本章小结

- 在Java语言中可以将多个对象存储在一个集合中，这些集合对象又称为容器对象。Java SE设计者将这些集合对象根据其差异抽象出来，所抽象出来的数据结构和操作（算法）统称为Java集合框架（Java Collection Framework）。集合框架提供了一系列可以作为集合类型的类，这些类封装了对集合中元素的各种操作。
- Java集合框架的抽象层由一组接口组成，不同的接口用于定义不同类型的集合组。在Java SE中，所有的接口和类都定义在java.util包中，Java集合框架中常用的接口包括Collection、List、Set、SortedSet、Map和SortedMap，常用的类包括ArrayList、LinkedList、HashSet、TreeSet、HashMap和TreeMap。
- Collection接口是Collection层次结构中的根接口。Collection接口的常用方法包括单元素的添加和删除操作、元素查询操作、组操作和转换操作。
- 迭代器（Iterator）本身也是一个对象，它的工作就是遍历并获取集合中的对象，而程序员不必知道或关心该集合的内部结构。Collection接口的iterator()方法可以返回一个Iterator对象。使用Iterator接口的相应方法，可以从头至尾遍历集合，并安全地从Collection中删除元素。Iterator接口包含三个方法，分别是next()、hasNext()和remove()。

本章小结

- List 接口继承了Collection接口，它是一种允许出现重复元素的有序集合。List又称为列表，它除了继承Collection接口的方法外，还增加了一系列方法用于对列表元素进行定位（索引）访问，即面向位置的操作。
- 在Java集合框架中提供了两种List实现，分别是ArrayList和LinkedList。
- ArrayList是用大小可变的数组实现的List。它实现了所有List列表的操作，并允许包括null在内的所有元素。可以将ArrayList看成是能够自动增长容量的数组，也就是说当数组不够用的时候，再定义更大的数组，然后将数组元素拷贝到新的数组。
- LinkedList是采用双向循环链表实现的List，其中每个元素对象除了包含数据本身外，还包含两个引用，分别指向前一个元素和后一个元素。
- Set 接口也继承了Collection接口，它是不允许出现重复项的集合。
- 在Java集合框架中提供了两种Set实现，分别是HashSet和TreeSet（TreeSet实现了SortedSet接口）。

本章小结

- HashSet类使用散列表进行存储。在散列（hashing）中，可用键来确定惟一的一个值，称为散列码（hashcode），而散列码被用来当作与键相连的数据的存储下标。
- Map接口用于存储键值对(key-value pairs)，它描述了从不重复的键到值的映射，是一种将键映射到值的对象。一个映射不能包含重复的键，且每个键最多只能映射到一个值。
- 在Java集合框架中提供了两种Map实现，分别是HashMap和TreeMap（TreeMap实现了SortedMap接口）。
- HashMap（哈希映像）是最常用的Map接口的子类，它能满足用户对Map的一般需求。