* 一个ArrayList对象aList中有若干个字符串元素，现在遍历该ArrayList对象，删除其中所有值为abc的字符串元素。
* [html] [view plain](http://blog.csdn.net/yt741504089/article/details/47949689" \o "view plain) [copy](http://blog.csdn.net/yt741504089/article/details/47949689" \o "copy)
* package demo;
* import java.util.ArrayList;
* //一个ArrayList对象aList中有若干个字符串元素，现在遍历该ArrayList对象，删除其中所有值为abc的字符串元素。
* /\*\*
* \* 思路：
* \* 1.创建集合对象；
* \* 2.创建元素对象；
* \* 3.添加元素
* \* 4.遍历元素，删除指定元素
* \* 注意：正着遍历时，当集合中有连续的abc存在时，可能会出现删除不净的情况，需要避免
* \*      倒着遍历；正着遍历时没删除一个abc时，角标自动减一位，
* \*
* \*/
* public class Demo7 {
* public static void main(String[] args) {
* ArrayList<String> list = new ArrayList<String>();
* list.add("abc");
* list.add("bc");
* list.add("abc");
* list.add("abc");
* list.add("abc");
* list.add("bc");
* System.out.println(list);
* /\*//倒着遍历
* for (int i = list.size()-1; i >0; i--) {
* if(list.get(i).equals("abc")){
* list.remove(list.get(i));
* }
* }\*/
* //正着遍历
* for(int j=0;j<list.size();j++){
* if(list.get(j).equals("abc")){
* list.remove(j);
* //如果集合中有连续的abc存在，删除后，不j--，就会出现删除不净的情况
* j--;
* }
* }
* System.out.println(list);
* }
* }<strong>
* </strong>
* 取出一个字符串中字母出现的次数，如：“abcdekka27qoq”输出格式为a(2)b(1)k(2).....
* [html] [view plain](http://blog.csdn.net/yt741504089/article/details/47949689" \o "view plain) [copy](http://blog.csdn.net/yt741504089/article/details/47949689" \o "copy)
* package demo;
* import java.util.Set;
* import java.util.TreeMap;
* //取出一个字符串中字母出现的次数，如：“abcdekka27qoq”输出格式为a(2)b(1)k(2).....
* /\*\*
* \* 思路：
* \* 1.将字符串转换为字符数组
* \* 2.定义一个map集合，字符做键，数字做值，定义一个StringBuffer
* \* 3.遍历该数组，获取map集合的值，判断值是否为空，
* \*      是：直接添加
* \*      否：值++，在添加
* \* 4.获取所有键的集合，通过键来找值，
* \* 5.将键和值按指定格式添加到StringBuffer中
* \* 总结：这题还是比较综合的，数组的遍历，考到了string和StringBuffer，还有双列集合的键和值的获取,键的唯一性。
* \*
* \*/
* public class Demo9 {
* public static void main(String[] args) {
* String s="abcdekka27qoq";
* //字符串转换为字符数组
* char[] arr=s.toCharArray();
* StringBuffer bf = new StringBuffer();
* TreeMap<Character, Integer> map = new TreeMap<Character, Integer>();
* //遍历数组，来获取值，添加到集合
* for (int i = 0; i < arr.length; i++) {
* Integer in=map.get(arr[i]);
* if(in==null){
* map.put(arr[i], 1);
* }else{
* in++;
* map.put(arr[i], in);
* }
* }
* //获取键的集合
* Set<Character> set=map.keySet();
* //遍历键集合
* for(Character d:set){
* //获取键对应的值
* Integer dd=map.get(d);
* //按照指定的格式添加到StringBuffer
* bf.append(d+"("+dd+")");
* }
* System.out.println(bf);
* }
* }<strong>
* </strong>
* 编写程序，随机生成5个1-10之间的随机数，存入一个List集合中，编写方法对List集合进行排序。
* [html] [view plain](http://blog.csdn.net/yt741504089/article/details/47949689" \o "view plain) [copy](http://blog.csdn.net/yt741504089/article/details/47949689" \o "copy)
* package demo;
* import java.util.ArrayList;
* import java.util.Random;
* //编写程序，随机生成5个1-10之间的随机数，存入一个List集合中，编写方法对List集合进行排序，
* /\*\*
* \* 思路：
* \* 1.创建list集合，产生随机数的对象
* \* 2.向list集合添加5个随机数
* \* 3.用冒泡对集合排序（因为是对集合直接排序，元素替换的时候得用list集合的方法）
* \* 考点：产生随机数的类，list集合元素的添加，还有对集合直接排序。
* \*
* \*/
* public class Demo11 {
* public static void main(String[] args) {
* //创建list集合
* ArrayList<Integer> list = new ArrayList<Integer>();
* //创建产生随机数对象
* Random d = new Random();
* //添加5个随机数
* while(list.size()<5){
* Integer s=d.nextInt(10)+1;
* list.add(s);
* }
* //对集合直接进行冒泡排序
* for (int i = 0; i < list.size()-1; i++) {  //总共进行多少趟
* for (int j = 0; j < list.size()-1-i; j++) {
* if(list.get(j)>list.get(j+1)){
* int s=list.get(j);
* //int ss=list.get(j+1);
* //list.get(j)=list.get(j+1);
* //元素替换，使用list集合set方法
* list.set(j, list.get(j+1));
* list.set(j+1, s);
* }
* }
* }
* System.out.println(list);
* }
* }

**问题：**某班30个学生的学号为20070301-20070330,全部选修了Java程序设计课程，给出所有同学的成绩（可用随机数产生，范围60-100），请编写程序将本班各位同学的成绩按照从低到高排序打印输出。

**要求：**分别用List、Map、Set来实现，打印的信息包括学号、姓名和成绩。

**1、使用List集合来实现**

[?](http://www.jb51.net/article/95940.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84 | import java.util.ArrayList;  import java.util.Collections;  import java.util.Comparator;  import java.util.Iterator;  import java.util.LinkedList;  import java.util.TreeMap;        public class Test2{      public static void main(String[] args){        /\* 此处用ArrayList实现       \*       \* ArrayList<Student>al=new ArrayList<Student>();      for(int i=20070301,j=10;i<=20070330;i++,j++)      {        al.add(new Student(i,(int) (40\*Math.random()+60), "同学"+j));      }        //ArrayList排序借助Collections中的sort()方法实现。      Collections.sort(al, new Sortbygrade());      for(Student sd:al)      System.out.println(sd);        \*/        LinkedList<Student> lt=new LinkedList<Student>();      for(int i=20070301,j=10;i<=20070330;i++,j++)      {        lt.add(new Student(i,(int) (40\*Math.random()+60), "同学"+j));      }        //对链表排序      Collections.sort(lt, new Sortbygrade());      //输出链表      for(Student sd:lt)        System.out.println(sd);      }    }        //学生类  class Student{    int num,grade;    String name;      //构造函数    public Student(int num,int grade,String name){        this.num=num;      this.name=name;      this.grade=grade;    }      //此处必须覆写    public String toString(){  //   System.out.println("hi");      return "学号："+this.num+"\t"+"姓名:"+this.name+"  "+"成绩:"+this.grade;    }    }    //创建一个比较器类  class Sortbygrade implements Comparator<Student>{      @Override    public int compare(Student s1, Student s2) {        if(s1.grade>s2.grade)        return 1;      if(s1.grade<s2.grade)        return -1;      if(s1.grade==s2.grade)        return s1.name.compareTo(s2.name);      return 0;      }    } |

 输出结果如图：

对List集合框架的**总结：**

1、List集合其实是一个动态的数组，元素可以直接通过for循环取出，而不需要迭代。  
2、输出List集合时，会默认调用集合中存储对象的toString（）方法，所以在类中需要进行覆写。  
若不覆写toString( )方法，则必须使用

|  |  |
| --- | --- |
| 1  2  3  4  5 | for(int i=0;i<lt.size();i++)  {    Student s=lt.get(i);    System.out.println("学号："+s.num+"  姓名："+s.name+"  成绩："+s.grade);  } |

3、List集合的排序需要借助于Collections工具类，即Collections.Sort（list，new 比较器类（））方法。所以需要自定义一个比较器类，定义自己的比较规则。

**2、使用Set集合来实现**（1）使用TreeSet来实现

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60 | package com.package1;    import java.util.\*;    public class StuScore {    public static void main(String[] args) {      TreeSet<Student> ts=new TreeSet<Student>(new Com());    //添加元素进去    for(int i=20070301,j=1;i<=20070330;i++,j++)    {      ts.add(new Student(i,"同学"+j,(int) (40\*Math.random()+60)));    }      //迭代循环取出    Iterator<Student> it=ts.iterator();    while(it.hasNext())    {      Student o1=it.next();      System.out.println("学号："+o1.num+" "+"姓名："+o1.name+" "+" "+"成绩："+o1.grade);      }    }  }  //学生类  class Student  {  int num;  int grade;  String name;      public Student(int num, String name,int grade)  {    this.num=num;    this.name=name;    this.grade=grade;  }  }  class Com implements Comparator  {    @Override  public int compare(Object o1, Object o2) {      Student s1=(Student) o1;    Student s2=(Student) o2;    if(s1.grade>s2.grade)      return 1;    if(s1.grade<s2.grade)      return -1;    if(s1.grade==s2.grade)    {      return new Integer(s1.num).compareTo(new Integer(s2.num));    }    return 0;  }  } |

输出结果为：

学号：20070307  姓名：同学16    成绩：60  
学号：20070309  姓名：同学18    成绩：60  
对**TreeSet**的总结：  
1、元素不可以重复，而且TreeSet是有序的。  
2、两种排序方法：  
（1）自定义一个比较器类，比如class Com implementsComparator{ }  ，实现compare(Object o1, Object o2)方法，在其中定义比较规则。  
（2）让元素自身具备比较性。  
步骤：将add进TreeSet中的元素实现Comparable接口，并且覆盖compareTo方法。这种顺序也是元素的自然顺序，或者叫做默认顺序。

**方法1和方法2的区别：**

两种方法各有优劣, 用Comparable 简单, 只要实现Comparable 接口的对象直接就成为一个可以比较的对象,但是需要修改源代码。

用Comparator 的好处是不需要修改源代码, 而是另外实现一个比较器, 当某个自定义的对象需要作比较的时候,把比较器和对象一起传递过去就可以比大小了, 并且在Comparator 里面用户可以自己实现复杂的可以通用的逻辑,使其可以匹配一些比较简单的对象,那样就可以节省很多重复劳动了。

（2）使用**HashSet**来实现

[?](http://www.jb51.net/article/95940.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74 | package com.package1;    import java.util.\*;          public class StuScore {      public static void main(String[] args) {        HashSet<Student> hs=new HashSet<Student>();      //添加元素进去      for(int i=20070301,j=1;i<=20070330;i++,j++)      {          hs.add(new Student(i,"同学"+j,(int)    (40\*Math.random()+60)));      }        ArrayList<Student>li=new ArrayList(hs);        Collections.sort(li, new Sortbygrade());        for(Student ss:li)        System.out.println(ss);        }    }  //学生类  class Student  {    int num;    int grade;    String name;        public Student(int num, String name, int grade)    {      this.num=num;      this.name=name;      this.grade=grade;    }    public String toString(){      //System.out.println("hi");      return "学号："+this.num+"\t"+"姓名:"+this.name    +"  "+"成绩:"+this.grade;    }      }      class Sortbygrade implements Comparator{      @Override    public int compare(Object o1, Object o2) {        Student s1=(Student) o1;      Student s2=(Student) o2;      if(s1.grade>s2.grade)        return 1;      if(s1.grade<s2.grade)        return -1;  //   if(s1.grade==s2.grade)        return 0;    }    } |

输出结果如下：  
学号：20070310 姓名:同学19    成绩:60  
学号：20070330 姓名:同学39    成绩:62  
对**HashSet**的总结：  
1、HashSet中的元素不可以重复，如果重复添加，则只会显示一个。  
原理如下：  
HashSet:底层数据结构是哈希表。是线程不安全的。不同步。  
2、HashSet是如何保证元素唯一性的呢？  
答：是通过元素的两个方法，hashCode和equals来完成。  
如果元素的HashCode值相同，才会判断equals是否为true。如果元素的hashcode值不同，不会调用equals。  
3、对HashSet的排序，通过将Set集合转化为List集合，借助Collections.Sort( )方法实现排序。

**3、使用TreeMap来实现**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58 | package com.package1;    import java.util.Comparator;  import java.util.Iterator;  import java.util.Set;  import java.util.TreeMap;    public class TestTreeMap {      public static void main(String[] args) {      //1.创建集合      TreeMap<Student,Integer> tm=new TreeMap<Student,Integer>();      for(int i=20070301,j=10;i<=20070330;i++,j++)      {        int grade=(int) (40\*Math.random()+60);         //2、往集合对象中添加元素         tm.put(new Student(grade,"同学"+j),i);      }        //3.遍历集合 ，排序完成      Set<Student> k=tm.keySet();      Iterator<Student> it=k.iterator();        while(it.hasNext()){        Student key=it.next();        Integer num=tm.get(key);          System.out.println("学号："+num+"  "+"姓名："+key.name+"  "+"成绩:"+key.grade);      }    }    }    class Student implements Comparable<Student>{    int grade;    String name;      public Student(int grade,String name){      this.grade =grade;      this.name=name;    }      @Override    public int compareTo(Student o) {        if(this.grade>o.grade)        return 1;        if(this.grade==o.grade)        { //当成绩相同时，按照姓名排序         return this.name.compareTo(o.name);        }        return -1;      }        } |

 输出结果为：

学号：20070303    姓名：同学12    成绩:61  
学号：20070323    姓名：同学32    成绩:61  
学号：20070317    姓名：同学26    成绩:62  
学号：20070309    姓名：同学18    成绩:64  
学号：20070326    姓名：同学35    成绩:98  
学号：20070311    姓名：同学20    成绩:99

对**TreeMap**的总结：  
1、TreeMap默认对key进行排序，所以可将自定义对象放入key中，将代表学号的整型放入value中。对Key排序时，可以指定自定义对象中的某个属性来排序。  
2、Map集合使用put（）方法添加元素。  
3、Map集合的取出原理：将map集合转成set集合。在通过迭代器取出。map集合的两种取出方式：  
**（1）Set<k> keySet：**将map中所有的键存入到Set集合。因为set具备迭代器。所有可以迭代方式取出所有的键，在根据get方法。获取每一个键对应的值。  
**（2）Set<Map.Entry<k,v>> entrySet：**将map集合中的映射关系存入到了set集合中，而这个关系的数据类型就是：Map.Entry

以上就是本文的全部内容，希望对大家的学习有所帮助，也希望大家多多支持脚本之家。

**1为什么Map接口不继承Collection 接口？**

· Set是无序集合，并且不允许重复的元素

· List是有序的集合，并且允许重复的元素

· 而Map是键值对

· 它被视为是键的set和值的set的组合

· Map被设计为键值对的集合，所以不需要继承Collection 接口

**2HashMap和Hashtable之间的区别？**

· 同步或线程安全

· Null键和Null值

· 迭代值

· 默认容量大小

**3comparable 和 comparator的不同之处？**

· comparable接口实际上是出自java.lang包

· 它有一个 compareTo(Object obj)方法来将objects排序

· comparator接口实际上是出自 java.util 包

· 它有一个compare(Object obj1, Object obj2)方法来将objects排序

**4如何对Object的list排序？**

· 对objects数组进行排序，我们可以用方法Arrays.sort()方法

· 如果要对objects的集合进行排序，需要使用Collections.sort()

**5fail-fast 与 fail-safe 之间的区别？**

· Fail fast快速地报告任何的failure。无论何时任何一个问题都会引发 fail fast系统fails

· 在Java Fail fast 迭代器中，迭代objects集合有时会出现并发修改异常，出现这种情况有2个原因

· 如果一个线程正在迭代一个集合，而另一个线程同时试图修改这个集合

· 在调用remove()方法后，如何我们还试图去修改集合object

**6Iterator、ListIterator 和 Enumeration的区别？**

· Enumeration接口在Java1.2版本开始有，所以Enumeration是合法规范的接口

· Enumeration使用elements()方法

· Iterator对所有Java集合类都有实现

· Iterator使用iterator方法

· Iterator只能往一个方向前进

· ListIterator仅仅对List类型的类实现了

· ListIterator使用listIterator（）方法

**7Java 中 Set 与 List 有什么不同?**

· Set是一个不允许重复元素存在的集合

· Set没有索引

· Set仅仅允许一个null值

· Set有类：HashSet、LinkedHashMap、TreeSet

· List有索引

· List允许N个null值

· List可以按插入顺序显示

· List有类：Vector、ArrayList、LinkedList

**8arraylist 与 vector 的区别?**

· Vector 在Java的第一个版本就引入了，也就是说vector是一个合法规范的类

· ArrayList在Java1.2版本引入的，是Java 集合框架的组成部分

· Vector是同步的

· ArrayList是不同步的

**9什么类实现了List接口？**

· ArrayList

· LinkedList

· Vector

**10什么类实现了Set接口？**

· HashSet

· LinkedHashSet

· TreeSet

**11如何保证一个集合线程安全？**

· Vector, Hashtable, Properties 和 Stack 都是同步的类，所以它们都线程安全的，可以被使用在多线程环境中

· 使用Collections.synchronizedList(list)) 方法，可以保证list类是线程安全的

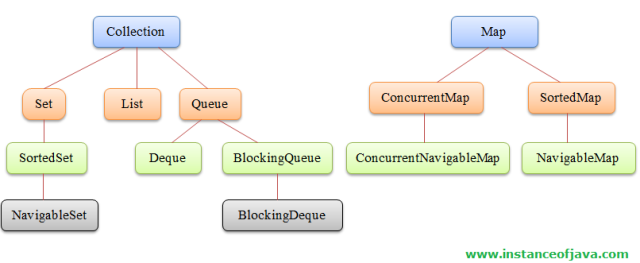
· 使用java.util.Collections.synchronizedSet()方法可以保证set类是线程安全的

**12是否可以往 TreeSet 或者 HashSet 中添加 null 元素？**

· 可以往 hashset 中添加一个 null

· TreeSet 也允许一个 null值

**13解释下Collection的接口继承关系？**



**14Iterator符合哪个设计模式？**

· Iterator 设计模式

**15HashSet 实现了哪个数据结构？**

· HashSet 内部实现了hashmap

**16为什么 Collection 不能继承 Cloneable 和 Serializable？**

· List和Set唯一继承 Collection 接口

· SortedMap 继承了 Map 接口

**17hashCode() 和 equals() 方法的重要性？如何在Java中使用它们？**

· hashCode() 和 equals() 方法定义在"object"类中

· 如果equals() 方法在比较2个对象时返回true，那么hashCode()的返回值必须得一样

**18array 和 arraylist 的区别？**

· Array类似object集合类型，大小固定

· Arraylist是同质和异质元素的集合

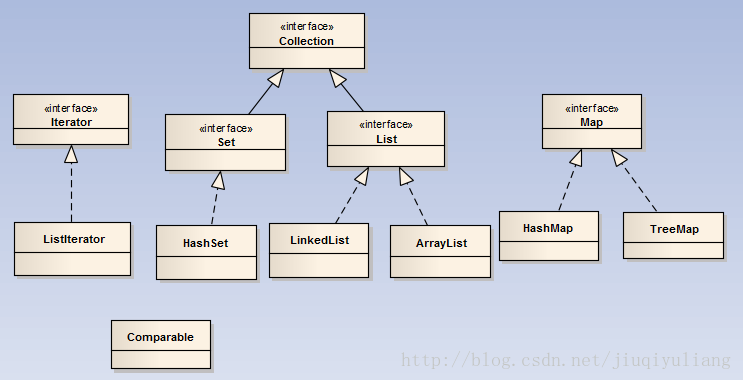
**19什么是 Properties 类？**

· Properties 是Hashtable的子类。它被用于维护值的list，其中它们的键、值都是String类型

**20如何将一个字符串转换为arraylist?**

· 使用 arrayList.toArray() 方法

首先，我们先看一下Collection集合的基本结构：



1、Collection接口

Collection是最基本集合接口，它定义了一组允许重复的对象。Collection接口派生了两个子接口Set和List，分别定义了两种不同的存储方式，如下：

2、 Set接口

Set接口继承于Collection接口，它没有提供额外的方法，但实现了Set接口的集合类中的元素是无序且不可重复。

特征：无序且不可重复。

3、  List接口

List接口同样也继承于Collection接口，但是与Set接口恰恰相反，List接口的集合类中的元素是对象有序且可重复。

特征：有序且可重复。

两个重要的实现类：ArrayList和LinkedList

1.ArrayList特点是有序可重复的

2.LinkedList是一个双向链表结构的。

4、Map接口

Map也是接口，但没有继承Collection接口。该接口描述了从不重复的键到值的映射。Map接口用于维护键/值对（key/value pairs）。

特征：它描述了从不重复的键到值的映射。

两个重要的实现类：HashMap和TreeMap

1.HashMap，中文叫散列表，基于哈希表实现，特点就是键值对的映射关系。一个key对应一个Value。HashMap中元素的排列顺序是不固定的。更加适合于对元素进行插入、删除和定位。

2.TreeMap，基于红黑书实现。TreeMap中的元素保持着某种固定的顺序。更加适合于对元素的顺序遍历。

5、Iterator接口

Iterator接口，在C#里有例外一种说法IEnumerator，他们都是集合访问器，用于循环访问集合中的对象。  
所有实现了Collection接口的容器类都有iterator方法，用于返回一个实现了Iterator接口的对象。Iterator对象称作迭代器，Iterator接口方法能以迭代方式逐个访问集合中各个元素，并可以从Collection中除去适当的元素。

[csharp] [view plain](http://blog.csdn.net/jiuqiyuliang/article/details/32697903" \o "view plain) [copy](http://blog.csdn.net/jiuqiyuliang/article/details/32697903" \o "copy)

<span style="font-size:10px;">Iterator it = collection.iterator(); // 获得一个迭代子

while(it.hasNext())

{

Object obj = it.next(); // 得到下一个元素

} </span>

6、Comparable接口

Comparable可以用于比较的实现，实现了Comparable接口的类可以通过实现comparaTo方法从而确定该类对象的排序方式。

总结

Collection集合分别派生自Collection和Map接口，Collection有两个常用子接口List和Set，分别表示有序可重复，无序不可重复的集合。而Map存储的是key-value的映射。

上面的总结看上去很多，很繁琐，其实你只要记得一点：collection是用于处理各种数据结构的，根据各种数据结构的特点理解，一切都会变简单。