链滴

集合框架 | 概念与 API

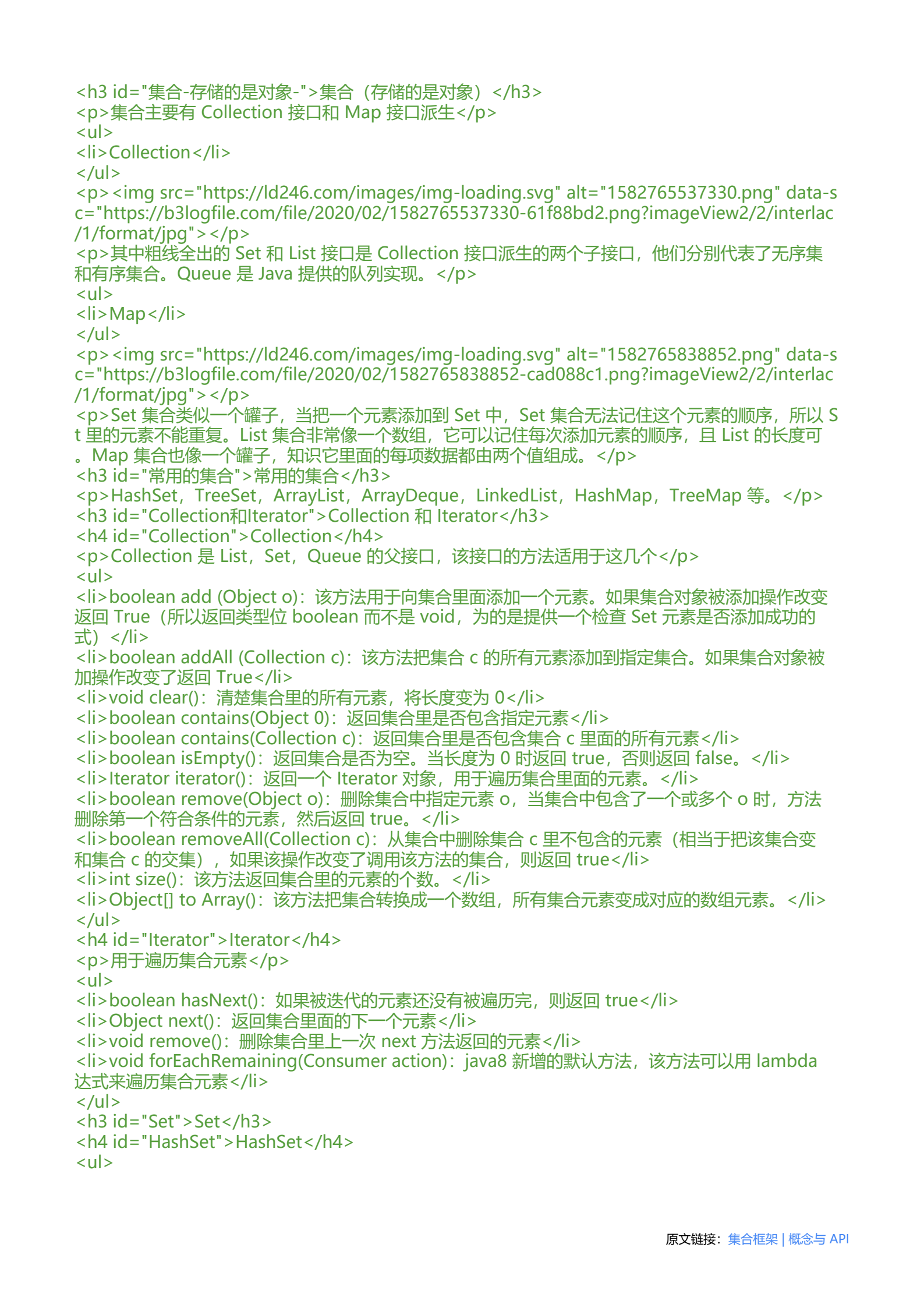
作者：douniwan

原文链接：https://ld246.com/article/1582989659487

来源网站：链滴

<h3 id="集合-存储的是对象-">集合（存储的是对象）</h3>
集合主要有 Collection 接口和 Map 接口派生

Collection


其中粗线全出的 Set 和 List 接口是 Collection 接口派生的两个子接口，他们分别代表了无序集和有序集合。Queue 是 Java 提供的队列实现。

Map

Set 集合类似一个罐子，当把一个元素添加到 Set 中，Set 集合无法记住这个元素的顺序，所以 Set 里的元素不能重复。List 集合非常像一个数组，它可以记住每次添加元素的顺序，且 List 的长度可。Map 集合也像一个罐子，知识它里面的每项数据都由两个值组成。
<h3 id="常用的集合">常用的集合</h3>
HashSet，TreeSet，ArrayList，ArrayDeque，LinkedList，HashMap，TreeMap 等。
<h3 id="Collection和Iterator">Collection 和 Iterator</h3>
<h4 id="Collection">Collection</h4>
Collection 是 List，Set，Queue 的父接口，该接口的方法适用于这几个

boolean add (Object o)：该方法用于向集合里面添加一个元素。如果集合对象被添加操作改变返回 True（所以返回类型位 boolean 而不是 void，为的是提供一个检查 Set 元素是否添加成功的式）
boolean addAll (Collection c)：该方法把集合 c 的所有元素添加到指定集合。如果集合对象被加操作改变了返回 True
void clear()：清楚集合里的所有元素，将长度变为 0
boolean contains(Object 0)：返回集合里是否包含指定元素
boolean contains(Collection c)：返回集合里是否包含集合 c 里面的所有元素
boolean isEmpty()：返回集合是否为空。当长度为 0 时返回 true，否则返回 false。
Iterator iterator()：返回一个 Iterator 对象，用于遍历集合里面的元素。
boolean remove(Object o)：删除集合中指定元素 o，当集合中包含了一个或多个 o 时，方法删除第一个符合条件的元素，然后返回 true。
boolean removeAll(Collection c)：从集合中删除集合 c 里不包含的元素（相当于把该集合变和集合 c 的交集），如果该操作改变了调用该方法的集合，则返回 true
int size()：该方法返回集合里的元素的个数。
Object[] to Array()：该方法把集合转换成一个数组，所有集合元素变成对应的数组元素。

<h4 id="Iterator">Iterator</h4>
用于遍历集合元素

boolean hasNext()：如果被迭代的元素还没有被遍历完，则返回 true
Object next()：返回集合里面的下一个元素
void remove()：删除集合里上一次 next 方法返回的元素
void forEachRemaining(Consumer action)：java8 新增的默认方法，该方法可以用 lambda 达式来遍历集合元素

<h3 id="Set">Set</h3>
<h4 id="HashSet">HashSet</h4>

元素不能重复
按 Hash 算法来存储集合中的元素
具有良好的存取和查找性能
不能保证元素的排列顺序，顺序可能与添加顺序不同
不是同步的，如果有两个或以上的线程同时修改 HashSet，则必须通过代码来保持同步
集合元素可以为 null

存入元素时，HashSet 调用该对象的的 HashCode 值，然后根据他们的值决定他们 HashSet 中存储位置。如果两个元素通过 equals()方法比较返回 true，但他们的 hashCode 方法的返值不相等，HashSet 将会把他存储在不同的位置，依然可以添加成功。如果 hashCode 相，而 equals 不同，也会认为是两个对象（通过链式结构来保存多个对象）</p

HashSet 中每个能存储元素的"槽位（slot）"通常被称为"桶"（bucket）
如果有多个元素的 hashcode 值相同，但他们通过 equals 方法比较返回 false，就需要在一个桶”
里面放多个元素，这样会导致性能下降。
<h5 id="重写hashcode的步骤">重写 hashcode 的步骤</h5>

把对象内每个有意义的实例变量（即每个参与 equals()方法比较标准的实例变量）计算出一个 int 类型的 hashCode 值。

用第一步算出来的多个 hashcode 值组合计算出一个 hashcode 的值

为了避免每个 hashcode 相加导致的相等，可以位每个 hashcode 值乘以一个质树后再相加。</
>

<h5 id="注意">注意</h5>
当程序把可变对象添加到 HashCode 中后，精良不要去修改集合元素参与计算 hashcode 和 eq
als 的实例遍历。
<h4 id="LinkedHashSet">LinkedHashSet</h4>

是 HashSet 的子类
也是根据 hashcode 来决定元素的存储位置，但他同时使用链表来维持元素的次序。
性能略低于 HashSet
虽然使用链表记录了集合元素的添加顺序，淡 LinkedHashSet 依然是 HashSet，因此他依然不许元素重复。

<h4 id="TreeSet">TreeSet</h4>

是 SortedSet 的实现类，TreeSet 可以确保集合元素处于排序状态。
与，HashSet 相比。TreeSet 提供了几个额外的方法

Comparator Comparator()：如果 TreeSet 采用了定制排序，则该方法返回定制排序所使用的 C
mparator；如果 TreeSet 采用了自然排序，则返回 null；
Object first()：返回集合中的第一个元素
Object last()：返回集合中的最后一个元素
Object lower(Object e)：返回集合中位于指定元素之前的元素（就是返回小于指定元素的最大

），参考元素不需要是 TreeSet 中的元素
Object higher(Object e)：返回集合中位于指定元素之后的元素（就是返回大于指定元素的最小），参考元素不需要是 TreeSet 中的元素
SortedSet subSet(Object fromElement , Object toElement)：返回此 Set 的子集，范围从 fromElement （包含）到 toElement（不包含）
SortedSet headSet(Object toElement)：返回此 Set 的子集，由小于 toElement 的元素组成
SortedSet tailSet(Object toElement)：返回此 Set 的子集，由大于 toElement 的元素组成

<h5 id="自然排序">自然排序</h5>

TreeSet 放入一个对象时会调用集合元素的 compareTo(Object obj)方法来比较集合元素的大小系，然后按升序排序，这种方式就是自然排序。
java 提供了一个 Compareable 接口，该接口定义了一个 compareTo(Object obj)方法，该方返回一个整数值，实现该接口的类必须实现该方法。实现了该方法的对象与另一个对象进行比较的时，比如 obj1.compareTo(obj2)，如果该方法返回 0，说明 obj1 与 obj2 相等，如果返回一个正整数明 obj1 大于 obj2，如果返回一个负数，表示 obj1 小于 obj2。

<img src="https://ld246.com/images/img-loading.svg" alt="1582774653739.png" data-sc="https://b3logfile.com/file/2020/02/1582774653739-68a1fea8.png?imageView2/2/interlace1/format/jpg">
<pre><code class="language-java highlight-chroma">package TreeSetTest;

import java.util.TreeSet;

class Student implements Comparable {
    /**
     * 学生的姓名
     */
    private String name;
    /**
     * 学生的年龄
     */
    private Integer age;

```
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-cm">/*·····
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">    * 省略的getter和setter
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">    */</span><span class="highlight-err">·····</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-nd">@Override</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-kd">public</span> <span class="highlight-kt">int</span> <span class="highlight-n
">compareTo</span><span class="highlight-o">(</span><span class="highlight-n">Objec
</span> <span class="highlight-n">o</span><span class="highlight-o">)</span> <span cl
ss="highlight-o">{</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">Student</span> <span class="highlight-n">student</span> <span class="high
ight-o">=</span> <span class="highlight-o">(</span><span class="highlight-n">Student<
span><span class="highlight-o">)</span><span class="highlight-n">o</span><span class
"highlight-o">;</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-k">return</span> <span class="highlight-k">this</span><span class="highlight-
">.</span><span class="highlight-na">getAge</span><span class="highlight-o">()</span
 <span class="highlight-o">&amp;gt;</span> <span class="highlight-n">student</span><span c
ass="highlight-o">.</span><span class="highlight-na">getAge</span><span class="highli
ht-o">()</span> <span class="highlight-o">?</span> <span class="highlight-mi">1</span
 <span class="highlight-o">:</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span cl
ss="highlight-k">this</span><span class="highlight-o">.</span><span class="highlight-na
>getAge</span><span class="highlight-o">()</span> <span class="highlight-o">&amp;lt;</spa
> <span class="highlight-n">student</span><span class="highlight-o">.</span><span clas
="highlight-na">getAge</span><span class="highlight-o">()</span> <span class="highligh
-o">?</span> <span class="highlight-o">-</span><span class="highlight-mi">1</span> <
pan class="highlight-o">:</span> <span class="highlight-mi">0</span>  <span class="highl
ght-o">;</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-o">}</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-o">}</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-cm">/**
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm"> * @author yaoqiuhong
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm"> * @create 2020-02-27 11:39
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm"> * @description
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm"> */</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-kd">public</span> <span class="highlight-kd">class</span> <span class="highlight-n
">TreeSetDemo</span> <span class="highlight-o">{</span>
```

```
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-kd">public</span> <span class="highlight-kd">static</span> <span class="highlig
t-kt">void</span> <span class="highlight-nf">main</span><span class="highlight-o">(</
pan><span class="highlight-n">String</span><span class="highlight-o">[]</span> <span c
ass="highlight-n">args</span><span class="highlight-o">)</span> <span class="highlight
o">{</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">// 创建一个学生对象1
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-n">Student</span> <span class="hi
hlight-n">student1</span> <span class="highlight-o">=</span> <span class="highlight-k"
new</span> <span class="highlight-n">Student</span><span class="highlight-o">();</spa
>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">student1</span><span class="highlight-o">.</span><span class="highlight-n
">setName</span><span class="highlight-o">(</span><span class="highlight-s">"yqh"</
pan><span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">student1</span><span class="highlight-o">.</span><span class="highlight-n
">setAge</span><span class="highlight-o">(</span><span class="highlight-mi">21</spa
><span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">// 创建一个学生对象2
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-n">Student</span> <span class="hi
hlight-n">student2</span> <span class="highlight-o">=</span> <span class="highlight-k"
new</span> <span class="highlight-n">Student</span><span class="highlight-o">();</spa
>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">student2</span><span class="highlight-o">.</span><span class="highlight-n
">setName</span><span class="highlight-o">(</span><span class="highlight-s">"yyw"</
pan><span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">student2</span><span class="highlight-o">.</span><span class="highlight-n
">setAge</span><span class="highlight-o">(</span><span class="highlight-mi">20</spa
><span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">// 创建一个TreeSet来存储学生对象
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-n">TreeSet</span> <span class="hig
light-n">treeSet</span> <span class="highlight-o">=</span> <span class="highlight-k">n
w</span> <span class="highlight-n">TreeSet</span><span class="highlight-o">();</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-cm">/*
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">         * 如果Student没有实现Comparable中的compareTo方法，那么下面将St
dent加入
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">         * treeSet中会出现java.lang.ClassCastException
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">         */</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">treeSet</span><span class="highlight-o">.</span><span class="highlight-na
>add</span><span class="highlight-o">(</span><span class="highlight-n">student1</sp
```

n>);
        <span class=
highlight-n">treeSet.<span class="highlight-na
>add(student2</sp
n>);
    <span class="h
ghlight-o">}
<span class="high
ight-o">}
</code></pre>

大部分对象在实现 compareTo(Object obj)的时候，需要将 Object 对象强转成本类对象经行某属性的比较。而又由于在向 TreeSet 中添加元素的时候会通过 compareTo(Object obj)方法来比较所以在向其中添加不同类的对象的时候，也会引发 java.lang.ClassCastException.所以需要注意的是向 TreeSet 中添加的应该是同一个类的对象。
当把一个对象加入 TreeSet 集合中，TreeSet 调用该对象的 compareTo(Object obj)方法跟其对象比较大小，然后根据红黑树结构找到他的存储位置。如果两个对象根据 compareTo(Object obj)方法返回的值为 0，新对象将无法添加到 TreeSet 中。
对于 TreeSet 而言，他判断两个对象是否相等的唯一标准是：两个对象通过 compareTo(Object bj)比较返回的值是否为 0，如果为 0，就认为相等，如果不为 0，就认为不相等。
当需要把一个类的对象放到 TreeSet 中时，应该保证该类中的 compareTo(Object obj 返回 0 时 ，他的 equals()方法返回 true 。
当改变了 TreeSet 集合里添加的元素的属性后，这将导致它与其他对象的大小顺序发生了变化，TreeSet 不会再次调整他们的顺序，甚至导致 TreeSet 中保存的这两个对象通过 compareTo(Object obj)方法比较返回 0。（正常情况下，为 0 时，后一个不会被插入）
当 TreeSet 中的元素的属性（跟 compareTo 方法中的比较有关的字段）有被修改过后，那么被改的属性就不能对修改过的元素进行 <code>remove</code> 操作。只能对没有改变的元素进行 <ode>remove</code> 操作。值得注意的是：对没有被修改的的元素进行 <code>remove</code> 操作后，TreeSet 会对集合中的元素从新索引（不是从新排序），接下来就可以删除 TresSet 中的所元素了。

<h5 id="定制排序">定制排序</h5>

TreeSet 的自然排序是根据集合元素的大小，TreeSet 将他们以升序排序，如果要实现定制排序例如通过降序排序（当然自然排序通过反写 compareTo 方法的比较逻辑也可以实现），则可以通过omparator 接口的帮助。该接口里面包含一个 int commpare(T o1, T o2)方法，该方法用于比较 o1 和 o2 的大小。如果该方法返回 0，说明 o1 与 o2 相等，如果返回一个正整数表明 o1 大于 o2，如返回一个负数，表示 o1 小于 o2。
如果需要实现定制排序，则需要在创建 TreeSet 集合对象的时候，提供一个 Comparator 对象与 TreeSet 集合关联，由该 Comparator 对象负责集合元素的排序逻辑。由于 Comparator 时一个函式接口，因此可使用 lambda 表达式来替代 Commparator
同样不可以插入相同的元素，不然会由 ClassCastException 异常

<pre><code class="language-java highlight-chroma"><span c
ass="highlight-cl">package <span class="highlight-nn
>TreeSetTest.TreeSetTest;

<span class="high
ight-kn">import java.util.TreeSet<span class="
ighlight-o">;

<span class="high
ight-kd">class Persion <span class="highlight-
">{

```
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-cm">/**
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">     * 人的姓名
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">     */</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-kd">private</span> <span class="highlight-n">String</span>   <span class="highli
ht-n">name</span><span class="highlight-o">;</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-cm">/**
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">     * 人的年龄
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">     */</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-kd">private</span> <span class="highlight-n">Integer</span>  <span class="highl
ght-n">age</span><span class="highlight-o">;</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-cm">/*.....
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">     * 省略的getter和setter
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">     */</span><span class="highlight-err">.....</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-nd">@Override</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-kd">public</span> <span class="highlight-n">String</span> <span class="highligh
-nf">toString</span><span class="highlight-o">()</span> <span class="highlight-o">{</s
an>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-k">return</span> <span class="highlight-s">"Persion{"</span> <span class="high
ight-o">+</span>
</span></span><span class="highlight-line"><span class="highlight-cl">                <span c
ass="highlight-s">"name='"</span> <span class="highlight-o">+</span> <span class="hig
light-n">name</span> <span class="highlight-o">+</span> <span class="highlight-sc">'\'
</span> <span class="highlight-o">+</span>
</span></span><span class="highlight-line"><span class="highlight-cl">                <span c
ass="highlight-s">", age="</span> <span class="highlight-o">+</span> <span class="highl
ght-n">age</span> <span class="highlight-o">+</span>
</span></span><span class="highlight-line"><span class="highlight-cl">                <span c
ass="highlight-sc">'}'</span><span class="highlight-o">;</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-o">}</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-o">}</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-cm">/**
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
```

```
s="highlight-cm"> * @author yaoqiuhong
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm"> * @create 2020-02-27 14:15
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm"> * @description
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm"> */</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-kd">public</span> <span class="highlight-kd">class</span> <span class="highlight-n
">TresSetTest</span> <span class="highlight-o">{</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-kd">public</span> <span class="highlight-kd">static</span> <span class="highlig
t-kt">void</span> <span class="highlight-nf">main</span><span class="highlight-o">(</
pan><span class="highlight-n">String</span><span class="highlight-o">[]</span> <span c
ass="highlight-n">args</span><span class="highlight-o">)</span> <span class="highlight
o">{</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">// 创建对象人1
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-n">Persion</span> <span class="hig
light-n">persion1</span> <span class="highlight-o">=</span> <span class="highlight-k">
ew</span> <span class="highlight-n">Persion</span><span class="highlight-o">();</span

</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">persion1</span><span class="highlight-o">.</span><span class="highlight-n
">setName</span><span class="highlight-o">(</span><span class="highlight-s">"YQH"</
pan><span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">persion1</span><span class="highlight-o">.</span><span class="highlight-n
">setAge</span><span class="highlight-o">(</span><span class="highlight-mi">21</spa
><span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">// 创建对象人2
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-n">Persion</span> <span class="hig
light-n">persion2</span> <span class="highlight-o">=</span> <span class="highlight-k">
ew</span> <span class="highlight-n">Persion</span><span class="highlight-o">();</span

</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">persion2</span><span class="highlight-o">.</span><span class="highlight-n
">setName</span><span class="highlight-o">(</span><span class="highlight-s">"YYW"</
pan><span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">persion2</span><span class="highlight-o">.</span><span class="highlight-n
">setAge</span><span class="highlight-o">(</span><span class="highlight-mi">20</spa
><span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">//创建TreeSet对象
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-n">TreeSet</span> <span class="hig
light-n">treeSet</span> <span class="highlight-o">=</span> <span class="highlight-k">n
w</span> <span class="highlight-n">TreeSet</span><span class="highlight-o">((</span>
span class="highlight-n">o1</span><span class="highlight-o">,</span><span class="highl
```

ght-n">o2) ->{</
pan>
  <span clas
="highlight-n">Persion p1 <span class="highlig
t-o">= (Persion</sp
n>)o1<span class="
ighlight-o">;
  <span clas
="highlight-n">Persion p2 <span class="highlig
t-o">= (Persion</sp
n>)o2<span class="
ighlight-o">;
  <span clas
="highlight-k">return p1<span class="highlight
o">.getAge()</spa
> < p2<span clas
="highlight-o">.getAge<span class="highlight
o">() ? 1 <
pan class="highlight-o">:
  p1.<span class="highlight-
a">getAge() ></
pan> p2.<span clas
="highlight-na">getAge() <span class="highligh
-o">? -1 <
pan class="highlight-o">: 0 <span class="highl
ght-o">;
  <span class=
highlight-o">});

  <span class=
highlight-n">treeSet.<span class="highlight-na
>add(persion1</spa
>);
  <span class=
highlight-n">treeSet.<span class="highlight-na
>add(persion2</spa
>);

  <span class=
highlight-n">System.<span class="highlight-na
>out.println</span
("21"<span class="hi
hlight-o">+((Persion
treeSet.<sp
n class="highlight-na">first()).<span class="high
ight-na">getAge());

  <span class="h
ghlight-o">}

<span class="high
ight-o">}
</code></pre>

<h4 id="EnumSet">EnumSet</h4>

是一个专门为枚举设计的集合类，EnumSet 中的所有元素都必须时指定枚举类型的指定值。该举类型是在创建 EnumSet 时显示或隐式地指定。Enum 的集合元素也是有序的，EnumSet 以枚举在 Enum 类内定义顺序来决定集合元素的顺序
以位向量的形式存储，这种存储形式非常紧凑，高效，英雌 EnumSet 都西昂占用内存很小，而运行效率很好。尤其是进行批量操作（如调用 containsAll()和 retainAll 方法）时，如果参数也是 EnmSet 集合，则该批量操作的执行速度也非常快。
不允许加入 null 元素，如果试图插入 null 元素，EnumSet 将抛出 NullPointerException 异常如果是想判断 EnumSet 是否包含 null 元素或试图删除 null 元素都不会抛出异常，只是删除操作将回 false。
没有提供构造器来创建该类的实例，程序应该通过他提供的类方法来创建 EnumSet。Enum 提了如下常用方法来创建 EnumSet 对象。

EnumSet allOf(Class elementType)：创建一个包含指定枚举类型所有枚举值的 EnumSet 集合
EnumSet complementOf(EnumSet s)：创建一个其元素类型与指定 EnumSet 里元素类型相的 EnsumSet 集合，新 EnsumSet 集合包含原 EnumSet 集合所不包含的，此枚举剩下的枚举值（新的 EnumSet 集合和原集合的集合元素加起来就是该枚举类的所有枚举值）
EnumSet copyOf(Collection c)：使用一个普通集合来创建 EnumSet 集合
EnumSet copyOf(EnumSet s)：创建一个与指定 EnumSet 具有相同元素类型，相同集合元素的 EnumSet 集合
EnumSet noneOf(Class elementType)：创建一个元素类型为指定枚举类型的空 EnumSet。</l>
EnumSet of(E first, E... rest)：创建一个包含一个或多个枚举值的 EnumSet 集合，传入的多个举值必须属于同一个枚举类
EnumSet range(E from, E to)：创建一个包含从 from 枚举值到 to 枚举值范围内所有枚举值 EnmSet 集合。

<pre><code class="language-java highlight-chroma">package enumSetTest;

import java.util.Collection;
import java.util.EnumSet;
import java.util.HashSet;

/**
 * 季节枚举类
 * */
enum Season<span class="highlight-o

>{

    <span class="h
ghlight-n">SPRING(<span class="highlight-mi"
1),SUMMER</span
(2<span class="hig
light-o">),FALL(</sp
n>3),<span class="
ighlight-n">WINTER(<span class="highlight-mi
>4) ;

    <span class="h
ghlight-cm">/**
<span cla
s="highlight-cm">   * 季节的代码
<span cla
s="highlight-cm">   * */
    <span class="h
ghlight-kd">private int <span class="highlight-
">code;

    <span class="h
ghlight-n">Season(
nt code){
        <span class=
highlight-k">this.c
de = code
span class="highlight-o">;
    <span class="h
ghlight-o">}

    <span class="h
ghlight-kd">public int <span class="highlight-n
">getCode(){
        <span class=
highlight-k">return code<span class="highlight
o">;
    <span class="h
ghlight-o">}
<span class="high
ight-o">}

<span class="high
ight-cm">/**
<span cla
s="highlight-cm"> * @author yaoqiuhong
<span cla
s="highlight-cm"> * @create 2020-02-27 20:26
<span cla
s="highlight-cm"> * @description
<span cla
s="highlight-cm"> */
<span class="high

ight-kd">public class <span class="highlight-n
">EnumSetTest {
    <span class="h
ghlight-kd">public static <span class="highlig
t-kt">void main(</
pan>String[] <span c
ass="highlight-n">args) <span class="highlight
o">{

        <span class=
highlight-c1">// 通过allOf(Class elementType)创建一个Season类型全部枚举值的EnumSet
<span cla
s="highlight-c1">   EnumSet <span class="hi
hlight-n">enumSet1 = <span class="highlight-n
>EnumSet.allOf</s
an>(Season<span cl
ss="highlight-o">.class<span class="highlight-
">);
        <span class=
highlight-n">System.<span class="highlight-na
>out.println</span
("enumSet1:" <span c
ass="highlight-o">+ enumSet1<span class="hi
hlight-o">);

        <span class=
highlight-c1">// 通过noneOf(Class elementType)创建个Season类型的空EnumSet
<span cla
s="highlight-c1">   EnumSet <span class="hi
hlight-n">enumSet2 = <span class="highlight-n
>EnumSet.noneOf
/span>(Season<span
class="highlight-o">.class<span class="highlig
t-o">);
        <span class=
highlight-n">System.<span class="highlight-na
>out.println</span
("enumSet2:" <span c
ass="highlight-o">+ enumSet2<span class="hi
hlight-o">);

        <span class=
highlight-c1">// 添加Season的元素到EnumSet中
<span cla
s="highlight-c1">   enumSet2<span class="h
ghlight-o">.add(</
pan>Season.<span c
ass="highlight-na">WINTER);
        <span class=
highlight-n">enumSet2.<span class="highlight-
a">add(Season</sp
n>.SUMMER<span
lass="highlight-o">);
        <span class=

```
highlight-n">System</span><span class="highlight-o">.</span><span class="highlight-na
>out</span><span class="highlight-o">.</span><span class="highlight-na">println</span
<span class="highlight-o">(</span><span class="highlight-s">"enumSet2:"</span> <span c
ass="highlight-o">+</span> <span class="highlight-n">enumSet2</span><span class="hi
hlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">// 通过of(E first, E... rest)指定枚举值创建Season类型的EnumSet
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-n">EnumSet</span> <span class="hi
hlight-n">enumSet3</span> <span class="highlight-o">=</span> <span class="highlight-n
>EnumSet</span><span class="highlight-o">.</span><span class="highlight-na">of</spa
><span class="highlight-o">(</span><span class="highlight-n">Season</span><span clas
="highlight-o">.</span><span class="highlight-na">SUMMER</span> <span class="highli
ht-o">,</span> <span class="highlight-n">Season</span><span class="highlight-o">.</sp
n><span class="highlight-na">WINTER</span><span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">System</span><span class="highlight-o">.</span><span class="highlight-na
>out</span><span class="highlight-o">.</span><span class="highlight-na">println</span
<span class="highlight-o">(</span><span class="highlight-s">"enumSet3:"</span> <span c
ass="highlight-o">+</span> <span class="highlight-n">enumSet3</span><span class="hi
hlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">// 通过range(E from, E to)创建一个从SPRING到WINTER的EnumSet
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-n">EnumSet</span> <span class="hi
hlight-n">enumSet4</span> <span class="highlight-o">=</span> <span class="highlight-n
>EnumSet</span><span class="highlight-o">.</span><span class="highlight-na">range</
pan><span class="highlight-o">(</span><span class="highlight-n">Season</span><span c
ass="highlight-o">.</span><span class="highlight-na">SUMMER</span> <span class="hig
light-o">,</span> <span class="highlight-n">Season</span><span class="highlight-o">.</
pan><span class="highlight-na">WINTER</span><span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">System</span><span class="highlight-o">.</span><span class="highlight-na
>out</span><span class="highlight-o">.</span><span class="highlight-na">println</span
<span class="highlight-o">(</span><span class="highlight-s">"enumSet4:"</span> <span c
ass="highlight-o">+</span> <span class="highlight-n">enumSet4</span><span class="hi
hlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">// 通过complementOf(EnumSet  s)创建一个enumSet4的补集的EnumSet
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-n">EnumSet</span> <span class="hi
hlight-n">enumSet5</span> <span class="highlight-o">=</span> <span class="highlight-n
>EnumSet</span><span class="highlight-o">.</span><span class="highlight-na">comple
entOf</span><span class="highlight-o">(</span><span class="highlight-n">enumSet4</s
an><span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">System</span><span class="highlight-o">.</span><span class="highlight-na
>out</span><span class="highlight-o">.</span><span class="highlight-na">println</span
<span class="highlight-o">(</span><span class="highlight-s">"enumSet5:"</span> <span c
ass="highlight-o">+</span> <span class="highlight-n">enumSet5</span><span class="hi
hlight-o">);</span>
```

```
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">// 创建一个元素为Season值得HashSet
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-n">Collection</span> <span class="h
ghlight-n">c</span> <span class="highlight-o">=</span> <span class="highlight-k">new
/span> <span class="highlight-n">HashSet</span><span class="highlight-o">();</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">c</span><span class="highlight-o">.</span><span class="highlight-na">add
/span><span class="highlight-o">(</span><span class="highlight-n">Season</span><span
class="highlight-o">.</span><span class="highlight-na">SUMMER</span><span class="hi
hlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">c</span><span class="highlight-o">.</span><span class="highlight-na">add
/span><span class="highlight-o">(</span><span class="highlight-n">Season</span><span
class="highlight-o">.</span><span class="highlight-na">WINTER</span><span class="high
ight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class
"highlight-c1">// 通过copyOf(Collection c)来复制c中的所有元素来创建一个EnumSet(Collection
象c中的所有元素都必须是同一个枚举类的枚举值)
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-n">EnumSet</span> <span class="hi
hlight-n">enumSet6</span> <span class="highlight-o">=</span> <span class="highlight-n
>EnumSet</span><span class="highlight-o">.</span><span class="highlight-na">copyOf<
span><span class="highlight-o">(</span><span class="highlight-n">c</span><span class
"highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">System</span><span class="highlight-o">.</span><span class="highlight-na
>out</span><span class="highlight-o">.</span><span class="highlight-na">println</span
<span class="highlight-o">(</span><span class="highlight-s">"c:"</span><span class="hi
hlight-o">+</span><span class="highlight-n">c</span><span class="highlight-o">+</spa
><span class="highlight-s">" enumSet6:"</span><span class="highlight-o">+</span><sp
n class="highlight-n">enumSet6</span><span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">// 通过copyOf(EnumSet  s)复制一个EnumSet
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-n">EnumSet</span> <span class="hi
hlight-n">enumSet7</span> <span class="highlight-o">=</span> <span class="highlight-n
>EnumSet</span><span class="highlight-o">.</span><span class="highlight-na">copyOf<
span><span class="highlight-o">(</span><span class="highlight-n">enumSet6</span><s
an class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">System</span><span class="highlight-o">.</span><span class="highlight-na
>out</span><span class="highlight-o">.</span><span class="highlight-na">println</span
<span class="highlight-o">(</span><span class="highlight-s">"enumSet6:"</span><span c
ass="highlight-o">+</span><span class="highlight-n">enumSet6</span><span class="hig
light-o">+</span><span class="highlight-s">" enumSet7:"</span><span class="highlight-o
>+</span><span class="highlight-n">enumSet7</span><span class="highlight-o">);</spa
>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
```

ghlight-o">}

<span class="high
ight-o">}

</code></pre>
<h4 id="各个Set得得性能比较">各个 Set 得得性能比较</h4>

HashSet 得性能总比 TreeSet 好（特别是最仓用类得添加/查询元素等操作）因为 TreeSet 需要外得红黑树算法来维护集合得次序。只有当需要一个保持排序得 Set 时，才应该使用 TreeSet。
LinkedHashSet 对于普通的插入，删除操作，LinkedHashSet 比 HashSet 要略慢一点，这是由维护链表所带来的额外开销造成的，但由于有了链表，遍历 LinkedHashSet 更快。
EnumSet 是所有 Set 实现类中性能最好的。但他只能保持同一个枚举类的枚举值作为集合元素</l
>
Set 的三个实现类，HashSet，TreeSet，enumSet 都是线程不安全的。

<h3 id="List">List</h3>

List 也是 Collection 的子接口，可以使用所有 Collection 中的方法。而且由于 List 是有序集合英尺 List 集合增加了一些感觉索引来操作集合的方法。

void add(int index , Object element)：将元素 element 插入到 index 处
boolean addAll(int index , Collection c)：将集合 c 所包含的所有元素插入到 List 集合的 index 处
Object get(int index)：返回集合 index 索引处的元素
int indexOf(Object o)：返回对象 o 在 List 集合中第一次出现的位置
int lastIndex(Object o)：返回对象 o 在 List 集合中最后一次出现的位置
Object remove(int index)：删除并返回 Index 处的元素
Object set(int index , Object element)：将 index 索引处的元素替换成 element 对象，返回替换的旧元素。（只能改变，不能向后面添加）
List subList(int formIndex , int toIndex)：返回从索引 formIndex（包含） 到索引 toIndex（包含）的所有元素组成的子集合
void replaceAll(UnaryOperator operator)：根据 operator 指定的计算规则重新设置 List 集合所有元素
void sort(Comparator c)：根据 Comparator 参数对 List 集合的元素排序。

通过 equals()方法来判断两个元素是否相等。
<h4 id="replaceAll和sort的演示">replaceAll 和 sort 的演示</h4>
<pre><code class="language-java highlight-chroma"><span c
ass="highlight-cl">   package <span class="highlight-
n">List;

    <span class="h
ghlight-kn">import java.util.ArrayList<span cla
s="highlight-o">;
    <span class="h
ghlight-kn">import java.util.List<span class="h
ghlight-o">;

    <span class="h
ghlight-cm">/**
<span cla

```
s="highlight-cm">     * @author yaoqiuhong
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">     * @create 2020-02-27 23:19
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">     * @description
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">     */</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-kd">public</span> <span class="highlight-kd">class</span> <span class="highligh
-nc">ListTest</span> <span class="highlight-o">{</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-kd">public</span> <span class="highlight-kd">static</span> <span class="highli
ht-kt">void</span> <span class="highlight-nf">main</span><span class="highlight-o">(<
span><span class="highlight-n">String</span><span class="highlight-o">[]</span> <span
class="highlight-n">args</span><span class="highlight-o">)</span> <span class="highligh
-o">{</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-c1">// 创建List对象
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>            <span class="highlight-n">List</span> <span class="highl
ght-n">list</span> <span class="highlight-o">=</span> <span class="highlight-k">new</
pan> <span class="highlight-n">ArrayList</span><span class="highlight-o">();</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-cm">/*
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">             * 添加元素
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">             * */</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-n">list</span><span class="highlight-o">.</span><span class="highlight-na">
dd</span><span class="highlight-o">(</span><span class="highlight-s">"如果那两个字"</
pan><span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-n">list</span><span class="highlight-o">.</span><span class="highlight-na">
dd</span><span class="highlight-o">(</span><span class="highlight-s">"没有颤抖"</span
<span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-n">list</span><span class="highlight-o">.</span><span class="highlight-na">
dd</span><span class="highlight-o">(</span><span class="highlight-s">"我不会发现"</sp
n><span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-n">list</span><span class="highlight-o">.</span><span class="highlight-na">
dd</span><span class="highlight-o">(</span><span class="highlight-s">"我难受"</span>
span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-n">System</span><span class="highlight-o">.</span><span class="highlight-
a">out</span><span class="highlight-o">.</span><span class="highlight-na">println</sp
n><span class="highlight-o">(</span><span class="highlight-s">"list中元素插入顺序：  "</sp
n><span class="highlight-o">+</span><span class="highlight-n">list</span><span class=
highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-c1">// 使用Comparator的lambda表达式对List集合排序
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>            <span class="highlight-n">list</span><span class="highli
```

ht-o">.sort(((</spa
>o1, <span class="h
ghlight-n">o2) -&gt
 {
 <span c
ass="highlight-n">String s1 <span class="highli
ht-o">= (String</sp
n>)o1<span class="
ighlight-o">;
 <span c
ass="highlight-n">String s2 <span class="highli
ht-o">= (String</sp
n>)o2<span class="
ighlight-o">;
 <span c
ass="highlight-k">return s1<span class="highli
ht-o">.length()</s
an> - s2<span class
"highlight-o">.length<span class="highlight-o
>();
 <span clas
="highlight-o">}));
 <span clas
="highlight-n">System.<span class="highlight-
a">out.println</sp
n>("list中元素通过sort排序后
序："+list<
pan class="highlight-o">);

 <span clas
="highlight-c1">// 使用Lambda表达式控制使用每个字符串的长度作为新的元素集合
<span cla
s="highlight-c1"> list<span class="highli
ht-o">.replaceAll(<
span>ele -> <spa
n class="highlight-o">((String<span class="high
ight-o">)ele).</spa
>length());
 <span clas
="highlight-n">System.<span class="highlight-
a">out.println</sp
n>("list中元素通过replaceAll
换后元素为："+list</
pan>);
 <span class=
highlight-o">}
 <span class="h
ghlight-o">}
</code></pre>
<h4 id="-----ListIterator"> ListIterator</h4>
与 Set 只提供一个 Iterator 不同，List 还额外提供一个 ListIterator 方法，该方法返回一个 ListIt
rator 对象，ListIterator 接口继承了 Iterator 接口，提供了专门操作 List 的方法。ListIterator 接口在
Iterator 接口上增加了一下方法。

boolean haspervious()：返回该迭代器关联的集合是否还有上一个元素
Object previous()：返回该迭代器的上一个元素
void add(Object o)：在指定位置插入一个元素

ListIterator 相比于 Iterator 增加了向前迭代的功能（Iterator 只能向后迭代）。而且 ListIterator 还能通过 add()方法向 List 中添加元素（Iterator 只能删除元素）
<h4 id="ArrayList-and-Vector">ArrayList and Vector</h4>

都是基于数组实现的 List 类，所以 ArrayList 和 Vector 类都封装了一个动态的，允许分配的 Object[]数组。
使用 initialCapacity 参数来设计该数组的长度，当向 ArrayList or Vector 添加的元素超过了数组的长度的时候，ArrayList 或 Vector 的 initialCapacity 自动增加。
对于普通场景，我们无需关心 initialCapacity 的大小。但如果向 ArrayList r Vector 添加大量元素的时候，可以使用 ensureCapacity(int minCapacity)方法</stron>一次性地增加 initialCapacity。这就可以减少分配的权重，从而提高性能！</l>
如果创建空的 ArrayList or Vector 时没有指定 initialCapacity 参数，则 Oject[]数组的长度默认为 10
void ensureCapacity(int minCapacity)：将 ArrayList or Vector 集合时 Object[]数组长度增加大于或等于 minCapacity 值。
void trimToSize()：调整 ArrayList or Vector 集合的 Object[]数组的个数当前元素的个数。该方法可以减少集合占用的存储空间。
ArrayList or Vector 的显著区别就是：ArrayList 时线程不安全的，Vector 则是线程全的。这里需要注意即使要保证 ArrayList 时线程安全的，也不推荐使用 Vector。这时候以使用 Collections 工具类，它可以将一个 ArrayList 变成线程安全的。

<h5 id="Stack">Stack</h5>

Vector 还提供一个 Stack 子类，它用于模拟"栈"这种数据结构。与其他集合一样，进栈出栈的素都是 Object,因此从栈中取出元素后必须进行类型转换，除非你只是使用 Object 具有的操作。提的方法

Object peek()：返回"栈"的第一个元素，但并不将该元素"pop"出栈
Object pop()：返回"栈"的第一个元素，但并将该元素"pop"出栈
void push(Object item)：将一个元素"push"进栈，最后一个进"栈"的元素总是位于"栈"顶
跟 Vectors 是一个较老的类，它同样是线程安全的，性能较差的。因此应该少用 Stack,如果需要"栈"这种数据结构，则可以考虑使用 ArrayDeque

<h4 id="Arrays-ArrayList">Arrays.ArrayList</h4>

有一个操作数组的工具类：Arrays，该工具类里提供了 asList(Object... a)方法，该方法可以把个数组或指定个数的对象转换成一个 List 集合，这个 List 集合既不是 ArrayList 实现类也不是 Vector 的实现类，而是 Arrays 的内部类 ArrayList 的实例。

Arrays.ArrayList 是一个固定长度的 List 集合，程序只能遍历访问该集合的元素，不可台南佳，除该集合里的元素。
<pre><code class="language-java highlight-chroma">package arrays;

```
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-kn">import</span> <span class="highlight-nn">java.util.Arrays</span><span class="h
ghlight-o">;</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-kn">import</span> <span class="highlight-nn">java.util.List</span><span class="highl
ght-o">;</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-cm">/**
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm"> * @author yaoqiuhong
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm"> * @create 2020-02-28 00:36
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm"> * @description
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm"> */</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-kd">public</span> <span class="highlight-kd">class</span> <span class="highlight-n
">ArraysDemo</span> <span class="highlight-o">{</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-kd">public</span> <span class="highlight-kd">static</span> <span class="highlig
t-kt">void</span> <span class="highlight-nf">main</span><span class="highlight-o">(</
pan><span class="highlight-n">String</span><span class="highlight-o">[]</span> <span c
ass="highlight-n">args</span><span class="highlight-o">)</span> <span class="highlight
o">{</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">// 利用Arrays.asList获取List
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-n">List</span> <span class="highlig
t-n">list</span> <span class="highlight-o">=</span> <span class="highlight-n">Arrays</
pan><span class="highlight-o">.</span><span class="highlight-na">asList</span><span c
ass="highlight-o">(</span><span class="highlight-s">"如果那两个字"</span><span class="
ighlight-o">,</span><span class="highlight-s">"没有颤抖"</span><span class="highlight-o
>,</span><span class="highlight-s">"我不会发现我难受"</span><span class="highlight-o">)
</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-n">System</span><span class="highlight-o">.</span><span class="highlight-na
>out</span><span class="highlight-o">.</span><span class="highlight-na">println</span
<span class="highlight-o">(</span><span class="highlight-n">list</span><span class="hi
hlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">// 遍历元素
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-n">list</span><span class="highligh
-o">.</span><span class="highlight-na">forEach</span><span class="highlight-o">(</spa
><span class="highlight-n">System</span><span class="highlight-o">.</span><span clas
="highlight-na">out</span><span class="highlight-o">::</span><span class="highlight-n"
println</span><span class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">// 试图增加，删除元素都会引发UnsupportedOperationException异常
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-n">list</span><span class="highligh
```

-o">.add(
span class="highlight-s">"增加元素会抛出UnsupportedOperationException异常"<spa
class="highlight-o">);
  <span class=
highlight-n">list.r
move("移除元素会抛
UnsupportedOperationException异常");
  <span class="h
ghlight-o">}
<span class="high
ight-o">}
</code></pre>

<h3 id="Queue">Queue</h3>

用于模拟队列这种数据结构（元素先进先出）
新元素插入（offer）到队列的尾部，访问元素（poll）操作会返回队列头部的元素。筒仓，队列允许所及访问队列中的元素
定义了如下方法：

void add(Object e)：将指定的元素加入此队列的尾部
Objec element()：获取队列头部的元素，但是不删除该元素
boolean offer(Objec e)：将指定元素加入此队列的尾部。当使用有容量限制的队列时，此方法常比 add(Object e)方法更好。
Objec peek()：获取队列头部的元素，但是不删除该元素。如果队列为空，返回 null
Objec poll()：获取队列头部的元素，并删除该元素，如果此队列为空，则返回 null
Objec remove()：获取队列头部的元素，并删除该元素
Queue 有一个 PriorityQueue 实现类。除此之外，Queue 还有一个 Deque 接口，Deque 代表个“双端队列”，双端队列可以同时从两端来添加，删除元素，因此 Deque 的实现类即可以当成队使用，也可当成栈使用。java 为 Deque 提供了 ArrayDeque 和 LinkedList 两个实现类

<h4 id="PriorityQueue">PriorityQueue</h4>

不按元素的插入顺序保存元素，二十按元素的大小从新排序，因此通过 peek()方法或者 poll 方取出队列中的元素时，并不是去除最先进入的元素而是取出最小的元素。
不允许插入 null 元素
两种排序

自然排序：元素实现 Comparable 接口
定制排序：创建 PriorityQueue 队列时，闯入一个 Comparaor 对象。
PriorityQueue 对元素的要求与 TreeSet 对元素的要求基本一致。

<h4 id="Deque接口与ArrayDeque实现类">Deque 接口与 ArrayDeque 实现类</h4>

<p>Deque 接口是 Queue 接口的字接口，Deque 代表一个“双端队列”，双端队列可以同时从两来添加/删除元素。提供如下方法允许从两端操作队列：

ttps://b3logfile.com/file/2020/02/1582823575256-8e51ab7f.png?imageView2/2/interlace/1/format/jpg">

使用栈这种数据结构的时候推荐使 ArrayDeque

<h4 id="LinkedList">LinkedList</h4>
LinkedList 是 List 接口的实现类，可以根据索引来随机访问集合中的元素。与此同时，LinkedLis 海慧寺心安了 Deque 接口，可以被当成双端队列来使用。
<img src="https://ld246.com/images/img-loading.svg" alt="1582823931076.png" data-sc="https://b3logfile.com/file/2020/02/1582823931076-2ffcf3b3.png?imageView2/2/interlace1/format/jpg">
<h4 id="各种线性表的性能分析">各种线性表的性能分析</h4>
<img src="https://ld246.com/images/img-loading.svg" alt="1582824030113.png" data-sc="https://b3logfile.com/file/2020/02/1582824030113-9a9c4304.png?imageView2/2/interlac/1/format/jpg">
<h3 id="Map">Map</h3>

Map 用于保存具有映射关系的数据，Map 里面保存着两组值，以组值用于保存 Map 里的 Key 另外一组值用于保存 Map 里的 Value，key 和 value 都可以是任何类型的数据。Map 的 Key 不允重复，即同一个 Map 的两个 key 通过 equals 方法总是返回 false。
key 和 value 是一对一的关系，即通过指定的 key，总能找到唯一的，确定的 value。如果把 Ma 的两组值拆开来看:

Map 里面的所有 Key 放在一起来看，他们就组成了一个 Set 集合（所有 Key 没有顺序，key 与 ey 之间不能重复）实际上 map 里面确实包含了一个 KeySet 方法，用于返回 Map 里面所有 Key 组的 Set 集合
Set 与 Map 的关系十分密切，虽然 Map 中放的元素是 Key-value 对，Set 集合中存放的元素单个对象，但如果把 Key-value 对中的 value 当成 key 的附庸：key 在哪里，value 就在哪里。这就可以对待 Set 一样来对待 Map 了。事实上 Map 提供了一个 Entry 内部类来封装 Key-Value 对，计算 Entry 存储时只考虑 Entry 封装的 Key。从 Java 的源码来看，java 先是实现了 Map，然后通包装一个所有 Value 都为 null 的 Map 就实现了 Set 集合。
Map 里面的所有 Value 放在一起来看，他们又非常类似一个 List：元素与元素之间可以重复，个元素可以根据索引来查找，知识 Map 中的索引不再使用整数值，而是使用另一个对象作为索引。果要从 List 集合去除元素，则需要提供该元素的数字索引；如果需要重 Map 里面取出元素，则需要供该 Value 的 Key 索引。因此，Map 有时候也被称为字典，或者关联数组。

Map 提供了如下接口

void clear()：删除该 Map 对象中的所有 Key-value 对
boolean containsKey(Object key)：查询 Map 中是否包含指定的 Key，如果包含返回 True</l >
boolean containsValue(Object value)：查询 Map 中是否包含一个或者多个的 value，如果含返回 True
Set entrySet()：返回 Map 中包含的 Key-value 对所组成的 Set 集合，每个集合元素都是 Map. ntry（Entry 是 Map 的内部类）对象
Object get(Object key)：返回指定的 key 所对应的 value，如果此 Map 中不包含该 Key，则回 null
boolean isEmpty()：查询该 Map 是否为空（即不包含任何的 Key-value 对）如果为空则返回 tre
Set ketSet()：返回该 Map 中所有 key 组成的 Set 集合
Object put(Object key , Object value)：添加一个 Key-value 对，如果 dangqianMap 中已

一个与该 Mao 相等的 Key-value 对，则新的 Key-value 对会覆盖原来的 Key-value 对。
void putAll(Map m)：将指定的 Map 中的 Key-value 对复制到本 Map 中
Object remove(Object key)：删除指定的 key-value 对，返回被删除 key 所关联的 value，果该 key 不存在，则返回 null
boolean remove(Object key , Object value)：java8 新增的方法，删除指定 key，value 所应的 key-value 对。如果删除成功，返回会 true，否者返回 false
int size()：返回该 Map 中的 key-value 的个数
Collection values()：返回该 Map 里所有 value 组成的 Collection

<h4 id="Entry">Entry</h4>

是 Map 集合的一个内部类，该类封装了一个 key-value 对。Entry 包含如下三个方法

Object getKey()：返回该 Entry 中包含的 key 值
Object getValue()：返回该 Entry 里包含的 Value 值
Object setValue(V value)：设置该 Entry 里包含的 Value 值，并返回新设置的 value 值。

<h4 id="HashMap-and-Hashtable">HashMap and Hashtable</h4>

Hashtable 线程安全，HashMap 线程不安全
Hashtable 不允许使用 null 作为 key 和 value。Hashtable 允许使用 null 作为 key 和 value（于 Map 的 key 不能重复，所以 HashMap 里最多只有一个 key-value 对的 Key 为 null。）
为什么是 Hashtable 而不是 HashTable。ashtable 是一个很老的类，它的命名甚至没有遵守 Java 的命名规范：每个单词的首字母都应该大写（后来大量的程序中使用了 Hashtable，所以这个类名就没修改为 HashTable）
为了成功的在 HashMap,Hashtable 中存储，获取对象，用作 Key 的对象必须实现 HahCode 方法和 equals 方法。判断两个 Key 是否相等的标准 是：通过 equals()方法比较返回 true，两个 key 的 hashCode 值也相等。判断两个 value 是否相等的标准：通过 equals 方法比较返回 true。</li

与 HashSet 一样，如果使用可变对象作为 HashMap 的 key，并且修改了作为 Key 的可变对象那么就会出现与 HashSet 一样的情形。

<h4 id="LinkedHashMap">LinkedHashMap</h4>

与 HashSet 有一个 LinkedHashSet 子类类似，HashMap 也有一个 LinkedHashMap 子类：LinedHashMap 也是使用双向链表来维护 key-value 对的顺序（只需要考虑 key 的次序）.
使用 LinkedHashMap 可以避免对 HahMap，Hashtable 里的 key-value 排序（只需要插入的候保持顺序即可），同时又可避免使用 TreeMap 所增加的成本。
性能略低于 HashMap

<h4 id="Properties">Properties</h4>

Properties 是 Hashtable 的子类，该对象在处理属性文件时特别方便。可以把 Map 对象和属文件关联起来。从而把 Map 对象中的 key-value 对写入属性中，也可以把属性文件中的“属性名 = 性值”加载到 Map 对象中。由于属性文件中的属性名和属性值都是字符串。所以 Properties 的 key-alue 都是字符串类型。改类提供了如下方法：

String getProperty(String key)：获取 Property 中指定属性名对应的属性值，类似与 Map 的 g

t(Object key)方法
String getProperty(String key , String defaultvalue)：与上一个方法类似。并且如果不存在指的 key 时，则方法指定默认值
Object setProperty(String key , String value)：设置属性值，类似与 Hashtable 的 put()方法
void load(InputStream instream)：从属性文件中加载 key-value 对，把加载到的 key-value 追加到 Properties 里（Properties 是 Hashtable 的子类，不保证 key-value 之间的次序）
void store(OutputStream out ,String comments)：将 Properties 中的 key-value 对输出到定的属性文件中。

<h4 id="TreeMap">TreeMap</h4>

是 SortedMap 接口的实现类。TreeMap 就是一个红黑树结构，每个 key-value 对即作为红黑的一个节点。TreeMap 存储 key-value 对，树妖对 key 节点进行排序。排序方法有两种。自然排序定制排序（与 TreeSet 相似）

<h4 id="WeakHashMap">WeakHashMap</h4>
与 HashMap 类似，与 HashMap 的区别在于 HashMap 的 key 保留了实际对象的强引用，这味着，只要 HashMap 对象不被销毁，该 HashMap 的所有 key 所引用的对象就不会被垃回收，HashMap 也不会自动删除这些 key 对应的 key-value 对：但 WeakHashMap 的ey 只保留了实际对象的弱引用，这意味着 WeakHashMap 对象的 key 所引用的对象没有其他强引用变量所引用，则这些 key 所引用的对象可能被垃圾回收，WeakHashMap 也能自动删除这些 key 对应的 key-value 对。
<h4 id="IdentityHashMap">IdentityHashMap</h4>
与 HashMap 类似，与 HashMap 的区别在于，它在处理两个 key 相等时比较特殊，在 Identit HashMap 中，当且仅当两个 key 严格相等（key1 == key2）时，IdentityHashMap 才会认为两个ey 相等。
<h4 id="EnumMap">EnumMap</h4>

EnumMap 内部以数组形式保存
EnumMap 根据 key 的自然顺序（即枚举值中的定义顺序）来维护 key-value 对的顺序。
不允许使用 null 作为 key，但允许 null 作为 value。如果试图使用 null 插入 key，那么就会抛出 NPE。如果只是查询是否包含 null 或者只是删除值为 null 的 key，那么都不会抛出 NPE
创建一个 EnumMap 必须指定一个枚举类。<code>EnumMap enumMap = new EnumMap(numClassName.class)</code>

<h4 id="各Map的性能分析">各 Map 的性能分析</h4>

HashMap 通常比 Hashtable 要快（Hashtable 是一个古老并且线程安全的集合）
TreeMap 通常比 HashMap，Hashtable 要慢（尤其在插入，删除时更慢），应为 TreeMap 用红黑树管理 key-value 对（红黑树的每个节点都是一个 key-value 对）
使用 TreeMap 的好处是可以让 Key-value 对总是处于有序状态，无需专门排序。当 TreeMap 填充后，就可以调用 keySet，取得由 key 组成的 Set，然后使用 toArray()方法生成 Key 的数组。接来使用 Arrays 的 binarySearch()方法在已排序的数组中快速的查询对象
LinkedHashMap 比 HashMap 要慢一点，因为他需要维护链表来保持 Map 中 Key-value 时的加顺序
一般建议使用 HashMap，因为 HashMap 就是为快速查询设计的（HashMap 地城其实也是采数组来存储 key-value 对）

<h4 id="HashMap与HashSet性能比较">HashMap 与 HashSet 性能比较</h4>

<h3 id="Collections">Collections</h3>
<h4 id="排序">排序</h4>

void reverse(List list)：反转指定 List 集合中元素的顺序
void shuffle(List list)：对 List 集合进行随机排序
void sort(List list)：根据元素的自然顺序对 List 集合的元素按升序排序
void sort(List list , Comparator c)：根据指定的 Comparator 产生的顺序对 List 集合元素进行序
void swap(List list, int i, int j)：将指定 List 集合中的 i 处元素和 j 处元素进行交换
void rotate(List list, int distance)：当 distance 为正数时，将 list 集合的后 distance 各元素“体”移到前面。当 distance 为负数时，将 list 集合的前 distance 各元素整体移到后面

<h4 id="查找替换">查找替换</h4>

int binarySerach(List list , Object key)：使用二分法搜索指定的 List 集合，以获得指定对象在 List 集合中的索引。如果要使该方法可以正常工作，则必须保证 List 中的元素已经处于有序状态。
Object max(Collection coll)：根据元素的自然顺序，返回给定集合中的最大元素
Object max(Collection coll , Comparator comp)：根据 Comparator 指定顺序，返回给定集中的最大元素
Object min(Collection coll)：根据元素的自然顺序，返回给定集合中的最小元素
Object min(Collection coll , Comparator comp)：根据 Comparator 指定顺序，返回给定集中的最小元素
void fill(List list , Object obj)：使用指定元素 obj 替换指定 List 集合中的所有元素
int frequency(Collection c , Object o)：返回指定集合中指定元素出现次数
int indexOfSublist(List source , List target)：返回子 List 对象在父 List 对象中第一次出现的位索引：如果父 List 中没有出现这样的子 List，则返回-1
int lastIndexOfSublist(List source , List target)返回子 List 对象在父 List 对象中最后一次出现位置索引：如果父 List 中没有出现这样的子 List，则返回-1
boolean replaceAll(List list ,Object oldVal ,Object newVal)：使用一个新值 newVal 替换 LIst 对象的所有旧值 oldVal

<h4 id="同步控制">同步控制</h4>

Collection 类中提供了多个 <code>synchronizedXxx()</code> 方法，该方法可以将只当集包装成线程同步的集合，从而解决并发访问集合时的线程安全问题
java 中常用的集合的实现类 HashSet，TreeSet，ArrayList，ArrayDeque，LInkedList，Hashap 和 TreeMap 都是线程不安全的。
Collection 可以通过 <code>Collection c = Collections.synchronizedCollection(new Collecton实现类) </code> 创建对应的线程安全版本
List 可以通过 <code>List list = Collections.synchronizedList(new List实现类)</code> 创对应的线程安全版本
Map 可以通过 <code>Map map = Collections.synchronizedMap(new Map实现类)</code> 创建对应的线程安全版本
Set 可以通过 <code>Set set = Collections.synchronizedSet(new Set实现类)</code> 创建应的线程安全版本

<h4 id="设置不可变集合">设置不可变集合</h4>

emptyXXX()：返回一个空的，不可变的集合对象，此处的集合既可以是 List，也可以是 SortedSt。Set，还可以时 Map，SortedMap

singletonXXX()：返回一个只包含指定对象（只有一个或一项元素）的，不可变的集合对象，此的集合即可以时 List，也还可以是 Map

unmodifiableXxx()：返回指定集合对象的不可变试图，此处的集合计可以是 List，也可以是 Se。SortedSet，还可以是 Map。SortedMap 等。

上面三个方法的参数都是原有的集合类型，返回值是该集合的“只读”版本。如果对其修改则会现 UnsupportedOperationException 异常。

