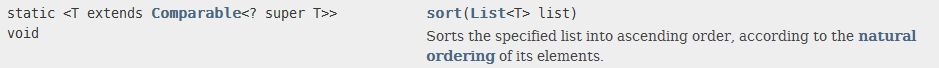
Java中的集合工具类Collection

1. **类Collections 存在于java.util 包中，直接继承于java.lang.Object，**是集合框架的**工具类**，其中的**所有方法**都是**静态的**。
2. 学习此工具类主要是掌握其中的主要方法，下面就介绍其静态方法。
3. 排序方法：sort方法：

（1） 利用**自然排序法**：（元素实现Comparable接口的compareTo方法）



（2） 利用**比较器**排序：（实现Comparator接口的compare方法）



**<T extends Comparable<? super T>> 作用：用来对T类型进行限制，用来说明T类型必须实现Comparable接口，且接口接收的必须为T类型或者其父类型。**

第一个**sort方法**的原理：

private static **<T extends Comparable<? super T>>** void mySort(List<T> list) {

int size = list.size();

for(int i = 1;i < size;i++) {

for(int j = 0;j < size -i;j++) {

if(list.get(j).compareTo(list.get(j+1)) > 0) {

swap(list,j,j+1);

}

}

}

}

private static <T> void swap(List<T> list, int j, int i) {

T temp = list.get(j);

list.set(j,list.get(j+1));

list.set(j+1,temp);

}

第一个**sort方法**的原理：

private static <T> void mySort(List<T> list,**Comparator<? super T> comparator**) {

int size = list.size();

for(int i = 1;i < size;i++) {

for(int j = 0;j < size -i;j++) {

if( comparator.compare(list.get(j),list.get(j+1)) > 0) {

swap(list,j,j+1);

}

}

}

}

private static <T> void swap(List<T> list, int j, int i) {

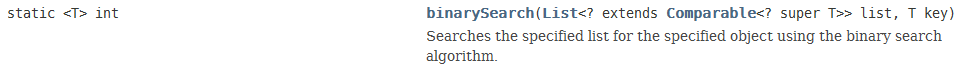
T temp = list.get(j);

list.set(j,list.get(j+1));

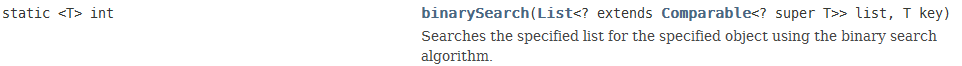
list.set(j+1,temp);

}

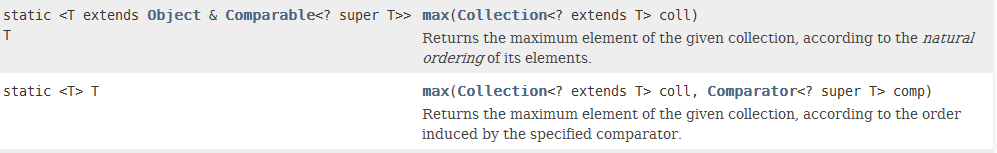
1. **查找方法**：binarySearch（）方法：利用二分查找方法查找元素，如果存在，则返回索引值，如果不存在则返回值为**此元素应当插入的点加1并取相反数**。目的是为了防止出现返回值为0时，可能为找到了元素在0处，或没找到应该在0出插入。
2. **自然排序**：



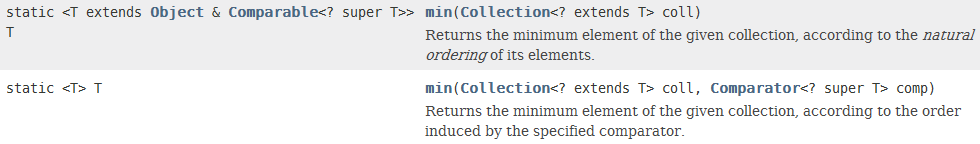
1. **比较器**：



1. **寻找最值**：
2. **max：**



1. **min：**



例子：

System.***out***.println( Collections.*min*(list) );

System.***out***.println( Collections.*max*(list) );

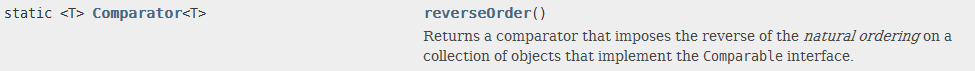
System.out.println(Collections.min(list,new CompatatorByLength()));

System.out.println(Collections.max(list,new CompatatorByLength()));

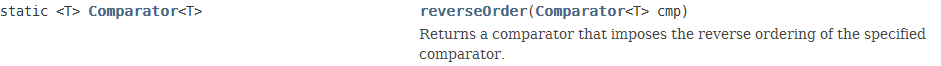
1. **倒置方法**：**reverse方法**：把集合倒置。



1. **reverseOrder方法**：倒置比较器，并**返回一个倒置的比较器**。即原来从大到小，返回的比较器从小到大。**逆序**
2. 把**自然排序的倒置方法**封装成一个**比较器**，并返回此比较器。



1. 把**指定比较器的倒置方法**封装成一个**比较器**，并返回此比较器。



1. **replaceAll方法**：替换方法。

原理：首先根据元素找到索引，然后再利用set进行替换。



1. **fill方法**：把集合中所有的元素**都**设置**某一个元素**。



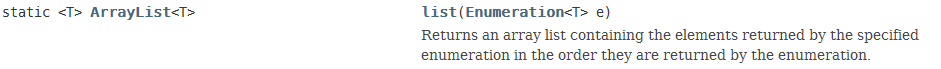
1. **shuffle方法**：相当于洗牌一样**，随机调换元素的顺序**。

shuffle 英 ['ʃʌf(ə)l] 美 ['ʃʌfl] v. 洗牌；推诿，推卸；搅乱





1. **Enumeration和Collection之间的转换**：
2. **Enumeration ==》 ArrayList**

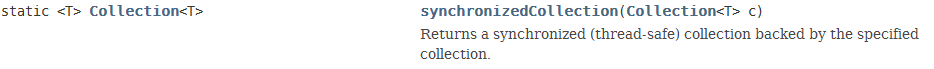


1. **Collection ===》Enumeration**



1. **给非同步的集合加锁**，即把一个非同步的集合变成一个同步的集合。

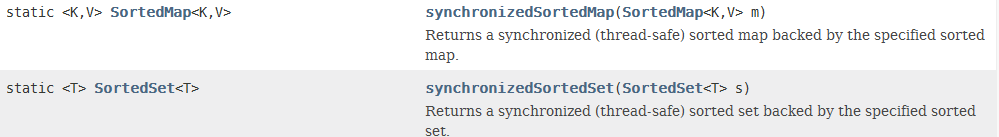
**除了Vector之外，大多数集合都是不同步的，为了防止集合的并发访问，需要给集合加锁，方法就是利用Collections中的synchronizedXxx方法。可以将Collection、List、Set、Map等集合加锁。**

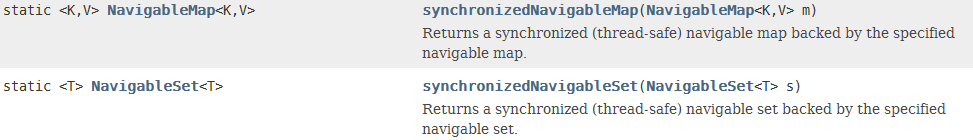












1. 代码仅供查看：

public class MyCollections {

public static List synList(List list) {

return new MyList(list); }

private class MyList implements List{

private List list;

private static final Object lock = new Object();

MyList(List list){

this.list = list;}

public boolean add(Object o) {

synchronized (lock) {

return list.add(o);

}

}

public boolean remove(Object o) {

synchronized (lock) {

return list.remove(o);

}}}}