本章讲述的操作各种数据结构类型的容器

集合和数组的区别：

数组：长度不可变，存储数据类型单一

集合：长度可变，存储数据类型多样化（集合中存放的数据均为相应数据的地址）

都具有基本功能：增删改查

Collection

|- List 有序，可重复 凡是可以操作角标的方法都是该体系特有的方法。

|-arrayList 底层数据结构：数组 查询快增删慢 线程不同步

|-linklist 底层数据结构：链表 查询稍慢，增删快 线程不同步

|-vector 底层数据结构：数组。 线程同步。被ArrayList替代了。因为效率低。

|- Set 无序，不重复 底层使用的map集合实现

|-HashSet 底层数据结构：哈希表

判断元素是否相同：HashCode值相同，才会判断equals是否为true。

|-TreeSet 底层数据结构：二叉树 集合内所有元素必须具有比较性 部分类自带比较性 单比较方式不需要，此时需要自定义比较器

判断元素是否相同：compareTo方法return 0.

Map集合，键值对存在

|-- HashTable 线程同步

|-- HashMap ， 线程非同步

该类是hashtable的轻量级实现，并且允许null键值对存在。hashTable不允许存在空键值对

枚举:是Vector特有的取出方式:

while(en.hasMoreElements()){System.out.println(en.nextElement());}

迭代器：一种访问集合的工具

|  |  |
| --- | --- |
| boolean | [**hasNext**](mk:@MSITStore:G:\Software\JCreater\JavaDocs\JDK1.6%20API帮助文档.CHM::/java/util/Iterator.html#hasNext())()            如果仍有元素可以迭代，则返回 true。 |
| [E](mk:@MSITStore:G:\Software\JCreater\JavaDocs\JDK1.6%20API帮助文档.CHM::/java/util/Iterator.html) | [**next**](mk:@MSITStore:G:\Software\JCreater\JavaDocs\JDK1.6%20API帮助文档.CHM::/java/util/Iterator.html#next())()            返回迭代的下一个元素。 |
| void | [**remove**](mk:@MSITStore:G:\Software\JCreater\JavaDocs\JDK1.6%20API帮助文档.CHM::/java/util/Iterator.html#remove())()            从迭代器指向的 collection 中移除迭代器返回的最后一个元素（可选操作）。 |

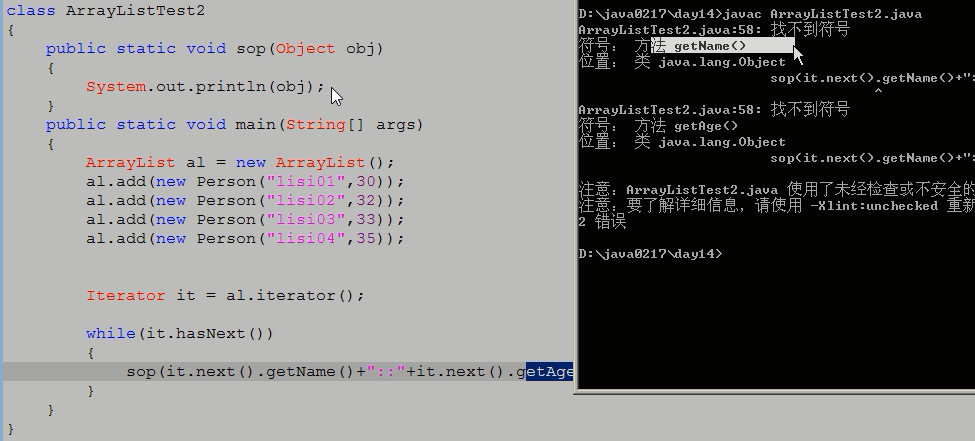
可通过集合方法Iterator()获取迭代器

迭代器（iterator）有时又称游标（cursor），是程式设计的软件设计模式，可在容器物件（container，例如链表或阵列）上遍访的接口，设计人员无需关心容器物件的内容。它可以把抽象容器和通用算法有机的统一起来。

java中：迭代器只有三个方法：判断是否存在下一个，取出当前，移除当前（功能很局限）

List集合特有的迭代器：ListIterator（功能更加强大） 是Iterator的子接口 具体功能查文档

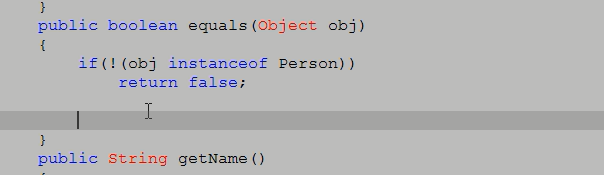
不可以并发使用集合方法和迭代器方法访问数据，会报告（并发修改异常）ConcurrrentModificationException异常



为何报该错？

子类对象可以赋值给父类对象（反过来呢）？ 父类对象可以强转为子类对象（那反过来呢）？

Contains 和 remove 都是通过调用底层的 equals方法实现的



泛型：

在使用java提供的对象时，什么时候写泛型呢？

通常在集合框架中很常见，

只要见到<>就要定义泛型。

其实<> 就是用来接收类型的。

当使用集合时，将集合中要存储的数据类型作为参数传递到<>中即可。

类： class myclass<Ele> {

Public Ele fun1(Ele e) //此处 不能操作 实参的（对象的）方法

{

e.funtion(); //错误 不能这样

Return e;

}

Public void main(string args[]){

Myclass<String> mc=new myclass<String>();

//这样声明的类 里面的方法就等如下：所以如果实参形参匹配 实参只能传入String类型的

Public string fun1(string e){ return e}

}

}

好处

1.将运行时期出现问题ClassCastException，转移到了编译时期。，

方便于程序员解决问题。让运行时问题减少，安全。，

2，避免了强制转换麻烦。（提升Object类型）

方法：Public<Ele> Ele fun1(Ele e){ return e;}

问题1：虽然使用泛型 可以使用很方便的方法或者类定义 就能操作多种数据类型 但是操作方式却是单一的 如果特定的数据类型 可能单一的操作方式操作不了该数据类型

解答：将泛型定义在方法上 就可以操作多种数据类型 但是操作具有局限性（无法操作特有对象的特性方法）

问题2：将泛型定义在类上和方法上的区别是什么？

区别：类中的方法的泛型定义跟随的是类的 而方法的泛型定义是跟随自己的

TreeSet 集合中的元素遵从自然排序，所以集合中的元素要么实现comparato接口

要么自定义一个comparator类 传入TreeSet 并且覆盖compare（T1,T2）方法

使用Object 接受任意对象时，如果需要调用该任意对象的方式时，需要使用强转 以提升Object的类型

思考一下 泛型的其他价值？

实现对象的比较：两种方法：

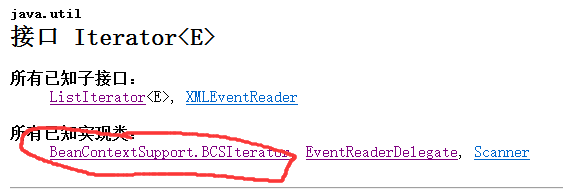
自定义比较器：comparator：覆盖compare方法，虽然接口中有equals方法，但是已经在父类中实现了 所以不必在比较器中覆写

实现comparable接口：并覆盖compareTo方法

之前的知识点：接口

在传统版本上，接口中的所有方法必须是非静态的，且是abstract的，且是public的。普通方法可以不写修饰符，也会默认为public和abstract，写上修饰符也正确。

疑问： 迭代器也是接口 为啥我们在使用迭代器的时候 没有实现其中的方法就可以使用迭代器了？

解答：红圈内的已经实现了的类中，包含了hasNext next remove 方法 所以我们使用迭代器的时候 不用实现这些方法

Collection是接口 该接口下继承了set list 等集合

Collections 是工具类，用于操作集合

Map集合：该集合存储键值对。一对一对往里存。而且要保证键的唯一性。

1，添加。

put(K key, V value)

putAll(Map<? extends K,? extends V> m)

2，删除。

clear()

remove(Object key)

3，判断。

containsValue(Object value)

containsKey(Object key)

isEmpty()

4，获取。

get(Object key)

size()

values()

entrySet()

keySet()

Map

|--Hashtable:底层是哈希表数据结构，不可以存入null键null值。该集合是线程同步的。jdk1.0.效率低。

|--HashMap：底层是哈希表数据结构，允许使用 null 值和 null 键，该集合是不同步的。将hashtable替代，jdk1.2.效率高。

|--TreeMap：底层是二叉树数据结构。线程不同步。可以用于给map集合中的键进行排序。

和Set很像。

其实大家，Set底层就是使用了Map集合。