

非贷款，0元入学，不1万就业不给1分钱学费，我们已千四年了！

笔记总链接：<http://bbs.itheima.com/thread-200600-1-1.html>

6. 集合

6.2 集合类

6.2.3 Vector、ArrayList、LinkedList

List:

[-Vector:内部是数组数据结构，是同步的。增删，查询都很慢。

[-ArrayList:内部是数组数据结构，是不同步的，替代了Vector，替代了Vector，查询的速度快。

[-LinkedList:内部是链表数据结构，是不同步的。增删元素的速度很快。

LinkedList方法:

addFirst();

addLast();

jdk1.6版本后新方法:

offerFirst()/与addFirst方法没有区别。

offerLast()/与addLast方法没有区别。

getFirst()/获取但不移除，如果链表为空，抛出NoSuchElementException。

getLast();

jdk1.6版本后新方法:

peekFirst()/获取但不移除，如果链表为空，返回null。

peekLast();

removeFirst()/获取并移除，如果链表为空，抛出NoSuchElementException。

removeLast();

jdk1.6版本后新方法:

pollFirst()/获取并移除，如果链表为空，返回null;

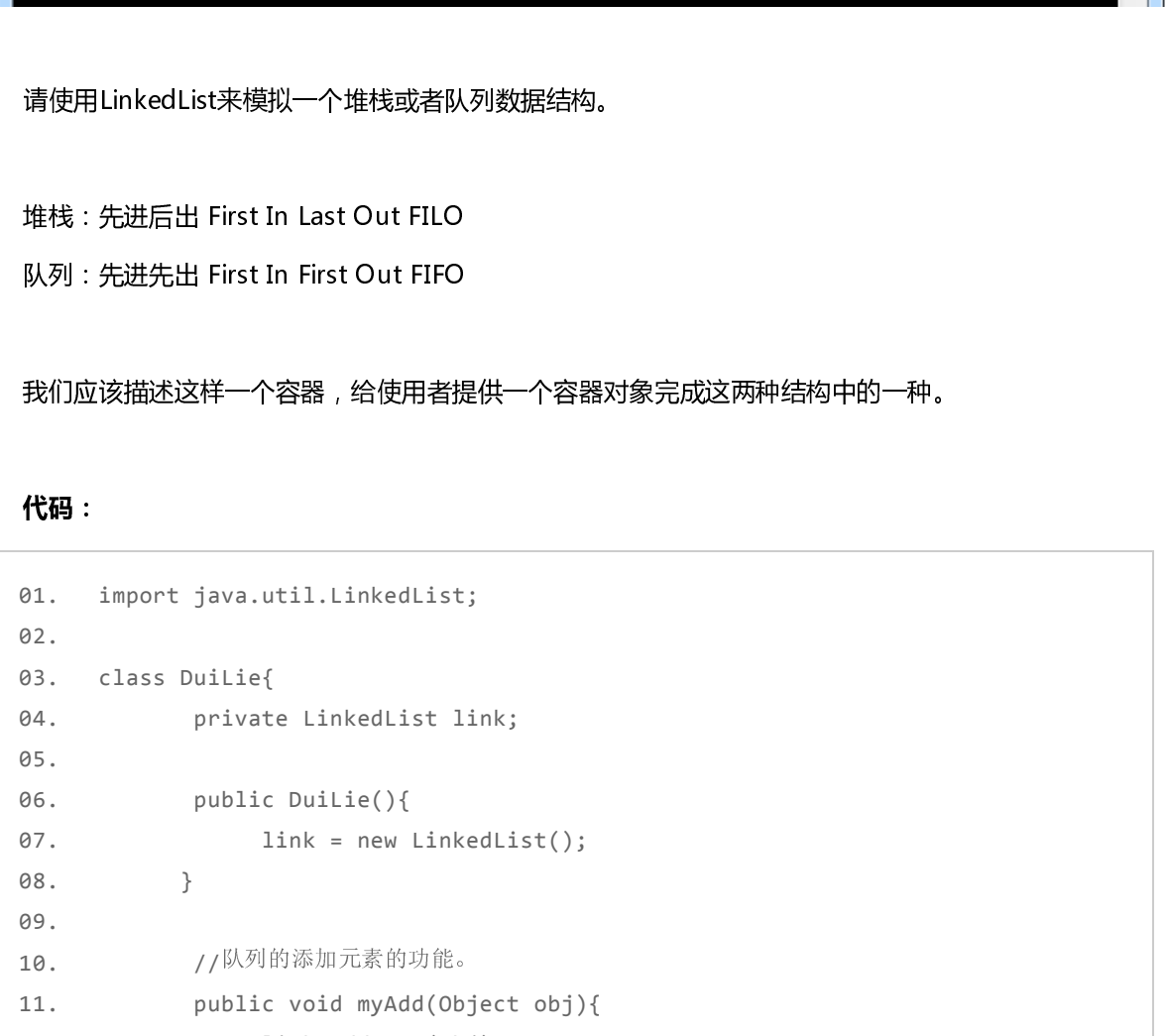
pollLast();

示例1:

```
01. import java.util.Enumeration;
02. import java.util.Iterator;
03. import java.util.Vector;
04.
05. public class VectorDemo{
06.     public static void main(String[] args){
07.         Vector v = new Vector();
08.
09.         v.addElement("abc1");
10.         v.addElement("abc2");
11.         v.addElement("abc3");
12.         v.addElement("abc4");
13.
14.         Enumeration en = v.elements();
15.
16.         while(en.hasMoreElements()){
17.             System.out.println("nextElement:" + en.nextElement());
18.         }
19.
20.         Iterator it = v.iterator();
21.
22.         while(it.hasNext()){
23.             System.out.println("next:" + it.next());
24.         }
25.     }
26. }
```

复制代码

运行结果:



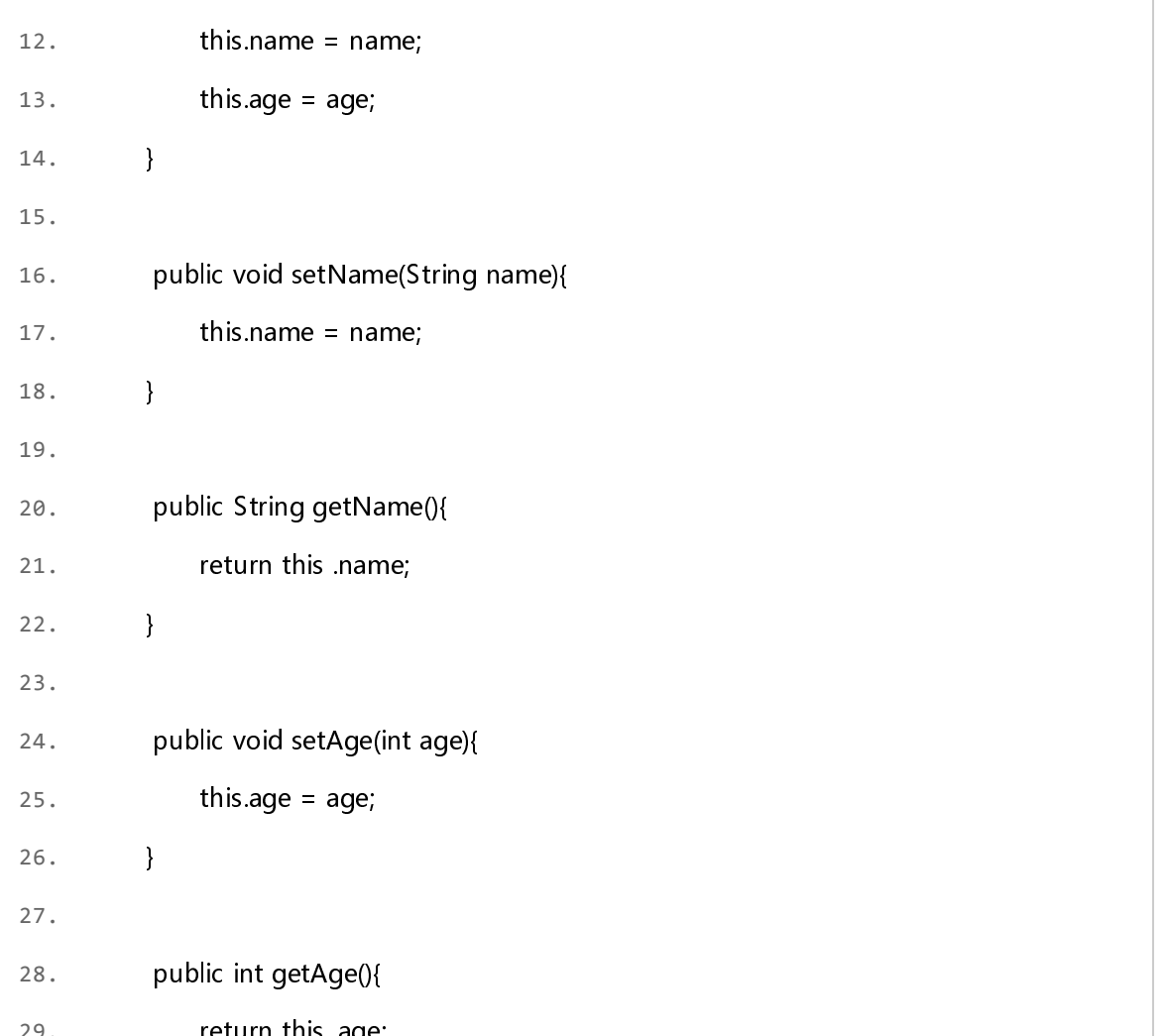
```
D:\code>day17>javac VectorDemo.java
注意: VectorDemo.java 使用了未经检查或不安全的操作。
注意: 要了解详细信息, 请使用 -Xlint:unchecked 重新编译。
D:\code>day17>java VectorDemo
nextElement:abc1
nextElement:abc2
nextElement:abc3
nextElement:abc4
next:abc1
next:abc2
next:abc3
next:abc4
```

示例2:

```
01. import java.util.Iterator;
02. import java.util.LinkedList;
03.
04. public class LinkedListDemo{
05.     public static void main(String[] args){
06.         LinkedList link = new LinkedList();
07.
08.         link.addFirst("abc1");
09.         link.addFirst("abc2");
10.         link.addFirst("abc3");
11.         link.addFirst("abc4");
12.
13.         Iterator it = link.iterator();
14.         while(it.hasNext()){
15.             System.out.println("next:" + it.next());
16.         }
17.
18.         System.out.println(link);
19.         System.out.println("getFirst:" + link.getFirst()); //获取第一个, 但
            是不删除。
20.         System.out.println("getLast:" + link.getLast());
21.
22.         System.out.println("removeFirst:" + link.removeFirst()); //获取第
            一个, 但是删除
23.         System.out.println("removeLast:" + link.removeLast());
24.
25.         //删除所有元素的方法
26.         while(!link.isEmpty()){
27.             System.out.println(link.removeFirst());
28.         }
29.     }
30. }
```

复制代码

运行结果:



```
D:\code>day17>javac LinkedListDemo.java
注意: LinkedListDemo.java 使用了未经检查或不安全的操作。
注意: 要了解详细信息, 请使用 -Xlint:unchecked 重新编译。
D:\code>day17>java LinkedListDemo
next:abc4
next:abc3
next:abc2
next:abc1
[abc4, abc3, abc2, abc1]
getFirst:abc4
getLast:abc1
removeFirst:abc1
abc3
abc2
```

请使用LinkedList来模拟一个堆栈或者队列数据结构。

堆栈: 先进后出 First In Last Out FILO

队列: 先进先出 First In First Out FIFO

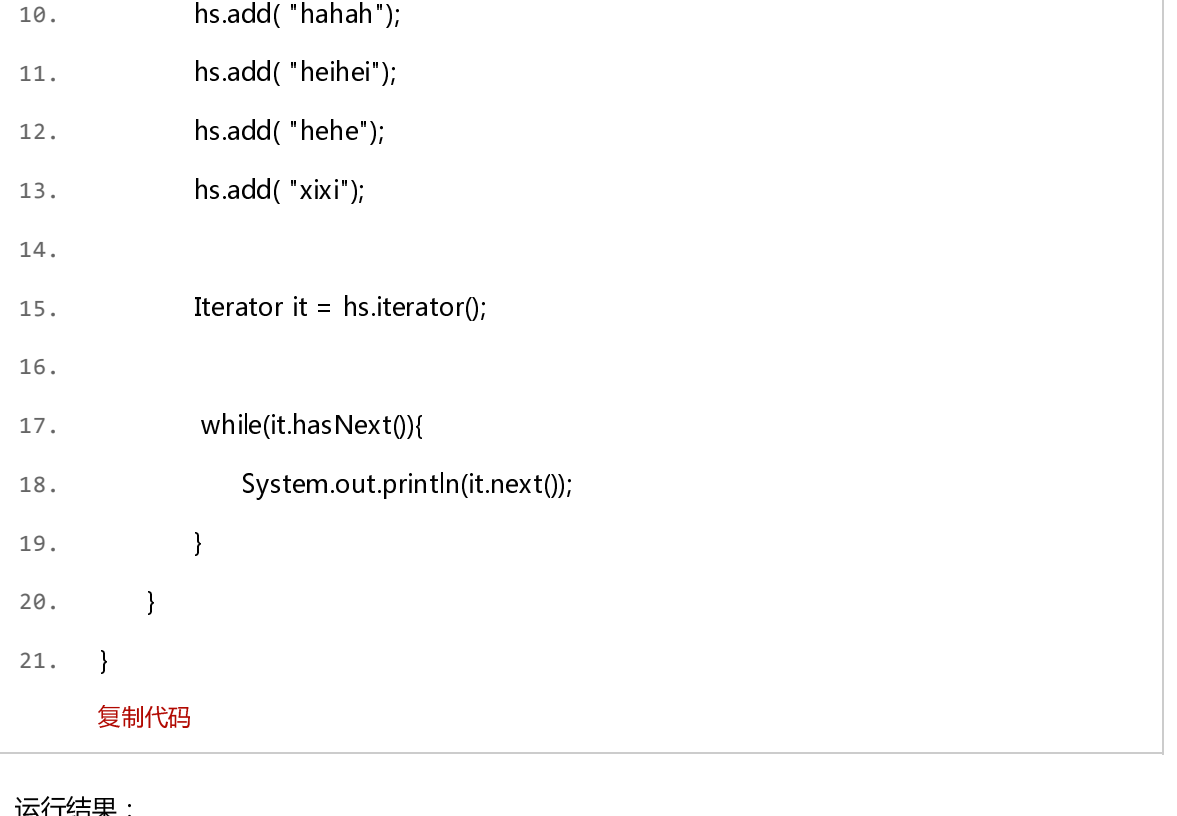
我们应该描述这样一个容器, 给使用者提供一个容器对象完成这两种结构中的一种。

代码:

```
01. import java.util.LinkedList;
02.
03. class Duilie{
04.     private LinkedList link;
05.
06.     public Duilie(){
07.         link = new LinkedList();
08.     }
09.
10.     //队列的添加元素的功能。
11.     public void myAdd(Object obj){
12.         link.addLast(obj);
13.     }
14.
15.     public Object myGet(){
16.         return link.removeFirst();
17.     }
18.
19.     public boolean isNull(){
20.         return link.isEmpty();
21.     }
22. }
23.
24. public class DuilieTest{
25.     public static void main(String[] args){
26.         Duilie dl = new Duilie();
27.
28.         dl.myAdd("abc1");
29.         dl.myAdd("abc2");
30.         dl.myAdd("abc3");
31.         dl.myAdd("abc4");
32.
33.         while(!dl.isNull()){
34.             System.out.println(dl.myGet());
35.         }
36.     }
37. }
```

复制代码

运行结果:



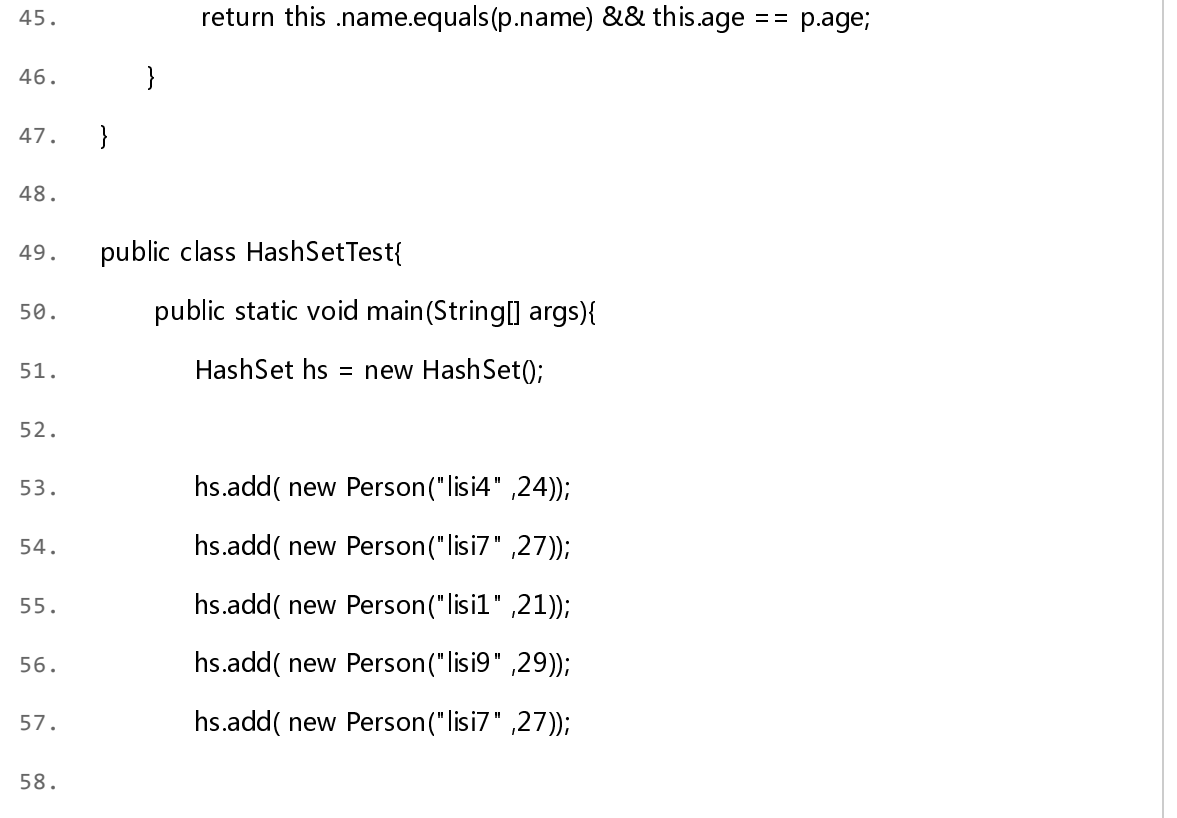
```
D:\code>day17>javac DuilieTest.java
注意: DuilieTest.java 使用了未经检查或不安全的操作。
注意: 要了解详细信息, 请使用 -Xlint:unchecked 重新编译。
D:\code>day17>java DuilieTest
abc1
abc2
abc3
abc4
```

示例3:

```
01. import java.util.ArrayList;
02. import java.util.Iterator;
03.
04. class Person{
05.     private String name;
06.     private int age;
07.
08.     public Person(){
09.     }
10.
11.     public Person(String name,int age){
12.         this.name = name;
13.         this.age = age;
14.     }
15.
16.     public void setName(String name){
17.         this.name = name;
18.     }
19.
20.     public String getName(){
21.         return this.name;
22.     }
23.
24.     public void setAge(int age){
25.         this.age = age;
26.     }
27.
28.     public int getAge(){
29.         return this.age;
30.     }
31. }
32.
33. public class ArrayListTest{
34.     public static void main(String[] args){
35.         ArrayList al = new ArrayList();
36.         al.add( new Person("lisi1" ,21));
37.         al.add( new Person("lisi2" ,22));
38.         al.add( new Person("lisi3" ,23));
39.         al.add( new Person("lisi4" ,24));
40.
41.         Iterator it = al.iterator();
42.         while(it.hasNext()){
43.             Person p = (Person)(it.next());
44.             System.out.println(p.getName()+ " " + p.getAge());
45.         }
46.     }
47. }
```

复制代码

运行结果:



```
D:\code>day17>javac ArrayListTest.java
注意: ArrayListTest.java 使用了未经检查或不安全的操作。
注意: 要了解详细信息, 请使用 -Xlint:unchecked 重新编译。
D:\code>day17>java ArrayListTest
lisi1:21
lisi2:22
lisi3:23
lisi4:24
```

Set元素不可以重复, 是无序。

Set接口中的方法和Collection一致。

[-HashSet: 内部数据结构是哈希表, 是不同步的。

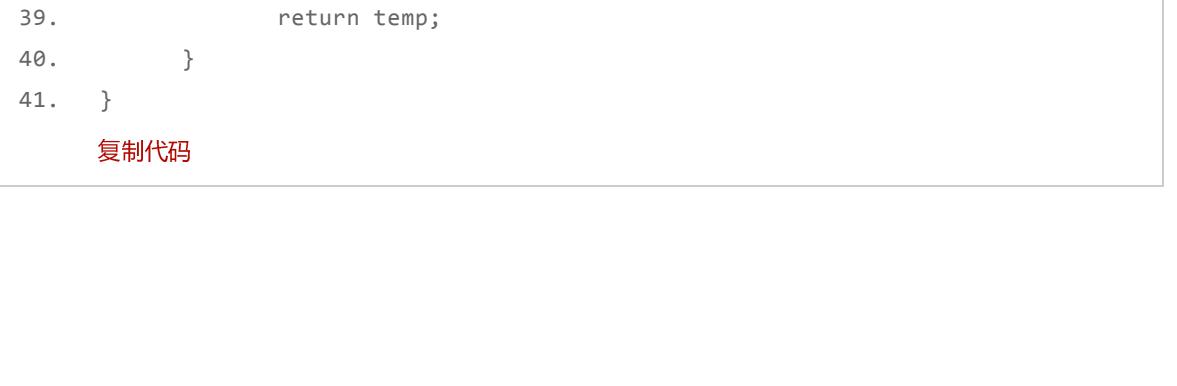
[-TreeSet: 可以对Set集合中的元素进行排序, 是不同步的。

示例4:

```
01. import java.util.HashSet;
02. import java.util.Iterator;
03. import java.util.Set;
04.
05. public class HashSetDemo{
06.     public static void main(String[] args){
07.
08.         Set hs = new HashSet();
09.
10.         hs.add("hahah");
11.         hs.add("heihei");
12.         hs.add("hehe");
13.         hs.add("xixi");
14.
15.         Iterator it = hs.iterator();
16.
17.         while(it.hasNext()){
18.             System.out.println(it.next());
19.         }
20.     }
21. }
```

复制代码

运行结果:



```
D:\code>day17>javac HashSetDemo.java
注意: HashSetDemo.java 使用了未经检查或不安全的操作。
注意: 要了解详细信息, 请使用 -Xlint:unchecked 重新编译。
D:\code>day17>java HashSetDemo
hehe
heihei
hahah
xixi
```

哈希表确定元素是否相同

1. 判断的是两个元素的哈希值是否相同。

如果相同, 再判断两个对象的内容是否相同。

2. 判断哈希值相同, 其实判断的是对象的HashCode方法。判断内容相同, 用的是equals方法。

P.S.

如果哈希值不同, 不需要判断equals。

往HashSet集合中存储Person对象。如果姓名和年龄相同, 视为同一个人, 视为相同元素。

示例5:

```
01. import java.util.HashSet;
02. import java.util.Iterator;
03.
04. class Person{
05.     private String name;
06.     private int age;
07.
08.     public Person(){
09.     }
10.
11.     public Person(String name,int age){
12.         this.name = name;
13.         this.age = age;
14.     }
15.
16.     public void setName(String name){
17.         this.name = name;
18.     }
19.
20.     public String getName(){
21.         return this.name;
22.     }
23.
24.     public void setAge(int age){
25.         this.age = age;
26.     }
27.
28.     public int getAge(){
29.         return this.age;
30.     }
31.
32.     public int hashCode(){
33.         return name.hashCode() + age * 39;
34.     }
35.
36.     public boolean equals(Object obj){
37.         if(this == obj){
38.             return true ;//同一个对象放两次, 直接返回true
39.         }
40.         if(!obj instanceof Person){
41.             throw new ClassCastException("类型错误");
42.         }
43.         Person p = (Person)obj;
44.
45.         return this.name.equals(p.name) && this.age == p.age;
46.     }
47. }
48.
49. public class HashSetTest{
50.     public static void main(String[] args){
51.         HashSet hs = new HashSet();
52.
53.         hs.add( new Person("lisi4" ,24));
54.         hs.add( new Person("lisi7" ,27));
55.         hs.add( new Person("lisi1" ,21));
56.         hs.add( new Person("lisi9" ,29));
57.         hs.add( new Person("lisi7" ,27));
58.
59.         Iterator it = hs.iterator();
60.
61.         while(it.hasNext()){
62.             Person p = (Person)(it.next());
63.             System.out.println(p.getName()+ "..." + p.getAge());
64.         }
65.     }
66. }
```

复制代码

运行结果:


```
D:\code>day17>javac HashSetTest.java
注意: HashSetTest.java 使用了未经检查或不安全的操作。
注意: 要了解详细信息, 请使用 -Xlint:unchecked 重新编译。
D:\code>day17>java HashSetTest
lisi2...22
lisi4...24
lisi1...21
lisi9...29
```

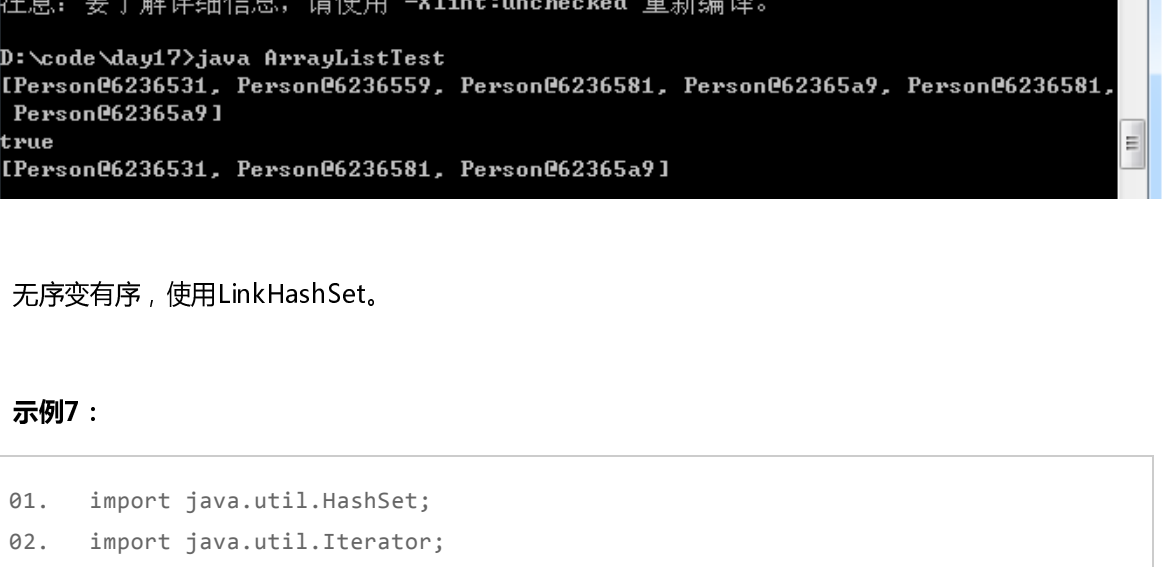
定义功能去除ArrayList中的重复元素。

示例6:

```
01. import java.util.ArrayList;
02. import java.util.Iterator;
03.
04. public class ArrayListTest{
05.     public static void main(String[] args){
06.         ArrayList al = new ArrayList();
07.         al.add( new Person("lisi1" ,21));
08.         al.add( new Person("lisi2" ,22));
09.         al.add( new Person("lisi3" ,23));
10.         al.add( new Person("lisi4" ,24));
11.         al.add( new Person("lisi3" ,23));
12.         al.add( new Person("lisi4" ,24));
13.
14.         System.out.println(al);
15.
16.         al = getSingleElement(al);
17.
18.         //remove底层用的是Equals方法
19.         System.out.println(al.remove( new Person("lisi2" ,22)));
20.         System.out.println(al);
21.     }
22.
23.     public static ArrayList getSingleElement(ArrayList al){
24.         //1. 定义一个临时容器
25.         ArrayList temp = new ArrayList();
26.
27.         //2. 迭代al集合
28.         Iterator it = al.iterator();
29.
30.         while(it.hasNext()){
31.             Object obj = it.next();
32.
33.             //3. 判断被迭代的元素是否在临时容器存在
34.             //contains方法依靠的是equals方法
35.             if(!temp.contains(obj)){
36.                 temp.add(obj);
37.             }
38.         }
39.         return temp;
40.     }
41. }
```

复制代码

运行结果：



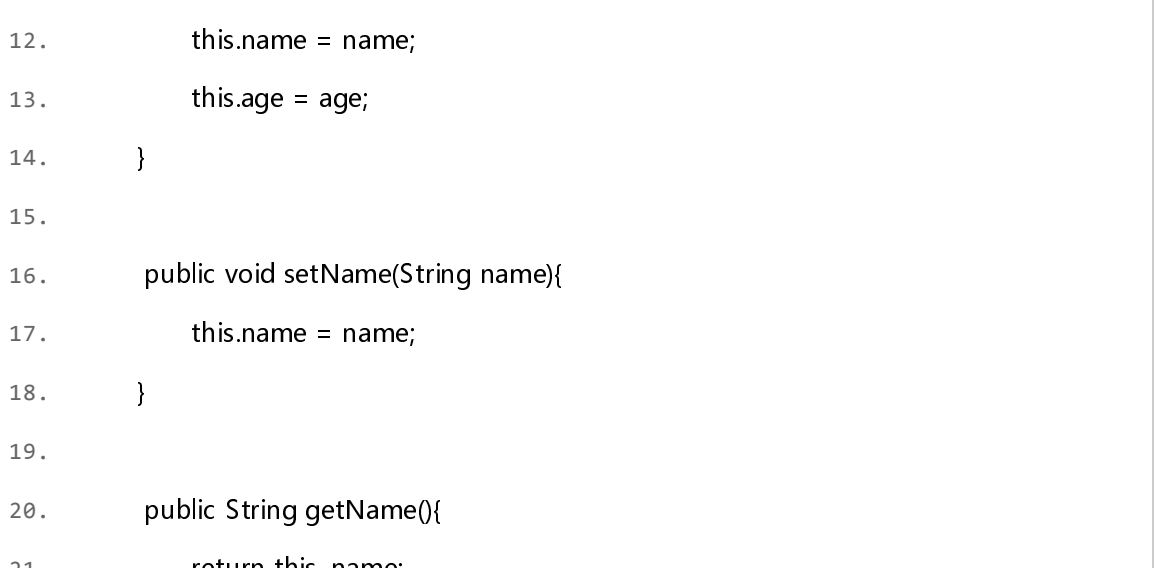
无序变有序，使用LinkHashSet。

示例7：

```
01. import java.util.HashSet;
02. import java.util.Iterator;
03. import java.util.LinkedHashSet;
04.
05. public class LinkedHashSetDemo{
06.     public static void main(String[] args){
07.         HashSet hs = new LinkedHashSet();
08.
09.         hs.add( "haha");
10.         hs.add( "hehe");
11.         hs.add( "heihei");
12.         hs.add( "xixii");
13.
14.         Iterator it = hs.iterator();
15.
16.         while(it.hasNext()){
17.             System.out.println(it.next());
18.         }
19.     }
20. }
```

复制代码

运行结果：



TreeSet判断元素唯一性的方式：就是根据比较方法的返回结果是否是0，是0，就是相同元素，不存。

示例8：

```
01. import java.util.Iterator;
02. import java.util.TreeSet;
03.
04. class Person implements Comparable{
05.     private String name;
06.     private int age;
07.
08.     public Person(){
09.     }
10.
11.     public Person(String name,int age){
12.         this.name = name;
13.         this.age = age;
14.     }
15.
16.     public void setName(String name){
17.         this.name = name;
18.     }
19.
20.     public String getName(){
21.         return this.name;
22.     }
23.
24.     public void setAge(int age){
25.         this.age = age;
26.     }
27.
28.     public int getAge(){
29.         return this.age;
30.     }
31.
32.     public int hashCode(){
33.         return name.hashCode() + age * 39;
34.     }
35.
36.     public boolean equals(Object obj){
37.         if(this == obj)
38.             return true;
39.
40.         if(!(obj instanceof Person))
41.             throw new ClassCastException("类型错误");
42.
43.         Person p = (Person)obj;
44.
45.         return this.name.equals(p.name) && this.age == p.age;
46.     }
47.
48.     public int compareTo(Object o){
49.         Person p = (Person)o;
50.
51.         //先按照年龄排序，再按照年龄排序，以年龄相同的人，没有存进去。
52.         int temp = this.age - p.age;
53.         return temp == 0?this.name.compareTo(p.name):temp;
54.     }
55. }
56.
57. public class TreeSetDemo{
58.     public static void main(String[] args){
59.         TreeSet ts = new TreeSet();
60.
61.         //以Person对象年龄进行从小到大排序
62.         ts.add( new Person("zhangsan" ,28));
63.         ts.add( new Person("wangwu" ,23));
64.         ts.add( new Person("lisi" ,21));
65.         ts.add( new Person("zhouqi" ,29));
66.         ts.add( new Person("zhaoliu" ,25));
67.
68.         Iterator it = ts.iterator();
69.
70.         while(it.hasNext()){
71.             Person p = (Person)it.next();
72.
73.             System.out.println(p.getName() + ":" + p.getAge());
74.         }
75.     }
76. }
```

复制代码

运行结果：



P.S.

如果自定义类实现了Comparable接口，并且TreeSet的构造函数中也传入了比较器，那么将以比较器的比较规则为准。

TreeSet集合的底层是二叉树进行排序的。

练习：对字符串进行长度排序。

代码：

```
01. import java.util.Comparator;
02. import java.util.Iterator;
03. import java.util.TreeSet;
04.
05. public class TreeSetTest{
06.     public static void main(String[] args){
07.         TreeSet ts = new TreeSet(new ComparatorByLen());
08.
09.         ts.add( "aaaa");
10.         ts.add( "zz");
11.         ts.add( "nbag");
12.         ts.add( "cba");
13.         ts.add( "abc");
14.
15.         Iterator it = ts.iterator();
16.
17.         while(it.hasNext()){
18.             System.out.println(it.next());
19.         }
20.     }
21. }
22.
23. class ComparatorByLen implements Comparator{
24.     public int compare(Object o1,Object o2){
25.         String s1 = (String)o1;
26.         String s2 = (String)o2;
27.
28.         int temp = s1.length() - s2.length();
29.
30.         return temp == 0?s1.compareTo(s2):temp;
31.     }
32. }
```

复制代码

运行结果：



~END~



~爱上海，爱黑马~



如果