

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

|--Vector:内部是数组数据结构,是同步的。增删,查询都很慢。

List:

04.

05.

06.

07.

05.

06.

08.

09.

10.

11. 12.

13.

15. 16. 17.

public class VectorDemo{

public static void main(String[] args){

Vector v = new Vector();

|--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: import java.util.Enumeration; 01. 02. import java.util.Iterator; import java.util.Vector; 03.

v.addElement("abc1"); 09. v.addElement("abc2"); 10. 11. v.addElement("abc3"); 12. v.addElement("abc4"); 13. Enumeration en = v.elements(); 14. 15. while(en.hasMoreElements()){ 16. System.out.println("nextElement:" + en.nextElement()); 17. } 18.

Iterator it = v.iterator(); 20. 21. while(it.hasNext()){ 22. System.out.println("next:" + it.next()); 23. } 24. 25. } 26. 复制代码 运行结果: _ _ _ X 📰 管理员: C:\Windows\system32\cmd.exe

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; import java.util.LinkedList; 02. 03. 04. public class LinkedListDemo{

public static void main(String[] args){

link.addFirst("abc1");

link.addFirst("abc2");

link.addFirst("abc3");

link.addFirst("abc4");

while(it.hasNext()){

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

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

import java.util.LinkedList;

public class DuiLieTest{

[abc4, abc3, abc2, abc1]

next:abc4 next:abc3 next:abc2 next:abc1

abc3 abc2

代码:

02.

24. 25.

26. 27.

30.

31.

32. 33.

34.

36. 37. }

运行结果:

示例3:

01.

02.

03.

04.

06.

33.

34.

35.

36.

37.

38.

39.

41.

42.

43.

44.

45.

P.S.

示例5:

01.

02.

03.

04.

05.

06.

07.

08.

09.

10.

11.

12.

14.

15.

16.

17.

18.

19.

20.

21.

22.

23.

24.

25.

26.

27.

28.

29.

30.

31.

32.

33.

34.

35.

36.

37.

60.

61.

62.

63.

64.

66.

示例6:

31. 32.

33.

34.

35.

36. 37.

38.

39.

40. 41.

复制代码

while(it.hasNext()){

}

定义功能去除ArrayList中的重复元素。

}

复制代码

}

Person p = (Person)it.next();

如果哈希值不同,不需要判断equals。

import java.util.HashSet;

import java.util.Iterator;

private String name;

private int age;

public Person(){

public Person(String name, int age){

public void setName(String name){

this.name = name;

public String getName(){

return this .name;

public void setAge(int age){

this.age = age;

public int getAge(){

return this .age;

public int hashCode(){

return name.hashCode() + age * 39;

public boolean equals(Object obj){

this.name = name;

class Person{

}

}

}

}

}

}

}

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。

public static void main(String[] args){

ArrayList al = new ArrayList();

al.add(new Person("lisi1",21));

al.add(new Person("lisi2" ,22));

al.add(new Person("lisi3",23));

al.add(new Person("lisi4",24));

Person p = (Person)(it.next());

System.out.println(p.getName() + ":" + p.getAge());

Ξ

Ξ

Iterator it = al.iterator();

while(it.hasNext()){

复制代码

🔤 管理员: C:\Windows\system32\cmd.exe

import java.util.ArrayList;

import java.util.Iterator;

private String name;

private int age;

class Person{

getFirst:abc4 getLast:abc1 removeFirst:abc4 removeLast:abc1

Iterator it = link.iterator();

LinkedList link = new LinkedList();

Ξ

System.out.println(link); 18. System.out.println("getFirst:" + link.getFirst()); //获取第一个, 但 19. 是不删除。 20. System.out.println("getLast:" + link.getLast()); 21. System.out.println("removeFirst:" + link.removeFirst()); //获取第 22. 一个,但是删除 System.out.println("removeLast:" + link.removeLast()); 23. 24. //删除所有元素的方法 25. while(!link.isEmpty()){ System.out.println(link.removeFirst()); 27. 28. } 29. } 30. } 复制代码 运行结果: 📷 管理员: C:\Windows\system32\cmd.exe

System.out.println("next:" + it.next());

class DuiLie{ 03. private LinkedList link; 04. 05. public DuiLie(){ link = new LinkedList(); 08. } 09. //队列的添加元素的功能。 10. 11. public void myAdd(Object obj){ link.addLast(obj); 12. 13. public Object myGet(){ 16. return link.removeFirst(); 17. } 18. public boolean isNull(){ 19. 20. return link.isEmpty(); } 22. 23.

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。

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

public static void main(String[] args){ DuiLie dl = new DuiLie();

> dl.myAdd("abc1"); dl.myAdd("abc2");

dl.myAdd("abc3");

dl.myAdd("abc4");

while(!dl.isNull()){

System.out.println(dl.myGet());

_ D X

```
07.
           public Person(){
08.
09.
          }
10.
11.
           public Person(String name,int age){
12.
               this.name = name;
               this.age = age;
13.
14.
          }
15.
           public void setName(String name){
16.
17.
               this.name = name;
          }
18.
19.
20.
           public String getName(){
               return this .name;
21.
22.
          }
23.
           public void setAge(int age){
24.
               this.age = age;
25.
26.
          }
27.
28.
           public int getAge(){
               return this .age;
29.
          }
30.
31.
      }
32.
       public class ArrayListTest{
```

46. } 47. 复制代码 运行结果: _ 0 X 🔤 管理员: C:\Windows\system32\cmd.exe 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 D:\code\day17>java ArrayListTest lisi1:21 lisi2:22 lisi3:23 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: import java.util.HashSet; 01. import java.util.Iterator; 03. import java.util.Set; 04. public class HashSetDemo{ 05. public static void main(String[] args){ 06. 07. Set hs = new HashSet(); 08. 09. hs.add("hahah"); 10. hs.add("heihei"); 11. hs.add("hehe"); 12. hs.add("xixi"); 13. Iterator it = hs.iterator(); 15. 16. while(it.hasNext()){ 17. System.out.println(it.next()); 18. } 19. 20. } 21. } 复制代码 运行结果: _ _ _ X 🔤 管理员: C:\Windows\system32\cmd.exe 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 he he 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。

if(this == obj) return true ;//同一个对象放两次,直接返回true 38. 39. if(!(obj instanceof Person)) 40. throw new ClassCastException("类型错误"); 41. 42. Person p = (Person)obj;43. 44. return this .name.equals(p.name) && this.age == p.age; 46. } } 47. 48. public class HashSetTest{ 49. public static void main(String[] args){ 50. HashSet hs = new HashSet(); 51. 52. hs.add(new Person("lisi4",24)); 53. hs.add(new Person("lisi7" ,27)); hs.add(new Person("lisi1",21)); 55. hs.add(new Person("lisi9",29)); 56. hs.add(new Person("lisi7",27)); 57. 58. Iterator it = hs.iterator(); 59.

运行结果: 🔤 管理员: C:\Windows\system32\cmd.exe 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 lisi7...27 lisi1...21 Ξ lisi4...24 lisi9...29

System.out.println(p.getName() + "..." + p.getAge());

import java.util.ArrayList; 01. import java.util.Iterator; 02. 03. public class ArrayListTest{ 04. public static void main(String[] args){ 05. ArrayList al = new ArrayList(); 06. al.add(new Person("lisi1" ,21)); 07. al.add(new Person("lisi2" ,22)); 08. al.add(new Person("lisi3" ,23)); 09. al.add(new Person("lisi4" ,24)); 10. 11. al.add(new Person("lisi3" ,23)); 12. al.add(new Person("lisi4" ,24)); 13. System.out.println(al); 14. 15. al = getSingleElement(al); 16. 17. //remove底层用的还是equals方法 18. System.out.println(al.remove(new Person("lisi2" ,22))); 19. System.out.println(al); } 22. public static ArrayList getSingleElement(ArrayList al){ 23. //1、定义一个临时容器 24. ArrayList temp = new ArrayList(); 25. 26. //2、迭代al集合 27. 28. Iterator it = al.iterator(); 29. while(it.hasNext()){ 30.

Object obj = it.next();

//3、判断被迭代到的元素是否在临时容器存在

//contains方法依靠的是equals方法

if(!temp.contains(obj)){

temp.add(obj);

}

return temp;

}

```
运行结果:
                                                                             _ _ _ X
画 管理员: C:\Windows\system32\cmd.exe
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
[Person@6236531, Person@6236559, Person@6236581, Person@62365a9, Person@6236581,
Person@62365a91
true
[Person@6236531, Person@6236581, Person@62365a9]
无序变有序,使用LinkHashSet。
示例7:
01.
      import java.util.HashSet;
02.
      import java.util.Iterator;
      import java.util.LinkedHashSet;
03.
05.
      public class LinkedHashSetDemo{
             public static void main(String[] args){
06.
07.
                   HashSet hs = new LinkedHashSet();
08.
                   hs.add( "hahaa");
09.
                   hs.add( "hehe");
10.
                   hs.add( "heihei");
12.
                   hs.add( "xixii");
13.
14.
                   Iterator it = hs.iterator();
15.
16.
                   while(it.hasNext()){
                         System.out.println(it.next());
17.
                   }
19.
             }
20.
      }
      复制代码
运行结果:
                                                                             📷 管理员: C:\Windows\system32\cmd.exe
D:\code\day17>javac LinkedHashSetDemo.java
注意: LinkedHashSetDemo.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 LinkedHashSetDemo
hahaa
he he
heihei
xixii
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;
       import java.util.TreeSet;
02.
03.
       class Person implements Comparable{
04.
           private String name;
05.
           private int age;
06.
07.
           public Person(){
08.
09.
          }
10.
           public Person(String name,int age){
11.
12.
               this.name = name;
               this.age = age;
13.
          }
14.
15.
           public void setName(String name){
16.
              this.name = name;
17.
18.
          }
19.
```

20. public String getName(){ 21. return this .name; } 22. 23. 24. public void setAge(int age){ this.age = age; 25. } 26. 27. public int getAge(){ 28. return this .age; 29.

30. } 31. public int hashCode(){ return name.hashCode() + age * 39; 33. } 34. public boolean equals(Object obj){ 36. if(this == obj) 37. return true; 39. if(!(obj instanceof Person)) 40. throw new ClassCastException("类型错误"); 41. 42. Person p = (Person)obj;43. return this .name.equals(p.name) && this.age == p.age; 45. } 46. 47. public int compareTo(Object o){ 48. Person p = (Person)o;49. 50.

//先按照年龄排序,再按照年龄排序,以免年龄相同的人,没有存进去。 51. 52. int temp = this.age - p.age; return temp = = 0?this.name.compareTo(p.name):temp; 53. } 54. } 55. 56. public class TreeSetDemo{ 57. public static void main(String[] args){ 59. TreeSet ts = new TreeSet(); 60. //以Person对象年龄进行从小到大的排序 61. ts.add(new Person("zhangsan" ,28)); 62. ts.add(new Person("wangwu" ,23)); 63. ts.add(new Person("lisi" ,21)); ts.add(new Person("zhouqi",29)); 65. ts.add(new Person("zhaoliu",25)); 66. 67. Iterator it = ts.iterator(); 68. 69. 70. while(it.hasNext()){ Person p = (Person)it.next(); 71. 72. 73. System.out.println(p.getName() + ":" + p.getAge()); } 74. } 75. 76. } 复制代码 运行结果: _ D X 画 管理员: C:\Windows\system32\cmd.exe D:\code\day17>javac TreeSetDemo.java 注意: TreeSetDemo.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 TreeSetDemo lisi:21 wangwu:23 zhaoliu:25 zhangsan:28 zhouqi:29 TreeSet对元素进行排序的方式一: 让元素自身具备比较功能,元素就需要实现Comparable接口, 覆盖compareTo方法。 如果不要按照对象中具备的自然顺序进行排序。如果对象中不具备自然顺序。怎么办? 可以使用TreeSet集合第二种排序方式: 让集合自身具备比较功能,定义一个类实现Comparator接口,覆盖compare方法。 将该类对象作为参数传递给TreeSet集合的构造函数。

示例9:

01. 02.

03.

04.

05.

08.

09. 10.

11.

12. 13.

14.

15.

16. 17.

18. 19.

20. 21.

22.

23.

24.

25. 26. 27.

28. 29.

30.

}

}

public class TreeSetDemo{

import java.util.Comparator;

//创建了一个根据Person类的name进行排序的比较器。

class ComparatorByName implements Comparator{

Person p1 = (Person)o1;

Person p2 = (Person)o2;

public int compare(Object o1,Object o2){

public static void main(String[] args){

//以Person对象年龄进行从小到大的排序

ts.add(new Person("lisi" ,21));

Iterator it = ts.iterator();

while(it.hasNext()){

ts.add(new Person("zhouqi" ,29));

ts.add(new Person("zhaoliu" ,25));

Person p = (Person)it.next();

ts.add(new Person("zhangsan" ,28)); ts.add(new Person("wangwu" ,23));

int temp = p1.getName().compareTo(p2.getName());

return temp == 0?p1.getAge()-p2.getAge() : temp;

TreeSet ts = new TreeSet(new ComparatorByName());

import java.util.Iterator;

import java.util.TreeSet;

31. System.out.println(p.getName() + ":" + p.getAge()); 32. 33. 34. } 35. 复制代码 运行结果: _ D X 📷 管理员: C:\Windows\system32\cmd.exe D:\code\day17>javac TreeSetDemo.java 注意: TreeSetDemo.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 TreeSetDemo lisi:21 wangwu:23 zhangsan:28 zhaoliu:25 zhouqi:29 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; import java.util.Iterator; 02. 03. import java.util.TreeSet; 04. public class TreeSetTest{ 05. public static void main(String[] args){ 06. TreeSet ts = new TreeSet(new ComparatorByLen()); 07. 08. ts.add("aaaa"); 09. ts.add("zz"); 10.

ts.add("nbag");

ts.add("cba");

ts.add("abc");

}

}

}

画 管理员: C:\Windows\system32\cmd.exe

D:\code\day17>java TreeSetTest

复制代码

Iterator it = ts.iterator();

System.out.println(it.next());

while(it.hasNext()){

class ComparatorByLen implements Comparator{

String s1 = (String)o1;

String s2 = (String)o2;

D:\code\day17>javac TreeSetTest.java 注意: TreeSetTest.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。

public int compare(Object o1,Object o2){

int temp = s1.length() - s2.length();

return temp == 0?s1.compareTo(s2):temp;

~END~

加米州 水水水水水 水水水水水水水水水

~爱上海,爱黑马~

_ D X

11.

12.

13. 14.

15. 16.

17.

18. 19.

20. 21.

22.

23.

24.

25.

26. 27.

28. 29.

30.

31. 32.

运行结果:

zz abc c ba aaaa nbag }