

```
非贷款,0元入学,不1万就业不给1分钱学费,我们已干四年了!
```

6.1 常用对象

6、集合

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

为了方便操作基本数据类型值,将其封装成了对象,在对象中定义了属性和行为丰富了该数据的操 用于描述该对象的类就称为基本数据类型对象包装类。

6.1.2 基本数据类型对象包装类

作。 byte Byte

Short short int Integer

Long Float

Character

long float

double Double char boolean Boolean

该包装对象主要用于基本类型和字符串之间的转换

基本类型-->字符串

2. 用String类中的静态方法valueOf(基本类型数值); 字符串-->基本类型 1. 使用包装类中的静态方法xxx parseXxx("xxx"); int parseInt(String str);

1. 基本类型数值+""

long parseLong(String str); boolean parseBoolean(String str); 只有Character没有parse方法。 2. 如果字符串被Integer进行对象的封装,可使用另一个非静态的方法,intValue。 将一个Integer对象转成基本数据类型值。

示例1: 01. public class WrapperDemo{ public static void main(String[] args){ 02.

3. 用Integer的静态方法valueOf(String s);

//整型最大值 System.out.println(Integer.MAX\_VALUE); //十进制转换成二进制 05. 06. System.out.println(Integer.toBinaryString(6)); 07.

08.

复制代码 运行结果:

画 管理员: C:\Windows\system32\cmd.exe

\_ D X

\_ \_ \_ X

- - X

D:\code\day16>javac WrapperDemo.java D:\code\day16>java WrapperDemo 2147483647

110 示例2:

public class WrapperDemo{ public static void main(String[] args){ 02. 03. Integer i = new Integer(5); System. out.println(Integer.parseInt( "123") + 1);

04. 05. 06. 复制代码 运行结果:

📷 管理员: C:\Windows\system32\cmd.exe

D:\code\day16>javac WrapperDemo.java

public static void main(String[] args){

if(Boolean.parseBoolean("true" ))

Integer i = new Integer("123" );

System.out.println(i.intValue());

System.out.println( "true");

D:\code\day16>java WrapperDemo 124 示例3: public class WrapperDemo{ 01.

复制代码 运行结果: 画 管理员: C:\Windows\system32\cmd.exe D:\code\day16>javac WrapperDemo.java

true 123

D:\code\day16>java WrapperDemo

String to String(int i,int radix);

}

画 管理员: C:\Windows\system32\cmd.exe

D:∖code∖day16>java WrapperDemo

其他进制-->十进制方法:

int parseInt(String s,int radix);

public static void main(String[] args){

System.out.println(Integer.parseInt("110",10));

System.out.println(Integer.parseInt( "3C",16));

private static final String SPACE\_SEPARATOR = " ";

String numStr = "20 79 9 -7 88 36 29";

numStr = sortStringNumber(numStr);

public static String sortStringNumber(String numStr){

String[] str\_arr = stringToArray(numStr);

public static void main(String[] args){

System.out.println(numStr);

System.out.println(numStr);

//1、将字符串变成字符串数组

//2、将字符串数组变成int数组

//3、对int数组排序

return temp;

else

return sb.toString();

Arrays.sort(num\_arr);

}

}

}

}

return arr;

return str\_arr;

mySortArray(num\_arr);

int[] num\_arr = toIntArray(str\_arr);

//4、将排序后的int数组变成字符串

String temp = arrayToString(num\_arr);

public static String arrayToString(int[] num\_arr){

for(int x = 0;  $x < num_arr.length$ ; x++){

sb.append(num\_arr[x] + SPACE\_SEPARATOR);

StringBuilder sb = new StringBuilder();

sb.append(num\_arr[x]);

public static void mySortArray(int[] num arr){

public static int[] toIntArray(String[] str\_arr){

int[] arr = new int[str\_arr.length];

for(int i = 0; i < arr.length;  $i++){$ 

arr[i] = Integer.parseInt(str\_arr[i]);

public static String[] stringToArray(String numStr){

String[] str\_arr = numStr.split(SPACE\_SEPARATOR);

6.2 集合类

对象用于封装特有数据,对象多了需要存储;如果对象的个数不确定,就使用集合容器进行存储。

- - X

Map

TreeMap

LinkedHashMap

Utilities

Collections

Arrays

Produces

TreeSet

HashMap

Set

Comparator

HashSet

LinkedHashSet

该对象必须依赖于具体容器,因为每一个容器的数据结构都不同,所以该迭代器对象是在容器中进行

对于使用容器者而言,具体的实现不重要,只要通过容器获取到该实现的迭代器的对象即可,也就是

if(x!=num\_arr.length - 1)

D:\code\day16>javac WrapperDemo.java

public class WrapperDemo{

public static void main(String[] args){

System.out.println(Integer.toBinaryString(60));

System.out.println(Integer.toOctalString(60)); System.out.println(Integer.toHexString(60));

System.out.println(Integer.toString(60,4));

02. 03.

06.

07. 08. 09.

整数具备不同的进制体现。 十进制-->其他进制方法: String to Binary String (int i); String toOctalString(int i); String to Hex String (int i);

示例4:

01.

03.

04.

05. 06. 07. 08. 复制代码 运行结果:

01. public class WrapperDemo{ 02. 03. 05. 06. }

运行结果:

110 60

需求:

代码:

"20 78 9 -7 88 36 29"

}

复制代码

📰 管理员: C:\Windows\system32\cmd.exe

D:∖code∖day16>java WrapperDemo

D:\code\day16>javac WrapperDemo.java

对一个字符串中的数值进行从小到大的排序。

示例5:

import java.util.Arrays; 01. 02. public class WrapperTest{ 04. 05.

07.

08.

10.

11.

12.

13.

14.

16.

17.

18.

19.

20.

}

21. 22. 23. 24. 25. 26.

27.

28.

29.

30.

31.

33.

34.

}

35. 36. 37. 38. 39. 40.

41.

42.

43.

45.

46.

47.

48.

49. 50. 51. 52. 53. 54.

56.

57.

58.

59.

运行结果:

}

}

复制代码

画 管理员: C:\Windows\system32\cmd.exe

D:\code\day16>javac WrapperTest.java D:\code\day16>java WrapperTest 20 79 9 -7 88 36 29 -7 9 20 29 36 79 88 集合类的由来:

集合特点:

1. 用于存储对象的容器。

2. 集合的长度是可变的。

3. 集合中不可以存储基本数据类型值。

不断的向上抽取,就形成了集合框架。

数组和集合类同是容器,有何不同?

ArrayList

6.2.1 Collection接口

框架的顶层Collection接口:

boolean remove(Object obj);

boolean contains(Object obj);

boolean containsAll(Collection coll);

boolean is Empty();判断集合中是否有元素。

内部实现的,也就是iterator方法在每个容器中的实现方式是不同的。

Iterator接口就是对所有的Collection容器进行元素取出的公共接口。

boolean removeAll(Collection coll);

集合容器因为内部的数据结构不同,有多种具体容器。

数组虽然也可以存储对象,但长度是固定的;集合长度是可变的。

Produces

Produces

Comparable

LinkedList

List

数组中可以存储基本数据类型,集合只能存储对象。 集合框架的构成及分类: Iterator ListIterator

P.S.

Collection的常见方法: 1、添加: boolean add(Object obj); boolean addAll(Collection coll); 2、删除:

void clear();

3、判断:

4、获取:

int size();

iterator方法。

Iterator iterator();

取出元素的方式:迭代器。

5、其他: boolean retainAll(Collection coll);取交集 Object to Array();将集合转成数组 示例1:

import java.util.\*;

public class CollectionDemo{

show(coll);

show(c1,c2);

public static void main(String[] args){

Collection coll = new ArrayList();

Collection c1 = new ArrayList();

Collection c2 = new ArrayList();

public static void show(Collection coll){

System.out.println( "coll:" + coll);

System.out.println( "coll:" + coll);

coll.remove( "abc2");//会改变集合的长度

System.out.println(coll.contains( "abc1"));

public static void show(Collection c1,Collection c2){

//2、删除元素,remove

//1、添加元素 , add

coll.add( "abc1");

coll.add( "abc2");

coll.add( "abc3");

//清空集合

//coll.clear();

//给c1添加元素

c1.add( "abc1");

c1.add( "abc2");

c1.add( "abc3");

c1.add( "abc4");

//给c2添加元素

c2.add( "abc2");

c2.add( "abc6");

c2.add( "abc7");

//演示addAll

c1.addAll(c2);

//演示removeAll

//演示containsAll

//演示retainAll

System.out.println( "c1:" + c1);

System.out.println( "c2:" + c2);

//将c2中的元素添加到c1中

//从c1集合中删除与c2集合相同的元素

System.out.println( "removeAll:" + b);

boolean b = c1.removeAll(c2);

boolean b1 = c1.containsAll(c2);

System.out.println( "containsAll:" + b1);

//取交集,保留和指定的集合相同的元素

System.out.println("c1、c2交集:" + c1);

\_ \_ \_ X

boolean b2 = c1.retainAll(c2);

public static void main(String[] args){

coll.add( "abc1");

coll.add( "abc2");

coll.add( "abc3");

coll.add( "abc4");

System.out.println(coll);

Iterator it1 = coll.iterator();

System.out.println(it1.next());

System.out.println(it2.next());

//for循环结束,Iterator变量内存释放,更高效

for(Iterator it2 = coll.iterator();it2.hasNext();){

while(it1.hasNext()){

//使用了Collection中的iterator()方法。调用集合中的迭代器方法,是为了获取集合中

Collection coll = new ArrayList();

System.out.println( "-----" );

01.

02.

03.

04.

05.

06.

07.

08.

09.

10.

11.

12.

13.

14.

15.

16.

17.

18.

19.

20.

}

21. 22. 23. 24. 25. 27. 28.

30.

31.

32.

33.

34.

35.

36.

37.

38.

39.

40.

41.

42.

43.

}

44. 45. 46. 47. 48. 49. 50. 51.

53.

54.

55.

56.

57.

58.

59.

60.

61.

62. 63. 65.

import java.util.ArrayList; 01. import java.util.Collection; 02. import java.util.Iterator; 03. 04. 05. public class IteratorDemo{

示例2:

17. 18.

}

}

复制代码 运行结果: 📷 管理员: C:\Windows\system32\cmd.exe D:\code\day16>javac CollectionDemo.java 注意: CollectionDemo.java 使用了未经检查或不安全的 注意: 要了解详细信息,请使用 -Xlint:unchecked 重新 D:\code\day16>java CollectionDemo coll:[abc1, abc2, abc3] coll:[abc1, abc3] true

06. 08. 09. 10. 11. 12. 13. 14. 15. 16. 的迭代器对象。

c1:[abc1, abc2, abc3, abc4] c2:[abc2, abc6, abc7] removeAll:true containsAll:false :1、c2交集:[]

运行结果:

19. 20. 21. 22. 23. 24. 25. 27. } 28.

复制代码

```
_ D X
📷 管理员: C:\Windows\system32\cmd.exe
D: \code\day16>javac IteratorDemo.java
注意: IteratorDemo.java 使用了未经检查或不安全的接
注意: 要了解详细信息,请使用 -Xlint:unchecked 重新
D:\code\day16>java IteratorDemo
[abc1, abc2, abc3, abc4]
abc1
abc2
abc3
abc1
abc2
abc3
 6.2.2 List, Set
 Collection
    |--List:有序(存入和取出的顺序一致),元素都有索引(角标),允许重复元素。
    |--Set:元素不能重复,无序。
 List:特有的常见方法。
 有一个共性特点就是都可以操作角标。
 1、添加
    void add(index,element);
    void addAll(index,collection);
```

2、删除 Object remove(index);

4、获取:

Object get(index);

int indexOf(object);

int lastIndexOf(object);

List subList(from,to);

3、修改 Object set(index,element);

List集合可以完成对元素的增删改查。 示例1: import java.util.ArrayList; 01.

import java.util.List; 02. 03. public class ListDemo{ 04. 06.

public static void main(String[] args){ List list = new ArrayList(); show(list); 07. 08. } 09. 10. //添加元素 11.

list.add( "abc1" ); 12. list.add( "abc2" ); 13. list.add( "abc3" ); 14. 15.

System.out.println(list); 16.

public static void show(List list){

17. //插入元素 18. list.add(1, "abc2"); 19. 20. //删除元素 21. System.out.println( "remove:" + list.remove(2)); 22. 23.

//修改元素 24. System.out.println( "set:" + list.set(1,"abc8" )); 25. 26.

//获取元素: //获取子列表

27. 28. 29. 30. 32. 33. 34. }

35. } 复制代码

运行结果:

System.out.println( "get:" + list.get(0)); 管理员: C:\Windows\system32\cmd.exe

System.out.println( "sublist:" + list.subList(1,2)); System.out.println(list);

D:\code\day16>javac ListDemo.java 注意: ListDemo.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\day16>java ListDemo [abc1, abc2, abc3] remove:abc2 set:abc2 get:abc1

sublist:[abc8] [abc1, abc8, abc3]

- - X

\_ 0 X

示例2: 01. import java.util.Iterator; 02. import java.util.List; import java.util.ListIterator;

import java.util.ArrayList;

public class ListDemo{ show(list); }

03. 04. 05. 06. public static void main(String[] args){ 07. List list = new ArrayList(); 08. 09. 10. 11. public static void show(List list){ 12. list.add( "abc1"); 13. list.add( "abc2"); 14. list.add( "abc3"); 15. list.add( "abc4"); 16. 17. Iterator it = list.iterator(); 18. 19.

while(it.hasNext()){ System.out.println( "next:" + it.next()); } //list特有的取出元素的方式之一 for(int x = 0; x < list.size();  $x++){$ 

20. 21. 22. 23. System.out.println( "get:" + list.get(x)); 25. 26. } } 27. } 28. 复制代码 运行结果:

画 管理员: C:\Windows\system32\cmd.exe D:\code\day16>javac ListDemo.java 注意: ListDemo.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\day16>java ListDemo next:abc1 next:abc2 next:abc3 next:abc4 get:abc1 get:abc2 get:abc3 get:abc4 示例3:

import java.util.ArrayList; 01. import java.util.Iterator; 02. import java.util.List; 03. 04.

public class ListDemo{ public static void main(String[] args){ List list = new ArrayList();

05. 06. 07. 08. list.add( "abc1"); 09. list.add( "abc2"); 10. list.add( "abc3"); 11. System.out.println( "list:" + list); 13.

14. Iterator it = list.iterator(); 15. 16. while(it.hasNext()){ 17. Object obj = it.next(); 19. if(obj.equals("abc2")){ 20. list.add( "abc9"); 21. } else{ 22. System.out.println( "next:" + obj); 23.

25.

26.

27.

28.

复制代码

运行结果: 画 管理员: C:∖Windows\system32\cmd.exe D:\code\day16>javac ListDemo.java 注意: ListDemo.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\day16>java ListDemo list:[abc1, abc2, abc3] next:abc1 [abc1, abc2, abc3] [abc1, abc2, abc3, abc9] Exception in thread "main" java.util.ConcurrentModificationException at java.util.AbstractList\$Itr.checkForComodification(AbstractList.java:3 72)

at ListDemo.main(ListDemo.java:18)

在迭代器过程中,不要使用集合操作元素,容易出现异常:

System.out.println(list);

at java.util.AbstractList\$Itr.next(AbstractList.java:343)

可以使用Iterator接口的子接口ListIterator来完成在迭代中对元素进行更多的操作。

ListIterator it = list.listIterator(); //获取列表迭代器对象

//它可以实现在迭代过程中完成对元素的增删改查。

//注意:只有list集合具备该迭代功能。

Object obj = it.next();

if(obj.equals("abc3" )){

it.add( "abc9");

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

~END~

~爱上海,爱黑马~

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

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

\_ D X

while(it.hasNext()){

while(it.hasPrevious()){

D:\code\day16>javac ListDemo.java 注意: ListDemo.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。

System.out.println( "list:" + list);

}

}

示例4: import java.util.ArrayList; import java.util.List; 02. import java.util.ListIterator; 03. 04. public class ListDemo{ 05. public static void main(String[] args){ 06. List list = new ArrayList(); 08. 09. list.add( "abc1"); list.add( "abc2"); 10. list.add( "abc3"); 11. 12. 13. System.out.println( "list:" + list);

java.util. Concurrent Modification Exception.

21. 23. 24. 25. 26. 27. 28. 30. 31. 32. 33. 34. 复制代码

运行结果:

📆 管理员: C:\Windows\system32\cmd.exe

D:\code\day16>java ListDemo

14.

16.

17. 18.

19.

20.

list:[abc1, abc2, abc3] hasNext:false hasPrevious:true previous:abc9 previous:abc3 previous:abc2 previous:abc1 list:[abc1, abc2, abc3, abc9]