

非贷款，0元入学，不1万就业不给1分钱学费，我们已于四年了！

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

6. 集合

6.1 常用对象

6.1.2 基本数据类型对象包装类

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作。

用于描述该对象的类就称为基本数据类型对象包装类。

```
byte    Byte
short   Short
int      Integer
long    Long
float    Float
double  Double
char     Character
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

1. 基本类型数值+""
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.parseInt("xxx");

```
int parseInt(String str);

long parseLong(String str);

boolean parseBoolean(String str);
```

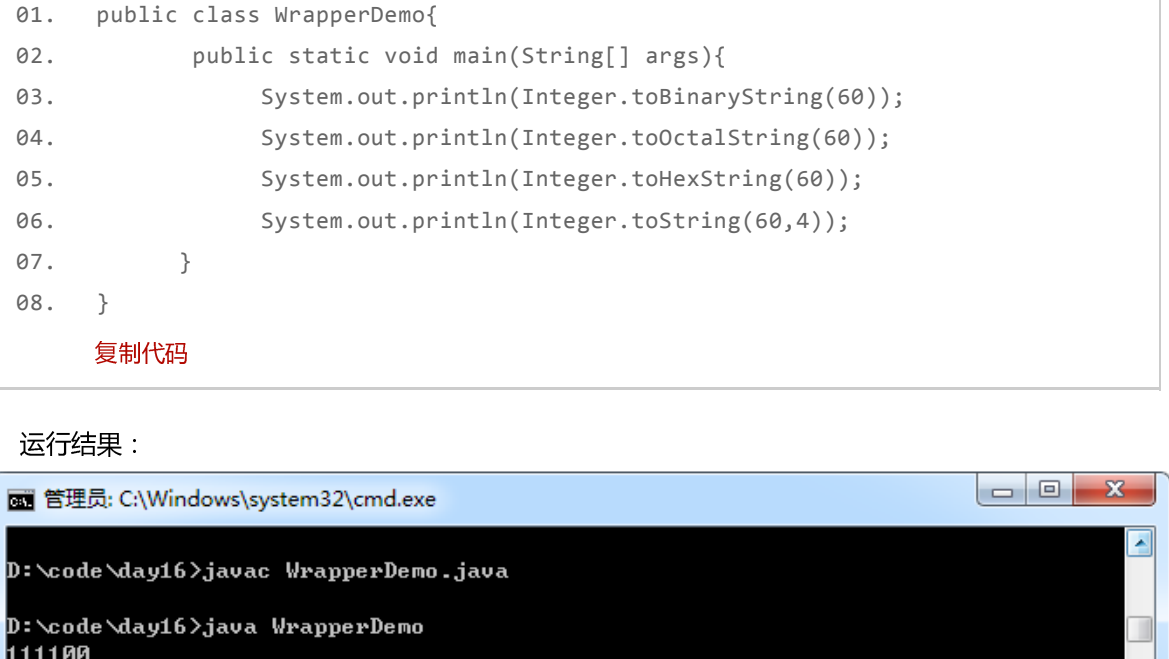
只有Character没有parse方法。
2. 如果字符串被Integer进行对象的封装，可使用另一个非静态的方法，intValue，
将一个Integer对象转换成基本数据类型值。
3. 用Integer的静态方法valueOf(String s);

示例1：

```
01. public class WrapperDemo{
02.     public static void main(String[] args){
03.         // 整型最大值
04.         System.out.println(Integer.MAX_VALUE);
05.         // 十进制转换成二进制
06.         System.out.println(Integer.toBinaryString(6));
07.     }
08. }
```

复制代码

运行结果：



```
D:\code>java WrapperDemo
2147483647
110
```

示例2：

```
01. public class WrapperDemo{
02.     public static void main(String[] args){
03.         Integer i = new Integer(5);
04.         System.out.println(Integer.parseInt("123") + 1);
05.     }
06. }
```

复制代码

运行结果：



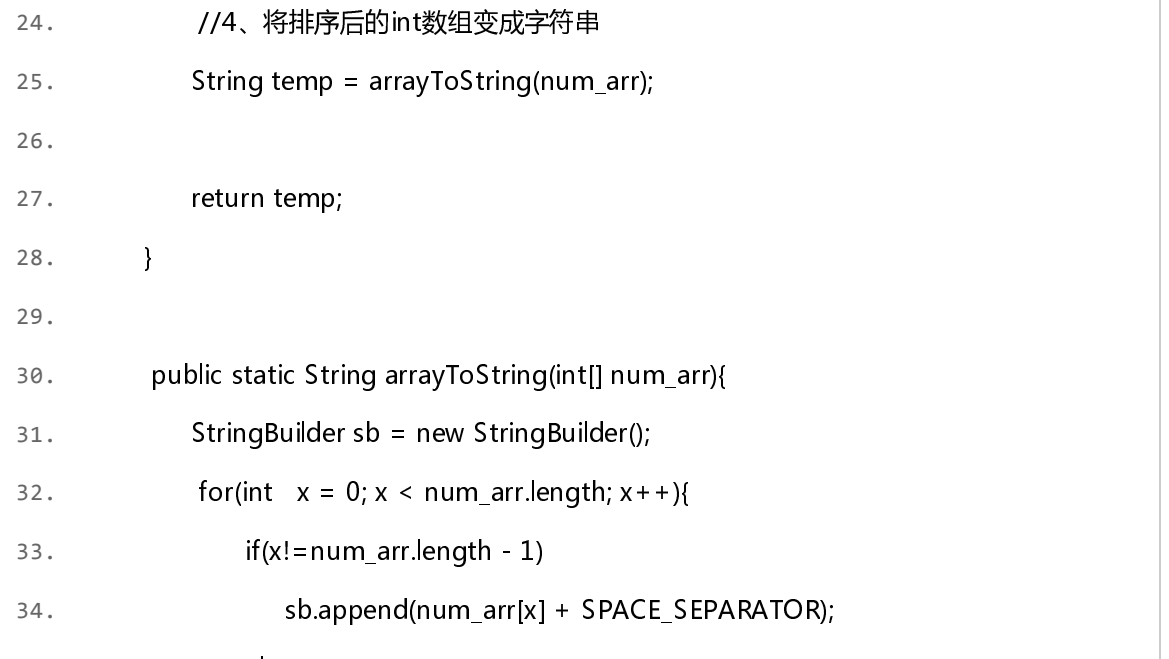
```
D:\code>java WrapperDemo
124
```

示例3：

```
01. public class WrapperDemo{
02.     public static void main(String[] args){
03.         if(Boolean.parseBoolean("true"))
04.             System.out.println("true");
05.
06.         Integer i = new Integer("123");
07.         System.out.println(i.intValue());
08.     }
09. }
```

复制代码

运行结果：



```
D:\code>java WrapperDemo
true
123
```

整数具备不同的进制体现。

十进制-->其他进制方法：

String toBinaryString(int i);

String toOctalString(int i);

String toHexString(int i);

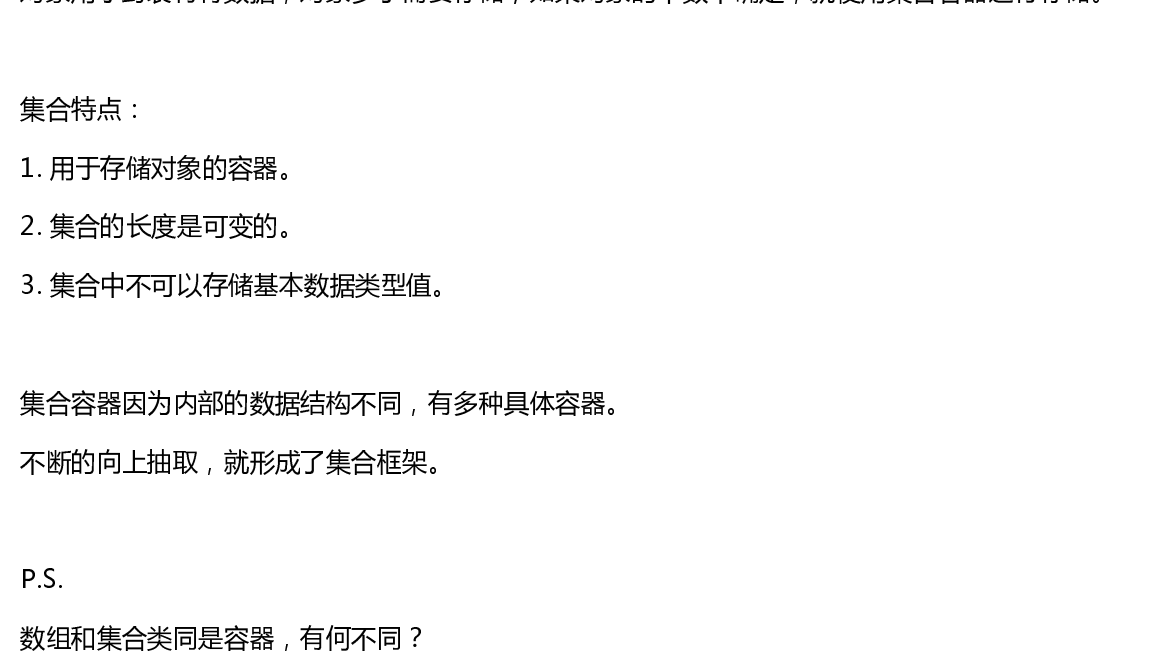
String toString(int i,int radix);

示例4：

```
01. public class WrapperDemo{
02.     public static void main(String[] args){
03.         System.out.println(Integer.toBinaryString(60));
04.         System.out.println(Integer.toOctalString(60));
05.         System.out.println(Integer.toHexString(60));
06.     }
07. }
```

复制代码

运行结果：



```
D:\code>java WrapperDemo
11100
74
3c
330
```

其他进制-->十进制方法：

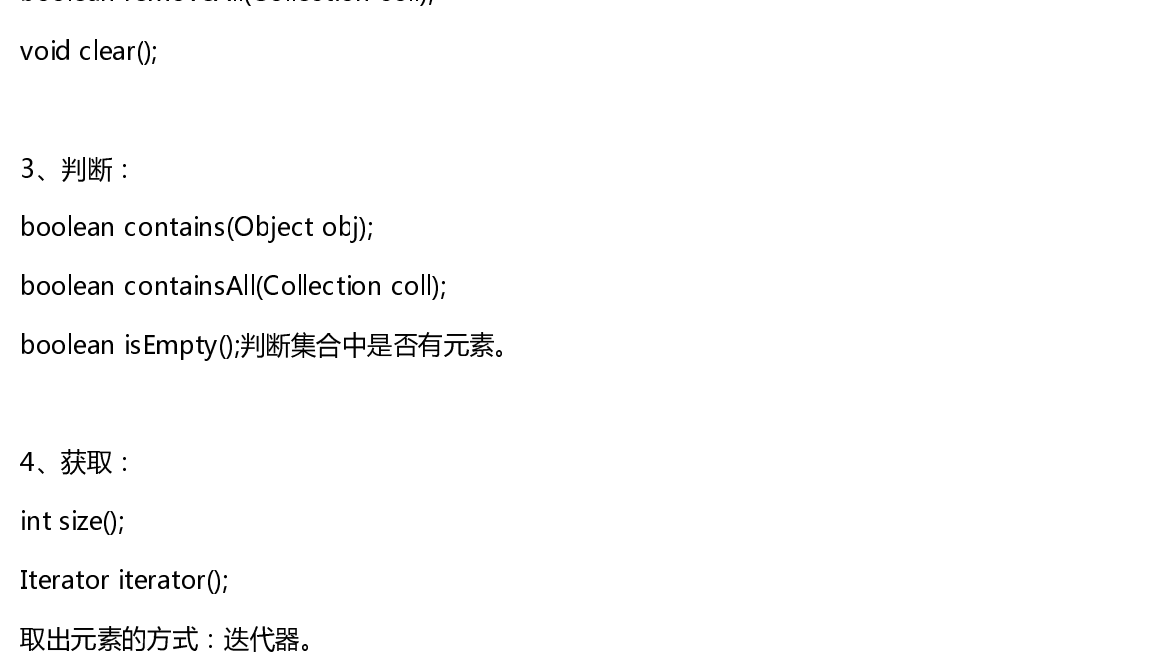
int parseInt(String s,int radix);

示例5：

```
01. public class WrapperDemo{
02.     public static void main(String[] args){
03.         System.out.println(Integer.parseInt("110",10));
04.         System.out.println(Integer.parseInt("3C",16));
05.     }
06. }
```

复制代码

运行结果：



```
D:\code>java WrapperDemo
110
30
```

需求：

对一个字符串中的数值进行从小到大的排序。

'20 78 9 -7 88 36 29'

代码：

```
01. import java.util.Arrays;
02.
03. public class WrapperTest{
04.
05.     private static final String SPACE_SEPARATOR = " ";
06.
07.     public static void main(String[] args){
08.         String numStr = "20 79 9 -7 88 36 29";
09.         System.out.println(numStr);
10.         numStr = sortStringNumber(numStr);
11.         System.out.println(numStr);
12.     }
13. }
```

```
14. public static String sortStringNumber(String numStr){
15.     //1、将字符串变成字符串数组
16.     String[] str_arr = stringToArray(numStr);
17.
18.     //2、将字符串数组变成int数组
19.     int[] num_arr = toIntArray(str_arr);
20.
21.     //3、对int数组排序
22.     mySortArray(num_arr);
23.
24.     //4、将排序后的int数组变成字符串
25.     String temp = arrayToString(num_arr);
26.
27.     return temp;
28. }
```

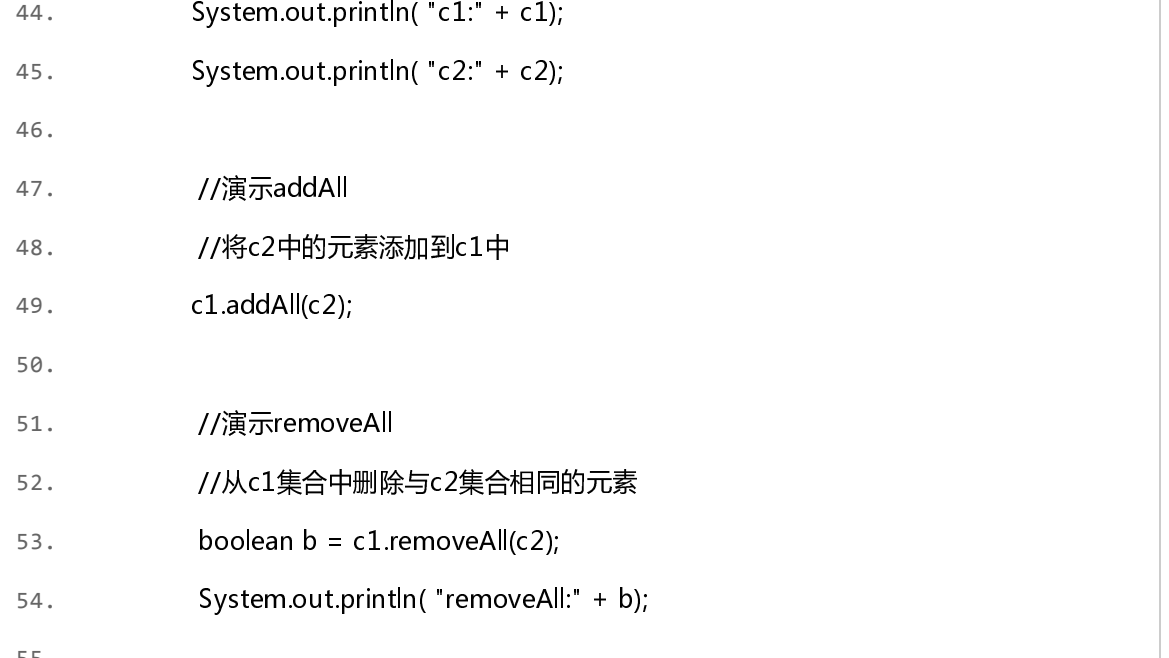
```
29.
30. public static String arrayToString(int[] num_arr){
31.     StringBuilder sb = new StringBuilder();
32.     for(int x = 0; x < num_arr.length; x++){
33.         if(x!=num_arr.length - 1)
34.             sb.append(num_arr[x] + SPACE_SEPARATOR);
35.         else
36.             sb.append(num_arr[x]);
37.     }
38.     return sb.toString();
39. }
```

```
40.
41. public static void mySortArray(int[] num_arr){
42.     Arrays.sort(num_arr);
43. }
44.
45. public static int[] toIntArray(String[] str_arr){
46.     int[] arr = new int[str_arr.length];
47.
48.     for(int i = 0; i < arr.length; i++){
49.         arr[i] = Integer.parseInt(str_arr[i]);
50.     }
51.
52.     return arr;
53. }
```

```
54.
55. public static String[] stringToArray(String numStr){
56.     String[] str_arr = numStr.split(SPACE_SEPARATOR);
57.     return str_arr;
58. }
59. }
```

复制代码

运行结果：



```
D:\code>java WrapperTest
20 79 9 -7 88 36 29
-7 9 28 29 36 79 88
```

6.2 集合类

集合类的由来：

对象用于封装带有数据，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。

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。

集合容器因为内部的数据结构不同，有多种具体容器。

不断的向上抽取，就形成了集合框架。

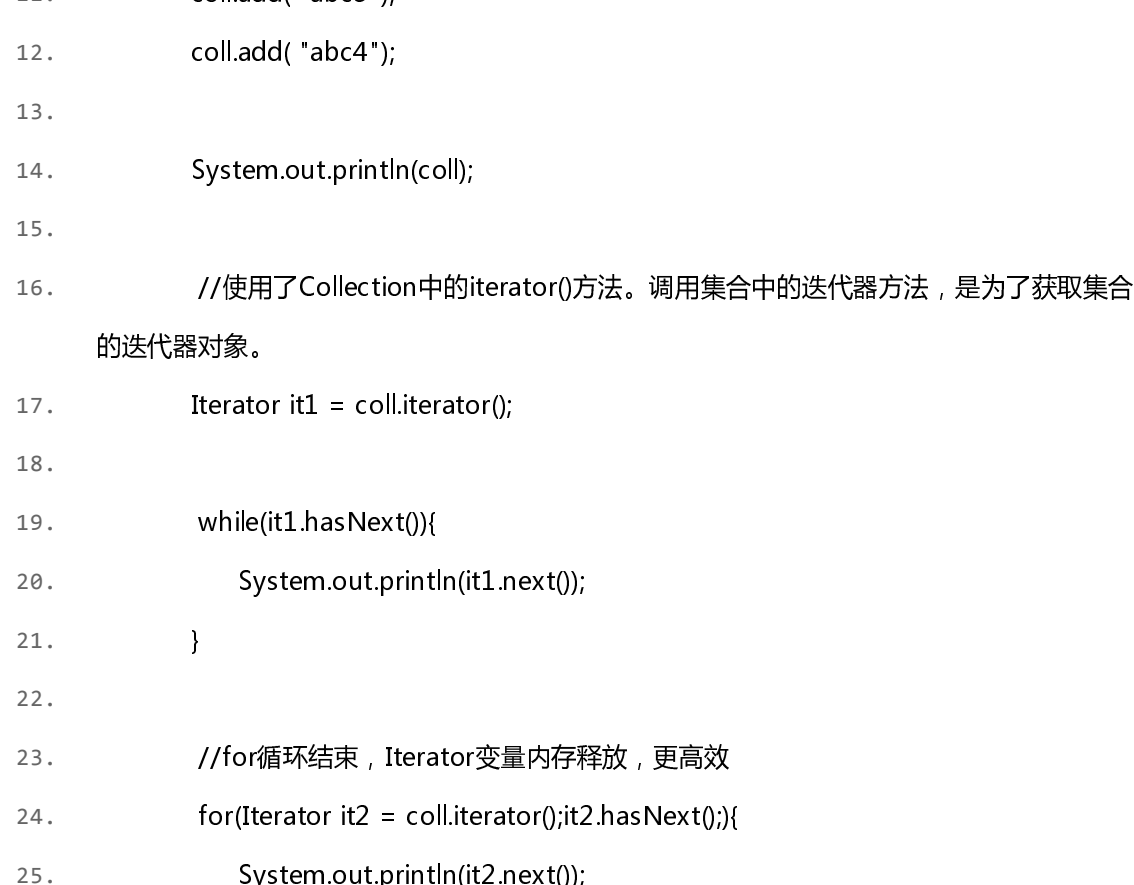
P.S.

数组和集合类同是容器，有何不同？

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。

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：



6.2.1 Collection接口

框架的顶层Collection接口：

Collection的常见方法：

- 1、添加：

```
boolean add(Object obj);
boolean addAll(Collection coll);
```

- 2、删除：

```
boolean remove(Object obj);
boolean removeAll(Collection coll);
void clear();
```

- 3、判断：

```
boolean contains(Object obj);
boolean containsAll(Collection coll);
boolean isEmpty();判断集合中是否有元素。
```

- 4、获取：

```
int size();
Iterator iterator();
```

取出元素的方式：迭代器。

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。

对于使用者而言，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Iterator方法。

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

- 5、其他：

```
boolean retainAll(Collection coll);取交集
Object toArray();将集合转成数组
```

示例1：

```
01. import java.util.*;
02.
03. public class CollectionDemo{
04.
05.     public static void main(String[] args){
06.         Collection coll = new ArrayList();
07.         show(coll);
08.
09.         System.out.println("-----");
10.
11.         Collection c1 = new ArrayList();
12.         Collection c2 = new ArrayList();
13.         show(c1,c2);
14.     }
15. }
```

```
16. public static void show(Collection coll){
17.     //1、添加元素，add
18.     coll.add("abc1");
19.     coll.add("abc2");
20.     coll.add("abc3");
21.     System.out.println("coll: " + coll);
22.
23.     //2、删除元素，remove
24.     coll.remove("abc2");//会改变集合的长度
25.     System.out.println("coll: " + coll);
26.
27.     //清空集合
28.     coll.clear();
29.     System.out.println(coll.contains("abc1"));
30. }
```

```
31.
32. public static void show(Collection c1,Collection c2){
33.     //给c1添加元素
34.     c1.add("abc1");
35.     c1.add("abc2");
36.     c1.add("abc3");
37.     c1.add("abc4");
38.
39.     //给c2添加元素
40.     c2.add("abc2");
41.     c2.add("abc6");
42.     c2.add("abc7");
43.
44.     System.out.println("c1: " + c1);
45.     System.out.println("c2: " + c2);
46.
47.     //演示addAll
48.     //将c2中的元素添加到c1中
49.     c1.addAll(c2);
50.
51.     //演示removeAll
52.     //从c1集合中删除与c2集合相同的元素
53.     boolean b = c1.removeAll(c2);
54.     System.out.println("removeAll: " + b);
55.
56.     //演示containsAll
57.     boolean b1 = c1.containsAll(c2);
58.     System.out.println("containsAll: " + b1);
59.
60.     //演示retainAll
61.     //取交集，保留指定的集合相同的元素
62.     boolean b2 = c1.retainAll(c2);
63.     System.out.println("c1、c2交集: " + c1);
64. }
65. }
```

复制代码

运行结果：


```
D:\code>java CollectionDemo
coll:[abc1, abc2, abc3]
true
-----
c1:[abc1, abc2, abc3, abc4]
c2:[abc2, abc6, abc7]
removeAll:true
containsAll:false
c1、c2交集:[]
```

示例2：

```
01. import java.util.ArrayList;
02. import java.util.Collection;
03. import java.util.Iterator;
04.
05. public class IteratorDemo{
06.     public static void main(String[] args){
07.         Collection coll = new ArrayList();
08.
09.         coll.add("abc1");
10.         coll.add("abc2");
11.         coll.add("abc3");
12.         coll.add("abc4");
13.
14.         System.out.println(coll);
15.
16.         //使用了Collection中的iterator()方法。调用集合中的迭代器方法，是为了获取集合中的迭代器对象。
17.         Iterator it1 = coll.iterator();
18.
19.         while(it1.hasNext()){
20.             System.out.println(it1.next());
21.         }
22.
23.         //for循环结束，Iterator变量内存释放，更高效
24.         for(Iterator it2 = coll.iterator();it2.hasNext()){
25.             System.out.println(it2.next());
26.         }
27.     }
28. }
```

复制代码

运行结果：


```
D:\code>java IteratorDemo
[abc1, abc2, abc3, abc4]
abc1
abc2
abc3
abc4
```




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);
```

3、修改

```
Object set(index,element);
```

4、获取：

```
Object get(index);  
int indexOf(object);  
int lastIndexOf(object);  
List subList(from,to);
```

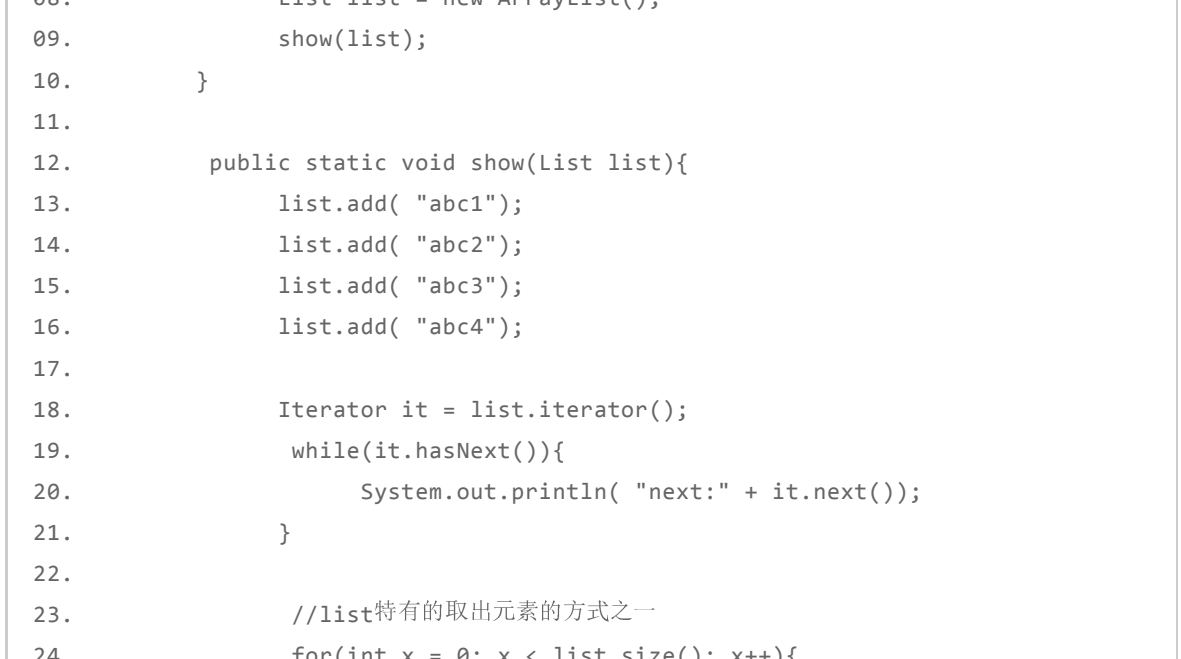
List集合可以完成对元素的增删改查。

示例1：

```
01. import java.util.ArrayList;  
02. import java.util.List;  
03.  
04. public class ListDemo{  
05.     public static void main(String[] args){  
06.         List list = new ArrayList();  
07.         show(list);  
08.     }  
09.  
10.     public static void show(List list){  
11.         //添加元素  
12.         list.add("abc1");  
13.         list.add("abc2");  
14.         list.add("abc3");  
15.  
16.         System.out.println(list);  
17.  
18.         //插入元素  
19.         list.add(1,"abc2");  
20.  
21.         //删除元素  
22.         System.out.println("remove:" + list.remove(2));  
23.  
24.         //修改元素  
25.         System.out.println("set:" + list.set(1,"abc8"));  
26.  
27.         //获取元素：  
28.         System.out.println("get:" + list.get(0));  
29.  
30.         //获取子列表  
31.         System.out.println("sublist:" + list.subList(1,2));  
32.  
33.         System.out.println(list);  
34.     }  
35. }
```

复制代码

运行结果：



示例2：

```
01. import java.util.ArrayList;  
02. import java.util.Iterator;  
03. import java.util.List;  
04. import java.util.ListIterator;  
05.  
06. public class ListDemo{  
07.     public static void main(String[] args){  
08.         List list = new ArrayList();  
09.         show(list);  
10.     }  
11.  
12.     public static void show(List list){  
13.         list.add("abc1");  
14.         list.add("abc2");  
15.         list.add("abc3");  
16.         list.add("abc4");  
17.  
18.         Iterator it = list.iterator();  
19.         while(it.hasNext()){  
20.             System.out.println("next:" + it.next());  
21.         }  
22.  
23.         //list特有的取出元素的方式之一  
24.         for(int x = 0; x < list.size(); x++){  
25.             System.out.println("get:" + list.get(x));  
26.         }  
27.     }  
28. }
```

复制代码

运行结果：



示例3：

```
01. import java.util.ArrayList;  
02. import java.util.Iterator;  
03. import java.util.List;  
04.  
05. public class ListDemo{  
06.     public static void main(String[] args){  
07.         List list = new ArrayList();  
08.  
09.         list.add("abc1");  
10.         list.add("abc2");  
11.         list.add("abc3");  
12.  
13.         System.out.println("list:" + list);  
14.  
15.         Iterator it = list.iterator();  
16.  
17.         while(it.hasNext()){  
18.             Object obj = it.next();  
19.  
20.             if(obj.equals("abc2")){  
21.                 list.add("abc9");  
22.             }else{  
23.                 System.out.println("next:" + obj);  
24.             }  
25.  
26.             System.out.println("hasNext:" + it.hasNext());  
27.             System.out.println("hasPrevious:" + it.hasPrevious());  
28.         }  
29.     }  
30. }
```

复制代码

运行结果：



~END~



~爱上海，爱黑马~

