

一、Arraylist动态数组

java.util.arraylist

如果存储的是一系列主类型，最好使用数组，但如果存储的是对象，最好使用集合。下面介绍Arraylist，是一种常用的集合。Arraylist是一种动态数组类型，Arraylist对象既有数组特征，也有链表特征。其实实现了list接口。

(一) 构造方法

构造方法摘要

<code>ArrayList()</code>	构造一个初始容量为 10 的空列表。
<code>ArrayList(Collection<? extends E> c)</code>	构造一个包含指定 collection 的元素的列表，这些元素是按照该 collection 的迭代器返回它们的顺序排列的。
<code>ArrayList(int initialCapacity)</code>	构造一个具有指定初始容量的空列表。

E表示集合中元素的类型

例如：创建一个空的数组链表，用来存放String类型的对象：

```
ArrayList<String> list = new ArrayList<String>();
```

创建一个指定初始容量的数组链表：

```
ArrayList<Integer> list = new ArrayList<Integer>(7);
```

注意：ArrayList对象只能**存放对象**，不能存放基础数据类型的数据。

(二) 常用方法

参见API

```
ArrayList<people> peoples = new ArrayList<people>(); //people是一个类，peoples是包含很多people类的对象的动态数组
```

常用方法	使用方法	用途
add	peoples.add()	将指定的元素添加到此列表的尾部（可以用于初始化ArrayList动态数组）
remove(int index)	peoples.remove()	移除动态数组该位置的对象
get(int index)	peoples.get()	返回此列表中指定位置上的元素(这样打出来的是对象的地址，那怎么才能访问到内部的内容呢？？)=>请看迭代器的妙用

例如：

```
public class Test{
    private ArrayList<People> peoples = new ArrayList<People>();
}
```

```

public Test(){           //在构造方法中进行初始化，把一个个的对象加上去
    People p = new People("关羽",80,90,90);
    peoples.add(p);
    p = new People("张飞",70,90,80);
    peoples.add(p);
    p = new People("赵云",85,95,90);
    peoples.add(p);
    p = new People("黄忠",85,80,70);
    peoples.add(p);
    p = new People("刘备",85,80,75);
    peoples.add(p);
}

```

(三) 遍历方法

方法一：神奇的迭代器

java.util.iterator

```

Iterator<People> it = peoples.iterator();//这里不是使用new哦！！
while(it.hasNext()){
    People p = it.next();//it往后移动一个单位
    System.out.println("三国排名【"+p.getRank()+"】："+p.getName()+" 智慧
("+p.getZhihui()+"),攻击("+p.getGongji()+"),防御("+p.getFangyu()+"");
    //迭代器直接println出来的是地址，对象直接直接println出来的是也是句柄地址，需要深入到属性
    才可以正常打印
}

```

迭代器是一种iterator接口，Iterator，它总是用同一种逻辑来遍历集合。使得客户端自身不需要来维护集合的内部结构，所有的内部状态都由Iterator来维护。客户端不用直接和集合进行打交道，而是控制Iterator向它发送向前向后的指令，就可以遍历集合。（实现遍历的大统一ArrayList、LinkedList、HashSet...）

它有三种方法：

boolean	hasNext()	如果仍有元素可以迭代，则返回 true。
E	next()	返回迭代的下一个元素。
void	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）。

Iterator.next() returns the **value** of the next collection element and moves forward to *just after* the next element.

- calling next() when hasNext() is false results in a NoSuchElementException

使用next () 同时会移动迭代器的指



向。

?? 注意：在使用Iterator的时候禁止对所遍历的容器进行改变其大小结构的操作。例如：在使用Iterator进行迭代时，如果对集合进行了add、remove操作就会出现ConcurrentModificationException异常。

(remove不太会用??)

方法二：使用for循环

```
for (int i=0; i<peoples.size(); i++) {  
    People p = list.get(i);  
}
```

举例：迭代器学习

```
import java.util.*;  
  
public class ALTest{  
    public static void main(String[] args){  
        //ArrayList al = new ArrayList(); 这是不安全操作  
        ArrayList<String> al = new ArrayList<String>();  
        al.add("Java");  
        al.add("C++");  
        al.add("Pascal");  
        al.add("Perl");  
        //Iterator it = al.iterator();  
        Iterator<String> it = al.iterator();  
        while(it.hasNext()){  
            //String str = (String)it.next();  
            String str = it.next();  
            System.out.println("Item :"+str);  
        }  
        for(it=al.iterator();it.hasNext();){  
            String str = it.next();  
            //String str = (String)it.next();  
            System.out.println("remove :"+str);  
            it.remove(); //使用迭代器的remove不能指定位置  
        }  
    }  
}
```

运行结果：

```
Item :Java
Item :C++
Item :Pascal
Item :Perl
remove :Java
remove :C++
remove :Pascal
remove :Perl
```

```
import java.util.*;

public class ALTest1{
    public static void main(String[] args){
        ArrayList<String> al = new ArrayList<String>();
        al.add("Java");
        al.add("C++");
        al.add("Pascal");
        al.add("Perl");
        for(int i=0;i<al.size();i++){//另一种遍历方式
            String str = (String)al.get(i);
            System.out.println("Item :"+str);
        }

        for(int i=0;i<al.size();i++){
            String str = (String)al.get(i);
            System.out.println("remove :"+str);
            al.remove(i);    //使用ArrayList本身的remove可以指定位置
        }
    }
}
```

运行结果:

```
Item :Java
Item :C++
Item :Pascal
Item :Perl
remove :Java
remove :Pascal
[C++, Pascal, Perl]
remove :Pascal
[C++, Perl]
```

第一轮移除了java后，al动态数组变成了只包含“c++ pascal perl”的数组，随后remove (1)即移除第二个“pascal”，数组长度变成2，i=2不满足条件退出循环。

实验考试题

```
import java.util.*;

//请完成★★★中规定的内容
public class Test{
    private ArrayList<People> peoples = new ArrayList<People>();
```

```

public Test(){
    People p = new People("关羽",80,90,90);
    peoples.add(p);
    p = new People("张飞",70,90,80);
    peoples.add(p);
    p = new People("赵云",85,95,90);
    peoples.add(p);
    p = new People("黄忠",85,80,70);
    peoples.add(p);
    p = new People("刘备",85,80,75);
    peoples.add(p);
}

```

//按智慧+攻击+防御总和进行排序，将序号写入people对象中，排名第一是1，顺序增加
 //排序实现形式任意，具体查阅JDK API帮助

```

public void ranking(){
    //★★★实验者实现(计算每个武将的总指数)
    Iterator<People> it = peoples.iterator();
    int arrayRank [] = new int[5];
    int i = 0;
    while(it.hasNext()){
        People p = it.next();
        int a = Math.round(p.getZhihui() + p.getGongji() + p.getFangyu());
        p.setRank(a);
        arrayRank[i] = a;
        i++;
    }
    //★★★实验者实现方法(对武将总指数进行排序，并设置对应people对象的rank值)
    for(int m = 0 ; m < arrayRank.length-1; m++){
        {
            for(int n = 0; n < arrayRank.length-1-m; n++){
                {
                    int temp;
                    if(arrayRank[n] < arrayRank[n+1])
                    {
                        temp = arrayRank[n];
                        arrayRank[n] = arrayRank[n+1];
                        arrayRank[n+1] = temp;
                    }
                }
            }
        }
        Iterator<People> itt = peoples.iterator();
        while(itt.hasNext()){
            People p = itt.next();
            int x = 0;
            for(int j = 0 ; j < arrayRank.length; j++){
                {
                    if(arrayRank[j] == p.getRank())
                    {
                        x = j;
                        break;
                    }
                }
            }
            p.setRank(x+1);
        }
        // System.out.println(Arrays.toString(arrayRank));
    }
}

```

```
//输出武将清单
public void listRank(){
    Iterator<People> it = peoples.iterator();
    while(it.hasNext()){
        People p = it.next();
        System.out.println("三国排名【"+p.getRank()+"】: "+p.getName()+" 智慧("+p.getZhihui()+"),攻击("+p.getGongji()+"),防御("+p.getFangyu()+")");
    }

}

public static void main(String[] args){
    Test t = new Test();
    System.out.println("排序显示-----");
    t.ranking();
    t.listRank();
}
}
```