**每日作业卷**

**JavaSE第07天**

# 关卡1

## 训练案例1

### 训练知识点

1. Collection基本方法

### 训练描述

练习Collection的add,remove,clear,size方法

### 操作步骤描述

1. 创建Colection对象.Collection是接口.所以创建子类ArrayList对象
2. 往集合中添加对象元素
3. 删除元素
4. 获取集合大小
5. 清空集合
6. 输出集合内容

**import** java.util.ArrayList;

**public** **class** Test1\_1 {

**public** **static** **void** main(String[] args) {

ArrayList<String> arr = **new** ArrayList<>();

arr.add("a");

arr.add("b");

arr.add("c");

arr.remove("c");

System.***out***.println(arr.size());

arr.clear();

System.***out***.println(arr.toString());

}

}



## 训练案例2

### 训练知识点

1.Iterator迭代器概述

### 训练描述

请回答hasNext()和next()方法的作用

### 操作步骤描述

1. hasNext()方法：用来判断集合中是否有下一个元素可以迭代。如果返回true,说明可以迭代。
2. next()方法：用来返回迭代的下一个元素，并把指针向后移动一位。

## 训练案例3

### 训练知识点

1.Iterator迭代器使用

### 训练描述

往ArrayList添加以下元素”abc1”, ”abc2”, ”abc3”, ”abc4”.使用迭代器获取ArrayList集合中的元素

### 操作步骤描述

1. 创建集合对象
2. 往集合中存放元素
3. 获取容器的迭代器
4. 使用迭代器判断是否有下一个元素
5. 使用迭代器对象获取集合中的元素

**import** java.util.ArrayList;

**import** java.util.Iterator;

**public** **class** Test1\_3 {

**public** **static** **void** main(String[] args) {

ArrayList<String> array=**new** ArrayList<String>();

array.add("abc1");

array.add("abc2");

array.add("abc3");

array.add("abc4");

Iterator<String> itr = array.iterator();

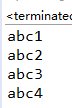
**while**(itr.hasNext()){

System.***out***.println(itr.next());

}

}

}



## 训练案例4

### 训练知识点

1.增强for循环

### 训练描述

往int[]添加以下元素{11, 22, 33}使用增强for遍历数组中的元素

### 操作步骤描述

1. 创建数组
2. 使用增强for循环获取数组中的元素

**public** **class** Test1\_4 {

**public** **static** **void** main(String[] args) {

**int**[] str={11, 22, 33};

**for**(**int** s:str){

System.***out***.println(s);

}

}

}



## 训练案例5

### 训练知识点

1.增强for循环

### 训练描述

往ArrayList添加以下元素” igeek”, ” igeek2”, ” igeek3”, ” igeek4”使用增强for遍历ArrayList中的元素

### 操作步骤描述

1. 创建ArrayList
2. 往ArrayList中添加元素
3. 使用增强for循环获取元素

**import** java.util.ArrayList;

**public** **class** Test1\_5 {

**public** **static** **void** main(String[] args) {

ArrayList<String> array=**new** ArrayList<String>();

array.add("igeek");

array.add("igeek2");

array.add("igeek3");

array.add("igeek4");

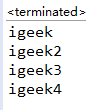
**for**(String s:array){

System.***out***.println(s);

}

}

}



## 训练案例6

### 训练知识点

1. 使用泛型的好处

### 训练描述

往ArrayList添加以下元素” abc”, ” igeek”, 5.练习ArrayList不加泛型和加泛型2种情况迭代出ArrayList中的元素.获取元素的长度.并说出泛型的好处

### 操作步骤描述

1. 定义method01方法
   1. 定义没有泛型的ArrayList
   2. 添加字符串
   3. 添加Integer类型
   4. 获取迭代器
   5. 判断是否有下一个元素
   6. 获取下一个元素,并强转为String类型
   7. 获取字符串的长度
2. 定义method02方法
   1. 定义有泛型的ArrayList
   2. 添加字符串
   3. 添加Integer类型
   4. 获取迭代器
   5. 判断是否有下一个元素
   6. 获取下一个元素,元素是String类型
   7. 获取字符串的长度
3. 调用这2个方法

**import** java.util.ArrayList;

**import** java.util.Iterator;

**public** **class** Test1\_6 {

**public** **static** **void** main(String[] args) {

*method01*();

System.***out***.println();

*method02*();

}

**public** **static** **void** method01(){

ArrayList array=**new** ArrayList();

array.add("abc");

array.add("igeek");

Integer count;

Iterator itr = array.iterator();

**while**(itr.hasNext()){

String a=(String)itr.next();

System.***out***.print(a.length()+" ");

}

}

**public** **static** **void** method02(){

ArrayList<String> array=**new** ArrayList<String>();

array.add("abc");

array.add("igeek");

Integer count;

Iterator<String> itr = array.iterator();

**while**(itr.hasNext()){

String a=(String)itr.next();

System.***out***.print(a.length()+" ");

}

}

}



## 训练案例7

### 训练知识点

1.泛型类

### 训练描述

自定义泛型类GenericClass<E>,包含E类型的成员变量,提供get/set方法,并使用这个泛型类

### 操作步骤描述

1. 自定义泛型类
2. 定义E类型的成员变量
3. 添加get/set方法
4. 创建泛型类对象.并指定具体类型
5. 调用set方法传入对应的类型
6. 调用get方法获取成员变量的值

**public** **class** Test1\_7 {

**public** **static** **void** main(String[] args) {

GenericClass<String> ge=**new** GenericClass<String>();

ge.setE("张三");

System.***out***.println(ge.getE());

}

}



## 训练案例8

### 训练知识点

1.泛型方法

### 训练描述

自定义泛型方法function.可以传入任意类型数据t.在这个方法中打印传入的t.并使用这个泛型方法

### 操作步骤描述

1. 定义泛型方法
2. 在泛型方法中,将传入的参数放到输出语句中
3. 调用function传入字符串
4. 调用function传入数字

**public** **class** function {

**public** **static** **void** main(String[] args) {

*function*("asda");

*function*(123);

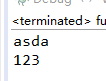
}

**public** **static** <T> **void** function(T t) {

System.***out***.println(t);

}

}



## 训练案例9

### 训练知识点

1.泛型接口

### 训练描述

自定义一个泛型接口Inter <E>,包含show(E e)抽象方法. 使用1.定义实现类时确定泛型的类型, 2.定义实现类时不确定泛型的类型2种方式定义实现类.并使用实现类

### 操作步骤描述

1. 定义泛型接口Inter<E>
2. 在Inter接口中定义抽象的show(E e)方法
3. 定义Imple01类.实现Inter接口,使用定义实现类时确定泛型的类型方式
4. 重写Imple01中的show(String e)方法.打印传入的内容
5. 定义Imple02类.实现Inter接口,定义实现类时不确定泛型的类型
6. 重写Imple02类中的show(E e)方法.打印传入的内容
7. 在测试类中创建Imple01对象
8. 调用Imple01类的show方法传入String参数
9. 在测试类中创建Imple02对象,并确定泛型为Integer类型
10. 调用Imple02类的show方法传入Integer参数

**public** **interface** Inter<E> {

**public** **abstract** **void** show(E e);

}

**public** **class** Imple01<String> **implements** Inter<String>{

@Override

**public** **void** show(String e) {

System.***out***.println(e);

}

}

**public** **class** Imple02<E> **implements** Inter<E> {

@Override

**public** **void** show(E e) {

System.***out***.println(e);

}

}

**public** **class** Test {

**public** **static** <E> **void** main(String[] args) {

Imple01<E> imple01=**new** Imple01<E>();

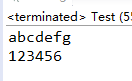
imple01.show((E) "abcdefg");

Imple02<Integer> imple02=**new** Imple02<Integer>();

imple02.show(123456);

}

}



# 关卡2

## 训练案例1

### 训练知识点

1. 自定义对象
2. 迭代器

### 训练描述

自定义学生类:包含姓名,年龄,成绩属性.私有成员变量,生成无参,有参构造方法,生成get/set方法.创建5个学生放到ArrayList中.使用迭代器获取每个学生信息.统计总分,平均分,最高分,最低分并输出

### 操作步骤描述

1. 定义学生类生成类无参,有参构造方法,生成get/set方法.
2. 定义ArrayList
3. 添加5个学生
4. 获取迭代器
5. 定义总分
6. 定义最高分
7. 定义最低分
8. 循环判断是否有下一个学生
9. 使用next方法获取学生对象
10. 将这个学生对象的分数添加到总分
11. 判断这个学生的分数是否大于最大分数
12. 这个学生的分数大于最大分数.最大分数等于这个学生的分数
13. 判断这个学生的分数是否大低最大分数
14. 这个学生的分数小于最小分数.最小分数等于这个学生的分数
15. 计算平均分(平均分 = 总分/人数)
16. 打印总分,最高分,最低分

**import** java.util.ArrayList;

**import** java.util.Iterator;

**public** **class** Test2\_1 {

**public** **static** **void** main(String[] args) {

ArrayList<Student> array=**new** ArrayList<Student>();

Student s1=**new** Student("张三",14,73);

Student s2=**new** Student("李四",15,83);

Student s3=**new** Student("王五",16,90);

Student s4=**new** Student("程晓",14,54);

Student s5=**new** Student("二丫",14,66);

array.add(s1);

array.add(s2);

array.add(s3);

array.add(s4);

array.add(s5);

Iterator<Student> itr = array.iterator();

**double** sum=0;

**double** max=array.get(0).getScore();

**double** min=array.get(0).getScore();

**while**(itr.hasNext()){

Student s=itr.next();

sum+=s.getScore();

**if**(s.getScore()>max){

max=s.getScore();

}

**if**(s.getScore()<min){

min=s.getScore();

}

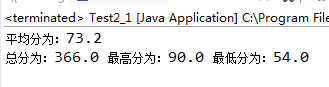
}

System.***out***.println("平均分为："+(sum/5));

System.***out***.println("总分为："+sum+" 最高分为："+max+" 最低分为："+min);

}

}



## 训练案例2

### 训练知识点

1. 自定义对象
2. 迭代器

### 训练描述

自定义人类:包含姓名,年龄,身高属性.私有成员变量,生成无参,有参构造方法,生成get/set方法.创建5个人放到ArrayList中.使用迭代器获取每个人的信息.找出最高的人,最矮的人并输出最高人和最矮人的信息.打印格式如下:最高的人是张三,身高1.80. 最矮的人是李四,身高1.60

### 操作步骤描述

1. 定义学人类生成类无参,有参构造方法,生成get/set方法
2. 定义ArrayList
3. 添加5个学生
4. 获取迭代器
5. 定义最高人
6. 定义最矮人
7. 使用迭代器循环判断是否有下一个人
8. 使用next方法获取学生对象
9. 判断这个人的的身高是否大于最大高度
10. 这个人的的身高大于最大高度.最大身高的人等于这个人
11. 判断这个人的的身高是否小于于最矮高度
12. 这个人的的身高小于于最矮高度.最矮身高的人等于这个人
13. 按照格式打印信息

**public** **class** Person {

**private** String name;

**private** **int** age;

**private** **double** height;

**public** Person(String name, **int** age, **double** height) {

**super**();

**this**.name = name;

**this**.age = age;

**this**.height = height;

}

**public** Person() {

**super**();

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

**public** **double** getheight() {

**return** height;

}

**public** **void** setheight(**double** height) {

**this**.height = height;

}

}

**import** java.util.ArrayList;

**import** java.util.Iterator;

**public** **class** Test2\_2 {

**public** **static** **void** main(String[] args) {

ArrayList<Person> array=**new** ArrayList<Person>();

Person p1=**new** Person("张三",14,1.80);

Person p2=**new** Person("李四",15,1.60);

Person p3=**new** Person("王五",16,1.70);

Person p4=**new** Person("程晓",14,1.74);

Person p5=**new** Person("二丫",14,1.66);

array.add(p1);

array.add(p2);

array.add(p3);

array.add(p4);

array.add(p5);

Iterator<Person> itr = array.iterator();

**double** sum=0;

**double** highest=array.get(0).getheight();

String hname = **null**;

String lname = **null**;

**double** lowest=array.get(0).getheight();

**while**(itr.hasNext()){

Person s=itr.next();

**if**(s.getheight()>=highest){

highest=s.getheight();

hname=s.getName();

}

**if**(s.getheight()<=lowest){

lowest=s.getheight();

lname=s.getName();

}

}

System.***out***.println(" 最高的人是"+hname+",身高"+highest+" 最矮的人是"+lname+",身高"+lowest);

}

}



## 训练案例3

### 训练知识点

1. 方法定义
2. 增强for

### 训练描述

ArrayList中有如下数据:”a”, ”b”, ”c”, ”c”, ”a”, ”b”, ”b”, ”b”, ”a”.定义名为:frequency(ArrayList arr, String key)的方法.arr是ArrayList集合,key是要查找的某个元素.使用增强for方式查找key在ArrayList中出现的次数.并将次数作为方法的返回值.在mian方法中调用frequency方法

### 操作步骤描述

1. 创建ArrayList
2. 添加元素
3. 定义frequency方法统计集合中指定元素出现的次数，
4. 在frequency方法中计数变量
5. 在frequency方法中使用增强for遍历传入的ArrayList集合.拿到每个元素
6. 如果遍历出来的元素是要查找的元素.计数器加1
7. 返回计数值
8. 在main中调用这个方法测试

**import** java.util.ArrayList;

**public** **class** Test2\_3 {

**public** **static** **void** main(String[] args) {

ArrayList<String> arr=**new** ArrayList<String>();

arr.add("a");

arr.add("b");

arr.add("c");

arr.add("c");

arr.add("a");

arr.add("b");

arr.add("b");

arr.add("b");

arr.add("a");

System.***out***.println(*frequency*(arr,"a"));

}

**public** **static** **int** frequency(ArrayList<String> arr,String key){

**int** times=0;

**for**(String string : arr){

**if**(string.equals(key)){

times++;

}

}

**return** times;

}

}



## 训练案例4

### 训练知识点

1.泛型方法

### 训练描述

编写一个泛形方法名称为swap，实现指定位置数组元素的交换.数组和要交换的索引作为方法参数

### 操作步骤描述

1. 编写一个泛形方法，数组是任意类型,并传入2个要交换位置的索引
2. 使用第三方变量的方式交换数组中的元素
3. 定义一个String数组,调用swap方法,交换指定索引的元素
4. 打印交换后的元素
5. 定义一个Integer数组,调用swap方法,交换指定索引的元素
6. 打印交换后的元素

**public** **class** Test2\_4 {

**public** **static** **void** main(String[] args) {

String[] strArr={"hello", "world", "java", "good", "bad"};

*swap*(strArr,0,1);

**for**(String str:strArr){

System.***out***.print(str+" ");

}

System.***out***.println();

Integer[] intArr={1,2,3,4,5,6};

*swap*(intArr,3,5);

**for**(**int** i:intArr){

System.***out***.print(i+" ");

}

}

**public** **static** <T> **void** swap(T[] array,**int** index1,**int** index2) {

T temp;

temp=array[index1];

array[index1]=array[index2];

array[index2]=temp;

}

}

## 训练案例5

### 训练知识点

1.泛型方法

### 训练描述

编写一个泛形方法，接收一个任意类型数组，并反转数组中的所有元素

### 操作步骤描述

1. 编写一个泛形方法reverse，接收一个任意数组
2. 在reverse方法中将数组传入的元素反转
3. 定义一个String数组,调用reverse方法,反转数组中的元素
4. 打印反转后的元素
5. 定义一个Integer数组,调用reverse方法,反转数组中的元素
6. 打印反转后的元素

**public** **class** Test2\_5 {

**public** **static** **void** main(String[] args) {

String[] arr = { "hello", "world", "java", "good", "bad" };

*reverse*(arr);

}

**public** **static** <T> **void** reverse(T[] arr) {

**for** (**int** i = 0; i < arr.length / 2; i++) {

T temp = arr[i];

arr[i] = arr[arr.length - 1 - i];

arr[arr.length - 1 - i] = temp;

}

System.***out***.println("反转后的数据:");

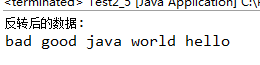
**for** (**int** i = 0; i < arr.length; i++) {

System.***out***.print(arr[i]+" ");

}

}

}



## 训练案例6

### 训练知识点

1.泛型方法

### 训练描述

编写一个泛形方法，接收一个任意类型数组，并将数组中的元素按照一定的格式打印.” [3, 6, 7, 1, 2] ”

### 操作步骤描述

1. 定义泛型方法,接收任意类型的数组
2. 在printArr中,打印"["
3. 使用for获取数组中的元素,打印 "元素, ",最后一个元素打印 "元素]"
4. 定义Integer数组
5. 定义String数组
6. 调用printArr打印数组

**public** **class** Test2\_6 {

**public** **static** **void** main(String[] args) {

Integer[] intArr = { 3, 6, 7, 1, 2 };

String[] strArr = { "hello", "world", "java", "good", "bad" };

*printArr*(intArr);

*printArr*(strArr);

}

**public** **static** <T> **void** printArr(T[] arr) {

System.***out***.print("[");

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

**if** (i != arr.length - 1) {

System.***out***.print(arr[i] + ",");

} **else** **if** (i == arr.length - 1) {

System.***out***.print(arr[i] + "]");

}

}

}

}



## 训练案例7

### 训练知识点

1.增强for

### 训练描述

有一个数组 String arr[]={"abc","bad","abc","aab","bad","cef","jhi"};创建一个ArrayList，将arr数组里面的元素添加进ArrayList中去，但ArrayList中元素不能重复 (arr数组不变)

### 操作步骤描述

1. 创建数组String arr[] = {"abc","bad","abc","aab","bad","cef","jhi"};
2. 创建ArrayList集合
3. 使用增强for遍历数组.拿到数组中的每个元素
4. 使用ArrayList的contains(Object 0)方法判断ArrayList中是否包含某个元素
5. 如果ArrayList中没有这个元素才添加往数组中添加

**import** java.util.ArrayList;

**public** **class** Test2\_7 {

**public** **static** **void** main(String[] args) {

String arr[] = {"abc","bad","abc","aab","bad","cef","jhi"};

ArrayList<String> array=**new** ArrayList<String>();

**for**(String s:arr){

**if**(!array.contains(s)){

array.add(s);

}

}

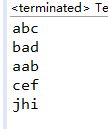
**for** (**int** i = 0; i < array.size(); i++) {

System.***out***.println(array.get(i));

}

}

}



## 训练案例8

### 训练知识点

1. 迭代器使用
2. 定义类

### 训练描述

自定义人类:包含姓名,年龄,身高属性.私有成员变量,生成无参,有参构造方法,生成get/set方法.创建5个人对象放到ArrayList中.使用迭代器获取每个人对象.将每个人的年龄加2岁.再使用增强for打印每个人的信息

### 操作步骤描述

1. 定义学人类生成类无参,有参构造方法,生成get/set方法
2. 定义ArrayList
3. 添加5个学生
4. 获取迭代器
5. 循环判断是否有下一个人
6. 使用next方法获取学生对象
7. 将学生的年龄加2岁,在使用setAge方法设置回去
8. 使用增强for获取每个学生,打印每个学生的信息

**import** java.util.ArrayList;

**import** java.util.Iterator;

**public** **class** Test2\_8 {

**public** **static** **void** main(String[] args) {

ArrayList<Person> array=**new** ArrayList<Person>();

Person p1=**new** Person("张三",14,1.80);

Person p2=**new** Person("李四",15,1.60);

Person p3=**new** Person("王五",16,1.70);

Person p4=**new** Person("程晓",14,1.74);

Person p5=**new** Person("二丫",14,1.66);

array.add(p1);

array.add(p2);

array.add(p3);

array.add(p4);

array.add(p5);

Iterator<Person> itr = array.iterator();

**while**(itr.hasNext()){

Person p=itr.next();

p.setAge(p.getAge()+2);

}

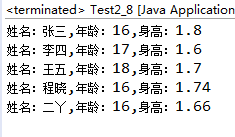
**for**(Person p:array){

System.***out***.println("姓名："+p.getName()+",年龄："+p.getAge()+",身高："+p.getheight());

}

}

}



## 训练案例9

### 训练知识点

1.泛型通配符

### 训练描述

练习课上泛型通配符案例

### 操作步骤描述

详见课上代码

**import** java.util.ArrayList;

**public** **class** Test2\_9 {

**public** **static** **void** main(String[] args) {

// 定义集合b,包含3个元素

ArrayList<String> listB = **new** ArrayList<String>();

listB.add("Jack");

listB.add("Rose");

listB.add("Trump");

// 使用集合b创建集合a

// listB中的元素类型String必须是listA中元素类型Object的子类

ArrayList<Object> listA = **new** ArrayList<Object>(listB);

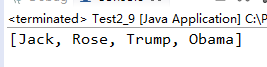
listA.add("Obama");

// 观察集合A

System.***out***.println(listA);

}

}



## 训练案例10

### 训练知识点

1.泛型通配符

### 训练描述

编写一个deleteElements方法，接收一个ArrayList,ArrayList中存放元素的类型不确定，在deleteElements方法的删除ArrayList中的第一个和最后一个元素

### 操作步骤描述

1. 定义deleteElements(ArrayList<?> array)方法
2. 在deleteElements方法中删除第一个元素
3. 删除最后一个元素
4. 创建ArrayList
5. 往ArrayList添加元素
6. 调用deleteElements方法传入array
7. 打印array

**import** java.util.ArrayList;

**public** **class** Test2\_10 {

**public** **static** **void** main(String[] args) {

ArrayList array = **new** ArrayList();

array.add("a");

array.add("b");

array.add("c");

array.add("c");

array.add("a");

array.add("b");

array.add("b");

array.add("b");

array.add("a");

*deleteElements*(array);

**for** (**int** i = 0; i < array.size(); i++) {

System.***out***.println(array.get(i));

}

}

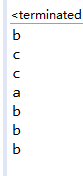
**public** **static** **void** deleteElements(ArrayList<?> array) {

array.remove(0);

array.remove(array.size() - 1);

}

}



# 关卡3

## 训练案例1

### 训练知识点

1. Collection的继承体系
2. Collection子类的特点

### 训练描述

画出Collection的继承体系.并写出ArrayList,LinkedList,HashSet,LinkedHashSet的特点

## 训练案例2

### 训练知识点

1. 泛型方法
2. 面向对象继承

### 训练描述

编写一个泛型方法求两个数之和.两个数可以传入int、long、float、double类型. 提示:泛型不能使用基本数据类型,需要使用引用数据类型.Integer,Long,Float,Double都是Number的子类. Number有doubleValue()方法

**public** **class** Test3\_2 {

**public** **static** **void** main(String[] args) {

*sum*(1,4.4);

*sum*(1.0f,4.4);

*sum*(99.2,43);

*sum*(563653,2425657.35f);

}

**public** **static** <T **extends** Number> **void** sum(T t1,T t2){

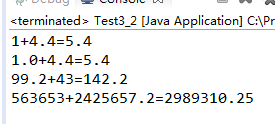
**double** sum = 0.0;

sum = t1.doubleValue() + t2.doubleValue();

System.***out***.println(t1+"+"+t2+"="+sum);

}

}



## 训练案例3

### 训练知识点

1.%运算符

2.嵌套for循环

### 训练描述

统计101-200之间有多少个素数，并输出所有素数。素数又叫质数，就是除了1和它本身之外，再也没有整数能被它整除的数。

**public** **class** Test3\_3 {

**public** **static** **void** main(String[] args) {

**int** count = 0;

**for** (**int** i = 101; i < 200; i += 2) {

**boolean** a = **false**;

**for** (**int** j = 2; j <= Math.*sqrt*(i); j++) {

**if** (i % j == 0) {

a = **false**;

**break**;

} **else** {

a = **true**;

}

}

**if** (a == **true**) {

count++;

System.***out***.print(i + " ");

}

}

System.***out***.println("\n 101-200之间素数个数为：" + count);

}

}

101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199

101-200之间素数个数为：21

## 训练案例4

### 训练知识点

1.斗地主案例

### 训练描述

1.完成斗地主案例

2.增加需求:哪个玩家拿到大王,就需要拿底牌

**import** java.util.ArrayList;

**import** java.util.Collections;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

ArrayList<String> poker = **new** ArrayList<String>();

// 定义花色集合

ArrayList<String> colors = **new** ArrayList<String>();

colors.add("♥");

colors.add("♠");

colors.add("♦");

colors.add("♣");

// 定义数字集合

ArrayList<String> numbers = **new** ArrayList<String>();

**for** (**int** i = 2; i <= 10; i++) {

numbers.add(i + "");

}

numbers.add("J");

numbers.add("Q");

numbers.add("K");

numbers.add("A");

// 嵌套迭代完成每张牌的组装

**for** (String thisColor : colors) {

**for** (String thisNumber : numbers) {

// 通过花色与数字拼写成对应的每张牌

String thisCard = thisColor + thisNumber;

// 将拼好的牌放到集合中

poker.add(thisCard);

}

}

// 补充大小王

poker.add("大☺");

poker.add("小☺");

// 洗牌 Collections类.shuffle方法可以将集合打乱顺序

Collections.*shuffle*(poker);

// 发牌：

// 将每个人以及底牌设计为ArrayList<String>,将最后3张牌直接存放于底牌，剩余牌通过对3取模依次发牌。

ArrayList<String> player1 = **new** ArrayList<String>();

ArrayList<String> player2 = **new** ArrayList<String>();

ArrayList<String> player3 = **new** ArrayList<String>();

ArrayList<String> dipai = **new** ArrayList<String>();

// 使用普通的for循环,通过索引处理纸牌

// 除了最后三张，其他的都发给玩家

**for** (**int** i = 0; i < poker.size() - 3; i++) {

// 获取当前这张牌

String thisCard = poker.get(i);

// 对3取模,决定给哪个玩家发牌

**if** (i % 3 == 0) {

player1.add(thisCard);

} **else** **if** (i % 3 == 1) {

player2.add(thisCard);

} **else** {

player3.add(thisCard);

}

}

// 最后3张是底牌

**for** (**int** i = poker.size() - 3; i < poker.size(); i++) {

// 获取当前这张牌

String thisCard = poker.get(i);

dipai.add(thisCard);

}

// 看牌：

// 直接打印每个集合

System.***out***.println(player1);

System.***out***.println(player2);

System.***out***.println(player3);

System.***out***.println(dipai);

**if** (player1.contains("大☺")) {

player1.addAll(dipai);

}**else** **if**(player2.contains("大☺")){

player2.addAll(dipai);

}**else** **if**(player3.contains("大☺")){

player3.addAll(dipai);

}

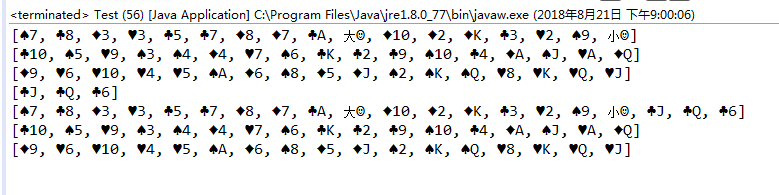
System.***out***.println(player1);

System.***out***.println(player2);

System.***out***.println(player3);

}

}



## 训练案例5

### 训练知识点

1. ArrayList使用
2. Collections. Shuffle()方法
3. 增强for

### 训练描述

1.一副牌有如下内容:有四种花色:♣♦♠♥,每种花色的牌 2 3 4 5 6 7 8 9 10 J Q K A

大王,小王

具体规则：

1.不要大王和小王

2.使52张牌打乱顺序

3.四个玩家,每个玩家随机获取2张牌,注意玩家抽走的牌不能再给其他玩家使用

4.将2张牌相加比较点数大小.A=1,J=11,Q=12,K=13

5.打印玩家中2张牌加起来最大的点数

**import** java.util.ArrayList;

**import** java.util.Collections;

**public** **class** Test3\_5 {

**public** **static** **int** TurnPoker(String point) {

**if** (point.equals("A"))

**return** 1;

**if** (point.equals("2"))

**return** 2;

**if** (point.equals("3"))

**return** 3;

**if** (point.equals("4"))

**return** 4;

**if** (point.equals("5"))

**return** 5;

**if** (point.equals("6"))

**return** 6;

**if** (point.equals("7"))

**return** 7;

**if** (point.equals("8"))

**return** 8;

**if** (point.equals("9"))

**return** 9;

**if** (point.equals("10"))

**return** 10;

**if** (point.equals("J"))

**return** 11;

**if** (point.equals("Q"))

**return** 12;

**if** (point.equals("K"))

**return** 13;

**return** 0;

}

**public** **static** **void** main(String[] args) {

ArrayList<String> poker = **new** ArrayList<String>();

// 定义花色集合

ArrayList<String> colors = **new** ArrayList<String>();

colors.add("♣");

colors.add("♦");

colors.add("♠");

colors.add("♥");

// 定义数字集合

ArrayList<String> numbers = **new** ArrayList<String>();

**for** (**int** i = 2; i <= 10; i++) {

numbers.add(i + "");

}

numbers.add("J");

numbers.add("Q");

numbers.add("K");

numbers.add("A");

**for** (String thisColor : colors) {

**for** (String thisNumber : numbers) {

String thisCard = thisColor + thisNumber;

poker.add(thisCard);

}

}

// 洗牌 Collections类.shuffle方法可以将集合打乱顺序

Collections.*shuffle*(poker);

// 发牌：

ArrayList<String> player1 = **new** ArrayList<String>();

ArrayList<String> player2 = **new** ArrayList<String>();

ArrayList<String> player3 = **new** ArrayList<String>();

ArrayList<String> player4 = **new** ArrayList<String>();

// 使用普通的for循环,通过索引处理纸牌

// 除了最后三张，其他的都发给玩家

**for** (**int** i = 0; i < 8; i++) {

// 获取当前这张牌

String thisCard = poker.get(i);

// 对3取模,决定给哪个玩家发牌

**if** (i % 4 == 0) {

player1.add(thisCard);

} **else** **if** (i % 4 == 1) {

player2.add(thisCard);

} **else** **if** (i % 4 == 2) {

player3.add(thisCard);

} **else** **if** (i % 4 == 3) {

player4.add(thisCard);

}

}

// 看牌：

// 直接打印每个集合

System.***out***.println(player1);

System.***out***.println(player2);

System.***out***.println(player3);

System.***out***.println(player4);

// 牌数相加

**int**[] sum = **new** **int**[4];

sum[0] = *TurnPoker*(player1.get(0).substring(1)) + *TurnPoker*(player1.get(1).substring(1));

sum[1] = *TurnPoker*(player2.get(0).substring(1)) + *TurnPoker*(player2.get(1).substring(1));

sum[2] = *TurnPoker*(player3.get(0).substring(1)) + *TurnPoker*(player3.get(1).substring(1));

sum[3] = *TurnPoker*(player4.get(0).substring(1)) + *TurnPoker*(player4.get(1).substring(1));

**int** max = 0;

**for** (**int** j = 0; j < sum.length; j++) {

System.***out***.print(sum[j]+" ");

**if** (sum[j] >= max) {

max = sum[j];

}

}

System.***out***.println("\n"+max);

}

}

