**每日作业卷答案**

**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

### 操作步骤答案

**public** **class** Test01 {

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

// 1.创建Colection对象.Collection是接口.所以创建子类ArrayList对象

Collection<String> coll = **new** ArrayList<String>();

// 2.往集合中添加对象元素

coll.add("igeek1");

coll.add("igeek2");

coll.add("igeek3");

// 3.删除元素

coll.remove("igeek2");

// 4.获取集合大小

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

// 5.清空集合

coll.clear();

// 6.输出集合内容

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

}

}

## 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

### 操作步骤答案

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

### 操作步骤答案

**public** **class** Test03 {

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

// 1.创建集合对象

Collection<String> coll = **new** ArrayList<String>();

// 2.往集合中存放元素

coll.add("abc1");

coll.add("abc2");

coll.add("abc3");

coll.add("abc4");

// 3.获取容器的迭代器

Iterator<String> it = coll.iterator();

// 4.使用迭代器判断是否有下一个元素

**while**(it.hasNext()){

// 5.使用迭代器对象获取集合中的元素

System.***out***.println(it.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

### 操作步骤答案

**public** **class** Test04 {

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

// 1.创建数组

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

// 2.使用增强for循环获取数组中的元素

**for** (**int** n : arr) {//变量n代表被遍历到的数组元素

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

}

}

}

## 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

### 操作步骤答案

**public** **class** Test05 {

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

// 1.创建ArrayList

Collection<String> coll = **new** ArrayList<String>();

// 2.往ArrayList中添加元素

coll.add("igeek1");

coll.add("igeek2");

coll.add("igeek3");

coll.add("igeek4");

// 3.使用增强for循环获取元素

**for**(String str : coll){ //变量Str代表被遍历到的集合元素

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

}

}

}

## 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

### 操作步骤答案

**public** **class** Test06 {

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

// method01();

*method02*();

}

**private** **static** **void** method02() {

// 1.定义有泛型的ArrayList

Collection<String> list = **new** ArrayList<String>();

// 2.添加字符串

list.add("abc");

list.add("igeek");

// 3.添加Integer类型

// list.add(5); // 当集合明确类型后，存放类型不一致就会编译报错

// 4.获取迭代器

Iterator<String> it = list.iterator();

// 5.判断是否有下一个元素

**while** (it.hasNext()) {

// 6.获取下一个元素,元素是String类型

String str = it.next();

// 7.获取字符串的长度

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

}

}

**private** **static** **void** method01() {

// 1.定义没有泛型的ArrayList

Collection list = **new** ArrayList<String>();

// 2.添加字符串

list.add("abc");

list.add("igeek");

// 3.添加Integer类型

list.add(5); // 没有泛型,任意类型都能放入

// 4.获取迭代器

Iterator it = list.iterator();

// 5.判断是否有下一个元素

**while** (it.hasNext()) {

// 6.获取下一个元素,并强转为String类型

String str = (String)it.next();

// 7.获取字符串的长度

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

}

}

}

使用迭代器的好处:

集合明确具体存放的元素类型，那么在使用迭代器的时候，迭代器也同样会知道具体遍历元素类型,存放类型不一致就会编译报错,避免了类型强转的麻烦.

## 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

### 操作步骤答案

**public** **class** Test07 {

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

// 4.创建泛型类对象.并指定具体类型

GenericClass<Integer> g = **new** GenericClass<Integer>();

// 5.调用set方法传入对应的类型

g.setElement(100);

// 6.调用get方法获取成员变量的值

Integer i = g.getElement();

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

}

}

// 1.自定义泛型类

**class** GenericClass<E> {

// 2.定义E类型的成员变量

**private** E element;

// 3.添加get/set方法

**public** **void** setElement(E element) {

**this**.element = element;

}

**public** E getElement() {

**return** element;

}

}

## 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

### 操作步骤答案

**public** **class** Test08 {

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

// 3.调用function传入字符串

*function*("Hello");

// 4.调用function传入数字

*function*(123);

}

// 1.定义泛型方法

**public** **static** <E> **void** function(E e) {

// 2.在泛型方法中,将传入的参数放到输出语句中

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

}

}

## 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

### 操作步骤答案

定义泛型接口Inter<E>

// 1.定义泛型接口Inter<E>

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

// 2.在Inter接口中定义抽象的show(E e)方法

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

}

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

// 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

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

// 4.重写Imple01中的show(String e)方法.打印传入的内容

@Override

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

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

}

}

定义Imple02类.实现Inter接口,定义实现类时不确定泛型的类型

// 5.定义Imple02类.实现Inter接口,定义实现类时不确定泛型的类型

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

// 6.重写Imple02类中的show(E e)方法.打印传入的内容

@Override

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

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

}

}

在测试类中创建Imple01对象

在测试类中创建Imple02对象,并确定泛型为Integer类型

**public** **class** Test09 {

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

// 7.在测试类中创建Imple01对象

Imple01 i1 = **new** Imple01();

// 8.调用Imple01类的show方法传入String参数

i1.show("Hello");

// 9.在测试类中创建Imple02对象,并确定泛型为Integer类型

Imple02<Integer> i2 = **new** Imple02<>();

// 10.调用Imple02类的show方法传入Integer参数

i2.show(1234);

}

}

# 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

### 操作步骤答案

定义学生类生成类无参,有参构造方法,生成get/set方法.

**public** **class** Student {

**private** String name;

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

**private** **double** score;

**public** Student() {}

**public** Student(String name, **int** age, **double** score) {

**this**.name = name;

**this**.age = age;

**this**.score = score;

}

// getxxx/setxxx

}

测试类:

**public** **class** Test01 {

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

// 2.定义ArrayList

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

// 3.添加5个学生

students.add(**new** Student("张三", 20, 99));

students.add(**new** Student("王昭君", 21, 88));

students.add(**new** Student("西施", 20, 80));

students.add(**new** Student("杨玉环", 20, 70));

students.add(**new** Student("凤姐", 22, 30));

// 4.获取迭代器

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

// 5.定义总分

**double** totalScore = 0;

// 6.定义最高分

**double** maxScore = students.get(0).getScore();

// 7.定义最低分

**double** minScore = students.get(0).getScore();

// 8.循环判断是否有下一个学生

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

// 9.使用next方法获取学生对象

Student student = itr.next();

// 10.将这个学生对象的分数添加到总分

totalScore += student.getScore();

// 11.判断这个学生的分数是否大于最大分数

**if** (maxScore < student.getScore()) {

// 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

maxScore = student.getScore();

} **else** **if** (minScore > student.getScore()) {

// 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

minScore = student.getScore();

}

}

// 15.计算平均分(平均分 = 总分/人数)

**double** avgScore = totalScore / students.size();

// 16.打印总分,最高分,最低分

System.***out***.println("总分 = " + totalScore + " ,平均分 = " + avgScore + " ,最高分 = " + maxScore + ",最低 = " + minScore);

}

}

## 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

### 操作步骤答案

Person类:

// 1.定义学人类生成类无参,有参构造方法,生成get/set方法.

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

**private** String name;

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

**private** **double** height

**public** Person() {}

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

**this**.name = name;

**this**.age = age;

**this**.height = height;

}

// getxxx/setxxx

}

测试类:

**public** **class** Test02 {

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

// 2.定义ArrayList

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

// 3.添加5个学生

persons.add(**new** Person("张三", 20, 1.66));

persons.add(**new** Person("王昭君", 21, 1.65));

persons.add(**new** Person("西施", 20, 1.70));

persons.add(**new** Person("杨玉环", 20, 1.72));

persons.add(**new** Person("凤姐", 22, 1.80));

// 4.获取迭代器

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

// 5.定义最高人

Person maxHeightPerson = persons.get(0);

// 6.定义最矮人

Person minHeightPerson = persons.get(0);

// 7.循环判断是否有下一个人

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

// 8.使用next方法获取学生对象

Person student = itr.next();

// 9.判断这个人的的身高是否大于最大高度

**if** (maxHeightPerson.getHeight() < student.getHeight()) {

// 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

maxHeightPerson = student;

} **else** **if** (minHeightPerson.getHeight() > student.getHeight()) {

// 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

minHeightPerson = student;

}

}

// 13.按照格式打印信息

System.***out***.println("最高的人是" + maxHeightPerson.getName() + ",身高 " +

maxHeightPerson.getHeight() + ". 最矮的人是" + minHeightPerson.getName()

+ ",身高 " + minHeightPerson.getHeight());

}

}

## 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

### 操作步骤答案

**public** **class** Test03 {

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

// 1.创建ArrayList

List<String> list = **new** ArrayList<>();

// 2.添加元素

list.add("a");

list.add("a");

list.add("a");

list.add("b");

list.add("b");

list.add("c");

list.add("d");

list.add("d");

list.add("d");

list.add("d");

list.add("d");

// 8.在main中调用这个方法测试

System.***out***.println(*frequency*(list, "a")); // 3

}

// 3.定义frequency方法统计集合中指定元素出现的次数，

**public** **static** **int** frequency(List<String> list, String str) {

// 4.在frequency方法中计数变量

**int** count = 0;

// 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

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

// 6.如果遍历出来的元素是要查找的元素.计数器加1

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

count++;

}

}

// 7.返回计数值

**return** count;

}

}

## 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

### 操作步骤答案

**public** **class** Test04 {

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

// 3.定义一个String数组,调用swap方法,交换指定索引的元素

String[] strs = {"a","b","c"};

*swap*(strs, 0, 2);

// 4.打印交换后的元素

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

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

}

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

// 5.定义一个Integer数组,调用swap方法,交换指定索引的元素

Integer[] arr = {1, 8, 9, 5, 7};

*swap*(arr, 0, 2);

// 6.打印交换后的元素

**for** (Integer i : arr) {

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

}

}

// 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

**public** **static** <T> **void** swap(T[] arr,**int** i,**int** j) {

// 2.使用第三方变量的方式交换数组中的元素

T temp = arr[i];

arr[i] = arr[j];

arr[j] = 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

### 操作步骤答案

**public** **class** Test05 {

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

// 3.定义一个String数组,调用reverse方法,反转数组中的元素

String[] strs = {"a","b","c"};

*reverse*(strs);

// 4.打印反转后的元素

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

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

}

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

// 5.定义一个Integer数组,调用reverse方法,反转数组中的元素

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

*reverse*(arr);

// 6.打印反转后的元素

**for** (Integer i : arr) {

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

}

}

// 1.编写一个泛形方法reverse，接收一个任意数组

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

// 2.在reverse方法中将数组传入的元素反转

**for** (**int** start = 0, end = arr.length - 1; start < end; start++, end--) {

T temp = arr[start];

arr[start] = arr[end];

arr[end] = temp;

}

}

}

## 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] ”

### 操作步骤答案

**public** **class** Test06 {

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

// 4.定义Integer数组

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

// 5.定义String数组

String[] strs = {"a","b"};

// 6.调用printArr打印数组

*printArr*(arr);

*printArr*(strs);

}

// 1.定义泛型方法printArr,接收任意类型的数组

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

// 2.在printArr中,打印"["

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

// 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]"

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

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

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

} **else** {

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)

### 操作步骤答案

**public** **class** Test07 {

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

// 1.创建数组String arr[] = {"abc","bad","abc","aab","bad","cef","jhi"};

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

// 2.创建ArrayList集合

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

// 3.使用增强for遍历数组.拿到数组中的每个元素

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

// 4.使用ArrayList的contains(Object 0)方法判断ArrayList中是否包含某个元素

**if**(!list.contains(str)) {

// 5.如果ArrayList中没有这个元素才添加往数组中添加

list.add(str);

}

}

// 6.输出ArrayList的元素

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

}

}

## 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

### 操作步骤答案

定义学人类生成类无参,有参构造方法,生成get/set方法

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

**private** String name;

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

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

**public** Person() {}

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

**this**.name = name;

**this**.age = age;

**this**.height = height;

}

// getXXX/setXXX

}

测试类:

**public** **class** Test08 {

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

// 2.定义ArrayList

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

// 3.添加5个学生

persons.add(**new** Person("张三", 20, 1.66));

persons.add(**new** Person("王昭君", 21, 1.65));

persons.add(**new** Person("西施", 20, 1.70));

persons.add(**new** Person("杨玉环", 20, 1.72));

persons.add(**new** Person("凤姐", 22, 1.80));

// 4.获取迭代器

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

// 5.循环判断是否有下一个人

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

// 6.使用next方法获取学生对象

Person p = itr.next();

// 7.将学生的年龄加2岁,在使用setAge方法设置回去

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

}

// 8.使用增强for获取每个学生,打印每个学生的信息

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

System.***out***.println(p.getName() + " 的身高是 " + p.getAge());

}

}

}

## 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

### 操作步骤答案

详见课上代码

## 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

### 操作步骤答案

**public** **class** Test10 {

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

// 4.创建ArrayList

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

// 5.往ArrayList添加元素

array.add("王昭君");

array.add("西施");

array.add("杨玉环");

array.add("貂蝉");

// 6.调用deleteElements方法传入array

*deleteElements*(array);

// 7.打印array

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

}

// 1.定义deleteElements(ArrayList<?> array)方法

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

// 2.在deleteElements方法中删除第一个元素

array.remove(0);

// 3.删除最后一个元素

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

### 操作步骤答案

## 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

### 操作步骤答案

**public** **class** Test02 {

// Number: 类上面 ctrl + t可以看到这个类的继承体系

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

**double** a = 10.5;

**double** b = 2.5;

// 3.调用sum方法,传入2个double值

**double** sum = *sum*(a, b);

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

**int** c = 10;

**int** d = 20;

// 4.调用sum方法,传入2个int值

System.***out***.println(*sum*(c, d));

System.***out***.println(*sum*(10, 1.2));

}

// 1.定义泛型方法sum(T a, T b).向上限定<T extends Number>让参数传递的时候只能传递Number及其子类

**public** **static** <T **extends** Number> **double** sum(T a, T b) {

// 2.参数a和b是T类型不能直接进行相加

// 3.将传递进来的a和b调用doubleValue方法获取到double类型的值,再进行加法运算

**return** a.doubleValue() + b.doubleValue();

}

}

## 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。

### 操作步骤答案

**public** **class** Test03 {

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

// 定义计数变量

**int** count = 0;

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

// 103 / 2 3 4 5 6 7 8 9 ... 102

**boolean** flag = **false**;

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

// 被其他数整除了

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

flag = **true**;

**break**;

}

}

**if** (!flag) { // 没有被其他数整除的是素数

count++;

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

}

}

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

System.***out***.println("在101~200之间一共有素数：" + count + "个");

}

}

## 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

### 操作步骤答案

/\*

\* 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

\*/

**public** **class** Test04 {

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

// 一副完整的牌

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

// 添加牌

*addPokers*(pokers);

// 洗牌

*washPokers*(pokers);

// 发牌

*dealPokers*(pokers);

}

/\*

\* 发牌

\* 3个玩家使用3个ArrayList来存放牌

\* 底牌也使用ArrayList来存放

\*/

**public** **static** **void** dealPokers(ArrayList<String> pokers) {

// 玩家1

ArrayList<String> player01 = **new** ArrayList<String>();

// 玩家2

ArrayList<String> player02 = **new** ArrayList<String>();

// 玩家3

ArrayList<String> player03 = **new** ArrayList<String>();

// 底牌

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

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

String poker = pokers.get(i);

**if** (i >= 51) {

diPai.add(poker);

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

player01.add(poker);

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

player02.add(poker);

} **else** **if**(i % 3 == 2) {

player03.add(poker);

}

}

// 看哪个玩家拿到大王,就将底牌也拿走

**if** (player01.contains("大王")) {

System.***out***.println("玩家1拿到大王,底牌也拿走");

player01.addAll(diPai);

} **else** **if** (player02.contains("大王")) {

System.***out***.println("玩家2拿到大王,底牌也拿走");

player02.addAll(diPai);

} **else** **if** (player03.contains("大王")) {

System.***out***.println("玩家3拿到大王,底牌也拿走");

player03.addAll(diPai);

}

// 将每个玩家的牌打印出来

System.***out***.println("玩家1的牌:");

**for** (String p : player01) {

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

}

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

System.***out***.println("玩家2的牌:");

**for** (String p : player02) {

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

}

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

System.***out***.println("玩家3的牌:");

**for** (String p : player03) {

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

}

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

System.***out***.println("底牌牌:");

**for** (String p : diPai) {

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

}

}

/\*

\* 洗牌

\*/

**public** **static** **void** washPokers(ArrayList<String> pokers) {

// 洗牌,其实就是将ArrayList中的元素顺序打乱

Collections.*shuffle*(pokers);

}

// 添加牌

**public** **static** **void** addPokers(ArrayList<String> pokers) {

String[] colors = {"♣", "♦", "♠", "♥"};

String[] numbers = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};

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

// 获取花色

String color = colors[i];

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

// 获取数字

String number = numbers[j];

// 拼接成一张牌

String poker = color + number;

// 添加到一副牌中

pokers.add(poker);

}

}

// 添加大小王

pokers.add("小王");

pokers.add("大王");

}

}

## 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

### 操作步骤答案

**public** **class** Test05 {

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

// 一副完整的牌

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

// 添加牌

*addPokers*(pokers);

// 洗牌

*washPokers*(pokers);

// 发牌

*dealPokers*(pokers);

}

/\*

\* 发牌

\* 3个玩家使用3个ArrayList来存放牌

\* 底牌也使用ArrayList来存放

\*/

**public** **static** **void** dealPokers(ArrayList<String> pokers) {

// 玩家1

ArrayList<String> player01 = **new** ArrayList<String>();

// 玩家2

ArrayList<String> player02 = **new** ArrayList<String>();

// 玩家3

ArrayList<String> player03 = **new** ArrayList<String>();

// 玩家4

ArrayList<String> player04 = **new** ArrayList<String>();

// 玩家已经拿走的牌的索引

ArrayList<Integer> removePokers = **new** ArrayList<Integer>();

*getPokers*(pokers, player01, removePokers);

*getPokers*(pokers, player02, removePokers);

*getPokers*(pokers, player03, removePokers);

*getPokers*(pokers, player04, removePokers);

System.***out***.println(player01);

System.***out***.println(player02);

System.***out***.println(player03);

System.***out***.println(player04);

// 定义一个数组放四个玩家的点数

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

continer[0] = *getPoints*(player01);

continer[1] = *getPoints*(player02);

continer[2] = *getPoints*(player03);

continer[3] = *getPoints*(player04);

// 定义最大点数

**int** max = 0;

**for** (**int** point : continer) {

**if** (max < point) {

max = point;

}

}

System.***out***.println("四个玩家中最大点数是: " + max);

}

/\*\*

\* 根据玩家的牌,计算出点数

\* **@param** player

\* **@return**

\*/

**public** **static** **int** getPoints(ArrayList<String> player) {

**int** point = 0;

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

**if** (str.contains("A")) {

point += 1;

} **else** **if** (str.contains("J")) {

point += 11;

} **else** **if** (str.contains("Q")) {

point += 12;

} **else** **if** (str.contains("K")) {

point += 13;

} **else** {

point += Integer.*valueOf*(str.substring(1));

}

}

**return** point;

}

/\*\*

\* 一个玩家拿2张牌

\* **@param** pokers 所有的牌

\* **@param** player 玩家

\* **@param** removePokers 已经拿走牌的序号

\*/

**public** **static** **void** getPokers(ArrayList<String> pokers, ArrayList<String> player, ArrayList<Integer> removePokers) {

Random ran = **new** Random();

// 如果玩家的牌小于2张,就拿一张

**while** (player.size() < 2) {

// 拿到牌的索引

**int** index = ran.nextInt(52);

// 看这张牌是否有人拿走过

**if** (!removePokers.contains(index)) {

// 这张牌没有被别人拿走.就给玩家

player.add(pokers.get(index));

// 这张牌已经被别人拿走,后面的人就不要来拿这张牌

removePokers.add(index);

}

}

}

/\*

\* 洗牌

\*/

**public** **static** **void** washPokers(ArrayList<String> pokers) {

// 洗牌,其实就是将ArrayList中的元素顺序打乱

Collections.*shuffle*(pokers);

}

// 添加牌

**public** **static** **void** addPokers(ArrayList<String> pokers) {

String[] colors = {"♣", "♦", "♠", "♥"};

String[] numbers = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};

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

// 获取花色

String color = colors[i];

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

// 获取数字

String number = numbers[j];

// 拼接成一张牌

String poker = color + number;

// 添加到一副牌中

pokers.add(poker);

}

}

}

}