**每日作业卷答案**

**JavaSE第08天**

# 关卡1

## 训练案例1

### 训练知识点

1. 栈和队列的特点

### 训练描述

请说出栈和队列的特点

### 操作步骤答案

请说出栈和队列的特点

\* 栈是后进先出(LIFO)的数据结构

\* 队列是先进先出(FIFO)的数据结构

## 训练案例2

### 训练知识点

1.数组和链表特点

### 训练描述

请说出数组和链表特点

### 操作步骤答案

请说出数组和链表特点

\* 数组的特点:连续的内存空间,查询快,添加和删除元素速度慢

\* 链表的特点:元素是不连续存放的,上一个元素记录下一个元素的地址,增删块,查询慢

## 训练案例3

### 训练知识点

1.LinkedList基本使用

### 训练描述

练习LinkedList基本方法: add, set, get, remove, clear, size方法

### 操作步骤答案

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

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

// 1.创建LinkedList

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

// 2.使用add方法添加元素

arr.add("西门吹雪");

arr.add("西门吹雪");

arr.add("西门吹雪");

arr.add("西门吹风");

arr.add("西门吹水");

// 3.使用add方法在指定索引添加元素

arr.add(2, "西门吹雨");

// 4.使用set方法修改指定位置索引

arr.set(0, "东门");

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

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

}

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

// 5.使用get方法获取指定索引的元素

System.***out***.println(arr.get(1));

// 6.使用size方法获取集合大小

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

// 7.使用remove方法删除指定索引的元素

arr.remove(3);

// 8.使用clear清空集合中的元素

arr.clear();

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

}

}

## 训练案例4

### 训练知识点

1.LinkedList特有方法

### 训练描述

练习LinkedList特有方法: addFirst, addLast, getFirst, getLast, removeFirst, removeLast方法

### 操作步骤答案

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

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

// 1.创建LinkedList

LinkedList<String> linked = **new** LinkedList<String>();

// 2.使用add方法添加元素

linked.add("刘德华");

linked.add("张学友");

// 3.使用addFirst添加元素到集合最前面

linked.addFirst("黎明");

// 4.使用addLast添加元素到集合最后面

linked.addLast("郭富城");

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

// 5.使用getFirst获取集合第一个元素

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

// 6.使用getLast获取集合最后一个元素

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

// 7.使用removeLast删除集合第一个元素

String first = linked.removeFirst();

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

// 8.使用removeLast删除集合最后一个元素

String last = linked.removeLast();

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

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

// 9.使用clear清空集合

linked.clear();

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

}

}

## 训练案例5

### 训练知识点

1.HashSet基本使用

### 训练描述

往HashSet中添加字符串"zhangsan", "lisi", "wangwu", "zhangsan".使用迭代器获取HashSet中的元素.

### 操作步骤答案

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

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

// 1.创建HashSet集合

HashSet<String> hs = **new** HashSet<>();

// 2.使用add方法往HashSet添加元素

hs.add("zhangsan");

hs.add("lisi");

hs.add("wangwu");

hs.add("zhangsan");

// 3.使用迭代器获取HashSet中的元素

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

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

String str = itr.next();

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

}

}

}

## 训练案例6

### 训练知识点

1.hashCode和equals方法

### 训练描述

请简述HashSet去除重复元素的原理

### 操作步骤答案

请简述HashSet去除重复元素的原理

\* 1.调用被添加元素的hashCode(),和HashSet中已有元素的hashCode比较是否相同

\* 2.如果不相同,直接存储

\* 3.如果相同.调用equals方法比较是否相同

\* 4.不相同,直接存储元素

\* 5.相同,认为是同一元素.不存储

## 训练案例7

### 训练知识点

1.HashSet存储自定义类型

### 训练描述

定义人类,包含姓名和年龄属性.创建4个人存储到HashSet中.姓名和年龄相同的人看做同一人不存储

### 操作步骤答案

Person类:

// 1.定义Person类.包好姓名年龄属性,重写hashCode()和equals()方法

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

**private** String name;

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

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

**this**.name = name;

**this**.age = age;

}

@Override

**public** String toString() {

**return** name + "," + age;

}

@Override

**public** **int** hashCode() {

**final** **int** prime = 31;

**int** result = 1;

result = prime \* result + age;

result = prime \* result + ((name == **null**) ? 0 : name.hashCode());

**return** result;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

Person other = (Person) obj;

**if** (age != other.age)

**return** **false**;

**if** (name == **null**) {

**if** (other.name != **null**)

**return** **false**;

} **else** **if** (!name.equals(other.name))

**return** **false**;

**return** **true**;

}

}

测试类

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

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

// 2.创建HashSet用于存储Person类型

HashSet<Person> hashSet = **new** HashSet<Person>();

// 3.添加多个Person到HashSet中

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

hashSet.add(**new** Person("西施", 21));

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

hashSet.add(**new** Person("貂蝉", 19));

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

hashSet.add(**new** Person("貂蝉", 19));

// 4.遍历获取HashSet中的内容

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

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

}

}

}

## 训练案例8

### 训练知识点

1. LinkedHashSet基本使用

### 训练描述

使用LinkedHashSet存储以下元素:"王昭君","王昭君","西施","杨玉环","貂蝉".使用迭代器和增强for循环遍历LinkedHashSet

### 操作步骤答案

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

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

// 1.创建LinkedHashSet

LinkedHashSet<String> lhSet = **new** LinkedHashSet<String>();

// 2.使用add方法添加元素到LinkedHashSet

lhSet.add("王昭君");

lhSet.add("王昭君");

lhSet.add("王昭君");

lhSet.add("西施");

lhSet.add("杨玉环");

lhSet.add("貂蝉");

// 3.使用迭代器获取LinkedHashSet中的元素

Iterator<String> iterator = lhSet.iterator();

**while** (iterator.hasNext()) {

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

}

// 4.使用增强for获取LinkedHashSet中的元素

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

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

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

}

}

}

# 关卡2

## 训练案例1

### 训练知识点

1.HashSet

### 训练描述

ArrayList有以下元素: "a","f","b","c","a","d"利用HashSet对ArrayList集合去重(最终结果: ArrayList中没有重复元素)

### 操作步骤答案

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

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

// 1.创建ArrayList

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

// 2.使用add方法往ArrayList添加元素

list.add("a");

list.add("f");

list.add("b");

list.add("c");

list.add("a");

list.add("d");

// 3.创建HashSet.用于将ArrayList中重复的元素去除

HashSet<String> set = **new** HashSet<>();

// 4.调用HashSet的addAll方法,将ArrayList中的元素添加到HashSet中

set.addAll(list);

// 5.清空list的所有元素

list.clear();

// 6.将set集合中的元素再添加回ArrayList集合

list.addAll(set);

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

}

}

## 训练案例2

### 训练知识点

1. HashSet使用

### 训练描述

向HashSet集合添加姓名{张三,李四,王五,二丫,钱六,孙七},将二丫删除,添加王小丫

### 操作步骤答案

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

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

// 1.创建HashSet集合

HashSet<String> hs = **new** HashSet<String>();

// 2.使用add方法往HashSet中添加元素

hs.add("张三");

hs.add("李四");

hs.add("王五");

hs.add("二丫");

hs.add("钱六");

hs.add("孙七");

// 3.使用remove方法删除HashSet中的二丫

hs.remove("二丫");

// 4.使用add方法往HashSet中添加王小丫

hs.add("王小丫");

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

}

}

## 训练案例3

### 训练知识点

1. LinkedList使用
2. Collection的contains(Object o)方法
3. 增强for和迭代器

### 训练描述

已知数组存放一批QQ号码.QQ号码最长为11位,最短为5位

String[] strs = {"12345","67891",1"2347809933","98765432102","67891","12347809933"}

将该数组里面的所有qq号都存放在LinkedList中,将list中重复元素删除,将list中所有元素用两种方式打印出来

### 操作步骤答案

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

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

// 1.定义QQ号码数组String[] strs = {"12345","67891",1"2347809933","98765432102","67891","12347809933"}

String[] strs = { "12345", "67891", "2347809933", "98765432102", "67891", "12347809933" };

// 2.创建LinkedList

LinkedList<String> qqList = **new** LinkedList<>();

// 3.遍历strs获取每个qq号码

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

// 4.判断LinkedList是否已经存在这个qq号码

**if** (!qqList.contains(qq)) {

// 5.不存在这个qq号码则添加到LinkedList中

qqList.add(qq);

}

}

// 6.增强for遍历

**for** (String qq : qqList) {

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

}

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

// 7.迭代器遍历

Iterator<String> iterator = qqList.iterator();

**while** (iterator.hasNext()) {

String string = iterator.next();

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

}

}

}

## 训练案例4

### 训练知识点

1. HashSet

### 训练描述

定义一个Student类,包含名称,年龄,性别(姓名,年龄,性别完全相同视为同一学生)

创建10个Student对象,至少有两个学生姓名,年龄,性别完全相同. 把这10个学生添加到Set集合中,不可以重复,遍历Set集合打印学生信息,使用两种方式

### 操作步骤答案

Student类

// 1.定义一个Student类,包含名称,年龄,性别重写hashCode()和equals()方法

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

// 名称

**private** String name;

// 年龄

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

// 性别

**private** String gender;

**public** Student() {

}

**public** Student(String name, **int** age, String gender) {

**this**.name = name;

**this**.age = age;

**this**.gender = gender;

}

**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** String getGender() {

**return** gender;

}

**public** **void** setGender(String gender) {

**this**.gender = gender;

}

@Override

**public** **int** hashCode() {

**final** **int** prime = 31;

**int** result = 1;

result = prime \* result + age;

result = prime \* result + ((gender == **null**) ? 0 : gender.hashCode());

result = prime \* result + ((name == **null**) ? 0 : name.hashCode());

**return** result;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

Student other = (Student) obj;

**if** (age != other.age)

**return** **false**;

**if** (gender == **null**) {

**if** (other.gender != **null**)

**return** **false**;

} **else** **if** (!gender.equals(other.gender))

**return** **false**;

**if** (name == **null**) {

**if** (other.name != **null**)

**return** **false**;

} **else** **if** (!name.equals(other.name))

**return** **false**;

**return** **true**;

}

@Override

**public** String toString() {

**return** "Student [name=" + name + ", age=" + age + ", gender=" + gender

+ "]";

}

}

测试类:

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

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

// 2.创建HashSet集合

HashSet<Student> set = **new** HashSet<>();

// 3.使用Student类创建10个学生,把这10个学生添加到Set集合中

set.add(**new** Student("张三", 18, "男"));

set.add(**new** Student("小王", 16, "女"));

set.add(**new** Student("倩倩", 19, "女"));

set.add(**new** Student("张伟", 18, "男"));

set.add(**new** Student("张三", 18, "男"));

set.add(**new** Student("老王", 38, "男"));

set.add(**new** Student("张三", 18, "男"));

set.add(**new** Student("小刘", 20, "女"));

set.add(**new** Student("张三", 18, "男"));

set.add(**new** Student("天二", 21, "男"));

// 4.使用迭代器遍历HashSet

Iterator<Student> iterator = set.iterator();

**while**(iterator.hasNext()) {

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

}

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

// 5.使用增强for遍历集合

**for**(Student stu : set){

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

}

}

}

## 训练案例5

### 训练知识点

1.HashSet

### 训练描述

有一个数组 String arr[]={"abc","bad","abc","aab","bad","cef","jhi"};创建一个ArrayList，将数组里面的元素添加进ArrayList，但元素不能重复(使用HashSet去除重复元素)

### 操作步骤答案

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

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

// 1.创建arr数组

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

// 2.创建HashSet集合

HashSet<String> hs = **new** HashSet<>();

// 3.遍历arr数组

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

// 4.将arr数组中的元素添加进HashSet中.重复的就不会添加进去了

hs.add(str);

}

// 5.创建ArrayList集合,将HashSet的元素添加到ArrayList中

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

// 6.遍历ArrayList打印元素

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

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

}

}

}

## 训练案例6

### 训练知识点

1. HashSet
2. IO

### 训练描述

ArrayList集合有以下元素:{"张柏芝","刘德华","张亮","张靓颖","杨颖","黄晓明","张柏芝","刘德华"}.1.使用HashSet将重复的元素去除.2.编写一个方法:将去重后的人员写入到D:\\a.txt中.3.编写一个方法:将D:\\a.txt中所有姓张的人员信息读取出来并打印到控制台上

### 操作步骤答案

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

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

// 1.定义ArrayList

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

// 2.使用add方法往ArrayList添加指定的内容

arr.add("张柏芝");

arr.add("刘德华");

arr.add("张亮");

arr.add("张靓颖");

arr.add("杨颖");

arr.add("黄晓明");

arr.add("张柏芝");

arr.add("刘德华");

// 3.创建HashSet.将ArrayList的内容放入HashSet去除重复内容

HashSet<String> hs = **new** HashSet<>(arr);

// 9.调用save方法,传入去重后的名称

*save*(hs);

// 15.调用read方法

*read*();

}

// 10.定义read()方法读取指定文件内容

**public** **static** **void** read() **throws** IOException {

// 11.创建BufferedReader字符输入流

BufferedReader br = **new** BufferedReader(**new** FileReader("a.txt"));

System.***out***.println("读取到姓张的名称:");

String line = **null**;

// 12.读取文件中的名称

**while** ((line = br.readLine()) != **null**) {

// 13.判断名称是否以张开头

**if** (line.startsWith("张")) {

// 14.打印以张开头的名称

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

}

}

}

// 4.定义save(HashSet<String> hs)方法

**public** **static** **void** save(HashSet<String> hs) **throws** IOException {

// 5.在save方法中创建BufferedWriter字符输出流

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("a.txt"));

// 6.遍历获取每个名称

**for** (String name : hs) {

// 7.将名称写入指定文件中

bw.write(name);

bw.newLine();

}

// 8.关闭流

bw.close();

}

}

## 训练案例7

### 训练知识点

1.HashSet

### 训练描述

编写一个程序，获取10个1至20的随机数，要求随机数不能重复。使用迭代器把最终的结果输出到控制台。

### 操作步骤答案

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

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

// 1.创建HashSet用于保存随机数

HashSet<Integer> hs = **new** HashSet<>();

// 2.创建Random用于产生随机数

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

// 3.使用while循环判断hs的size是否小于10

**while** (hs.size() < 10) {

// 4.hs的size小于10就生成一个随机数

**int** num = ran.nextInt(20) + 1;

// 5.将随机数添加到HashSet,重复的随机数HashSet不会添加

hs.add(num);

}

// 6.获取迭代器

Iterator<Integer> itr = hs.iterator();

// 7.使用迭代器循环判断是否有下一个元素

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

// 8.获取到下一个元素.打印出来

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

}

}

}

## 训练案例8

### 训练知识点

1. LinkedHashSet

### 训练描述

键盘录入一个字符串.去掉其中重复字符, 打印出不同的那些字符.必须保证顺序.例如输入: aaaabbbcccddd,打印结果为: abcd

### 操作步骤答案

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

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

// 1.创建Scanner对象,用于键盘录入

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("请输入一个字符串");

// 2.调用Scanner的nextLine()方法,让用户输入一个字符串

String line = sc.nextLine();

// 3.创建LinkedHashSet.用于去除重复的字符串,并保证迭代顺序

LinkedHashSet<Character> chs = **new** LinkedHashSet<>();

// 4.将字符串串转成char[]

**char**[] charArray = line.toCharArray();

// 5.使用增强for循环遍历每个字符

**for** (**char** c : charArray) {

// 6.将每个字符添加到LinkedHashSet中

chs.add(c);

}

// 7.使用增强for打印LinkedHashSet中的内容

**for** (**char** c : chs) {

System.***out***.print(c);

}

}

}

## 训练案例9

### 训练知识点

1.HashSet

### 训练描述

编写一个程序，键盘录入多个数字,直到录入-1结束.将录入的重复数据去除,计算这些不重复数据的总和和平均值

### 操作步骤答案

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

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

// 1.创建Scanner对象,用于键盘录入

Scanner sc = **new** Scanner(System.***in***);

// 2.创建HashSet.用于去除重复的数字

HashSet<Integer> hs = **new** HashSet<>();

// 3.不停循环,让用户输入数字

**while** (**true**) {

System.***out***.println("请输入一个数字");

// 4.调用Scanner的nextInt()方法,让用户输入一个数字

**int** num = sc.nextInt();

// 5.如果用户输入-1,跳出循环

**if** (num == -1) {

**break**;

}

// 6.不是-1将数字添加到HashSet

hs.add(num);

}

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

// 7.定义求和变量sum

**int** sum = 0;

// 8.使用增强for获取HashSet中的每个元素

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

// 9.将每个数字添加到sum中

sum += i;

}

// 10.输出求和的结果

System.***out***.println("总数是: " + sum);

// 11.输出平均值,平均值 = 总数 / 个数

System.***out***.println("平均值是: " + sum / hs.size());

}

}

## 训练案例10

### 训练知识点

1.HashSet

### 训练描述

有2个ArrayList,元素如下: array1 = "a","b","a","c","d",array2 = "e","f","a","d","g",将两个集合重复项去除后合并成一个集合

### 操作步骤答案

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

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

// 1.创建ArrayList array1

ArrayList<String> array1 = **new** ArrayList<String>();

// 2.使用add方法为array1存放对应的元素

array1.add("a");

array1.add("b");

array1.add("a");

array1.add("c");

array1.add("d");

// 3.创建ArrayList array2

ArrayList<String> array2 = **new** ArrayList<String>();

// 4.使用add方法为array2存放对应的元素

array2.add("e");

array2.add("f");

array2.add("a");

array2.add("d");

array2.add("g");

// 5.创建HashSet用于存放去除重复的元素

HashSet<String> hs = **new** HashSet<>();

// 方法1,简单方法

// hs.addAll(arr1);

// hs.addAll(arr2);

// 方法2

// 6.遍历array1,获取到每个元素,将每个元素添加到HashSet中

**for** (String str : array1) {

hs.add(str);

}

// 7.遍历array2,获取到每个元素,将每个元素添加到HashSet中

**for** (String str : array2) {

hs.add(str);

}

// 8.输出HashSet中去除重复后的元素

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

}

}

## 训练案例11

### 训练知识点

1.HashSet

### 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HashSet中(姓名和年龄相同的认为是同一学生).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

Student类:

// 1.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.hashCode方法和equals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;

}

**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** getScore() {

**return** score;

}

**public** **void** setScore(**double** score) {

**this**.score = score;

}

@Override

**public** **int** hashCode() {

**final** **int** prime = 31;

**int** result = 1;

result = prime \* result + age;

result = prime \* result + ((name == **null**) ? 0 : name.hashCode());

**return** result;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

Student other = (Student) obj;

**if** (age != other.age)

**return** **false**;

**if** (name == **null**) {

**if** (other.name != **null**)

**return** **false**;

} **else** **if** (!name.equals(other.name))

**return** **false**;

**return** **true**;

}

}

测试类:

**public** **class** Test11 {

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

// 1.创建HashSet,用于存放学生

HashSet<Student> hs = **new** HashSet<>();

// 2.使用add方法添加5个学生

hs.add(**new** Student("王昭君", 21, 99));

hs.add(**new** Student("西施", 21, 88));

hs.add(**new** Student("杨玉环", 20, 77));

hs.add(**new** Student("貂蝉", 19, 66));

hs.add(**new** Student("貂蝉", 19, 89));

hs.add(**new** Student("貂蝉", 18, 89));

// 3.使用迭代器获取一个学生.

Student student = hs.iterator().next();

// 4.定义总分变量

**double** totalScore = 0;

// 5.定义最高分变量,初始化值为迭代出来的学生的成绩

**double** maxScore = student.getScore();

// 6.定义最低分变量,初始化值为迭代出来的学生的成绩

**double** minScore = student.getScore();

// 7.使用增强for循环获取遍历HashSet,获取到每个学生

**for** (Student stu : hs) {

// 8.在增强for中将学生的分数添加到总分

totalScore += stu.getScore();

// 9.在增强for中判断最高分是否小于这个学生的分数,如果最高分小于这个学生的分数,将这个学生的分数设置为最高分

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

maxScore = stu.getScore();

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

// 10.在增强for中判断最低分是否大于这个学生的分数,如果最低分大于这个学生的分数,将这个学生的分数设置为最低分

minScore = stu.getScore();

}

}

// 11.计算平均分(平均分 = 总分 / 人数)

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

// 12.按照格式打印

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

}

}

## 训练案例12

### 训练知识点

1. ArrayList
2. 正则表达式

### 训练描述

定义一个方法filterChars(ArrayList<String> list)将传入的ArrayList<String>集合中的每个元素中的字母过滤掉,比如list有2个元素{"hello123java", "11world555java"}，返回结果{"123", "11555"}

### 操作步骤答案

**public** **class** Test12 {

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

// 1.定义ArrayList strs存放元素

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

// 2.使用add方法往ArrayList中添加元素,元素包含字母和数字

strs.add("hello123java");

strs.add("123java");

strs.add("hello123");

strs.add("hello123ja45va");

strs.add("111hel0lo123java");

// 7.调用filterChars方法

*filterChars*(strs);

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

}

// 3.定义filterChars(ArrayList<String> list)方法

**public** **static** **void** filterChars(ArrayList<String> list) {

// 4.在filterChars方法中使用for循环拿出每个元素

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

String str = list.get(i);

// 5.替换掉字符串中所有的字母

String newStr = str.replaceAll("[a-zA-Z]+","");

// 6.使用替换后的字符串覆盖原来的字符串

list.set(i, newStr);

}

}

}

# 关卡3

## 训练案例

### 训练知识点

1.HashSet

### 训练描述

双色球规则:双色球每注投注号码由6个红色球号码和1个蓝色球号码组成。红色球号码从1—33中选择；蓝色球号码从1—16中选择.请随机生成一注双色球号码

### 操作步骤答案

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

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

// 1.创建Random随机数对象

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

// 2.创建HashSet用于保存不重复的红球

HashSet<Integer> hs = **new** HashSet<>();

// 3.循环判断红球数量是否小于6个

**while** (hs.size() < 6) {

// 4.球数量小于6个就产生一个红球.添加到HashSet中

**int** num = ran.nextInt(33) + 1;

hs.add(num);

}

// 5.在生成一个红球

**int** blueBall = ran.nextInt(16) + 1;

// 6.打印中奖号码

System.***out***.println("双色球中奖号码:");

System.***out***.print("红球是: ");

**for** (Integer redBall : hs) {

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

}

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

System.***out***.println("蓝球是: " + blueBall);

}

}

## 训练案例2

### 训练知识点

1. HashSet
2. 增强for

### 训练描述

一个学科中有若干班级，每一个班级又有若干学生。整个学科一个大集合，若干个班级分为每一个小集合(集合嵌套之HashSet嵌套HashSet)。要求如下

1、 学生类有两个属性，姓名和年龄，并定义有参构造、无参构造和getter/setter方法.姓名和年龄相同的视为同一学生

2、向班级集合中添加若干个学生（至少两个学生）。

2、向学科集合中添加所有班级(至少创建两个班级)。

3、使用两种方法遍历学科集合，并打印出所有学生。

### 操作步骤答案

Student类:

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

// 姓名

**private** String name;

// 年龄

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

**public** Student() {

}

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

**super**();

**this**.name = name;

**this**.age = age;

}

**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;

}

@Override

**public** **int** hashCode() {

**final** **int** prime = 31;

**int** result = 1;

result = prime \* result + age;

result = prime \* result + ((name == **null**) ? 0 : name.hashCode());

**return** result;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

Student other = (Student) obj;

**if** (age != other.age)

**return** **false**;

**if** (name == **null**) {

**if** (other.name != **null**)

**return** **false**;

} **else** **if** (!name.equals(other.name))

**return** **false**;

**return** **true**;

}

}

测试类:

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

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

// 1.创建HashSet作为一个班级集合储存学生

HashSet<Student> class1 = **new** HashSet<Student>();

class1.add(**new** Student("zhangsan", 20));

class1.add(**new** Student("lisi", 25));

class1.add(**new** Student("lisi", 25));

// 创建HashSet作为一个班级集合储存学生

HashSet<Student> class2 = **new** HashSet<Student>();

class2.add(**new** Student("wanwu", 23));

class2.add(**new** Student("zhaoli", 27));

// 创建HashSet<HashSet<Student>>作为一个学科集合储存班级

HashSet<HashSet<Student>> xueKe = **new** HashSet<HashSet<Student>>();

xueKe.add(class1);

xueKe.add(class2);

System.***out***.println("--------------增强for遍历----------------");

// 使用两种方法遍历学科集合，并打印出所有学生。

// 增强for

**for** (HashSet<Student> cls : xueKe) {

**for** (Student student : cls) {

System.***out***.println(student.getName() + "---" + student.getAge());

}

}

System.***out***.println("--------------迭代器遍历----------------");

// 迭代器遍历

Iterator<HashSet<Student>> xueKeItr = xueKe.iterator();

**while** (xueKeItr.hasNext()) {

HashSet<Student> cls = xueKeItr.next();

Iterator<Student> clsItr = cls.iterator();

**while** (clsItr.hasNext()) {

Student stu = clsItr.next();

System.***out***.println(stu.getName() + "---" + stu.getAge());

}

}

}

}

## 训练案例3

### 训练知识点

1.循环

### 训练描述

有一堆零件数量不超过1000个,如果4个零件一组最后剩余2个,如果7个零件一组最后剩余3个,如果9个零件一组最后剩余5个,请计算这堆零件个数有几种可能

### 操作步骤答案

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

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

**int** count = 0;

**for** (**int** i = 6; i < 1000; i++) {

**if** ((i % 4 == 2) && (i % 7 == 3) && (i % 9 == 5)) {

count++;

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

}

}

System.***out***.println("共有 " + count + " 情况");

}

}

## 训练案例4

### 训练知识点

1.嵌套循环

### 训练描述

将50元兑换成5元,2元和1元.每种面额的钱不能为0张.共有几种兑换方法.每种兑换方法的钱是怎么样的?

### 操作步骤答案

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

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

// 假设i是5元的,j是2元的,k是1元的

**for** (**int** i = 1; i < 10; i++) { // 5元的

**for** (**int** j = 1; j < 25; j++) { // 2元的

**for** (**int** k = 1; k < 50; k++) { // 1元的

**if** (i \* 5 + j \* 2 + k == 50) {

System.***out***.println("5元的 " + i + " 张,2元的 " + j + " 张,1元的 " + k + " 张");

}

}

}

}

}

}