第 5 章　数　　组

任务　歌手大奖赛评分程序

5.1　任务描述

在歌手大赛中(见图 5-1),经常会有这样的场景:歌手演唱完毕,请评委亮分,评委打完分后,主持人最后宣布,去掉一个最高分,去掉一个最低分,该选手最后平均得分为 90 分。本章的任务是要你来设计实现这样一款评分程序,假设有 10 个评委为参赛的选手打分,分数为 1～100 分,选手最后得分为:去掉一个最高分和一个最低分的其余 8 个分数的平均值。

图 5-1　歌手大奖赛

5.2　任务分析

根据任务描述,可以抽出如下数学模型:

该选手最后得分=(10 个评委打分之和－最高分－最低分)/8

我们要解决的第一个问题是得出 10 个评委打分之和,这是典型的累加和问题,相信你很快就能写出如下代码:

```
double sum=0;                    //存放最终得分
for(int i=1;i<=10;i++){
    //读入评委打分
    sum+=score;
}
```

这样,sum 中存放的就是 10 个评委打分之和。

那接下来,如何去掉最高分呢?

你会发现上面虽然实现了求和,但是最终只有最后一个评委的打分被记录下来存在了 score 变量中,之前评委的打分全被覆盖了。此时再想求 10 个分数中的最高分就必须重新

输入评委的打分，然后进行比较，比较的过程也简直是噩梦：

```
double max=score1;                    //假定第一个数是最大的
if(score2>max)
    max=score2;
if(score3>max)
    max=score3;
if(score4>max)
    max=score4;
…
```

我实在不忍心继续写下去了，像这种做法，如果 20 个评委呢？那又得添加一堆的代码，看起来这是件很恐怖的事情。幸亏 Java 语言为人们提供了能够存放大量数据的容器，它的出现可以使人们在内存中一下子开辟一片连续的存储空间，而不是像上面的做法把成绩一个个地保存在离散的变量里面。这个容器就好比是大家宿舍里的书柜，有了书柜我们可以整整齐齐地把书放上去而不至于到处乱扔。大家一定迫不及待地想要知道这个容器究竟是什么了吧？那就是**数组**。数组的特点有两个。

（1）数组里面存放的是同类型的数据。

（2）数组是有下标的，并且这个下标是从 0 开始的。自从有了数组，程序员数数就是从 0 开始数了。

通过本章的学习，读者将掌握在 Java 中如何声明和创建一维数组，掌握数组的初始化和数组的遍历，理解 Java 中的内存分配情况，掌握如何求最值问题，掌握经典的排序算法——冒泡排序和选择法排序，掌握二分查找法，了解二维数组和对象数组的创建，本章大家还将了解 Arrays 类常用方法的使用，大家会发现，有了它排序和查找问题将变得如此简单。

5.3 相关知识

5.3.1 一维数组的声明和创建

1. 一维数组的声明

要想使用数组，首先要对数组进行声明。在 Java 中，声明一维数组的语法格式如下：

```
数据类型[] 数组名;                    //声明数组
```

或

```
数据类型 数组名[];                    //声明数组
```

以上两种格式都可以声明一个数组，其中数据类型既可以是基本的数据类型（byte、short、int、long、float、double、char、boolean），又可以是引用数据类型，在 Java 中除了 8 种基本的数据类型，其他所见到的都是引用数据类型。在这里读者可以将引用数据类型想象成电视机的遥控器，通过遥控器可以遥控电视机。在本章知识拓展对象数组的创建和使用一节可以使大家进一步地理解引用数据类型的含义。数组名可以是任意合法的标识符。

声明数组的目的就是告诉计算机该数组中存放的数据类型是什么。通过数组的声明可

知，数组中存放的都是同一数据类型的数据。例如：

```
float[] score;     //声明一个存放成绩的数组，数据类型为 float，数组名为 score
String[] fruits    //声明一个存放水果名称的数组，数据类型为 String，数组名为 fruits
```

注意

对这两种语法格式而言，在开发中优先选用第一种格式。因为第一种格式不仅具有更好的语意，也具有更好的可读性。对于“数据类型[]数组名;”方式，一看数据类型后面跟着一个中括号，我们就知道声明的是一个数组，而不是普通的变量。但第二种格式“数据类型数组名[];”的可读性就差了，看起来好像定义了一个类型为“数据类型”的变量，而变量名是“数组名[]”，这与真实的含义相去甚远。可能有些读者非常喜欢第二种格式，从现在开始就不要再使用这种糟糕的方式了。C＃就不再支持第二种声明数组的语法，它只支持第一种声明数组的语法。

另外，在 Java 语言中声明数组时不能指明数组的长度，即数组中元素的个数。例如：

```
float score[5];                //非法，编译通不过
```

2. 一维数组的创建

声明数组只是得到了一个存放数组的变量，但并没有真正为数组元素分配内存空间，所以此时还不能使用数组。如何将数组真正地创建出来呢？Java 中给人们提供了一个很形象的关键字——new。其语法格式如下：

```
数组名=new 数据类型[数组长度];          //创建数组，为数组开辟了存储空间
```

其中，数组长度就是数组中能够存放的元素的个数，显然数组长度应该是正整数。例如：

```
score=new float[5];
fruits=new String[10];
```

当然，也可以在声明一个数组的同时将数组创建出来。其语法格式如下：

```
数据类型 数组名[]=new 数据类型[数组长度];
```

其中，数组长度一旦声明就不能再修改了。例如：

```
float[] score=new float[5];       //声明一个名为 score 的数组，同时开辟了 5 个小格子
String[] fruits=new String[10];   //声明一个名为 fruits 的数组，同时开辟了 10 个小格子
```

5.3.2 Java 中的内存管理

如果声明了一个基本数据类型的变量，就会在内存中开辟一块存储空间，那么声明完一个数组并且创建后，它在内存中又是怎么样的呢？为了分析数组在声明和创建时的内存变化情况，首先来简单地了解一下 Java 中的内存管理。

Java 把内存区域划分为好几块：栈区、堆区、代码区和常量池区。

暂且不管代码区和常量池区，主要来看栈区和堆区。

栈区：主要存放的是人们在程序中所定义的一些基本数据类型的变量以及引用数据类型的变量。

堆区：所有 new 出来的东西全都存放在堆区。

Java 中的内存管理主要管理的是内存的分配和释放。

分配：内存的分配是由程序完成的，对于基本数据类型，通过声明变量，直接就会在内存中开辟相应大小的存储空间；对于引用数据类型，程序员需要通过关键字 new 动态地申请内存空间。

释放：在栈中分配的内存，当超出变量的作用域后会自动地释放该变量所分配的内存空间；在堆中分配的内存空间，由 Java 虚拟机的垃圾回收机制负责其空间的释放。

5.3.3 一维数组内存分析

了解了 Java 中内存管理之后，围绕下面的例子来看一下一维数组在声明和创建时的内存变化情况。

【例 5-1】 声明一个存放 5 个整数的数组并创建数组，输出数组中每个元素的值。

【问题分析】

声明数组：由于这个数组是存放整数的，所以根据数组声明的语法，可以进行如下声明：

```
int[] num;
```

此时内存情况是怎样的呢？Java 虚拟机会为 num 在栈内存开辟一块存储空间，如图 5-2 所示。

创建数组：声明数组只是在栈内存中开辟了一块存储空间，一块空间里面只能存放一个数值，肯定是不能存下 5 个整数的。只有真正地创建数组，才会开辟能够存放 5 个整数的连续的存储空间，即执行“num＝new int[5];”。大家想想此时这片连续的存储空间是开辟在内存什么地方呢？根据 5.3.2 节对 Java 内存管理的学习可知，所有 new 出来的东西都是分配在堆内存中的。因此，此时的内存变化情况如图 5-3 所示。

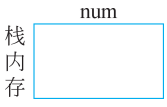

图 5-2 声明 num 数组时的内存情况

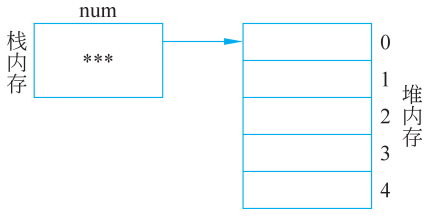

图 5-3 创建数组后内存的变化情况

此时 num 变量里面究竟存了什么值使人们通过它可以访问堆内存中开辟的连续空间的内容呢？让我们一起将 num 的值输出来探探究竟。

【程序实现】

```
//ArrayPrintDemo.java
public class ArrayPrintDemo {
    public static void main(String[] args) {
        int[] num;                                  //声明数组 num
        num=new int[5];                             //创建数组
        System.out.println("num="+num);             //输出数组名
    }
}
```

【运行结果】

```
num=[I@c17164
```

【程序说明】

程序运行后，我们发现一个奇怪的结果[I@c17164，这是什么呢？这是JVM给堆内存中那片连续存储空间分配的首地址。是不是感觉恍然大悟了呢？因为存了首地址，人们通过数组名就可以找到堆内存中开辟的空间的首地址了，而数组中空间是连续的，这样就可以依次找到数组中的每个元素。

5.3.4 数组的遍历

1. 数组元素的访问

由于数组名里面存放的是一个地址值，那么堆内存的那些小格子里面到底存放的是什么呢？如何能够输出每个小格子里面的内容呢？即如何访问每个数组元素呢？在Java中，访问数组元素的语法格式如下：

```
数组名[下标值];
```

如num[0]表示数组的第1个元素。

【例5-2】 输出例5-1所创建数组的每个元素的值。

【程序实现】

```
public class ArrayPrintDemo {
    public static void main(String[] args) {
        int[] num;
        num=new int[5];
        System.out.println("num="+num);
        System.out.println("数组中的元素为：");
        System.out.println(0+"\t"+num[0]);
        System.out.println(1+"\t"+num[1]);
        System.out.println(2+"\t"+num[2]);
        System.out.println(3+"\t"+num[3]);
        System.out.println(4+"\t"+num[4]); }
    }
```

【运行结果】

```
num=[I@c17164
```

数组中的元素为：

```
0    0
1    0
2    0
3    0
4    0
```

注意

通过输出结果，我们发现创建了数组后，数组中每个元素是有初始值的，这个值称为数组的默认值。这个默认值与声明数组时所指定的数据类型有关：如果是基本数据类型，如int类型的默认值为0，float类型的默认值为0.0，boolean类型的默认值为false；如果是引用

数据类型，其默认值为 null。读者可以通过程序验证一下。

2. 一维数组的遍历

观察一下例 5-2 可发现，以下这段代码是非常有规律的：

```
System.out.println(0+"\t"+num[0]);
System.out.println(1+"\t"+num[1]);
System.out.println(2+"\t"+num[2]);
System.out.println(3+"\t"+num[3]);
System.out.println(4+"\t"+num[4]);
```

如果定义一个变量 i，这段代码就可以统一成：System.out.println(i+"\t"+num[i])，那么这条语句就变成了可以重复执行的语句，套上循环结构，由于事先能够确定循环 5 次，所以使用 for 结构进行一维数组的遍历最合适。

使用 for 循环结构重新改写例 5-2 的这一段代码：

```
for (int i=0; i<5; i++) {
    System.out.println(i+"\t"+num[i]);
}
```

在第 4 章循环结构的知识拓展部分曾提到过 for-each 结构，对数组的遍历输出也可以使用这个结构。

```
for(int n:num){
    System.out.println(n);
}
```

它可以理解为“在每次循环中，将 num 数组的下一个元素赋给变量 n，然后对其进行输出”。可见 for-each 结构更加简化了遍历数组的代码。但要注意的是，这种结构只能用来访问数组的元素，而不能对元素进行修改，因为它无法访问数组元素的下标。

5.3.5 一维数组的初始化

如何给数组中的每个元素赋值呢？可以通过以下 3 种方式。

1. 数组声明与为数组元素分配空间并赋值的操作分开

例如：

```
int[] num;
num=new int[5];
num[0]=10;
num[1]=20;
num[2]=30
…
```

缺点：一个个地赋值太麻烦了。

2. 在声明数组的同时为数组元素分配空间并赋值

其语法格式如下：

```
数据类型[] 数组名={值 1,值 2,值 3,…,值 n};
```

例如：

```
int[] num={10,20,30,40,50};
```

这个声明没有用 new 来创建数组对象。当编译器遇到包含有初始化表的数组声明时，会自动计算表中元素的数量，并将它作为数组的大小，然后在“幕后”进行合适的 new 操作。

3. 从控制台接收键盘输入进行循环赋值

```
Scanner input=new Scanner(System.in);
for(int i=0;i<5;i++){
    num[i]=input.nextInt();
}
```

注意

直接创建并赋值的方式必须一并完成，如下代码是不合法的：

```
int[] num;
num={10,20,30,40,50};                    //编译报错
```

【例 5-3】 要求 20 名同学对学生食堂饭菜的质量进行 1～5 的评价(1 表示很差，5 表示很好)。将这 20 个结果输入整型数组，并对打分结果进行分析。

【问题分析】

我们希望统计出每个分数对应的学生人数。学生最终的打分情况可以借助于一个整型数组 answers 来保存。首先定义一个包含 20 个打分结果的 answers 数组，然后再定义一个包含 6 个元素的 frequency 数组来统计各种评价的次数，frequency 中的每个元素此时都被看成了一个得分的计数器，其默认的初始值为 0。在这里为何要将数组长度定义为 6 呢？我们想让 frequency[1]统计的是分值 1 的次数，frequency[2]统计的是分值 2 的次数，以此类推，这样正好一一对应起来，这里忽略掉 frequency[0]。比如读入第一个学生的评价，他的评价是 5 分，就将 frequency[5]的计数值加 1。当遍历完 answers 数组后，对应的 frequency 数组里面的值就是每个分数对应的学生人数。

【程序实现】

```
1    public class StudentPoll {
2        public static void main(String[] args) {
3            int[] answers={3, 1, 2, 5, 4, 2, 2, 3, 4, 5,
4                           1, 2, 3, 4, 2, 1, 3, 2,4, 2};
5            int[] frequency=new int[6];
6            for (int i=0; i<20; i++) {
7                frequency[answers[i]]++;
8            }
9            System.out.println("分值\t学生数");
10           for (int i=1; i<6; i++) {
11               System.out.println(i+"\t"+frequency[i]);
12           }
13       }
14   }
```

【程序说明】

(1) 第 3 行和第 4 行采用初始化数组中所说的第 2 种方式完成了数组中每个元素的赋值，这个数组是存放学生问卷结果的。

(2) 第 5 行定义了包含 6 个元素的 frequency 数组，默认初始值均为 0。这个数组是统计各个分数对应的学生人数的。学生评价结果取值范围只可能是[1,5]，评价为 1 的让

frequency[1]计数器进行累加，以此类推。

(3) 第 6～8 行循环遍历 answers 数组中的每个元素，将其值 answers[i]取出来，answers[i]的取值只可能是[1,5]，然后将对应的 frequency[answers[i]]的值加 1。

(4) 第 10～12 行遍历 frequency 数组即可知道各个分值对应的学生人数。

【运行结果】

```
分值        学生数
1           3
2           7
3           4
4           4
5           2
```

5.3.6 一维数组的应用

1. 最值问题

【例 5-4】 输入 10 个整数，输出其中的最大值和最小值。

【问题分析】

求最大值最小值的问题，通常采用“打擂台”的方式，先站上来一个人就认为他是最厉害的。然后第 2 个人上来和站在这个擂台上的人打，谁失败了谁下台，这样站在擂台上的就是这两个人中最厉害的。然后第 3 个人上来和站在这个擂台上的人打，谁失败谁就下台，这样站在擂台上的人是 3 个人中最厉害的。然后一直比，直到最后一个人再比时他一定是在和前面已经比过的最厉害的那个人在比。台上剩下的自然就是最厉害的。

回到这个问题，可以进行如下设计。

(1) 定义一个一维数组 num，用于保存输入的 10 个整数。

(2) 假设第一个数 num[0]既是最大的，也是最小的。将其放入 max(大家可以将其想象成是擂台)，即执行“max=num[0];”同时也将其放入 min 中，即执行“min=num[0];”。

(3) 将 num[1]和 max 比较，如果 num[1]>max，则将 max=num[1]；如果 num[1]<min，则将 min=num[1]。显然，这是典型的分支结构。

```
if(num[1]>max)
    max=num[1];
if(num[1]<min)
    min=num[1];
```

这样一来 max 中存放的是两个数中最大的，min 中存放的是两个数中最小的。

(4) 将 num[2]和 max 比较，如果 num[2]>max，则将 max=num[2]；如果 num[2]<min，则将 min=num[2]。

```
if(num[2]>max)
    max=num[2];
if(num[2]<min)
    min=num[2];
```

这样一来 max 中保存的就是 3 个数中最大的，min 中保存的是 3 个数中最小的。

此时我们观察一下(3)和(4)，如果用 i 表示当前遍历的那个元素的下标，则(3)和(4)可

以统一成以下代码：

```
if(num[i]>max)
    max=num[i];
if(num[i]<min)
    min=num[i];
```

上述代码重复执行直到 i 为 9。显然这里要套上循环结构，由于事先能确定循环次数，优先选用 for 循环结构。

(5) 最后 max 中存放的即为 10 个数中最大的，min 中存放的即为 10 个数中最小的。

【程序实现】

```
1   import java.util.Scanner;
2   public class MaxAndMinDemo {
3       public static void main(String[] args) {
4           int[] num=new int[10];
5           Scanner input=new Scanner(System.in);
6           System.out.println("请输入 10 个整数: ");
7           for(int i=0;i<10;i++){
8               num[i]=input.nextInt();
9           }
10           int max=num[0];
11           int min=num[0];
12           for(int i=1;i<10;i++){
13               if(num[i]>max){
14                   max=num[i];
15               }
16               if(num[i]<min){
17                   min=num[i];
18               }
19           }
20           System.out.println("10 个数中最大值为: "+max);
21           System.out.println("10 个数中最小值为: "+min);
22       }
23   }
```

【程序说明】

(1) 第 1 行，导入包语句，将和输入有关的类 Scanner 引入进来。

(2) 第 4 行，声明并创建了一个能够存储 10 个整数的一维数组。

(3) 第 7～9 行，使用 for 循环，接收从键盘中输入的 10 个整数将其存入一维数组。

(4) 第 10 行，假设第一个数(即 num[0])就是最大的。

(5) 第 11 行，假设第一个数(即 num[0])就是最小的。

(6) 第 12～19 行，使用 for 循环依次遍历 num 数组中的每个元素，将其与 max 和 min 进行比较，每次 max 中存放的都是遍历过元素中最大的，min 中存放的都是遍历过元素中最小的。

(7) 第 20 行和第 21 行，输出 max 和 min 的值。

注意

在程序中，有两处出现了数组的长度 10，数组长度一旦要发生改变，这两处都需要进行修改，有没有可能做到“一改全改”呢？通过数组所提供的 length 属性就可以解决这个问

题。究竟属性是什么？大家暂可先不用管。先会用它即可。其语法格式为

```
数组名.length
```

在本例中可以将数组长度的地方替换为 num.length。

【运行结果】

```
请输入 10 个整数：
8 10 7 8 9 6 5 6 8 7
10 个数中最大值为：10
10 个数中最小值为：5
```

2. 排序问题

【例 5-5】 将 5 个数按照从小到大的顺序排列起来。

排序问题可以借助于一些经典的算法来解决。这里给大家介绍两种经典的排序算法：冒泡排序和选择排序。

1）冒泡排序

【算法描述】

冒泡排序(Bubble Sort)算法的基本思想是：依次比较相邻的两个数，将小数放在前面，大数放在后面。即在第一趟：首先比较第一个数和第二个数，将小数放前，大数放后。然后比较第二个数和第三个数，将小数放前，大数放后，如此继续，直至比较最后两个数，将小数放前，大数放后。至此第一趟结束，将最大的数放到了最后。在第二趟：仍从第一个数开始比较，将小数放前，大数放后，一直比较到倒数第二个数(倒数第一的位置上已经是最大的)，第二趟结束，在倒数第二的位置上得到一个新的最大数。如此下去，重复以上过程，直至最终完成排序。由于在排序过程中总是小数往前放，大数往后放，相当于气泡往上升，所以称为冒泡排序。

【问题分析】

假设对 5 个数(9,7,5,8,4)应用冒泡排序，排序的过程如下。

第一趟排序的过程如图 5-4 所示。

经过第一趟排序后，最大数 9 沉底。

第二趟排序的过程如图 5-5 所示。

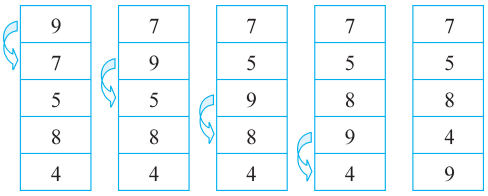

图 5-4　冒泡排序的第一趟排序

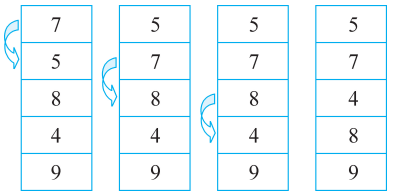

图 5-5　冒泡排序的第二趟排序

在第二趟排序时，只需要比较前 4 个数，因为经过第一趟排序后，9 已经排在了合适的位置。经过第二趟排序后，数字 8 沉底。

第三趟排序的过程如图 5-6 所示。

经过第三趟排序后，数字 7 沉底。

第四趟排序的过程如图 5-7 所示。