第 5 章

显示班级成绩单

学习目标

- 理解为什么使用数组；
- 掌握应用数组进行程序设计；
- 学习字符串类的常用方法。

5.1 示 例 程 序

5.1.1 班级平均成绩

第 4 章介绍了如何计算多个学生的平均成绩。如果想显示一个班学生的平均成绩和每个人的成绩，这时就需要将输入的多个学生成绩进行保存。可以把多个学生的成绩保存到一个数组中，然后进行计算并得到结果。程序代码如程序 5.1 所示。

【程序 5.1】 程序 StudentInfo.java。

```
import java.util. Scanner;
public class StudentInfo{
    public static void main(String [] args){
        final int SIZE = 5;
        double grade[] = new double[SIZE];
        double averageGrade = 0;
        Scanner sc = new Scanner(System.in);
        for(int i=0; i<SIZE; i++){
            grade[i] = sc.nextDouble();
        }
        for(int i=0; i<SIZE; i++){
            averageGrade = averageGrade + grade[i];
        }
        averageGrade = averageGrade / SIZE;
        System.out.println("平均成绩:" + averageGrade);
        for(int i=0; i<SIZE; i++){
            System.out.println("学生成绩:" + grade[i]);
        }
    }
}
```

程序5.1的编译和运行结果如图5.1所示。程序5.1中定义了数组来保存学生的成绩，数组定义语句如下：

```
double grade[] = new double[SIZE];
```

定义double类型的数组grade，数组的长度为SIZE。需要注意的是数组不是基本类型，而是引用类型，因此需要开辟数据的存储空间。new double[SIZE]的作用就是开辟数组需要的存储空间，开辟空间的大小可以存放SIZE个double数值。每个数组元素占用一个double类型数据的存储空间。上面定义的数组下标范围是0到SIZE－1，共计SIZE个元素，一般使用for循环来处理数组中的每一个元素。

```
D:\program\unit5\5-1\1-1>java StudentInfo
78
89.5
66.5
81
93
平均成绩：81.6
学生成绩：78.0
学生成绩：89.5
学生成绩：66.5
学生成绩：81.0
学生成绩：93.0
```

图5.1　程序5.1运行结果

采用循环语句读入每个学生的成绩，语句grade[i] = sc.nextDouble()的作用是把读入的双精度类型成绩数据存放到grade数组的第i个元素中，这样通过循环把所有的成绩都读入到数组grade中。后面的计算和显示过程与前面的相似，循环将数组grade中的元素累加求和，然后计算并输出平均成绩，最后也是通过循环显示数组grade中每个元素的值。由于学生成绩保存在数组grade中，因此最后可以再次把学生成绩显示出来。

5.1.2　显示最高成绩

程序5.1显示了一个班的学生成绩，如果还想知道一个班学生的最高成绩是多少，并把最高成绩显示出来，需要增加一段程序找到最高成绩。改进后的代码如程序5.2所示。

【程序5.2】 修改程序StudentInfo.java，显示最高成绩。

```
import java.util. Scanner;
public class StudentInfo{
    public static void main(String [] args){
        final int SIZE = 5;
        double grade[] = new double[SIZE];
        double averageGrade = 0;
        double maxGrade = 0;
        Scanner sc = new Scanner(System.in);
```

```
            for(int i=0; i<SIZE; i++){
                grade[i] = sc.nextDouble();
            }
             maxGrade = grade[0];
            for(int i=0; i<SIZE; i++){
                averageGrade = averageGrade + grade[i];
                if(maxGrade < grade[i]){
                    maxGrade = grade[i];
                }
            }
            averageGrade = averageGrade / SIZE;
            System.out.println("平均成绩:" + averageGrade);
            System.out.println("最高成绩:" + maxGrade);
            for(int i=0; i<SIZE; i++){
                System.out.println("学生成绩:" + grade[i]);
            }
        }
    }
```

程序 5.2 的编译和运行结果如图 5.2 所示。

```
D:\program\unit5\5-1\1-2>javac StudentInfo.java

D:\program\unit5\5-1\1-2>java StudentInfo
78
89.5
66.5
81
93
平均成绩：81.6
最高成绩：93.0
学生成绩：78.0
学生成绩：89.5
学生成绩：66.5
学生成绩：81.0
学生成绩：93.0
```

图 5.2　程序 5.2 运行结果

找出数组中最大值的方法是，先假设第一个元素是最大的，把它放到变量 maxGrade 中，然后和数组中所有的元素逐一比较，如果发现比 maxGrade 大的元素，就把这个元素放到 maxGrade 中。通过循环，对比数组中所有的元素，最后 maxGrade 中存放的值就是数组中的最大值。

有的读者可能会想：是否可以将最大成绩 maxGrade 的初值设为 0，逐个与学生成绩进行比较，找到最大值。这个方法在本题中是可以的，但是不够安全，有的时候可能会出问题。假设所有的学生成绩都是负的，这时候得到的最大值就有问题了。比较安全的方法是把第一个成绩设置为最大值，然后进行比较。

5.2 相关知识

5.2.1 一维数组

数组是用来存放一组相同类型数据的复合数据类型，常见的数组是一维数组，一维数组声明的格式定义如下：

```
数组元素类型 数组名[ ];
```

或者：

```
数组元素类型[ ] 数组名;
```

例如程序5.2中定义数组double grade[]，数组元素是double类型，数组名为grade。数组定义后，还需要创建数组。创建数组就是为数组分配存储空间，使用关键字new来完成。例如程序5.2中的创建语句grade[] = new double[SIZE]，创建一个有SIZE个double类型元素的数组，数组的名字是grade。

通过下标来访问数组元素，例如程序5.2中的语句maxGrade = grade[i]，读取grade的第i个元素，把它赋给变量maxGrade。

5.2.2 多维数组

同样，也可以定义二维数组或者是多维数组，二维数组的定义格式如下：

```
数组元素类型 数组名[ ][ ];
```

二维数组的创建方法也是相同的，例如double grade[][] = new double[5][6]，表示定义了一个具有5行6列的二维数组。使用二维数组元素的方式也与一维数组相同，通过下标来进行访问，例如：

grade[3][4] = 23.4，表示将数组grade的第3行第4列元素赋值为23.4。

Java程序中使用数组的情况相对C程序要少，多数情况下会使用集合类ArrayList来代替数组，有关集合类的定义和使用参见第21章。

5.2.3 String类

第2章介绍了一种基本类型char，用于存储一个字符。如果程序中需要使用由多个字符组成的字符串，就需要用到String类。例如，语句String name="张三";创建了一个String类的对象name，值是字符串“张三”。创建字符串对象还可以使用String类的构造方法来实现，例如String str2=new String("Hello")。关于构造方法将在第8章讲解。

字符串的常用操作包括字符串比较、求字符串长度、获取指定位置字符等，这些操作都是通过调用String类的相应方法来实现，部分常用的方法如表5-1所示。

表 5-1　String 类的方法

方　法　名	功 能 简 介
char charAt(int index)	返回字符串中第 index 个字符，索引 index 值从 0 开始计算
boolean equals(String stra)	判断当前字符串的内容是否与字符串 stra 内容相同，如相同返回 true，否则返回 false
int indexOf(String stra)	返回 stra 字符串在当前字符串中的索引位置，如果没有找到返回 −1
int length()	返回字符串的长度
String replace(char oldChar, char newChar)	将当前字符串中所有 oldChar 字符替换为 newChar
String substring(int beginIndex, int endIndex)	返回字符串的子串，子串从 beginIndex 开始，直到 endIndex − 1 结束
String toLowerCase()	将字符串中所有大写字母变成小写
String toUpperCase()	将字符串中所有小写字母变成大写
String trim()	去掉字符串两端的空格

字符串的方法还有很多，如果想要了解更多的方法，可以查看 String 类的 API 文档。表 5-1 中的方法是程序设计中最常用的方法。下面给出一个例子来演示字符串的使用，如程序 5.3 所示。

【程序 5.3】 字符串常用方法示例。

```
public class StringTest {
    public static void main(String[] args) {
        String s0 = "Program";
        String s1 = "Java " + s0;                    //字符串连接
        String s2 = "Java program";                  //注意 p 小写
        String s3 = "Java Program";                  //注意 P 大写
        String s4 = "Java 程序设计";
        //字符串比较
        System.out.println("----字符串比较----");
        System.out.println("s1 和 s2 的比较结果为 " + s1.equals(s2));
        System.out.println("s1 和 s3 的比较结果为 " + s1.equals(s3));
        //字符串长度
        System.out.println("----字符串长度----");
        System.out.println(s2.length( ));
        System.out.println(s4.length( ));
        //获取字符
        System.out.println("----获取字符----");
        System.out.println(s1.charAt(0));
        //字符替换
        System.out.println("----字符替换----");
```

```
        System.out.println(s2.replace('J','j'));
        //大小写转换
        System.out.println("----大小写转换----");
        System.out.println(s2.toUpperCase( ));
        System.out.println(s3.toLowerCase ( ));
        //子串查找
        System.out.println("----子串查找----");
        System.out.println(s1.indexOf(s0));
        System.out.println(s1.indexOf("a"));
        System.out.println(s1.indexOf("abc"));
    }
}
```

程序 5.3 的运行结果如图 5.3 所示。

```
D:\program\unit5\5-2\2-1>java StringTest
----字符串比较----
s1和s2的比较结果为 false
s1和s3的比较结果为 true
----字符串长度----
12
8
----获取字符----
J
----字符替换----
java program
----大小写转换----
JAVA PROGRAM
java program
----子串查找----
5
1
-1
```

图 5.3　程序 5.3 运行结果

需要特别说明的是，比较两个字符串是否相等时，建议使用 equals()方法，而不是使用等号“==”，关于二者的区别将在第 8 章中讲解。

5.3　训 练 程 序

输入一个班学生的成绩，先显示所有及格成绩，再显示所有不及格成绩；最后显示及格人数和不及格人数。

5.3.1　程序分析

在程序 5.1 基础上来完成这个程序，同样还是使用数组来保存学生成绩，使用循环输入学生成绩。再设计一个循环判断每个学生的成绩是否及格，如果及格则输出，这样就可以显示所有及格的成绩。同理可以显示所有不及格的成绩。定义两个计数器分别用来记录及格学生人数和不及格学生人数。

5.3.2　参考程序

【程序 5.4】 程序 StudentInfo.java。

```
import java.util. Scanner;
public class StudentInfo{
    public static void main(String [] args){
        final int SIZE = 5;
        double grade[] = new double[SIZE];
        int pass=0;                          //定义变量 pass 对及格成绩进行计数
        int fail=0;                          //定义变量 fail 对不及格成绩进行计数
        System.out.println("请输入" + SIZE+"个学生的成绩");
        Scanner sc = new Scanner(System.in);
        for(int i=0; i<SIZE; i++){
            grade[i] = sc.nextDouble();
        }
        //统计及格人数,并输出成绩
        System.out.println("及格的学生成绩:");
        for(int i=0;i<SIZE;i++){
            if (grade[i]>=60){
                pass++;
                System.out.println(grade[i]);
            }
        }
         //统计不及格人数,并输出成绩
        System.out.println("不及格的学生成绩:");
        for(int i=0;i<SIZE;i++){
            if (grade[i]<60){
                fail++;
                System.out.println(grade[i]);
            }
        }
        System.out.println("及格的学生有"+pass+"人");
        System.out.println("不及格的学生有"+fail+"人");
    }
}
```

程序 5.4 的编译和运行结果如图 5.4 所示。

需要对输入的学生成绩进行两遍处理,先使用循环处理所有的成绩,显示及格的学生成绩;再次循环处理所有成绩,显示不及格学生成绩。为了处理方便,使用数组将学生的成绩保存起来。程序中先输入 5 个学生成绩,显示学生成绩的同时统计及格学生人数,不及格学生人数,最后显示出来。

```
D:\program\unit5\5-3\3-1>javac StudentInfo.java

D:\program\unit5\5-3\3-1>java StudentInfo
请输入5个学生的成绩
78
66.5
59
81
93
及格的学生成绩：
78.0
66.5
81.0
93.0
不及格的学生成绩：
59.0
及格的学生有4人
不及格的学生有1人
```

图 5.4　程序 5.4 运行结果

5.3.3　进阶训练

【程序 5.5】 计算圆的面积进阶 4。程序 4.7 中我们实现了多次连续计算圆的面积，现在我们使用数组实现一次性的输入多个圆半径，半径保存在数组中，程序将所有圆的面积计算出来并输出。

程序设计思路分析：

(1) 首先输入圆的个数 n；

(2) 定义长度为 n 的数组 r 和 area，分别存储 n 个圆的半径和面积；

(3) 利用循环，输入各个圆的半径并存入数组 r；

(4) 利用循环，依次从 r 中取出半径值，并计算圆的面积，计算结果存放在数组 area 中；输出计算结果；

(5) 本题也可以将两个循环合并为一个。

读者按照给出的设计思路，完成程序 5.5 的编写和编译，运行程序，查看结果是否符合题目要求。

5.4　拓 展 知 识

5.4.1　数组讨论

数组访问中经常会遇到的问题是数组下标越界。例如程序 5.1 中定义的数组 double grade[] = new double[SIZE]，这个数组的下标范围是 0 到 SIZE－1，共计 SIZE 个元素。常见的错误是访问了下标为 SIZE 的数组元素，这样程序运行时就会报错，例如程序段：

```
int SIZE = 5;
double grade[] = new double[SIZE];
double averageGrade = 0;
grade[SIZE] = 96;
```

这个程序段运行时会抛出异常，如图 5.5 所示。

```
D:\program\unit5\5-3\3-1>javac StudentInfo.java

D:\program\unit5\5-3\3-1>java StudentInfo
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
        at StudentInfo.main(StudentInfo.java:7)
```

图 5.5　数组下标越界异常

这个例子说明两个问题。第一个问题是使用数组时，应该注意代码中是否存在下标越界。上面例子比较简单，实际应用中，数组下标可能是复杂的表达式或方法的返回值，这时候需要小心处理。第二个问题是如果出现数组下标越界的情况应该有相应的异常处理机制，这样可以保证程序的正确性和有效性。有关异常处理机制参见第 17 章。

一般对数组进行处理时经常使用 for 循环，例如程序 5.1 中的程序段：

```
for(int i=0; i<SIZE; i++){
    averageGrade = averageGrade + grade[i];
}
```

这是常见的程序段，使用循环变量 i 来访问数组 grade 中的每个元素。需要注意一般在循环体中不要修改循环变量 i 的值。如果在循环中不小心对变量 i 的值进行了修改，程序运行可能会出问题。例如上面程序段修改为：

```
for(int i=0; i<SIZE; i++){
    averageGrade = averageGrade + grade[i];
     i++;
}
```

在这段程序中，循环变量 i 的值在循环体中被修改，因此得到的结果会出现错误。这也是使用循环处理数组经常遇到的问题。

5.4.2　数组的存储

程序中定义的每一个变量，到了程序运行期间都会对应一个内存单元，变量的值保存在内存单元中，关于变量定义的详细说明见第 2 章。数组本质上就是一组变量，每个数组元素都是一个变量。一个数组变量可以存放多个相同类型的数据。例如，程序 5.4 中定义的数组变量 grade，定义如下：

```
int SIZE = 5;
double grade[] = new double[SIZE];
```

和普通变量类似，这段代码运行时数组变量 grade 会在内存中对应一段存储区域，里面包括 SIZE(图中的 SIZE 为 5)个存储单元，示意图如图 5.6 所示。数组 grade 占用 5 个连续的存储空间，每个存储空间分别对应下标为 0～4 的数组元素。如果想给下标为 2 的

数组元素赋一个双精度的数值 67.8，则使用语句 grade[2]＝67.8，通过下标实现对数组元素的访问。

图 5.6 只是一个示意图，用来说明一个数组在内存中的存储方式。在实际的 Java 虚拟机中，变量 grade 是保存在运行栈的数据区中，而图 5.6 中数组的存储空间则是在堆区。定义数组的语句：

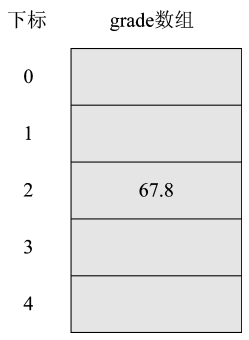

图 5.6　数组内存示意图

```
double grade[] = new double[SIZE]
```

表示的具体含义是，定义一个 double 类型数组，数组名字是 grade，变量 grade 保存在栈上。new double[SIZE]表示在堆中开辟一个存储空间，大小是 SIZE 个 double 类型的存储空间。定义语句中的“＝”的作用是让 grade 指向堆中开辟的空间。相关内容参考第 8 章中对象和实例部分的讲解。最后做个简单的总结，本章学习了关键字 new，用于创建数组空间，表 5-2 中用粗体表示。其余关键字将在后续章节讲解。

表 5-2　Java 关键字

abstract	extends	native	return	throw
assert***	finally	**new**	static	throws
catch	implements	package	strictfp**	transient
class	import	private	super	try
default	instanceof	protected	synchronized	void
enum****	interface	public	this	volatile

5.5　实做程序

1. 定义一个数组来保存教师工资，编写程序找出并显示最高工资。要点提示，参考程序 5.2 的实现。

2. 定义一个数组来保存教师工资，编写程序找出并显示最高工资，指出是第几个工资最高。要点提示：

（1）参考程序 5.2 的实现；

（2）增加一个变量 k 来记录数值中最高工资对应的下标。

3. 定义一个数组来保存教师工资，把最高工资换到数组中的第一个位置，输出交换后的数组。要点提示：

（1）找出最高工资和对应下标，例如 s[i]；

（2）与第一个交换，交换是 s[0]和 s[i]，可以使用中间变量 temp 实现。

4. 在实做程序 3 基础上修改程序，将教师工资按从高到低顺序进行排序。要点提示：