JAVA

8



## 4.4　数　　组

数组是 Java 语言以及其他编程语言的一种[illegible]数据结构，它[illegible]面做单个介绍。8 种基本数据类型，当处理一系列的同类数据时，利用数组来操作会非常方便。

### 4.4.1　数组的概念

在解决现实问题时，经常需要处理一批类似的数据，如对 6 位同学[illegible]成绩进行处理，如果利用基本数据类型的话，那就必须定义 6 个变量：result1、result2、result3、result4、result5 和 result6。如果有 60 位同学，那就需要定义 60 个基本数据类型的变量了，这是很不合理的。

为了便于处理一批同类型的数据，Java 语言引入了数组类型，以处理像线性表、矩阵等结构的数据。数组类型是由其他基本数据类型按照一定的组织规则构造出来的带有分量的构造类型，即数组是由具有相同数据类型的分量组成的结构。

以用 result[0]，result[1]，result[2]，…，result[57]，result[58]，result[59]来表示。

在 Java 中，数组是一种特殊的对象，数组与对象的使用一样，都需要定义、创建和释放。在 Java 语言中，[illegible]方式来获取所需的存储空间，或者用直接初始化的方式来创建，而对存储空间的释放则由垃圾收集器自动回收。

数组作为一种特殊的数据类型，具有以下特点：首先，数组中的每个元素都是相同数据类型的；其次，数组中的这些相同数据类型的元素是通过数组下标来标识的，并且该下标从 0 开始；最后，数组元素在内存中是连续存放的。

下面介绍常用的一维数组和二维数组(也可以称之为多维数组)的声明与创建。

### 4.4.2 数组的声明和创建

#### 1. 一维数组

一维数组的声明格式如下：

```
数据类型 [ ] 数组名;
```

或：

```
数据类型  数组名[ ];
```

其中，数据类型指明了数组中各元素的数据类型，包括基本数据类型和[illegible]类型(如数组或类)；数组名为一个合法的标识符，"[ ]"指示该变量为数组类型变量。如下的声明语句：

```
short [ ] x
```

或：

```
short  x[ ];
```

以上两种定义格式都是正确的，表示声明了一个短整型的数组，数组名为 x，数组中的每个元素的类型都是 short。需要注意的是，Java 语言在定义数组时，不能为一维数组指定数组元素的个数，如下面的声明语句是错误的：

```
short  x[60];
```

数组元素的数目[illegible]，这点与很多其他编程语言不同。那么，[illegible]

创建数组后，Java 语言规定，创建数组可以有两种方式：初始化方式和 new 操作符。初始化方式是指直接给数组的每一个元素指定一个初始值，系统自动根据所给出的数为数组分配相应的存储空间，这种创建数组的方式适用于数组元素较少的情形。其形式如下：

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

下面的语句分别定义并创建了一个含有 6 个元素的短整型数组和一个含有 6

的字符数组：

```
short  x[ ] = {1, 2, 3, 4, 5, 6};
char  ch[ ] = {'a', 'b', 'c', 'd', 'e', 'f'};
```

然而，先定义数组，再创建初始化数组则是错误的，例如，上述语句如果改写成下面的形式则是错误的：

```
short  x[ ];
x = {1, 2, 3, 4, 5, 6};  //编译出错
```

对于数组较大的情况，即数组元素较多时，用初始化方式显然不妥，这时就应采用第二种方式，即 new 操作符方式。其一般形式如下。

```
数据类型  数组名[ ] = new  数据类型[元素个数];
```

或：

```
数据类型  数组名[ ];
数组名= new  数据类型[元素个数];
```

利用 new 操作符方式创建的数组元素会自动被初始化为一个默认值：对于整型，默认值为 0；对于浮点型，默认值为 0.0；对于布尔型，则默认为 false 等。当然，在创建完数组后，用户也可以通过正常的访问方式对数组中的元素进行赋值，例如：

```
short  x[ ] = new short[6];
x[0] = 9;
x[1] = 8;
x[2] = 7;
x[3] = 6;
x[4] = 5;
x[5] = 10;
```

注意：

该数组的元素个数是 6 个，因此下标为从 0 至 5，千万不要对其他下标的元素进行访问，如 x[6]，x[10]等将会发生数组下标越界的错误。

一般地，数组的元素个数称为该数组的长度，它可以通过数组对象的 length 属性来获取。请看如下程序段：

```
short x[ ] = new short[6];
int len = x.length;
for(int i=0;i<len;i++)   //通过循环给每个数组元素赋值
  x[i]=i*2;
for(int i=0;i<len;i++)   //通过循环输出每个数组元素的值
  System.out.print(x[i] + "  ");
```

上述程序运行结果如下：

```
0  2  4  6  8  10
```

由此可以看出，利用数组对象的length属性可以很方便地实现遍历访问数组的每一个元素。

### 2. 二维数组

二维数组的声明格式如下：

```
数据类型 [ ][ ] 数组名;
```

或：

```
数据类型 数组名[ ][ ];
```

其中，数据类型和数组名的规定同一维数组一样，所不同的是多了一个中括号。例如：

```
short  [ ][ ] x;
float  y[ ][ ];
```

分别声明了二维短整型数组 x 和二维单精度浮点型数组 y，与一维数组一样，声明二维数组时也不能指定具体的长度，一般习惯将第一个中括号称为“行维”，第二个中括号称为“列维”，相应地，访问二维数组的元素时，需要同时提供行下标和列下标。

创建二维数组同样可以采用两种方式：初始化方式和 new 操作符方式。例如：

```
short [ ][ ] x = {{1,2,3},{4,5,6},{7,8,9}};
float  y[ ][ ]={{0.1,0.2},{0.3,0.4,0.5},{0.6,0.7,0.8,0.9}};
```

上述语句为采用初始化方式创建的两个二维数组。其中，x 为 3 行 3 列的等长数组，而 y 为非等长数组，第 1 行有 2 列，第 2 行有 3 列，第 3 行则有 4 列。初学者应该注意，Java 语言支持非等长数组并不代表其他语言也支持，如 C、Pascal 等就不支持。

上述二维数组如果采用 new 操作方式来创建，则 x 数组可以用如下语句创建：

```
short [ ][ ] x = new short[3][3];
```

而 y 数组则相对复杂一些：

```
float  y[ ][ ]= new float[3][];
y[0] = new float[2];
y[1] = new float[3];
y[2] = new float[4];
```

由此可见，非等长数组由于各列元素的个数不同，因此只能采取各列分别单独进行创建的方式，显然这种写法要稍微繁琐一些。

创建了二维数组后，就可以对数组元素进行访问了，上面说过，访问二维数组元素需要同时提供行下标和列下标，例如：

```
x[0][0] = 1;    x[0][1] = 2;    x[0][2] = 3;
x[1][0] = 4;    x[1][1] = 5;    x[1][2] = 6;
x[2][0] = 7;    x[2][1] = 8;    x[2][2] = 9;
```

上面 9 条语句分别对每一个数组元素进行了赋值，赋值后的状态与前面采用初始化方式创建的 x 数组相同。同理，对于非等长数组 y 的各元素赋值如下：

```
y[0][0] = 0.1;    y[0][1] = 0.2;
y[1][0] = 0.3;    y[1][1] = 0.4;    y[1][2] =0.5;
y[2][0] = 0.6;    y[2][1] = 0.7;    y[2][2] = 0.8;    y[2][3] = 0.9;
```

当要创建的[illegible]

```
char  str[][] = {{'T',
                 {'L', 'o', 'v', 'e'},
                 {'C', 'h', 'i', 'n', 'a'}};
int   z[][] = new [10][10];
for(int i=0;i<z.length;i++)   //通过外循环遍历数组每一行
  for(int j=0;j<z[i].length;j++)   //通过内循环遍历数组每一列
        z[i][j] = i*10+j;          //通过行下标和列下标访问数组元素
```

需要特别注意的是：z.length 总值代表二维数组 z 的行数，即行维的长度，而 z[i].length 的值则代表二维数组的第 i 行的元素个数，即列长度。因此，这两重嵌套循环结构遍历访问二维数组的程序对于非等长数组也是适用的。

对于二维字符数组 str 来说，str.length 的值应该为 3，而 str[0].length、str[1].length 和 str[2].length 的值分别为 1、4 和 5，str[1][1]的值则为字符'o'。

### 4.4.3　数组的应用举例

数组适合用来存储和处理相同类型的一批数据，下面给出几个关于数组的应用例子。

【例 4-16】某同学参加了高数、英语、Java 语言、线性代数和物理 5 门课程的考试，假定成绩分别为 70、86、77、90 和 82，请用数组来存放这些成绩，并计算 5 门课程的最高分和平均分。

```
public class Score
{
   public static void main(String[] args)
   {
      int x[]={70,86,77,90,82};
      int max=0; //临时变量
      int sum=0; //总分
      for(int i=0;i<x.length;i++)
      {
        if(x[i]>max)
           max=x[i];
```

```
            max=a[i];
        }
        System.out.println("最高分："+max);
        System.out.println("平均分："+sum*1.0/a.length);  //结果为小数
    }
}
```

程序的运行结果如下：

```
最高分：90
平均分：81.0
```

【例 4-17】某班同学参加了高数、英语、Java 语言、线性代数和物理 5 门课程的考试，假定成绩已公布，请编写一程序，通过键盘录入全部成绩，并计算输出每位同学的课程最高分、最低分和平均分，以及每一门课程的班级最高分、最低分和平均分。

```
import java.io.*;
public class Scores
{
    public static void main(String[] args)throws IOException
    {
        int max=0;    //最高分
        int min=100; //最低分
        int sum=0;    //总分
        System.out.print("请输入学生数：");
        InputStreamReader reader=new InputStreamReader(System.in);
        BufferedReader input=new BufferedReader(reader);
        String temp=input.readLine();
        //输入学生人数 n
        int n = Integer.parseInt(temp);
        int x[][]=new int[n][5];
        //录入成绩
        for(int i=0;i<n;i++)
        {
          for (int j=0;j<5 ;j++ )
          {
                System.out.print((i+1)+"号同学"+(j+1)+"号课程分数");
                temp=input.readLine();
                x[i][j] = Integer.parseInt(temp);
          }
        }
        //计算并输出每一位同学的课程最高分、最低分和平均分
        for(int i=0;i<n;i++)
        {
          for(int j=0;j<5;j++)
          {
```

```
                max=x[i][j];
            if (x[i][j]<min)
                min=x[i][j];
            sum+=x[i][j];
        }
        System.out.println((i+1)+"号同学最高分："+max);
        System.out.println((i+1)+"号同学最低分："+min);
        System.out.println((i+1)+"号同学平均分："+sum/5.0);
        max=0;
        min=100;
        sum=0;
        }
        //计算并输出每一门课程的班级最高分、最低分和平均分
          for(int i=0;i<5;i++)
            {
            for (int j=0;j<n ;j++ )
            {
                if (x[j][i]>max)
                    max=x[j][i];
                if (x[j][i]<min)
                    min=x[j][i];
                sum+=x[j][i];
            }
             System.out.println((i+1)+"号课程的班级最高分："+max);
             System.out.println((i+1)+"号课程的班级最低分："+min);
             System.out.println((i+1)+"号课程的班级平均分："+sum*1.0/n);
             max=0;
             min=100;
             sum=0;
        }
    }
}
```

某次的程序运行结果如下：

```
请输入学生数：2          (为简单起见，这里假定只有 2 位同学)
1 号同学 1 号课程分数 70
1 号同学 2 号课程分数 50
1 号同学 3 号课程分数 90
1 号同学 4 号课程分数 88
1 号同学 5 号课程分数 67
2 号同学 1 号课程分数 92
2 号同学 2 号课程分数 76
2 号同学 3 号课程分数 81
2 号同学 4 号课程分数 63
2 号同学 5 号课程分数 87
1 号同学最高分：90
1 号同学最低分：50
```

```
1 号同学平均分：73.0
2 号同学最高分：92
2 号同学最低分：63
2 号同学平均分：79.8
1 号课程的班级最高分：92
1 号课程的班级最低分：70
1 号课程的班级平均分：81.0
2 号课程的班级最高分：76
2 号课程的班级最低分：50
2 号课程的班级平均分：63.0
3 号课程的班级最高分：90
3 号课程的班级最低分：81
3 号课程的班级平均分：85.5
4 号课程的班级最高分：88
4 号课程的班级最低分：63
4 号课程的班级平均分：75.5
5 号课程的班级最高分：87
5 号课程班级最低分：67
5 号课程班级平均分：77.0
```

【例 4-18】试用冒泡法对 10，50，20，30，60 和 40 数列进行降序排列。

```
public class BubbleSort
{
    public static void main(String[] args)
    {
        int x[]={10,50,20,30,60,40};
        int temp; //临时变量
        for(int i=1;i<x.length;i++)    //比较趟次
        for (int j=0;j<x.length-i;j++)    //在某趟中逐对比较
        {
          if(x[j]<x[j+1])
          {   //交换位置
              temp=x[j];
               x[j]=x[j+1];
               x[j+1]=temp;
          }
        }
        for(int i=0;i<x.length;i++)
          System.out.print(x[i]+" ");     //遍历输出排序的数组元素
    }
}
```

程序最后的输出结果如下：

```
60 50 40 30 20 10
```

冒泡排序法的基本思路如下：对一个具有 n 个元素的数列，首先通过比较第 1 和第 2

个元素，若为降序，则不动，若为升序，则将两数做一个对调，然后再比较第 2 和第 3 个元素，依次类推，当比较 n-1 次以后，则最小的数就排在了最后的位置上；第二趟对前 n-1 个数做同样操作，将次小数排至倒数第 2 的位置上；依次类推，经过 n-1 趟比较后，整个数列就从无序变为有序了。由于在每一趟比较过程中，都会将其中最小的数推至最后面去，就像水底泡泡上升一样，故取名为冒泡排序。

上述程序总共进行了 5 趟排序，排序过程如下所示。

第 1 趟：50,20,30,60,40,10　(10 冒出来了)
第 2 趟：50,30,60,40,20　(20 也冒出来了)
第 3 趟：50,60,40,30　(30 冒出来)
第 4 趟：60,50,40　(40 冒出来)
第 5 趟：60,50　(50 冒出来，此时只剩最后一个数 60，因此排序完毕)

【例 4-19】矩阵相乘运算。

```
public class MatrixMultiply{
    public static void main(String args[]){
        int i,j,k;
        //创建二维数组 a
        int a[][]=new int [2][3];
        //创建并初始化二维数组 b
        int b[][]={{1,2,3,4},{5,6,-7,-8},{9,10,-11,-12}};
        //创建二维数组 c
        int c[][]=new int[2][4];
        for (i=0;i<2;i++)
            for (j=0; j<3 ;j++)
                //遍历 a 数组并赋值
                a[i][j]=(i+2)*(j+3);
        for (i=0;i<2;i++){
            for (j=0;j<4;j++){
                c[i][j]=0;
                for(k=0;k<3;k++)
                    c[i][j]+=a[i][k]*b[k][j];
            }
        }
        System.out.println("********矩阵 C********");
        //输出矩阵 C
        for(i=0;i<2;i++){
            for (j=0;j<4;j++)
                System.out.print(c[i][j]+" ");
            System.out.println();
        }
    }
}
```



程序运行结果如下：

```
********矩阵 C********
136 160 -148 -160
204 240 -222 -240
```

上述程序首先利用数组存放 2 行 3 列矩阵 a 和 3 行 4 列矩阵 b，通过循环结构实现矩阵的遍历赋值和相乘运算，并将矩阵相乘的结果存放至 2 行 4 列的矩阵 c 中，最后，将结果矩阵 c 打印输出。