第 3 章

数　组

本章学习目标

数组是由类型相同的元素顺序组成的一个集合，每个数组都有一个唯一的名称，即数组名。通过数组名和下标可以方便地访问数组中的每个成员。本章主要讲述 Java 中一维和二维数组的创建和使用方法。通过本章的学习，应该重点掌握以下主要内容：

（1）一维数组的定义和产生方法。

（2）一维数组的初始化方法。

（3）一维数组的引用方法。

（4）二维数组的创建和引用方法。

3.1　数组使用初探

Java 程序中通常使用变量来存放各种类型的数据，如将 10 个数分别存放在 10 个变量中，但这会非常麻烦，因为需要的变量名实在太多了。这时就可以考虑用一个数组变量来存放它们，并通过一个下标来访问存入数组中的每个成员。

【例 3.1】 求 10 个学生某门课程的平均成绩。

分析： 如果用变量 s1，s2，…，s10 来保存 10 个学生的成绩，显然不合适（变量太多且随学生人数递增），而如果使用数组 s 来保存这些成绩，则问题可以处理如下：

```
package code0301;
public class AvgGrade {
    public static void main(String args[]) {
        int total = 0;
        int s[] = { 75, 69, 80, 85, 93, 97, 79, 77, 68, 90 };     //定义数组变量 s 并赋值
        double avg = 0;
        for (int i = 0; i < 10; i++) {
            total = total + s[i];                                  //访问数组成员 s[i]
        }
        avg = total / 10.0;                                        //求平均分 avg
        System.out.println("The average score is :" + avg);
    }
}
```

程序运行结果：

```
The average score is :81.3
```

该问题的求解涉及两个关键点：

(1) 定义数组变量 s,将所有值{75，69，80，85，93，97，79，77，68，90}存入其中。

(2) 通过数组变量 s 加下标 i 的形式访问数组中每个成员：total = total + s[i]。

3.2 一维数组

与简单变量一样,数组必须先定义后使用。定义数组时,除了要给定数组的名称、数组成员类型,还要为其分配内存空间并进行初始化。

3.2.1 定义数组

一维数组的定义格式如下：

```
数据类型 数组名[ ];
```

其中,数据类型是指数组成员的类型,可以为基本数据类型,也可以为引用数据类型。[ ]为数组标记。在定义数组时,不允许在[ ]内指定数组元素的个数。例如：

```
int a[ ],  boolean temp[ ],  String s[ ];
```

表示数组 a 的数据类型为 int,temp 的数据类型为 boolean,s 的数据类型为类 String。

提示：定义数组时,另外一种可选形式为直接将方括号放在变量类型的后边。例如：

```
int[ ] a,  boolean[ ] temp,  String[ ] s
```

3.2.2 生成数组

数组定义只是建立了一种数组的引用,还必须使用关键字 new 为其分配内存空间,否则它是无法被访问的。基本形式如下：

```
数组变量名= new 数据类型[数组长度]
```

例如：

```
char s[];  s=new char[5];
```

的作用是为数组 s 分配 5×2B 大小的空间。也可以在定义数组的同时为之分配内存空间,例如：

```
int temp[]=new int[10];
```

3.2.3 初始化数组

当使用关键字 new 生成数组时,数组中的每个成员都会被自动初始化,初始值依据数组的类型而定。例如：

```
数值型的初始值:0;              字符型的初始值:'\0';
布尔型的初始值:false;          类对象的初始值:null;
```

也可以通过主动赋值的方式对数组进行初始化。例如

```
int s[ ]=new int[3];          s[0]=1; s[1]=2; s[2]=3
```

还可以将数组的定义、内存空间分配、初始化工作放在一条语句中,格式如下:

```
数据类型 数组名[ ]={值 1,…,值 n}
```

这种数组初始化形式的特点是无须给出数组的长度,系统会根据初始值的数量自动确定数组的长度,并依次将每个数组成员放在数组中。例如:

```
int s[ ]={1,2,3};
```

使得数组长度为 3,且 s[0]=1;s[1]=2;s[2]=3。

数组不能整体赋值,例如,下列语句是错误的:

```
char s[]=new char[5];          s[]={'a','b','c','d','e'};     //错误
```

提示:必须给出数组长度,而且数组一旦创建,就不允许再增加它的空间。

3.2.4 访问数组

数组成员的访问是通过数组名和下标来实现的,Java 中的下标是从 0 开始的,依次递增 1,如果数组 temp 的长度为 10,则数组下标为 0~9。这些成员分别表示为

```
temp[0],temp[1],…,temp[9]
```

对数组中任意成员的引用形式为:数组名[下标]。

例如,temp[4]表示引用数组 temp 的第 5 个元素。数组元素的用法和平常使用的普通变量没有什么差异,也可以被赋值或参与其类型允许的各种运算等。例如:

```
int i=temp[0]%2;
temp[1]=temp[0] * 5;
System.out.println("value="+temp[2]);
```

使用数组时要注意数组是否越界,每个数组一个属性 length,可以通过数组名.length 来获取数组的长度,即数组元素的个数。

3.2.5 实用案例 3.1:求一维数组的最大值及位置

为给一维数组进行随机赋值,本例采用 Math.random()方法产生随机数。

【例 3.2】 求一维数组的最大值及位置。

```
package Code0302;
public class MaxNum {
    public static void main(String[] args) {
        final int ARRAY_SIZE = 10;
        int a[] = new int[ARRAY_SIZE];
        int max = 0;
        int index = 0;                              //存储最大元素的位置
        for (int i = 0; i < a.length; i++) {        //本例中 a.length=10
            a[i] = (int) (Math.random() * 10);      //产生随机数,并对数组成员赋值
            System.out.print("  " + a[i]);
        }
```

```
            System.out.println();
            max = a[0];
            for (int j = 1; j < ARRAY_SIZE; j++) {
                if (a[j] > max)                          //判断当前位置的数是否大于最大值 max
                {
                    max = a[j];
                    index = j;                           //替换当前的最大值,记住对应位置
                }
            }
            System.out.println("A[" + index + "] has maximum value " + a[index]);

        }
    }
```

程序可能的输出结果：

```
2  0  0  1  9  5  1  4  2  7
A[4] has maximum value 9
```

分析：第一个 for 循环依次将随机数读入下标为 i 的数组中，完成初始化操作；第二个 for 循环依次取出每个数组元素 a[j]的值，并判断其是否为当前最大值。

提示：如果例 3.2 中第二个 for 循环改为 for(int j=1;j<=ARRAY_SIZE;j++)，则出现数组下标越界的情况，因为 a[10]并不存在。

3.3 二维数组

到目前为止使用的都是一维数组，但有时应用中需要使用多维数组才能充分地存储数据。例如，(x,y)坐标系统中，如果需要存储坐标 x 和坐标 y 来标识一个点，则需要使用二维数组，其中一维用于存储坐标 x，另一维用于存储坐标 y。

将数组的下标数称为数组的维数，维数大于 1 的数组称为多维数组。因为多维数组的操作比较相似，本节重点介绍二维数组。

3.3.1 定义二维数组

二维数组的定义与一维数组相似：

```
数据类型  数组名[ ][ ];
```

两个方括号表示这是一个二维数组。当然，它也支持另一种形式：

```
数据类型[ ][ ] 数组名;
```

与一维数组相似，二维数组的创建可以先定义数组，再分配内存空间，也可以同时进行这两项工作。例如：

```
int temp[][];  temp=new int[2][3];
int temp[][]=new int[2][3];
```

这样即创建了一张二维结构的数组，其中行数为 2，列数为 3。也可将其看作一个长度为 2 的一维数组，每个数组成员又是一个长度为 3 的一维数组。这些数组成员按行排列为

```
temp[0][0], temp[0][1], temp[0][2]
temp[1][0], temp[1][1], temp[1][2]
```

同样,可以在定义数组的同时对其进行初始化。例如:

```
int temp[][]={{1,2},{3,4},{5,6}}
```

系统自动根据初始值的情况,为数组指定大小长度,并依次将值填入相应的单元中,例如初始化后 temp[0][1]=2,temp[2][0]=5。

注意:等式右边大括号内嵌套的大括号不能省略,它代表数组的一行及其组成。

3.3.2 二维数组元素的引用

引用二维数组元素的方式如下:

```
数组名[下标 1][下标 2]
```

下标仍然是从 0 开始逐渐递增的,同样在引用时要注意数组的越界问题。二维数组也有 length 属性,可以求每一维数组的长度。例如:

```
int temp[][]=new int[3][5];
System.out.println(temp.length);          //求二维数组的长度实际是求它的行数 3
System.out.println(temp[0].length);       //每个数组成员又是一个一维数组,其长度为 5
```

【例 3.3】 求二维数组中各元素的和。

```
package code0303;
public class SumAll{
    public static void main(String args[]){
    int total=0;
    int arr[][]=new int[3][4];
    for(int i=0;i<arr.length;i++){                //初始化并显示二维数组
            for(int j=0;j<arr[i].length;j++){
                arr[i][j]=i+j;
                System.out.print(" " + arr[i][j]);
            }
            System.out.println();
        }
        for(int i=0;i<arr.length;i++)             //求和
            for(int j=0;j<arr[i].length;j++){
                total = total +arr[i][j];
            }
        System.out.println(" The Sum is:"+total);
    }
}
```

程序运行结果:

```
0 1 2 3
1 2 3 4
2 3 4 5
The Sum is:30
```

3.3.3 实用案例 3.2:求两个矩阵的乘积

根据矩阵的乘法规则,两个矩阵相乘将产生一个新的矩阵,如 a[4,3]×b[3,2]将产生一

个 r[4,2]的新矩阵,其中的某个元素 r[i][j]=a[i][0]×b[0][j]+a[i][1]×b[1][j]+a[i][2]×b[2][j]。例如:

$$x=\begin{pmatrix}1 & 6\\ 3 & 8\end{pmatrix}\times\begin{pmatrix}2 & 2\\ 9 & 7\end{pmatrix}$$

其计算过程为

$$1\times2+6\times9=56$$
$$1\times2+6\times7=44$$
$$3\times2+8\times9=78$$
$$3\times2+8\times7=62$$

因此结果为$\begin{pmatrix}56 & 44\\ 78 & 62\end{pmatrix}$。

【例 3.4】 求两个矩阵的乘积。

```
public class MatrixMultiply {
    public void multiply(int[][] a,int[][] b) {
        int [][] r = new int[4][2];          //r 用于存放运算结果
        int tmp =0;
        for (int k =0; k < r[0].length; k++) {
            //双重循环,遍历 a 矩阵
            for (int i = 0; i< a.length; i++ ) {
                tmp = 0;
                for (int j = 0; j < a[0].length; j++) {
                    tmp += a[i][j] * b[j][k] ;
                }
                r[i][k]=tmp;
            }
        }
         for (int i = 0; i < r.length; i++) {
            for (int j = 0; j < r[0].length; j++) {
                System.out.print(r[i][j] + "\t");
            }
            System.out.println();

        }
    }
    public static void main(String[] args) {
        int[][] a = new int[][] {
                    { 1, 2, 3 },
                    { 4, 5, 6 },
                    { 7, 8, 9 },
                    { 11, 12, 13} };
        int[][] b = new int[][]{
                    {1,2},
                    {3,4},
                    {5,6} } ;
        MatrixMultiply ma=new MatrixMultiply();
        ma.multiply(a,b);
    }
}
```

3.4 Arrays 类

为了更方便地操作数组,Java 在包 java.util 定义一个叫 Arrays 的类,该类包含了如下几个用 static 修饰的静态方法。

(1) static type[] copyOf(type[] original,int length):将 original 数组复制为一个新数组,其中 length 为新数组的长度。

(2) static int binarySearch(type[] a,type key):使用二分搜索法在类型为 type 的数组中搜索类型为 type 的指定值 key。

(3) static boolean equals(type[] a,type[] b):比较两个类型为 type 的数组是否相等。

(4) static void fill(type[] a, type val):用一个指定的值 val 填充类型为 type 的数组 a。

(5) static void fill(type[] a, int fromIndex, int toIndex, type val):与前一个方法类似,但填充时仅仅针对下标为 fromIndex 到 toIndex-1 的数组元素赋值为 val。

(6) static void sort(type[] a):对类型为 type 的数组排序。

上述每个方法的具体描述,请参考 JDK 文档。

【例 3.5】 Arrays 类的基本使用。

```
package code0304;
import java.util.Arrays;
public class ArraysDemo {
    public static void main(String[] args) {
        Integer array[] = new Integer[9];
        for (int i = 1; i < 10; i++)
            array[i - 1] = (int) (Math.random() * 100);
        //显示,排序数组
        System.out.print("原内容: ");
        display(array);
        Arrays.sort(array);
        System.out.print("排序后: ");
        display(array);
        //将值-1 分配给数组 array 中下标从 0~3-1 位置上的元素
        Arrays.fill(array, 0, 3, -1);
        System.out.print("执行 fill()后: ");
        display(array);
        //搜索 39
        System.out.print("值 39 的位置 ");
        int index = Arrays.binarySearch(array, 39);
        System.out.println(index);
    }
    static void display(Integer array[]) {
        for (int i = 0; i < array.length; i++)
            System.out.print(array[i] + " ");
        System.out.println("");
    }
}
```

程序运行结果:

```
原内容: 90 48 81 14 3 35 95 4 97
排序后: 3 4 14 35 48 81 90 95 97
```

```
执行 fill()后: -1 -1 -1 35 48 81 90 95 97
值 39 的位置 -5
```

提示：Java 8 对 Arrays 类的功能进行了增强。如增加了一些新的方法，可以利用多 CPU 的并行处理能力提高数组排序、赋值等操作性能；parallelSort(type[] a)方法与之前介绍的 sort()方法相似，但可以在多 CPU 上并行实现。

实用案例 3.3　对数组按中文名称排序

【例 3.6】　对数组按中文名称排序。

```
package code0304;
import java.text.Collator;
import java.util.Arrays;
import java.util.Comparator;
public class SortByChinese {
    public static void main(String[] args) {
        String[] arrStrings = {"计算机","长江", "通信","数学"};
        Arrays.sort(arrStrings);
        for (int i = 0; i < arrStrings.length; i++)
            System.out.println(arrStrings[i]);
        System.out.println("------------------------------");
        //Collator 类用来执行区分语言环境的字符串比较,这里选择用 CHINA
        Comparator comparator = Collator.getInstance(java.util.Locale.CHINA);
        //根据指定比较器产生的顺序对指定对象数组进行排序
        Arrays.sort(arrStrings, comparator);
        for (int i = 0; i < arrStrings.length; i++)
            System.out.println(arrStrings[i]);
    }
}
```

程序运行结果：

```
数学
计算机
通信
长江
------------------------------
长江
计算机
数学
通信
```

Arrays 类中的 sort()方法缺省，那么按照数组中数值的大小或字母顺序进行排序，但这种处理方式对中文是无效的，如上例运行结果。为此，我们使用了类 Arrays 中另一种形式的 sort()方法：sort(T[] a, Comparator<? super T> c)，它可以根据指定比较器 Comparator 产生的顺序对对象数组进行排序。为获取 Comparator 对象，可以通过方法 Collator.getInstance()实现，其中参数 java.util.Locale.CHINA 表示按中文语言进行排序。

3.5 数组实训任务

【任务描述】

编写一个模拟的 Java 发牌程序,要求将 2 副牌(即 108 张牌)发给 4 个人,并留 8 张底牌,最后输出底牌和每个人手中的牌。

【任务分析】

(1) 首先确定扑克牌在计算机中的表达方式,因为计算机中无法表达扑克牌中的花色,因此最好为每张牌设定一个编号,拟定的编号规则如下:

红桃按照从小到大依次为 1～13。

方块按照从小到大依次为 14～26。

黑桃按照从小到大依次为 27～39。

梅花按照从小到大依次为 40～52。

小王为 53,大王为 54。

(2) 由于可发的牌和每个玩家手中的牌都需要记录,且其由多个有序数据组成,因此需要设计两个数组,以存放 108 张牌和每个玩家手中的牌。

(3) 整个程序由以下 4 部分构成:

① 按照以上编号规则初始化一个包含 108 个数字的数组。

② 每次随机从该数组中抽取一个数字,分配给保存玩家数据的数组。

③ 循环输出每个玩家手中的牌。

④ 最后输出底牌。

【任务解决】

【例 3.7】 模拟发牌。

```
package code0305;
import java.util.*;
public class CardPlay{
    public static void main(String[] args){
        int[] total = new int[108];                    //存储 108 张牌的数组
        int[][] player = new int[4][25];               //存储 4 个玩家的牌
        int leftNum = 108;                             //当前剩余牌的数量
        int ranNumber;
        Random random = new Random();                  //生成 Random 对象,用以生成随机数
        for(int i = 0;i < total.length;i++){           //初始化一维数组
            total[i] = (i + 1) % 54;
            if(total[i] == 0){                         //处理大小王编号
                total[i] = 54;
            }
        }
        //循环发牌
        for(int i = 0;i < 25;i++){//为每个人发牌
            for(int j = 0;j < player.length;j++){   /生成随机下标
                ranNumber = random.nextInt(leftNum); //random.nextInt 方法生成随机数
                player[j][i] = total[ranNumber];    //发牌
                total[ranNumber] = total[leftNum - 1];     //删除已经发过的牌
                leftNum--;
```

```
            }
        }
        //循环输出玩家手中的牌
        for(int i = 0;i < player.length;i++){        //通过两层循环遍历二维数组
            System.out.print("玩家" + i+ "的牌:");
            for(int j = 0;j < player[i].length;j++){
                System.out.print(" " + player[i][j]);
            }
            System.out.println();
        }
        //底牌
        System.out.print("底牌:");
        for(int i = 0;i < 8;i++){
            System.out.print(" " + total[i]);
        }
        System.out.println();
    }
}
```

程序运行结果：

```
玩家 0 的牌: 52 15 12 53 32 24 48 36 24 11 54 27 14 22 3 8 10 40 43 46 40 45 37 41 22
玩家 1 的牌: 49 25 5 47 20 35 45 51 29 17 10 26 20 51 3 1 29 43 31 9 33 30 32 4 53
玩家 2 的牌: 42 7 13 38 34 50 12 6 39 49 33 23 14 31 7 19 39 48 16 18 35 5 41 9 11
玩家 3 的牌: 44 2 30 26 28 46 44 19 21 23 25 21 15 13 1 28 34 52 4 8 6 47 17 2 27
底牌: 50 42 18 37 16 38 36 54
```

习题与思考

1. 为了定义 3 个整型数组 a1、a2、a3，下面声明正确的语句是哪一个？

   A. int Array[] a1,a2; int a3[]={1,2,3,4,5};

   B. int [] a1,a2; int a3[]={1,2,3,4,5};

   C. int a1,a2[]; int a3={1,2,3,4,5};

   D. int [] a1,a2; int a3=(1,2,3,4,5);

2. 编程实现两个同行列数矩阵的加法运算。

3. 将给定数组 int a[]={78,23,56,34,12,45,67,89}按从小到大顺序进行排序并输出。

4. 给出下列程序段运行的结果：

```
public static void main(String args[])
{
    int array[]={1,2,3,4,5};
    printarray(array);
    could_modify(array);
    printarray(array);
}
```

```
static void could_modify(int a[]){
    for(int i=0;i<a.length;i++)
        a[i]*=i;
}
static void printarray(int a[])
{
    for(int i=0;i<a.length();i++)
    System.out.println(a[i]);
}
```

5. 数组创建后,可否通过设置变量 length 的大小来增减数组的大小?