第5章

数组

本章主要学习 Java 语言的数组及其应用，数组的创建与使用，Arrays 类的常用方法，二维数组及其应用。

本章主要内容：

- 数组的定义
- 访问数组元素
- 数组元素的复制
- 可变参数方法
- Arrays 类
- 二维数组的使用

5.1 创建和使用数组

视频讲解

数组是几乎所有编程语言都提供的一种数据存储结构。所谓**数组**(array)是指名称相同、下标不同的一组变量，它用来存储一组类型相同的数据。下面就来介绍如何声明、创建和使用数组。

5.1.1 数组定义

使用数组一般需要两个步骤：①声明数组，声明数组名称和元素的数据类型；②创建数组，为数组元素分配存储空间。

1. 声明数组

使用数组之前需要声明，声明数组就是告诉编译器数组名和数组元素类型。数组声明可以使用下面两种等价形式。

```
元素类型 []数组名;
元素类型 数组名[];
```

方括号可放在数组名前或后

这里，元素类型可以是基本数据类型(如 boolean 型或 char 类型)，也可以是引用数据类型(如 String 或 Account 类型等)。数组名是一个引用变量。方括号指明变量为数组变量，它既可以放在变量前面也可以放在变量后面。推荐放在变量前面，这样更直观。

例如,下面的代码声明了两个数组。

```
double [ ]marks;
Account [ ]accounts;
```

【注意】 数组声明不能指定数组元素的个数,这一点与 C/C++不同。

上面声明的数组,它们的元素类型分别为 double 型和 Account 类型。在 Java 语言中,数组是引用数据类型,也就是说数组是一个对象,数组名就是对象名(或引用名)。数组声明实际上是声明一个引用变量。如果数组元素为引用类型,则该数组称为**对象数组**,如上面的 accounts 就是对象数组。所有数组都继承了 Object 类,因此,可以调用 Object 类的所有方法。

【提示】 Java 语言的数组是一种引用数据类型,即数组是对象。数组继承 Object 类的所有方法。

2. 创建数组

数组声明仅声明一个数组对象引用,而创建数组是为数组的每个元素分配存储空间。创建数组使用 new 语句,一般格式为:

```
数组名 = new 元素类型[size];        ←┤ 这里 size 指定大小
```

该语句的功能是分配 size 个指定元素类型的存储空间,并通过数组名引用。下面的代码创建了 marks 数组和 accounts 数组。

```
marks = new double[5];              //数组包含 5 个 double 型元素
accounts = new Account[3];          //数组包含 3 个 Account 型元素
```

【注意】 Java 数组可以动态创建,也就是它的大小可以在运行时指定。

数组的声明与创建可以写在一个语句中,例如:

```
double [ ]marks = new double[5];       ←┤ 声明同时创建数组
Account [ ]accounts = new Account[3];
```

当用 new 运算符创建一个数组时,系统就为数组元素分配了存储空间,这时系统根据指定的长度创建若干存储空间并为数组的每个元素指定默认值。数值型数组元素默认值是 0,字符型元素的默认值是'\u0000',布尔型元素的默认值是 false。如果数组元素是引用类型,其默认值是 null。

上面两个语句分别分配了 5 个 double 型和 3 个 Account 类型的空间,并且每个元素使用默认值初始化。上面两个语句执行后效果如图 5-1 所示。数组 marks 的每个元素都被初始化为 0.0,而数组 accounts 的每个元素被初始化为 null。

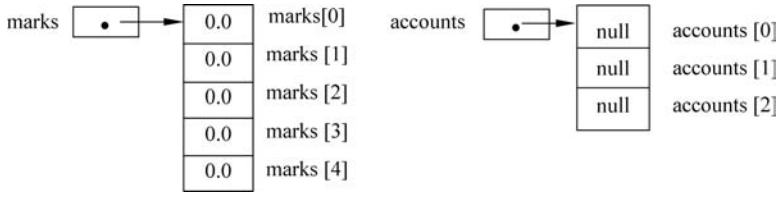

图 5-1 marks 数组和 accounts 数组示意

对于引用类型数组(对象数组),它的每个元素初值为 null,因此,还需要创建数组元素对象。

```
accounts[0] = new Account(103,"张三",3000.0);
accounts[1] = new Account(104,"王五",5000.0);
accounts[2] = new Account(105,"李四",8000.0);
```

上面语句执行后效果如图 5-2 所示。

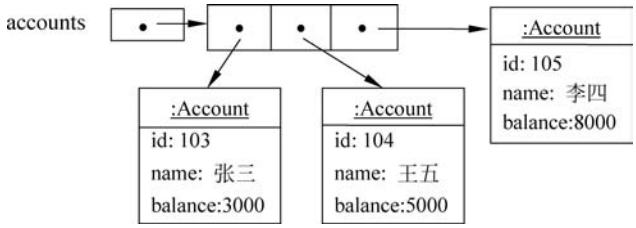

图 5-2 accounts 数组创建后的效果

5.1.2 访问数组元素

声明了一个数组,并使用 new 运算符为数组元素分配内存空间后,就可以使用数组中的每一个元素了。数组元素的使用方式是:

```
数组名[下标]
```

通过数组名和下标访问数组元素,下标从 0 开始,到数组的长度减 1。例如,上面定义的 accounts 数组包含 3 个元素,所以只能使用 accounts[0]、accounts[1]和 accounts[2]这三个元素。数组一经创建大小不能改变。

数组作为对象提供一个 length 成员变量,它的值是数组元素个数,访问该成员变量的方法为"数组名.length"。

【案例 5-1】 数组的声明、创建以及元素和 length 成员的使用。

【程序 5-1】 ArrayDemo.java

```
package com.boda.xy;
public class ArrayDemo {
  public static void main(String[] args) {
    var marks = new double[5];  ←┤ 用 var 声明数组不加方括号
    marks[0] = 79;
    marks[1] = 84.5;
    marks[2] = 63;
    marks[3] = 90;
    marks[4] = 98;
    System.out.println(marks[2]);
    System.out.println(marks.length);
    //输出每个元素值
    for (var i = 0; i < marks.length; i++) {
      System.out.print(marks[i] + "  " );
    }
  }
}
```

运行结果如图 5-3 所示。

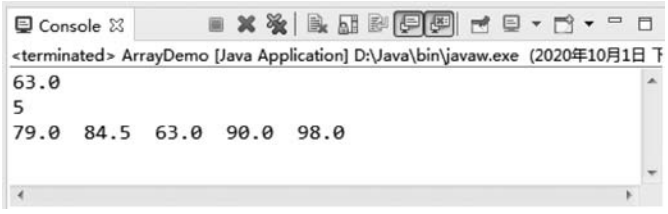

图 5-3 案例运行结果

程序中使用 for 循环迭代数据的每个元素。使用这种方法不但可以从前到后访问元素，也可以逆序访问元素，还可以访问部分元素以及对访问的元素进行修改(如加分)，因此这种方式对数组元素的操作比较灵活。

脚下留神

为了保证安全性，Java 运行时系统要对数组元素的范围进行越界检查，若数组元素下标超出范围，将抛出 ArrayIndexOutOfBoundsException 运行时异常。例如，下面的代码会抛出异常。

```
System.out.println(marks[5]);
```

5.1.3 数组初始化器

声明数组同时可以使用初始化器对数组元素初始化，它是在一对大括号中给出数组的每个元素值。这种方式适合数组元素较少的情况，这种初始化也称为静态初始化。

```
double[] marks = {79, 84.5, 63, 90, 98};
Account[] accounts = {new Account(103, "张三", 3000),
         new Account(104, "王五", 5000), new Account(105,"李四",8000)};
```

上面两句还可以写成如下的形式。

```
var marks = new double[]{79, 84.5, 63,90, 98};
var accounts = new Account[]{new Account(103, "张三", 3000),
                        new Account(104, "王五", 5000),
                        new Account(105,"李四",8000)};
```

用这种方法创建数组不能指定大小，系统根据元素个数确定数组大小。另外，可以在最后一个元素后面加一个逗号，以方便扩充。

脚下留神

对于数组局部变量的类型推断，如果声明数组使用 new 运算符创建数组，则可以使用 var 声明数组，并且声明的数组变量不带方括号，例如：

```
var marks = new double[5];
```

但是如果使用数组初始化器创建数组，则不能使用 var 声明数组，例如，下面的代码会发

生编译错误。

```
var marks = {79, 84.5, 63,90, 98};
```

但使用下面的格式是合法的，因为这里指定了数组元素的类型。

```
var marks = new double[]{79, 84.5, 63,90, 98};
```

5.1.4 增强的 for 循环

如果程序只需顺序访问数组中每个元素，可以使用增强的 for 循环，它是 Java 5 新增的功能。增强的 for 循环可以用来迭代数组和对象集合的每个元素。它的一般格式为：

```
for(type identifier: expression) {
    //循环体
}
```

该循环的含义为：对 expression(数组或集合)中的每个 type 类型的元素 identifier，执行一次循环体中的语句。这里，type 为数组或集合中的元素类型。expression 必须是一个数组或集合对象。

下面使用增强的 for 循环实现求数组 marks 中各元素的和，代码如下。

```
double sum = 0;
for(var score : marks){
    sum = sum + score;
}
System.out.println("总成绩 = " + sum);
```

【提示】 使用增强的 for 循环只能按顺序访问数组元素，并且只能使用元素而不能对元素进行修改。

5.2 数组的应用

视频讲解

5.2.1 数组元素的复制

经常需要将一个数组的元素复制到另一个数组中。首先应该想到将数组元素一个一个地复制到目标数组中。设有一个数组 source，其中有 4 个元素，现在定义一个数组 target，与原来数组类型相同，元素个数相同。使用下面的方法将源数组的每个元素复制到目标数组中。

```
int[] source = {10,30,20,40};                    //源数组
int[] target = new int[source.length];           //目标数组
for(var i = 0; i < source.length; i++)
   target[i] = source[i] ;
```

除上述方法外，还可以使用 System 类的 arraycopy()方法，格式如下。

```
public static void arraycopy(Object src, int srcPos,
                             Object dest, int destPos,int length)
```

src 为源数组，srcPos 为源数组的起始下标，dest 为目标数组，destPos 为目标数组下标，length 为复制的数组元素个数。下面的代码实现将 source 中的每个元素复制到数组 target 中。

```
int[] source = {10,30,20,40};
int[] target = new int[source.length];
System.arraycopy(source, 0, target, 0, 4);
```

使用 arraycopy()方法可以将源数组的一部分元素复制到目标数组中。但要注意，如果目标数组不足以容纳源数组元素，会抛出异常。

【案例 5-2】 数组元素的复制与异常。

【程序 5-2】 ArrayCopyDemo.java

```
package com.boda.xy;
public class ArrayCopyDemo{
  public static void main(String[] args){
    int[] a = {1,2,3,4};
    int[] b = {8,7,6,5,4,3,2,1};
    int[] c = {10,20};
    try{
      System.arraycopy(a, 0, b, 0, a.length);
      //下面语句发生异常,目标数组 c 容纳不下源数组 a 的元素
      System.arraycopy(a, 0, c, 0, a.length);
    }catch(ArrayIndexOutOfBoundsException e){
      System.out.println(e);
    }
    for(var elem: b){
      System.out.print(elem + "  ");
    }
    System.out.println();
    for(var elem: c){
      System.out.print(elem + "  ");
    }
  }
}
```

运行结果如图 5-4 所示。

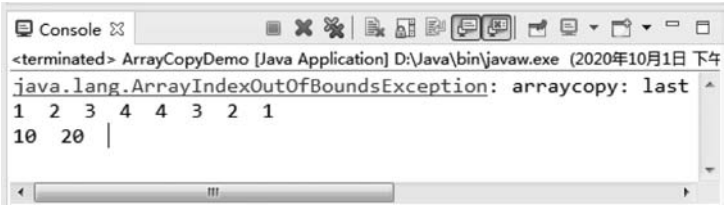

图 5-4 案例运行结果

注意，不能使用下列方法试图将数组 source 中的每个元素复制到 target 数组中。

```
int[] source = {10,30,20,40};
int[] target = source;     //这是引用赋值
```

上述两条语句实现对象的引用赋值，两个数组引用指向同一个数组对象，如图 5-5 所示。

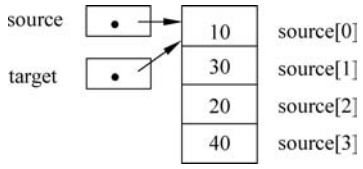

图 5-5 将 source 赋值给 target 的效果

5.2.2 数组参数与返回值

数组对象可以作为参数传递给方法，例如，下面的代码定义了一个求数组元素和的方法。

```
public static double sumArray(double array[]){
  var sum = 0.0;
  for(var i = 0; i < array.length; i++){
    sum = sum + array[i];
  }
  return sum;
}
```

【注意】 由于数组是对象，因此将其传递给方法时按引用传递。当方法返回时，数组对象不变。但要注意，如果在方法体中修改了数组元素的值，则该修改反映到返回的数组对象。

一个方法也可以返回一个数组对象，例如，下面的方法返回参数数组的元素反转后的一个数组。

```
public static int[] reverse(int[] list){
  var result = new int[list.length];          //创建一与参数数组大小相同的数组
  for(var i = 0, j = result.length - 1; i < list.length; i++, j--){
    result[j] = list[i];                      //实现元素反转
  }
  return result;                              //返回数组
}
```

有了上述方法，可以使用如下语句实现数组反转。

```
int[] list = {6, 7, 8, 9, 10};
int[] list2 = reverse(list);
```

5.2.3 可变参数方法

Java 语言允许定义方法（包括构造方法）带可变数量的参数，这种方法称为可变参数（variable argument）方法。具体做法是在方法参数列表的最后一个参数的类型名之后、参数名之前使用省略号，例如：

```
public static double average(double ... values){
    //方法体
}
```

这里,参数 values 被声明为一个 double 型值的序列。其中,参数的类型可以是引用类型。对可变参数的方法,调用时可以为其传递任意数量的指定类型的实际参数。在方法体中,编译器将为可变参数创建一个数组,并将传递来的实际参数值作为数组元素的值,这就相当于为方法传递一个指定类型的数组。

【案例 5-3】 可变长度参数方法的应用。

【程序 5-3】 VarargsDemo.java

```
package com.boda.xy;
public class VarargsDemo{
   public static double average(double … values){
      var sum = 0.0;
      for(var value:values){
         sum = sum + value;          //求数组元素之和
      }
      var average = sum / values.length;
       return average;
   }
   public static void main(String[] args){
      System.out.println("平均值 = " + average(60,70,86));
   }
}
```

运行结果如图 5-6 所示。

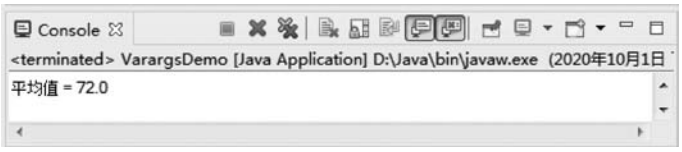

图 5-6 案例运行结果

程序定义了带可变参数的方法 average(),它的功能是返回传递给该方法的多个 double 型数的平均值。该程序调用了 average()方法并为其传递三个参数,输出结果为 72.0。

在可变参数的方法中还可以有一般的参数,但是可变参数必须是方法的最后一个参数,例如,下面定义的方法也是合法的。

```
public static double average(String name,double … values){
    //方法体
}
```

【注意】 在调用带可变参数的方法时,可变参数是可选的。如果没有为可变参数传递一个值,那么编译器将生成一个长度为 0 的数组。如果传递一个 null 值,将产生一个运行时 NullPointerException 异常。

视频讲解

5.3 案例研究:数组起泡排序

1. 问题描述

在程序设计中,排序是一种常见的操作。例如,一个数组可能存放某个学生的各科成绩,

现要求将成绩按从低到高的顺序输出，这就需要为数组排序。排序有多种方法，如起泡排序、选择排序、插入排序、快速排序等。本案例采用起泡排序法（bubble sort）对一个数组按升序排序。

2. 运行结果

该案例对包含下面 8 个数的数组升序（从小到大顺序）排序。

```
75, 53, 32, 12, 46, 199, 17,  54
```

运行结果如图 5-7 所示，这里输出了每趟排序的结果。

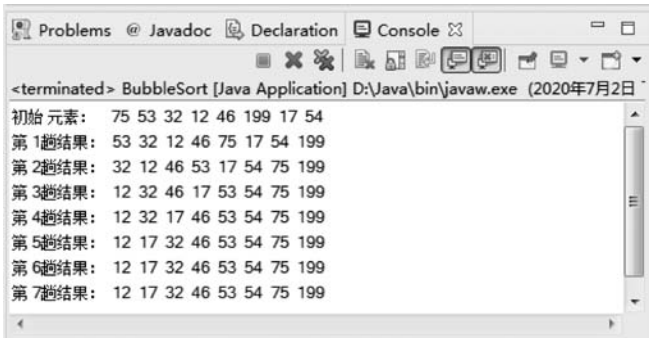

图 5-7　案例运行结果

3. 任务目标

（1）学会分析“数组起泡排序”案例的实现思路。

（2）根据设计思路独立完成“数组起泡排序”案例代码的编写、编译和运行。

（3）掌握数组的循环迭代、元素比较与交换的方法。

（4）了解如何使用 Arrays 类的 sort 方法对数组元素排序。

4. 设计思路

假设给定如下整型数组：

```
int[]a = {75, 53, 32, 12, 46, 199, 17, 54};
```

根据问题要求，采用起泡排序法对该数组升序排序，设计思路如下。

从数组的第一个元素开始，与它后面的元素比较，如果逆序（即 a[j] > a[j+1]），则交换两个元素的位置，如果正序，则不交换。经过第一趟排序后，结果如下。

```
53, 32, 12, 46, 75, 17, 54, 199
```

可以看到，经过第一趟排序，找到了数组中的最大值，而经过比较后，小的元素交换到前面，就像水泡向上冒。第二趟排序时，只需比较到倒数第二个元素为止，即可找到次大的元素，因此这里使用二重循环结构。

5. 代码实现

该案例实现代码如程序 5-4 所示。

【程序 5-4】 BubbleSort.java

```
package com.boda.xy;
public class BubbleSort {
  public static void main(String[] args) {
     int[]a = {75, 53, 32, 12, 46, 199, 17, 54};
     System.out.print("初始 元素:   ");
     for(var n:a) {
        System.out.print("  " + n);
     }
     System.out.println();
     for(var i = 0; i < a.length - 1; i++) {
        for(var j = 0; j < a.length - i - 1; j ++) {
           if(a[j] > a[j + 1]) {
              int t = a[j];
              a[j] = a[j + 1];
              a[j + 1] = t;
           }
        }
        System.out.print("第 " + (i + 1) + "趟结果: ");
        for(var n:a) {
           System.out.print(" " + n);
        }
        System.out.println();
     }
  }
}
```

若逆序,则交换元素

关于这个起泡排序算法,这里直接写在 main()方法中,实际上可以定义一个方法实现排序,然后在 main()方法中调用它,如下所示。

```
public static void bubbleSort(int[] a){
   for(var i = 0; i < a.length; i++) {
       for(var j = 0; j < a.length - i - 1; j ++) {
          if(a[j] > a[j + 1]) {
              int t = a[j];
              a [j] = a[j + 1];
              a[j + 1] = t;
          }
       }
   }
}
```

之后在 main()方法中调用 bubbleSort()方法,代码如下。

```
public static void main(String[] args) {
   int[]a = {75,53, 32, 12, 46, 199, 17,  54};
   System.out.print("排序前: ");
   for(var n:a) {
        System.out.print(" " + n);
   }
```

```
        bubbleSort(a);
        System.out.print("\n排序后：");
        for(var n:a) {
            System.out.print(" " + n);
        }
    }
```

调用 bubbleSort()方法为数组 a 排序

运行结果如图 5-8 所示。

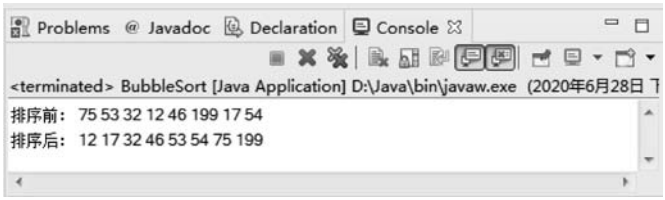

图 5-8 案例运行结果

【提示】 使用 5.4 节讨论的 java.util.Arrays 类的 sort()方法可以对任何类型元素的数组进行排序，还可以对数组执行查找和其他操作。

5.4 java.util.Arrays 类

视频讲解

java.util.Arrays 类定义了若干静态方法对数组操作，包括对数组排序、在已排序的数组中查找指定元素、数组元素的复制、比较两个数组是否相等、将一个值填充到数组的每个元素中。

- public static int binarySearch (int[] a, int key)：根据给定的键值，查找该值在数组中的位置，如果找到指定的值，则返回该值的下标值。如果查找的值不包含在数组中，方法的返回值为(－插入点－1)。插入点为指定的值在数组中应该插入的位置。
- public static void sort(int[] a)：对数组 a 按自然顺序排序。
- public static void sort(int[] a, int fromIndex, int toIndex)：对数组 a 中的元素从起始下标 fromIndex 到终止下标 toIndex 之间的元素排序。
- public static double[] copyOf(double[] original, int newLength)：方法的 original 参数是源数组，newLength 参数是新数组的长度。
- public static void fill (int[] a, int val)：用指定的 val 值填充数组 a 中的每个元素。
- public static boolean equals(boolean[] a, boolean[] b)：比较布尔型数组 a 与 b 是否相等。
- public static String toString(int[] a)：将数组 a 的元素转换成字符串，它有多个重载的版本，方便对数组的输出。

上述操作都有多个重载的方法，可用于所有的基本数据类型和 Object 类型。

【案例 5-4】 用 Arrays 的 sort()方法排序数组。

【程序 5-5】 SortDemo.java

```
package com.boda.xy;
import java.util.Arrays;
```

```
public class SortDemo{
   public static void main(String[] args){
       int[]a = {75,53, 32, 12, 46, 199, 17,  54};
       System.out.print("排序前: ");
       System.out.println(Arrays.toString(a));

       Arrays.sort(a);      ←┤ 调用 Arrays.sort()方法对数组 a 排序
       System.out.print("\n 排序后: ");
       System.out.println(Arrays.toString(a));
   }
}
```

运行结果如图 5-9 所示。

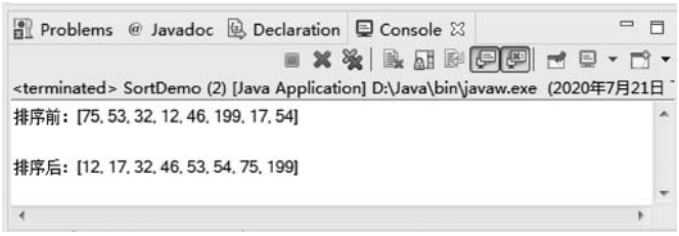

图 5-9　案例运行结果

视频讲解

5.5　案例研究：桥牌随机发牌(一)

1. 问题描述

桥牌是一种文明、高雅、竞技性很强的智力游戏，由 4 个人分两组玩。桥牌使用普通扑克牌去掉大小王后的 52 张牌，分为梅花(C)、方块(D)、红心(H)和黑桃(S)4 种花色，每种花色有 13 张牌，从大到小的顺序为：A(最大)、K、Q、J、10、9、8、7、6、5、4、3、2(最小)。

打桥牌首先需发牌。本案例就是通过编程实现随机发牌，最后，按顺序显示输出 4 个玩家得到的牌。

2. 运行结果

案例运行结果如图 5-10 所示。

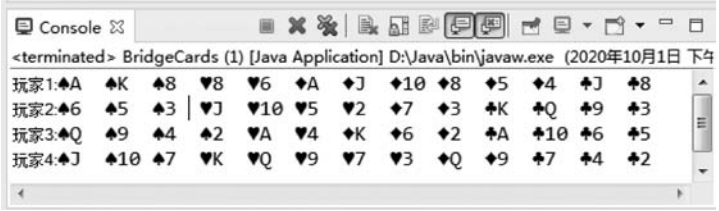

图 5-10　案例运行结果

从输出结果可以看到，每个玩家得到 13 张牌，并且每个玩家的牌按花色和大小进行了排序。

3. 任务目标

(1) 学会分析“桥牌随机发牌(一)”案例的实现思路。

(2) 根据设计思路独立完成“桥牌随机发牌(一)”案例代码的编写、编译和运行。

(3) 掌握数组的声明、初始化和修改元素的方法。

(4) 学会如何使用 Arrays 类的 sort 方法对数组元素排序。

4. 设计思路

根据问题描述,本案例的设计思路如下。

(1) 可以使用一个有 52 个元素的数组存储 52 张牌。为了区分 52 张牌,使用不同的元素值即可,为了方便这里使用 0～51。设元素值 0～12 为黑桃,13～25 为红桃,26～38 为方块,39～51 为梅花。在创建数组后为每个元素赋值,如下所示。

```
int [] deck = new int[52];
for(var i = 0; i < deck.length; i++)          //填充每个元素
  deck[i] = i;
```

(2) 为实现随机发牌,需要打乱数组元素的值,这里对每个元素随机生成一个整数下标,将当前元素与产生的下标的元素交换,循环结束后,数组中的元素值被打乱。

```
for(var i = 0; i < deck.length;i++){
    //随机产生一个元素下标 0～51
   int index = (int)(Math.random() * deck.length);
   int temp = deck[i];          //将当前元素与产生的下标的元素交换
   deck[i] = deck[index];
   deck[index] = temp;
}
```

(3) 牌的顺序打乱后,4 个玩家依次从 52 张牌取出 13 张牌。第 1 个 13 张牌属于第 1 个玩家,第 2 个 13 张牌属于第 2 个玩家,第 3 个 13 张牌属于第 3 个玩家,第 4 个 13 张牌属于第 4 个玩家。最后要求每个玩家的牌按顺序输出,因此需要对每个玩家的牌排序,这里使用 Arrays 类的 sort()方法,它可以对数组部分元素排序,例如对第 3 个玩家的牌排序,使用下面的代码。

```
Arrays.sort(deck,26,39);
```

(4) 最后根据每张牌的数值转换为牌的名称(如♥K)。为此,定义两个 String 数组,如下所示。

```
String[] suits = {"♠","♥","♦","♣"};
String[] ranks = {"A","K","Q","J","10","9","8","7",
                  "6","5","4","3","2"};
```

根据下面的代码确定每张牌的花色和次序,然后输出。

```
String suit = suits[deck[i]/13];                //确定花色
String rank = ranks[deck[i] % 13];              //确定次序
System.out.printf(" % s % - 3s",suit,rank);
```

5. 代码实现

该案例的完整实现代码如程序 5-6 所示。

【程序 5-6】 BridgeCards.java

```
package com.boda.xy;
import java.util.Arrays;
public class BridgeCards {
   public static void main(String[]args){
      int[]deck = new int[52];
      String[] suits = {"♠","♥","♦","♣"};
      String[] ranks = {"A","K","Q","J","10","9","8","7",
                        "6","5","4","3","2"};
      //初始化每一张牌
      for(var i = 0; i < deck.length;i++)
         deck[i] = i;
      //打乱牌的次序
      for(var i = 0; i < deck.length;i++){
         //随机产生一个元素下标 0～51
         int index = (int)(Math.random() * deck.length);
         int temp = deck[i]; //将当前元素与产生的元素交换
         deck[i] = deck[index];
         deck[index] = temp;
      }
      //对指定范围的数组元素排序
      Arrays.sort(deck,0,13);
      Arrays.sort(deck,13,26);
      Arrays.sort(deck,26,39);
      Arrays.sort(deck,39,52);
      //显示所有 52 张牌
      for(var i = 0; i < 52; i++){
         switch(i % 13) {
            case 0 -> System.out.print("玩家" + (i/13 + 1) + ":");
         }
         String suit = suits[deck[i]/13];            //确定花色
         String rank = ranks[deck[i] % 13];          //确定次序
         System.out.printf(" %s %-3s",suit,rank);
         if((i + 1) % 13 == 0) {
            System.out.println();
         }
      }
   }
}
```

视频讲解

5.6 二维数组

Java 语言中的数组元素还可以是一个数组，这样的数组称为数组的数组或二维数组。

5.6.1 二维数组的定义

二维数组的使用也分为声明、创建两个步骤。

1. 二维数组声明

二维数组有下面三种等价的声明格式。

```
元素类型 [][]数组名;          ←┤ 推荐使用第一种格式
元素类型 数组名[][];
元素类型 []数组名[];
```

这里，元素类型可以是基本类型，也可以是引用类型，数组名为合法的变量名。推荐使用第一种格式声明二维数组。下面的语句声明了一个整型二维数组 matrix 和一个 String 型二维数组 cities。

```
int [][] matrix;
String [][] cities;
```

2. 创建二维数组

创建二维数组就是为二维数组的每个元素分配存储空间。系统先为高维分配引用空间，然后再顺次为低维分配空间。二维数组的创建也使用 new 运算符，分配空间有两种方法，下面是直接为每一维分配空间。

```
var matrix = new int[2][3];          //直接为每一维分配空间
```

这种方法适用于数组的低维具有相同个数的数组元素。在 Java 中，二维数组是数组的数组，即数组元素也是一个数组。上述语句执行后创建的数组如图 5-11 所示，二维数组 matrix 有两个元素：matrix[0]和 matrix[1]，它们又都是数组，各有三个元素。图 5-11 中共有三个对象：matrix、matrix[0]和 matrix[1]。

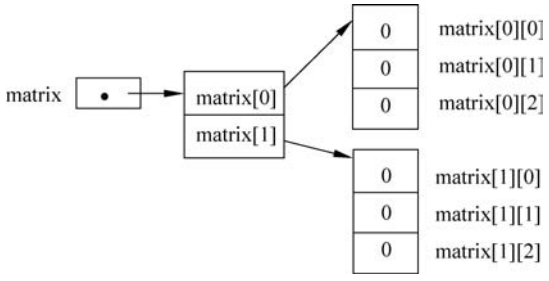

图 5-11 matrix 数组元素空间的分配

创建了二维数组后，它的每个元素被指定为默认值。上述语句执行后，数组 matrix 的 6 个元素值都被初始化为 0。

在创建二维数组时，也可以先为第一维分配空间，然后再为第二维分配空间。

```
var matrix = new int[2][];          //先为第一维分配空间
matrix[0] = new int[3];             //再为第二维分配空间
matrix[1] = new int[3];
```

5.6.2 数组元素的使用

访问二维数组的元素，使用下面的形式。

```
数组名[下标1][下标2]
```

其中,下标1和下标2可以是整型常数或表达式。同样,每一维的下标也是从0到该维的长度减1。

下面的代码给matrix数组元素赋值。

```
matrix[0][0] = 80;
matrix[0][1] = 75;
matrix[0][2] = 78;
matrix[1][0] = 67;
matrix[1][1] = 87;
matrix[1][2] = 98;
```

下面的代码输出matrix[1][2]元素值。

```
System.out.println(matrix[1][2]);
```

与访问一维数组一样,访问二维数组元素时,下标也不能超出范围,否则抛出异常。可以用matrix.length得到数组matrix的大小,结果为2,用matrix[0].length得到matrix[0]数组的大小,结果为3。

对二维数组的第一维通常称为行,第二维称为列。要访问二维数组的所有元素,应该使用嵌套的for循环。如下面的代码输出matrix数组中的所有元素。

```
for(var i = 0; i < matrix.length; i++){
   for(var j = 0; j < matrix[0].length; j ++){
      System.out.print(matrix[i][j] + "  ");
   }
   System.out.println();              //换行
}
```

运行结果如图5-12所示。

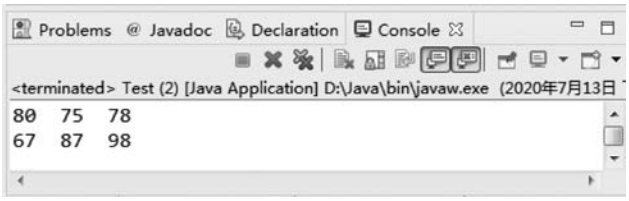

图5-12 运行结果

同样,在访问二维数组元素的同时,可以对元素进行处理,如计算行的和或者列的和等。

5.6.3 数组初始化器

对于二维数组,也可以使用初始化器在声明数组的同时为数组元素初始化,例如:

```
int[][] matrix = {{15,56,20,-2},
                  {10,80,-9,31},
                  {76,-3,99,21},};
```

matrix 数组是 3 行 4 列的数组。多维数组每一维也都有一个 length 成员表示数组的长度。matrix.length 的值是 3，matrix[0].length 的值是 4。

5.6.4 不规则二维数组

Java 的二维数组是数组的数组，对二维数组声明时可以只指定第一维的大小，第二维的每个元素可以指定不同的大小，例如：

```
var cities = new String[2][];              //cities 数组有 2 个元素
cities[0] = new String[3];                 //cities[0]数组有 3 个元素
cities[1] = new String[2];                 //cities[1]数组有 2 个元素
```

这种方法适用于低维数组元素个数不同的情况，即每个数组的元素个数可以不同。对于引用类型的数组，除了为数组分配空间外，还要为每个数组元素的对象分配空间。

```
cities[0][0] = new String("北京");
cities[0][1] = new String("上海");
cities[0][2] = new String("广州");
cities[1][0] = new String("伦敦");
cities[1][1] = new String("纽约");
```

cities 数组元素空间的分配情况如图 5-13 所示，图中共有 8 个对象。

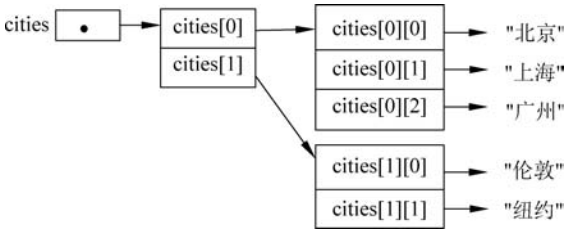

图 5-13 cities 数组元素空间的分配

【案例 5-5】 不规则数组的应用。杨辉三角形，又称帕斯卡三角形，是二项式系数在三角形中的一种几何排列。假设要打印输出前 10 行杨辉三角形，可以用不规则数组存储。

【程序 5-7】 Triangle.java

```
package com.boda.xy;
public class Triangle{
  public static void main(String[] args){
    var level = 10;
    var triangle = new int[level][];
    for(var i = 0;i < triangle.length;i++)
      triangle[i] = new int[i + 1];
    //为 triangle 数组的每个元素赋值
    triangle [0][0] = 1;
    for(var i = 1;i < triangle.length;i++){
      triangle[i][0] = 1;
      for(var j = 1; j < triangle[i].length - 1; j++)
        triangle[i][j] = triangle[i - 1][j - 1] + triangle[i - 1][j];
      triangle[i][triangle[i].length - 1] = 1;
```

```
        }
        //打印输出 triangle 数组的每个元素
        for(var i = 0;i < triangle.length; i++){
          for(var j = 0;j < triangle[i].length;j++)
            System.out.print(triangle[i][j] + " ");
          System.out.println();  //换行
        }
      }
    }
```

运行结果如图 5-14 所示。

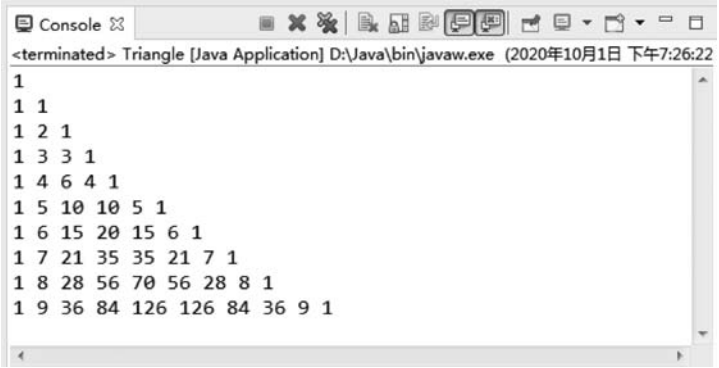

图 5-14　案例运行结果

【提示】 Java 支持多维数组，例如，下面的代码声明并创建了一个三维数组。

double [][][] sales = new double[3][3][4];

视频讲解

5.7　案例研究：考试成绩统计

1. 问题描述

有 5 名学生学年考试成绩如下。

科目 1	科目 2	科目 3	科目 4
80	75	78	93
67	87	98	65
86	72	60	76
76	80	76	63
82	70	90	67

编写程序，使用二维数组存储学生成绩，并完成下列操作。

(1) 计算并输出每名学生的总成绩。

(2) 打印输出总成绩最高的行号及总成绩。

(3) 打印输出每科最高分及所在行号。

2. 运行结果

案例运行结果如图 5-15 所示。

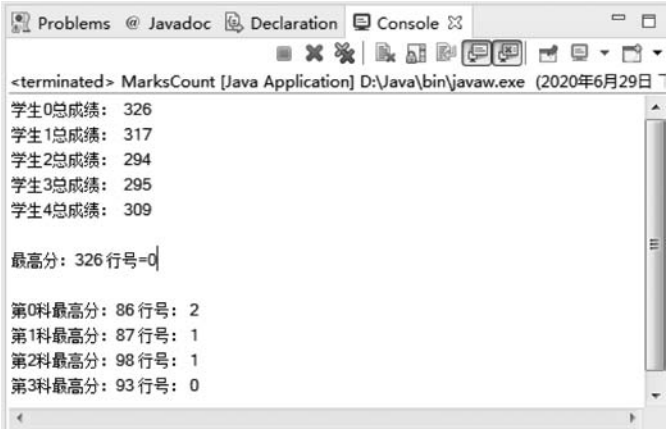

图 5-15 案例运行结果

3. 任务目标

(1) 学会分析“考试成绩统计”案例的实现思路。

(2) 根据设计思路独立完成“考试成绩统计”案例代码的编写、编译和运行。

(3) 掌握二维数组的声明、初始化和元素迭代的方法。

(4) 学会如何对二维数组元素按行和按列计算。

4. 设计思路

根据问题描述，本案例的设计思路如下。

(1) 使用二维数组存放学生的各科成绩，第一行存放第一个学生成绩，第二行存放第二个学生成绩，以此类推。数组的行数为学生数，列数为科目数。

(2) 计算并输出每名学生的总成绩。可以单独定义一个一维数组，存放每个学生的总成绩。

(3) 要打印输出总成绩最高的行号及总成绩，可以从总成绩数组中找到最大值以及它的下标(下标即是行号)。

(4) 打印输出每科最高分及所在行号。可以对成绩数组迭代，计算每列的最大值并输出。

5. 代码实现

该案例的完整实现代码如程序 5-8 所示。

【程序 5-8】 MarksCount.java

```
package com.boda.xy;
public class MarksCount {
  public static void main(String[] args) {
     int [][]marks = {{80, 75, 78, 93},
                      {67, 87, 98, 65},
                      {86,72, 60, 76},
                      {76,80, 76, 63},
                      {82, 70, 90, 67}};
     int [] totals = new int[5];   //存放 5 个学生总成绩
     for(var row = 0; row < marks.length; row++) {
       for(var col = 0; col < marks[row].length; col++) {
```

```
            totals[row] = totals[row] + marks[row][col];
        }                                   求每个学生总成绩
    }
    int max_total = totals[0];
    int index = 0;
    for(var s = 0; s < totals.length ;s++) {
      if(totals[s] > max_total) {
         max_total = totals[s];      ←┤ 求最高分 max_total 及下标 index
         index = s;
      }
      System.out.println("学生" + s + "总成绩:  " + totals[s]);
    }
    System.out.println();
    System.out.print("最高分:" + max_total + " 行号 =" + index + "\n\n");
    for(var col = 0; col < marks[0].length; col++) {
      int max = marks[0][col];
      int r = 0;                        ←┤ max 为列中最大值,r 为行号
      for(var row = 1; row < marks.length; row++) {
         if(marks[row][col] > max) {
            max = marks[row][col];
            r = row;
         }
      }
      System.out.println("第" + col +"科最高分:" + max + " 行号:" + r);
    }
  }
}
```

视频讲解

5.8 案例研究：打印输出魔方数

1. 问题描述

在中国传统文化中，有一种图表叫九宫图，它是在 9 个格子中放置数字 1～9，使得结果的每行、每列以及对角线上的 3 个数字之和相等，如图 5-16 所示。对于该图，古人还给出了图中数字的记忆方法："戴九履一，左三右七，二四为肩，六八为足，五居中央"。

8	1	6
3	5	7
4	9	2

图 5-16 九宫格

数学上已经证明，对于任何 $n \times n$ 个自然数（1，2，…，n^2，n 为奇数）都有这样的排列，使得每行、每列和每条对角线上数的和都相等。这些数称为魔方数。编写程序，要求用户从键盘输入要打印的魔方数（如 5），程序计算并输出魔方数。

2. 运行结果

运行程序，当输入 5 时运行结果如图 5-17 所示。

3. 任务目标

（1）学会分析"打印输出魔方数"案例的实现思路。

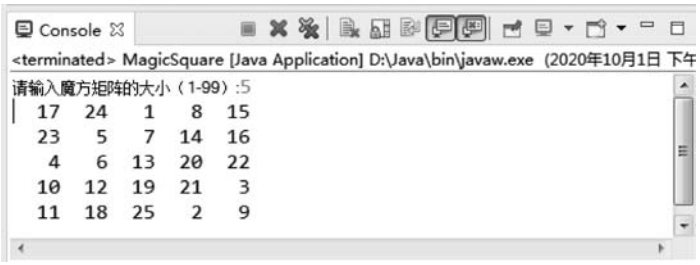

图 5-17 案例运行结果

(2) 根据设计思路独立完成“打印输出魔方数”案例代码的编写、编译和运行。

(3) 掌握二维数组的声明、初始化和修改元素的方法。

4. 设计思路

对于 n 为奇数的魔方数，可以按下列方法将 $1\sim n^2$ 的数填充到二维数组的元素中。首先把 1 放在第 0 行的中间，剩下的数 $2,3,\cdots,n^2$ 依次向上移动一行，并向右移动一列。当可能越过数组边界时需要“绕回”到数组的另一端。例如，如果需要把下一个数放到 -1 行，就将其存储到 $n-1$ 行(最后一行)；如果需要把下一个数放到第 n 列，就将其存储到第 0 列。如果某个特定的数组元素已被占用，就把该数存储在前一个数的正下方。

根据上述规则，可按下列步骤设计实现本案例。

(1) 首先从键盘输入一个正奇数 size，然后定义一个 size×size 的二维数组，代码如下。

```
nt [][] magic = new int[size][size];
```

(2) 将 1 填充到数组的第一行中间元素中，代码如下。

```
int row = 0, col = size/2;
magic[row][col] = 1;          //将 1 填到第一行中间元素
```

(3) 将 2～size×size 填充到数组其他元素中，这里重点是确定下一个元素填充的位置。通过定义临时变量存放元素的行和列，如果下一个位置超出数组元素范围，将改变其行或列的值，最终确定正确位置。

(4) 输出填充了数值的二维数组的每个元素。

5. 代码实现

该案例的完整实现代码如程序 5-9 所示。

【程序 5-9】 MagicSquare.java

```
package com.boda.xy;
import java.util.Scanner;
public class MagicSquare {
   public static void main(String[] args) {
      var input = new Scanner(System.in);
      int size;
      System.out.print("请输入魔方矩阵的大小(1-99):");
      size = input.nextInt();
      if(size < 0 && size % 2 == 0){
         System.out.println("你输入的数不是奇数,不能计算魔方数!");
```

```
            System.exit(0);
        }
        var magic = new int[size][size];
        //用 1 到 size * size 值填充每个元素
        int row = 0, col = size/2;
        magic[row][col] = 1;        //将 1 填到第一行中间元素
        for(var n = 2; n <= size * size; n++){
            int tempx,tempy;
            tempx = row;
            tempy = col;
            tempx = tempx - 1;                  ←┤ 找下一个元素位置,行减 1、列加 1
            tempy = tempy + 1;

            if(tempx < 0)                       ←┤ 行超出范围,变到最后一行
              tempx = size - 1;
            if(tempy == size)                   ←┤ 列超出范围,变到第一列
                tempy = 0;

            f(magic[tempx][tempy]!= 0){
              magic[row + 1][col] = n;          ←┤ 当前位置被占,数放到当前位置下方
              row++;
            }else if(magic[tempx][tempy] == 0) {
              magic[tempx][tempy] = n;          ←┤ 数存放到当前元素
              row = tempx;
              col = tempy;
            }
        }
        //输出数组元素
        for(var i = 0; i < magic.length; i ++){
            for(var j = 0 ;j < magic.length; j ++){
              System.out.printf(" % 4d",magic[i][j]);
            }
            System.out.println();
        }
    }
}
```

关于魔方矩阵,本案例只计算输出奇数的矩阵。实际上,对于偶数的矩阵,也有一些满足这个特性,但有些不满足。读者可自己到网上查找资料,了解它的特性。

小结

本章介绍了 Java 语言的数组,包括数组的定义、元素的访问、元素的复制,数组参数和返回值以及可变参数方法。Arrays 类中定义了一些方法可对数组操作。Java 也支持多维数组并且可以创建不规则数组。

第 6 章将介绍面向对象的基本特征,包括封装性、继承性和多态性,还将介绍对象转换与多态。

习题与实践

一、填空题

1. 写一个语句声明并创建一个名为 numbers 的有 10 个元素的 double 类型的数组：_____。

2. 写一个语句，将 8.88 赋给 numbers 数组的最后一个元素：_____。

3. 写一个语句，打印输出 numbers 数组的前两个元素的和：_____。

4. 如果访问 numbers 数组的 numbers[10]元素，将抛出 _____异常。

5. 可使用 for 循环和增强的 for 循环访问数组元素，如果要修改元素值，应使用_____循环。

6. 写一个语句实现将 numbers 数组复制到 target 数组：_____。

7. 有一个方法声明如下，调用该方法需为其传递_____参数。

```
public static double average(double … values)
```

二、判断题

1. Java 数组声明时不能指定大小，下面的声明是错误的。(　　)

```
int[5] numbers;
```

2. 有一个名为 nums 的数组，要得到该数组的大小可以使用 nums.length()方法。(　　)

3. 访问数组元素，若超出范围将抛出 IndexOutOfBoundsException 异常。(　　)

4. 下面的声明并初始化数组的语句是正确的。(　　)

```
var source = {1, 2, 3, 4, 5, 6, 7};
```

5. 下面代码的输出结果是 6。(　　)

```
int [][]x = {{1,2},{3,4,5},{6,7,8,9}};
int [][]y = x;
System.out.println(y[2][1]);
```

三、选择题

1. 下面叙述正确的两个选项是(　　)。

A. 数组中的每个元素都有相同的类型　　B. 一旦数组被声明，大小就不能改变

C. 一旦数组被创建，大小就不能改变　　D. 数组中的元素必须是基本数据类型

2. 下面对数组的声明并初始化，正确的是(　　)。

A. int arr[];　　B. int arr[5];

C. int arr[5] = {1,2,3,4,5};　　D. int arr[] = {1,2,3,4,5};

3. 下面语句声明的数组能存放 10 个整型数的是(　　)。

A. int[] foo;　　B. int foo[];

C. int foo[10];　　D. Object[] foo;

E. Object foo[10];

4. 执行下列程序段后 foo 的值为(　　)。

```
String foo = "blue";
boolean[] bar = new boolean[1];
if(bar[0]){
     foo = "green";
}
```

A. ""　　B. null　　C. blue　　D. green

5. 下列对数组的声明和初始化正确的有(　　)。

A. int[] j = new int[2]{5,10};　　B. int j[5] = {1,2,3,4,5};

C. int j[] = {1,2,3,4,5};　　D. int j[][] = new int[10][];

6. 下面代码的输出是(　　)。

```
public class Test{
   public static void main(String[] args){
       int list[] = {1, 2, 3, 4, 5, 6};
       for(int i = 1; i < list.length; i++)
          list[i] = list[i-1];
       for(int i = 0; i < list.length; i++)
          System.out.print(list[i] + " ");
   }
}
```

A. 1 2 3 4 5 6　　B. 1 1 2 3 4 5　　C. 1 1 1 1 1 1　　D. 1 2 2 2 2 2

7. 下列代码运行后 baz 的结果为(　　)。

```
int index = 1;
int foo[] = new int[3];
int bar = foo[index];
int baz = bar + index;
```

A. 0　　B. 1　　C. 2　　D. 抛出一个异常

8. 下列程序段的运行结果为(　　)。

```
int [][] array = {{1, 2},{3, 4},{5,6}};
int sum = 0;
for(int i = 0; i < array.length; i++)
      sum = sum + array[i][0];
System.out.println(sum);
```

A. 12　　B. 9　　C. 21　　D. 18

四、讨论题

1. 以下哪些语句是正确的?

```
int i = new int(30);
double []d = new double[30];
char[] r = new char(1…30);
int i[] = (5, 4, 3, 2);
float f[] = {2.3, 4.5, 6.6};
char[] c = new char();
```

2. 下面代码的输出是什么？

```
int x = 3;
int[] numbers = new int[x + 2];
x = 6;
System.out.println("x is " + x);
System.out.println("size is " + numbers.length);
```

3. 找出并修改下面代码中的错误。

```
public class Test{
   public static void main(String[] args){
       double [100] num;
       for(int i = 0; i < num.length; i++)
         num(i) = Math.random() * 100;
   }
}
```

4. 有下面的声明语句，要求使用三种方法完成将 source 数组中的 3、4、5 三个元素值复制到 target 数组中。

```
int [] source = {1, 2, 3, 4, 5, 6, 7};
int [] target = new int[3];
```

5. 下面的代码试图对数组中的内容反转，解释为什么是错误的，请对代码进行修改。

```
int [] list = {1, 2, 3, 4, 5};
for (int i = 0, j = list.length - 1; i < list.length; i++, j-- ){
     int temp = list[i];
     list[i] = list[j];
     list[j] = temp;
}
```

6. 下面方法头的声明有什么错误？

```
public static void print(String ... names, double ... numbers)
public static void print(double ... numbers, String name)
public static double ... print(double d1,double d2)
```

7. 方法 printMax()定义如下。

```
public static void printMax(double ... numbers){
     if(numbers.length == 0){
        System.out.println("没有传递参数");
        return;
     }
     double result = numbers[0];
     for(int i = 1; i < numbers.length; i ++)
        if(numbers[i] > result)
           result = numbers[i];
        System.out.println("最大值是: " + result);
}
```

可以使用下面的语句调用 printMax()方法吗?

```
printMax(1, 2, 2, 1, 4);
printMax(new double[]{1, 2, 3});
printMax(new int[]{1, 2, 3});
```

8. 给出下面代码的输出结果。

```
int [] arr = new int[]{1, 3, 5, 0, 4, 6};
int [] index = new int[]{0, 1, 3, 2, 3, 4, 2, 0, 0, 5, 5};
String tel = " " ;
for (var i : index){
      tel += arr[i];
}
System.out.println("电话: " + tel);
```

9. 给出下面代码的输出结果。

```
int [] list1 = {2, 4, 7, 10};
Arrays.fill (list1, 7);
System.out.println(Arrays.toString(list1));
int [] list2 = {2, 4, 7, 10};
System.out.println(Arrays.toString(list2));
System.out.println(Arrays.equals(list1,list2));
```

10. 以下哪些语句是正确的?

```
int [][] r = new int[2];
int [] x = new int[];
int [][] y = new int[3][];
int [][] z = {{1, 2}};
int [][] m = {{1, 2},{2, 3}};
int [][] n = {{1, 2},{2, 3}, };
```

五、简答题

1. 数组下标是什么类型的? 最小的下标是多少? 如何表示数组名为 a 的第 3 个元素?
2. 当程序尝试访问下标不合法的数组元素时会发生什么?
3. 编写语句完成:

(1) 创建一个含 10 个 double 型值的数组。

(2) 将 5.8 赋值给数组中最后一个元素。

(3) 显示数组前两个元素的和。

(4) 编写循环计算数组中所有元素的和。

(5) 编写循环找出数组的最小值。

(6) 随机产生一个下标,然后显示该下标所对应的数组元素。

(7) 使用数组初始化语法创建另一个初始值为 3.5、5.5、4.52 和 6.6 的数组。

4. 写出下列程序的输出结果。

```
public class ArrayTest{
   public static void main(String[] args){
```

```
        float f1[],f2[];
        f1 = new float[10];
        f2 = f1;
        System.out.println("f2[0] = " + f2[0]);
    }
}
```

5. 写出下列程序的运行结果。

```
public class ArrayDemo{
    public static void main(String[] args){
        int[] a = new int[1];
        modify(a);
        System.out.println("a[0] = " + a[0]);
    }
    public static void modify(int[] a){
        a[0]++;
    }
}
```

6. 写出下列程序的运行结果。

```
public class Test {
    public static void swap(int[] a){
        int temp = a[0];
        a[0] = a[1];
        a[1] = temp;
    }
    public static void main(String[]args){
        int[] a = {1, 2};
        swap(a);
        System.out.println("a[0] = " + a[0] + ",a[1] = " + a[1]);
    }
}
```

7. 写出下列程序的运行结果。

```
public class ArrayDemo{
    public static void main(String[] args){
        int[] array = {1,2,3,4,5};
        printArray(array);
        modify(array);
        printArray(array);
    }
    static void modify(int[] a){
        for(int i = 0; i < a.length; i++)
            a[i] = a[i] * i;
    }
    static void printArray(int[] a){
        for(int i = 0; i < a.length; i++)
            System.out.print(a[i] + "\t");
        System.out.println();
    }
}
```

8. 写出下列程序的运行结果。

```
public class VarargsDemo{
   public static void main(String[] args){
      System.out.println(largest(12, -12,45,4,345,23,49));
      System.out.println(largest( -43, -12, -705, -48, -3));
   }
   private static int largest(int … numbers){
      int currentLargest = numbers[0];
      for(int number:numbers){
         if(number > currentLargest){
            currentLargest = number;
         }
      }
      return currentLargest;
   }
}
```

9. 下面两行的声明是否等价？怎样声明更好些？

```
int []j, k[] ;
int j[], k[][] ;
```

10. 下面代码的输出是什么？

```
int [][] array = new int[5][6];
int [] x = {1, 2};
array[0] = x;
System.out.println("array[0][1] is " + array[0][1]);
```

六、编程题

1. 编写程序，用数组初始化器初始化一个有 5 个 double 型数的数组 numbers，求数组中的最大、最小元素值及各元素的和与平均值。

2. 编写程序，从键盘输入 5 个整数，并存放到一个数组中，然后计算所有元素的和、最大值、最小值及平均值。

3. 编写一个方法，求出一个 double 型数组中的最小元素。

```
public static double min(double[] array)
```

编写测试程序，提示用户输入 5 个 double 型数，并存放到一个数组中，然后调用这个方法返回最小值。

4. 编写程序，用数组初始化器初始化一个有 5 个 int 型数的数组 source，分别使用下面的方法将数组元素复制到目标数组 target 中。

(1) 用 for 循环一个一个元素地复制。

(2) 使用 System 类的 arraycopy()方法复制。

(3) 用 Arrays 类的 copyOf()或 copyOfRange()方法复制。

5. 编写程序，定义一个有 8 个元素的整型数组，然后使用选择排序法对该数组按升序排序。选择排序法先找到数列中最小的数，然后将它和第一个元素交换。接下来，在剩下的数中

找到最小数，将它和第二个元素交换，以此类推，直到数列中仅剩一个数为止。

6. 编程打印输出斐波那契数列的前 20 个数。斐波那契数列中第一个和第二个数都是 1，以后每个数是前两个数之和，用公式表示为：$f_1 = f_2 = 1$，$f_n = f_{n-1} + f_{n-2}(n \geqslant 3)$。要求使用数组存储斐波那契数。

7. 编写一个方法，计算给定的两个数组之和，格式如下。

```
public static int[] sumArray(int[] a, int[] b)
```

要求返回的数组元素是两个参数数组对应元素之和，不对应的元素直接赋给相应的位置，例如，{1,2,4} + {2,4,6,8}={3,6,10,8}。

8. 编写一个方法，合并给定的两个数组，并以升序返回合并后的数组，格式如下。

```
public static int[] arrayMerge(int[] a, int[] b)
```

例如，一个数组是{16,13,15,18}，另一个数组是{29,36,100,9}，返回的数组应该是{9,13,15,16,29,36,100}。

9. 编写程序，从一副 52 张的牌中选出 4 张，然后计算它们的和。A、J、Q 和 K 分别表示 1、11、12 和 13。程序应该显示得到和为 24 的选牌次数。

10. 有下面矩阵 ***A***，编写程序求矩阵 ***A*** 的转置，即得到下面 ***B*** 的结果。

$$\boldsymbol{A} = \begin{pmatrix} 1 & 3 & 5 \\ -3 & 6 & 0 \\ 13 & -5 & 7 \\ -2 & 19 & 25 \end{pmatrix} \quad \boldsymbol{B} = \begin{pmatrix} 1 & -3 & 13 & -2 \\ 3 & 6 & -5 & 19 \\ 5 & 0 & 7 & 25 \end{pmatrix}$$

11. 编程求解约瑟夫(Josephus)问题：有 12 个人排成一圈，从 1 号开始报数，凡是数到 5 的人就离开，然后继续报数，试问最后剩下的一个人是谁？

12. 编程求解八皇后问题。要求在 8×8 的国际象棋盘上放置 8 个皇后(每个皇后只能占一个格子)，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上，问有多少种摆法。图 5-18 就是其中一个解。

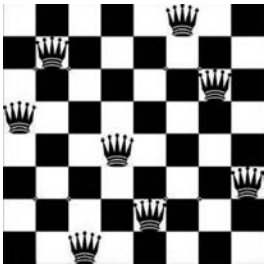

图 5-18 八皇后问题的一个解