# 尚马教育 JAVA 基础课程

# Java数组与方法

文档编号：A04

创建日期： 2017-04-12

最后修改日期：2019-09-23

版 本 号：V3.0

电子版文件名：尚马教育-第一阶段-4.Java数组与方法专题课程.docx

**文档修改记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| 更新日期 | 更新作者 | 更新说明 | 版本号 |
| 2017-07-30 | 张元林 | 初始版本 | V1.0 |
| 2018-08-01 | 王绍成 | Java基础版本更新 | V2.0 |
| 2019-08-09 | 徐丽莎 | Java基础版本更新 | V3.0 |

**主讲人：**

**徐丽莎**

目录

[尚马教育 JAVA 基础课程](#_Toc20943)

[Java数组与方法](#_Toc10852)

[1. 数组](#_Toc16100)

[1.1. 数组特性](#_Toc5851)

[1.2. 一维数组](#_Toc15878)

[1.2.1. 语法](#_Toc11524)

[1.2.2. 案例](#_Toc7552)

[1.2.3. 课堂练习](#_Toc4825)

[1.3. 排序](#_Toc8404)

[1.3.1. 冒泡排序](#_Toc9599)

[1.3.2. 选择排序](#_Toc24983)

[1.3.3. 插入排序](#_Toc3378)

[1.4. 二维数组](#_Toc28714)

[1.4.1. 语法](#_Toc17817)

[1.4.2. 案例](#_Toc31543)

[2. Arrays](#_Toc26064)

[3. 方法](#_Toc15387)

[3.1. 方法的组成部分](#_Toc3495)

[3.1.1. 访问权限修饰符](#_Toc6057)

[3.1.2. 普通修饰符](#_Toc18916)

[3.1.3. 返回值数据类型](#_Toc7649)

[3.1.4. 方法名(目前唯一的)](#_Toc3980)

[3.1.5. 形式参数(形参)](#_Toc10392)

[3.1.6. 方法体](#_Toc11902)

[3.2. 定义(创建)方法](#_Toc6177)

[3.3. 方法调用](#_Toc24039)

[3.3.1. 调用无参无返回值方法](#_Toc26228)

[3.3.2. 调用无参有返回值方法](#_Toc22875)

[3.3.3. 调用有参无返回值方法](#_Toc19568)

[3.3.4. 调用有参有返回值方法](#_Toc31627)

[3.4. 方法重载overload](#_Toc21017)

[4. 作业](#_Toc27645)

## 数组

* 前面我们一直在学习数据类型，Java中有基本数据类型，引用类型。
* 基本数据类型一共有8种
* 引用类型包括类，接口，枚举，注解，虽然这些概念后面才会深入学习
* 试想一下，如果我们班级进行了考试，一共30位学生，我要统计一下平均分，该如何写这个程序呢？可以考虑声明30个变量，然后再一个一个累加起来，再除以30？
* 想起来就很繁琐对不对？那有没有更好的办法，能把这60个数存起来。那就是本节我们要学习的数组，数组也是引用类型的一种。
* 数组是一组类型相同的数据的集合，也就是说，数组中可以存储多个数据，但是这些数据的类型必须相同；
* 数组能够作为数据的容器使用，把多个数据集中存储；
* 存储在数组中的数据，都有相应的索引值，可以方便获取或修改；
* 当需要同时保存多个类型相同的变量并进行处理时，可以考虑用数组，例如：多个人的成绩、多个员工的薪资……

### 数组特性

* Java的数组是引用类型
* Java的数组长度一经确定不能改变；例如，一个数组的长度是10，那么最多能存10个数据，如果保存第11个就会出错；
* 数组在内存中是连续分配，所以读取速度快
* 实际应用中，常常无法确定变量的数量，后续我们将学习集合框架，实现可变长度的数据容器；
* 数组中存储的数据称为数组的元素（Element）；
* 数组本身是引用类型，但是数组中的元素可以是基本数据类型，也可以是引用类型；
* 也就是说，即可以有存储基本数据类型int的数组，也可以有存储引用类型String的数组，但是数组本身是引用类型
* 数组中的元素有索引值，索引值从0开始
* 也就是说，如果一个数组的长度是10，那么索引值就是0-9，也就是第一个元素的索引值是0，第二个的索引值是1，以此类推，通过索引值可以方便访问元素

### 一维数组

* 元素都是单个数据

#### 语法

* 声明一维数组
  + 数组元素类型[ ] 变量名称; (推荐使用)
  + 数组元素类型 变量名称[ ] ;
  + 例如：
  + int[] a; 或 int a[];
  + String[] s; 或 String s[]；
  + 不论数组中元素是什么类型，以上声明形式都适用。
  + 只是声明一个变量，没有初始化操作。
* 一维数组的初始化
  + 数组元素类型[ ] 变量名称=new 数组元素类型[数组长度];
  + 数组元素类型[ ] 变量名称=new 数组元素类型[]{用逗号隔开元素的具体值};
  + 数组元素类型[ ] 变量名称= {用逗号隔开元素的具体值};
  + 例如:

|  |
| --- |
| // a1的长度为5，元素的值为默认值0  **int**[] a1 = **new** **int**[5];  // a2的长度为3，元素的值为1,4,10  **int**[] a2 = **new** **int**[] { 1, 4, 10 };  // a3的长度为4，元素的值为34,23,4,10  **int**[] a3 = { 34, 23, 4, 10 }; |

#### 案例

* 获得数组元素内容
* 对数组元素重新赋值
* 遍历数组元素内容

|  |
| --- |
| //使用数组存储5个学生的成绩  //1.创建数组变量  **int**[] scores = **new** **int**[5];//scores数组变量能够存储5个元素数据 默认数据  System.***out***.println(scores);  //获得数组里面的元素  System.***out***.println("第1个元素:"+scores[0]);  System.***out***.println("第2个元素:"+scores[1]);  System.***out***.println("第3个元素:"+scores[2]);  System.***out***.println("第4个元素:"+scores[3]);  System.***out***.println("第5个元素:"+scores[4]);  System.***out***.println(scores[5]);//java.lang.ArrayIndexOutOfBoundsException: 5  //数组空间大小 通过属性 length进行获取  System.***out***.println("数组的元素个数:"+scores.length);  //赋值  scores[0] = 100;  scores[2] = 90;  scores[3] = 80;  //循环遍历输出数组的元素 for  **for**(**int** index=0,len = scores.length;index<len;index++) {  System.***out***.println("第"+(index+1)+"个元素:"+scores[index]);  }  //增强for循环  **int** count = 1;  **for**(**int** sc:scores) {  System.***out***.println("第"+count+"个元素:"+sc);  count++;  } |

* 动态录入数据存储数组
* 获得数组元素最值

|  |
| --- |
| **public** **static** **void** main(String[] args) {  //动态录入学生个数以及学生的成绩，求出学生的平均成绩)  //最值  Scanner input = **new** Scanner(System.***in***);  System.***out***.println("请录入学生的人数：");  **int** studentNum= input.nextInt();  **double** totalScore = 0;  **int**[] scores = **new** **int**[studentNum];  //循环的录入每个学生的成绩  **for**(**int** i=0;i<studentNum;i++) {  System.***out***.println("请录入第"+(i+1)+"学生的成绩");  **int** score = input.nextInt();  scores[i] = score;  totalScore+=score;  }  **double** avg = totalScore/studentNum;  System.***out***.println("学生的平均成绩:"+avg);  System.***out***.println("所有学生的成绩:"+Arrays.*toString*(scores));  **int** maxScore = scores[0];  **int** minScore = scores[0];  **for**(**int** sc:scores) {  //结合三元运算符  maxScore = (maxScore>sc)?maxScore:sc;  minScore = (minScore<sc)?minScore:sc;  }  **for**(**int** index=1;index<studentNum;index++) {  maxScore = (maxScore>scores[index])?maxScore:scores[index];  minScore = (minScore<scores[index])?minScore:scores[index];  }  System.***out***.println("学生成绩的最大值:"+maxScore);  System.***out***.println("学生成绩的最小值:"+minScore);  } |

* 复制数组的元素到一个新数组中

|  |
| --- |
| **public** **static** **void** main(String[] args) {  //复制数组的元素到一个新数组中  **int**[] nums = {1,2,3,4};  //将nums数组的元素复制到一个新的数组中  **int**[] newNums = nums;//内存地址值  nums[0] = 100;  //修改nums的元素 对newNums的数据有没有影响?  System.***out***.println("--------------------------");  System.***out***.println("nums:"+Arrays.*toString*(nums));  System.***out***.println("newNums:"+Arrays.*toString*(newNums));  System.***out***.println("---------------------------");  //一个数组的元素变化对另外一个数组的元素没有任何影响的  //不共用一块内存 就可以了  **int**[] newNums2 = **new** **int**[nums.length];  //遍历获取原数组的元素 赋值到新的数组中  **int** index = 0;  **for**(**int** num:nums) {  newNums2[index] = num;  index++;  }  nums[0] = 1000;  System.***out***.println("newNums2:"+Arrays.*toString*(newNums2));  System.***out***.println("nums:"+Arrays.*toString*(nums));  System.***out***.println("newNums:"+Arrays.*toString*(newNums));  //Arrays.copyOf(原数组,新数组的空间大小); 数组元素的复制  **int**[] newNums3 = Arrays.*copyOf*(nums, 5);  nums[0] = 2000;  System.***out***.println("newNums3:"+Arrays.*toString*(newNums3));  System.***out***.println("nums:"+Arrays.*toString*(nums));  **int**[] arr = {1,2,3,4};  //扩大数组的空间 手动扩容  arr = Arrays.*copyOf*(arr, arr.length+100);  } |

* 比较两个数组数据是否一致

|  |
| --- |
| **public** **static** **void** main(String[] args) {  // 比较两个数组数据是否一致  **int**[] arr1 = { 1, 2, 3 };  **int**[] arr2 = { 1, 2, 3 };  System.***out***.println(arr1.equals(arr2));//false  **if**(arr1==**null** || arr2==**null**) {  System.***out***.println("数组不能为null的");  **return**;  }  **int** len1 = arr1.length;  **int** len2 = arr2.length;  **if** (len1 != len2) {  System.***out***.println("数组元素不一致");  **return**;  }  // 元素个数一致 相同索引的元素数据是否一致  **for** (**int** index = 0; index < len1; index++) {  **if** (arr1[index] != arr2[index]) {  System.***out***.println("数组元素不一致");  **return**;  }  }  System.***out***.println("数组元素数据一致的");  System.***out***.println(Arrays.*toString*(arr1));  System.***out***.println(Arrays.*toString*(arr2));  //Arrays.equals(数组1,数组2); 比较2个数组的元素是否一致 boolean  System.***out***.println(Arrays.*equals*(arr1, arr2));//true  } |

#### 课堂练习

* 有一列乱序的字符，‘a’,‘c’,‘u’,‘b’,‘e’,‘p’,‘f’,‘z’，排序并按照英文字母表的逆序输出
* 有一个数列：8，4，2，1，23，344，12

1）循环输出数列的值

2）求数列中所有数值的和

3）猜数游戏：从键盘中任意输入一个数据，判断数列中是否包含此数

* 李雷要去买一部手机，他询问了4家店的价格，分别是2899元，2950元，2850元和3100元，显示输出最低价

### 排序

* 实现对数组元素按照升序(降序)排列。
* 目前主要针对基本类型数据。

#### 冒泡排序

* 原理:将相邻的两个元素进行比较，最大的值放在右端。
* N个数字要排序完成，总共进行N-1趟排序，每i趟的排序次数为(N-i)次，所以可以用双重循环语句，外层控制循环多少趟，内层控制每一趟的循环次数。
* 代码演示

|  |
| --- |
| **public** **static** **void** main(String[] args) {  // 对数组元素进行排序(升序)  **int**[] array = { 20, 9, 30, 2 };  System.***out***.println("排序之前:" + Arrays.*toString*(array));  **for** (**int** i = 1, len = array.length; i < len; i++) {// 控制排序轮数  **for** (**int** index = 0; index < len - i; index++) {// 每一轮排序里面要比较多少次  // 相邻元素相互比较  **if** (array[index] < array[index + 1]) {// 升序 判断左边元素是否比右边的元素要大  // 左边元素大  // 左右元素交换位置(2个变量数据相互转换)  **int** temp = array[index];  array[index] = array[index + 1];  array[index + 1] = temp;  }  }  }  System.***out***.println("排序之后:" + Arrays.*toString*(array));  } |

#### 选择排序

* 原理:从第一个元素开始，分别与后面的元素相比较，找到最小的元素与第一个元素交换位置；从第二个元素开始，分别与后面的元素相比较，找到剩余元素中最小的元素，与第二个元素交换；重复上述步骤，直到所有的元素都排成由小到大为止。
* 选择排序是对冒泡排序的改进，它的比较次数与冒泡排序相同，但交换次数要小于冒泡排序。当数据量较大时，效率会有很大的提升.
* 代码演示

|  |
| --- |
| **public** **static** **void** main(String[] args) {  // 对数组元素进行排序(升序)  **int**[] array = { 2, 9, 30, 2,100,1,50 };  System.***out***.println("排序之前:" + Arrays.*toString*(array));  **for** (**int** i = 0, len = array.length; i < len - 1; i++) {  **int** min = array[i];// 最小值  **int** minIndex = i;  **for** (**int** index = i + 1; index < len; index++) {  **if** (min > array[index]) {// 有比所认为的最小值还要小的元素  min = array[index];//  minIndex = index;// 最小值的索引  }  }  // 交换位置（交换1次）  **int** temp = array[i];  array[i] = min;  array[minIndex] = temp;  }  System.***out***.println("排序之后:" + Arrays.*toString*(array));  } |

#### 插入排序

* 原理

将指针指向某个元素，假设该元素左侧的元素全部有序，将该元素抽取出来，然后按照从右往左的顺序分别与其左边的元素比较，遇到比其大的元素便将元素右移，直到找到比该元素小的元素或者找到最左面发现其左侧的元素都比它大，停止.

此时会出现一个空位，将该元素放入到空位中，此时该元素左侧的元素都比它小，右侧的元素都比它大；

指针向后移动一位，重复上述过程。每操作一轮，左侧有序元素都增加一个，右侧无序元素都减少一个。

* 代码演示

|  |
| --- |
| **public** **static** **void** main(String[] args) {  // 对数组元素进行排序(升序)  **int**[] array = { 20, 9, 30, 2 };  System.***out***.println("排序之前:" + Arrays.*toString*(array));  **for** (**int** i = 1, len = array.length; i < len; i++) {// 外层循环 控制轮数 i索引  **int** temp = array[i];  **int** leftIndex = i - 1;  **while** (leftIndex >= 0 && temp > array[leftIndex]) {//......  // 右移  array[leftIndex+1] = array[leftIndex];//....  leftIndex--;  }  // 填补空位  array[leftIndex + 1] = temp;//.....  }  System.***out***.println("排序之后:" + Arrays.*toString*(array));  } |

### 二维数组

* 二维数组的元素是一维数组。

#### 语法

* 声明二维数组
  + 数组元素类型[ ][] 变量名称;
  + 例如：
  + int[][] a;
  + 只是声明一个变量，没有初始化操作。
* 初始化二维数组
  + 数据类型[][] 变量名 = new 数据类型[m][n];

|  |
| --- |
| m表示这个二维数组有多少个一维数组  n表示每一个一维数组中有多少个元素  例如:  **int**[][] arr = **new** **int**[3][2];  定义了一个二维数组arr, 有3个一维数组元素，每个一维数组里面有2个元素。 |

* + 数据类型[][] 变量名 = new 数据类型[m][];

|  |
| --- |
| m表示这个二维数组有多少个一维数组  这一次没有直接给出一维数组的元素个数，可以动态的给出。  例如：  **int**[][] arr = **new** **int**[3][];  arr[0] = **new** **int**[2];  arr[1] = **new** **int**[3];  arr[2] = **new** **int**[1]; |

* + 数据类型[][] 变量名 = {{元素…},{元素…},{元素…}};
  + int[][] arr = {{1,2,3},{4,6},{6}};

#### 案例

* 动态录入班级数，以及每个班级学生的人数，以及每个学生的成绩
* 遍历输出每个班级每个学生的成绩

|  |
| --- |
| **public** **static** **void** main(String[] args) {  //动态录入班级数以及每个班级的学生数，对每个班级的学生动态录入成绩。  Scanner input = **new** Scanner(System.***in***);  System.***out***.println("录入班级数：");  **int** roomNum = input.nextInt();  **double**[][] score = **new** **double**[roomNum][];//声明二维数组里面有roomNum个元素  **for** (**int** i = 0; i < roomNum; i++) {  System.***out***.println("请录入第" + (i+1) + "个班级的学生数:");  **int** studentNum = input.nextInt();  //对二维数组的每个元素进行初始化操作  score[i] = **new** **double**[studentNum];  **for** (**int** j = 0; j < studentNum; j++) {  System.***out***.println("请录入第" + (i+1) + "个班级第" + (j+1) + "个学生的成绩:");  **double** studentScore = input.nextDouble();  //将成绩存入二维数组中  score[i][j] = studentScore;  }  }  //遍历输出每个班级每个学生的成绩。  **int** count1 = 1;  **for**(**double**[] dou:score) {  **int** count2 =1;  **for**(**double** sc:dou) {  System.***out***.println("第" + count1 + "个班级第" + count2 + "个学生的成绩:"+sc);  count2++;  }  count1++;  }  } |

## Arrays

* 操作数组元素的工具类。在java.util.Array 使用此类，需要导入此类所在的包。

#### 常用方法

* + Arrays.toString(数组名); 将数组内容转换成字符串进行输出
  + Arrays.copyOf(源数组,新数组长度); 复制数组元素
  + Arrays.equals(数组1，数组2);
  + Arrays.sort(数组)---->字面量类型数组 默认升序

## 方法

### 方法的组成部分

#### 访问权限修饰符

* + public
  + private
  + protected
  + 缺省(default)

#### 普通修饰符

* + static
  + abstract
  + final
  + .....

#### 返回值数据类型

* 分为2类:
  + 无返回值数据类型 void
  + 有返回值数据类型 基本数据类型+引用数据类型

#### 方法名(目前唯一的)

* 驼峰命名，首字母小写，第二个单词首字母大写

#### 形式参数(形参)

* 无参
* 有参：任意类型 基本数据类型+引用数据类型
  + 字面量: 值传递
  + 引用: 引用传递

#### 方法体

### 定义(创建)方法

* 方法与方法是同级的关系，都在类里面。
* 结合返回值和形式参数进行方法分类

#### 无参无返回值方法

#### 无参有返回值方法

#### 有参无返回值方法

#### 有参有返回值方法

### 方法调用

* 在方法里面调用其他方法。(避免出现自己调用自己，出现死循环的情况)

#### 调用无参无返回值方法

* 方法名();

#### 调用无参有返回值方法

* 方法名();
* 返回值数据类型 变量 = 方法名();

#### 调用有参无返回值方法

* 方法名(实际参数1,实际参数2,实际参数3....);
* 实参数据类型与形参类型保持一致。

#### 调用有参有返回值方法

### 方法重载overload

* 以上我们说到在一个类中，方法的名称是唯一的，不可重复的，但是在很多地方我们会发现一个类中是可以出现同名的方法的。
* 比如：Arrays.toString()

#### 特征

* 方法名一致
* 形参类型不同
* 不考虑返回值和修饰符

## 作业

* 在歌唱比赛中，共有10位评委进行打分，在计算歌手得分时，去掉一个最高分，
* 去掉一个最低分，然后剩余的8位评委的分数进行平均，就是该选手的最终得分。
* 输入每个评委的评分，求某选手的得分
* 声明一个字符串的数组，空间为5个。使用循环接收五个学生的姓名。再使用循环输出这五 个学生的姓名
* 声明一个int型的数组，循环接收8个学生的成绩，计算这8个学生的总分、平均分、最高分、最低分
* 现在有如下一个数组： int[] oldArr={1,3,4,5,0,0,6,6,0,5,4,7,6,7,0,5}; 要求将以上数组中的0项去掉，将不为0的值存入一个新的数组，生成新的数组为

Int[] newArr={1,3,4,5,6,6,5,4,7,6,7,5};

* 有一整数数组，{1,3,-1,5,-2}; 将数据复制到新数组中 ，要求逆序输出新数组中的数，同时并将小于0的元素按0存储
* Int[] array = {2,432,5221,235,2,5352,1,53,5,3,5364,2,2,63,3533,2,53,532,532};

求所有元素和。

输出所有奇数下标元素。如：array[1]

输出所有元素中，值为奇数的。

将所有元素乘二。

将所有元素加到第一个元素中。

将奇数位置元素存到B数组中

偶数元素存到C数组中

分别逆序输出B数组和C数组

* 从键盘接受10个整数保存在数组中，将10个数中最大的与第一个元素交换，最小的与最后一个元素交换，其余元素位置不变，输出数组元素的值
* 熟练创建方法以及调用方法
* 熟练使用方法形式参数以及返回值的问题