# Javase第四课:

## 回顾

### 分支结构 :

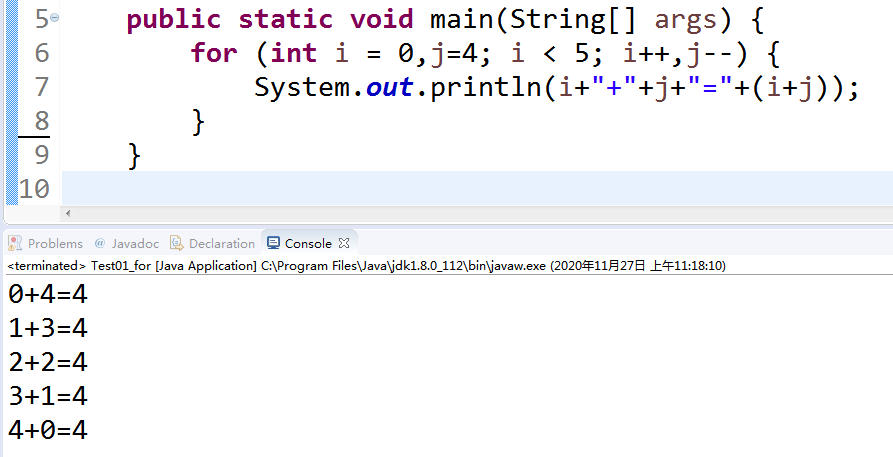
1. if(){} if(){}else{} if(){}else if(){}else{} if(){if(){}}else{}
2. switch(key){case 常量: 语句;break;}

### If和Switch的区别:

1. 区间 (某个范围)的判断优先选择多重if;
2. 等值可以使用switch(推荐)或者if
3. switch中的常量判断的类型规定:

int/byte/short/char/enum(jdk1.5)/String(jdk1.7)

### 循环结构:

  
数据类型[] 变量名; 数据类型 变量名[];

语法:

数据类型[] 变量名=new 数据类型[长度];

数据类型[] 变量名=new 数据类型[]{值1,值2};

数据类型[] 变量名={值1,值2};

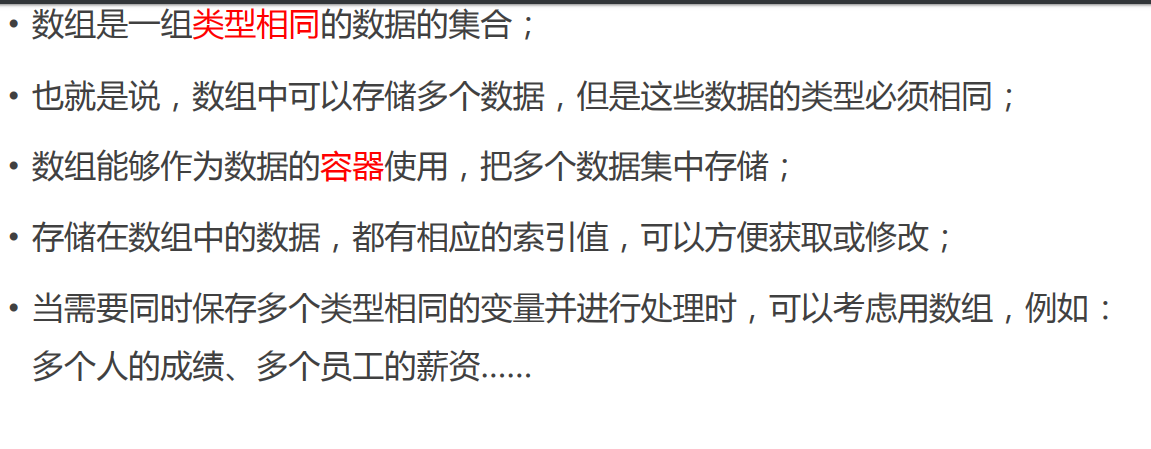
长度:length

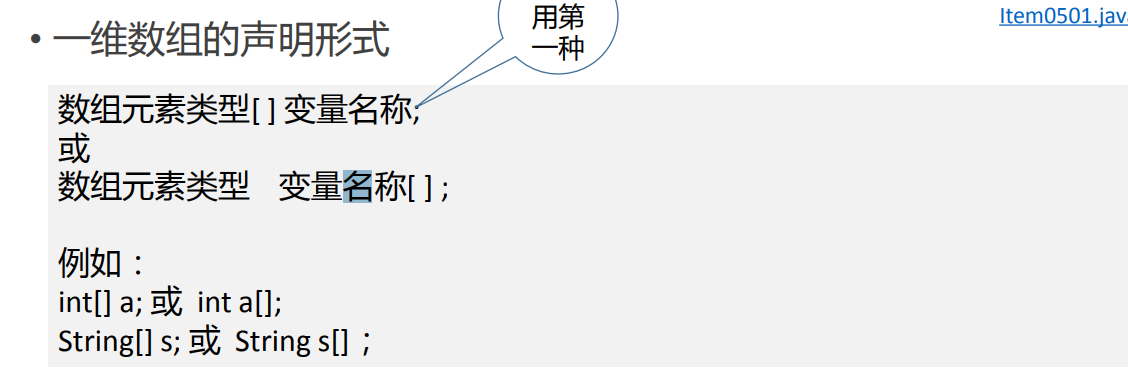
遍历:for循环 for(int i=0;i<数组名.length;i++) {数组名[i]}

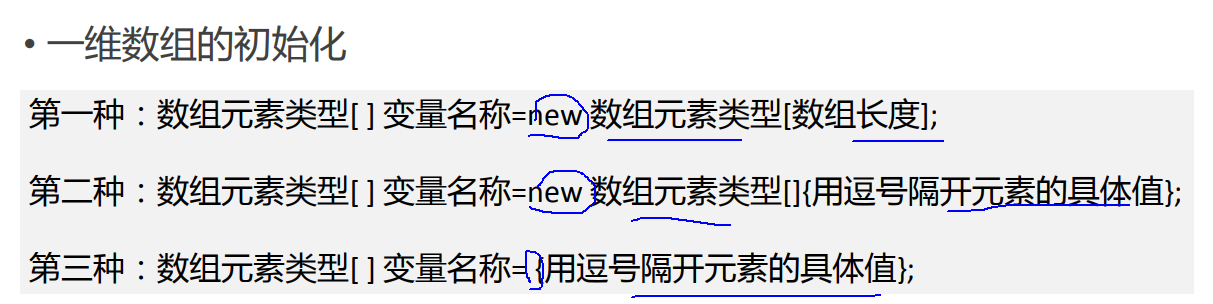
for(数据类型 变量名:数组名){}

## 作业讲解：

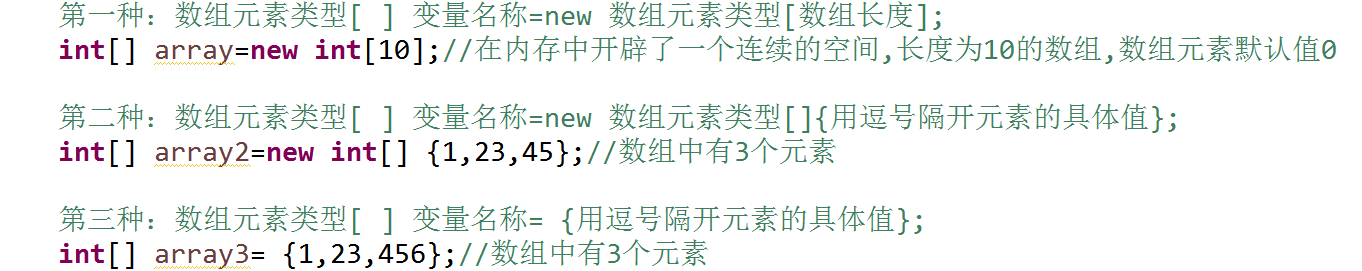
## 数组：







### 定义数组并初始化



### 数组的访问:

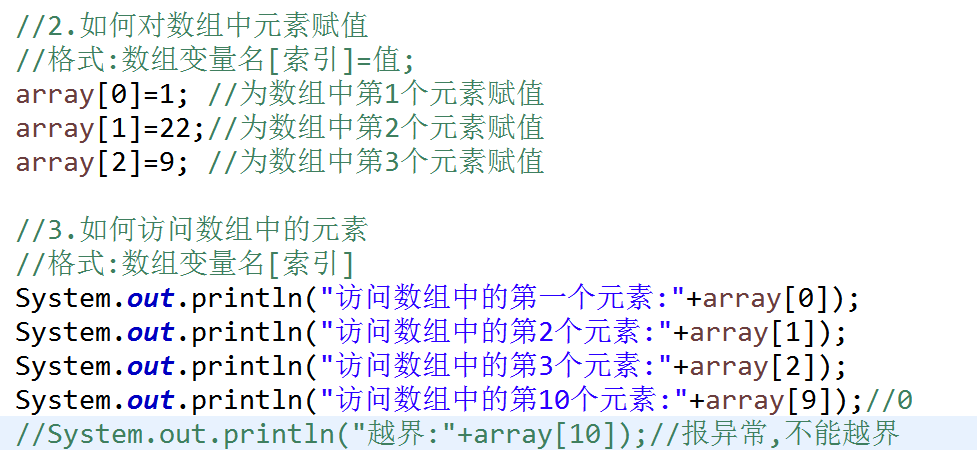
数组是引用类型,数组名array保存的是数组的首地址,指向堆中数组的具体内容

数组元素使用索引表示,索引从0开始,如:array[0],array[1].....

语法:

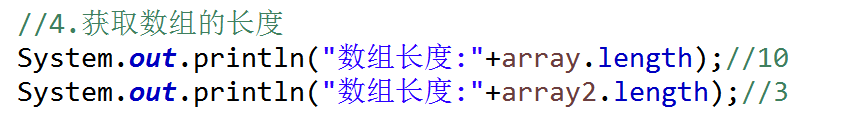
**数组变量名[索引];**//索引从0开始,到长度-1

注意:如果访问数组的下标越界,会出现异常信息



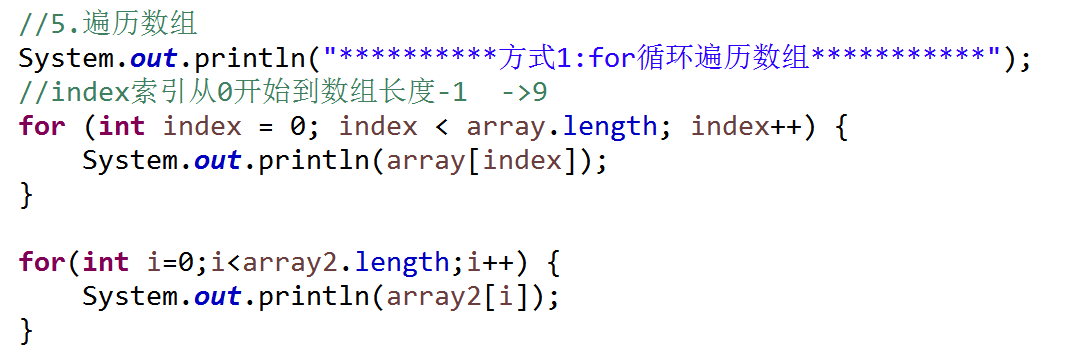
### 数组的长度

使用 数组变量名**.length属性**可以返回数组的长度

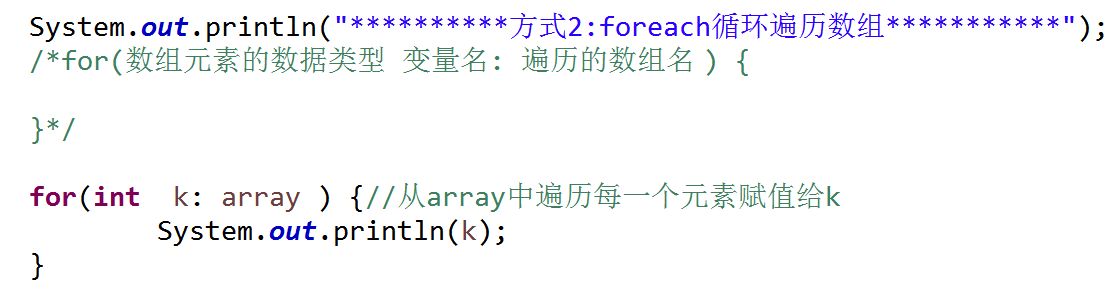


### 数组的遍历

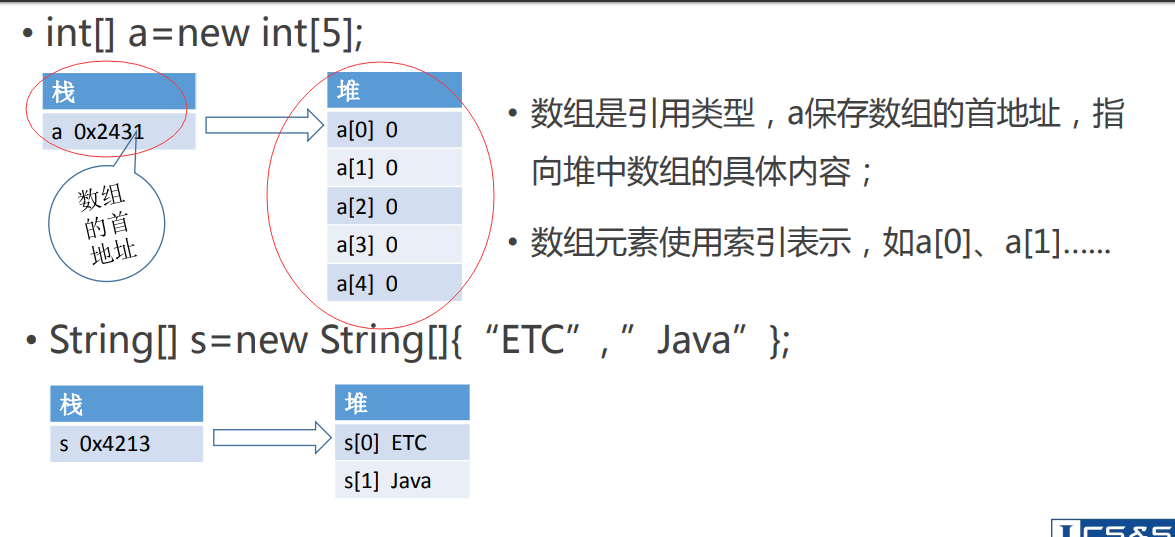
#### 1.for循环



#### 2.增强for循环

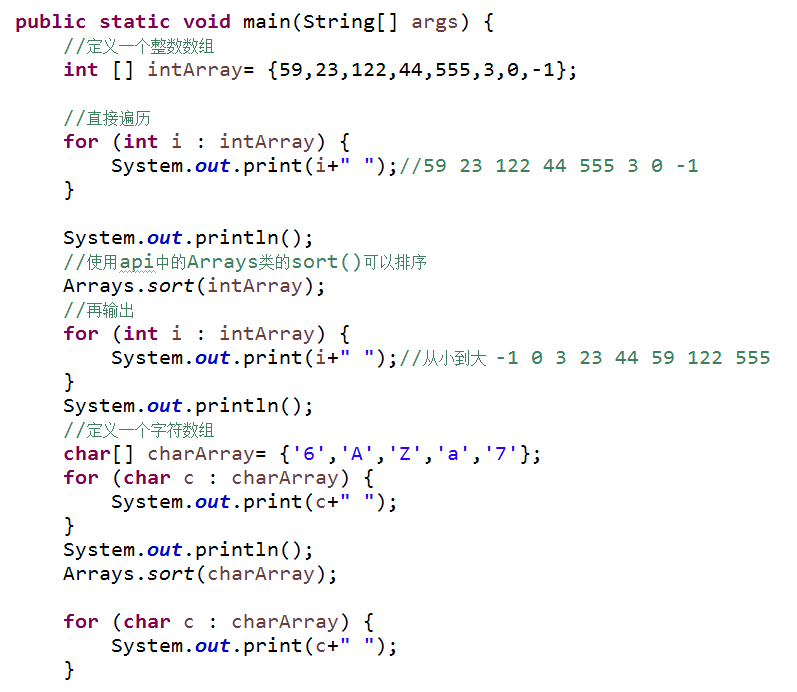


### 数组的存储:



### 数组的排序

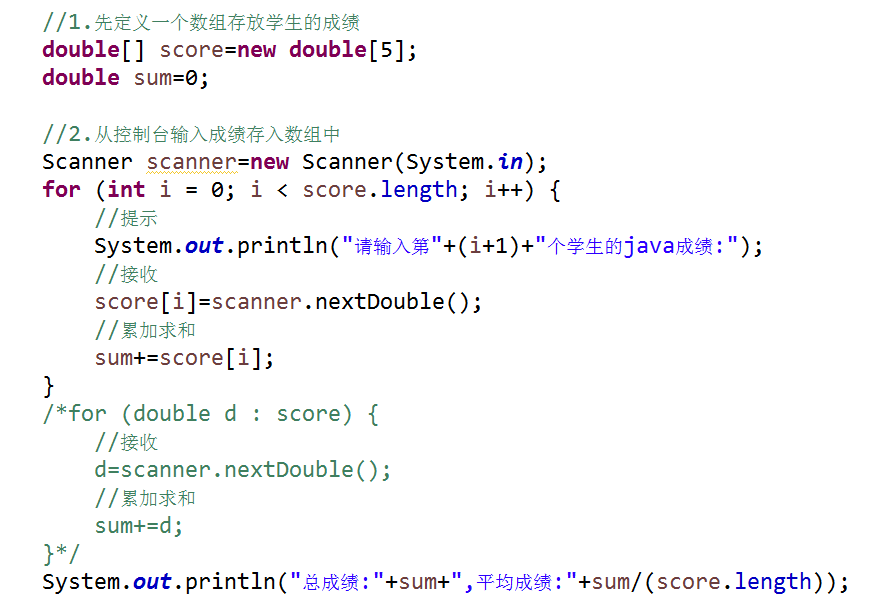
java API中有个类Arrays.定义了大量的sort(),可以对数组中的元素进行排序



## 数组&常见算法问题:

### 和和平均值(累加，总和除以个数)

1. 从控制台接收5个学生的java成绩，求和和平均分;



### 最值(最大值 最小值)

假设第一个元素的值为最大，然后接下来用相邻的值和第一个值比较，保留大的那个值;再用这个值和下一个相邻的元素比较，保留最大的;以此类推 ;直到最后一个元素，得到了所有数字中的最大值;

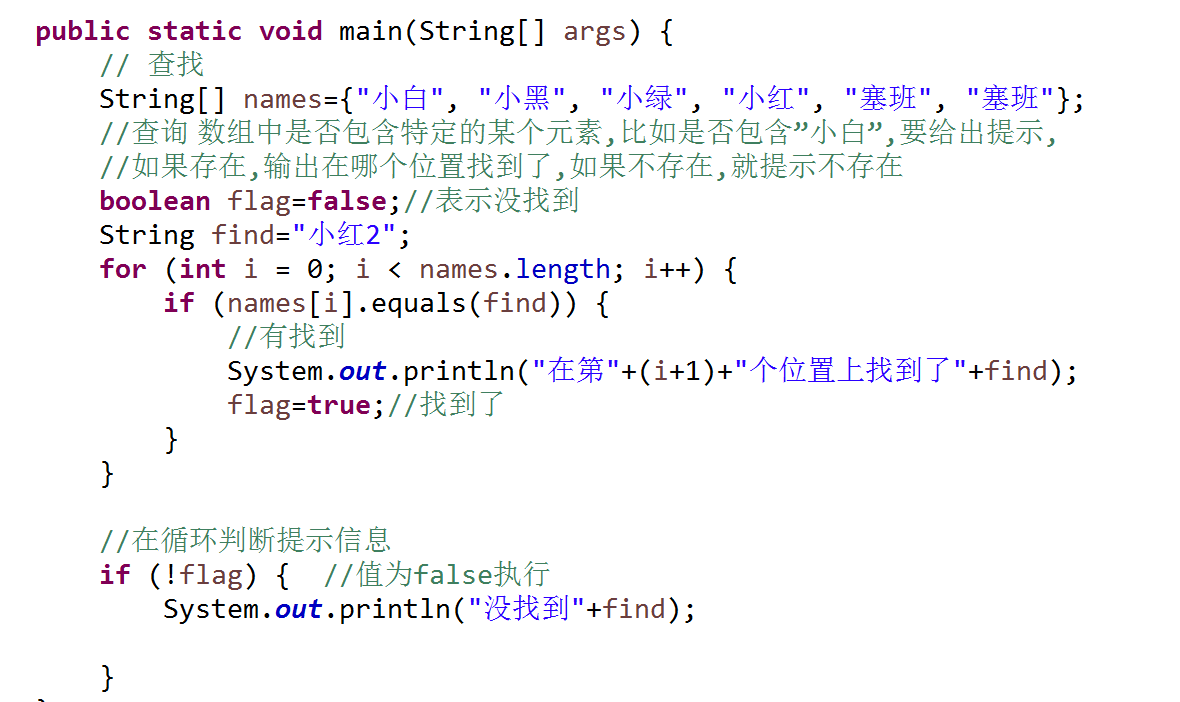


### 查找

String[] names={“小白”, “小宏”, “小兰”, “小黑”, “小绿”, “小白”};

//查询 数组中是否包含特定的某个元素,比如是否包含”小白”,要给出提示,如果存在,输出在哪个位置找到了,如果不存在,就提示不存在

String find=”小白”



### 插入算法(了解)

在一个已经按照特定顺序排好的数组中，我们要插入一个新的元素进来，要求，插入新的元素之后，数组元素依然是有序的.

Int array[]={1,3,5,7,9,0}



### 排序算法:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 12 | 7 | -1 | 8 | 99 |

希望以上一组数能够按照一定的顺序(从大到小/从小到大)重新排列;

相邻的两个元素依次比较，根据比较条件，来交换位置;

第一轮开始：比较了4次,结果是得到了5个数中的最大值

12和7比,用>比较,如果条件为真,则交换两个元素

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | 12 | -1 | 8 | 99 |

12和下一个相邻元素 12>-1, 则交换两个元素

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | -1 | 12 | 8 | 99 |

12和下一个相邻元素 12>8, 则交换两个元素

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | -1 | 8 | 12 | 99 |

继续比较: 12和下一个相邻元素 12>99, 则交换两个元素,为假,什么都不做

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | -1 | 8 | 12 | **99(最大值)** |

第二轮开始：比较了3次,结果是得到了剩下4个数中的最大值

原始数据:

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | -1 | 8 | 12 |

|  |  |  |  |
| --- | --- | --- | --- |
| -1 | 7 | 8 | 12 |

|  |  |  |  |
| --- | --- | --- | --- |
| -1 | 7 | 8 | 12 |
| -1 | 7 | 8 | **12(最大值)** |

继续下一轮：

…

第三轮开始: 比较了2次,结果是得到了剩下3个数中的最大值

第四轮开始: 比较了1次,结果是得到了剩下2个数中的最大值

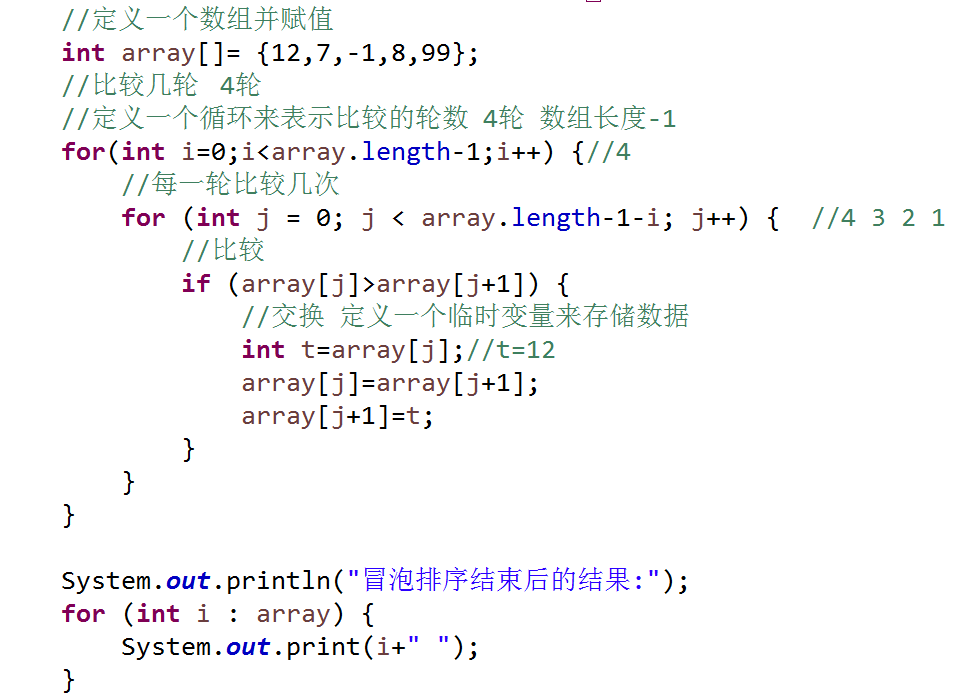
小结:5个数字,比较了4轮;

每一轮内部比较了几次: 4 3 2 1

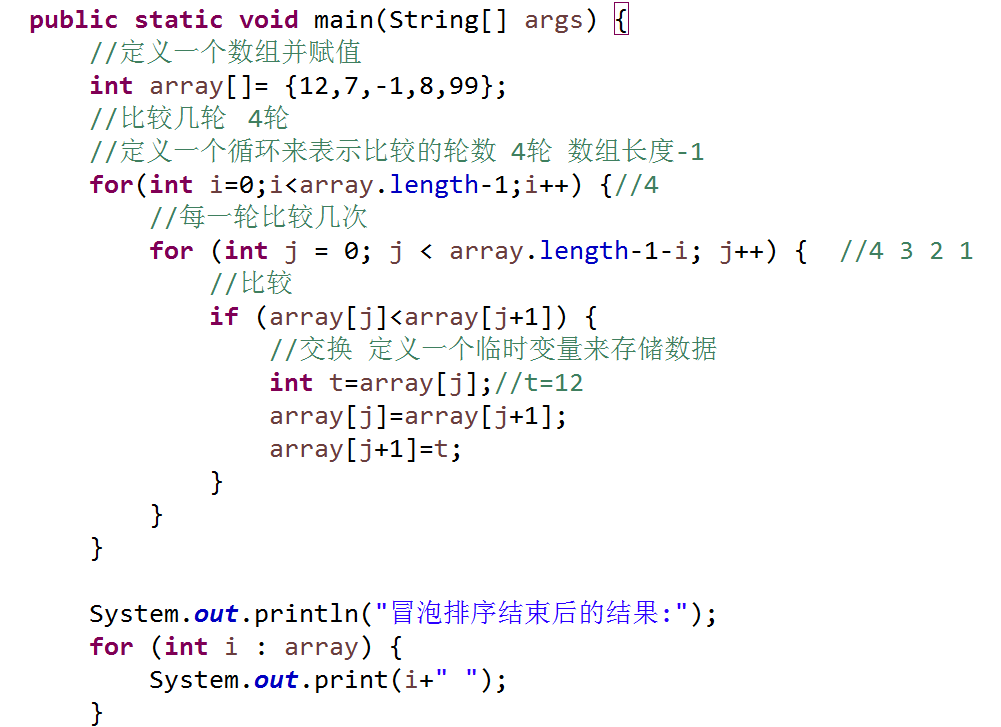
满足条件我们做什么?交换两个相邻的元素

### 参考代码：

冒泡排序: 嵌套循环; 外控制比较的轮数(长度-1) ;内循环控制的是每轮比较的次数(长度-i-1) ; i是外循环变量; 内循环中比较相邻的元素( 使用内循环变量作为下标 array[j]>array[j+1]) ;结果 应该是从小到大;



从大到小:



## 二维数组:

二维数组，在内存中的结构和一维数组是一致的;但是我们可以将其**理解**为一个平面结构;



二维数组可以理解成一个表格：有行有列，行有下标，列也有下标

### 定义和赋值:

数组元素类型[ ][ ] 变量名称=new 数组元素类型[一维长度] [二维长度];

数组元素类型[ ][ ] 变量名称=new 数组元素类型[行数] [列数];

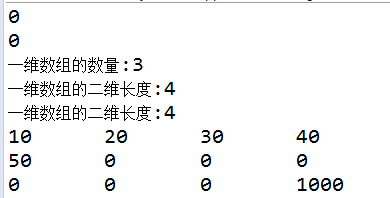
如果同时确定一维和二维的长度，则表示数组的元素是等长的一维数组

int[ ][ ] array=new int[5] [6];//构建一个5行6列的二维数组,这个二维数组中包含5个一位数组,每个一维数组的长度是6(里面包含6个元素)

如果数组元素不是等长的一维数组，可以不指定二维长度；

int[ ][ ] array=new int[5] [];



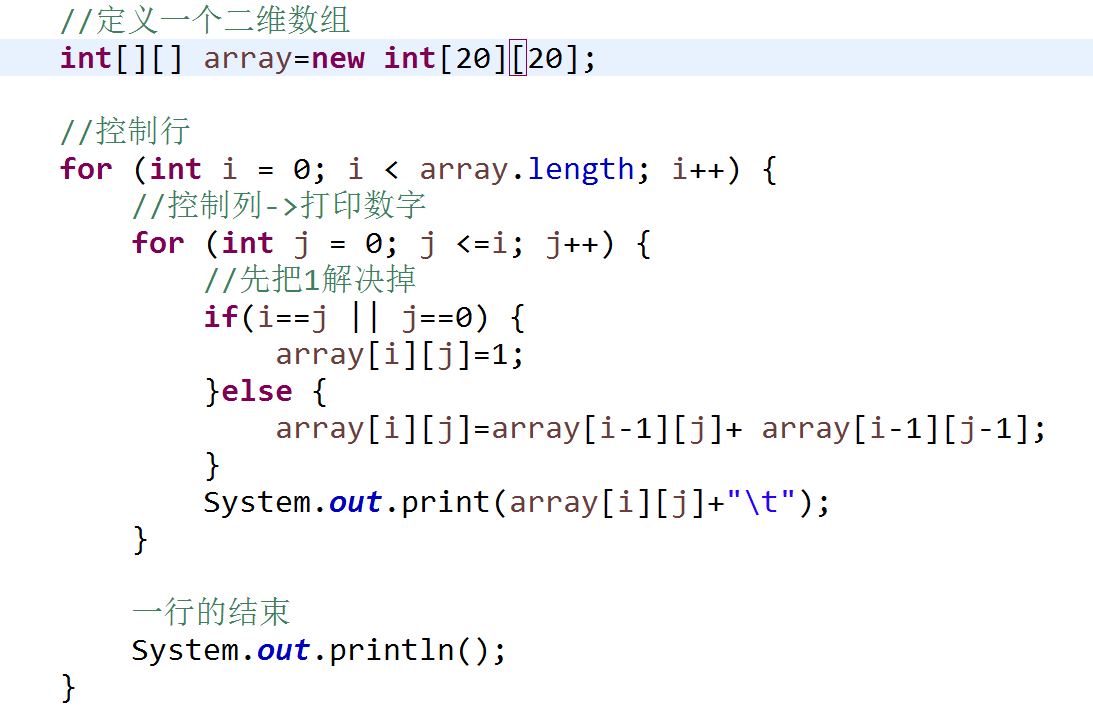


### 案例:杨辉三角

行(i) 列(j) 第一列值为1,以及每一行的最后一列,if(i==j || j==0)

Array[i][j]=array[i-1][j]+ array[i-1][j-1]

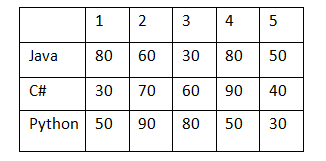
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1(0,0) |  |  |  |  |
| 1(1,0) | 1(1,1) |  |  |  |
| 1 | 2(2,1) | 1(2,2) |  |  |
| 1 | 3 | 3 | 1(3,3) |  |
| 1 | 4 | 6 | 4 | 1(4,4) |



# 作业:

1. 实现等腰三角形的杨辉三角 [5][]
2. 定义二维数组，统计班级5个学生的Java,C#,Python课程的成绩，并求***每门课程***的平均分;

感谢小灰灰的图;



数组定义: double score[][] = new double[3][5];

day06

3.综合练习:[选做]

简单的控制台小程序:

1. 用户登录: 用户名和密码的输入，根据用户名和密码来判断 ;登录成功
2. 用户菜单选项:最大值/最小值/排序/抽奖/退出
3. 当某个功能结束之后，我们再自动回到用户菜单选项

提示：

package day07;

import java.util.Scanner;

public class TestEx {

public static void main(String[] args) {

// 方法的调用

denglu();

}

/\*\*

\* 登录的方法

\*/

public static void denglu() {

// TODO Auto-generated method stub

System.out.println("请输入用户名 密码...");

String username = "admin";

if ("admin".equals(username)) {

System.out.println("登陆成功");

// 去菜单页面

caidan();

} else {

System.out.println("登陆失败");

}

}

/\*\*

\* 菜单的方法

\*/

public static void caidan() {

while (true) {

System.out.println("请输入您的操作:0 退出 1 最大值 2 最小值 3 排序 ");

Scanner input = new Scanner(System.in);

int n = input.nextInt();

switch (n) {

case 1:

zuidazhi();

break;

default:

break;

}

}

}

public static void zuidazhi() {

System.out.println("最大值");

}

}

## 总结:

1：理解：一组类型相同，有序的数据的集合。类型是指我们学过数据类型，有序是指数组中的元素的排序（位置）顺序

2：为什么使用数组？

3.数组的定义

方式1：数据类型[] 数组名 = new 数据类型[常量] //常量表示数组的长度

方式2：数据类型 数组名[] = new 数据类型[常量] //常量表示数组的长度

方式3：数据类型[] 数组名 = { 元素1，元素2，元素3,。。。 }

4.数组的访问--

通过数组名[索引值] 索引值从0开始，到数组长度-1结束; eg:int a[] = new int[5] 访问数组a[0]--a[4]

如果超出范围，出现异常：数组越界

5.数组的赋值--数组名[索引值] = "值"

6.数组的长度：通过length属性，结合for循环遍历数组中的元素

7.掌握数组的简单应用:求和，求平均值，最大值，最小值，查找

8. 二维数组可以理解成一个表格：有行有列，行有下标，列也有下标

9. 定义二维数组 数据类型[][] 数组名 = new 数据类型[行数][列数]

## 作业：

预习面向对象:

1. 