



# 第4章 数 组

## Java中print、printf、println的区别

- ◆ printf主要是继承了C语言的printf的一些特性，可以进行**格式化输出**
- ◆ print就是一般的标准输出，但是**不换行**
- ◆ println和print基本没什么差别，就是最后**会换行**

# Java中print、 printf、 println的区别

◆ `System.out.printf("the number is: d",t);`

参照JAVA API的定义如下：

- ◆ **'d'** 整数结果被格式化为十进制整数
- ◆ **'o'** 整数结果被格式化为八进制整数
- ◆ **'x', 'X'** 整数结果被格式化为十六进制整数
- ◆ **'e', 'E'** 浮点结果被格式化为用计算机科学记数法表示的十进制数
- ◆ **'f'** 浮点结果被格式化为十进制数
- ◆ **'g', 'G'** 浮点根据精度和舍入运算后的值，使用计算机科学记数形式或十进制格式对结果进行格式化。
- ◆ **'a', 'A'** 浮点结果被格式化为带有效位数和指数的十六进制浮点数

# Java中print、printf、println的区别

常用方法	
<b>next()</b>	查找并返回来自此扫描器的下一个完整标记
<b>nextBoolean()</b>	将输入信息的下一个标记扫描为布尔值
<b>nextByte()</b>	将输入信息的下一个标记扫描为Byte
<b>nextDouble()</b>	将输入信息的下一个标记扫描为Double
<b>nextInt()</b>	将输入信息的下一个标记扫描为Int
<b>nextLine()</b>	次扫描器执行当前行，并返回跳过的输入信息
<b>nextLong()</b>	将输入信息的下一个标记扫描为Long
<b>nextShort()</b>	将输入信息的下一个标记扫描为Short

# next() 与 nextLine() 区别

## next():

- 一定要读取到有效字符后才可以结束输入。
- 对输入有效字符之前遇到的空白，next() 方法会自动将其去掉。
- 只有输入有效字符后才将其后面输入的空白作为分隔符或者结束符。
- next() 不能得到带有空格的字符串。

## nextLine():

- 以Enter为结束符,也就是说 nextLine()方法返回的是输入回车之前的所有字符。
- 可以获得空白

# 第四章 数组

- ◆ 创建数组
- ◆ 初始化一维数组
- ◆ 数组名的使用
- ◆ 数组作为方法

# 为什么需要数组3-1



计算全班（36人）的平均分

36个变量太繁琐

```
int stu1 = 95;
```

```
int stu2 = 89;
```

```
int stu3 = 79;
```

```
int stu4 = 64;
```

```
int stu5 = 76;
```

```
int stu6 = 88;
```

.....

```
avg =
```

```
(stu1+stu2+stu3+stu4+stu5...+stu36)/36
```

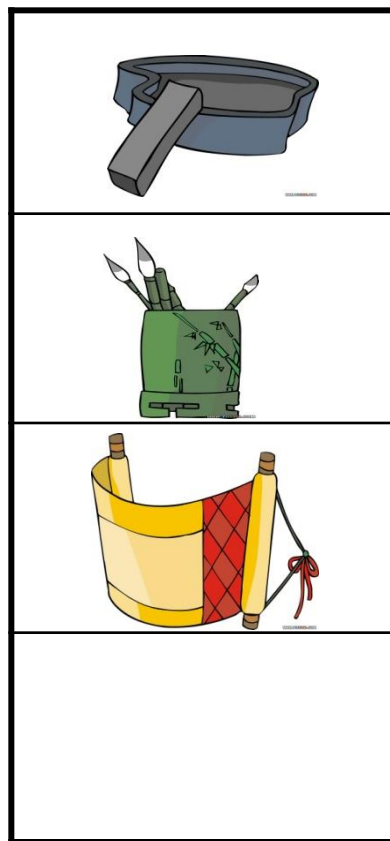
36个变量

# 为什么需要数组3-2

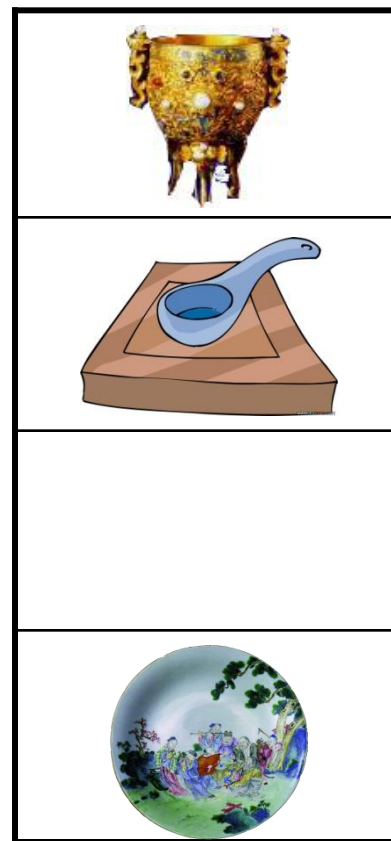
## 生活案例：博物架

**好办法**——分类放，易于找

- 1、格子提供了存储空间
- 2、每一类别都起一个名字
- 3、每件物品都有个标号



字画类



古玩类



# 为什么需要数组3-3

- ◆ 类比博物架：可不可以把数据归类存放？
- ◆ 分类存放不同类型的数据

1
5
20
6
80

int类型

12.5
15.6
66.78
99.5
88.9

double类型

a
g
h
f
u

char类型

## 4.1 创建数组

- 数组声明与基本数据类型类似，要指明数据类型和变量名，但声明时要增加方括号（[ ]）表示数组。
- 创建数组有两个步骤：
  - 声明数组变量。
  - 为数组元素分配存储空间。

## 4.1 创建数组

- 例如，声明一个整数数组：

```
int [ ] intArray ;  
int intArray[];
```

- 数组类型和数组名后，只能有一个方括号（[ ]）。声明数组，不会为数组元素分配存储空间，所以声明时没有数组元素的个数。

## 4.1 创建数组

数组在初始化时才会为数组元素分配空间。数组元素的存储空间是根据数据类型和数组元素的个数计算。因此，**数组初始化时要提供数组元素的个数**。例如，定义一个包括10个整数的数组，并初始化数组元素：

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

new运算符为已经定义了元素类型和个数的数组开辟空间。数组大小定义后就不能更改。new分配空间的同时，将初始化数组元素。

# 数组在内存中的存储



内存

80

整型变量  
`int a = 80;`

100

98

67

78

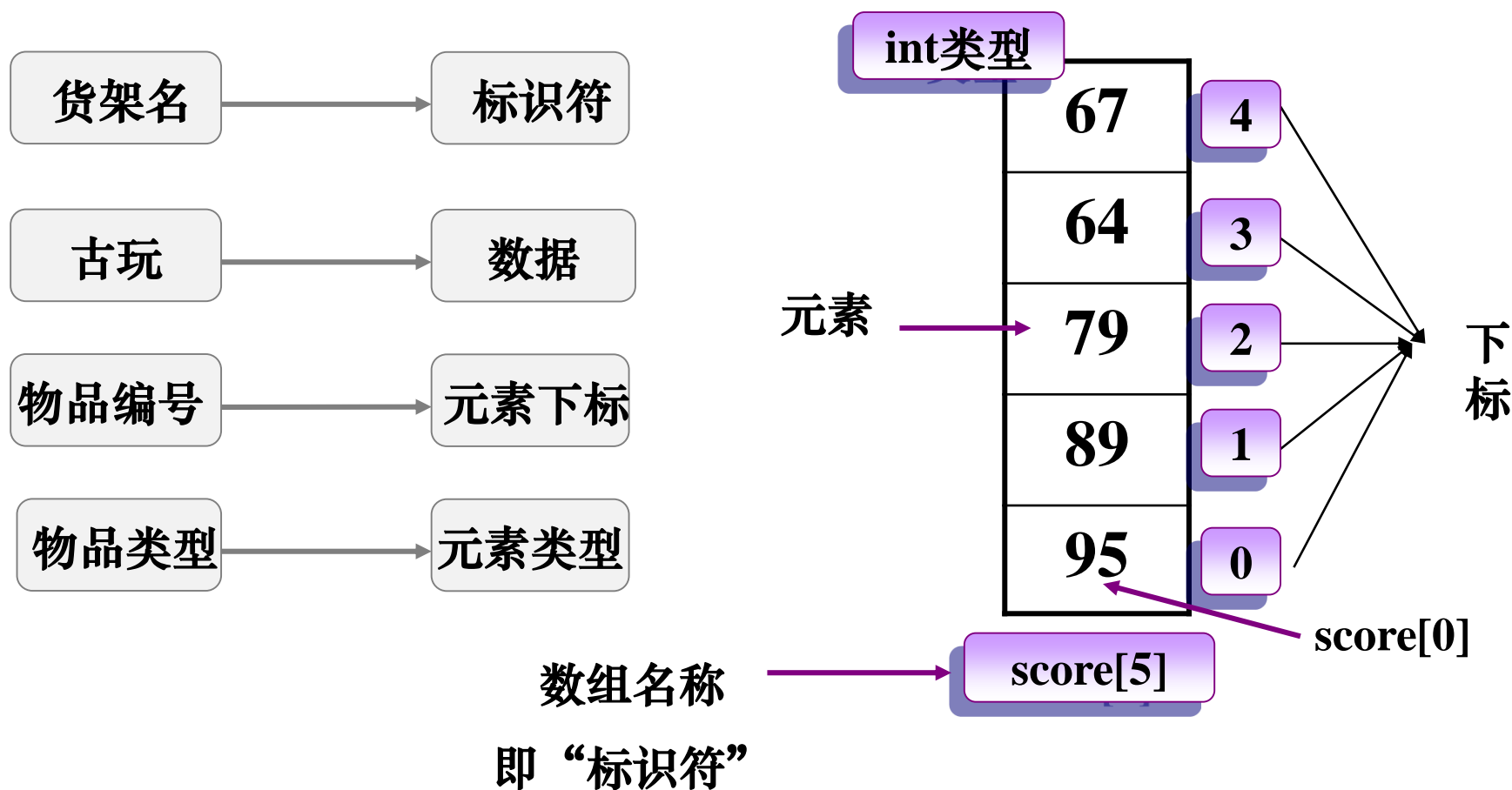
...

82

整型数组  
`int[ ] b = {100, 98, ...};`

# 数组定义

数组是一个变量，连续存储相同**数据类型**的一种结构



# 如何使用数组

## 使用数组：

1、声明数组

```
int[ ] a;
```

2、分配空间

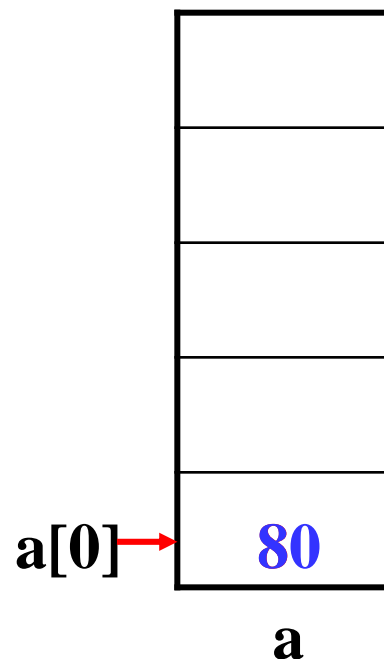
```
a = new int[5];
```

3、赋值

```
a [0] = 8;
```

4、处理数据

```
a [0] = a[0] * 10;
```



只声明未赋值，数组中个元素的值默认为0

# 数组的声明

## 1 声明数组: 告诉计算机数据类型是什么

```
int [ ] score1;           //Java成绩
```

```
int score2[ ];           //C成绩
```

```
String[ ] name;          //学生姓名
```



数据类型 数组名[ ];

数据类型[ ] 数组名;



# Java内存管理

## 栈内存

- ◆ 方法中定义的**基本类型变量**和**对象的引用变量**都在方法的栈内存中分配。
- ◆ 在一段代码块（花括号{ }之间）定义的变量，Java就在栈中为这个变量分配内存空间。
- ◆ **超过变量的作用域**后，Java会自动释放掉分配内存空间。

# Java内存管理

## 堆内存

- ◆ 由new运算符创建的对象或数组。
- ◆ 由Java虚拟机的自动垃圾回收器管理。
- ◆ 堆中产生了数组或对象后，还可以在栈中定义一个特殊的变量。如果该变量的取值等于数组或对象在堆内存中的首地址，栈中的这个变量就成了数组或对象的引用变量。
- ◆ 引用变量是普通的变量，定义时在栈中分配。
- ◆ 数组和对象本身在堆中分配，所占据的内存不会被释放。

## 4.2 初始化一维数组

- 初始化数组就是对已经定义好的数组元素赋初值。

```
int numbers [ ] = new int [ 10];    //创建数组
```

```
for (int i = 0; i < 9; i++)
```

```
numbers[i] = i + 1;                //为每个数组元素  
                                   //赋值为 i+1
```

# 数组赋值2-1

## ③ 赋值：向分配的格子里放数据 .....

```
score[0] = 89;
```

```
score[1] = 79;
```

```
score[2] = 76;
```

```
.....
```

score[2]

score[1]

score[0]

76

79

89

30

太麻烦！能不能一起赋值？

# 数组赋值2-2

解 决

## 方法1: 边声明边赋值

```
int[ ] score = {89, 79, 76};
```

```
int[ ] score = new int[ ]{89, 79, 76};
```

## 方法2: 动态地从键盘录入信息并赋值

```
Scanner input = new Scanner(System.in);  
for(int i = 0; i < 30; i++){  
    score[i] = input.nextInt();  
}
```

# 使用数组求平均分2-1

## 4 对数据进行处理: 计算5位学生的平均分

利用学生人数与数组下标的关系，从键盘输入学生的成绩



```
Console X
<terminated> A5_5 [Java A]
请输入第1个学生
23
请输入第2个学生
34
请输入第3个学生
45
请输入第4个学生
56
请输入第5个学生
54
5个学生的平均分为42
```

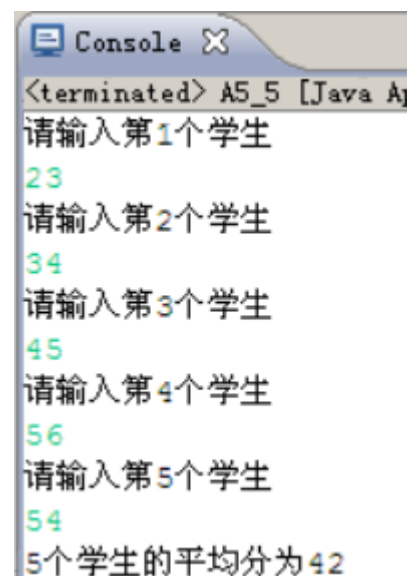
60
80
90
70
85

成绩单

# 使用数组求平均分2-2

## 4 对数据进行处理: 计算5位学生的平均分

```
Scanner input = new Scanner(System.in);  
  
for(int i = 0; i < 5; i++){  
    score[i] = input.nextInt();  
}
```



The screenshot shows a Java console window titled "Console" with a close button. The text inside the window is as follows:

```
<terminated> A5_5 [Java A  
请输入第1个学生  
23  
请输入第2个学生  
34  
请输入第3个学生  
45  
请输入第4个学生  
56  
请输入第5个学生  
54  
5个学生的平均分为42
```

# 常见错误3-1

```
public class zxw{  
    public static void  main(String[ ] args){  
        int[ ] score=new int[ ];  
        score[0]=89;  
        score[1]=63;  
        System.out.print(score[0]);  
    }  
}
```



# 常见错误3-1

```
public class zxw{  
    public static void main(String[ ] args){  
        int[ ] score=new int[ ];  
        score[0]=89;  
        score[1]=63;  
        System.out.print(score[0]);  
    }  
}
```

编译出错，没有写明数组的大小

# 常见错误3-2

```
public class zxw{  
    public static void main(String[ ] args){  
        int[ ] score=new int[2 ];  
        score[0]=89;  
        score[1]=63;  
        score[2]=63;  
        System.out.print(score[0]);  
    }  
}
```

## 常见错误3-2

```
public class zxw{  
    public static void main(String[ ] args){  
        int[ ] score=new int[2 ];  
        score[0]=89;  
        score[1]=63;  
        score[2]=63;  
        System.out.print(score[0]);  
    }  
}
```

编译出错，数组越界

# 常见错误3-3

```
public class zxw{  
    public static void main(String[ ] args){  
        int[ ] score=new int[5 ];  
        score={60,90,80,65,39};  
  
        int[ ] score2;  
        score2={60,90,80,65,39};  
    }  
}
```

# 常见错误3-3

```
public class zxw{  
    public static void main(String[ ] args){  
        int[ ] score=new int[5 ];  
        score={60,90,80,65,39};  
  
        int[ ] score2;  
        score2={60,90,80,65,39};  
    }  
}
```

编译出错，创建数组并赋值的方式必须在一条语句中完成

## 4.2.1 静态初始化

- 静态初始化：直接将数组元素值被花括号括起来并通过逗号分开。

```
Int intArray [ ]={1,2,3,4} ;
```

```
String stringArray[ ]={"abc", "cde", "fgh"} ;
```

编译器通过初始化数组元素值的数量确定数组的大小。静态初始化适用于数组元素的个数较少，且初始元素可以枚举的情况。

## 4.2.2 动态初始化

- 根据数组类型又可分为**简单类型**和**引用类型**，它们的初始化步骤有所不同。

**简单类型**：用new运算符分配内存空间，同时给数组元素赋值为默认值，不同的数据类型，默认值不同。

```
int intArray [ ] ;
```

```
intArray= new int [5] ;
```

- **引用类型**：对引用类型的数组初始化，需要分两步完成。
  - ◆ 为数组分配每个元素的引用空间。
  - ◆ 为每个数组元素分配空间并赋初值。

```
String StringArray [ ] ;
```

```
StringArray=new String [3 ] ;
```

```
StringArray[0]=new String (“how”);
```

```
StringArray[1]=new String (“are”);
```

```
StringArray[2]=new String (“you”);
```



- 数组创建后，通过下标访问数组元素

**StringArray[ index];**

- 每个数组都有一个属性**length**，指明数组的长度。

例如：“**intArray.length** 指明数组**intArray**的长度。

# 数组的length属性2-1

## ④ 对数据进行处理：输入学生人数，求这些学生的平均分

数组名.length：查看数组的长度



```
Console X
<terminated> AS_6 [Java Appli
10
请输入第1个学生
324
请输入第2个学生
434
请输入第3个学生
545
请输入第4个学生
656
请输入第5个学生
434
请输入第6个学生
546
请输入第7个学生
7676
请输入第8个学生
545
请输入第9个学生
535
请输入第10个学生
545
10个学生的平均分为1224
```

# 数组的length属性2-2

## 4 对数据进行处理: 计算5位学生的平均分

```
int sum=0;
int num=0;
Scanner input = new Scanner(System.in);
System.out.print("请输入学生人数: ");
num=input.nextInt();
int[ ] score=new int(num);
for(int i = 0; i < score.length; i++){
    System.out.print("请输入第"+(i+1)+"位学生的成绩: ");
    score[i] = input.nextInt();
    sum+=score[i];
}
System.out.print("学生的平均成绩为: "
    +(double)sum/score.length);
```



```
<terminated> A5_6 [Java Appli
10
请输入第1个学生
324
请输入第2个学生
434
请输入第3个学生
545
请输入第4个学生
656
请输入第5个学生
434
请输入第6个学生
546
请输入第7个学生
7676
请输入第8个学生
545
请输入第9个学生
535
请输入第10个学生
545
10个学生的平均分为1224
```

## 4.3 数组名的使用

数组是引用类型变量，数组名中存储的不是数据元素，而是第一个数组元素的地址，对数组的操作多数要通过数组名。

## 4.4 .1 数组作为方法的参数

- ◆ 基本数据类型作为方法的参数时，实参是将数值传递给方法。
- ◆ 数组不是基本数据类型，作为方法参数时，情况是不同的。
- ◆ 数组是复合数据类型，作为方法参数，数组将引用即数组的首地址传递给方法而不是数组元素值。

例如：书上例4.3

## 4.4.2 数组作为返回类型

数组可以作为方法的输入参数，也可以作为方法的返回值。方法返回数组类型，实际返回的是数组名，即指向数组的引用。

## 4.5 增强的for 循环

- 处理数组时，经常需要用到循环，for循环是使用最频繁的。
- 在for循环中，循环计数器在循环体内可以作为数组的下标。JDK
- 5.0增强了for循环的功能。

## 4.5 增强的for 循环

增强的for循环在每次迭代中都通过一个变量存储数组中连续的数组元素，增强的for循环语法为：

```
for ( dataType item: arrayName) {  
    System.out.println(item);  
}
```



## 4.5 增强的for 循环

- ◆ 普通的for循环访问数组元素语句：

```
for(int i=0;i<a.length;i++)  
    System.out.print(a[i]+ " ");
```

- ◆ 用增强的for循环：

```
for ( int i: a )  
    System.out.print(i);
```

## 4.5 增强的for 循环

与标准的for循环相比，这种方式更加简洁。但是，如果**修改数组元素**，增强的for循环就不再适用。用增强的for循环修改数组元素**会引起编译错误**，同时也会使程序运行**可靠性降低、安全性差**。因此只有在以下情况才可以使用增强的for循环：

- ◆ 需要**访问整个数组**（不是数组的一部分）
- ◆ 需要**读取数组中的元素**，而不是修改。
- ◆ 不需要使用数组下标完成其他处理。

# 利用Array类中的toString方法

调用 **Array.toString(a)**，返回一个包含数组元素的字符串，这些元素被放置在括号内，并用逗号分开。

```
int[] array = {1,2,3,4,5};
```

```
System.out.println(Arrays.toString(array));
```

## 4.7 多维数组

**Java中没有真正的多维数组，只有数组的数组。在应用上很像C语言的多维数组，但是不同。Java中多维数组不一定是规则矩阵形式**

## 4.7.1. 二维数组定义

二维数组定义的一般格式为：

```
type arrayName[ ][ ];
```

或

```
type [ ][ ] arrayName;
```

## 4.7.2 初始化二维数组

### ◆ 静态初始化

```
int intArray[ ][ ]={{1,2},{2,3},{3,4,5}};
```

### ◆ 动态初始化

```
arrayName = new type [length1][length2];  
int a[ ][ ] = new int[2][3];
```

# 数组小结4-1

## Java中都有哪些数组类型

**int**

**int[]**

**double**

**double[]**

**char**

**char[]**

**float**

**float[]**

**Scanner**

**Scanner[]**

**Random**

**Random[]**

# 数组小结4-2

## 使用数组的方式

方法1:

```
int[ ] score ;  
  
score=new int[36];  
  
score[0]=89;
```

方法2:

```
int[ ]score=new int[36];  
score[0]=89;
```

方法3:

```
int[ ] score = new int[ ]{89, 79, 76};  
或: int[ ]score={89,79,76};
```



# 数组小结4-3

声明数组并分配空间，但未赋值，数组各元素默认值？

## ◆ 数值类型

```
int[ ] score=new int[3];  
System.out.println(score[1]);
```

运行结果：默认值为0

## ◆ 字符串：

```
String[ ] name = new String[3 ];  
System.out.println(name[1]);  
if(name[1]==null){  
    System.out.println(“这个数组下标为1的没有数据”);  
}
```

运行结果：默  
认为null

# 数组小结4-3



● 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

# 数组小结4-4



现场编程

小明要去买手机，他询问了4家店的价格，分别是2800元，2900元，2750元和3100元，显示输出最低价

//计算最小值

```
int min=list[0]
```

```
for(int i=0;i<list.length;i++){
```

```
    if(min>list[i]){
```

```
        min=list[i]; //交换
```

```
    }
```

```
}
```



# 循环录入5位学生的java成绩，进行升序排序后输出结果

## 使用java.util.Arrays类

- ◆ java.util包提供了许多存储数据的结构和有用的方法
- ◆ Arrays类提供许多方法操纵数组，例如：排序，查询
- ◆ Arrays类的sort()方法：对数组进行升序排列

**Arrays.sort(数组名) ;**

# 数组排序2-2



## 完整代码演示

```
import java.util.* //导入包
```

```
.....
```

```
int[ ] score = new int[5];
```

```
Scanner in = new Scanner(System.in);
```

```
System.out.print("请输入5位同学的成绩:");
```

```
for (int i=0; i<5;i++){
```

```
    score[i] = in.nextInt();    //依次录入5位同学的成绩
```

```
}
```

```
Arrays.sort(score); // 排序
```

```
for (int j=0;j<score.length;j++){
```

```
    System.out.println(score[j]); // 输出结果
```

```
}
```

循环输入学生的成绩  
并存储在数组中

数组中的元素被  
重新排列

循环输出数组  
中的信息

# 数组排序小结



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

```
import java.util.* //导入包
```

```
.....
```

```
String[ ] zimu = new String[8];
```

```
Scanner in = new Scanner(System.in);
```

```
for (int i=0; i<5;i++){
```

```
    System.out.print("请输入第"+(i+1)+"个字母:");
```

```
    zimu[i] = in.nextInt();    //依次录入8个字母
```

```
}
```

```
Arrays.sort(zimu); // 排序
```

```
for (int j=zimu.length-1;j>=0;j--){
```

```
    System.out.println(zimu[j]); // 输出结果
```

```
}
```

循环输入字母并存储  
在数组中

数组中的元素被  
重新排列

循环输出数组  
中的信息



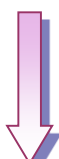

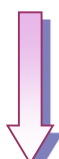

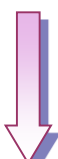


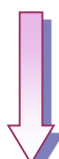
## 1、数组的复制



arrA:

2	1	5	8	7	6	3	10
---	---	---	---	---	---	---	----

arrB:

							
2	1	5	8	7	6	3	10



## 1、数组的复制

演示完整代码

.....

```
int[ ] shuzuA = {1,2,2,3,4,5,6,7};  
int[ ] shuzuB =new int[shuzuA.length];  
for (int i=0; i<shuzuA.length;i++){  
    shuzuB[i ] = shuzuA[ i] ; //依次将shuzuA的8个元素值复制给shuzuB  
}  
for (int j=0;j<shuzuB.length;j++){  
    System.put.println(shuzuB[j]); // 输出结果  
}
```

数组B的长度就是数  
组A的长度

循环复制数组  
元素值





示例

## 2、数组的查找：查找叫“麻子”的同学



分析

arrA:

↓	↓	↓	↓
张三	李四	王二	麻子

每个元素依次查找，找到“麻子”显示“找到此人！”未找到显示“查无此人！”



## 2、数组的查找：查找叫“麻子”的同学

完整代码

```
System.out.print ( “请输入要查找的学生的姓名： ” );
String name=in.next();
String[ ] xm ={” 张三” , ” 李四” , ” 王二” , ” 麻子” };
boolean flag=false;
for (int i=0; i<xm.length;i++){
    if(name.equals(xm[i])){
        flag=true ; //查到将flag标识置为真
        break;
    }
}
if(flag==true){
    System.out.println( “找到此人！ ” );    // 输出结果
}
else{
    System.out.println( “查无此人！ ” );
}
```

设置一个标识



示例



分析

## 3、数组的插入

数组中个元素按升序排列，当最后一个数被替换，则需重新排序

与前一个元素比较大小，比前一个数小时，两者交换

1	7	9	16	18	21	24	55
---	---	---	----	----	----	----	----





## 3、数组的插入

演示完整代码

```
int[ ] shz = {1,32,34,55,61,77};  
System.put.println("请输入要插入的数据:");  
int num = in.nextInt( );  
shz(shz.length-1) = num;  
for (int i=shz.length-1;i>0;i--){  
    if (shz[i]<shz[i-1]){  
        int t = shz[i];  
        shz[i] = shz[i-1];  
        shz[i-1] = t ; // 后一个元素小于前一个元素时, 交换值  
    }  
    for (int j=0;j<shz.length;j++){  
        System.put.println(shz[j]); // 输出结果  
    }  
}
```

将输入的数据插入为  
最后一个元素

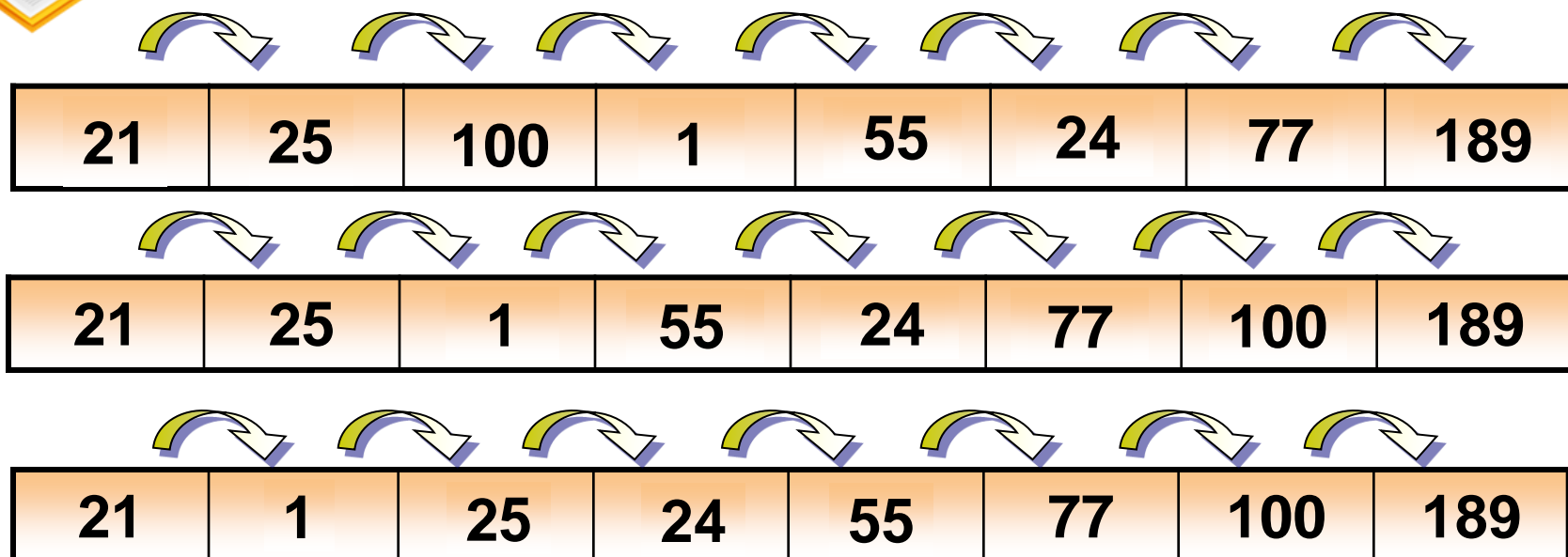
比较交换

输出结果



## 4、数组的冒泡排

数组中冒泡排序算法  
是数组中的最基本的  
算法





## 4、数组的冒泡排序

演示完整代码

```
int[ ] shz = {98,32,21,19,61,55};
```

```
for(int i=0;i<shz.length-1;i++){
```

循环嵌套（双重  
循环）

```
    for (int j=0;j<shz.length-1;j++){
```

```
        if (shz[j]>shz[j+1]){
```

```
            int t = shz[j];
```

```
            shz[j] = shz[j+1];
```

```
            shz[j+1] = t ;    // 后一个元素小于当前元素时, 交换值
```

```
        }
```

```
    }
```

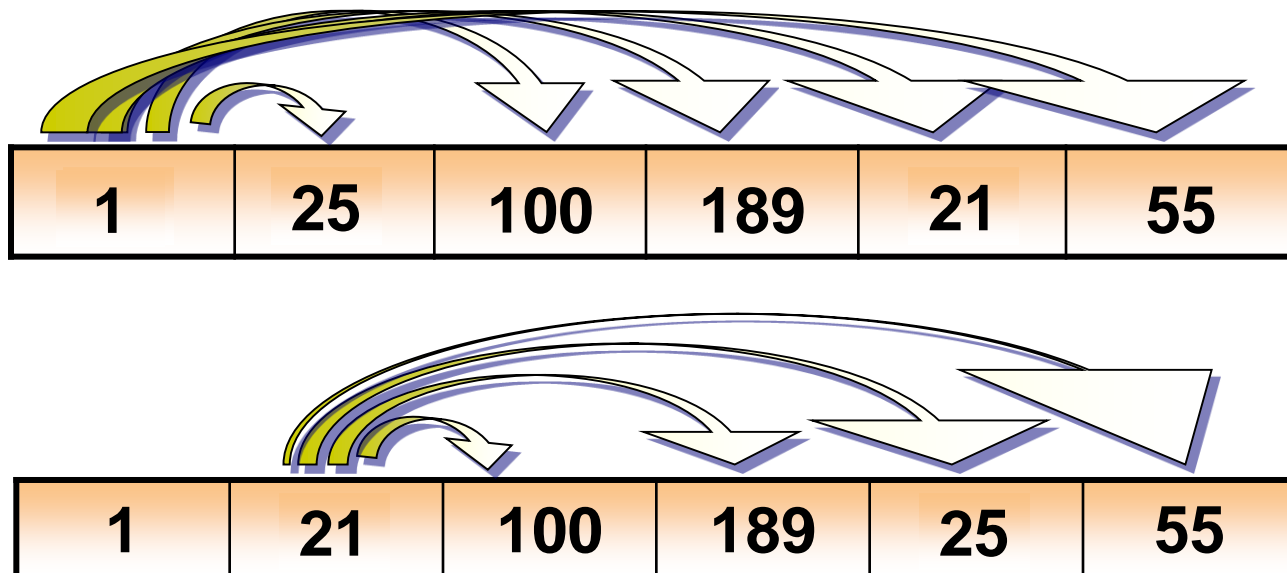
```
}
```

重复  
shz.length-1  
遍



## 5、数组的选择排序

数组中选择排序算法  
比冒泡排序高效





## 5、数组的选择排序

演示完整代码

```
int[ ] shz = {98,32,21,19,61,55};  
for(int i=0;i<shz.length-1;i++){  
    for (int j=i+1;j<shz.length;j++){  
        if (shz[j]<shz[i]){  
            int t = shz[i];  
            shz[i] = shz[j];  
            shz[j] = t ;    // 后一个元素小于当前元素时, 交换值  
        }  
    }  
}
```

循环嵌套（双重循环）  
设置擂主，共设置  
shz.length-1遍

每个擂主与其  
余元素比较