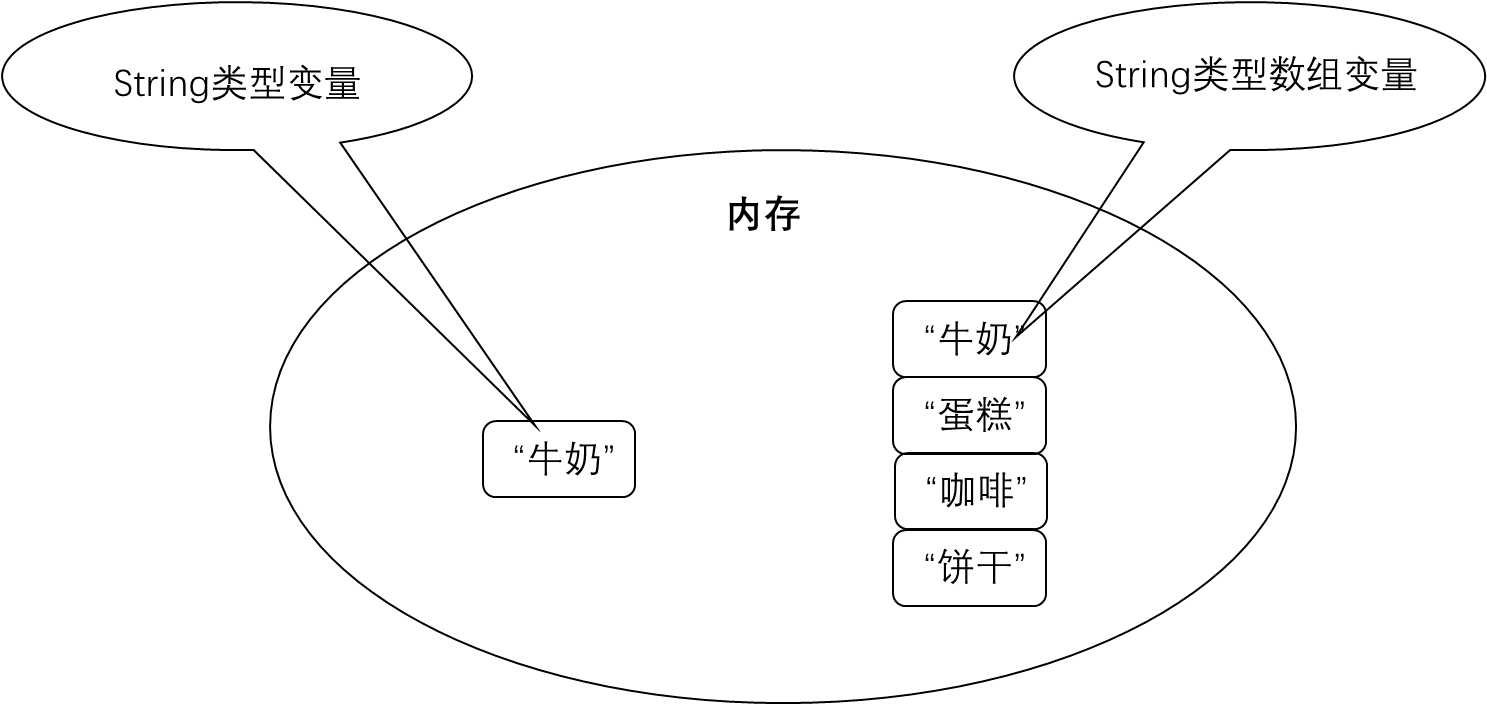
**数组**

* 数组是**相同类型数据**的有序集合



声明一个变量就是在内存空间划出**一块**合适的空间

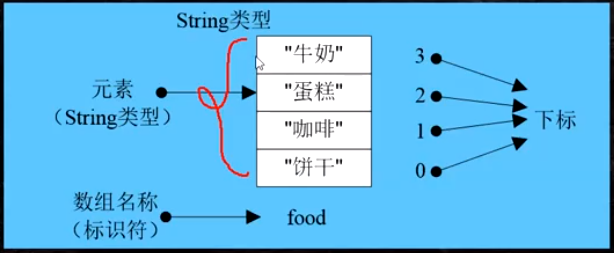
声明一个数组就是在内存空间划出**一串**连续空间

-相同类型的若干个数据，按照一定先后次序（不是指大小排序，而是插入顺序）排列组合而成

-其中，每一个数据称作一个数组元素

-每个数组元素可以通过一个下标(即索引值)来访问它们，从0开始

-数组长度固定不变，避免数组越界



如这里我定义的数组只能放下4个元素，想放第5个是放不进去的。这时会抛异常：ArrayIndexOutOfBoundsException:Index 5 out of bounds for length 5

* 数组特点：

-其长度是确定的。数组一旦被创建，它的大小就是不可以改变的

-数组一旦设定好长度，里面的**元素是不允许增减的**，只能实现对元素的重新赋值

-其元素必须是相同类型，不允许出现混合类型

-数组中的**元素可以是任何数据类型**，包括基本类型和引用类型

* 数组属于引用类型

-length, elements of the arry

练习

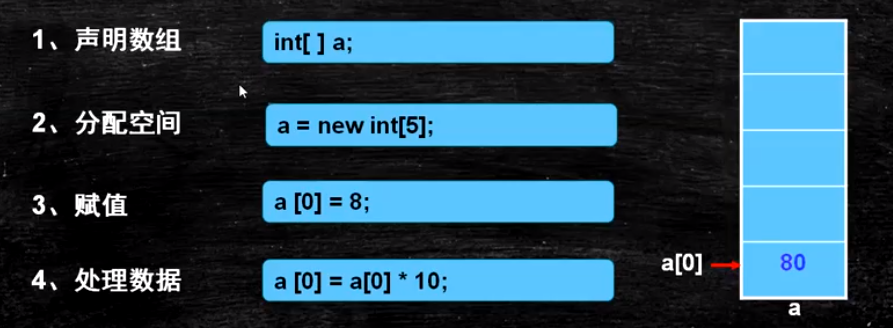
下列哪组数据能存储在数组中？数组的类型是什么？

-“王俊凯”，“易烊千玺”,“王源” 可以，string类型

-8，98，“c”，23 不可以，int类型和string类型

-98.1，341.2，34.3 可以，double类型

* 使用数组四步走：



**//声明数组  
int**[] array;  
**//创建内存空间**array = **new int**[5];  
**//数组赋值**array[0]=0;  
array[1]=1;  
array[2]=2;  
array[3]=3;  
array[4]=4;  
**//数组操作**System.***out***.println(array[0]);

* 数组的几种创建方式：

**//声明并申请空间（下面的三种写法都可以）  
int**[] array1 = **new int**[5];  
**int** [] array2 = **new int**[5];  
**int** array3[] = **new int**[5];  
**//声明数组并赋值（赋值的同时是也就把它的空间申请出来了）  
int**[] array4 = **new int**[]{1,2,3,4,5};  
**//直接初始化操作  
int**[] array5 = {1,2,3,4,5};

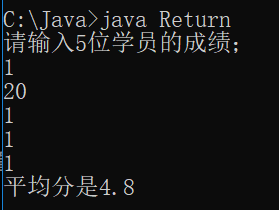
* 获取数组的长度：array.length

比如现在声明了一个长度为5的数组，但是并没有给它赋值，这时array.length显示为5

* 使用数组求平均分：

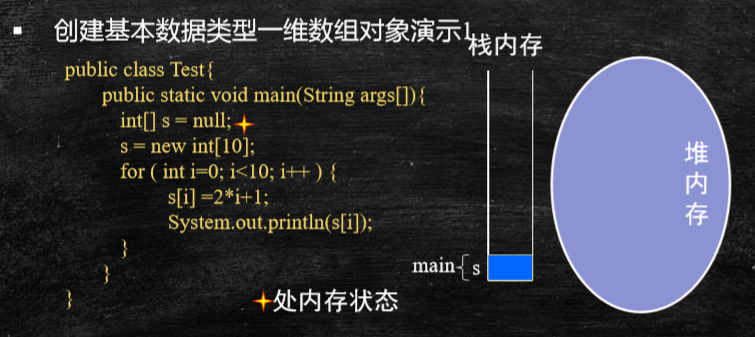
**import** java.util.Scanner;  
**public class** Number {  
 **public static void** main(String[] args) {  
 **int**[] scores = **new int**[5];  
 **int** sum = 0;  
 Scanner input = **new** Scanner(System.***in***);  
 System.***out***.println(**"请输入5位学员的成绩："**);  
 **for**(**int** i=0;i<scores.**length**;i++){  
 scores[i]=input.nextInt();  
 sum = sum + scores[i];  
 }  
 System.***out***.println(**"平均分是"**+(**double**)sum/scores.**length**);  
 }  
}

在输入5名学员的成绩的时候，它们之间可以用空格，tab或者Enter做分隔，系统可以自动以它们为分隔符辨认现在已经有几个元素，并判断什么时候可以输出结果。比如我现在以Enter键做分隔，显示的效果如下

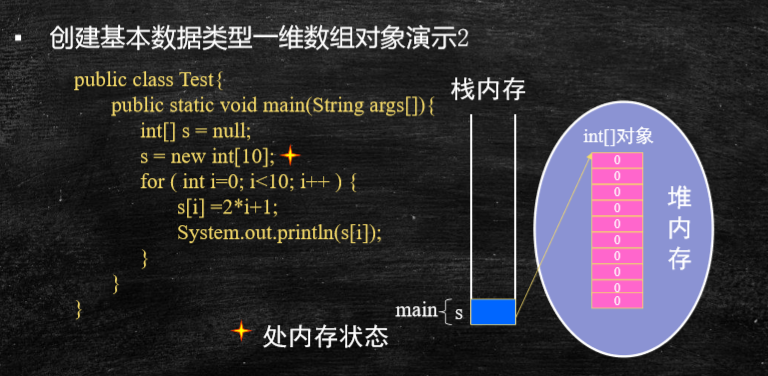


* 创建数组的内存过程演示：

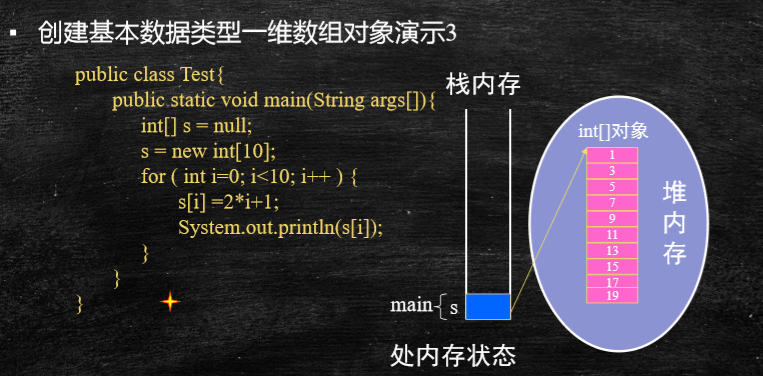
栈里面存的是变量（不管是普通类型还是引用类型）



先定义一个数组，并把它放到栈内存中



new以后就在堆内存中开辟了一个空间用来装数组里的元素



给数组赋值，并且将这个数组在堆内存的首地址存储在栈内存中以供以后调用数组（元素）

* 数组初始化：
* 动态初始化

数组定义与为数组元素分配空间并赋值的操作分开进行

int a[] = null;

a = new int[3];

a[0] = 3;

a[1] = 5;

a[2] = 2;

* 静态初始化

除了用new关键字来产生数组以外，还可以直接在定义数组的同时就为数组元素分配空间并赋值

格式： 类型 [ ] 数组名 = {元素2[，元素2 ……]};

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

* 数组元素的默认初始化

数组是引用类型，它的元素相当于类的实例变量，因此数组一经分配空间，其中的每个元素也被按照实例变量同样的方式被隐式初始化。即当创建完成数组之后相当于在方法外定义了一个变量，此时数组中的值是有默认值的。默认值是什么取决于定义的数组的类型。

int 的默认类型：0

String的默认类型：null

布尔的默认类型：false

想要在方法里面定义一个值必须要进行初始化，例如：

int i;

System.out.println(i);

会报错，说尚未初始化变量i

在定义数组的时候，不用非要进行初始化，例如：

int[ ] array = new int[5];

System.out.println(array[0]);

显示0（即int类型的默认值）

* 数组的界限

-定义并用运算符new为之分配空间后，才可以引用数组中的每个元素

-数组元素的引用方式：arrayName[index]

index为数组元素下标，可以是整型常量或整型表达式。如a[3], b[i], c[6\*i];

数组元素下标从0开始，长度为n的数组合法下标取值范围：0 ~ n-1

-每个数组都有一个属性length指名它的长度，如：a.length指名数组a的长度（元素个数）

-起点和终点

起点： 数组名[0]

终点： 数组名[length-1]

* 用数组知识输出A~Z：

**char**[] s;  
s = **new char**[26];  
**for**(**int** i=0;i<26;i++){  
 s[i]=(**char**)(**'A'**+i);  
 System.***out***.println(s[i]);  
}

这里s[i] = (char)('A'+i);必须要用(char)，不然会报错：从int转换到char可能会有损失

**排序算法：**

* 数组相当于数据结构的一种实现，很多数据在进行存储的时候需要使用数组

数据结构：

线性表

非线性表

树

图

队列

栈

* 数组经常会用来考排序算法
* 排序算法

1. 冒泡排序

概念：

是一种简单的排序算法，它重复地走访过要排序的数列，一次比较两个元素，如果它们的顺序错误就把它们交换过来。走访数列的工作是重复地进行知道没有再需要交换，也就是说该数列已经排序完成。这个算法的名字由来是因为越小的元素会经过交换慢慢浮到数列的顶端

原理：

-如果元素大小关系不正确，交换这两个数（在本例中a>b）

-比较一对相邻元素（a，b）

-重复步骤1和2，直到我们叨叨数组的末尾（最后一对是第（N-2）和（N-1））

项，因为我们的数组从零开始

-到目前为止，最大的元素将在最后的为止，然后我们将N减少1，并重复步

骤1，直到N=1

方法：

**public class** Number {  
 **public static void** main(String[] args) {  
 **//定义数组  
 int**[] array = **new int**[]{4,1,7,2,6};  
 **//用冒泡排序法将数组进行排序操作，从小到大  
 for** (**int** i=0;i<array.**length**-1;i++){  
 **//数组一共有5个元素，一共需要4轮大循环  
 //但是因为i从0开始，所以要小于array.length-1，即0~3，为4次大循环  
 for**(**int** j=0;j<array.**length**-1-i;j++){  
 **//第1大轮里要经过4小轮，第2大轮里要经过3小轮...第4大轮里要经过1**

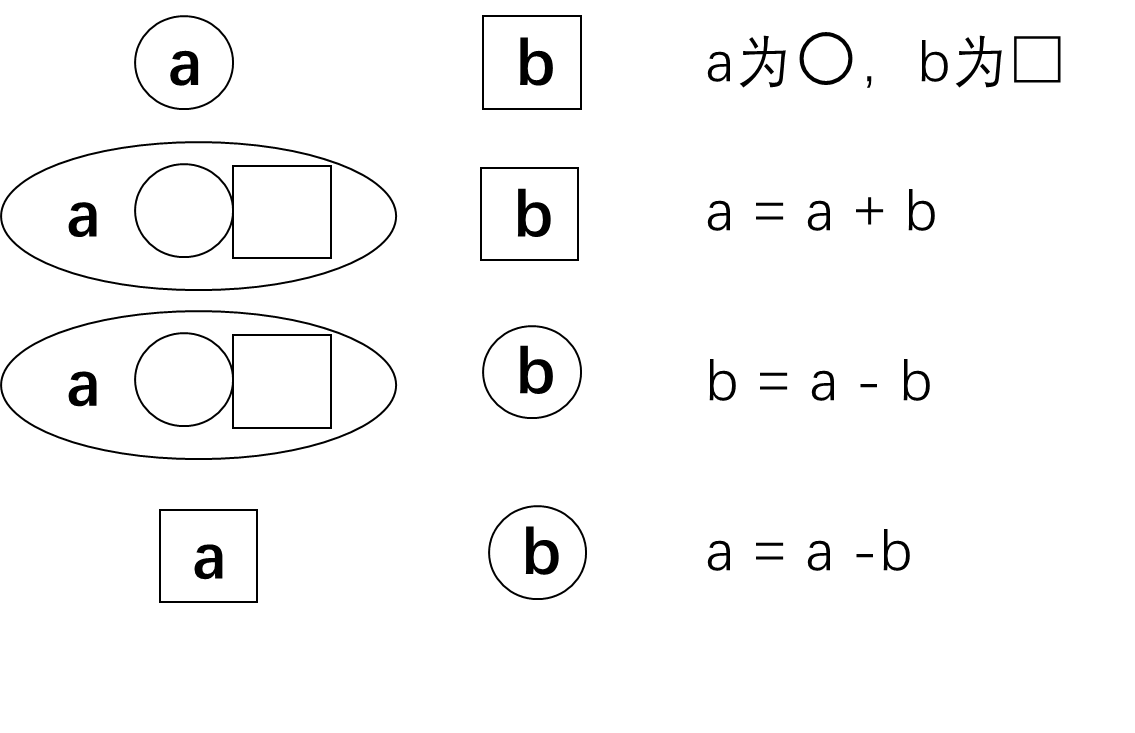
**小轮  
 //由此可见 j = array.length-i  
 //而j也从0开始，因此j<array.length-1-i时都要进行循环  
 if** (array[j]>array[j+1]){  
 **int** tmp = array[j];  
 array[j] = array[j+1];  
 array[j+1] = tmp;  
 **//进行前后两个数的比较，前面的数大于后面的数的时候两个数位置交换** }  
 }  
 }  
 **for** (**int** k=0;k<array.**length**;k++){  
 System.***out***.print(array[k]+**"\t"**);  
 **//在上面的大for循环循环完了以后会得到一个排序好的数组  
 //要将它们打印出来还需要个循环** }  
 }  
}

这里两个数交换的地方如果不用加入第三变量的方法也可以使用下面的方法：

a = a + b

b = a – b

a = a - b



1. 选择排序

概念：

一种简单直观的排序算法，工作机制：首先在未排序序列中找到最小（大）元素，存放到排序序列的起始位置，然后再从剩余未排序元素中继续寻找最小（大）元素，然后放到已排序序列的末尾。以此类推，直到所有元素均排序完毕

原理：

1. 在[L…N-1]范围内找出最小项目X的位置
2. 用第L项交换X
3. 将下限L增加1并重复步骤1直到L=N-2

方法：

**public class** Number {  
 **public static void** main(String[] args) {  
 **int**[] array = **new int**[]{4,1,7,2,6};  
 **for** (**int** i=0;i<array.**length**;i++){  
 **for**(**int** j=i+1;j<array.**length**;j++){  
 **if** (array[j]<array[i]){  
 **int** tmp = array[j];  
 array[j] = array[i];  
 array[i] = tmp;  
 }  
 }  
 }  
 **for** (**int** k=0;k<array.**length**;k++){  
 System.***out***.print(array[k]+**"\t"**);  
 }  
 }  
}

3.插入排序

4.快速排序

* 排序算法的时间复杂度，空间复杂度

衡量一个数据结构是否合适的衡量标准

* 排序算法的稳定性

排序之后的值跟排序之前的值位置是否发生变化

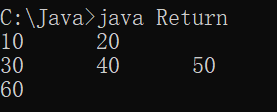
**二维数组**

可以称作数组的数组

定义二维数组的时候一定要注意，必须要给定数组的长度

**public class** Number {  
 **public static void** main(String[] args) {  
 **//3这个数字的地方一定要有，表示占用多大的大格子  
 int**[][] arr = **new int**[3][];  
  
 **//创建二维数组的对象，给后面的[]定义长度：** arr[0] = **new int**[2];  
 arr[1] = **new int**[3];  
 arr[2] = **new int**[1];  
 **//相当于我占了一个有3个格子的空间，  
 //其中这3个格子每个格子又分别对应2个，3个，1个小格子  
  
 //赋值** arr[0][0] = 10;  
 arr[0][1] = 20;  
 arr[1][0] = 30;  
 arr[1][1] = 40;  
 arr[1][2] = 50;  
 arr[2][0] = 60;  
  
 **for**(**int** i = 0;i<arr.**length**;i++){  
 **for** (**int** j=0;j<arr[i].**length**;j++){  
 System.***out***.print(arr[i][j]+**"\t"**);  
 }  
 System.***out***.println();  
 }  
 }  
}

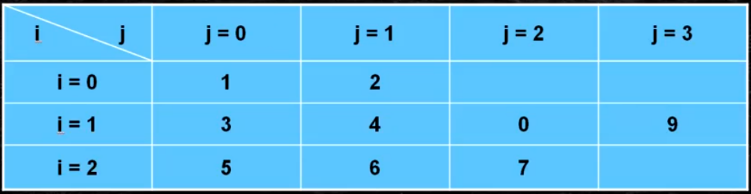
执行结果：



举例：

int [ ][ ] a = {{1,2},(3,4,0,9),{5,6,7}};

Java中多维数组不必须是规则矩阵形式



**Java.util.Arrays**

该类提供了关于数组操作的API

**API**是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件的以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节

-打印数组 toString方法

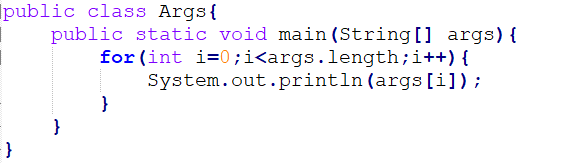
-比较两个数组是否相同 equals方法

-数组查找 binarySearch方法

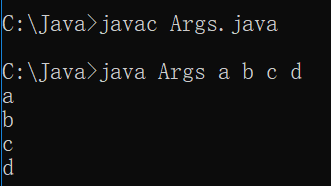
-数组排序 sort方法（从小到大排序）

**import** java.util.Arrays;  
**public class** Number {  
 **public static void** main(String[] args) {  
 **int**[] number = **new int**[]{3,4,5,2,1};  
 Arrays.*sort*(number);  
 **for**(**int** i=0;i<number.**length**;i++){  
 System.***out***.print(number[i]+**"\t"**);  
 }  
 }  
}

**命令行参数用法**



执行结果：



命令行输入java Args a b c d，用空格分隔，就会将输入的这些打印出来

这是因为main方法里定义了一个字符串的数组args，当输入完以后就会放到这个数组里，

然后用下标就可以找到每个元素的位置，用打印方法将其打印出来