第04天 java基础语法

今日内容介绍

* Random
* 数组

# Random

## 产生整数随机数

### Random的使用步骤

我们想产生1~100(包含1和100)的随机数该怎么办呢? 我们不需要自己去写算法,因为Java已经为我们提供好了产生随机数的类---Random:

 作用：

* + 用于产生一个随机数
* 使用步骤(和Scanner类似)
  + 导包
    - import java.util.Random;
  + 创建对象
    - Random r = new Random();
  + 获取随机数
    - int number = r.nextInt(10);
    - 产生的数据在0到10之间，包括0，不包括10。
    - 括号里面的10是可以变化的，如果是100，就是0-100之间的数据

### 案例代码一:

package com.itheima;

import java.util.Random;

/\*

\* Random:用于产生随机数

\*

\* 使用步骤：

\* A:导包

\* import java.util.Random

\* B:创建对象

\* Random r = new Random();

\* C:获取随机数

\* int number = r.nextInt(10);

\* 获取的是0-10之间的随机数，包括0，不包括10

\*

\* 需求：如何获取到一个1-100之间的随机数呢?

\*/

public class RandomDemo {

public static void main(String[] args) {

// 创建对象

Random r = new Random();

for (int x = 0; x < 10; x++) {

// 获取随机数

int number = r.nextInt(10);

// 输出随机数

System.out.println("number:" + number);

}

System.out.println("--------------------");

// 如何获取到一个1-100之间的随机数呢?

int i = r.nextInt(100) + 1;

System.out.println("i:" + i);

}

}

## 猜数字小游戏案例:

### 系统产生一个1-100之间的随机数，请猜出这个数据是多少。

### 案例代码二:

package com.itheima;

import java.util.Random;

import java.util.Scanner;

/\*

\* 猜数字小游戏案例

\* 系统产生一个1-100之间的随机数，请猜出这个数据是多少。

\* 分析：

\* A:系统产生一个随机数1-100之间的。

\* int number = r.nextInt(100) + 1;

\* B:键盘录入我们要猜的数据

\* 用Scanner实现

\* C:比较这两个数据(用if语句)

\* 大了：给出提示大了

\* 小了：给出提示小了

\* 猜中了：给出提示，恭喜你，猜中了

\* D:多次猜数据，而我们不知道要猜多少次，怎么办呢?

\* while(true) {循环的内容}

\*/

public class RandomTest {

public static void main(String[] args) {

// 系统产生一个随机数1-100之间的。

Random r = new Random();

int number = r.nextInt(100) + 1;

while(true){

// 键盘录入我们要猜的数据

Scanner sc = new Scanner(System.in);

System.out.println("请输入你要猜的数字(1-100)：");

int guessNumber = sc.nextInt();

// 比较这两个数据(用if语句)

if (guessNumber > number) {

System.out.println("你猜的数据" + guessNumber + "大了");

} else if (guessNumber < number) {

System.out.println("你猜的数据" + guessNumber + "小了");

} else {

System.out.println("恭喜你,猜中了");

break;

}

}

}

}

# 数组

## 数组概述

需求：现在需要统计某公司员工的工资情况，例如计算平均工资、找到最高工资等。假设该公司有80名员工，用前面所学的知识，程序首先需要声明80个变量来分别记住每位员工的工资，然后在进行操作，这样做会显得很麻烦。为了解决这种问题，Java就提供了数组供我们使用。

那么数组到底是什么呢?有什么特点呢?通过上面的分析：我们可以得到如下两句话：

数组是存储多个变量(元素)的东西(容器)

这多个变量的数据类型要一致

## 数组的定义格式

### 数组概念

数组是存储同一种数据类型多个元素的容器。

数组既可以存储基本数据类型，也可以存储引用数据类型。

### 数组的定义格式

格式1：数据类型[] 数组名;

格式2：数据类型 数组名[];

注意：这两种定义做完了，数组中是没有元素值的。

## 数组的初始化

### 数组初始化概述：

Java中的数组必须先初始化,然后才能使用。

所谓初始化：就是为数组中的数组元素分配内存空间，并为每个数组元素赋值。

### 数组的初始化方式

#### 动态初始化:初始化时只指定数组长度，由系统为数组分配初始值

格式：数据类型[] 数组名 = new 数据类型[数组长度];

数组长度其实就是数组中元素的个数。

举例：

int[] arr = new int[3];

解释：定义了一个int类型的数组，这个数组中可以存放3个int类型的值。

#### 案例代码三:

package com.itheima\_01;

/\*

\* 数组：存储同一种数据类型的多个元素的容器。

\*

\* 定义格式：

\* A:数据类型[] 数组名;

\* B:数据类型 数组名[];

\* 举例：

\* A:int[] a; 定义一个int类型的数组，数组名是a

\* B:int a[]; 定义一个int类型的变量，变量名是a数组

\*

\* 数组初始化：

\* A:所谓初始化，就是为数组开辟内存空间，并为数组中的每个元素赋予初始值

\* B:我们有两种方式对数组进行初始化

\* a:动态初始化 只指定长度，由系统给出初始化值

\* b:静态初始化 给出初始化值，由系统决定长度

\*

\* 动态初始化：

\* 数据类型[] 数组名 = new 数据类型[数组长度];

\*/

public class ArrayDemo {

public static void main(String[] args) {

//数据类型[] 数组名 = new 数据类型[数组长度];

int[] arr = new int[3];

/\*

\* 左边：

\* int:说明数组中的元素的数据类型是int类型

\* []:说明这是一个数组

\* arr:是数组的名称

\* 右边：

\* new:为数组分配内存空间

\* int:说明数组中的元素的数据类型是int类型

\* []:说明这是一个数组

\* 3:数组的长度，其实就是数组中的元素个数

\*/

}

}

#### 静态初始化:初始化时指定每个数组元素的初始值，由系统决定数组长度

#### 案例代码四:

package com.itheima\_01;

/\*

\* 静态初始化的格式：

\* 数据类型[] 数组名 = new 数据类型[]{元素1,元素2,...};

\*

\* 简化格式：

\* 数据类型[] 数组名 = {元素1,元素2,...};

\*

\* 举例：

\* int[] arr = new int[]{1,2,3};

\*

\* 简化后：

\* int[] arr = {1,2,3};

\*/

public class ArrayDemo2 {

public static void main(String[] args) {

//定义数组

int[] arr = {1,2,3};

//输出数组名和元素

System.out.println(arr);

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

System.out.println(arr[1]);

System.out.println(arr[2]);

}

}

## 数组的内存分配

### JVM内存划分

Java 程序在运行时，需要在内存中的分配空间。为了提高运算效率，就对空间进行了不同区域的划分，因为每一片区域都有特定的处理数据方式和内存管理方式。

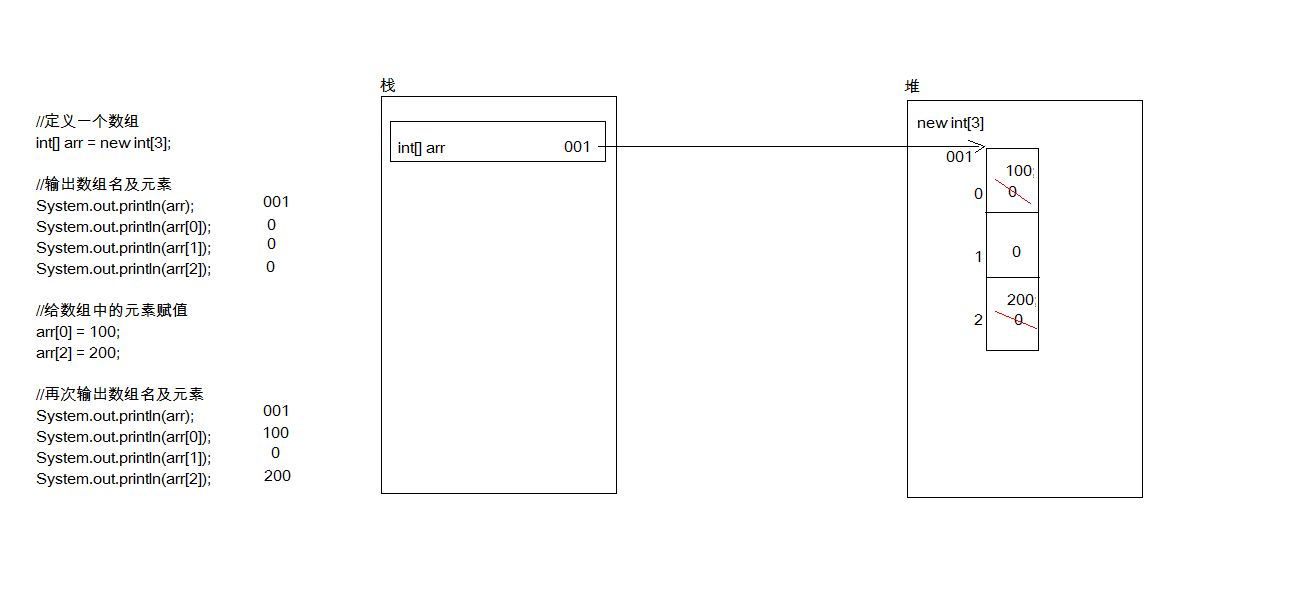
栈 存储局部变量

堆 存储new出来的东西

方法区 (面向对象进阶讲)

本地方法区 (和系统相关)

寄存器 (给CPU使用)



### 一个数组的内存图

定义一个数组，输出数组名及元素。然后给数组中的元素赋值，再次输出数组名及元素

#### 案例代码五:

package com.itheima\_01;

/\*

\* 需求：定义一个数组，输出数组名及元素。然后给数组中的元素赋值，再次输出数组名及元素。

\*/

public class ArrayTest {

public static void main(String[] args) {

//定义一个数组

int[] arr = new int[3];

//输出数组名及元素

System.out.println(arr);

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

System.out.println(arr[1]);

System.out.println(arr[2]);

//给数组中的元素赋值

arr[0] = 100;

arr[2] = 200;

//再次输出数组名及元素

System.out.println(arr);

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

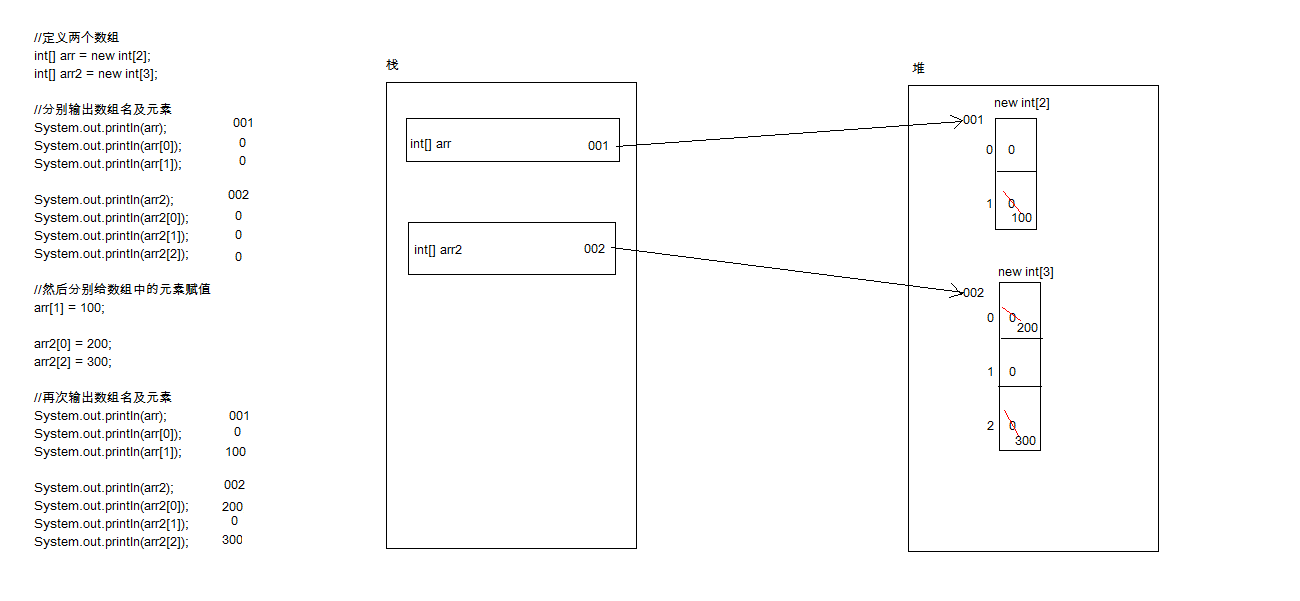
System.out.println(arr[1]);

System.out.println(arr[2]);

}

}

#### 代码内存图解:



### 两个数组的内存图

定义两个数组，分别输出数组名及元素。然后分别给数组中的元素赋值，分别再次输出数组名及元素

#### 案例代码六:

package com.itheima\_01;

/\*

\* 需求：定义两个数组，分别输出数组名及元素。然后分别给数组中的元素赋值，分别再次输出数组名及元素。

\*/

public class ArrayTest2 {

public static void main(String[] args) {

//定义两个数组

int[] arr = new int[2];

int[] arr2 = new int[3];

//分别输出数组名及元素

System.out.println(arr);

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

System.out.println(arr[1]);

System.out.println(arr2);

System.out.println(arr2[0]);

System.out.println(arr2[1]);

System.out.println(arr2[2]);

//然后分别给数组中的元素赋值

arr[1] = 100;

arr2[0] = 200;

arr2[2] = 300;

//再次输出数组名及元素

System.out.println(arr);

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

System.out.println(arr[1]);

System.out.println(arr2);

System.out.println(arr2[0]);

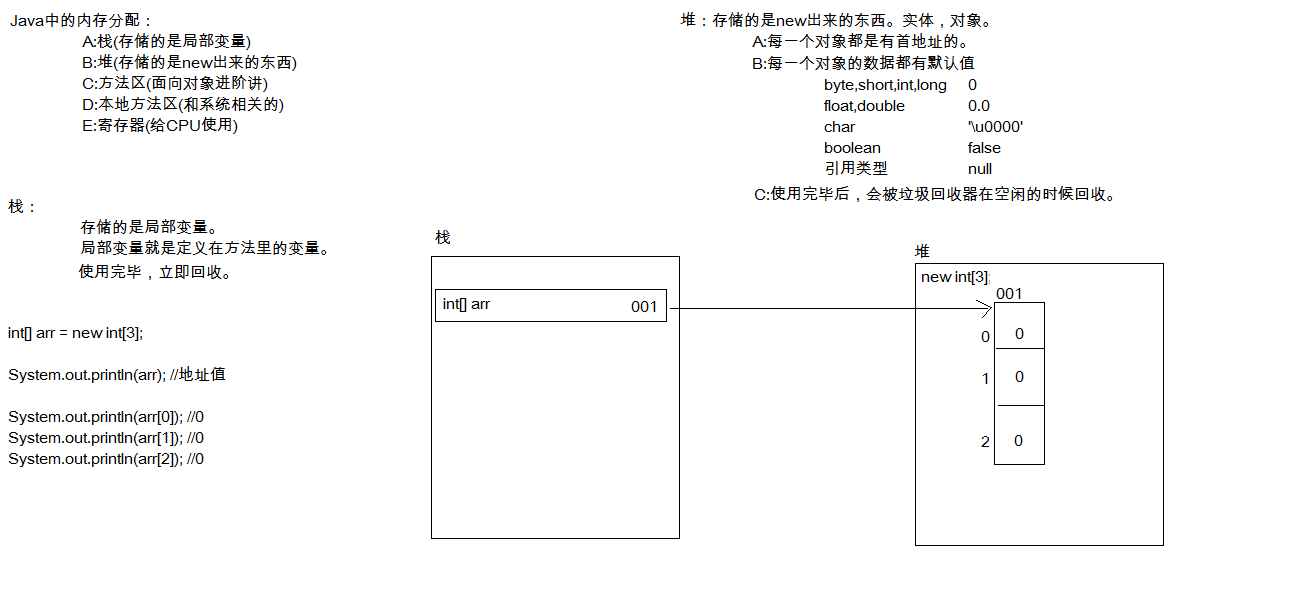
System.out.println(arr2[1]);

System.out.println(arr2[2]);

}

}

#### 代码内存图解:



### 两个数组指向同一个地址的内存图

定义两个数组，先定义一个数组，赋值，输出。然后定义第二个数组的时候把第一个数组的地址赋值给第二个数组。然后给第二个数组赋值，再次输出两个数组的名及元素

#### 案例代码七:

/\*

\* 需求：定义两个数组，先定义一个数组，赋值，输出。然后定义第二个数组的时候把第一个数组的地址赋值给第二个数组。

\* 然后给第二个数组赋值，再次输出两个数组的名及元素。

\*/

public class ArrayTest3 {

public static void main(String[] args) {

// 先定义一个数组，赋值，输出

int[] arr = new int[3];

arr[0] = 100;

arr[1] = 200;

arr[2] = 300;

System.out.println(arr);

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

System.out.println(arr[1]);

System.out.println(arr[2]);

// 然后定义第二个数组的时候把第一个数组的地址赋值给第二个数组

int[] arr2 = arr;

// 然后给第二个数组赋值

arr2[0] = 111;

arr2[1] = 222;

arr2[2] = 333;

// 再次输出两个数组的名及元素

System.out.println(arr);

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

System.out.println(arr[1]);

System.out.println(arr[2]);

System.out.println(arr2);

System.out.println(arr2[0]);

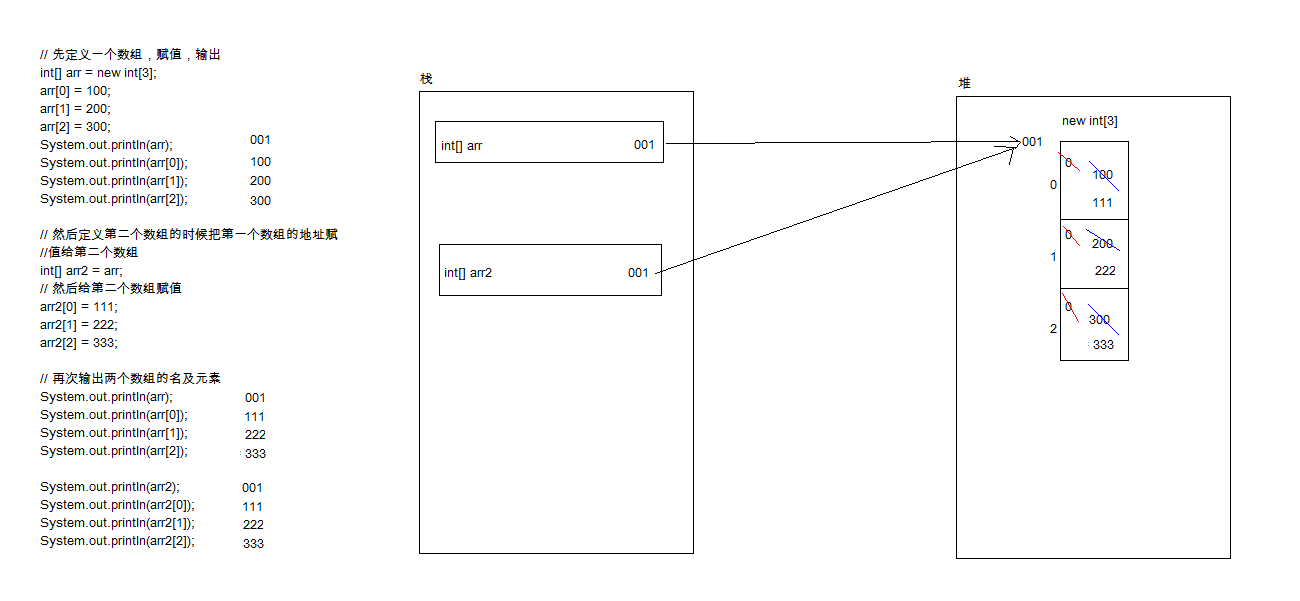
System.out.println(arr2[1]);

System.out.println(arr2[2]);

}

}

#### 代码内存图解:



## 数组的使用

### 数组的元素访问

#### 案例代码八:

package com.itheima\_01;

/\*

\* 数组：存储同一种数据类型的多个元素的容器。

\*

\* 定义格式：

\* A:数据类型[] 数组名;

\* B:数据类型 数组名[];

\* 举例：

\* A:int[] a; 定义一个int类型的数组，数组名是a

\* B:int a[]; 定义一个int类型的变量，变量名是a数组

\*

\* 数组初始化：

\* A:所谓初始化，就是为数组开辟内存空间，并为数组中的每个元素赋予初始值

\* B:我们有两种方式对数组进行初始化

\* a:动态初始化 只指定长度，由系统给出初始化值

\* b:静态初始化 给出初始化值，由系统决定长度

\*

\* 动态初始化：

\* 数据类型[] 数组名 = new 数据类型[数组长度];

\*/

public class ArrayDemo {

public static void main(String[] args) {

//数据类型[] 数组名 = new 数据类型[数组长度];

int[] arr = new int[3];

/\*

\* 左边：

\* int:说明数组中的元素的数据类型是int类型

\* []:说明这是一个数组

\* arr:是数组的名称

\* 右边：

\* new:为数组分配内存空间

\* int:说明数组中的元素的数据类型是int类型

\* []:说明这是一个数组

\* 3:数组的长度，其实就是数组中的元素个数

\*/`

System.out.println(arr); //[I@3fa5ac,地址值

//我们获取到地址值没有意义，我要的是数组中的数据值，该怎么办呢?

//不用担心，java已经帮你想好了

//其实数组中的每个元素都是有编号的，编号是从0开始的，最大的编号就是数组的长度-1

//用数组名和编号的配合我们就可以获取数组中的指定编号的元素

//这个编号的专业叫法：索引

//格式：数组名[编号] -- 数组名[索引]

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

System.out.println(arr[1]);

System.out.println(arr[2]);

}

}

### 数组使用的两个小问题

#### 案例代码九:

package com.itheima\_02;

/\*

\* 两个常见小问题：

\* A:java.lang.ArrayIndexOutOfBoundsException

\* 数组越界异常

\* 产生的原因：就是你访问了不存在的索引元素。

\* B:java.lang.NullPointerException

\* 空指针异常

\* 产生的原因：数组已经不指向堆内存的数据了，你还使用数组名去访问元素。

\* 为什么我们要记住这样的小问题呢?

\* 编程不仅仅是把代码写出来，还得在出现问题的时候能够快速的解决问题。

\*/

public class ArrayDemo {

public static void main(String[] args) {

// 定义数组

int[] arr = { 1, 2, 3 };

//System.out.println(arr[3]);

//引用类型：类,接口,数组

//常量：空常量 null，是可以赋值给引用类型的

//arr = null;

System.out.println(arr[1]);

}

}

## 一维数组练习

### 一维数组遍历

#### 案例代码十:

package com.itheima\_03;

/\*

\* 需求：数组遍历(依次输出数组中的每一个元素)

\* 获取数组中元素的个数：数组名.length

\*/

public class ArrayTest {

public static void main(String[] args) {

// 定义数组

int[] arr = { 11, 22, 33, 44, 55 };

// 原始做法

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

System.out.println(arr[1]);

System.out.println(arr[2]);

System.out.println(arr[3]);

System.out.println(arr[4]);

System.out.println("--------------------");

// 用for循环改进

for (int x = 0; x < 5; x++) {

System.out.println(arr[x]);

}

System.out.println("--------------------");

//为了解决我们去数数组中元素个数的问题，数组就提供了一个属性：length

//用于获取数组的长度

//格式：数组名.length

System.out.println("数组共有："+arr.length+"个");

System.out.println("--------------------");

for(int x=0; x<arr.length; x++) {

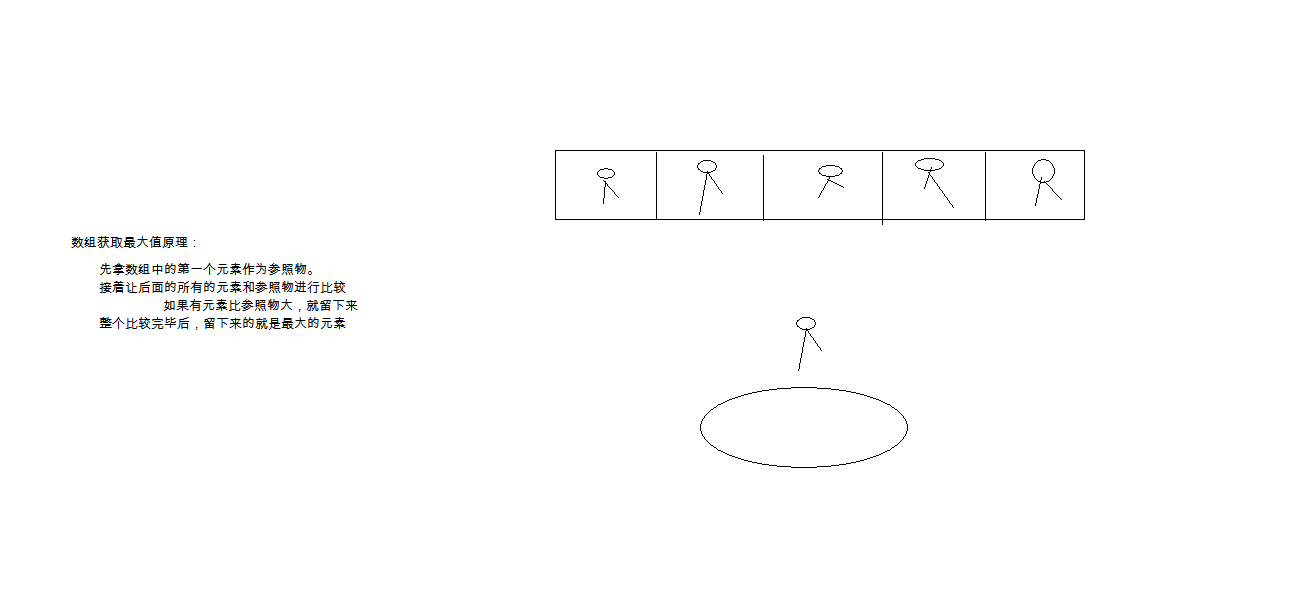
System.out.println(arr[x]);

}

}

}

### 数组操作之获取最值:



#### 案例代码十一:

package com.itheima\_03;

/\*

\* 需求：数组获取最值(获取数组中的最大值最小值)

\*/

public class ArrayTest2 {

public static void main(String[] args) {

//定义数组

int[] arr = {12,98,45,73,60};

//定义参照物

int max = arr[0];

//遍历数组，获取除了0以外的所有元素，进行比较

for(int x=1; x<arr.length; x++) {

if(arr[x] > max) {

max = arr[x];

}

}

System.out.println("数组中的最大值是："+max);

}

}

## 二维数组

### 二维数组概述

我们黑马程序员的Java基础班每个班有很多个学生，所以，可以用数组来存储，而我们又同时有很多个Java基础班。这个也应该用一个数组来存储。如何来表示这样的数据呢?Java就提供了二维数组供我们使用。

由此可见：其实二维数组其实就是一个元素为一维数组的数组。

### 二维数组格式

定义格式

数据类型[][] 数组名;

数据类型 数组名[][]; 不推荐

数据类型[] 数组名[]; 不推荐

初始化方式

数据类型[][] 变量名 = new 数据类型[m][n];

数据类型[][] 变量名 = new 数据类型[][]{{元素…},{元素…},{元素…}};

简化版格式：数据类型[][] 变量名 = {{元素…},{元素…},{元素…}};

#### 案例代码十二:

package com.itheima\_04;

/\*

\* 二维数组：就是元素为一维数组的数组。

\*

\* 定义格式：

\* A:数据类型[][] 数组名;

\* B:数据类型 数组名[][]; 不推荐

\* C:数据类型[] 数组名[]; 不推荐

\*

\* 如何初始化呢?

\* A:动态初始化

\* 数据类型[][] 数组名 = 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的元素有多少个

\* B:静态初始化

\* 数据类型[][] 数组名 = new 数据类型[][]{{元素...},{元素...},{元素...},...};

\* 简化格式：

\* 数据类型[][] 数组名 = {{元素...},{元素...},{元素...},...};

\*/

public class ArrayArrayDemo {

public static void main(String[] args) {

// 数据类型[][] 数组名 = {{元素...},{元素...},{元素...},...};

int[][] arr = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

System.out.println(arr); // [[I@104c575

System.out.println(arr.length); // 二维数组中的一维数组的个数

System.out.println(arr[0]);// [I@3fa5ac

System.out.println(arr[0].length);

System.out.println(arr[1]);// [I@95cfbe

System.out.println(arr[2]);// [I@179dce4

//我如何获取到一个二维数组的元素呢?

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

System.out.println(arr[1][1]);

System.out.println(arr[2][0]);

}

}

### 二维数组的遍历

遍历思想:首先使用循环遍历出二维数组中存储的每个一维数组,然后针对每个遍历到的一维数组在使用循环遍历该一维数组中的元素

#### 案例代码十三:

package com.itheima\_04;

/\*

\* 二维数组遍历

\*

\* System.out.println():输出内容并换行

\* System.out.print():输出内容

\* System.out.println():换行

\*/

public class ArrayArrayTest {

public static void main(String[] args) {

// 定义二维数组

int[][] arr = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

// 二维数组中的一维数组名称：二维数组名[索引]

// arr[0] 其实就是二维数组中的第一个一维数组的名称

// arr[1] 其实就是二维数组中的第二个一维数组的名称

// arr[2] 其实就是二维数组中的第三个一维数组的名称

// for (int x = 0; x < arr[0].length; x++) {

// System.out.println(arr[0][x]);

// }

// System.out.println("hello");

// System.out.println("world");

// System.out.print("hello");

// System.out.print("world");

/\*

// 第一个一维数组的元素

for (int x = 0; x < arr[0].length; x++) {

System.out.print(arr[0][x] + " ");

}

System.out.println();

// 第二个一维数组的元素

for (int x = 0; x < arr[1].length; x++) {

System.out.print(arr[1][x] + " ");

}

System.out.println();

// 第三个一维数组的元素

for (int x = 0; x < arr[2].length; x++) {

System.out.print(arr[2][x] + " ");

}

System.out.println();

\*/

// for(int y=0; y<3; y++) {

// for (int x = 0; x < arr[y].length; x++) {

// System.out.print(arr[y][x] + " ");

// }

// System.out.println();

// }

for(int y=0; y<arr.length; y++) {

for (int x = 0; x < arr[y].length; x++) {

System.out.print(arr[y][x] + " ");

}

System.out.println();

}

}

}