一.数组的定义和访问

数组(array)：就是一种容器，可以同时存放多个数据值。并且具有如下特点：

* 数组是一种引用数据类型
* 数组当中的多个数据，类型必须统一
* 数组的长度在程序运行期间不可改变

1.数组的定义

(1).动态初始化数组：

数据类型[] 数组名称 = new 数据类型[数组长度];

(2).静态初始化的标准格式：

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

(3).静态初始化的省略格式：

数据类型[] 数组名称 = { 元素1, 元素2, ... };

* 数据类型：也就是数组当中保存的数据，全都是统一的什么类型。可以是基本数据类型，也可以是引用数据类型中的String类型
* 数组名字：满足标识符的命名规则和规范(小驼峰式)，可以使用名字操作数组
* new：创建数组使用的关键字，代表创建数组的动作

2.数组的访问

每一个存储到数组的元素，都会自动的拥有一个编号，从0开始，依次为：0，1，2，3......，这个编号称为数组的索引。可以通过数组的索引访问到数组中的元素。具体语法格式如下：（注意：索引值从0开始，一直到 "数组的长度-1" 为止）

数组名称[索引值];

#.注意：

a.动态初始化（指定长度）：直接指定数组当中的数据元素个数，而没有指定具体的数据内容，数据内容可后续进行赋值

静态初始化（指定内容）：不直接指定数据个数多少，而是直接将具体的数据内容进行指定，可自动推算出来数组长度

b.动态初始化可以拆分成为两个步骤：先定义一个数组，再指定数组长度

int arrayA[];

arrayA = new int[10];

静态初始化的标准格式，也可以拆分成为两个步骤：先定义一个数组，再指定数组内容

int arrayB[];

arrayB = new int[]{5, 10, 15, 20};

静态初始化的省略格式，不能拆分成为两个步骤

int arrayC[];

arrayC = {5, 10, 15, 20};

c.直接打印数组名称，得到的是数组对应的：内存地址哈希值。

int[] array = { 10, 20, 30 };

System.out.println(array); // [I@75412c2f：[ 代表数组，I代表int类型，@代表后面的数字为十六进制

d.使用动态初始化数组的时候，数组当中的元素并未被赋值，但是有默认的值存储在数组里面。规则如下：

|  |  |
| --- | --- |
| 整数型(byte，short，int，long) | 0 |
| 浮点数型(float，double) | 0.0 |
| 字符型(char) | '\u0000'（不可见的字符） |
| 布尔型(boolean) | false |
| 引用类型(字符串，数组，类，接口，Lambda) | null |

静态初始化其实也有默认值的过程，只不过系统自动马上将默认值替换成为了大括号当中的具体数值。

e.数组的长度属性

每个数组都具有长度，而且是固定的。Java中赋予了数组一个属性，可以获取到数组的长度，语句为： 数组名.length ，属性length的执行结果是数组的长度，是一个int类型数字。由此可以推断出，数组的最大索引值为：数组名.length - 1

public static void main(String[] args) {

int[] array = {10, 20, 30, 3, 5, 4, 6, 7, 8, 8, 65, 4, 44, 6, 10, 3};

int len = array.length; // 数组的length属性的返回值是一个int类型的数字

System.out.println("数组array的长度是：" + len);

System.out.println("数组array的最大索引是：" + (len - 1) );

}

二.数组原理内存图

1.内存概述

内存是计算机中的重要原件，为计算机的临时存储区域，作用是运行程序。我们编写的程序是存放在硬盘中的，在硬盘中的程序是不会运行的，必须放进内存中才能运行，运行完毕后会清空内存。由于Java虚拟机是一种运行在操作系统上的假想的计算机，所以Java虚拟机也具有内存空间。Java虚拟机要运行程序，必须要对内存进行空间的分配和管理。

2.Java虚拟机的内存划分

为了提高运算效率，JVM就对其内存空间进行了不同区域的划分，每一片区域都有特定的数据处理方式和内存管理方式。Java虚拟机的内存一般划分为以下5个部分：

(1).栈(Stack)：方法运行时使用的内存，方法必须在栈中运行。存放方法的局部变量，局部变量一旦超出其作用域，立刻从栈内存中消失。

(2).堆(Heap)：存储数组或者对象，凡是new出来的东西，都存储在堆内存中。

堆内存里的东西都有一个十六进制的地址值，且堆内存里面的数据都有默认值，规则如下：

|  |  |  |
| --- | --- | --- |
|  | 数据类型 | 默认值 |
| 基本类型 | 整数型（byte，short，int，long） | 0 |
|  | 浮点数型（float，double） | 0.0 |
|  | 字符型（char） | '\u0000'（不可见的字符） |
|  | 布尔型（boolean） | false |
| 引用类型 | 字符串，数组，类，接口，Lambda | null |

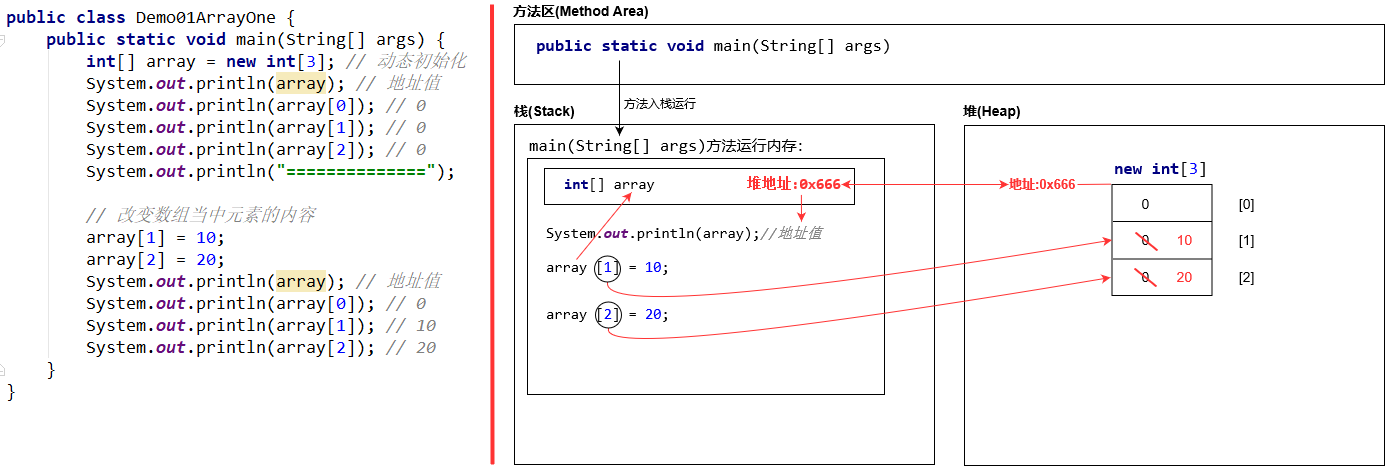
(3).方法区(Method Area)：存放 .class字节码文件，其中包含方法的相关信息。（但方法必须在栈中运行）

(4).本地方法栈(Native Method Stack)：JVM在使用操作系统功能的时候使用，与开发无关。

(5).寄存器(pc Register)：给CPU使用，与开发无关。

3.数组在内存中的存储

(1).一个数组的内存图



①.类Demo01ArrayOne经过编译生成Demo01ArrayOne.class文件，JVM运行该字节码文件时，将这个字节码文件存放在方法区，其中存放着方法的相关信息。

②.方法必须要在栈中才能运行，所以main方法首先进栈执行。此时会在栈内存中开辟一个main方法的运行内存空间。

③.在方法中创建局部变量时，JVM会在main方法运行区中开辟一块空间用来存储变量。

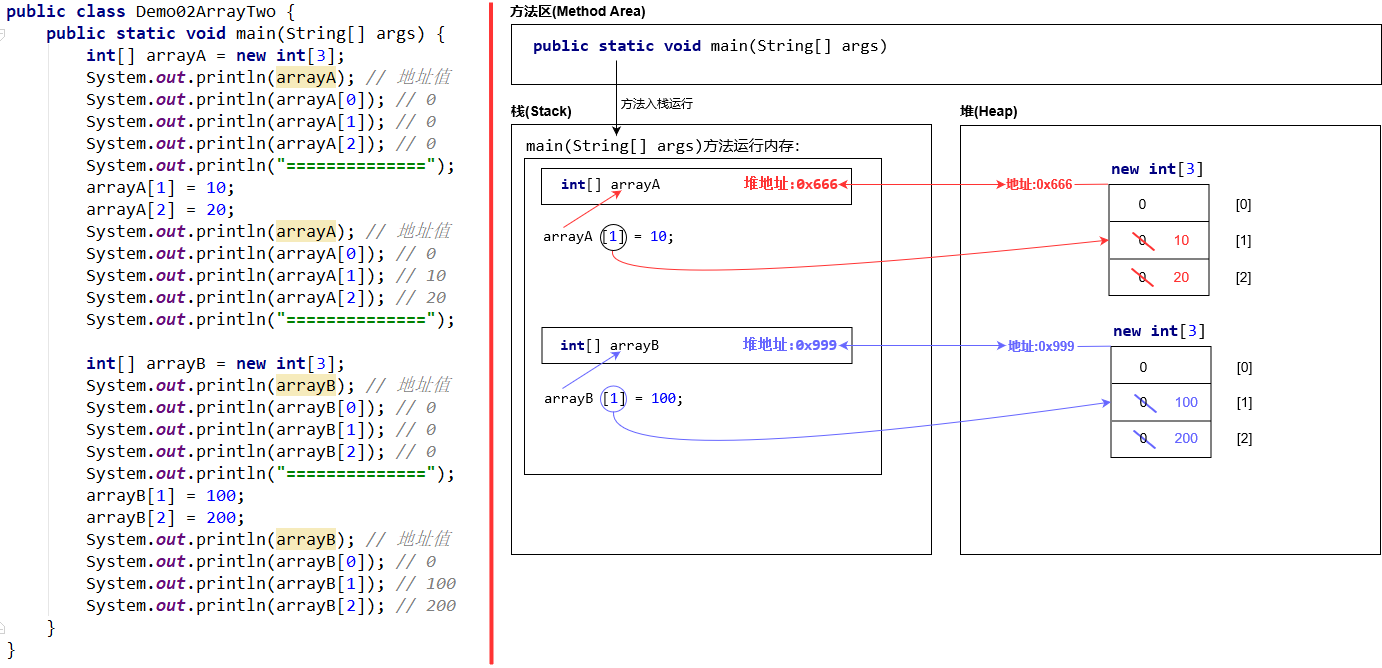
但是数组类型的数据不一样，int[] array 相当于以数组类型 int[] 作为数据类型，创建一个名为array的变量，称为引用类型变量array。首先会在main方法运行区内，为该引用类型变量开辟一块空间。然后new int[3]会使JVM在堆内存中开辟一块空间（根据数组的长度将此空间划分为具有索引的许多块）用来存储数组数据，并将数组的堆内存地址返回给引用类型变量array存储。则 int[] array = new int[3]; 为将堆内存地址赋值给在栈内存中的引用类型变量array。

④.System.out.println(array)；由于引用类型变量array其中存放的是堆地址，故直接打印该引用类型变量输出的是数组在堆内存中的地址。

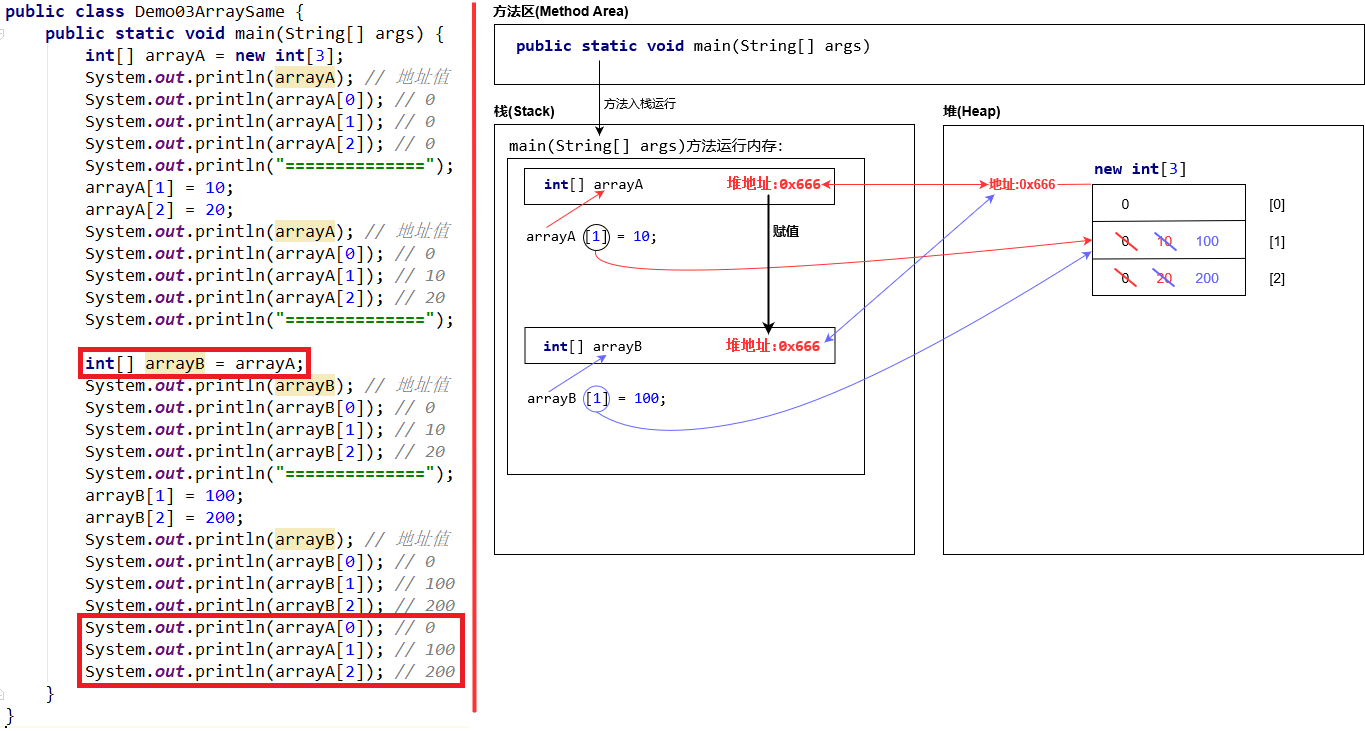
⑤.array[1] = 10；在栈空间中根据数组名找到引用类型变量的内存区，再根据其中存放的地址在堆内存中找到数组，并根据索引值找到数组元素对其进行赋值操作。(System.out.println(array[0])；同理)

#.注意：引用类型变量array的使用与其他变量没有差别，只不过其数据类型为数组类型 int[]，其中存储的是数组的堆地址。每次使用数组array[2]时，都要根据变量array中的地址值，去堆内存中寻址查找数据，相当于使用此变量去引用堆内存中的数据，故称之为引用类型变量，数组也因此称为引用数据类型。

(2).两个数组的内存图

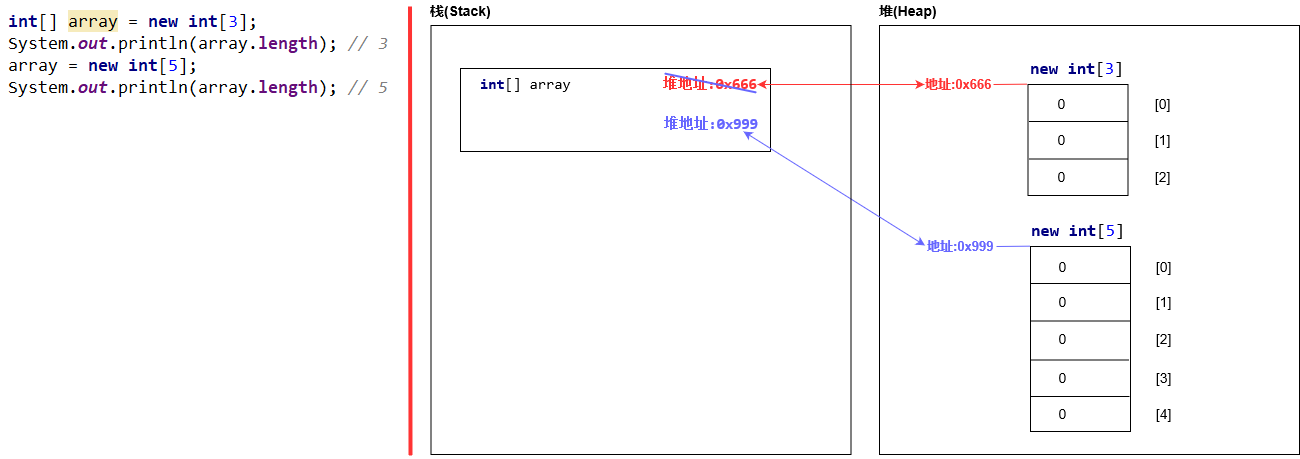


(3).两个引用类型变量指向同一个数组的内存图



将数组arrayA的堆内存地址赋值给另一个引用类型变量arrayB，使两个引用类型变量都指向堆内存中的同一个数组。此时对两个不同数组的操作相当于对同一个数组的操作。

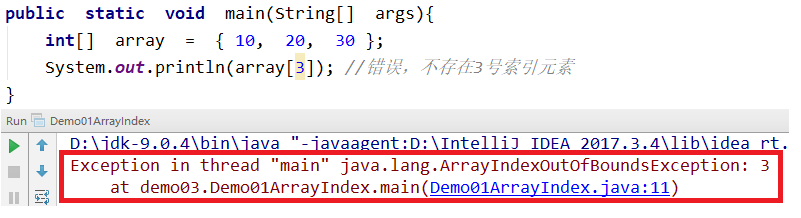
#.利用数组的内存图来理解：数组一旦创建，在程序运行期间，数组的长度不可改变。



由此可见，只要有new int[]就能在堆内存中开辟空间存储数组，并返回数组在堆内存中的地址值。并且数组的长度在程序运行期间不可改变，只有通过改变new int[]中的数字，才能改变数组长度。

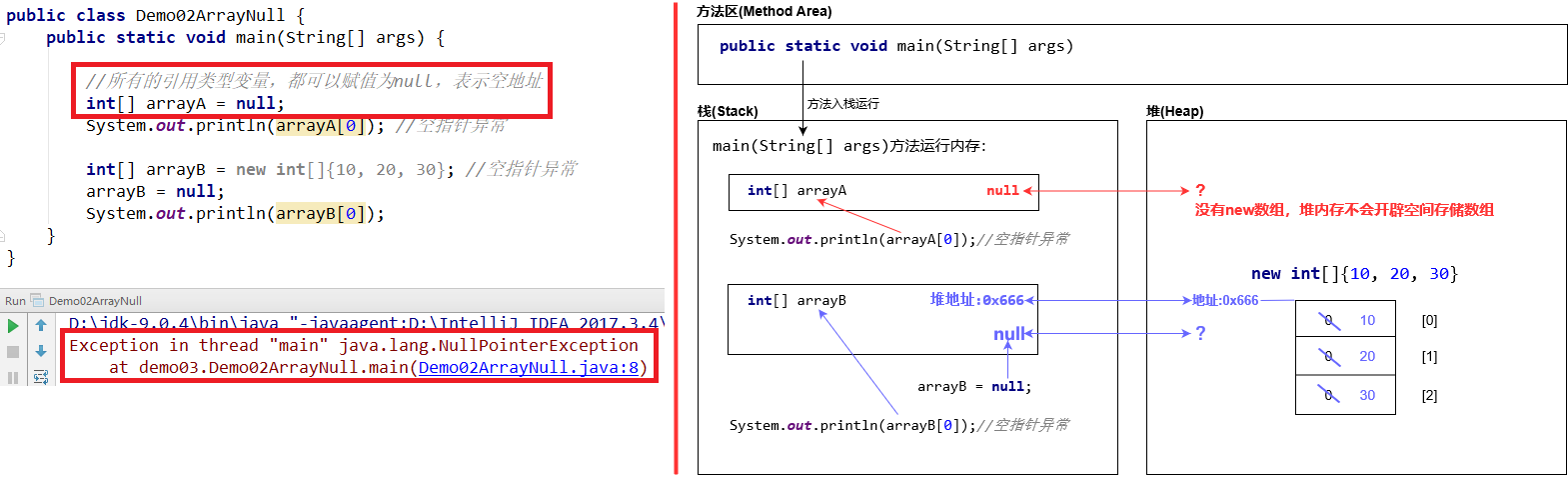
三.数组的常见操作

1.数组索引越界异常



创建含有三个元素的数组array，数组索引就是0，1，2，没有索引3，因此不能访问数组中不存在的索引。程序运行后，将会抛出 java.lang.ArrayIndexOutOfBoundsException 数组索引越界异常。

2.数组空指针异常

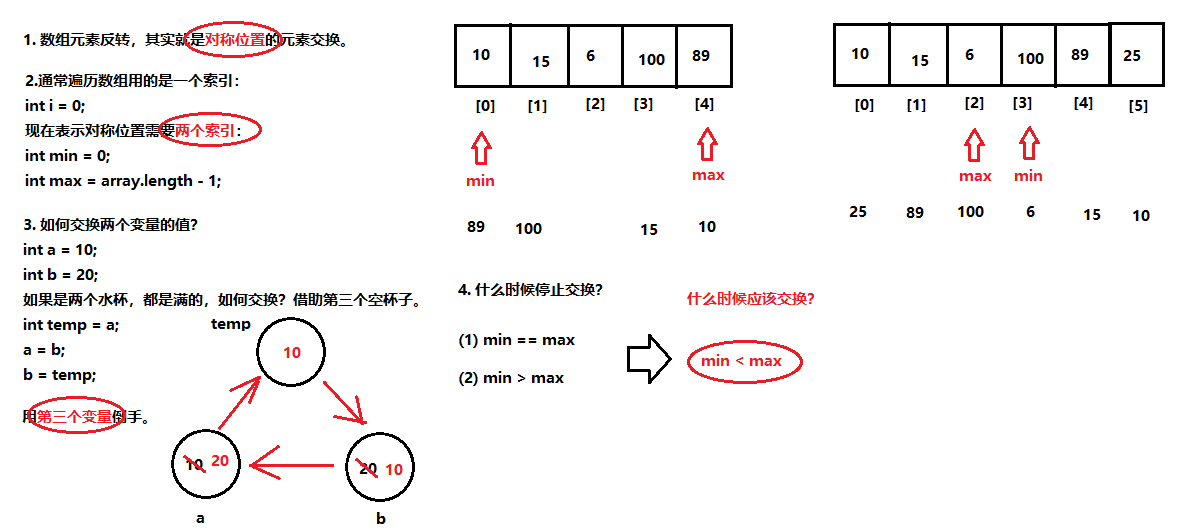


一旦引用类型变量为null，此时我们再对数组数据进行操作，就会导致引用类型变量在堆内存空间中找寻不到地址，此时会抛出 java.lang.NullPointerException 空指针异常。

3.数组的遍历：利用循环将数组中的每个元素依次获取出来，就是数组的遍历。遍历是数组操作中的基石。

4.数组获取最大值、最小值元素

5.数组元素反转：其实就是对称位置的元素交换，具体实现思路如下：



a.借助第三个临时变量temp来交换两个变量的值

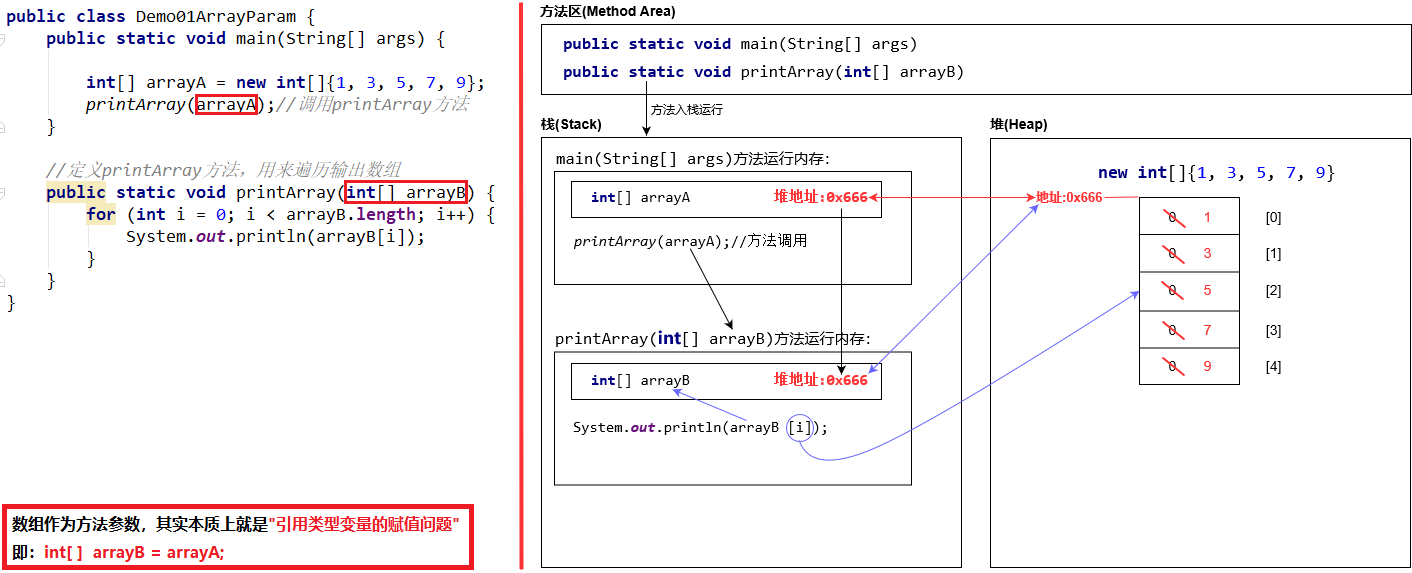
b.当for循环中有多个循环变量时：初始化语句可以同时初始化多个循环变量，步进表达式也可以同时对多个循环变量进行改变

四.数组作为方法的参数和返回值

在以前的学习中，方法的参数和返回值，使用的都是基本数据类型。作为引用数据类型的数组也可作为方法的参数和返回值。数组作为方法的参数和返回值时，其实是以其引用类型变量来作为方法的参数和返回值。即方法的实参、形参和返回值，都是数组类型的引用类型变量，传递与返回的都是数组的堆内存地址。

1.数组作为方法参数\_传递地址

数组作为方法参数进行传递，是以引用类型变量来作为方法的实参和形参，因此传递的是数组的堆内存地址。即实参、形参这两个引用类型变量此时指向同一个数组，因此在方法体内对形参数组的操作，就相当于对实参数组的操作。而对于基本数据类型的实参变量和形参变量，在方法体内改变形参变量并不会影响到实参变量，因为二者在不同的内存区域。



2.数组作为方法返回值\_返回地址

数组作为方法返回值，是以引用类型变量来作为方法的返回值，因此返回的是数组的堆内存地址。同时方法调用处要使用相同数据类型的引用类型变量来接收地址值，即返回值、接收值这两个引用类型变量此时指向同一个数组。

