一、深入数组

（一）定义：   
1、数组是编程语言中最常见的一种数据结构，可用于存储多个数据。   
Java的数组要求所有的数组元素具有相同的数据类型

2、数组是一个引用数据类型，数组的变量只是一个引用，数组元素和数组变量在内存里是分开存放的。

（二）初始化

1、静态初始化：初始化时由程序员显示指定每个数组元素的初始值，由系统决定数组长度。

2、动态初始化：初始化时程序员只指定数组长度，由系统为数组元素分配初始值。

指定初始值时，系统按如下规则分配初始值：

类型 初始值

byte、short、int、long 0

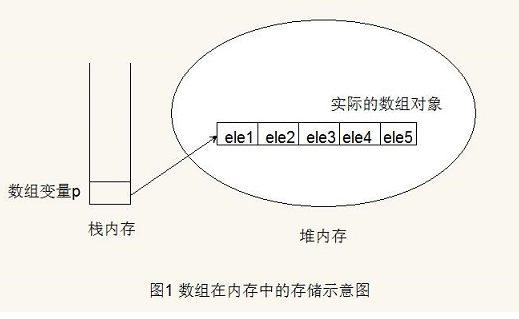
float、double 0.0

char ‘\u0000’

boolean false

类、接口、数组 null

（三）数组在内存中存储

1）定义一个整型数组，并初始化 int[] p = new int[5]; 则：

附注:

《1》栈内存和堆内存

当一个方法执行时，每个方法都会建立自己的内存栈，

在这方法内定义的变量将会逐个放入这块栈内存里，

随着方法的执行结束，这个方法的内存栈也将自然销毁。

所有在方法中定义的局部变量都是存放在栈内存中的；

在程序中创建一个对象时，这个对象将被保存到运行时数据区中，

以便反复利用（因为对象的创建成本通常较大），

这个运行时数据区就是堆内存。

《2》堆内存中的对象不会随方法的结束而销毁，

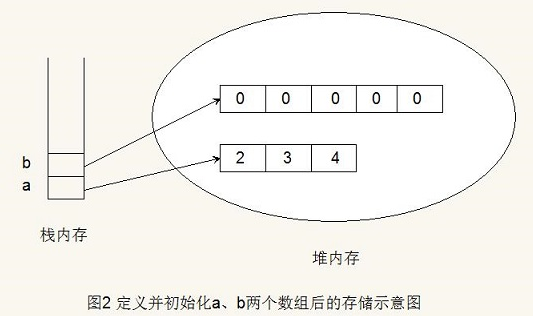
即使方法结束后，这个对象还可能被另一个引用变量所引用（在方法的参数传递时很常见），则这个对象依然不会被销毁。只有当一个对象没有任何引用变量指向它时，系统的垃圾回收器才会在合适的时候回收它。

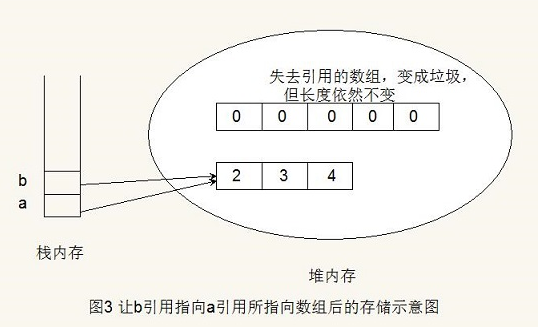
2）定义两个数组a和b,将a的值赋给b，也就是让b引用指向a的引用指向的数组。

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

int[] b = new int[4];

b = a;



当程序定义并初始化了a、b两个数组后，   
系统内存中实际产生了4块内存区，   
其中栈内存中有两个引用变量：a和b;   
堆内存那种也有两块内存区，分别用于存储a和b引用所指向的数组本身。

3）定义Person类

public class Person {

public int age;

public double height;

public void info() {

System.out.println("年龄：" + age + ",身高：" + height);

}

}

测试类

public class ReferenceArrayTest {

public static void main(String[] args) {

// 定义一个students数组变量，其类型是Person[]

Person[] students = new Person[2];

Person zhang = new Person();

zhang.age = 10;

zhang.height = 130;

Person lee = new Person();

lee.age = 20;

lee.height = 180;

//将zhang变量赋值给第一个数组元素

students[0] = zhang;

//将lee变量赋值给第二个数组元素

students[1] = lee;

//下面两行代码结果一样，因为lee和student[1]

//指向的是同一个Person实例

lee.info();

students[1].info();

}

}

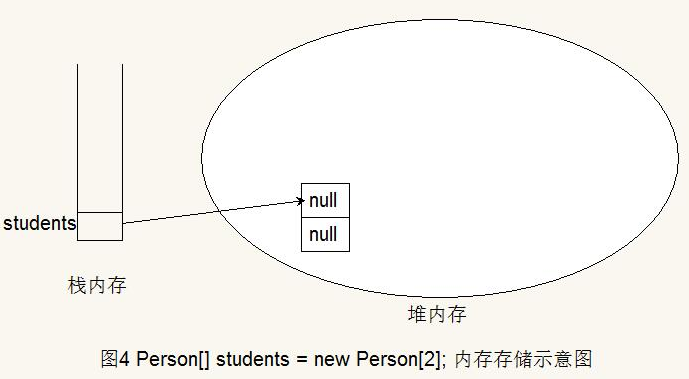


图4可以看出，students数组的两个数组元素都是引用，而且这个引用未指向任何有效的内存，因此这个数组的元素还不能使用！

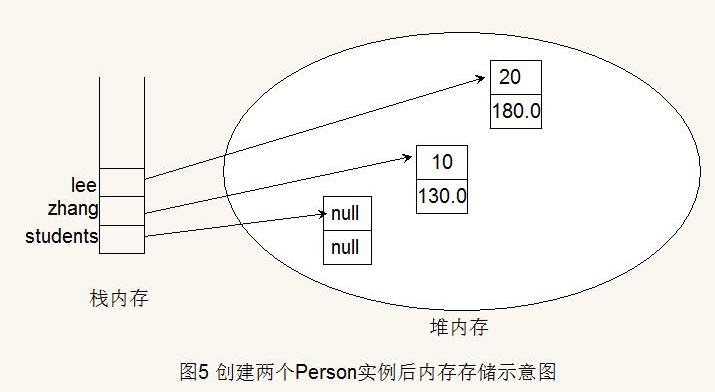
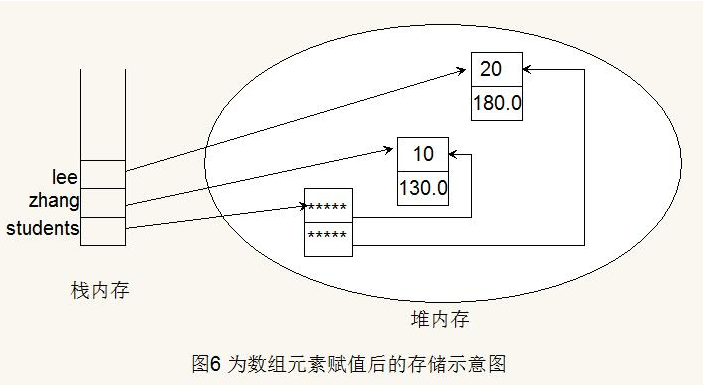


图5可以看出，定义了zhang和lee两个Person实例，



定义这两个实例实际上分配了4块内存，在栈内存中存储了zhang和lee两个引用变量，还在堆内存中存储了两个Person实例。图6可以看出，程序将zhang和lee一次赋给数组元素，则students数组的两个数组元素将会指向有效的内存区。没有多维数组，Java语言里提供了支持多维数组的语法。 如果从数组底层的运行机制上来看，没有多维数组！ Java里数组是引用类型，因此数组变量其实是一个引用， 这个引用指向真实的数组内存，如果数组元素也是引用类型， 指向真正的数组内存，则看起来像多维数组。数组是静态数据结构，即密集表；

举例说明：   
public class TwoDimensionTest {

public static void main(String[] args) {

// 定义一个二维数组

int[][] a;

// 把a当作一维数组进行初始化，初始化a是一个长度为4的数组

// a数组的元素又是引用类型

a = new int[4][];

// 把a当作一维数组进行遍历，遍历a的每个元素

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

System.out.println(a[i]);// null null null null

}

// 初始化a的第一个元素

a[0] = new int[2];

// 访问a数组的第一个元素所指向数组的第二个元素

a[0][1] = 6;

// a数组的第一个元素是一个一维数组，遍历这个一维数组

for (int i = 0; i < a[0].length; i++) {

System.out.println(a[0][i]);

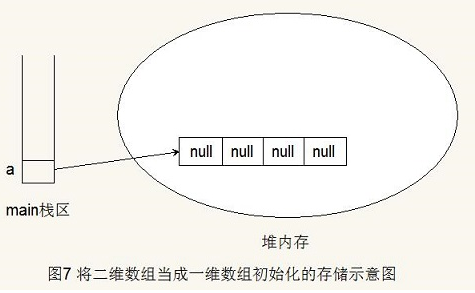
}

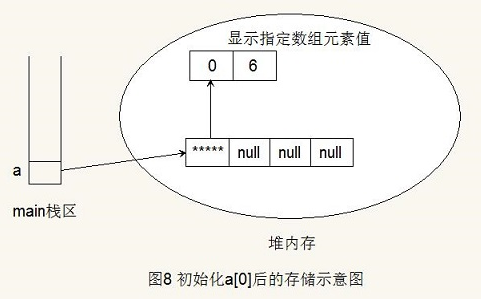
}

}

二维数组的实际定义：

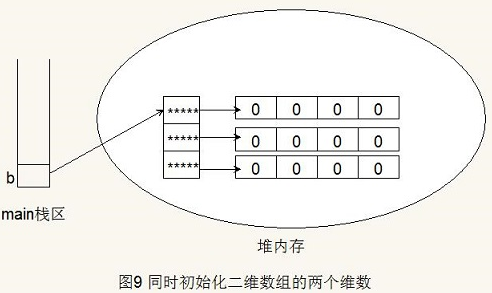
数组类型[][] 数组名 = new 数组类型[一维数组的个数][每一个一维数组中元素的个数];





程序采用动态初始化a[0]数组，   
因此系统将为a[0]所引用数组的每个元素默认分配0，   
程序显示的将a[0]的第二个元素赋值为6。   
int[][] b = new int[3][4];同时初始化二维数组的两个维数。

该代码定义了一个b数组变量，这个数组变量指向一个长度为3的数组，   
这个数组的元素又是一个数组类型，它们各指向对应长度为4的int[]数组，   
每个数组元素的值为0。



结论：二维数组是一维数组，其数组元素是一维数组；   
三位数组也是一维数组，其数组元素是二维数组……从这个角度来看，   
Java语言里没有多维数组！

