Day05 - 数组

今日学习内容：

JVM的内存模型

数组的定义

数组的静态初始化

数组的动态初始化

数组的基本操作

二维数组的操作

今日学习目标：

了解JVM的内存模型中的栈、堆、方法区

必须掌握数组定义的语法

必须掌握数组的静态初始化

必须掌握数组的动态初始化

必须掌握数组的基本操作-获取长度

必须掌握数组的基本操作-获取元素值

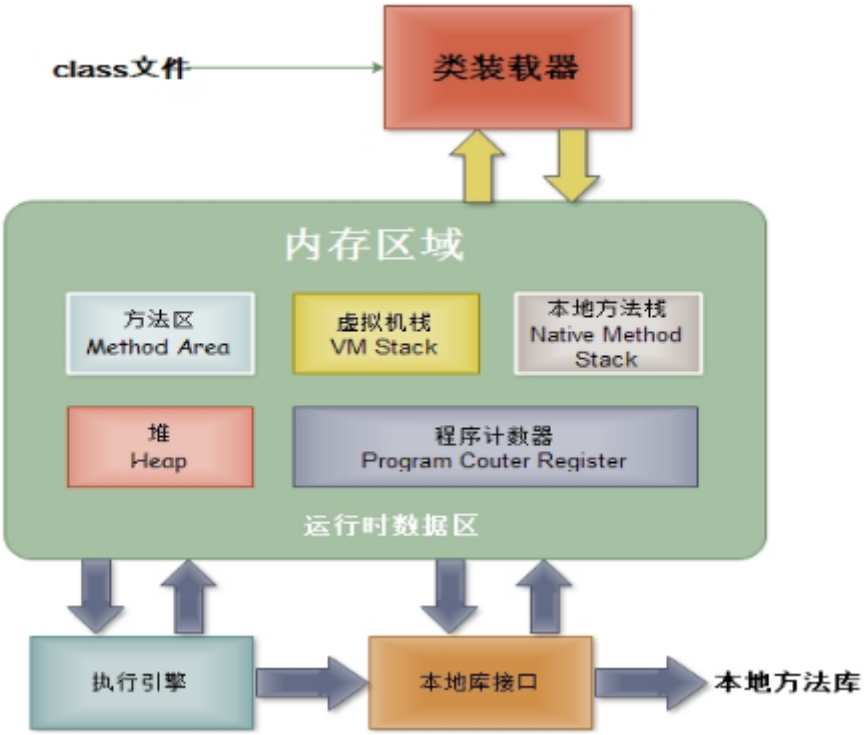
必须掌握数组的基本操作-设置元素值

必须掌握数组的基本操作-遍历元素（for循环和for-each循环）

了解二维数组的定义和初始化操作

第五章、数组

1、JVM内存模型（掌握概念）



JVM内存划分，人为的根据不同内存空间的存储特点以及存储的数据。

- 程序计数器：当前线程所执行的字节码的行号指示器。
- 本地方法栈：为虚拟机使用的native方法服务。
- **方法区**：线程共享的内存区域，**存储已被虚拟机加载的类信息**、常量、静态变量即时编译器编译后的代码数据等（这个区域的内存回收目标主要是针对常量池的回收和对类型的卸载）。
- **Java虚拟机栈**：简称栈区（stack），**每个方法被执行的时候都会同时创建一个栈帧**用于存储该方法的局部**变量**、操作栈、动态链接、方法出口等信息。
  - 每当调用一个方法时，创建一个栈帧（方法执行时的内存），存放了当前方法的局部变量，当方法调用完毕，该方法的栈帧就被销毁了。
  - 一句话：**java方法执行时，在栈区执行**
- **Java堆**：简称堆区(heap),被所有线程共享的一块内存区域，在虚拟机启动时创建。**所有的对象实例以及** `数组` **都要在堆上分配内存。**
  - **每次使用new关键字，就表示在堆内存中开辟一块新的存储空间。**

GC(Garbage Collection)，垃圾回收器。

Java的自动垃圾回收机制可以简单理解为，不需要程序员手动的去控制内存的释放。当JVM内存资源不够用的时候，就会自动地去清理堆中无用对象（没有被引用到的对象）所占用的内存空间。

```
// jvm 内存模型
/**
* 1 方法区：用于存储字节码(xx.class)信息，也即类信息
* 2 栈区：方法执行时在栈区分配内存空间。一句话:java方法执行时，在栈区执行。
* 3 堆区：数组的内存都分配在该区。该区变量有一个标志--new操作符，表示在堆内存中开辟一块新的存储空间。
*/
```

2、数组定义（重点）

```
// 问题:定义一个变量，存储一个学生的年龄?
int age = 17;

// 问题2：如果班级有100人，那么是否需要100个变量呢？
int age1 = 20;
int age2 = 21;
int age3 = 18;
...
```

2.1 什么是数组(了解)

在之前我们可以通过一个变量表示一个学生的年龄，如果现在需要表示全班100个人的年龄岂不是需要定义100个变量来分别表示。这样的操作太麻烦了，为了解决这种问题，Java就提供了数组(Array)（多个数据组合在一起的新的存储形式）

所谓**数组(Array)**，把**具有** 相同类型 **的多个常量值** 有序组织 **起来的一种数据形式。**这些按一定顺序排列的多个数据称为数组。

- 数组中的每一个常量值称之为数组元素(item / element)
- 数组中使用**索引/下标**(index)来表示元素存放的位置，索引从0开始，步长是1，有点像Excel表格的行号。

索引	0	1	2	3	4	5	6	7
元素	50	10	30	40	20	60	70	80

数组在内存中是一段连续的内存空间。

2.2 定义语法（重点）

回忆定义变量的语法：

```
数据类型 变量;
int age;
```

数组是一种新的数据类型，可以用于声明变量(也就申请内存空间)

数组的定义语法：

```
数组元素类型[] 数组名;
例如:
int[] ages;
```

理解：

1. 可以把int[]看成是一种数据类型——int类型的数组类型。
2. int[] 数组可以看出，该数组中的元素类型是int类型的。
3. String[] 数组可以看出，该数组中的元素是String类型的。

3、数组的初始化（重点）

数组在定义后，必须初始化才能使用。所谓初始化，就是在**堆内存**中给数组分配存储空间，并为每一个元素赋上初始值，有两种方式：

- 静态初始化；
- 动态初始化；

数组的长度是固定的，无论以哪种，一旦初始化完成，数组的长度（元素的个数）就固定了，不能改变，除非重新对该初始化。

如果我们事先知道元素是多少，选用静态初始化；
事先不知道元素是多少(但知道需要多少个的空间)，选用动态初始化。

3.1 静态初始化（重点）

程序员直接为每一个数组元素设置初始化值，而数组的长度由系统(JVM)决定。

初始化语法：

```
数组元素类型[]  数组名 = new 数组元素类型[]{元素1,元素2,元素3,.......};
例如：
int[] nums = new  int[]{1,3,5,9,7};

简单写法：
int[] nums = {1,3,5,9,7};// 简单写法,定义和初始化必须同时写出来
```

3.2 静态初始化内存分析（理解即可）

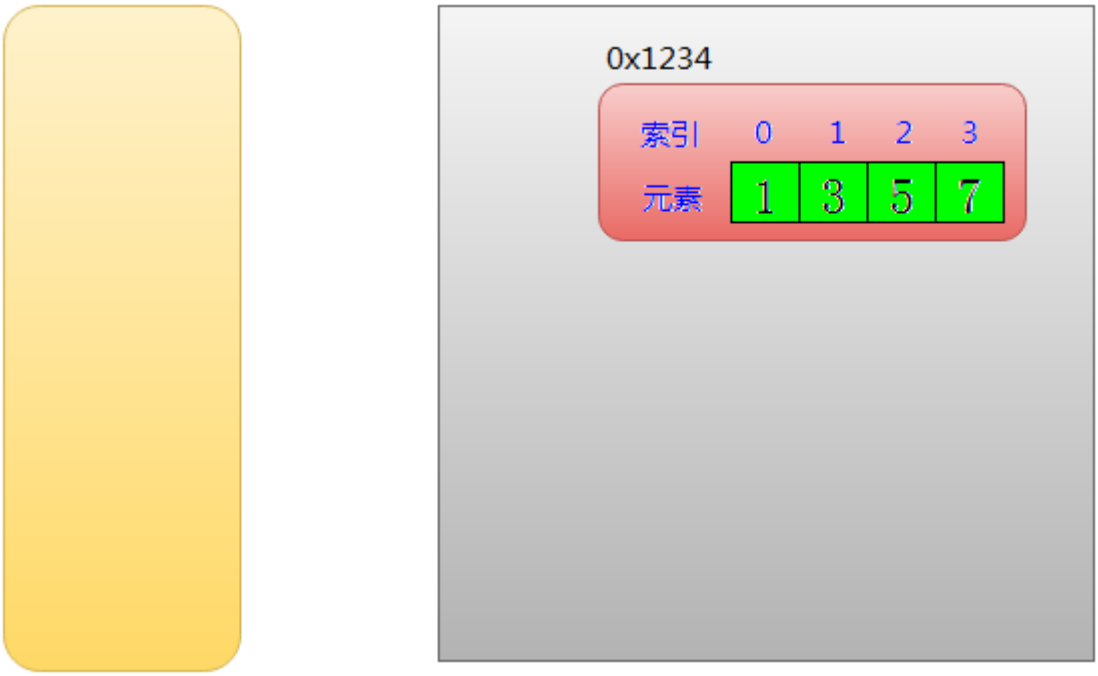
```
public class ArrayDemo1 {
    public static void main(String[] args) {
        // 定义并初始化数组
        int[] nums = new int[] { 1, 3, 5, 7 };
        System.out.println("数组长度=" + nums.length);
        // 重新初始化数组
        nums = new int[] { 2, 4, 8 };
        System.out.println("数组长度=" + nums.length);
    }
}
```

对上述代码做内存分析

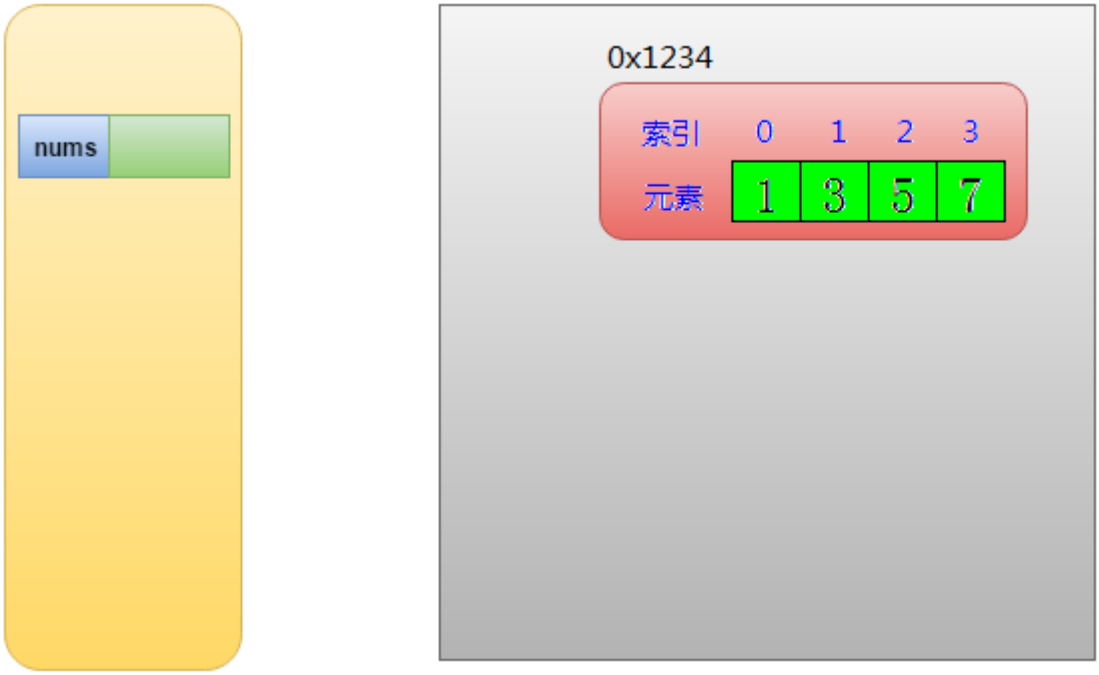
int[] nums = new int[] { 1, 3, 5, 7 };

这一行代码可以分成三步：

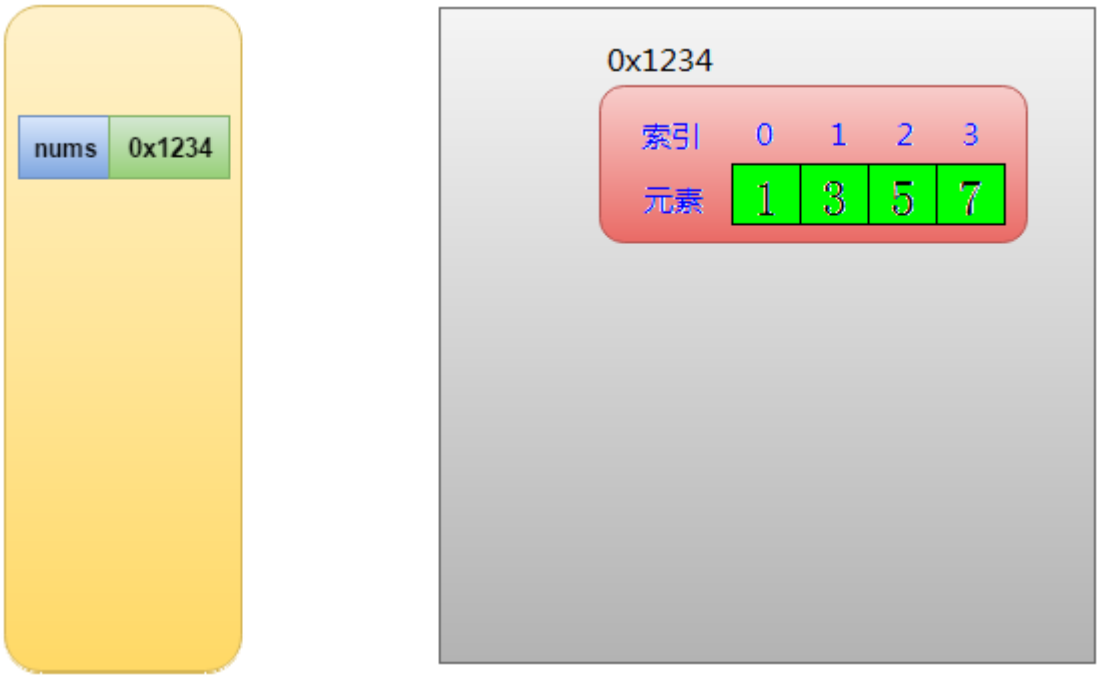
- 在堆内存中开辟一块空间，用来存储数组数据:new int[] { 1, 3, 5, 7 }



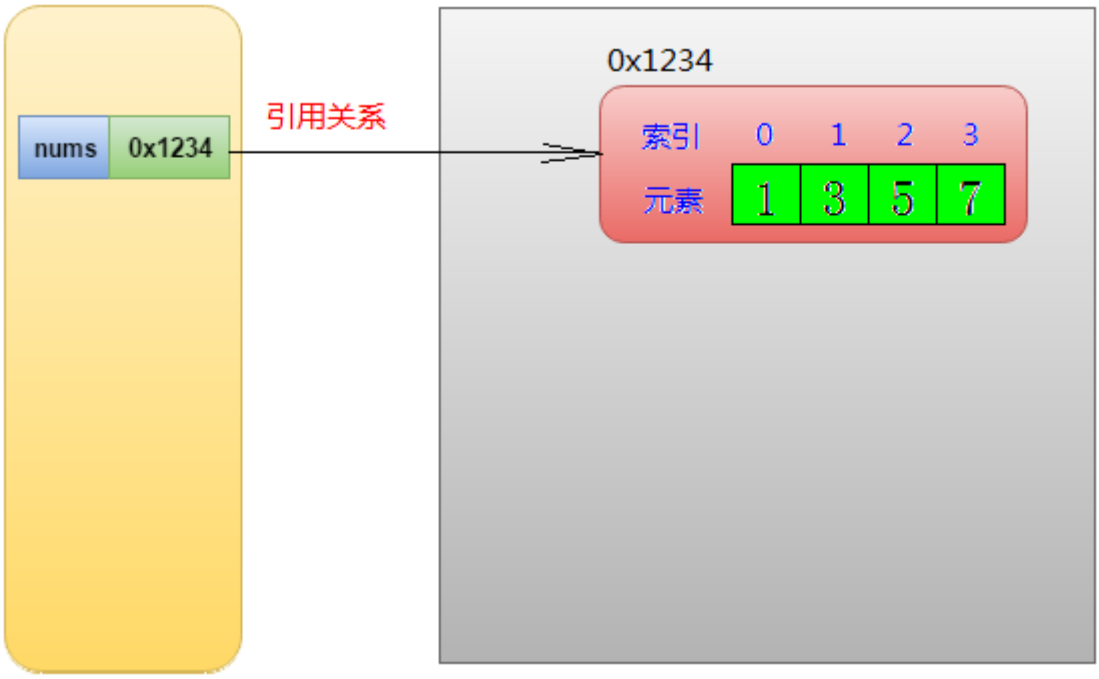
- 在栈中开辟一块空间nums



- 把堆空间表示数组的内存地址赋给nums



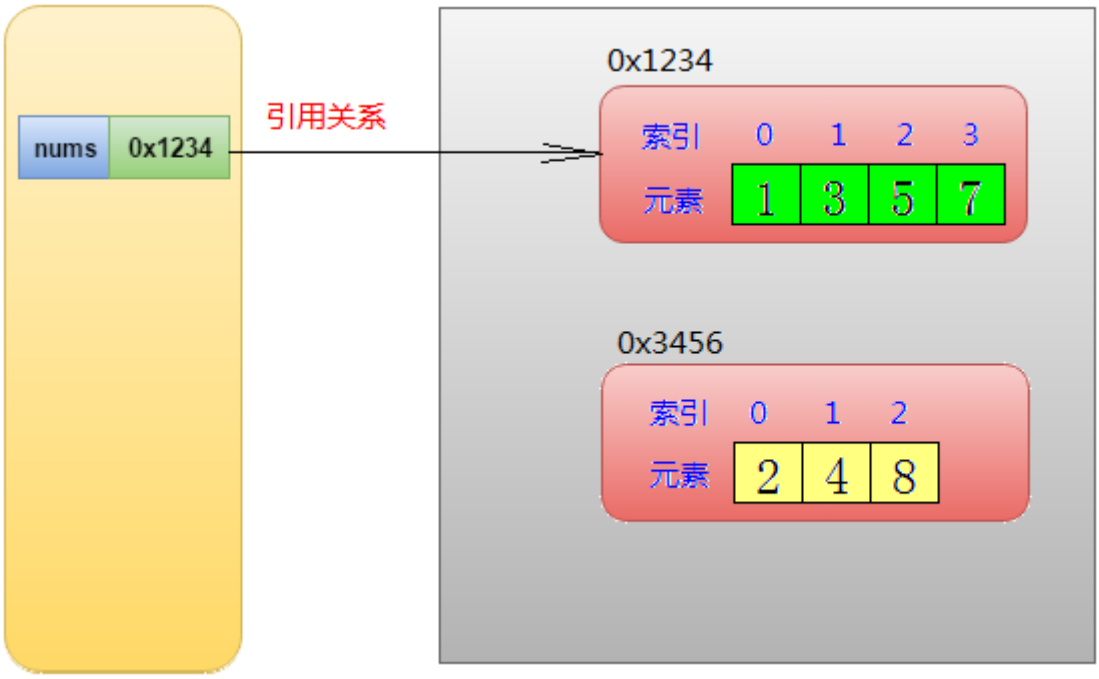
这种把内存地址赋给一个变量，也被称之为引用关系（为了更清晰有人习惯画一根箭头来表示这种关系），也就是说nums变量引用了堆中某一块内存地址，**当操作nums变量的时候，其实底层操作的是nums所引用内存地址里面的数据。**



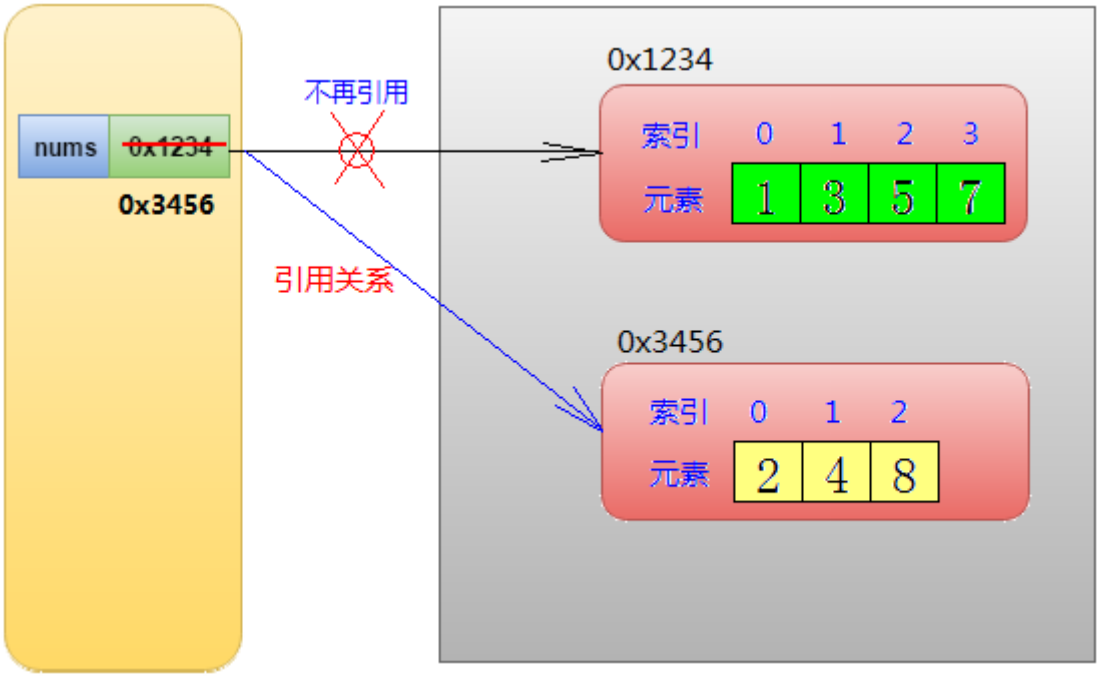
所以此时，通过nums.length代码来查看nums数组中有几个元素，结果很明显是4。

nums = new int[] { 2, 4, 8 };

- 因为存在new，说明又会在堆空间开辟一块新的空间，赋初始值。



- 并把内存地址重新赋给nums变量，nums原来所引用的地址将被覆盖掉。



所以此时，通过nums.length代码来查看nums数组中有几个元素，结果很明显是3。

注意：此时地址为0x1234的内存空间没有被任何变量所引用，所以该空间就变成了无用的垃圾，就等着垃圾回收器回收该空间。

如果存在一行代码，如下：

nums = null;

null表示不再引用堆中的内存空间，那么此时nums就好比是没有初始化的，不能使用。

3.3 动态初始化（重点）

程序员只设置数组元素个数，开发者可以提前把数组内存空间申请好，然后再程序运行过程中添值。

```
语法：
数组元素类型[]  数组名 = new 数组元素类型[length];
// 例如
int[] nums = new  int[5];
```

不同数据类型的初始值：

数据类型	默认初始化值
byte、short、int、long	0
float、double	0.0
char	一个空字符（空格），即'\u0000'
boolean	false
引用数据类型	null，表示不引用任何对象

int[] arr1 = new int[3]; int类型数组，每一个元素的初始值为0

double[] arr2 = new double[5]; double类型数组，每一个元素初始值为0.0

String[] arr3 = new String[2]; String类型数组，每一个元素的初始值为null

注意：不能同时指定元素值和数组长度，反例如下:

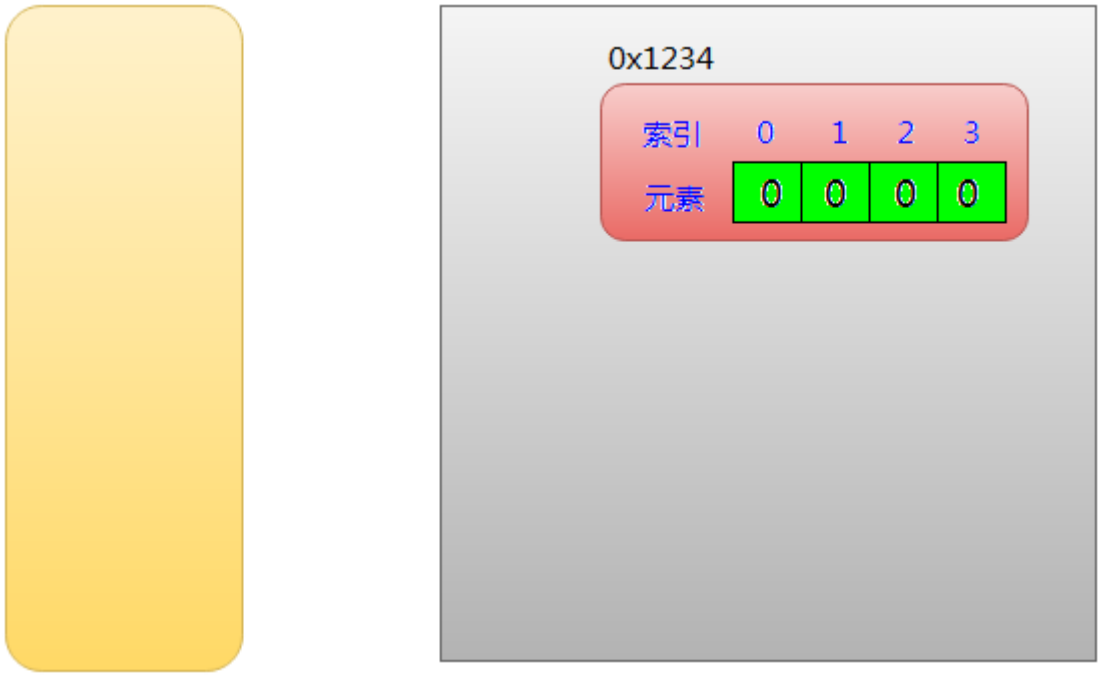
int[] nums = new int[5]{1,3,5,7,9};

3.4 动态初始化内存分析（理解即可）

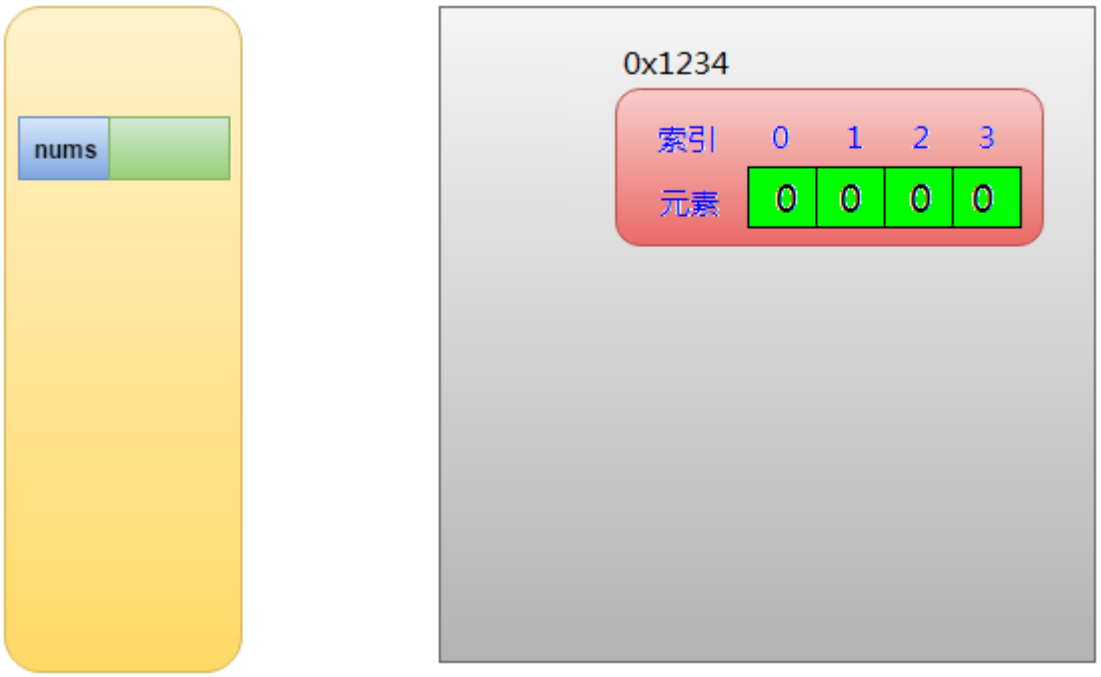
```
public class ArrayDemo2 {
    public static void main(String[] args) {
        // 定义并初始化数组
        int[] nums = new int[4];
        System.out.println("数组长度=" + nums.length);

        // 重新初始化数组
        nums = new int[5];
        System.out.println("数组长度=" + nums.length);
    }
}
```

这一行代码，确实可以分成三步：int[] nums = new int[4];

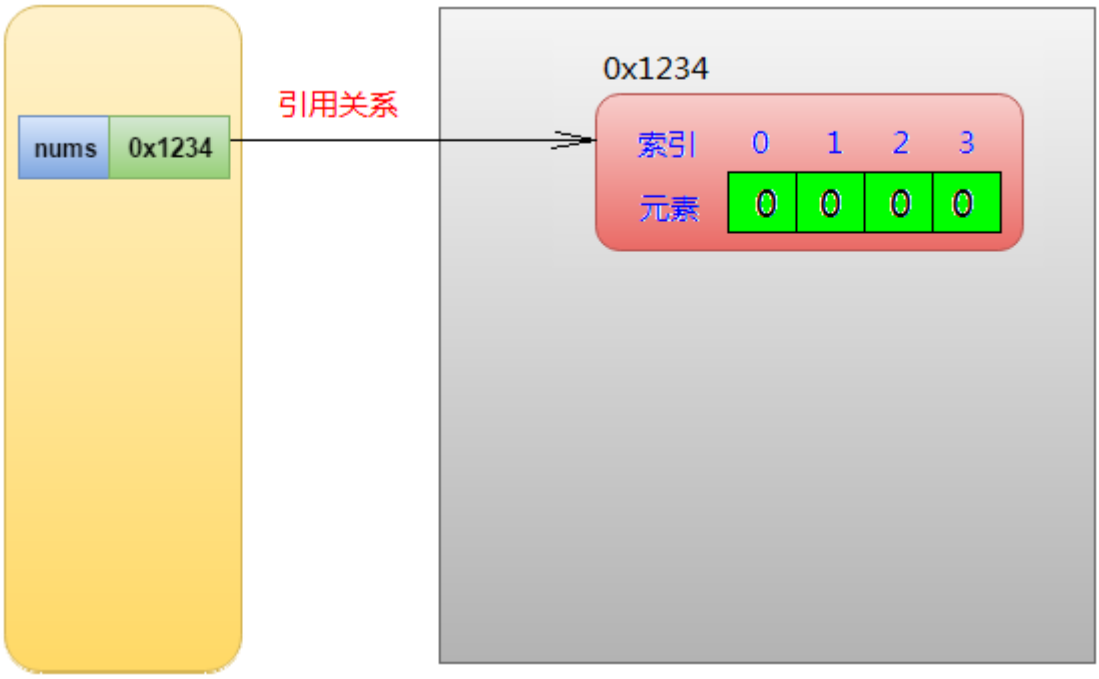
- 在堆内存中开辟一块空间，用来存储数组数据



- 在栈中开辟一块空间nums



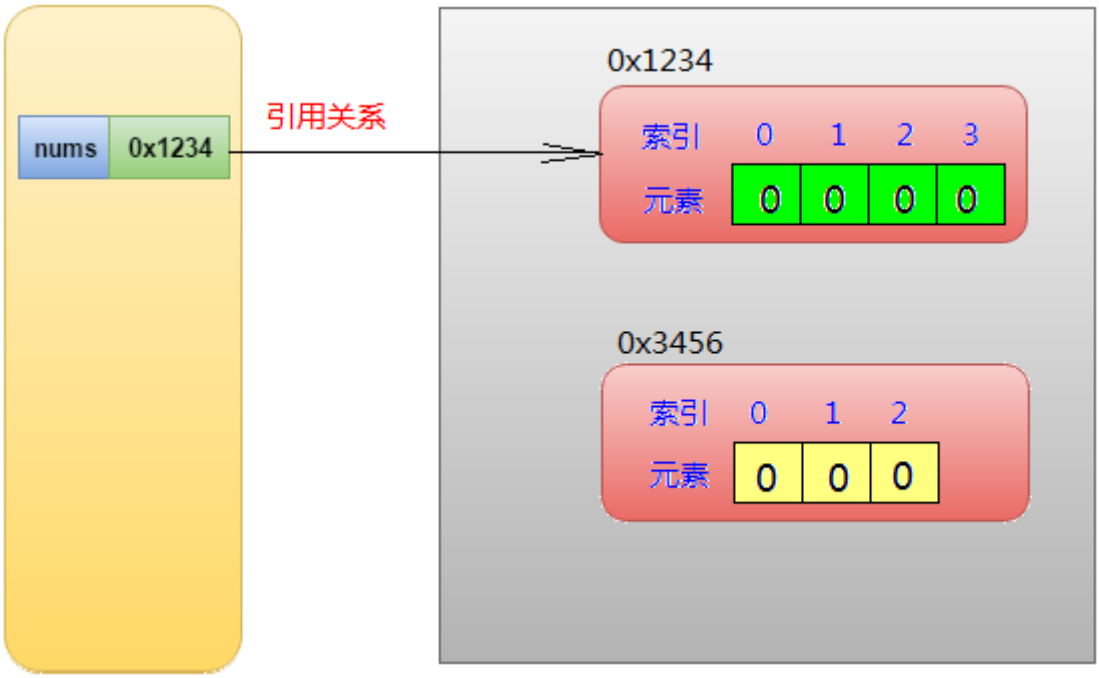
- 把堆空间表示数组的内存地址赋给nums



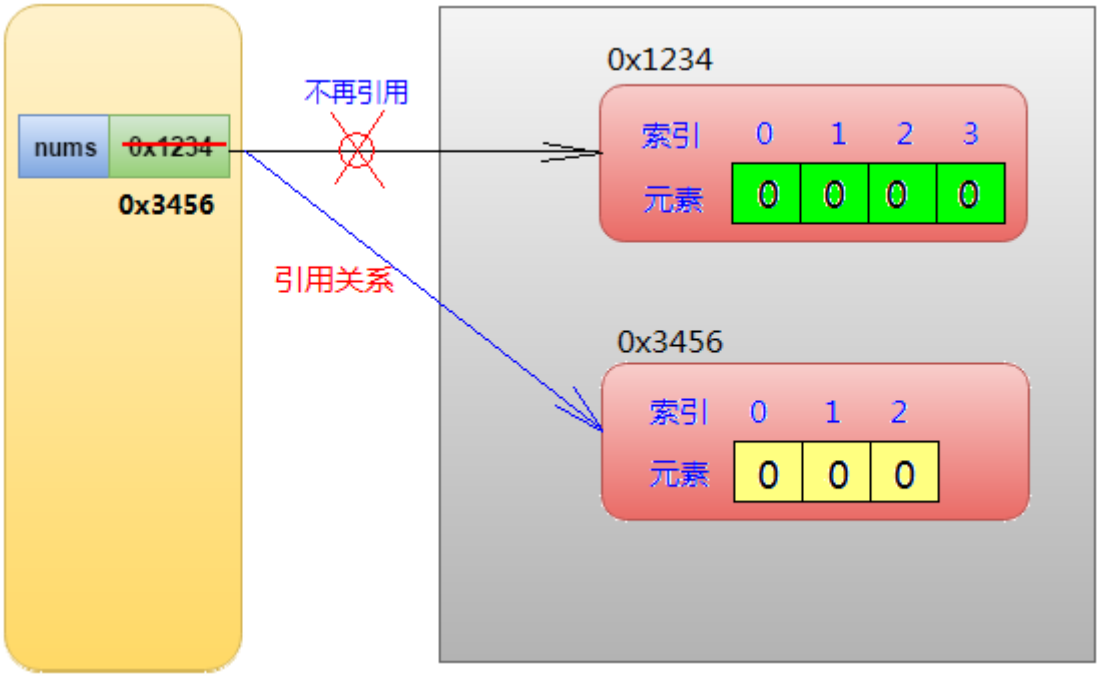
所以此时，通过nums.length代码来查看nums数组中有几个元素，结果很明显是4。

nums = new int[3] ;

- 因为存在new，说明又会在堆空间开辟一块新的空间，赋初始值。



- 并把内存地址重新赋给nums变量，nums原来所引用的地址将被覆盖掉。

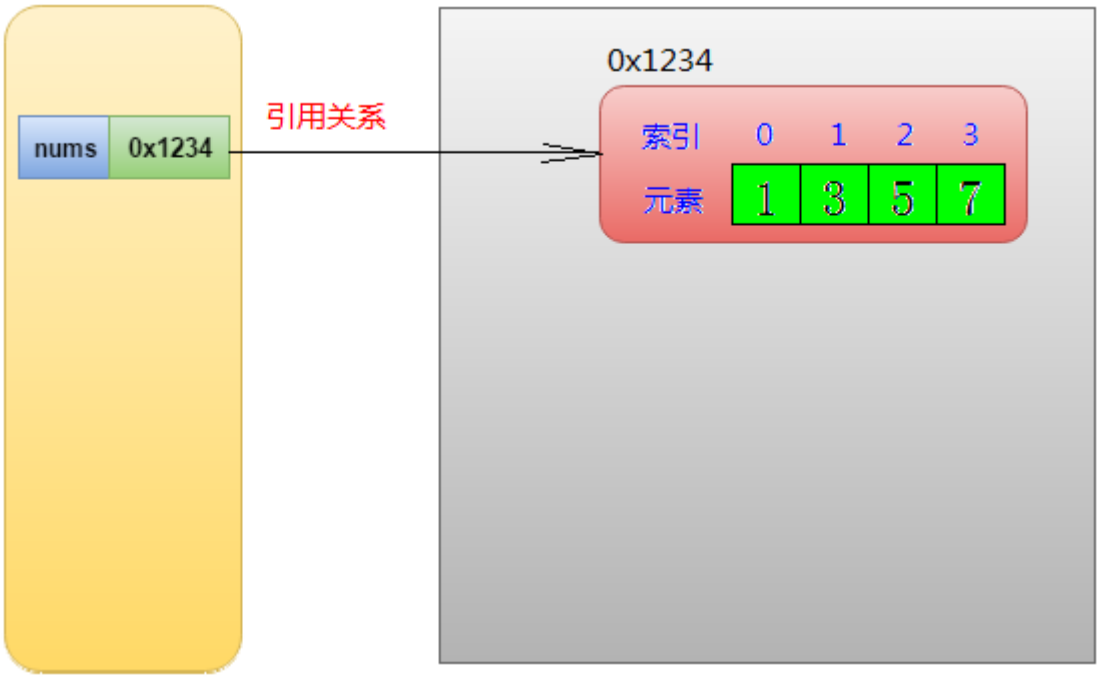


所以此时，通过nums.length代码来查看nums数组中有几个元素，结果很明显是3。

4、数组操作（重点）

4.1 基本操作（重点）

```
int[] nums = new int[]{1,3,5,7};
```



- **获取数组长度。**获取数组元素的个数

```
语法：
int size =  数组名.length;

例如:nums.length 可以获取nums数组元素的个数
int size = nums.length;  // 输出结果4
```

- 获取元素值

```
语法：
元素类型 变量名 = 数组名[index];
数组的索引从0开始，最大索引值是数组长度-1，那么索引范围是 [0, 数组名.length-1]

// 获取第一个元素
int ele1 = nums[0];     // 输出1
// 获取第二个元素：
int ele2 = nums[1];     // 输出3
// 获取第四个元素：
int ele4 = nums[3];     // 输出7
```

- 设置元素值。把一个新值存入数组对应索引位置的空间中，原来的值会被覆盖。

```
语法：
数组名[index] = 值;

// 设置第二个元素值为30
nums[1] =  30;

// 获取第二个元素
int  ele2 = nums[1];    // 输出30
```

常见的异常

1. NullPointerException：空指针异常（空引用异常）

   操作了一个尚未初始化或者没有分配内存空间的数组

2. **ArrayIndexOutOfBoundsException：数组的索引越界异常**

   操作的数组的索引不在[0，数组名.length-1]范围内

4.3 数组遍历操作 （重点）

迭代数组，也叫遍历数组（获取出数组中每一个元素）

获取第一个元素：int ele1 = nums[0];　　输出1

获取第二个元素：int ele2 = nums[1];　　输出30

获取第三个元素：int ele3 = nums[2];　　输出5

获取第四个元素：int ele4 = nums[3];　　输出7

发现，循环遍历的次数是数组元素的个数，每一次获取元素只有索引在变化，范围是[0 , 数组名.length - 1]。

```
int[] nums = new int[] { 1, 3, 5, 7 };

for (int index = 0; index < nums.length; index++) {
    int ele = nums[index];//index依次是 0、1、2、3
    System.out.println(ele);
}
```

- 使用for-each（增强for循环）操作数组

```
for(数组元素类型 变量： 数组){
    // 使用变量
}
```

使用for-each操作数组更简单，因为可以不关心索引，其底层原理依然是上述的for循环操作数组。

```
int[] nums = new int[] { 1, 3, 5, 7 };

for (int ele : nums) {
    System.out.println(ele);
}
```

4.2.使用循环操作数组（重点）

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

需求1：找出数组中元素22第一次出现的索引位置

```
public class Demo06Array {
    public static void main(String[] args) {
        int[] arr = {11,22,33,44,55,22,66};
        int target = 220;         // 要查找的目标元素
        int index = -1;           // 记录元素第一次出现的索引

        // step 1: 依次取出数组的每个元素
        for(int i = 0; i < arr.length; i++) {
            int item = arr[i];
            // step 2: 用取出的元素和要找的目标元素22进行比较
            if( item == target ) {
                // step 3: 如果元素和22相等，停止查找并打印输出，否则继续查找
                index = i;
                break;
            }
        }

        if(index < 0) {
            System.out.println("未找到");
        } else {
            System.out.println("找到，位置：" + index);
        }
    }
}
```

需求2：求出int类型数组中最大元素值

思路：

1. 暂定第一个元素为最大，用变量max存起来
2. 用max和后面的元素比较，如果元素比max大，则把这个元素赋值给max
3. 遍历完成后，max存的数值就是最大值

```
public class ArrayDemo4 {
    public static void main(String[] args) {
        int[] nums = new int[] { 11, 22, 33, 44, 22, 55 };
        // step 1: 定义一个变量max 用于存储最大值，假设第一个元素最大
        int max = arr[0];

        // step 2: 从1号位置开始，依次取出每个元素
        for(int i = 1;i < arr.length;i++) {
            int ele = arr[i];
            // step 3: 取出来的元素和max比较，如果取出来的元素大于max，把大的元素存入max
中，直到全部比较完成
            if( ele > max ) {
                max = ele;
            }
        }
        // step 4: 打印max
        System.out.println("max = " + max);
    }
}
```

需求3：按照某种格式打印数组元素，数组元素放在方括号[]中,相邻元素使用逗号分隔开。如上述数组打印出来，效果如：[11, 22, 33, 44, 22, 55]

思路：

1. 定义一个String类型的变量，准备用来拼接指定格式的字符串
2. 将 [拼接到字符串中
3. 通过循环遍历的方式，将每个元素拼接到字符串中，每个元素后面拼接一个逗号，
4. 如果是最后一个元素，不是拼接逗号，而是拼接一个]

```
public class ArrayDemo5 {
    public static void main(String[] args) {
        int[] nums = new int[] { 11, 22, 33, 44, 22, 55 };
        String str = "[";//str表示结果字符串,先拼一个[符号
        for (int index = 0; index < nums.length; index++) {
            //把每一个元素拼接在str后面
            str = str + nums[index];
            //如果是最后一个元素,则不拼接,而是]
            if(index == nums.length-1) {
                str = str + "]";
            }else {
                //如果不是最后一个元素拼接,
                str = str + ", ";
            }
        }
```

```
        System.out.println(str);
    }
}
```

5、二维数组（了解）

在之前，数组的每一个元素就是一个个的值，这种数组我们称之为一维数组。如：

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

这个数组中存储的是int类型的元素，该数组就是一个一维数组。

二维数组，就是数组中的每一个元素都是一个一维数组。

二维数组就是用在装一维数组的数组

5.1 二维数组的定义和初始化

定义和静态初始化一维数组的语法：

```
数组元素类型[] 数组名 = new 数组元素类型[]{值1，值2，值3，...};
如：int[] nums = new int[]{1,3,5,7};
```

定义和静态初始化二维数组的语法：

```
数组元素类型[][] 数组名 = new 数组元素类型[][] {数组1，数组2，数组3，...};
```

注意，二维数组中的元素类型是一维数组，把数组元素类型[]看成一个整体，表示数据类型。

```
public class ArrayInArrayDemo1 {
    public static void main(String[] args) {
        //定义三个一维数组
        int[] arr1 = { 1, 2, 3 };
        int[] arr2 = { 4, 5 };
        int[] arr3 = { 6 };

        //把三个一维数组存储到另一个数组中,那么该数组就是二维数组
        int[][] arr = new int[][] { arr1, arr2, arr3 };
    }
}
```

更简单的写法(字面量写法)

```
int[][] arr = new int[][] {
    { 1, 2, 3 },
    { 4, 5 },
    { 6 }
};
或者
int[][] arr = {
    { 1, 2, 3 },
    { 4, 5 },
    { 6 }
};
```

定义和动态初始化二维数组的语法：

`数组元素类型[][] 数组名 = new 数组元素类型[x][y];`

x表示二维数组中有几个一维数组

y表示每一个一维数组中有几个元素。

`int[][] arr = new int[3][5];`

System.out.println(arr.length); //输出3

5.2 获取二维数组的元素

因为二维数组表示数组的中的数组，如果要获取数组的每一个元素值，则需要两个循环嵌套。

```
//二维数组
int[][] arrInArr = new int[][] {
    { 1, 2, 3 },
    { 4, 5 },
    { 6 }
};
```

使用for循环：

```
for (int index = 0; index < arrInArr.length; index++) {
    //取出每一个一维数组
    int[] arr= arr[index];
    //迭代一维数组
    for (int j = 0; j < arr.length; j++) {
        int ele = arr[j];
        System.out.print(ele + " ");
    }
    System.out.println();
}
```

使用for-each：

```
for (int[] arr : arrInArr) {
    //arr2为每次遍历出来的一维数组
    for (int ele : arr) {
        //ele为从arr一维数组中遍历出来的元素
        System.out.print(ele + " ");
    }
    System.out.println();
}
```