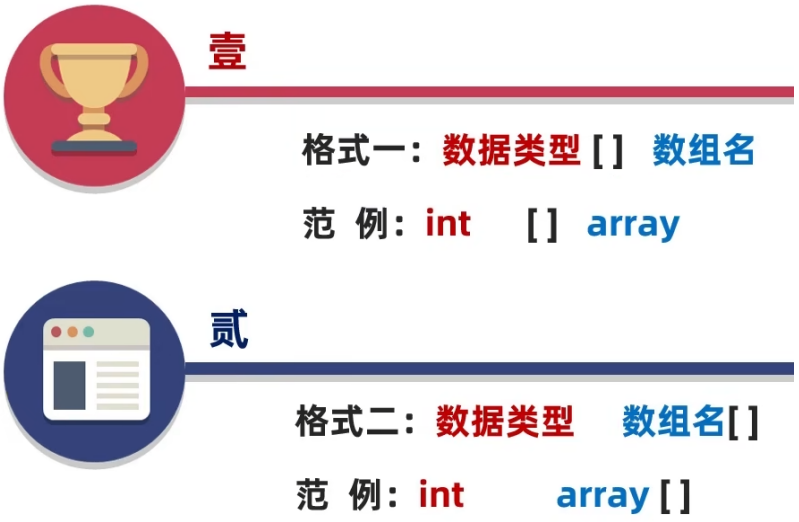
08 数组

- **1 介绍**：存储**同种数据类型**的多个值
  - 数组容器在存储数据的时候，需要结合隐式转换考虑。
  - 例如：int类型的数组容器 （boolean ✗ byte ✓ short ✓ int ✓ double ✗）
  - 例如：double类型的数组容器（byte ✓ short ✓ int ✓ long ✓ float ✓ double ✓）
  - 建议：容器的类型，和存储的数据类型保持一致
- **2 定义格式**：



- **3 数组静态初始化:**
  - 完整格式

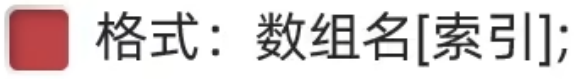
    数据类型[] 数组名 = new 数据类型[] { 元素1，元素2，元素3... };
  - 简化格式

    数据类型[] 数组名 = { 元素1，元素2，元素3... };
- **4 数组的地址值：表示数组在内存中的位置**

```
int[] arr = {1,2,3,4,5};
System.out.println(arr);//[I@6d03e736
```
地址值

```
double[] arr2 = {1.1,2.2,3.3};
System.out.println(arr2);//[D@568db2f2
```
地址值

  - [：表示当前是一个数组
  - D：表示当前数组里面的元素都是double类型的
  - @：表示间隔符号
  - 568db2f2：才是真正地址值(16进制)
- **5 数组元素访问**



  - **索引**：

索引：也叫做下标，角标。

索引的特点：从0开始，逐个+1增长，连续不间断

- **6 数组遍历 for（int i =0; i<arr.length; i++）**
  - 数组的长度属性length的调用方式
    - 数组名.length
  - 快捷生成数组的遍历方式：**数组名.fori**
- **7 数组动态初始化：**
  - 只指定数组长度，有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];
  - 数值默认初始化值的规律
    - 整数：默认0
    - 小数：默认0.0
    - 字符：默认'\u0000' （空格）
    - 布尔：默认false
    - 引用数据类型：默认 null
- **8 静态和动态初始化的区别**

动态初始化：手动指定数组长度，由系统给出默认初始化值。

只明确元素个数，不明确具体数值，推荐使用动态初始化

```
举例：使用数组容器来存储键盘录入的5个整数。
int[] arr = { ? ? ? ? ? };
int[] arr = new int[5];
```

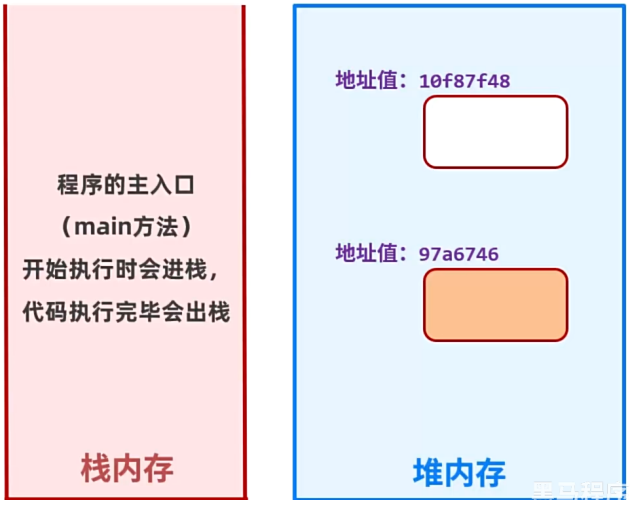
静态初始化：手动指定数组元素，系统会根据元素个数，计算出数组的长度。

需求中已经明确了要操作的具体数据，直接静态初始化即可。

```
举例：将全班的学生成绩存入数组中 11，22，33
int[] arr = {11, 22, 33};
```

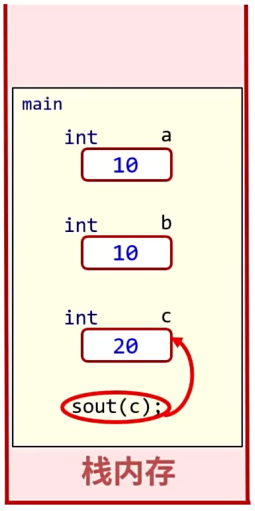
- **9 数组的常见问题**
  - 索引越界：知道索引范围（0~数组长度-1）
- **10 数组常见操作**
  - 求最值
  - 求和
  - 交换数据
  - 打乱数据
- **11 Java的内存分配**

- 栈　方法运行时使用的内存，比如main方法运行，进入方法栈中执行
- 堆　存储对象或者数组，new来创建的，都存储在堆内存
- 方法区　存储可以运行的class文件
- 本地方法栈　JVM在使用操作系统功能的时候使用，和我们开发无关
- 寄存器　给CPU使用，和我们开发无关

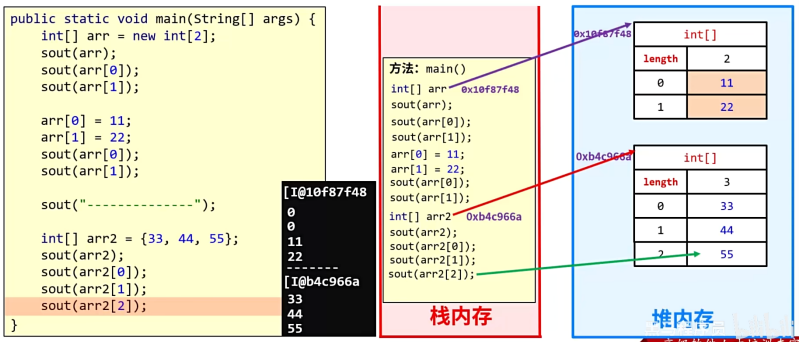


- main方法的内存图

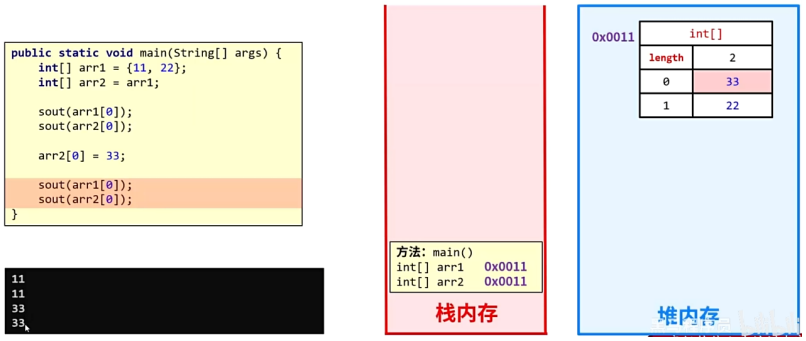
```
public static void main(String[] args) {
    int a = 10;
    int b = 10;
    int c = a + b;

    sout(c);

}
```



- 数组的内存图



- **两个数组指向同一个空间的内存图**



当两个数组指向同一个小空间时，其中一个数组对小空间中的值发生了改变，那么其他数组再次访问的时候都是修改之后的结果了。

- **12 二维数组：数组中存数组**
  - **应用场景**：把数据分组管理时需要
  - **初始化**：
    - **静态初始化**
      - 格式：数据类型[][] 数组名 = new 数据类型[][] {{元素1,元素2},{元素1, 元素2}};
      - 范例：int[][] arr = new int[][]{{11,22},{33,44}};
      - 简化格式：数据类型[][] 数组名 = {{元素1,元素2}, {元素1, 元素2}};
      - 范例：int[][] arr = {{11,22},{33,44}};
      - 范例：int arr[][] = {{11,22},{33,44}};

```java
//以后建议这样定义，把每一个一维数组，单独写成一行
//注意：每一个一维数组其实是二维数组中的元素，所以每一个一维数组之间需要用逗号隔开。
int[][] arr3 = {
        {1, 2, 3},
        {4, 5, 6, 7, 8}
};
//2.获取元素
//arr[i][j]
//arr：二维数组
//i：二维数组的索引，获取出来的是里面的一维数组
//j：表示一维数组中的索引，获取出来的就是真正的元素

//3、二维数组遍历
//外循环：遍历二维数组，得到里面的每一个一维数组
for (int i = 0; i < arr3.length; i++) {
    //i：表示二维数组中的每一个索引
    //arr3[i]：表示二维数组中的每一个元素（一维数组）
    //内循环：遍历一维数组，得到里面的每一个元素
    for (int j = 0; j < arr3[i].length; j++) {
        //j:表示一维数组中的每一个元素
        System.out.print(arr3[i][j] + " ");
    }
    System.out.println();
}
```

- **动态初始化**
  - 格式：数据类型[][] 数组名 = new 数据类型[m][n];

    m表示这个二维数组，可以存放多少个一维数组

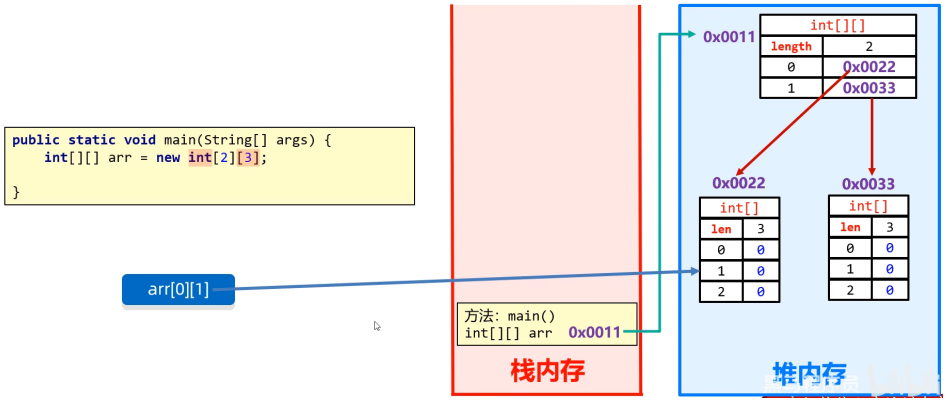
    n表示每一个一维数组，可以存放多少个元素

  - 范例：int[][] arr = new int[2][3];

    该数组可以存放2个一维数组，每个一维数组中可以存放3个int类型元素

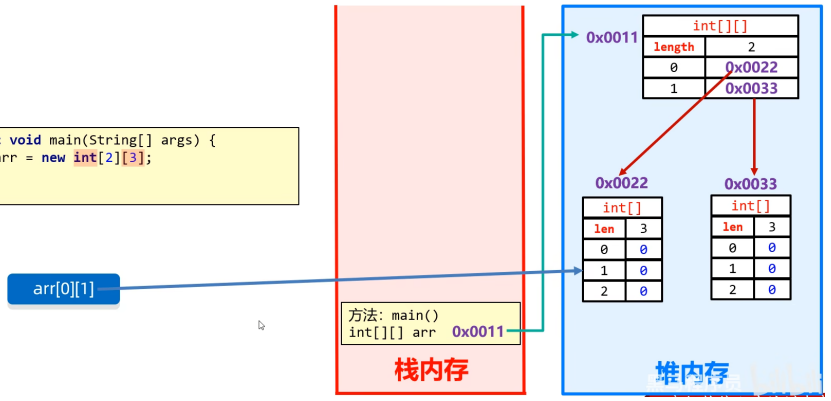
- **二维数组的内存图**

二维数组的内存图



二维数组的内存图

```
public static void main(String[] args) {
    int[][] arr = new int[2][3];

}
```



二维数组的内存图

特殊情况二:

```
public static void main(String[] args) {
    int[][] arr = new int[2][3];
    int[] arr1 = {11, 22};
    int[] arr2 = {44, 55, 66};

    arr[0] = arr1;
    arr[1] = arr2;

}
```