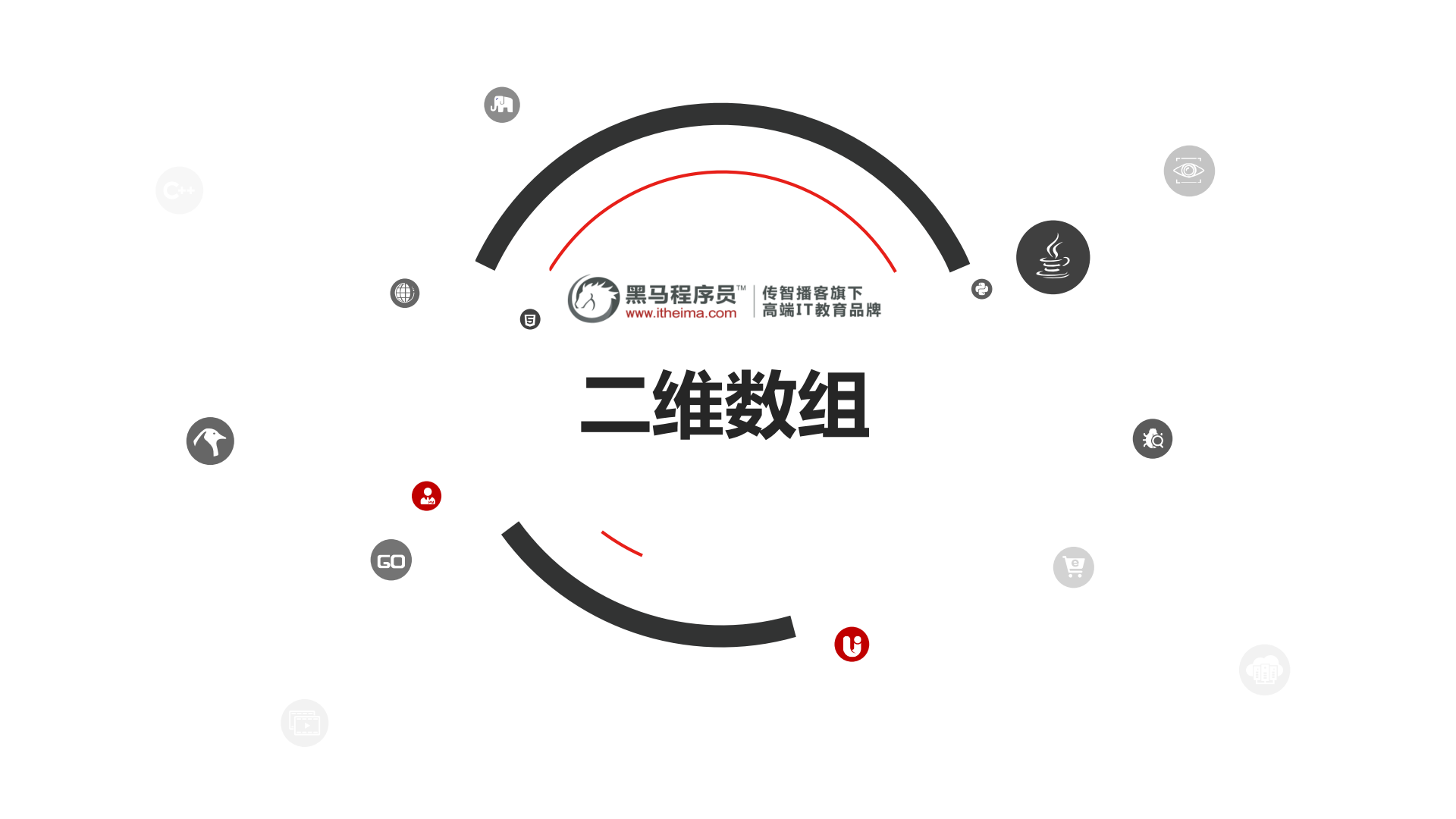


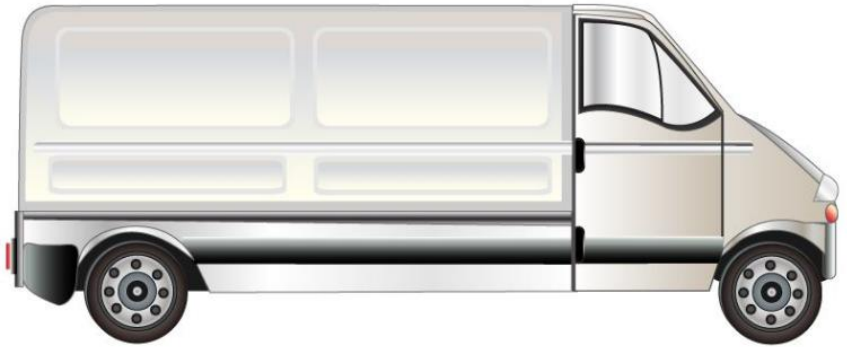
二维数组

目录 Contents

二维数组概述

什么是二维数组

- 二维数组也是一种**容器**，不同于一维数组，该容器存储的都是一维数组容器



二维数组概述

为什么要有二维数组?

- 某公司季度，和月份统计的数据如下：单位(万元)

  第一季度：22,66,44

  第二季度：77,33,88

  第三季度：25,45,65

  第四季度：11,66,99

目录 Contents

二维数组初始化之动态初始化

二维数组定义格式

- 格式1：数据类型 [ ] [ ] 变量名;
- 范例：int [ ] [ ] arr;

- 格式2：数据类型 变量名[ ] [ ];
- 格式3：数据类型[ ] 变量名[ ];

- 范例：int arr[ ] [ ];
- 范例：int[ ] arr[ ];

二维数组初始化之动态初始化

二维数组动态初始化

- 格式：数据类型 [ ] [ ] 变量名 = 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];

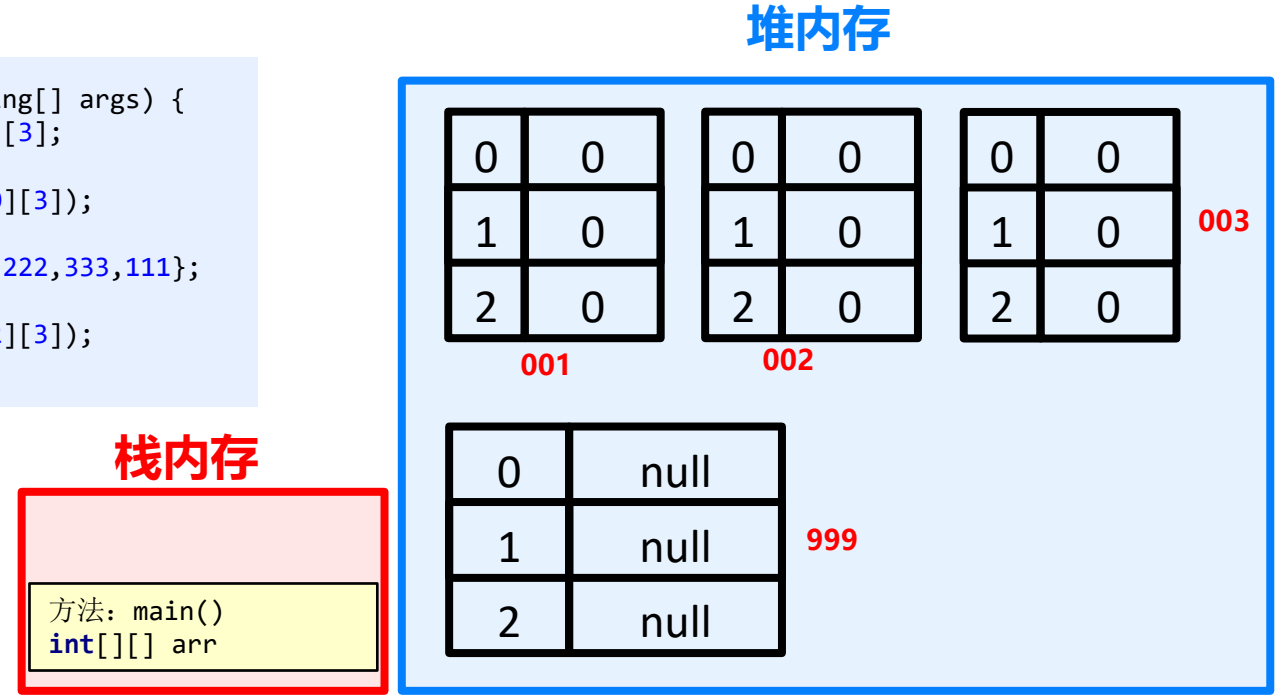
该数组可以存放2个一维数组，每个一维数组中可以存放3个int类型元素

二维数组初始化之动态初始化

二维数组

```
public static void main(String[] args) {
    int[][] arr = new int[3][3];
    arr[0][3] = 11;
    System.out.println(arr[0][3]);

    int[] smallArray = {111,222,333,111};
    arr[2] = smallArray;
    System.out.println(arr[2][3]);
}
```

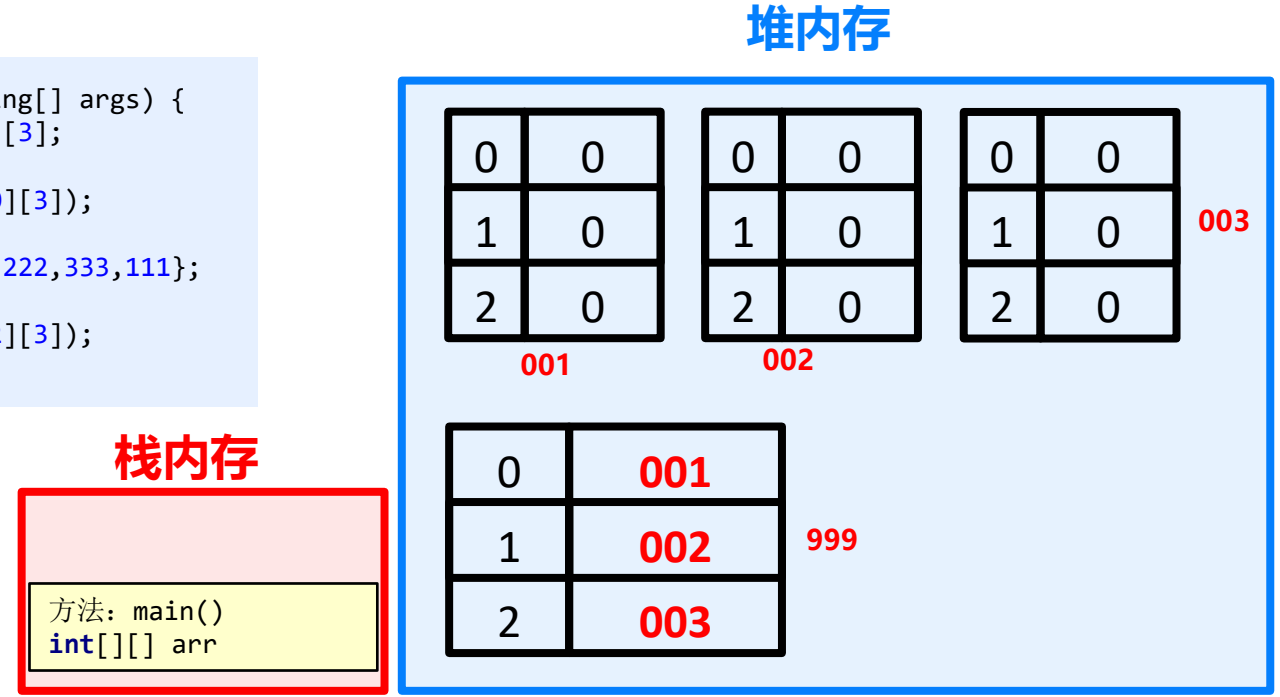


二维数组初始化之动态初始化

二维数组

```
public static void main(String[] args) {
    int[][] arr = new int[3][3];
    arr[0][3] = 11;
    System.out.println(arr[0][3]);

    int[] smallArray = {111,222,333,111};
    arr[2] = smallArray;
    System.out.println(arr[2][3]);
}
```

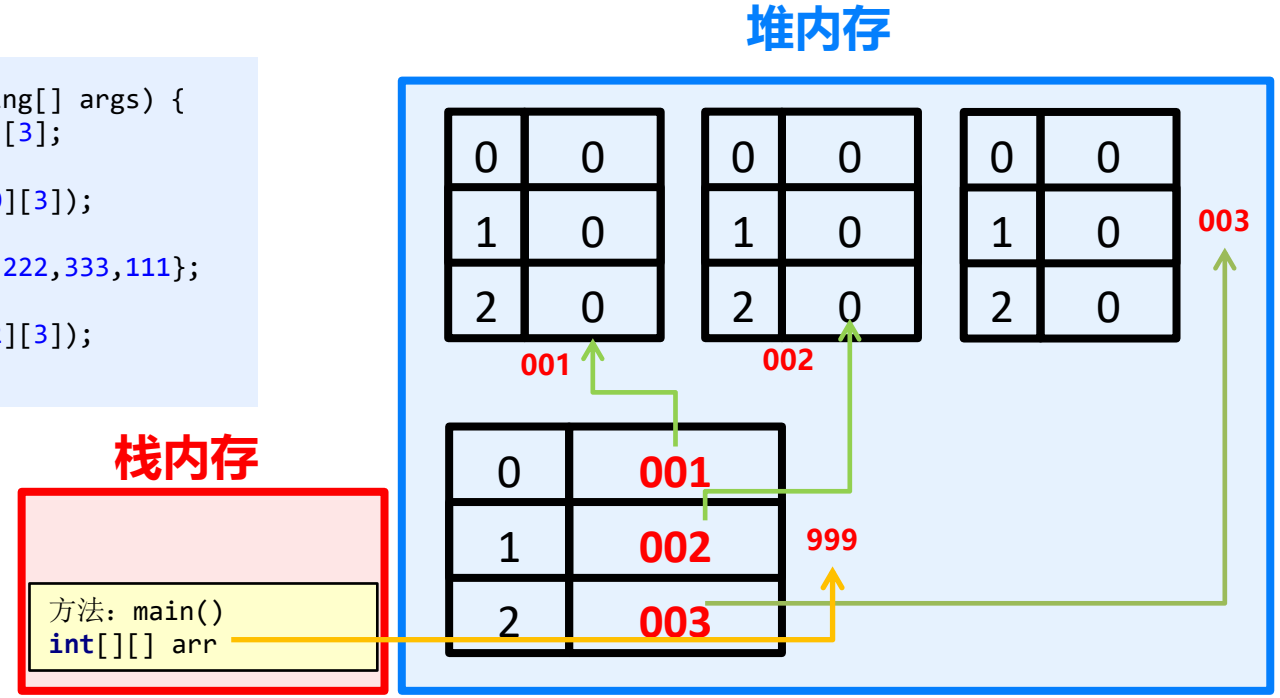


二维数组初始化之动态初始化

二维数组

```
public static void main(String[] args) {
    int[][] arr = new int[3][3];
    arr[0][3] = 11;
    System.out.println(arr[0][3]);

    int[] smallArray = {111,222,333,111};
    arr[2] = smallArray;
    System.out.println(arr[2][3]);
}
```

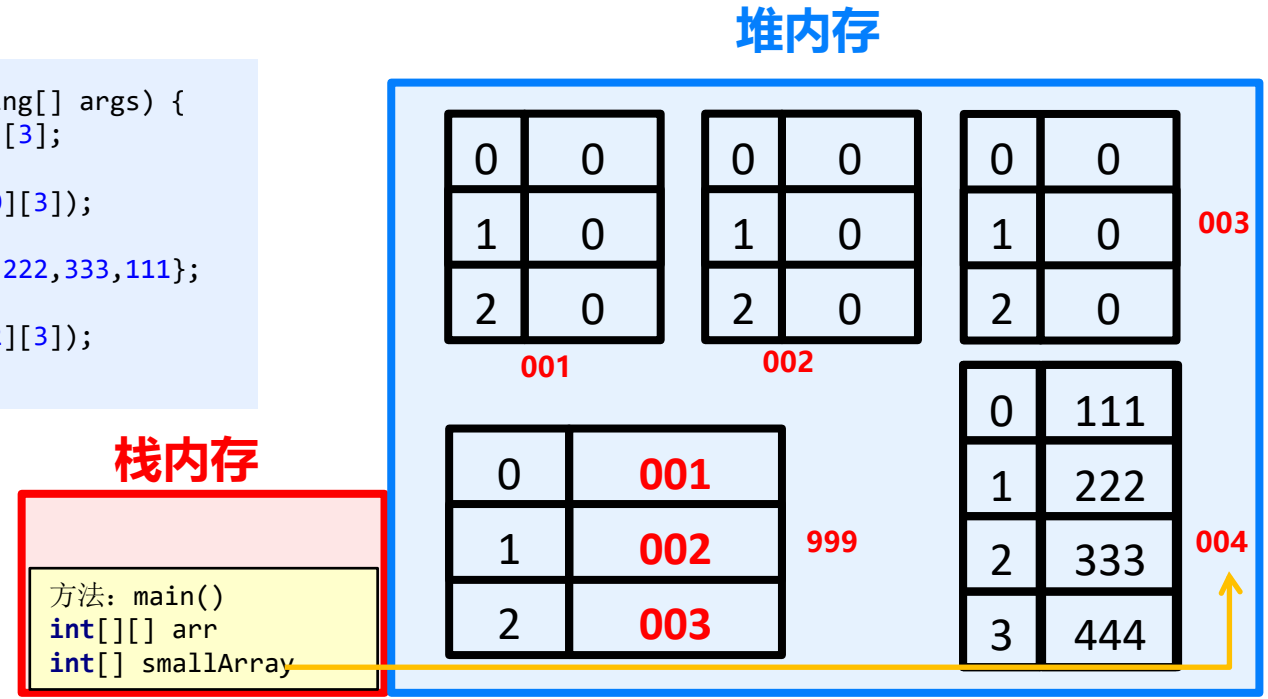


二维数组初始化之动态初始化

二维数组

```
public static void main(String[] args) {
    int[][] arr = new int[3][3];
    arr[0][3] = 11;
    System.out.println(arr[0][3]);

    int[] smallArray = {111,222,333,111};
    arr[2] = smallArray;
    System.out.println(arr[2][3]);
}
```



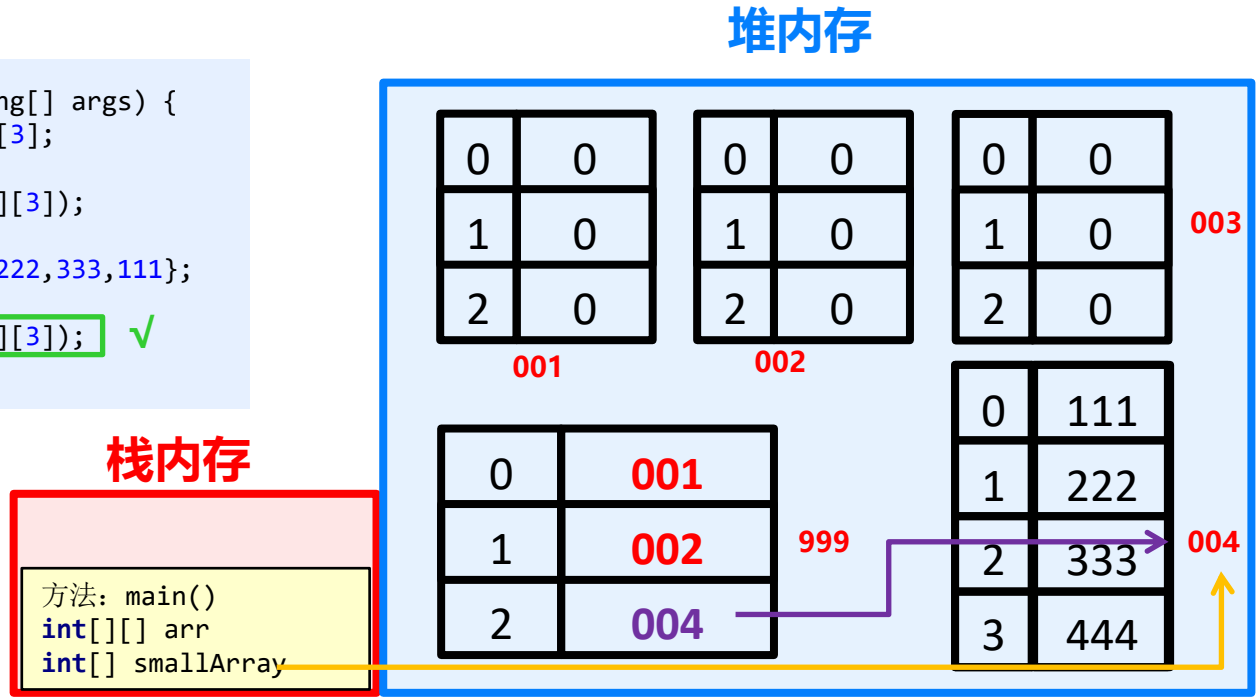
二维数组初始化之动态初始化

二维数组

```
public static void main(String[] args) {
    int[][] arr = new int[3][3];
    arr[0][3] = 11;
    System.out.println(arr[0][3]);

    int[] smallArray = {111,222,333,111};
    arr[2] = smallArray;
    System.out.println(arr[2][3]);
}
```

√



目录 Contents

二维数组初始化之静态初始化

二维数组静态初始化

- 格式：数据类型 [ ] [ ] 变量名 = 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} };

二维数组初始化之静态初始化

二维数组静态初始化

- 简化格式：数据类型 [ ] [ ] 变量名 = { {元素1,元素2}, {元素1, 元素2} };
- 范例：int [ ] [ ] arr = { {11,22} , {33,44} };

目录 Contents

二维数组-遍历

二维数组遍历

- 已知一个二维数组 arr = { {11 , 22 , 33} , {33 , 44 , 55} }；遍历该数组，取出所有元素并打印

实现思路：

①：遍历二维数组，取出里面每一个一维数组

②：在遍历的过程中，对每一个一维数组继续完成遍历，获取内部存储的每一个元素

```
for (int i = 0; i < arr.length; i++) {
        // arr[i] 就是每一个一维数组
        for (int j = 0; j < arr[i].length; j++) {

        }

}
```

二维数组-求和

案例：公司年销售额求和

- 某公司季度和月份统计的数据如下：单位(万元)

  第一季度：22,66,44

  第二季度：77,33,88

  第三季度：25,45,65

  第四季度：11,66,99

实现思路：

①：定义求和变量，准备记录最终累加结果

②：使用二维数组来存储数据，每个季度是一个一维数组，再将4个一维数组装起来

③：遍历二维数组，获取所有元素，累加求和

④：输出最终结果