Java 中的数组

- **数组定义**

  数组是一种数据结构，是**相同类型**数据的**有序集合**。

  数组中的每一个数据称作一个**元素**，每个元素可以通过一个**索引（下标）**来访问。

  - 数组的特点：

    1）**长度是确定的**。数组一旦被创建，它的大小不可改变。

    2）Java 中数组的元素类型必须是**相同类型**，不允许出现混合类型 *（Python 是可以的）*。

    3）数组的元素类型可以是任何数据类型。

    4）数组有索引：**索引从 0（第 2 个元素）开始，到 数组.length-1（最后一个元素） 结束**。

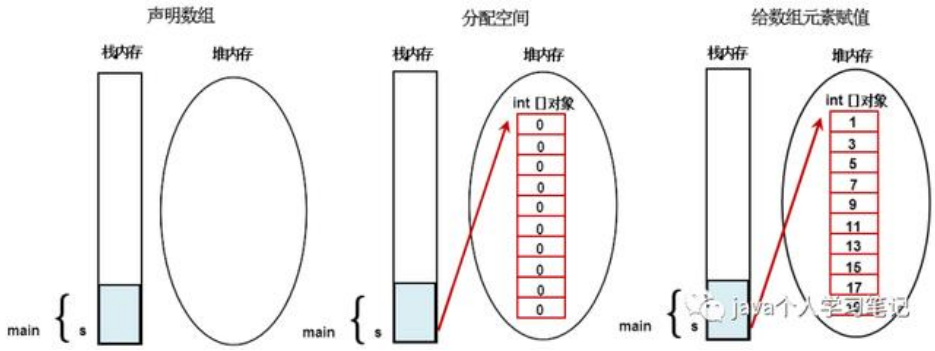
- **数组的创建**

  以一维数组为例：type[ ] arr_name; //比如 int[] iArray;

  注意：*声明数组并没有实例化任何对象*，只有在实例化数组对象时，JVM （Java 虚拟机）才分配空间，这时才与长度有关；**声明一个数组时并没有数组真正被创建**； **构造一个数组，必须指定长度**。

  - 创建基本数据类型一维数组

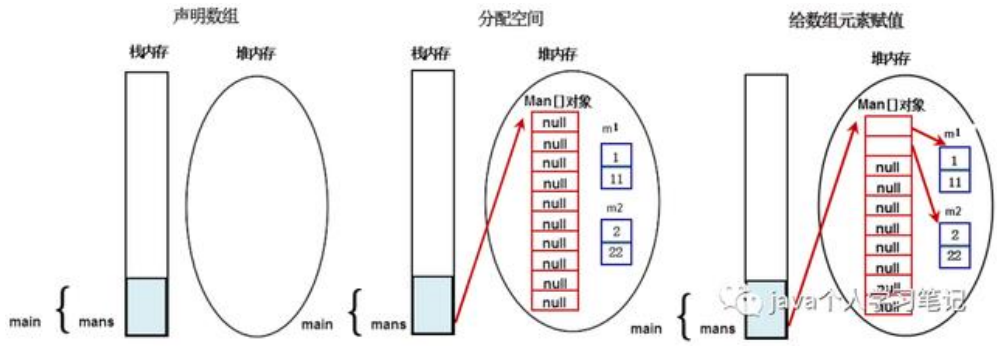
```
public class Test {
    public static void main(String args[ ]) {
        int[ ] s; // 声明数组，s 是一个 int 数组，此时只有 1 个变量 s，并没有给 s 分配数组空间
        s = new int[10]; // 创建具有 10 个元素的数组，此时给数组分配空间
        for (int i = 0; i < 10; i++) {
            s[i] = 2 * i + 1; //给数组元素赋值
            System.out.println(s[i]);
        }
    }
}
```



  - 创建 Class 数据类型一维数组

```
class Man{ //定义一个 Man 类型
    private int age;
    private int id;
    public Man(int id, int age) {
        this.age = age;
        this.id = id;
    }
}
public class AppMain {
    public static void main(String[ ] args) {
        Man[ ] mans; //声明引用类型数组；此时只有 mans 变量，并没有给 mans 分配数组空间
        mans = new Man[10]; //创建 mans，具有 10 个元素，给引用类型数组分配空间；
        Man m1 = new Man(1,11); //每个元素是 Class 类型，需要单独创建实例（注意和基础类型区别）
        Man m2 = new Man(2,22);
        mans[0]=m1;//给引用类型数组元素赋值;
```

```
        mans[1]=m2;//给引用类型数组元素赋值;
    }
}
```



- **数组的初始化**

  数组三种初始化方式：**静态初始化、动态初始化、默认初始化**。

  - **静态初始化：定义数组的同时指定数组元素（**即同时为数组元素分配空间并赋值）。

    ```
    int[] a = {1, 2, 3};// 静态初始化基本类型数组;
    Man[] mans = {new Man(1, 1), new Man(2, 2)};// 静态初始化引用类型数组;
    ```

  - **动态初始化：**数组定义时未指定元素数据，需要通过 new 创建数组并赋值。

    ```
    int[] a = new int[2];//先分配空间;
    a[0] = 1;//给数组元素赋值;
    a[1] = 2;//给数组元素赋值;
    ```

  - **默认初始化：**数组元素相当于对象的属性，按属性的方式被默认初始化。

    ```
    int a[] = new int[2]; // 默认值：0,0
    boolean[] b = new boolean[2]; // 默认值：false,false
    String[] s = new String[2]; // 默认值：null, null
    ```

- **数组常用操作**

  - **遍历：**通过循环遍历数组的所有元素

    1）**普通 for 循环进行遍历（通过下标）：遍历时可以读取、修改元素的值**

    ```
    public class Test {
        public static void main(String[] args) {
            int[] a = new int[4];
            //初始化数组元素的值
            for(int i=0;i<a.length;i++){
                a[i] = 100*i;
            }
            //读取元素的值
            for(int i=0;i<a.length;i++){
                System.out.println(a[i]);
            }
        }
    }
    ```

    2）**for-each 循环：专用于读取数组或容器所有元素，不能修改元素值**

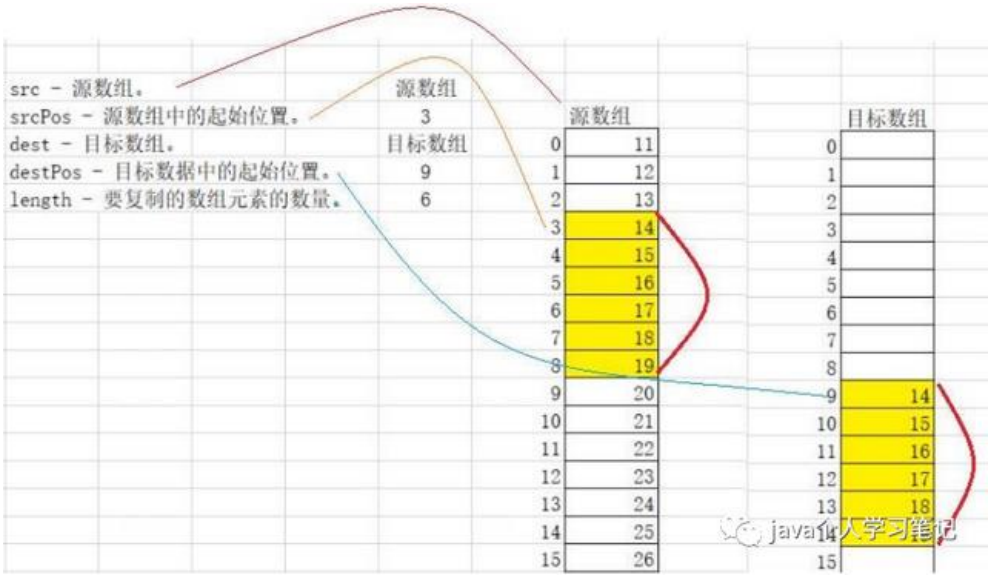
    ```
    public class Test {
        public static void main(String[] args) {
            String[] ss = {"aa", "bbb", "ccc", "ddd"};
            for (String temp : ss) { //增强 for 循环
                System.out.println(temp);
            }
    ```

```
        }
    }
```

- **拷贝：将某个数组的内容拷贝到另一个数组中**

  System.**arraycopy**（object src，int srcpos，object dest， int destpos，int length）

  该方法可以将 src 数组里的元素值赋给 dest 数组的元素。各参数说明如下：

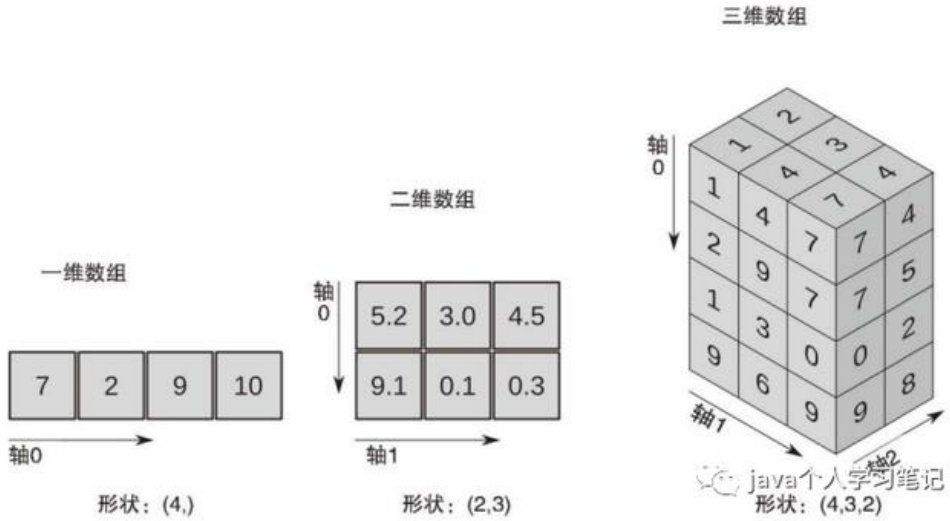



```
public class Test {
    public static void main(String args[]) {
        String[] s = {"阿里","腾讯","京东","搜狐","网易"};
        String[] sBak = new String[6];
        System.arraycopy(s,0,sBak,0,s.length);
        for (int i = 0; i < sBak.length; i++) {
            System.out.print(sBak[i]+ "\t");
        }
    }
}
```

- **多维数组**

  多维数组可以看作是以数组为元素的数组。有二维、三维、甚至更多维数组，但是实际开发中用的非常少，最多到二维数组。

  二维数组，可以看作一个一维数组，每一个元素都还是一个数组。

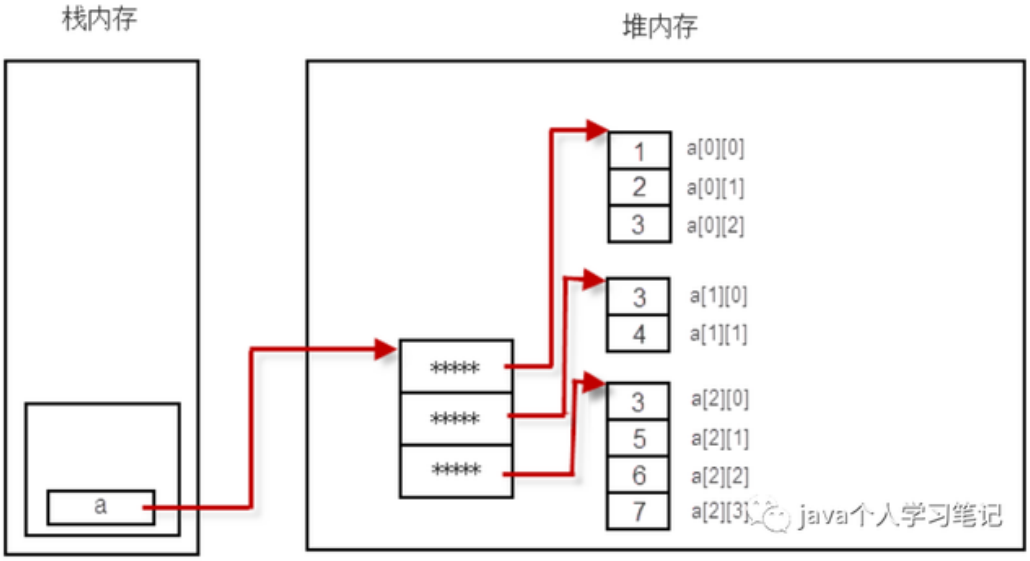


- **二维数组的声明**

```
public class Test {
    public static void main(String[] args) {
        //多维数组的声明和初始化应按从低维到高维的顺序进行
        int[][] a = new int[3][]; //创建了一个二维数组，其第一维度为 3 个元素
        a[0] = new int[2];  //第一个元素被创建成 2 个 int 的数组
        a[1] = new int[4];
        a[2] = new int[3];
    }
}
```

静态初始化

```
public class Test {
    public static void main(String[] args) {
        int[][] a = {{1, 2, 3}, {3, 4}, {3, 5, 6, 7}}; //声明的时候直接初始化二维数组
        System.out.println(a[2][3]);  //通过二维下标访问二维数组的元素
    }
}
```



动态初始化

```
import java.util.Arrays;
public class Test {
    public static void main(String[] args) {
        int[][] a = new int[3][];
        // a[0] = {1,2,5}; //本句是错误，没有声明类型就初始化
        a[0] = new int[] {1, 2};
        a[1] = new int[] {2, 2};
        a[2] = new int[] {2, 2, 3, 4};
        System.out.println(a[2][3]);
        System.out.println(Arrays.toString(a[0]));
        System.out.println(Arrays.toString(a[1]));
        System.out.println(Arrays.toString(a[2]));
    }
}
```