数组

目录

Contents

数组是什么

- 数组就是用来存储一批同种类型数据的容器。

例子

20, 10, 80, 60, 90

```
int[] arr = {20, 10, 80, 60, 90};
```

张三, 李四, 西门

```
String[] names = {"张三", "李四", "王五"};
```

静态初始化数组

- 定义数组的时候直接给数组赋值。

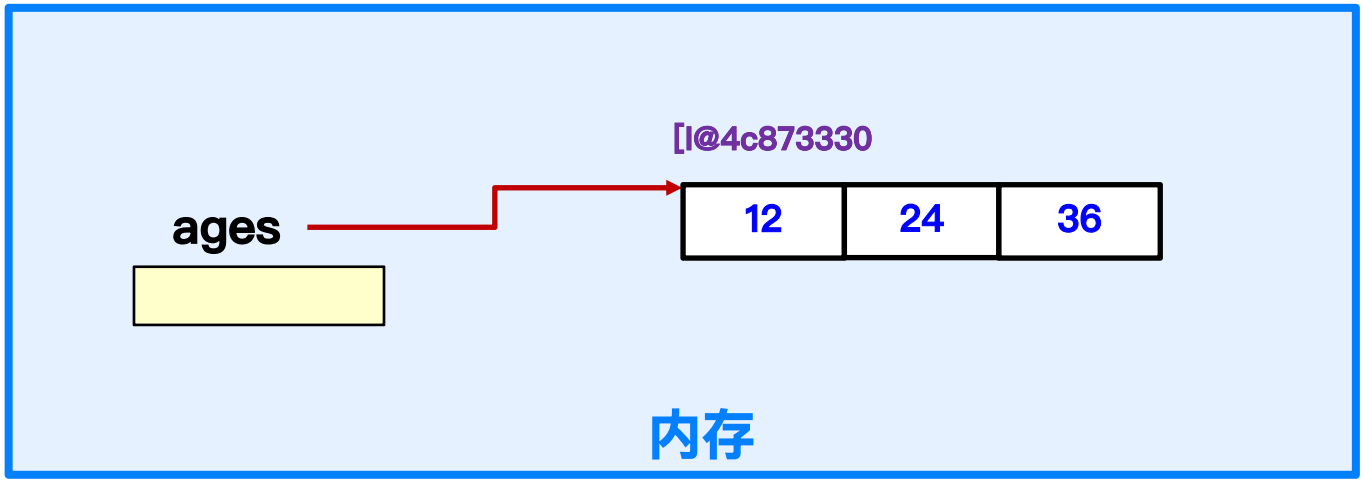
静态初始化数组的格式：

```
// 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... };
double[] scores = new double[]{89.9, 99.5, 59.5, 88.0};
int[] ages = new int[]{12, 24, 36}
```

```
// 简化格式
数据类型[] 数组名 = { 元素1，元素2 ，元素3，... };
int[] ages = {12, 24, 36};
```

数组的基本原理

```
int[] ages = {12, 24, 36};
```



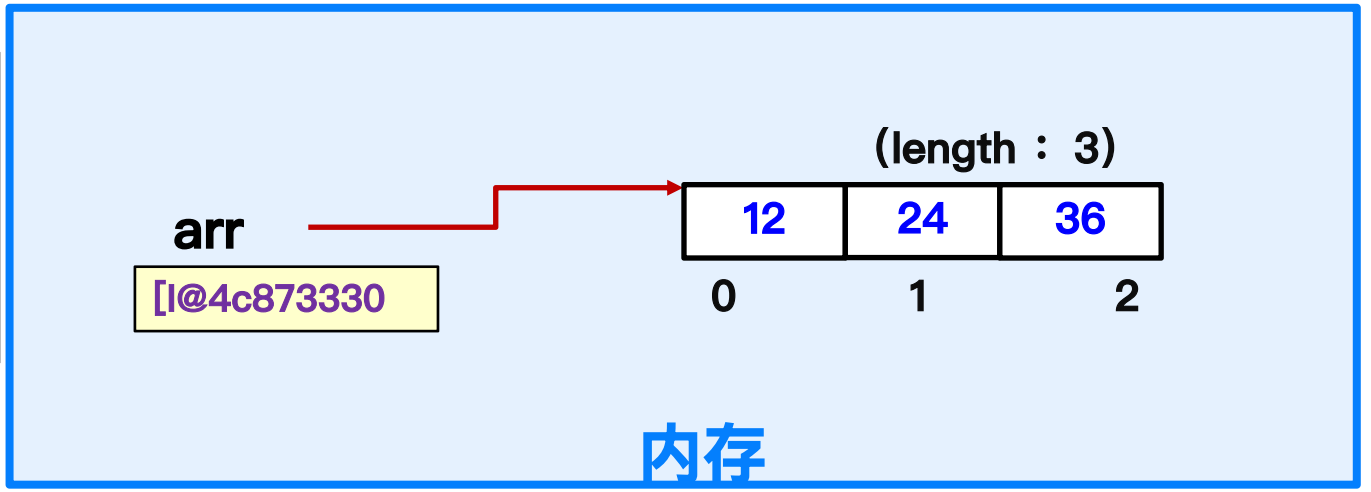
注意：数组变量名中存储的是数组在内存中的地址，数组是引用类型。

数组的访问

数组名称[索引]

```
// 取值
System.out.println(arr[0]); // 12
// 赋值
arr[2] = 100;
System.out.println(arr[2]); // 100
```

int[] arr = {12, 24, 36};



数组的长度属性：length

```
// 获取数组的长度（就是数组元素的个数）
System.out.println(arr.length); // 3
```

问题：数组的最大索引可以怎么表示?

数组名. length − 1 // 前提：元素个数大于0

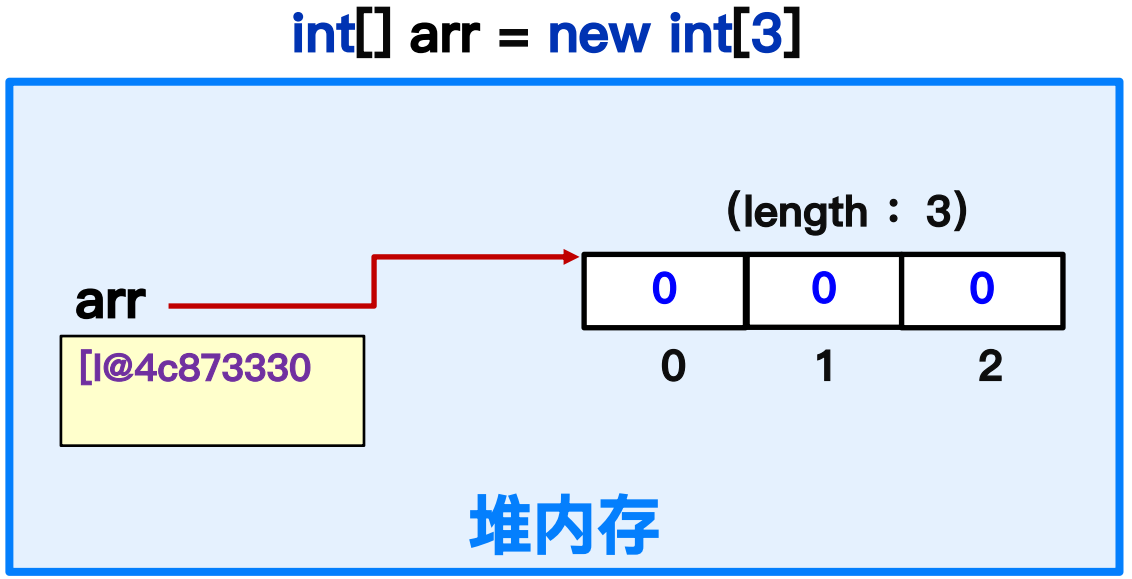
数组的动态初始化

- 定义数组的时候只确定元素的类型和数组的长度，之后再存入具体数据。

数组的动态初始化格式：

```
数据类型[]  数组名 = new 数据类型[长度];
int[] arr = new int[3];
```

```
// 后赋值
arr[0] = 10;
System.out.println(arr[0]); // 10
```



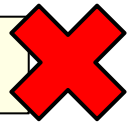
元素默认值规则：

数据类型	明细	默认值
基本类型	byte、short、char、int、long	0
	float、double	0.0
	boolean	false
引用类型	类、接口、数组、String	null

两种初始化的的使用场景总结、注意事项说明：

- 动态初始化：只指定数组长度，后期赋值，适合开始知道数据的数量，但是不确定具体元素值的情况。
- 静态初始化：开始就存入元素值，适合一开始就能确定元素值的情况。
- 两种格式的写法是独立的，不可以混用。

```
int[] arrs = new int[3]{30，40,50};
```



数组的几个注意事项：

- “数据类型[] 数组名”也可以写成“数据类型 数组名[] ”。

```
int[] ages =...;
int ages[] =...;
```

```
double[] scores = ...;
double scores[] = ...;
```

- 什么类型的数组存放什么类型的数据，否则报错。

```
int[] arrs = new int[]{30, 40, "李四"};
```

数组的几个注意事项：

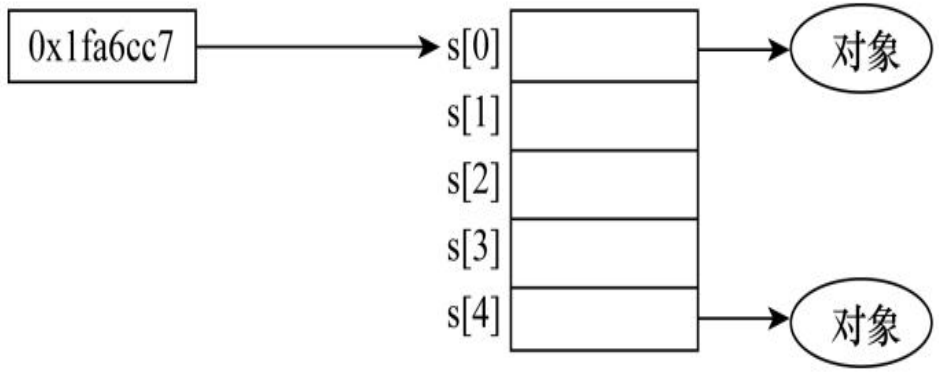
- **数组一旦定义出来，程序执行的过程中，长度、类型就固定了。**
- 数组的每个元素具有相同的数据类型。
- 数组类型可以是任何数据类型，包括基本类型和引用类型。
- 数组变量属于**引用类型**，数组也是**对象**。

也可以开辟**引用类型**数组空间，以Student类为例：

```
Student s[];                //声明数组
s=new Student[5];//创建数组元素
```

或者声明数组的同时创建：

```
Student s[]=new Student[5];
```



使用 new 关键字创建数组元素后，在堆空间开辟了 5 个 Student 型连续的**存储空间**，用来存储 Student 型对象的引用。每个空间仍为引用型。

总结

1、动态初始化的写法是什么样的?

```
数据类型[] 数组名 = new 数据类型[长度];
int[] ages = new int[4];
```

2、两种数组定义时的特点和场景有什么区别

- **当前已经知道存入的元素值，用静态初始化。**
- **当前还不清楚要存入哪些数据，用动态初始化。**

3、动态初始化数组后元素的默认值是什么样的?

- **byte、short、int 、char、long类型数组元素的默认值都是0**
- **float、double类型数组元素的默认值都是0.0**
- **boolean类型数组元素的默认值是false、String类型数组元素的默认值是null**

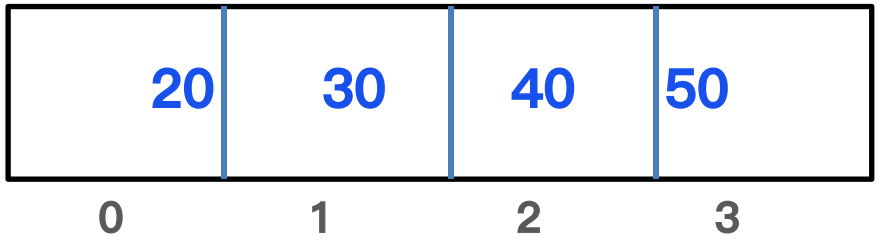
目录

Contents

数组遍历介绍

- 遍历：就是一个一个数据的访问。

```
int[] ages = {20, 30, 40, 50};
for (int i = 0; i < ages.length; i++) {
    System.out.println(ages[i]);
}
```



案例 数组遍历–求和

需求：某部门5名员工的销售额分别是：16、26、36、6、100，请计算出他们部门的总销售额。

分析：

① 把这5个数据拿到程序中去 ---> 使用数组

```
int[] money = {16, 26, 36, 6, 100};
```

② 遍历数组中的每个数据，然后在**外面定义求和变量**把他们累加起来。

```
int sum = 0;
for (int i = 0; i < money.length; i++) {
    // i = 0 1 2 3 4
    sum += money[i];
}
```

目录

Contents

二维数组

二维数组是一种行列结构的数组，可以通过如下方式定义和初始化。

```
int[][] matrix = new int[2][3];      // 定义一个2行3列的二维数组
```

上述代码定义了一个名为matrix的二维数组，该数组有2行3列。

要注意的是，二维数组的每行元素也是一个数组。

二维数组的初始化方式有以下几种。

（1）使用new关键字创建一个初始值为0的二维数组。

```
int[][] matrix = new int[2][3];
```

（2）直接指定二维数组的初始值。

```
int[][] matrix = {{1,2,3}, {4,5,6}};
```

（3）使用new关键字和数组构造器来初始化二维数组。

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

二维数组使用两个下标来访问元素，第1个下标表示行号，第2个下标表示列号，行号与列号都从0开始计数。

```
int[][] matrix = {{1,2,3}, {4,5,6}};

System.out.println(matrix[0][1]);          // 输出2
System.out.println(matrix[1][2]);          // 输出6

matrix[1][0] = 7;                          // 修改第二行第一列的值为7
System.out.println(matrix[1][0]);  // 输出7
```

【例7-2】二维数组的应用

定义一个二维数组，并初始化，最后输出二维数组的所有元素。

```
package ch07;
public class TestTwoArray {
    public static void main(String[] args) {
                int[][] matrix = {{1,2,3}, {4,5,6}};
                for(int i = 0; i < matrix.length; i++){
                    for(int j = 0; j < matrix[i].length; j++){
                        System.out.print(matrix[i][j] + " ");
                    }
                    System.out.println();  // 换行输出下一行
                }
        }
}
```

例7-2中，输出整个二维数组的所有元素，每行之前换行显示。为了保证循环的正确性，需要分别使用数组的length属性获取行数和列数。

```
1 2 3
4 5 6
```