

数组

Java Platform Standard Edition

Java教学部

课程目标

CONTENTS

ITEMS **1** 数组的概念

ITEMS **2** 数组的声明与赋值

ITEMS **3** 数组的下标

ITEMS **4** 数组的遍历

ITEMS **5** 数组的排序

ITEMS **6** 数组的应用

为什么使用数组

- 如何存储100名学生的成绩?
 - 办法：使用变量存储，重复声明100个double类型变量即可。
 - 缺点：麻烦，重复操作过多。
- 如何让100名学生成绩全部+1?
 - 办法：100个变量重复相同操作，直至全部完毕。
 - 缺点：无法进行统一的操作。

数组的概念

- 概念：一组连续的存储空间，存储多个相同数据类型的值。



特点：

1. 类型相同
2. 长度固定

数组的创建

```
public class TestCreateArray {  
    public static void main(String[] args) {  
  
        int[] a = new int[5];  
  
    }  
}
```

声明int数组类型变量
定义变量名为a

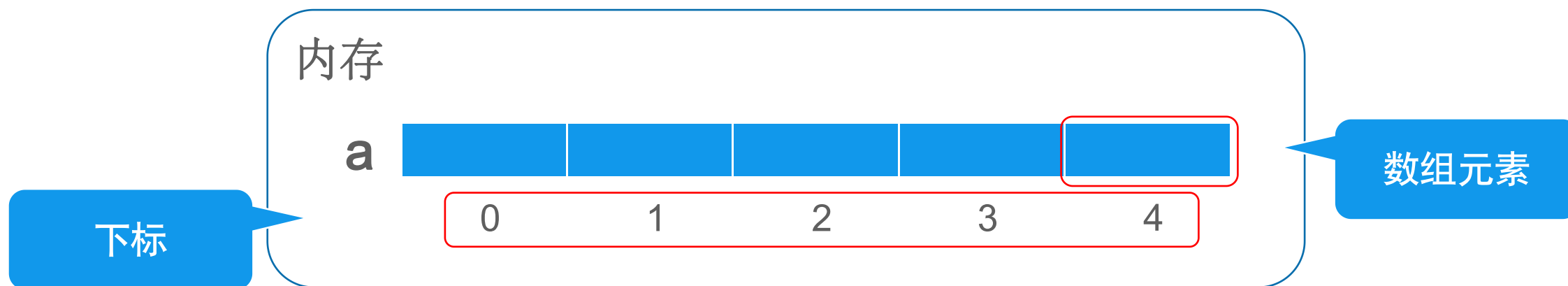
分配长度为5的连续空间

内存



可以存储5个int类型的值

数组的使用



- 数组中的每个数据格被称为“数组元素”。
- 对每个元素进行赋值或取值的操作被称为“元素的访问”。
- 访问元素时，需要使用“下标”（从0开始，依次+1，自动生成）。
- 访问的语法：数组名[下标]; //例如 存：a[0]=10; 取：a[0];

数组的使用

```
public class TestCreateArray {  
    public static void main(String[] args) {  
        int[] a = new int[5];  
        a[0]=5;  
        a[1]=3;  
        a[2]=4;  
        a[3]=7;  
        a[4]=10;  
        System.out.println(a[0]);  
        System.out.println(a[1]);  
        System.out.println(a[2]);  
        System.out.println(a[3]);  
        System.out.println(a[4]);  
    }  
}
```

创建数组

依次赋值

依次取值

运行结果:

5
3
4
7
10

下标的范围

```
public class TestCreateArray {  
    public static void main(String[] args) {  
        int[] a = new int[5];  
        a[0]=5;  
        a[1]=3;  
        a[2]=4;  
        a[3]=7;  
        a[4]=10;  
        System.out.println(a[0]);  
        System.out.println(a[1]);  
        System.out.println(a[2]);  
        System.out.println(a[3]);  
        System.out.println(a[4]);  
        System.out.println(a[5]);  
    }  
}
```

有效下标范围：0 ~ 数组长度-1

访问无效下标，会导致数组下标越界

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5

数组的遍历

- 遍历：从头至尾，逐一对数组的每个元素进行访问。

```
public class TestVisitArray {  
    public static void main(String[] args) {  
        int[] a = new int[5];  
        a[0]=5;  
        a[1]=3;  
        a[2]=4;  
        a[3]=7;  
        a[4]=10;  
        for (int i = 0; i < a.length; i++) {  
            System.out.println(a[i]);  
        }  
    }  
}
```

数组名.length 可动态获得数组长度。

使用循环变量“i”充当下标，
逐一访问数组中的每个元素。

数组的默认值

```
public class TestDefaultValue {  
    public static void main(String[] args) {  
        int[] a = new int[5];  
  
        for (int i = 0; i < 5; i++) {  
            System.out.println(a[i]);  
        }  
    }  
}
```

在没有为数组元素赋值的情况下，
依旧可以正确访问。

运行结果：

0
0
0
0
0

数组默认值：

整数： 0
小数： 0.0
字符： \u0000
布尔： false
其他： null

- 先声明、再分配空间：

数据类型[] 数组名;

数组名 = new 数据类型[长度];

- 声明并分配空间：

数据类型[] 数组名 = new 数据类型[长度];

- 声明并赋值（繁）：

数据类型[] 数组名 = new 数据类型[] {value1,value2,value3,...};

- 声明并赋值（简）：

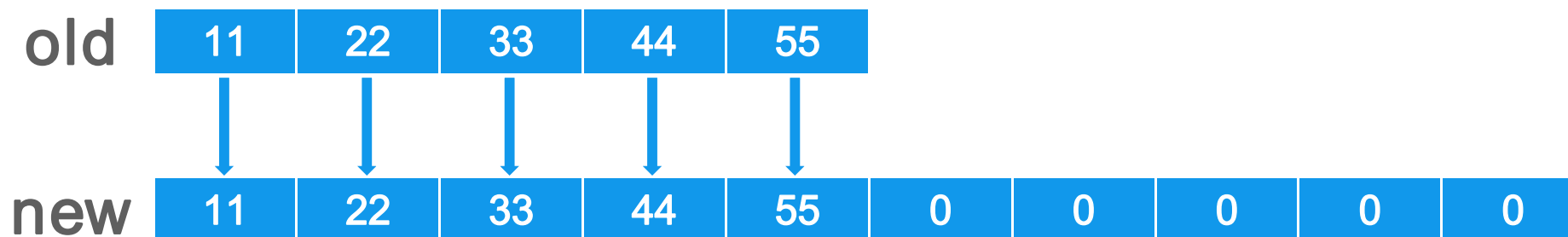
数据类型[] 数组名 = {value1,value2,value3,...}; //显示初始化，注意：不可换行

- 统计int类型数组中所有元素的总和。
- 统计int类型数组中所有元素的平均值。

数组的扩容

- 创建数组时，必须显示指定长度，并在创建之后不可更改长度。
- 扩容的思路：
 - 创建大于原数组长度的新数组。
 - 将原数组中的元素依次复制到新数组中。

内存

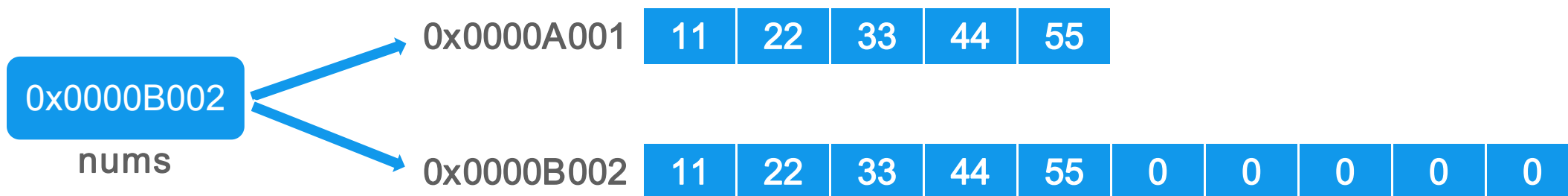


- 循环将原数组中所有元素逐一赋值给新数组。
- `System.arraycopy(原数组, 原数组起始, 新数组, 新数组起始, 长度);`
- `java.util.Arrays.copyOf(原数组, 新长度);`//返回带有原值的新数组。

地址的替换


- 数组作为引用类型之一，其变量中存储的是数组的地址。
- 完成元素复制后，需将新数组地址，赋值给原变量进行替换。

内存



数组类型的参数

```
public class TestArrayParameter {  
    public static void main(String[] args) {  
        int[] nums = {111, 222, 333, 444, 555};  
        printArray(nums);  
    }  
  
    public static void printArray(int[] oneArray) {  
        for (int i = 0; i < oneArray.length; i++) {  
            System.out.println(oneArray[i]);  
        }  
    }  
}
```



假设nums地址为:
0x0000A001

参数传入后
oneArray地址为:
0x0000A001

运行结果:

111
222
333
444
555

- 方法调用时，将nums中的地址赋值给oneArray，此时二者指向同一个数组。

数组类型的返回值

```
public class TedtReturnedValue {  
    public static void main(String[] args) {  
        int[] oa = {111,222,333,444,555};  
        int[] na = expand(oa);  
        for (int i = 0; i < na.length; i++) {  
            System.out.println(na[i]);  
        }  
    }  
  
    public static int[] expand(int[] oldArray){  
        int[] newArray = new int[oldArray.length*2];  
        for (int i = 0; i < oldArray.length; i++) {  
            newArray[i] = oldArray[i];  
        }  
        return newArray;  
    }  
}
```

返回长度为10的
newArray

运行结果:

111
222
333
444
555
0
0
0
0
0

- 创建新数组，长度为原数组的2倍，新数组中保留原有数据，返回新数组。

可变长参数

- 概念：可接收多个同类型实参，个数不限，使用方式与数组相同。
- 语法：数据类型... 形参名 //必须定义在形参列表的最后，且只能有一个。

```
public class TestArrayParameter {  
    public static void main(String[] args) {  
        printArray(111,222,333,444,555);  
    }  
  
    public static void printArray(int... oneArray){  
        for (int i = 0; i < oneArray.length; i++) {  
            System.out.println(oneArray[i]);  
        }  
    }  
}
```

可为可变长参数赋予
0 ~ N 个实际参数

定义int型可变长参数

运行结果：

111
222
333
444
555

- 冒泡排序：相邻的两个数值比较大小，互换位置。
- 选择排序：固定值与其他值依次比较大小，互换位置。
- JDK排序： `java.util.Arrays.sort(数组名);` //JDK提供（升序）

二维数组

- 概念：一维数组中的一维数组；数组中的元素，还是数组。

	A	B	C
1	NAME	AGE	SEX
2	Tom	20	M
3	Jack	21	M
4	Marry	19	F
5	Annie	20	F

当查找excel中的某个单元格时，需要两个下标，
n代表行，m代表列，
二维数组相当于一个多行多列的表格

查找X表中的"B3"单元格
二维数组的语法为：X[3][B]
行下标在前，列下标在后

二维数组的赋值

```
public class Test2DArray {  
    public static void main(String[] args) {  
        int[][] array = new int[3][5];  
        array[0][0] = 10;  
        array[0][3] = 20;  
        array[1][1] = 30;  
        array[1][2] = 40;  
        array[2][4] = 50;  
    }  
}
```

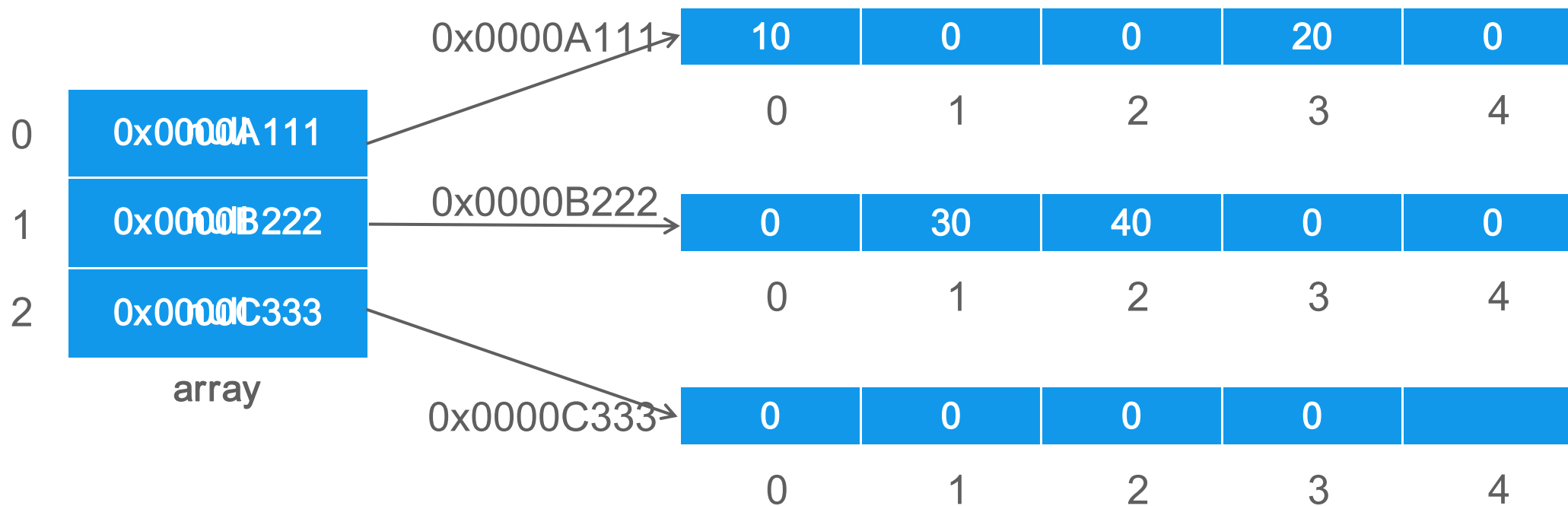
逻辑

0	10	0	0	20	0
1	0	30	40	0	0
2	0	0	0	0	50
	0	1	2	3	4

使用双下标访问二维数组中的元素，
第一个下标代表：行号（高维下标），
第二个下标代表：列号（低维下标）。

二维数组的内存分配

内存



高维数组中的每一个元素，保存了低维数组的地址。访问array[0]等价于在访问0x0000A111

二维数组的访问

```
public class Test2DArray {  
    public static void main(String[] args) {  
        int[][] array = new int[3][5];  
        array[0][0] = 10;  
        array[0][3] = 20;  
        array[1][1] = 30;  
        array[1][2] = 40;  
        array[2][4] = 50;  
        for (int i = 0; i < array.length; i++) {  
            for (int j = 0; j < array[i].length; j++) {  
                System.out.print(array[i][j]);  
            }  
            System.out.println();  
        }  
    }  
}
```

访问低维长度:
array[0].length
首个低维数组的长度

访问低维数组元素:
array[0][0]
首个低维数组的首个元素

二维数组创建语法

- 先声明、再分配空间：

数据类型[] 数组名;

数组名 = new 数据类型[高维长度][低维长度];

- 声明并分配空间：

数据类型[] 数组名 = new 数据类型[高维长度][低维长度];

- 声明并赋值（繁）：

数据类型[] 数组名 = new 数据类型[高维长度][]; //不规则数组，自行new低维数组

- 声明并赋值（简）：

数据类型[] 数组名 = { {v1,v2,v3},{v4,v5},{v6,v7,v8,v9} }; //显示初始化

- 杨辉三角

- 数组的概念：
 - 一组连续的存储空间，存储多个相同数据类型的值。
- 数组的声明与赋值：
 - 数据类型[] 数组名 = new 数据类型[长度];
 - 数组名[下标] = 值;
- 数组的遍历：
 - 从头至尾，逐一对数组的每个元素进行访问。
- 数组的排序：
 - 冒泡排序、选择排序、JDK快速排序。
- 数组的应用：
 - 数组复制、数组扩容、数组参数、数组返回值、二维数组。