

Java程 序 设 计

第7讲：数组、矩阵操作与 Arrays 工具库

- 1 本讲目标
- 2 循环一:数组的真身
- 3 循环二:越界与二维
- 4 循环三:Arrays 工具箱(机动)
- 5 收束:自查与课后

本讲目标:下课前,你应该能做到

三件事

1. **会用**:声明、初始化、遍历数组,下标范围 $[0, N - 1]$ 不错位;
2. **会解释**:直接打印数组为什么出乱码、越界为什么崩;
3. **会借力**:用 Arrays 工具类完成排序、查找、拷贝。

课堂节奏

- 3 个循环,每个循环:讲 10 分钟 → 动手 5 分钟;
- 抢答先自己猜,再验证;实操限时完成,截图发课程群。

开场预测:直接打印数组,输出什么?

【随堂抢答·概念检测】先猜,30 秒后现场验证

```
int[] arr = {1, 2, 3};  
System.out.println(arr);
```

- A. [1, 2, 3]
- B. 类似 [I@15db9742 的乱码
- C. 编译报错

两种初始化:先占格子 vs 一次装满

动态初始化:给长度

```
int[] scores = new int[5];  
double[] temps = new double[10];  
// 元素先拿默认值: 0 / 0.0 / false / null
```

静态初始化:给内容

```
int[] primes = {2, 3, 5, 7, 11};  
String[] names = {"A", "B"};  
// 长度由元素个数自动推断
```

一个数组只装一种类型,装多少在创建时就焊死了。

内存真身:连续的一排格子



- 上讲悬念兑现: **下标从 0 开始**, 因为下标是 **偏移量**—— $\text{arr}[i]$ 的地址 = 首地址 + $i \times$ 元素大小;
- 连续内存 $\rightarrow$ 任意下标一步到位, 随机访问 $O(1)$ 。

两种遍历:要下标用 for,只读用 for-each

```
int[] data = {10, 20, 30, 40};

for (int i = 0; i < data.length; i++) {    // 要下标/要写
    System.out.printf("data[%d] = %d%n", i, data[i]);
}

for (int val : data) {                      // 只读,最简
    System.out.println("元素: " + val);
}
```

抢答 1: new 出来的数组里是什么?

【随堂抢答·概念检测】`int[] a = new int[3]; boolean[] b = new boolean[2];` 之后 `a[0]`、`b[0]` 是?

- A. 0 和 false
- B. null 和 null
- C. 0 和 true
- D. 随机垃圾值

【知识要点】默认值表

- JVM 给堆里的新数组**自动零值初始化**:数值 0、浮点 0.0、boolean false、引用 null;
- 对比 C 语言:局部数组不初始化是脏值——Java 替你兜了底;
- 开场谜底预告:知道数组在堆里,就能猜到 `println(arr)` 打的是什么——**下一页揭晓**。

开场谜底:打印出来的是“地址”

```
System.out.println(arr);           // [I@15db9742  
System.out.println(Arrays.toString(arr)); // [1, 2, 3]
```

- [I@...: [I = 整型数组,后面是十六进制**哈希地址**——呼应第 6 讲:数组变量存的就是地址;
- 想看内容,永远用 `Arrays.toString(arr)`。

实操 1: 班级成绩统计分析器

【随堂实操·动手练笔】ScoreStatistics • 限时 5 分钟

编写 ScoreStatistics.java:

1. 定义 10 个成绩的数组(自己编数据);
2. 遍历求最高、最低、平均分与及格率;
3. Arrays.sort 后输出中位数。

完成标志:四个统计量 + 中位数都打出来 → 截图发课程群。

找茬 1: 只多写了一个等号

【随堂找茬·避坑挑错】这段遍历为什么运行时崩？

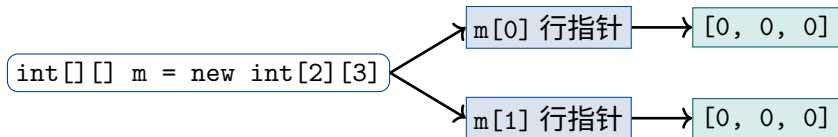
```
int[] arr = new int[5];  
for (int i = 0; i <= arr.length; i++) {  
    System.out.println(arr[i]);  
}
```

- A. 正常打 5 个
- B. 抛 `ArrayIndexOutOfBoundsException`
- C. 打 6 个 0
- D. 编译报错

【知识要点】边界法则

- `length = 5`, 合法下标 `[0, 4]`; `i = 5` 时 → `ArrayIndexOutOfBoundsException`;
- `<=` 是越界惯犯: 循环条件永远写 `i < arr.length`;
- 对比第 5 讲的缓冲区思维: 数组就是定长缓冲区, 越界就是读到了别人的地盘。

二维数组:数组的数组



- 本质: **一维数组的数组**——外层装的是每行的地址;
- 既然每行是独立数组,就能不等长:`new int[3][]` 再逐行分配(三角矩阵)。

矩阵转置:行列互换

```
public static int[][] transpose(int[][] m) {  
    int rows = m.length, cols = m[0].length;  
    int[][] t = new int[cols][rows];  
    for (int i = 0; i < rows; i++) {  
        for (int j = 0; j < cols; j++) {  
            t[j][i] = m[i][j];    // 行列坐标互换  
        }  
    }  
    return t;  
}
```

【随堂实操·动手练笔】MatrixTransposeDemo · 限时 5 分钟

编写 MatrixTransposeDemo.java:

1. 初始化 3×4 整数矩阵;
2. 用转置算法生成 4×3 并逐行打印。

完成标志: 打印出 4 行 3 列且元素行列互换正确 → 截图发课程群。

Arrays: 数组操作的瑞士军刀

方法	用途
<code>toString(arr)</code>	打印成 [1, 2, 3]
<code>sort(arr)</code>	原地升序排序(双轴快排)
<code>binarySearch(arr, k)</code>	二分查找, $O(\log n)$
<code>copyOf(arr, len)</code>	扩容 / 截断拷贝
<code>fill(arr, val)</code>	整组填充

- 底层搬运更快的是 `System.arraycopy(src, srcPos, dest, destPos, len)`——JVM 直连内存拷贝；
- `binarySearch` 的前提: 数组必须已升序, 乱序结果不可预期。

抢答 2: arraycopy 五个参数谁是谁?

【随堂抢答·概念检测】`System.arraycopy(src, 1, dest, 2, 3)`——数据落到 `dest` 的起始下标是?

- A. 0
- B. 1
- C. 2
- D. 3

【知识要点】五参数顺序:源、源起点、目标、目标起点、长度

- (src, 1, dest, 2, 3):从 src[1] 起取 3 个,放进 dest[2..4];
- 记法:源在前,目标在后;各自带起点;最后是长度;
- 越界同理:起点 + 长度不得超过各自数组长度。

1. **会用**:统计器与转置你独立跑通了吗?
2. **会解释**: [I@15db9742 是什么? 为什么 `<= length` 会崩?
3. **会借力**:Arrays 五件套能报出几个?

没做到的那条,就是课后任务的第一优先级。

工程视野

连续内存带来 CPU 缓存友好——高性能网络收发缓冲区几乎都是数组,紧凑就是吞吐率。

1. **练习**: 有序数组上用 `binarySearch` 定位目标值, 并亲手制造一次“乱序数组查找”观察错误结果;
2. **AI 实战**: 把你的统计器发给 AI, 请它专查数组越界与空数组风险。

下讲预告: 数组定长装不下怎么办? ——排序算法与 `ArrayList`。

在线主页与课程资源

■ 个人主页:

<https://matnoble.top>

■ 课程专栏:

<https://matnoble.top/courses/>

■ GitHub 仓库:

<https://github.com/MatNoble>

自由交流与提问 (Q&A)

“学贵善疑,逻辑筑基”

感谢大家的专注聆听!

课件最新版本与补充资料将持续同步至课程主页。

现在进入自由提问与交流时间。