Java程 序 设 计

第 7 讲： 数组、 矩阵操作与 Arrays 工具库

厦门大学嘉庚学院

XIAMEN UNIVERSITY TAN KAH KEE COLLEGE

CONTENTS

数组声明与两种初始化方式

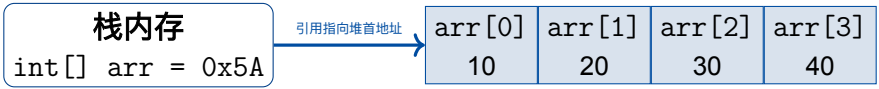
动态初始化 (指定长度)

```java
// 系统为各元素赋予默认初值 (0 / 0.0 / false / null)
int[] scores = new int[5];
double[] temps = new double[10];
```

静态初始化 (直接指定元素)

```java
// 由编译器自动根据元素个数推断长度
int[] primes = {2, 3, 5, 7, 11};
String[] names = new String[]{"A", "B"};
```

数组物理内存连续分配模型



【知识要点】核心物理特性

数组在堆中占据一段**连续物理内存空间**；支持基于下标索引的 $O(1)$ 时间复杂度极速随机访问！

传统 for 循环 vs 增强型 for-each

【代码演练】两种遍历对比 (ArrayMemoryDemo.java)

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

// 方式 1: 传统索引下标遍历 (支持读写与按索引操作)
for (int i = 0; i < data.length; i++) {
    System.out.printf("索引 [%d] = %d\n", i, data[i]);
}

// 方式 2: 增强型 for-each 循环 (语法极其简洁, 适合只读遍历)
for (int val : data) {
    System.out.println("元素值: " + val);
}
```

随堂找茬 1：ArrayIndexOutOfBoundsException 异常

【随堂找茬·避坑挑错】看看以下代码为什么会在运行时发生崩溃？

```java
int[] arr = new int[5]; // 合法下标为 0, 1, 2, 3, 4

// 致命 Bug：循环条件写成了 i <= arr.length
for (int i = 0; i <= arr.length; i++) {
    System.out.println(arr[i]); // 当 i=5 时抛出 ArrayIndexOutOfBoundsException!
}
```

【知识要点】避坑指南

数组长度为 N 时，最大有效下标为 $N-1$；循环条件必须写为 `i < arr.length`!

随堂抢答 1：动态初始化默认值预测

【随堂抢答·概念检测】问题：执行 int[] a = new int[3]; boolean[] b = new boolean[2]; 后，a[0] 与 b[0] 分别是？

A. `a[0] = 0, b[0] = false`

B. `a[0] = null, b[0] = null`

C. `a[0] = 0, b[0] = true`

D. 内存未初始化，包含随机垃圾值

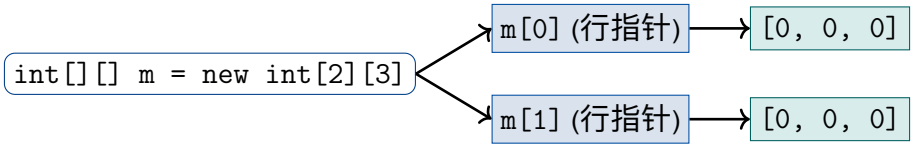
随堂抢答 1：答案与深度解析

【知识要点】正确答案：【A 】

原理解析：

- Java 堆内存在分配时会自动进行零值初始化；
- 数值型（byte, short, int, long）默认初始化为 `0`；
- 浮点型默认 `0.0`，布尔型默认 `false`，引用类型默认 `null`。

二维数组的物理本质:数组的数组



【代码演练】矩阵转置核心实现 (MatrixTransposeDemo.java)

```java
public static int[][] transpose(int[][] matrix) {
    int rows = matrix.length;
    int cols = matrix[0].length;
    int[][] transposed = new int[cols][rows]; // 行列互换
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            transposed[j][i] = matrix[i][j]; // 元素行列坐标交换
        }
    }
    return transposed;
}
```

java.util.Arrays 工具类常用 API

工具方法	方法功能说明	示例用法
`Arrays.toString(arr)`	将数组格式化为字符串	`"[1, 2, 3]"` 打印输出
`Arrays.sort(arr)`	高性能**双轴快速排序**	`Arrays.sort(scores)`
`Arrays.binarySearch(arr, k)`	有序数组**二分查找** ($O(\log n)$)	查找目标值索引下标
`Arrays.copyOf(arr, newLen)`	数组扩容或截断拷贝	数组容量动态调整
`Arrays.fill(arr, val)`	将指定值填充覆盖整个数组	快速初始化数组

随堂找茬 2：直接打印数组输出哈希地址

【随堂找茬·避坑挑错】看看以下代码输出为什么是一串乱码？

```
1 int[] arr = {1, 2, 3};
2 System.out.println(arr); // 输出：[I@15db9742（类型标志+哈希码!）
```

【知识要点】正确做法

必须调用 `Arrays.toString(arr)` 将数组转换为可读的字符串形式输出："[1, 2, 3]"！

随堂实操练笔 1：班级成绩统计分析器

【随堂实操·动手练笔】任务 1：编写 ScoreStatistics.java

编写 `ScoreStatistics.java`，实现：

1. 定义一个包含 10 个学生成绩的数组；
2. 遍历数组计算出班级最高分、最低分、平均分与及格率；
3. 调用 `Arrays.sort()` 升序排列并输出中位数。

随堂实操练笔 2：二维图像矩阵转置

【随堂实操·动手练笔】任务 2：编写 MatrixTransposeDemo.java

编写 `MatrixTransposeDemo.java`，实现：

1. 初始化一个 3×4 的二维整数矩阵；
2. 实现转置算法生成 4×3 矩阵并格式化打印输出；
3. 验证转置前后矩阵行数与列数的对称性。

工程视野与缓存局部性

【工程视野】内存连续性与硬件亲和

连续数组内存具极佳 CPU 缓存局部性（Cache Locality）；在高性能网络传输中，紧凑内存是提升吞吐率的关键。

【实战技巧】代码审查与辅助开发

实战建议：借助 AI 生成矩阵运算与快速傅里叶变换（FFT）模板，优化循环遍历顺序提升 Cache 命中率。

【随堂实操与课后思考】课后思考与巩固

- **练习**：编写基于二分查找的数组快速定位程序。
- **思考**：原生数组定长不可变，通信突发海量数据时有何痛点？

System.arraycopy 底层极速内存拷贝

【知识要点】JVM 本地系统调用 (JNI C++ 内存拷贝)

- 语法：`System.arraycopy(src, srcPos, dest, destPos, length);`
- 直接通过底层 CPU 块内存搬移指令执行，性能远超 Java 层的 for 循环逐个拷贝！

```
1 int[] src = {10, 20, 30, 40, 50};
2 int[] dest = new int[5];
3 System.arraycopy(src, 0, dest, 0, src.length);
```

不规则二维数组 (Ragged Array) 内存分配

【代码演练】每行具有不同长度的二维数组

```java
int[][] triangle = new int[3][]; // 仅指定行数
triangle[0] = new int[1]; // 第 0 行 1 列
triangle[1] = new int[2]; // 第 1 行 2 列
triangle[2] = new int[3]; // 第 2 行 3 列
```

随堂抢答 3：System.arraycopy 参数顺序

【随堂抢答·概念检测】问题：System.arraycopy(src, 1, dest, 2, 3) 表示从 src 下标 1 开始拷贝 3 个元素到 dest 的哪个起始下标?

A. 0

B. 1

C. 2

D. 3

随堂抢答 3:答案与深度解析

【知识要点】正确答案:【C 】

原理解析:

- 参数 1:源数组;参数 2:源数组起始位置;
- 参数 3:目标数组;参数 4:目标数组起始位置(即 2);
- 参数 5:拷贝长度(即 3 个元素)。

随堂抢答 4：二分查找前提条件

【随堂抢答·概念检测】问题：调用 Arrays.binarySearch(arr, key) 进行快速二分查找的前提条件是？

A. 数组长度必须是 2 的整数次幂
B. 数组必须已经按照升序排好序
C. 数组元素必须全为正数
D. 无任何前提条件

随堂抢答 4：答案与深度解析

【知识要点】正确答案：【B 】

原理解析：

- 二分查找基于区间单调性；
- 若在无序数组中调用二分查找，查找结果将具有不可预期的错误与不确定性！

课程互动与讲师主页

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`

思考：Questions & Answers

“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。