

Java程 序 设 计

第 8 讲：经典排序算法与 ArrayList 动态列表

- 1 本讲目标
- 2 循环一:排序三兄弟
- 3 循环二:ArrayList——会长大的数组
- 4 循环三:著名陷阱(找茬专场)
- 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. 会排:手写冒泡排序,说得出每一轮过后数组长什么样;
2. 会装:用 ArrayList 增删改查,装对包装类;
3. 会避坑:遍历中删除元素的正确姿势。

课堂节奏

- 3 个循环,每个循环:讲 10 分钟 → 动手 6 分钟;
- 抢答先自己猜,再验证;实操限时完成,截图发课程群。

【随堂抢答·概念检测】数组 {5, 2, 8, 1} 走完第一轮冒泡(相邻比较、大数右移)后是?

- A. {1, 2, 5, 8}
- B. {2, 5, 1, 8}
- C. {2, 1, 5, 8}
- D. {5, 2, 8, 1} 不变

逐对比较:5 和 2、5 和 8、8 和 1。

【知识要点】冒泡一轮回放

- 5 比 2 大 → 换: {2, 5, 8, 1}; 5 比 8 小 → 不动; 8 比 1 大 → 换: {2, 5, 1, 8};
- 一轮的成果 = 最大值沉到末尾, 下一轮少看一格;
- 提速窍门: 整轮无交换 = 已有序, swapped 标志提前收工 (最好 $O(n)$)。

冒泡排序:代码不到十行

```
public static void bubbleSort(int[] arr) {  
    for (int i = 0; i < arr.length - 1; i++) {  
        boolean swapped = false;  
        for (int j = 0; j < arr.length - 1 - i; j++) {  
            if (arr[j] > arr[j + 1]) {  
                int t = arr[j]; arr[j] = arr[j+1]; arr[j+1] = t;  
                swapped = true;  
            }  
        }  
        if (!swapped) break;    // 已有序,提前收工  
    }  
}
```

选择排序:挑最小放前面

```
for (int i = 0; i < arr.length-1; i++) {  
    int minIdx = i;  
    for (int j = i+1; j < arr.length; j++)  
        if (arr[j] < arr[minIdx]) minIdx = j;  
    int t = arr[i]; arr[i] = arr[minIdx];  
    arr[minIdx] = t;  
}
```

平均都是 $O(n^2)$; 工业界用 `Arrays.sort()` (双轴快排 $O(n \log n)$)。

插入排序:摸牌插位

- 像抓扑克牌:新牌从右往左找位置插进去;
- 数据近乎有序时接近 $O(n)$, 实战最快的小规模排序。

【随堂实操·动手练笔】BubbleSortDemo · 限时 6 分钟

编写 BubbleSortDemo.java:

1. 生成 10 个随机整数数组, 打印排序前内容;
2. 优化版冒泡升序排序, 统计比较次数与交换次数;
3. 打印排序后内容。

完成标志: 前后两组数据 + 两个次数 → 截图发课程群。

为什么要它:数组的两个痛点

原生数组

- 长度焊死,装不下也不能缩;
- 插入 / 删除要自己搬后面的元素。

ArrayList

- 底层还是数组,但满了自动扩容;
- add / remove / get 帮你搬;
- 泛型 ArrayList<T> 锁定元素类型。

底层真相:扩容 = 新建更长的数组 + 整体搬移,默认扩为 1.5 倍($\text{old} + (\text{old} \gg 1)$)。

方法	功能	复杂度
<code>add(e) / add(i, e)</code>	尾部追加 / 中间插入	$O(1) / O(n)$
<code>get(i) / set(i, e)</code>	读 / 改指定位置	$O(1)$
<code>remove(i)</code>	删除指定位置(后续前移)	$O(n)$
<code>size()</code>	当前元素个数	$O(1)$

- 排序交给工具类:`Collections.sort(list)` 升序,加 `Collections.reverseOrder()` 降序。

装箱规则:集合里不收基本类型

```
ArrayList<int> bad = new ArrayList<>(); // 编译报错!  
ArrayList<Integer> good = new ArrayList<>();  
good.add(95); // 自动装箱: int -> Integer  
int val = good.get(0); // 自动拆箱: Integer -> int
```

- 泛型只收引用类型:int 要用 Integer,double 要用 Double;
- 装、拆都是编译器自动完成,写起来和基本类型一样顺手。

抢答 1: List.of 出来的列表, 还能加吗?

【随堂抢答·概念检测】List.of("A", "B") 创建后调用 add("C") 会?

- A. 成功添加
- B. 抛 UnsupportedOperationException
- C. 静默忽略
- D. 编译报错

【知识要点】不可变集合

- `List.of(...)` 造的是不可修改列表:增删改一律运行时异常;
- 要继续增删,套一层:`new ArrayList<>(List.of(...))`;
- 用途:常量配置表、固定选项——改不了正是它的安全价值。

实操 2: 班级成绩排行榜

【随堂实操·动手练笔】StudentRosterManager • 限时 6 分钟

编写 StudentRosterManager.java:

1. 用 `new ArrayList<>(List.of(...))` 装入 8 个成绩;
2. `Collections.sort(list, Collections.reverseOrder())` 降序;
3. 打印第 1 名到第 N 名榜单(含名次)。

完成标志: 降序榜单且第一名正确 → 截图发课程群。

【随堂找茬·避坑挑错】批量剔除所有"B",运行会发生什么?

```
ArrayList<String> names =  
    new ArrayList<>(List.of("A", "B", "B", "C"));  
for (String s : names) {  
    if ("B".equals(s)) names.remove(s);  
}
```

- A. 删干净了
- B. 抛 ConcurrentModificationException
- C. 只删一个 B
- D. 编译报错

【知识要点】根因与两种标准解法

- for-each 底层是**迭代器**在计数,集合自己 remove 打乱了它的账本 → `ConcurrentModificationException`;
- 解法一(现代):`names.removeIf("B"::equals)`; 一行搞定;
- 解法二(经典):迭代器遍历,用 `it.remove()` 删——让迭代器自己动手。

1. **会排**: {2, 5, 1, 8} 那道手推题, 现在闭眼能走两轮吗?
2. **会装**: `ArrayList<int>` 错在哪? `List.of` 为什么不能加?
3. **会避坑**: 边遍历边删, 两种解法说得出口?

没做到的那条, 就是课后任务的第一优先级。

工程视野

没有“放之四海皆准”的最优算法, 只有场景约束下的权衡——时间、空间、稳定性, 工程决策的本质是取舍。

1. **练习**: 实现插入排序, 用“近乎有序”与“完全乱序”两组数据对比耗时;
2. **AI 实战**: 让 AI 用 Mermaid 画出冒泡一轮的交换过程图, 对照你手推的轨迹。

下讲预告: 语法篇收官——从下一讲起, 进入面向对象的世界。

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。