

Java程 序 设 计

第6讲： 方法——定义、 值传递与重载

- 1 本讲目标
- 2 循环一:方法与调用
- 3 循环二:参数传递(本讲核心)
- 4 循环三:方法重载
- 5 循环四:递归与栈溢出(机动)
- 6 收束:自查与课后

本讲目标:下课前,你应该能做到

三件事

1. **会写**:定义并调用一个方法,说清返回值类型与形参;
2. **会解释**:为什么 swap 换不动两个 int,却换得动数组元素;
3. **会重载**:用同一个名字写一组 add 方法,说清什么算重载。

课堂节奏

- 4 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。

一段逻辑要用三次,怎么办?

复制粘贴三份

- 改一处,漏两处;
- 变量还可能重名冲突;
- 逻辑越复杂,坑越多。

抽成方法,写一次用 N 次

```
static void printStars(int n) {  
    for (int i = 1; i <= n; i++) {  
        System.out.print("*");  
    }  
    System.out.println();  
}  
  
printStars(5);  printStars(3);
```

方法定义:五要素一张图

定义

```
//      修饰符    返回值    方法名(形参列表)
public static int sum(int a, int b) {
    int s = a + b;    // 方法体
    return s;         // 返回值
}
```

调用

```
int r = sum(10, 20);

System.out.println(
    sum(1, 2));
```

- 无返回值 → 写 `void`;形参是占位的局部变量,调用时按位置接收实参;
- 返回值类型必须与 `return` 的数据一致。

【随堂实操·动手练笔】`max(int, int)` · 限时 3 分钟

新建 `MethodLab1.java`:

1. 写方法 `max(int a, int b)`, 返回较大者;
2. 在 `main` 中调用 `max(13, 27)` 并打印。

完成标志: 控制台输出 `max = 27` → 截图发课程群, 举手。

抢答 1: swap 能换成功吗?

【随堂抢答·概念检测】调用 swap(x, y) 后,输出是?

```
static void swap(int a, int b) {  
    int t = a;  a = b;  b = t;  
}  
public static void main(String[] args) {  
    int x = 1, y = 2;  
    swap(x, y);  
    System.out.println(x + " " + y);  
}
```

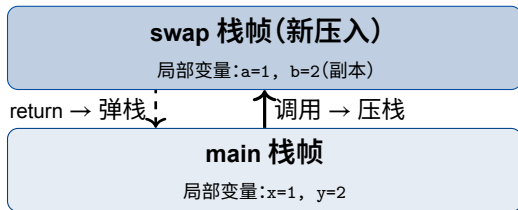
- A. 2 1(交换成功)
- B. 1 2(交换失败)
- C. 编译报错
- D. 运行时异常

先独立写下答案——30 秒后现场验证。

【知识要点】方法拿到的是副本

- 输出 1 2:swap 内部确实交换了——但换的是 a、b;
- a、b 是 x、y 的副本,方法一结束就销毁;
- 悬念:为什么换不掉? 看完“栈帧”,你要能自己说清。

看底层:方法调用的栈帧(Stack Frame)



- 每次调用压入一个新栈帧,return 即弹栈销毁;
- 形参 a、b 住在 swap 自己的栈帧里,与 x、y 是两份内存。

定论:Java 只有一种传参——值传递

一、基本类型:传数值副本

- 复制一份数值给形参;
- 方法内改形参,外部实参不变;
- 这就是 `swap(int, int)` 失败的原因。

一句话:数值是副本,地址也是副本——改副本不影响原值,顺着地址改对象才影响。

二、引用类型:传地址副本

- 复制一份地址给形参;
- 两个地址指向同一个堆对象;
- 通过地址改对象内容,外部看得见。

抢答 2:升级版——数组也换不动吗?

【随堂抢答·概念检测】输出是? 先想 10 秒:这次谁动了“地址”?

```
static void change(int x, int[] arr) {  
    x = 999;  
    arr[0] = 999;  
}  
  
public static void main(String[] args) {  
    int num = 10;  
    int[] list = {10, 20};  
    change(num, list);  
    System.out.printf("num=%d, list[0]=%d\n", num, list[0]);  
}
```

- A. num=10, list[0]=10
- B. num=999, list[0]=999
- C. num=10, list[0]=999
- D. num=999, list[0]=10

【知识要点】副本的两种命运

- num:int 副本,方法里改的是副本,原值 10 纹丝不动;
- list:**地址副本**,与实参指向同一个堆数组,arr[0]=999 改的是堆里那块内存 → 外部同步可见;
- 记住这句:改副本无效,改对象有效。

【随堂实操·动手练笔】ValueSwapDemo · 限时 5 分钟

新建 MethodLab2.java:

1. `swap(int[] arr)`: 交换 `arr[0]` 与 `arr[1]`, main 验证**成功**;
2. `swap(int a, int b)`: 交换两个 `int`, 验证**失败**。

完成标志: 两行输出——数组版换成功、int 版换失败 → 截图发课程群。

为什么 println 什么都能打印?

println 有 10 多个重载版本: int、double、char、String……同一个名字,按参数自动匹配。

重载(Overload)判定:同名 + 形参列表不同

写法

参数个数不同 add(int,int) / add(int,int,int)

参数类型不同 add(int,int) / add(double,double)

类型顺序不同 f(int,double) / f(double,int)

仅返回值不同

是否构成重载

✓

✓

✓

✗ 编译报错

找茬 1:这两个方法为什么报错?

【随堂找茬·避坑挑错】先判断:这段代码能编译通过吗?

```
public class Calculator {  
    public int compute(int a, int b) {  
        return a + b;  
    }  
    public double compute(int a, int b) {  
        return (double)(a + b);  
    }  
}
```

两个方法仅返回值不同——30 秒,说出你的判断。

【知识要点】报错与原理

- 报错: `compute(int, int)` is already defined;
- 调用 `compute(1, 2)` 时,编译器凭参数列表选方法,返回值帮不上忙;
- 结论:重载只看方法名 + 形参列表(个数 / 类型 / 顺序)。

抢答 3: print(5) 会调用谁?

【随堂抢答·概念检测】调用 print(5), 输出是?

```
static void print(int x)      { System.out.println("int: " + x); }  
static void print(double x) { System.out.println("double: " + x); }  
  
public static void main(String[] args) {  
    print(5);  
}
```

- A. int: 5
- B. double: 5.0
- C. 编译歧义报错
- D. 随机调用一个

【知识要点】方法绑定规则

- 5 是 int $\rightarrow$ 存在 `print(int)`, 精确匹配, 直接绑定;
- 没有 int 版本时才自动提升: `int` $\rightarrow$ `double`, 调用 `print(double)`;
- 顺序: 先精确, 再提升, 永不随机。

【随堂实操·动手练笔】MethodLab3 ·限时 7 分钟

新建 MethodLab3.java,写三个重载:

1. `add(int a, int b);`
2. `add(double a, double b);`
3. `add(int a, int b, int c);`

依次调用并打印:`add(1,2)`、`add(1.5,2.5)`、`add(1,2,3)`。

加餐:给 `add` 补一条 Javadoc(`@param` / `@return`)。

完成标志:三行输出 3、4.0、6 → 截图发课程群。

递归:方法调用自己

```
static long factorial(int n) {  
    if (n <= 1) return 1;           // 1. 基线条件: 出口  
    return n * factorial(n - 1);    // 2. 递归推进: 向出口收敛  
}
```

- 两要素缺一不可:基线条件 + 递归推进;
- 每层调用压一个栈帧:factorial(5) 最多同时压 5 层。

找茬 2:这段求和代码会怎样?

【随堂找茬·避坑挑错】调用 `sum(5)`, 会发生什么?

```
static int sum(int n) {  
    return n + sum(n - 1);  
}
```

- A. 返回 15
- B. 返回 0
- C. 运行一阵后崩溃
- D. 编译报错

【知识要点】StackOverflowError 的来历

- 缺基线条件, sum 永远推进: $\text{sum}(5) \rightarrow \text{sum}(4) \rightarrow \dots$ 无限压栈;
- 线程栈默认约 1MB, 瞬间耗尽 $\rightarrow$ **StackOverflowError** 进程崩溃;
- 回扣第 9 页栈帧图: 每个栈帧都占内存, 压栈不是免费的。

加餐:可变长参数(varargs)

```
static int sum(String tag, int... nums) {  
    int total = 0;  
    for (int n : nums) total += n;  
    return total;  
}  
// 调用: sum("A", 1, 2, 3) -> 6
```

- 语法:类型... 参数名,编译器自动打包成数组;
- 铁律:最多一个,且必须是最后一个参数;
- 想一想:若放在中间,编译器怎么知道实参到哪结束?

1. **会写**:max 和三个 add,你独立写出来了吗?
2. **会解释**:swap 之谜——能用“副本 + 栈帧”一句话说清吗?
3. **会重载**:三要素说得出吗?“仅返回值不同”为什么不行?

没做到的那条,就是你课后任务的第一优先级。

工程视野

方法是“分工协作、各司其职”的最小单元:单一职责、高内聚低耦合。

1. **练习**:斐波那契写两版(递归 / 循环),对比 $n = 40$ 时的耗时;
2. **AI 实战**:让 AI 为三个 `add` 重载生成 Javadoc 与 JUnit 测试,人工核对边界(负数、超大数)。

下讲预告:数组——为什么下标从 0 开始?

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。