

第12课

把配方装进方法

方法 调用与复用

从重复指令到清楚的流程

提取方法

定义一个准备步骤



按名调用

main安排顺序



核对结果

重构后行为不变

会把一段指令装进有名字的方法；能从 main 调用方法并解释执行顺序；能识别重复代码并安全重构；最终把下矿准备拆成五个清楚的小技能。

前置背包：变量、条件、循环和第11课的像素房子。方法会整理已有语法，不会凭空召唤新的钻石。

阅读方式：先猜结果，再运行核对；每次只改一个条件。

本章使用 JDK 21，无需预览功能。故事与插画借用方块世界，练习运行在普通 Java 控制台。

开始前：把示例放对地方

本章使用 JDK 21，不需要预览功能。每个普通示例都给出完整程序，按标注的文件名保存。多个示例各放在自己的文件夹，避免辅助类同名。

终端命令：把中文占位符替换成实际名称

```
javac -encoding UTF-8 示例文件名.java  
java 示例类名
```

类名区分大小写。完整挑战代码若分成多段，请按顺序接在同一个文件中，不要为续段另建类。运行核对只列程序输出；需要输入的地方另给输入序列。

学习动作：先写下你的预测，运行后比较，再只改一个条件。标有“故意错误”的实验用于认识 Bug；它们与可直接运行的示例有清楚区分。

01 每次都重抄配方吗

阿方负责出发前检查。他把“检查镐子、装食物、带火把”复制了三遍：白天一遍、夜里一遍、同桌说再来一遍。结果火把数量改了两处，第三处仍停在旧版本。

方法像一张有名字的配方卡。把一组动作写一次，需要用方法名调用。修改规则时只改配方卡，不必追着三只复制粘贴怪物跑。

工作台正在笑重复代码

重复

多处保存同样代码

风险

改漏一处，行为不一致

方法

定义一次，调用多次

带着四个问题出发

- 方法的定义和调用有什么区别？
- 为什么 main 仍然是程序出生点？
- 怎样给方法起一个能看懂的动作名？
- 重构后怎样证明行为没有改变？

世界设定：本课仍使用 Java 21 控制台程序。方法负责组织代码，不会直接修改真实 Minecraft 世界。

02 给一组指令起名字

先找完全相同或概念相同的动作，再把它们搬进方法。方法名用动词开头，例如 packFood、checkArmor、printDivider。

重复代码变成一张配方卡

先记录

保存原来的输出

提取

重复动作移入方法

再比较

调用两次，核对六行

L12Example02.java • 完整独立示例

```
public class L12Example02 {
    static void prepareMining() {
        System.out.println("检查镐子");
        System.out.println("装入 16 个面包");
        System.out.println("带上 64 根火把");
    }
    public static void main(String[] args) {
        prepareMining();
        prepareMining();
    }
}
```

运行核对

检查镐子
装入 16 个面包
带上 64 根火把
检查镐子
装入 16 个面包
带上 64 根火把

重构目标：减少重复，但不改变输出。先记录旧输出，再提取方法，最后逐行比较新旧结果。

03 定义和调用 void 方法

定义方法只是把配方写进书里；调用方法才是真的拿材料开工。下面的类可以直接编译运行。

方法由方法头和方法体组成

static

本例可从main直接调用

void

执行动作，不返回值

方法名

如packFood，描述动作

方法体

花括号内的指令

L12Example03.java • 完整独立示例

```
public class L12Example03 {  
    static void packFood() {  
        System.out.println("装入 16 个面包");  
    }  
    static void checkArmor() {  
        System.out.println("检查护甲耐久");  
    }  
    public static void main(String[] args) {  
        packFood();  
        checkArmor();  
    }  
}
```

运行核对

装入 16 个面包
检查护甲耐久

读法：看到 packFood(); 就读作“调用 packFood 方法”。结尾的圆括号和分号都不能掉队。

04 调用顺序和返回路线

Java 遇到方法调用时，会暂时离开当前位置，进入被调用的方法。方法执行到右花括号后，程序回到调用处的下一行。

main 发出任务 方法完成后返回

main

打印出发

ringBell

调用一次，打印一声
叮

再调用

再次打印叮

继续

回到main，打印下矿

L12Example04.java • 完整独立示例

```
public class L12Example04 {
    static void ringBell() { System.out.println("叮"); }
    public static void main(String[] args) {
        System.out.println("出发");
        ringBell();
        ringBell();
        System.out.println("下矿");
    }
}
```

运行核对

出发
叮
叮
下矿

预测技巧：先只看 main 的调用顺序，再分别展开每个方法。不要一上来把整份代码揉成一团看。

05 一个方法只做一件事

doEverything() 听起来很强，实际像一个塞满腐肉、钻石和十七把木锄的箱子。方法最好完成一个清楚的动作，并用动词描述它。

方法名就是箱子外面的标签

检查装备

checkEquipment

装食物

packFood

选择矿洞

chooseMine

安排顺序

prepareTrip调用三个步骤

L12Example05.java • 完整独立示例

```
public class L12Example05 {
    static void checkEquipment() { System.out.println("检查装备"); }
    static void packFood() { System.out.println("装食物"); }
    static void chooseMine() { System.out.println("选择矿洞"); }
    static void prepareTrip() {
        checkEquipment();
        packFood();
        chooseMine();
    }
    public static void main(String[] args) {
        prepareTrip();
    }
}
```

运行核对

检查装备
装食物
选择矿洞

命名检查

- 方法名用小驼峰，例如 checkArmor。
- 名字应描述动作，不写 method1。
- 方法过长时，找能单独命名的小步骤。

判断标准：如果你不能用一句短话说清方法做什么，它可能承担了太多任务。

06 把像素房子拆成技能

上一课的像素房子可以先做“无参数版”重构：屋顶负责前两行，墙体负责其余五行并在格子内判断门窗。下一课再用参数让同一方法处理不同尺寸。

屋顶与墙体分工，门窗留在正确格子里

printRoof

输出前两行屋顶

printBody

逐格处理墙、门、窗

main

按顺序调用两个方法

核对

输出与第11课一致

06 把像素房子拆成技能（续）

PixelMethods.java • 完整独立示例

```
public class PixelMethods {
    static void printRoof() {
        System.out.println("  ^^^^  ");
        System.out.println(" ^^^^^^ ");
    }
    static void printBody() {
        for (int row = 2; row < 7; row++) {
            for (int col = 0; col < 9; col++) {
                char block = ' ';
                if (col == 4 && row >= 3 && row <= 5) block = 'D';
                else if ((col == 1 || col == 7)
                        && (row == 3 || row == 4)) block = 'O';
                else if (row == 2 || row == 6 || col == 0 || col == 8)
                    block = '#';
                System.out.print(block);
            }
            System.out.println();
        }
    }
    public static void main(String[] args) {
        printRoof();
        printBody();
    }
}
```

运行核对

```
  ^^^^
 ^^^^^
#####
#O  D  O#
#O  D  O#
#   D   #
#####
```

门窗必须在对应格子里决定，不能先打印整面墙，再把门另起一行。这个重构版与第11课逐字符一致：每行9个字符，共7行。

边界提醒：本课先练组织代码。固定字符串版本适合重构练习，但真正可变尺寸的建筑仍需要参数和循环。

07 复用方法和小测试

复用不是盲目多调用。每次调用都应该有明确位置，方法本身也要能单独检查。

一次写好 多处调用

divider

定义一条分隔线

第一次调用

标题前打印

第二次调用

标题后打印

小测试

先验证一个方法

L12Example07.java • 完整独立示例

```
public class L12Example07 {
    static void divider() { System.out.println("====="); }
    public static void main(String[] args) {
        divider();
        System.out.println("方块学院");
        divider();
    }
}
```

运行核对

```
=====
方块学院
=====
```

三类检查

- 正常：按计划调用全部方法。
- 边界：只调用一个方法或调用两次。
- 错误：故意拼错方法名，观察编译器提示。

复用不等于隐藏：方法名应让读者知道发生了什么；不要把十个神秘步骤藏进 go()。

08 方法故障诊所

方法类 Bug 常分两类：编译器直接拦住，或程序能跑但少做一步。第二类更会伪装，所以要核对调用清单。

三个常见方法 Bug

少调用

编译成功，动作缺失

嵌套方法

Java拒绝这种声明

名字拼错

无法找到方法

修复

一次只改一个问题

L12Example08.java • 完整独立示例

```
public class L12Example08 {  
    static void packFood() { System.out.println("装食物"); }  
    public static void main(String[] args) {  
        System.out.println("出发");  
    }  
}
```

运行核对

出发

这个程序能编译，却没有装食物：定义方法不会自动执行。修复时在“出发”之前添加 packFood();。

故障实验：Java 不允许在方法体内再声明方法。下面故意保留错误，请先预测，再尝试修复。

BadNestedMethod.java • 故意错误示例

```
public class BadNestedMethod {  
    public static void main(String[] args) {  
        static void checkArmor() { }  
    }  
}
```

急诊顺序：先确认方法写在类里面、其他方法外面；再核对名称和圆括号；最后检查 main 是否真的调用。

09 挑战 蓝图 下矿准备流程

挑战 交付物是 MiningPrep.java。程序必须包含五个自定义方法，并由 main 按合理顺序调用。

五个方法组成一条出发流水线



建筑契约

- 五个方法都有清楚的动作名。
- 每个方法至少输出一条可验证信息。
- main 只负责安排流程，不重复方法内部的输出。
- 正常运行一次，再删除一个调用制造逻辑错误。

验收表

步骤	方法	验收输出
1	checkPickaxe	镐子耐久已检查
2	packFood	食物已装入
3	packTorches	火把已装入
4	wearArmor	护甲已穿好
5	printPlan	显示下矿路线

最小版本：先写两个方法并成功调用，再逐个增加。一次新增一个方法，Bug 就不会在五个矿洞里同时躲猫猫。

10 挑战 实现 五个方法排队

新建 MiningPrep.java。类名和文件名一致。这些准备方法不需要输入，也不返回值。void 表示没有返回值；static 让它们可从 static main 直接调用。两者是不同的设计选择。

完整下矿准备程序

定义

五个方法保存指令

调用

main安排五次调用

执行

每次完成后回到main

验收

依次输出五条信息

验收：能编译；输出五条；调用顺序合理；删除 packTorches(); 后能指出少了哪一步，再恢复并运行。

10 挑战 实现 五个方法排队 (续)

MiningPrep.java • 完整独立示例

```
public class MiningPrep {  
    static void checkPickaxe() {  
        System.out.println("镐子耐久已检查");  
    }  
    static void packFood() {  
        System.out.println("装入 16 个面包");  
    }  
    static void packTorches() {  
        System.out.println("装入 64 根火把");  
    }  
    static void wearArmor() {  
        System.out.println("铁甲已穿好");  
    }  
    static void printPlan() {  
        System.out.println("路线: 主矿道 -> 补给站 -> 回家");  
    }  
    public static void main(String[] args) {  
        checkPickaxe();  
        packFood();  
        packTorches();  
        wearArmor();  
        printPlan();  
    }  
}
```

运行核对

镐子耐久已检查
装入 16 个面包
装入 64 根火把
铁甲已穿好
路线: 主矿道 -> 补给站 -> 回家

11 拓展任务与通关复盘

作业要交源代码、一次运行结果和一句 Bug 记录。只交截图不算，因为截图不会告诉你分号躲在哪里。

三档任务继续整理代码

木镐

三个提示方法

铁镐

房屋重构，输出不变

钻石镐

四阶段挑战流程

复盘

说明定义与调用的区别

三档作业

- 木镐：写 morningTip、nightTip、dangerTip 三个提示方法，并从 main 各调用一次。
- 铁镐：把第11课像素房子的屋顶与含门窗的墙体分别装进方法，保持输出不变。
- 钻石镐：设计 挑战 战流程，至少包含准备、开战、第二阶段、胜利四个方法。

验收表

自评	能	还要练
我能区分定义和调用	☐	☐
我能解释执行顺序	☐	☐
我能把重复代码提取成方法	☐	☐

下一课：第13课给方法增加参数和返回值。同一张配方卡将能接收不同材料，并把计算结果送回调用处。

自己的验收记录

运行之前写预期，运行之后记录实际结果。正常案例说明程序会做事；边界案例说明它知道何时停；错误案例说明它能拒绝不合理的请求。

把证据写进记录

类别	你选择的输入	预期与实际
正常	一个常用值或一次成功操作	是否逐行一致？
边界	最小值、最大值及相邻值	有没有少一格、多一步？
错误	非法数量、缺失对象或错误语法	是否明确拒绝？状态是否保持？

- 把一个 Bug 写成：触发条件、现象、原因、最小修复。
- 整理一个值得记住的规则，用自己的话解释，不能只复制定义。
- 改变一个示例，保留测试记录，让结果证明你的理解。

技术核对依据：Java SE 21 语言规范，作用域与访问控制（第6章）、类与构造器（第8章）、方法调用（第15章）。 <https://docs.oracle.com/javase/specs/jls/se21/html/index.html>