

第32课

给服务器加命令

用参数、权限和补全发放工具包



一条 `/kit` 命令先经过调用者、权限与参数检查，才把物品放进背包。你将同时照顾玩家、控制台和满背包情况，让失败也得到清楚的回应。

前置背包：第31课的Gradle项目与Paper测试服。复制项目为BlockAcademyL32，保留构建脚本，将settings中的项目名改为BlockAcademyL32。

阅读方式：先猜结果，再运行核对；每次只改一个条件。

环境：JDK 21、普通 Paper 1.21.11、同版本客户端。服务器插件由 Paper 加载；只有明确标注的规则测试使用 main。

开始前：先认清运行位置

前30课打好了普通 Java 程序的基础。这一阶段把代码接到 Paper 服务器：Minecraft 客户端负责进入世界，服务器加载插件，事件和命令把玩家动作交给你的方法。

本书固定使用 JDK 21 与 Paper 1.21.11，客户端版本也保持一致。下载页面上的最新版本可能已经变了，按版本号选择。本书采用普通 plugin.yml 与 Bukkit API；本阶段没有声称支持 Folia。

图框中的短例是局部片段，先读本节的位置说明，不要各自保存成插件。完整文件会明确给出文件名；同一文件分成多段时，按编号连接，包名、import 与外层类只出现一次。

- 预测：先写成功条件和预期消息。
- 运行：构建成功后，在专用本机测试服安装并重启。
- 核对：普通玩家、管理身份与控制台分别走对应路径。

场景插画帮助理解故事；图解解释真实规则，不作为 IDE 或游戏实测截图。做 Java 修改后重新构建、停服替换新 JAR，再启动；不要把旧 JAR 的表现当作新代码结果。

01 先认清是谁在发命令

村民在控制台输入领工具包命令，程序却找他要背包。控制台没有手，更没有快捷栏。命令第一关先认清调用者，再数参数，最后才打开配方箱。

把命令当作一条有门卫的路线：调用者类型、权限、参数数量、参数内容都通过后，才修改背包。失败也是一次需要清楚回复的处理，别让玩家只看到沉默。

先认清是谁在发命令

玩家

能领取到背包

控制台

得到明确拒绝

共同规则

检查完毕才改变物品

返回值：onCommand 返回 true 表示本次命令已处理，包括已解释的失败。返回 false 通常让服务器发送 plugin.yml 中的 usage。

安全取得玩家 • 位置见本节说明

```
if (!(sender instanceof Player player)) {
    sender.sendMessage("请在游戏里领取工具包");
    return true;
}
player.sendMessage(Component.text("背包主人确认完毕"));
```

故障实验：下面故意保留错误，仅在练习副本中观察。不要把它合并进可用插件。

错误的强制转换 • 故意错误，先预测再修复

```
Player player = (Player) sender; // 控制台执行会出错
```

02 先数参数再打开格子

输入 `/kit miner` 时，`kit` 是命令名，`args` 里只有 `miner`；`args[0]` 不是 `kit`。玩家只敲 `/kit` 时数组长度是0，此时读取第0格就越界。

先数参数

长度必须为1

→

再读第0格

它是miner，不是kit

→

最后认配方

未知选项明确拒绝

沿箭头进入下一步前，必须满足前一步条件；被拒绝的分支提前返回。

条件先行：先检查长度，再读取元素。参数不是越多越聪明，多出来的单词也要有明确规则。

只接受一个参数 · 位置见本节说明

```
if (args.length != 1) {
    player.sendMessage(Component.text("用法: /kit miner 或 /kit builder"));
    return true;
}
```

不受系统语言影响的比较 · 位置见本节说明

```
String name = args[0].toLowerCase(Locale.ROOT);
```

核对表

输入	长度	结果
<code>/kit</code>	0	显示用法
<code>/kit miner</code>	1	选择矿工包
<code>/kit miner extra</code>	2	显示用法
<code>/kit wizard</code>	1	提示未知工具包

03 命令清单和权限要配套

建立 lesson32 包，主类叫 KitPlugin，替换资源目录中的 plugin.yml。练习默认允许玩家领取，代码仍检查权限，方便以后由权限管理系统收回某个玩家的资格。

命令清单和权限要配套

命令kit

清单与getCommand对应

权限节点

代码在执行前检查

默认规则

本章default为true

选择处理位置：本课由代码统一检查权限并发送中文提示，commands.kit 下没有再写 permission。default: true 是本课的练习规则，不是所有服务器的默认选择。

本课完整 plugin.yml • 完整文件

```
name: BlockAcademyL32
version: 1.0.0
main: de.teacherma.lesson32.KitPlugin
api-version: '1.21.11'
author: TeacherMa
website: https://www.teacher-ma.de
commands:
  kit:
    description: 领取课堂工具包
    usage: /kit <miner|builder>
permissions:
  blockacademy.kit:
    default: true
```

执行前查权限 • 位置见本节说明

```
if (!player.hasPermission("blockacademy.kit")) {
    player.sendMessage(Component.text("你还没有领取权限"));
    return true;
}
```

04 背包满了物品去哪儿

addItem 尝试把物品塞进背包，并返回没有放进去的部分。返回的 Map 记录剩余物品，键是输入物品的序号，不是背包槽位。不能只说“领取成功”就把剩余物品忘掉。

背包满了物品去哪儿



练习策略：本课把剩余物品掉到玩家脚边，附近玩家可能捡走，掉落物也可能消失。测试在平坦空地；正式服可改成“先腾位置再领取”。

发物品并处理剩余 · 位置见本节说明

```
var leftovers = player.getInventory().addItem(  
    new ItemStack(Material.TORCH, 16));  
for (ItemStack item : leftovers.values()) {  
    player.getWorld().dropItemNaturally(player.getLocation(), item);  
}
```

反馈要符合结果 · 位置见本节说明

```
String message = leftovers.isEmpty()  
    ? "全部放入背包" : "剩余物品掉在脚边";  
player.sendMessage(Component.text(message));
```

05 按 Tab 给玩家一点提示

命令补全只推荐可能的输入，真正执行时仍要检查参数。否则玩家手动输入一个奇怪名字，就能绕过只存在于菜单里的规则。

按 Tab 给玩家一点提示

补全提示

只推荐合法名字

执行验证

手动输入仍需检查

未知前缀

返回空列表

复习第30课：这里的 filter 保留匹配名字，toList 收集结果。补全方法返回空列表时，不再提供额外建议。

从列表筛选补全项 · 位置见本节说明

```
String prefix = args[0].toLowerCase(Locale.ROOT);
return List.of("miner", "builder").stream()
    .filter(name -> name.startsWith(prefix)).toList();
```

不要访问不存在的参数 · 位置见本节说明

```
if (args.length != 1) return List.of();
if (!sender.hasPermission("blockacademy.kit")) return List.of();
```

核对表

当前参数	推荐
空字符串	miner 和 builder
m	miner
b	builder
z	空列表

06 工具包配方先写在纸上

今天的 挑战 是两种工具包。先把配方写清，再把输入、判断、发放和反馈串起来。练习版可重复领取；冷却功能会在第34课学会后再加。

工具包配方先写在纸上

miner

石镐1把、火把16个

builder

木板32个、玻璃16个

未知名字

返回null后停止发放

空结果不发放：items 为 null 表示未知配方，先发送说明并 return，再走背包逻辑。下一页和再下一页合起来是一份完整源文件。

用 switch 选配方 • 位置见本节说明

```
ItemStack[] items = switch (name) {
    case "miner" -> new ItemStack[] {
        new ItemStack(Material.STONE_PICKAXE),
        new ItemStack(Material.TORCH, 16)};
    default -> null;
};
```

核对表

工具包	物品一	物品二
miner	石镐 1把	火把 16个
builder	橡木木板 32个	玻璃 16个

07 工具包程序上半部

新建 src/main/java/de/teacherma/lesson32/KitPlugin.java。先复制本页，再在后面紧接下一页；中间不加包名、import 或额外的大括号。

工具包程序上半部

一份源文件

前后两部分顺序连接

先检查

类型、权限和参数

通过后

才进入配方与发放

发现错误要尽早：getCommand 为 null 通常说明资源里的命令名没对上。requireNonNull 会在启动时给出明确线索。

KitPlugin 第1部分 • 本段1/2，顺序连接

```
package de.teacherma.lesson32;

import java.util.List;
import java.util.Locale;
import java.util.Objects;
import net.kyori.adventure.text.Component;
import org.bukkit.Material;
import org.bukkit.command.Command;
import org.bukkit.command.CommandSender;
import org.bukkit.entity.Player;
import org.bukkit.inventory.ItemStack;
import org.bukkit.plugin.java.JavaPlugin;

public final class KitPlugin extends JavaPlugin {
    @Override
    public void onEnable() {
        var kit = Objects.requireNonNull(getCommand("kit"), "请检查 plugin.yml");
        kit.setExecutor(this);
        kit.setTabCompleter(this);
    }

    @Override
    public boolean onCommand(CommandSender sender, Command command,
                             String label, String[] args) {
        if (!(sender instanceof Player player)) {
```

07 工具包程序上半部（续）

KitPlugin 第1部分 • 本段2/2, 顺序连接

```
        sender.sendMessage("请在游戏里领取工具包");
        return true;
    }
    if (!player.hasPermission("blockacademy.kit")) {
        player.sendMessage(Component.text("你还没有领取权限"));
        return true;
    }
    if (args.length != 1) {
        player.sendMessage(Component.text("用法: /kit miner 或 /kit builder"));
        return true;
    }
}
```

08 工具包程序下半部

下面紧接上一页。发放、剩余物品和补全都放在同一个类里。保持两个 @Override 方法的边界，大括号数量别让苦力怕替你数。

工具包程序下半部

发放

addItem并处理剩余

反馈

说明实际结果

补全

按当前前缀过滤

构建部署：使用第31课的构建脚本与本课plugin.yml。运行 .\gradlew.bat clean build，把BlockAcademyL32的JAR放入测试服并重启。

KitPlugin 第2部分 • 本段1/2，顺序连接

```

ItemStack[] items = switch (args[0].toLowerCase(Locale.ROOT)) {
    case "miner" -> new ItemStack[] {
        new ItemStack(Material.STONE_PICKAXE),
        new ItemStack(Material.TORCH, 16)};
    case "builder" -> new ItemStack[] {
        new ItemStack(Material.OAK_PLANKS, 32),
        new ItemStack(Material.GLASS, 16)};
    default -> null;
};
if (items == null) {
    player.sendMessage(Component.text("只认识 miner 和 builder"));
    return true;
}
var leftovers = player.getInventory().addItem(items);
for (ItemStack item : leftovers.values()) {
    player.getWorld().dropItemNaturally(player.getLocation(), item);
}
player.sendMessage(Component.text(leftovers.isEmpty()
    ? "工具包已放入背包" : "背包已满，剩余物品掉在脚边"));
return true;
}

@Override
public List<String> onTabComplete(CommandSender sender, Command command,
    String alias, String[] args) {

```

08 工具包程序下半部（续）

KitPlugin 第2部分 • 本段2/2, 顺序连接

```
        if (!sender.hasPermission("blockacademy.kit") || args.length != 1)
            return List.of();
        String prefix = args[0].toLowerCase(Locale.ROOT);
        return List.of("miner", "builder").stream()
            .filter(name -> name.startsWith(prefix)).toList();
    }
}
```

09 把失败路径也走一遍

先腾出背包做成功测试，再主动制造空参数、陌生参数和满背包。测试的目标是知道每种输入会发生什么，不能只凭“我领到了一次镐子”就宣布通关。

把失败路径也走一遍

正常

两种包都符合配方

边界

零参数、满背包

拒绝

未知选项、控制台、无权限

权限测试：默认允许领取。老师若已有权限插件，可明确拒绝 blockacademy.kit，再验证不发物品；不要把默认允许误判为代码没有检查。

核对表

操作	预期
玩家 /kit miner	1把石镐和16个火把
玩家 /kit builder	32木板和16玻璃
控制台 kit miner	提示在游戏里领取，无异常
/kit 或 /kit wizard	清楚反馈，不发物品
背包填满后领取	脚边有剩余物品，消息准确
输入 /kit m 后按Tab	补全miner



马老师教编程
www.teacher-ma.de

13 / 16

10 让命令更容易维护

当命令越来越多，每个分支都写一段巨大的发放代码，修一次就像挖整座山。把配方提成方法，命令方法只负责检查和调用。

停下来想一想：读取 `args[0]` 之前，应该先完成什么判断？零参数为何不能靠后面的配方分支补救？

让命令更容易维护

职责分明

命令负责检查

配方方法

只返回物品清单

测试同步

改规则也改预期

想通这一步：先检查 `args.length`。越界发生在读取时，后面的分支还没机会执行；说明用法后应提前返回。

提取一个配方方法 • 位置见本节说明

```
private ItemStack[] minerKit() {  
    return new ItemStack[] {  
        new ItemStack(Material.STONE_PICKAXE),  
        new ItemStack(Material.TORCH, 16)};  
}
```

帮助消息也要一致 • 位置见本节说明

```
player.sendMessage(Component.text("用法: /kit miner 或 /kit builder"));  
// plugin.yml 中 usage 写成同样的选择范围
```

- 把大小写改成 MINER，预期仍识别矿工包。
- 把未知选项改为 wizard，预期没有任何物品变化。
- 把工具包数量改成配方表中的新值，再更新验收记录。

11 拓展挑战与自己的验收记录

留下源代码、实际运行结果和一条问题记录。先用自己的话解释，再运行验证；结果与预测不同的地方，往往最值得讨论。

拓展任务与通关复盘

木镐

实现名字问候

铁镐

完成两种包与失败记录

钻石镐

按玩家增加独立冷却

下一站：第33课用事件管理哪些方块可以挖。
官方延伸阅读：jd.papermc.io/paper/1.21.11/org/bukkit/command/CommandExecutor.html

- 木镐：实现 /hello，向玩家发送带名字的问候；控制台也能得到合理提示。
- 铁镐：完成两种工具包，提交空参数和满背包的测试记录。
- 钻石镐：学完第34拓展给领取命令加入每人30秒冷却，并验证两名玩家互不影响。
- 验收要求：用玩家和控制台分别测试，补齐无权限、零参数、未知配方、满背包场景；确认失败不会发物品。

核对表

我能独立完成	已掌握	再练一次
先检查发送者和参数	☐	☐
能处理背包剩余物品	☐	☐
能解释权限与补全	☐	☐

把证据留给下一次冒险

把“能编译”“数字规则通过”和“游戏实际表现正确”分成三栏。没有运行的场景写待测；发现 Bug 时记录触发条件、预期、实际、最小修复与复测结果。

你的验证记录

类别	你选择的操作	留下的证据
正常	本课的一次成功路径	版本与实际输出
边界	结束前一刻、最大值或满背包	与合同逐项比较
拒绝	错误参数、无权限或不相关事件	反馈清楚，状态未误改

- 解释本课每条箭头为什么从这里指向那里；并列卡片没有先后含义。
- 先改一个规则，再同时修改说明和预期，用同一输入复测。

技术核对：本课以 Paper 1.21.11 API 和 Java SE 21 为准。参考文档可能切换到较新版本，查阅时注意版本号。

<https://jd.papermc.io/paper/1.21.11/>
<https://docs.papermc.io/paper/dev/>
<https://docs.oracle.com/en/java/javase/21/docs/api/>