

FABRIC 1.20 / JDK 21

第 1 课

启动第一个 Fabric MOD

打开工程，进入世界，找到自己的启动日志。



真实测试世界：模板在玩家加入时生成的草方块台阶。

本课作品

跑通指定模板，并亲手写出一条经过实机验证的启动消息。

90-110 分钟 | [simon-minecraft-template-1.20](#)

今天先把工作台准备好；第 2 课再注册独立的入门方块。

前置知识：会使用键盘、鼠标和文件夹

teacher-ma.de/minecraft-mod-lesson.php?lesson=1&lang=zh

01 / 11

先跑通，再改一行

不需要先背熟 Java，也不需要从零新建一个 MOD 工程。

这节课用老师提供的现成模板。第一轮原样运行，得到可靠起点；第二轮只增加一条日志，验证你写的 Java 确实执行了。先不改模组 ID、方块代码和依赖版本。

IntelliJ IDEA	编辑、定位文件，查看运行控制台；它不是 Minecraft 游戏本身。
JDK 21	提供 Java 开发与运行工具。按这个模板要求选择 JDK，不只安装 JRE。
Gradle Wrapper	使用工程自带的构建工具配置，下载依赖、编译与启动开发客户端。
Fabric	Loader 识别并加载 MOD；Fabric API 提供模组开发时使用的接口。

1. 打开工程 / 2. 核对 JDK / 3. 原样启动 / 4. 验证日志 / 5. 修改再测

用时与通关证据

课堂操作约 90-110 分钟，首次下载时间另计。通关要同时留下：入口日志、可进入的测试世界，以及重启后出现的 Lesson 1 ready!。

先预测：游戏能打开，是否就证明你改的那一行 Java 已经执行？

02 / 11

打开整个工程

先解压到普通文件夹；不要在压缩包预览中直接运行。

模板工程根目录示例

```
C:\TeacherMa\Mod\simon-minecraft-template-1.20
```

在 IntelliJ 选择 File → Open，选中上面的工程文件夹。学生电脑可放在其他位置；根目录必须能看到 build.gradle 和 gradlew.bat。只打开 .java 文件会丢失完整 Gradle 工程。

第一次打开	让 IntelliJ 识别并加载 Gradle 工程，等待依赖下载与同步完成。
已经打开过	从最近项目进入同一个工程，不再用 New Project 创建另一份空工程。
同步成功的线索	能看到 Gradle 工具窗口及工程任务；源码不再大面积标红。
同步仍在下载	首次需要联网；下载时长取决于网络。失败时看具体错误，不反复点击运行。

目录地图：修改 src，查看 run，不修改 build

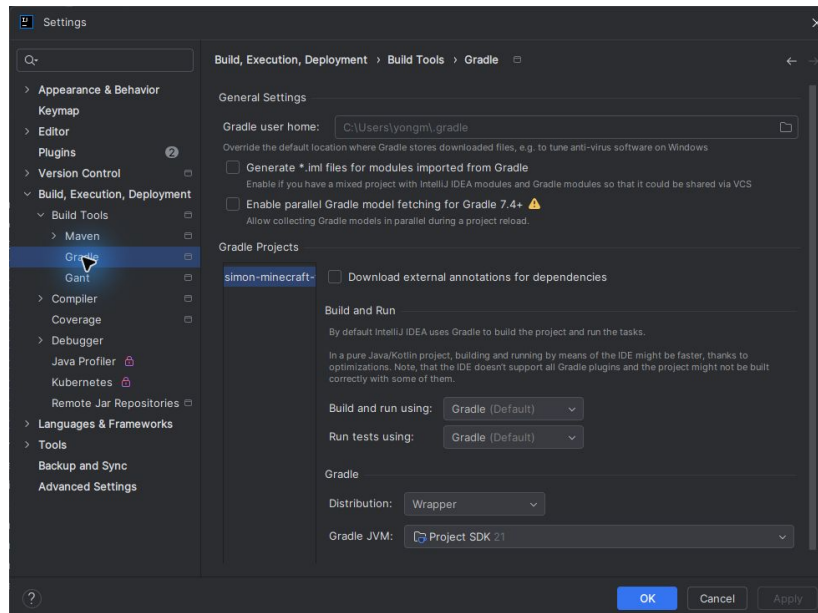
```
project root/  
  build.gradle, gradle.properties, gradlew.bat  
  src/main/java/           Java source  
  src/main/resources/      JSON and textures  
  run/                     game data and logs  
  build/                   generated output
```

只改 src 里的源码和资源；不要把 build 中的生成文件当作原始文件。

03 / 11

核对模板要求的 JDK 21

运行要求与编译目标不同：检查两个设置位置。



真实设置页：Distribution 使用 Wrapper，Gradle JVM 指向 Project SDK 21。

Project SDK

File → Project Structure → Project: SDK 选择已安装的 JDK 21。

Gradle JVM

Settings → Build, Execution, Deployment → Build Tools → Gradle：选择 JDK 21，或指向 21 的 Project SDK；Distribution 保留 Wrapper。

不要看到 17 就把所有设置都改成 17

当前模板的 fabric.mod.json 写着 java >=21；build.gradle 的 options.release=17 是编译目标，不等于这个模板允许用 Java 17 启动。保留模板版本，只把运行工具配置到 JDK 21。

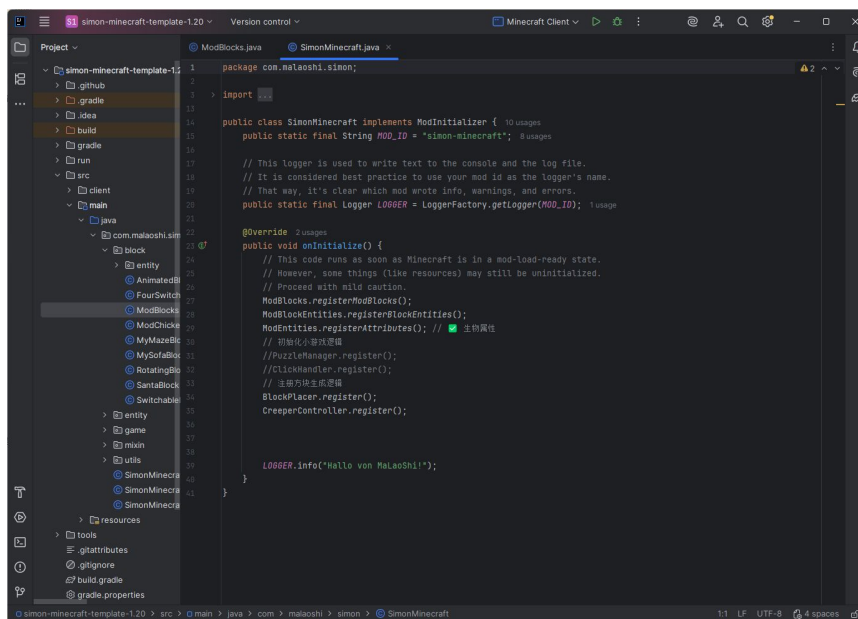
04 / 11

找到 Java 入口

先观察，不整段替换。

打开这个文件

```
src/main/java/com/malaoshi/simon/SimonMinecraft.java
```



真实 IntelliJ 截图：本课使用的 SimonMinecraft 入口类。

原文件中不同位置的两行：只用于定位

```
public static final String MOD_ID = "simon-minecraft";  
LOGGER.info("Hallo von MaLaoShi!");
```

这两行在文件中的不同位置。MOD_ID 是模组标识；LOGGER.info 会输出日志。先找到 onInitialize() 的大括号，看到底部的原有启动消息。

保留模板已有功能

ModBlocks、ModBlockEntities、BlockPlacer 等调用已经存在，暂时不必理解每个细节。保留全部注册与事件调用，不要用节选代码覆盖整个类或整个 onInitialize()。

05 / 11

Fabric 如何找到入口类

打开 `src/main/resources/fabric.mod.json`；本节只阅读。

`fabric.mod.json` 阅读节选，不是完整配置

```
{
  "id": "simon-minecraft",
  "environment": "*",
  "entrypoints": {
    "main": [
      "com.malaoshi.simon.SimonMinecraft"
    ],
    "client": [
      "com.malaoshi.simon.SimonMinecraftClient"
    ]
  }
}
```

这是阅读节选，不是完整配置。原文件还有 `schemaVersion`、`version`、`depends`、`mixins` 等字段，全部保留；今天不改 JSON。

main	指向实现 <code>ModInitializer</code> 的通用入口；客户端启动也会加载这部分内容。
client	指向实现 <code>ClientModInitializer</code> 的客户端入口，例如渲染；不要把它当通用入口。
完整类名	<code>package com.malaoshi.simon</code> 加类名 <code>SimonMinecraft</code> ，组成 <code>main</code> 中的完整路径。

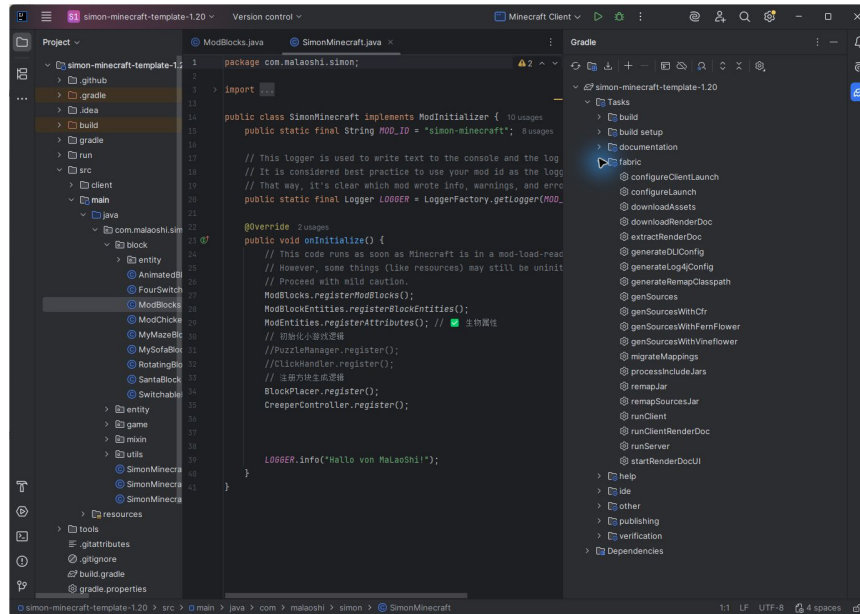
本课只核对，不改命名空间

JSON 的 `id` 与 Java 的 `MOD_ID` 都是 `simon-minecraft`。后续资源也用它；不要只修改其中一个位置，造成入口、注册与资源不一致。

06 / 11

启动 runClient

Gradle → 工程名 → Tasks → fabric → runClient



真实 IntelliJ Gradle 面板：在 fabric 分组中找到 runClient。

双击 runClient，等待编译与资源加载，然后切到弹出的 Minecraft 窗口。游戏开按时任务会持续运行，这不是卡住；不要反复启动多个客户端。

备用方式：在工程根目录的 PowerShell 中执行

```
.\gradlew.bat runClient
```

两种方式选一种

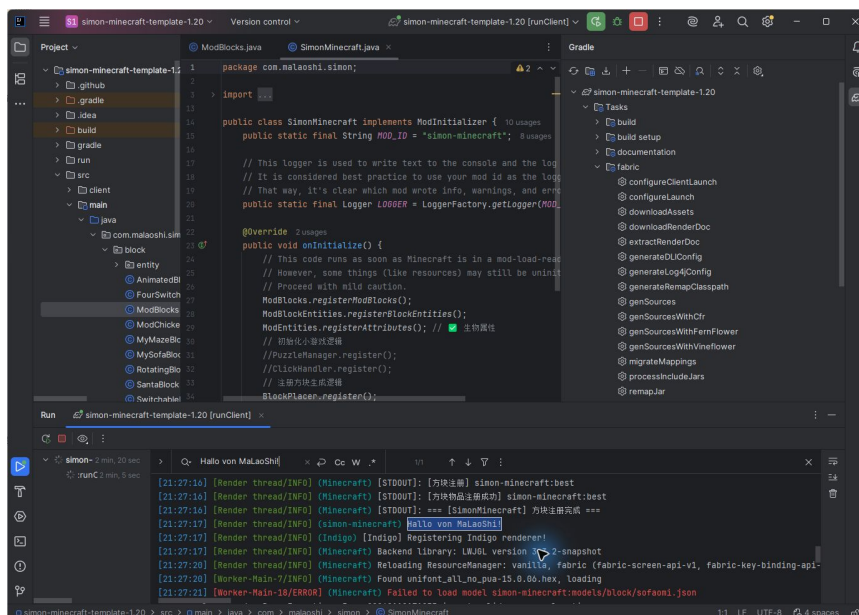
上面的命令是备用路径，只在工程根目录执行。IDE 中也可选择已生成的 Minecraft Client 配置再点 Run。不要把 SimonMinecraft.java 当普通 Java main 程序运行。

07 / 11

找到原模板启动日志

打开底部 Run 控制台，用 Ctrl+F 搜索完整消息。

在控制台搜索 Hallo von MaLaoShi!。找到来自 simon-minecraft 的这一行，再继续进入世界。若控制台滚动很快，使用搜索，不靠肉眼翻找。



真实运行控制台：已找到原有启动消息。

实际日志内容，省略时间和线程前缀

(simon-minecraft) Hallo von MaLaoShi!

INFO

信息级别，不是错误。日志还可能包含时间、线程名和日志来源。

simon-minecraft

Logger 的名字，帮助判断这条消息来自哪个模组。

日志证明什么

证明执行到了这一行；不自动证明每个方块模型或贴图都完整。

日志在哪里?

运行时看 IDE 的 Run 控制台；也可查看工程 run/logs/latest.log。LOGGER.info 不会自动发送游戏聊天消息。

08 / 11

进入独立测试世界

建议创造模式、允许作弊、和平难度和超平坦世界。



游戏实拍：从 IntelliJ 启动后进入的新建课堂世界。

创建

单人游戏 → 创建新的世界；起名 Lesson01Test。不要在重要旧存档中做实验。

观察

W/A/S/D 移动，鼠标转向，空格跳跃。当前模板在玩家加入时会自动生成草方块台阶。

记录

记住世界名称与截图位置；F2 保存截图到 run/screenshots。截图是游戏实拍，不是 3D 插图。

台阶属于模板，日志属于入口

台阶由模板已有的 BlockPlacer 生成，不是今天新增的代码。进入世界并找到日志，完成基础验收；模板中的实验模型缺项，另行记录和修复。

09 / 11

添加自己的启动消息

先正常退出游戏，再回到 SimonMinecraft.java。

在已有 onInitialize() 中保留旧行，并添加新行

```
LOGGER.info("Hallo von MaLaoShi!");  
LOGGER.info("Lesson 1 ready!");
```

保留原来的第一行，在它下面、onInitialize() 的结束大括号之前，添加第二行。今天不另建类、不添加 import，也不删除其他注册调用。

LOGGER.info(...)	调用模板已有 Logger 的 info 方法，把括号中的文字写入日志。
-------------------------	--------------------------------------

"Lesson 1 ready!"	双引号里的内容是字符串。使用英文半角双引号，不用中文弯引号。
--------------------------	--------------------------------

分号 ;	结束这条 Java 语句。漏分号会导致编译错误，通常会标红。
-------------	--------------------------------

大小写与位置	LOGGER 要大写，info 要小写；新行放在方法大括号内，不放在类外。
---------------	---------------------------------------

先预测，再启动

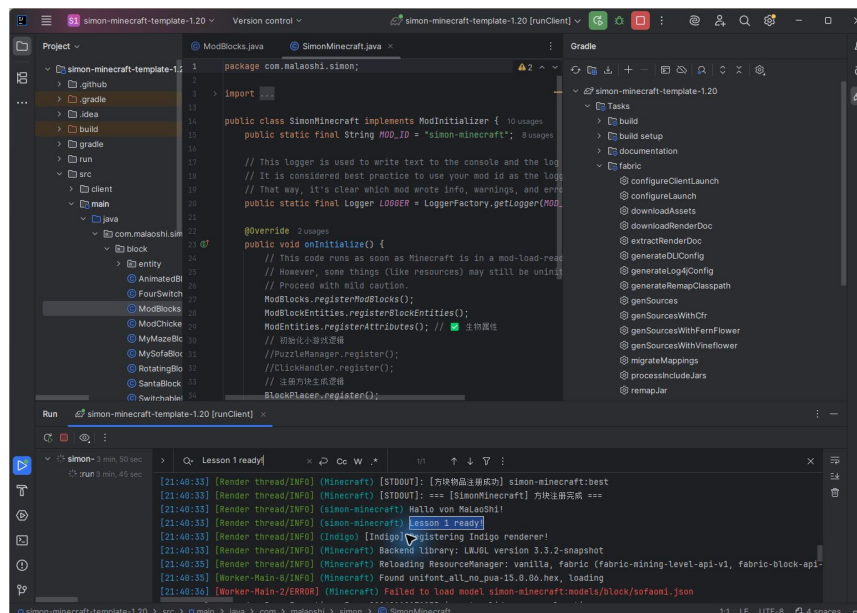
这行代码只改变启动日志，不会改变方块外观、生成一个新方块或发送聊天文字。保存文件后重新运行 runClient，再查找 Lesson 1 ready!。

小练习：把字符串改成自己的英文启动消息，保持调用、引号与分号不变。

10 / 11

重启后验证修改

第二次启动后，搜索 Lesson 1 ready!。



真实第二次运行控制台：新添加的 Java 消息已执行。

第二次启动的实际日志，省略时间前缀

(simon-minecraft) Lesson 1 ready!

改 Java

关闭旧客户端、保存源码，再运行 runClient，才会重新编译并加载。

F3+T

重载客户端资源；不会重新编译 Java，也不会重新执行入口。

退出世界再进入

通常仍是同一个客户端进程，不等于重新启动 MOD 入口。

runClient 与 build 是不同的测试

runClient 验证启动与游戏；build 验证编译、检查和产物生成。退出游戏后，在工程根目录执行 .\gradlew.bat build，查看 BUILD SUCCESSFUL；它不能替代游戏实测。

排错与通关清单

按出错阶段定位，不要一次改动多个版本或文件。

找不到 Gradle 任务	核对打开的是含 build.gradle 的根目录；检查 Gradle 同步结果。
依赖下载失败	查看具体网络或仓库错误；确认联网后重新同步，不先升级所有依赖。
Java 版本不符	核对 Project SDK、Gradle JVM；命令行启动也要使用符合模板要求的 JDK。
新消息没出现	核对改的是该工程的 onInitialize()；关闭旧客户端、保存并重启。
编译报错或崩溃	先看带源码行号的错误；沿异常原因链定位相关 Caused by，并保留完整日志。
局部模型缺项	原教材截图中的 sofaomi 模型缺项已存在，本次仍能进世界。其他 ERROR 按具体原因排查。

完成这五项，就可以进入下一课

1. 能找到工程根目录与 src。
2. 能说明 main 如何找到入口类。
3. 找到原有启动日志。
4. 能进入独立测试世界。
5. 重启后找到自己的消息，并区分 Java 重启与资源重载。

下一课：沿用今天的工程和启动方法，注册独立的入门方块 starter_block。它会有正常图标、中文名称，并能拿取和放置。

官方参考

官方参考用于理解概念与设置路径；本课版本以指定模板为准。

Fabric：工程结构与入口

Fabric：fabric.mod.json

JetBrains：Gradle JVM 设置