

Java 專案前置作業

TYIC 桃高資訊社

引言

在學習完基本的 **Java** 語法和知識後
我們可以透過做專案來檢測自己的實力
並學習到更多無法只透過投影片和練習題學習到的東西
因此接下來我們會製作一個自己的 **Minecraft 模組(mod)**

範例模組版本：

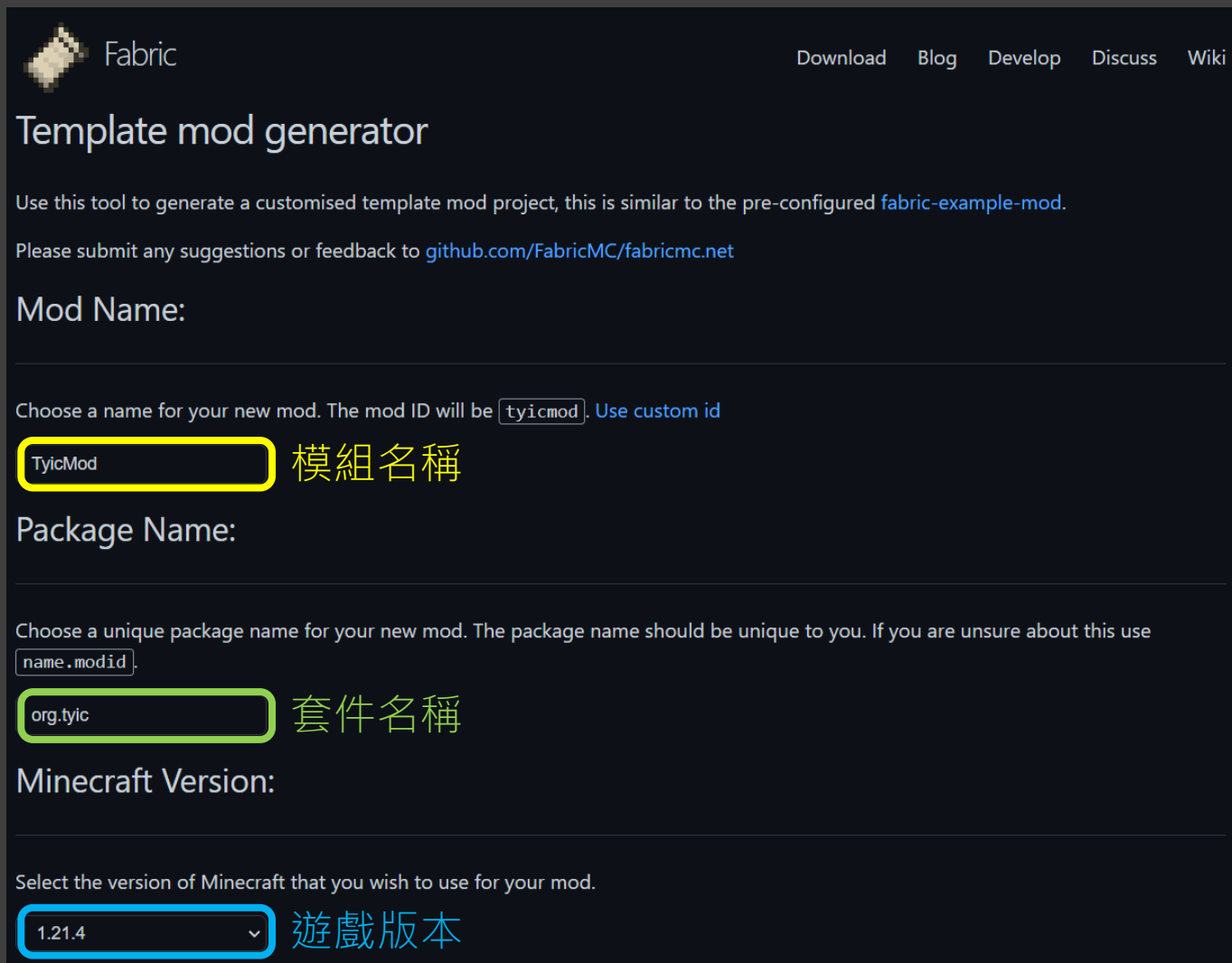
Minecraft 版本：Java Edition 1.21.4 with Fabric

JDK 版本：Java SE 21 以上

生成樣板

首先至 **Fabric template mod generator** 網站
(<https://fabricmc.net/develop/template/>)

將**模組名稱**和**套件名稱**填入
並選擇想要的**遊戲版本**
本教學以 **1.21.4** 為例
不同版本模組寫法會有差異



The screenshot shows the 'Fabric Template mod generator' website. At the top, there's a navigation bar with links for 'Download', 'Blog', 'Develop', 'Discuss', and 'Wiki'. The main heading is 'Template mod generator'. Below it, a paragraph explains the tool's purpose: 'Use this tool to generate a customised template mod project, this is similar to the pre-configured fabric-example-mod.' It also provides a link for suggestions or feedback: 'github.com/FabricMC/fabricmc.net'. The form has three main sections: 1. 'Mod Name:' with a text input field containing 'TyicMod' and a label '模組名稱' to its right. 2. 'Package Name:' with a text input field containing 'org.tyic' and a label '套件名稱' to its right. 3. 'Minecraft Version:' with a dropdown menu showing '1.21.4' and a label '遊戲版本' to its right. Each input field is highlighted with a colored border (yellow for Mod Name, green for Package Name, and blue for Minecraft Version).

生成樣板

將 **Data Generation**
和 **Split client and common sources**

兩個選項勾選

接著點選下方的

Download Template

就會下載模組樣板

將模組樣板

解壓縮至資料夾

Advanced Options:

☐ Kotlin Programming Language

[Kotlin](#) is an alternative programming language that can be used to develop mods. The [Fabric Kotlin language adapter](#) is used to enable support for creating Fabric Kotlin mods.

☐ Mojang Mappings

Use Mojang's official mappings rather than Yarn. Note that Mojang's mappings come with a usable yet more restrictive license than Yarn. Use them at your own risk.

☒ Data Generation

This option configures the [Fabric Data Generation API](#) in your mod. This allows you to generate resources such as recipes from code at build time.

☒ Split client and common sources

A common source of server crashes comes from calling client only code when installed on a server. This option configures your mod to be built from two source sets, client and main. This enforces a clear separation between the client and server code.

 **Download Template (.ZIP)**

For setup instructions please see the [fabric wiki page](#) that relates to the IDE that you are using. This template is available under the CC0 license. Feel free to learn from it and incorporate it in your own projects.

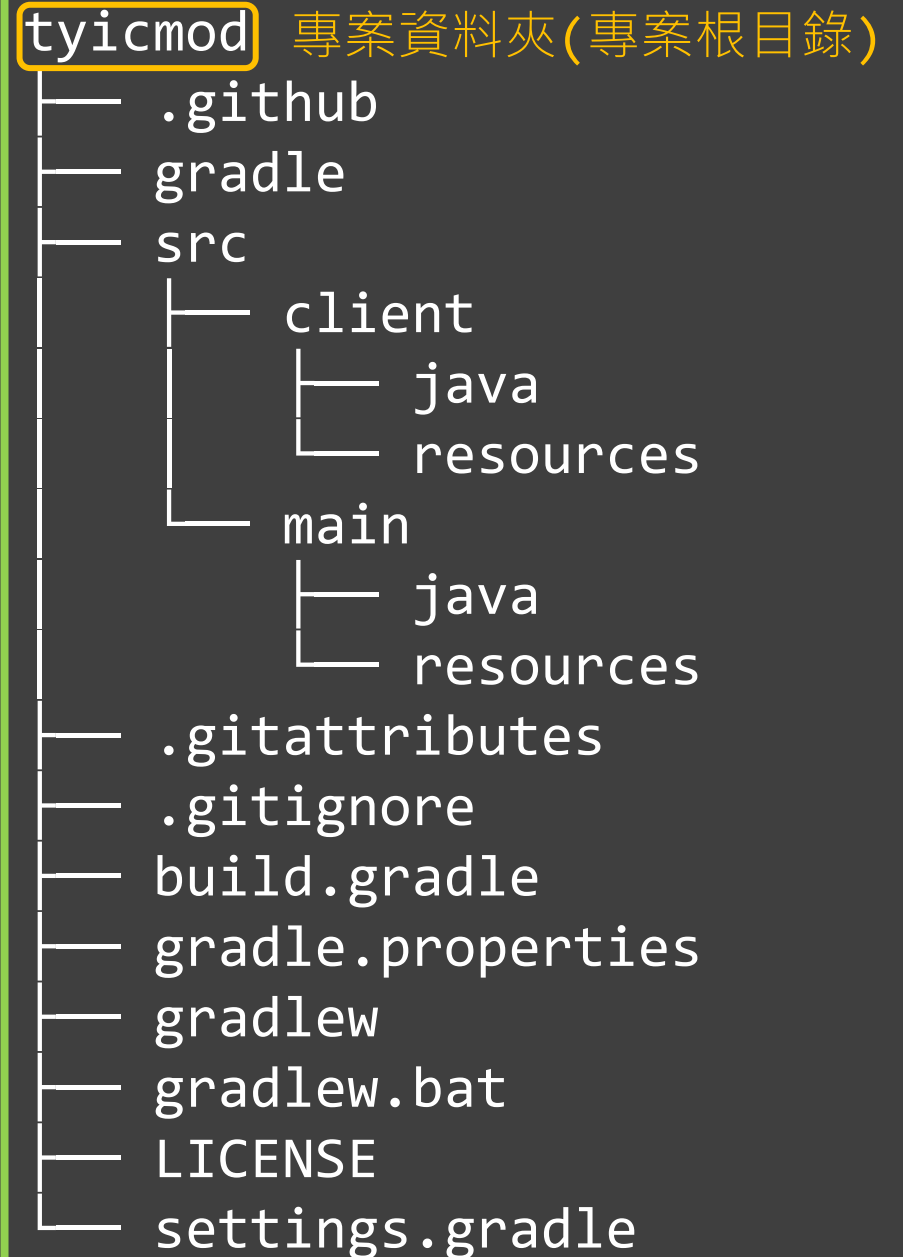
 BY-NC-SA

The contents of this website, unless otherwise indicated, are licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).

NOT AN OFFICIAL MINECRAFT PRODUCT. NOT APPROVED BY OR ASSOCIATED WITH MOJANG.

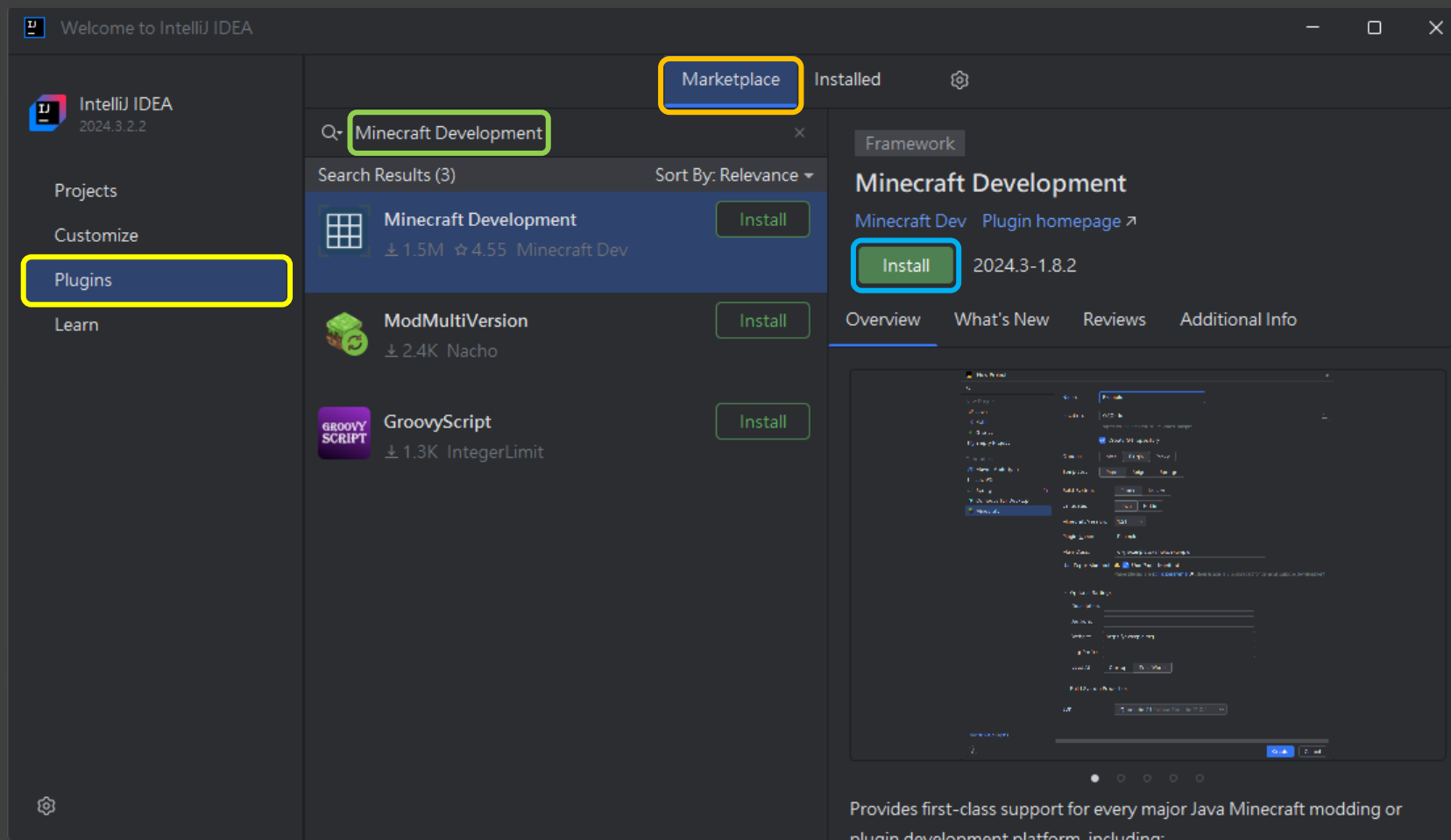
解壓縮樣板

解壓縮後的資料夾結構應該類似如右
注意解壓縮時不可資料夾嵌套，如下



安裝 Minecraft Development 插件

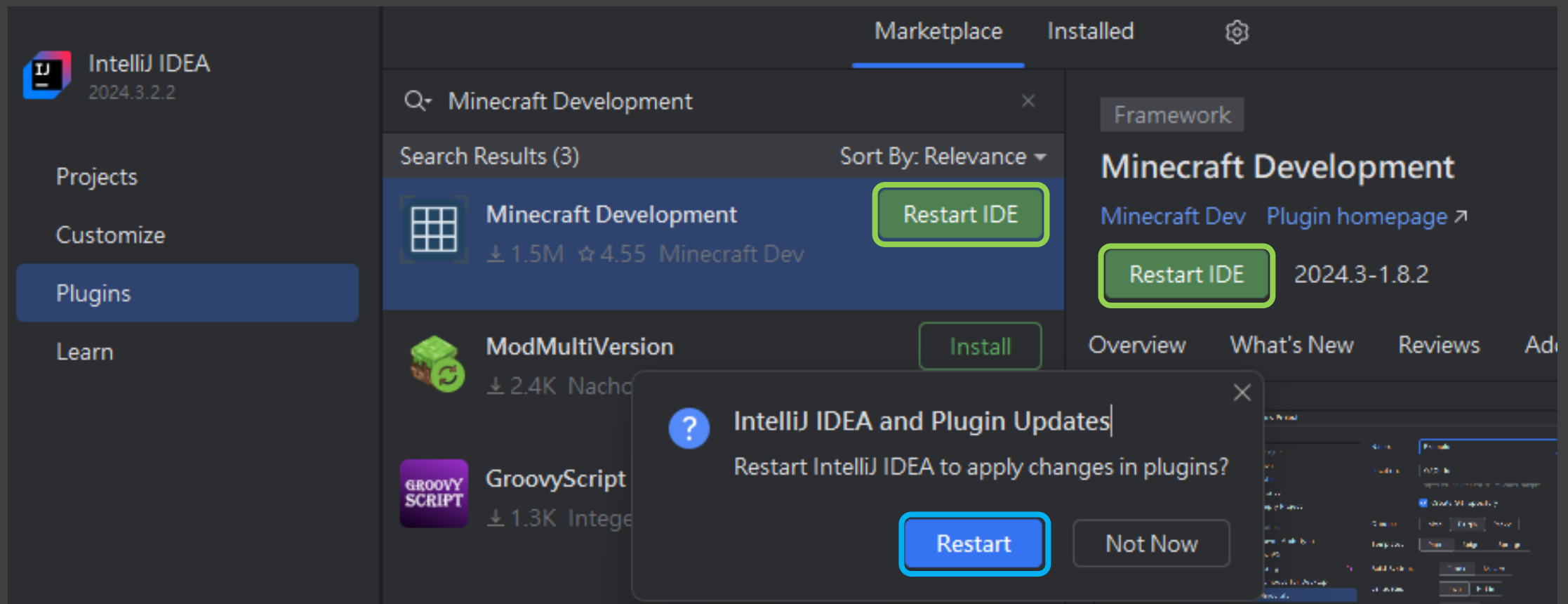
開啟 IntelliJ IDEA 後，點擊左方的 **Plugin**，選擇 **Marketplace** 搜尋 **Minecraft Development**，並點擊 **Install** 安裝



安裝 Minecraft Development 插件

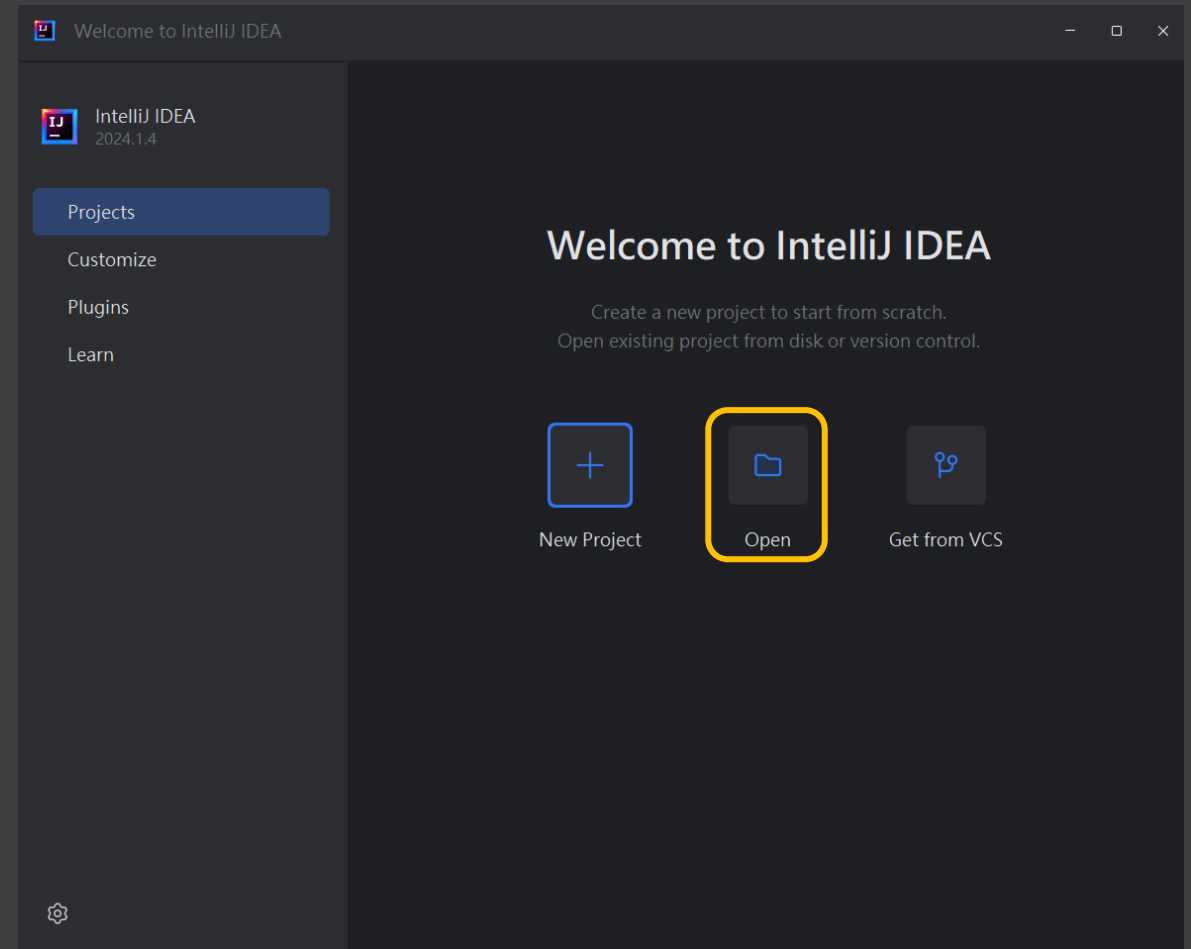
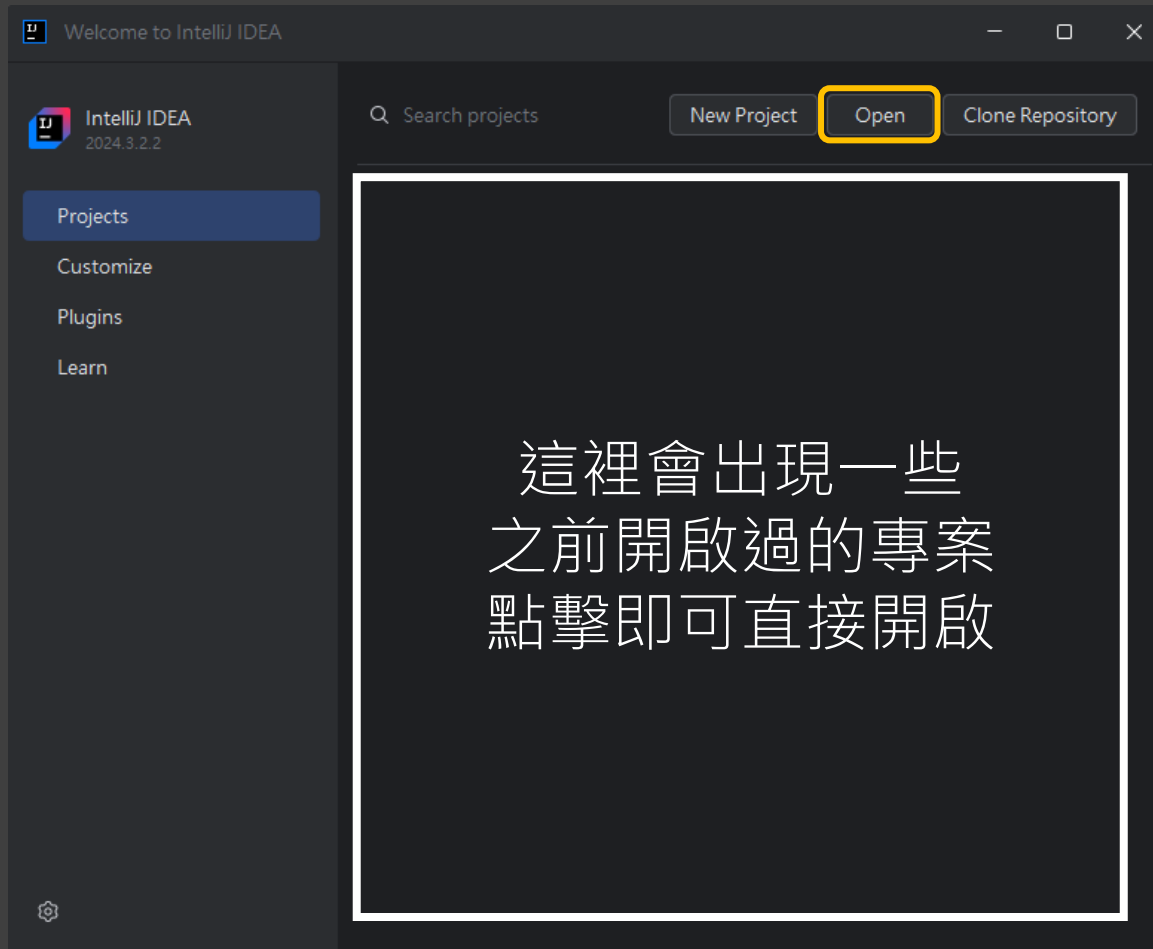
安裝完成後，選擇 **Restart IDE**

並選擇 **Restart** 重新啟動 IDE 以完成**插件**安裝



開啟專案

IDE 重新啟動後，點擊 **Open**，選擇專案資料夾，開啟該專案
也可將專案資料夾拖入 **IntelliJ IDEA**，同樣也能開啟專案

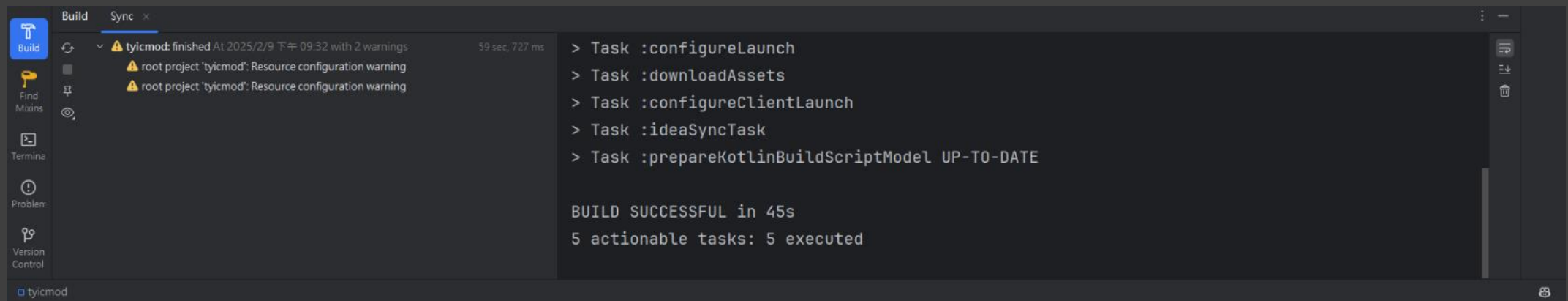


載入 Gradle 專案

開啟專案後，會自動進行載入 Gradle 專案的任務(task)
點擊左下方 **Build** 即會顯示任務執行的進度和輸出



等待任務完成再繼續，任務完成後會輸出 **SUCCESSFUL**

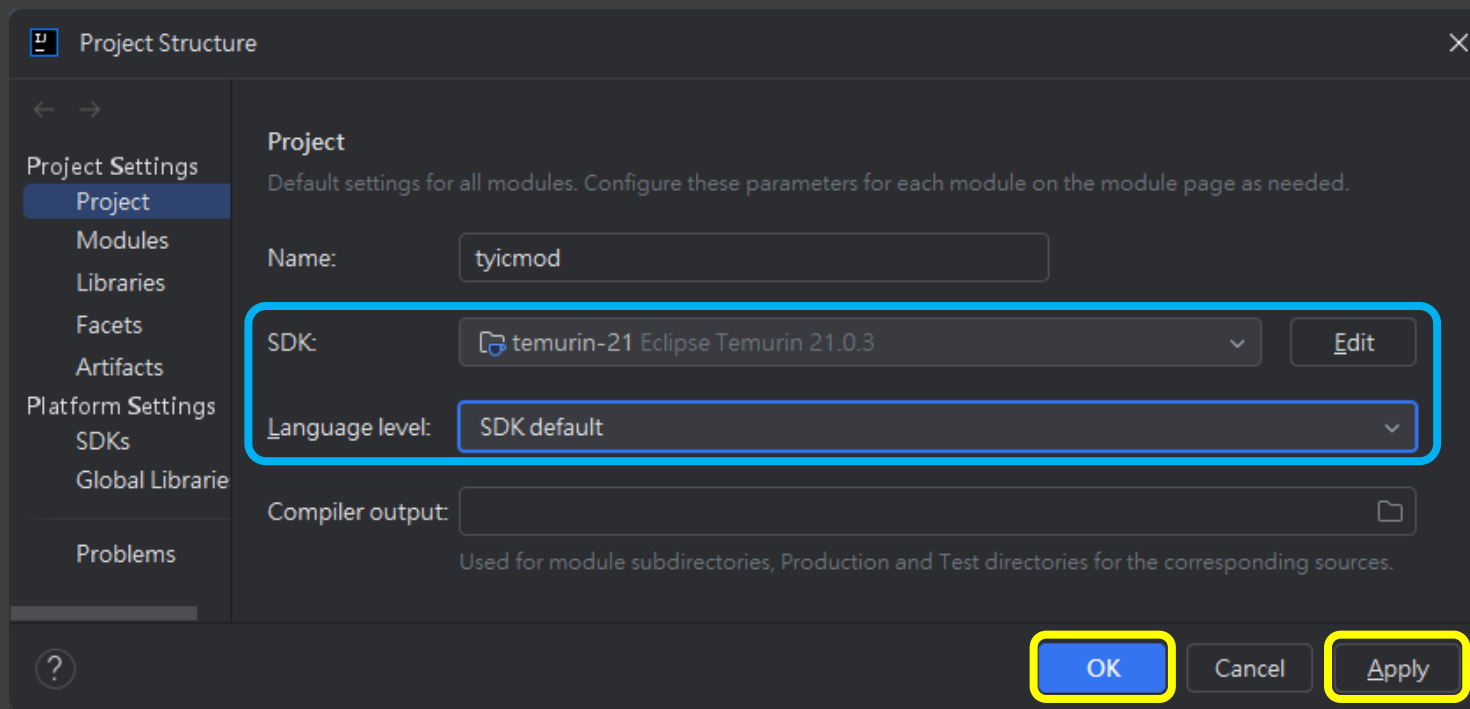
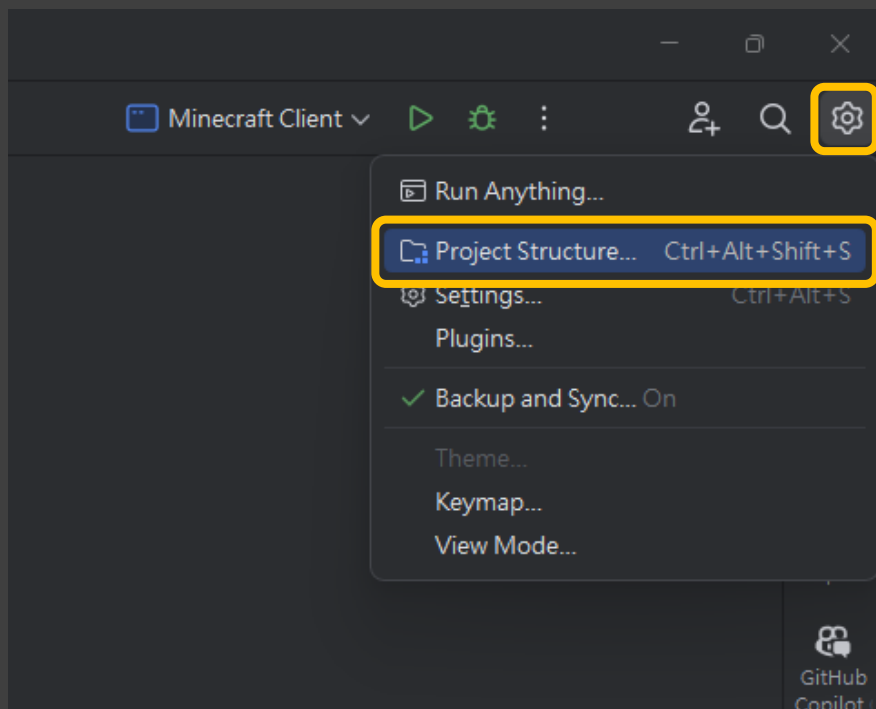


設定 JDK

載入完成後，點擊右上方設定 -> Project Structure

將 SDK 設為 Java 21，Language level 設為 SDK default

設定完成後依序點擊 Apply 套用，OK 完成



執行遊戲

右上方可選擇執行的設定檔和使用過的任務
選定想要執行的設定檔或任務後
點擊執行或除錯按鈕即可運行

預設會有三個執行設定檔：

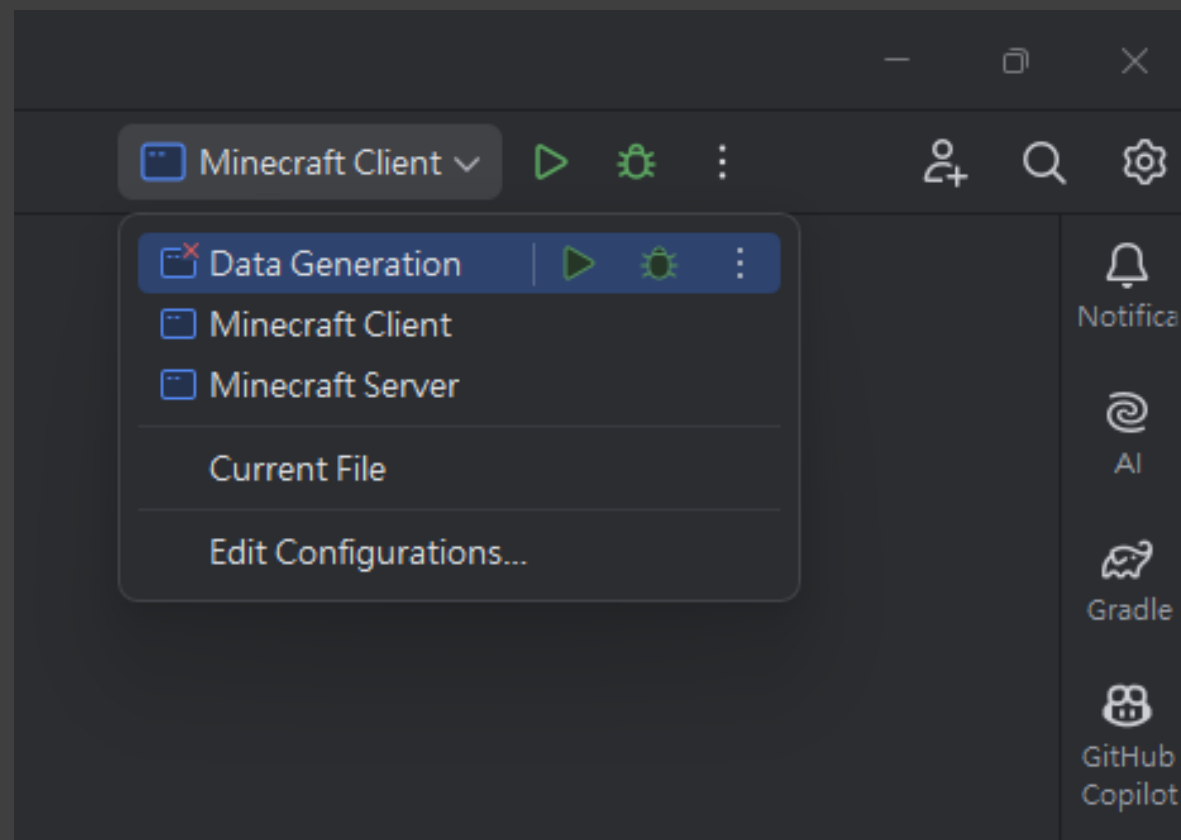
Data Generation，資料生成

Minecraft Client，Minecraft 客戶端

Minecraft Server，Minecraft 伺服器端

須注意，此 Minecraft 客戶端為離線模式

即只能加入 `online-mode` 為 `false` 的伺服器



執行遊戲

客戶端和伺服器的工作目錄皆為 `"./run"` 目錄

關於詳細的伺服器開設定教學，可參考 [Minecraft Wiki](#)：

中文：「教學：架設Java版伺服器」

英文："Tutorial:Setting up a server"

須注意：

1. 第一次開啟伺服器須將 `eula.txt` 中的 `eula` 改為 `true`
2. 須將 `server.properties` 中的 `online-mode` 設為 `false`
3. 加入伺服器時 `ip` 位址須填入 `localhost`，代表本機

專案結構

專案資料夾結構大致如右

其中的 **"./src"** 資料夾

為專案原始碼和資源的主要存放之處

main 和 **client** 為**模組(module)**

每個**模組**各自擁有自己的原始碼和資源資料夾

main **模組**為伺服器端和客戶端通用

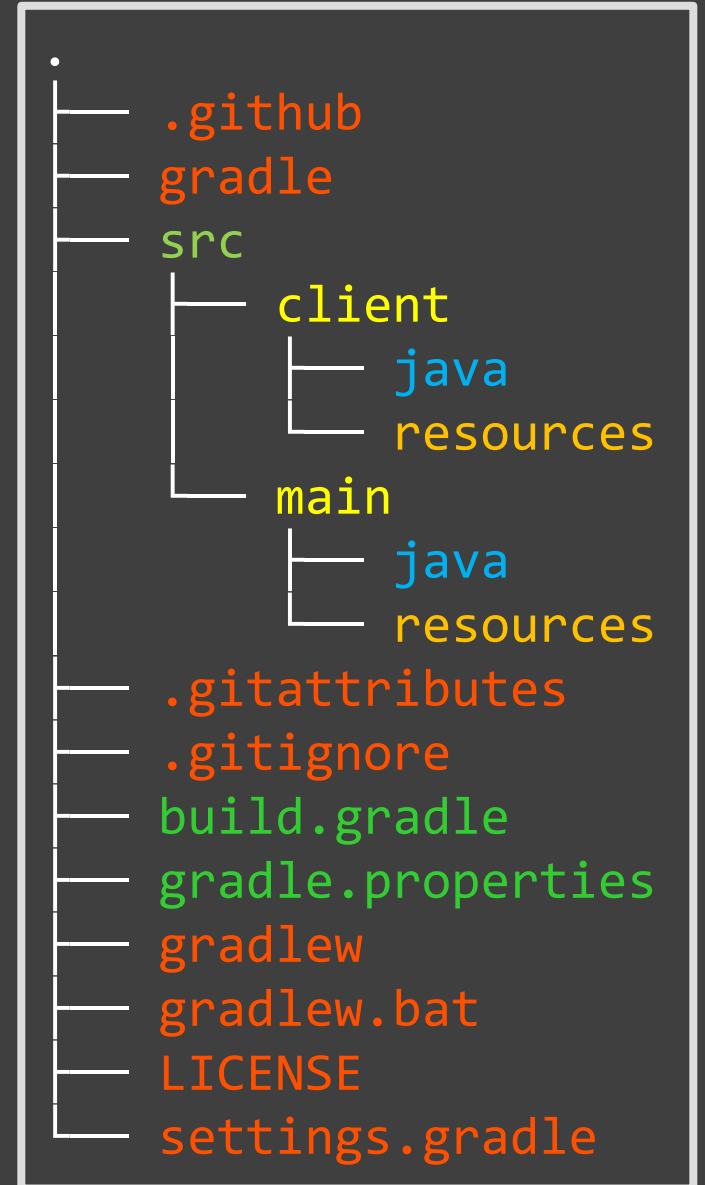
client **模組**為客戶端專屬

java 資料夾為原始碼資料夾

resources 資料夾為資源資料夾

深綠色資料夾和檔案之後會提到

紅色資料夾和檔案基本上可以不用理會



路徑

路徑(path)是一種用目錄名稱和檔案名稱表示一個檔案的方式
分為絕對路徑(absolute path)和相對路徑(relative path)

絕對路徑：一個完整的路徑

無論在何處此路徑皆指向同一檔案或資料夾

如："D:\Riot Games\VALORANT\live\VALORANT.exe"

相對路徑：一個不完整的路徑

會以當前工作目錄(working directory)

為起點去指向一個檔案或資料夾

如：工作目錄為 "D:\Riot Games\VALORANT\live"

則相對路徑 ".\VALORANT.exe" 與上方絕對路徑所指檔案相同

路徑

路徑中的 "." 表示當前目錄

".." 表示父目錄

在 Linux 中使用斜線 "/" 表示子目錄

在 Windows 中使用反斜線 "\" 表示子目錄，但也支援使用 "/"

因反斜線也用於跳脫字元，故有時會在路徑中以 "\\\" 表示

一般情況下，皆使用斜線 "/" 表示子目錄

除非在 Windows 的絕對路徑中才使用反斜線 "\"

路徑

以右方資料夾結構為例，在 **Windows** 系統下

若 **src** 目錄的絕對路徑為 **"C:\src"**，則：

icon.png 檔案的絕對路徑為

"C:\src\main\resources\assets\tyicmod\icon.png"

assets 目錄的絕對路徑為

"C:\src\main\resources\assets"

icon.png 相對於 **src** 目錄的相對路徑為

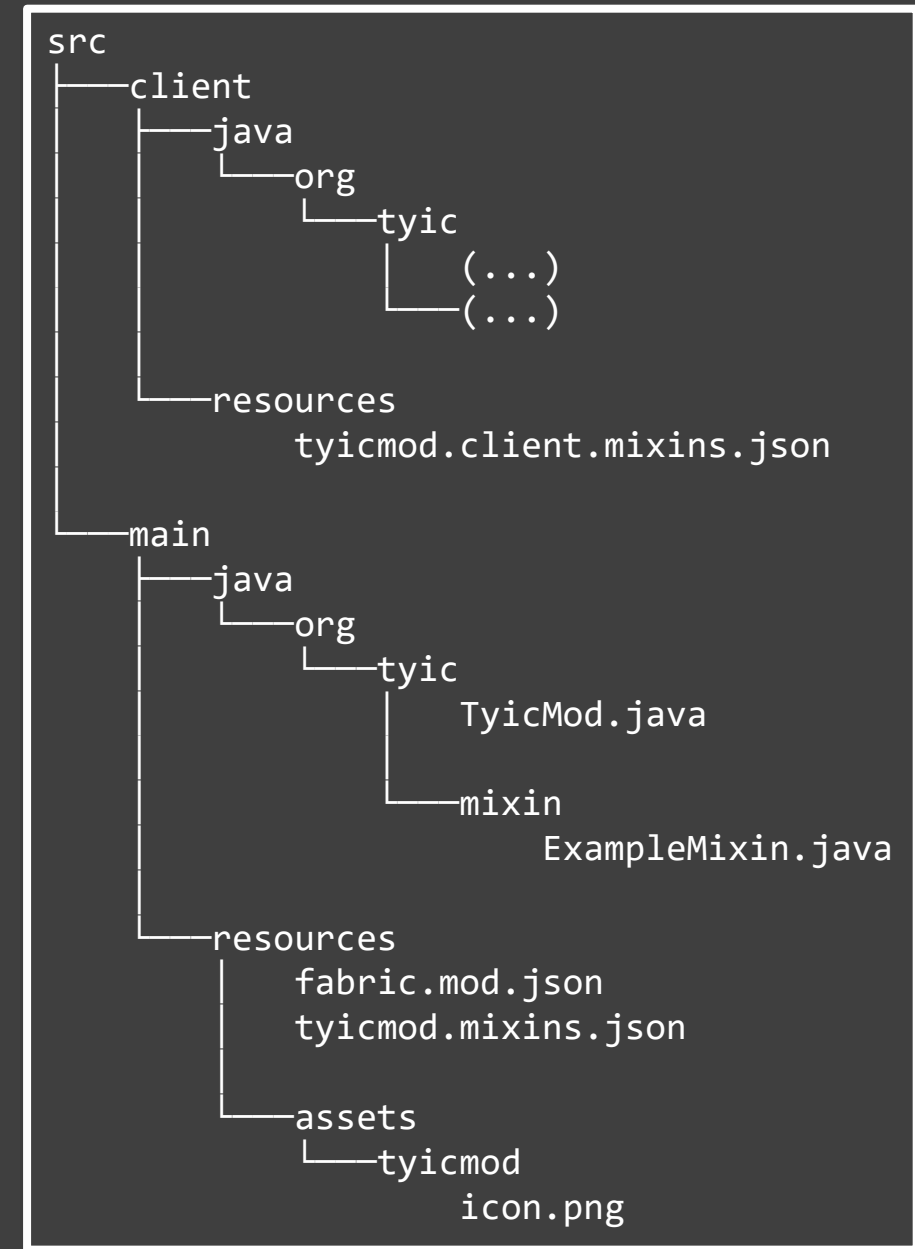
"./main/resources/assets/tyicmod/icon.png"

assets 目錄相對於 **src** 目錄的相對路徑為

"./main/resources/assets"

icon.png 檔案相對於 **assets** 目錄的相對路徑為

"./tyicmod/icon.png"



Gradle

Gradle 是一個自動組建工具(build automation)

透過寫好的腳本

自動管理專案所需依賴(dependence)

以及控制專案如何編譯和壓縮

專案根目錄下的

gradlew 和 gradlew.bat 兩個檔案

即為該專案的 gradle 腳本

前者用於 Linux，後者用於 Windows



Gradle

根目錄下的 `build.gradle` 檔案

是用來控制編譯、依賴、任務

`setting.gradle` 檔案是用來控制各個依賴的關係

而 `gradle.properties` 則是用來控制一些數值設定

這些數值部分用於控制 `gradle` 執行

部分則會在 `build.gradle` 中被使用

gradle.properties

`gradle.properties` 中的大部分欄位都很好理解

其中 `Minecraft` 和 `Fabric` 版本資訊

 小數點版本介紹

可在 <https://fabricmc.net/develop/> 下方找到

`mod_version` 代表模組的版本

使用小數點版本來表示

小數點版本為 `major.minor.build`

`major` 表大版本號，增加表重大更新

`minor` 表次版本號，增加表次要更新

`build` 表小版本號，增加表小更新

```
# Done to increase the memory available to gradle.
org.gradle.jvmargs=-Xmx1G
org.gradle.parallel=true

# Fabric Properties
# check these on https://fabricmc.net/develop
minecraft_version=1.21.4
yarn_mappings=1.21.4+build.8
loader_version=0.16.10

# Mod Properties
mod_version=1.0.0
maven_group=org.tyic
archives_base_name=tyicmod

# Dependencies
fabric_version=0.115.1+1.21.4
```

gradle.properties

Gradle

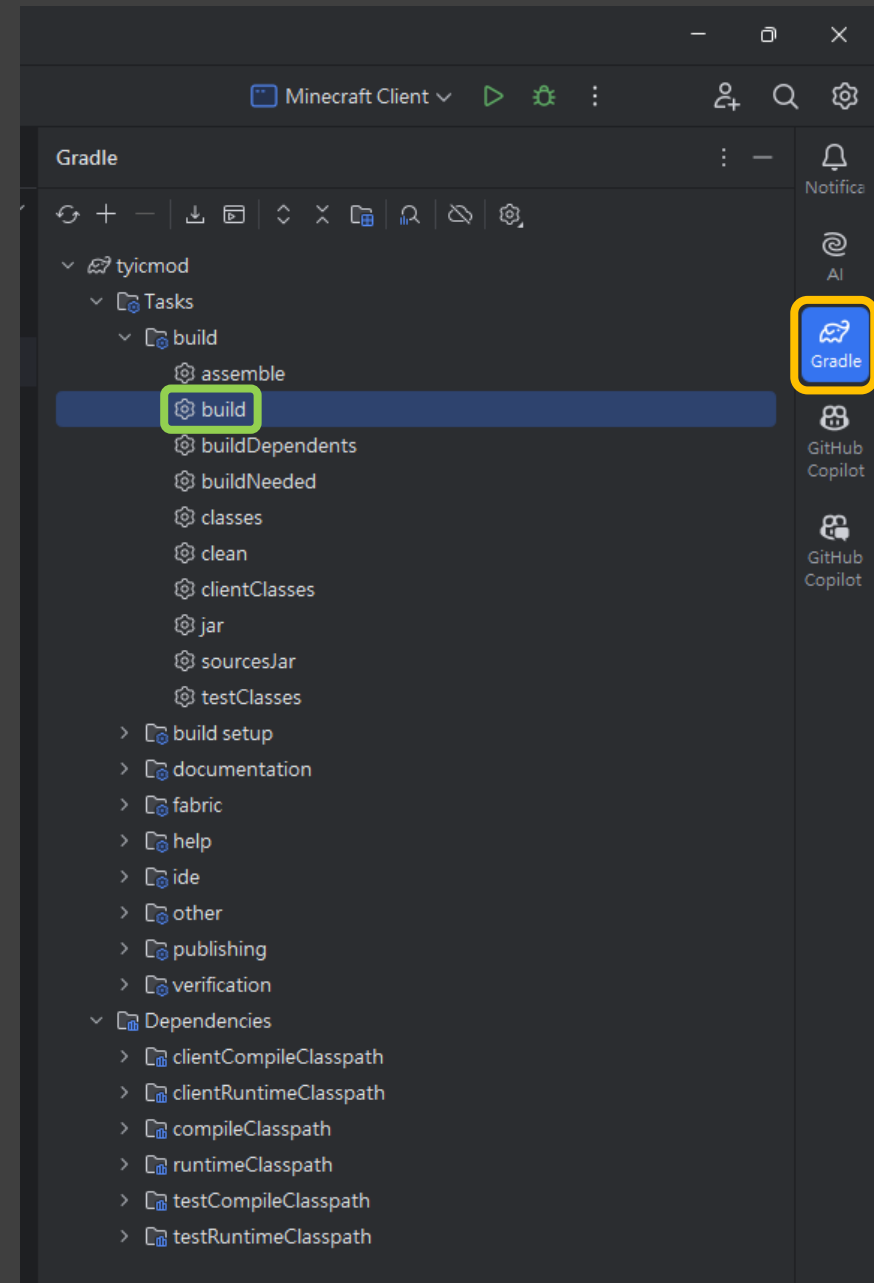
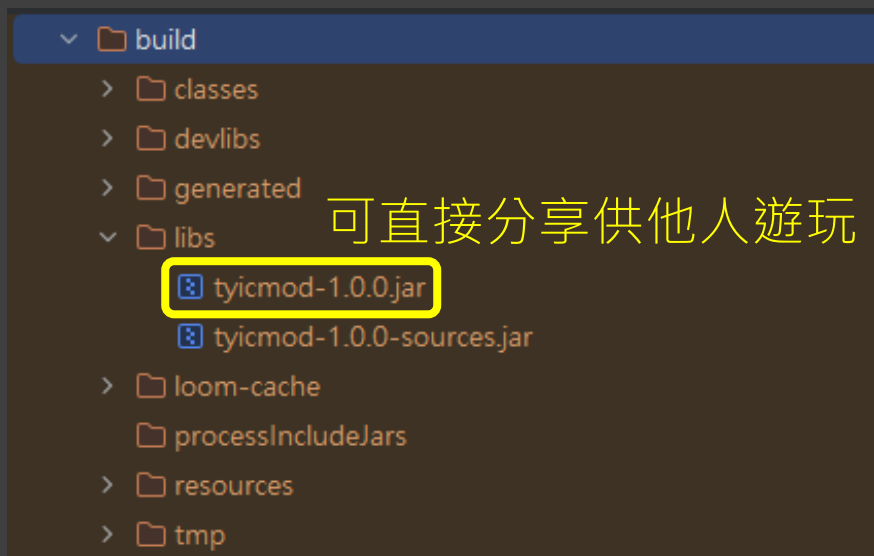
點擊右方 **Gradle**

即可看見許多 **gradle** 任務

其中 **build** 任務執行後

./build/libs 中會出現 **.jar** 檔案

其中檔名無多餘後綴的檔案即為模組檔案
可進行發佈或分享供他人遊玩



JSON

JSON (JavaScript Object Notation)

是一種簡易的資料交換格式，容易編寫也很容易解析
常常用於各個程式之間的資料交換，或是用於撰寫設定檔
在各種程式中無處不見
可以直接以一個字串表示
也可寫在檔案中，副檔名為 `.json`

大部分語言都有處理 `JSON` 的函式庫



JSON

JSON 中只有 6 種資料型別：

數值：整數或小數

字串：以一對雙引號(`"`)夾住的一段文字

陣列：以一對中括號(`[]`)夾住的若干元素，以逗號(`,`)分隔

布林：`true` 或 `false`

空：`null`

物件：以一對大括號(`{}`)夾住的若干鍵值對，但鍵須為字串

每個鍵值對以逗號(`,`)分隔，每個鍵和值以冒號(`:`)分隔

JSON

右方為

JSON 範例：

```
{
  "id": "EX1_116",
  "dbfId": 559,
  "name": "Leeroy Jenkins",
  "text": "<b>Charge</b>...",
  "flavor": "At least he has...",
  "artist": "Gabe from Penny Arcade",
  "attack": 6,
  "cardClass": "NEUTRAL",
  "collectible": true,
  "cost": 5,
  "elite": true,
  "faction": "ALLIANCE",
  "health": 2,
  "mechanics": [
    "BATTLECRY",
    "CHARGE"
  ],
  "rarity": "LEGENDARY",
  "set": "EXPERT1",
  "type": "MINION"
}
```

json

```
{
  "type": "singleChoice",
  "number": 1,
  "description": "Look at the picture...",
  "image": "112_english_1.png",
  "passing_rate": null,
  "discrimination_index": null,
  "choices": [
    {
      "answer": "A",
      "description": "bag"
    },
    {
      "answer": "B",
      "description": "basket"
    },
    {
      "answer": "C",
      "description": "bowl"
    },
    {
      "answer": "D",
      "description": "box"
    }
  ],
  "correct_answer": "B"
}
```

json

fabric.mod.json

`./src/main/resources/fabric.mod.json`

為一個 **json** 檔案，用於控制模組資訊

其中大部分內容很容易理解

將游標移動至**鍵**上方也會顯示其註解

唯須注意，**depends** 的型別是物件

該物件每個**鍵**為前置模組 **id**

值為字串，表前置模組可接受版本範圍

```
{
  "schemaVersion": 1,
  "id": "tyicmod",
  "version": "${version}",
  "name": "TyicMod",
  "description": "(...)",
  "authors": [
    "Me!"
  ],
  "contact": {
    "homepage": "https://fabricmc.net/",
    "sources": "(...)"
  },
  "license": "CC0-1.0",
  "icon": "assets/tyicmod/icon.png",
  "environment": "*",
  "entrypoints": {
    "main": [
      "org.tyic.TyicMod"
    ],
    "client": [
      "org.tyic.TyicModClient"
    ],
    "fabric-datagen": [
      "org.tyic.TyicModDataGenerator"
    ]
  },
  "mixins": [
    "tyicmod.mixins.json",
    {
      "config": "tyicmod.client.mixins.json",
      "environment": "client"
    }
  ],
  "depends": {
    "fabricloader": ">=0.16.10",
    "minecraft": "~1.21.4",
    "java": ">=21",
    "fabric-api": "*"
  },
  "suggests": {
    "another-mod": "*"
  }
}
```

json

icon.png

`./src/main/resources/assets/{modname}/icon.png`

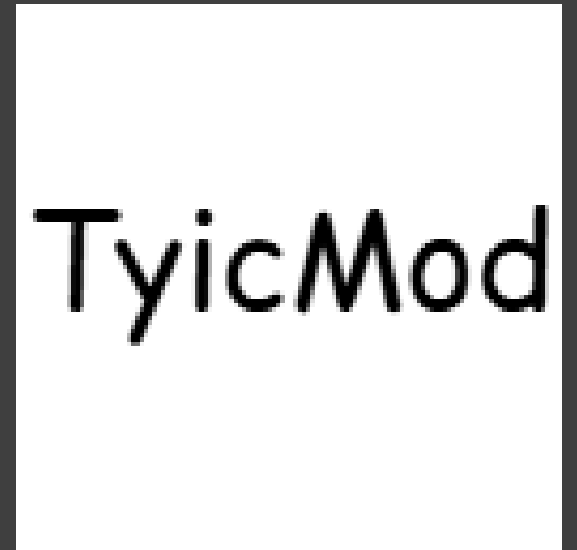
為模組的圖標

可替換成其他 **png** 格式圖片

但此圖片須為正方形(1:1)

且解析度須為 2 的次方數

預設為寫著模組名稱的自動生成圖片，如右



Minecraft 原始碼

！！！！注意！！！！

根據 Minecraft EULA

不得發布完整未混淆

Minecraft 原始碼

Minecraft 原始碼

點擊畫面左方的

Project -> External Libraries

即可展開程式庫(library)列表

其中的 net.minecraft

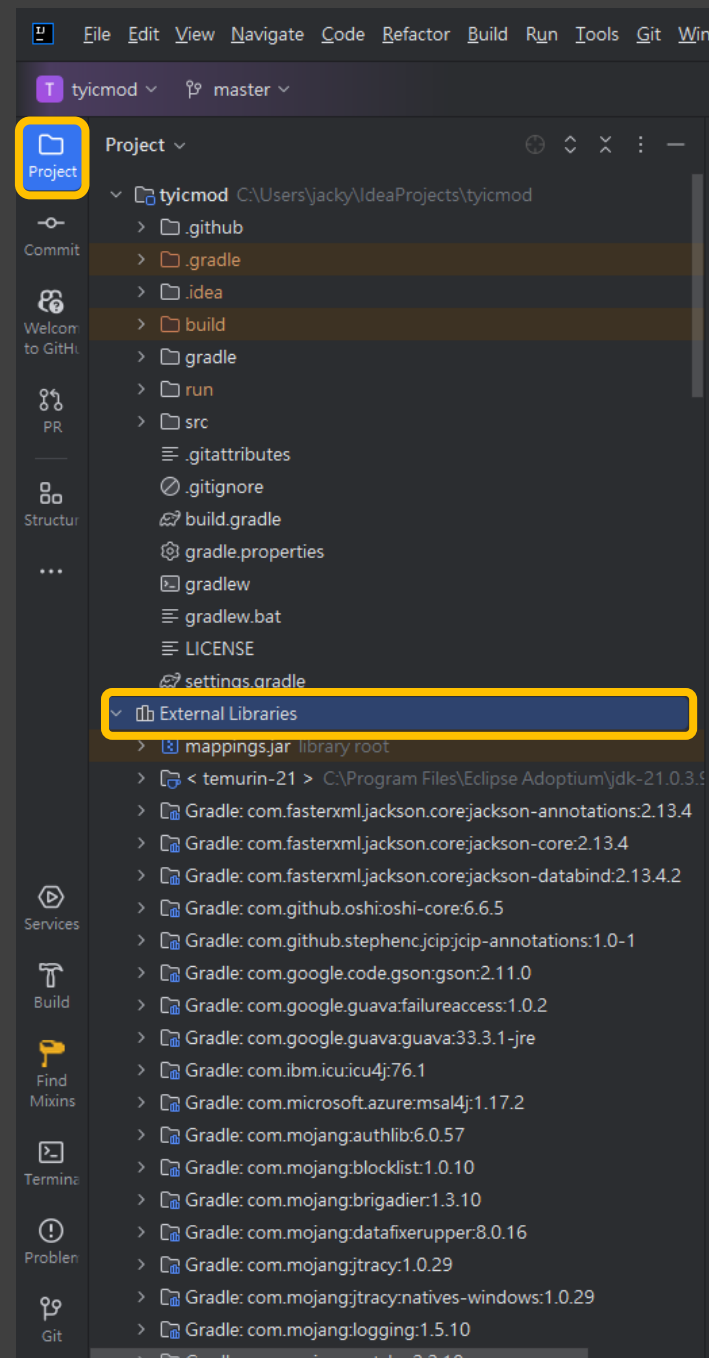
即為 Minecraft 原始碼程式庫

common 表通用程式碼

大部分程式碼和遊戲資料在此

clientOnly 表客戶端獨有

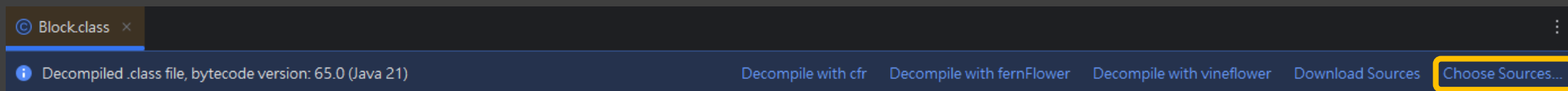
主要為渲染相關及材質



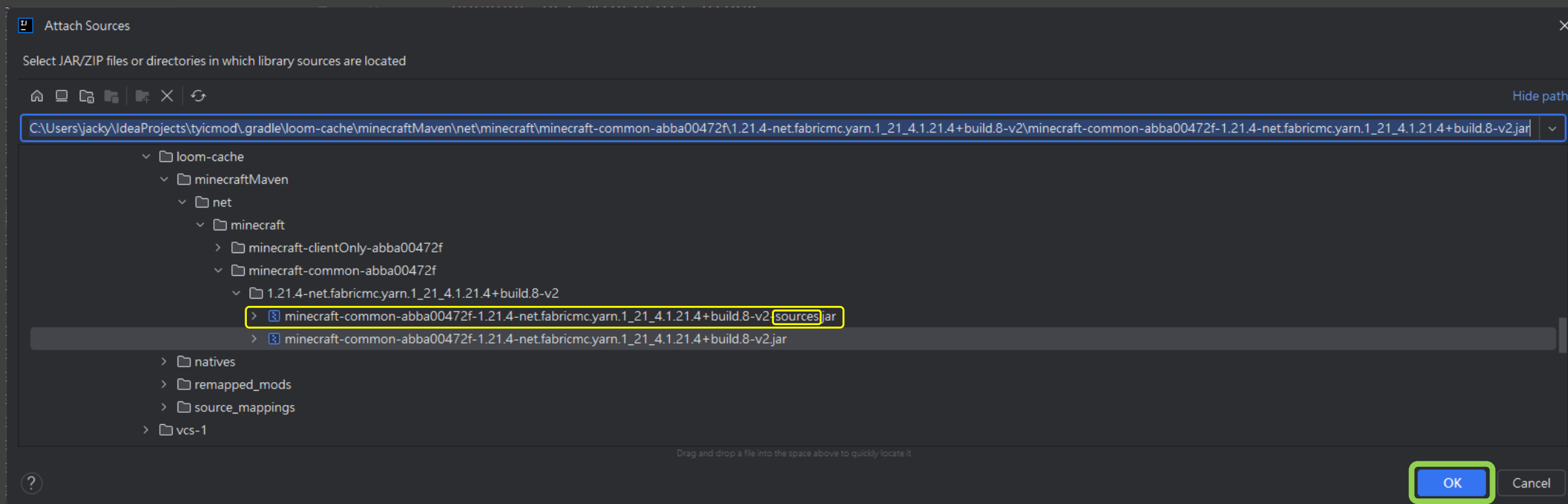
```
> Gradle: net.minecraft:minecraft-clientOnly:abba00472f:1.21.4-net.fabricmc:yarn.1_21_4.1.21.4+build.8-v2
> Gradle: net.minecraft:minecraft-common:abba00472f:1.21.4-net.fabricmc:yarn.1_21_4.1.21.4+build.8-v2
```

Minecraft 原始碼

點開原始碼，上方可能會出現此藍條：

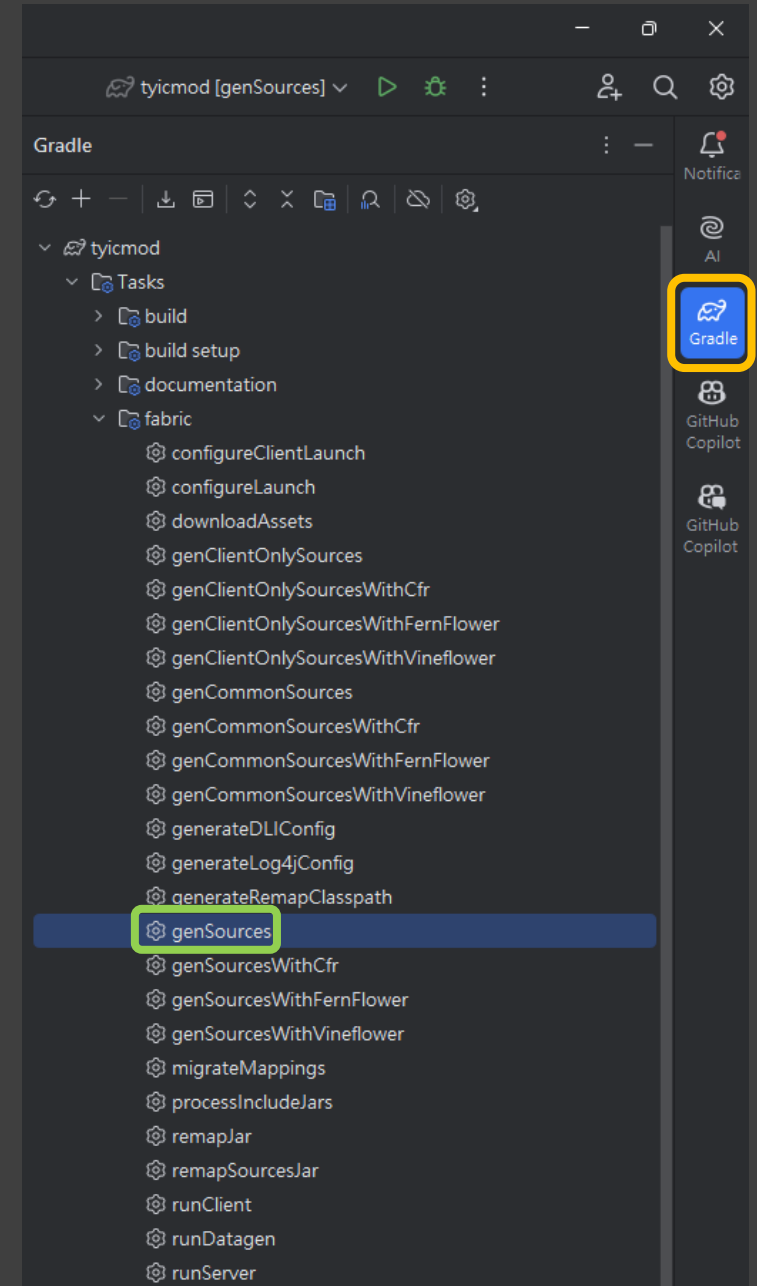


點擊 **"Choose Sources..."**，會彈出一個視窗選擇原始碼
若有以 **"sources"** 結尾的 **jar** 檔案，請選擇並點擊 **OK**



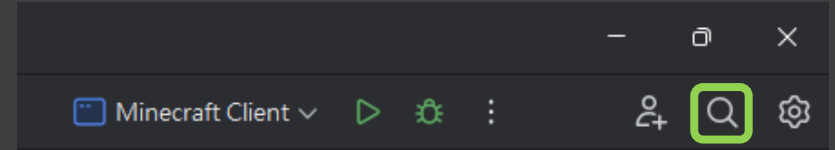
Minecraft 原始碼

若沒有以 **"sources"** 結尾的 jar 檔案
則點擊 **Cancel** 後
點擊右方 **Gradle**
並執行 **genSources** 任務
再重新點擊 **"Choose Sources..."**
選擇以 **"sources"** 結尾的 jar 檔案
並點擊 **OK**

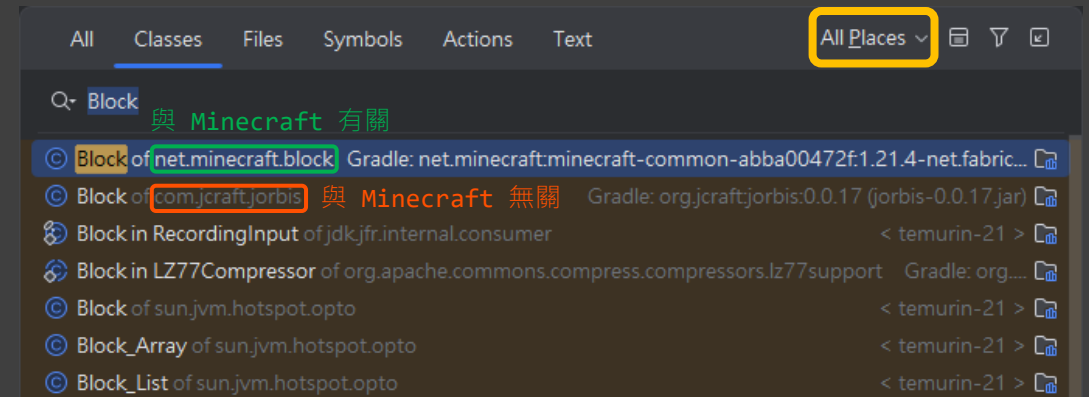
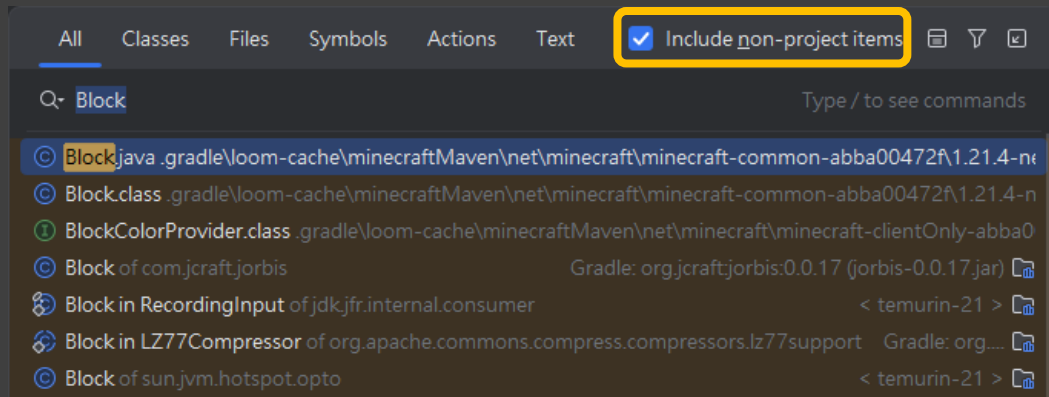


搜尋程式庫

連續按兩下 **Shift** 或點擊右上方的**放大鏡**
即可使用 **Search Everywhere**



記得需勾選 **"Include non-project items"**
或將搜尋範圍改成 **"All Places"** 才能確保搜尋的到
另需注意，可能很多個**程式庫**中有相同名稱的東西
可以從**套件**名稱或所處**路徑**來判斷，也可以直接試錯



初始化器

在 `main` 和 `client` 模組原始碼的初始套件下
分別會有 `ModName` 類別和 `ModNameClient` 類別
他們分別實作 `net.fabricmc.api.ModInitializer` 介面
和 `net.fabricmc.api.ClientModInitializer` 介面
並分別實現 `onInitialize` 方法和 `onInitializeClient` 方法
當模組被載入時，便會呼叫這兩個方法
進行模組初始化(`initialization`，簡稱 `init`)

初始化器

ModName 類別

還有兩個

公開靜態不可變欄位

其中 MOD_ID 欄位

會在之後使用到

```
package org.tyic;

import net.fabricmc.api.ModInitializer;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class TyicMod implements ModInitializer {
    public static final String MOD_ID = "tyicmod";

    // This logger is used to write text to the console and the log file.
    // It is considered best practice to use your mod id as the logger's name.
    // That way, it's clear which mod wrote info, warnings, and errors.
    public static final Logger LOGGER = LoggerFactory.getLogger(MOD_ID);

    @Override
    public void onInitialize() {
        // This code runs as soon as Minecraft is in a mod-load-ready state.
        // However, some things (like resources) may still be uninitialized.
        // Proceed with mild caution.

        LOGGER.info("Hello Fabric world!");
    }
}
```

java

```
package org.tyic;

import net.fabricmc.api.ClientModInitializer;

public class TyicModClient implements ClientModInitializer {
    @Override
    public void onInitializeClient() {
        // This entrypoint is suitable for setting up client-specific logic, such as rendering.
    }
}
```

java