

信息科学与工程学院

软件工程 实验报告

**系 别 信息科学与工程学院**

**专 业 计算机科学与技术**

**年 级 2021级**

**姓 名 徐昊博**

**指导教师 阮彤**

**2024 学年 第 二 学期**

**实验七、基于Maven Build的自动构建**

1. **实验目的**

1. 理解什么是自动构建，对比什么是手动构建。

2. 理解JUnit测试框架的用途。

**二、实验主要内容**

**（1）了解什么是MAVEN以及MAVEN的原理，相关知识**

**1、Maven的核心功能**

依赖管理：Maven自动管理项目所需的库和插件依赖，通过定义POM文件中的依赖项，Maven会自动下载所需的JAR文件及其依赖项。

构建管理：Maven提供了一套标准化的构建过程，包括编译、测试、打包、部署等步骤，通过插件来实现这些过程。

生命周期管理：Maven有一个预定义的生命周期（例如，default、clean、site），其中每个生命周期包含一系列的阶段，每个阶段可以绑定不同的任务。

插件扩展：Maven通过插件来扩展其功能，几乎所有的任务都是通过插件来实现的，包括编译代码、运行测试、生成报告等。

项目报表：Maven可以生成项目的各种报表，包括代码覆盖率、单元测试结果、依赖关系等。

**2、Maven的原理**

POM文件：Maven项目的核心是POM文件（pom.xml），其中包含项目的配置信息和依赖项声明。POM文件定义了项目的构建过程、依赖关系和插件。

仓库：Maven使用仓库来存储和管理依赖项。默认的中央仓库是Maven Central，但也可以配置使用公司内部的私有仓库。Maven在构建项目时会从这些仓库中下载所需的依赖项。

生命周期：Maven定义了一系列的生命周期，每个生命周期包含多个阶段。常见的生命周期包括：

clean：清理上一次构建的输出。

default：处理项目的构建，包括编译、测试、打包等。

site：生成项目的站点文档。

插件：Maven的功能主要通过插件实现。插件是在Maven执行不同生命周期阶段时实际运行的代码。例如，编译插件负责编译代码，测试插件负责运行单元测试，打包插件负责打包项目等。

依赖管理：Maven通过在POM文件中声明依赖项来自动管理项目的依赖关系。Maven会根据依赖关系树下载和解析所需的JAR包，确保项目有正确的版本和依赖关系。

配置继承和聚合：Maven支持POM文件的继承和聚合。父POM文件可以定义通用的配置和依赖关系，子项目可以继承这些配置。聚合允许多个子项目组合成一个多模块项目，共享配置和依赖关系。

**3、工作流程**

创建POM文件：编写包含项目基本信息、依赖关系和插件的POM文件。

执行Maven命令：使用Maven命令（如mvn clean install）来构建项目。

Maven解析POM文件：Maven读取并解析POM文件，确定项目的依赖关系和构建过程。

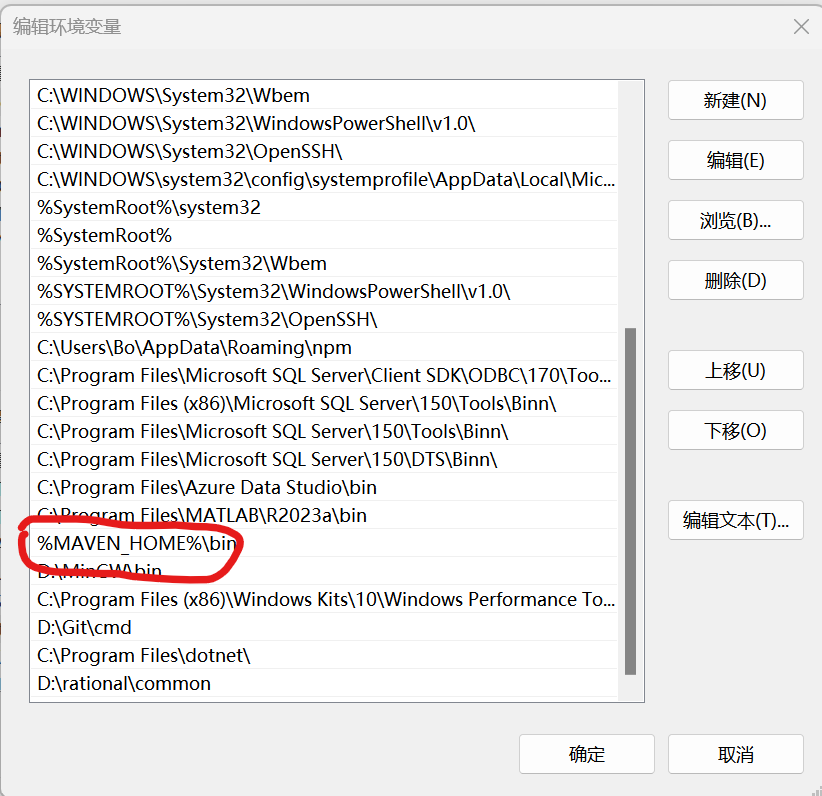
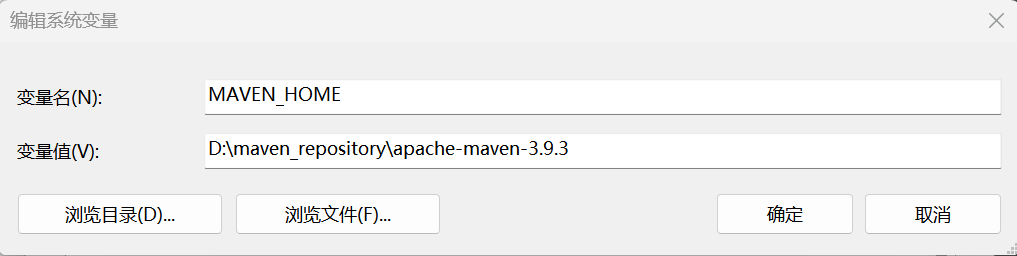
下载依赖项：Maven从配置的仓库中下载所需的依赖项。

执行插件任务：根据定义的生命周期和阶段，Maven执行相应的插件任务，如编译、测试、打包等。

生成输出：Maven根据构建过程生成最终的构建输出，如JAR包、WAR包或报告等。

**（2）通过cmd命令进行基于MAVEN基于Maven Build的自动构建**

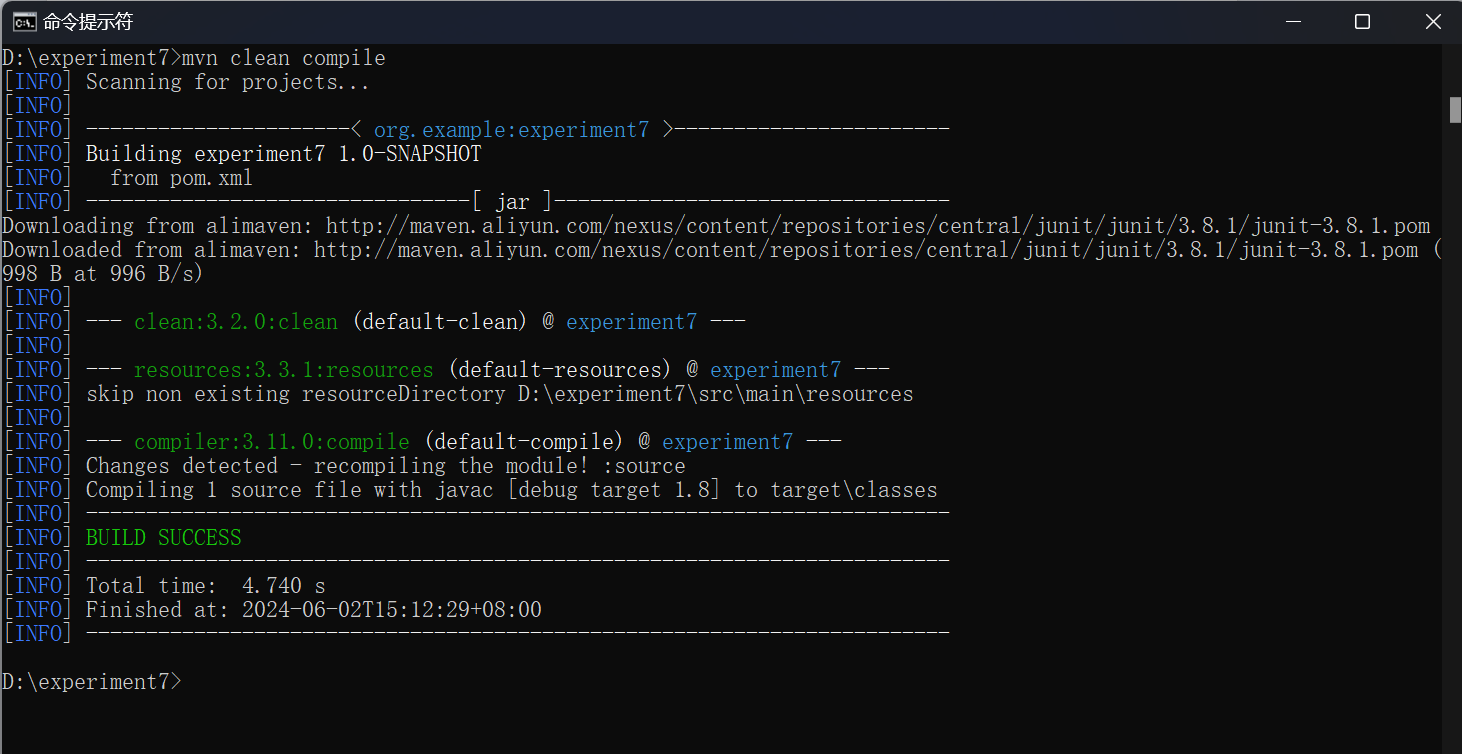
**三、实验过程**

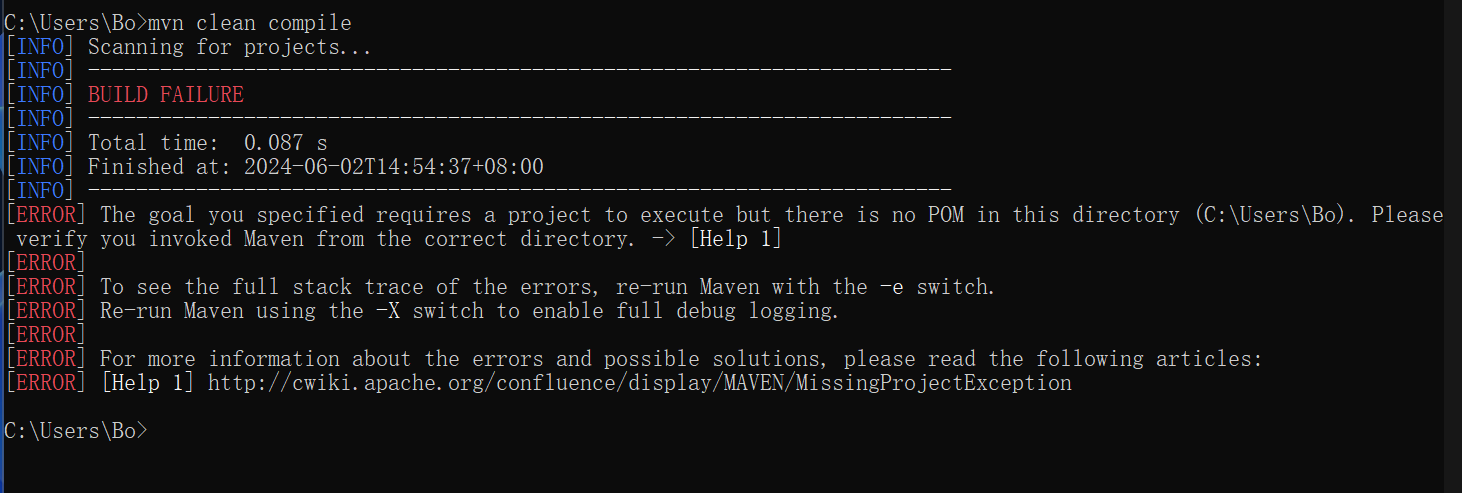
**（1）下载MAVEN，进行环境变量的配置如下所示：**

在cmd当中输入echo &MAVEN\_HOME%和mvn -version如下可见已经下载成功

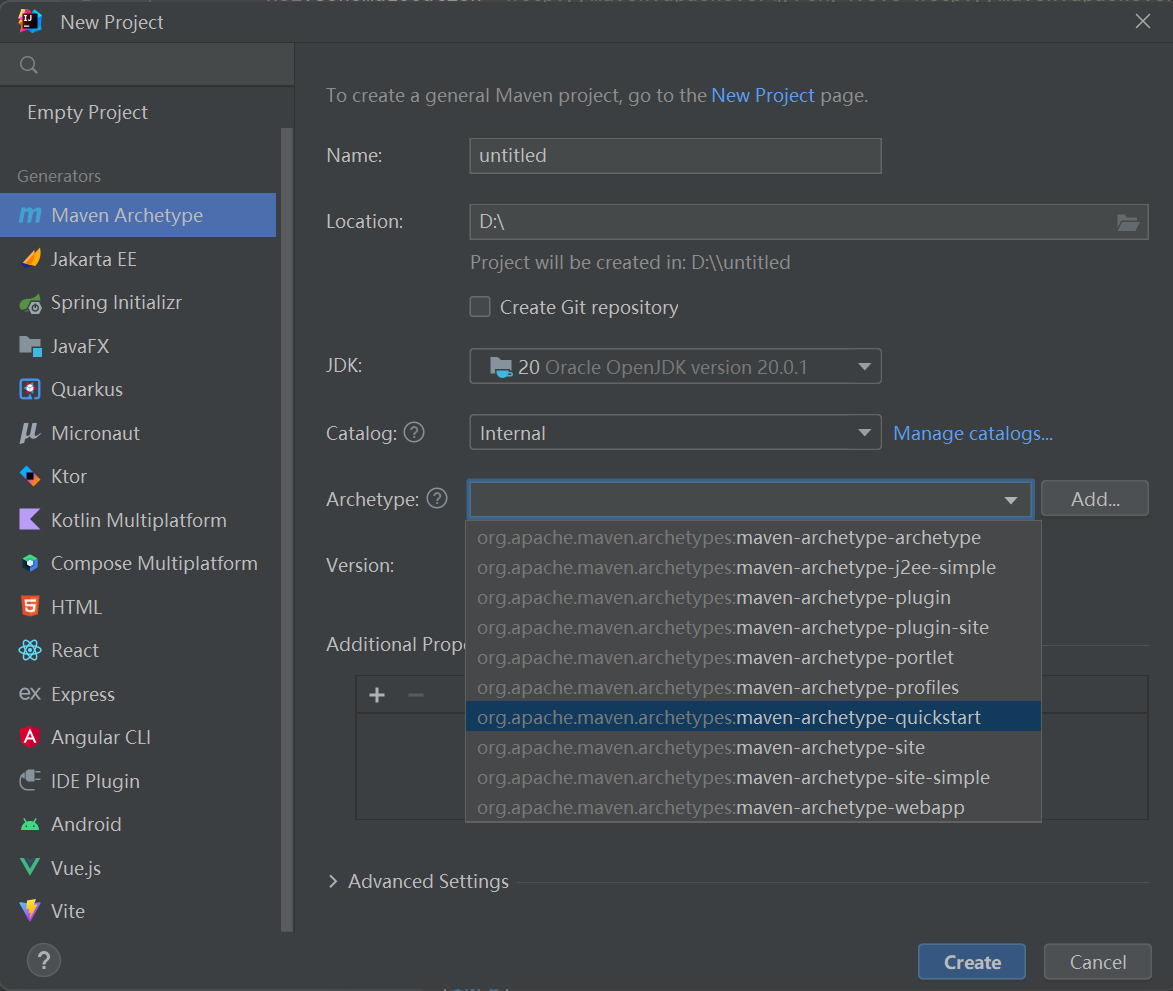
****

1. **在系统目录下键入mvn clean compile,结果如下:**

****

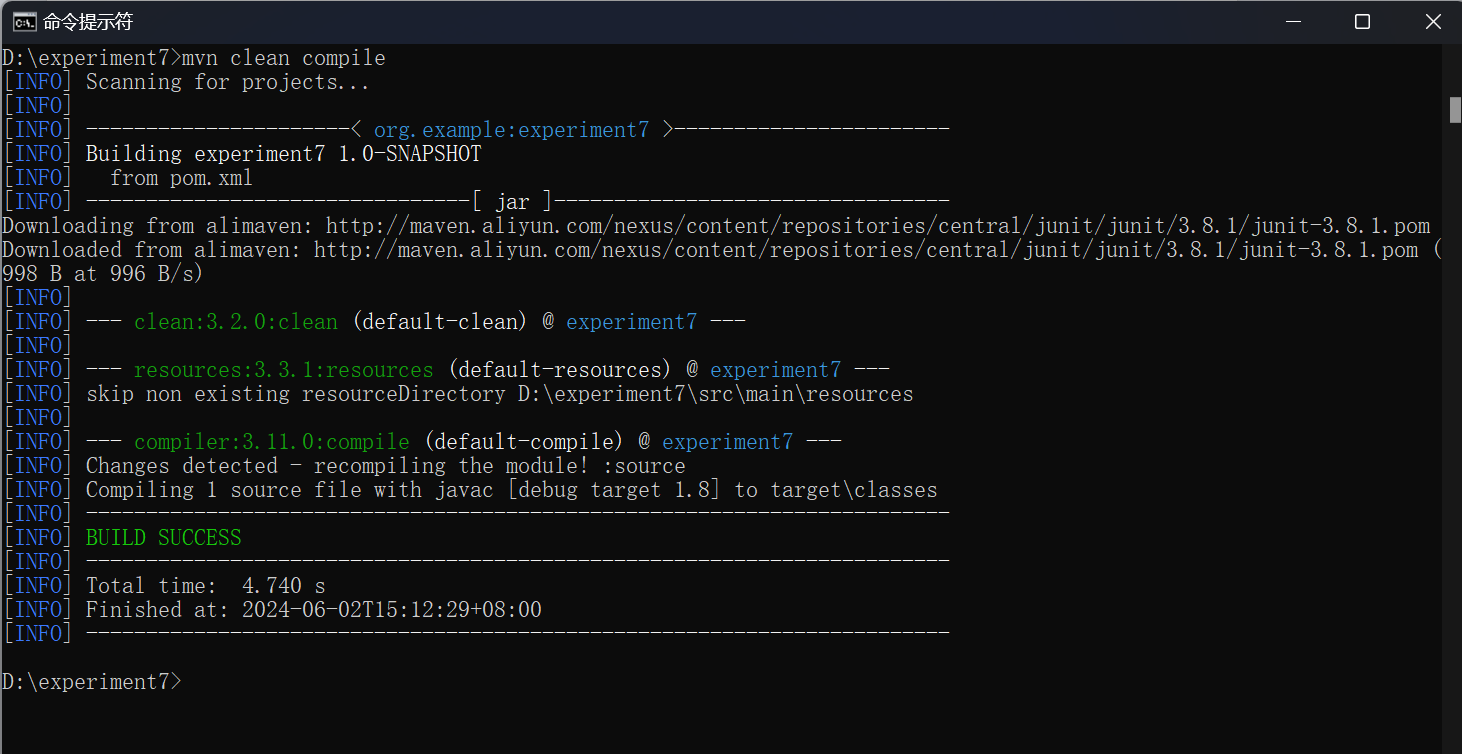
运行之后发现BUILD FAILURE,因为路径不对，找不到pom.xml，根本原因在于没有新建项目

在IDEA中新建MAVEN项目，Archetype选择为maven-archetype-quickstart如下所示

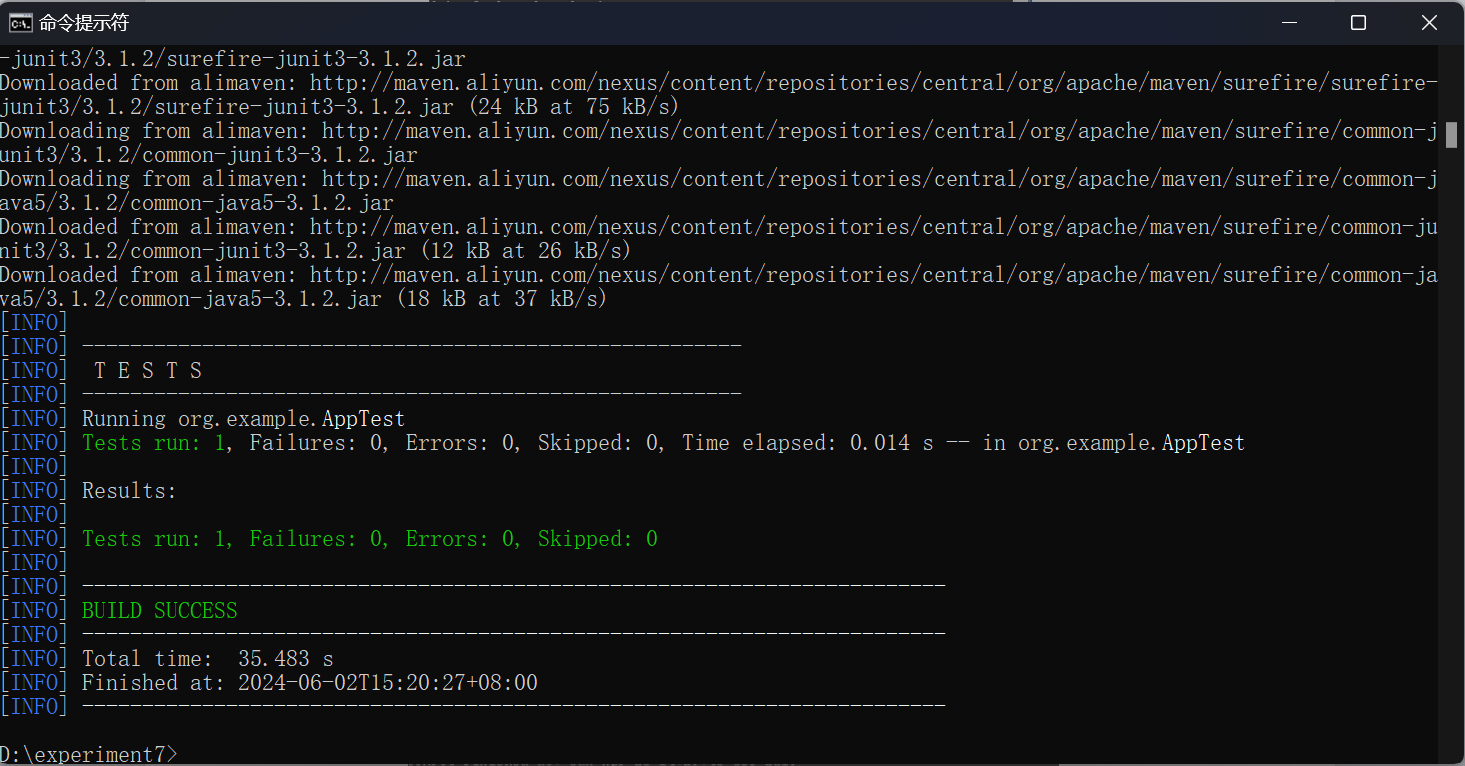


查找到新建的项目之后进入pom.xml的目录D:\experiment7\target

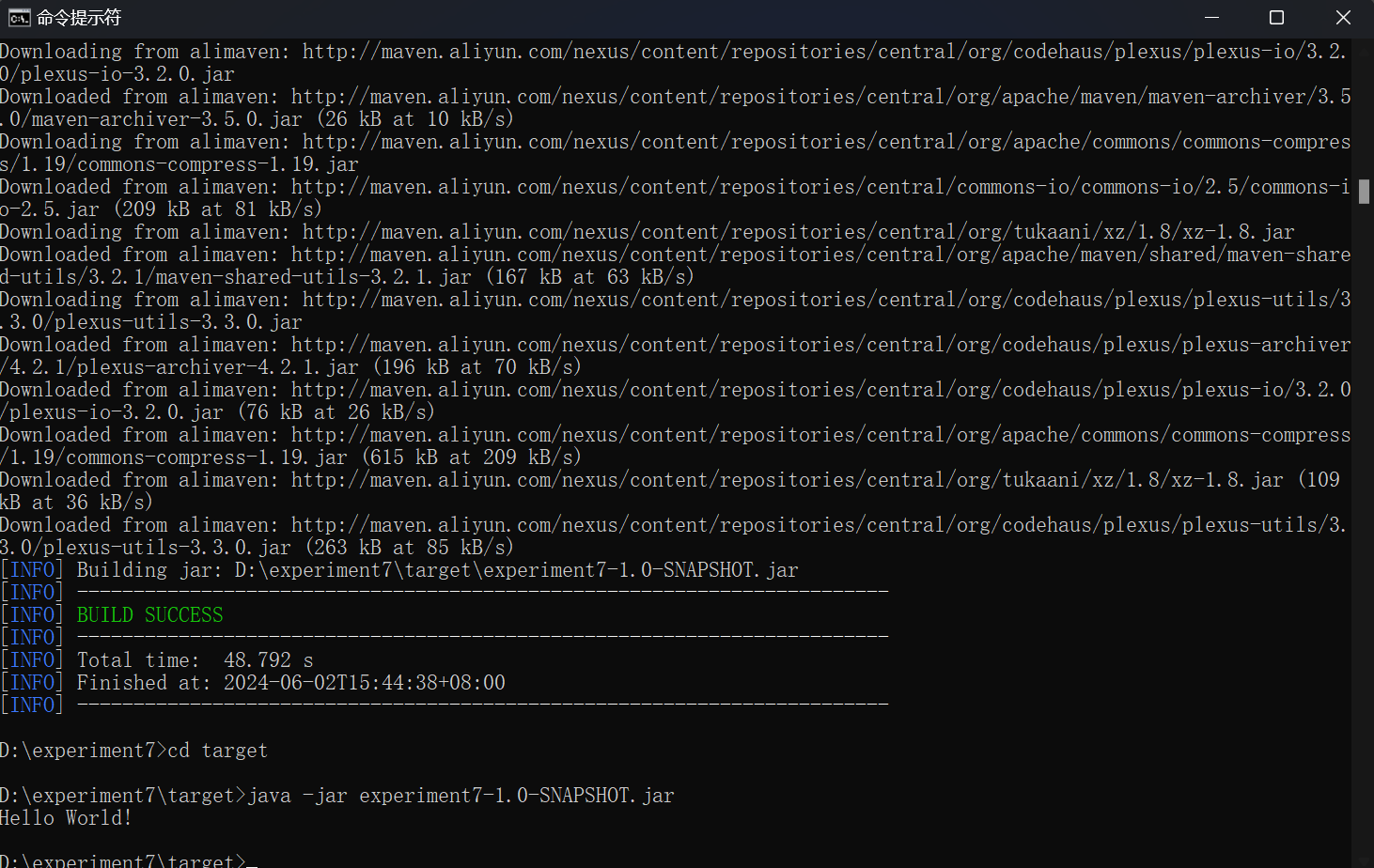
再次键入mvn clean compile,查看编译结果BUILD SUCCESS:



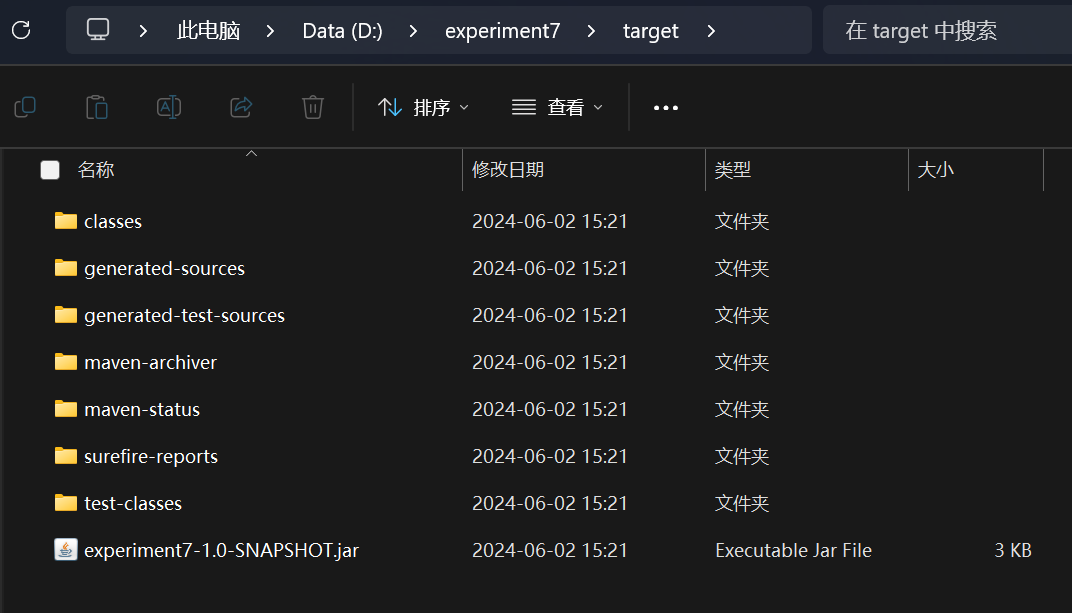
得到新增target目录

**（3）键入命令mvn clean test测试**

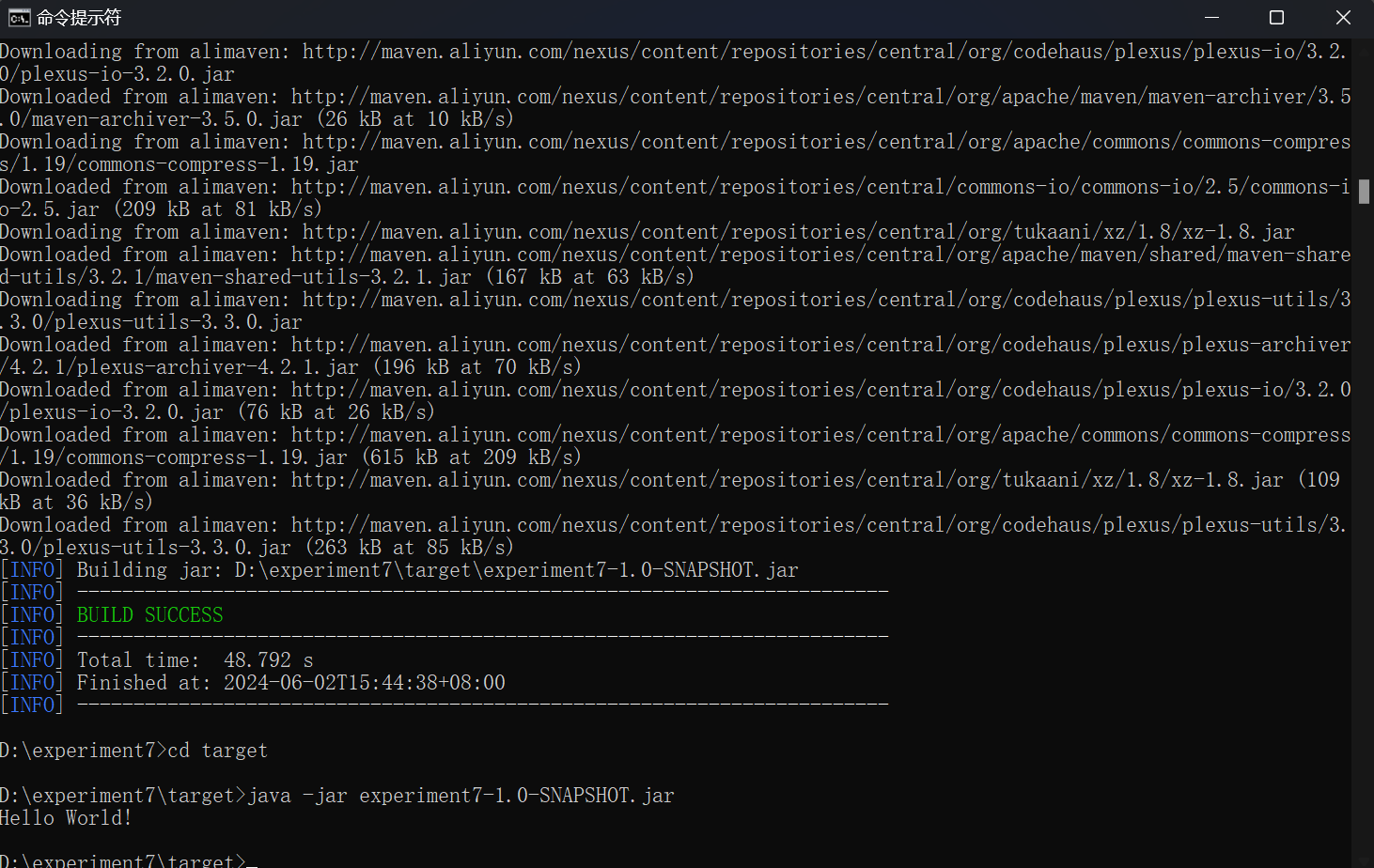
**（4）键入mvn clean package:**

****

1. **查看target目录，存在experiment7-1.0-SNAPSHOT.jar文件：**



1. **Cd命令进入target目录下，键入java –jar experiment7-1.0-SNAPSHOT.jar运行experiment7-1.0-SNAPSHOT.jar文件，结果如下：**



1. **实验总结和收获**

**1、实验总结**

在本次实验中，我们通过使用 Maven 工具实现了一个简单的 Java 项目的自动构建过程。实验的主要任务包括：

1. 安装和配置 Maven 环境。

2. 创建一个新的 Maven 项目。

3. 配置 `pom.xml` 文件以指定项目的依赖和构建信息。

4. 编写 Java 代码并配置 Maven 插件以生成包含主清单属性的 JAR 文件。

5. 使用 Maven 构建项目并生成 JAR 文件。

6. 运行生成的 JAR 文件验证结果。

**2、实验收获**

1. 掌握了 Maven 的基本使用方法：

学习了如何使用 Maven 创建和管理 Java 项目，包括项目的目录结构和配置文件。

了解了 Maven 项目的生命周期，以及如何使用 Maven 命令进行项目的清理、编译、打包等操作。

2. 深入理解了pom.xml文件的配置：

了解了pom.xml文件中各个元素的作用，包括项目属性、依赖管理和插件配置。

掌握了如何通过配置maven-jar-plugin插件来生成包含主清单属性的可执行 JAR 文件。

3. 增强了自动化构建的能力：

通过实践，体验了 Maven 自动化构建工具的强大功能，体会到了自动化构建在项目开发和管理中的重要性。

学习了如何利用 Maven 进行项目的依赖管理，避免了手动下载和配置依赖库的繁琐过程。

4. 提高了项目构建效率：

通过 Maven 的自动化构建流程，显著提高了项目的构建效率，减少了人为错误的可能性。

了解了如何使用 Maven 构建大型项目，并将项目划分为多个模块进行管理。

5. 实用的调试和运行技巧：

学习了如何在 IntelliJ IDEA 等集成开发环境中配置和运行 Maven 项目，掌握了调试和运行 Java 项目的实用技巧。

通过命令行和 IDE，熟悉了不同环境下构建和运行 Maven 项目的方法。

本次实验通过实际操作和配置，使我全面掌握了 Maven 的使用方法和自动构建技术。这些技能不仅提高了我的开发效率，还为我今后参与大型项目的开发和管理打下了坚实的基础。在未来的项目中，我将更加熟练地使用 Maven 进行依赖管理和项目构建，提升项目开发的自动化水平和质量。