Lab Program 6

Build and Run a Java Application with Maven. Migrate the same Application to Gradle. Create a Maven Project. Understand the POM file, Dependency Management and Plugins.

PART – 1: Build and Run a Java Application with Maven

1. Prerequisites required

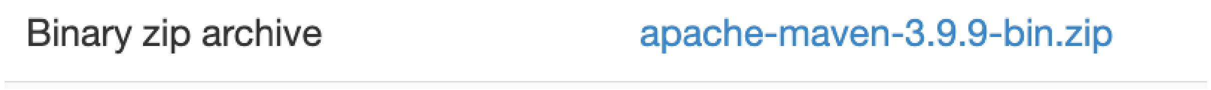
- Java JDK (version 21 preferable) must be installed.
- Confirm Java installation by typing the below command in command prompt:

```
java -version
```

Step 1: Download and install Maven from: https://maven.apache.org/download.cgi

- Install Binary Zip file only.

->apache-maven-3.9.6-bin.zip



Binary zip archive   apache-maven-3.9.9-bin.zip

✕ Do NOT Download:

- src.zip – contains source code (not needed for installation)
- .tar.gz – intended for Linux/Unix systems

Step 2 : Add Maven to your system path:

- MAVEN_HOME → Path to Maven folder
- Add MAVEN_HOME/bin to the PATH

After Downloading:

1. **Extract** the zip file to a location like:

   C:\Program Files\Apache\Maven\apache-maven-3.9.6

2. **Set environment variables** using setx (as explained below)
3. Open Command Prompt (Run as Administrator) and execute the following :

```
> setx MAVEN_HOME "C:\Program Files\Apache\Maven\apache-maven-3.9.6" /M

> setx PATH "%PATH%;C:\Program Files\Apache\Maven\apache-maven-3.9.6\bin" /M
```

4. Restart your Command Prompt and check:

   -> Check whether is Maven is properly installed

5. Open Command Prompt and execute the following :

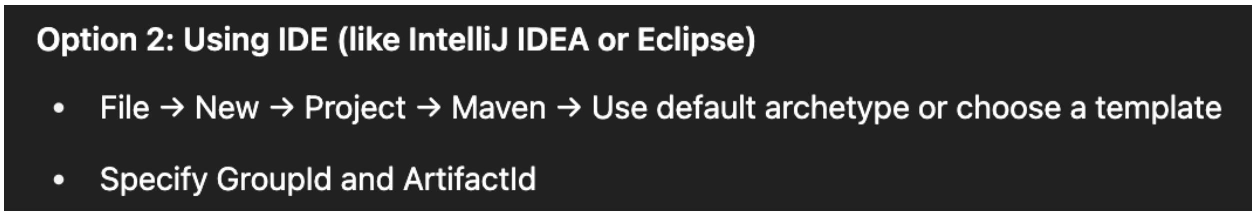
```
> mvn -v
```

Step 3: Create a Maven Project

-> Type the following in the cmd prompt

> mvn archetype:generate -DgroupId=com.example -DartifactId=demo-project -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

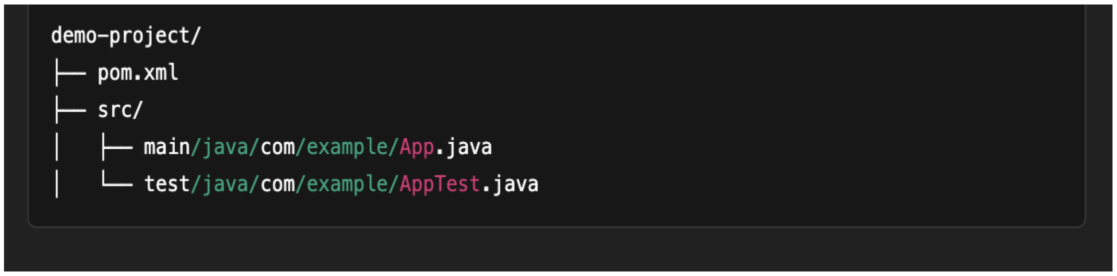
OR



Option 2: Using IDE (like IntelliJ IDEA or Eclipse)

- File → New → Project → Maven → Use default archetype or choose a template
- Specify GroupId and ArtifactId

-> After the Execution the above cmd on the command prompt, the project structure looks like below.



```
demo-project/
├── pom.xml
├── src/
│   ├── main/java/com/example/App.java
│   └── test/java/com/example/AppTest.java
```

Step 4: Add a Dependency

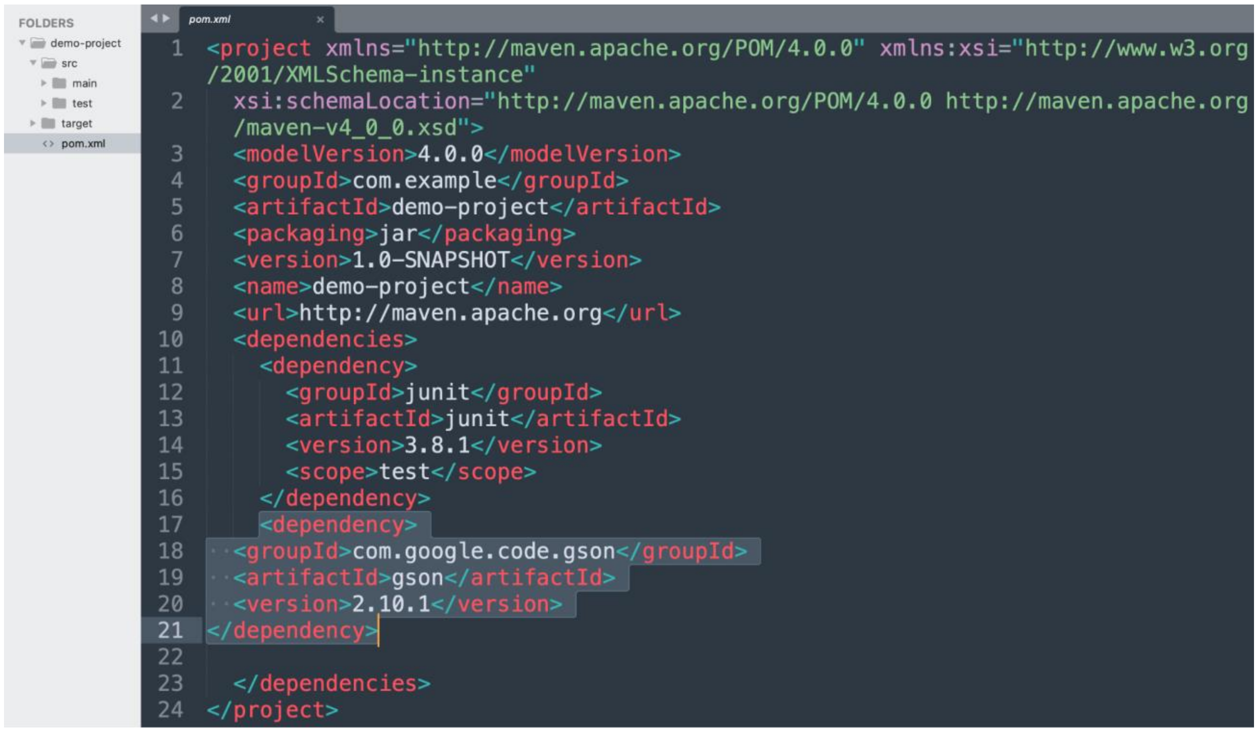
-> Step 4 : Go to the demo-project folder and understand the usage of pom.xml file.

-> To use a library, add its dependency in the <dependencies> block,

Example: Add Gson (for JSON parsing):

-> Add the below dependency code section to the pom.xml file.

```
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.10.1</version>
</dependency>
```



```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>demo-project</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>demo-project</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.10.1</version>
</dependency>

  </dependencies>
</project>
```

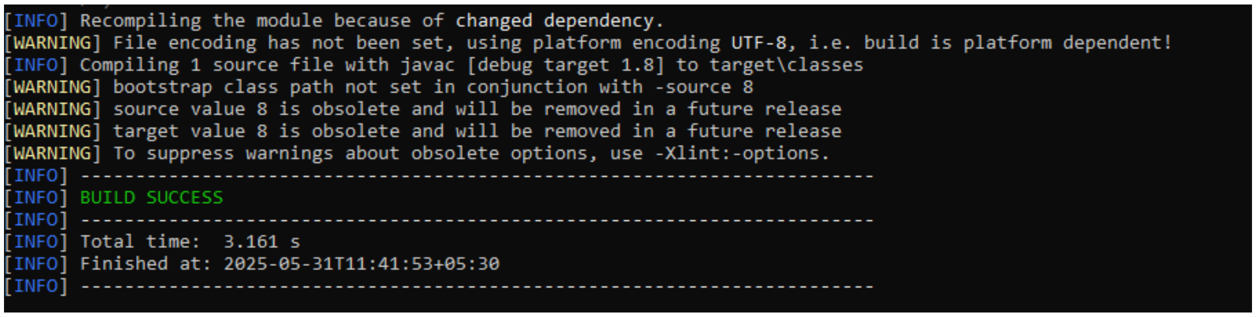
-> In the same way you can add any number of dependencies required for your application.

Step 5: Building Maven Project

-> Change the directory to demo-project :

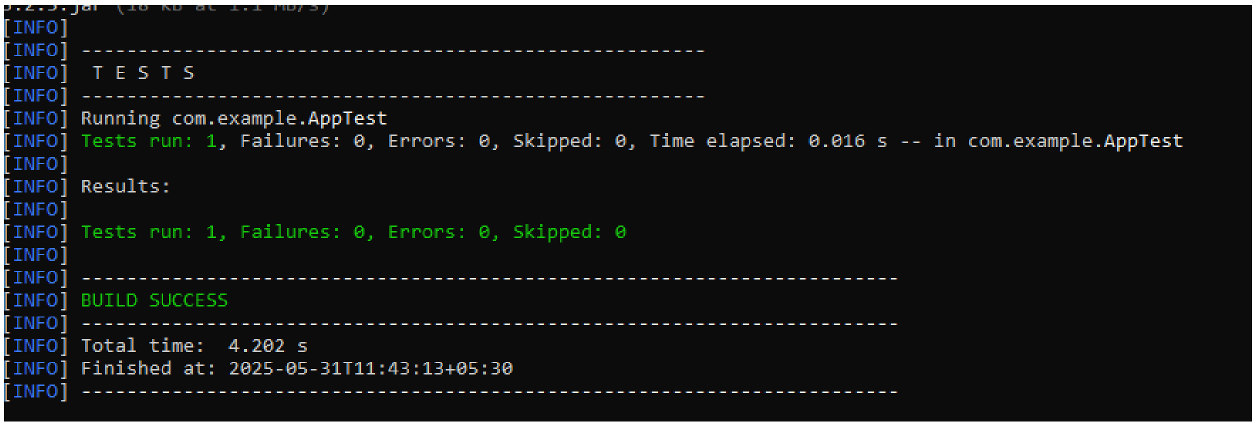
> **cd demo-project**

> **mvn compile**



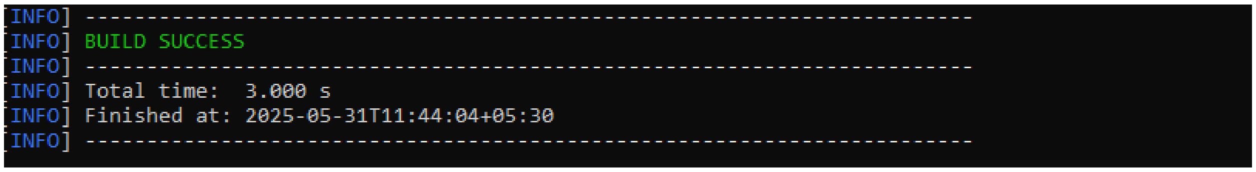
```
[INFO] Recompiling the module because of changed dependency.
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file with javac [debug target 1.8] to target\classes
[WARNING] bootstrap class path not set in conjunction with -source 8
[WARNING] source value 8 is obsolete and will be removed in a future release
[WARNING] target value 8 is obsolete and will be removed in a future release
[WARNING] To suppress warnings about obsolete options, use -Xlint:-options.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.161 s
[INFO] Finished at: 2025-05-31T11:41:53+05:30
[INFO] ------------------------------------------------------------------------
```

> mvn test



```
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.016 s -- in com.example.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  4.202 s
[INFO] Finished at: 2025-05-31T11:43:13+05:30
[INFO] ------------------------------------------------------------------------
```

> mvn package

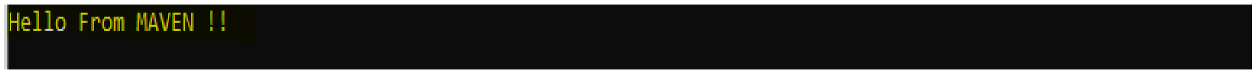


```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.000 s
[INFO] Finished at: 2025-05-31T11:44:04+05:30
[INFO] ------------------------------------------------------------------------
```

Step 6: Run the JAR File

-> If you have a main() method in App.java, run the JAR like this:

java -cp target/demo-project-1.0-SNAPSHOT.jar com.example.App



```
Hello From MAVEN !!
```

Note : The above steps is to run a Java Application using Maven.

PART 2: Migrate the Same Application to GRADLE

Step 1: Delete Maven-Specific Files

In the demo-project/ folder:

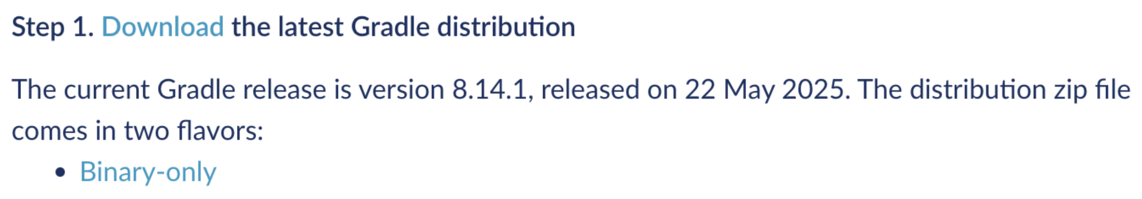
- Delete pom.xml

Do **NOT** delete the src/ folder — we reuse the code.

Step 2: Installation of Gradle

Download Gradle(version 8.x.x and above)

- Go to the official site: https://gradle.org/releases/
- Download the **binary-only ZIP** (not the complete source code).


Step 1. Download the latest Gradle distribution

The current Gradle release is version 8.14.1, released on 22 May 2025. The distribution zip file comes in two flavors:

- Binary-only

-> click on Binary - only link , the binary zip file will be downloaded.

1. Extract the zip file to **C:\Gradle\gradle-8.x.x** folder location. (8.x.x - > specify the downloaded version)
2. **Set environment variables** using setx (as explained below)
3. Open Command Prompt (Run as Administrator) and execute the following :

```
> setx GRADLE_HOME "C:\Gradle\gradle-8.x.x" /M
```

[Note: 8.x.x - > specify the downloaded version]

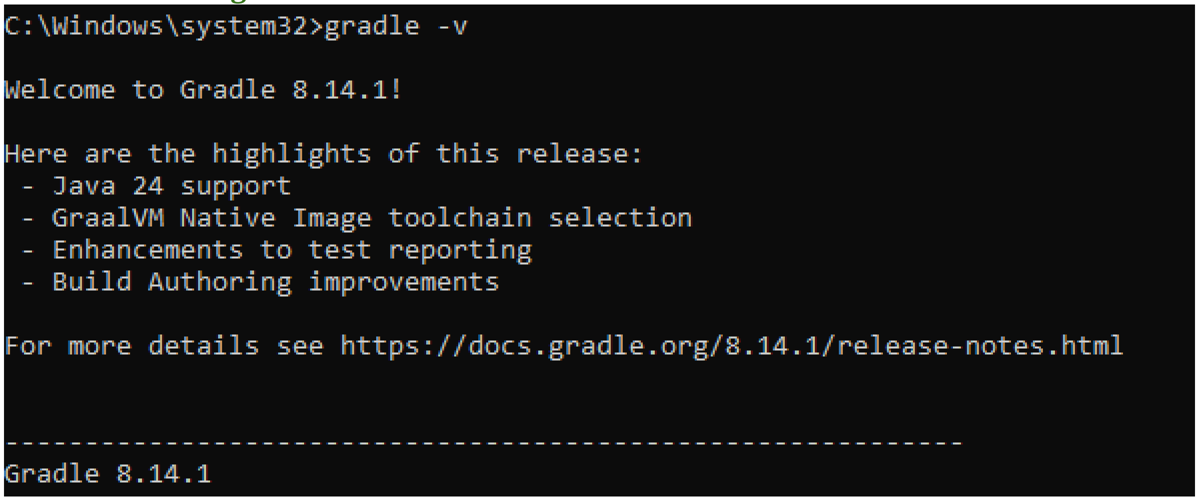
```
> setx PATH "%PATH%;%GRADLE_HOME%\bin" /M
```

4. Restart your Command Prompt and check:

-> Check whether is Gradle is properly installed

5. Open Command Prompt and execute the following :

> **gradle -v**



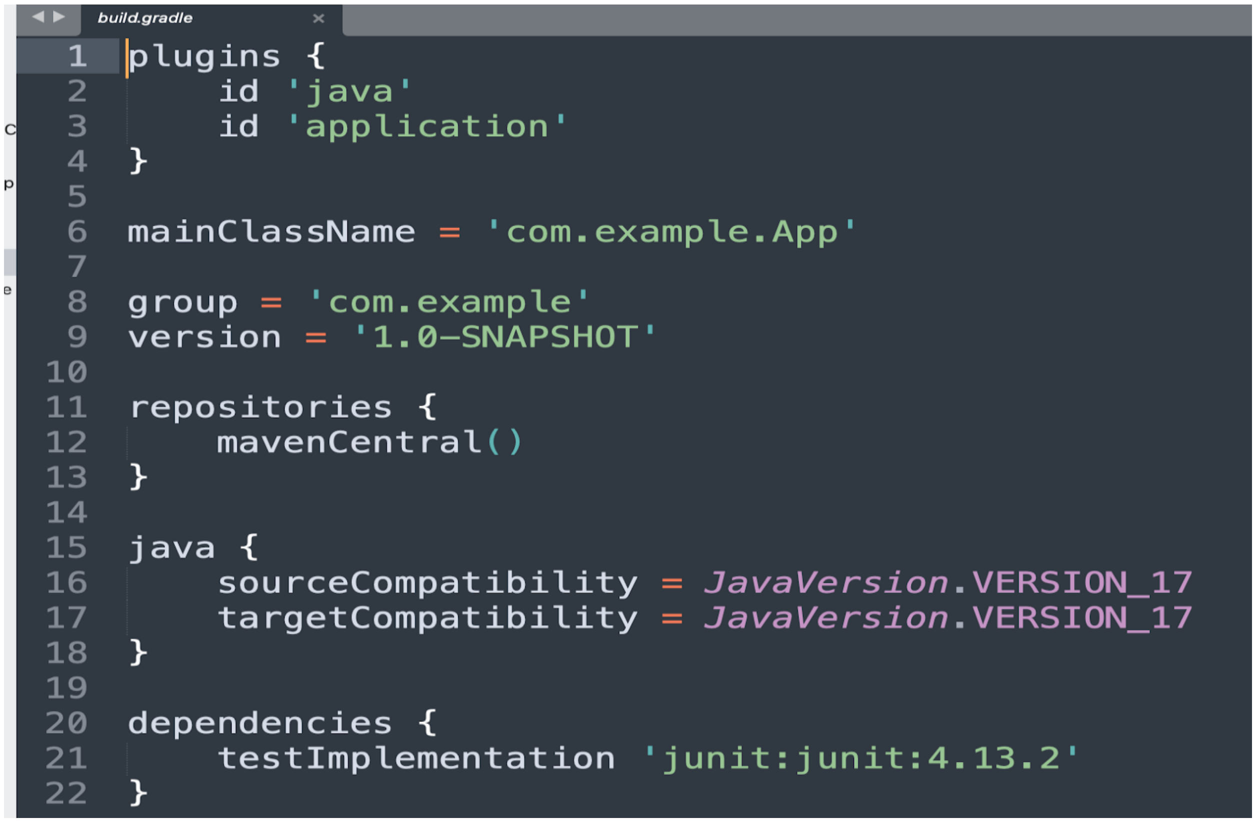
```
C:\Windows\system32>gradle -v

Welcome to Gradle 8.14.1!

Here are the highlights of this release:
 - Java 24 support
 - GraalVM Native Image toolchain selection
 - Enhancements to test reporting
 - Build Authoring improvements

For more details see https://docs.gradle.org/8.14.1/release-notes.html


------------------------------------------------------------
Gradle 8.14.1
```

Step 3: Create Gradle Files

Create two new files in the demo- project folder

File 1: build.gradle

```
plugins {
   id 'java'
}

group = 'com.example'
version = '1.0-SNAPSHOT'

repositories {
   mavenCentral()
}

dependencies {
   testImplementation 'junit:junit:4.13.2'
}
```



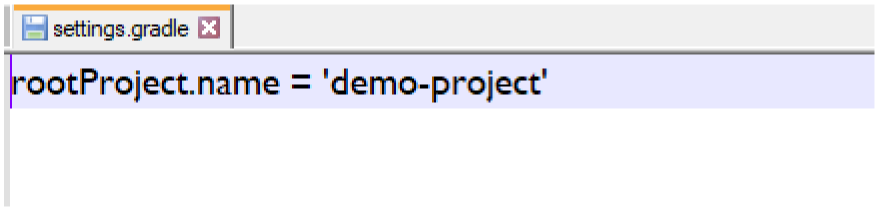
```
plugins {
	id 'java'
	id 'application'
}

mainClassName = 'com.example.App'

group = 'com.example'
version = '1.0-SNAPSHOT'

repositories {
	mavenCentral()
}

java {
	sourceCompatibility = JavaVersion.VERSION_17
	targetCompatibility = JavaVersion.VERSION_17
}

dependencies {
	testImplementation 'junit:junit:4.13.2'
}
```

Note : incase of Version discrepancies add the below code to build.grade file

```
java {
   sourceCompatibility = JavaVersion.VERSION_17
   targetCompatibility = JavaVersion.VERSION_17
}
```

File 2: settings.gradle

rootProject.name = 'demo-project



```
rootProject.name = 'demo-project'
```

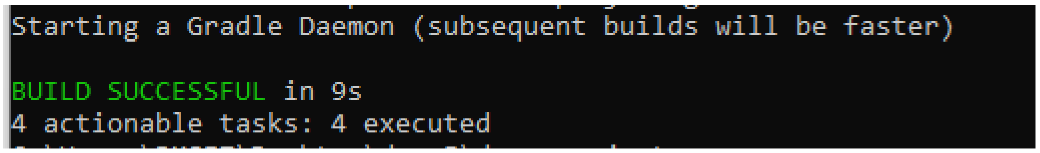
The Project Structure looks as below

```
demo-project/
├── build.gradle
├── settings.gradle
└── src/
    ├── main/java/com/example/App.java
    └── test/java/com/example/AppTest.java
```

Step 4: Build with Gradle

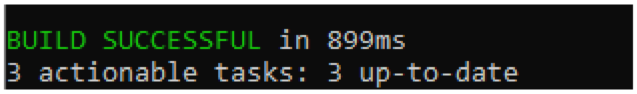
1. In the demo-project/ folder, run:

> gradle build



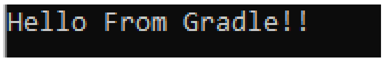
```
Starting a Gradle Daemon (subsequent builds will be faster)

BUILD SUCCESSFUL in 9s
4 actionable tasks: 4 executed
```

> gradle test



```
BUILD SUCCESSFUL in 899ms
3 actionable tasks: 3 up-to-date
```

2. Run the Gradle
   > java -cp build/libs/demo-project-1.0-SNAPSHOT.jar com.example.App



```
Hello From Gradle!!
```

Note : The above steps is to run a Java Application using Gradle.
