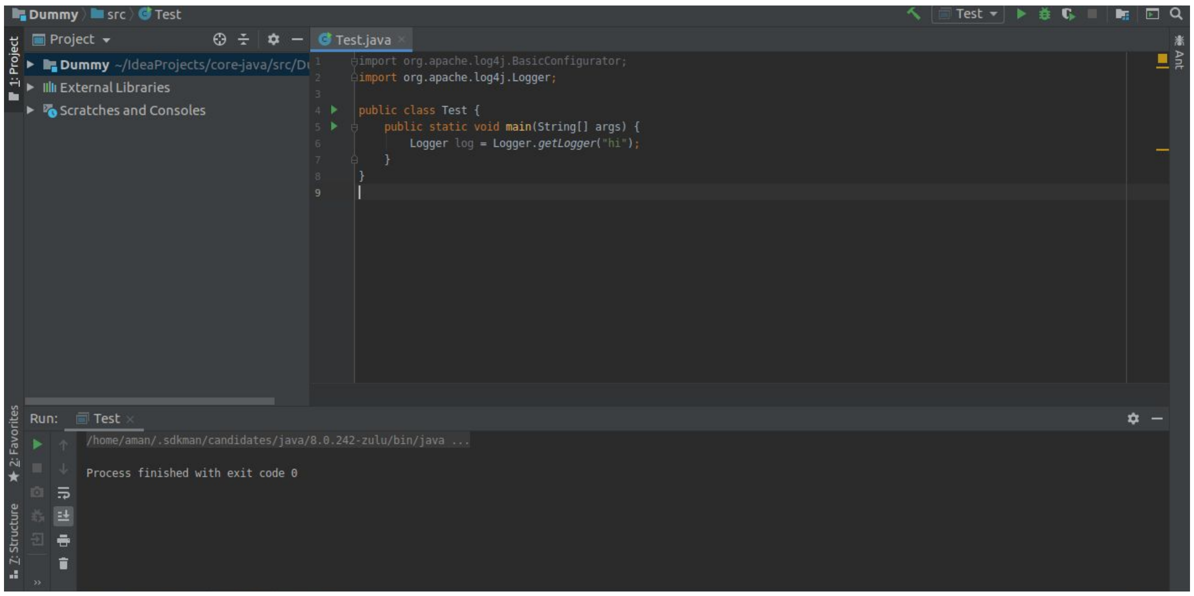
1. Create a non maven project and add use log4j for logging.

SOL.

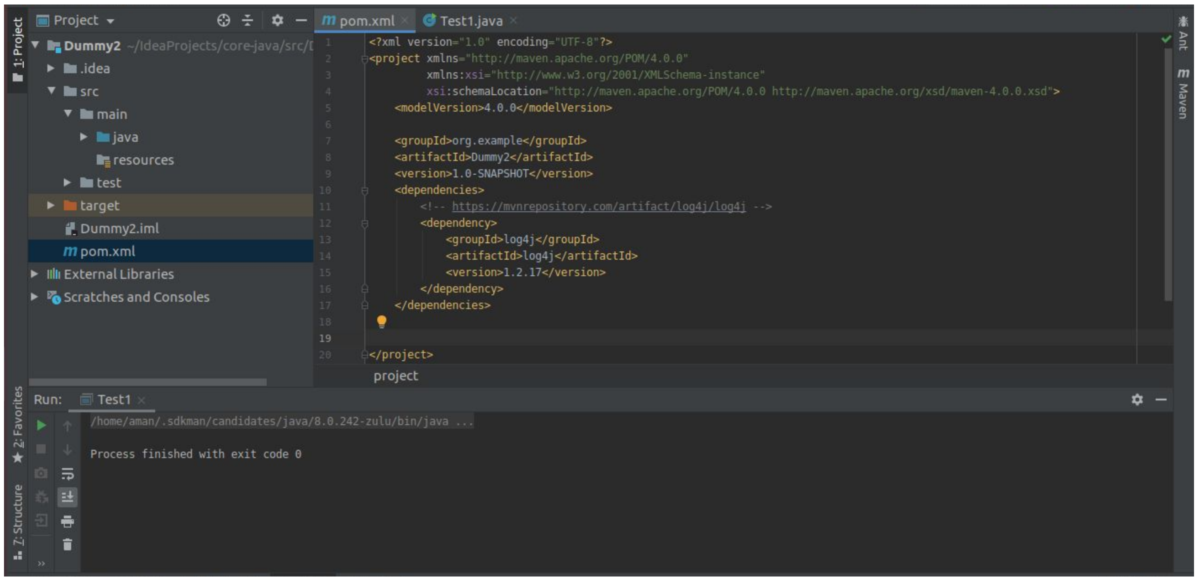


```java
import org.apache.log4j.BasicConfigurator;
import org.apache.log4j.Logger;

public class Test {
    public static void main(String[] args) {
        Logger log = Logger.getLogger("hi");
    }
}
```

```
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...

Process finished with exit code 0
```

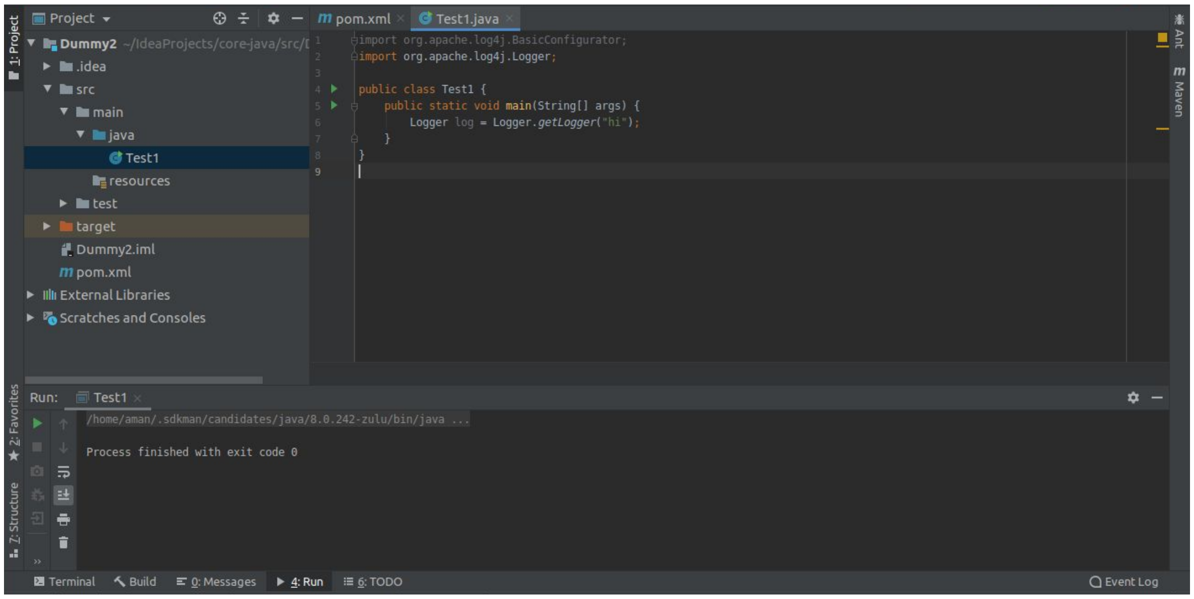
2.Create a maven project and use log4j for logging.

SOL.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<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/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>Dummy2</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
        <!-- https://mvnrepository.com/artifact/log4j/log4j -->
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.17</version>
        </dependency>
    </dependencies>


</project>
```

```
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...

Process finished with exit code 0
```

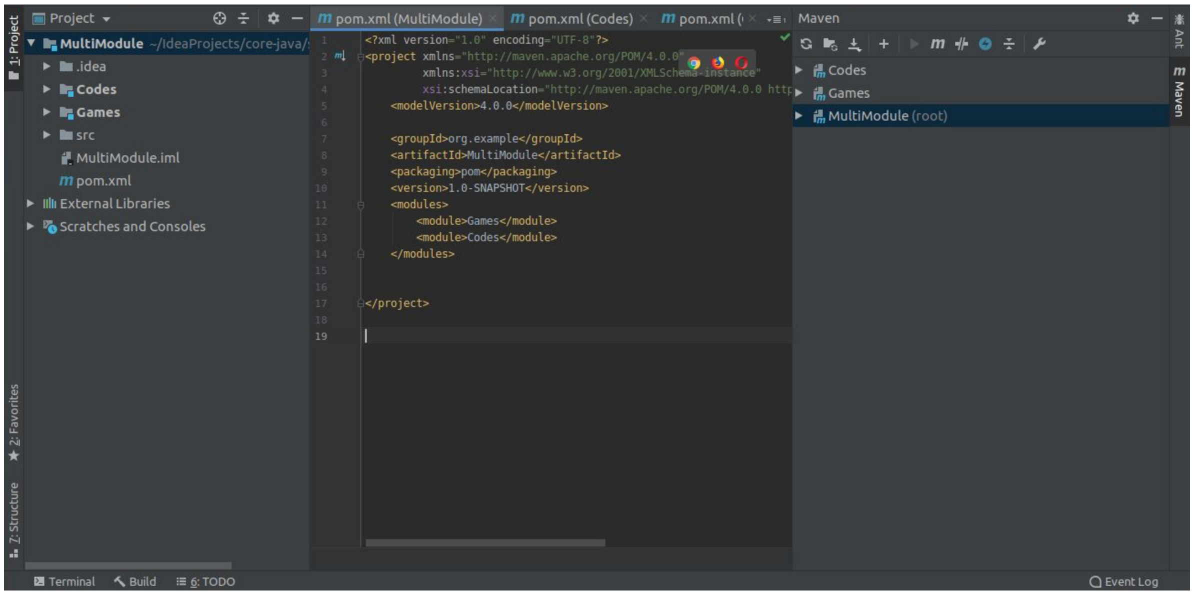


```java
import org.apache.log4j.BasicConfigurator;
import org.apache.log4j.Logger;

public class Test1 {
    public static void main(String[] args) {
        Logger log = Logger.getLogger("hi");
    }
}
```

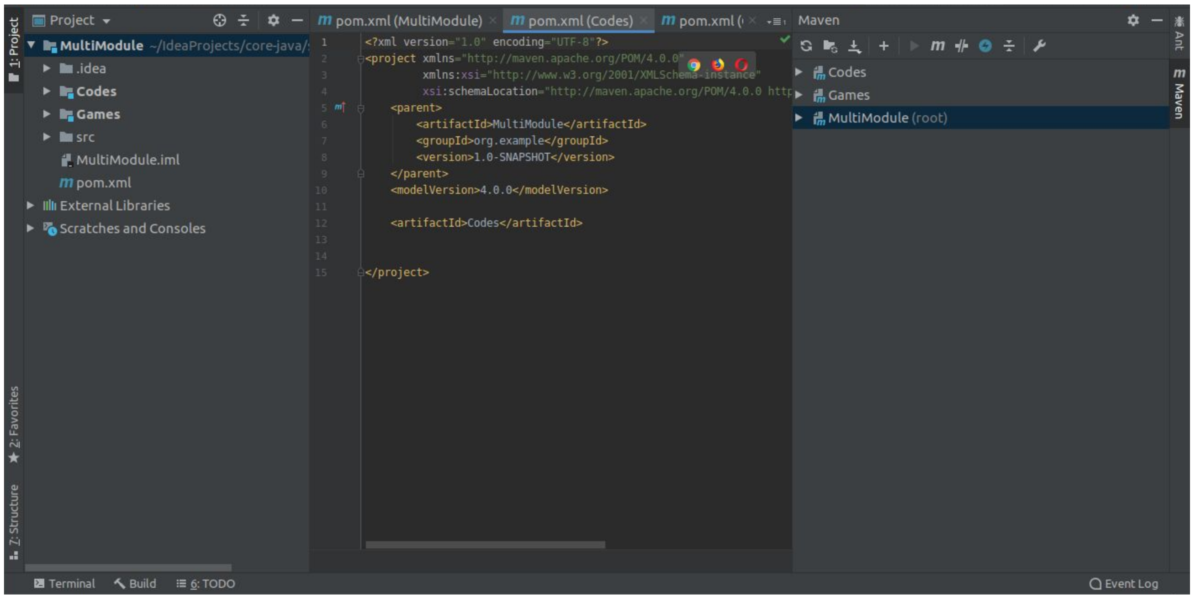
```
/home/aman/.sdkman/candidates/java/8.0.242-zulu/bin/java ...

Process finished with exit code 0
```

3.Create a maven project using an archetype(multi module).

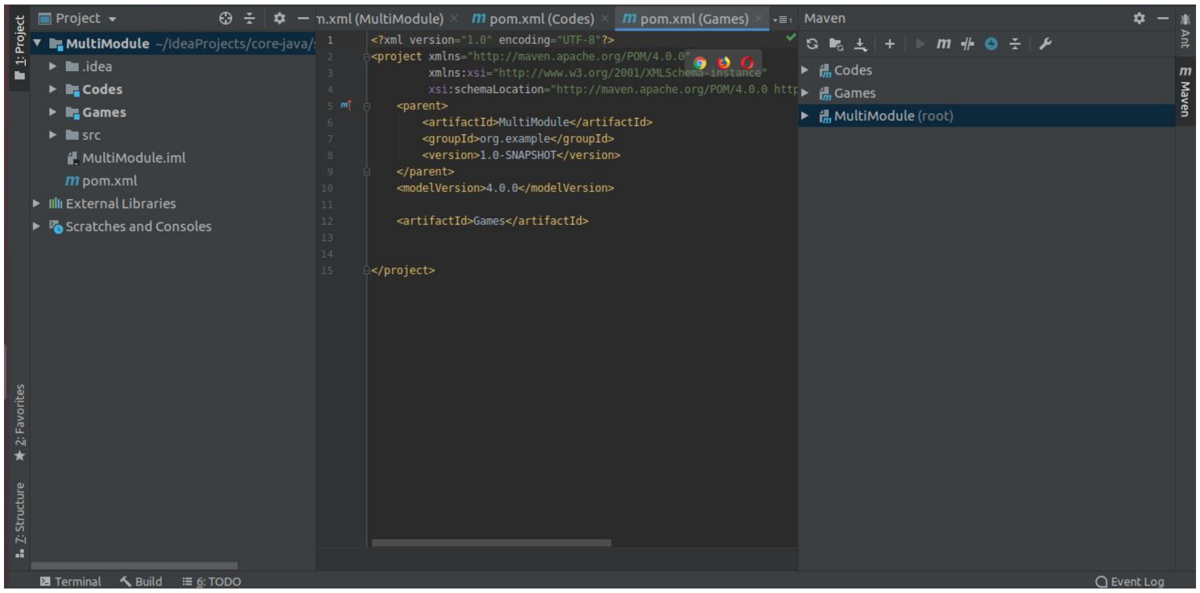
SOL.



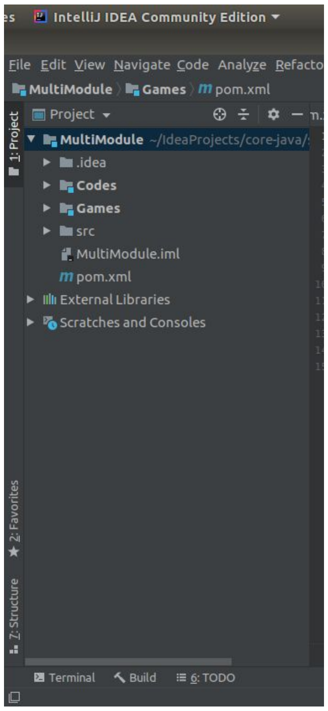
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>MultiModule</artifactId>
    <packaging>pom</packaging>
    <version>1.0-SNAPSHOT</version>
    <modules>
        <module>Games</module>
        <module>Codes</module>
    </modules>


</project>

```



```xml
<?xml version="1.0" encoding="UTF-8"?>
<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
    <parent>
        <artifactId>MultiModule</artifactId>
        <groupId>org.example</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>Codes</artifactId>


</project>
```



```xml
<?xml version="1.0" encoding="UTF-8"?>
<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
    <parent>
        <artifactId>MultiModule</artifactId>
        <groupId>org.example</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>Games</artifactId>


</project>
```



IntelliJ IDEA Community Edition
File Edit View Navigate Code Analyze Refacto
MultiModule 〉 Games 〉 pom.xml
Project
MultiModule ~/IdeaProjects/core-java/
.idea
Codes
Games
src
MultiModule.iml
pom.xml
External Libraries
Scratches and Consoles
Terminal
Build
6: TODO

4.Explain different tags (plugins, dependency, parent, profile, properties, modules and project related(i.e. modelVersion, groupId, artifactId, packaging, version, description)) of POM file created using archetype.

SOL.

plugins: Maven is actually a plugin execution framework where every task is actually done by plugins. Maven Plugins are generally used to −

- create jar file
- create war file
- compile code files
- unit testing of code
- create project documentation
- create project reports

A plugin generally provides a set of goals, which can be executed using the following syntax −

→ mvn [plugin-name]:[goal-name]

dependencies: In Maven, dependency is another archive—JAR, ZIP, and so on—which your current project needs in order to compile, build, test, and/or to run. The dependencies are gathered in the pom.xml file, inside of a <dependencies> tag.

parent: A parent POM can be declared with packaging pom. It is not meant to be distributed because it is only referenced from other projects.

Maven parent pom can contain almost everything and those can be inherited into child pom files like:-

- Common data – Developers' names, SCM address, distribution management etc.
- Constants – Such as version numbers
- Common dependencies – Common to all child. It has same effect as writing them several times in individual pom files.
- Properties – For example plugins, declarations, executions and IDs.
- Configurations
- Resources

profile: A profile in Maven is an alternative set of configuration values which set or override default values. Using a profile, you can customize a build for different environments. Profiles are configured in the *pom.xml* and are given an identifier. Then you can run Maven with a command-line flag that tells Maven to execute goals in a specific profile.

properties: When a Maven Project Property is referenced, the property name is referencing a property of the Maven Project Object Model (POM). We can use Maven properties in a *pom.xml* file or in any resource that is being processed by the Maven Resource plugin's filtering features. A property is always surrounded by *${* and *}.*

module: A Maven module is a sub-project. With Maven, you can control the versions of these modules and the dependencies between these modules. Each module will produce an artifact.

modelVersion: It is the sub element of the project. It specifies the modelVersion. It should be set to 4.0.0.

groupId: It is the sub element of the project. It specifies the id for the project group.

artifactId: It is the sub element of the project. It specifies the id for the artifact (project). An artifact is something that is either produced or used by a project. Examples of artifacts produced by Maven for a project include: JARs, source and binary distributions, and WARs

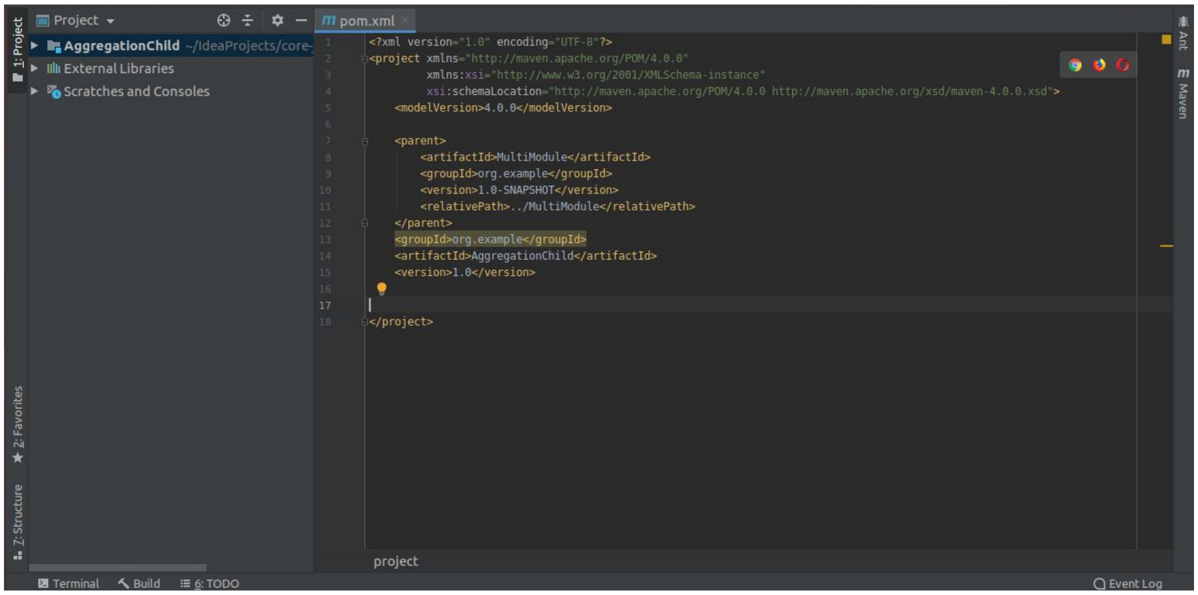
version: It is the sub element of the project. It specifies the version of the artifact under given group.

packaging: defines packaging types such as jar, war etc.

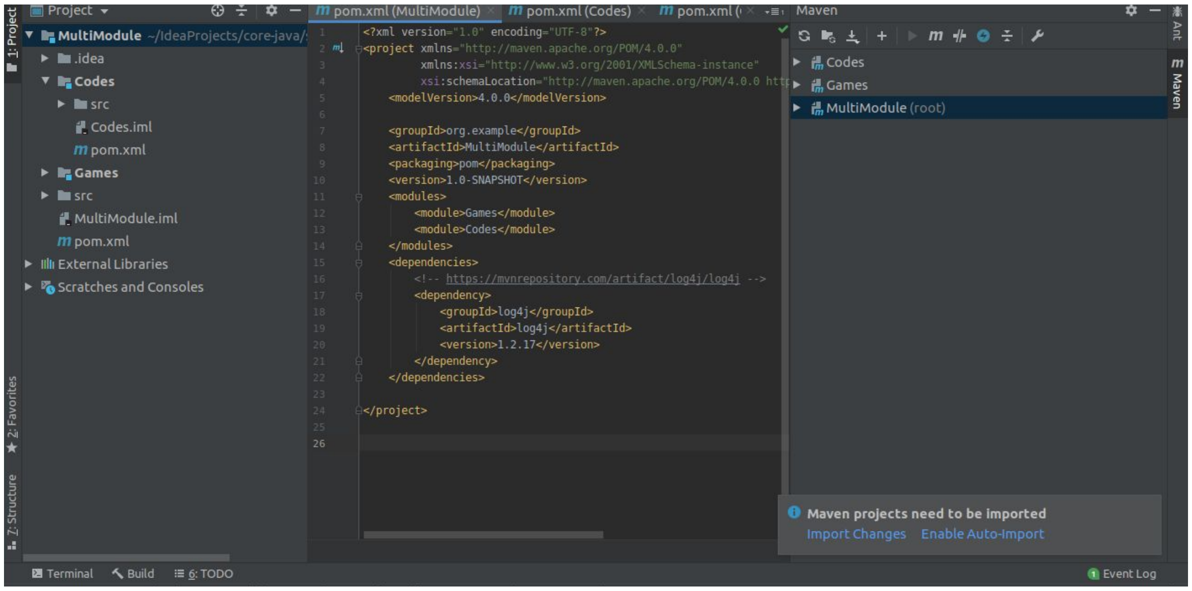
5.Demonstrate the inheritance and aggregation of POM.

SOL.

<u>AGGREGATION</u>



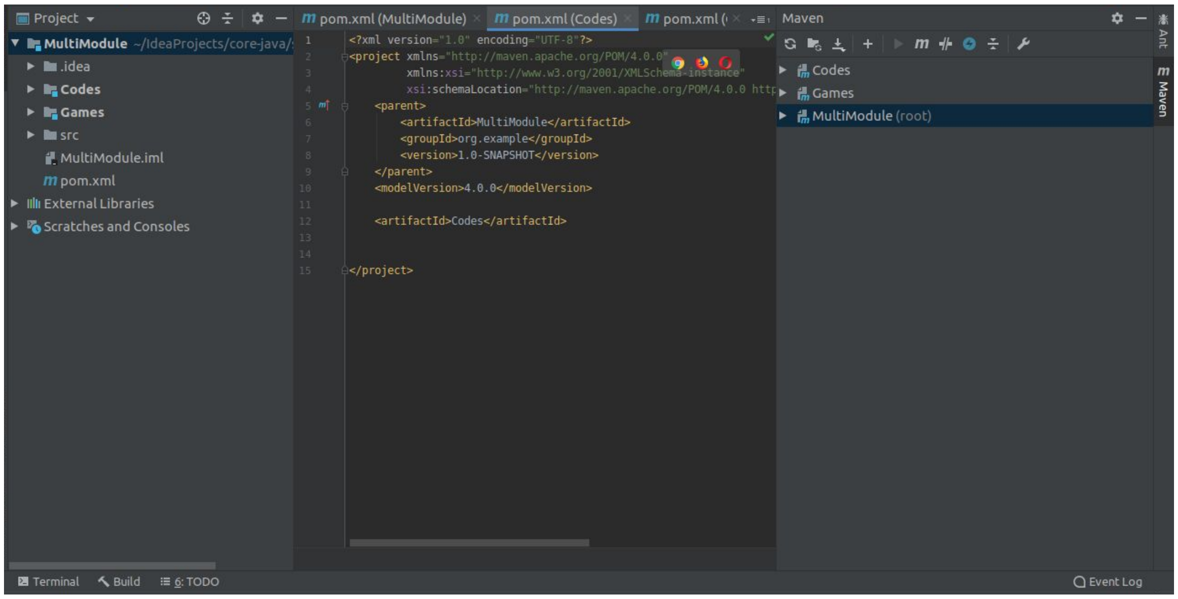
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <artifactId>MultiModule</artifactId>
        <groupId>org.example</groupId>
        <version>1.0-SNAPSHOT</version>
        <relativePath>../MultiModule</relativePath>
    </parent>
    <groupId>org.example</groupId>
    <artifactId>AggregationChild</artifactId>
    <version>1.0</version>


</project>
```



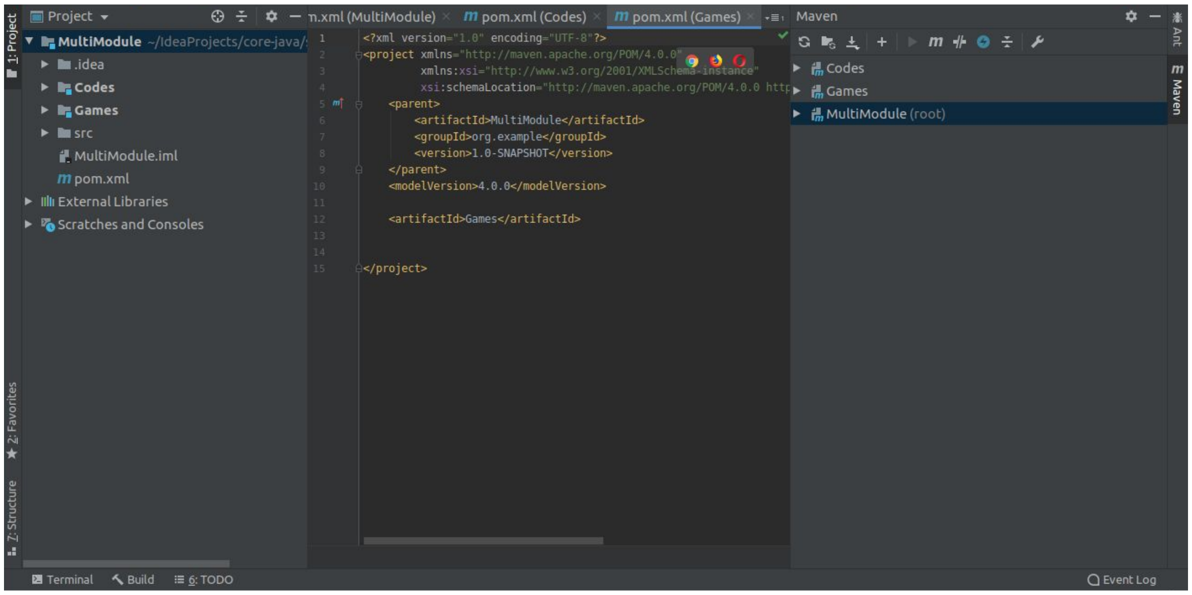
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>MultiModule</artifactId>
    <packaging>pom</packaging>
    <version>1.0-SNAPSHOT</version>
    <modules>
        <module>Games</module>
        <module>Codes</module>
    </modules>
    <dependencies>
        <!-- https://mvnrepository.com/artifact/log4j/log4j -->
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.17</version>
        </dependency>
    </dependencies>

</project>
```

Maven projects need to be imported
Import Changes Enable Auto-Import

INHERITANCE



```xml
<?xml version="1.0" encoding="UTF-8"?>
<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
    <parent>
        <artifactId>MultiModule</artifactId>
        <groupId>org.example</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>Codes</artifactId>


</project>
```



```xml
<?xml version="1.0" encoding="UTF-8"?>
<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
    <parent>
        <artifactId>MultiModule</artifactId>
        <groupId>org.example</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>Games</artifactId>


</project>
```

6.Point out the differences between Gradle and Maven.

SOL.

<u>**GRADLE**</u>	**VS**	<u>**MAVEN**</u>
An open-source build automation System build on Apache Ant and Apache Maven		A software project management and comprehension tool primarily used with Java based project
Does not use XML.		Uses XML.
Scripts are short and clean.		Scripts are not as short and clean.
Written in Java, Gradle and Kotlin.		Written in Java.
Allows structuring the build, supporting multi-build projects, increases productivity, provide an easy way to migrate and different techniques to manage builds.		Makes the build process easier, provides guidelines for best practices in development and allow transparent migration to new features.

7.What is the purpose of mvn clean install and its usage in the project

SOL.

Maven, during its use will gather all sorts of intermediate resources in it’s target folder.

mvn clean will remove the target folder and all the intermediate results.

When you build with Maven, the goal is to create some artifacts for deployment, this can be a jar for a library, a war for a web application or something else altogether. If we do a mvn install, it will:

1. Generate whatever it needs,
2. Compile the sources,
3. Copy other resources,
4. Create the artifact for your project,
5. Run unit tests,
6. Copy the artifact to the local mvn repository (this is usually $HOME/.m2/repository).

So a **mvn clean install** will first clean the target and then run the steps above.