

## CREACION DE PROYECTO EN MAVEN CON ECLIPSE

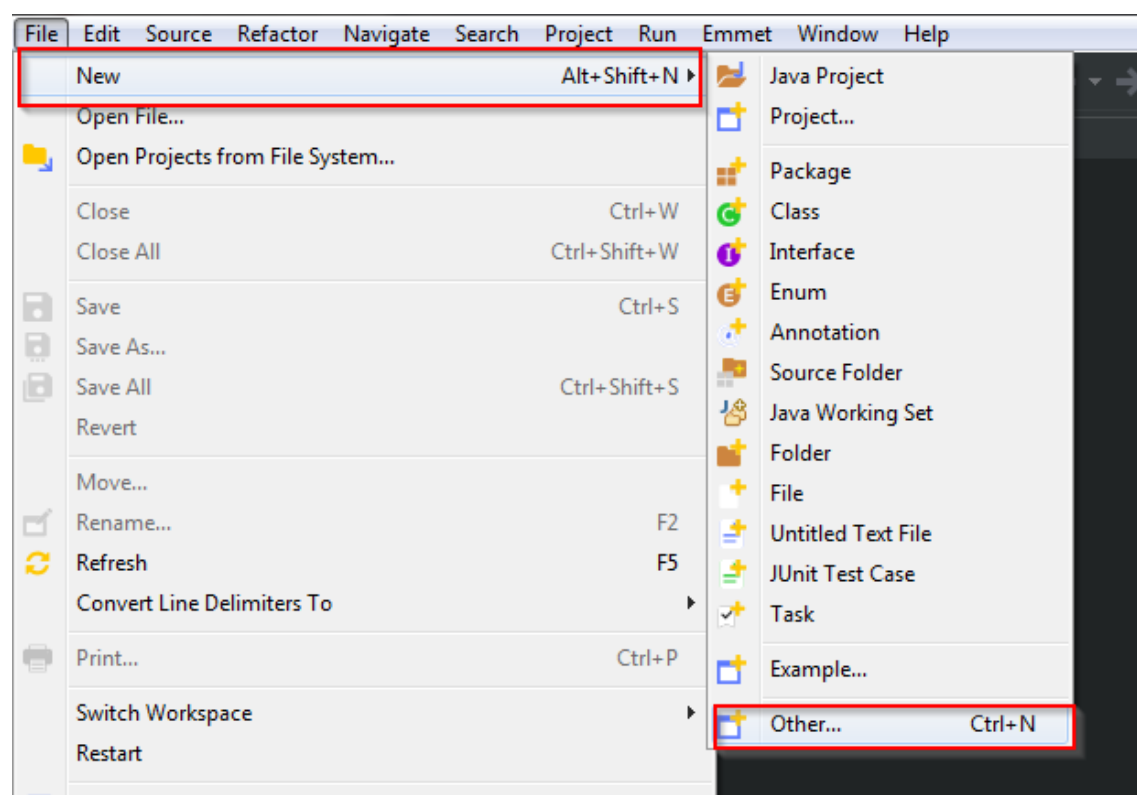
### RESOLUCIÓN DE EJERCICIOS

A continuación encontrará el desarrollo de los ejercicios que resolvió en la semana. Contraste las respuestas entregadas por el docente con las desarrolladas por usted. En caso que no coincidan, y persistan dudas, le invitamos a repasar los contenidos y/ o consultar con su profesor.

#### 1.1 Crear un proyecto de Maven en Eclipse

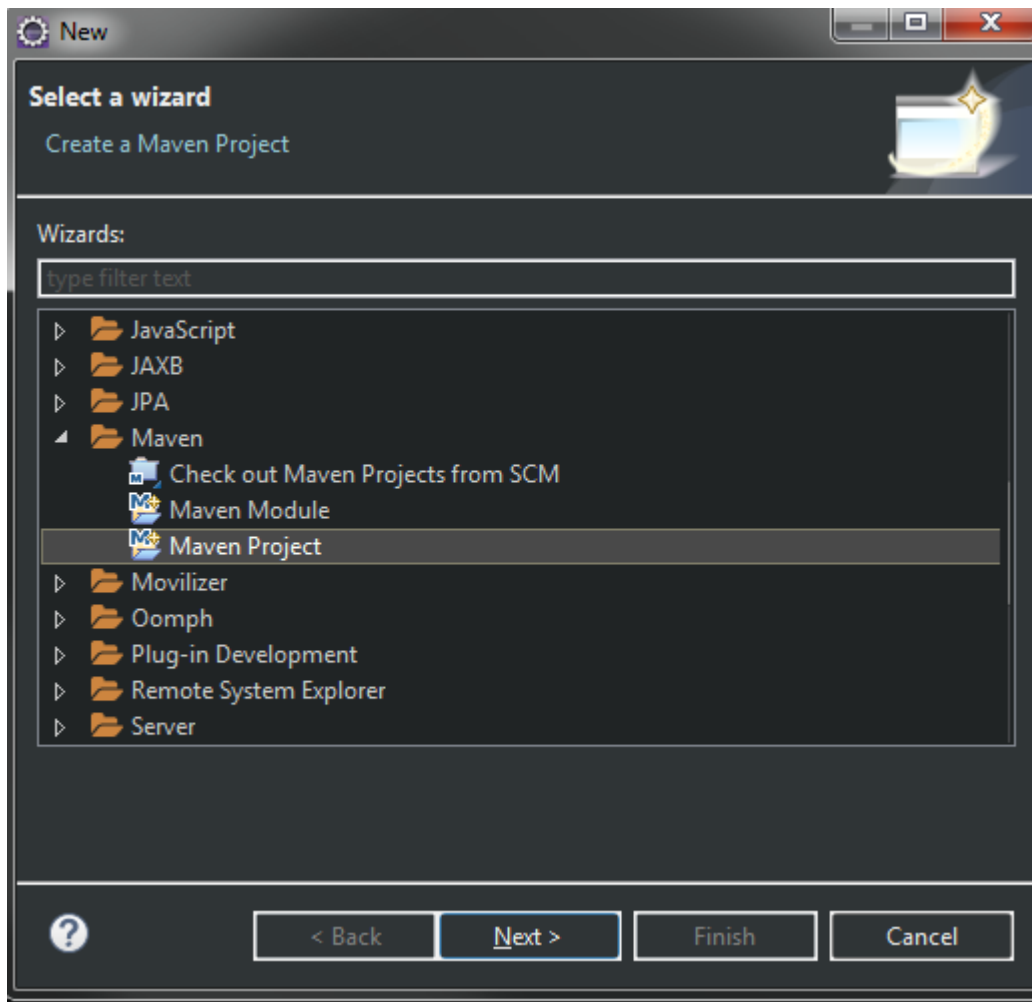
Paso 1:

Debemos crear un nuevo proyecto Maven en File → New → Other.



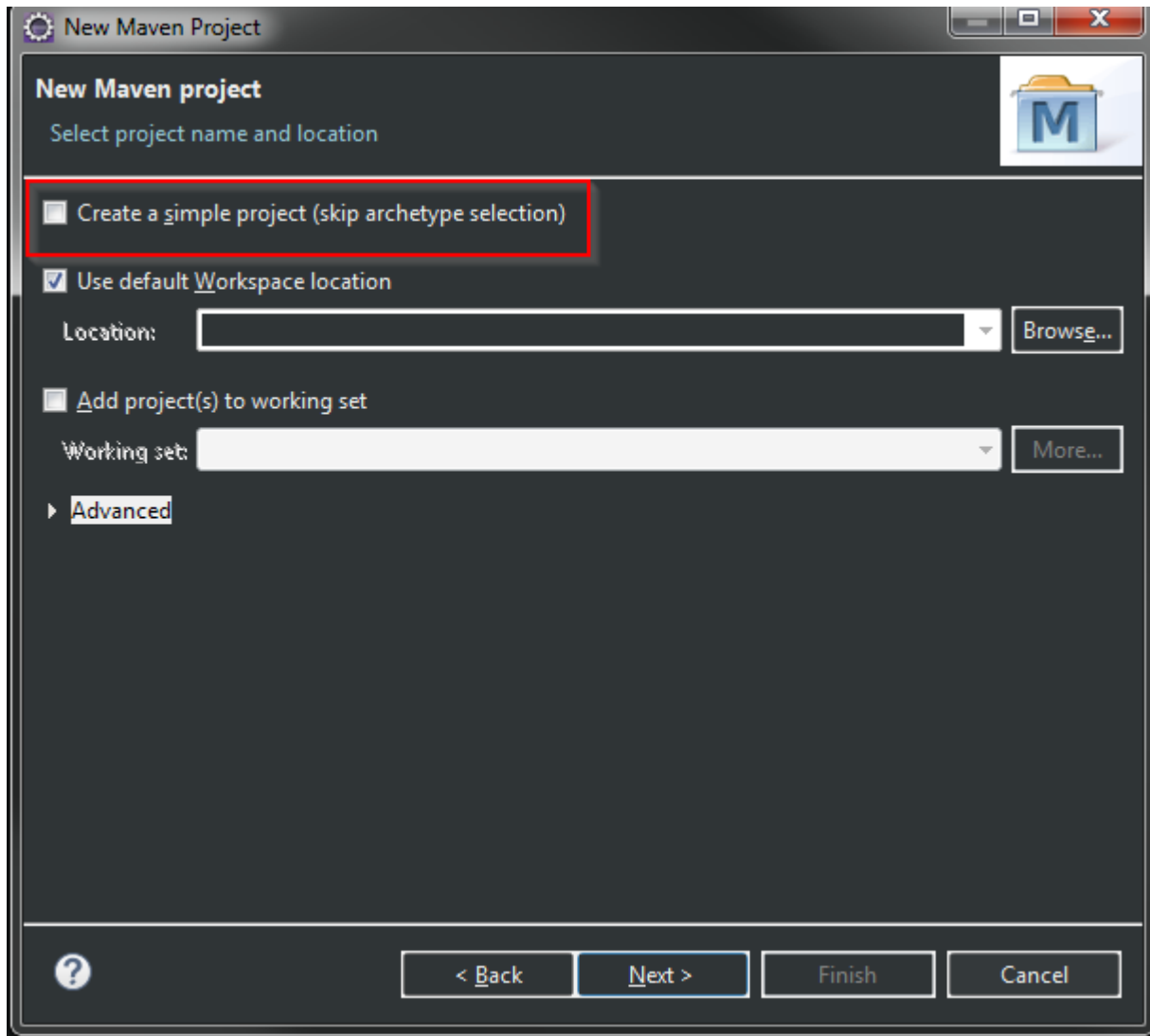
Paso 2:

Seleccionamos Maven → Maven Project y presionamos Next.



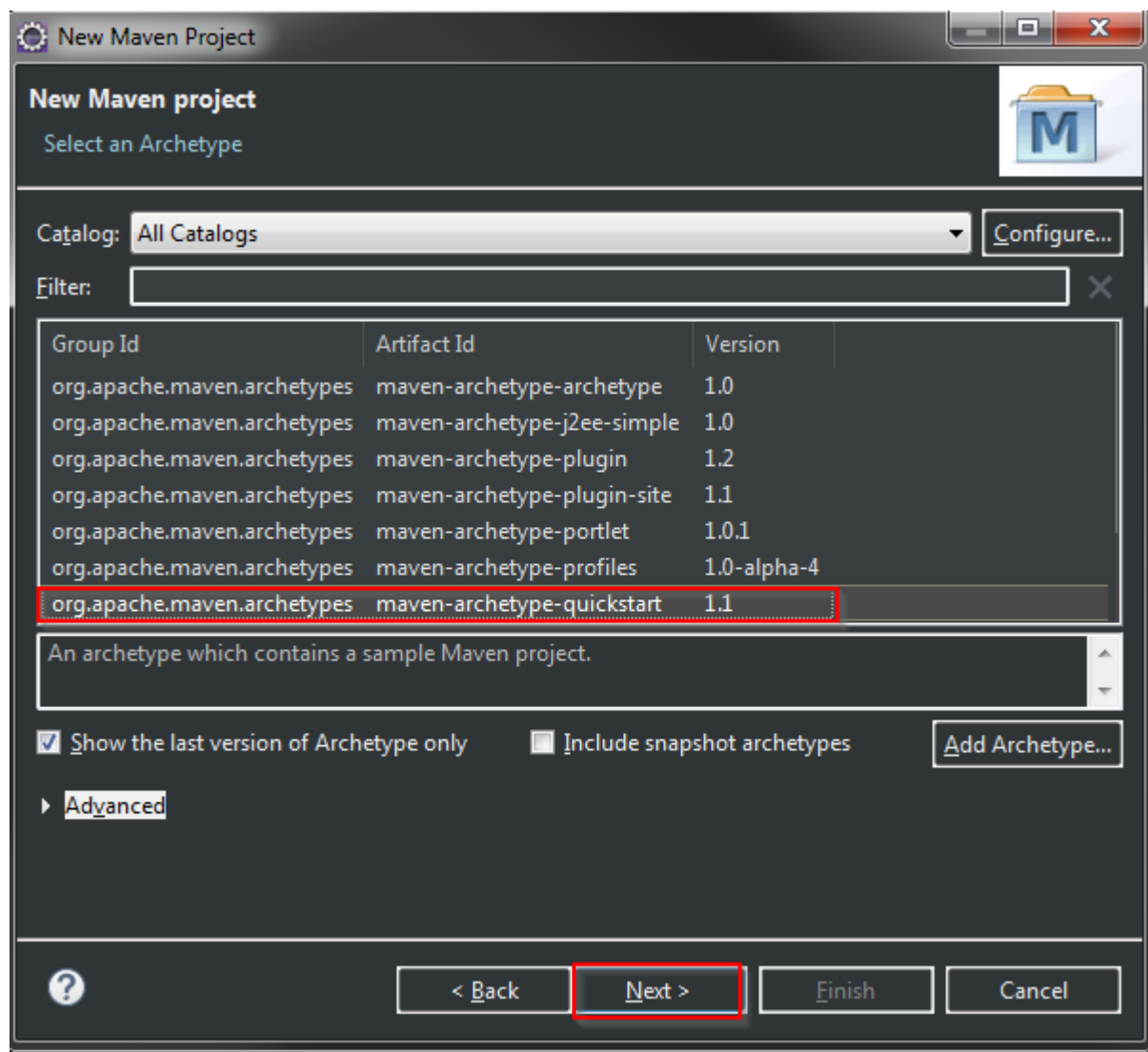
## Paso 3:

Tendremos que elegir la configuración de nuestro proyecto en Maven. En la siguiente ventana desmarcamos la opción Create a simple Project (skip archetype selection) ya que en los pasos posteriores vamos a elegir la creación de un arquetipo de las opciones que nos muestra el asistente.



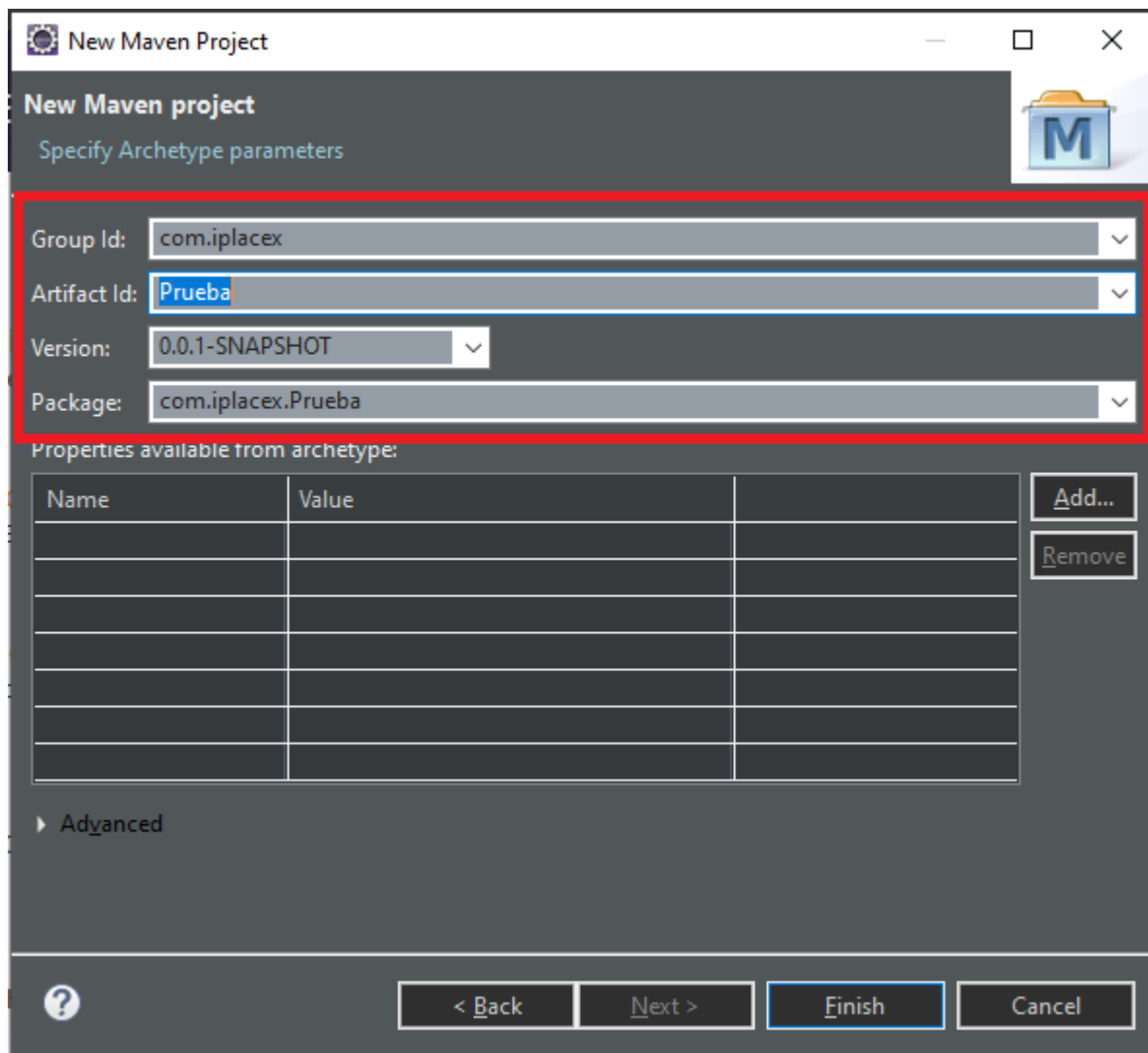
## Paso 4:

En la nueva ventana podemos seleccionar un arquetipo del listado o cargar uno desde una ubicación. En nuestro caso seleccionaremos el de groupId org.apache.maven.archetypes y Artifact id maven-archetype-quickstart y presionamos Next.



Paso 5:

Como último paso del asistente, tendremos que indicar nuestro propio groupId y el artifactId. Esto se realiza porque en el anterior paso los usaron los creadores del artefacto, sobrescribiéndolos lo establecemos como queramos.



**New Maven Project**

Specify Archetype parameters

Group Id:

Artifact Id:

Version:

Package:

Properties available from archetype:

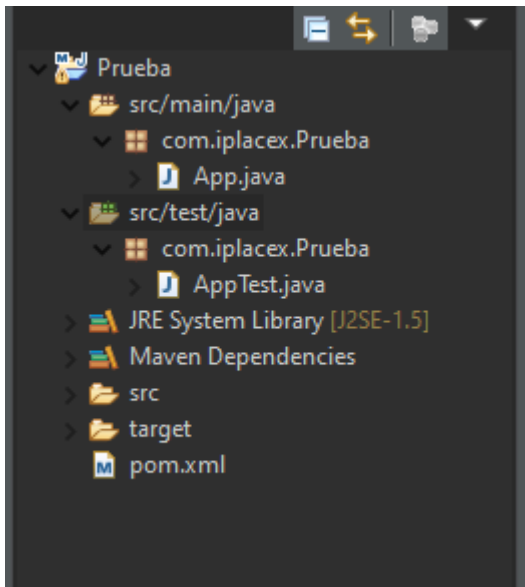
Name	Value

Advanced

< Back Next > Finish Cancel

Paso 6:

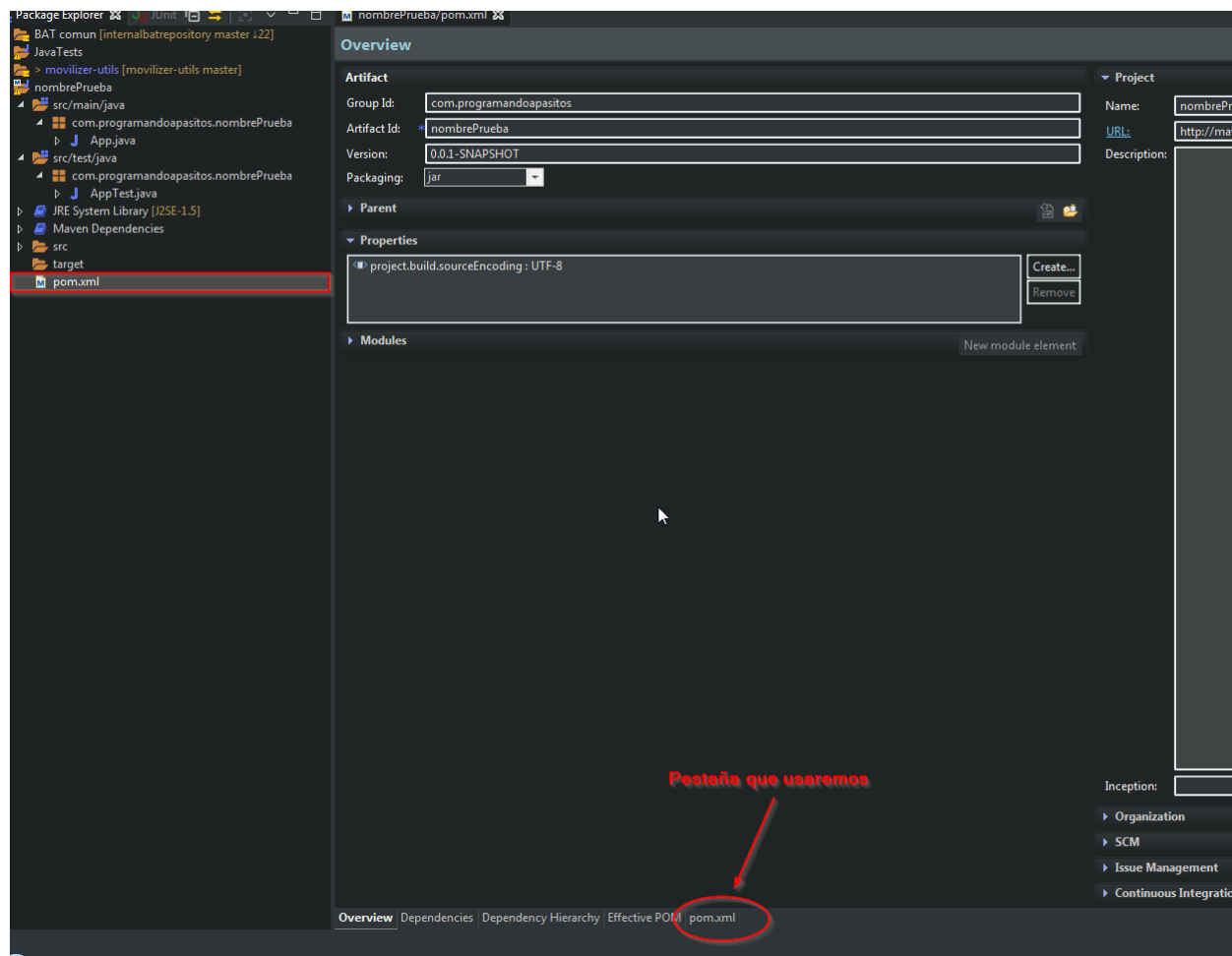
Presionamos Finish y ya tendremos nuestro proyecto. La estructura quedará de la siguiente forma:



## Configurar el archivo pom.xml

Ahora que ya tenemos creado el proyecto Maven es hora de configurar el POM para hacerlo funcionar.

Paso 1: Para ello hacemos doble click en el archivo pom.xml y en la pestaña inferior elegimos la sección pom.xml para poder ver el código XML.



Cuando entremos veremos un archivo XML similar a esto

```
<?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>com.iplacex</groupId>
  <artifactId>Prueba</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>Prueba</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Del código podemos intuir bastante bien con lo mencionado anteriormente. Del resto, el tag de properties indicará las propiedades de nuestro proyecto. Por ejemplo, la codificación de los ficheros en UTF-8 como vemos en esa línea.

Luego tenemos dependencies, que es la sección donde gestionaremos nuestras dependencias.

Antes tenemos que conseguir que Maven se encargue por nosotros de gestionar el JRE. Esto se hace incluyendo un nuevo plugin en este archivo. Así que nos ponemos debajo de <dependencies> y escribimos lo siguiente:



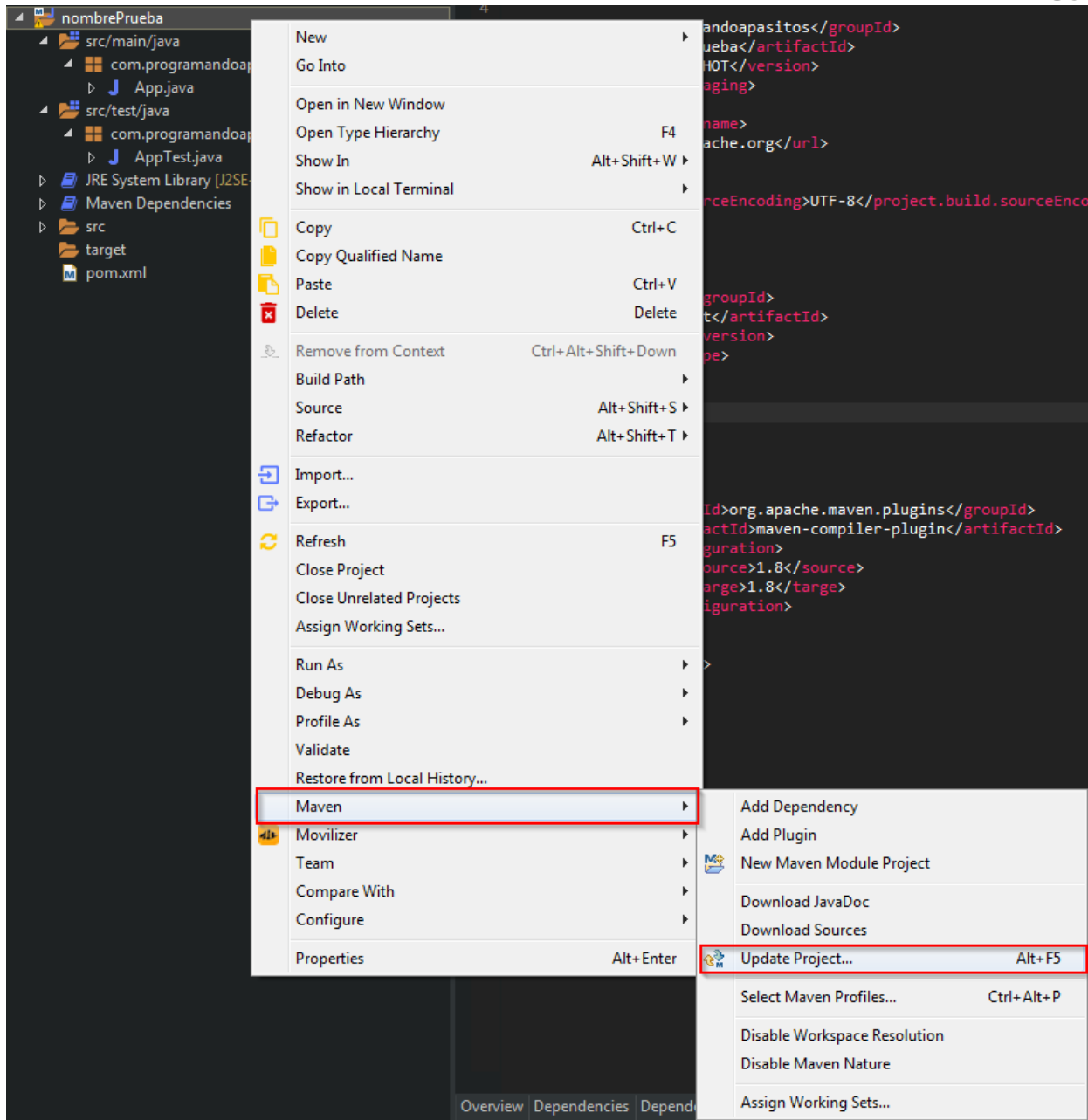
```
<build>
  <pluginManagement>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </pluginManagement>
</build>
```

Dentro de build hemos creado la etiqueta pluginManagement para gestionar los plugins de Maven. Y dentro de eso hemos creado una llamada plugins que será la que los contendrá.

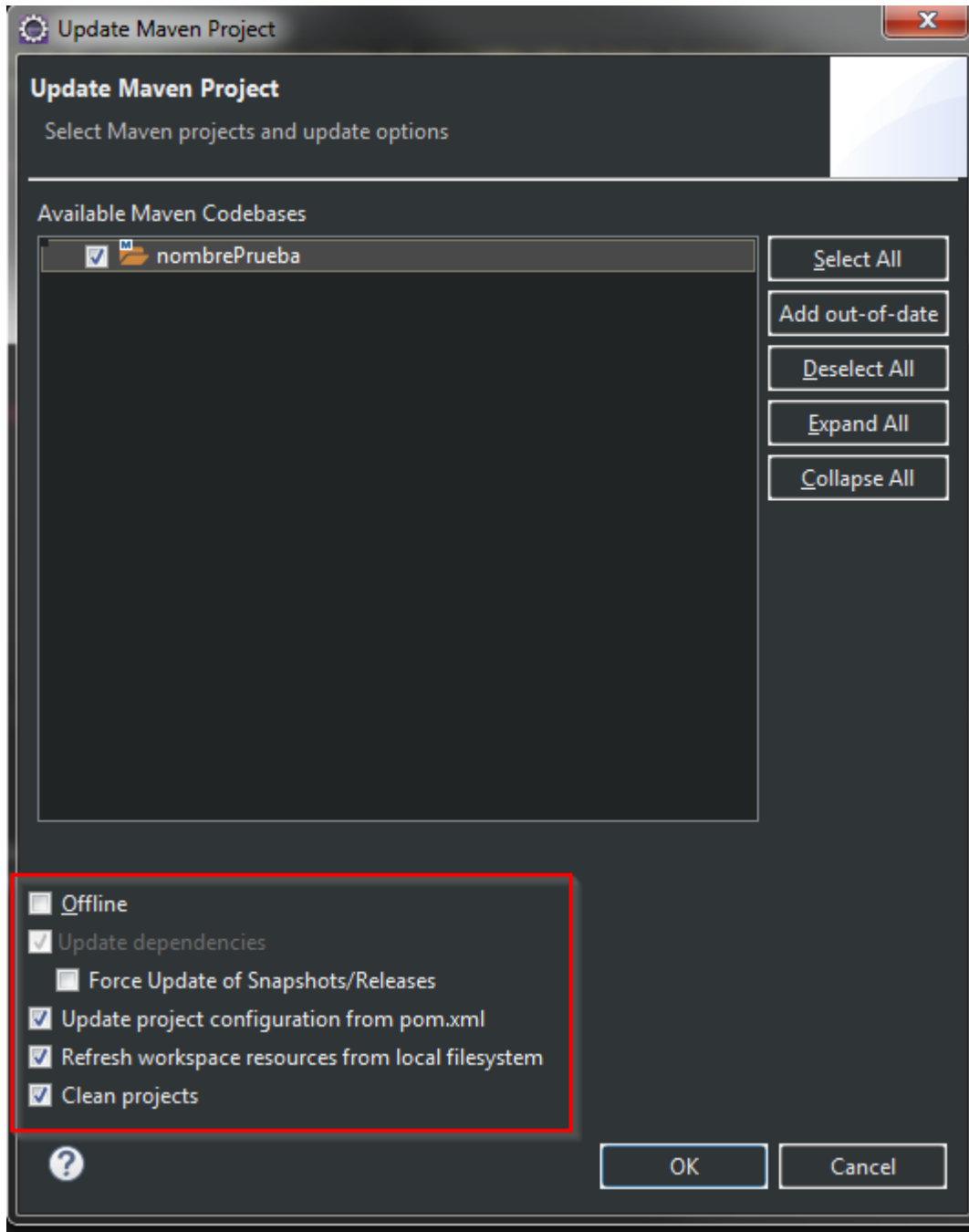
Después añadimos un nuevo plugin, en este caso maven-compiler-plugin. Y luego indicamos, en su configuración, que el source y el target apunten a la versión que tengamos del JRE en nuestro dispositivo. En este ejemplo es el 1.8.

Ahora guardamos el fichero y no te preocupes si ves un error en el explorador de proyectos, en el siguiente paso se solucionará.

Vamos al proyecto Maven y pulsamos botón derecho → Maven → Update Project...



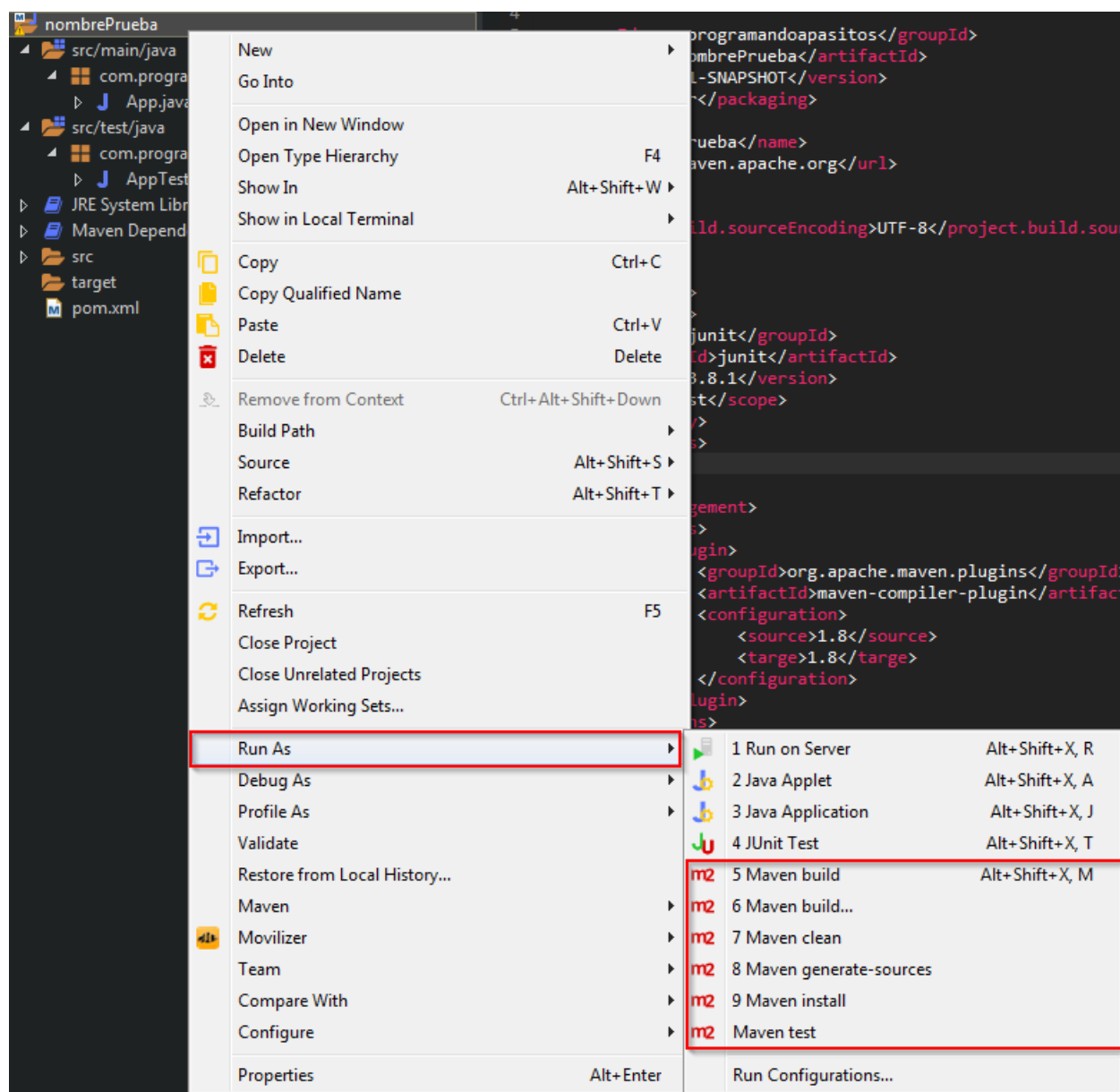
En la ventana que se abrirá marcamos con un check nuestro proyecto dejando la siguiente configuración



Cuando acabe el proceso de actualización veremos cómo desaparece el error de nuestro proyecto.

Realizado esto, vamos a lanzar por primera vez nuestro proyecto Maven y aprovecharemos para ver los ciclos de vida posibles que tiene.

Hacemos clic con el botón derecho sobre nuestro proyecto y vamos a Run As. Ahí nos aparecerán diversas opciones



¿Qué hace cada una de ellas?

Para nuestra prueba, vamos a elegir la opción de Maven install.

Se nos mostrará una ventana de consola con el proceso actual de dicha acción.

```

terminated- C:\Program Files\Java\jdk-8.0.111\bin\java.exe (28 x64 2017-11-28-18)
[INFO] Compiling 1 source file to C:\Vovcoders\EclipseWorkspace\nombrePrueba\target\test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ nombrePrueba ---
[INFO] Surefire report directory: C:\Vovcoders\EclipseWorkspace\nombrePrueba\target\surefire-reports

T E S T S

Running com.programandacassinos.nombrePrueba.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.003 sec

Results:

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ nombrePrueba ---
[INFO] Building jar: C:\Vovcoders\EclipseWorkspace\nombrePrueba\target\nombrePrueba-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ nombrePrueba ---
[INFO] Installing C:\Vovcoders\EclipseWorkspace\nombrePrueba\target\nombrePrueba-0.0.1-SNAPSHOT.jar to C:\Users\Inazio\.m2\repository\com\programandacassinos\nombrePrueba\0.0.1-SNAPSHOT.jar
[INFO] Installing C:\Vovcoders\EclipseWorkspace\nombrePrueba\com.xml to C:\Users\Inazio\.m2\repository\com\programandacassinos\nombrePrueba\com.xml
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 4.534 s
[INFO] Finished at: 2017-07-28T11:28:55+02:00
[INFO] Final Memory: 13M/194M
[INFO]

```

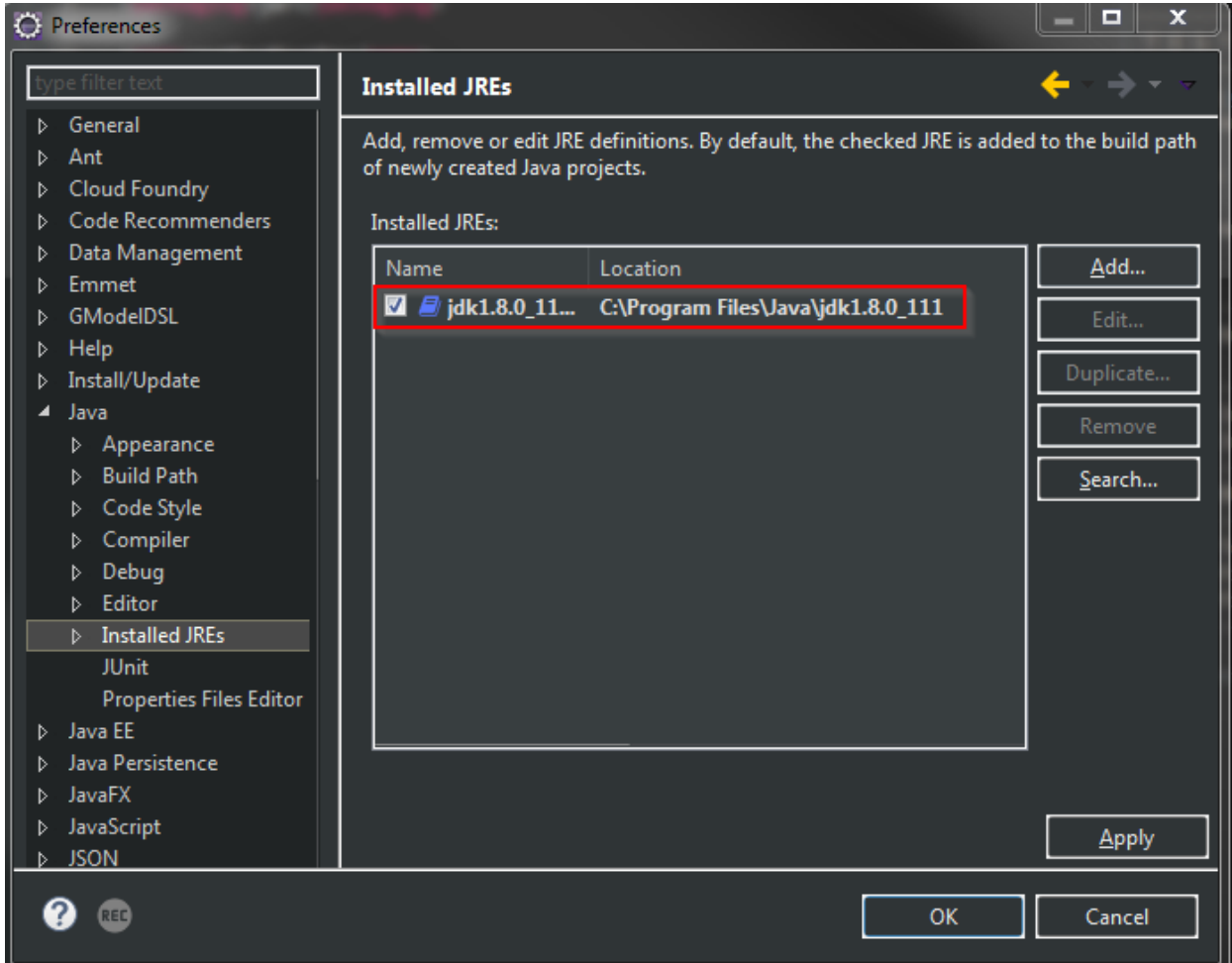
Si vemos un mensaje de BUILD SUCCESS como en la imagen de arriba significa que el proceso se ha completado con éxito.

Es posible que, en vez de un mensaje de éxito, veamos lo siguiente:

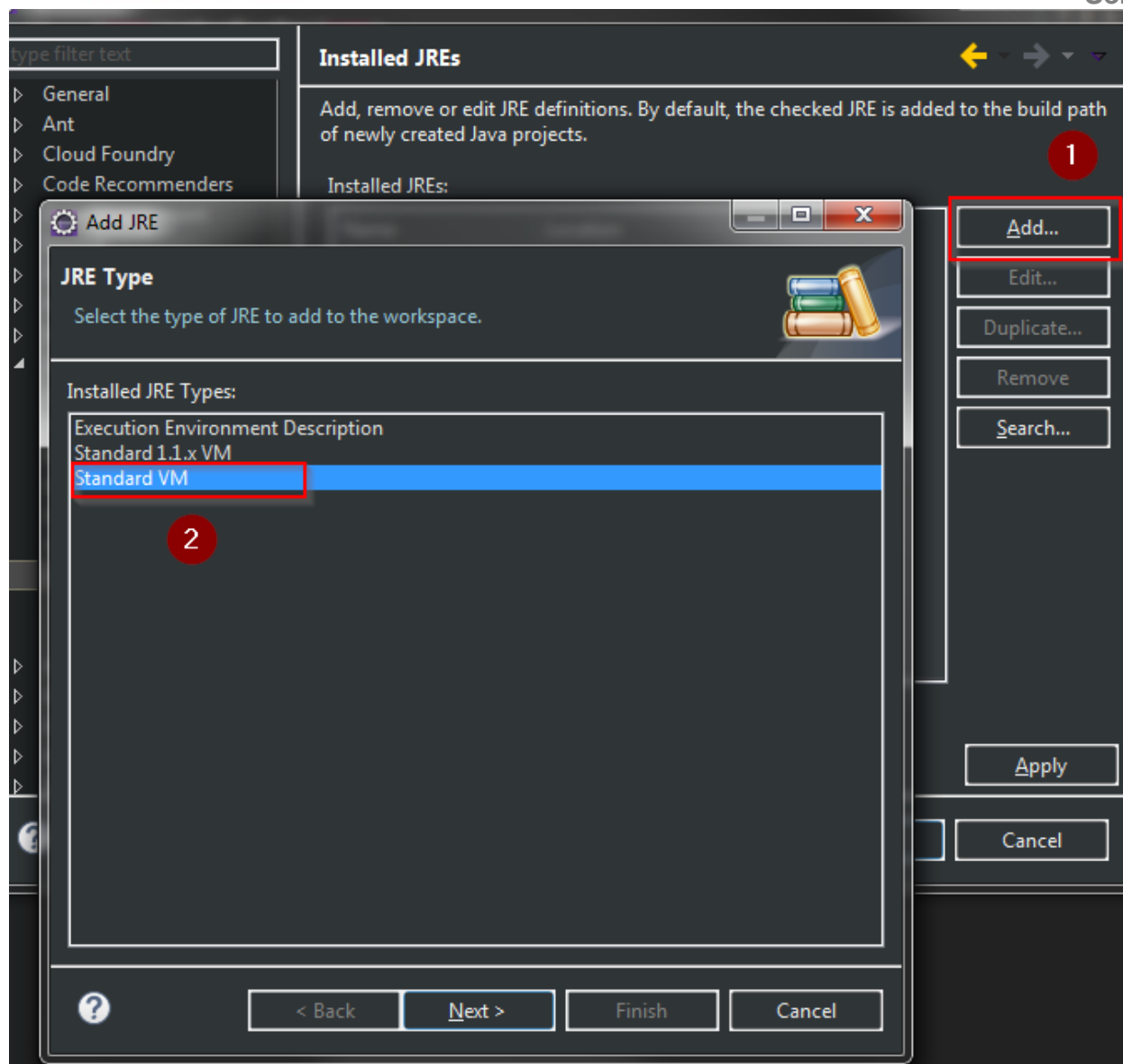
**No compiler is provided in this environment. Perhaps you are running on a JRE rather than a JDK**

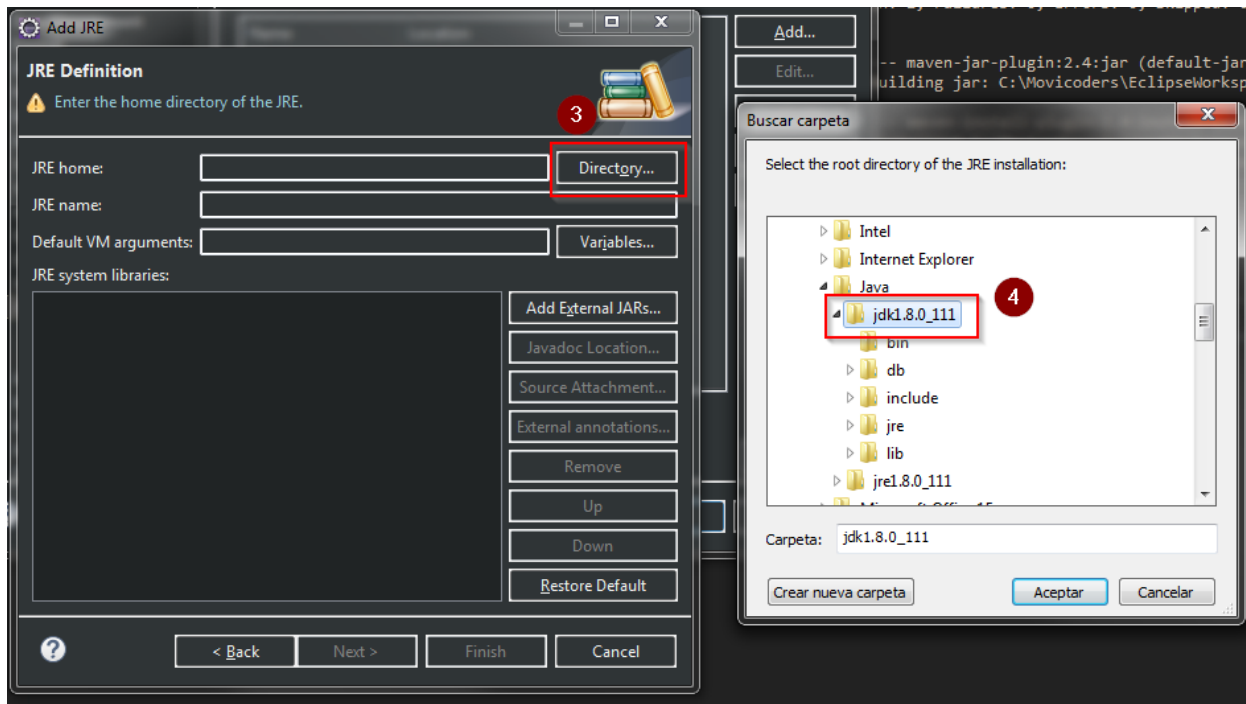
Si nos da este error quiere decir que tenemos mal configurada la ruta al JDK en el workspace de Eclipse.

Para solucionarlo hay que ir a Window → Preferences → Java → Installed JRE. Lo que debe aparecer es la versión de JDK



Si en vez de eso tenemos el JRE es el motivo de que veamos ese error. Para subsanarlo pulsamos sobre Add → Standard VM → Directory y elegimos la carpeta de instalación de nuestro JDK





Aplicamos los cambios, aceptamos y ya podemos repetir el proceso del Maven install habiendo corregido ese error.



## 1.2 Agregar dependencias del conector JDBC de MySQL.

Para gestionar las dependencias, dentro del POM debemos fijarnos en el tag dependencies. Por defecto tenemos cargada la librería de Junit, que nos servirá para ver un ejemplo de cómo añadir más librerías.

Tenemos, al igual que con los plugin, los tag de groupId y artifactId. Aparte de eso, lo que nos interesa también es el tercer tag, version, que nos permitirá indicar la versión de la librería y, en caso de tener que modificarla, bastará con cambiar ese valor y volver a hacer un install.

¿Dónde podemos conseguir las dependencias?

Existe una página web llamada MVN Repository que cuenta con un amplio abanico de dependencias organizadas por temas.

Simplemente entraremos en la dependencia que nos interese, por ejemplo, el conector JDBC de MySQL, elegiremos la versión que queramos cargar en nuestro proyecto y en la parte inferior de la página nos aparecerá el código XML de Maven que deberemos copiar en el POM.

Home » [mysql](#) » [mysql-connector-java](#) » 5.1.6

**Note:** There is a new version for this artifact

**New Version**



## MySQL Connector/J » 5.1.6

MySQL JDBC Type 4 driver

License	<a href="#">GPL 2.0</a>
Categories	<a href="#">MySQL Drivers</a>
HomePage	<a href="http://dev.mysql.com/usingmysql/java/">http://dev.mysql.com/usingmysql/java/</a>
Date	(May 08, 2015)
Files	<a href="#">Download (JAR)</a> <b>(686 KB)</b>
Repositories	<a href="#">Central</a> <a href="#">Clojars</a> <a href="#">WSO2</a>
Used By	<b>2,076 artifacts</b>

[Maven](#)

[Gradle](#)

[SBT](#)

[Ivy](#)

[Grape](#)

[Leiningen](#)

[Buildr](#)

```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.6</version>
</dependency>
```

☒ Include comment with link to declaration

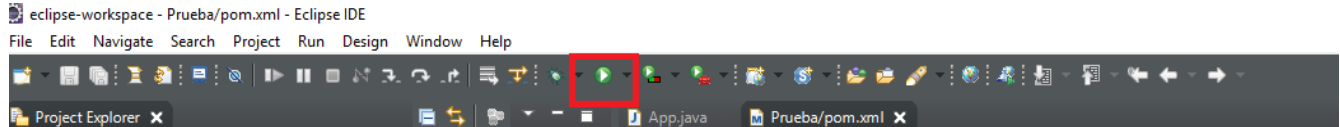
De ese modo nuestro POM quedaría así

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4
5   <groupId>com.iplacex</groupId>
6   <artifactId>Prueba</artifactId>
7   <version>0.0.1-SNAPSHOT</version>
8   <packaging>jar</packaging>
9
10  <name>Prueba</name>
11  <url>http://maven.apache.org</url>
12
13  <properties>
14    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15  </properties>
16
17  <dependencies>
18    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
19    <dependency>
20      <groupId>mysql</groupId>
21      <artifactId>mysql-connector-java</artifactId>
22      <version>8.0.16</version>
23    </dependency>
24
25    <dependency>
26      <groupId>junit</groupId>
27      <artifactId>junit</artifactId>
28      <version>3.8.1</version>
29      <scope>test</scope>
30    </dependency>
31  </dependencies>
32
33
34
35 </project>
36
```

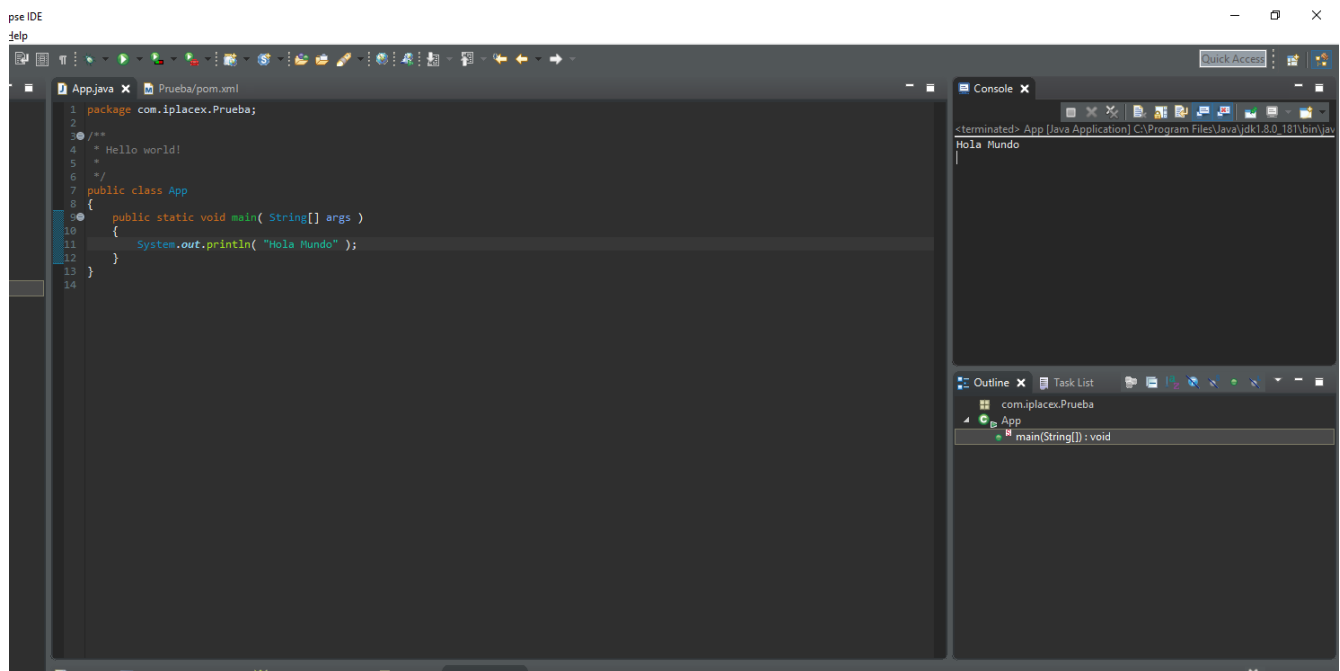
Y después de hacer un Maven install ya estaremos en condiciones de usar nuestra nueva dependencia.

### 1.3 Realizar una prueba básica

Para realiza prueba de nuestro proyecto presionamos el incono de play indicado en la siguiente imagen.



A continuación, veremos el resultado de la ejecución del proyecto en java desde Maven



## RESPUESTAS ESPERADAS

1. ¿Cuál es la importancia de Maven en el desarrollo de proyectos como herramienta de gestión?

La importancia de Maven es construir nuestro código fuente del proyecto, ejecutar los casos de prueba, gestionar las dependencias de nuestro proyecto e importar las librerías automáticamente desde un repositorio remoto olvidándonos de hacer manualmente esta tarea. Permite la creación y descarga de plantillas de proyectos para tener la estructura ya creada, por ejemplo, en proyectos web o conectores. Crear la documentación del proyecto, desplegar el proyecto (AKA artefacto) en servidor y por supuesto, realizar una perfecta integración con Git, SVN, Jira, etc.

2. ¿Cuáles son las ventajas de utilizar dependencias en un proyecto?

Maven está construido en la idea de reutilización, y más específicamente, a la reutilización de la lógica de construcción. Tiene una librería de utilidades y funciones buenas y útiles. Es configurable, enfatizando seriamente a que los usuarios deben adherirse a su concepto de un modelo de proyecto estándar tanto como sea posible.

3. ¿Cuáles son los componentes de un proyecto Maven y cuáles son las herramientas de desarrollo de software (IDE) que permiten trabajar con Maven?

Artefacto: es un proyecto que lo gestiona Maven y que incluye un fichero llamado pom.xml

POM: Son las siglas de Project Object Model. Es un fichero XML que contiene la configuración del artefacto. Más adelante trabajaremos con él. En Android su similar sería el AndroidManifest.

Group id: El identificador único para crear nuestro artefacto. Se suele poner el mismo que en un paquete java. com.<nombre\_proyecto>

Herramientas de desarrollo que permiten trabajar con Maven son:

- Eclipse
- JetBrains IntelliJ IDEA
- Netbeans IDE.

Dependencias que podemos agregar a nuestro proyecto en Maven:

- Mysql
- Oracle JDBC
- SDK y librerías de Amazon Web Services: S3, SQS Java Messaging library, AWS Java SDK for Amazon DynamoDB.