Lesson 01 Demo 01

Using JUnit in Maven Project

Objective: To use JUnit in the Maven project and test the project by configuring the project with the new compiler plugin

Tool required: Eclipse IDE

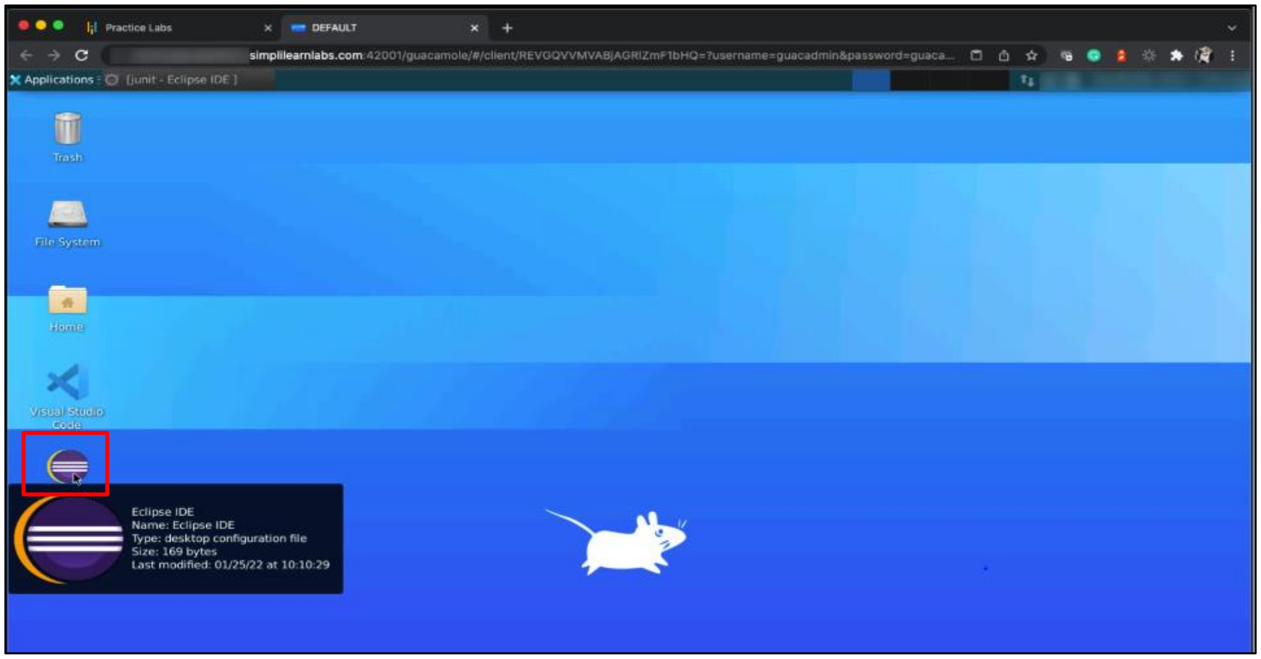
Prerequisites: None

Steps to be followed:

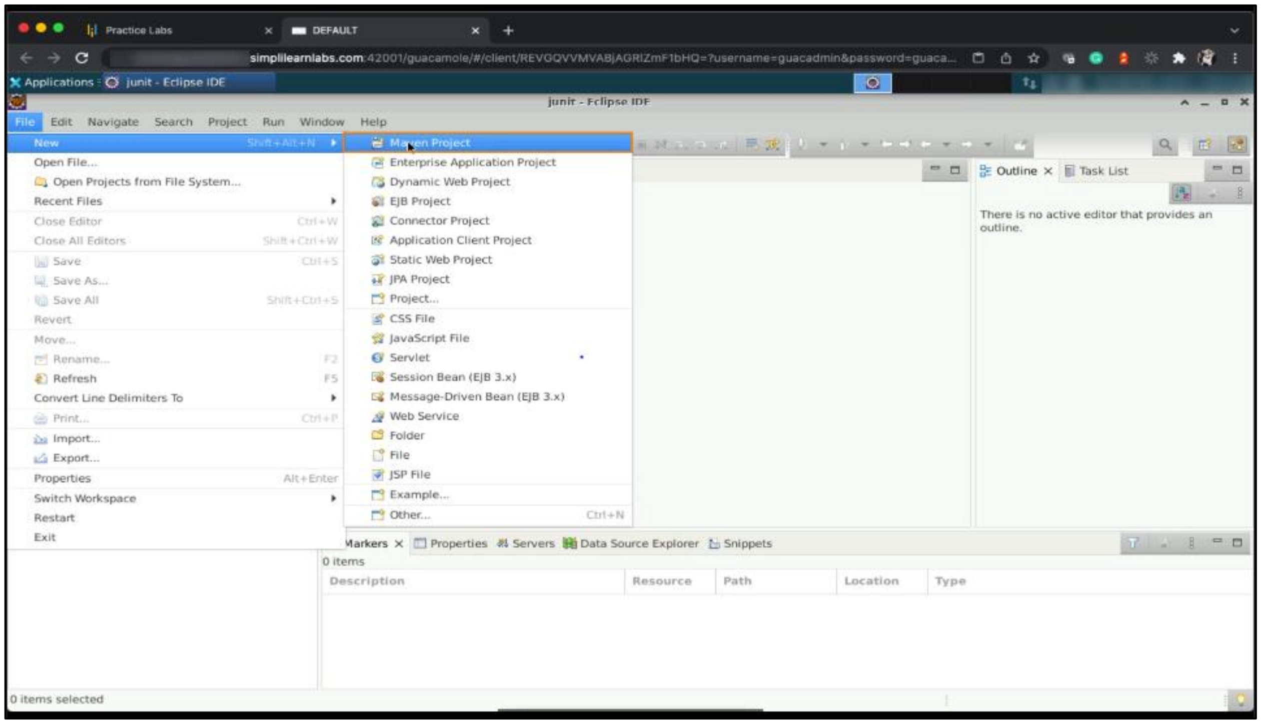
1. Creating a new Maven project
2. Configuring the project with a new compiler plugin
3. Creating a test method

Step 1: Creating a new Maven project

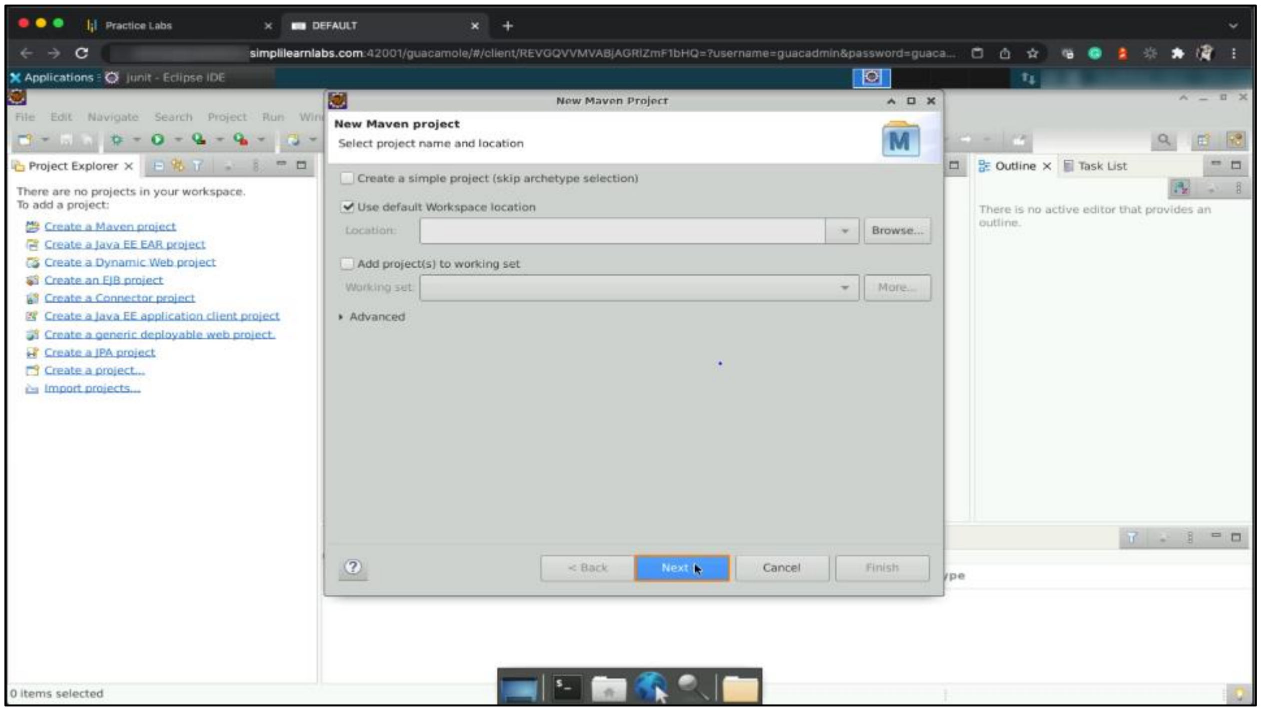
1.1 Open **Eclipse IDE**



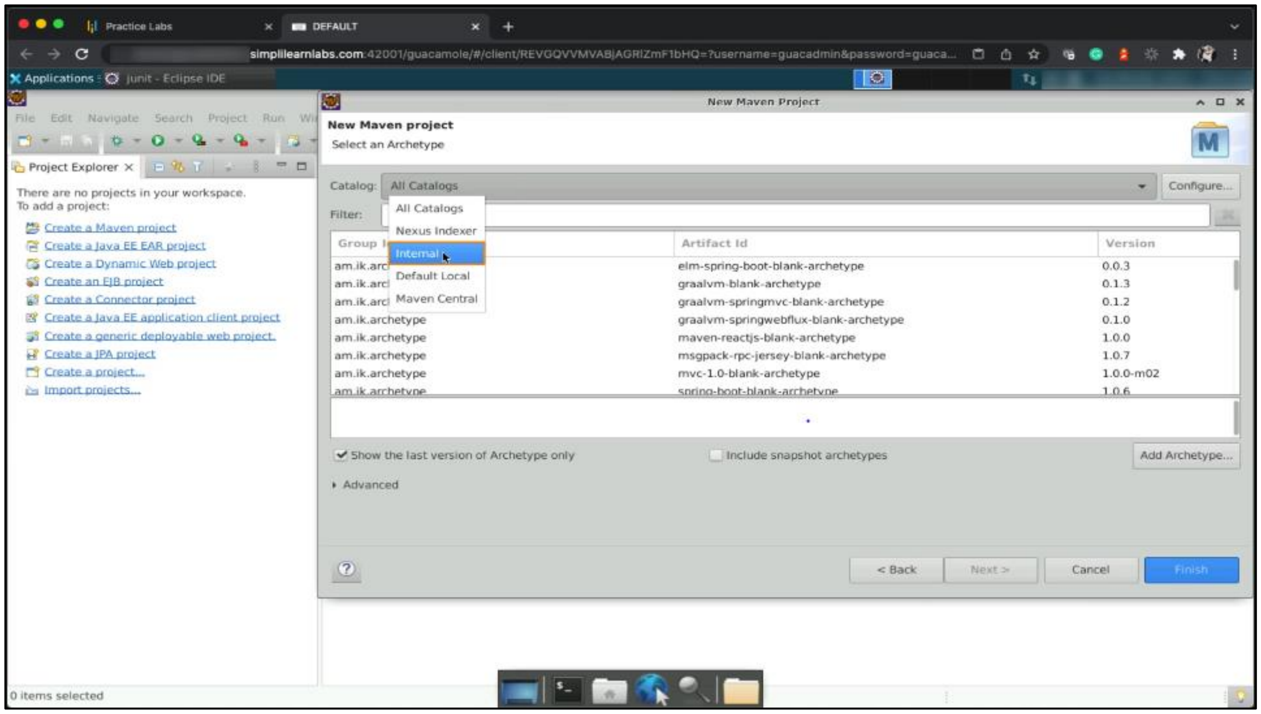
1.2 Click on **Files**, then click on **New,** and create a new **Maven project**



1.3 Click on **Next** and choose the catalog as **Internal**


New Maven Project
New Maven project
Select project name and location
Browse...
More...
Advanced
< Back
Next >
Cancel
Finish
Project Explorer
There are no projects in your workspace.
To add a project:
Create a Maven project
Create a Java EE EAR project
Create a Dynamic Web project
Create an EJB project
Create a Connector project
Create a Java EE application client project
Create a generic deployable web project.
Create a JPA project
Create a project...
Import projects...
Outline
Task List
There is no active editor that provides an outline.
0 items selected

☐ Create a simple project (skip archetype selection)
☑ Use default Workspace location
Location:
☐ Add project(s) to working set
Working set:


New Maven project

Select an Archetype

Catalog: All Catalogs

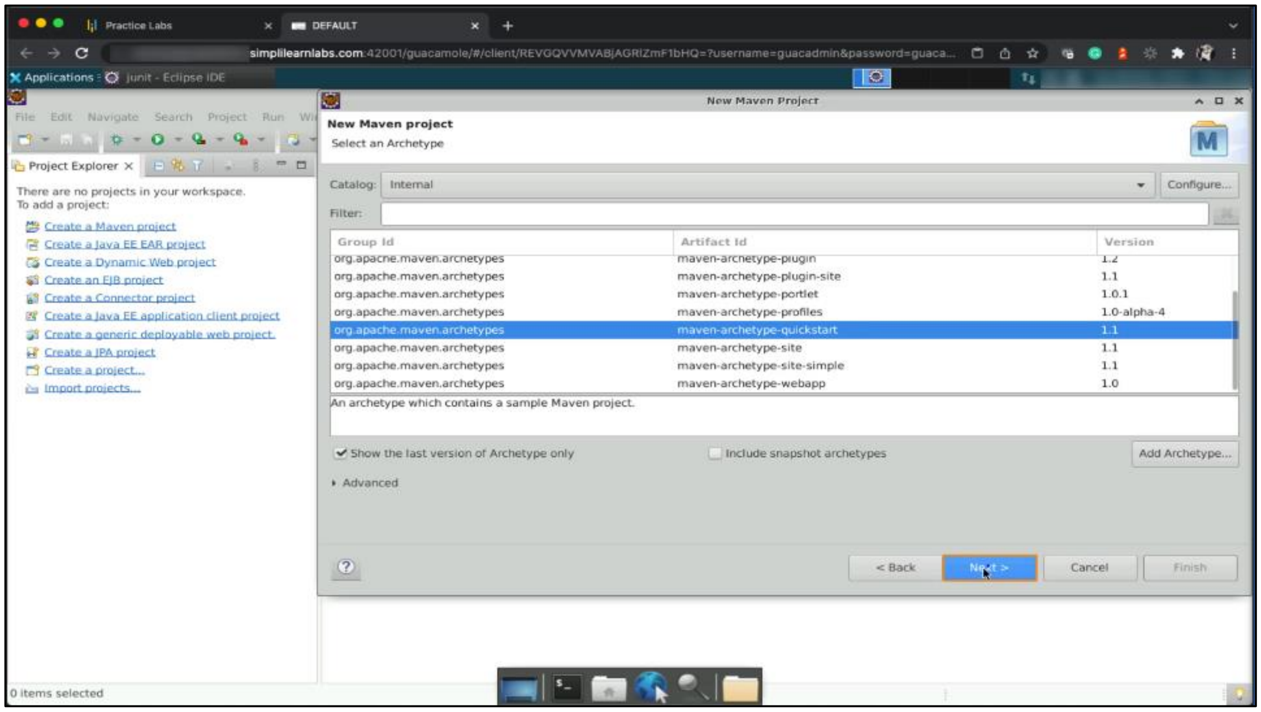
Group Id	Artifact Id	Version
am.ik.arc	elm-spring-boot-blank-archetype	0.0.3
am.ik.arc	graalvm-blank-archetype	0.1.3
am.ik.arc	graalvm-springmvc-blank-archetype	0.1.2
am.ik.archetype	graalvm-springwebflux-blank-archetype	0.1.0
am.ik.archetype	maven-reactjs-blank-archetype	1.0.0
am.ik.archetype	msgpack-rpc-jersey-blank-archetype	1.0.7
am.ik.archetype	mvc-1.0-blank-archetype	1.0.0-m02
am.ik.archetype	spring-boot-blank-archetype	1.0.6

✓ Show the last version of Archetype only

Include snapshot archetypes

Advanced

1.4 Select a **maven-archetype** as **quickstart** and click on **Next**


New Maven project

Select an Archetype

Catalog: Internal

Group Id	Artifact Id	Version
org.apache.maven.archetypes	maven-archetype-plugin	1.2
org.apache.maven.archetypes	maven-archetype-plugin-site	1.1
org.apache.maven.archetypes	maven-archetype-portlet	1.0.1
org.apache.maven.archetypes	maven-archetype-profiles	1.0-alpha-4
org.apache.maven.archetypes	maven-archetype-quickstart	1.1
org.apache.maven.archetypes	maven-archetype-site	1.1
org.apache.maven.archetypes	maven-archetype-site-simple	1.1
org.apache.maven.archetypes	maven-archetype-webapp	1.0

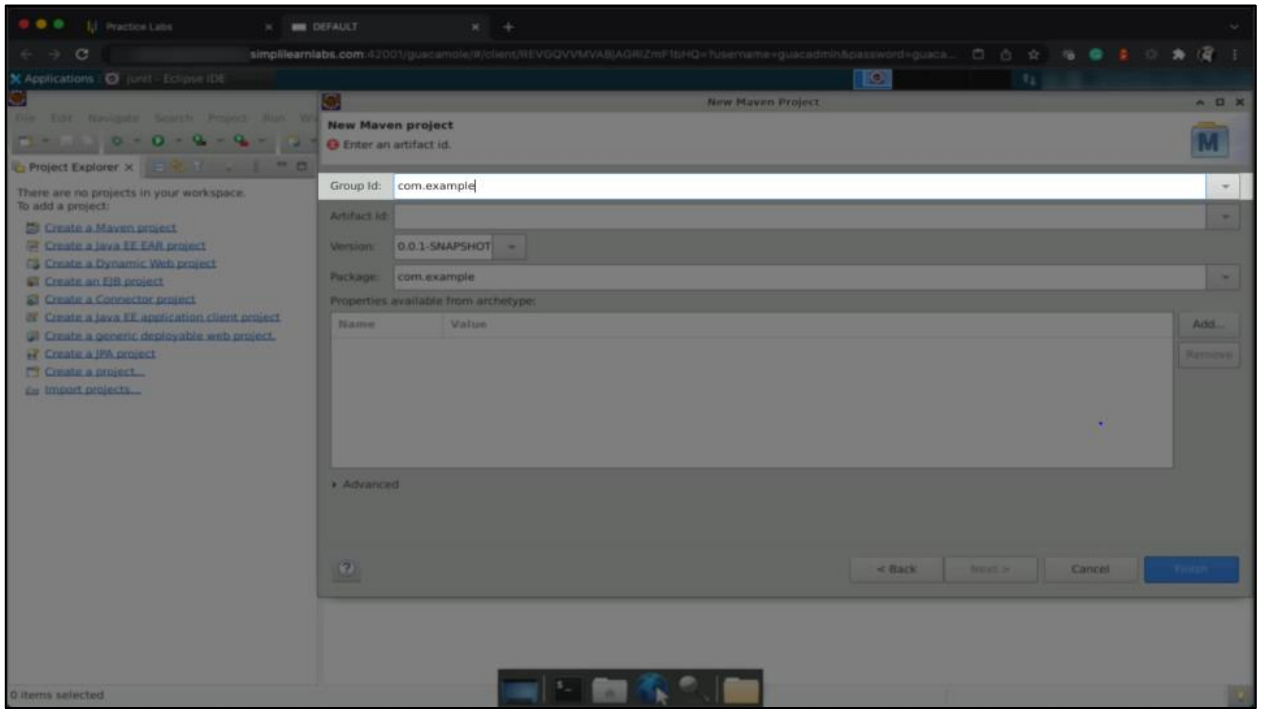
An archetype which contains a sample Maven project.

✓ Show the last version of Archetype only

Include snapshot archetypes

Advanced

1.5 Name the **Group Id** as **com.example**


New Maven project

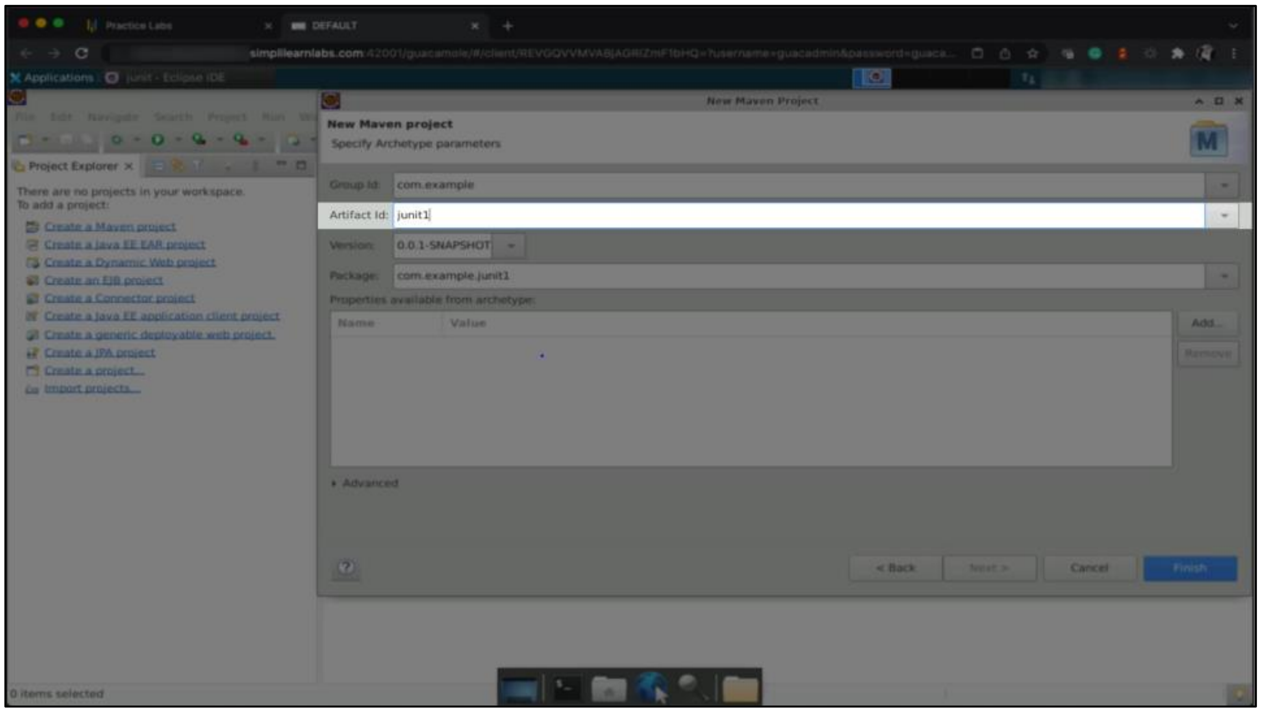
Enter an artifact id.

Group Id: com.example

Version: 0.0.1-SNAPSHOT

Package: com.example

1.6 Name the **Artifact Id** as **junit1** and select **Finish**


New Maven project

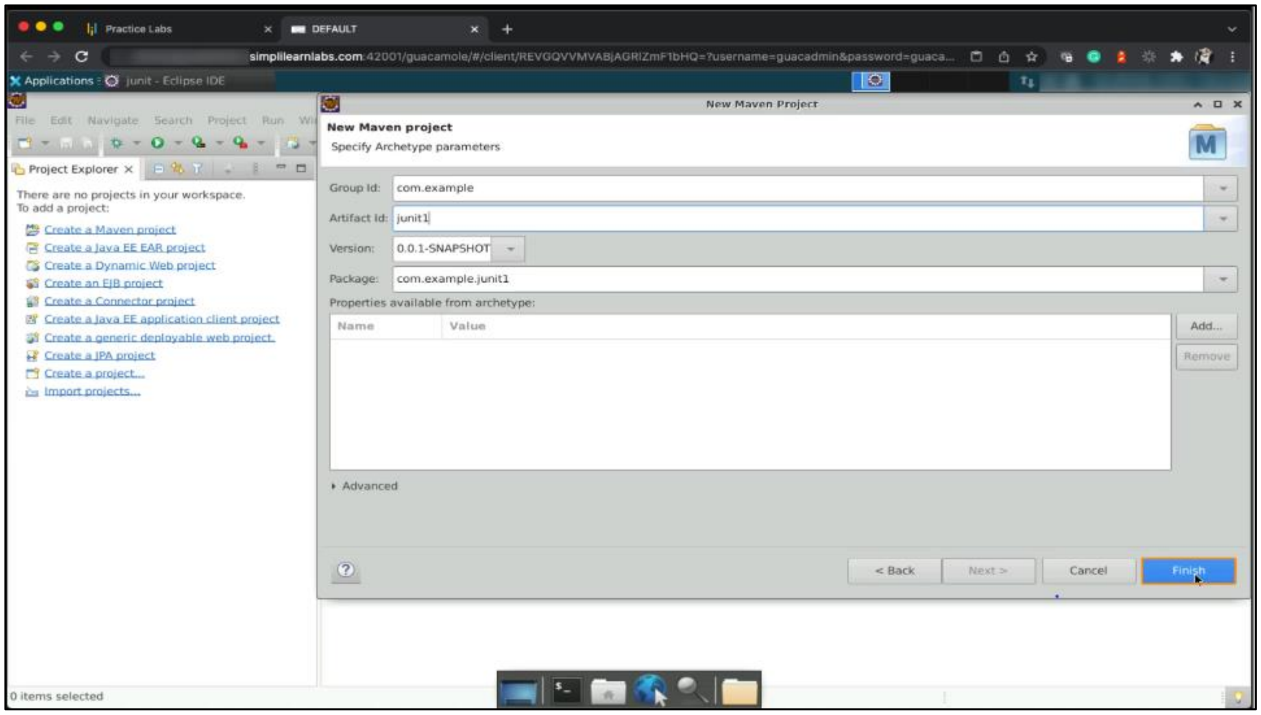
Specify Archetype parameters

Group Id: com.example

Artifact Id: junit1

Version: 0.0.1-SNAPSHOT

Package: com.example.junit1


New Maven project

Specify Archetype parameters

Group Id: com.example

Artifact Id: junit1

Version: 0.0.1-SNAPSHOT

Package: com.example.junit1

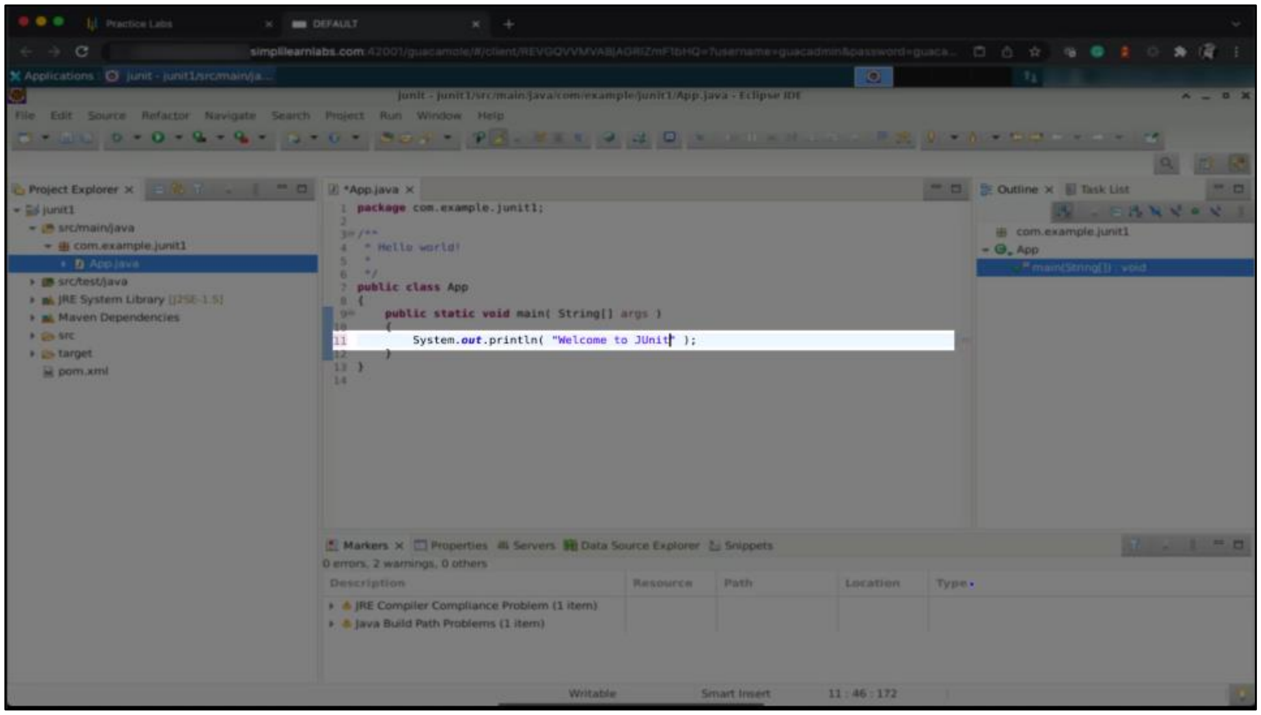
Properties available from archetype:

Name	Value

There are no projects in your workspace.
To add a project:

- Create a Maven project
- Create a Java EE EAR project
- Create a Dynamic Web project
- Create an EJB project
- Create a Connector project
- Create a Java EE application client project
- Create a generic deployable web project.
- Create a JPA project
- Create a project...
- Import projects...

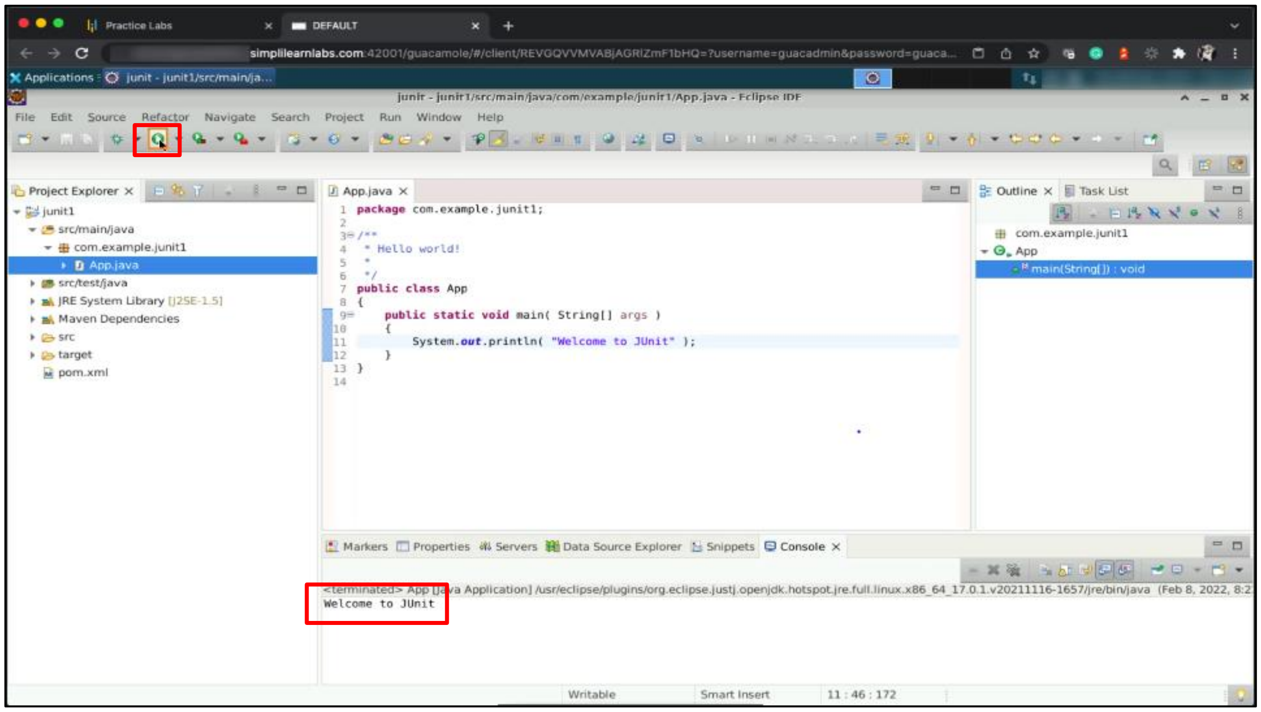
1.7 Click on the **junit1** project, go to **src/main/java > com.example.junit1,** and open the **App.java** file. Change the print statement to **Welcome to Junit**


```
package com.example.junit1;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Welcome to JUnit" );
    }
}
```

0 errors, 2 warnings, 0 others

Description	Resource	Path	Location	Type
JRE Compiler Compliance Problem (1 item)				
Java Build Path Problems (1 item)				

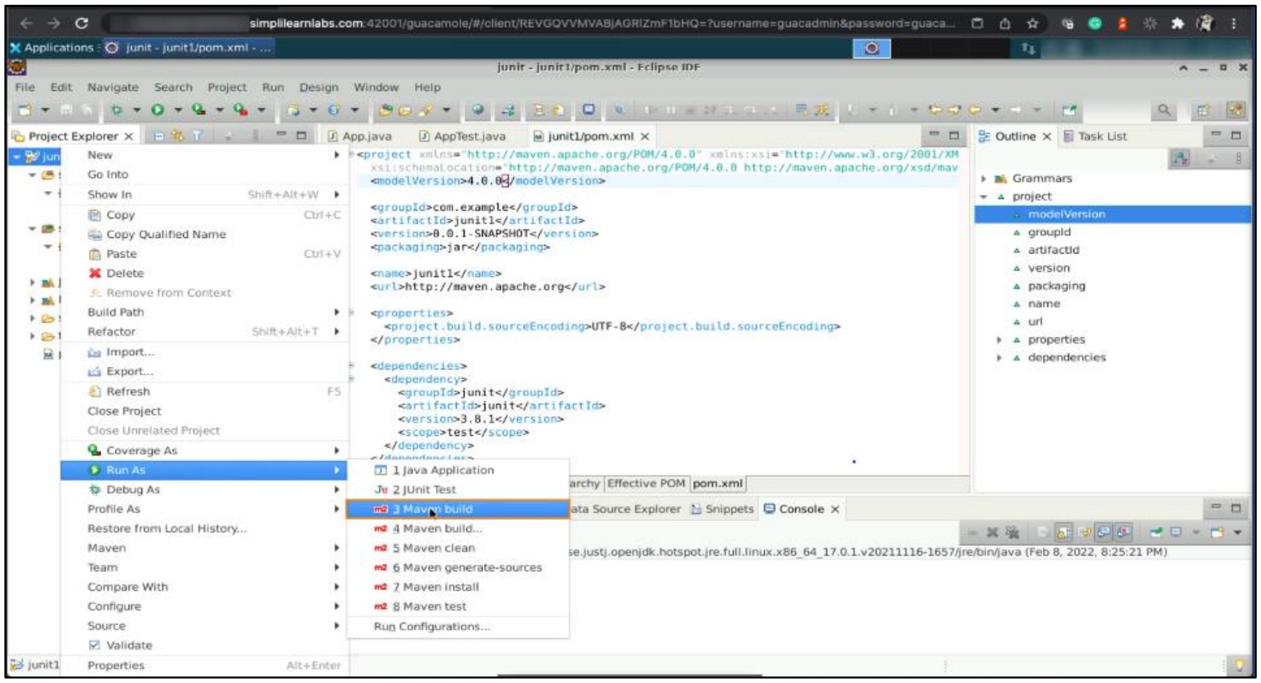
1.8 Run the code. You will see the output **Welcome to JUnit** in the console.



```java
package com.example.junit1;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Welcome to JUnit" );
    }
}
```

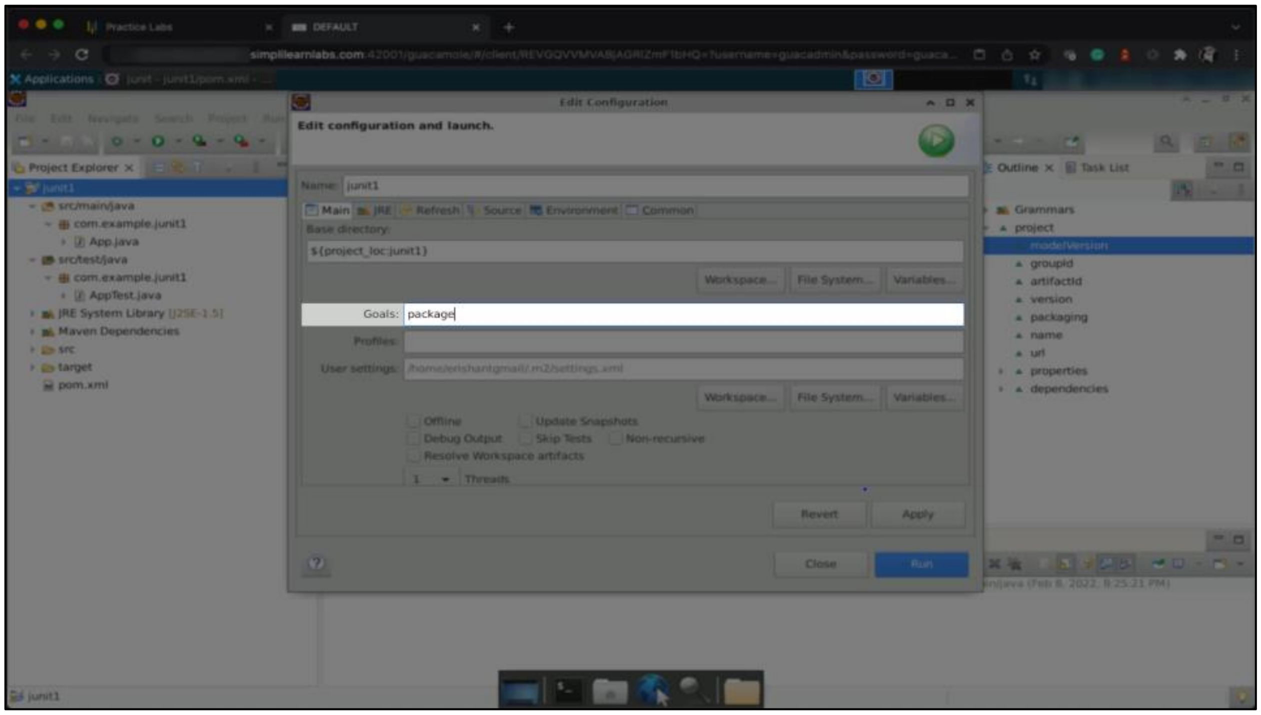
```
<terminated> App [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Feb 8, 2022, 8:2
Welcome to JUnit
```

1.9 Now, run the project as Maven. For that, right-click on project name **junit1,** select **Run As,** then select **Maven build**

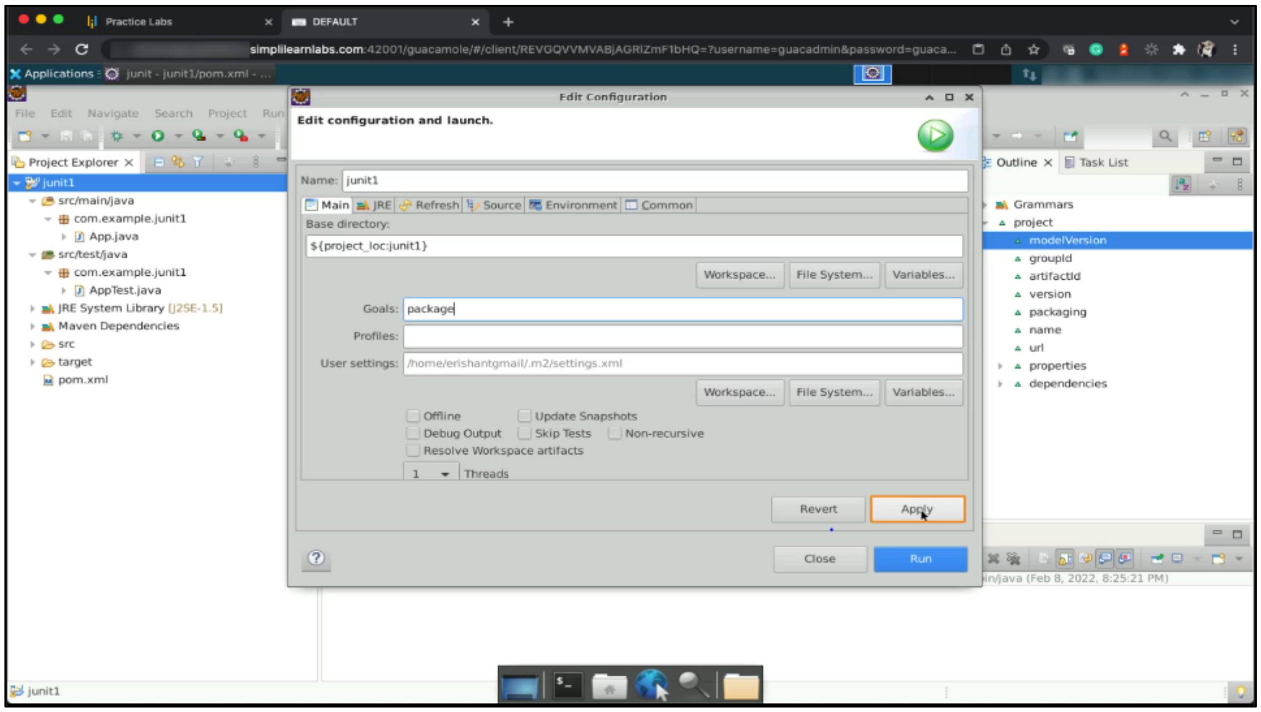


```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XM
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/mav
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>junit1</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>junit1</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>
```

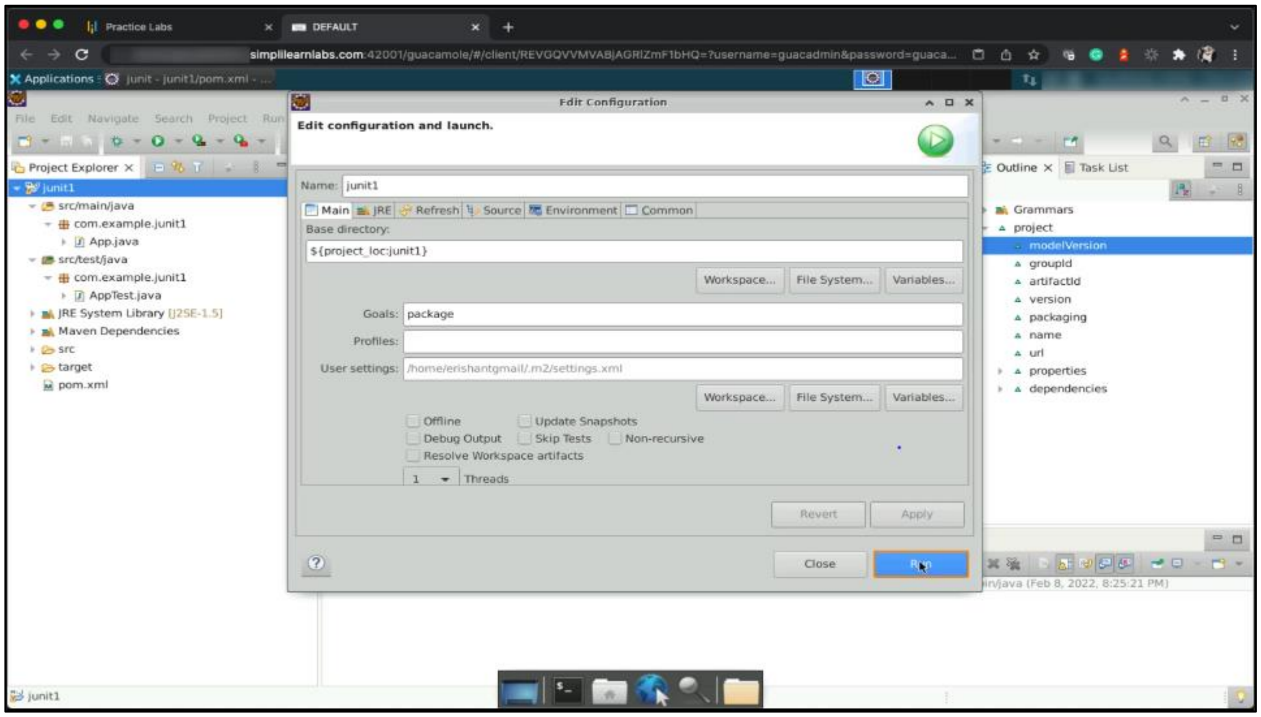
1.10 Name the **Goals** as **package**


Edit Configuration
Edit configuration and launch.
Name: junit1
Base directory: ${project_loc:junit1}
Goals: package
Profiles:
User settings: /home/erishantgmail/.m2/settings.xml
Revert
Apply
Close
Run

1.11 **Apply** the **Goals** setting


Edit Configuration
Edit configuration and launch.
Name: junit1
Base directory: ${project_loc:junit1}
Goals: package
Profiles:
User settings: /home/erishantgmail/.m2/settings.xml
Offline
Update Snapshots
Debug Output
Skip Tests
Non-recursive
Resolve Workspace artifacts
1 Threads
Revert
Apply
Close
Run

1.12 Click on the **Run** option



Edit Configuration

Edit configuration and launch.

Name: junit1

Base directory: ${project_loc:junit1}

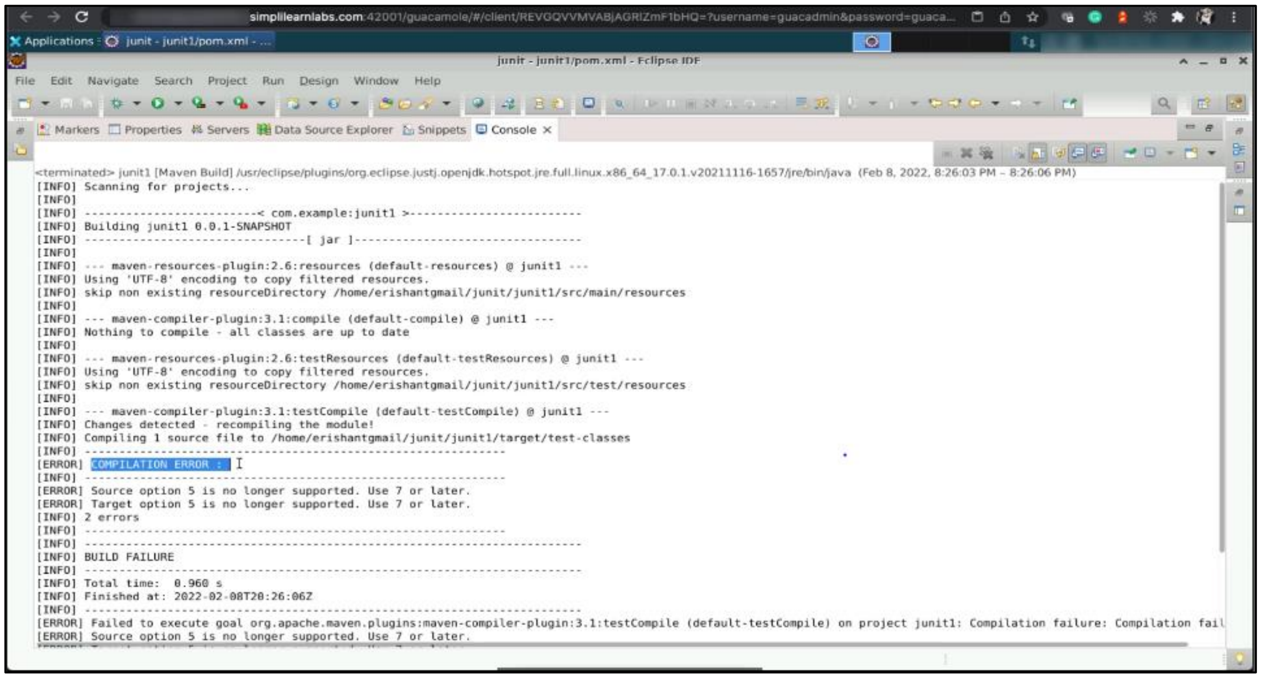
Goals: package

Profiles:

User settings: /home/erishantgmail/.m2/settings.xml

Threads: 1

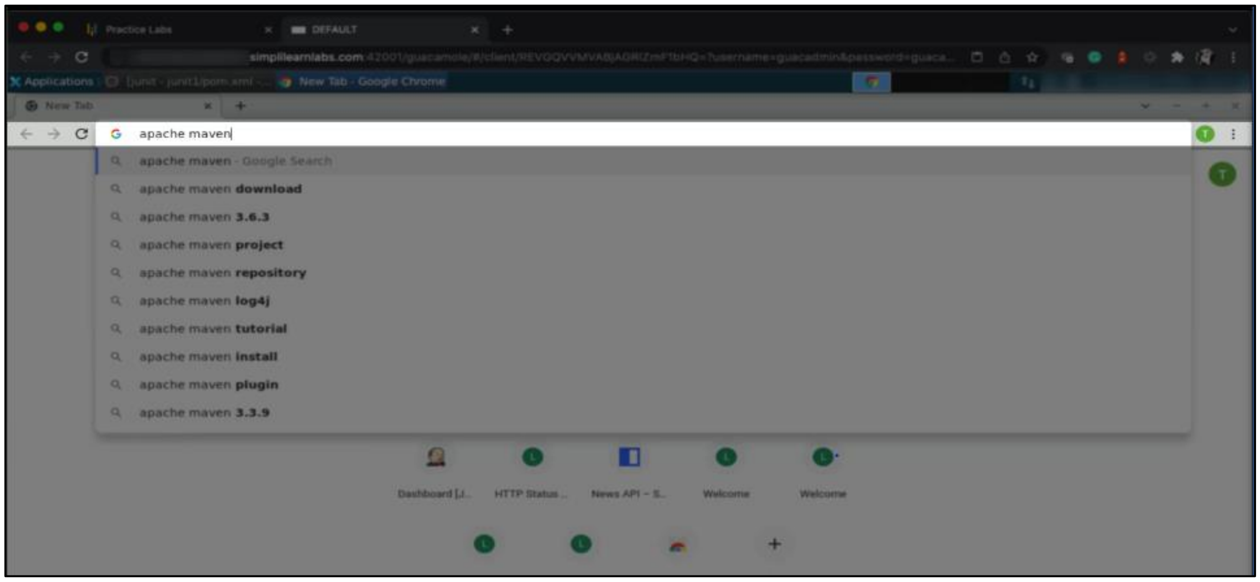
The version error will give you the output with a **COMPILATION ERROR**. Now, let us update the configuration for the version update.



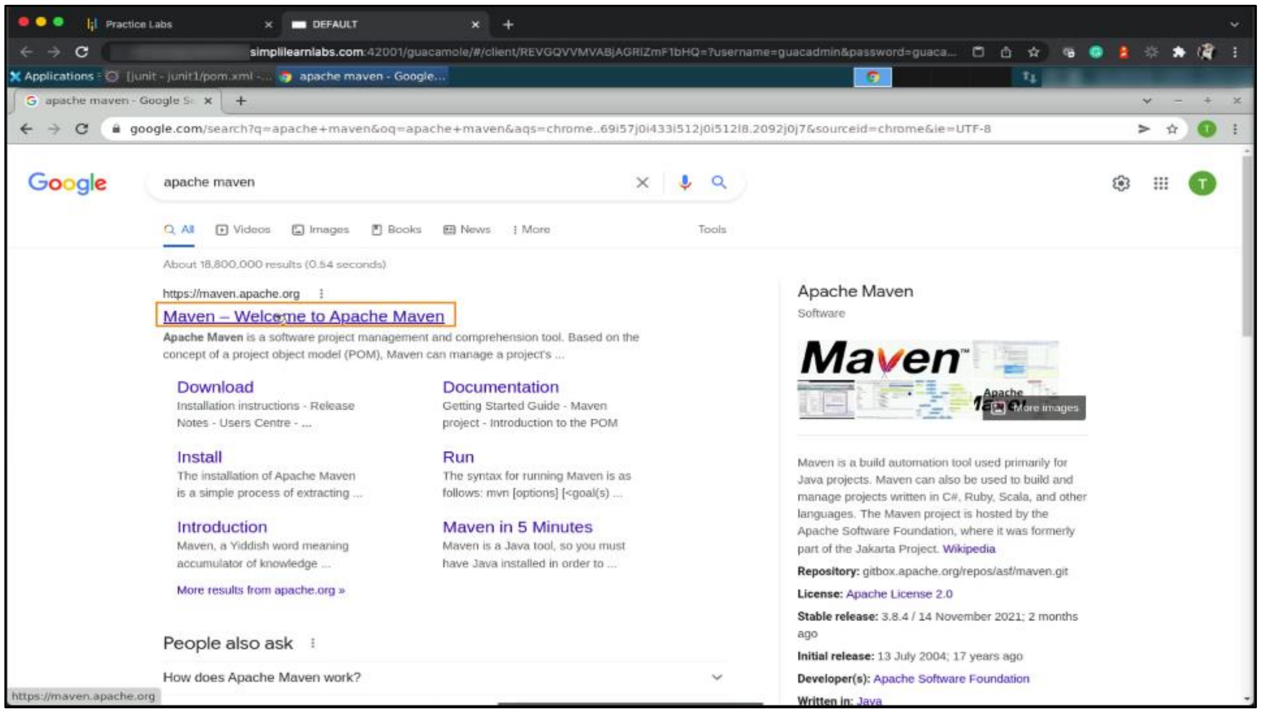
```
<terminated> junit1 [Maven Build] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Feb 8, 2022, 8:26:03 PM – 8:26:06 PM)
[INFO] Scanning for projects...
[INFO]
[INFO] -------------------------< com.example:junit1 >-------------------------
[INFO] Building junit1 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ junit1 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/erishantgmail/junit/junit1/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ junit1 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ junit1 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/erishantgmail/junit/junit1/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ junit1 ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /home/erishantgmail/junit/junit1/target/test-classes
[INFO] -------------------------------------------------------------
[ERROR] COMPILATION ERROR :
[INFO] -------------------------------------------------------------
[ERROR] Source option 5 is no longer supported. Use 7 or later.
[ERROR] Target option 5 is no longer supported. Use 7 or later.
[INFO] 2 errors
[INFO] -------------------------------------------------------------
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.960 s
[INFO] Finished at: 2022-02-08T20:26:06Z
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.1:testCompile (default-testCompile) on project junit1: Compilation failure: Compilation fail
[ERROR] Source option 5 is no longer supported. Use 7 or later.
```

Step 2: Configuring the project with a new compiler plugin

2.1 Open the **browser** and search for **apache maven**



2.2 Select **Maven** official documentation



About 18,800,000 results (0.54 seconds)

https://maven.apache.org

Maven – Welcome to Apache Maven

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's ...

Download
Installation instructions - Release Notes - Users Centre - ...

Documentation
Getting Started Guide - Maven project - Introduction to the POM

Install
The installation of Apache Maven is a simple process of extracting ...

Run
The syntax for running Maven is as follows: mvn [options] [<goal(s) ...

Introduction
Maven, a Yiddish word meaning accumulator of knowledge ...

Maven in 5 Minutes
Maven is a Java tool, so you must have Java installed in order to ...

More results from apache.org »

People also ask

How does Apache Maven work?

Apache Maven
Software

Maven is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. The Maven project is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project. Wikipedia

Repository: gitbox.apache.org/repos/asf/maven.git

License: Apache License 2.0

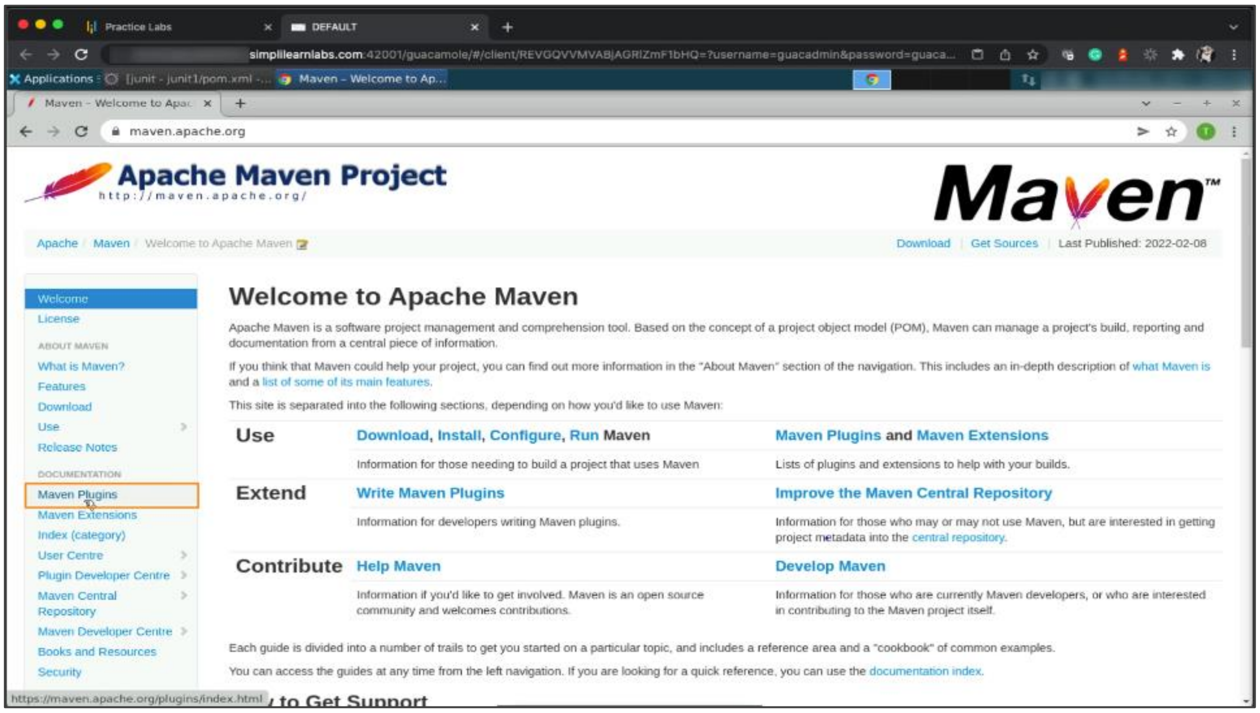
Stable release: 3.8.4 / 14 November 2021; 2 months ago

Initial release: 13 July 2004; 17 years ago

Developer(s): Apache Software Foundation

Written in: Java

2.3 Select **Maven Plugins** from the documentation section



Welcome to Apache Maven

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

If you think that Maven could help your project, you can find out more information in the "About Maven" section of the navigation. This includes an in-depth description of what Maven is and a list of some of its main features.

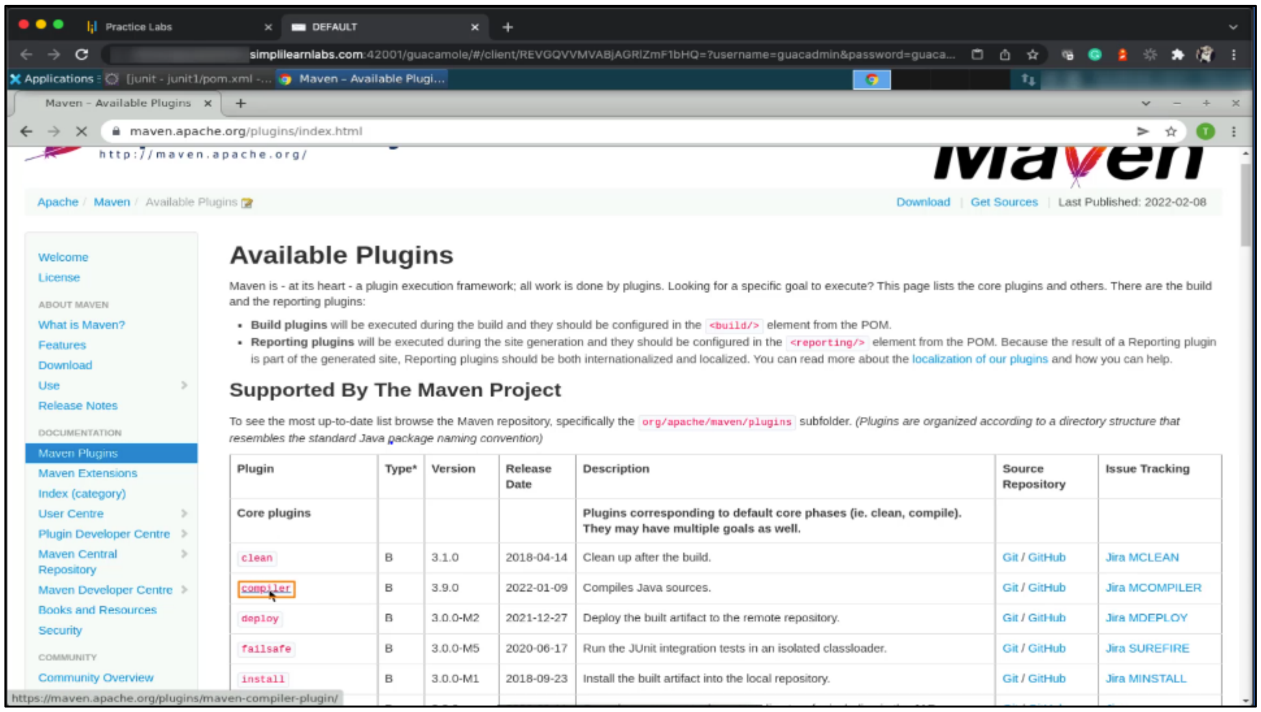
This site is separated into the following sections, depending on how you'd like to use Maven:

Use	Download, Install, Configure, Run Maven Information for those needing to build a project that uses Maven	Maven Plugins and Maven Extensions Lists of plugins and extensions to help with your builds.
Extend	Write Maven Plugins Information for developers writing Maven plugins.	Improve the Maven Central Repository Information for those who may or may not use Maven, but are interested in getting project metadata into the central repository.
Contribute	Help Maven Information if you'd like to get involved. Maven is an open source community and welcomes contributions.	Develop Maven Information for those who are currently Maven developers, or who are interested in contributing to the Maven project itself.

Each guide is divided into a number of trails to get you started on a particular topic, and includes a reference area and a "cookbook" of common examples.

You can access the guides at any time from the left navigation. If you are looking for a quick reference, you can use the documentation index.

2.4 Open the **compiler** plugin



Available Plugins

Maven is - at its heart - a plugin execution framework; all work is done by plugins. Looking for a specific goal to execute? This page lists the core plugins and others. There are the build and the reporting plugins:

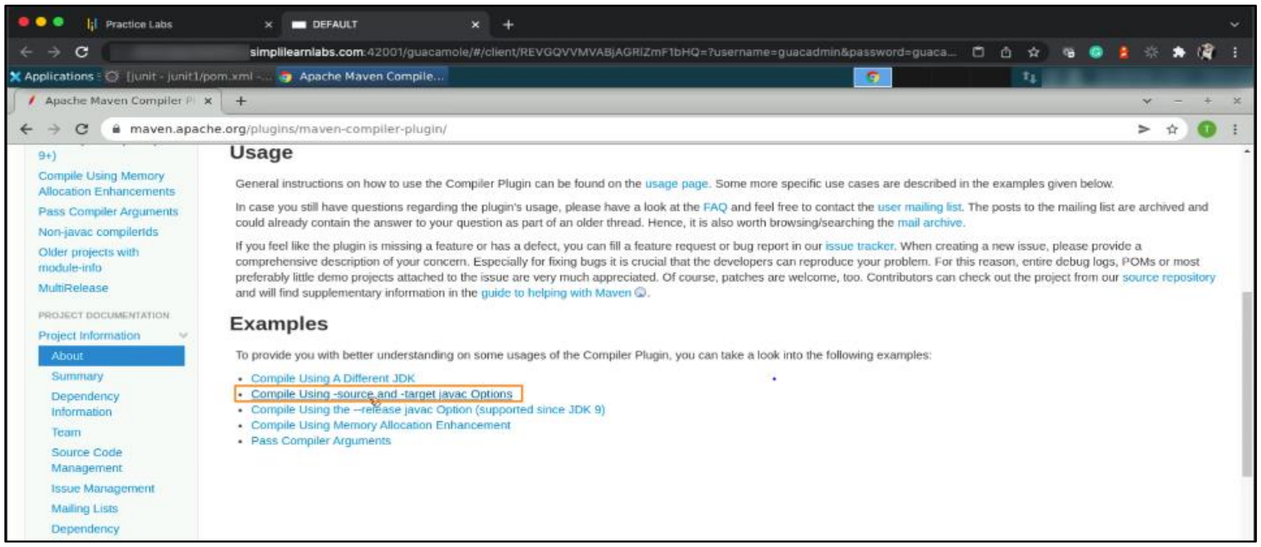
- **Build plugins** will be executed during the build and they should be configured in the `<build/>` element from the POM.
- **Reporting plugins** will be executed during the site generation and they should be configured in the `<reporting/>` element from the POM. Because the result of a Reporting plugin is part of the generated site, Reporting plugins should be both internationalized and localized. You can read more about the localization of our plugins and how you can help.

Supported By The Maven Project

To see the most up-to-date list browse the Maven repository, specifically the `org/apache/maven/plugins` subfolder. *(Plugins are organized according to a directory structure that resembles the standard Java package naming convention)*

Plugin	Type*	Version	Release Date	Description	Source Repository	Issue Tracking
Core plugins				Plugins corresponding to default core phases (ie. clean, compile). They may have multiple goals as well.		
clean	B	3.1.0	2018-04-14	Clean up after the build.	Git / GitHub	Jira MCLEAN
compiler	B	3.9.0	2022-01-09	Compiles Java sources.	Git / GitHub	Jira MCOMPILER
deploy	B	3.0.0-M2	2021-12-27	Deploy the built artifact to the remote repository.	Git / GitHub	Jira MDEPLOY
failsafe	B	3.0.0-M5	2020-06-17	Run the JUnit integration tests in an isolated classloader.	Git / GitHub	Jira SUREFIRE
install	B	3.0.0-M1	2018-09-23	Install the built artifact into the local repository.	Git / GitHub	Jira MINSTALL

2.5 Scroll down the compiler page. In **Examples,** select **Compile Using -source and -target javac Options**



simplilearnlabs.com:42001/guacamole/#/client/REVGQVVMVABjAGRIZmF1bHQ=?username=guacadmin&password=guaca...

maven.apache.org/plugins/maven-compiler-plugin/

9+)
Compile Using Memory Allocation Enhancements
Pass Compiler Arguments
Non-javac compilerIds
Older projects with module-info
MultiRelease

PROJECT DOCUMENTATION
Project Information
About
Summary
Dependency Information
Team
Source Code Management
Issue Management
Mailing Lists
Dependency

Usage

General instructions on how to use the Compiler Plugin can be found on the usage page. Some more specific use cases are described in the examples given below.

In case you still have questions regarding the plugin's usage, please have a look at the FAQ and feel free to contact the user mailing list. The posts to the mailing list are archived and could already contain the answer to your question as part of an older thread. Hence, it is also worth browsing/searching the mail archive.

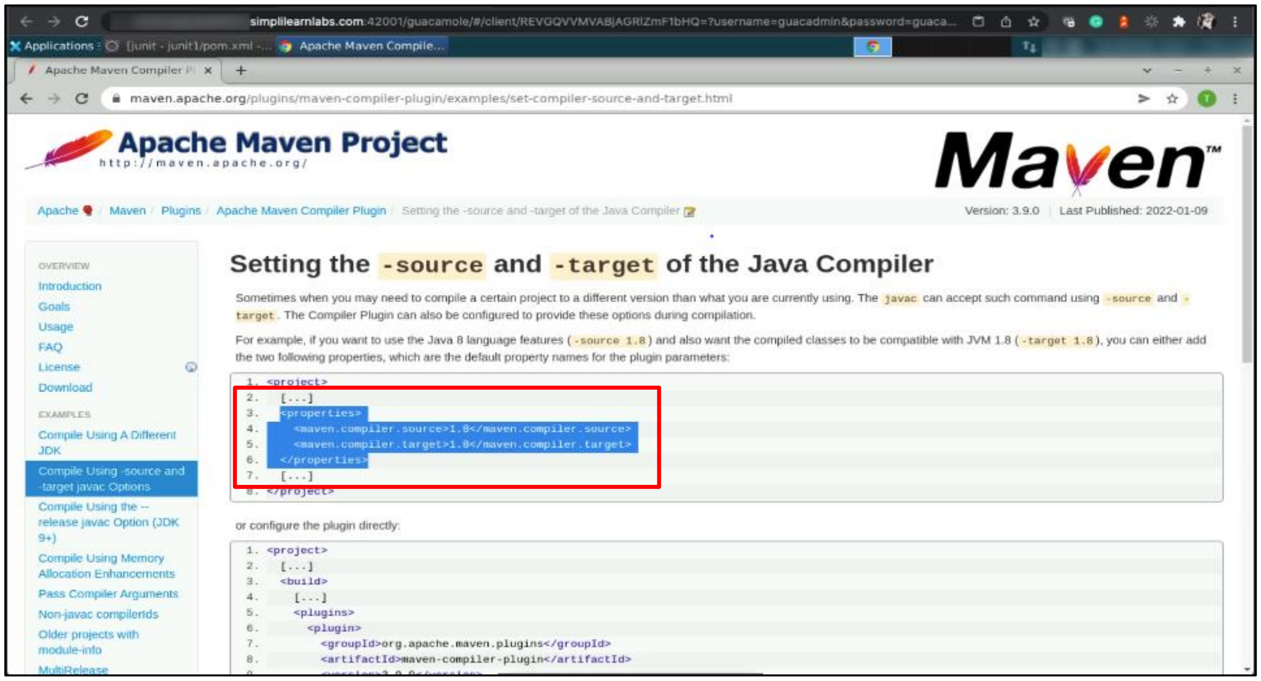
If you feel like the plugin is missing a feature or has a defect, you can fill a feature request or bug report in our issue tracker. When creating a new issue, please provide a comprehensive description of your concern. Especially for fixing bugs it is crucial that the developers can reproduce your problem. For this reason, entire debug logs, POMs or most preferably little demo projects attached to the issue are very much appreciated. Of course, patches are welcome, too. Contributors can check out the project from our source repository and will find supplementary information in the guide to helping with Maven.

Examples

To provide you with better understanding on some usages of the Compiler Plugin, you can take a look into the following examples:

- Compile Using A Different JDK
- Compile Using -source and -target javac Options
- Compile Using the --release javac Option (supported since JDK 9)
- Compile Using Memory Allocation Enhancement
- Pass Compiler Arguments

2.6 Copy the maven properties, go back to the **pom.xml** file in the editor, and paste the copied maven properties code inside **<properties>** tag



simplilearnlabs.com:42001/guacamole/#/client/REVGQVVMVABjAGRIZmF1bHQ=?username=guacadmin&password=guaca...

maven.apache.org/plugins/maven-compiler-plugin/examples/set-compiler-source-and-target.html

Apache / Maven / Plugins / Apache Maven Compiler Plugin / Setting the -source and -target of the Java Compiler | Version: 3.9.0 | Last Published: 2022-01-09

OVERVIEW
Introduction
Goals
Usage
FAQ
License
Download

EXAMPLES
Compile Using A Different JDK
Compile Using -source and -target javac Options
Compile Using the --release javac Option (JDK 9+)
Compile Using Memory Allocation Enhancements
Pass Compiler Arguments
Non-javac compilerIds
Older projects with module-info
MultiRelease

Setting the `-source` and `-target` of the Java Compiler

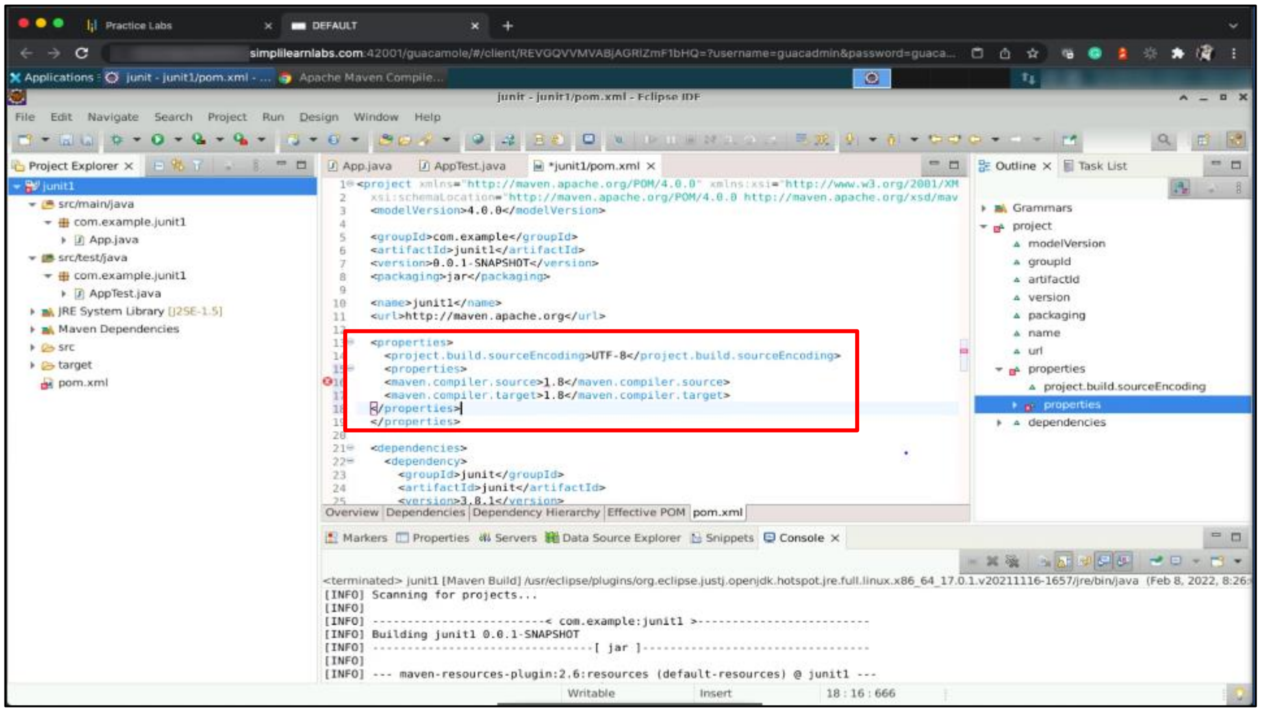
Sometimes when you may need to compile a certain project to a different version than what you are currently using. The `javac` can accept such command using `-source` and `-target`. The Compiler Plugin can also be configured to provide these options during compilation.

For example, if you want to use the Java 8 language features (`-source 1.8`) and also want the compiled classes to be compatible with JVM 1.8 (`-target 1.8`), you can either add the two following properties, which are the default property names for the plugin parameters:

```
<project>
  [...]
  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  [...]
</project>
```

or configure the plugin directly:

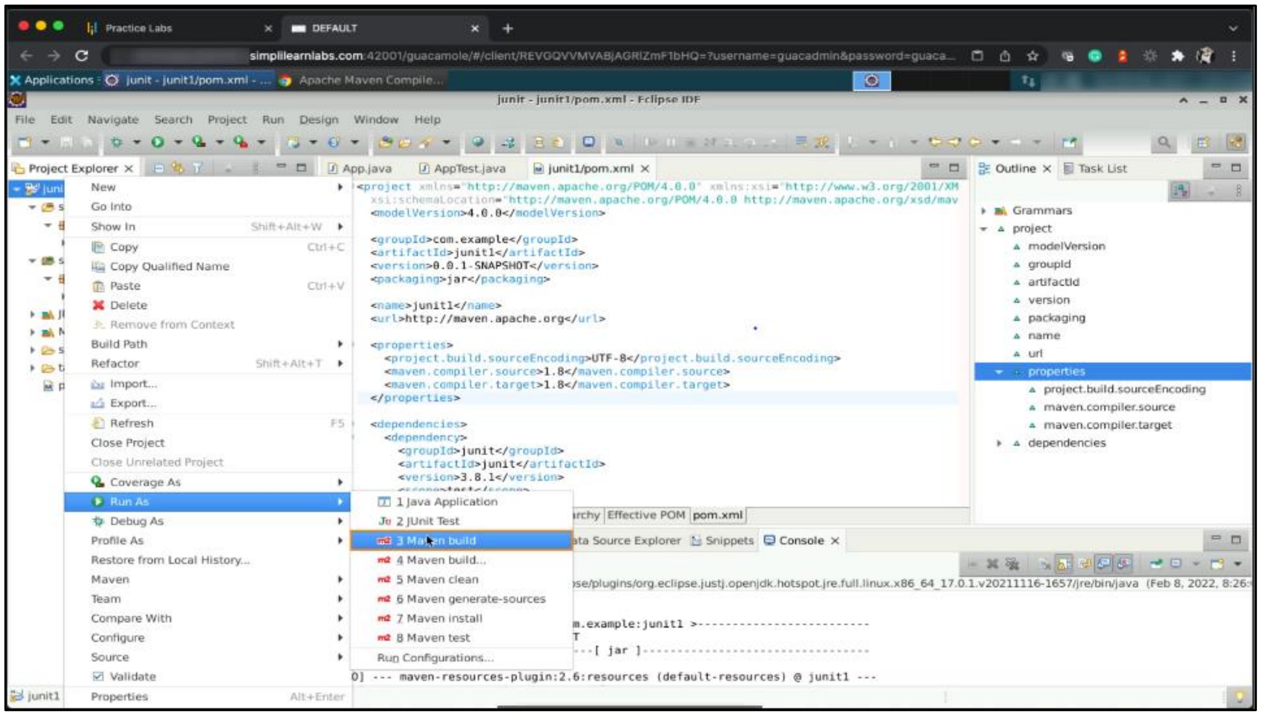
```
<project>
  [...]
  <build>
    [...]
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
```



```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XM
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/mav
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>junit1</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>junit1</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
```

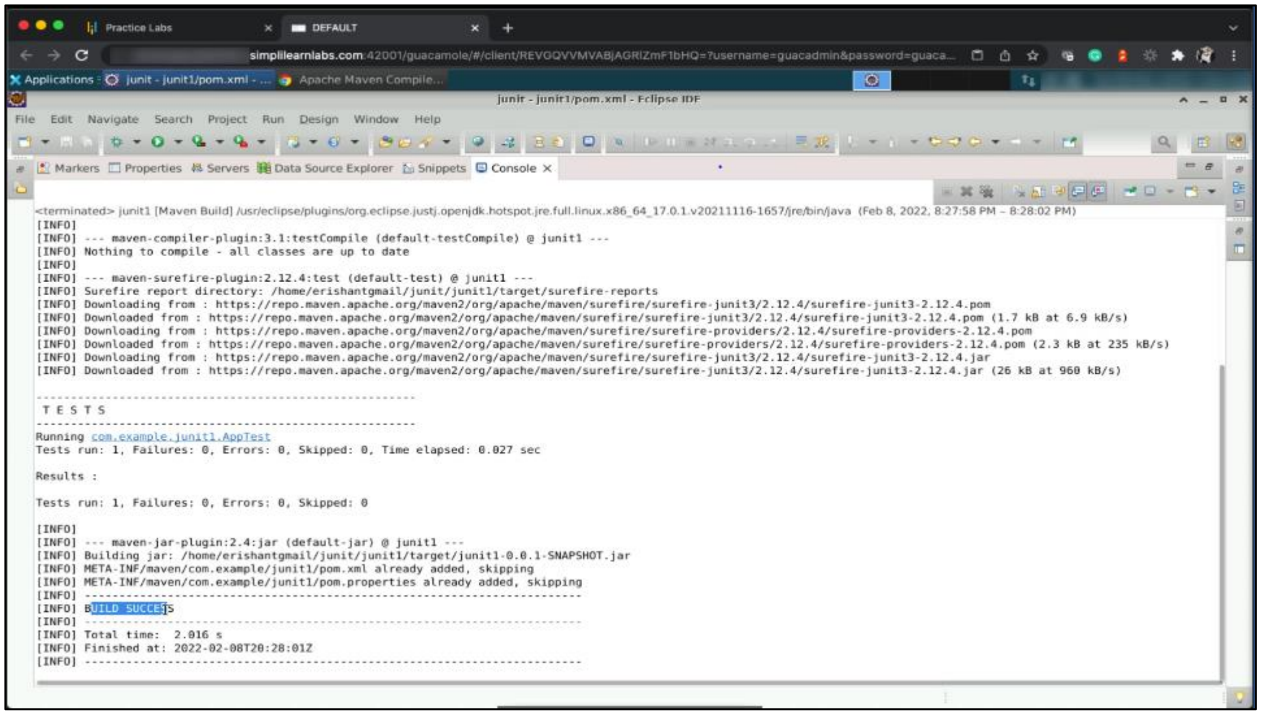
```
<terminated> junit1 [Maven Build] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Feb 8, 2022, 8:26:
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< com.example:junit1 >------------------------
[INFO] Building junit1 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]--------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ junit1 ---
```

2.7 Now, run the project again as **Maven build** as followed in step 1.9



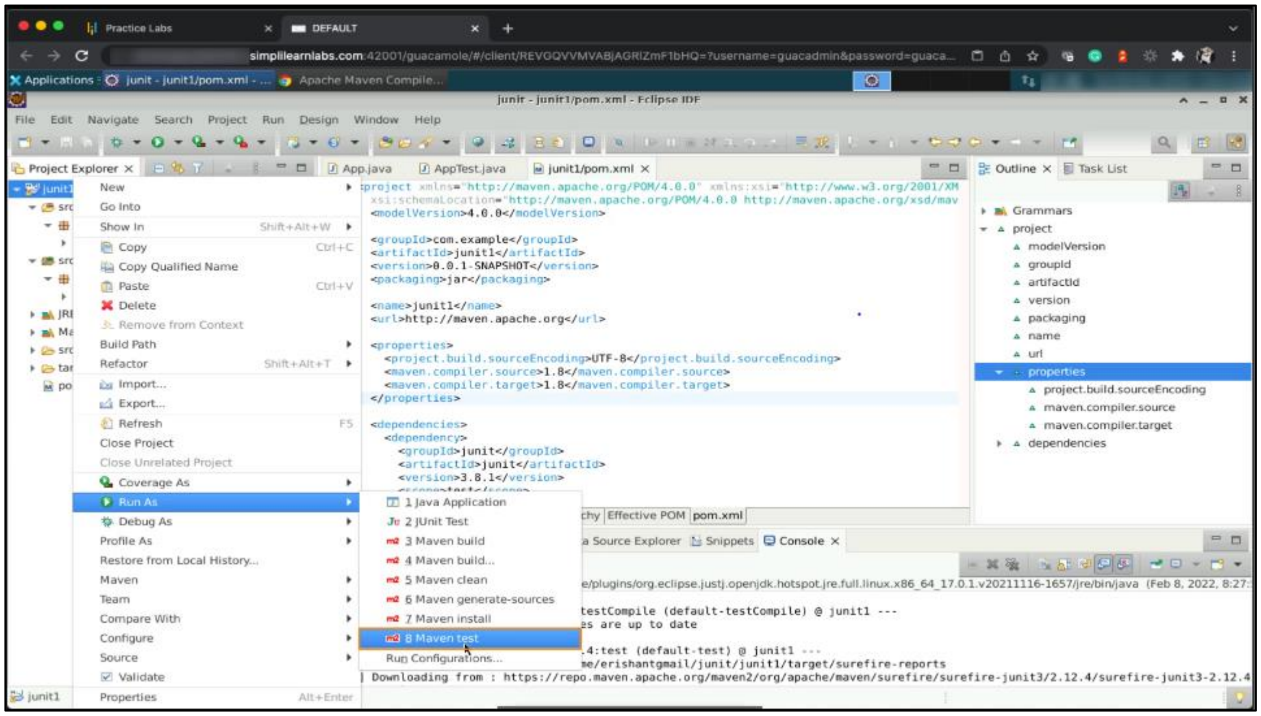
```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XM
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/mav
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>junit1</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>junit1</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
```

You can see the **BUILD SUCCESS** in the console as the output.



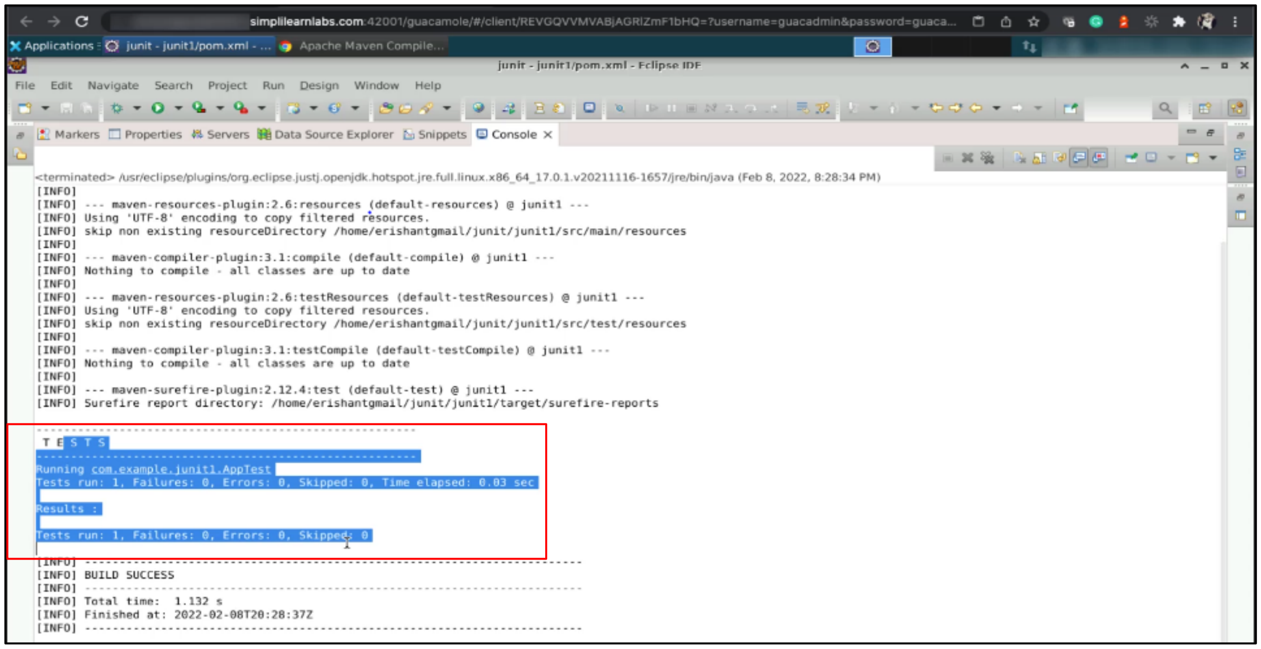
```
<terminated> junit1 [Maven Build] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Feb 8, 2022, 8:27:58 PM – 8:28:02 PM)
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ junit1 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ junit1 ---
[INFO] Surefire report directory: /home/erishantgmail/junit/junit1/target/surefire-reports
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit3/2.12.4/surefire-junit3-2.12.4.pom
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit3/2.12.4/surefire-junit3-2.12.4.pom (1.7 kB at 6.9 kB/s)
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-providers/2.12.4/surefire-providers-2.12.4.pom
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-providers/2.12.4/surefire-providers-2.12.4.pom (2.3 kB at 235 kB/s)
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit3/2.12.4/surefire-junit3-2.12.4.jar
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit3/2.12.4/surefire-junit3-2.12.4.jar (26 kB at 960 kB/s)

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.example.junit1.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.027 sec

Results :

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

[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ junit1 ---
[INFO] Building jar: /home/erishantgmail/junit/junit1/target/junit1-0.0.1-SNAPSHOT.jar
[INFO] META-INF/maven/com.example/junit1/pom.xml already added, skipping
[INFO] META-INF/maven/com.example/junit1/pom.properties already added, skipping
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.016 s
[INFO] Finished at: 2022-02-08T20:28:01Z
[INFO] ------------------------------------------------------------------------
```

2.8 Come back to the project and run the project as a **Maven test**



```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XM
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/mav
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>junit1</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>junit1</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
```

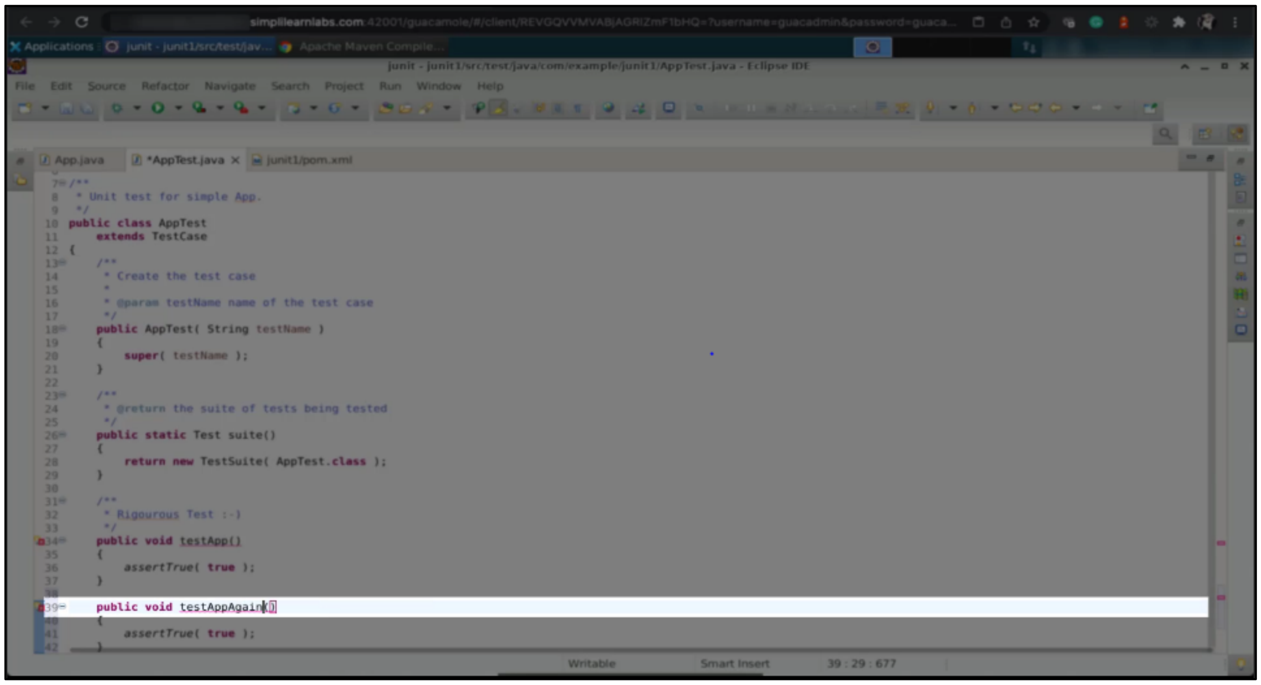
You can see that test cases are executed.



```
<terminated> /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Feb 8, 2022, 8:28:34 PM)
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ junit1 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/erishantgmail/junit/junit1/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ junit1 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ junit1 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/erishantgmail/junit/junit1/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ junit1 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ junit1 ---
[INFO] Surefire report directory: /home/erishantgmail/junit/junit1/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.example.junit1.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.03 sec

Results :

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

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.132 s
[INFO] Finished at: 2022-02-08T20:28:37Z
[INFO] ------------------------------------------------------------------------
```

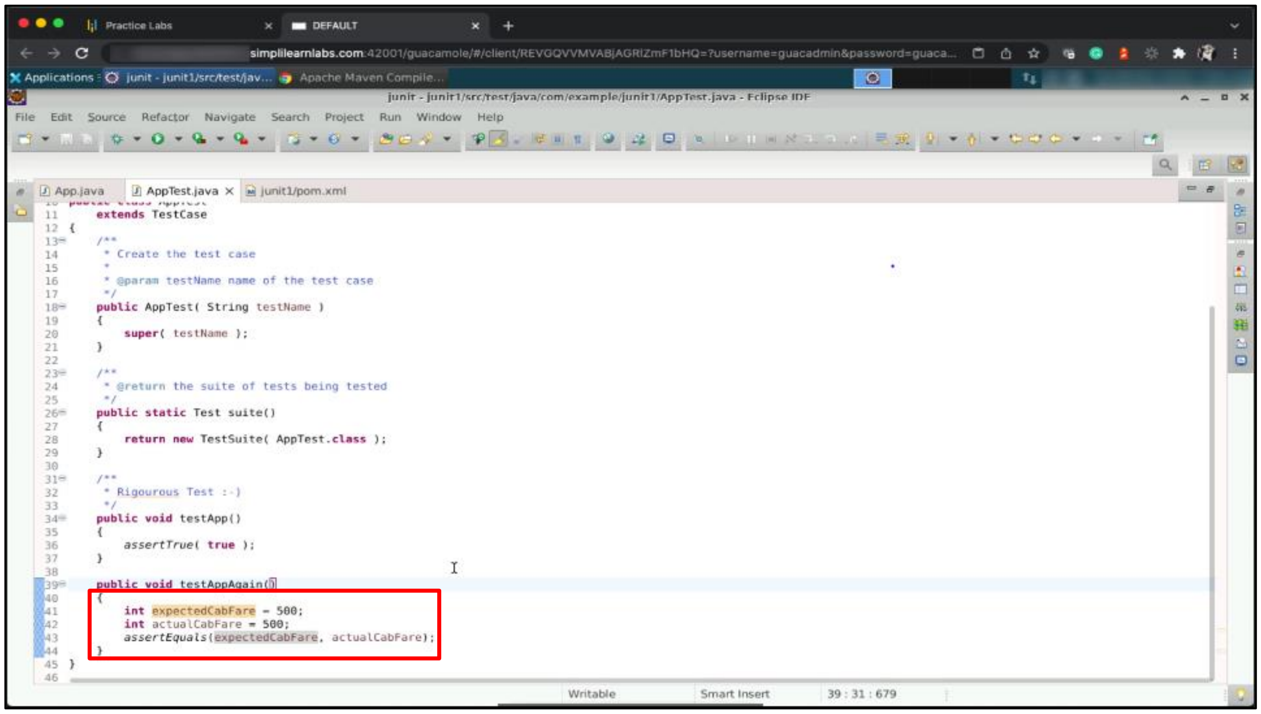
Step 3: Creating a test method

3.1 Go to the project, inside **src/test/java** the **AppTest.java** class is already created. Open the file and create a new method **testAppAgain** for testing



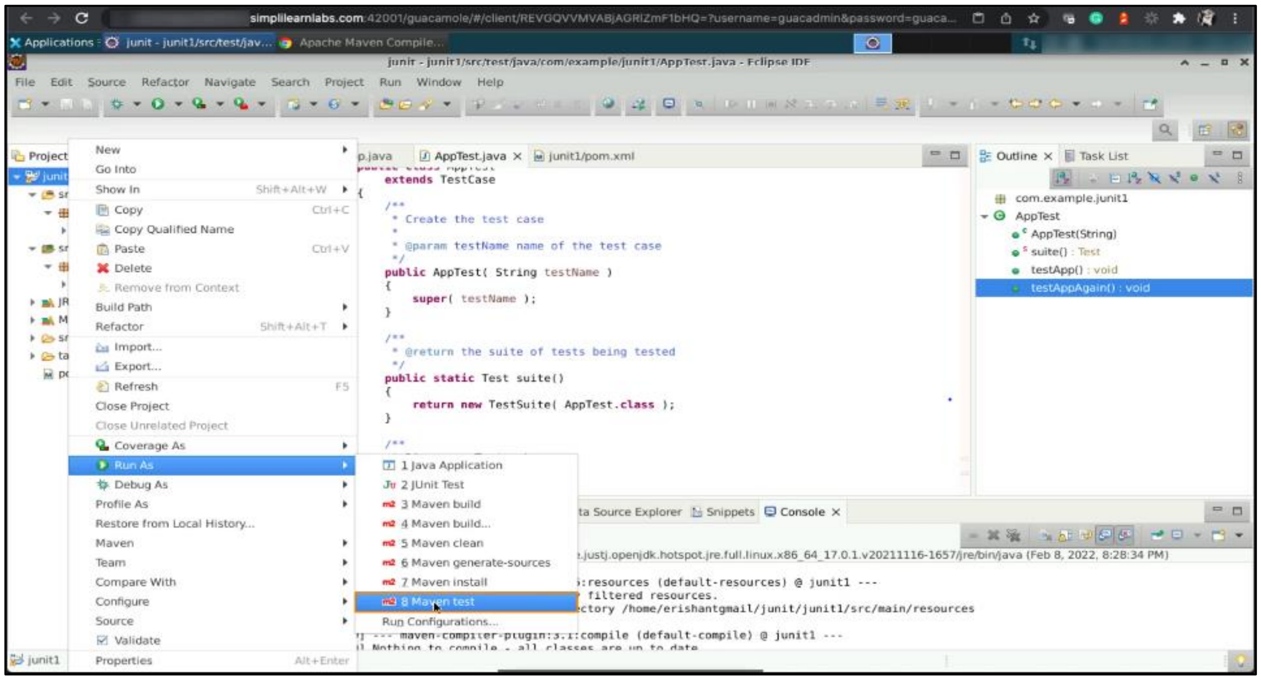
```java
/**
 * Unit test for simple App.
 */
public class AppTest
    extends TestCase
{
    /**
     * Create the test case
     *
     * @param testName name of the test case
     */
    public AppTest( String testName )
    {
        super( testName );
    }

    /**
     * @return the suite of tests being tested
     */
    public static Test suite()
    {
        return new TestSuite( AppTest.class );
    }

    /**
     * Rigourous Test :-)
     */
    public void testApp()
    {
        assertTrue( true );
    }

    public void testAppAgain()
    {
        assertTrue( true );
    }
```

3.2 Create one more method **assertEquals()** in **testAppAgain()** method and local variables **expectedCabFare** and **actualCabFare** to pass from this method for testing



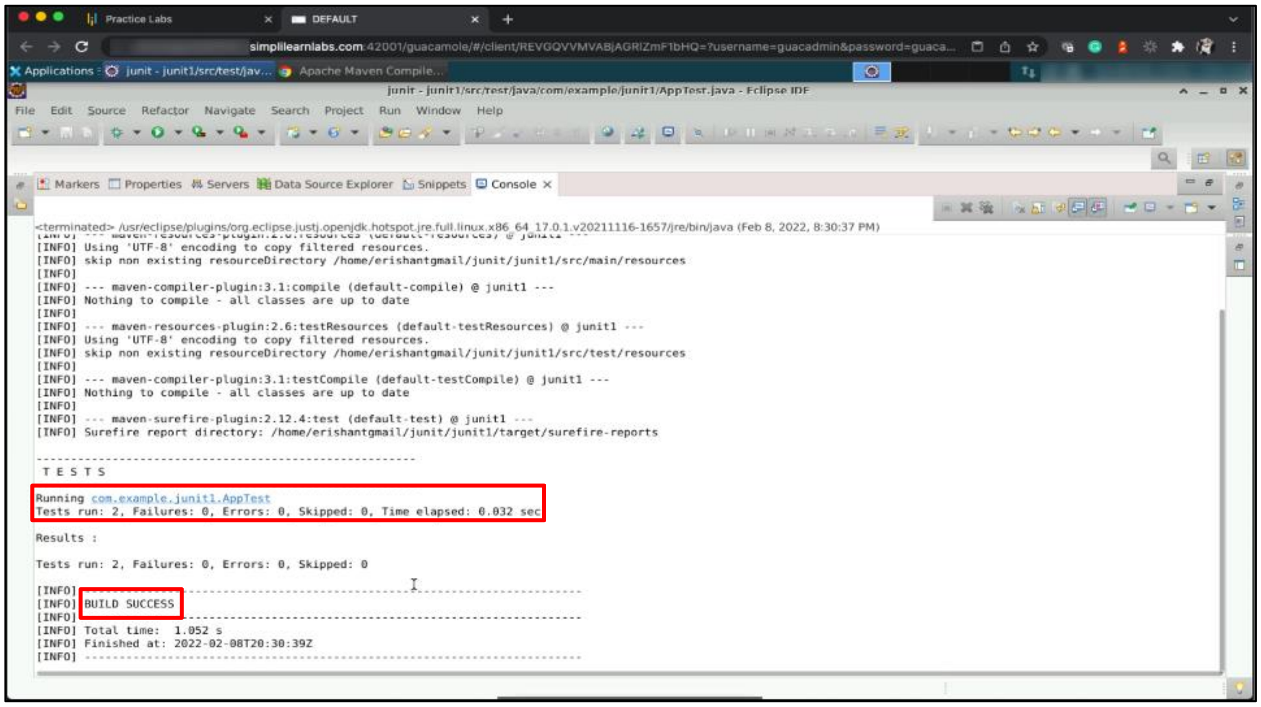
```
    extends TestCase
{
    /**
     * Create the test case
     *
     * @param testName name of the test case
     */
    public AppTest( String testName )
    {
        super( testName );
    }

    /**
     * @return the suite of tests being tested
     */
    public static Test suite()
    {
        return new TestSuite( AppTest.class );
    }

    /**
     * Rigourous Test :-)
     */
    public void testApp()
    {
        assertTrue( true );
    }

    public void testAppAgain()
    {
        int expectedCabFare = 500;
        int actualCabFare = 500;
        assertEquals(expectedCabFare, actualCabFare);
    }
}
```

3.3 Run the project as a **Maven test**



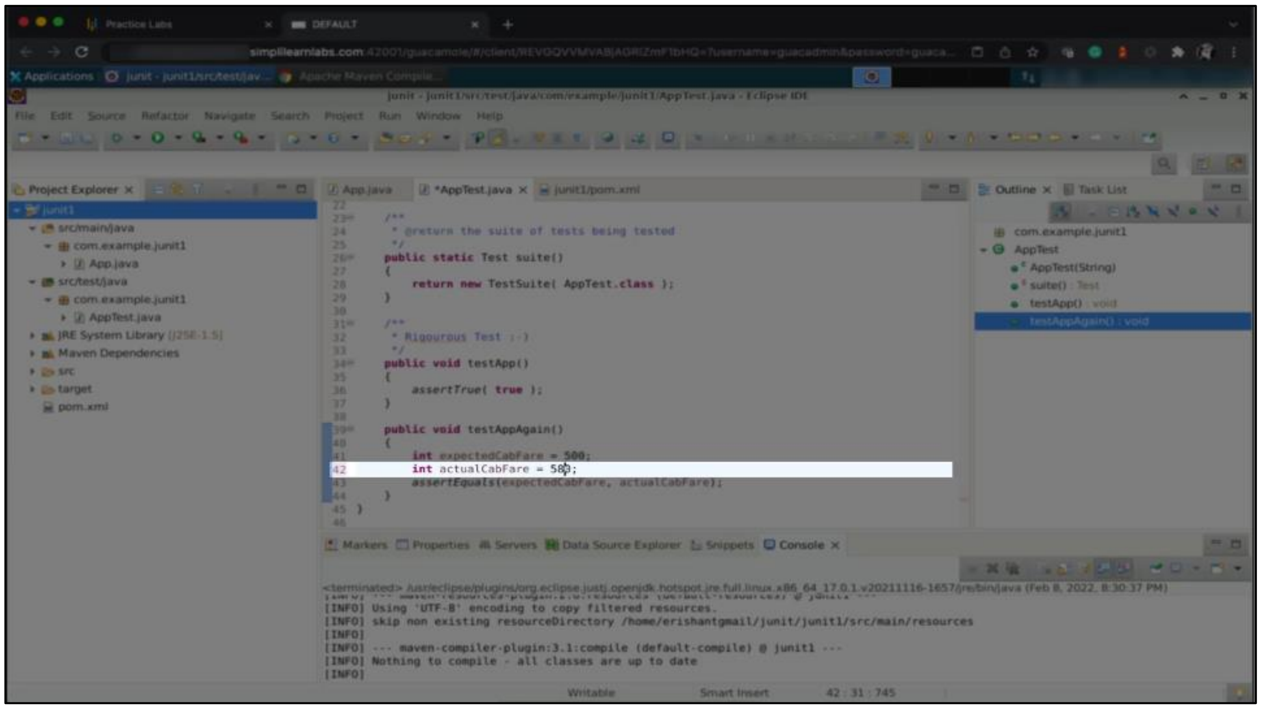
```
.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Feb 8, 2022, 8:28:34 PM)

:resources (default-resources) @ junit1 ---
 filtered resources.
ctory /home/erishantgmail/junit/junit1/src/main/resources

--- maven-compiler-plugin:3.1:compile (default-compile) @ junit1 ---
Nothing to compile - all classes are up to date
```

You can see now, the 2 test cases have passed successfully.



```
<terminated> /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Feb 8, 2022, 8:30:37 PM)
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/erishantgmail/junit/junit1/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ junit1 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ junit1 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/erishantgmail/junit/junit1/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ junit1 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ junit1 ---
[INFO] Surefire report directory: /home/erishantgmail/junit/junit1/target/surefire-reports

-------------------------------------------------------
 T E S T S

Running com.example.junit1.AppTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.032 sec

Results :

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.052 s
[INFO] Finished at: 2022-02-08T20:30:39Z
[INFO] ------------------------------------------------------------------------
```

3.4 Now, change **actualCabFare** to **580**

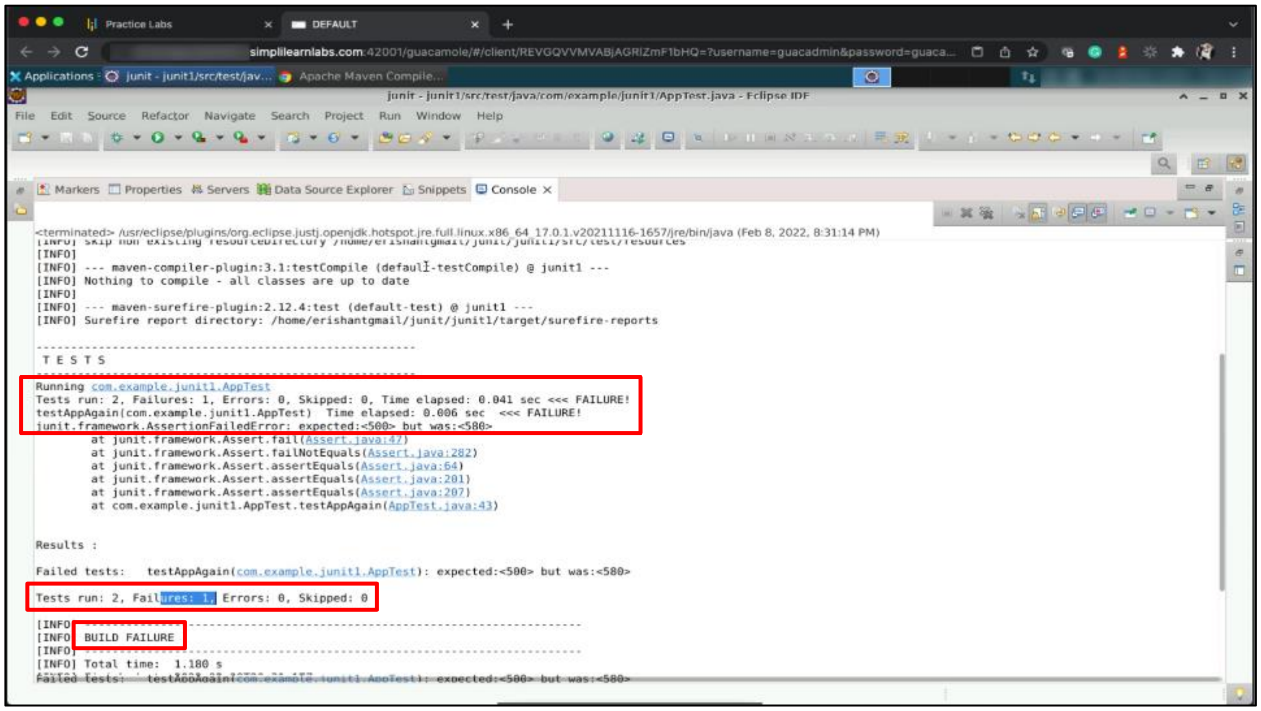


```java
    /**
     * @return the suite of tests being tested
     */
    public static Test suite()
    {
        return new TestSuite( AppTest.class );
    }

    /**
     * Rigourous Test :-)
     */
    public void testApp()
    {
        assertTrue( true );
    }

    public void testAppAgain()
    {
        int expectedCabFare = 500;
        int actualCabFare = 580;
        assertEquals(expectedCabFare, actualCabFare);
    }
}
```

```
<terminated> /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Feb 8, 2022, 8:30:37 PM)
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/erishantgmail/junit/junit1/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ junit1 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
```

3.5 Again, run the project as a **Maven test**

You can see in the console that the build failed because 1 test case failed.



```
<terminated> /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Feb 8, 2022, 8:31:14 PM)
[INFO] skip non existing resourceDirectory /home/erishantgmail/junit/junit1/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ junit1 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ junit1 ---
[INFO] Surefire report directory: /home/erishantgmail/junit/junit1/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.example.junit1.AppTest
Tests run: 2, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.041 sec <<< FAILURE!
testAppAgain(com.example.junit1.AppTest)  Time elapsed: 0.006 sec  <<< FAILURE!
junit.framework.AssertionFailedError: expected:<500> but was:<580>
        at junit.framework.Assert.fail(Assert.java:47)
        at junit.framework.Assert.failNotEquals(Assert.java:282)
        at junit.framework.Assert.assertEquals(Assert.java:64)
        at junit.framework.Assert.assertEquals(Assert.java:201)
        at junit.framework.Assert.assertEquals(Assert.java:207)
        at com.example.junit1.AppTest.testAppAgain(AppTest.java:43)


Results :

Failed tests:   testAppAgain(com.example.junit1.AppTest): expected:<500> but was:<580>

Tests run: 2, Failures: 1, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.180 s
Failed tests:   testAppAgain(com.example.junit1.AppTest): expected:<500> but was:<580>
```

This is how we can create a Maven Project and use JUnit in it. When we create a Maven project, JUnit automatically configures, and test classes are created by default. Whenever the test case fails, JUnit throws an error.