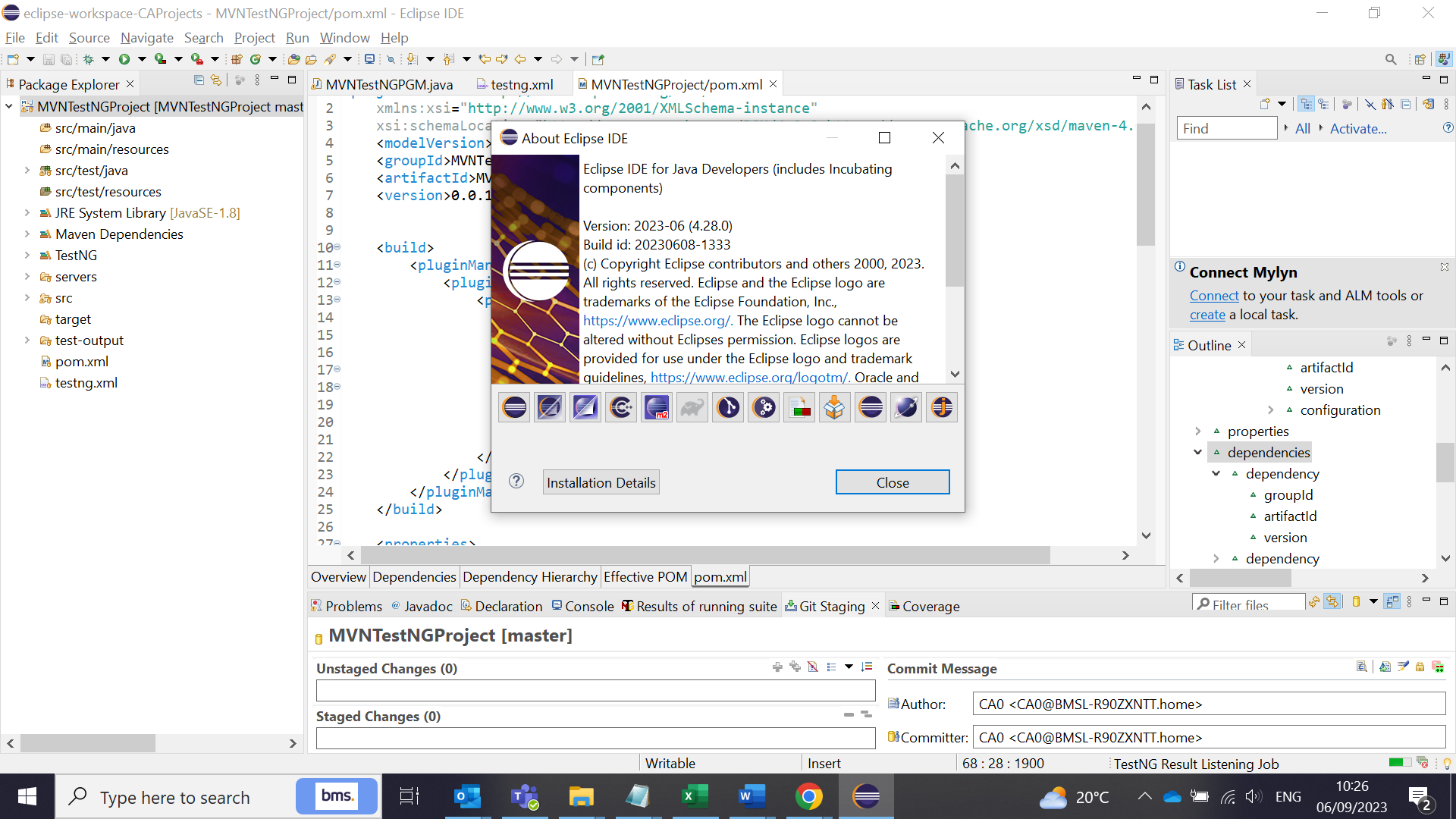
How to create TestNG project in Eclipse:

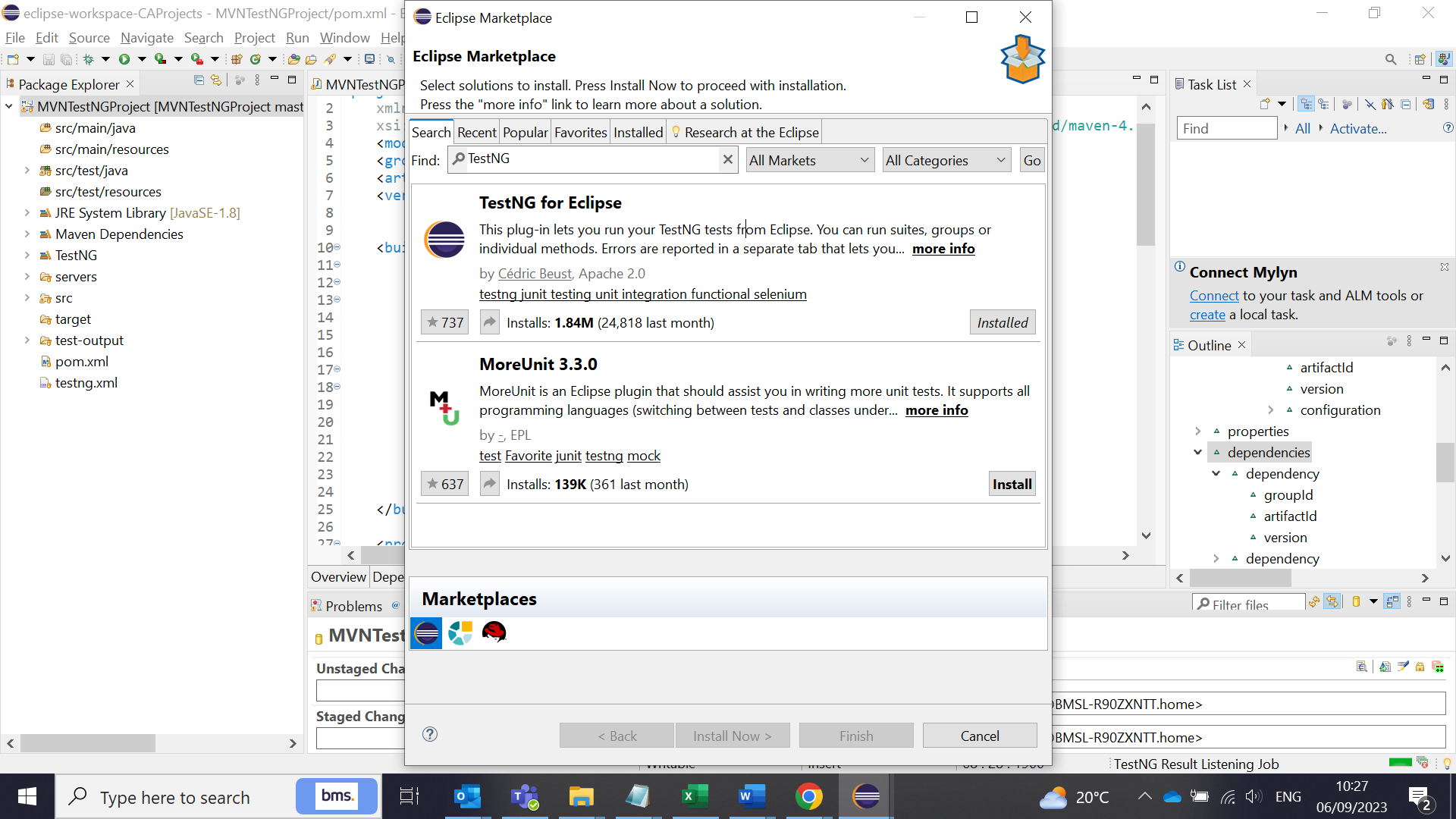
1) Download latest version of Eclipse - Eclipse IDE for Java Developers (includes Incubating components)

Version: 2023-06 (4.28.0)

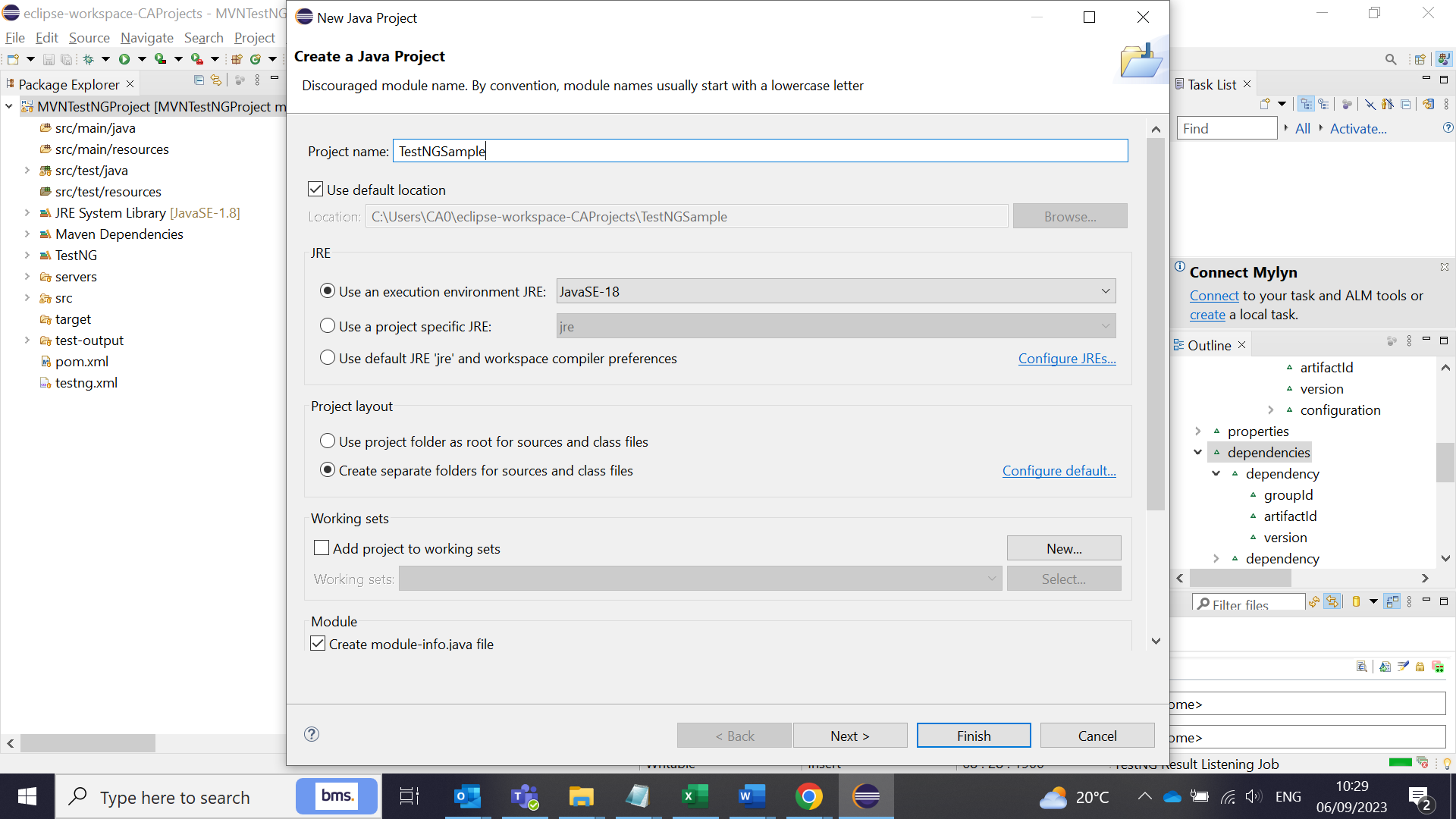
Build id: 20230608-1333



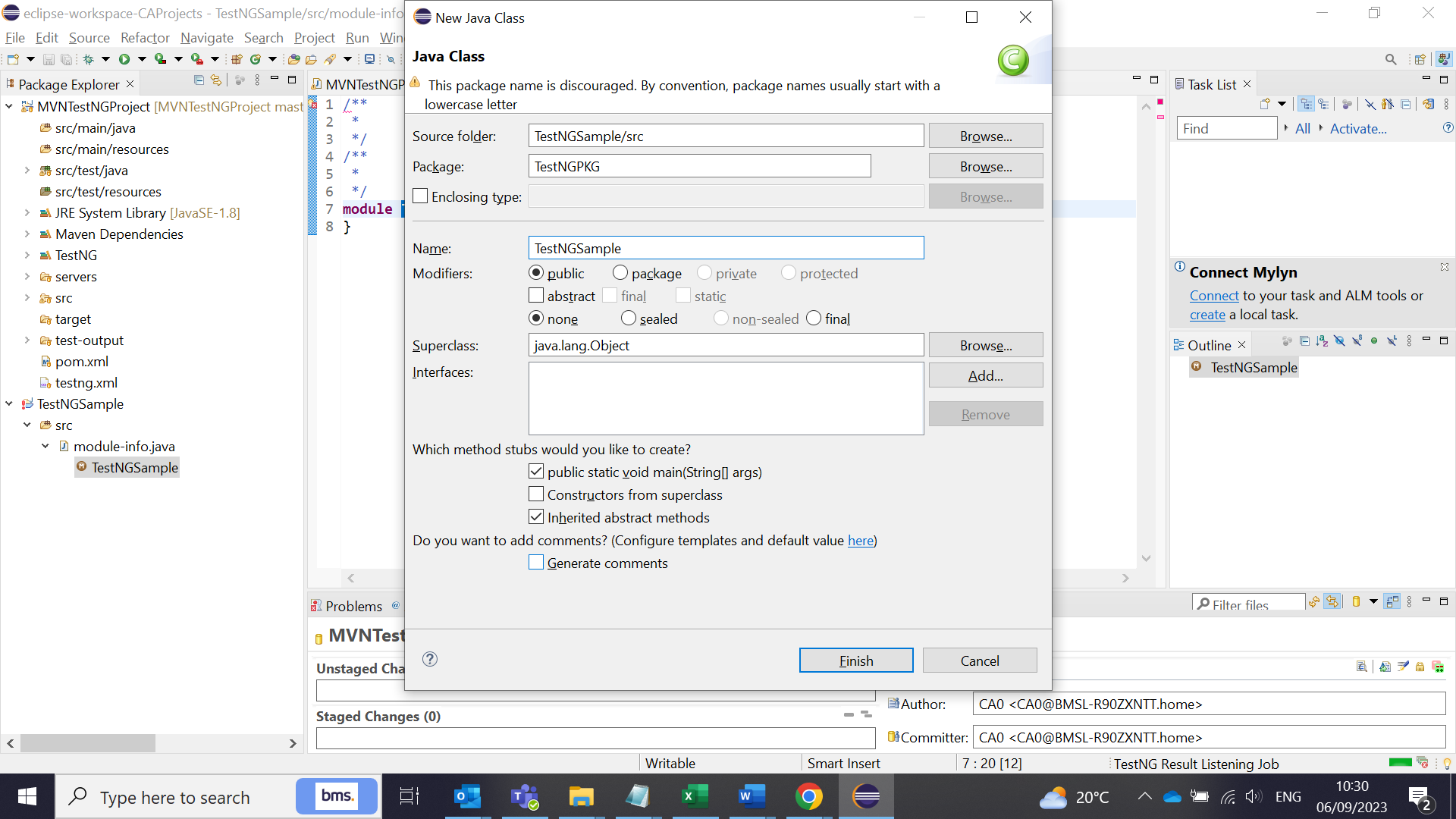
2) Go to Market Place and find TestNG in Eclipse and install the TestNG plugin



3) Create a Java project:

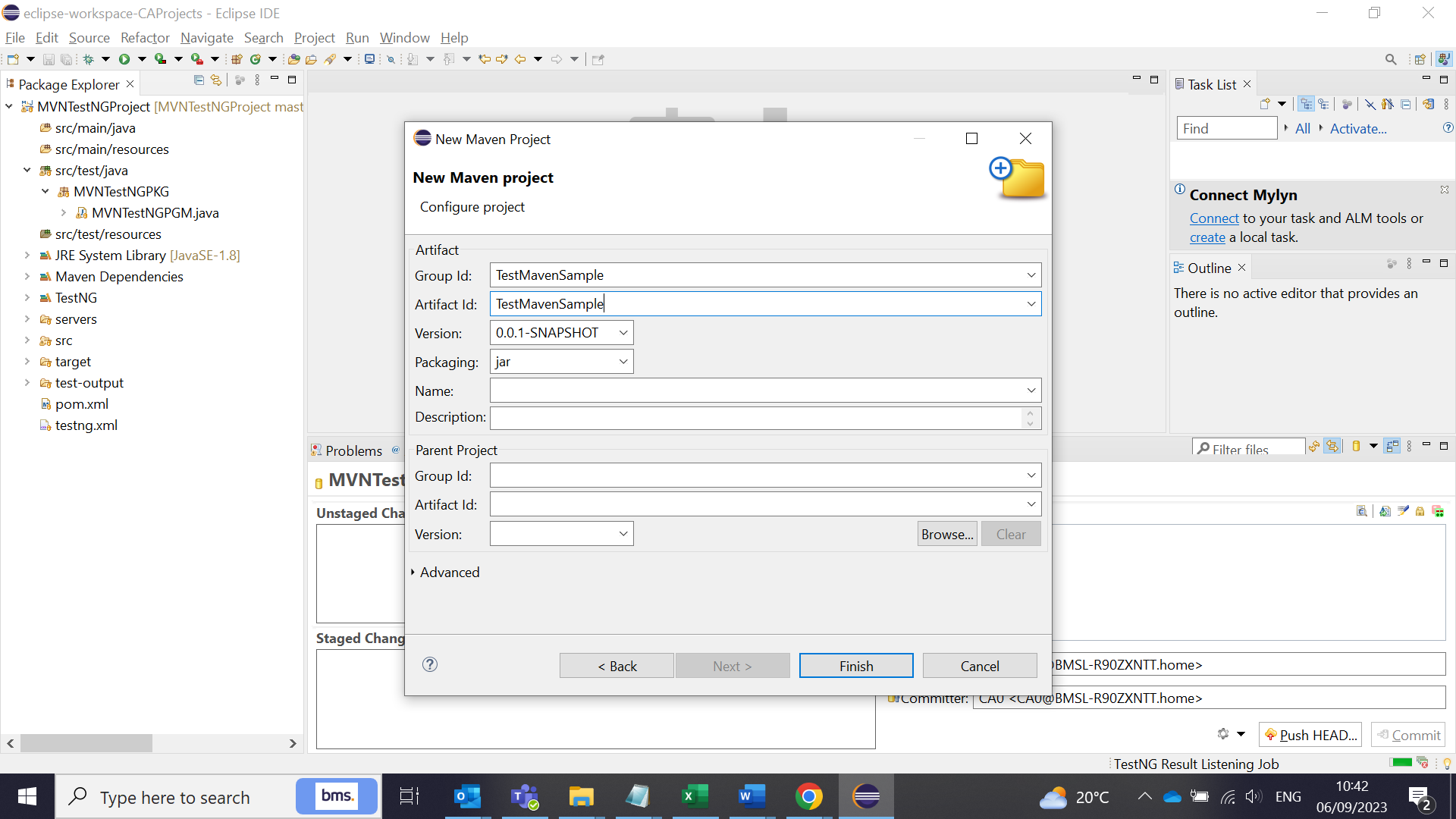


Create a Class file:

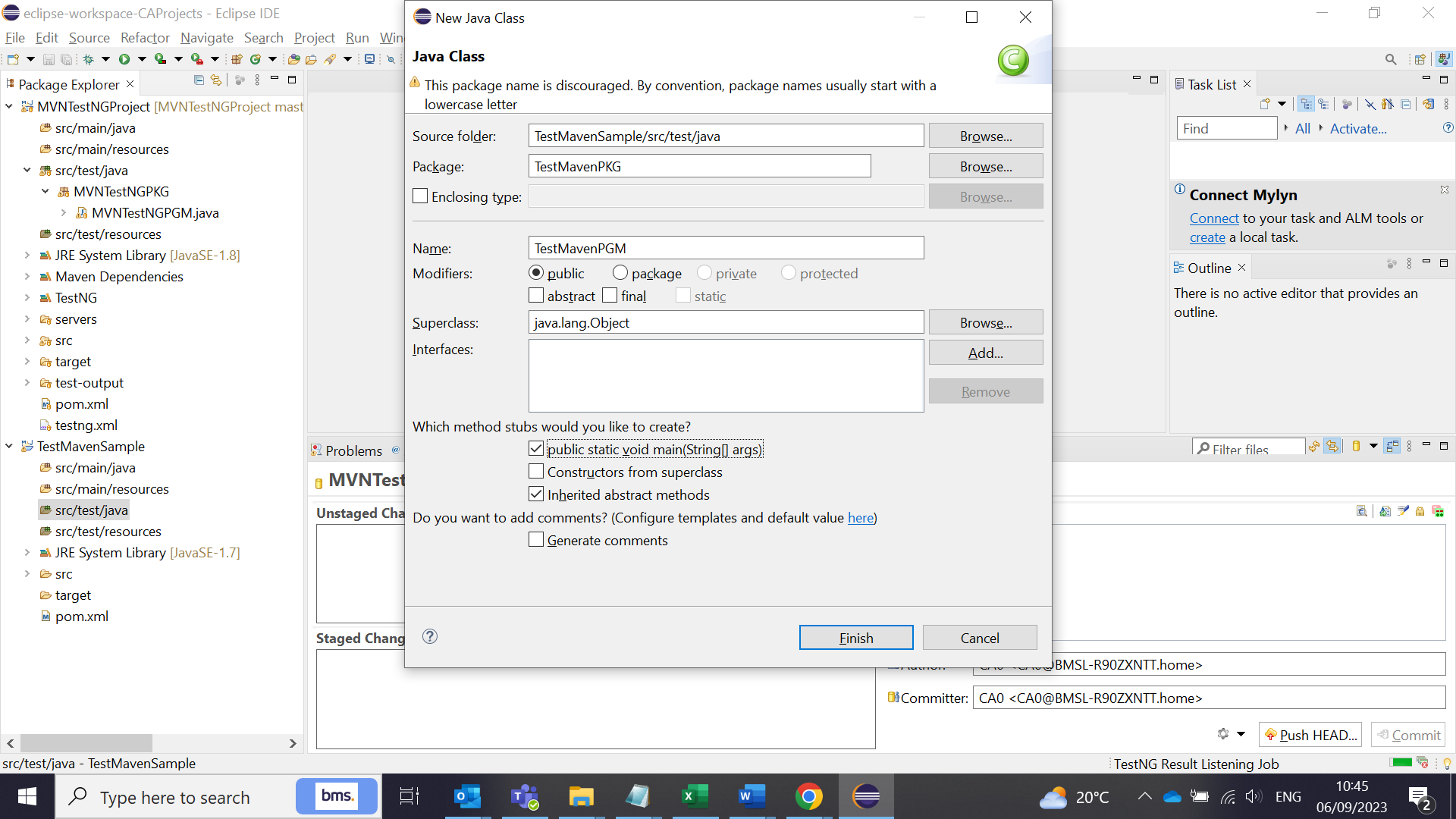


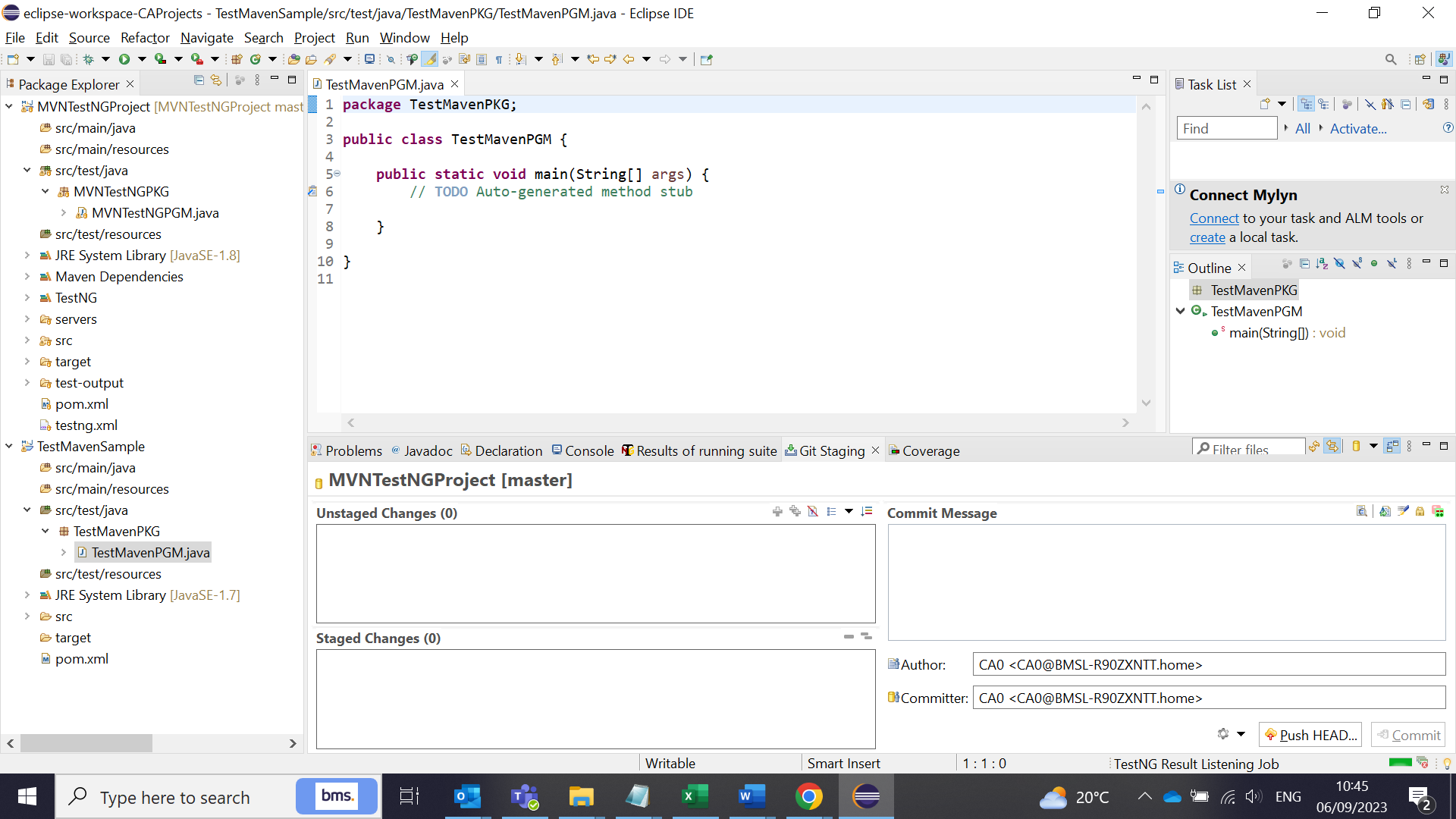
**How to Create a Maven Project:**

1. Create a Maven project using the New -> option (Maven)

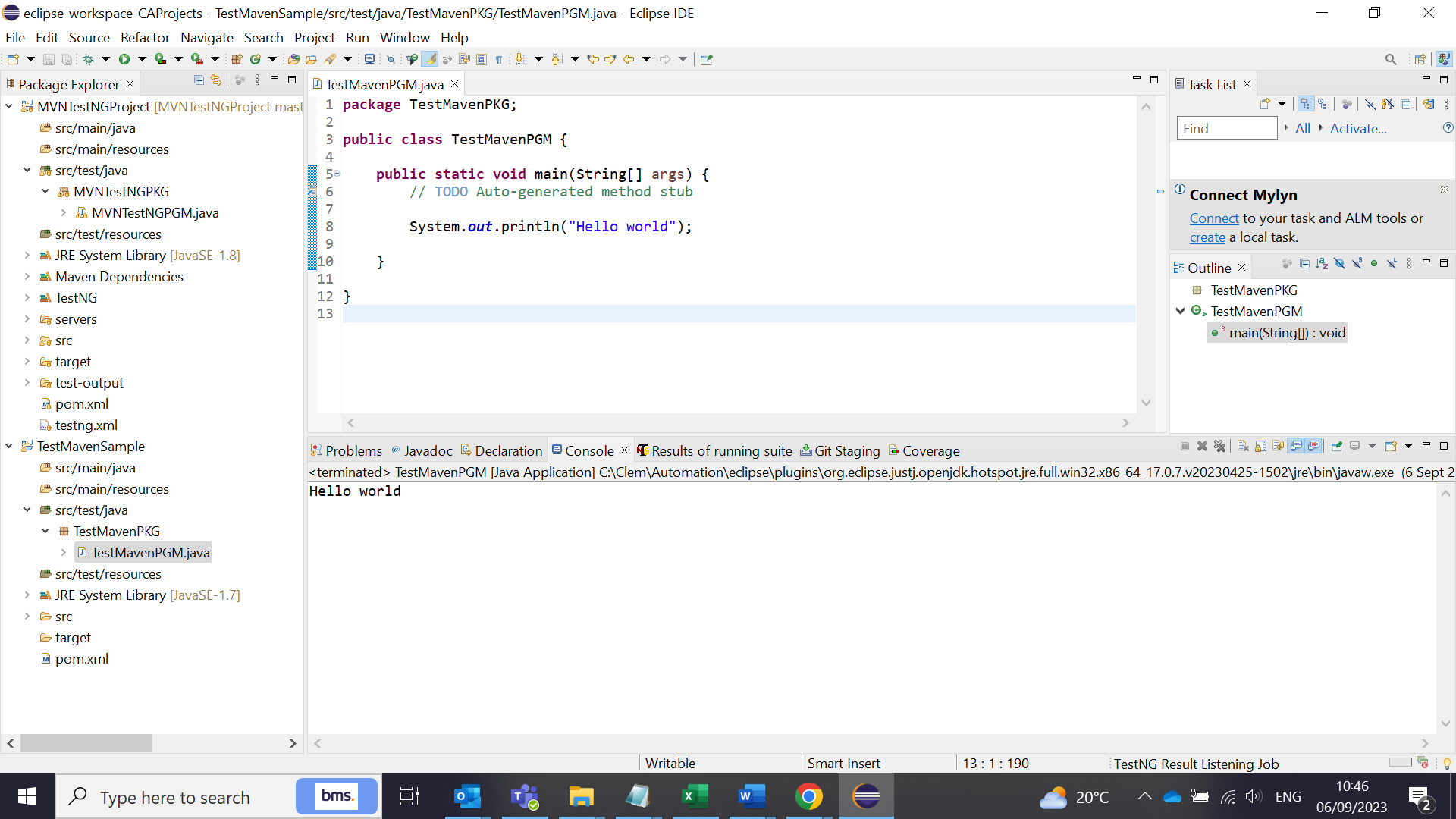


1. Create a class file after the Maven project folder structure created. All tests will be in the src/test/java folders.

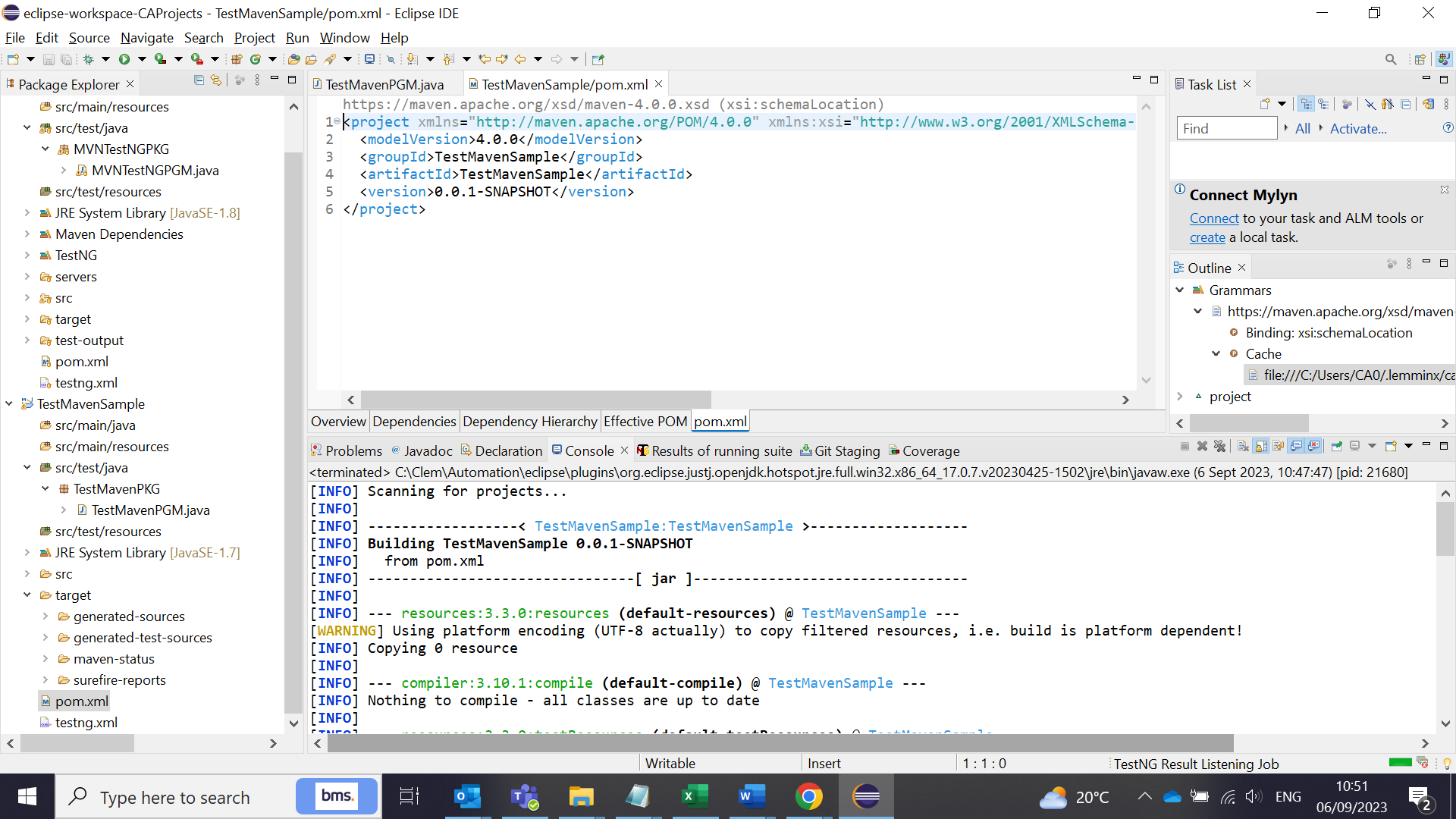




1. Add a line of code : hello world echo and Run as ‘Java application’

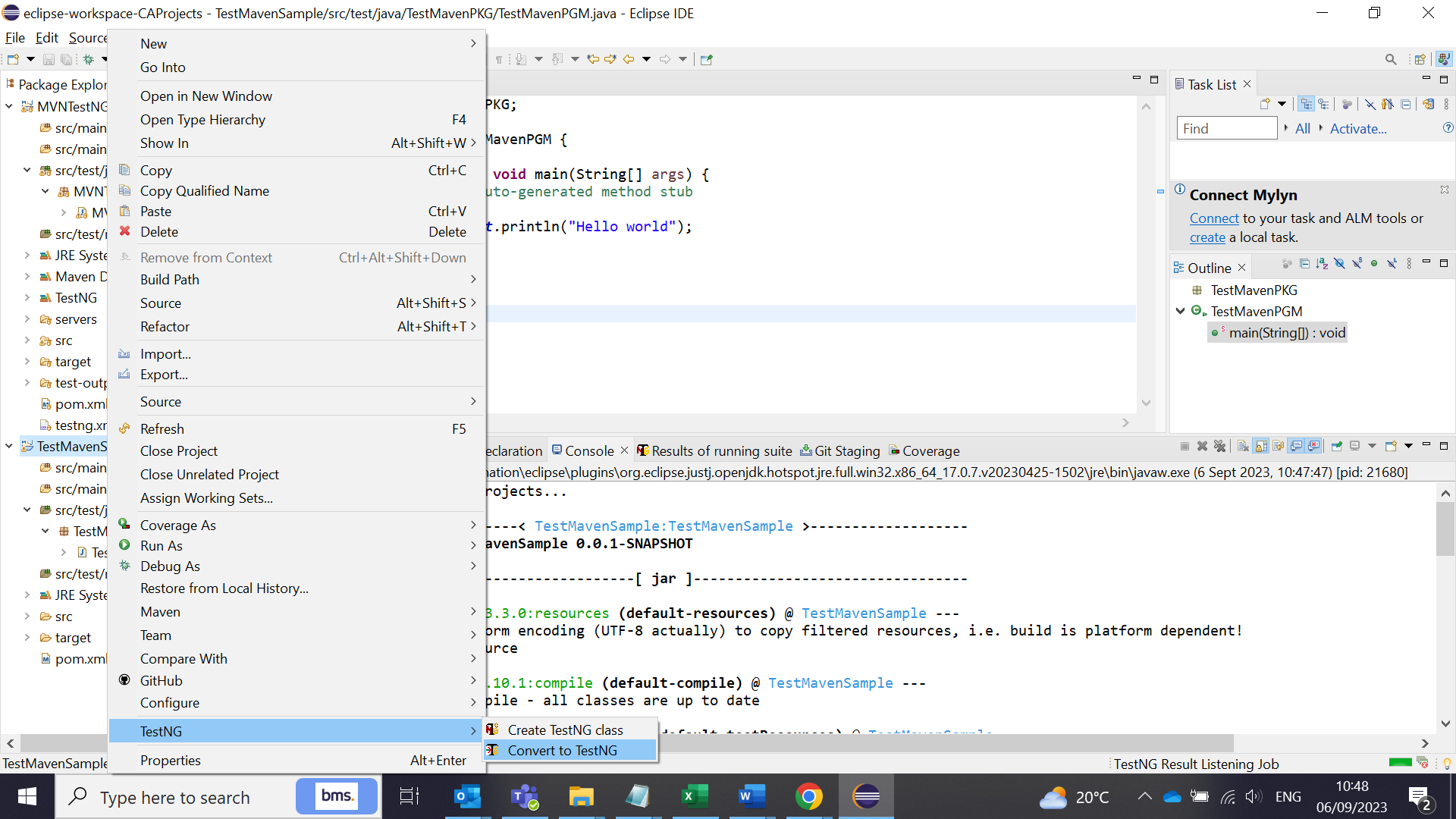


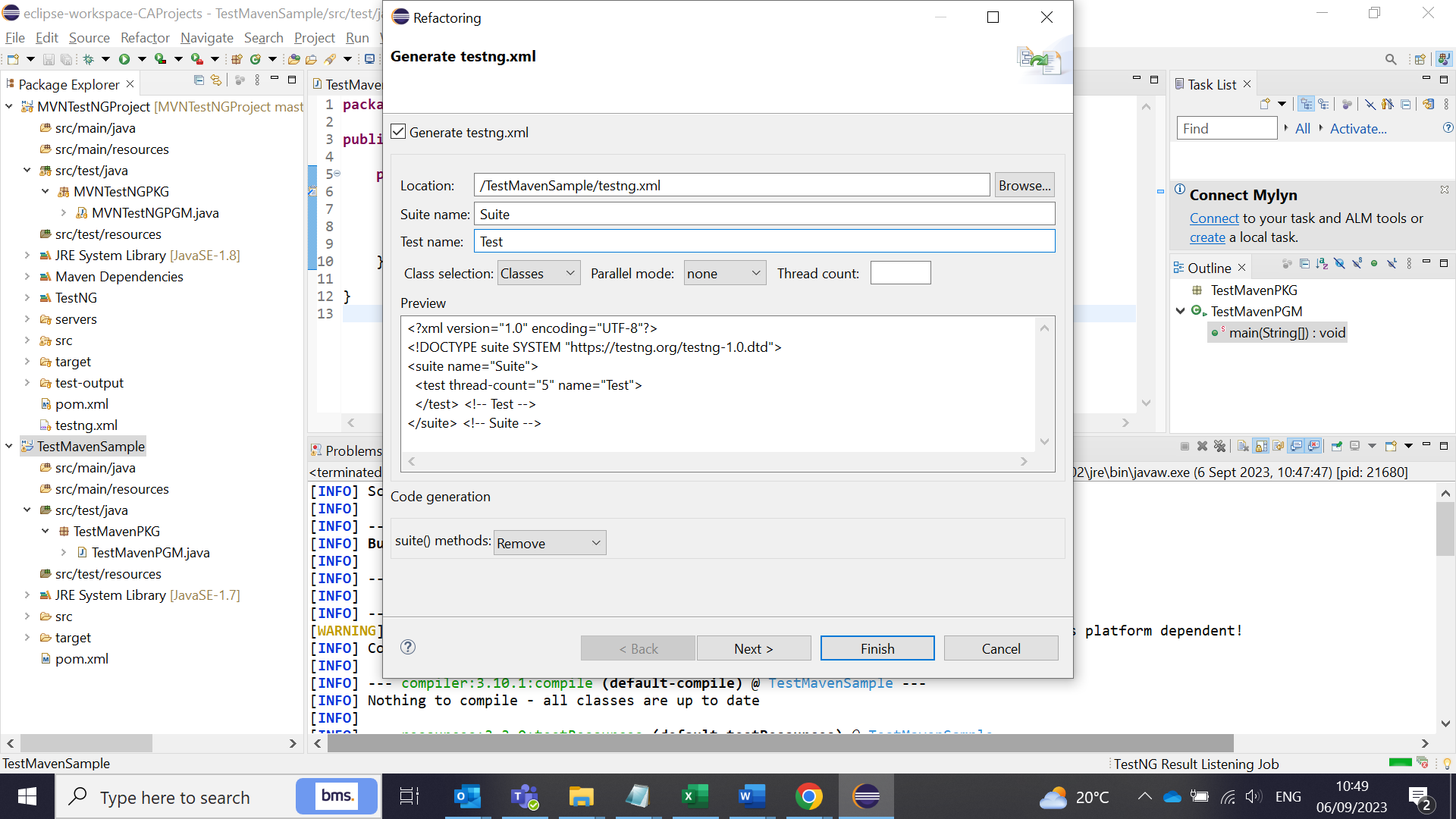
Pom.xml create:



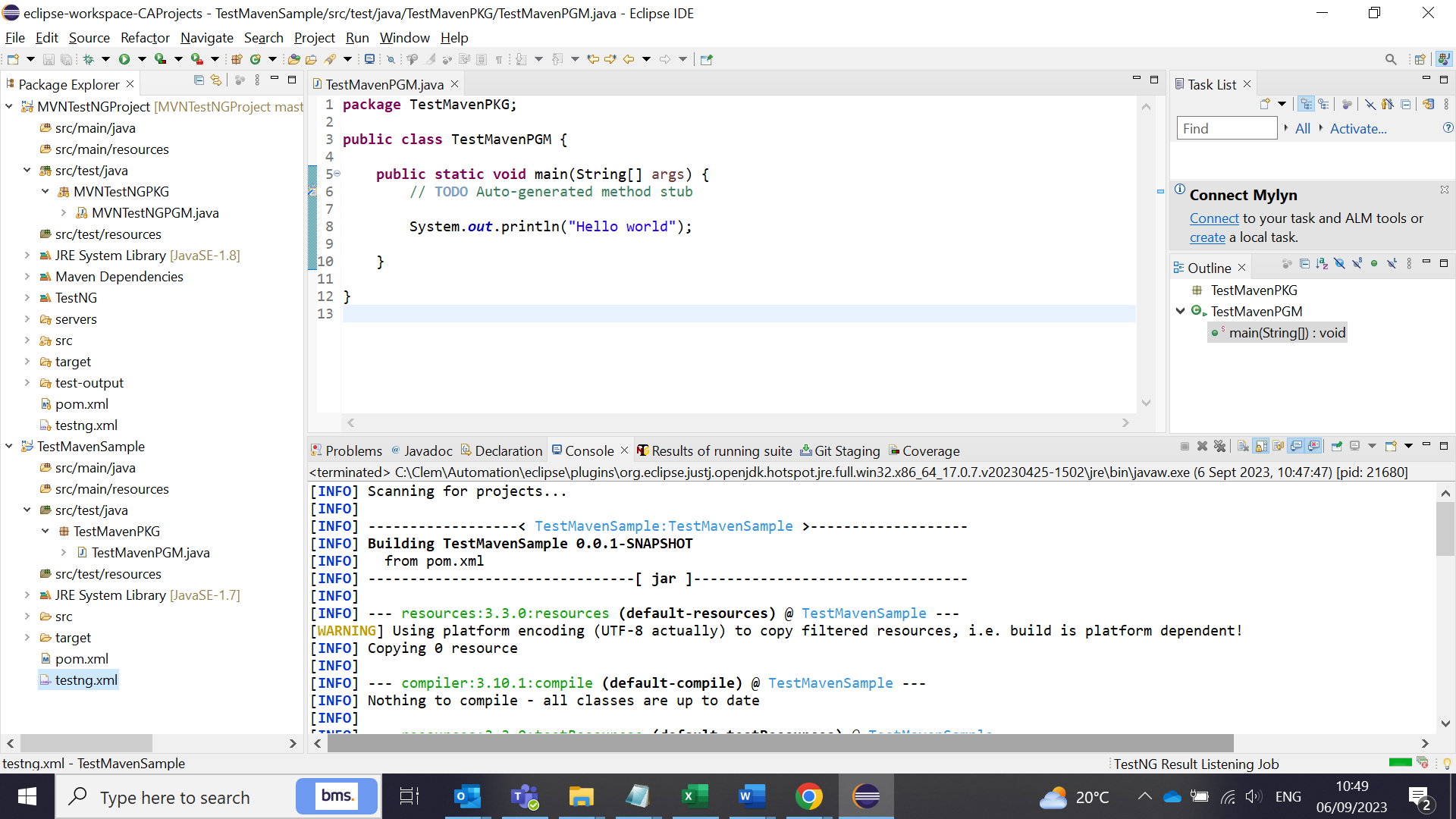
Maven Project is created now. A new pom.xml file created in the folder structure. Now, We need to convert this to a TestNG project First.

**To convert Maven project to a Test NG project:**

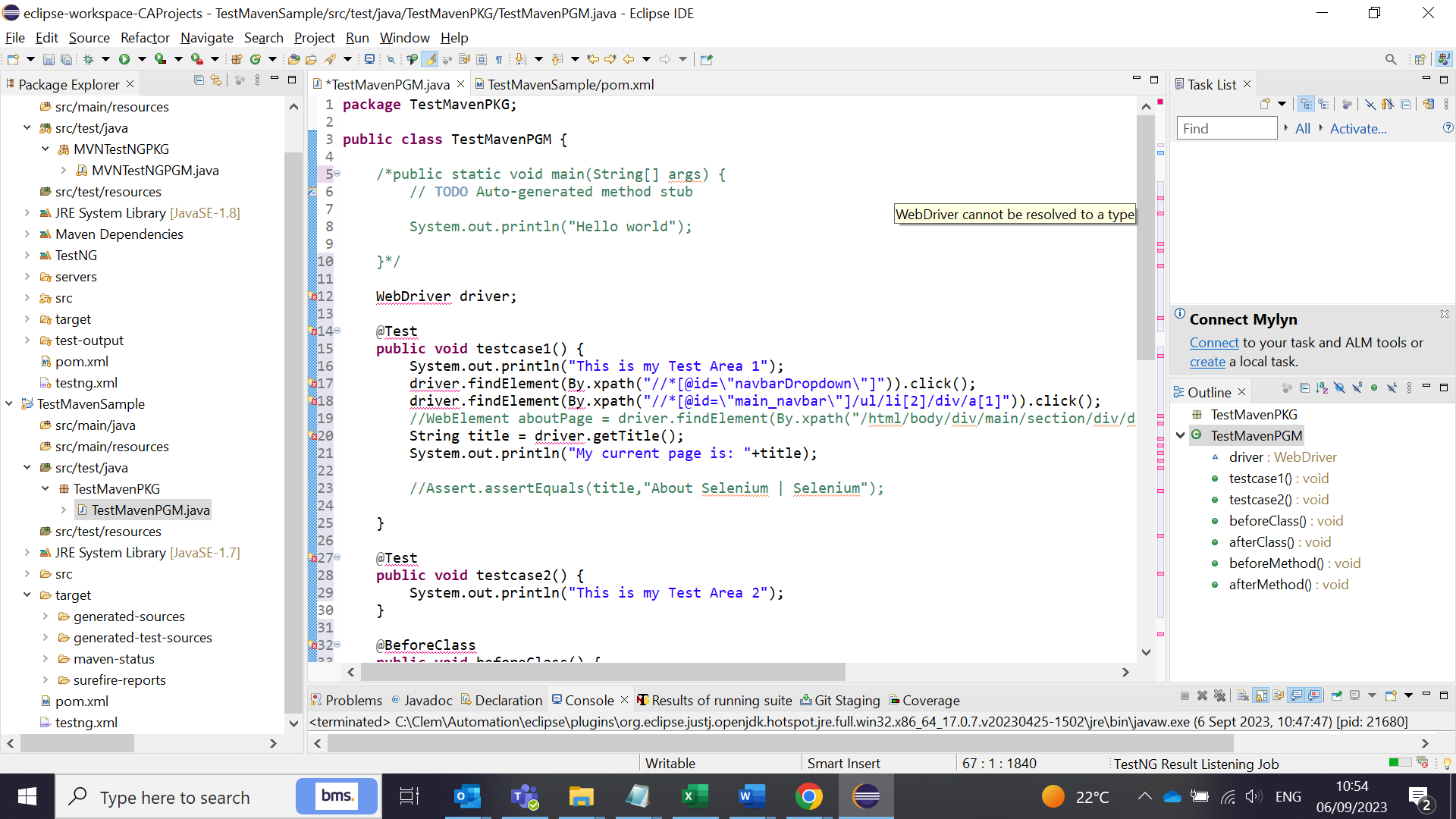




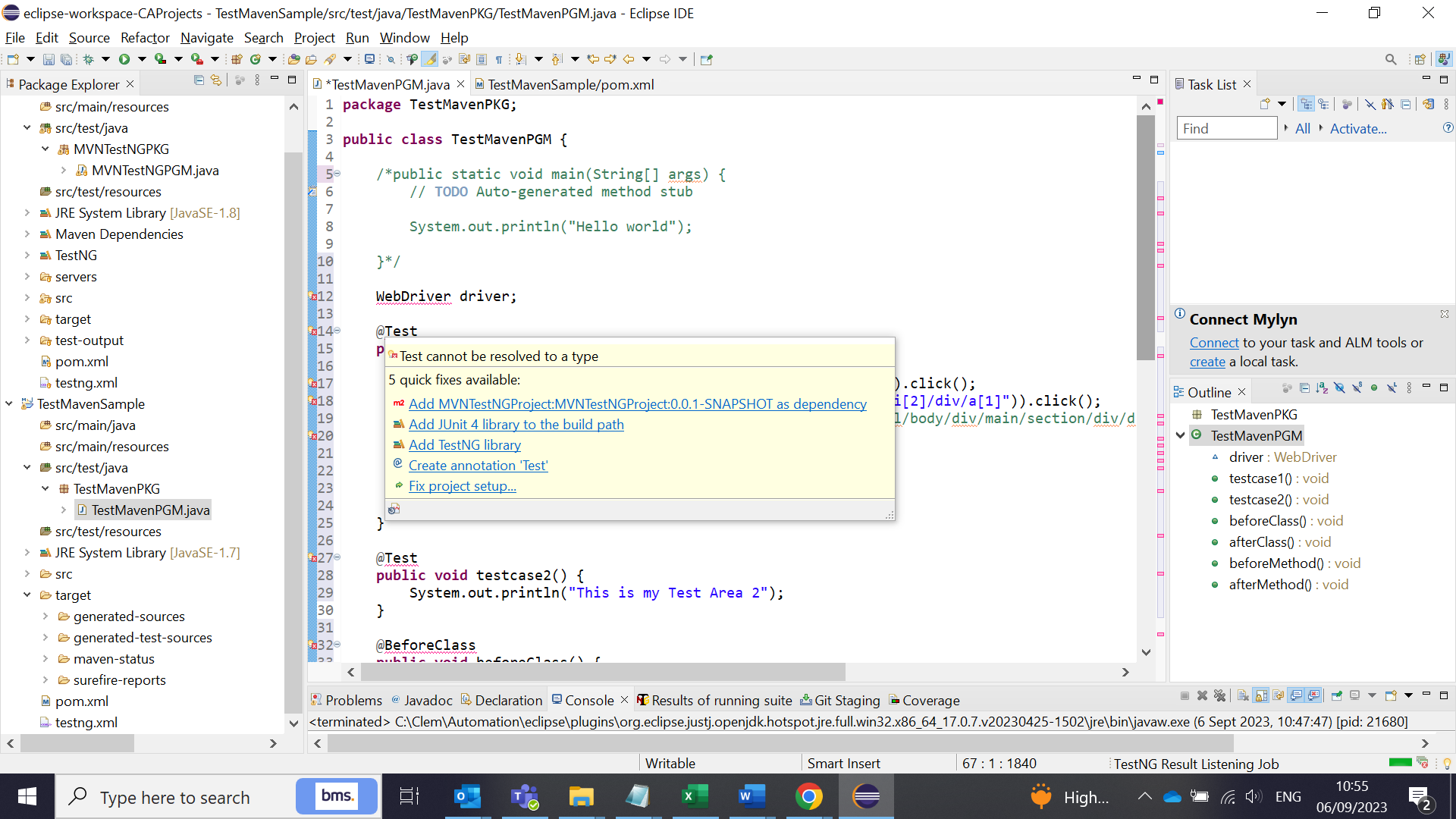
A new TestNG.xml file created in the folder structure:



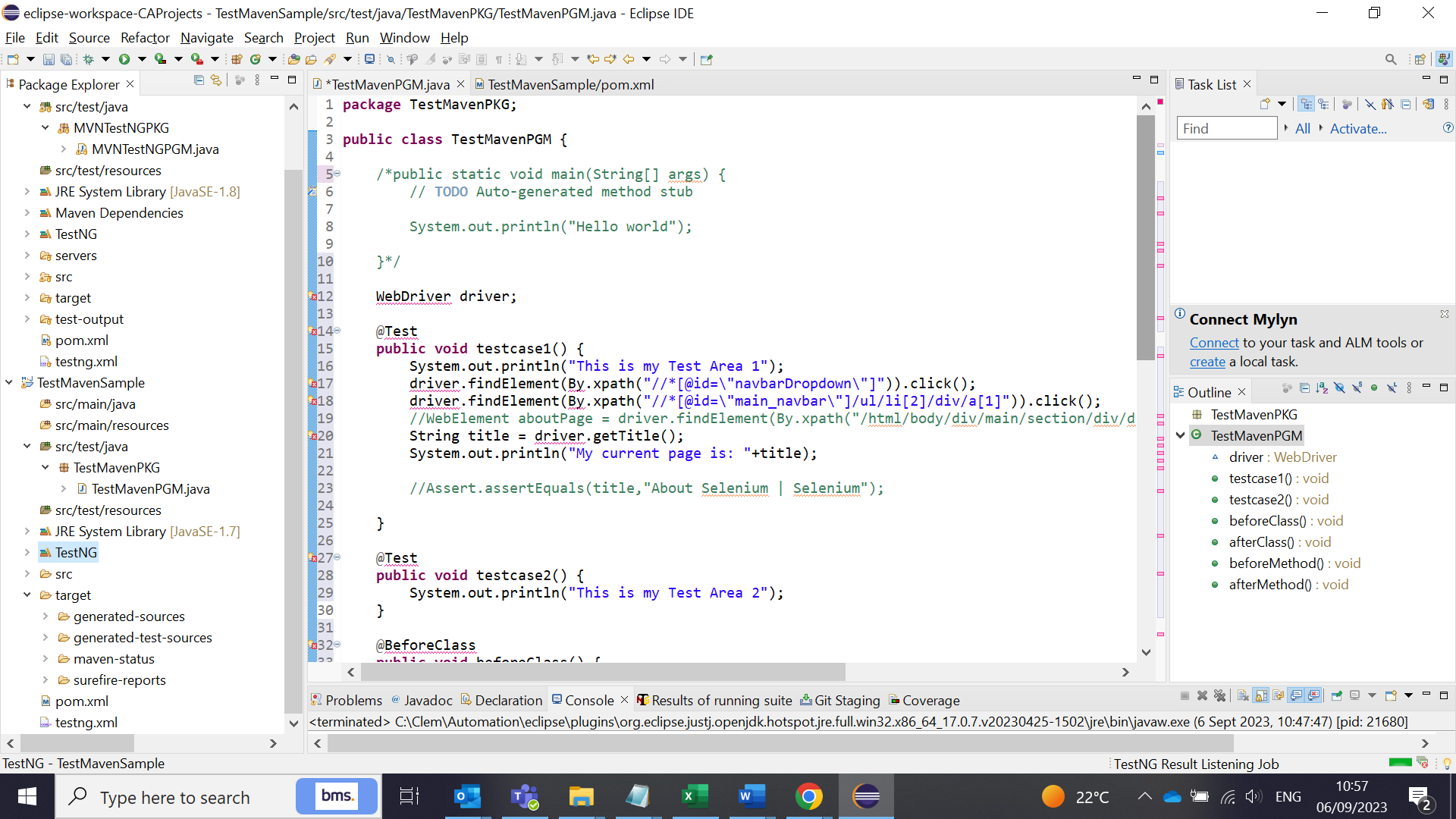
Now add the TestNG program code in the Java class as follows:



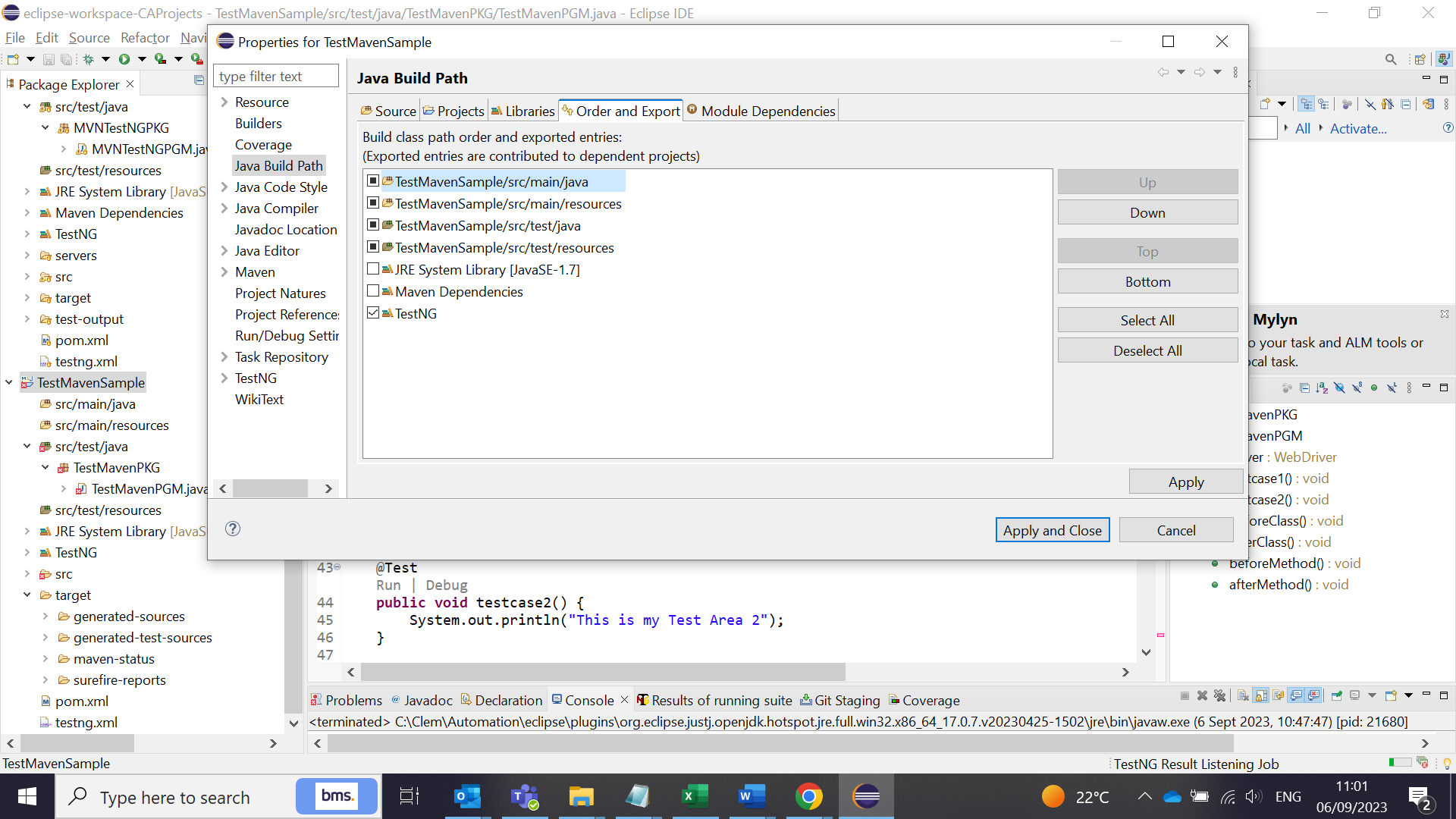
Now, we see lots of errors related to TestNG. So, we need to add the Test NG folder path to the project to overcome this. To do this just hoover over the Red error text such as @Test annotations, and add the TestNG library from the pop up suggestions.



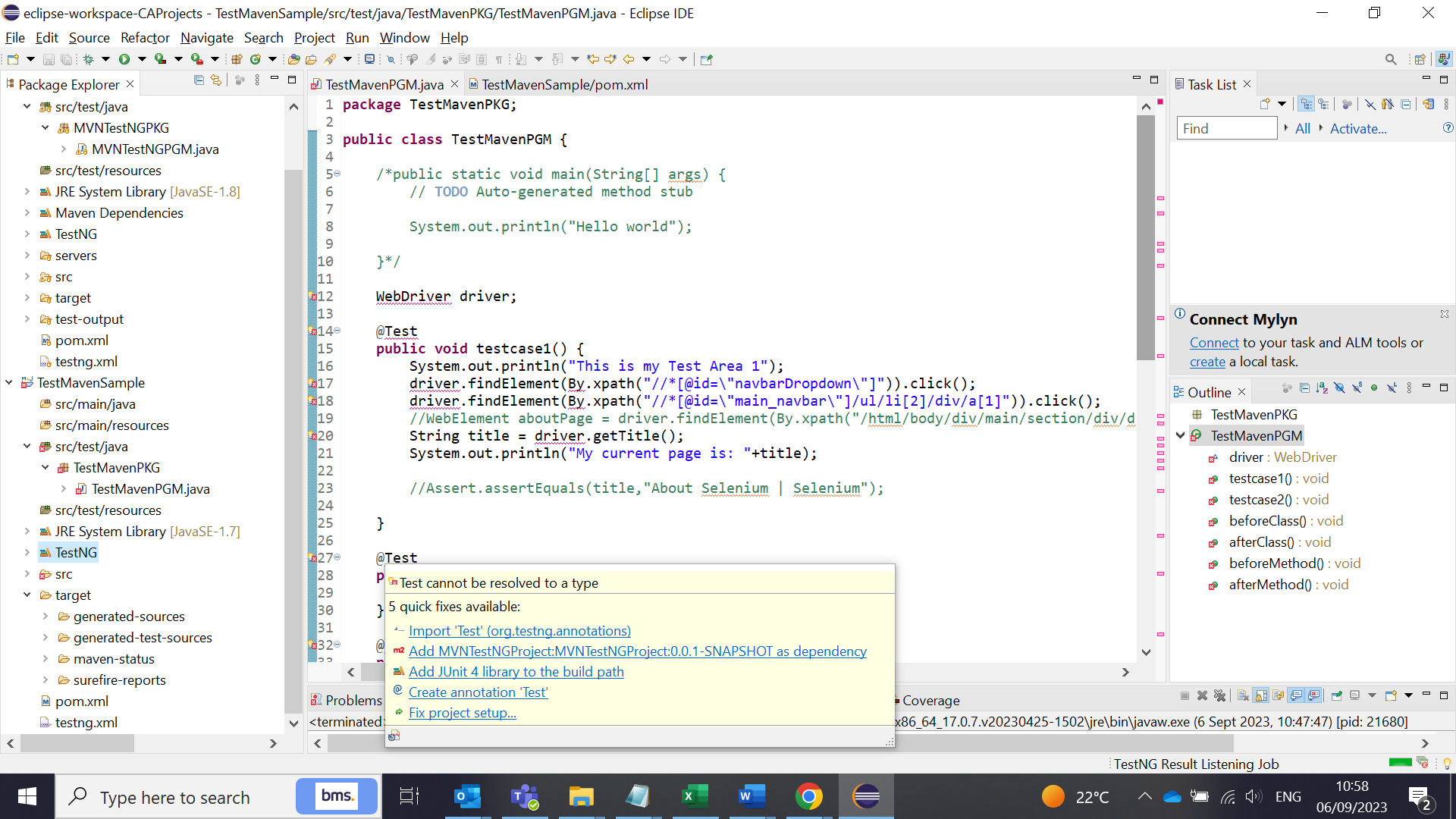
Now TestNG library has been added in the folder structure:



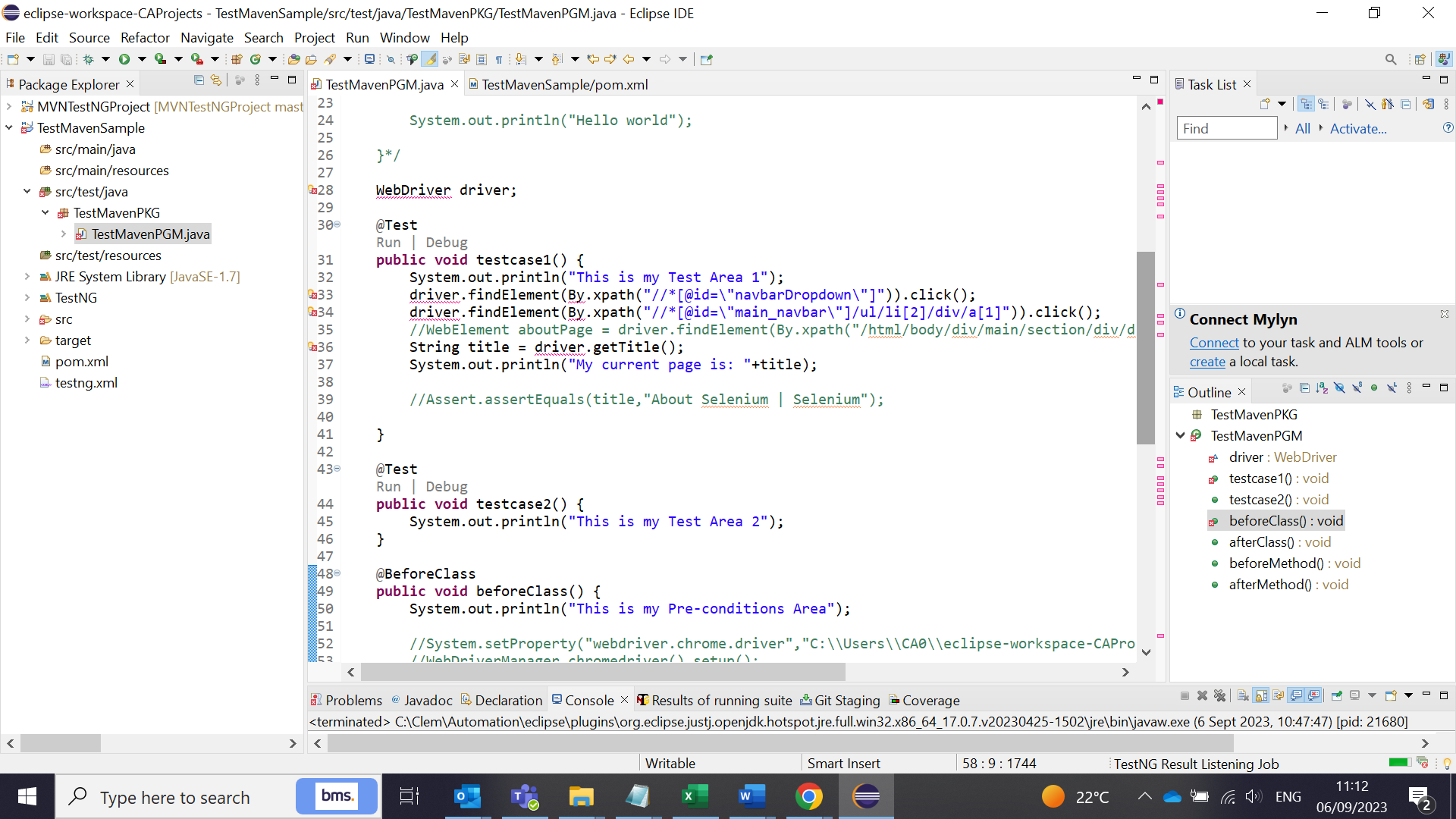
Now add the TestNG library in the build path as follows:



Still the errors are in the Red test for TestNG annotations. Now, we need to import the TestNG annotations packages in the program code.



Now the TestNG errors are gone. But the Selenium Driver errors exists.



There are two options to overcome the selenium issues.

1. Add the Selenium webdriver dependency (4.11.0) stable version to Maven pom.xml. This option is very feasible and easy to maintain the project dependencies.
2. Second download the selenium webdriver (4.11.0) jars from Maven Repository add the Selenium webdriver jars manually in the Reference path

Once you added the Maven dependency in the pom.xml file, there will be a Maven dependency folder created in the project folder structure. All dependencies added there. Now the Webdriver Red errors are gone.

<dependencies>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

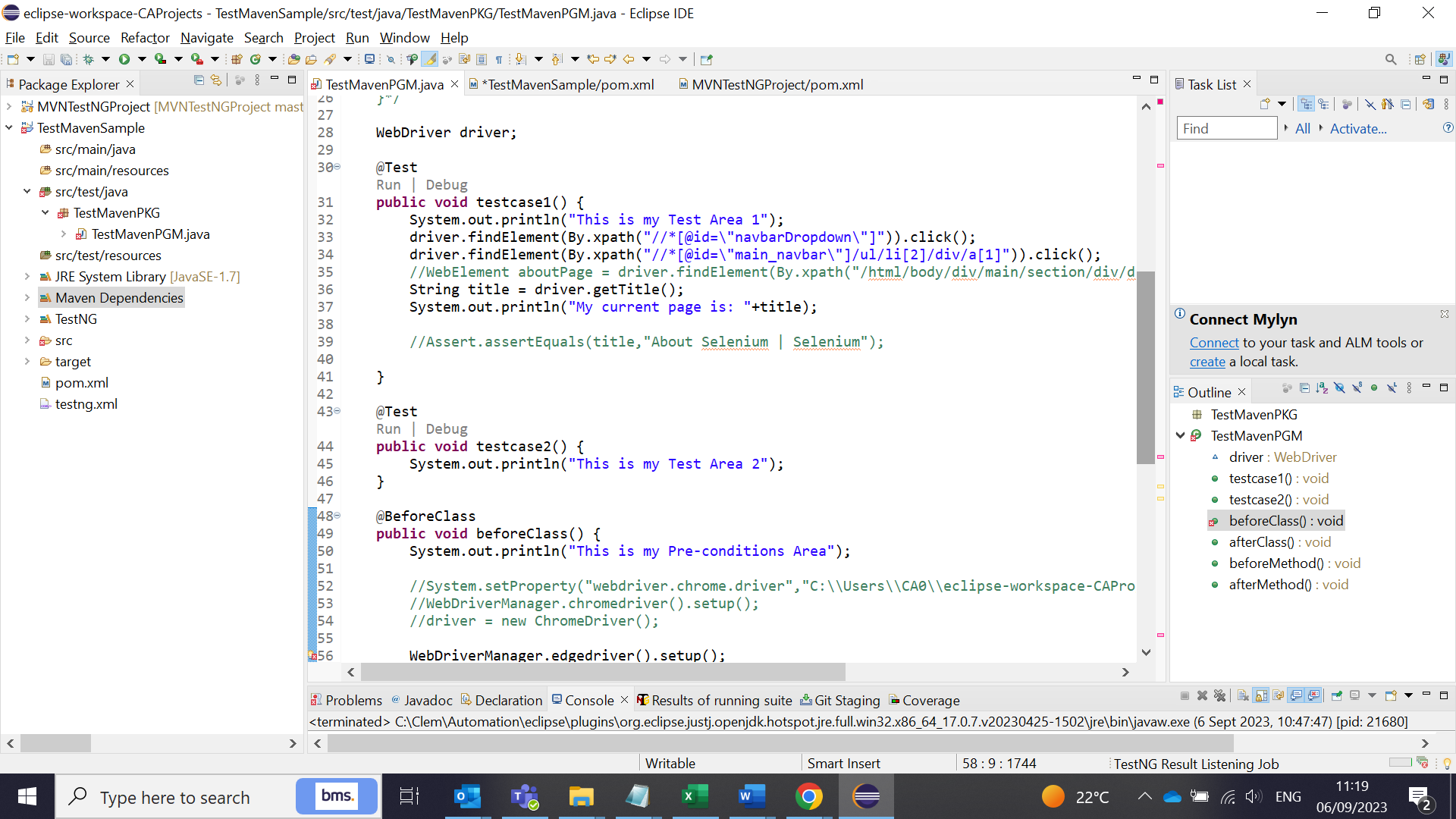
<artifactId>selenium-java</artifactId>

<version>4.11.0</version>

</dependency>

</dependencies>





Wedriver manager should be added to pom.xml. It will maintain all the browser exes and we do not need to specifically mention it in the code for the path and exe files for the browsers.

<!--

https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->

<dependency>

<groupId>io.github.bonigarcia</groupId>

<artifactId>webdrivermanager</artifactId>

<version>5.5.3</version>

</dependency>

Add all the dependencies to the POM>xml file and save it. Whenever you add the dependencies a new files will be added to the Maven dependency folder. To get the dependency code, go to Maven Repository and find the plugins and copy the code and paste it in the POM.xml file.

**Useful Dependencies are below:**

Add 2 SLFD mvn dependencies in Pom.xml to avoid Maven Run Logger errors.

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>2.0.7</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>2.0.7</version>

<scope>test</scope>

</dependency>

3)A line of code in selenium test : WebDriverManager.chromedriver().setup(); sets the browser drivers automatically. We need add the webdriver manager in POM.xml file. But the latest pom driver version not supporting the chrome 116 version currently.

<!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->

<dependency>

<groupId>io.github.bonigarcia</groupId>

<artifactId>webdrivermanager</artifactId>

<version>5.5.3</version>

</dependency>

4) To run all the TestNG test cases created in the TestNG.xml, we need to add the below "maven-surefire-plugin" from the usuage link to the pom.xml : https://maven.apache.org/surefire/maven-surefire-plugin/usage.html

<build>

<pluginManagement>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.1.2</version>

</plugin>

</plugins>

</pluginManagement>

</build>

5)After adding the Step 4, add the configuration(after version tag) in the code in pom.xml file as follows:

after <version>3.1.2</version>

add this <configuration>

<suiteXmlFiles>

<suiteXmlFile>testng.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

**To update the Testng.xml file:**

Add the <Package. Class file name> in the TestNG.xml file to run the project.

<classes>

<class name="TestMavenPKG.TestMavenPGM" />

</classes>

It will look like this:

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name="Suite">

<test thread-count="5" name="Test">

<classes>

<class name="TestMavenPKG.TestMavenPGM" />

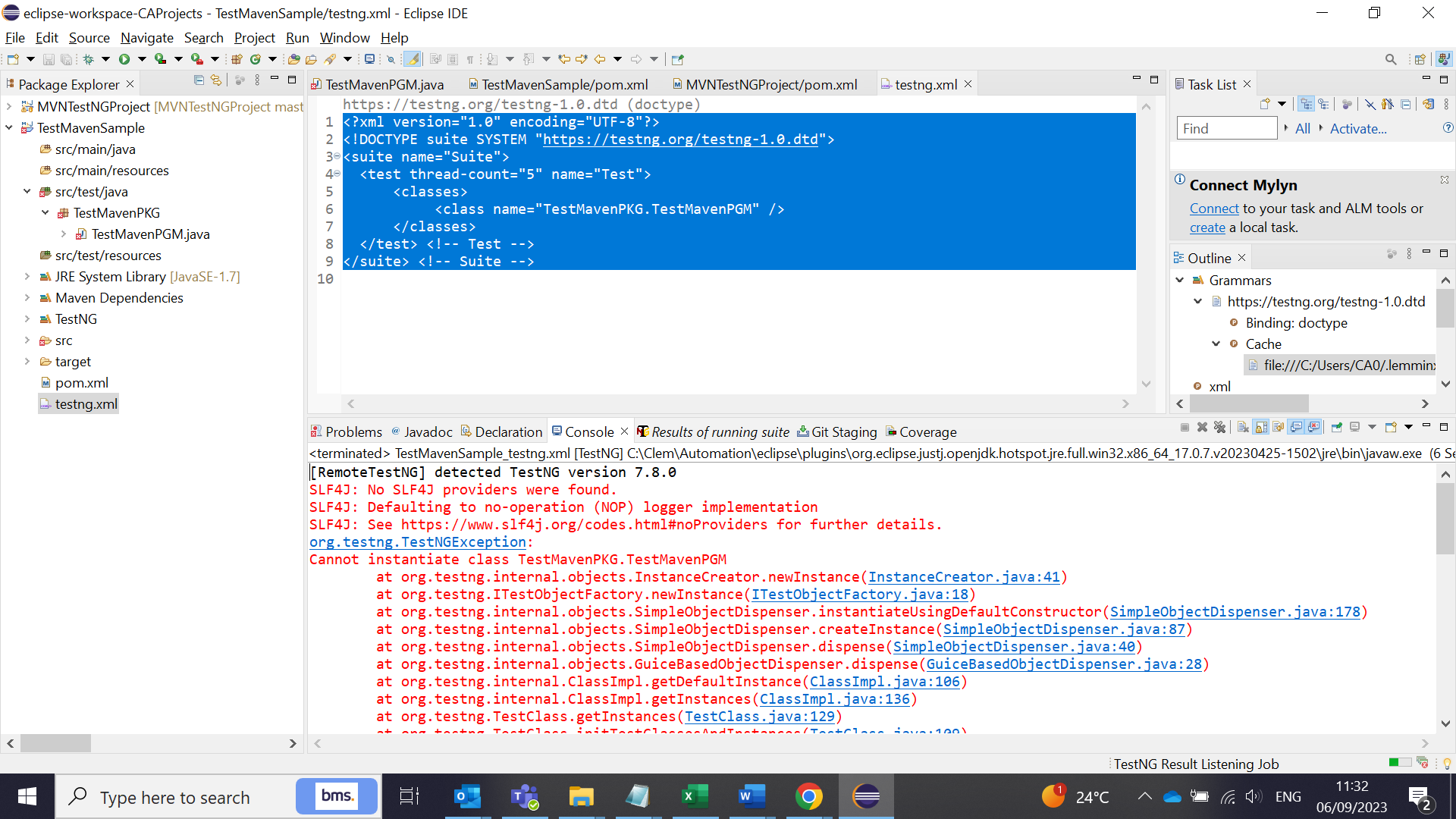
</classes>

</test> <!-- Test -->

</suite> <!-- Suite -->



When you do not have required dependencies in the pom.xml, you will get this errors:



To overcome this, add the 2 MVN dependencies as follows in the pom.xml:

<!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-api Logger api

file error correction 1-->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>2.0.7</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-simple Logger

api file error correction 2-->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>2.0.7</version>

<scope>test</scope>

</dependency>

Also add the TestNG version (6.14.3) which is stable for Java 1.8, java 1.7. Anything above 7.6 needs Java 11. So better to use the 6.14.3.

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

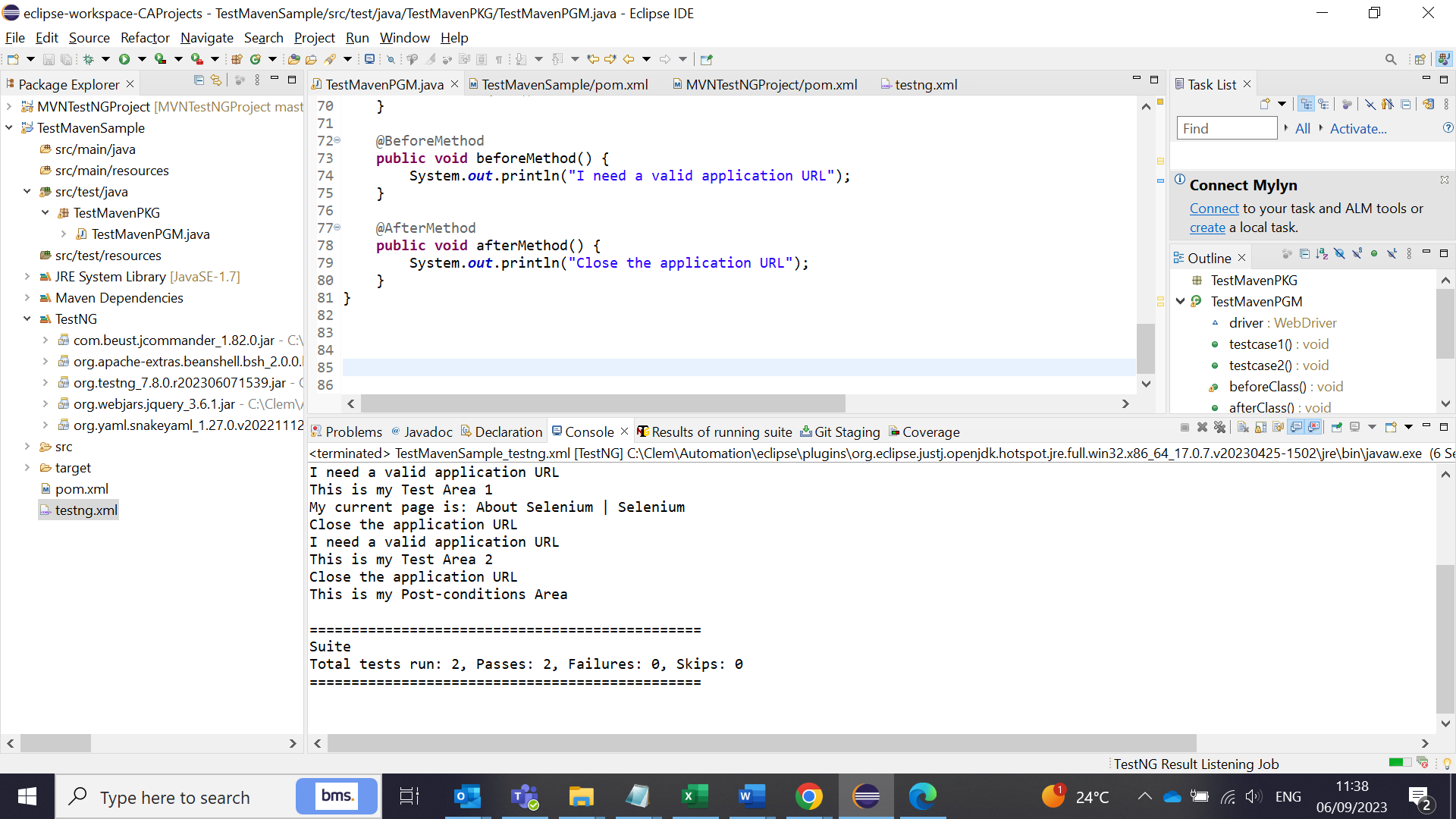
<artifactId>testng</artifactId>

<version>6.14.3</version>

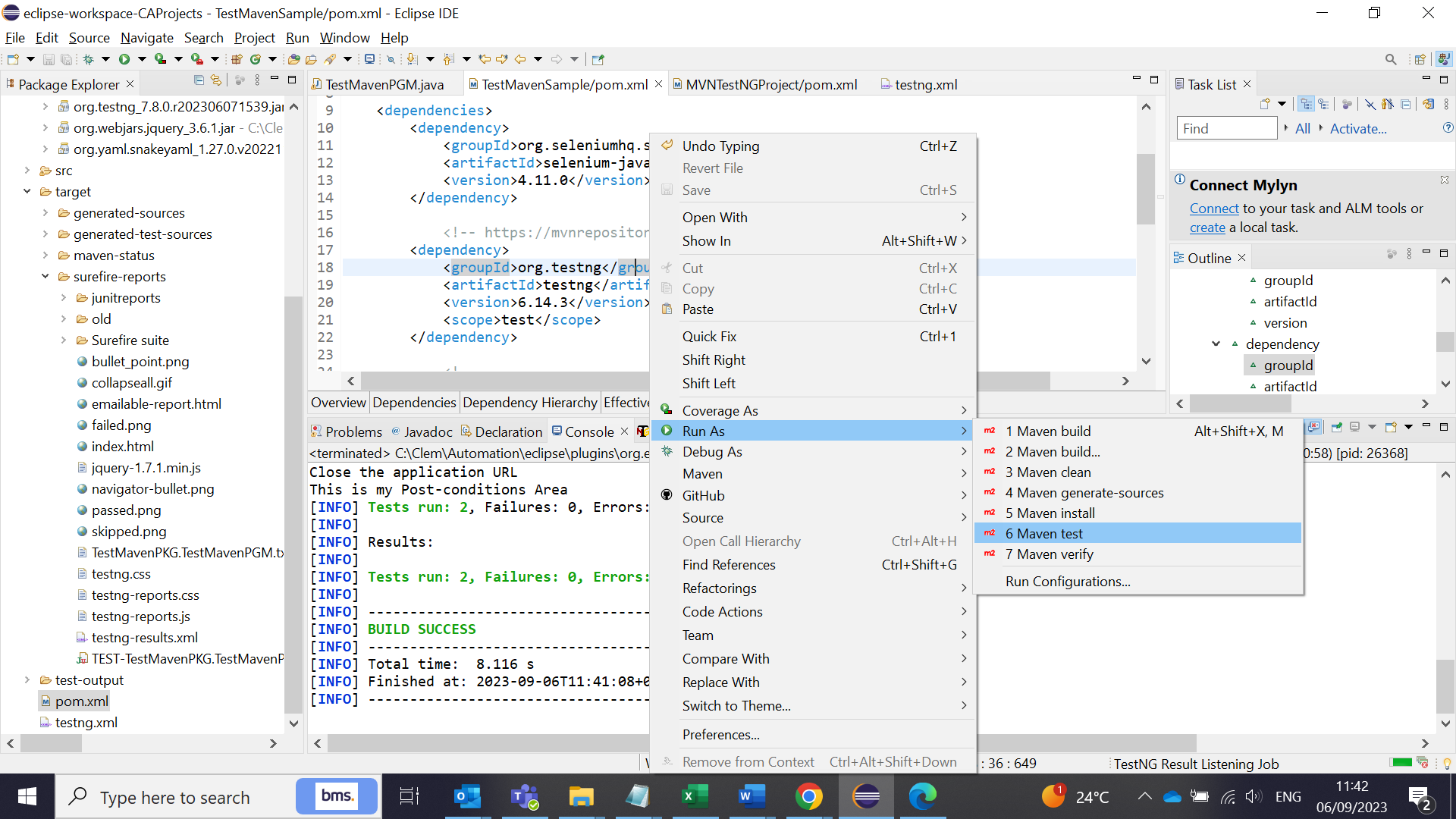
<scope>test</scope>

</dependency>

The code will run smoothy (TestNG run):



The code will run smoothy (Maven test):



Reports will be generated in target folder for Maven reports.target folder will have all the Maven reports (index.html). Right click show in - system explorer option to see in the browser.

Maven clean = will clean all the folders

Maven install = will clean and run the maven tests

Maven test = will clean, build and run the test.

test-output folder will have all the TestNG reports (index.html)

Add your tests based on your project needs by creating packages, class files, utils and so on. Add the required dependencies in the maven pom.xml file as and when required. Mostly the dependency versions might create problems for any errors. So, use the correct version which supports the platform and apps.

**Final POM XML-EdgeBrowser TestNG Maven**

<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>MVNTestNGProject</groupId>

<artifactId>MVNTestNGProject</artifactId>

<version>0.0.1-SNAPSHOT</version>

<build>

<pluginManagement>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.1.2</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>testng.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

</pluginManagement>

</build>

<properties>

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

</properties>

<dependencies>

<!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-api Logger api

file error correction 1-->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>2.0.7</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-simple Logger

api file error correction 2-->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>2.0.7</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.14.3</version>

<scope>test</scope>

</dependency>

<!--

https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>4.11.0</version>

</dependency>

<!--

https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->

<dependency>

<groupId>io.github.bonigarcia</groupId>

<artifactId>webdrivermanager</artifactId>

<version>5.5.3</version>

</dependency>

<!-- https://mvnrepository.com/artifact/junit/junit -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**Final Selenium TestNG Program Code - EdgeBrowser TestNG Maven**

package MVNTestNGPKG;

import org.testng.Assert;

import org.testng.annotations.AfterClass;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

import io.github.bonigarcia.wdm.WebDriverManager;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.edge.EdgeDriver;

public class MVNTestNGPGM {

/\*public static void main(String[] args) {

System.out.println("hello world");

}\*/

WebDriver driver;

@Test

public void testcase1() {

System.out.println("This is my Test Area 1");

driver.findElement(By.xpath("//\*[@id=\"navbarDropdown\"]")).click();

driver.findElement(By.xpath("//\*[@id=\"main\_navbar\"]/ul/li[2]/div/a[1]")).click();

//WebElement aboutPage = driver.findElement(By.xpath("/html/body/div/main/section/div/div/div/h1"));

String title = driver.getTitle();

System.out.println("My current page is: "+title);

Assert.assertEquals(title,"About Selenium | Selenium");

}

@Test

public void testcase2() {

System.out.println("This is my Test Area 2");

}

@BeforeClass

public void beforeClass() {

System.out.println("This is my Pre-conditions Area");

//System.setProperty("webdriver.chrome.driver","C:\\Users\\CA0\\eclipse-workspace-CAProjects\\MVNTestNGProject\\servers\\chromedriver.exe");

//WebDriverManager.chromedriver().setup();

//driver = new ChromeDriver();

WebDriverManager.edgedriver().setup();

driver = new EdgeDriver();

driver.manage().window().maximize();

driver.manage().deleteAllCookies();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("https://www.selenium.dev");

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

}

@AfterClass

public void afterClass() {

System.out.println("This is my Post-conditions Area");

//driver.quit();

}

@BeforeMethod

public void beforeMethod() {

System.out.println("I need a valid application URL");

}

@AfterMethod

public void afterMethod() {

System.out.println("Close the application URL");

}

}

**Final TestNG XML - EdgeBrowser TestNG Maven**

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name="Demo Test Suite">

<test thread-count="5" name="MVN TestNG Test Demo">

<classes>

<class name="MVNTestNGPKG.MVNTestNGPGM" />

</classes>

</test> <!-- Test -->

</suite> <!-- Suite -->

**Useful Links:**

Business Justification

-----------------------------

The listed tools are used to automate the test scripts we plan for the projects in BMS as part of our testing approach. The regression test suites will also be automated using these tools and we develop automation code development which would be a beneficial and a cost saving for the projects across BMS. Hence we need to install the software and tools in the system to develop automation code. It is also approved and installed in other resource’s system in the testing team.

Java 1.7 -

Eclipse IDE - Eclipse is a free open-source integrated development environment

TestNG - TestNG is a free and open source testing framework inspired from JUnit and NUnit

Selenium - open-source automated testing suite for web applications

Maven (These are open-source tools). - Maven is a popular open-source build tool developed by the Apache Group to build, publish, and deploy several projects at once for better project managemen

Notepad++

MVN repository

---------------------

https://www.youtube.com/watch?v=vjYRLa3JV8E

Install Java, Eclipse, Set MVN dependencies

------------------------------------------------------------

https://www.youtube.com/watch?v=QFHMzWqDPiA

https://mvnrepository.com/

https://www.eclipse.org/downloads/packages/release/2022-12/r

https://www.youtube.com/watch?v=vmwc\_TK07SU - File upload and download in selenium

https://www.youtube.com/watch?v=nOsimnAzSng - Junit test case creation

https://googlechromelabs.github.io/chrome-for-testing/#stable - New Chrome drivers downloads

DevOps pipeline for selenium tests (build pipeline)

-------------------------------------------------------------------

https://www.youtube.com/watch?v=5S7ev-GMyOM - Good one full CI/CD, Git Hub commands

Maven & Automation (good one)

---------------------------------------------

https://www.youtube.com/watch?v=-25Q76lnB50 - Part 1

https://www.youtube.com/watch?v=cTzvrCm9P9E - Part 2

https://www.youtube.com/watch?v=UNB-bCcdSjE - Convert Java project to Maven in Eclipse

https://www.youtube.com/watch?v=iNPE7DwpxIU - How to Create First TestNG Class using Maven Project | Selenium Framework using Java | SDET

https://www.youtube.com/watch?v=BlI2NGarmvY - Part 3 | Apache Maven Tutorial | Selenium with maven Project | Maven Surefire Plugin | Maven Report

channel good one - Suresh SDET Automation

Git hub - eclipse project commit & push

---------------------------------------------------------

https://www.youtube.com/watch?v=LPT7v69guVY - Good

https://github.com/naveenanimation20 - All codes download

https://www.youtube.com/watch?v=gO20QGT6aW8 - GIT Push error fix (Suresh SDET Automation - channel)

https://www.youtube.com/watch?v=mM4ZJ2iXenE -

https://www.youtube.com/watch?v=5S7ev-GMyOM

https://www.youtube.com/watch?v=\_XQm7TyZBYA

(Eclipse IDE for Java Developers)

Java JDK above 8 is always good. Use 11 or 20

Test NG - 6.14.2 is the stable version. Get from MVN repository

Slenium - Selenium-java - 3.141.59 good version