Apache framework

1. **Apache POI** is an open-source library developed and distributed by Apache Foundation. Moreover, it is mainly used to create, read, and edit **Microsoft Office** files, majorly Excel files in Java programs. Moreover, it is distributed as a JAR, which provides various methods to manipulate Microsoft Excel files.

How to download:

1. First, navigate to the **Apache POI** webpage. After that, click on the **Download** link in the left menu. Moreover, it is as highlighted below:

2. Secondly, clicking on the Download link will navigate to the page showing the latest release of **Apache POI**. Additionally, it is as highlighted below:

```
find . -name "*.sha512" -type f -execdir sha512sum -c {} \;
find . -name "*.asc" -exec gpg --no-secmem-warning --verify {} \;
```

Sample verification of poi-bin-3.5-FINAL-20090928.tgz

```
% gpg --import KEYS
gpg: key 12DAE9BE: "Glen Stampoultzis <glens at apache dot org>" not changed
gpg: key 4CEED75F: "Nick Burch <nick at gagravarr dot org>" not changed
gpg: key 84B5A42E: "Rainer Klute <rainer.klute at gmx dot de>" not changed
gpg: key F5BB52CD: "Yegor Kozlov <yegor.kozlov at gmail dot com>" not changed
gpg: Total number processed: 4
gpg:               unchanged: 4
% gpg --verify poi-bin-3.5-FINAL-20090928.tgz.asc poi-bin-3.5-FINAL-20090928.tgz
gpg: Signature made Mon Sep 28 10:28:25 2009 PDT using DSA key ID F5BB52CD
gpg: Good signature from "Yegor Kozlov <yegor.kozlov at gmail dot com>"
gpg:                  aka "Yegor Kozlov <yegor at dinom dot ru>"
gpg:                  aka "Yegor Kozlov <yegor at apache dot org>"
Primary key fingerprint: 7D77 0C77 6CE7 754E E6AF  23AA 6934 0A02 F5BB 52CD
% gpg --fingerprint F5BB52CD
pub   1024D/F5BB52CD 2007-06-18 [expires: 2012-06-16]
      Key fingerprint = 7D77 0C77 6CE7 754E E6AF  23AA 6934 0A02 F5BB 52CD
uid                  Yegor Kozlov <yegor.kozlov at gmail dot com>
uid                  Yegor Kozlov <yegor at dinom dot ru>
uid                  Yegor Kozlov <yegor at apache dot org>
sub   4096g/7B45A98A 2007-06-18 [expires: 2012-06-16]
```

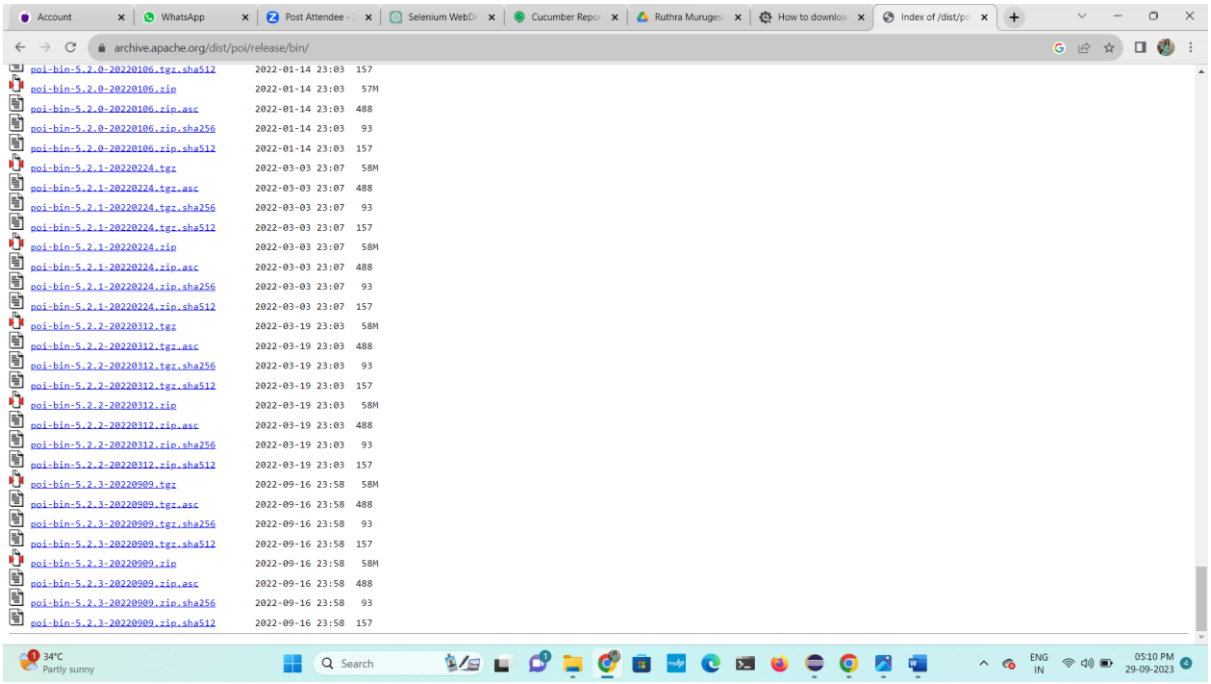
Release Archives

Apache POI became a top level project in June 2007 and POI 3.0 artifacts were re-released. Prior to that date POI was a sub-project of Apache Jakarta.

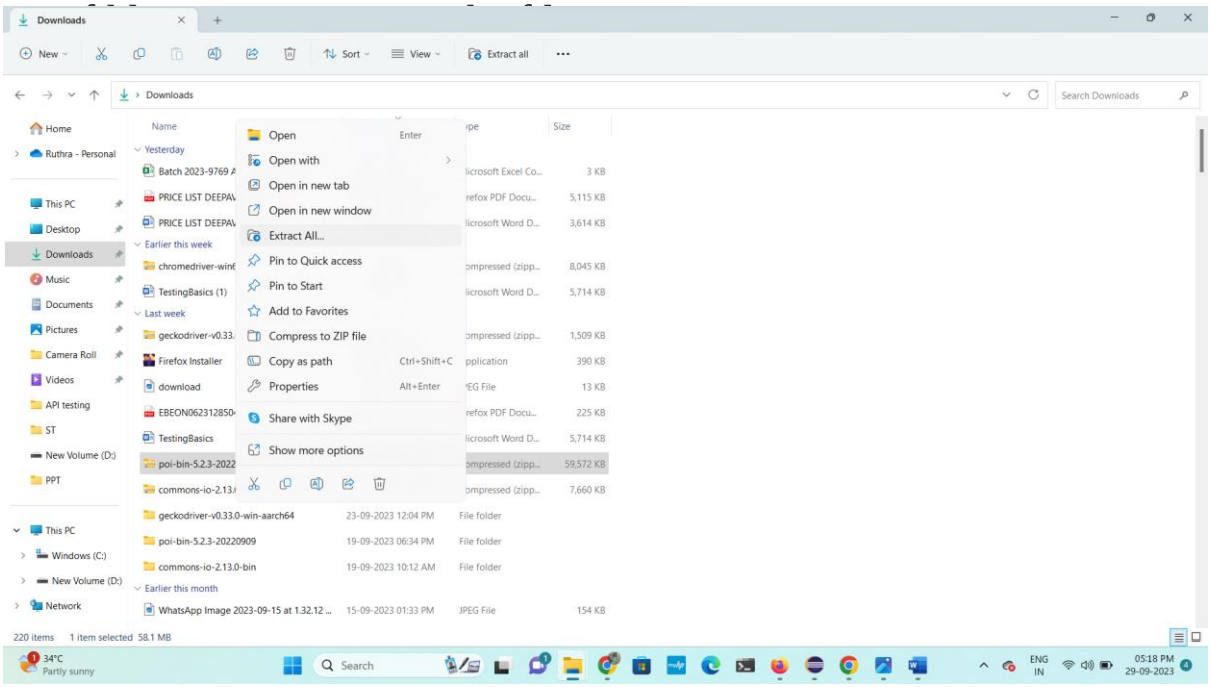
- Source Artifacts
- Binary Artifacts
- Artifacts from prior to 3.0

23 The Apache Software Foundation — Last Published: 09/29/2023 05:19:05

3. Thirdly, you can either click on the **"Latest Stable Release Link "** (as shown by marker 1), which will scroll the page down to the binaries of Apache POI (as shown by marker 2), or can directly scroll down to the section of binaries shown by marker 2. Subsequently, after clicking on the **"zip "** file, it will navigate to the page showing various download links as shown below:

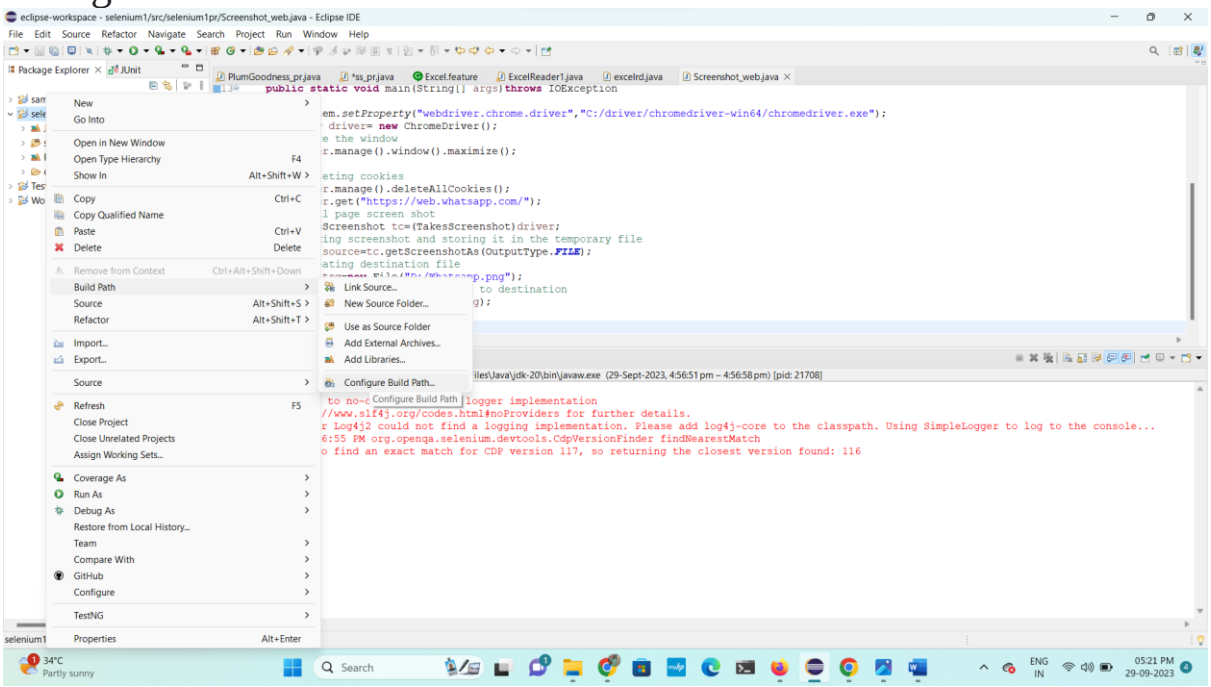


Above diagram click [poi-bin-5.2.3-20220909.tgz](poi-bin-5.2.3-20220909.tgz) the link.

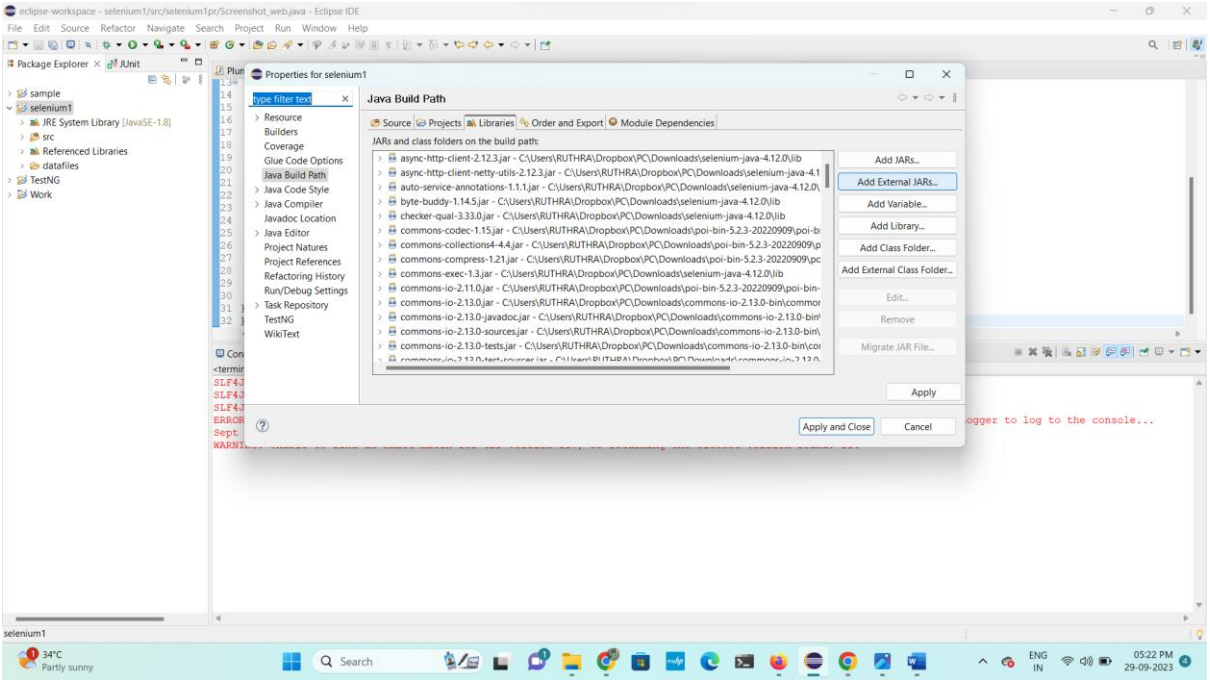


Eclipse configuration

Create a package or already create selenium package we will configure



Click Configure Build Path➔ go to libraries➔ add external jar files.

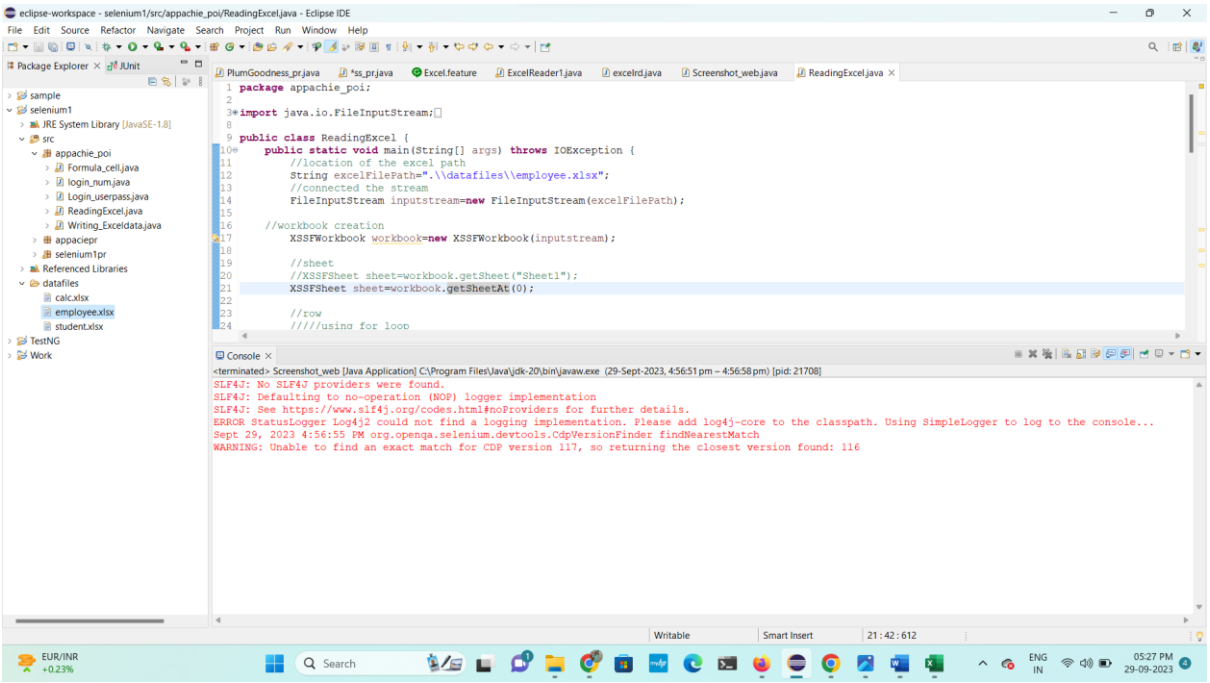


Add Apache jar files ➔ apply and close

2.create class:

We are creating external Excel files to read the data

Inside we are create one folder and add the excel data:




```
package appachie_poi;
```

```
import java.io.FileInputStream;

import java.io.IOException;

import java.util.Iterator;

import org.apache.poi.xssf.usermodel.*;

public class ReadingExcel {

	public static void main(String[] args) throws IOException {

		//location of the excel path

		String excelFilePath=".\\datafiles\\employee.xlsx";

		//connected the stream

		FileInputStream inputstream=new
FileInputStream(excelFilePath);

	//workbook creation

		XSSFWorkbook workbook=new
XSSFWorkbook(inputstream);



		//sheet

		//XSSFSheet sheet=workbook.getSheet("Sheet1");

		XSSFSheet sheet=workbook.getSheetAt(0);



		//row

		/////using for loop



	/*	int rows=sheet.getLastRowNum();

		//coloumn
```

```
int cols=sheet.getRow(1).getLastCellNum();


for(int r=0;r<rows;r++)//outer for loop
{
    XSSFRow row=sheet.getRow(r);


    for(int c=0;c<cols;c++) {


        XSSFCell cell=row.getCell(c);


        switch(cell.getCellType())//type of cell
        {
            case STRING:
System.out.print(cell.getStringCellValue());break;
            case
NUMERIC:System.out.print(cell.getNumericCellValue());break;
            case BOOLEAN:
System.out.print(cell.getBooleanCellValue());break;
        }
        System.out.print(" | ");
    }
    System.out.println();
}*/
////////// Iterator ///////////////
```

```
Iterator iterator=sheet.iterator();

//we have read the all columns and  rows we are using
the iterator method.

while(iterator.hasNext()) {// has next will check the
particular object there are not



XSSFRow row=(XSSFRow)
iterator.next();//capture each individual value

//this row contain the multiple cell



Iterator cellIterator=row.cellIterator();

while(cellIterator.hasNext()) {



XSSFCell cell= (XSSFCell)
cellIterator.next();



switch(cell.getCellType())//type of cell

{

case STRING:
System.out.print(cell.getStringCellValue());break;

case
NUMERIC:System.out.print(cell.getNumericCellValue());break;

case BOOLEAN:
System.out.print(cell.getBooleanCellValue());break;

}
```

```
                System.out.print(" | ");
            }
            System.out.println();
            }
}
}
```

Output:

```
Console ×
<terminated> ReadingExcel [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (29-Sept-2023, 5:27:26 pm – 5:27:27 pm) [pid: 17760]
ERROR StatusLogger Log4j2 could not find a logging implementation. Please add log4j-core to the classpath. Using SimpleLogger to log to the console...
Employee name | Employee state | salary |
Ruthra | Tamilnadu | 52000.0 |
Selva | Kerala | 56000.0 |
Shanthini | pandicherry | 53000.0 |
Magesh | karataka | 65000.0 |
Lithish | Manipur | 62000.0 |
Kajol | Pujab | 73000.0 |
Swetha | Odisha | 43000.0 |
Aishu | GOA | 54000.0 |
```

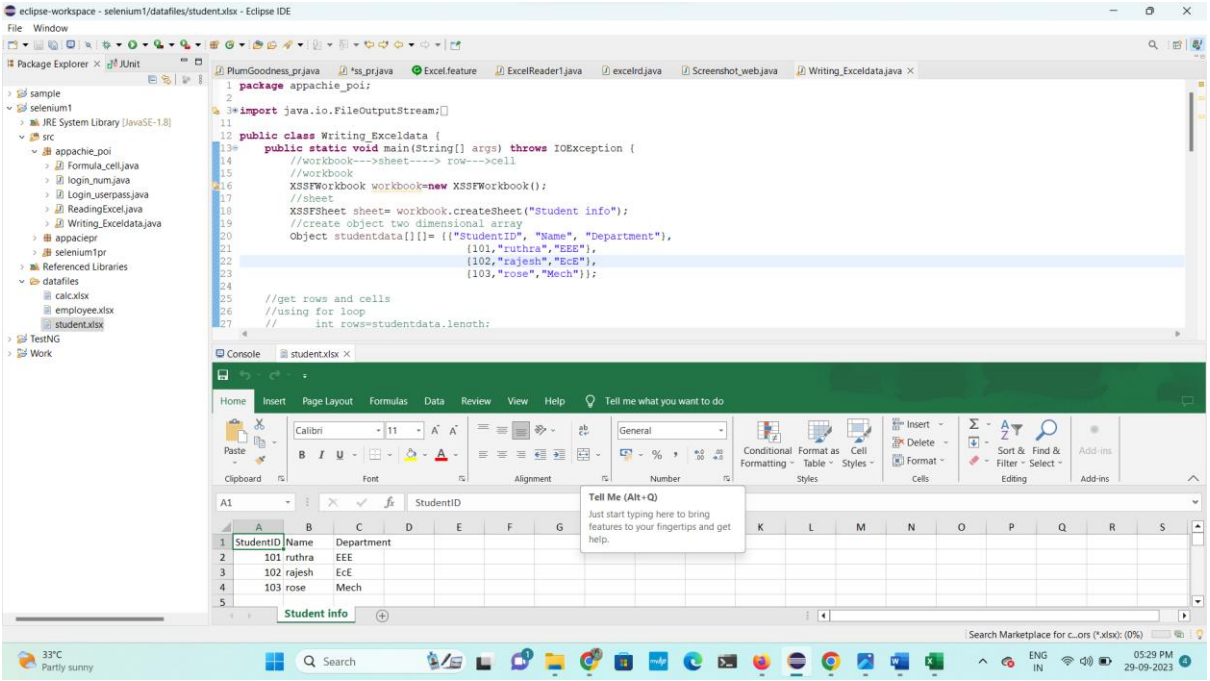
Write the file:

```
package appachie_poi;

import java.io.FileOutputStream;
import java.io.IOException;
import java.util.EventObject;


import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.apache.poi.xssf.usermodel.*;

public class Writing_Exceldata {
    public static void main(String[] args) throws IOException {
        //workbook--->sheet----> row--->cell
        //workbook
        XSSFWorkbook workbook=new XSSFWorkbook();
```

```
		//sheet
		XSSFSheet sheet= workbook.createSheet("Student
info");
		//create object two dimensional array
		Object studentdata[][]= {{"StudentID", "Name",
"Department"},

	{101,"ruthra","EEE"},

	{102,"rajesh","EcE"},

	{103,"rose","Mech"}};

	//get rows and cells
	//using for loop
	//		int rows=studentdata.length;
								//coloumn
	//	int cols=studentdata[0].length;

	//	System.out.println(rows);//4
	//	System.out.println(cols);//3

	/*	for(int r=0;r<rows;r++)//outer for loop
		{
			XSSFRow row=sheet.createRow(r);

			for(int c=0;c<cols;c++) {

				XSSFCell cell=row.createCell(c);//0
				Object value =studentdata[r][c];

				if(value instanceof String)
					cell.setCellValue((String)value);
				if(value instanceof Integer)
					cell.setCellValue((Integer)value);
				if(value instanceof Boolean)
					cell.setCellValue((Boolean)value);
			}
		}*/

	////	for each loop
```

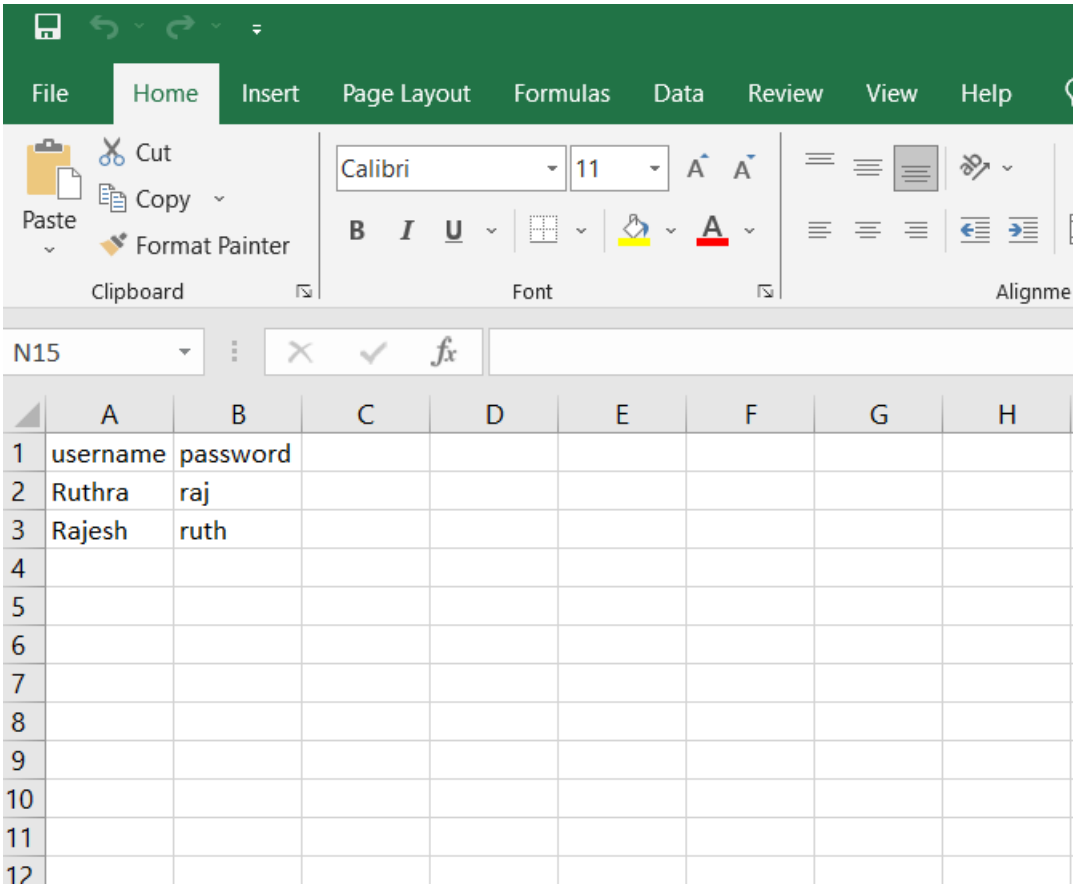
```
            int rowCount=0;
            for(Object[] studdata:studentdata) {
            XSSFRow row=sheet.createRow(rowCount++);
            int columnCount=0;
            for(Object value:studdata) {

                XSSFCell cell=row.createCell(columnCount++);
                if(value instanceof String)
                    cell.setCellValue((String)value);
                if(value instanceof Integer)
                    cell.setCellValue((Integer)value);
                if(value instanceof Boolean)
                    cell.setCellValue((Boolean)value);
            }


            String filepath=".\\datafiles\\student.xlsx";
            FileOutputStream outstream=new
FileOutputStream(filepath);
            workbook.write(outstream);

            outstream.close();

            System.out.println("the student file is successfully
updated");
        }
}

}
```

Output



Test Driven Data

We are create excel file outside the folder




```
package appachie_poi;
```

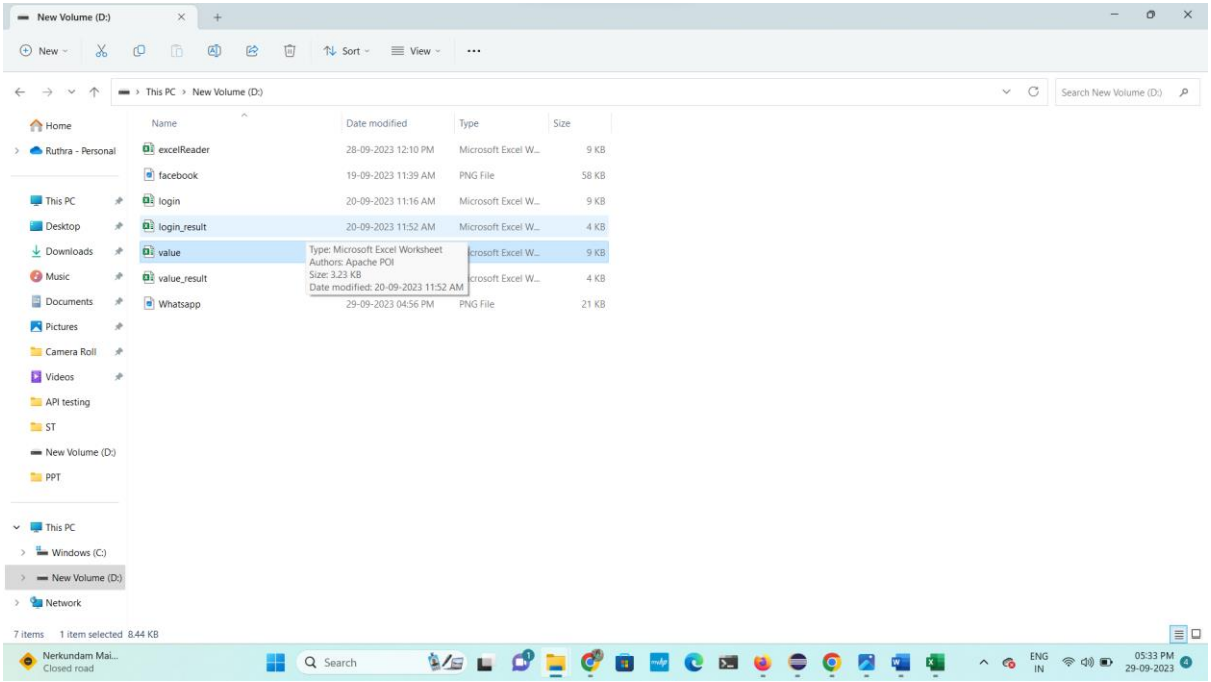
```
import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import java.io.*;
import java.util.*;


public class Login_userpass {
        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);

            System.out.print("Enter your username: ");
            String username = sc.nextLine();
            System.out.print("Enter your password: ");
            String password = sc.nextLine();
            sc.close();

            try {
                boolean loginValid = isLoginValid("D://login.xlsx",
username, password);

                // Append the result to the output Excel file
                writeToResultExcel(loginValid);

                if (loginValid) {
                    System.out.println("Login successful!");
                } else {
                    System.out.println("check the credatials!");
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        public static boolean isLoginValid(String filePath, String
username, String password) throws IOException {
            FileInputStream file = new FileInputStream(new
File(filePath));
            Workbook workbook = new XSSFWorkbook(file);
            Sheet sheet = workbook.getSheetAt(0);

            Iterator<Row> rowIterator = sheet.iterator();
```

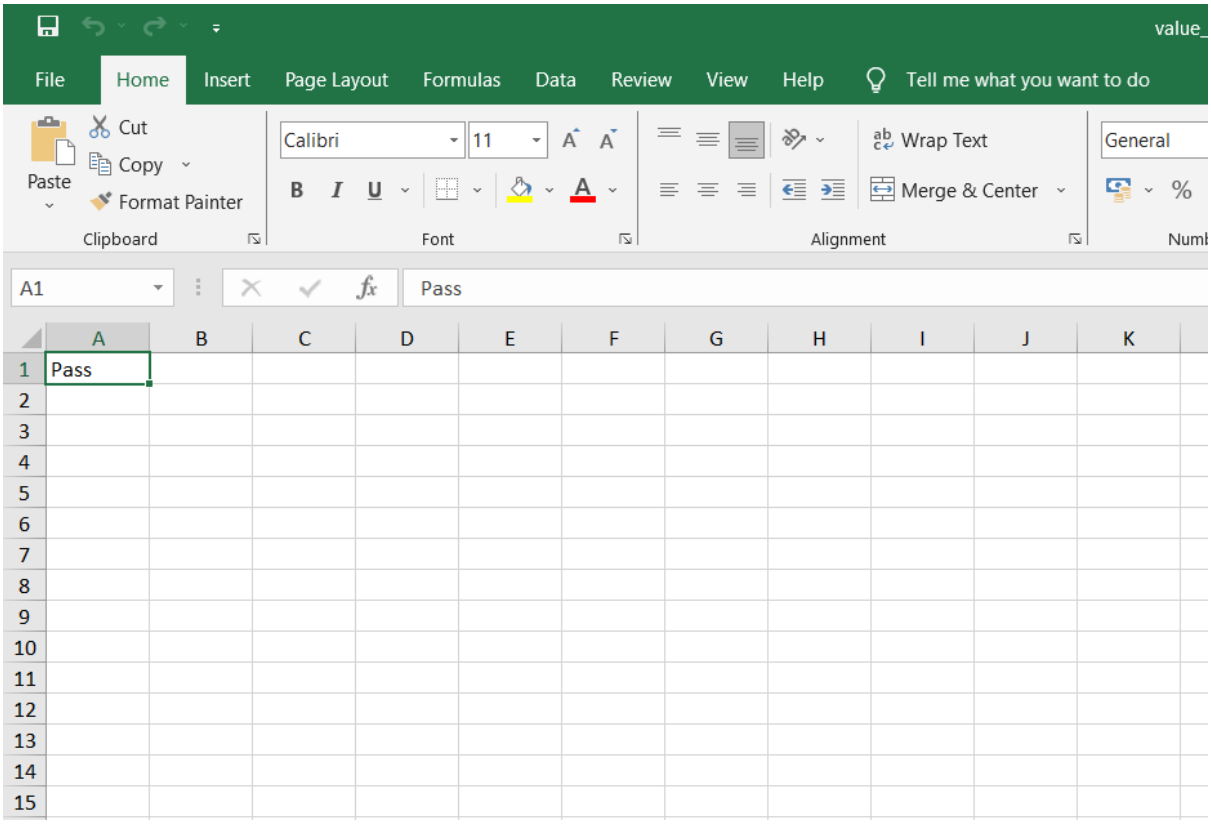
```
            while (rowIterator.hasNext()) {
                Row row = rowIterator.next();
                Cell usernameCell = row.getCell(0); // username is in
the first column
                Cell passwordCell = row.getCell(1); //  password is in
the second column

                String storedUsername =
usernameCell.getStringCellValue();
                String storedPassword =
passwordCell.getStringCellValue();

                if (storedUsername.equals(username) &&
storedPassword.equals(password)) {
                    workbook.close();
                    file.close();
                    return true; // Login successful
                }
            }
        workbook.close();
            file.close();
            return false; // check credantials
        }

        public static void writeToResultExcel(boolean loginValid)
throws IOException {
            FileInputStream resultFile = null;
            Workbook workbook = null;

            try {
                // Open the existing result Excel illana it create
automatiaclly
                File outputFile = new File("D://login_result.xlsx");
                if (outputFile.exists()) {
                    resultFile = new FileInputStream(outputFile);
                    workbook = new XSSFWorkbook(resultFile);
                } else {
                    workbook = new XSSFWorkbook();
                }
                Sheet sheet = workbook.getSheet("Login Result");
                if (sheet == null) {
```

```
                sheet = workbook.createSheet("Login Result");
            }
        int lastRowNum = sheet.getLastRowNum();
            Row row = sheet.createRow(lastRowNum + 1);
            Cell cell = row.createCell(0);

            if (loginValid) {
                cell.setCellValue("Pass");
            } else {
                cell.setCellValue("Fail");
            }

            FileOutputStream fileOut = new
FileOutputStream(outputFile);
            workbook.write(fileOut);
            fileOut.close();
        } finally {
            if (workbook != null) {
                workbook.close();
            }
            if (resultFile != null) {
                resultFile.close();
            }
        }
    }
}
```

Output

Externally data automatically created result file.





TestNG

Definition

- **TestNG** is a testing framework inspired from **JUnit** and **NUnit** but introducing some new functionalitythat makes it more powerful and easier to use.
- It is an open-source automated testing framework; where **NG** ofTest**NG** means **N**ext **G**eneration.
- TestNG is similar to JUnit but it is much more powerful than JUnit but still, it's inspired by JUnit.
- It is designed to be better than JUnit, especially when testing integrated classes. Pay special thanks to Cedric Beust who is thecreator of TestNG.

USES

TestNG eliminates most of the limitations of the older framework and gives the developer the ability to write more flexible and powerful tests with help of easy annotations, grouping, sequencing & parametrizing.

What are the Benefits of TestNG:-

1. It gives the ability to produce **HTML Reports** of execution
2. **Annotations** made testers life easy
3. Test cases can be **Grouped & Prioritized** more easily
4. **Parallel** testing is possible
5. Generates **Logs**
6. Data **Parameterization** is possible

Test Case Writing process in TestNG

- **Step 1 -** Write the business logic of the test
- **Step 2 -** Insert TestNG annotations in the code
- **Step 3 -** Add the information about your test (e.g. the class names, methods names, groups names, etc...) in a testng.xml file
- **Step 4 -** Run TestNG

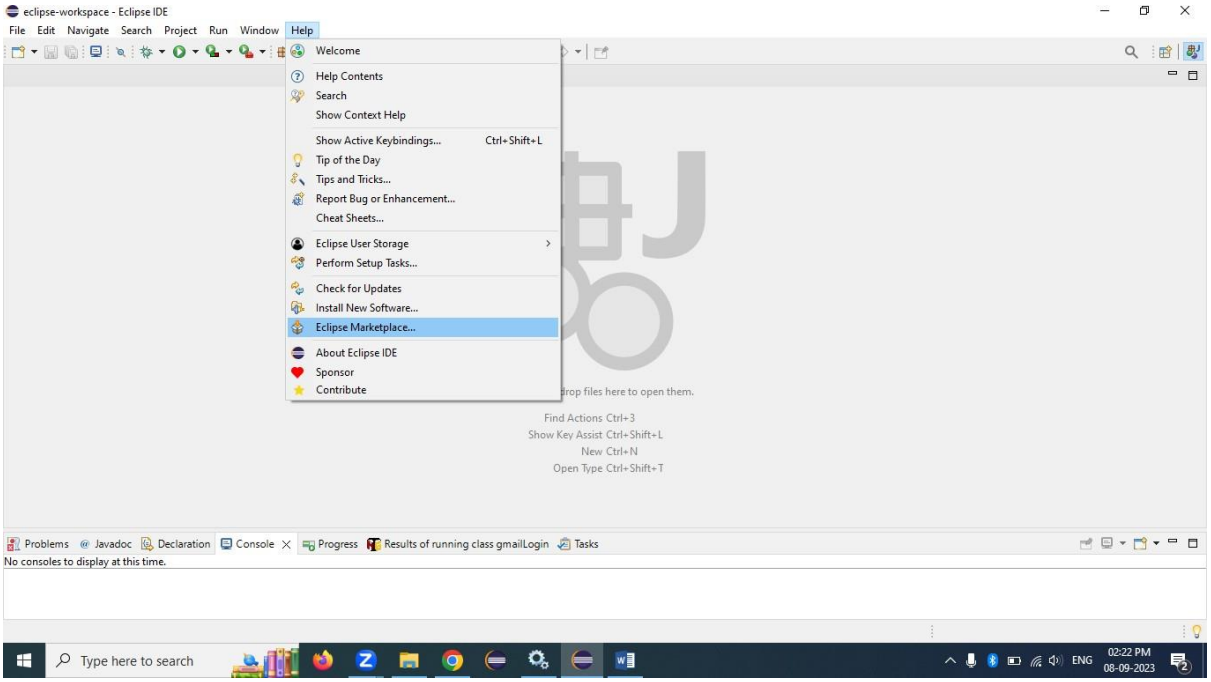
What are the different Annotations are present inTestNG?

- **@BeforeSuite**: The annotated method will be run before all tests in this suite have run.
- **@AfterSuite**: The annotated method will be run after all tests in this suite have run.
- **@BeforeTest**: The annotated method will be run before any test method belonging to the classes inside the tag is run.

- **@AfterTest**: The annotated method will be run after all the test methods belonging to the classes inside the tag have run.
- **@BeforeGroups**: The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.
- **@AfterGroups**: The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.
- **@BeforeClass**: The annotated method will be run before the first test method in the current class is invoked.
- **@AfterClass**: The annotated method will be run after all the test methods in the current class have been run.
- **@BeforeMethod**: The annotated method will be run before each test method.
- **@AfterMethod**: The annotated method will be run after each test method.
- **@Test**: The annotated method is a part of a test case.

How to configure TestNG in EclipseStep1: Install TestNG

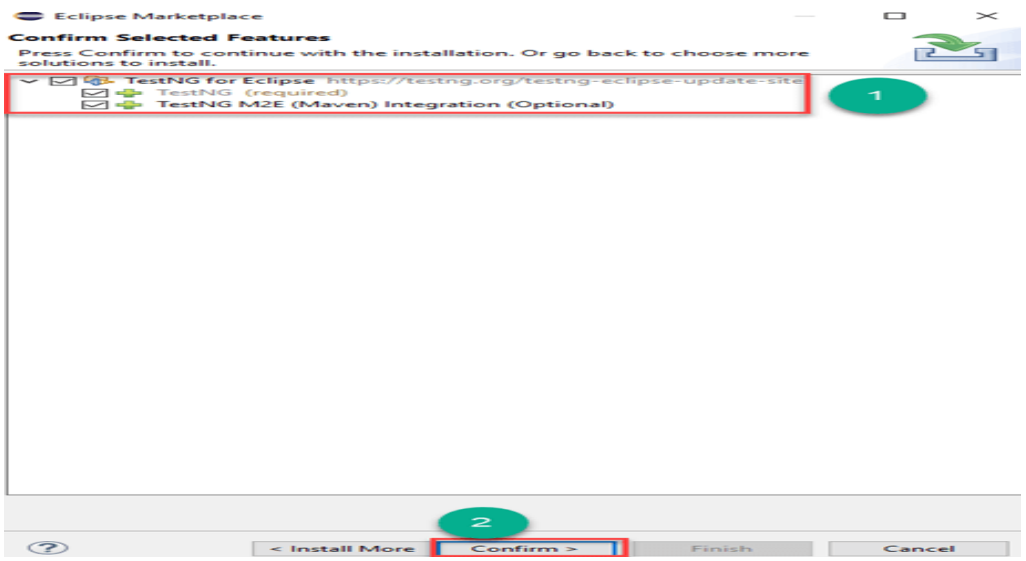
in Eclipse HelpàEclipse marketplaceàClick

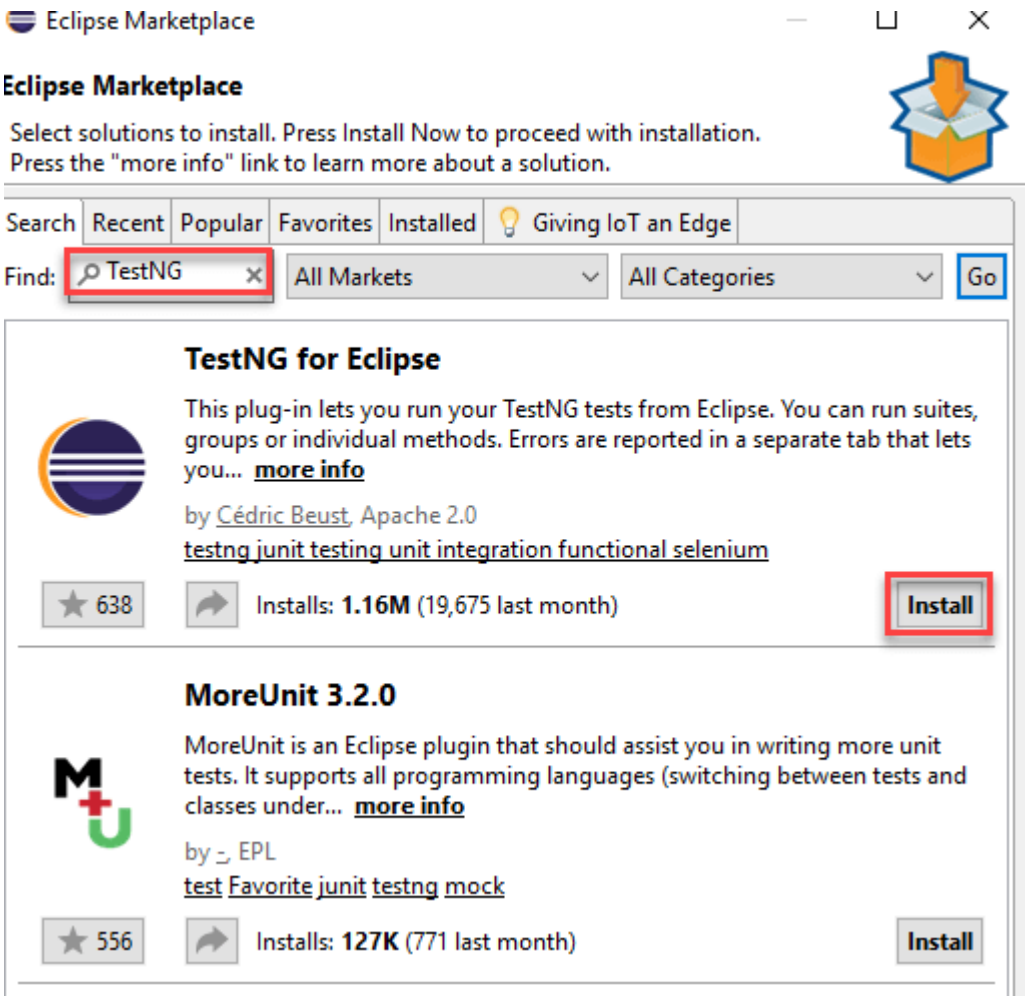


Step 2: Search for TestNG

1. Searchbar àType **TestNG** àClick Enter

2. You will see TestNG àClick **Install**





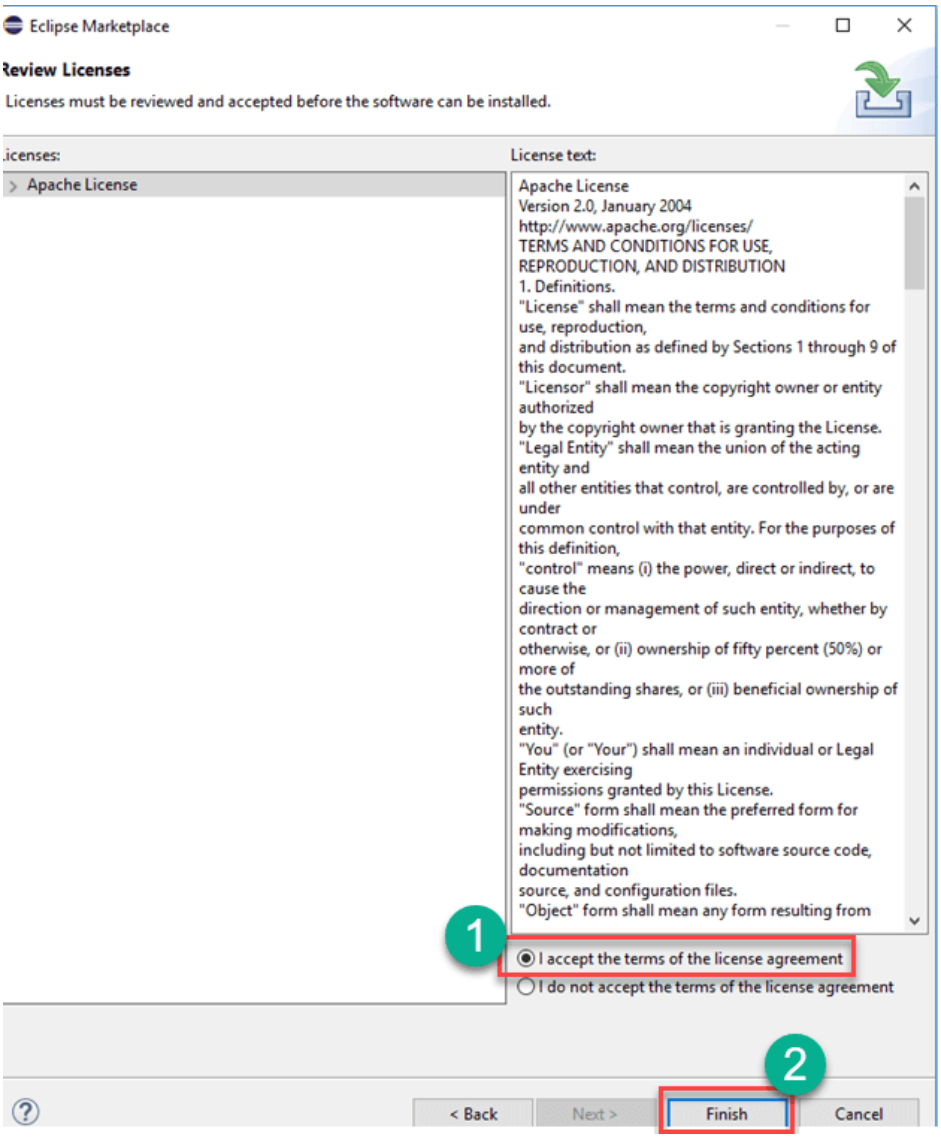
Step 3: After clicking **Install,** it shows like belowpage. **It takes some time to install**…

1. Click **confirm**

Step 4: It ask for the license to accept

1. Click **I accept the terms for the license agreement.**

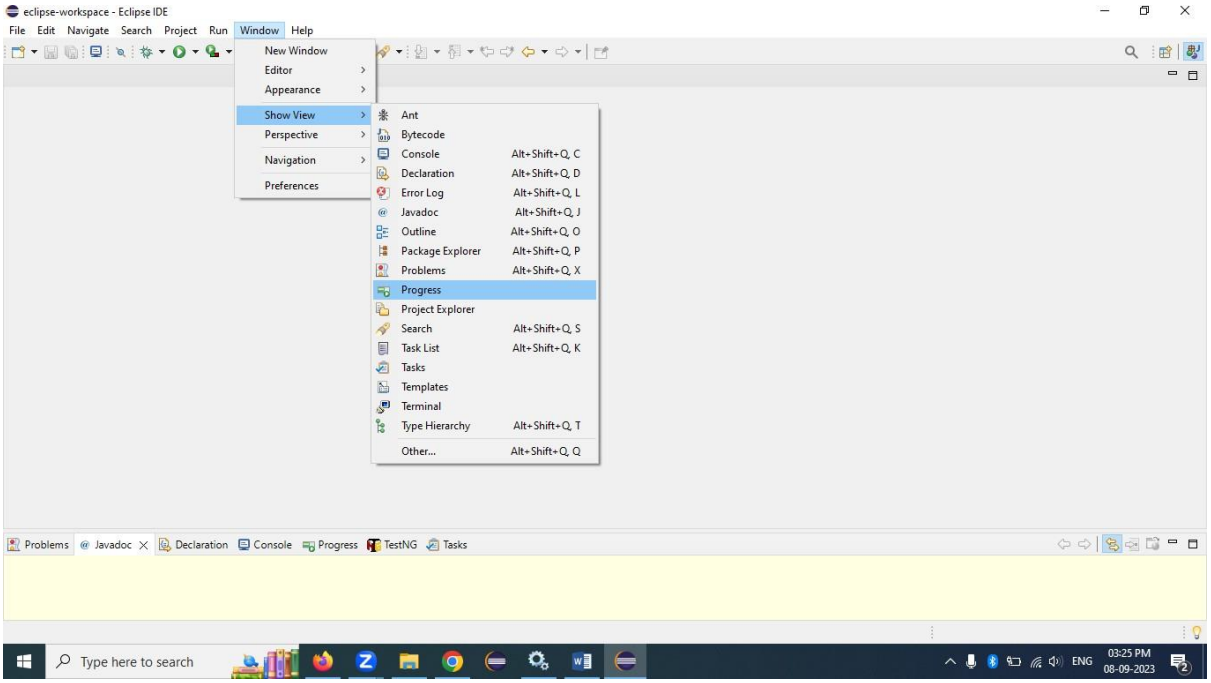
2. Click **Finish.**

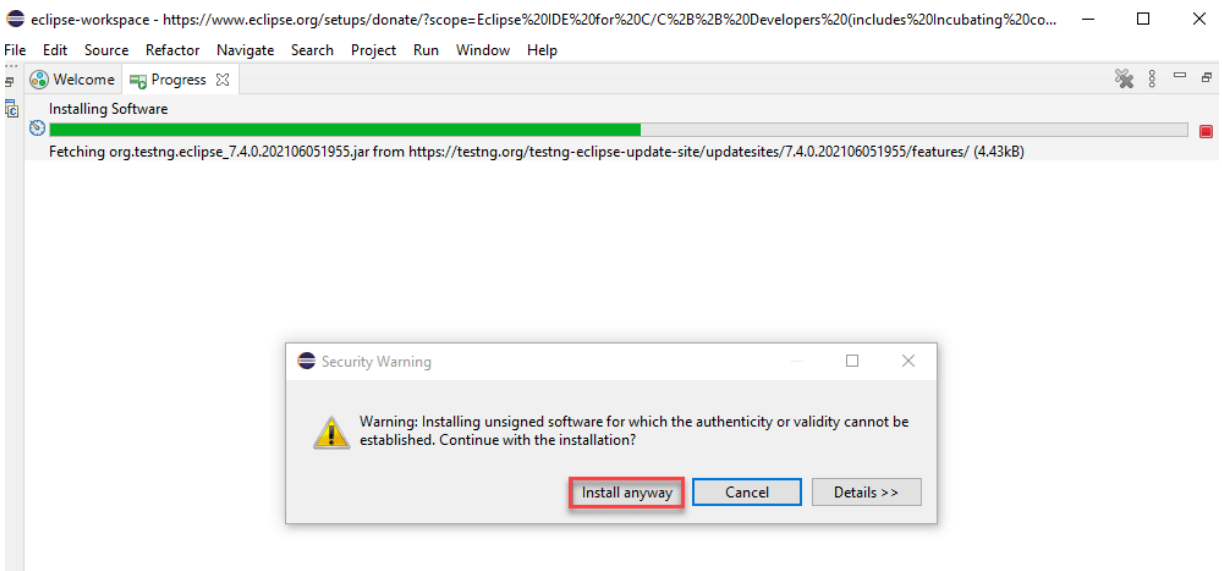


Step 5: Viewing **installation progress**

1. **windowàShow viewàProgress**
2. You will see the **progress** at below
3. I f it asks for any security warningàClick

   Install anyway.

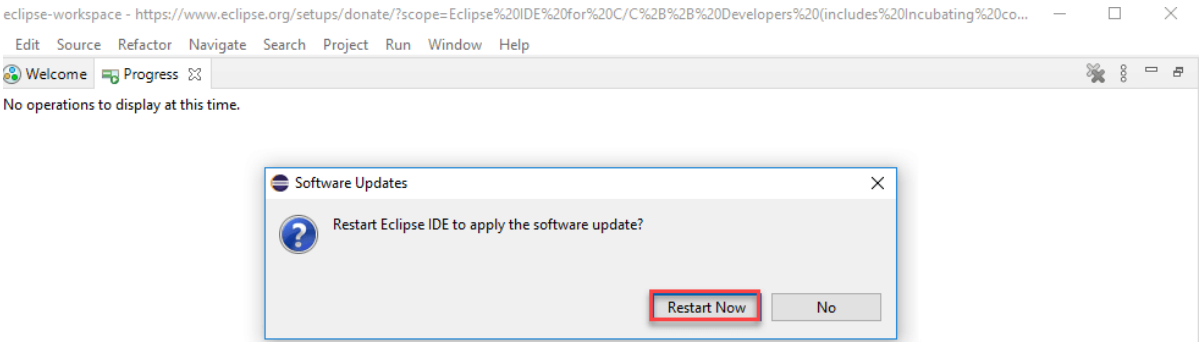




Step 6: After Installation

1. It asks for **Restart now.**

2. Click **Restart Now**

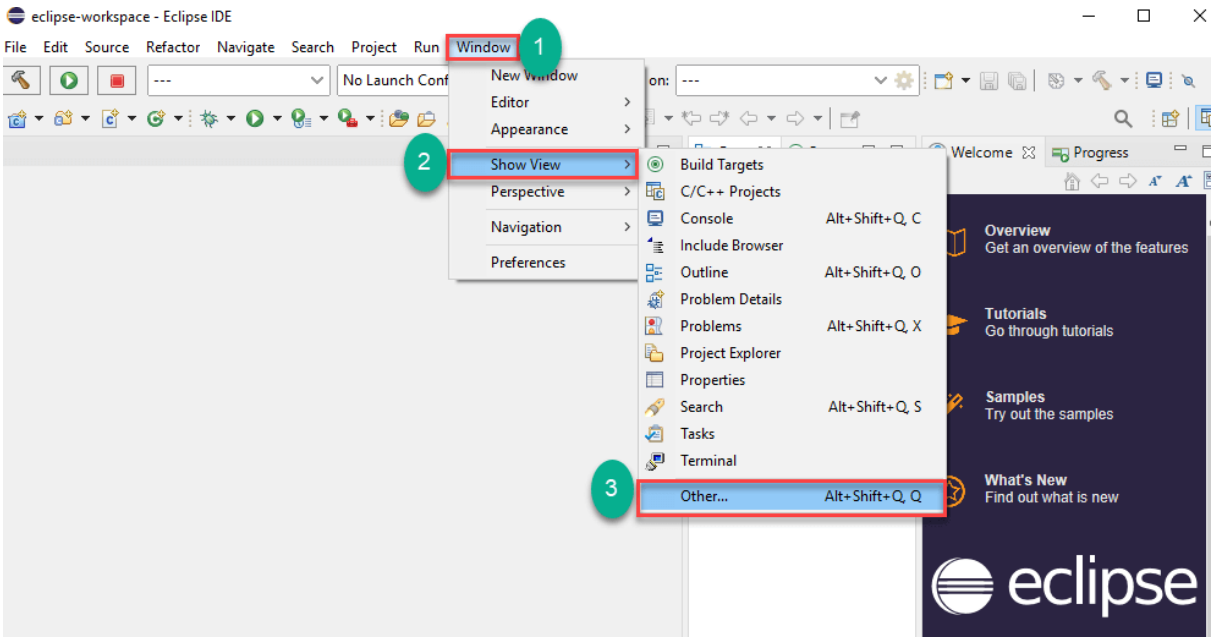


Step 7: Checking whether **TestNG is installed ornot?**

1. Click **window**

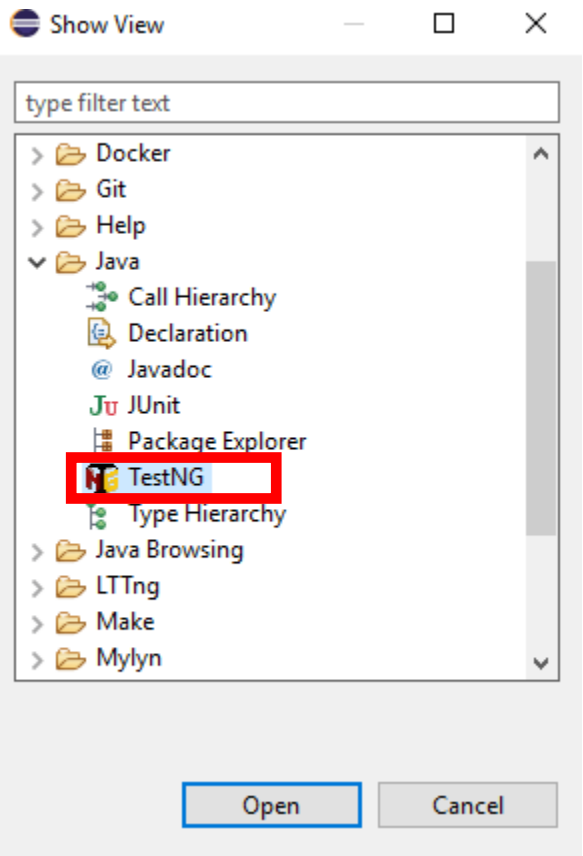
2. Click **show view**

3. Click **others**



Step 8: TestNG is present??

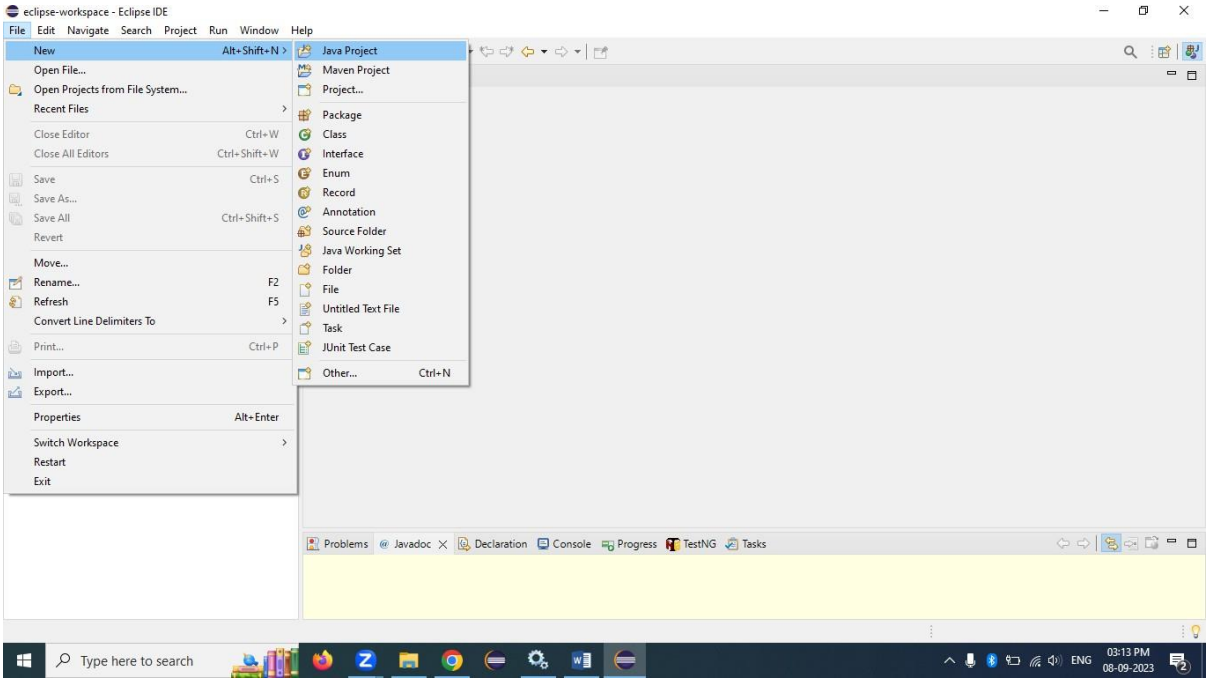
1. Click **Java Folder**
2. Inside **TestNG** is present



How to Create New Java project?

Step 1: Creating new project

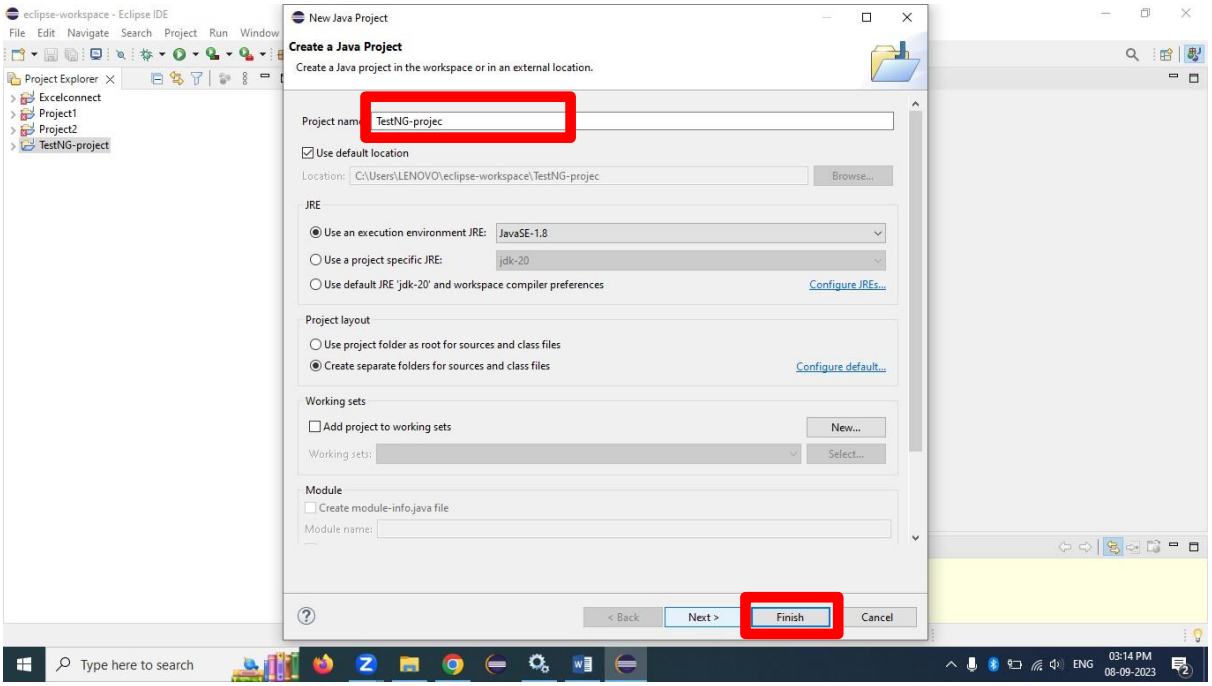
1. **File**à **New**à **Java project**



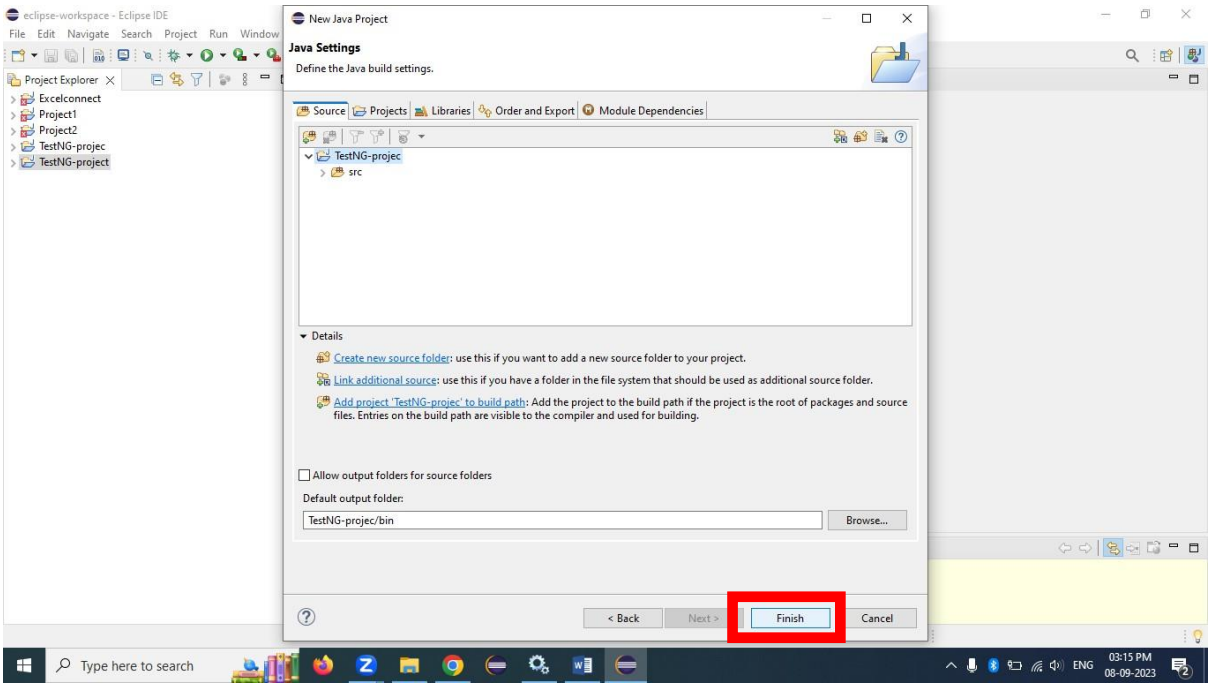
Step 2: You can name your project as your wish

1.I named as **TestNg-project**

2. Click **Next**



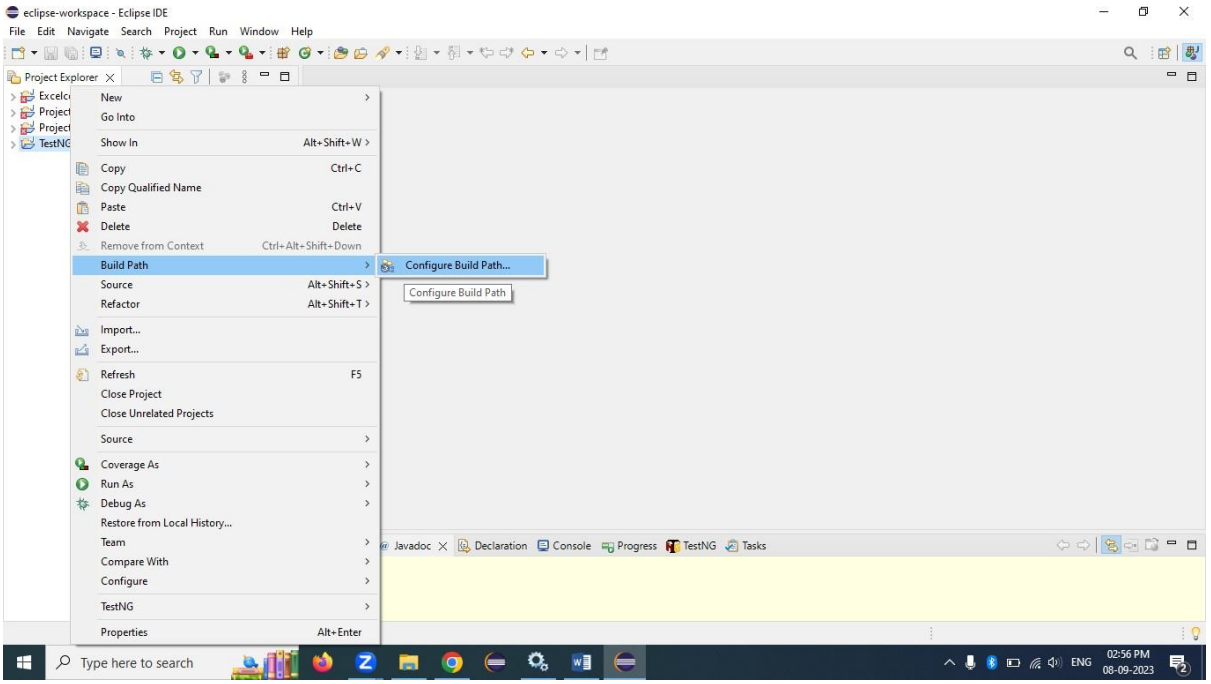
Step 3: Click **Finish**



How to configure Jar files insideTestNG?

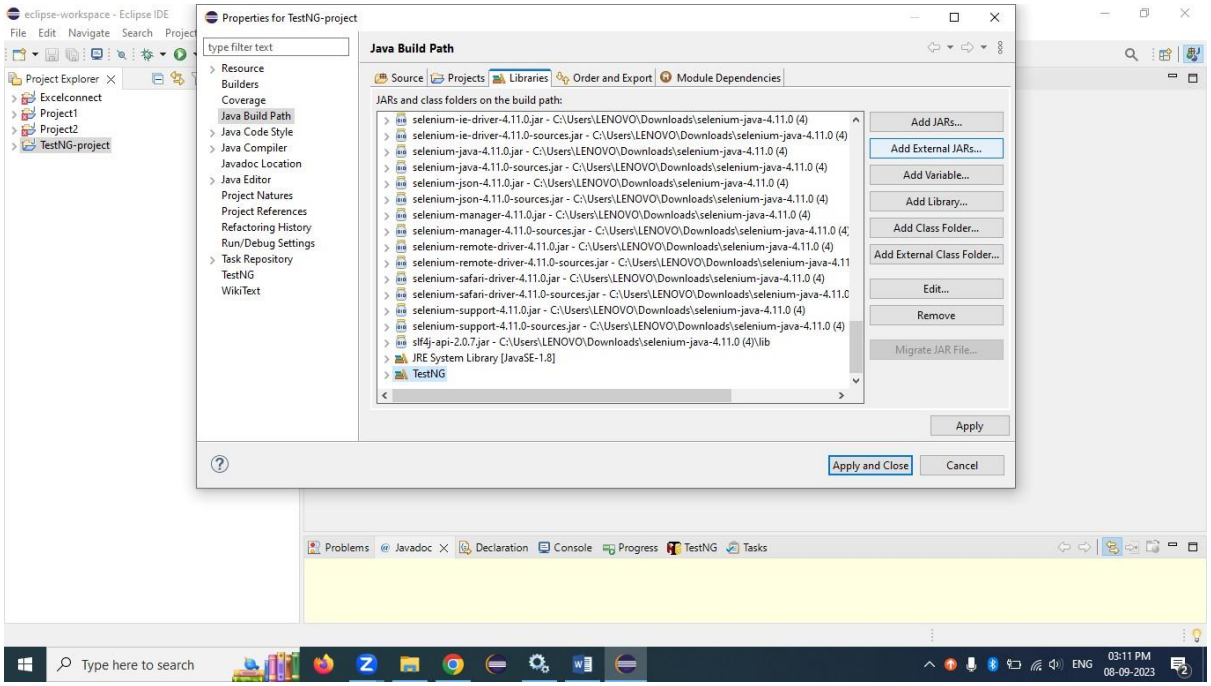
Step 1: Configure path

1. Right click on **TestNG-project** (Name of theproject)
2. Click **Build Path**
3. Click **Configure Build Path**



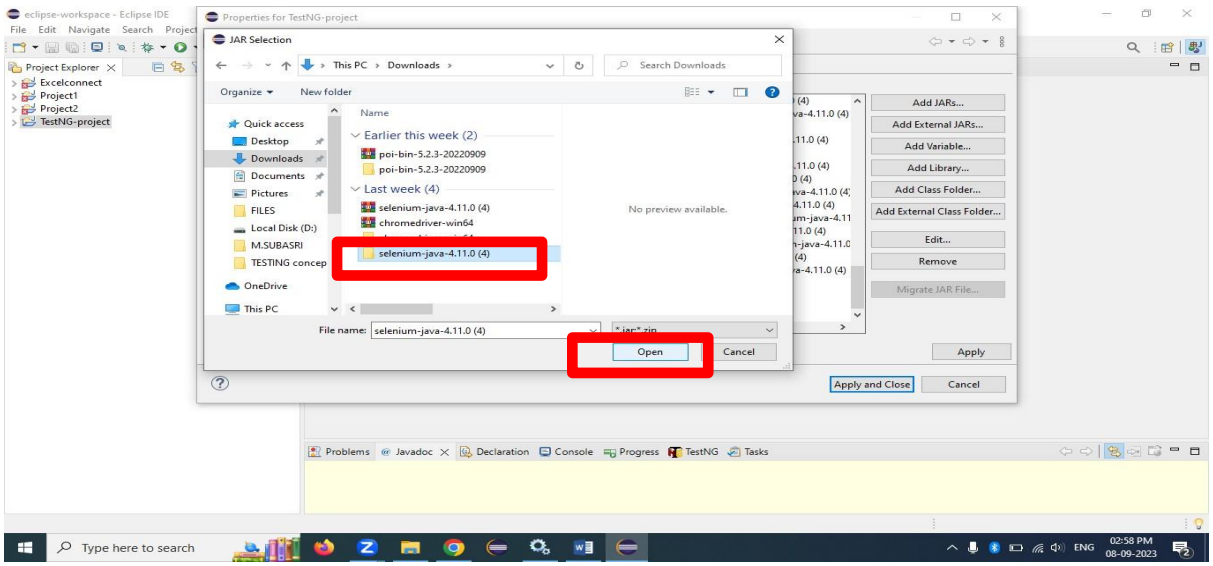
Step 2: Have to **configure Selenium insideTestNG**

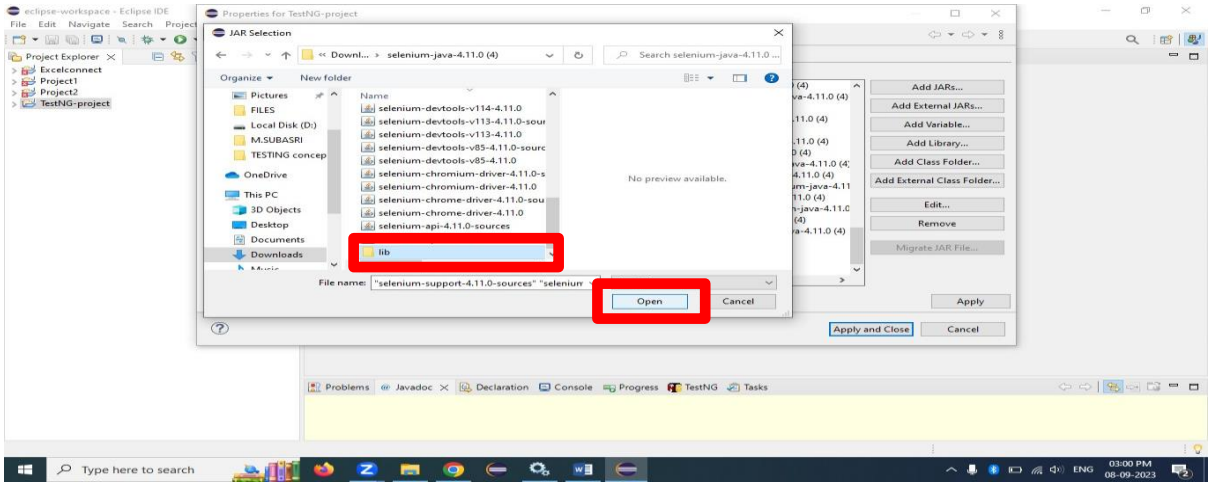
1. Click **libraries**
2. Inside you will see **TestNG**à**Click** that
3. Click **Add External Jars**



Step 3:The folder will open

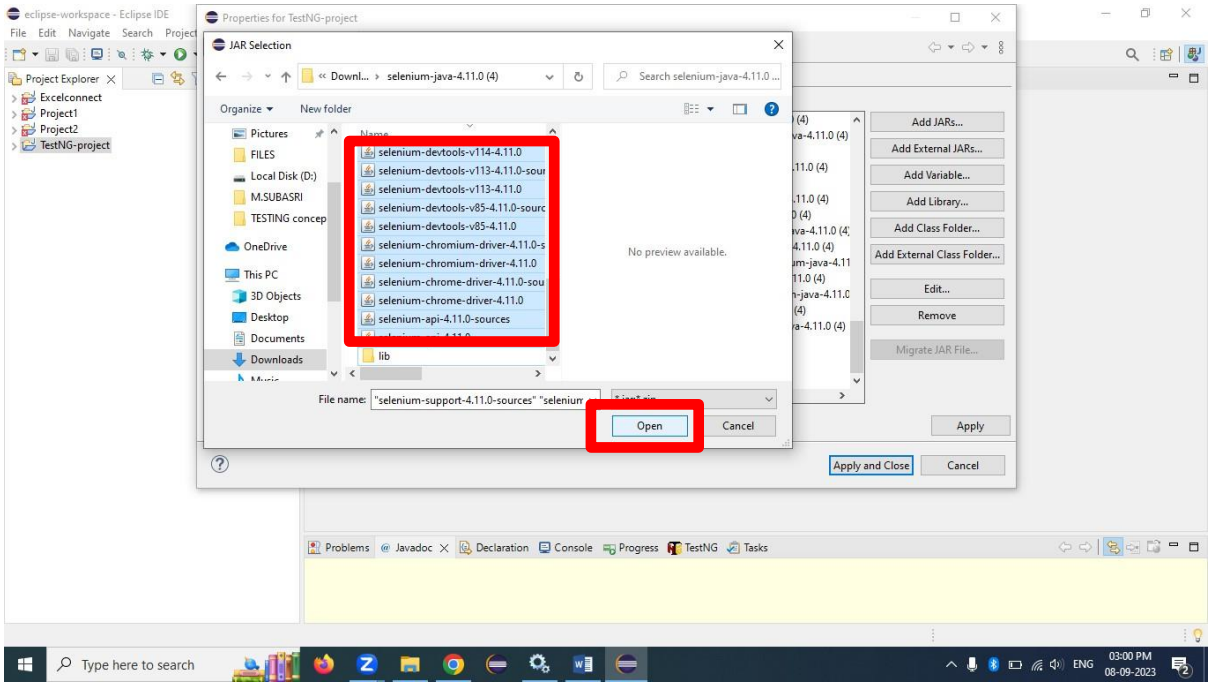
1. Click **Selenium folder** that you have downloaded already.
2. If you not yet downloaded selenium, you candownload from here, https://www.selenium.dev/ (optional)





3. Click **open**

Step 4: Select all jar files **except lib folderàOpen**

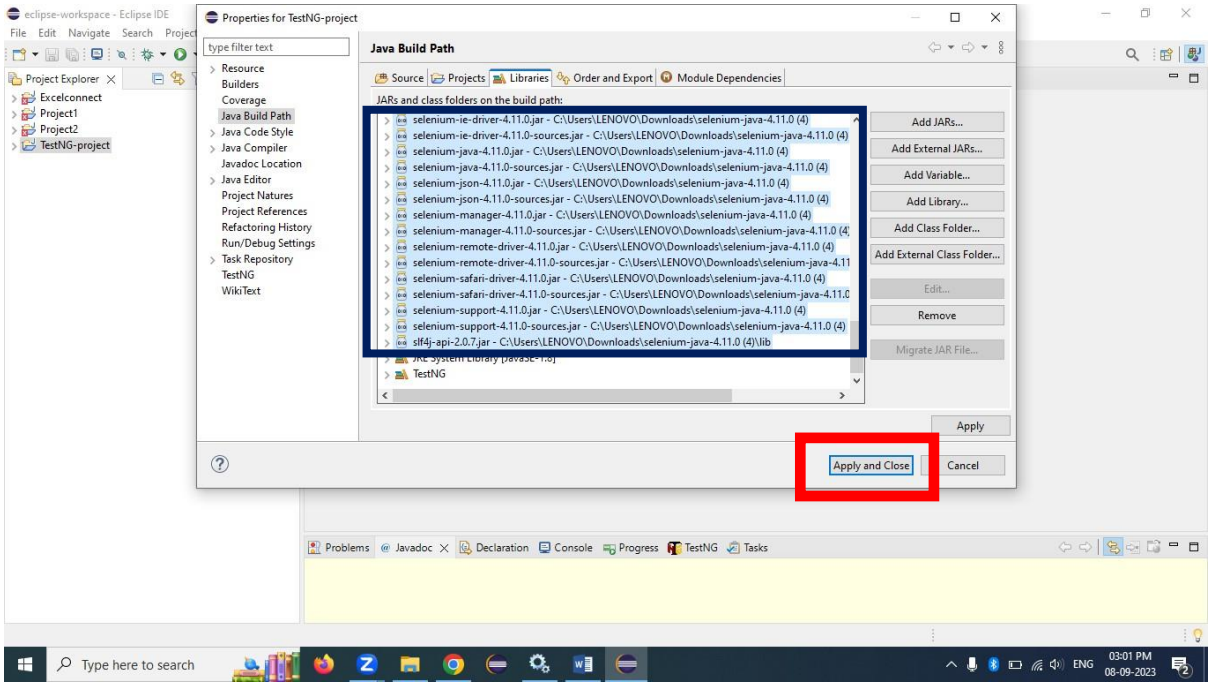


Step 5: Now **select lib folderàOpen**

Step 6: All selenium jar files will be uploaded toTestNG.

Now all jar files will be configured, that isshown in blue color box in below image.

Click **Apply and Close**



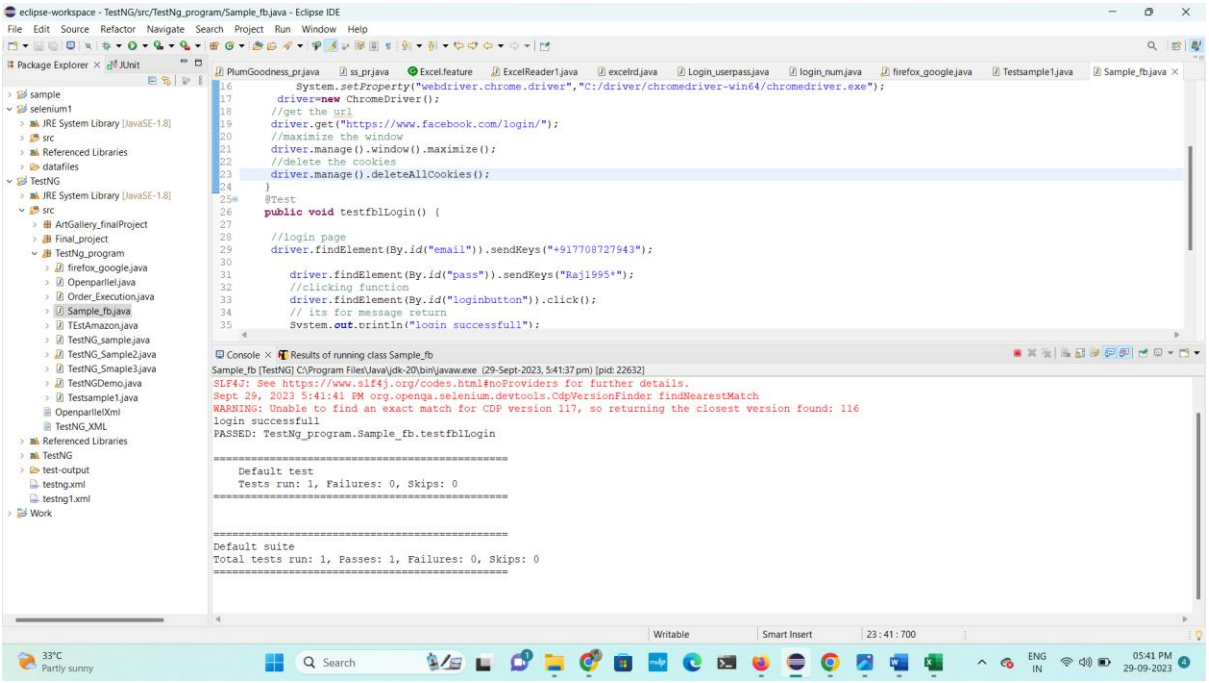

```
package TestNg_program;


import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.annotations.AfterMethod;
```

```
import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;



public class Sample_fb {

private WebDriver driver;



 @BeforeMethod

   public void setUp() {



    System.setProperty("webdriver.chrome.driver","C:/
    driver/chromedriver-win64/chromedriver.exe");

  driver=new ChromeDriver();

 //get the url

 driver.get("https://www.facebook.com/login/");

 //maximize the window

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

 //delete the cookies
```

```
 driver.manage().deleteAllCookies();

}

@Test

  public void testfblLogin() {



 //login page


   driver.findElement(By.id("email")).sendKeys("+917
   708727943");




        driver.findElement(By.id("pass")).sendKeys("R
   aj1995*");

   //clicking function

   driver.findElement(By.id("loginbutton")).click();

   // its for message return

   System.out.println("login successfull");

}

@AfterMethod

public void tearDown() {
```

```
        if (driver != null) {

            driver.quit();

        }



    }

    }
```

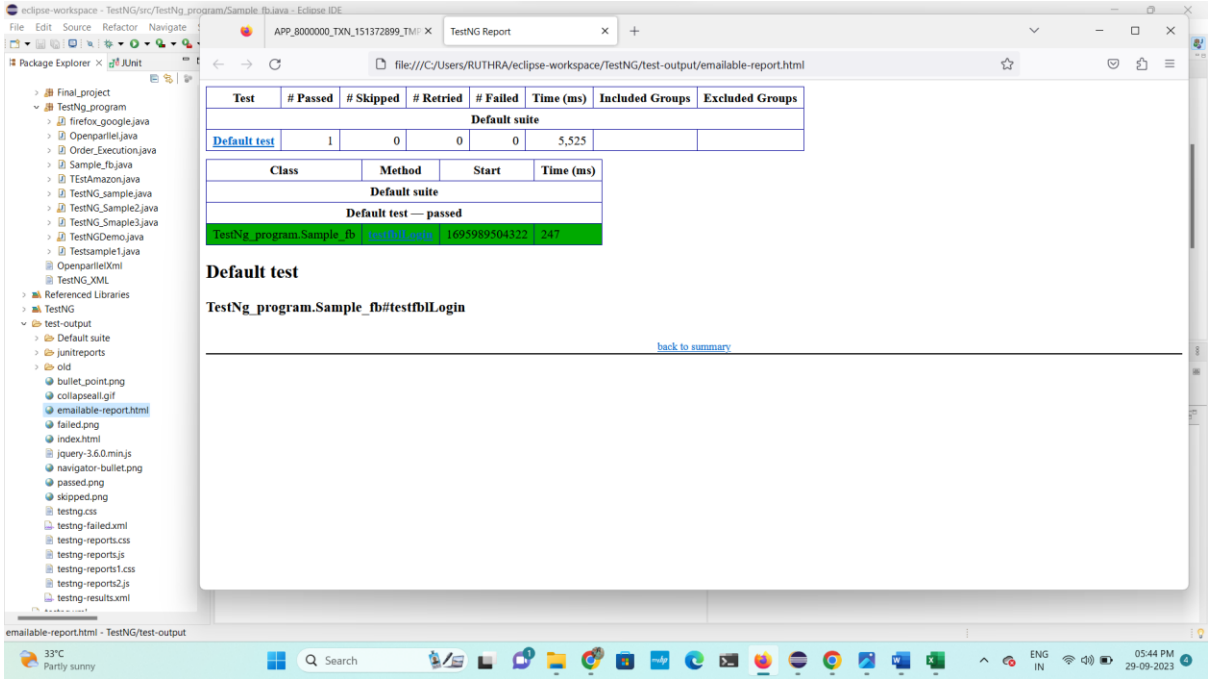
Output



Once Right click the TestNg package to click refress

Below test output file shows.

And verify our reports

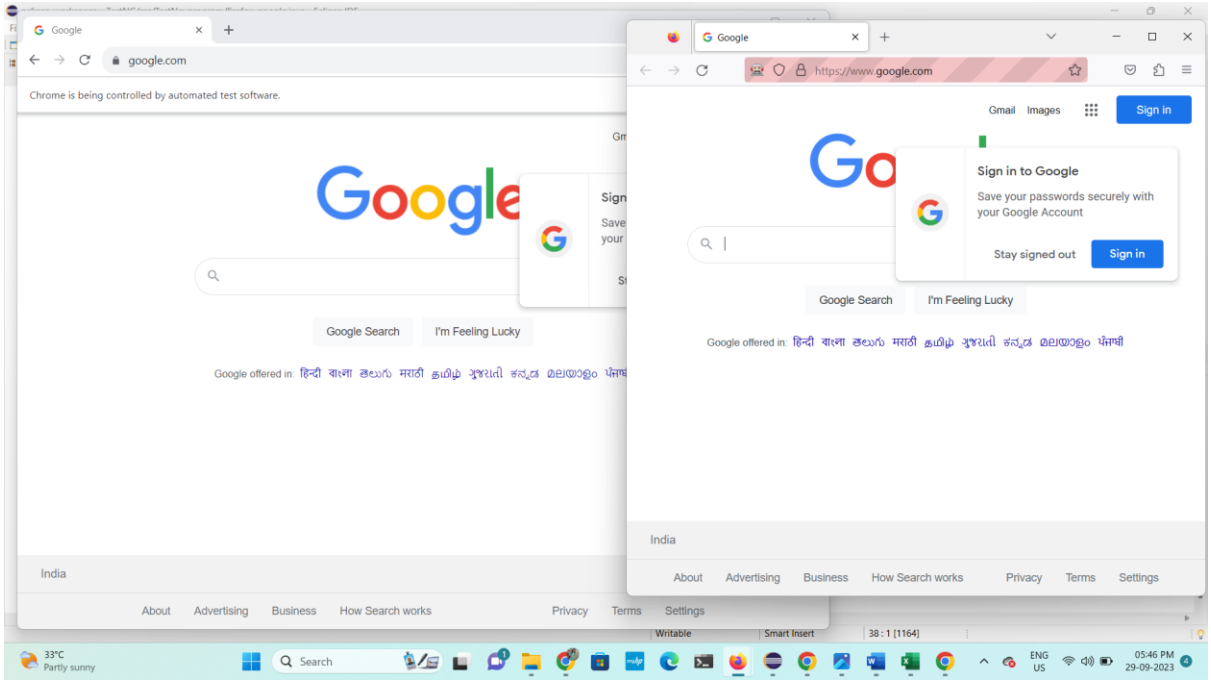


Crossbrowser Testing:

```
package TestNg_program;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.Test;

                                                                    //
cross - browser testing

public class firefox_google {

        @Test

         public void ChromeDriver() {
```

```
System.setProperty("webdriver.chrome.driver","C:/driver/chromedri
ver-win64/chromedriver.exe");
                WebDriver driver=new ChromeDriver();
                 driver.get("https://www.google.com/");
        }
        @Test
    public void fire_fox() throws InterruptedException  {

System.setProperty("webdriver.geccko.driver","C:/Users/RUTHRA/
Dropbox/PC/Downloads/geckodriver-v0.33.0-win-
aarch64/gecckodriver.exe");
        WebDriver amaz = new FirefoxDriver();

        amaz.get("https://www.google.com/");
      //  amaz.findElement(By.id("APjFqb")).sendKeys("iphone");


      //  Thread.sleep(3000); // Adding a sleep to see the results before
closing the browser

      //  amaz.quit(); // Use quit() to close the browser and release
resources
    }
}
```

Output

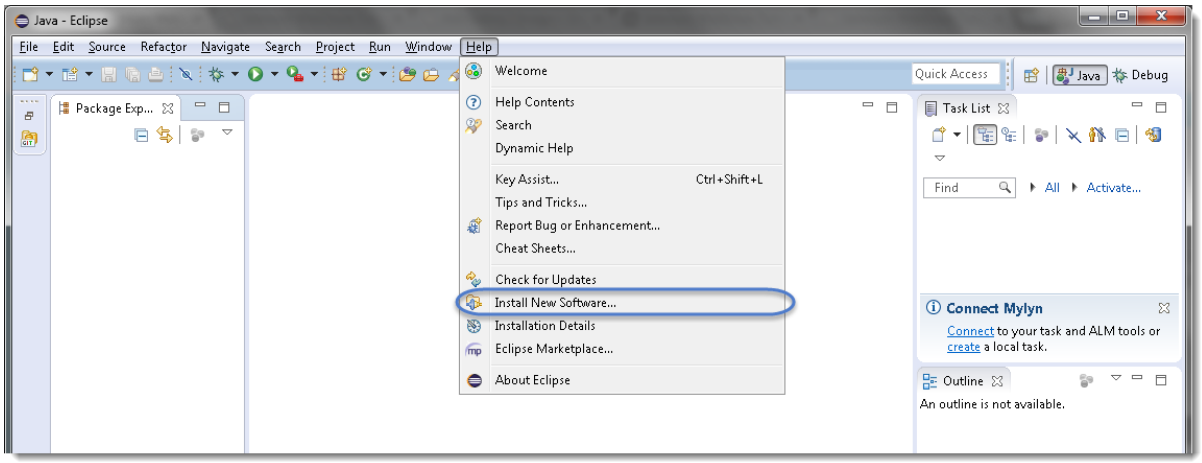


CUCUMBER

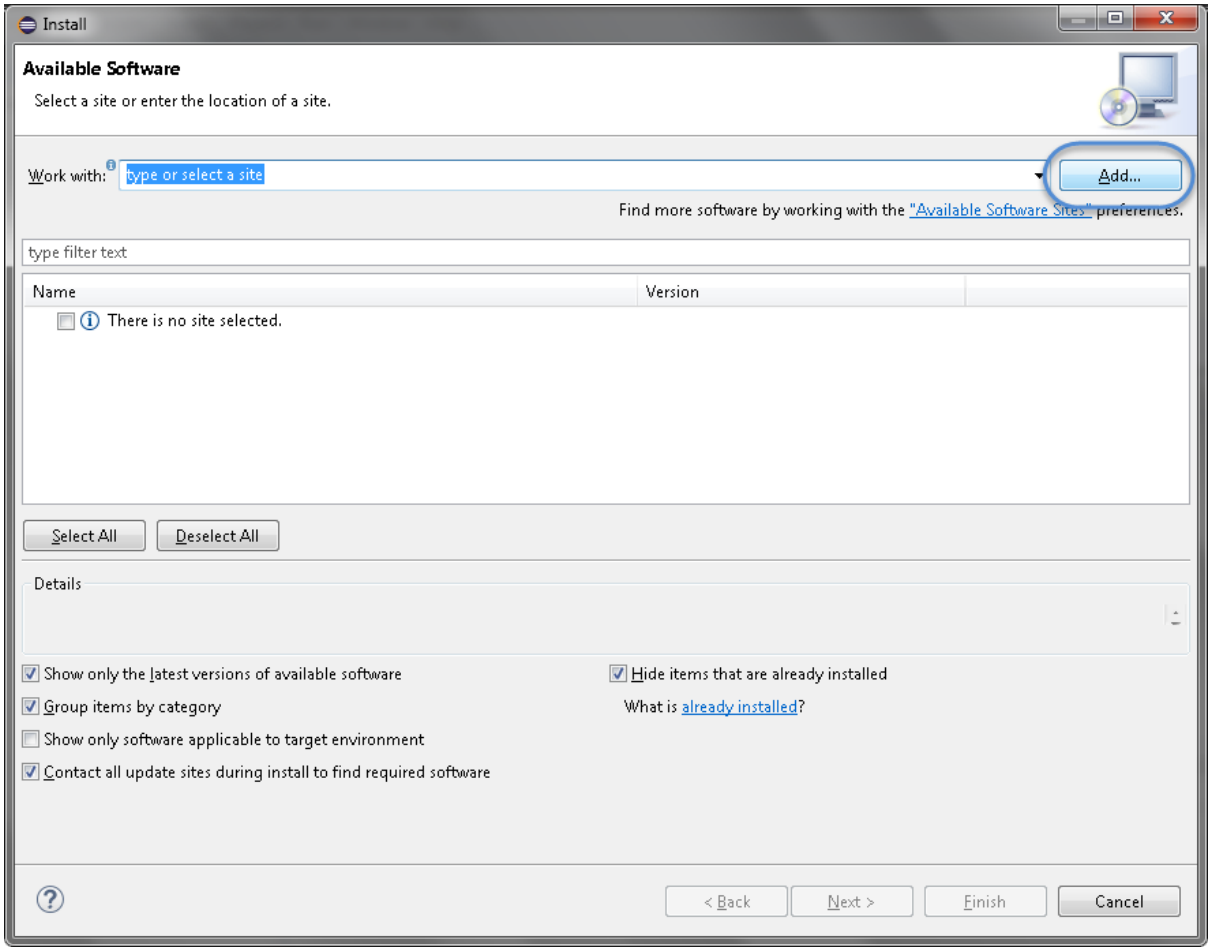
Cucumber is a testing framework which supports Behavior Driven Development (BDD). It lets us define application behavior in plain meaningful English text using a simple grammar defined by a language called Gherkin.

Steps to follow:

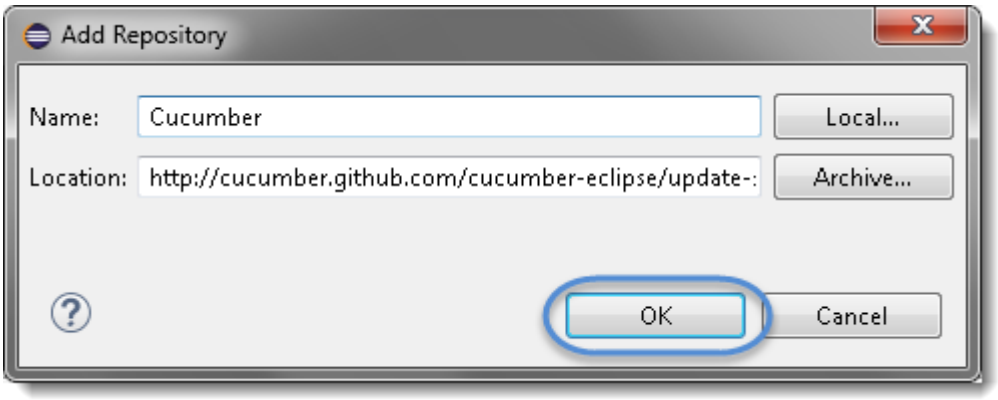
1. Launch the Eclipse IDE and from Help menu, click "**Install New Software**".



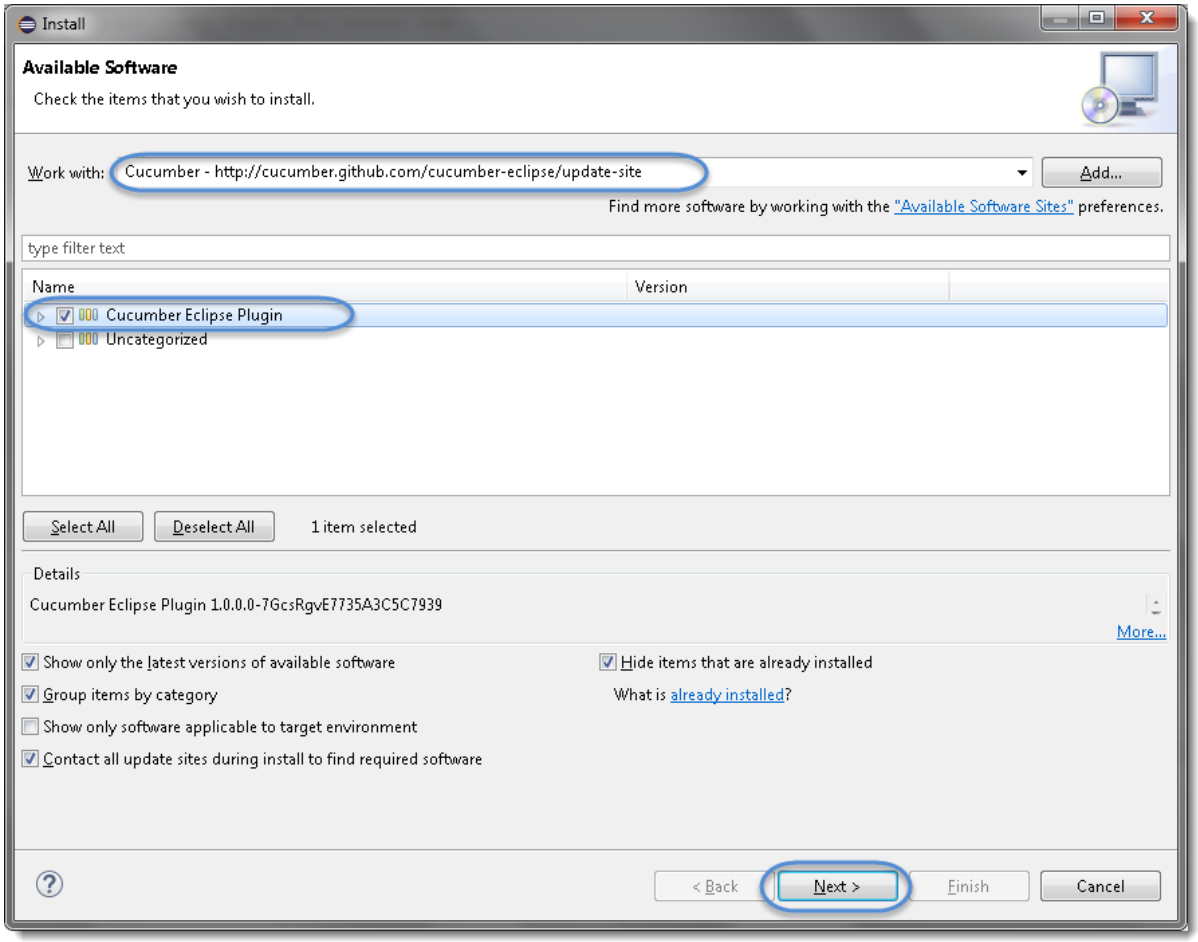
2. You will see a dialog window, click “**Add**” button.



3. Type name as you wish, let's take “**Cucumber**” and type “**http://cucumber.github.com/cucumber-eclipse/update-site**” as location. Click **OK**.

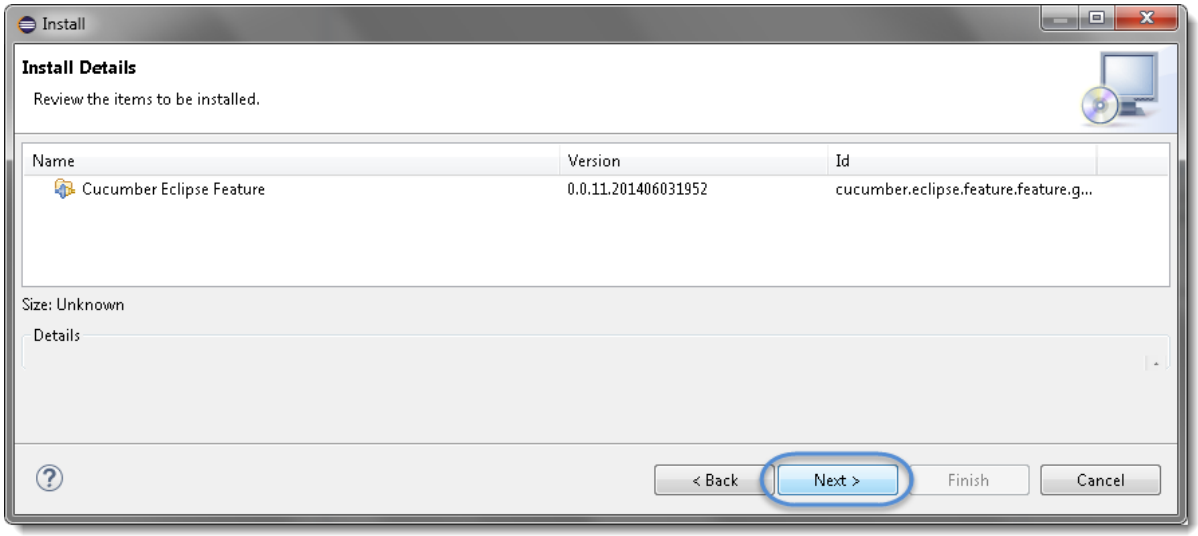


4. You come back to the previous window but this time you must see **Cucumber Eclipse** Plugin option in the available software list. Just **Check** the box and press “**Next**” button.

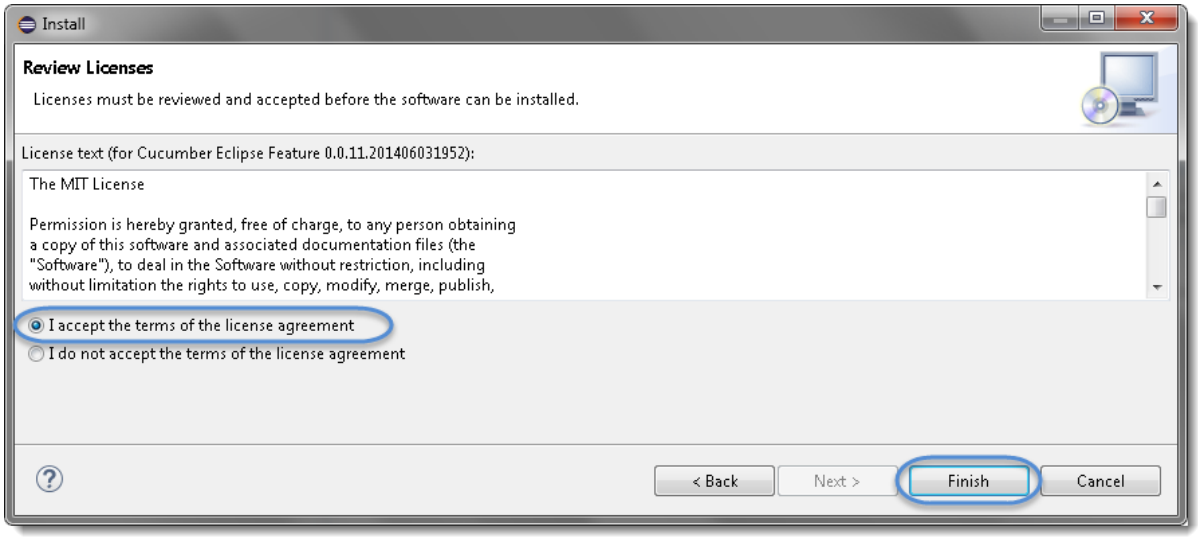


Note: If running behind a proxy server and you get a 'HTTP Proxy Authentication Required' error you may need to contact a system administrator to set up your proxy server settings.

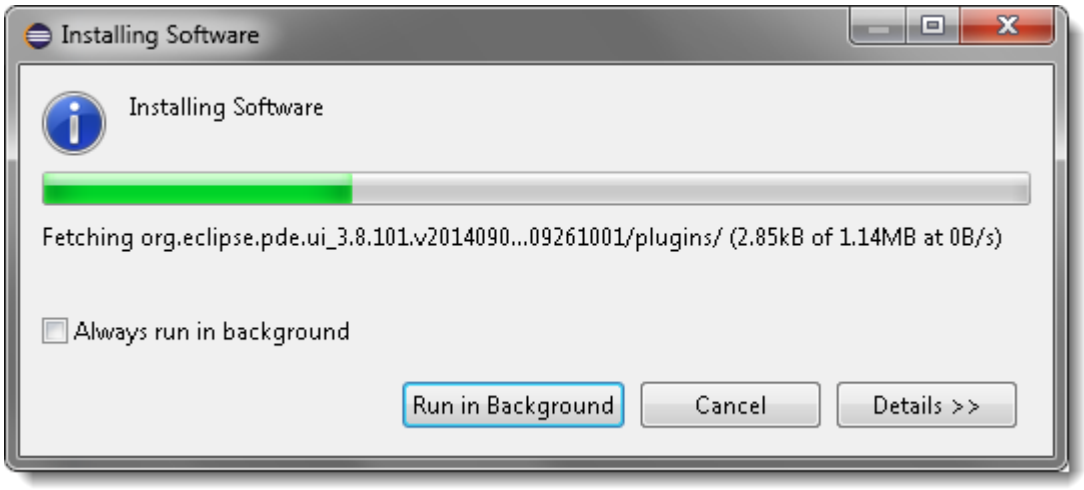
5. Click on **Next**.



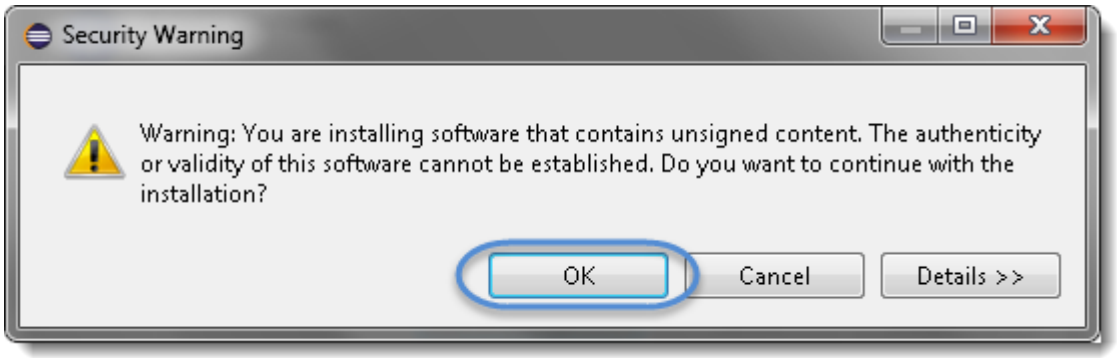
6. Click "**I accept the terms of the license agreement**" then click **Finish**.



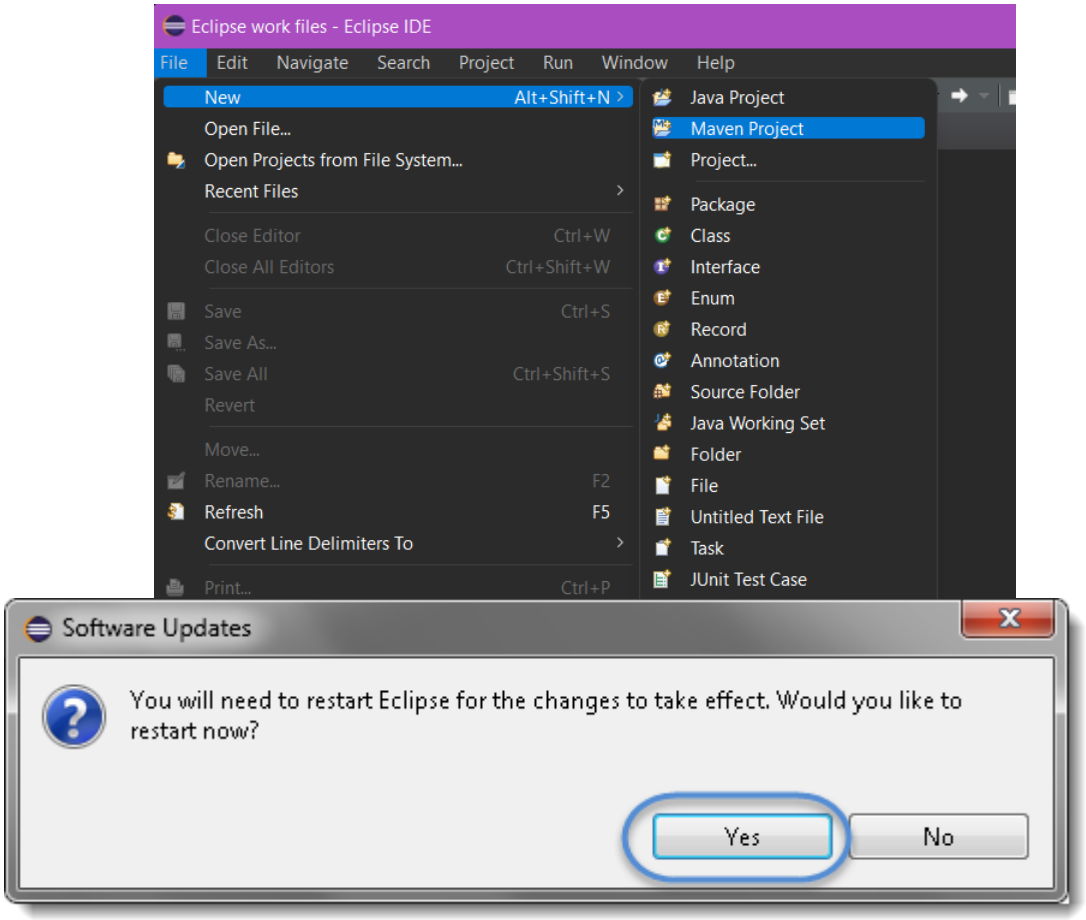
7. Let it install, it will take few seconds to complete.



8. You may or may not encounter a Security warning, if in case you do just click **OK**.



9. You are all done now, just click **Yes**.



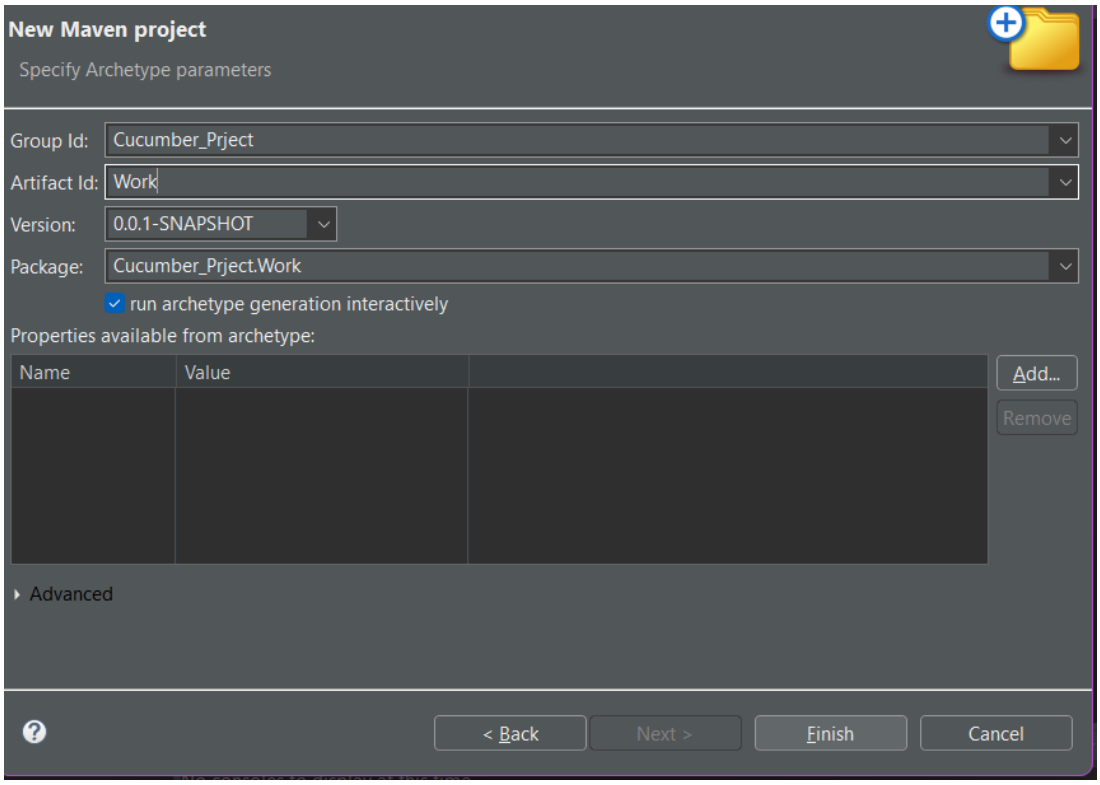
MAVEN

Create a New Maven Project:

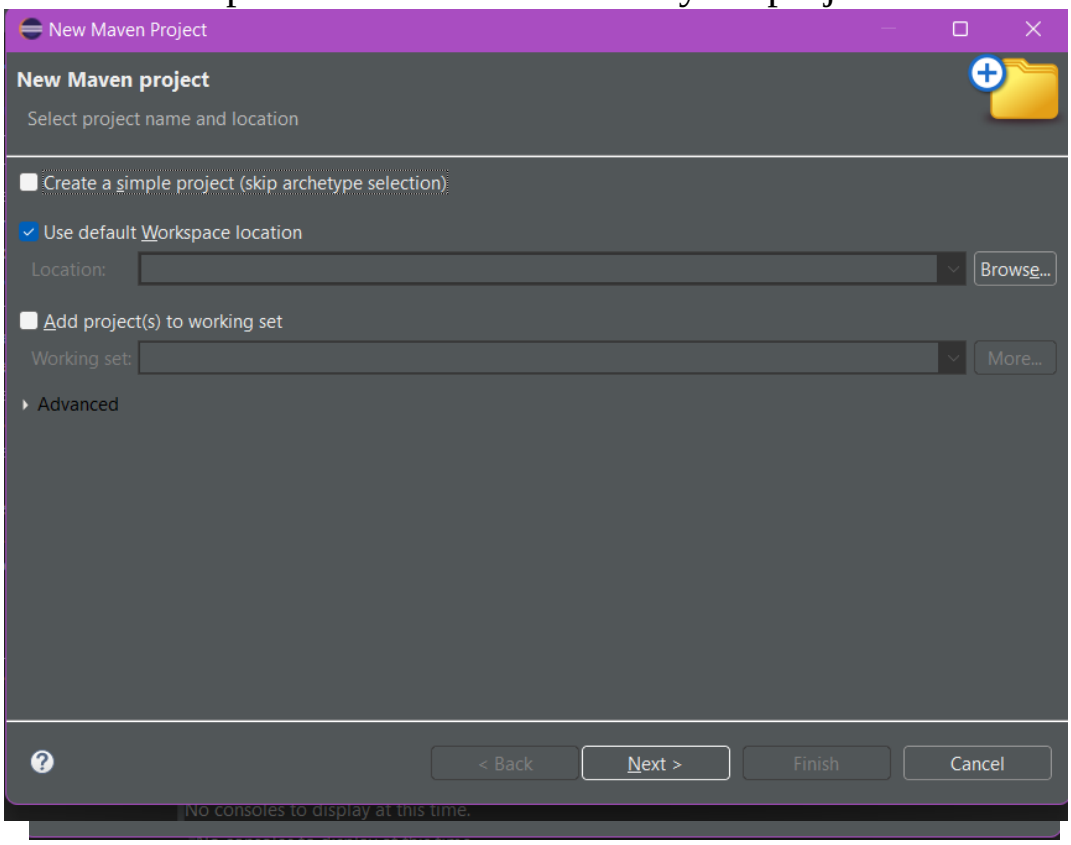
Open Eclipse and go to "File" -> "New" -> "Maven Project

Click Next. (Use Default workspace or you can change the location

Select "Catalog (Internal & choose ID maven- archetype-quickstart)" and click "Next."



Fill in the "Group Id" and "Artifact Id" for your project and click



"Finish."

After Finish, it loading in the console and **asking for Yes?**

```
[INFO] Generating project in Interactive mode
[INFO] Archetype repository not defined. Using the one from [org.apache.mave
[INFO] Using property: groupId = Cucumber_Prject
[INFO] Using property: artifactId = Work
[INFO] Using property: version = 0.0.1-SNAPSHOT
[INFO] Using property: package = Cucumber_Prject.Work
Confirm properties configuration:
groupId: Cucumber_Prject
artifactId: Work
version: 0.0.1-SNAPSHOT
package: Cucumber_Prject.Work
 Y: :
```

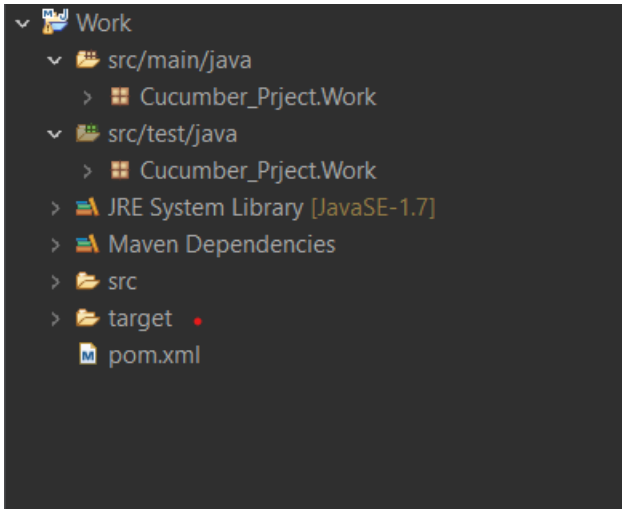
```
Problems  Javadoc  Declaration  Console ×  Progress  TestNG
C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (20-Sep-2023, 10:07:27 am) [pid: 10724]
[INFO] Generating project in Interactive mode
[INFO] Archetype repository not defined. Using the one from [org.apache.maven.archetypes:maven-a
[INFO] Using property: groupId = Cucumber_Prject
[INFO] Using property: artifactId = Work
[INFO] Using property: version = 0.0.1-SNAPSHOT
[INFO] Using property: package = Cucumber_Prject.Work
Confirm properties configuration:
groupId: Cucumber_Prject
artifactId: Work
version: 0.0.1-SNAPSHOT
package: Cucumber_Prject.Work
 Y: : Y
```

Type '**Y**' click enter

Project Build Successful

```
<terminated> C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (13-Sep-2023, 2:50:50 pm) [pid: 25668]
[INFO] Parameter: groupId, Value: Cucuwork
[INFO] Parameter: artifactId, Value: Cucumberwork
[INFO] Parameter: packageName, Value: Cucuwork.Cucumberwork
[INFO] Parameter: version, Value: 0.0.1-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\Users\kavi1\OneDrive\Desktop\Eclipse work files\Cuc
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:40 min
[INFO] Finished at: 2023-09-13T14:52:34+05:30
[INFO] ------------------------------------------------------------------------
```

Maven Project File Structure

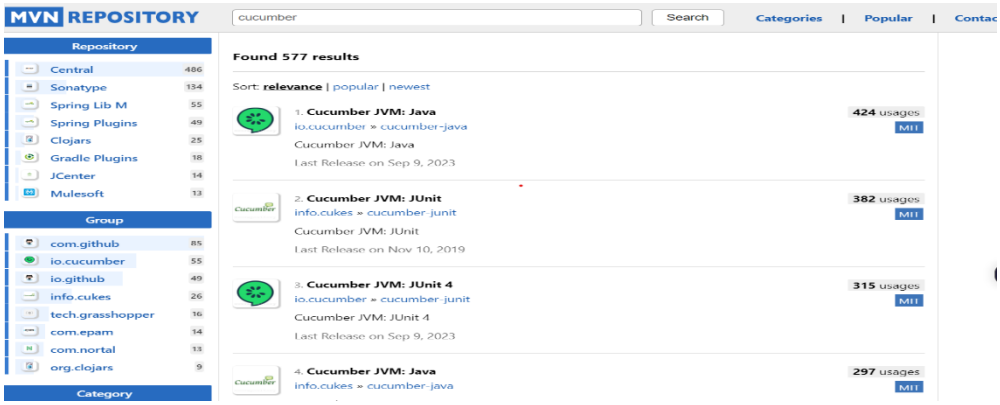


Cucumber Framework

MAVEN DEPENDENCY

Go to Maven Repository -> https://mvnrepository.com/

Search for cucumber framework



Click the dependency you need

Cucumber JVM: Java

Cucumber JVM: Java

License	MIT
Categories	Testing Frameworks & Tools
Tags	testing
Ranking	#1096 in MvnRepository (See Top Artifacts) #48 in Testing Frameworks & Tools
Used By	424 artifacts

Central (112)

	Version	Vulnerabilities	Repository	Usages	Date
7.14.x	7.14.0		Central	22	Sep 09, 2023
7.13.x	7.13.0		Central	37	Jul 07, 2023
7.12.x	7.12.1		Central	29	Jun 02, 2023
	7.12.0		Central	31	Apr 29, 2023
7.11.x	7.11.2		Central	37	Mar 23, 2023
	7.11.1		Central	36	Jan 31, 2023
	7.11.0		Central	38	Jan 12, 2023
7.10.x	7.10.1		Central	20	Dec 16, 2022
	7.10.0		Central	12	Dec 11, 2022
7.9.x	7.9.0		Central	30	Nov 01,

Click the Version based on the latest version or usages.

Copy the content and paste to the POM.xml

Cucumber JVM: Java » 7.13.0

Cucumber JVM: Java

License	MIT
Categories	Testing Frameworks & Tools
Tags	testing
Date	Jul 07, 2023
Files	pom (8 KB) jar (721 KB) View All
Repositories	Central
Ranking	**#1096 in MvnRepository (See Top Artifacts)** **#48 in Testing Frameworks & Tools**
Used By	**424 artifacts**

Note: There is a new version for this artifact

New Version	7.14.0

Maven | Gradle | Gradle (Short) | Gradle (Kotlin) | SBT | Ivy | Grape | Leiningen | Buildr

```
<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-
java -->
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>7.13.0</version>
</dependency>
```

☑ Include comment with link to declaration

Copied to clipboard!

Use same procedure for other dependency like selenium, apache poi,

testNG, etc.,

Add Cucumber Dependencies to the pom.xml:

Open the pom.xml file and add the Cucumber dependencies for Java and JUnit:

```
<dependencies>

    <!-- Selenium WebDriver -->
   <dependency>
   <groupId>org.seleniumhq.selenium</groupId>
   <artifactId>selenium-java</artifactId>
   <version>2.47.1</version>
   </dependency>


    <!-- Cucumber Dependencies -->
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>7.0.0</version> <!-- Use the latest version available -->
</dependency>

<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>7.0.0</version> <!-- Use the same version as cucumber-
java -->
</dependency>


<!-- JUnit Dependency -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version> <!-- Use the latest version available -->
    <scope>test</scope>
</dependency>
```

```
        <dependency>
      <groupId>org.apache.poi</groupId>
      <artifactId>poi</artifactId>
      <version>5.0.0</version> <!-- Use the latest version -->
   </dependency>

     <!-- Apache POI for Excel -->
    <dependency>
      <groupId>org.apache.poi</groupId>
      <artifactId>poi-ooxml</artifactId>
      <version>3.17</version>
    </dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>
   <groupId>org.testng</groupId>
   <artifactId>testng</artifactId>
   <version>7.8.0</version>
   <scope>test</scope>
</dependency>

  </dependencies>
</project>
```

Example

project-root

```
│
├── src
│   ├── main
│   │   └── java
│   │       └── Feature
```

```
│   │         └── fblogin.feature
│   └── test
│       └── java
│           ├── stepdefinitions
│           │   └── StepDefinitions.java
│           └── runners
│               └── TestRunner.java
└── pom.xml
```

- src/main/java/Feature/fblogin.feature: This is where your Cucumber feature file will reside, defining the behavior in a Gherkin syntax.
- src/test/java/stepdefinitions/StepDefinitions.java: This Java class will contain the step definitions that map Gherkin steps to Java code.
- src/test/java/runners/TestRunner.java: This Java class will act as the JUnit runner to execute your Cucumber tests.
- pom.xml: The Maven POM (Project Object Model) file containing project configuration and dependencies.

Fblogin.feature

```
Feature: demo

 Scenario: Login functionality exists

Given I have open the browser

When I open Facebook website

Then Login button should exits
```

- **StepDefinition.java**

```
package facebook;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.WebElement;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.When;
import io.cucumber.java.en.Then;

public class StepDefnition {
  WebDriver driver = null;
        @Given("^I have open the browser$")
  public void openBrowser() {
        System.setProperty("webdriver.chrome.driver",
"path to chrome Driver”);
    driver = new ChromeDriver();
  }
        @When("^I open Facebook website$")
  public void goToFacebook() {
    driver.navigate().to("https://www.facebook.com/");
  }
 @Then("^Login button should exits$")
  ublic void loginButton() {
   if(driver.findElement(By.name("login")).isEnabled())
{
    System.out.println("Test 1 Pass");
    } else {
      System.out.println("Test 1 Fail");
    }
    driver.close();
  }
}
```

- **TestRunner.java**

```
package TestRun;
import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;


        @RunWith(Cucumber.class)
        @CucumberOptions(
                        features                                        =
{"src/main/java/feature/fblogin.feature"},
                        publish=true,
                        glue = {"stepDefinition"}) // StepDefinition
package name
        public class TestRuner {

}
```

- **Links to refer**
- https://www.edureka.co/community/53904/login-test-for-gmail-with-cucumber-and-selenium-webdriver

- https://www.numpyninja.com/post/how-to-read-data-from-excel-sheet-in-bdd-cucumber-framework

HOW TO READ THE EXCEL DATA in cucumber

Create one folder in src/test/resources

Add excel sheet with data.

Feature File

Feature: Gmail Login

```
Scenario: User Login with valid username and password from Excel
  Given I navigate to the Gmail login page
  When I enter Gmail username and password from Excel
  And I click the Gmail login button
  Then I should be logged into Gmail
```

StepDefinition

```
package Excelhandle;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
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.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

import java.io.FileInputStream;
import java.io.IOException;

public class writeDataSteps {

   private WebDriver driver;
   private String username;
   private String password;

   @Given("I navigate to the Gmail login page")
   public void iNavigateToGmailLoginPage() {
```

```
        System.setProperty("webdriver.chrome.driver",      "path      to
chrome");
        driver = new ChromeDriver();
        driver.get("https://mail.google.com/");
    }

    @When("I enter Gmail username and password from Excel")
    public void iEnterGmailUsernameAndPasswordFromExcel() {
        String excelFilePath = "src/test/resources/Exceldata/login.xlsx";
        int sheetIndex = 0; // Assuming data is in the first sheet

        try (FileInputStream fis = new FileInputStream(excelFilePath);
             Workbook workbook = new XSSFWorkbook(fis)) {

            Sheet sheet = workbook.getSheetAt(sheetIndex);
            Row row = sheet.getRow(1); // Assuming data is in the first
row

            username = row.getCell(0).getStringCellValue();
            password = row.getCell(1).getStringCellValue();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @When("I click the Gmail login button")
    public void iClickGmailLoginButton() {
        WebElement                    emailField                    =
driver.findElement(By.id("identifierId"));
        emailField.sendKeys(username);

        WebElement                    nextButton                    =
driver.findElement(By.id("identifierNext"));
        nextButton.click();
    }
```

```
    @Then("I should be logged into Gmail")
    public void iShouldBeLoggedInToGmail() {
        System.out.println("Move to password field");
        WebDriverWait wait = new WebDriverWait(driver, 30);
        WebElement passwordInput =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpat
h("//*[@id =\"password\"]/div[1]/div / div[1]/input")));
        WebElement passwordField =
driver.findElement(By.xpath("//*[@id =\"password\"]/div[1]/div /
div[1]/input"));
        passwordField.sendKeys(password);

        WebElement nextButton =
driver.findElement(By.id("passwordNext"));
        nextButton.click();
    }
}
```

RunnerClass

```
package Excelhandle;

import org.junit.runner.RunWith;
import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)
@CucumberOptions(
    features = "src/main/java/Featurefiles/write_data.feature",
    glue = {"Excelhandle"},
    publish=true,
    plugin = {"pretty", "html:target/cucumber-reports"}
)
public class writeDataRunner {
}
```

Output:

```
Problems  Javadoc  Declaration  Console ×  Progress  Results of running class draganddrop
<terminated> writeDataRunner (1) [JUnit] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe  (20-Sep-2023, 1:12:49 pm – 1:13:05 pm) [pid: 22968]

Scenario: User Login with valid username and password from Excel # src/main/java/Featurefiles/write_data.feature:3
Starting ChromeDriver 117.0.5938.62 (25a7172909a4cba7355365cf424d7d7eb35231f4-refs/branch-heads/5938@{#1146}) on port 28480
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
  Given I navigate to the Gmail login page                    # Excelhandle.writeDataSteps.iNavigateToGmailLoginPage()
ERROR StatusLogger Log4j2 could not find a logging implementation. Please add log4j-core to the classpath. Using SimpleLogger to log to the conso
  When I enter Gmail username and password from Excel         # Excelhandle.writeDataSteps.iEnterGmailUsernameAndPasswordFromExcel()
  And I click the Gmail login button                          # Excelhandle.writeDataSteps.iClickGmailLoginButton()
Move to password field
  Then I should be logged into Gmail                          # Excelhandle.writeDataSteps.iShouldBeLoggedInToGmail()

  View your Cucumber Report at:
  https://reports.cucumber.io/reports/338bb03e-7733-42b4-9e76-c49007d32ee7

  This report will self-destruct in 24h.
  Keep reports forever: https://reports.cucumber.io/profile
```

Cucumber report