## Feature files-

We can have as many feature files as needed.

Feature files created using gherkin keyword. Gherkin is domain specific language also known as DSL.

One feature file can have “n” number of features.

One feature file can have multiple scenarios and multiple examples.

Feature file will have scenarios -> scenarios will have steps -> steps have to be written with gherkin keywords like given, when, then etc

Based on this feature file, developers will start tdd approach, qa’s will start with automation etc.

## Step definition-

Any programming language can be used.

Here we create classes like “step1.java” and so on.

Every annotation from feature file will have a corresponding method.

Every method is known as step.

Step definitions also interact with utility classes like dbutil, seleniumutil etc.

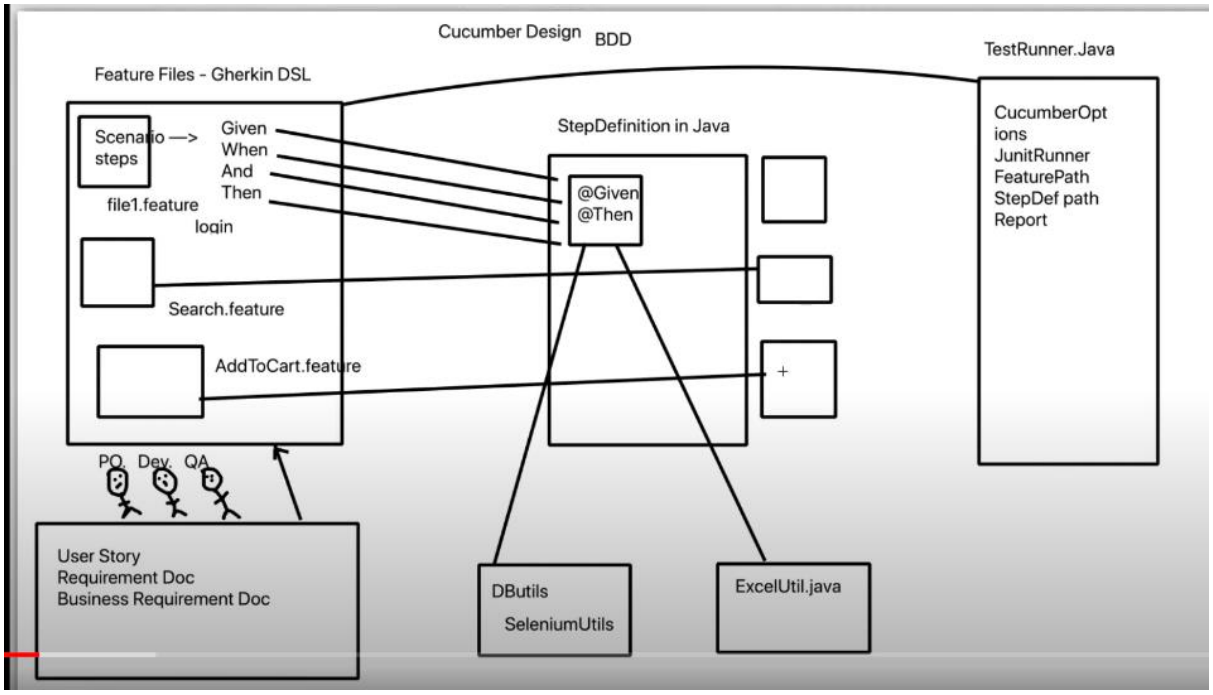
We can have “n” number of step definition files for different features like login, search, add to cart etc and map them to the respective feature file.

## To run the code-

We can use individual feature files or use runner class.

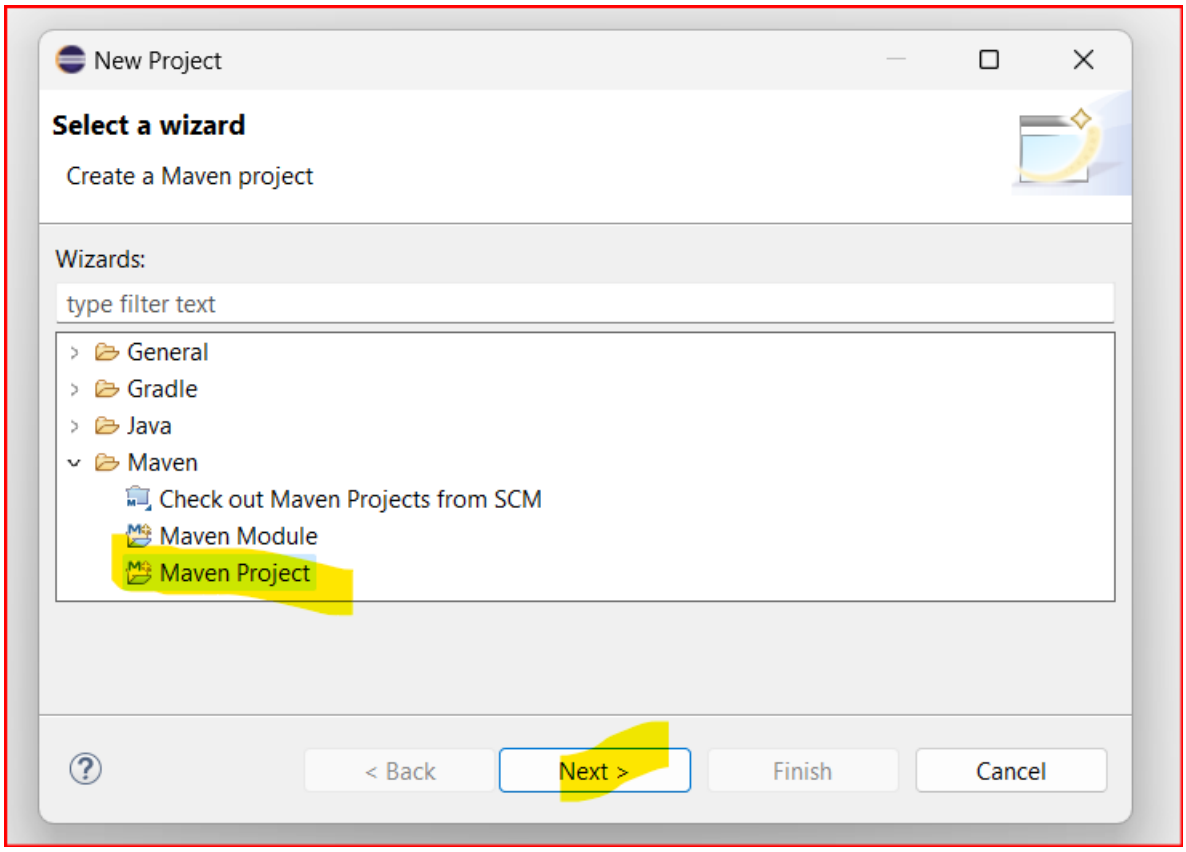
## Test runner –

Will interact with feature file. Runner -> Feature -> step definition.

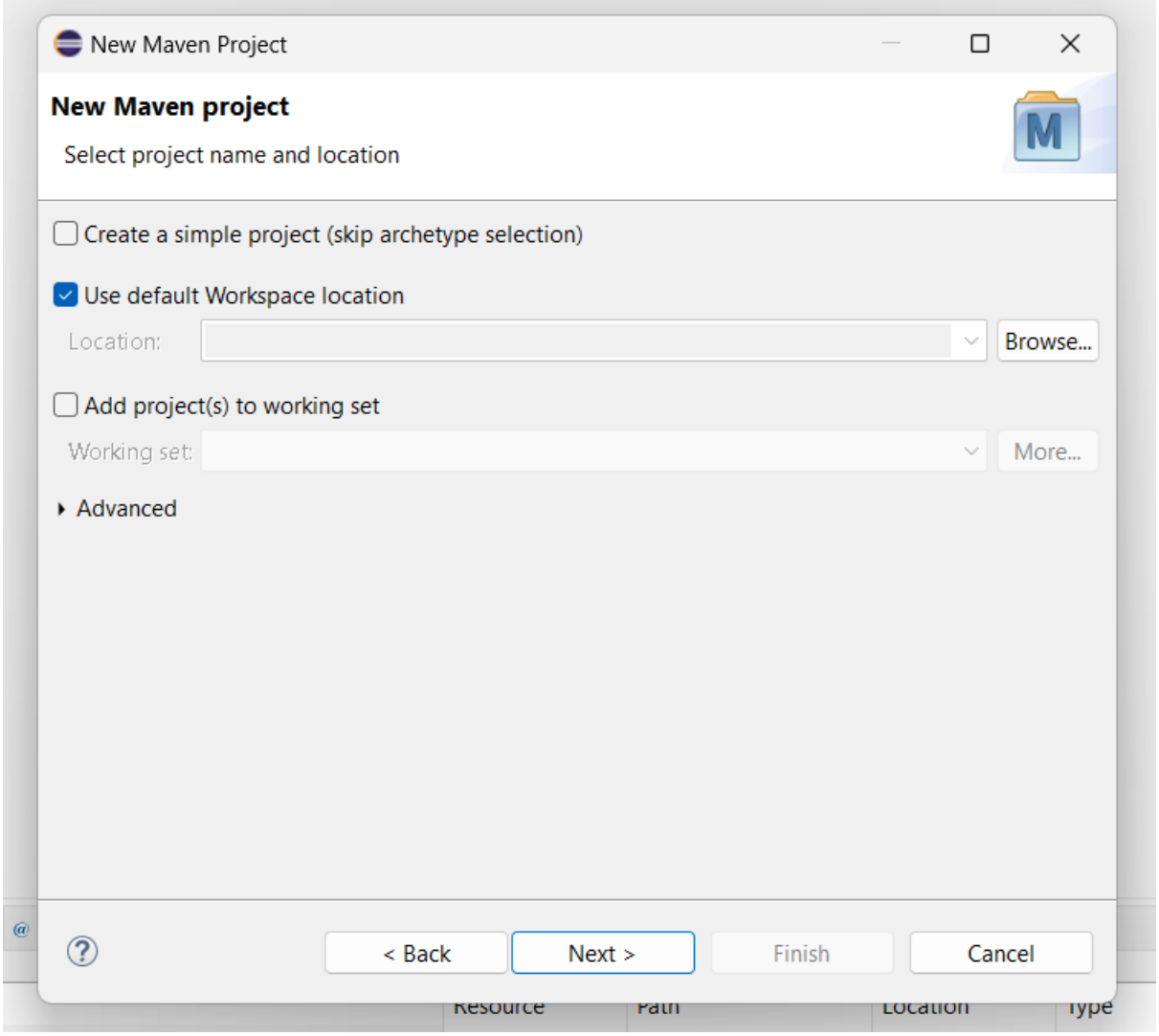



## Let's start cucumber project-

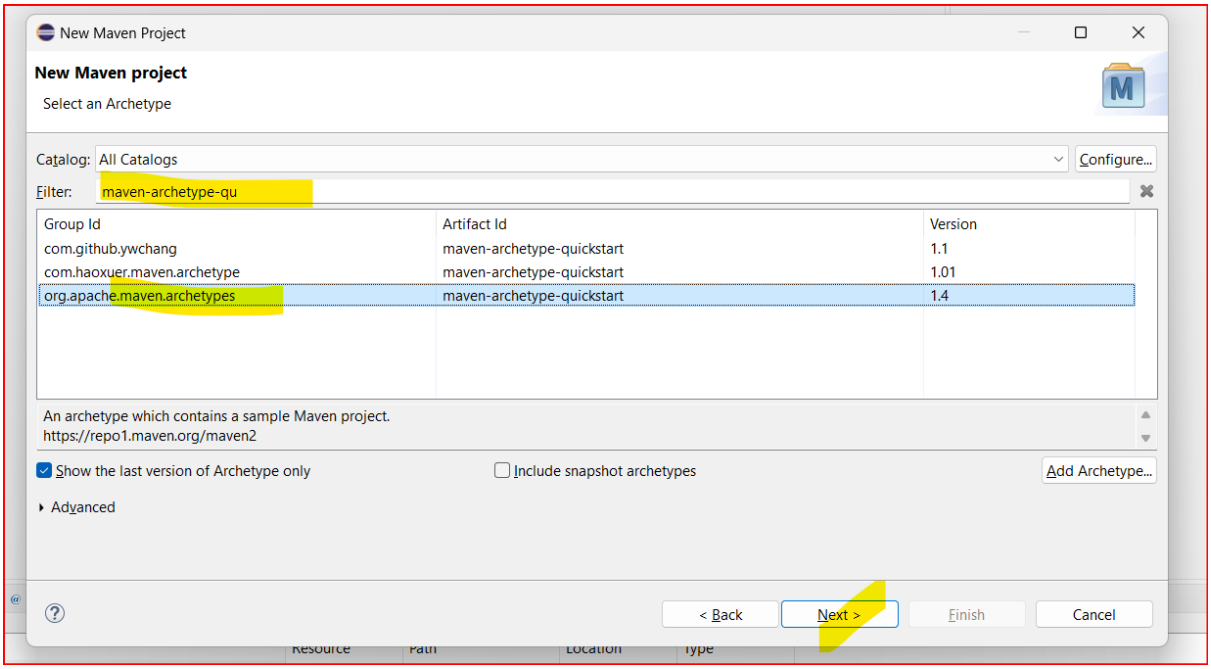
Maven project creation:



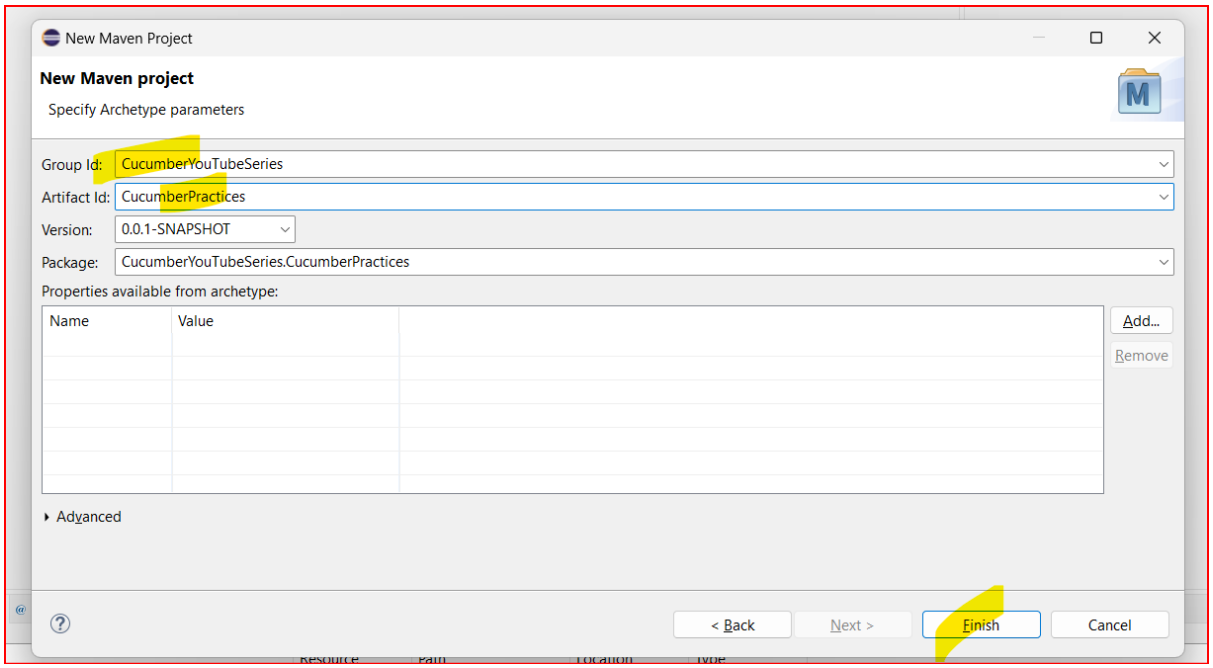

Click next:



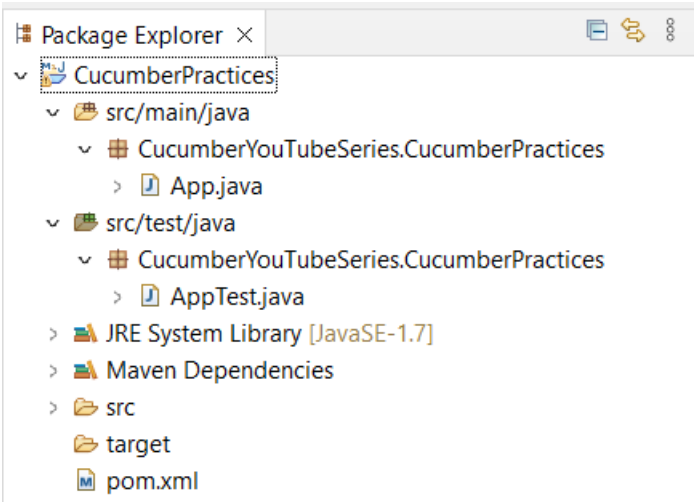

Search for maven plugin:



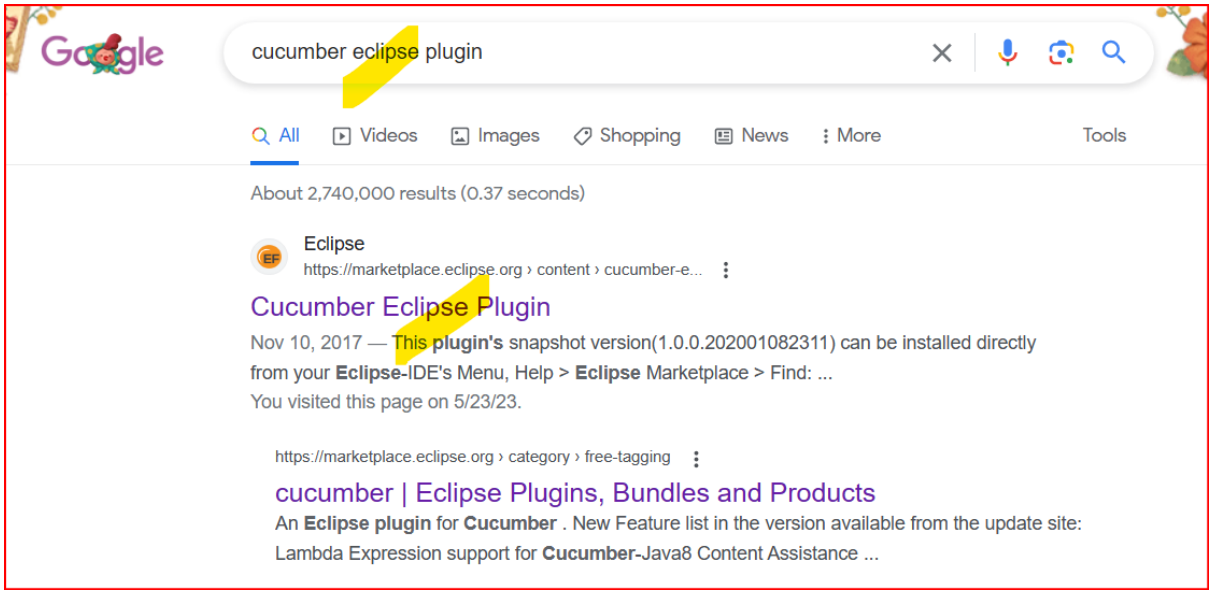
Give group id, artifact id:



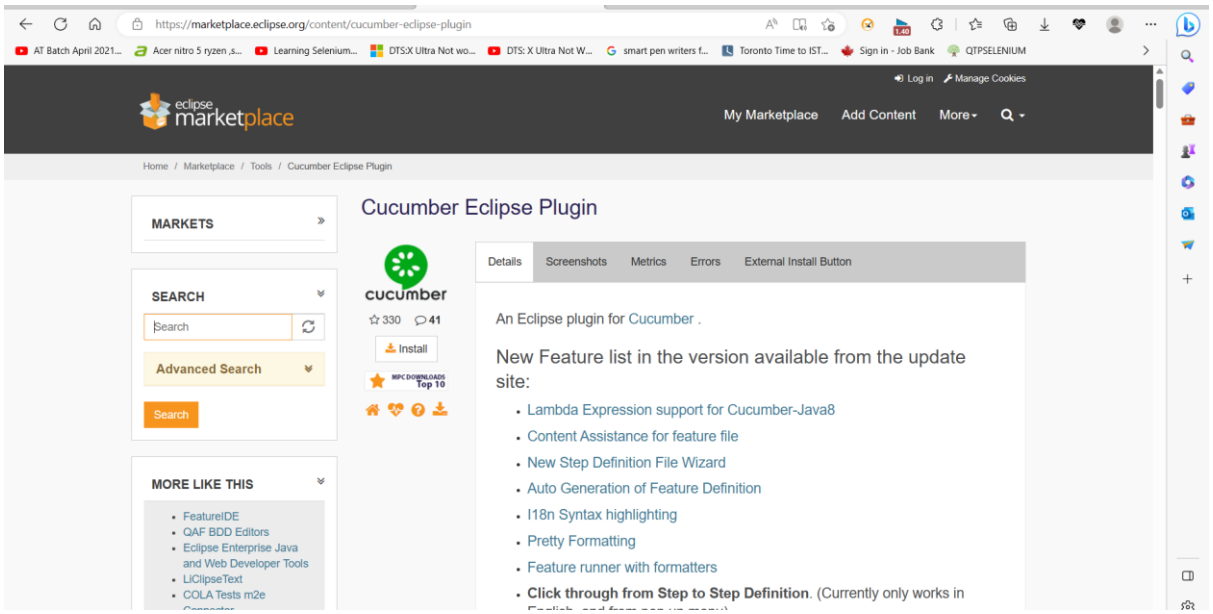

Base project structure once created:




Download the below mentioned plugin to make the code colourful and readable:




Lots of good features



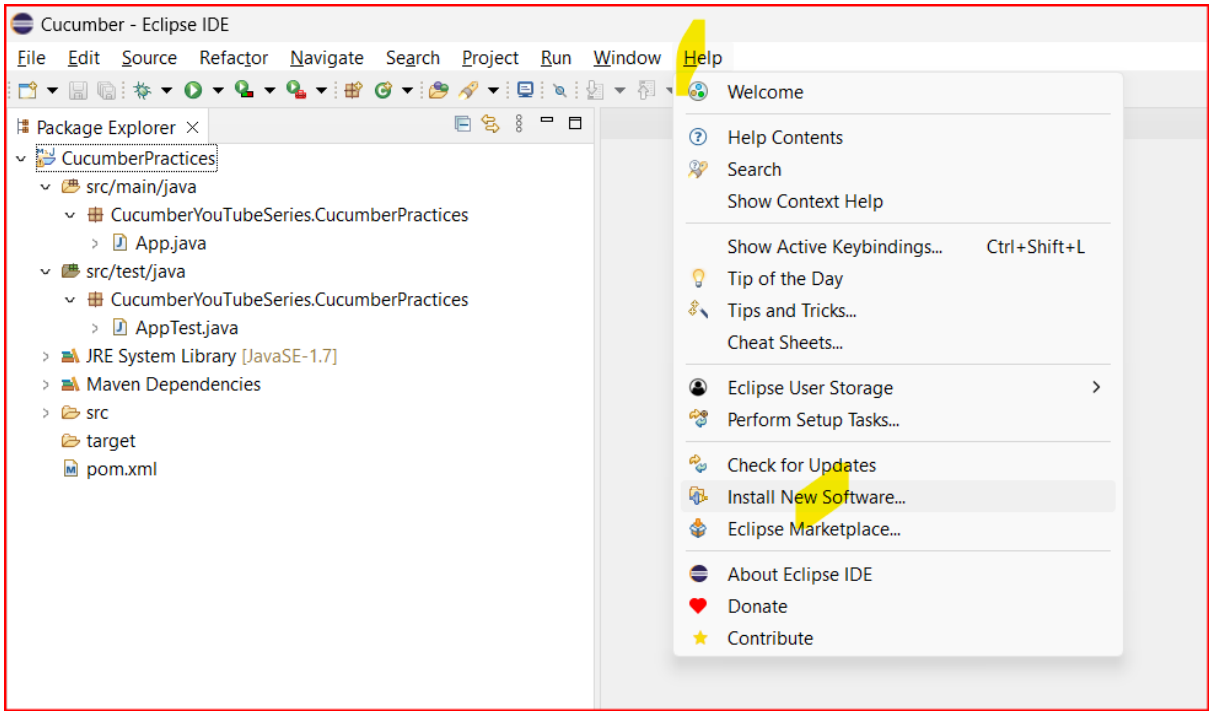

One way to install is drag and drop into eclipse.

But we will use the below link-

[Cucumber-Eclipse > Update Site]

Copy the browser url.

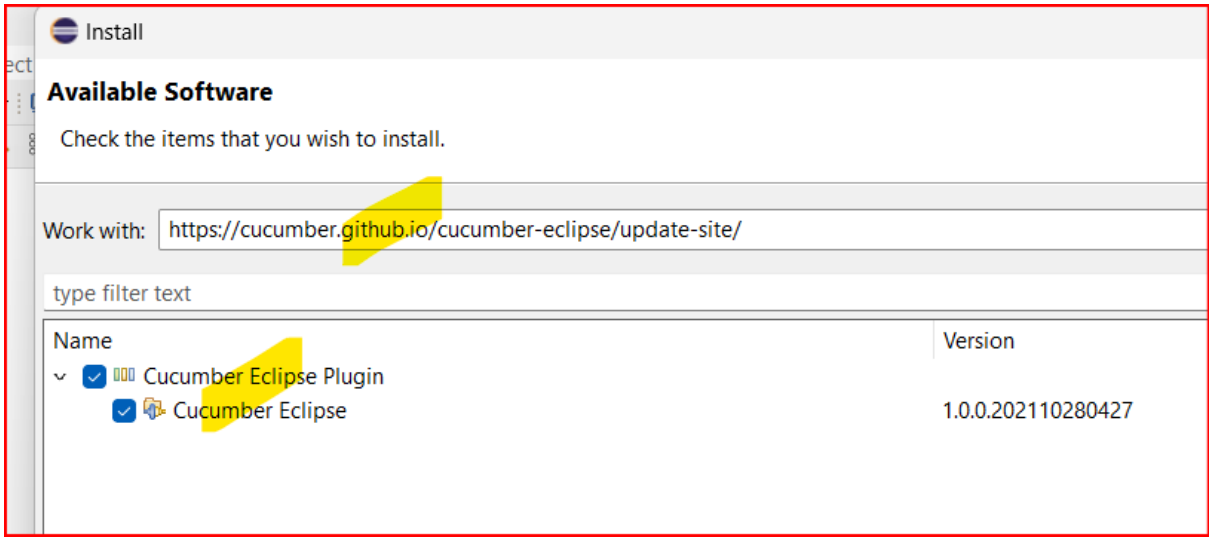
Open eclipse and go to the below location:




Enter the link.

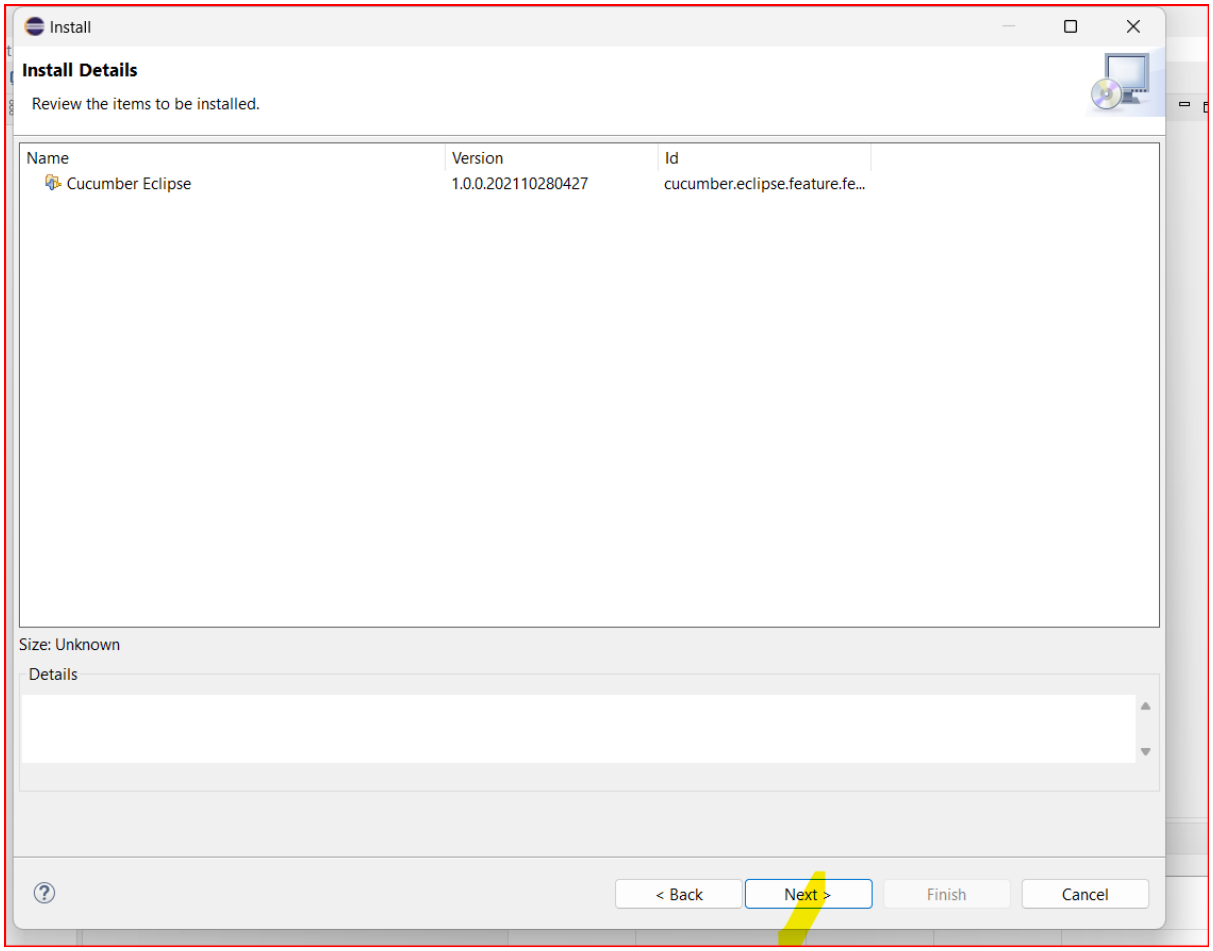
Hit enter.

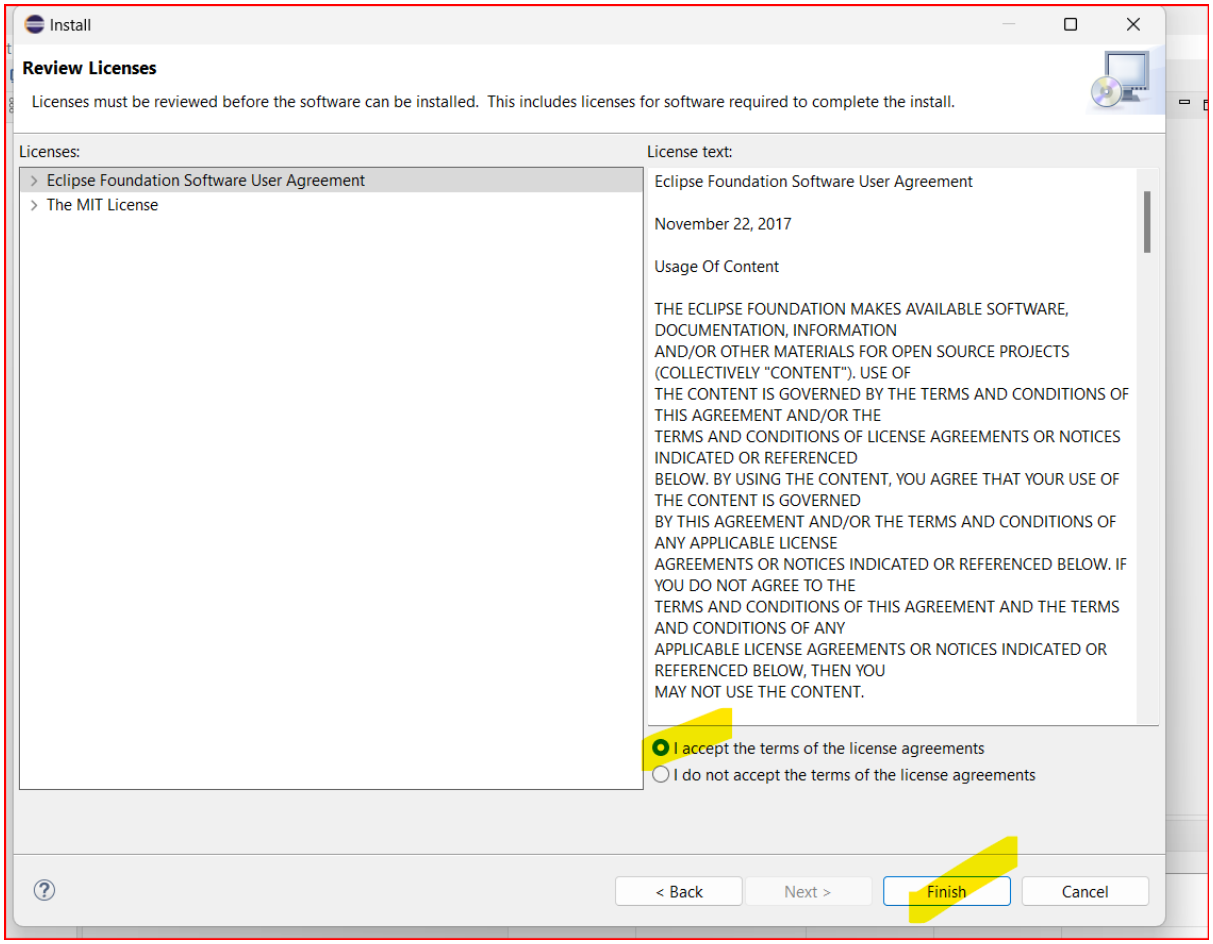
Select all the checkboxes inside Name field.



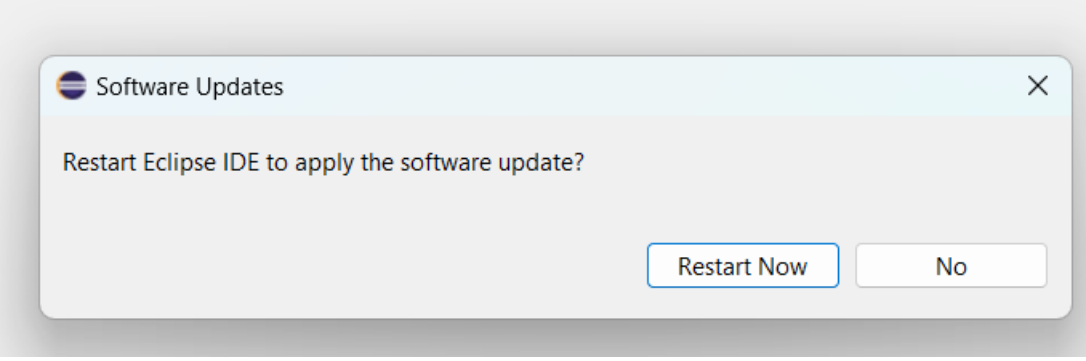

Click next.

7

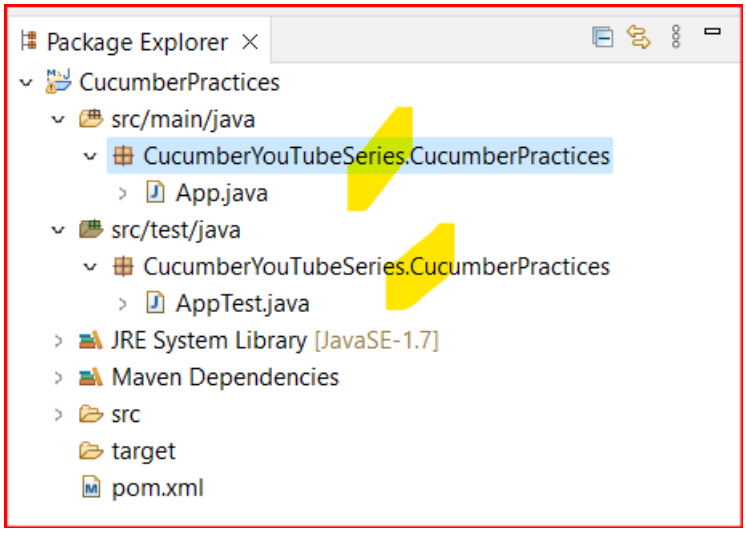

Install
Install Details
Review the items to be installed.
Name
Version
Id
Cucumber Eclipse
1.0.0.202110280427
cucumber.eclipse.feature.fe...
Size: Unknown
Details
< Back
Next >
Finish
Cancel




You may have to restart eclipse for it to work.

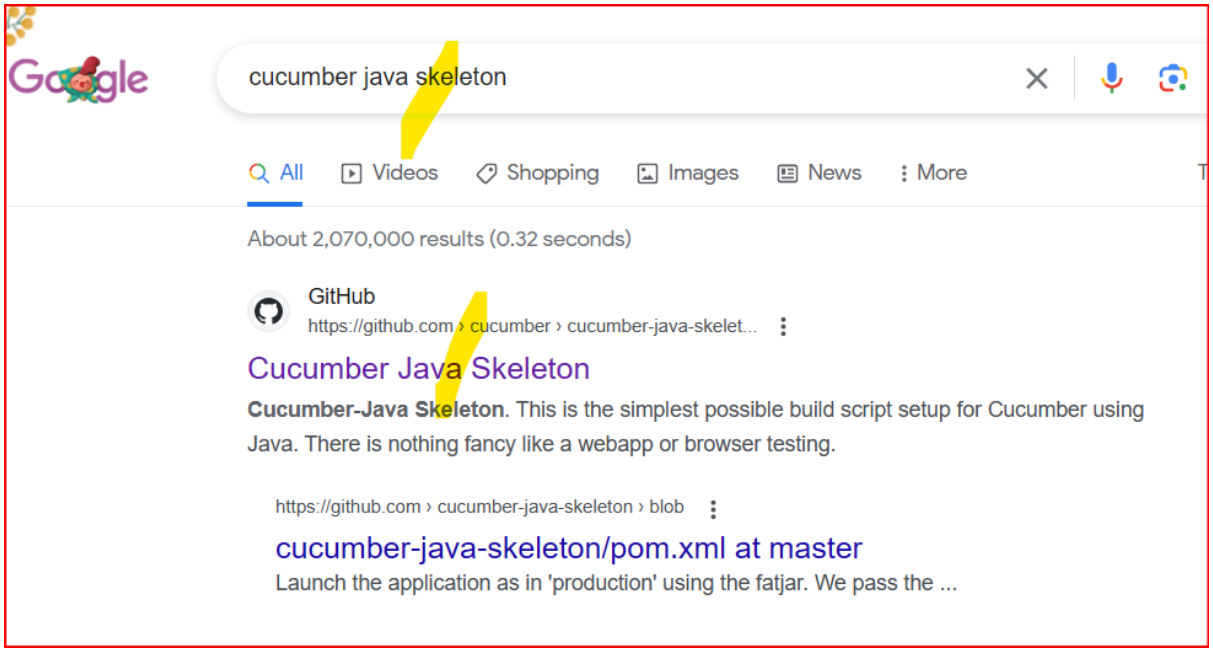



Delete the dummy packages:

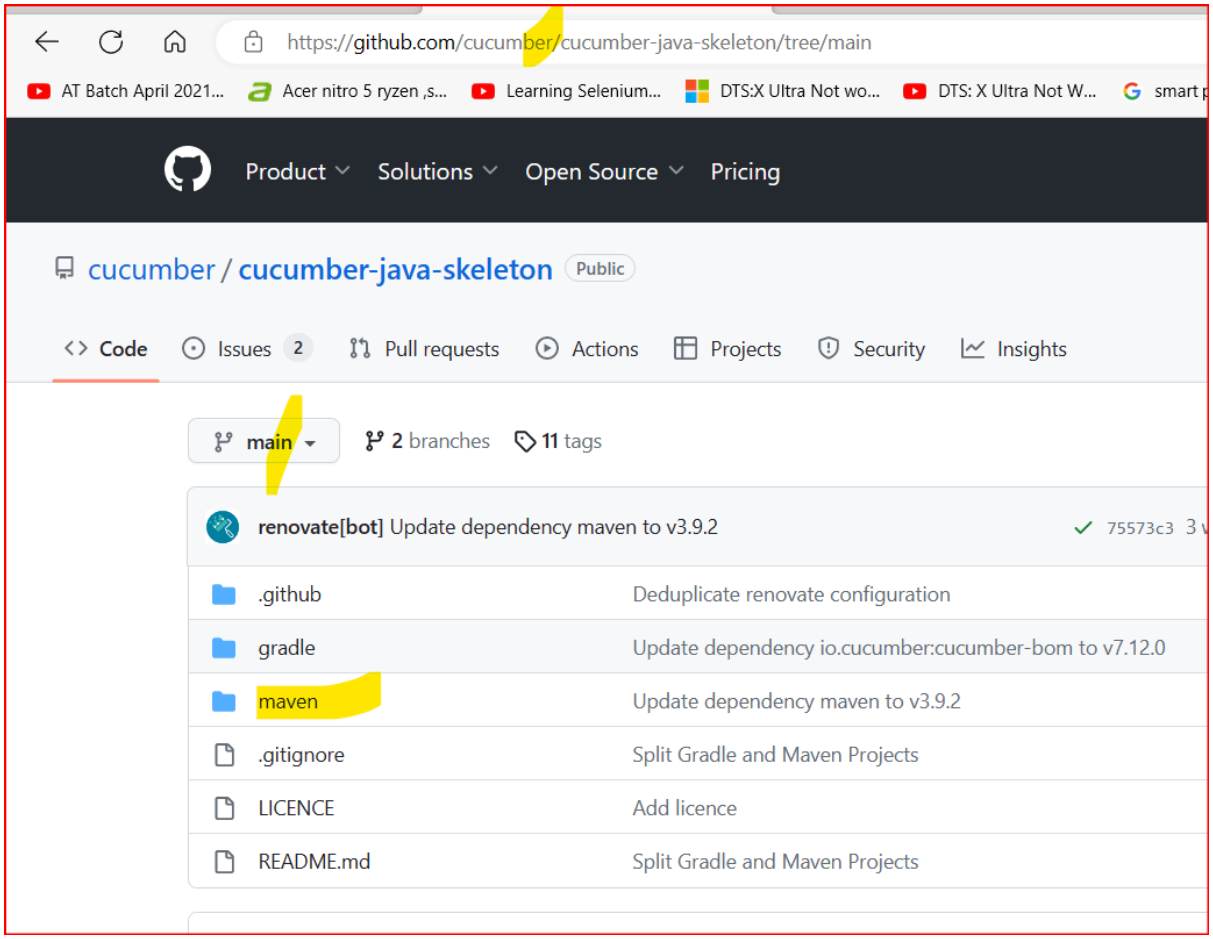



## Let's add the required dependencies for project-

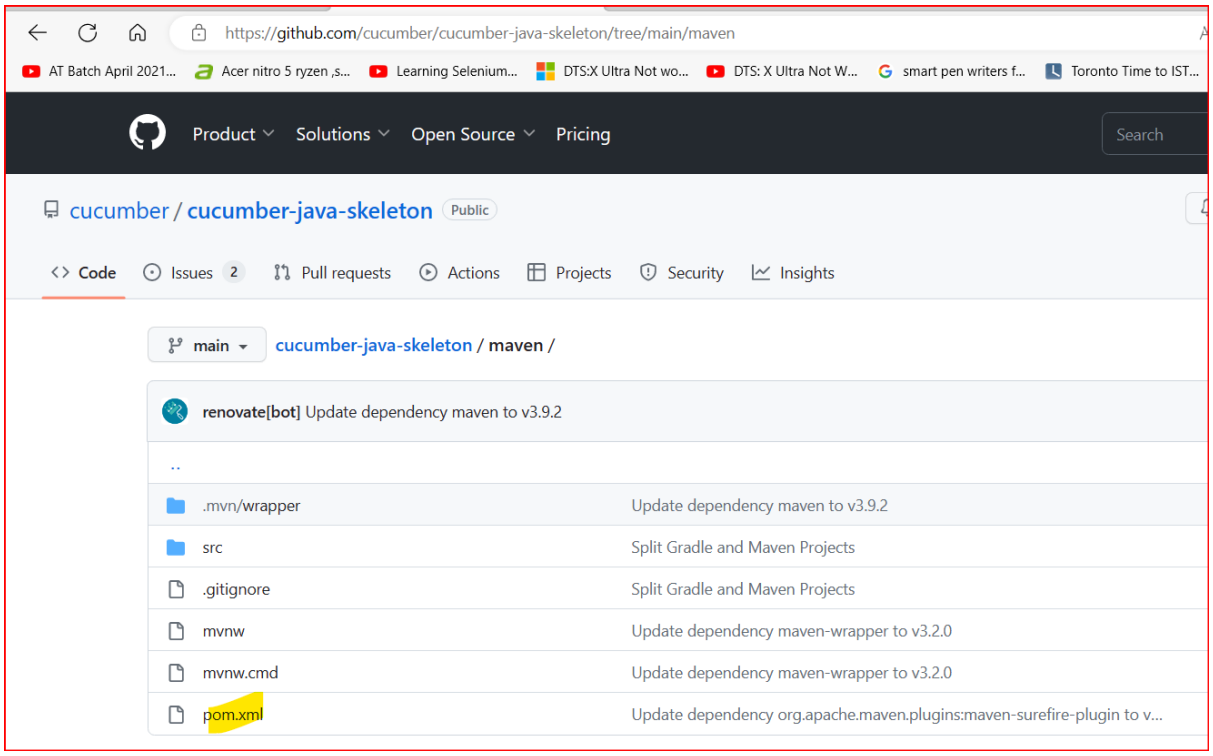
Search in google:




Click maven:




Click pom.xml:




Copy the tags and values inside properties:

```
10
11      <properties>
12          <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
13          <java.version>1.8</java.version>
14      </properties>
```

Open your pom and remove the existing property from properties.

Paste the new one.

Am adding as the one Naveen added as there seems to be some changes.

Added as per what Naveen added in his code:

```
15
16      <properties>
17          <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
18          <java.version>1.8</java.version>
19          <junit.version>4.13.1</junit.version>
20          <cucumber.version>6.9.0</cucumber.version>
21          <maven.compiler.version>3.8.1</maven.compiler.version>
22          <maven.surefire.version>2.22.2</maven.surefire.version>
23      </properties>
24
```

Then go to dependency:

We only need the three dependencies below:

Note- the latest code in link has different types of dependencies due to changes in cucumber.


Copy them.

Remove existing dependency from pom and paste the copied one.

```
    <dependencies>
        <dependency>
            <groupId>io.cucumber</groupId>
            <artifactId>cucumber-java</artifactId>
            <version>${cucumber.version}</version>
            <scope>test</scope>
        </dependency>

        <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -->
        <dependency>
            <groupId>io.cucumber</groupId>
            <artifactId>cucumber-junit</artifactId>
            <version>${cucumber.version}</version>
            <scope>test</scope>
        </dependency>

        <!-- https://mvnrepository.com/artifact/junit/junit -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

Copy the plugins from build section.

Taken screenshot of Naveen's computer.

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>${maven.compiler.version}</version>
                <configuration>
                    <encoding>UTF-8</encoding>
                    <source>${java.version}</source>
                    <target>${java.version}</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>${maven.surefire.version}</version>
            </plugin>
        </plugins>
    </build>
</project>
```

Remove existing build from pom.

Paste the build here.

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>${maven.compiler.version}</version>
                <configuration>
                    <encoding>UTF-8</encoding>
                    <source>${java.version}</source>
                    <target>${java.version}</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>${maven.surefire.version}</version>
            </plugin>
        </plugins>
    </build>
</project>
```

This is how we define variables in xml.

These variables are referring to the value stored in them.

Cucumber version in dependencies is coming from cucumber version in properties.

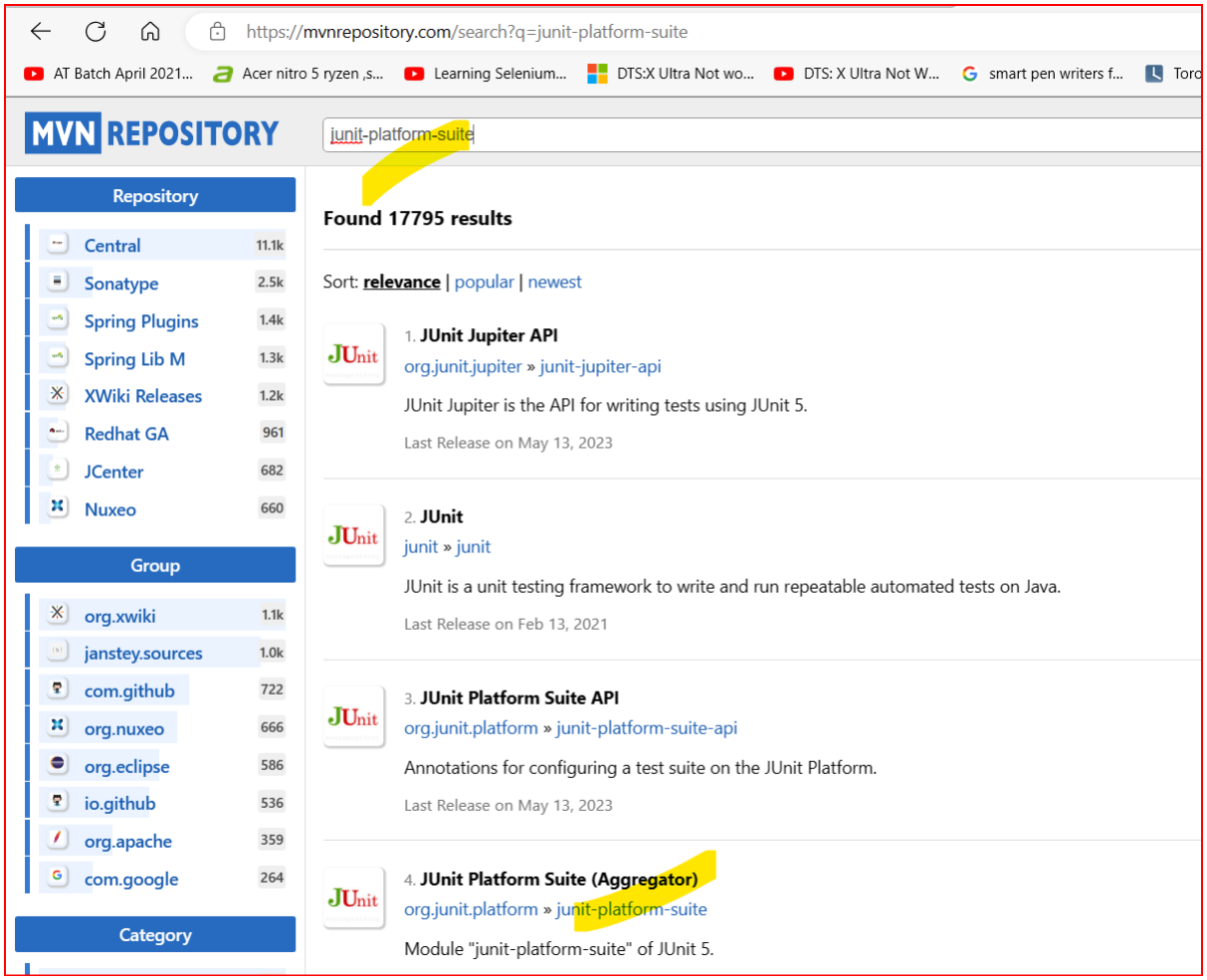
```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      <java.version>1.8</java.version>
      <junit.version>4.13.1</junit.version>
      <cucumber.version>6.9.0</cucumber.version>
      <maven.compiler.version>3.8.1</maven.compiler.version>
      <maven.surefire.version>2.22.2</maven.surefire.version>
</properties>

<dependencies>
  <dependency>
          <groupId>io.cucumber</groupId>
          <artifactId>cucumber-java</artifactId>
          <version>${cucumber.version}</version>
          <scope>test</scope>
      </dependency>
```

Note-

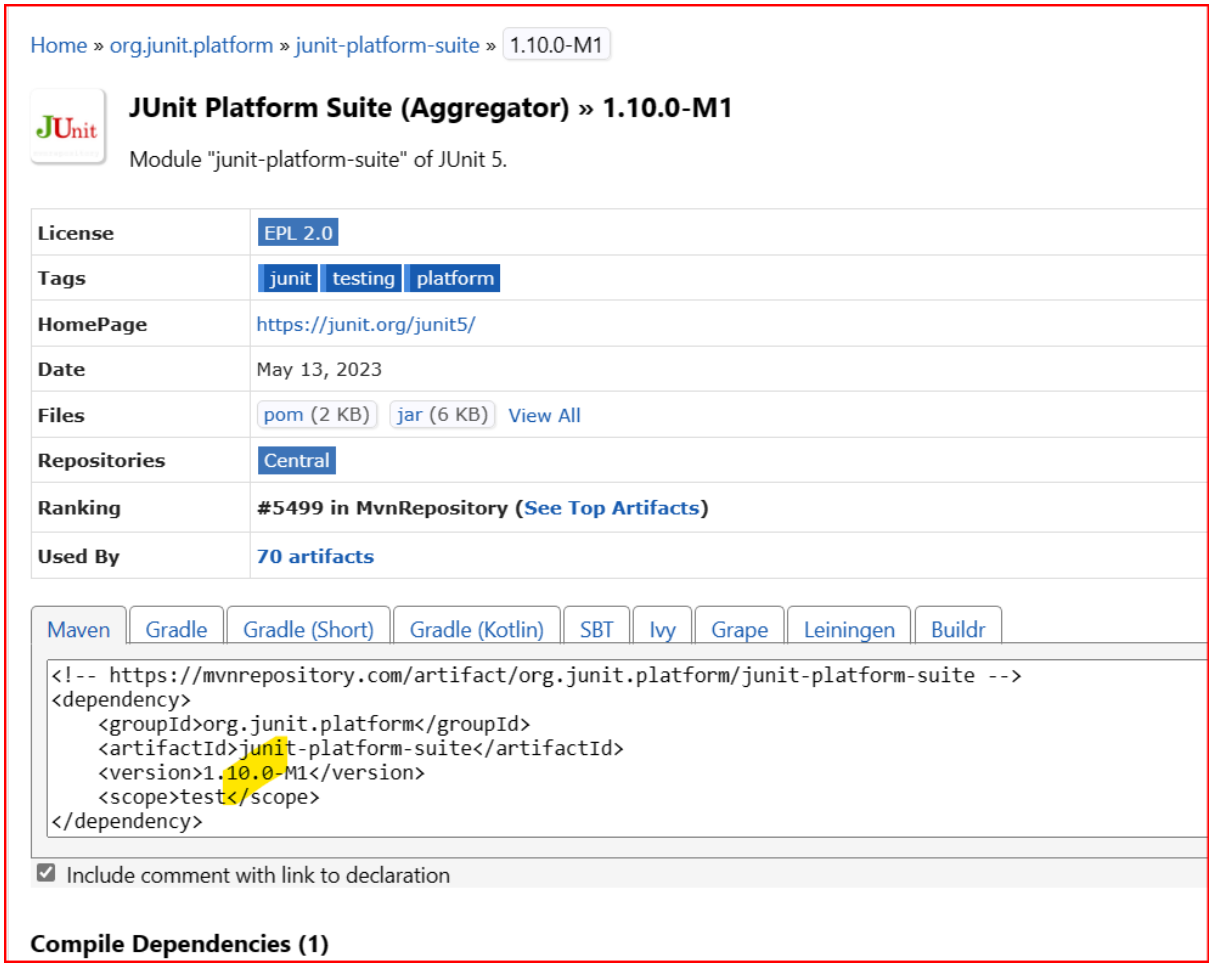
When you build first time it may throw errors for version number not mentioned.

So, what you do is, go to mvn repository and search using artifact id.

Example,

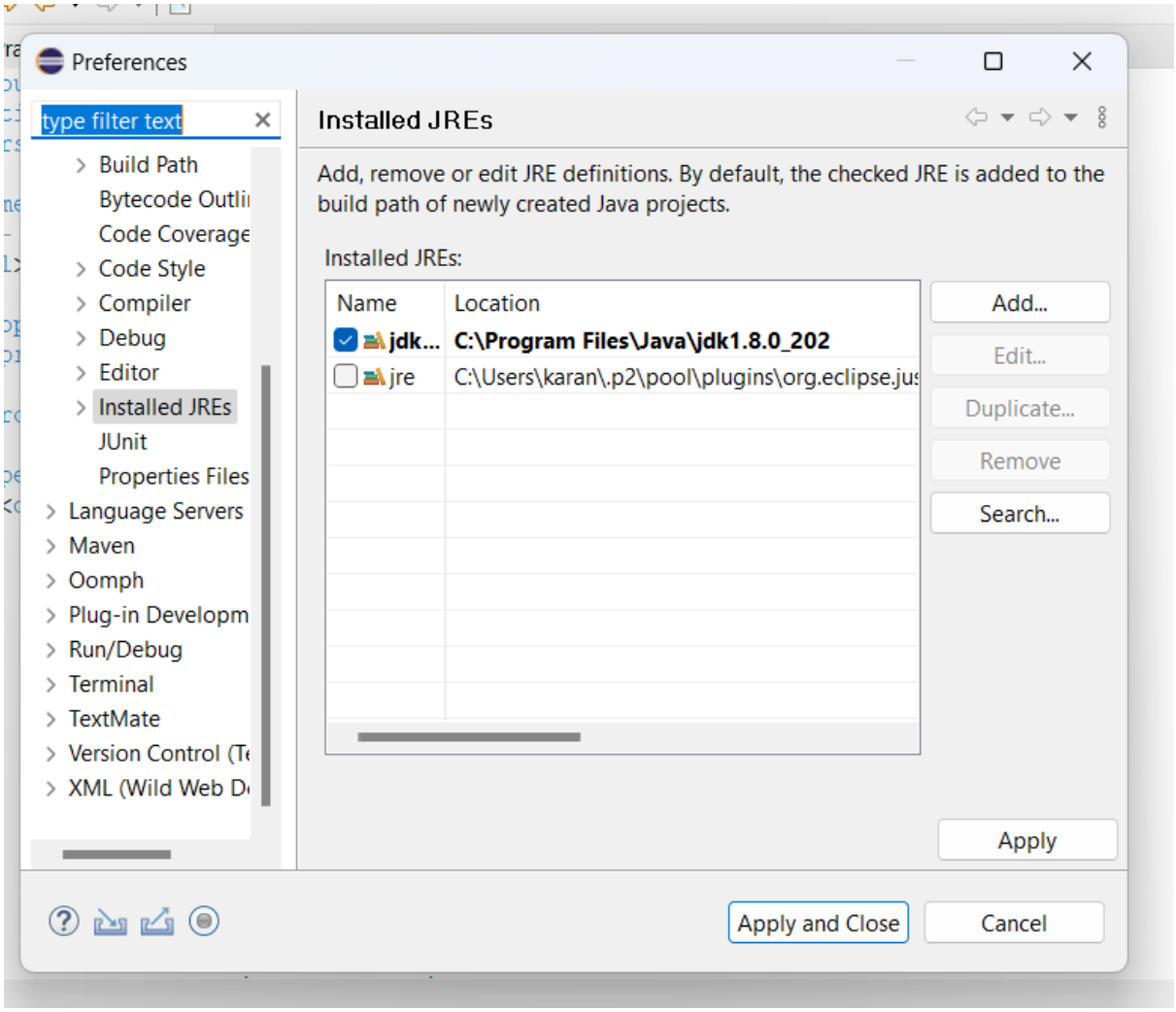



And get the version number:

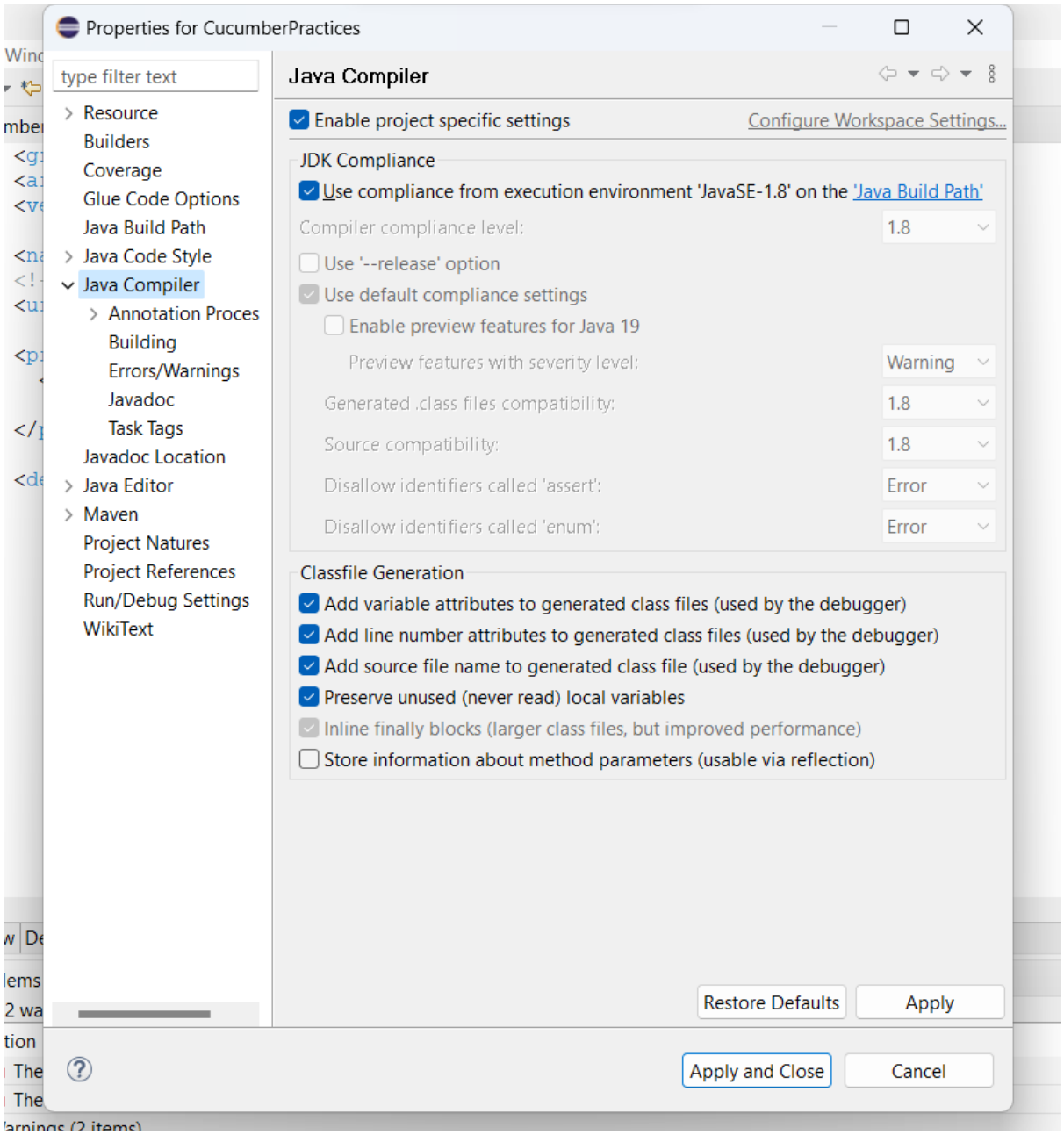



How to resolve this warning - “Description Resource Path Location Type The compiler compliance specified is 1.8 but a JRE 17 is used CucumberPractices Compiler Compliance JRE Compiler Compliance Problem”

Windows -> preferences -> java -> installed jres -> add the jdk folder -> apply -> apply and close



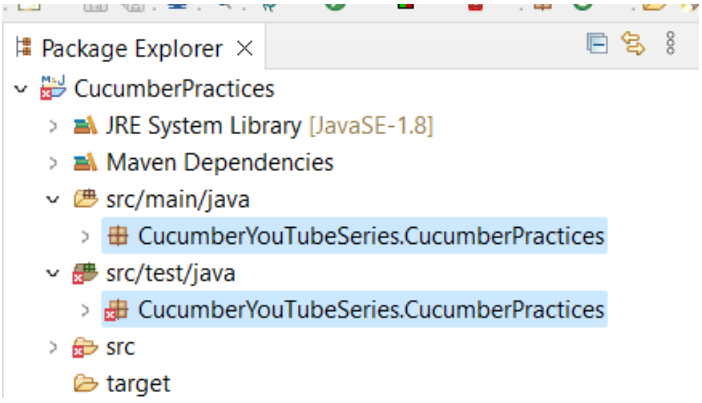

Right click on project -> properties -> java compiler -> and change the java compliance.




See in pom if the version matches.

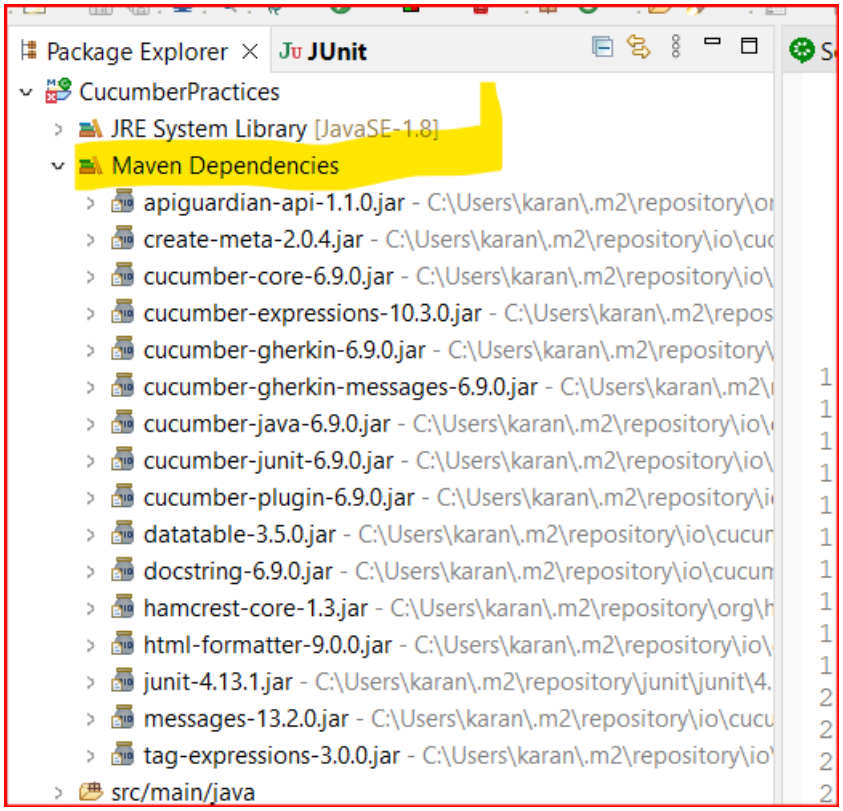
```
14
15   <properties>
16     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
17         <java.version>1.8</java.version>
18   </properties>
```

Remove the dummy packages:




These are created by default.

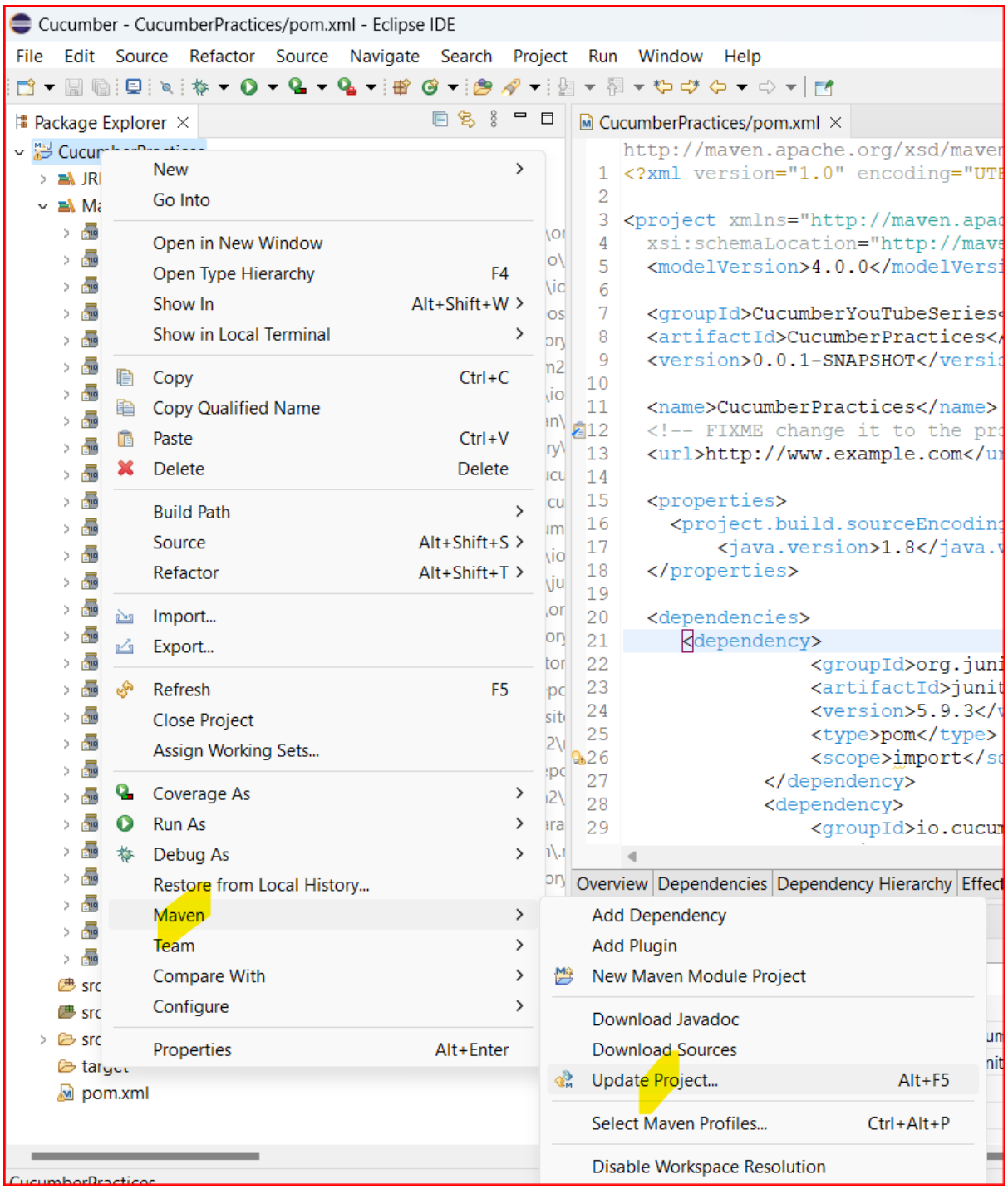
Expand maven dependencies and we can see cucumber present:

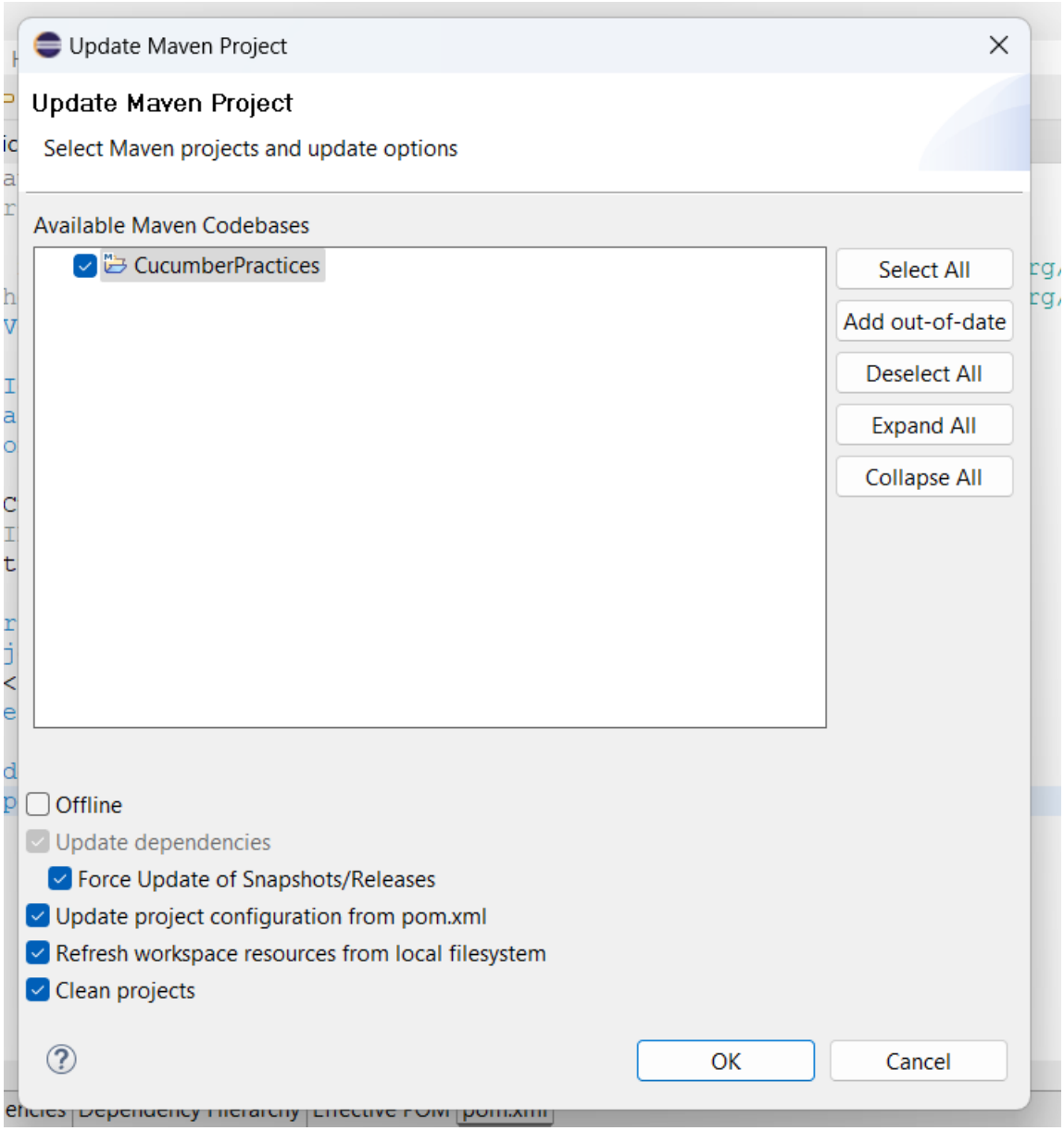



Build again the project:

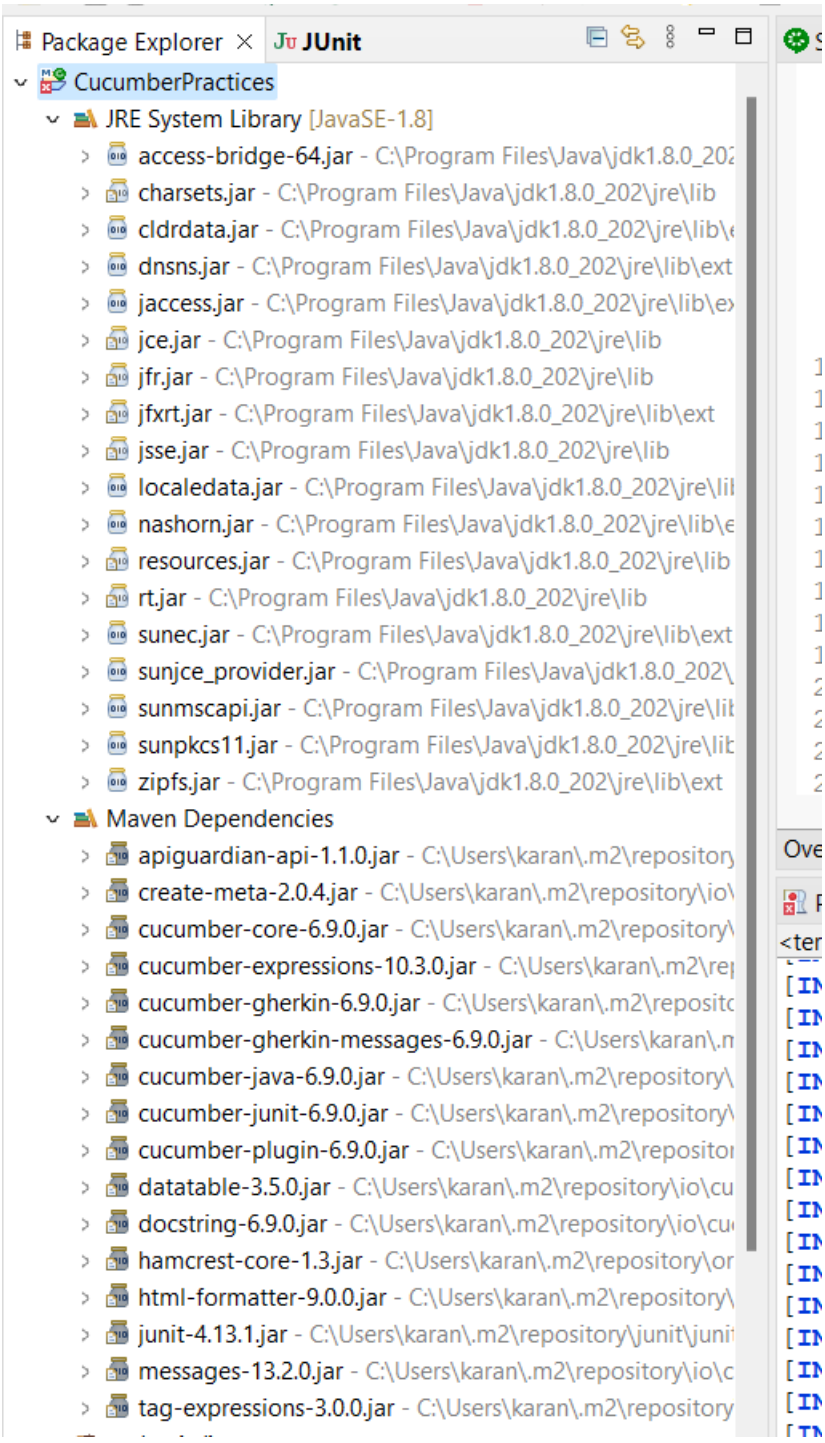
Right click -> maven -> update project.

Select force update and click ok.


Cucumber - CucumberPractices/pom.xml - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer
CucumberPractices/pom.xml
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Show in Local Terminal
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Assign Working Sets...
Coverage As
Run As
Debug As
Restore from Local History...
Maven
Team
Compare With
Configure
Properties Alt+Enter
pom.xml
Add Dependency
Add Plugin
New Maven Module Project
Download Javadoc
Download Sources
Update Project... Alt+F5
Select Maven Profiles... Ctrl+Alt+P
Disable Workspace Resolution
Overview
Dependencies
Dependency Hierarchy

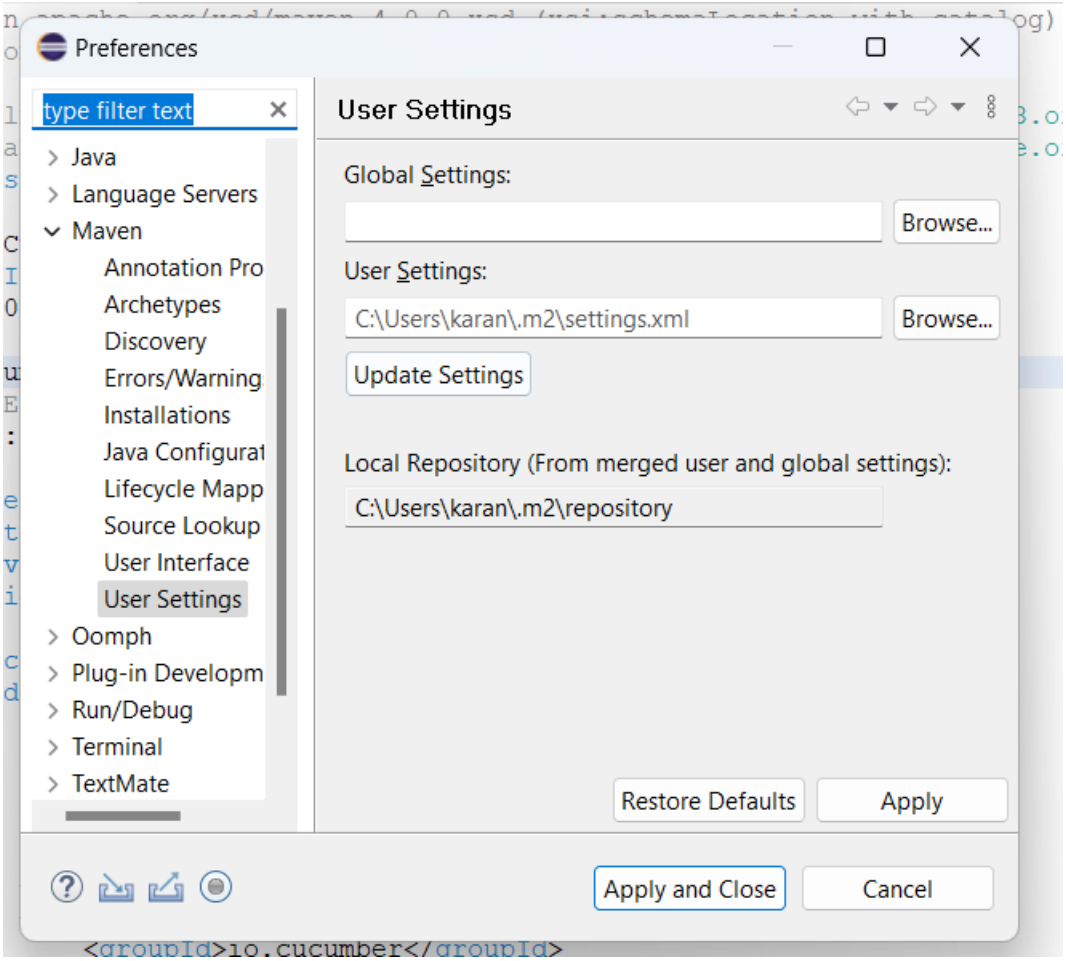



You can see the project being built in bottom right corner and all latest dependencies updated in jre and maven.




## Where is .m2 located in eclipse-

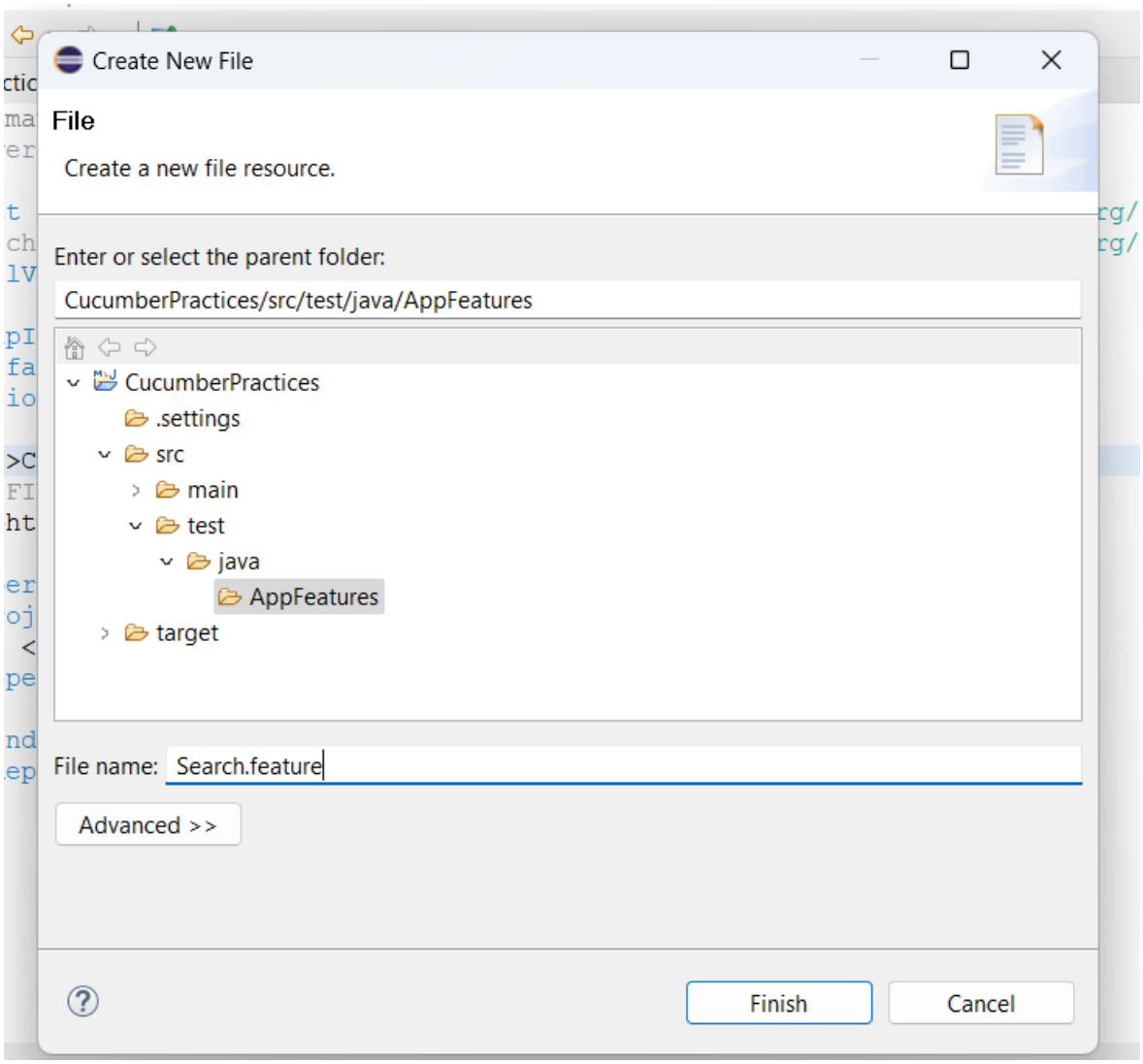
Windows -> preferences -> maven -> user settings



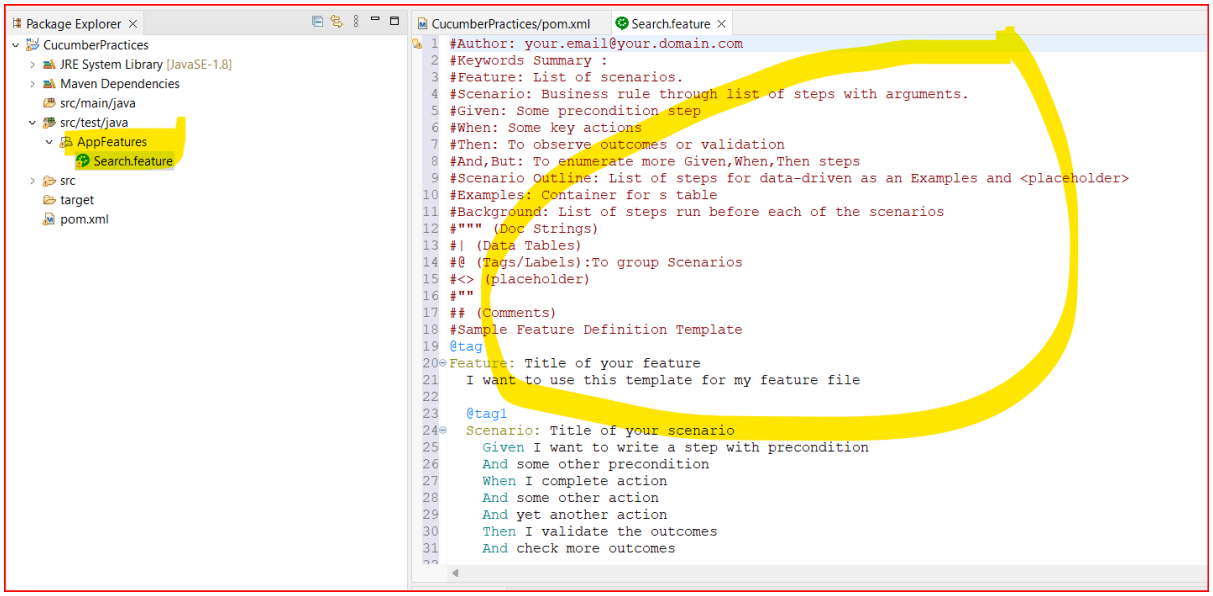

Create new package under src/test/java called as AppFeatures.

Create a feature file.

Extension has to be given as “.feature”.

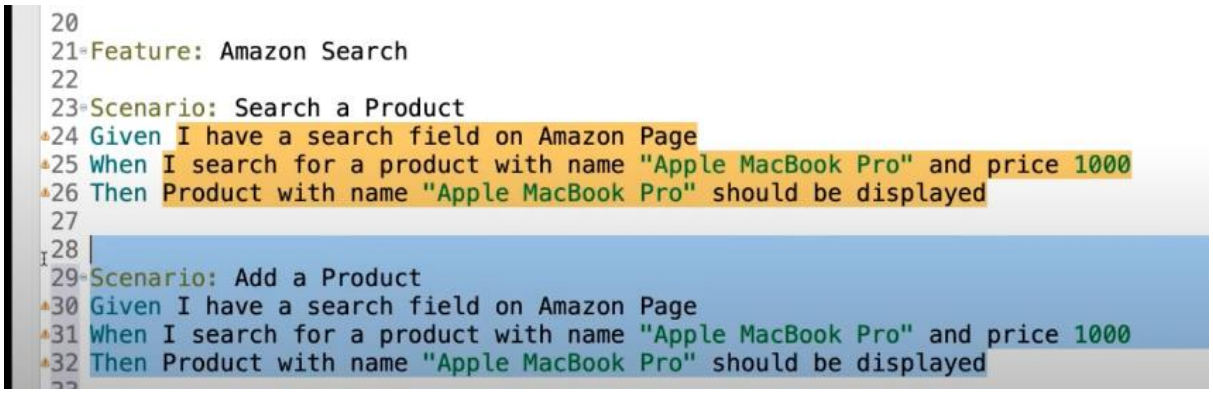



Sample feature file got created.



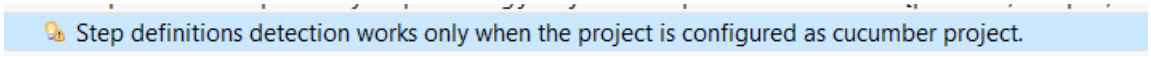

One feature file can have multiple scenarios.

This is the way to write multiple scenarios:



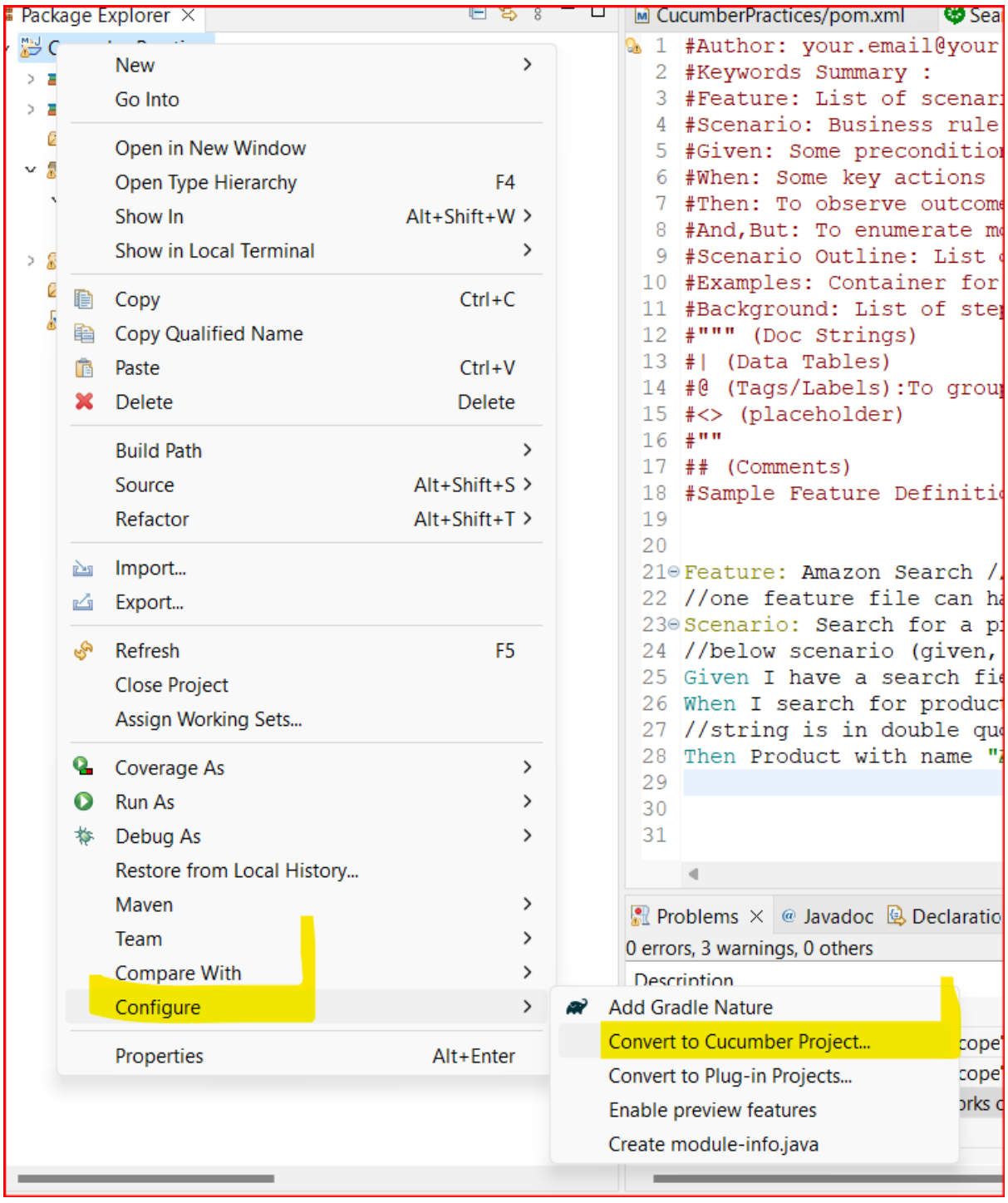
```
20
21 Feature: Amazon Search
22
23 Scenario: Search a Product
24 Given I have a search field on Amazon Page
25 When I search for a product with name "Apple MacBook Pro" and price 1000
26 Then Product with name "Apple MacBook Pro" should be displayed
27
28
29 Scenario: Add a Product
30 Given I have a search field on Amazon Page
31 When I search for a product with name "Apple MacBook Pro" and price 1000
32 Then Product with name "Apple MacBook Pro" should be displayed
```

Getting this warning in console:

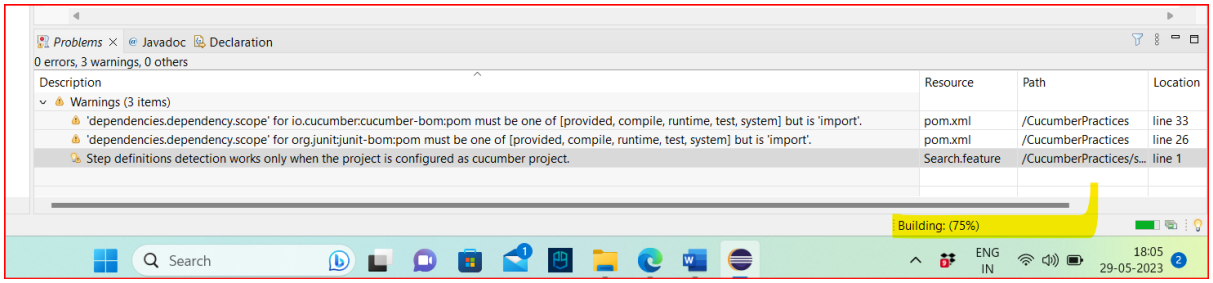



How to resolve it:

Right click on project -> configure -> convert to cucumber.




It will show building....




Now get this warning on the steps:

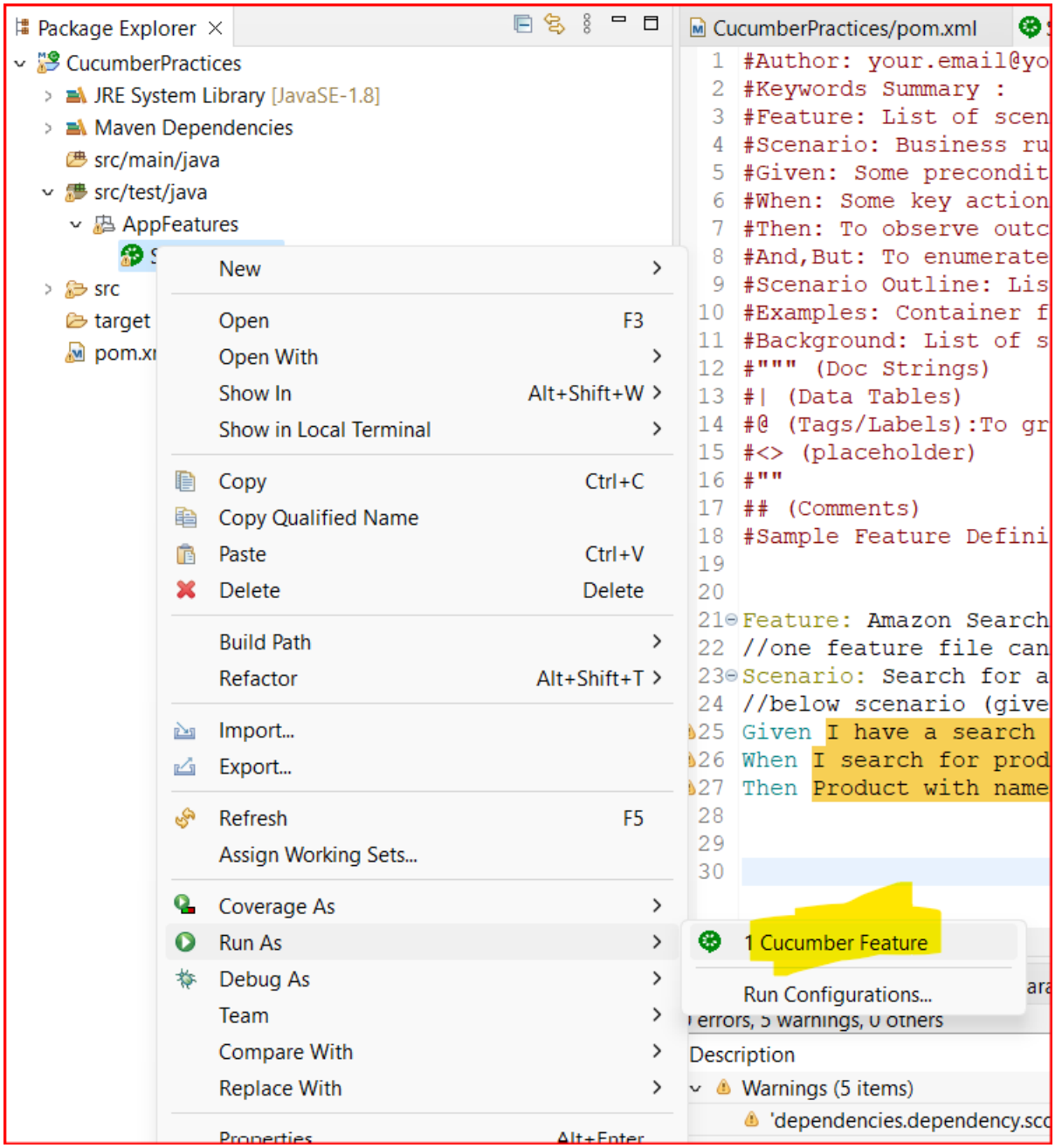
```
23 Scenario: Search for a product //here we give the scenario name
24 //below scenario (given, when, then, and) etc are known as steps
25 Given I have a search field on Amazon page //given can be considered as pre-condition
26 Step 'I search for product with name "Apple" and price is 1000 //string is in double quotes' does not have a matching glue code
27 Then Product with name "Apple" should be displayed
28
29
30
```

Why?

Glue code means there is no step definition mentioned.

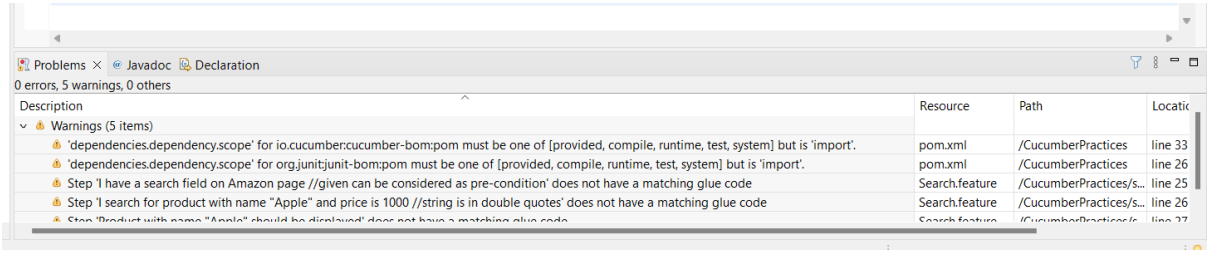
Now when we right click on feature file and run as cucumber project there will be no output:

Right click on feature and run as:



Console:

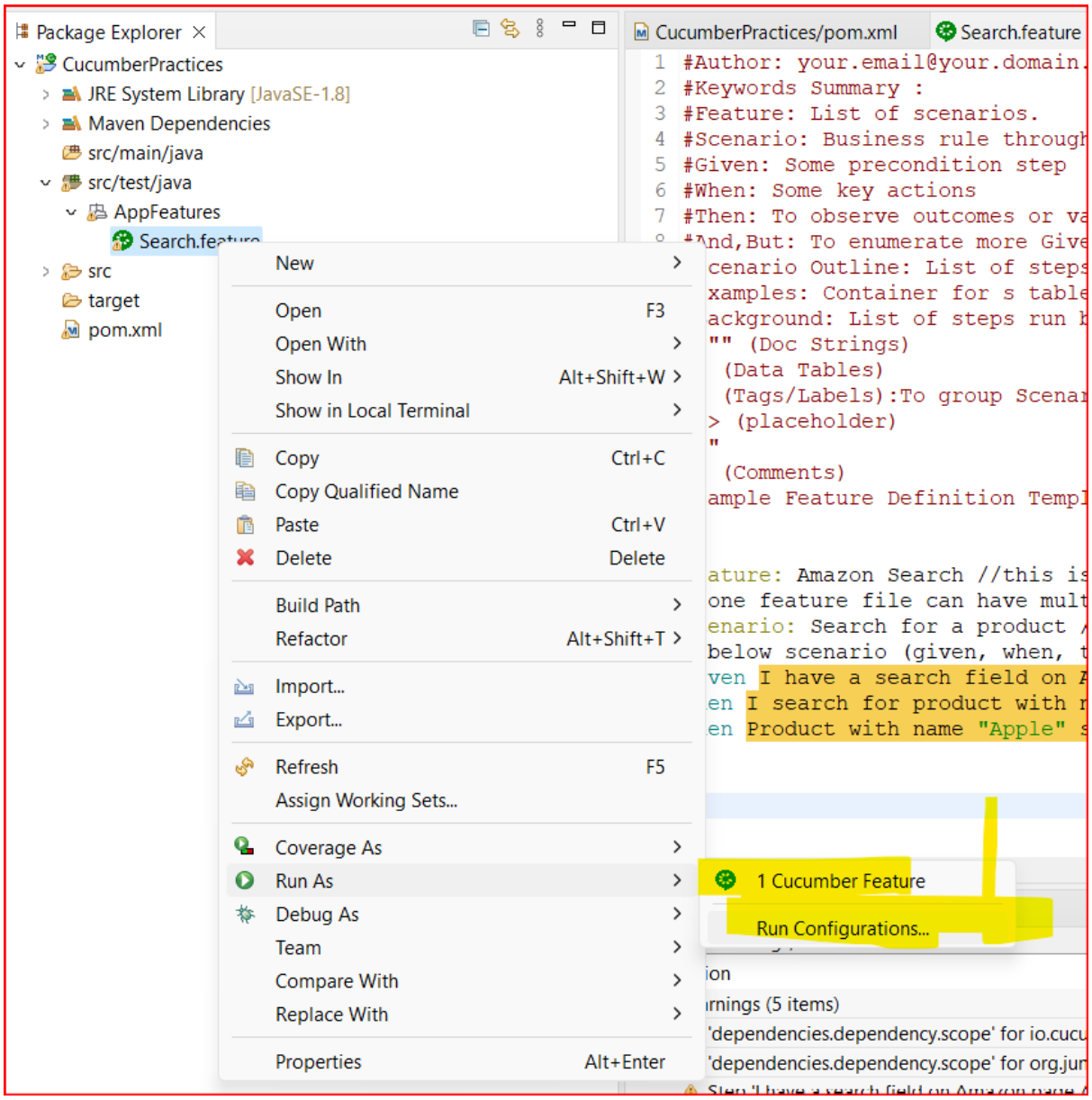
Nothing changed.



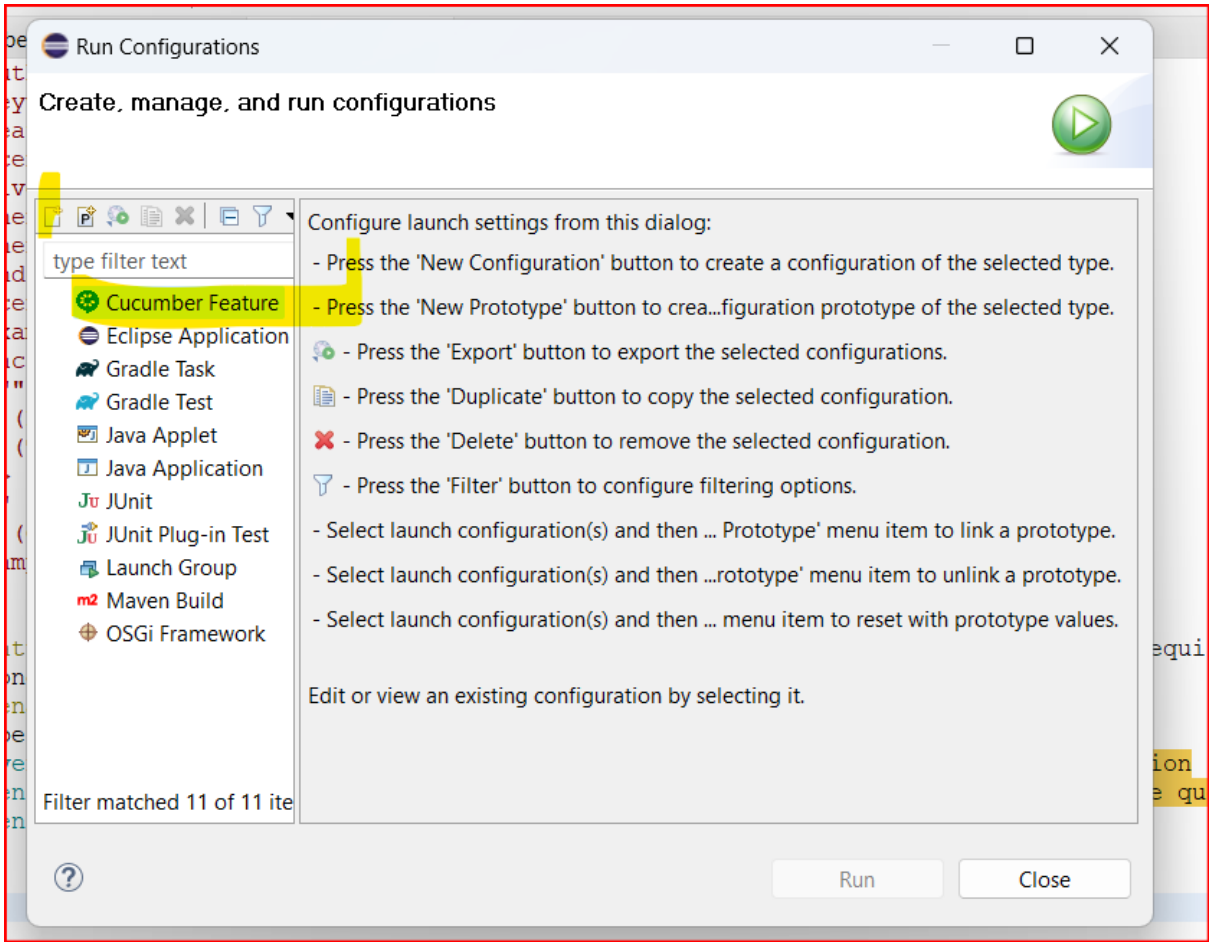

So, we need to configure the feature files in run configuration.

Right click on feature.

Run as -> run configuration.

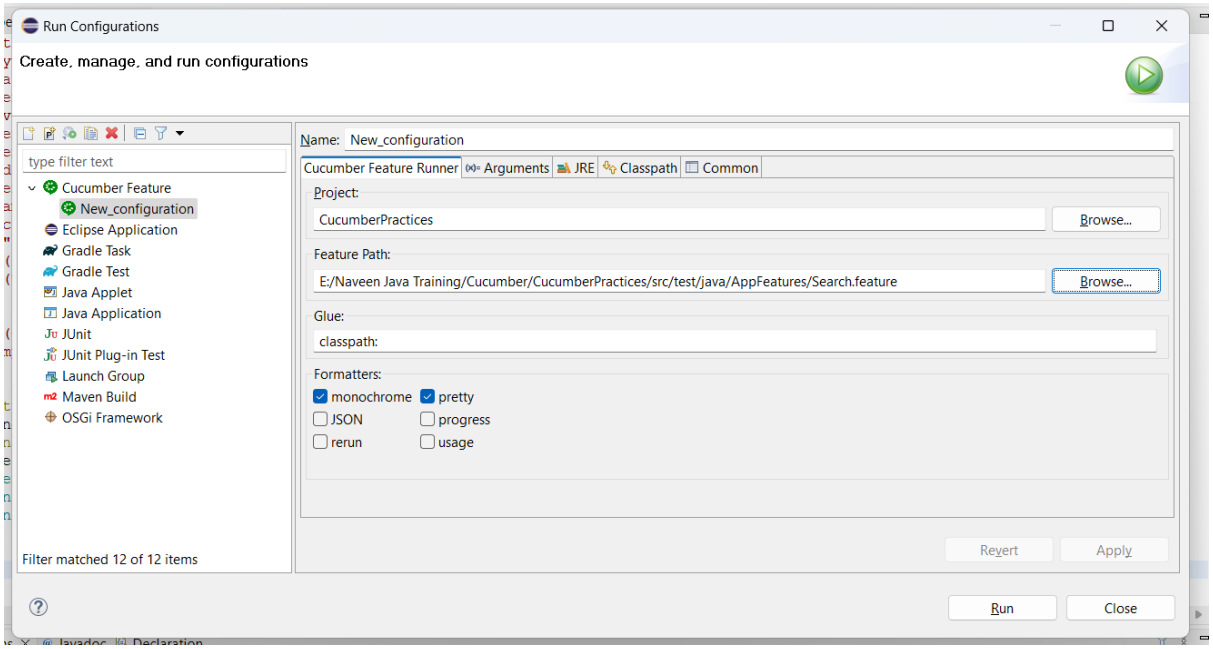


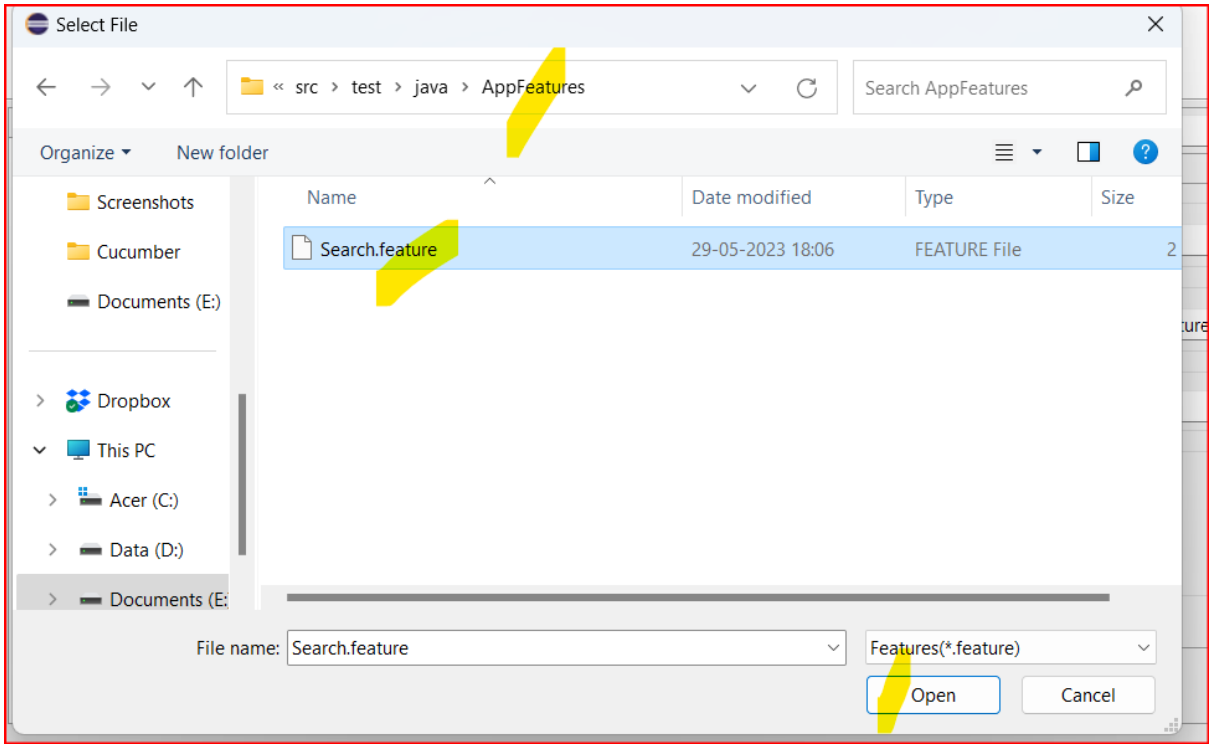

Select the “+” icon after clicking on cucumber.




Like this new window will come.

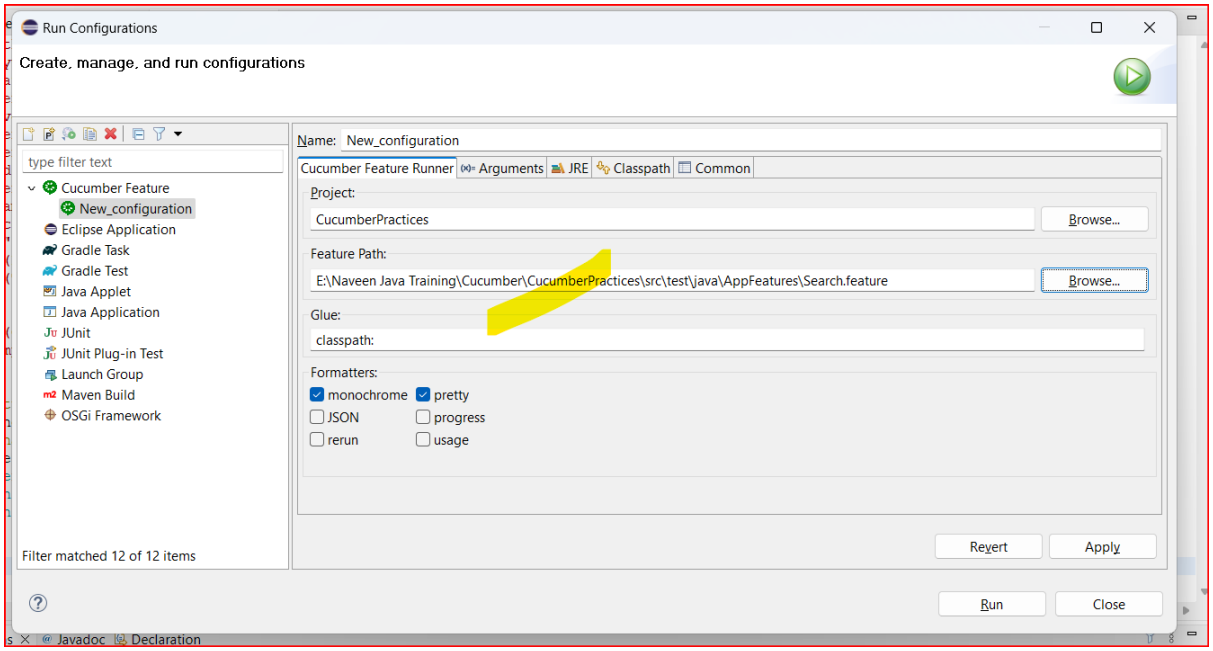
Click browse and go till the feature file path on your computer.






Click open.

Path has been updated.

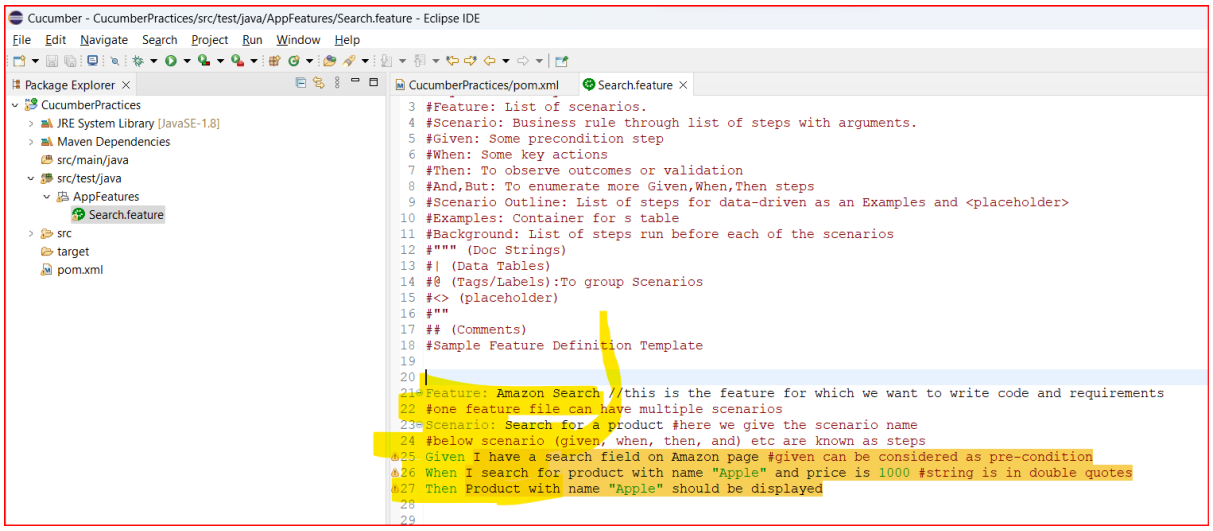



Click apply and then run.

## For commenting in feature file use #-

```
#Feature: List of scenarios.
#Scenario: Business rule through list of steps with arguments.
#Given: Some precondition step
#When: Some key actions
#Then: To observe outcomes or validation
#And,But: To enumerate more Given,When,Then steps
#Scenario Outline: List of steps for data-driven as an Examples and <placeholder>
#Examples: Container for s table
#Background: List of steps run before each of the scenarios
#""" (Doc Strings)
#| (Data Tables)
#@ (Tags/Labels):To group Scenarios
#<> (placeholder)
#""
## (Comments)
#Sample Feature Definition Template


```

Feature file created:



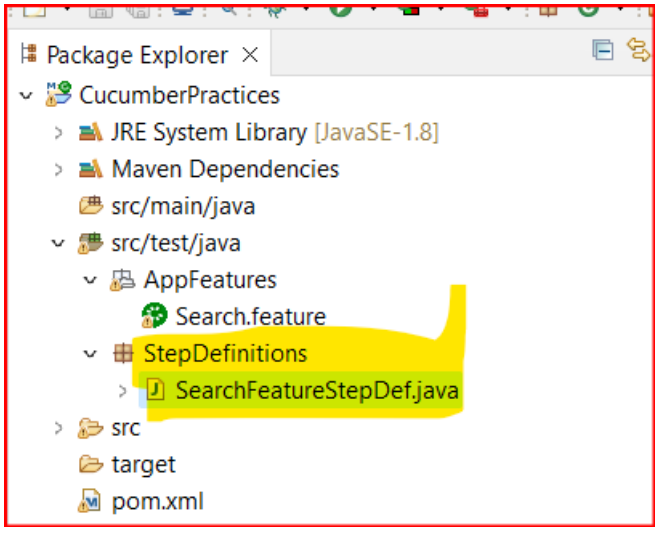

We run it.

We get below in console.

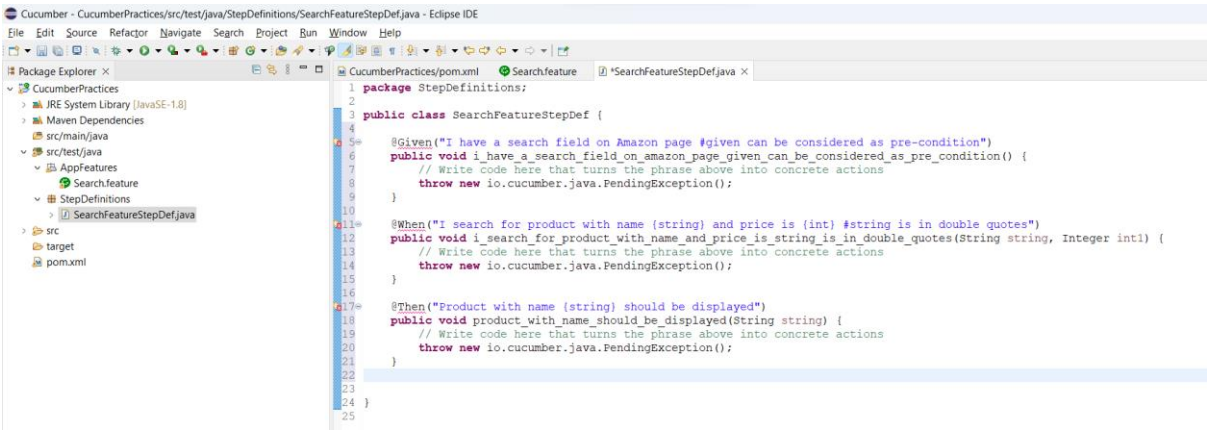
```
May 30, 2023 8:30:29 AM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use
io.cucumber.core.cli.Main

Scenario: Search for a product #here we give the scenario name
# src/test/java/AppFeatures/Search.feature:23
  Given I have a search field on Amazon page #given can be considered as
pre-condition
  When I search for product with name "Apple" and price is 1000 #string
is in double quotes
  Then Product with name "Apple" should be displayed

Undefined scenarios:
file:///E:/Naveen%20Java%20Training/Cucumber/CucumberPractices/src/test/j
ava/AppFeatures/Search.feature:23 # Search for a product #here we give
the scenario name

1 Scenarios (1 undefined)
```

```
3 Steps (2 skipped, 1 undefined)
0m0.100s


You can implement missing steps with the snippets below:

@Given("I have a search field on Amazon page #given can be considered as
pre-condition")
public void
i_have_a_search_field_on_amazon_page_given_can_be_considered_as_pre_condi
tion() {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}

@When("I search for product with name {string} and price is {int} #string
is in double quotes")
public void
i_search_for_product_with_name_and_price_is_string_is_in_double_quotes(St
ring string, Integer int1) {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}

@Then("Product with name {string} should be displayed")
public void product_with_name_should_be_displayed(String string) {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}
```

Create package for step definition and add a class for search feature:




Copy the methods from console and paste in step definition class:




We have to import given, when, then etc from io.cucumber.java package:

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

public class SearchFeatureStepDef {

    @Given("I have a search field on Amazon page")
    public void i_have_a_search_field_on_amazon_page() {
        // Write code here that turns the phrase above ir
        throw new io.cucumber.java.PendingException();
    }

    @When("I search for product with name {string} and pr
    public void i_search_for_product_with_name_and_price_
        // Write code here that turns the phrase above ir
        throw new io.cucumber.java.PendingException();
    }

    @Then("Product with name {string} should be displayed
    public void product_with_name_should_be_displayed(Str
        // Write code here that turns the phrase above ir
        throw new io.cucumber.java.PendingException();
    }
```

Whatever step you have written in feature will be mapped in step definition:

Feature file:

```
#below scenario (given, when, then, and) et
Given I have a search field on Amazon page
#given can be considered as pre-condition
```

Step definition file:

```
    @Given("I have a search field on Amazon page")
```

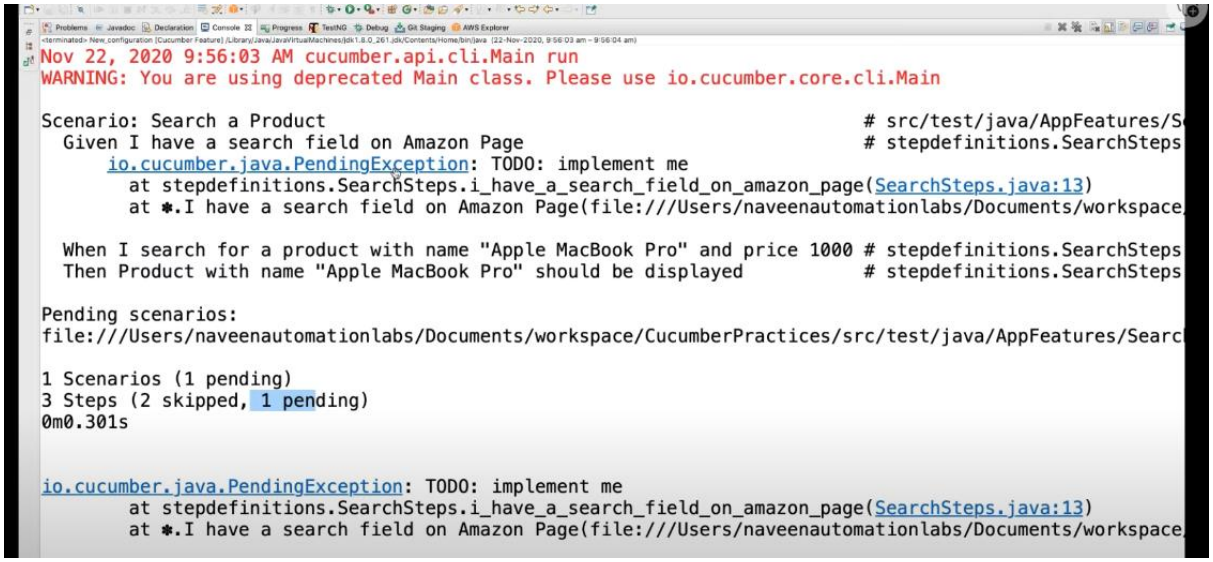
Cucumber generally gives same method name what we have written for given, when, then etc along-with underscore:

```
    @Given("I have a search field on Amazon page")
    public void i_have_a_search_field_on_amazon_page() {
        // Write code here that turns the phrase above into concrete actions
        throw new io.cucumber.java.PendingException();
    }
```

This exception is thrown because there is no java code:

```
    @Given("I have a search field on Amazon page")
    public void i_have_a_search_field_on_amazon_page() {
        // Write code here that turns the phrase above into concrete actions
        throw new io.cucumber.java.PendingException();
    }
```

If you run the feature file again:




It throws this exception at first step only because there is no java code.

Also, we can see there are three steps. 2 skipped as test could not reach step 2 and step 3 as there is no code. Step 1 is pending because there is no code, and the pending exception is thrown as we need to write some code for program to run.

## First code-

Feature file:

```
#Author: your.email@your.domain.com
#Keywords Summary :
#Feature: List of scenarios.
#Scenario: Business rule through list of steps with arguments.
#Given: Some precondition step
#When: Some key actions
#Then: To observe outcomes or validation
#And,But: To enumerate more Given,When,Then steps
#Scenario Outline: List of steps for data-driven as an Examples and
<placeholder>
#Examples: Container for s table
#Background: List of steps run before each of the scenarios
```

```
#""" (Doc Strings)
#| (Data Tables)
#@ (Tags/Labels):To group Scenarios
#<> (placeholder)
#""
## (Comments)
#Sample Feature Definition Template


Feature: Amazon Search
#this is the feature for which we want to write code and requirements
#one feature file can have multiple scenarios
Scenario: Search for a product
#here we give the scenario name
#below scenario (given, when, then, and) etc are known as steps
Given I have a search field on Amazon page
#given can be considered as pre-condition
When I search for product with name "Apple" and price is 1000
#string is in double quotes
Then Product with name "Apple" should be displayed
```

Step definition file for the above feature:

```
package StepDefinitions;

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

public class SearchFeatureStepDef {

	@Given("I have a search field on Amazon page")
	public void i have a search field on amazon page() {
	    System.out.println("step 1 - i am on search page");
	}

	@When("I search for product with name {string} and price is {int}")
	public void i_search_for_product_with_name_and_price_is(String
productName, Integer price) {
	   System.out.println("step 2 - search product with name " +
productName + " and price is " + price);
	}

	@Then("Product with name {string} should be displayed")
	public void product_with_name_should_be_displayed(String
productName) {
	    System.out.println("step 3 - product with " + productName + " :
is displayed");
	}


}
```

Run feature file and check in console:

```
May 30, 2023 8:46:21 AM cucumber.api.cli.Main run
```

```
WARNING: You are using deprecated Main class. Please use
io.cucumber.core.cli.Main

Scenario: Search for a product                                    #
src/test/java/AppFeatures/Search.feature:24
step 1 - i am on search page
  Given I have a search field on Amazon page                      #
StepDefinitions.SearchFeatureStepDef.i_have_a_search_field_on_amazon_page
()
step 2 - search product with name Apple and price is 1000
  When I search for product with name "Apple" and price is 1000 #
StepDefinitions.SearchFeatureStepDef.i_search_for_product_with_name_and_p
rice_is(java.lang.String,java.lang.Integer)
step 3 - product with Apple : is displayed
  Then Product with name "Apple" should be displayed             #
StepDefinitions.SearchFeatureStepDef.product_with_name_should_be_displaye
d(java.lang.String)

1 Scenarios (1 passed)
3 Steps (3 passed)
0m0.147s
```

## How are we achieving the tester driven development approach-

According to bdd, create feature first. Write test cases. Run it and it fails. Refactor code so that the test cases pass. Keep doing this till all test cases passed.

## Lets see how tdd works in action-

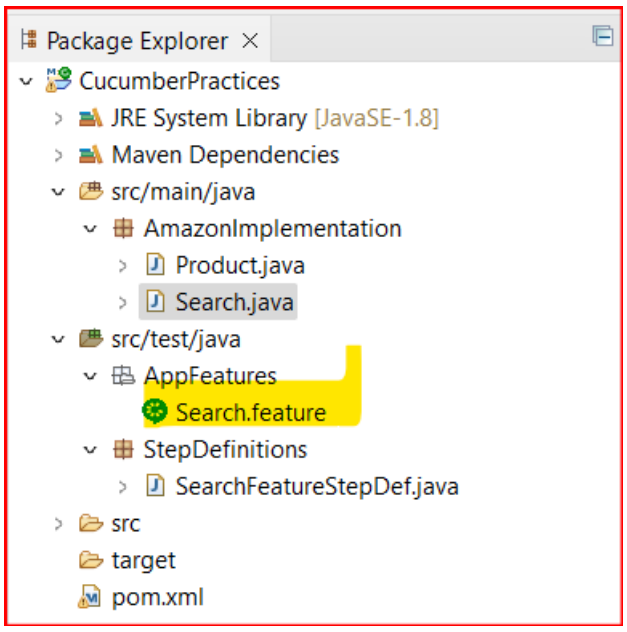
We write features.

We write step definition.

Run the code.

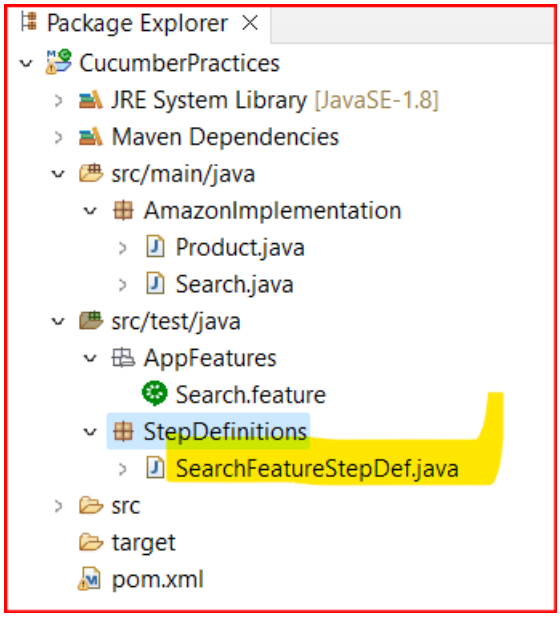
Fix defects and ensure all features and test cases are passing and we do this process continuously at run time.

Feature file:




```
#Author: your.email@your.domain.com
#Keywords Summary :
#Feature: List of scenarios.
#Scenario: Business rule through list of steps with arguments.
#Given: Some precondition step
#When: Some key actions
#Then: To observe outcomes or validation
#And,But: To enumerate more Given,When,Then steps
#Scenario Outline: List of steps for data-driven as an Examples and
<placeholder>
#Examples: Container for s table
#Background: List of steps run before each of the scenarios
#""" (Doc Strings)
#| (Data Tables)
#@ (Tags/Labels):To group Scenarios
#<> (placeholder)
#""
## (Comments)
#Sample Feature Definition Template


Feature: Amazon Search
#this is the feature for which we want to write code and requirements
#one feature file can have multiple scenarios
Scenario: Search for a product
#here we give the scenario name
#below scenario (given, when, then, and) etc are known as steps
Given I have a search field on Amazon page
#given can be considered as pre-condition
When I search for product with name "apple" and price is 1000
#string is in double quotes
Then Product with name "apple" should be displayed
```

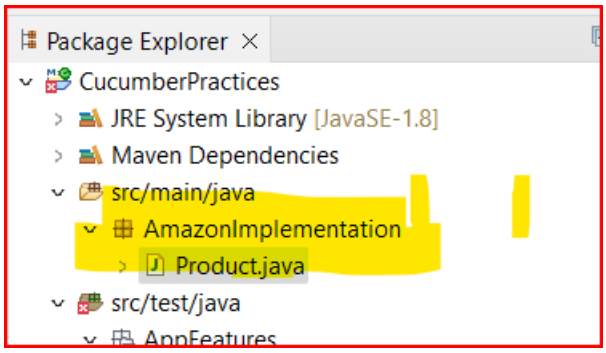
Step definition:




```java
package StepDefinitions;

import AmazonImplementation.Product;
import AmazonImplementation.Search;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import org.junit.jupiter.api.Assertions;

public class SearchFeatureStepDef {

	Product product;
	Search search;

	@Given("I have a search field on Amazon page")
	public void i_have_a_search_field_on_amazon_page() {
	    System.out.println("step 1 - i am on search page");
	}

	@When("I search for product with name {string} and price is {int}")
	public void i_search_for_product_with_name_and_price_is(String
productName, Integer price) {
	   System.out.println("step 2 - search product with name " +
productName + " and price is " + price);

	   product =new Product(productName, price);
	}

	@Then("Product with name {string} should be displayed")
	public void product_with_name_should_be_displayed(String
productName) {
	    System.out.println("step 3 - product with " + productName + " :
is displayed");
	    search=new Search();
	    String productNameReturned=search.displayProductName(product);
	    System.out.println("returned product is " +
productNameReturned);
	    Assertions.assertEquals(product.getProductName(),
productNameReturned);
	}
```

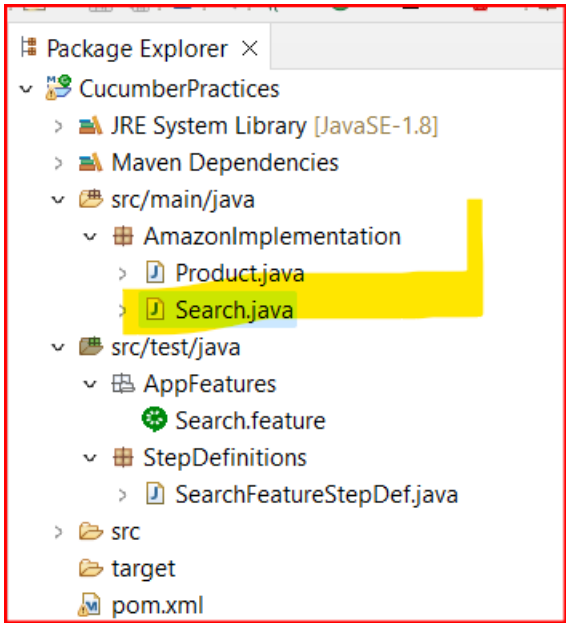
```
}
```

Create product class:




```java
package AmazonImplementation;

import java.util.ArrayList;
import java.util.List;

public class Product {

	private String productName;
	private int price;

	public Product(String productName, int price) {
		this.productName = productName;
		this.price = price;
	}

	public String getProductName() {
		return productName;
	}

	public void setProductName(String productName) {
		this.productName = productName;
	}

	public int getPrice() {
		return price;
	}

	public void setPrice(int price) {
		this.price = price;
	}

	// this method will list of products in the form of string
	public List<String> getProductList() {
		List<String> prodList = new ArrayList<>();
		prodList.add("apple");
		prodList.add("hp");
		prodList.add("samsung");
		prodList.add("bourbon");
		return prodList;
	}

}
```

Search implementation class:




```
package AmazonImplementation;

public class Search {

	public String displayProductName(Product product) { // here we will
take the product name returned from
		// product.java class
		if
(product.getProductList().contains(product.getProductName())) {
			return product.getProductName();
		} else {
			return null;
		}
		// or we can simply write, because if above return is
satisfied that will be the
		// thing which will be returned to method
		// return null;
	}

}
```

Note-

When you get error as “List cannot be resolved to type, or Array List cannot be resolved to type” it means we have to add the import <package name> statement.

```
import java.util.ArrayList;
import java.util.List;
```

Note-

Any error which says “cannot be resolved to type” means that we didn’t import the needed packages.

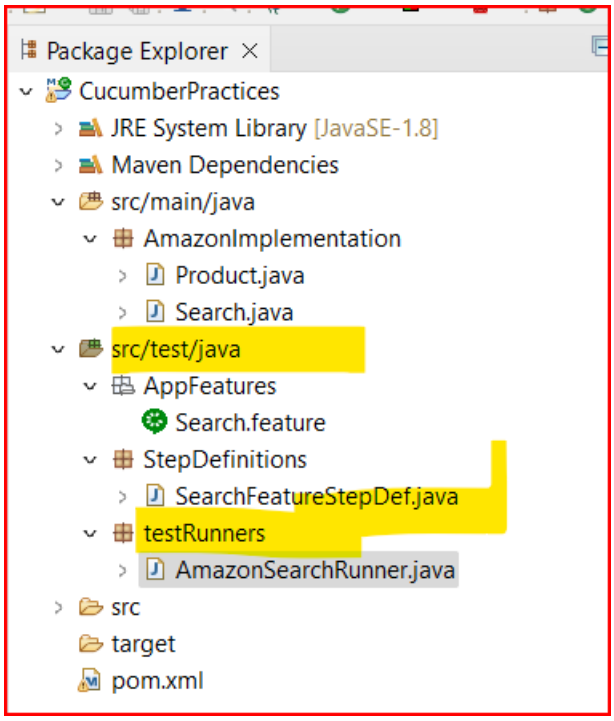
Note-

“Then” is used for assertions generally.

Runner.java:

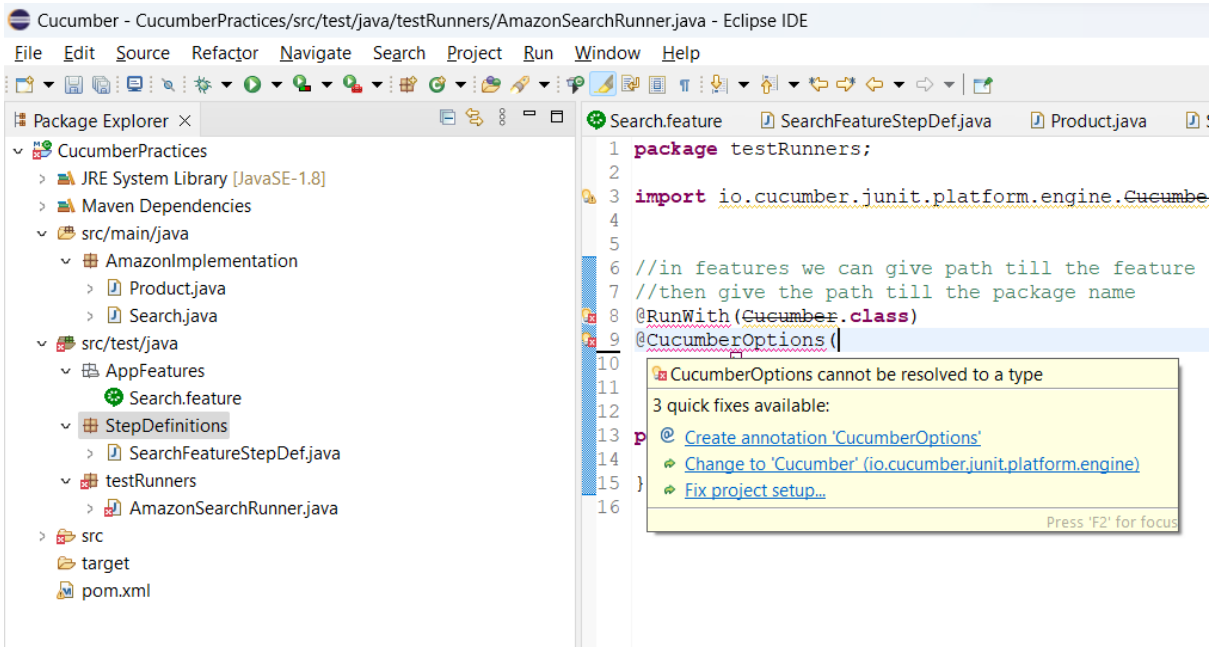
Every time we cannot run feature by feature.

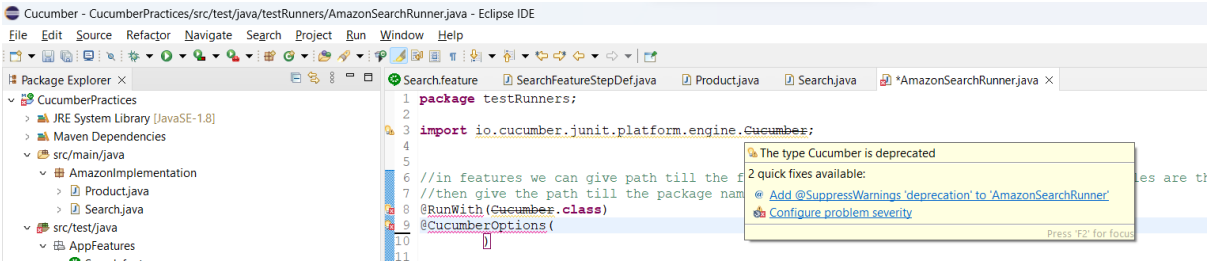
Runner class location:



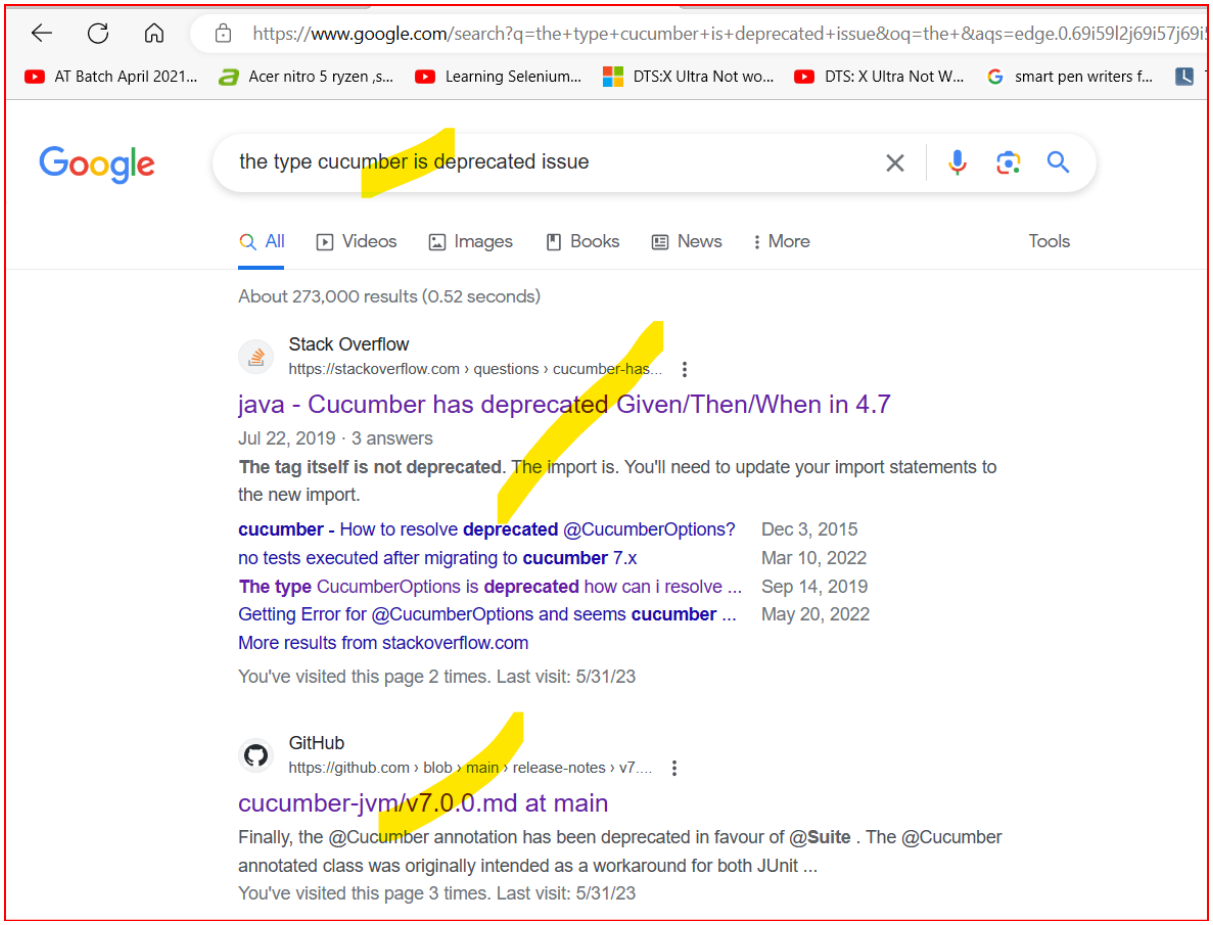

You will get this error in runner class:

the type cucumber is deprecated issue.






Search for this in google:




Go to this link and it says what changes to do:

[cucumber-jvm/v7.0.0.md at main · cucumber/cucumber-jvm · GitHub]

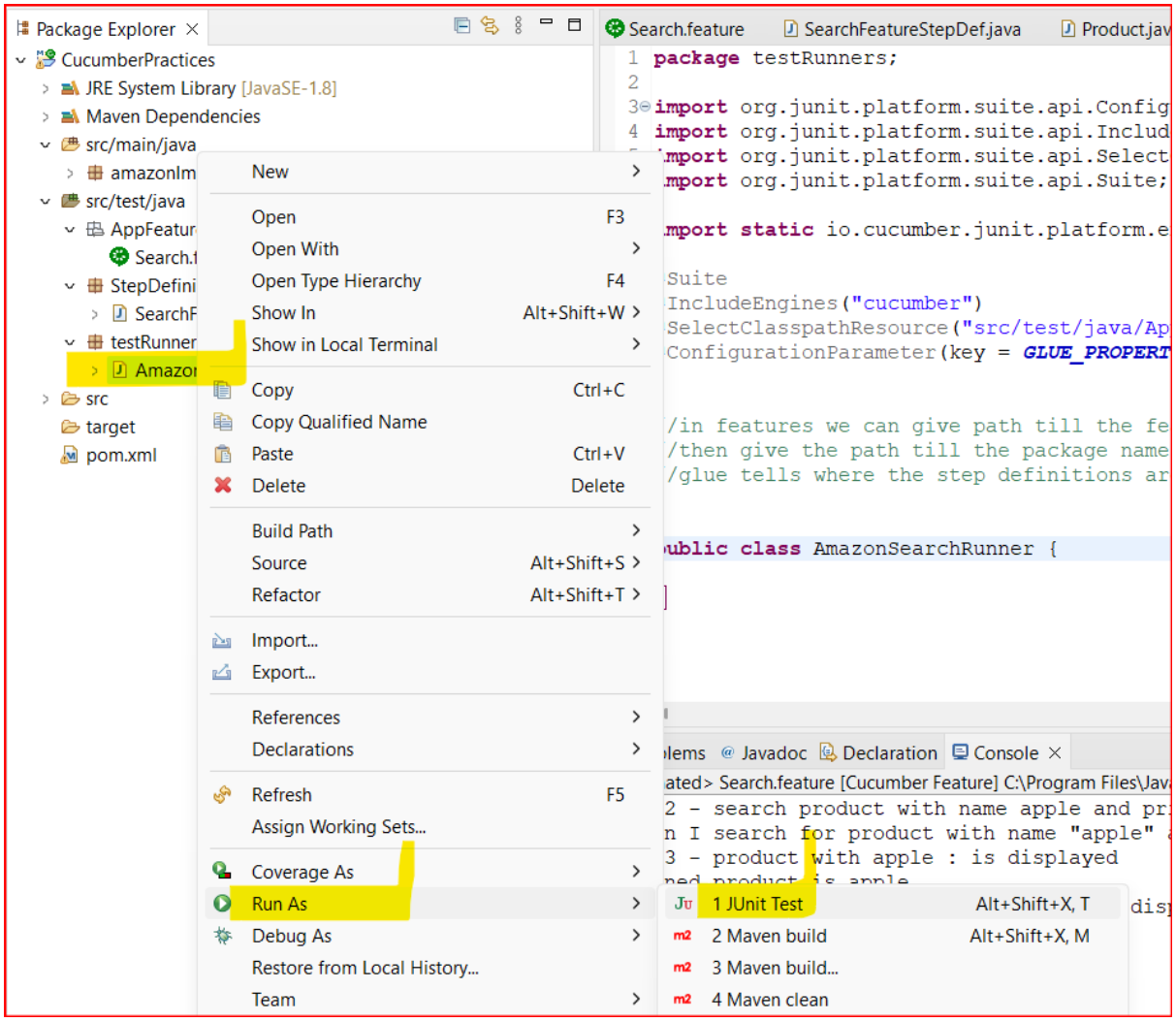
these are the snippets from what to change:

```
package com.example;

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

@RunWith(Cucumber.class)
@CucumberOptions(glue = "com.example.application", features = "classpath:com/example/application")
public class RunCucumberTest {

}
```

As a declarative JUnit 5 test suite:

```
package com.example;

import org.junit.platform.suite.api.ConfigurationParameter;
import org.junit.platform.suite.api.IncludeEngines;
import org.junit.platform.suite.api.SelectClasspathResource;
import org.junit.platform.suite.api.Suite;

import static io.cucumber.junit.platform.engine.Constants.GLUE_PROPERTY_NAME;

@Suite
@IncludeEngines("cucumber")
@SelectClasspathResource("com/example/application")
@ConfigurationParameter(key = GLUE_PROPERTY_NAME, value = "com.example.application")
public class RunCucumberTest {

}
```

Run the runner file as junit test because it’s made of junit:




Chat gpt solved the issue-

If you're encountering an error stating that the import cannot be resolved, it typically means that the required dependency for Cucumber is missing or not correctly configured in your project.

xml Copy code

```
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>7.12.0</version>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>7.12.0</version>
    <scope>test</scope>
</dependency>
```

2. Ensure that you have the necessary imports in your runner class. The required imports for the code you provided should include:

java Copy code

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

3. If you have already added the dependencies and the necessary imports, try rebuilding your project or refreshing your IDE to ensure that the dependencies are correctly resolved.

How to solve the error-

how to solve this error - java.lang.NoSuchMethodError: org.junit.platform.commons.util.ClassNamePatternFilterUtils.excludeMatchingClassNames(Ljava/lang/String;)Ljava/util/function/Predicate; at org.junit.platform.launcher.core.StackTracePruningEngineExecutionListener.<init>(StackTracePruningEngineExecutionListener.java:40) at org.junit.platform.launcher.core.EngineExecutionOrchestrator.selectExecutionListener(EngineExecutionOrchestrator.java:184) at org.junit.platform.launcher.core.EngineExecutionOrchestrator.execute(EngineExecutionOrchestrator.java:162) at org.junit.platform.launcher.core.EngineExecutionOrchestrator.execute(EngineExecutionOrchestrator.java:95) at

org.junit.platform.launcher.core.EngineExecutionOrchestrator.lambda$execute$0(EngineExecutionOrchestrator.java:60) at org.junit.platform.launcher.core.EngineExecutionOrchestrator.withInterceptedStreams(EngineExecutionOrchestrator.java:144) at org.junit.platform.launcher.core.EngineExecutionOrchestrator.execute(EngineExecutionOrchestrator.java:59) at org.junit.platform.launcher.core.DefaultLauncher.execute(DefaultLauncher.java:103) at org.junit.platform.launcher.core.DefaultLauncher.execute(DefaultLauncher.java:94) at org.junit.platform.launcher.core.DelegatingLauncher.execute(DelegatingLauncher.java:52) at org.junit.platform.launcher.core.SessionPerRequestLauncher.execute(SessionPerRequestLauncher.java:70) at org.eclipse.jdt.internal.junit5.runner.JUnit5TestReference.run(JUnit5TestReference.java:98) at org.eclipse.jdt.internal.junit.runner.TestExecution.run(TestExecution.java:40) at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.runTests(RemoteTestRunner.java:529) at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.runTests(RemoteTestRunner.java:756) at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.run(RemoteTestRunner.java:452) at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.main(RemoteTestRunner.java:210)

The error in your `pom.xml` file seems to be related to conflicting versions of JUnit dependencies. Here are the changes you can make to resolve the issue:

1. Remove the following dependency, as it is causing a conflict with the JUnit version specified in the `<dependencyManagement>` section:

xml Copy code

```xml
<dependency>
  <groupId>org.junit.platform</groupId>
  <artifactId>junit-platform-suite</artifactId>
  <version>1.10.0-M1</version>
  <scope>test</scope>
</dependency>
```

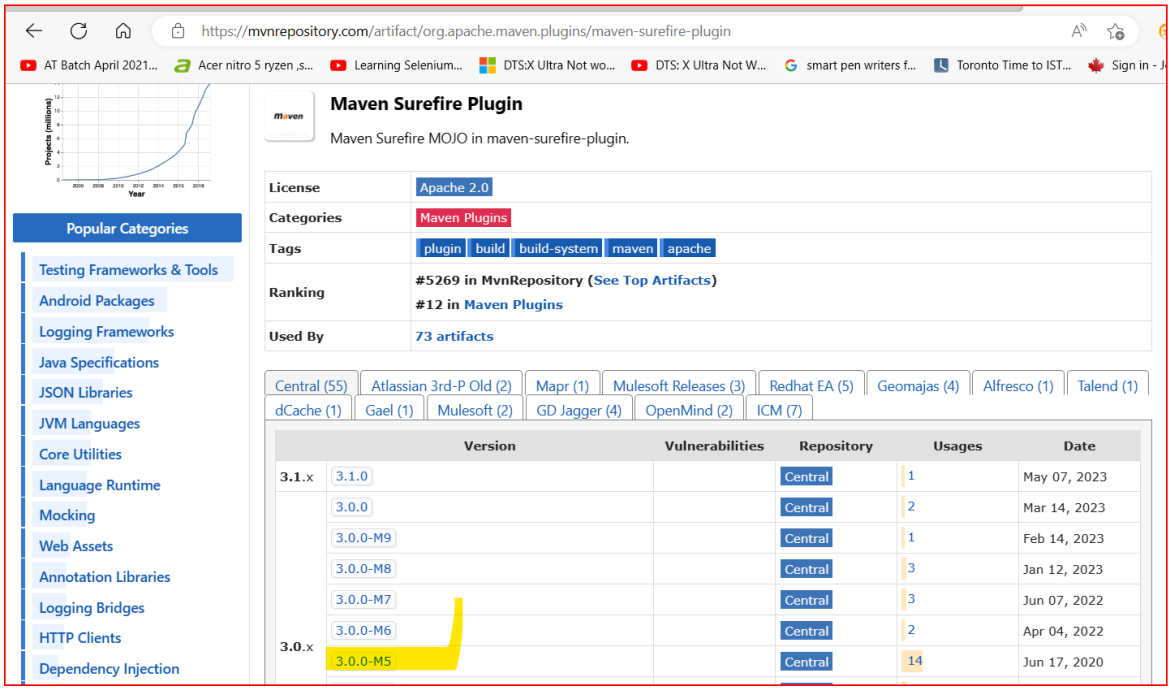
2. Update the version of the `maven-surefire-plugin` to match the JUnit version you are using. Change the `maven-surefire-plugin` configuration as follows:

xml Copy code

```
<plugin>
  <artifactId>maven-surefire-plugin</artifactId>
  <version>3.0.0-M5</version> <!-- Update the version to match your JUnit ve
  <executions>
    <execution>
      <id>default-test</id>
      <phase>test</phase>
      <goals>
        <goal>test</goal>
      </goals>
      <configuration>
        <properties>
          <configurationParameters>cucumber.junit-platform.naming-strategy=l
        </properties>
      </configuration>
    </execution>
  </executions>
  <configuration>
    <properties>
```

This is because see the seeding in mvnrepo.

This version has the highest seeds.

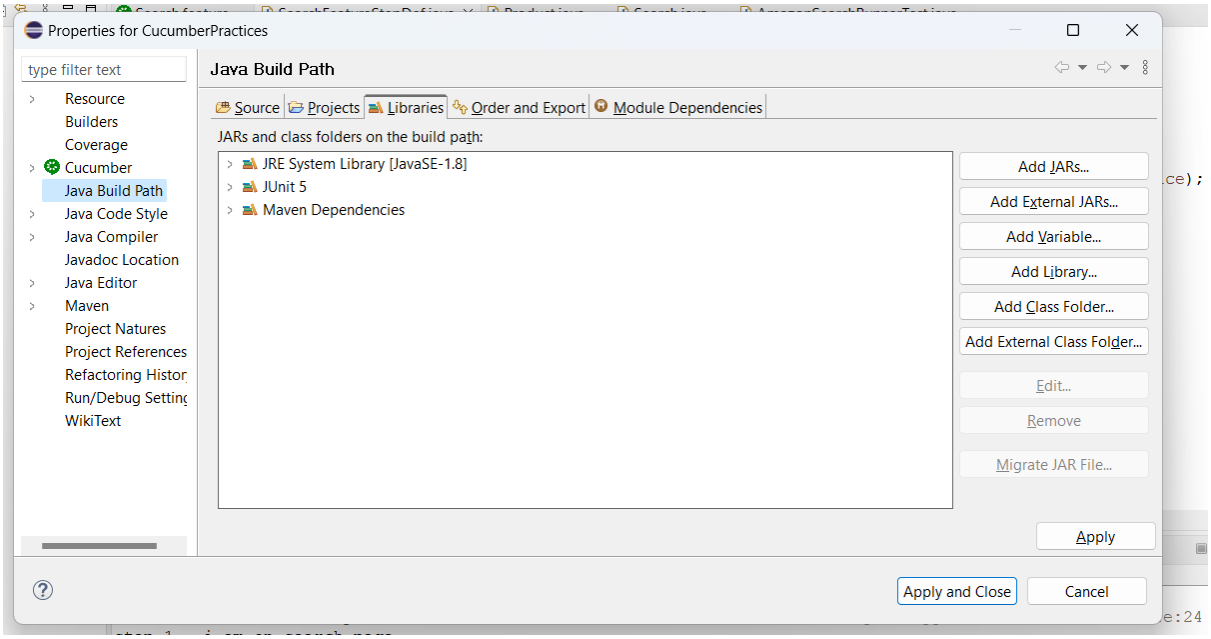



Right click on runner class and run as junit test:

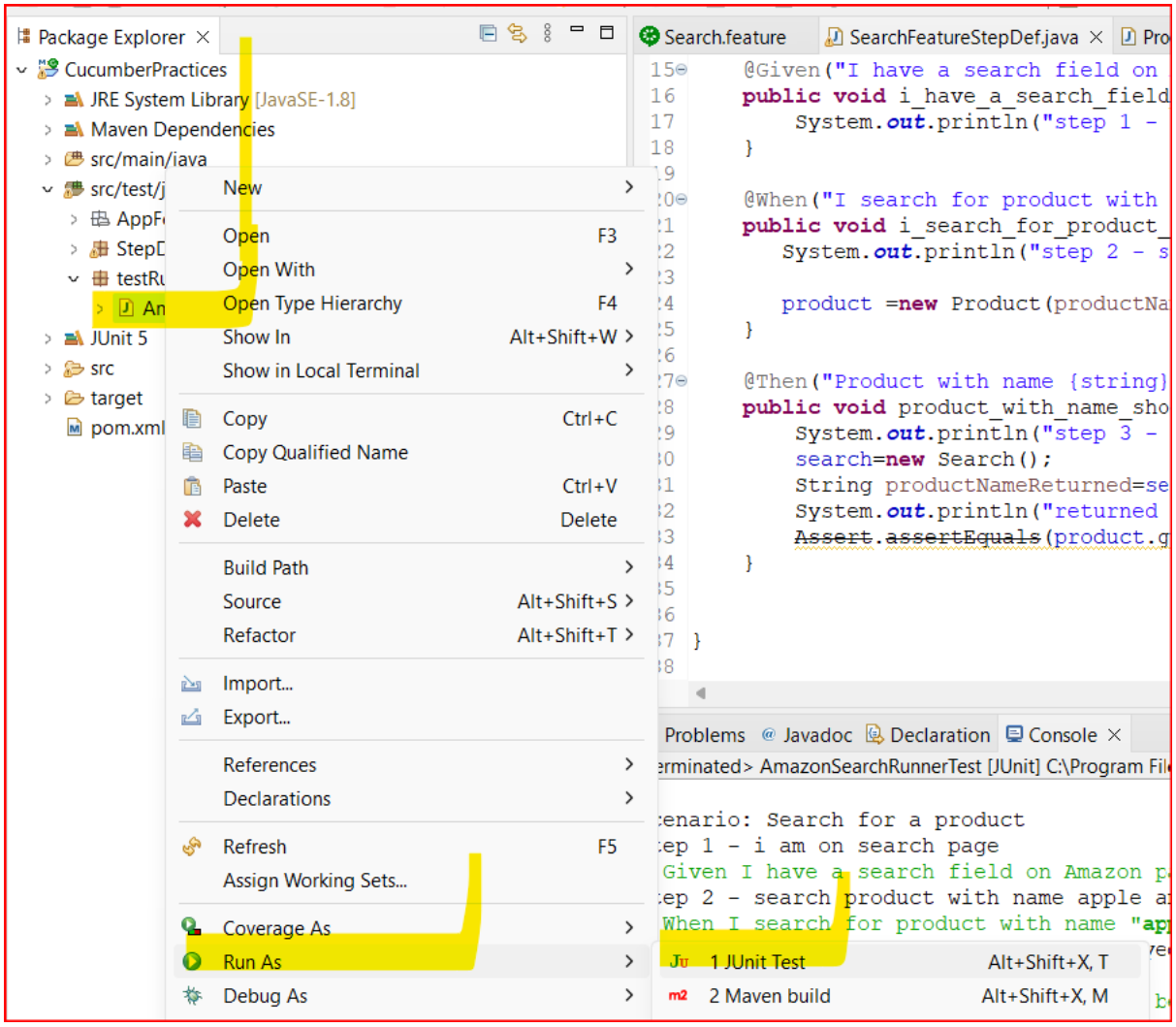
First time we get error as:

Problems launching junit tests - cannot find org.junit.platform.commons.annotations.testable on project build path. junit 5 tests can only be run if junit 5 is on build path.

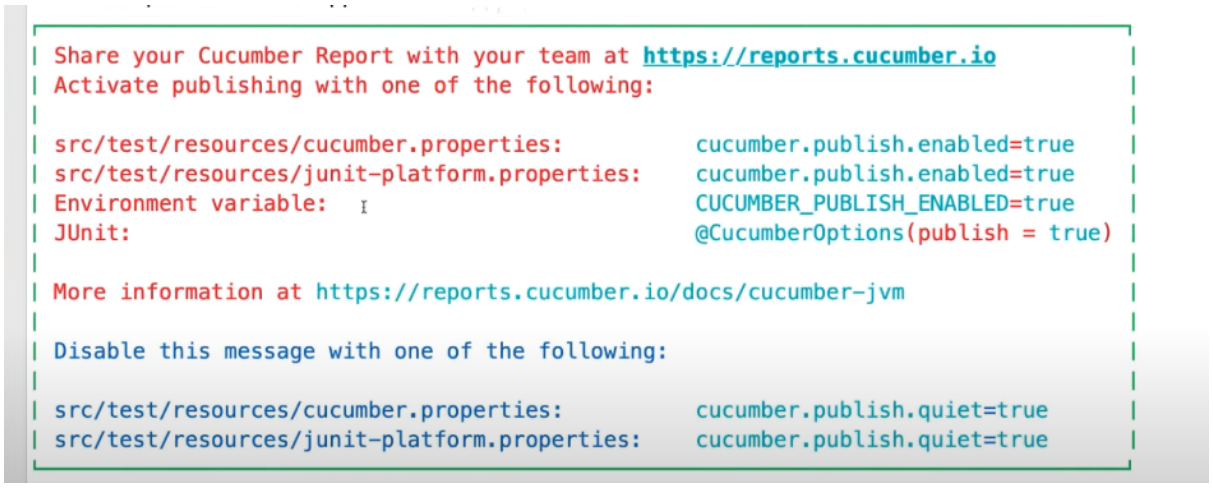
To solve this problem, right click on project -> properties -> java build path -> libraries -> >add library -> select junit -> select junit 5 -> keep clicking next and finish it. -> click apply -> > click apply and close




Run the runner class again as junit test:




In console what we are getting below is just for logging purpose and can be ignored:

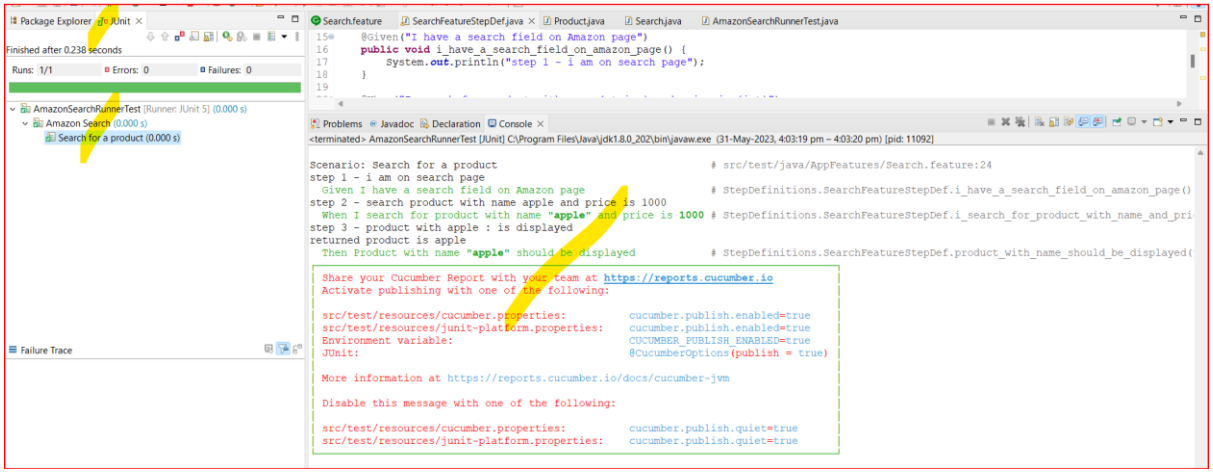



The above can be ignored.

Console output:


We can see junit tab in project explorer once project runs:

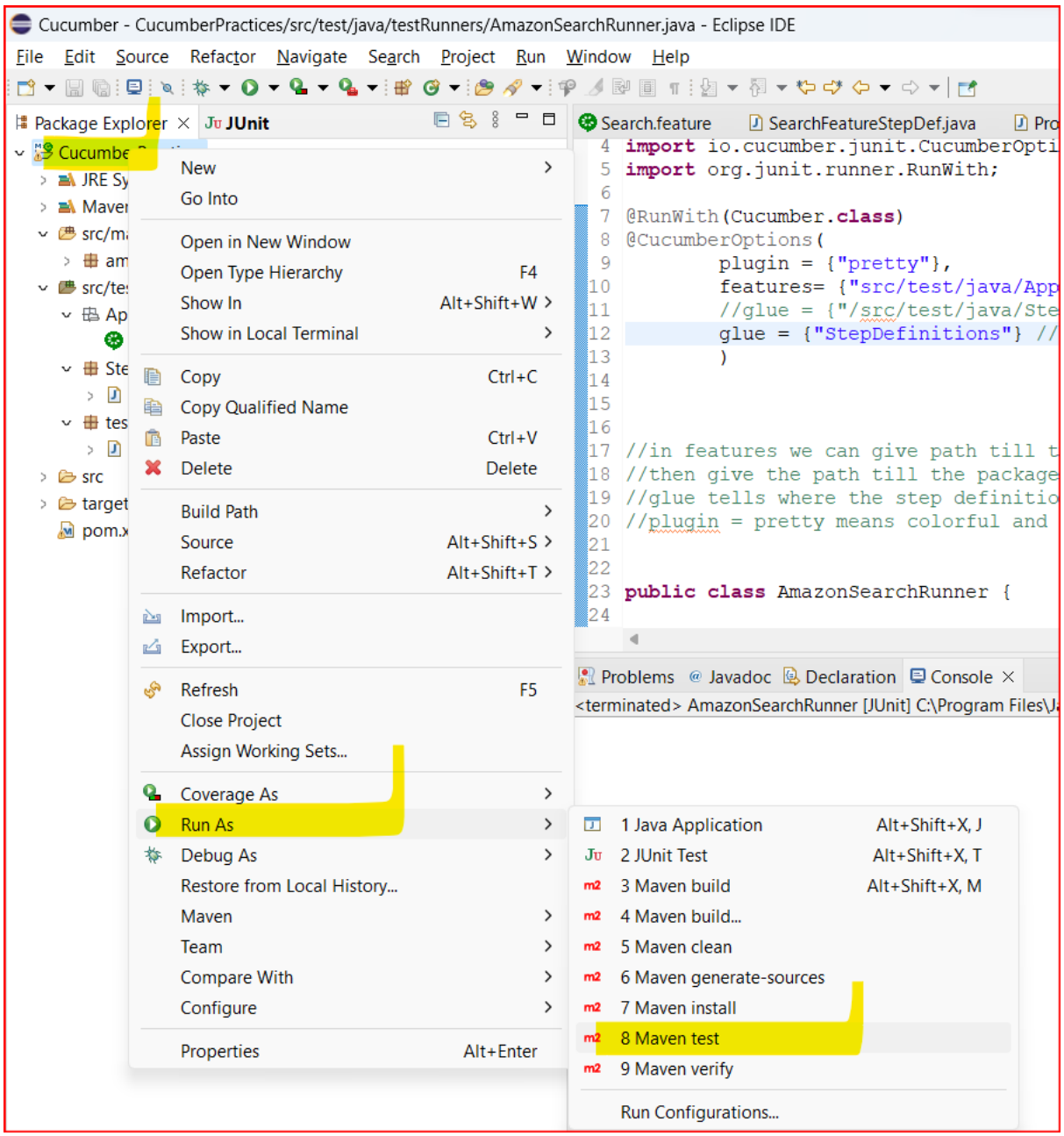
Green means passed.



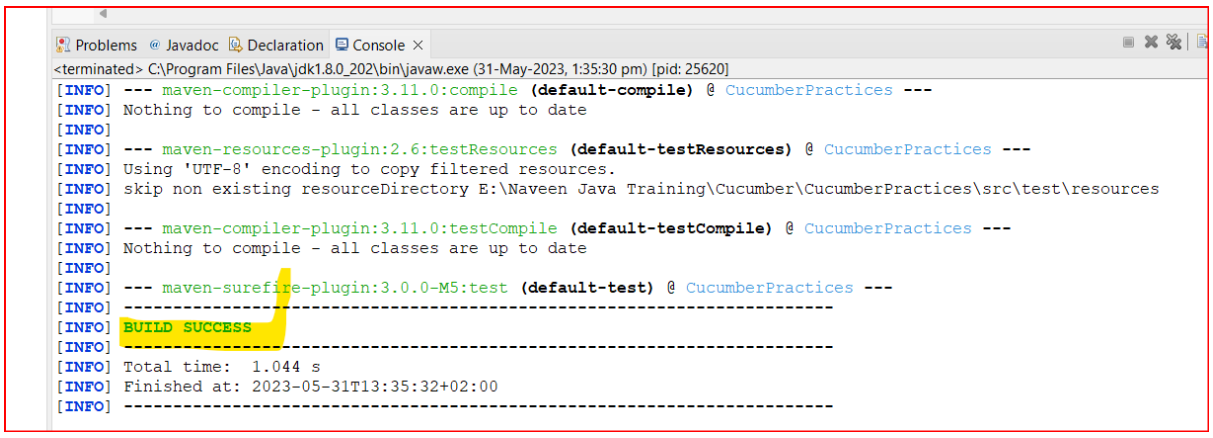

To run as maven we need to sure maven is installed in laptop.

Right click on project.

Run as maven test.



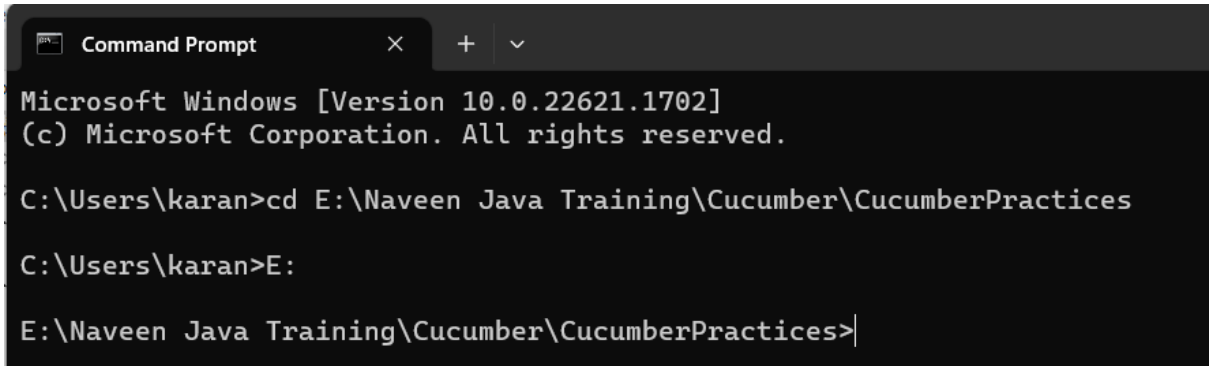

You should get build success.



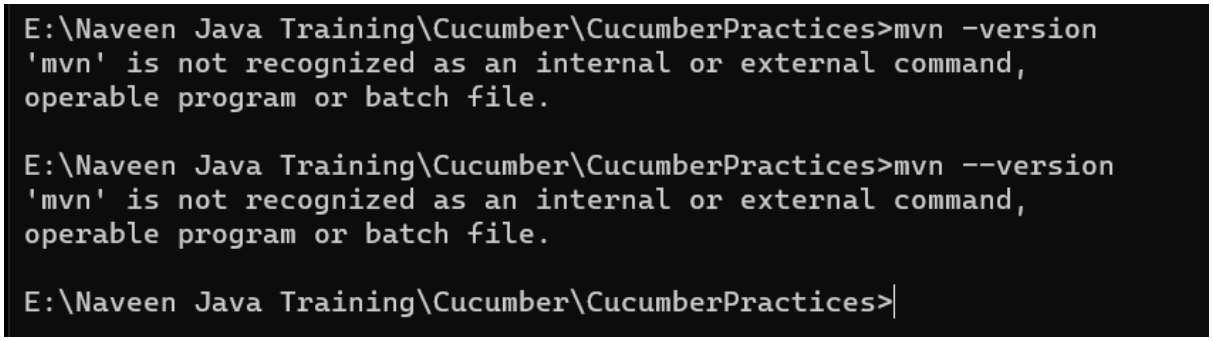
To run from command line using maven-

Open cmd.

Be in path of project.



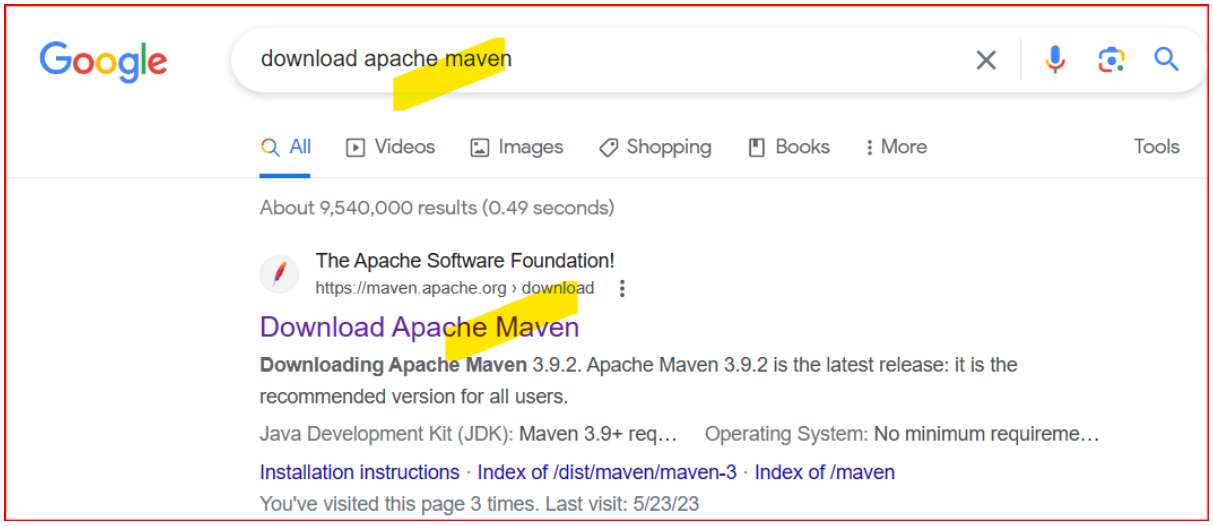

Check if maven is present.



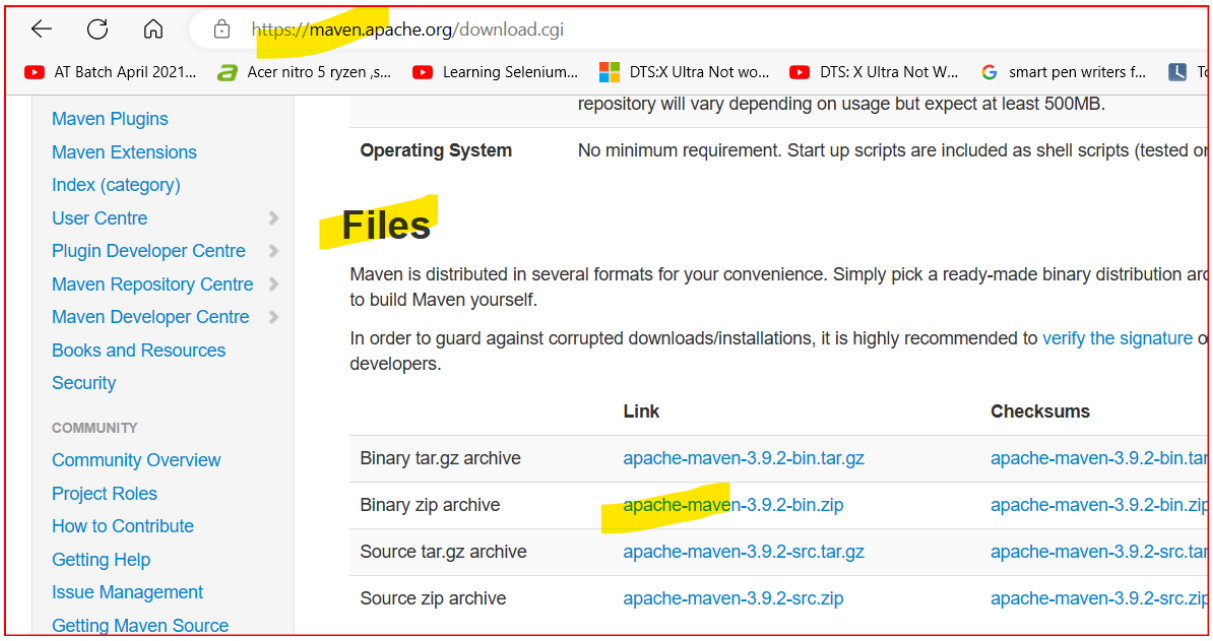

It is not present.

How to set it up.

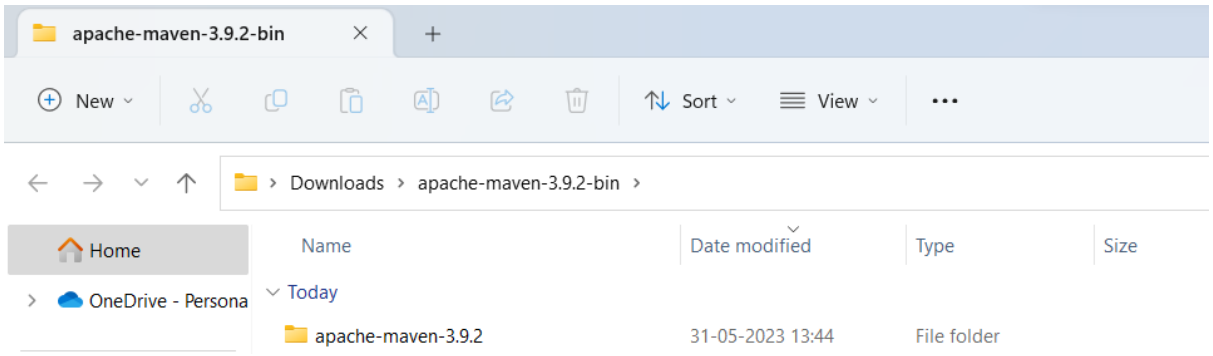
Goto google and search for download maven.




Once inside the link click on the highlighted zip file.

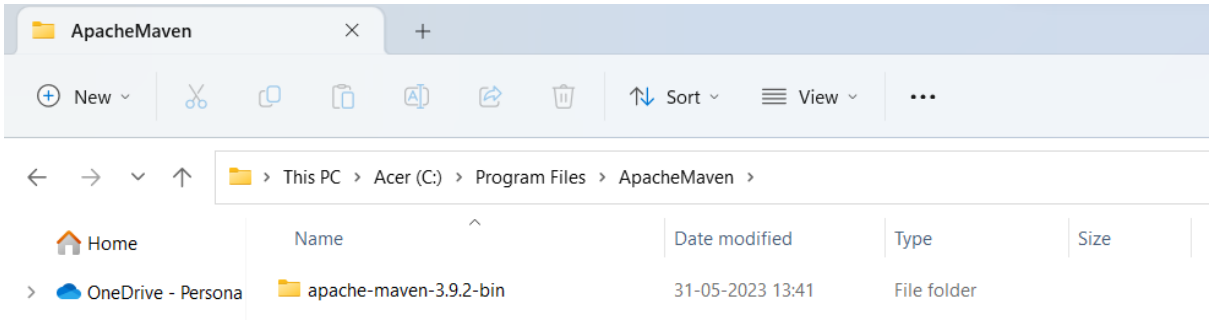



Extract it.



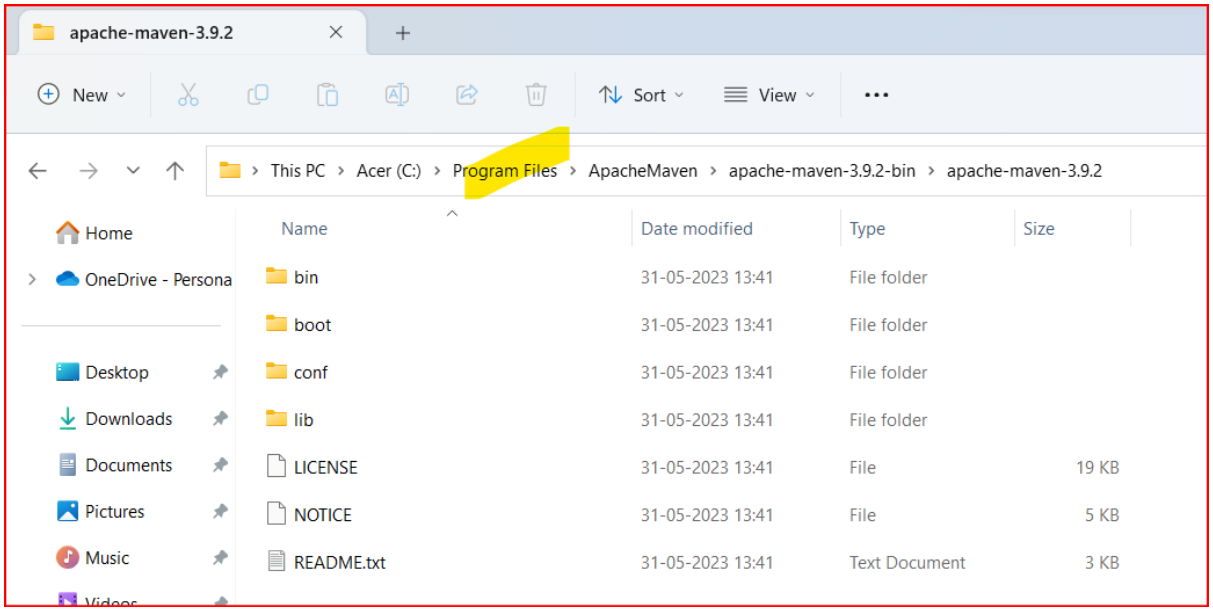

Copy this folder.

In c drive -> program files -> make a folder “apache maven” and paste this folder into that.




Go till this path mentioned in below image.

Copy the path.




Open this pc.

Environment variables.

In system variables add variable “MAVAN_HOME” in caps and in value paste the copied path.

Click ok.

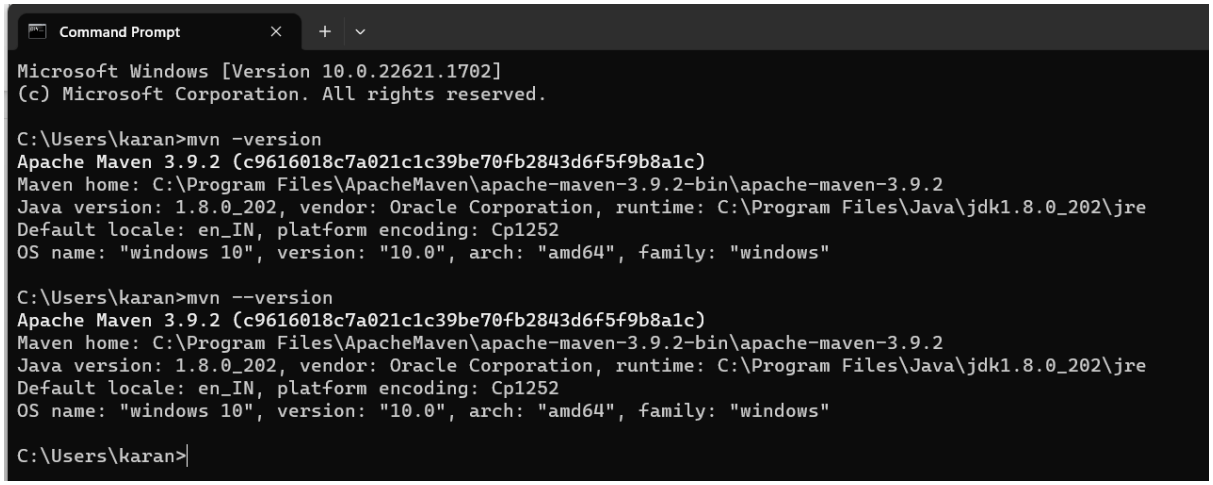
Now edit the path variable.

Write %MAVEN_HOME%\bin.

Click ok.

Type “mvn -version” or “mvn - -version”

You should see something like below:




Maven is installed.

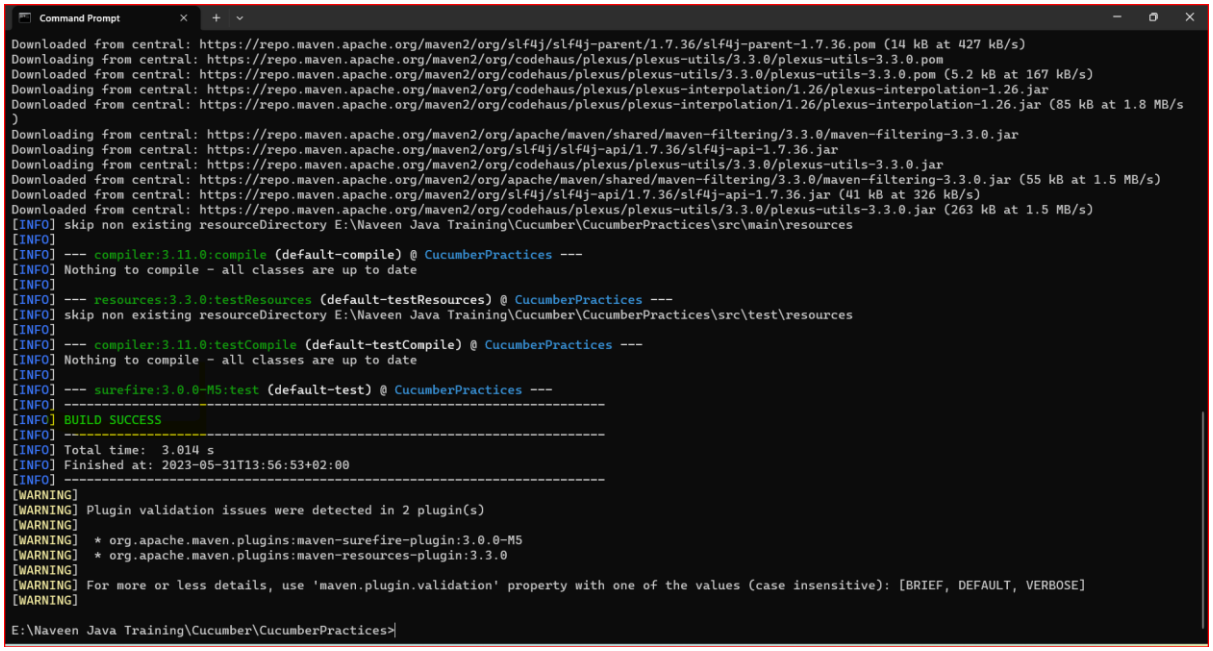
Now go to the path of the project.

Type this command.

```
E:\Naveen Java Training\Cucumber\CucumberPractices>mvn test
```

Hit enter.

You should get build success.



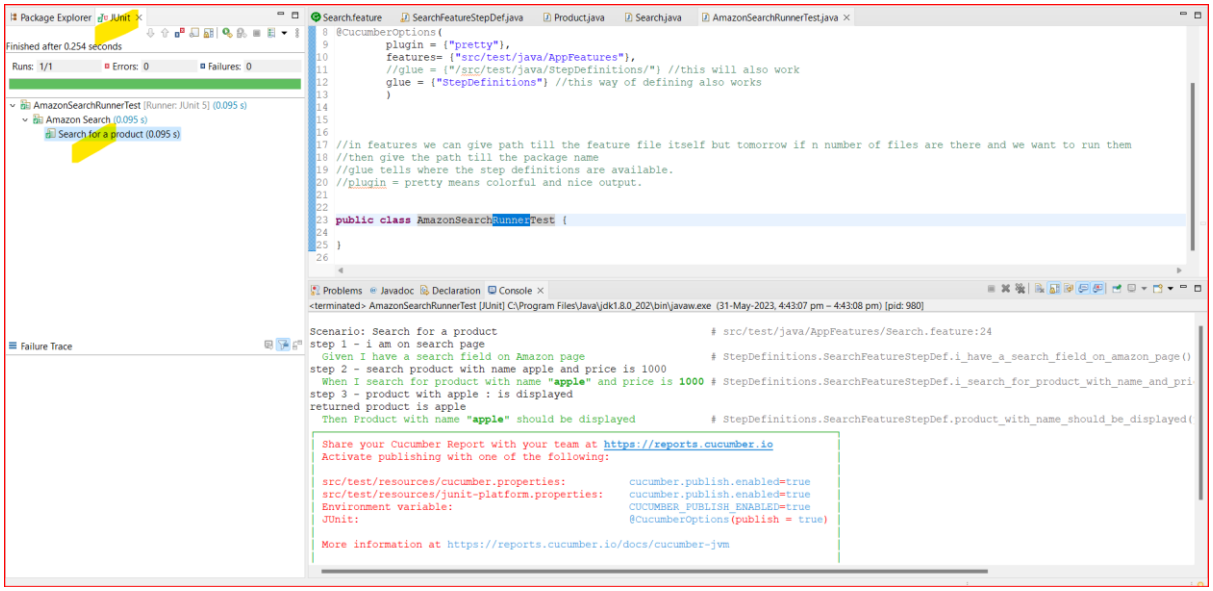

Now if we see we were not getting output when we ran using mvn we are only getting build success. No mention of steps passed/failed etc.

Junit needs the runner class or any class we run to end with name “Test”. This is pre-configured for junit that all the classes should end with the word “Test” in their name.

Changed name of the runner class file name:

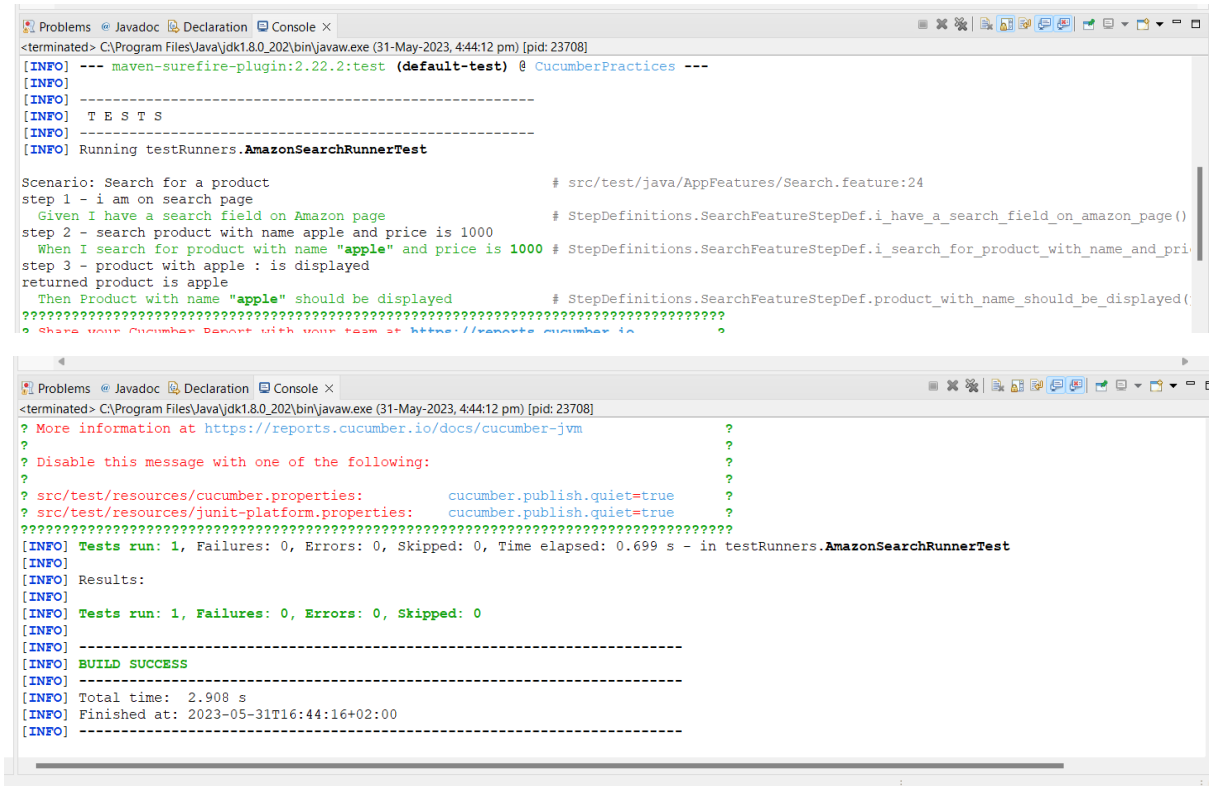
```
22
23 public class AmazonSearchRunnerTest {
24
25 }
```

Now run using junit test:



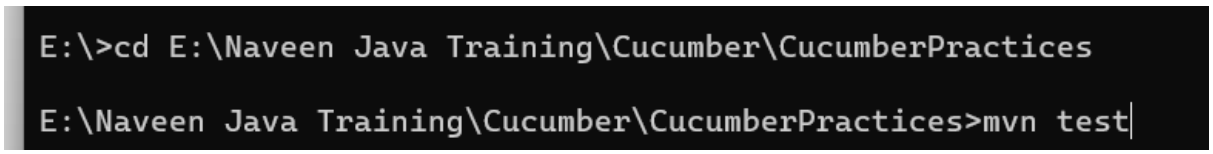

We can also right click on runner file and run as maven test. This will also work.

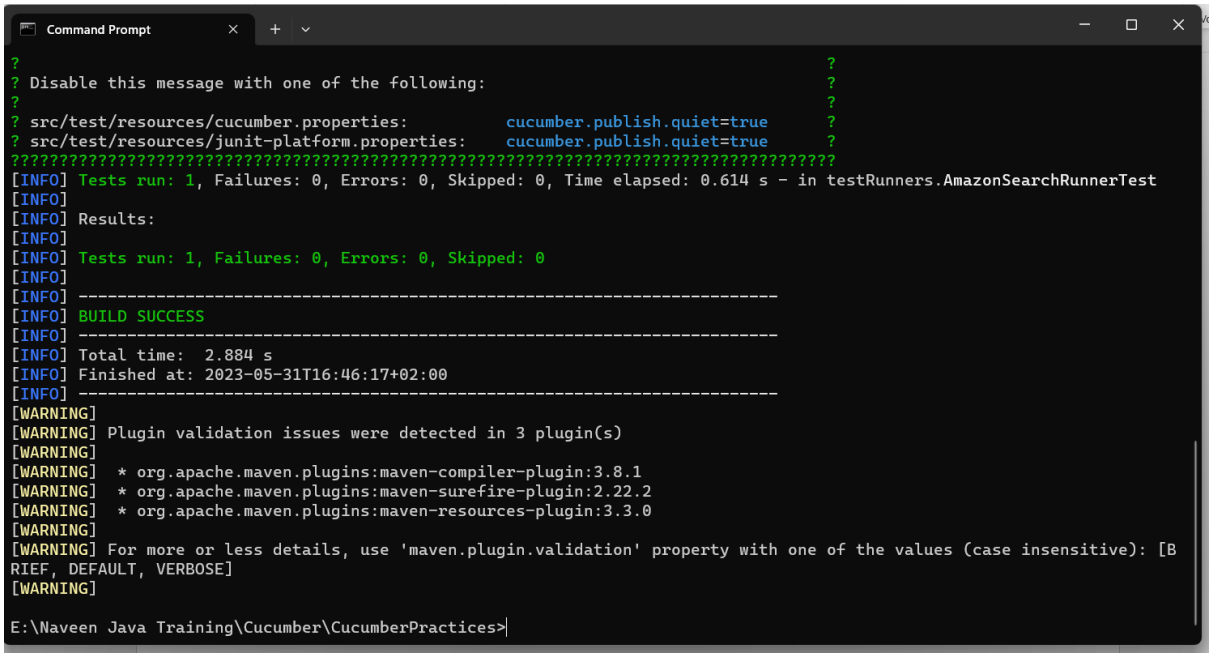
We are getting how much passed, features details etc.




Test run is 1, no failures, all passed. No errors, no skips.

Run from cmd:






Same bdd can be used for api automation.

**Given** will have the http verb call (get/put/post/delete).

**When** the “payload” is passed.

**Then** I get 200 response code or the response time.

We can use bdd for ui automation, api automation, mobile testing, unit testing, regression testing.

## Codes-

Pom file:

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

<project xmlns="http://maven.apache.org/POM/4.0.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>

	<groupId>CucumberYouTubeSeries</groupId>
	<artifactId>CucumberPractices</artifactId>
	<version>0.0.1-SNAPSHOT</version>

	<name>CucumberPractices</name>
	<!-- FIXME change it to the project's website -->
	<url>http://www.example.com</url>

	<properties>
		<project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
		<java.version>1.8</java.version>
		<junit.version>4.13.1</junit.version>
		<cucumber.version>6.9.0</cucumber.version>
		<maven.compiler.version>3.8.1</maven.compiler.version>
```

```
            <maven.surefire.version>2.22.2</maven.surefire.version>
      </properties>

      <dependencies>
            <dependency>
                  <groupId>io.cucumber</groupId>
                  <artifactId>cucumber-java</artifactId>
                  <version>${cucumber.version}</version>
                  <scope>test</scope>
            </dependency>

            <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-
junit -->
            <dependency>
                  <groupId>io.cucumber</groupId>
                  <artifactId>cucumber-junit</artifactId>
                  <version>${cucumber.version}</version>
                  <scope>test</scope>
            </dependency>

            <!-- https://mvnrepository.com/artifact/junit/junit -->
            <dependency>
                  <groupId>junit</groupId>
                  <artifactId>junit</artifactId>
                  <version>${junit.version}</version>
                  <scope>test</scope>
            </dependency>
      </dependencies>

      <build>
            <plugins>
                  <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-compiler-plugin</artifactId>
                        <version>${maven.compiler.version}</version>
                        <configuration>
                              <encoding>UTF-8</encoding>
                              <source>${java.version}</source>
                              <target>${java.version}</target>
                        </configuration>
                  </plugin>
                  <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-surefire-plugin</artifactId>
                        <version>${maven.surefire.version}</version>
                  </plugin>
            </plugins>
      </build>
</project>
```

Feature file:

```
#Author: your.email@your.domain.com
#Keywords Summary :
#Feature: List of scenarios.
#Scenario: Business rule through list of steps with arguments.
#Given: Some precondition step
#When: Some key actions
#Then: To observe outcomes or validation
#And,But: To enumerate more Given,When,Then steps
```

```
#Scenario Outline: List of steps for data-driven as an Examples and
<placeholder>
#Examples: Container for s table
#Background: List of steps run before each of the scenarios
#""" (Doc Strings)
#| (Data Tables)
#@ (Tags/Labels):To group Scenarios
#<> (placeholder)
#""
## (Comments)
#Sample Feature Definition Template


Feature: Amazon Search
#this is the feature for which we want to write code and requirements
#one feature file can have multiple scenarios
Scenario: Search for a product
#here we give the scenario name
#below scenario (given, when, then, and) etc are known as steps
Given I have a search field on Amazon page
#given can be considered as pre-condition
When I search for product with name "apple" and price is 1000
#string is in double quotes
Then Product with name "apple" should be displayed
```

Step definition file:

```
package StepDefinitions;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import junit.framework.Assert;
import amazonImplementation.Product;
import amazonImplementation.Search;

public class SearchFeatureStepDef {

	Product product;
	Search search;

	@Given("I have a search field on Amazon page")
	public void i_have_a_search_field_on_amazon_page() {
	    System.out.println("step 1 - i am on search page");
	}

	@When("I search for product with name {string} and price is {int}")
	public void i_search_for_product_with_name_and_price_is(String
productName, Integer price) {
	   System.out.println("step 2 - search product with name " +
productName + " and price is " + price);

	   product =new Product(productName, price);
	}

	@Then("Product with name {string} should be displayed")
	public void product_with_name_should_be_displayed(String
productName) {
```

```
            System.out.println("step 3 - product with " + productName + " :
is displayed");
            search=new Search();
            String productNameReturned=search.displayProductName(product);
            System.out.println("returned product is " +
productNameReturned);
            Assert.assertEquals(product.getProductName(),
productNameReturned);
        }


}
```

Product class:

```
package amazonImplementation;

import java.util.ArrayList;
import java.util.List;

public class Product {

        private String productName;
        private int price;

        public Product(String productName, int price) {
                this.productName = productName;
                this.price = price;
        }

        public String getProductName() {
                return productName;
        }

        public void setProductName(String productName) {
                this.productName = productName;
        }

        public int getPrice() {
                return price;
        }

        public void setPrice(int price) {
                this.price = price;
        }

        // this method will list of products in the form of string
        public List<String> getProductList() {
                List<String> prodList = new ArrayList<>();
                prodList.add("apple");
                prodList.add("hp");
                prodList.add("samsung");
                prodList.add("bourbon");
```

```
                return prodList;
        }

}
```

Search class:

```
package amazonImplementation;

public class Search {

      public String displayProductName(Product product) { // here we will
take the product name returned from
            // product.java class
            if
(product.getProductList().contains(product.getProductName())) {
                  return product.getProductName();
            } else {
                  return null;
            }
            // or we can simply write, because if above return is
satisfied that will be the
            // thing which will be returned to method
            // return null;
      }

}
```

Test runner:

```
package testRunners;

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

@RunWith(Cucumber.class)
@CucumberOptions(
                plugin = {"pretty"},
                features= {"src/test/java/AppFeatures"},
                //glue = {"/src/test/java/StepDefinitions/"} //this will also work
                glue = {"StepDefinitions"} //this way of defining also works
                )



//in features we can give path till the feature file itself but tomorrow if n number of files are there
and we want to run them
//then give the path till the package name
//glue tells where the step definitions are available.
//plugin = pretty means colorful and nice output.


public class AmazonSearchRunnerTest {
```

```
}
```

Output of the code:

```
Jun 02, 2023 5:19:02 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use
io.cucumber.core.cli.Main

Scenario: Search for a product                                    #
src/test/resources/AppFeatures/Search.feature:24
step 1 - i am on search page
  Given I have a search field on Amazon page                     #
StepDefinitions.SearchFeatureStepDef.i_have_a_search_field_on_amazon_page
()
step 2 - search product with name apple and price is 1000
  When I search for product with name "apple" and price is 1000 #
StepDefinitions.SearchFeatureStepDef.i_search_for_product_with_name_and_p
rice_is(java.lang.String,java.lang.Integer)
step 3 - product with apple : is displayed
returned product is apple
  Then Product with name "apple" should be displayed             #
StepDefinitions.SearchFeatureStepDef.product_with_name_should_be_displaye
d(java.lang.String)

1 Scenarios (1 passed)
3 Steps (3 passed)
0m0.468s


┌──────────────────────────────────────────────────────────────────────
──────────┐
│ Share your Cucumber Report with your team at
https://reports.cucumber.io          │
│ Activate publishing with one of the following:
│
│
│
│ src/test/resources/cucumber.properties:
cucumber.publish.enabled=true    │
│ src/test/resources/junit-platform.properties:
cucumber.publish.enabled=true    │
│ Environment variable:
CUCUMBER_PUBLISH_ENABLED=true    │
│ JUnit:
@CucumberOptions(publish = true) │
│
│
│ More information at https://reports.cucumber.io/docs/cucumber-jvm
│
│
│
│ Disable this message with one of the following:
│
│
│
│ src/test/resources/cucumber.properties:
cucumber.publish.quiet=true      │
│ src/test/resources/junit-platform.properties:
cucumber.publish.quiet=true      │
└──────────────────────────────────────────────────────────────────────
──────────┘
```