

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.

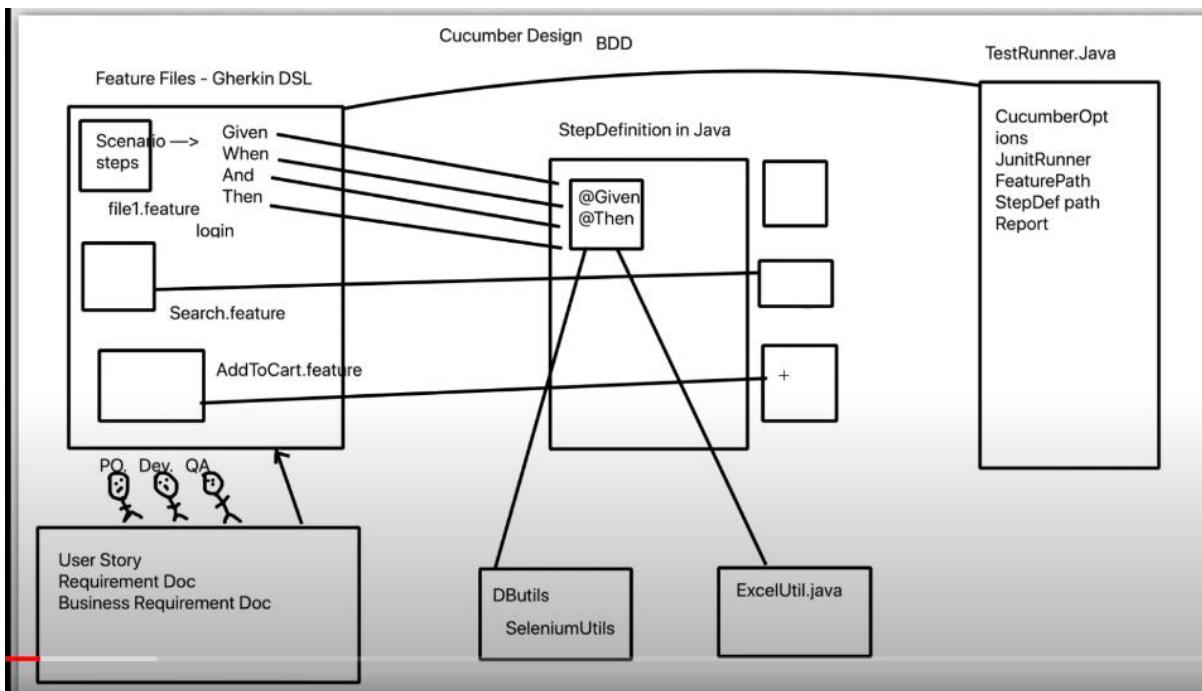
We can have “n” number of step definition files for different features like login, search, add to cart etc and map them to the respective feature file.

To run the code-

We can use individual feature files or use runner class.

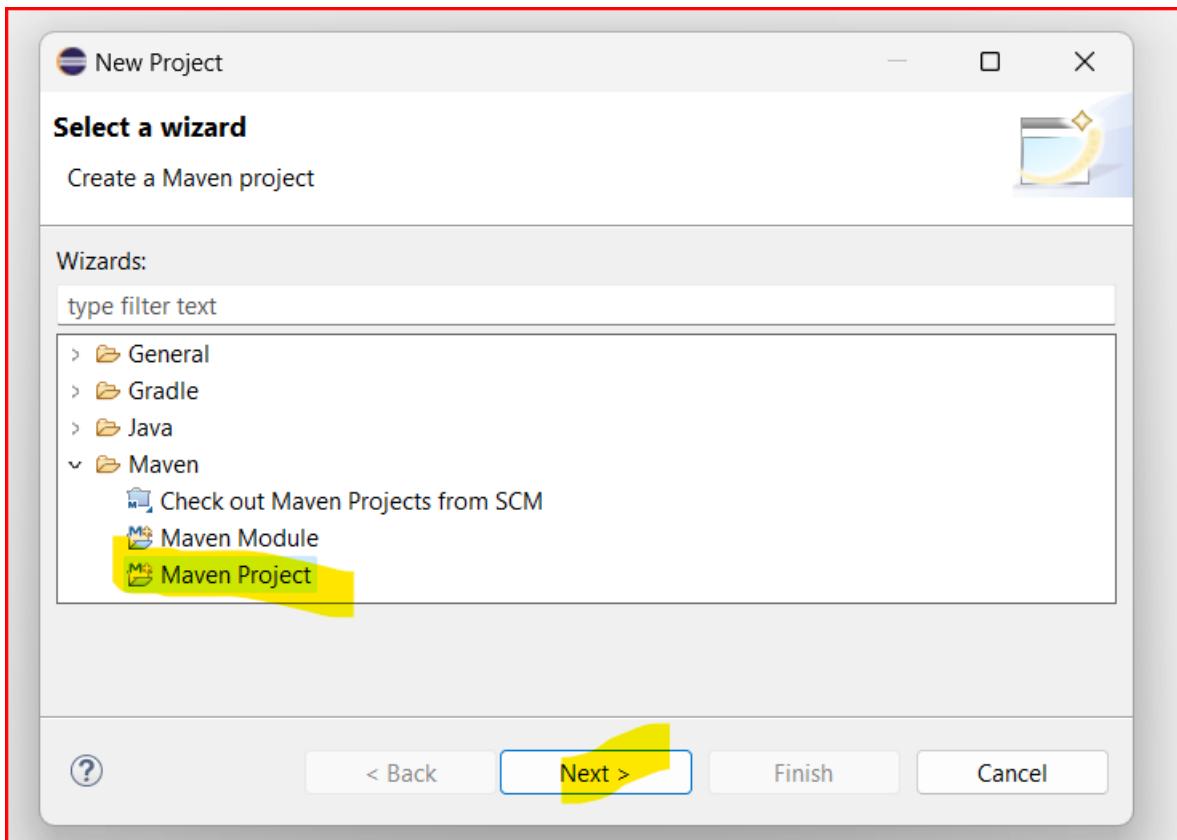
Test runner –

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

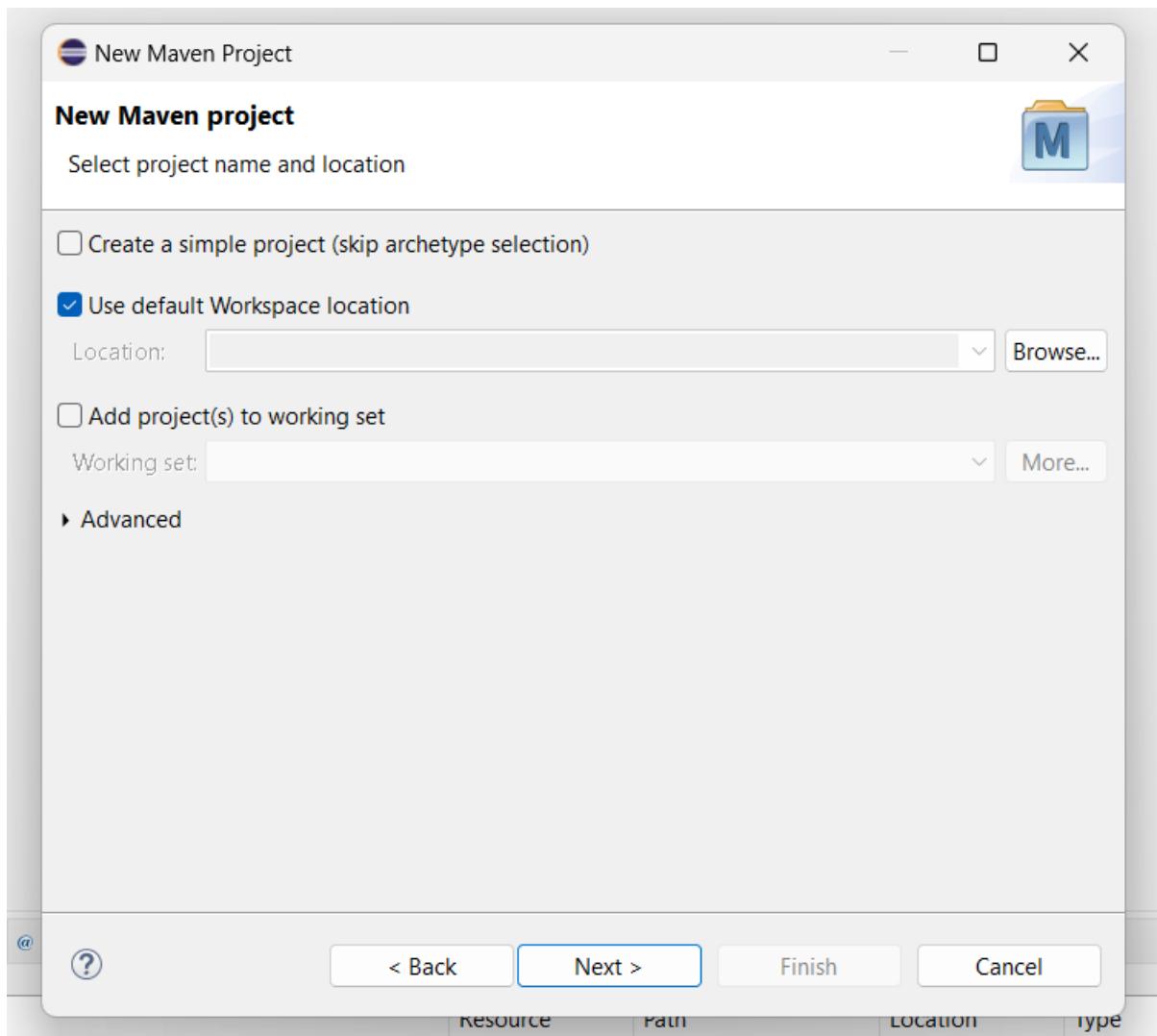


Let's start cucumber project-

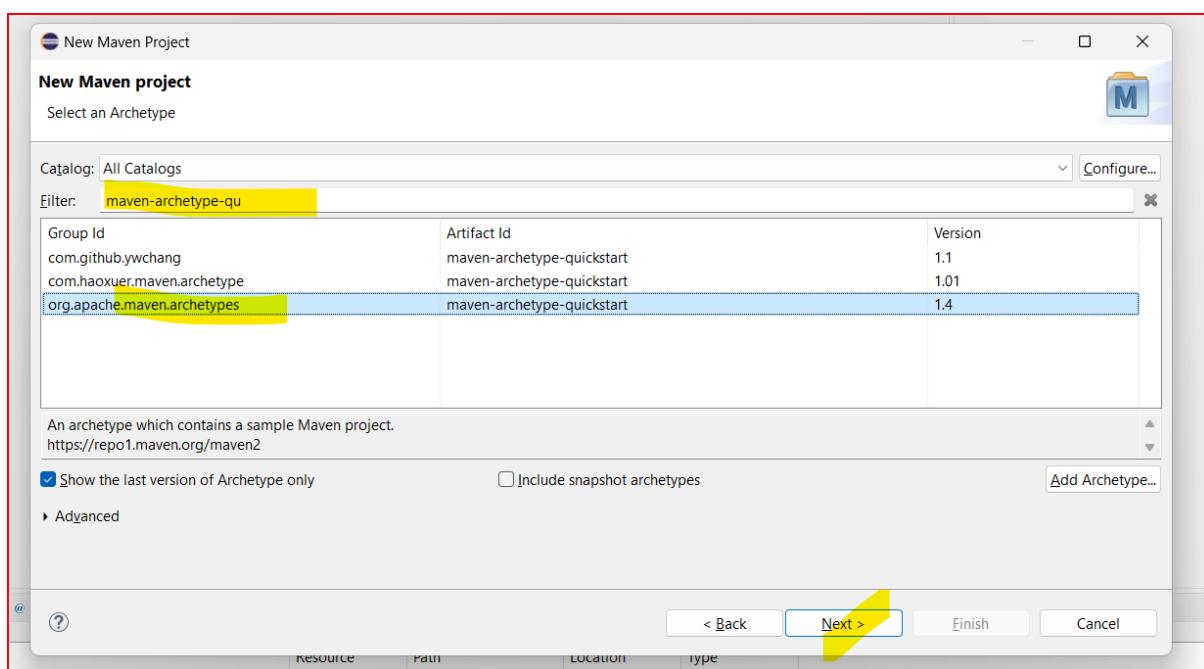
Maven project creation:



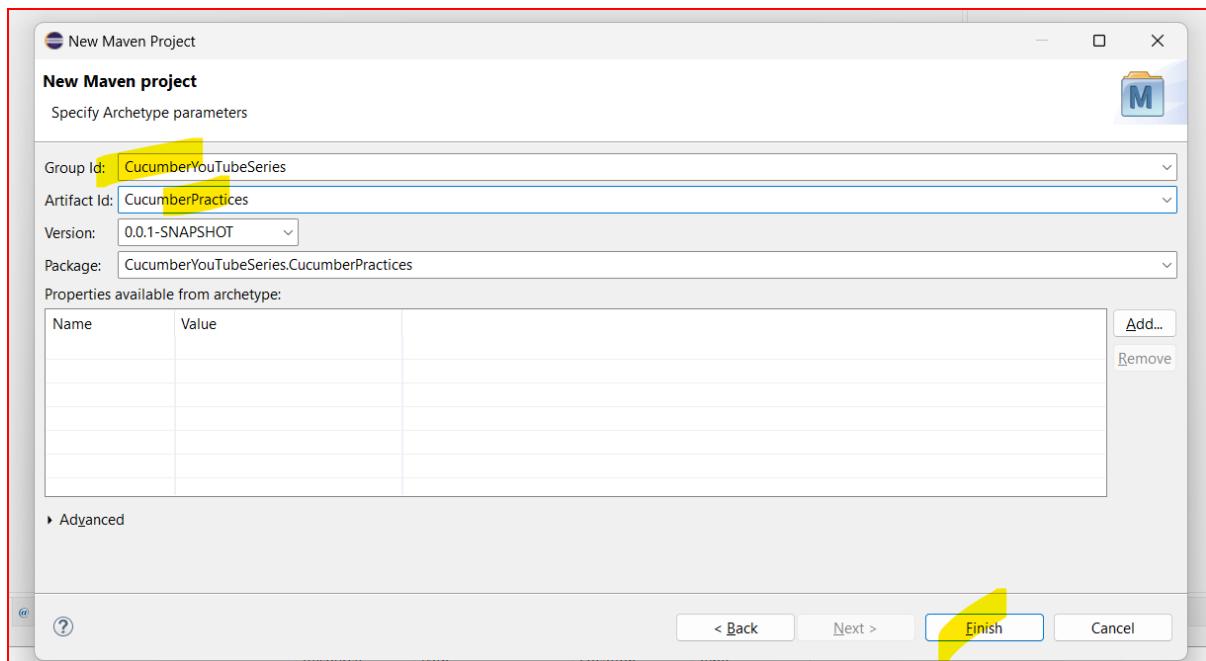
Click next:



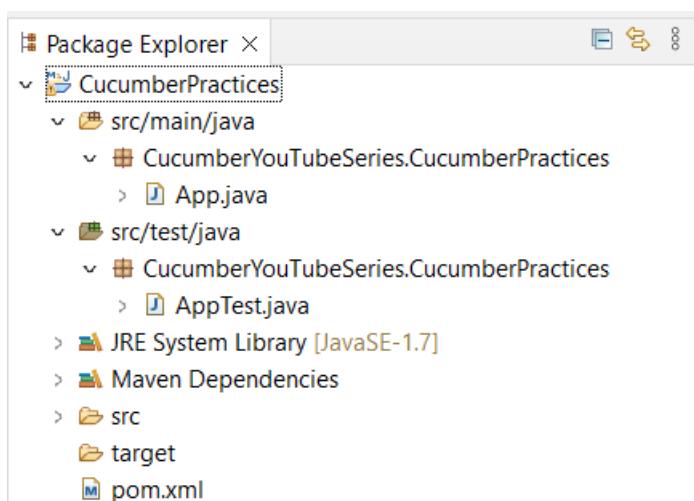
Search for maven plugin:



Give group id, artifact id:



Base project structure once created:



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

The screenshot shows a Google search results page for the query "cucumber eclipse plugin". A yellow box highlights the search bar. Below it, a red box highlights the first search result, which is a link to the Eclipse Marketplace. The result title is "Cucumber Eclipse Plugin". A yellow box highlights the plugin's description: "Nov 10, 2017 — This plugin's snapshot version(1.0.0.202001082311) can be installed directly from your Eclipse-IDE's Menu, Help > Eclipse Marketplace > Find: ...". Below this, another yellow box highlights the URL "https://marketplace.eclipse.org / category / free-tagging". The page also lists "Lambda Expression support for Cucumber-Java8 Content Assistance ...".

Lots of good features

The screenshot shows the Eclipse Marketplace interface. A yellow box highlights the search bar at the top. Below it, a red box highlights the "Cucumber Eclipse Plugin" entry in the search results. The page displays the plugin's details, including its name, rating (330 stars), download count (41), and an "Install" button. A yellow box highlights the "Details" tab. The description section mentions "Lambda Expression support for Cucumber-Java8" and other features like "Content Assistance for feature file" and "New Step Definition File Wizard".

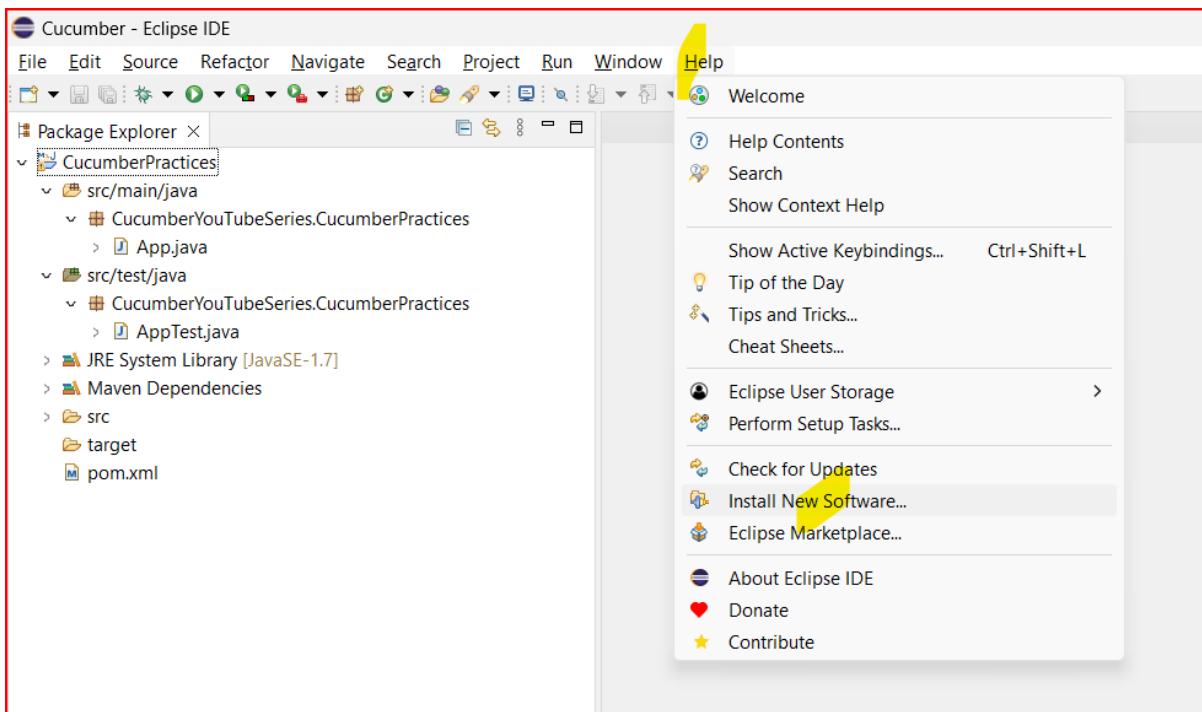
One way to install is drag and drop into eclipse.

But we will use the below link-

[Cucumber-Eclipse > Update Site](#)

Copy the browser url.

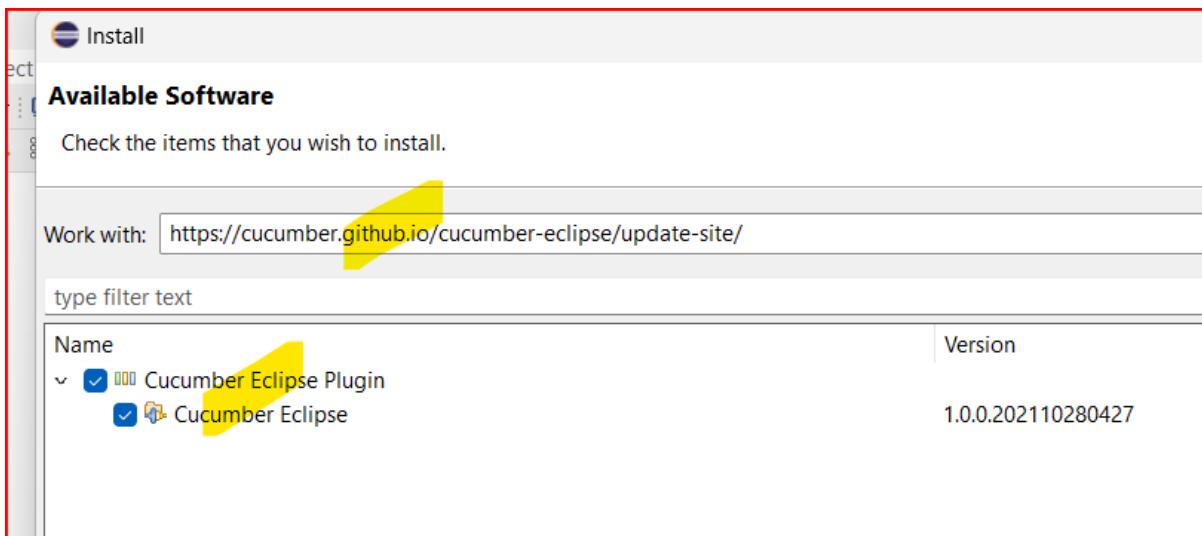
Open eclipse and go to the below location:



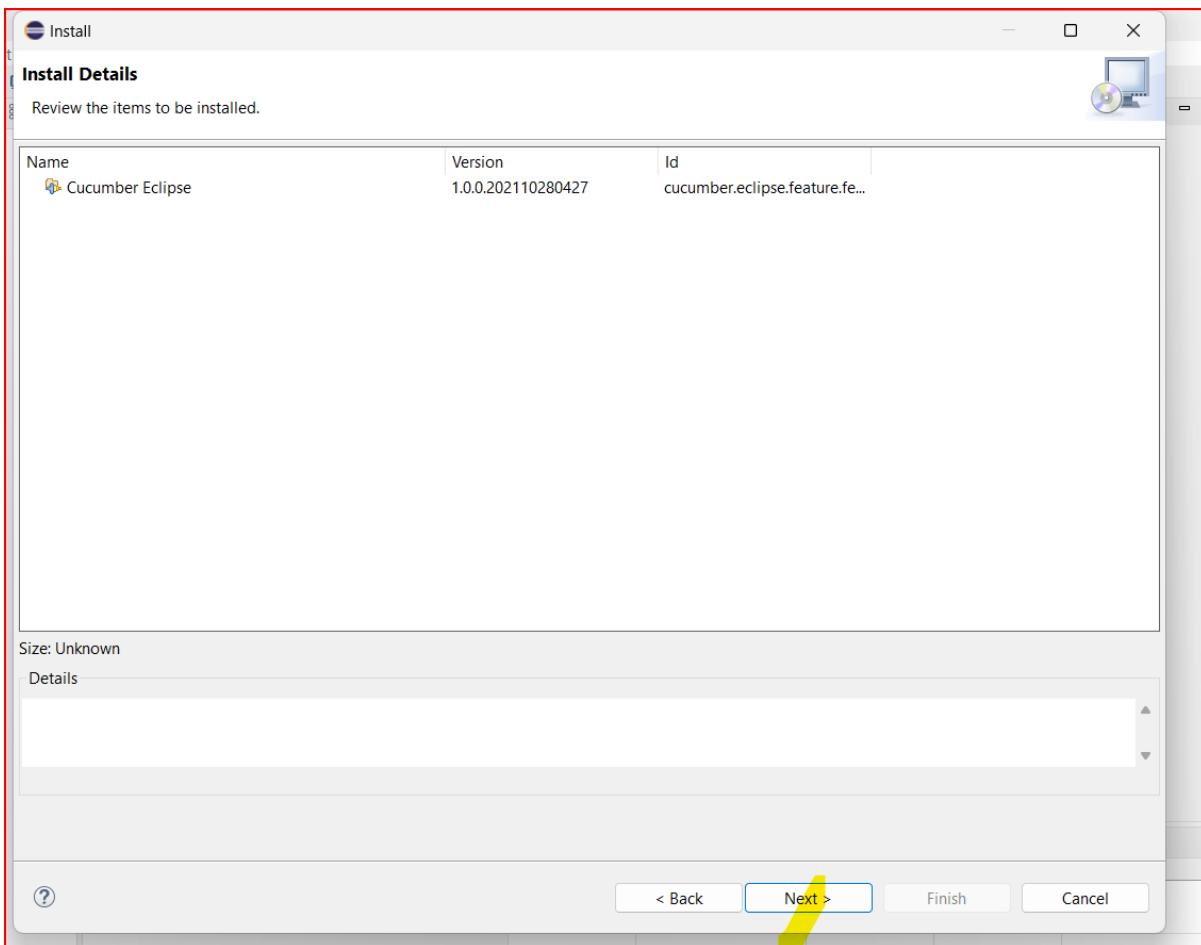
Enter the link.

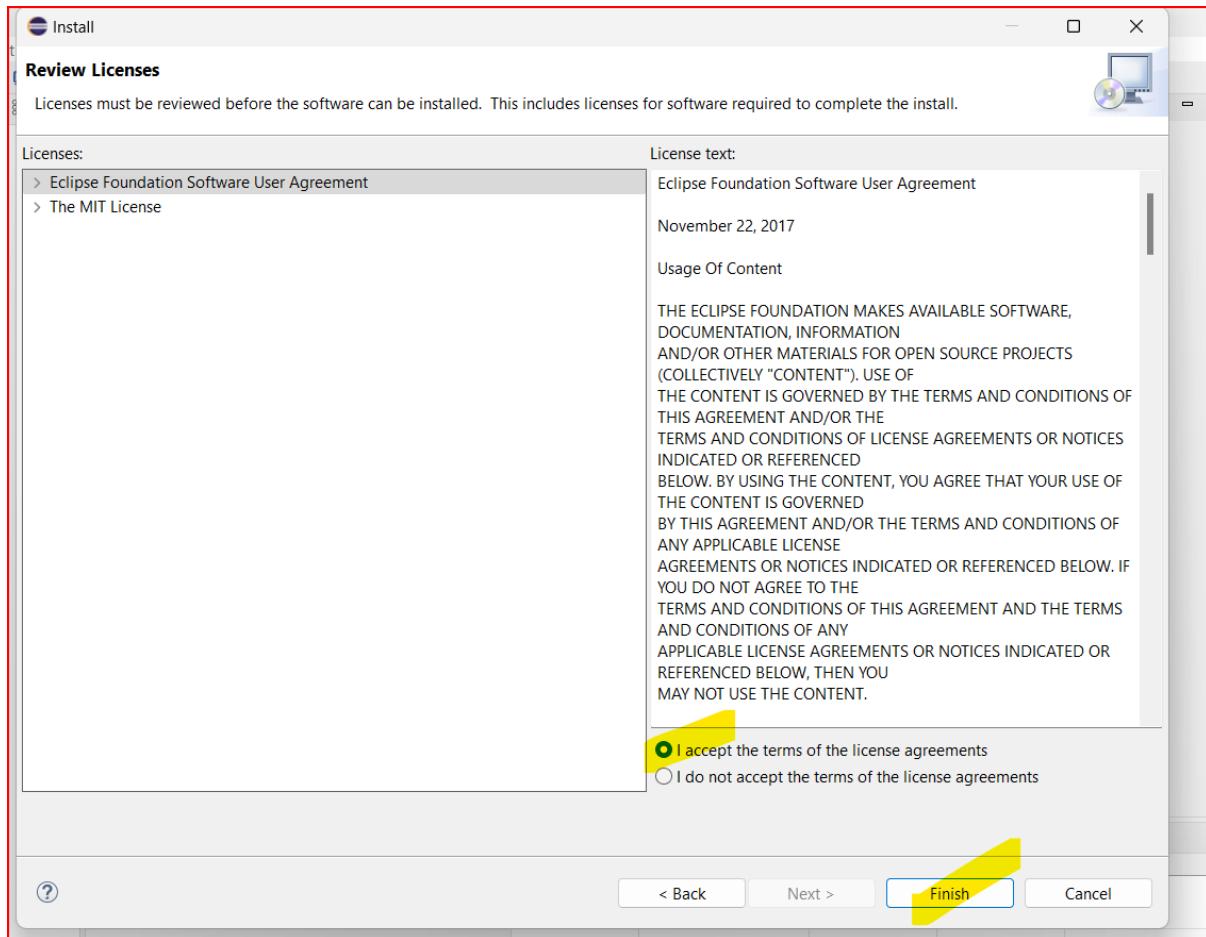
Hit enter.

Select all the checkboxes inside Name field.

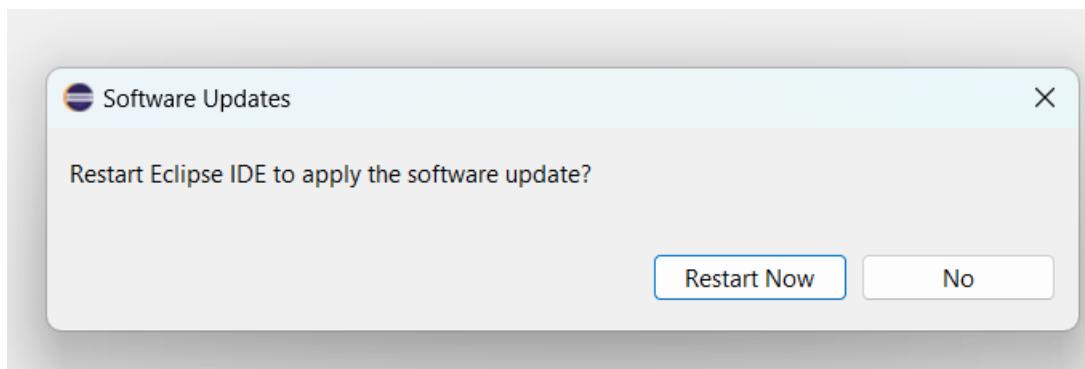


Click next.

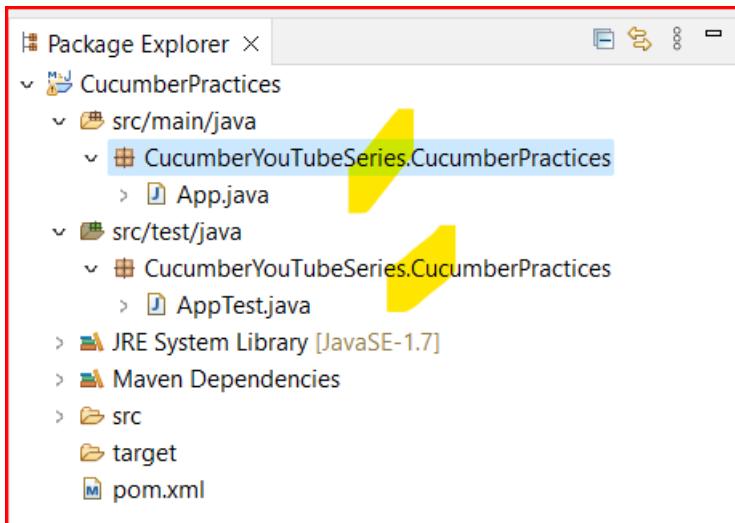




You may have to restart eclipse for it to work.



Delete the dummy packages:



Let's add the required dependencies for project-

Search in google:

cucumber java skeleton

All Videos Shopping Images News More

About 2,070,000 results (0.32 seconds)

[GitHub](https://github.com/cucumber/cucumber-java-skeleton)
https://github.com/cucumber/cucumber-java-skeleton... ::

Cucumber Java Skeleton

Cucumber-Java Skeleton. This is the simplest possible build script setup for Cucumber using Java. There is nothing fancy like a webapp or browser testing.

<https://github.com/cucumber/cucumber-java-skeleton/blob/master/pom.xml> ::

cucumber-java-skeleton/pom.xml at master

Launch the application as in 'production' using the fatjar. We pass the ...

Click maven:

<https://github.com/cucumber/cucumber-java-skeleton/tree/main>

Product Solutions Open Source Pricing

cucumber / cucumber-java-skeleton Public

Code Issues 2 Pull requests Actions Projects Security Insights

main ▾ 2 branches 11 tags

- renovate[bot] Update dependency maven to v3.9.2 ✓ 75573c3 3 weeks ago
- .github Deduplicate renovate configuration
- gradle Update dependency io.cucumber:cucumber-bom to v7.12.0
- maven Update dependency maven to v3.9.2
- .gitignore Split Gradle and Maven Projects
- LICENCE Add licence
- README.md Split Gradle and Maven Projects

Click pom.xml:

<https://github.com/cucumber/cucumber-java-skeleton/tree/main/maven>

Product Solutions Open Source Pricing

cucumber / cucumber-java-skeleton Public

Code Issues 2 Pull requests Actions Projects Security Insights

main ▾ cucumber-java-skeleton / maven /

- renovate[bot] Update dependency maven to v3.9.2
- ..
- .mvn/wrapper Update dependency maven to v3.9.2
- src Split Gradle and Maven Projects
- .gitignore Split Gradle and Maven Projects
- mvnw Update dependency maven-wrapper to v3.2.0
- mvnw.cmd Update dependency maven-wrapper to v3.2.0
- pom.xml Update dependency org.apache.maven.plugins:maven-surefire-plugin to v...

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.

```

20     <dependencies>
21         <dependency>
22             <groupId>io.cucumber</groupId>
23             <artifactId>cucumber-java</artifactId>
24             <version>${cucumber.version}</version>
25             <scope>test</scope>
26         </dependency>
27
28         <dependency>
29             <groupId>io.cucumber</groupId>
30             <artifactId>cucumber-junit</artifactId>
31             <version>${cucumber.version}</version>
32             <scope>test</scope>
33         </dependency>
34
35         <dependency>
36             <groupId>junit</groupId>
37             <artifactId>junit</artifactId>
38             <version>${junit.version}</version>
39             <scope>test</scope>
40         </dependency>
41     </dependencies>
42

```

Copy them.

Remove existing dependency from pom and paste the copied one.

```

44
25@    <dependencies>
26@      <dependency>
27@        <groupId>io.cucumber</groupId>
28@        <artifactId>cucumber-java</artifactId>
29@        <version>${cucumber.version}</version>
30@        <scope>test</scope>
31@      </dependency>
32
33@      <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -->
34@      <dependency>
35@        <groupId>io.cucumber</groupId>
36@        <artifactId>cucumber-junit</artifactId>
37@        <version>${cucumber.version}</version>
38@        <scope>test</scope>
39@      </dependency>
40
41@      <!-- https://mvnrepository.com/artifact/junit/junit -->
42@      <dependency>
43@        <groupId>junit</groupId>
44@        <artifactId>junit</artifactId>
45@        <version>${junit.version}</version>
46@        <scope>test</scope>
47@      </dependency>
48  </dependencies>
49
-- . . .

```

Copy the plugins from build section.

Taken screenshot of Naveen's computer.

```

43    <build>
44      <plugins>
45        <plugin>
46          <groupId>org.apache.maven.plugins</groupId>
47          <artifactId>maven-compiler-plugin</artifactId>
48          <version>${maven.compiler.version}</version>
49          <configuration>
50            <encoding>UTF-8</encoding>
51            <source>${java.version}</source>
52            <target>${java.version}</target>
53          </configuration>
54        </plugin>
55        <plugin>
56          <groupId>org.apache.maven.plugins</groupId>
57          <artifactId>maven-surefire-plugin</artifactId>
58          <version>${maven.surefire.version}</version>
59        </plugin>
60      </plugins>
61    </build>
62  </project>

```

Remove existing build from pom.

Paste the build here.

```

49
50<build>
51    <plugins>
52        <plugin>
53            <groupId>org.apache.maven.plugins</groupId>
54            <artifactId>maven-compiler-plugin</artifactId>
55            <version>${maven.compiler.version}</version>
56        <configuration>
57            <encoding>UTF-8</encoding>
58            <source>${java.version}</source>
59            <target>${java.version}</target>
60        </configuration>
61    </plugin>
62    <plugin>
63        <groupId>org.apache.maven.plugins</groupId>
64        <artifactId>maven-surefire-plugin</artifactId>
65        <version>${maven.surefire.version}</version>
66    </plugin>
67 </plugins>
68 </build>
69 </project>
70

```

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.

```

70
47 <properties>
48     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
49     <java.version>1.8</java.version>
50     <junit.version>4.13.1</junit.version>
51     <cucumber.version>6.9.0</cucumber.version>
52     <maven.compiler.version>3.8.1</maven.compiler.version>
53     <maven.surefire.version>2.22.2</maven.surefire.version>
54 </properties>
55
56 <dependencies>
57     <dependency>
58         <groupId>io.cucumber</groupId>
59         <artifactId>cucumber-java</artifactId>
60         <version>${cucumber.version}</version>
61         <scope>test</scope>
62     </dependency>
63

```

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,

The screenshot shows the MVN Repository search interface at <https://mvnrepository.com/search?q=junit-platform-suite>. A yellow highlight is placed over the search bar containing 'junit-platform-suite'. The search results page displays 17795 results. The first result is 'JUnit Jupiter API' from org.junit.jupiter, which is the API for writing tests using JUnit 5. It was last released on May 13, 2023. The second result is 'JUnit' from junit. The third result is 'JUnit Platform Suite API' from org.junit.platform. The fourth result is 'JUnit Platform Suite (Aggregator)' from org.junit.platform, which is the module "junit-platform-suite" of JUnit 5.

Repository	Group	Category
Central	org.xwiki	
Sonatype	janstey.sources	
Spring Plugins	com.github	
Spring Lib M	org.nuxeo	
XWiki Releases	org.eclipse	
Redhat GA	io.github	
JCenter	org.apache	
Nuxeo	com.google	

Found 17795 results

Sort: [relevance](#) | [popular](#) | [newest](#)

- 1. JUnit Jupiter API**
org.junit.jupiter » junit-jupiter-api
JUnit Jupiter is the API for writing tests using JUnit 5.
Last Release on May 13, 2023
- 2. JUnit**
junit » junit
JUnit is a unit testing framework to write and run repeatable automated tests on Java.
Last Release on Feb 13, 2021
- 3. JUnit Platform Suite API**
org.junit.platform » junit-platform-suite-api
Annotations for configuring a test suite on the JUnit Platform.
Last Release on May 13, 2023
- 4. JUnit Platform Suite (Aggregator)**
org.junit.platform » junit-platform-suite
Module "junit-platform-suite" of JUnit 5.

And get the version number:

Home » org.junit.platform » junit-platform-suite » 1.10.0-M1

 **JUnit Platform Suite (Aggregator) » 1.10.0-M1**

Module "junit-platform-suite" of JUnit 5.

License	EPL 2.0
Tags	junit testing platform
HomePage	https://junit.org/junit5/
Date	May 13, 2023
Files	pom (2 KB) jar (6 KB) View All
Repositories	Central
Ranking	#5499 in MvnRepository (See Top Artifacts)
Used By	70 artifacts

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

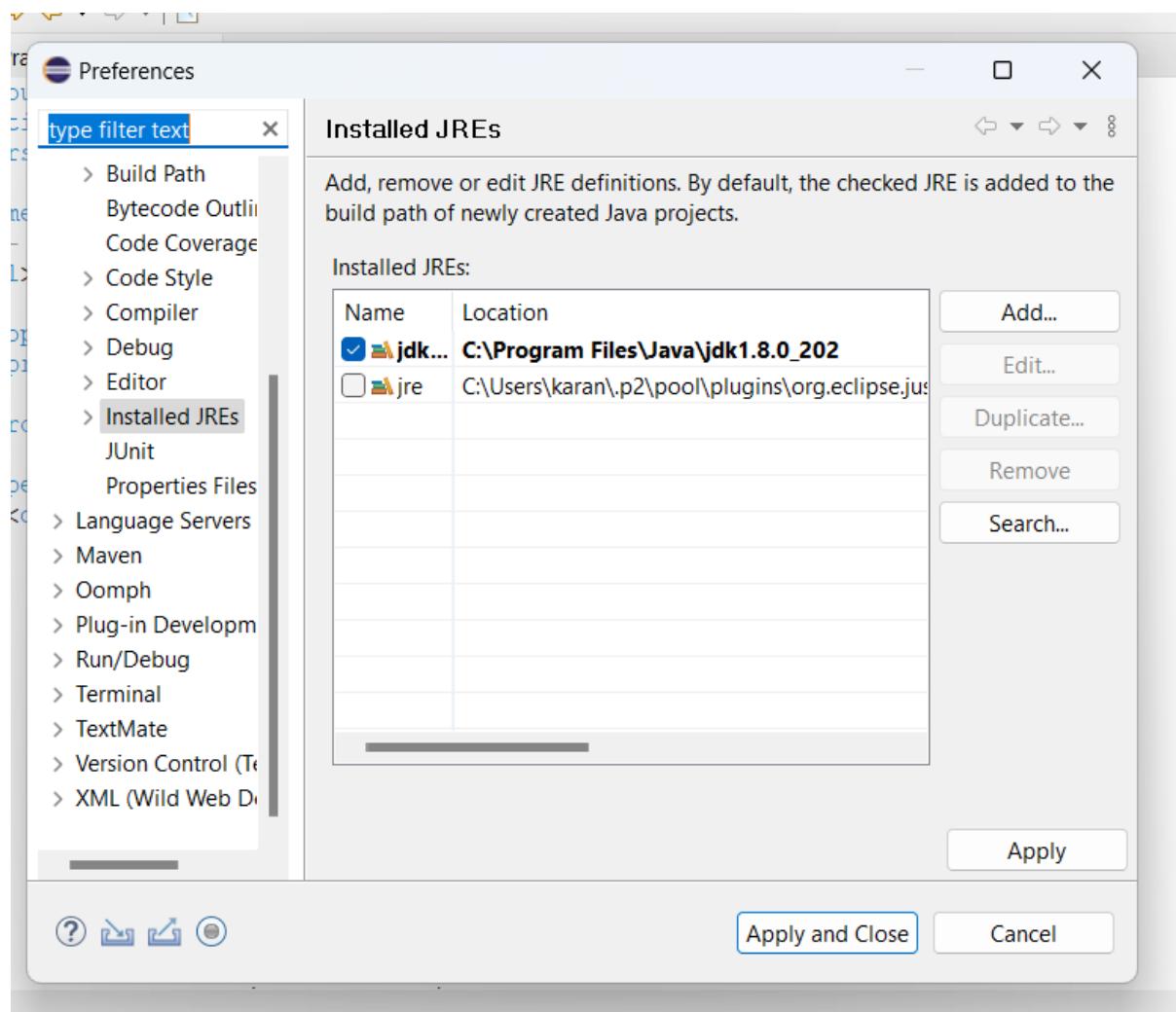
```
<!-- https://mvnrepository.com/artifact/org.junit.platform/junit-platform-suite -->
<dependency>
    <groupId>org.junit.platform</groupId>
    <artifactId>junit-platform-suite</artifactId>
    <version>1.10.0-M1</version>
    <scope>test</scope>
</dependency>
```

Include comment with link to declaration

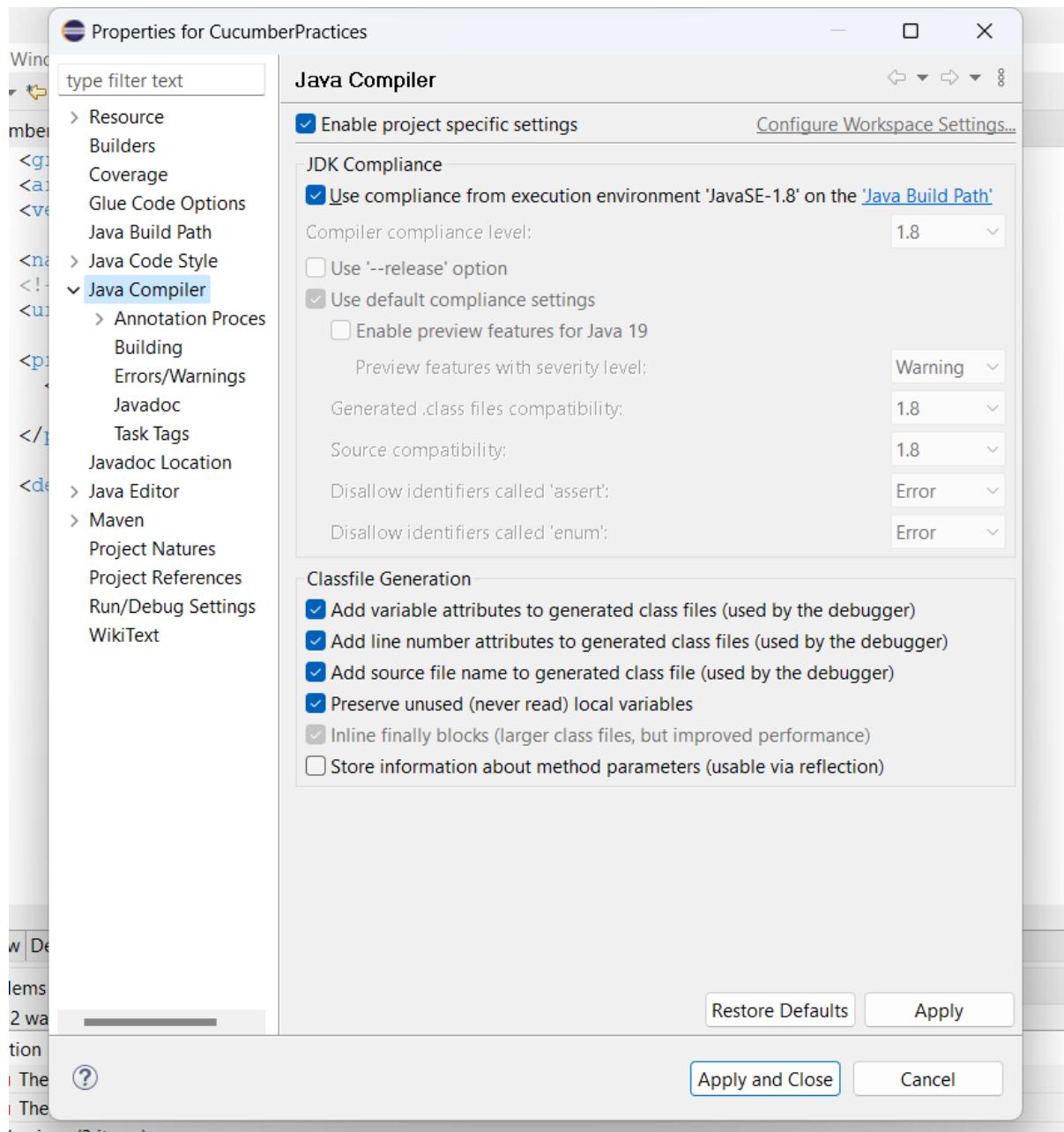
Compile Dependencies (1)

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.

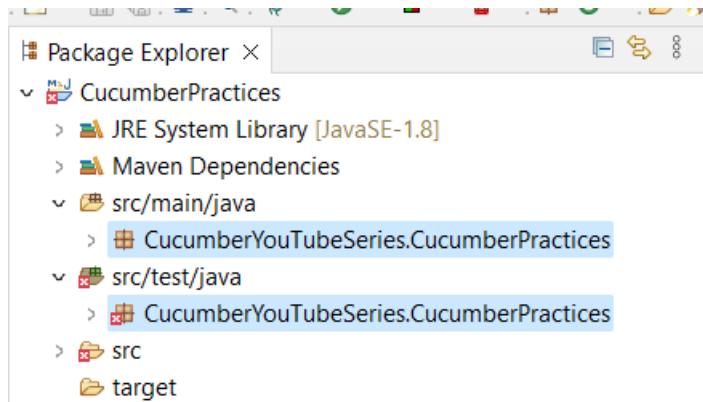


See in pom if the version matches.

```

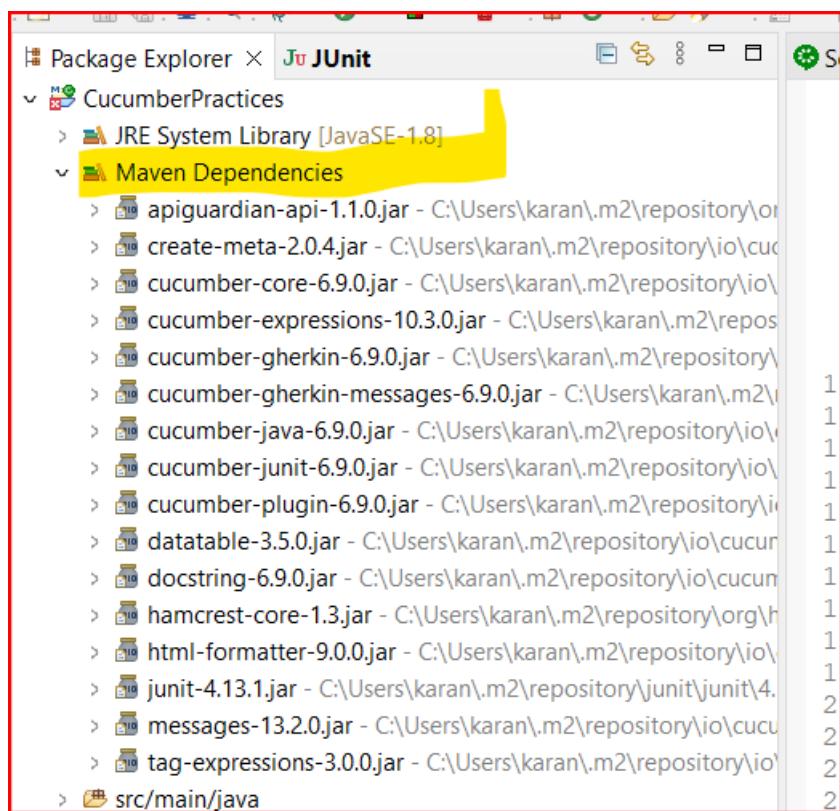
14
15 <properties>
16   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
17     <java.version>1.8</java.version>
18 </properties>
```

Remove the dummy packages:



These are created by default.

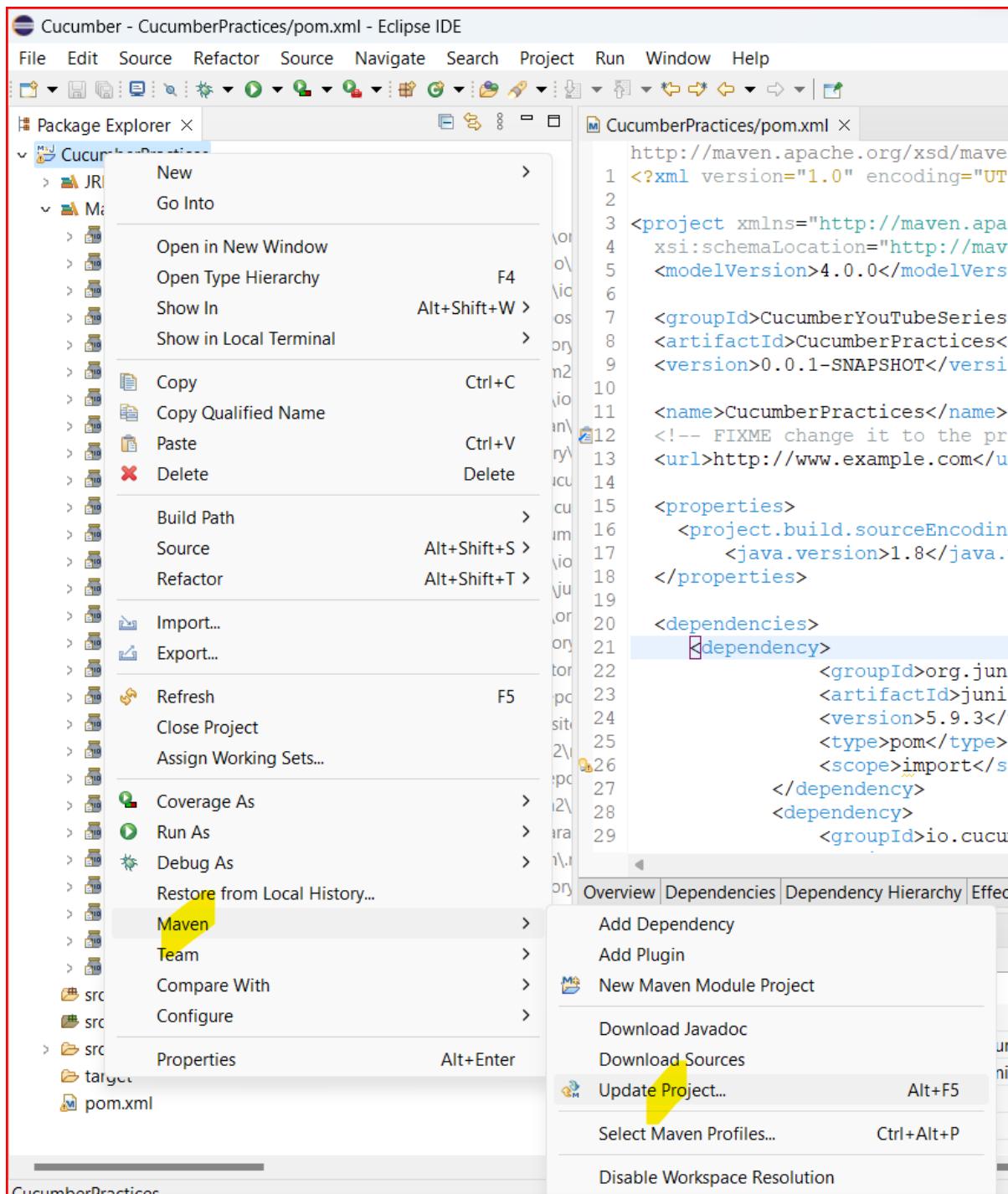
Expand maven dependencies and we can see cucumber present:

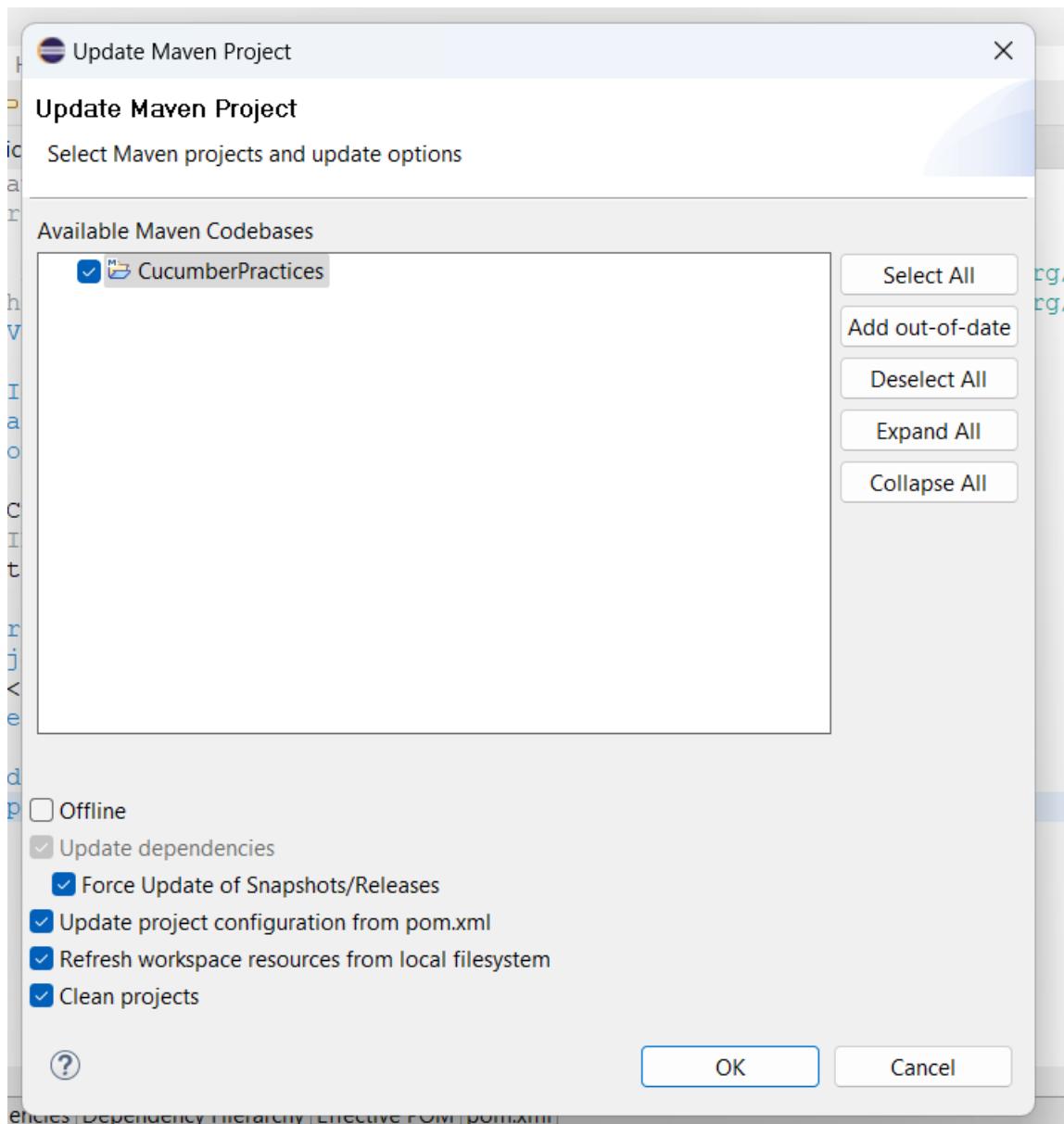


Build again the project:

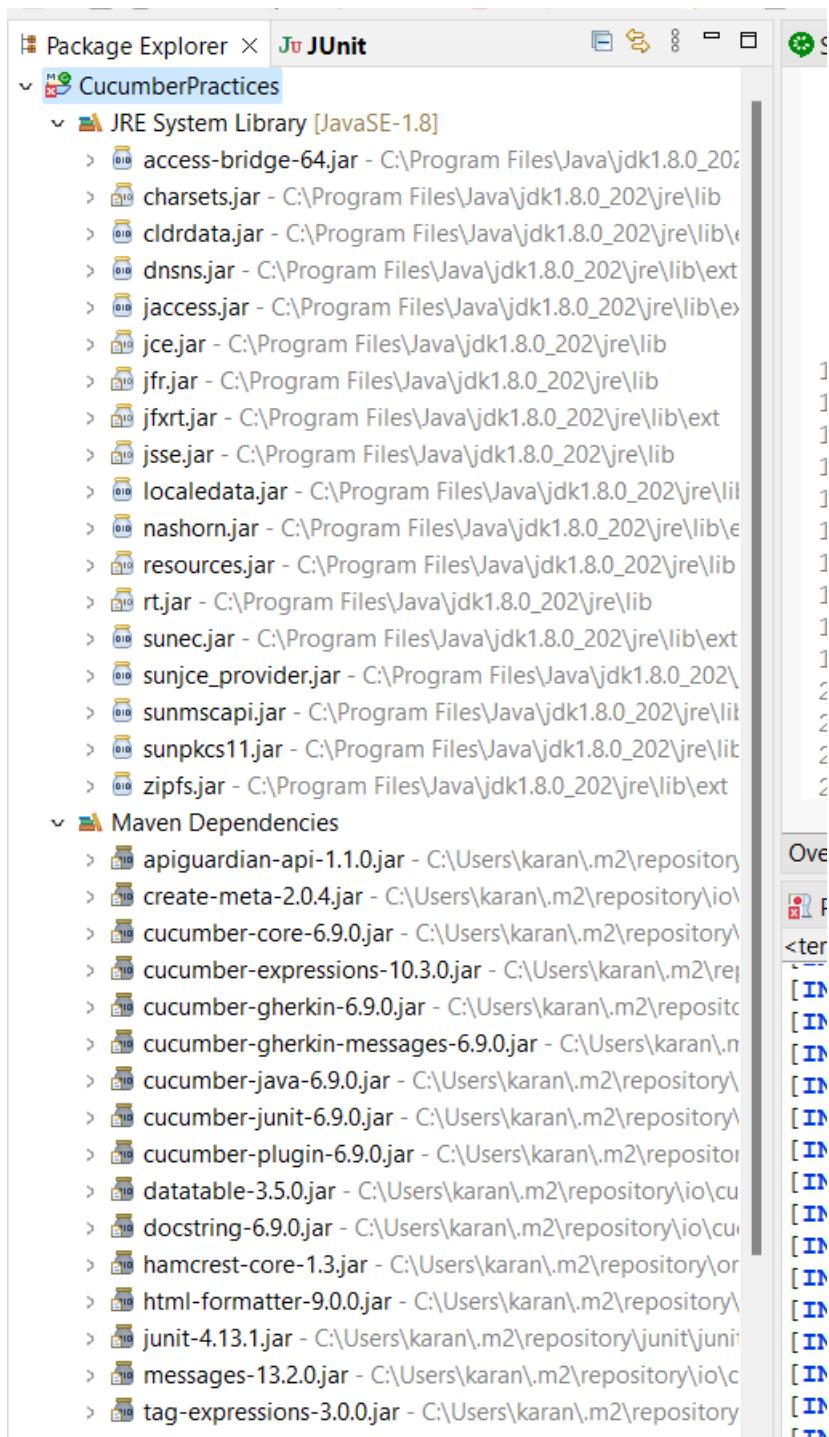
Right click -> maven -> update project.

Select force update and click ok.



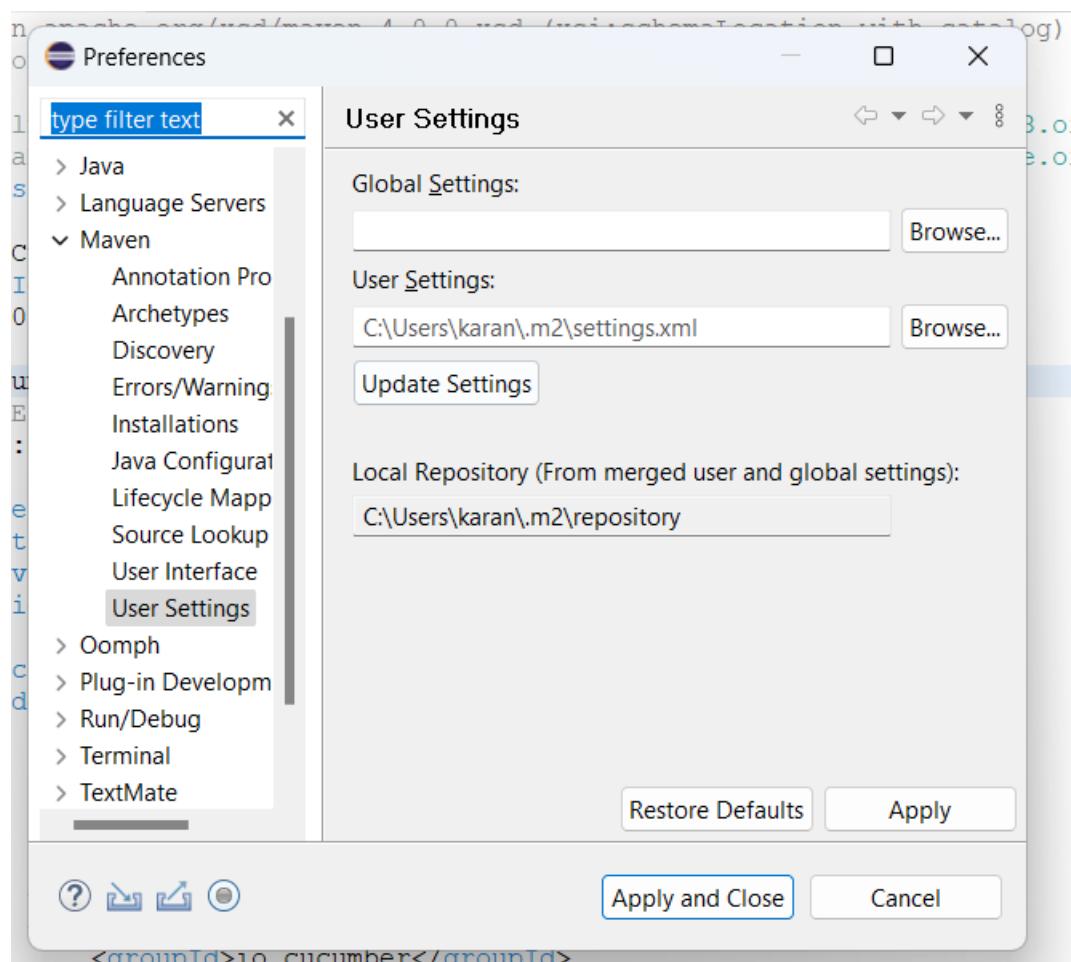


You can see the project being built in bottom right corner and all latest dependencies updated in jre and maven.



Where is .m2 located in eclipse-

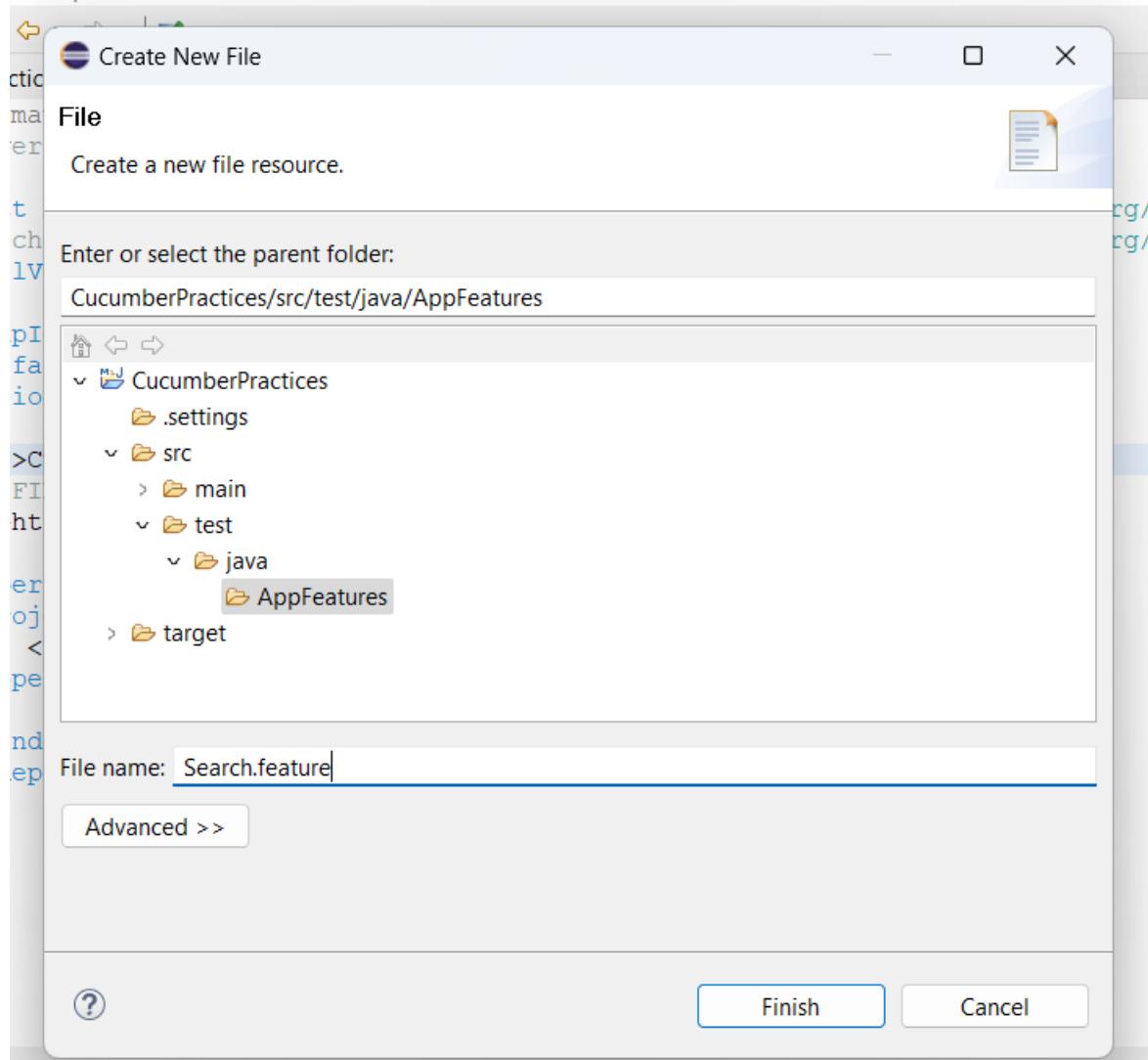
Windows -> preferences -> maven -> user settings



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.

```

1 #Author: your.email@your.domain.com
2 #Keywords Summary :
3 #Feature: List of scenarios.
4 #Scenario: Business rule through list of steps with arguments.
5 #Given: Some precondition step
6 #When: Some key actions
7 #Then: To observe outcomes or validation
8 #And,But: To enumerate more Given,When,Then steps
9 #Scenario Outline: List of steps for data-driven as an Examples and <placeholder>
10 #Examples: Container for a table
11 #Background: List of steps run before each of the scenarios
12 """ (Doc Strings)
13 #I (Data Tables)
14 #@ (Tags/Labels): To group Scenarios
15 #<> (placeholder)
16 """
17 ## (Comments)
18 #Sample Feature Definition Template
19 @tag
20@Feature: Title of your feature
21 I want to use this template for my feature file
22
23 @tag1
24@ Scenario: Title of your scenario
25 Given I want to write a step with precondition
26 And some other precondition
27 When I complete action
28 And some other action
29 And yet another action
30 Then I validate the outcomes
31 And check more outcomes
```

```

One feature file can have multiple scenarios.

This is the way to write multiple scenarios:

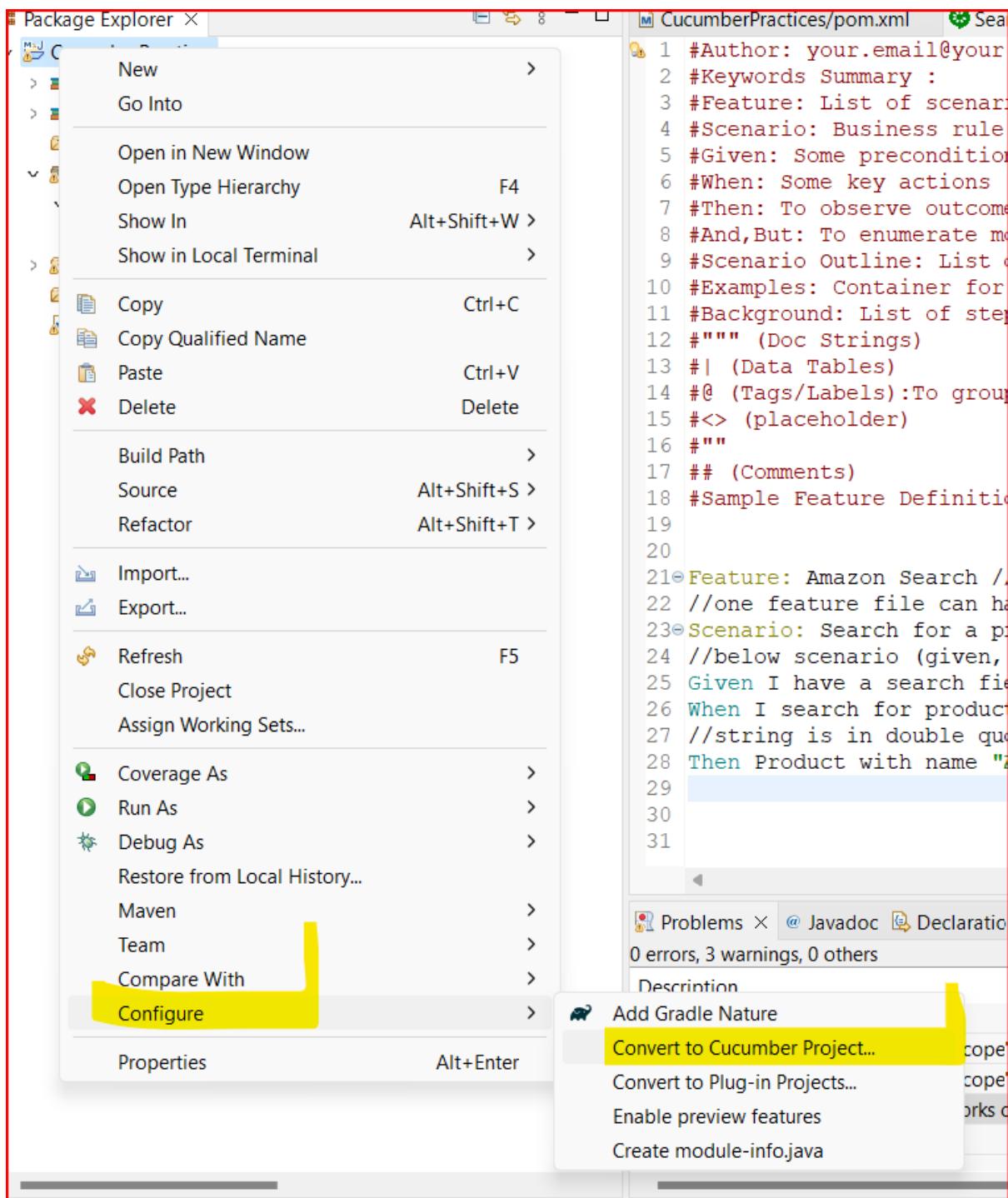
```
20
21 Feature: Amazon Search
22
23 Scenario: Search a Product
24 Given I have a search field on Amazon Page
25 When I search for a product with name "Apple MacBook Pro" and price 1000
26 Then Product with name "Apple MacBook Pro" should be displayed
27
28
29 Scenario: Add a Product
30 Given I have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

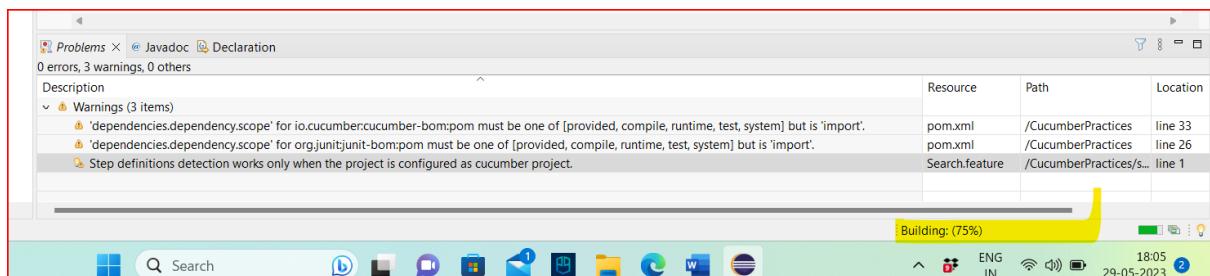
 Step definitions detection works only when the project is configured as cucumber project.

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[quotes]
27 Then Product with name "Apple" should be displayed
28
29
30

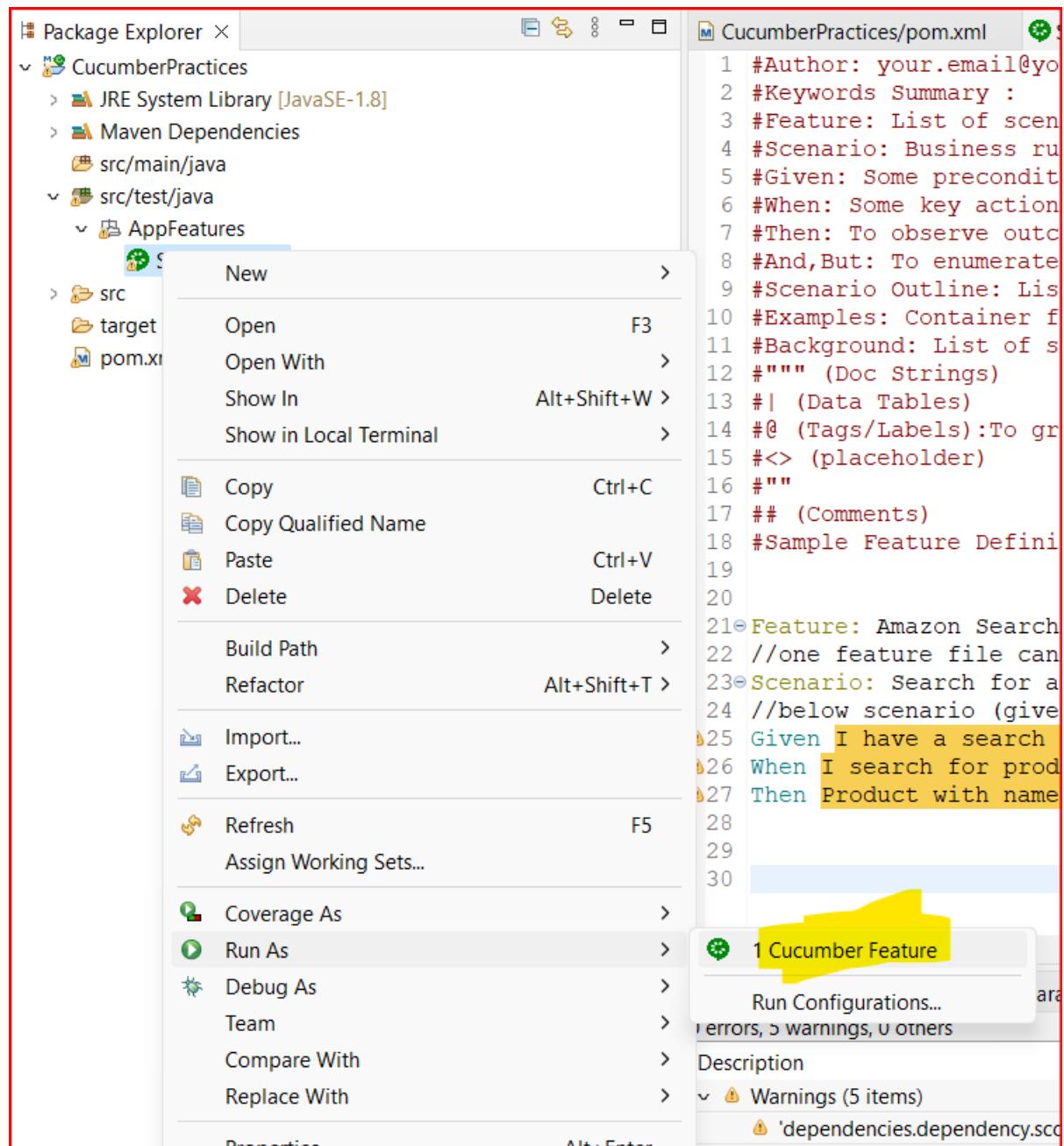
```

Why?

Glue code means there is no step definition mentioned.

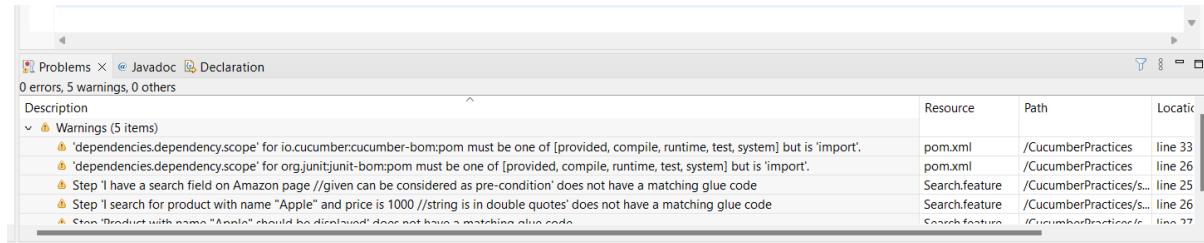
Now when we right click on feature file and run as cucumber project there will be no output:

Right click on feature and run as:



Console:

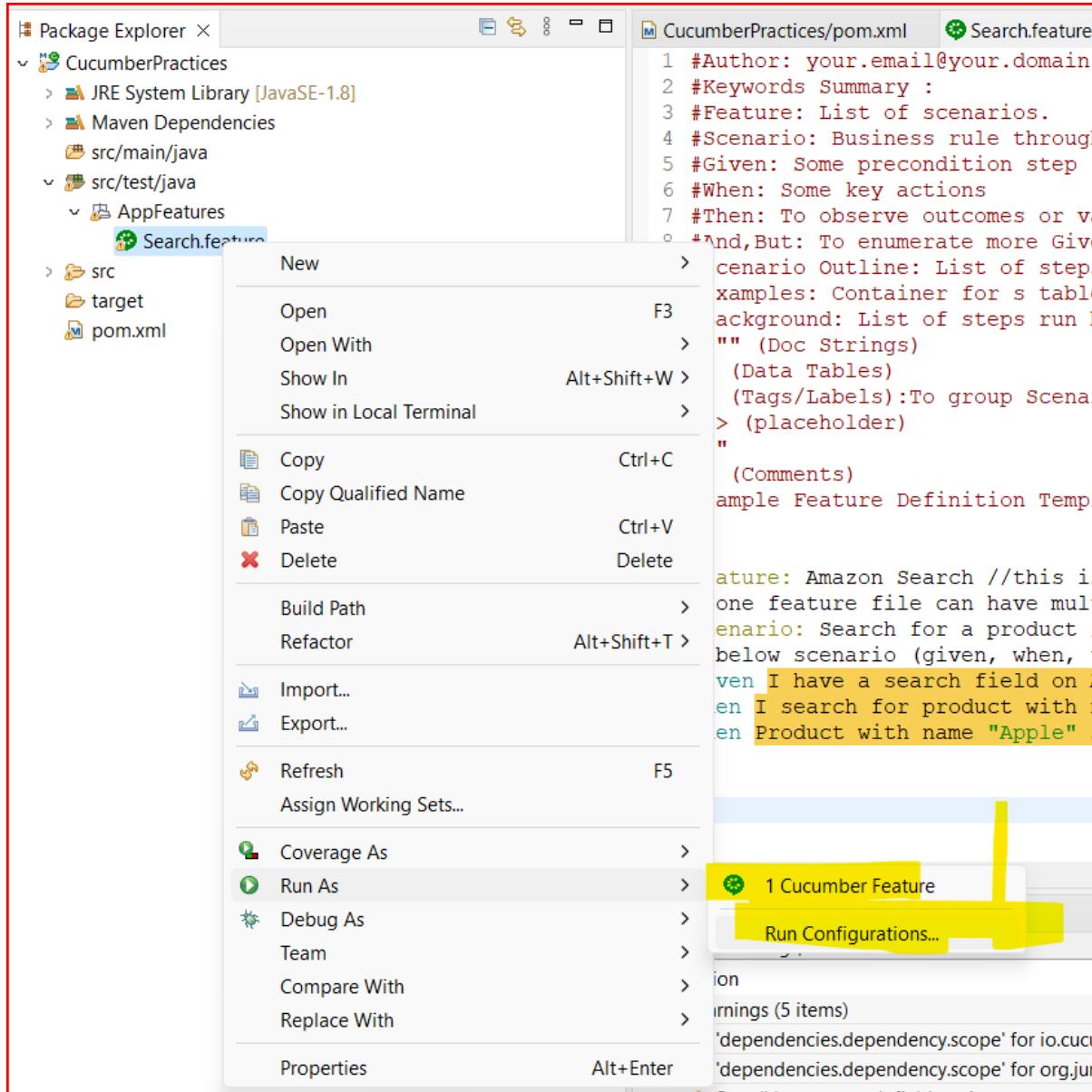
Nothing changed.



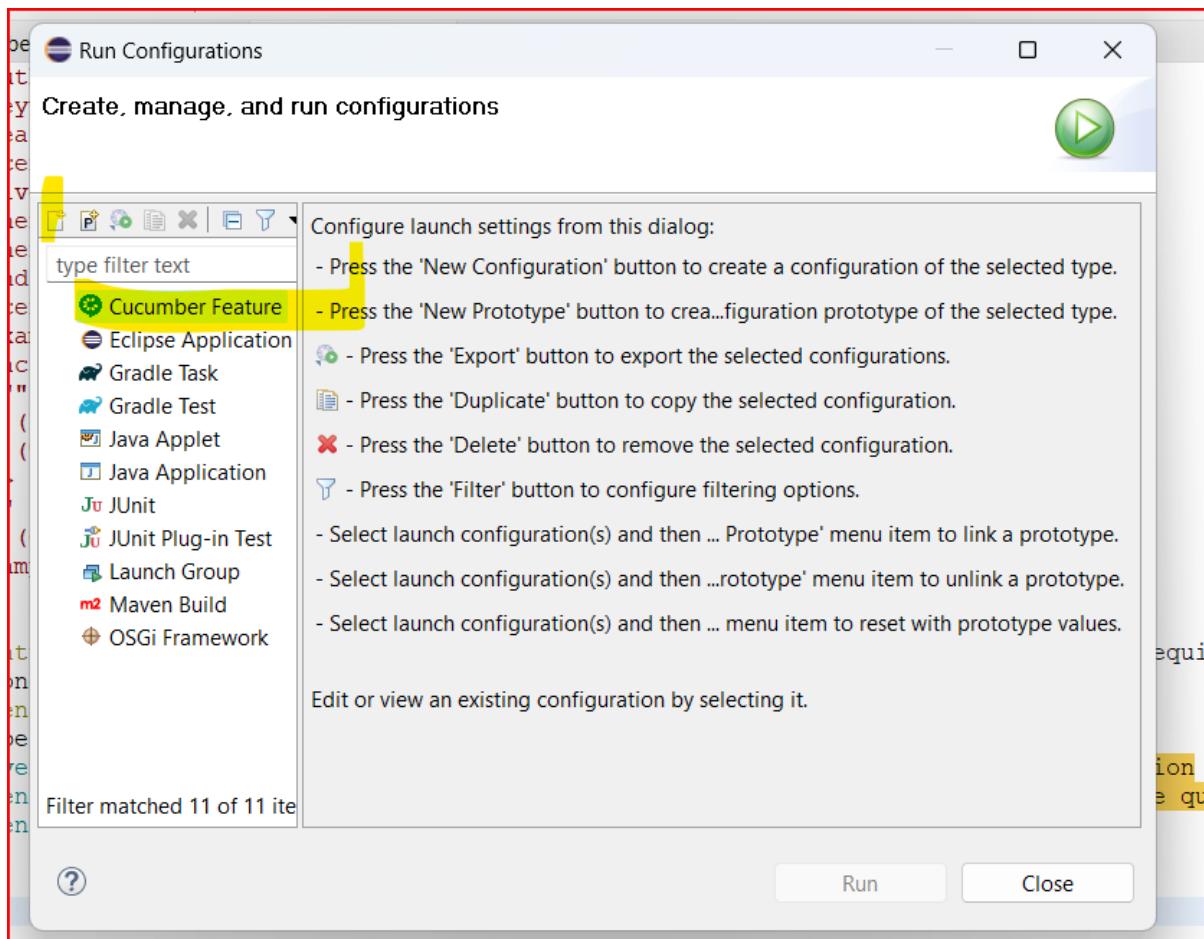
So, we need to configure the feature files in run configuration.

Right click on feature.

Run as -> run configuration.

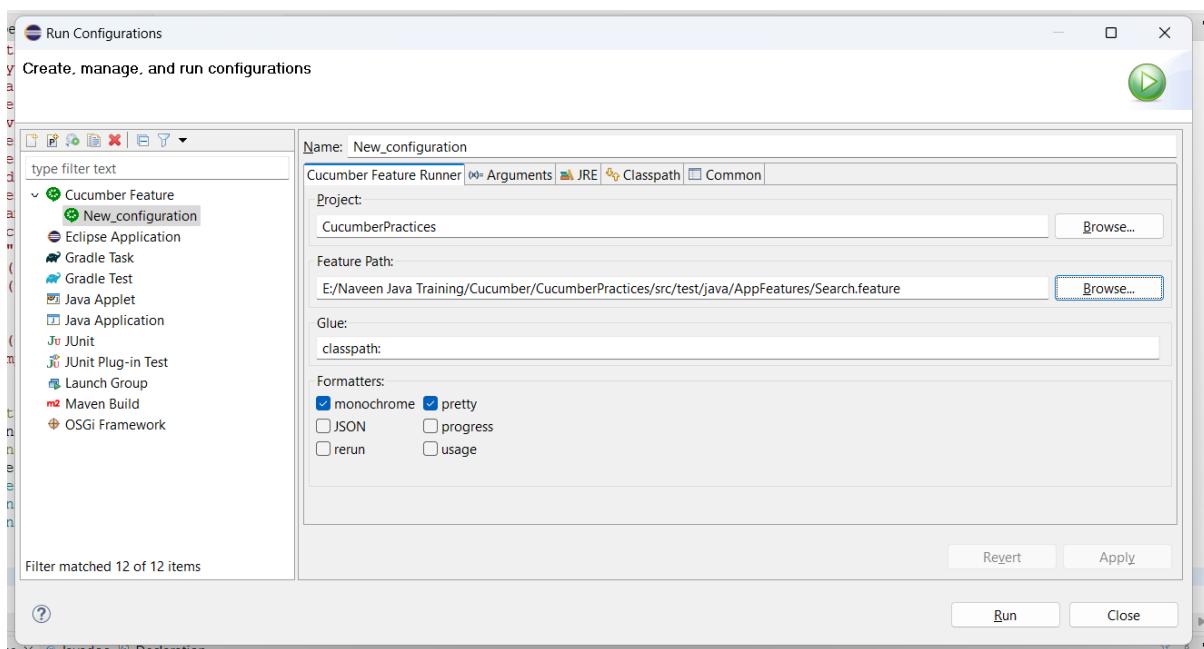


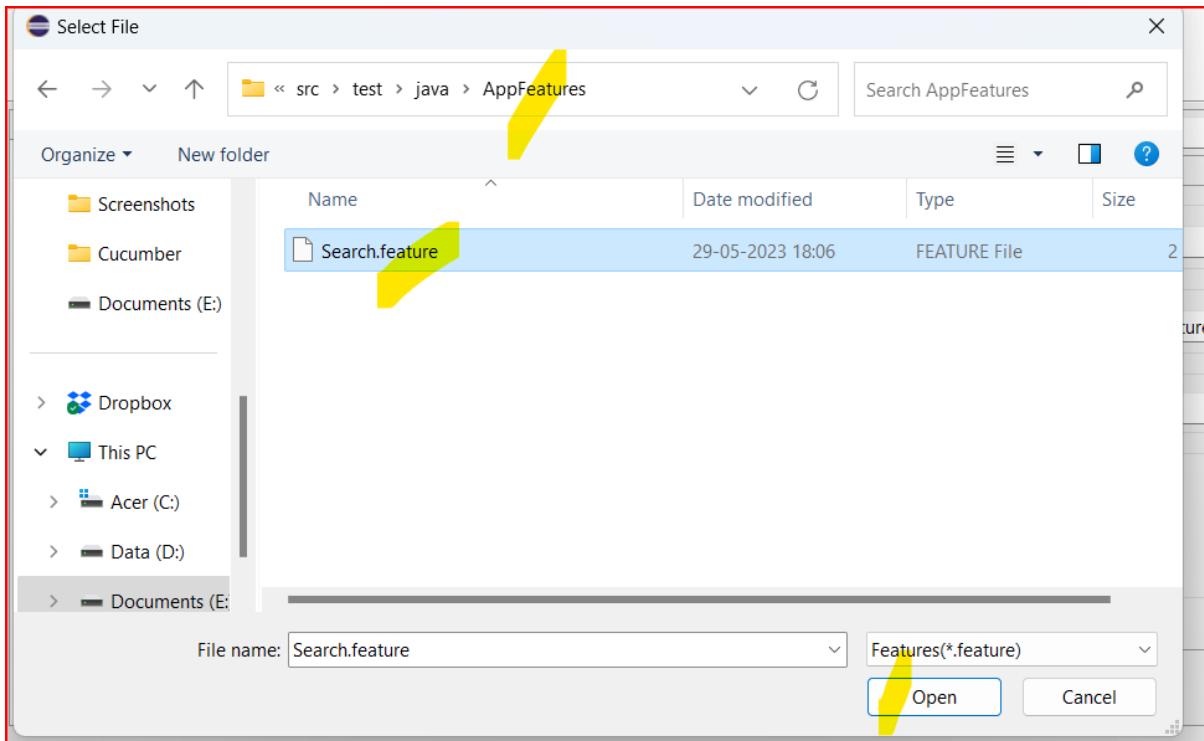
Select the "+" icon after clicking on cucumber.



Like this new window will come.

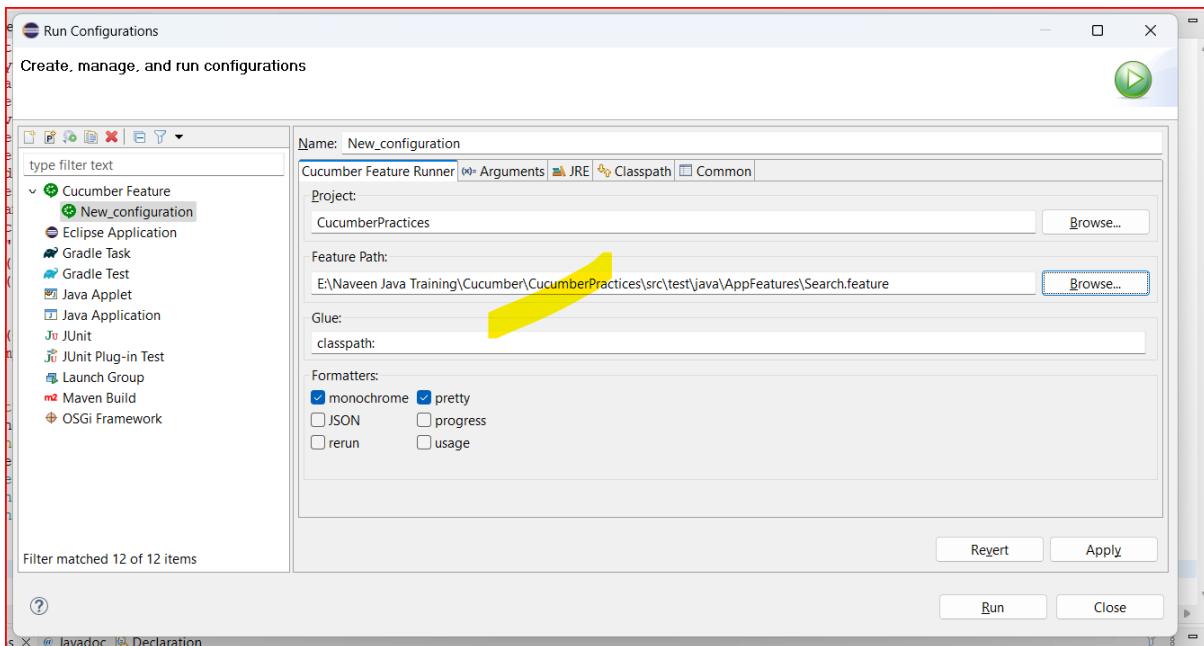
Click browse and go till the feature file path on your computer.





Click open.

Path has been updated.



Click apply and then run.

## For commenting in feature file use #-

```

3 #Feature: List of scenarios.
4 #Scenario: Business rule through list of steps with arguments.
5 #Given: Some precondition step
6 #When: Some key actions
7 #Then: To observe outcomes or validation
8 #And,But: To enumerate more Given,When,Then steps
9 #Scenario Outline: List of steps for data-driven as an Examples and <placeholder>
10 #Examples: Container for s table
11 #Background: List of steps run before each of the scenarios
12 """ (Doc Strings)
13 #| (Data Tables)
14 #@ (Tags/Labels):To group Scenarios
15 #<> (placeholder)
16 #
17 ## (Comments)
18 #Sample Feature Definition Template
19
20

```

### Feature file created:

```

Cucumber - CucumberPractices/src/test/java/AppFeatures/Search.feature - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Package Explorer X CucumberPractices pom.xml Search.feature X
CucumberPractices
 JRE System Library [JavaSE-1.8]
 Maven Dependencies
 src/main/java
 src/test/java
 AppFeatures
 Search.feature
 src
 target
 pom.xml
3 #Feature: List of scenarios.
4 #Scenario: Business rule through list of steps with arguments.
5 #Given: Some precondition step
6 #When: Some key actions
7 #Then: To observe outcomes or validation
8 #And,But: To enumerate more Given,When,Then steps
9 #Scenario Outline: List of steps for data-driven as an Examples and <placeholder>
10 #Examples: Container for s table
11 #Background: List of steps run before each of the scenarios
12 """ (Doc Strings)
13 #| (Data Tables)
14 #@ (Tags/Labels):To group Scenarios
15 #<> (placeholder)
16 #
17 ## (Comments)
18 #Sample Feature Definition Template
19
20
21 Feature: Amazon Search //this is the feature for which we want to write code and requirements
22 #one feature file can have multiple scenarios
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 When I search for product with name "Apple" and price is 1000 #string is in double quotes
27 Then Product with name "Apple" should be displayed
28
29

```

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
```

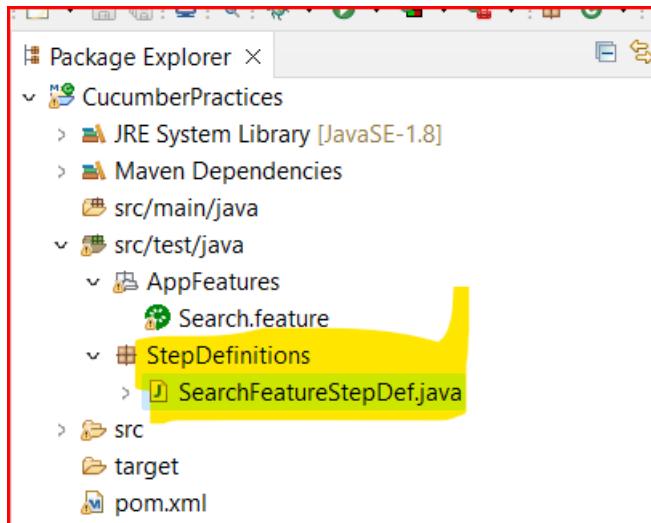
You can implement missing steps with the snippets below:

```
@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_condition() {
 // 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(String 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:

```

1 package StepDefinitions;
2
3 public class SearchFeatureStepDef {
4
5 @Given("I have a search field on Amazon page #given can be considered as pre-condition")
6 public void i_have_a_search_field_on_amazon_page_given_can_be_considered_as_pre_condition() {
7 // Write code here that turns the phrase above into concrete actions
8 throw new io.cucumber.java.PendingException();
9 }
10
11 @When("I search for product with name {string} and price is {int} #string is in double quotes")
12 public void i_search_for_product_with_name_and_price_is_string_is_in_double_quotes(String string, Integer int1) {
13 // Write code here that turns the phrase above into concrete actions
14 throw new io.cucumber.java.PendingException();
15 }
16
17 @Then("Product with name {string} should be displayed")
18 public void product_with_name_should_be_displayed(String string) {
19 // Write code here that turns the phrase above into concrete actions
20 throw new io.cucumber.java.PendingException();
21 }
22
23
24 }
25

```

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

```

2
3@import io.cucumber.java.en.Given;
4 import io.cucumber.java.en.Then;
5 import io.cucumber.java.en.When;
6
7 public class SearchFeatureStepDef {
8
9 @Given("I have a search field on Amazon page")
10 public void i_have_a_search_field_on_amazon_page() {
11 // Write code here that turns the phrase above into concrete actions
12 throw new io.cucumber.java.PendingException();
13 }
14
15 @When("I search for product with name {string} and price is {int}")
16 public void i_search_for_product_with_name_and_price_is_int(String string, Integer int1) {
17 // Write code here that turns the phrase above into concrete actions
18 throw new io.cucumber.java.PendingException();
19 }
20
21 @Then("Product with name {string} should be displayed")
22 public void product_with_name_should_be_displayed(String string) {
23 // Write code here that turns the phrase above into concrete actions
24 throw new io.cucumber.java.PendingException();
25 }

```

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

Feature file:

```

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

```

Step definition file:

```

9@ @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:

```

9@ @Given("I have a search field on Amazon page")
0 public void i_have_a_search_field_on_amazon_page() {
1 // Write code here that turns the phrase above into concrete actions
2 throw new io.cucumber.java.PendingException();
3 }

```

This exception is thrown because there is no java code:

```

9@ @Given("I have a search field on Amazon page")
10 public void i_have_a_search_field_on_amazon_page() {
11 // Write code here that turns the phrase above into concrete actions
12 throw new io.cucumber.java.PendingException();
13 }

```

If you run the feature file again:

```

Nov 22, 2020 9:56:03 AM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Search a Product
 Given I have a search field on Amazon Page
 io.cucumber.java.PendingException: TODO: implement me
 at stepdefinitions.SearchSteps.i_have_a_search_field_on_amazon_page(SearchSteps.java:13)
 at *I have a search field on Amazon Page(file:///Users/naveenautomationlabs/Documents/workspace)

 When I search for a product with name "Apple MacBook Pro" and price 1000 # stepdefinitions.SearchSteps
 Then Product with name "Apple MacBook Pro" should be displayed # stepdefinitions.SearchSteps

Pending scenarios:
file:///Users/naveenautomationlabs/Documents/workspace/CucumberPractices/src/test/java/AppFeatures/Search

1 Scenarios (1 pending)
3 Steps (2 skipped, 1 pending)
0m0.301s

io.cucumber.java.PendingException: TODO: implement me
 at stepdefinitions.SearchSteps.i_have_a_search_field_on_amazon_page(SearchSteps.java:13)
 at *I have a search field on Amazon Page(file:///Users/naveenautomationlabs/Documents/workspace)

```

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-

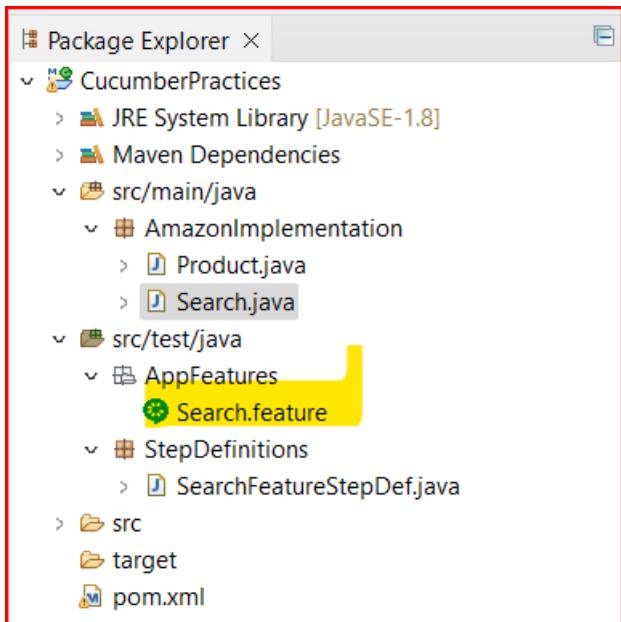
We write features.

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.

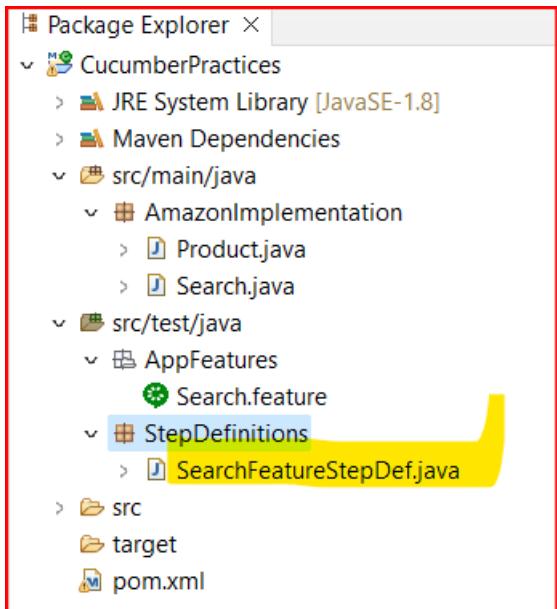
Feature file:



```
#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:



```

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);

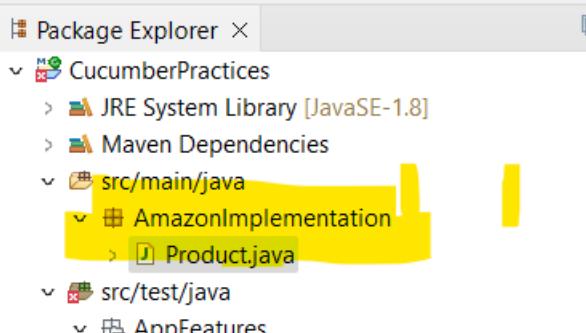
 product =new Product(productName, price);
 }

 @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:



```
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;
 }

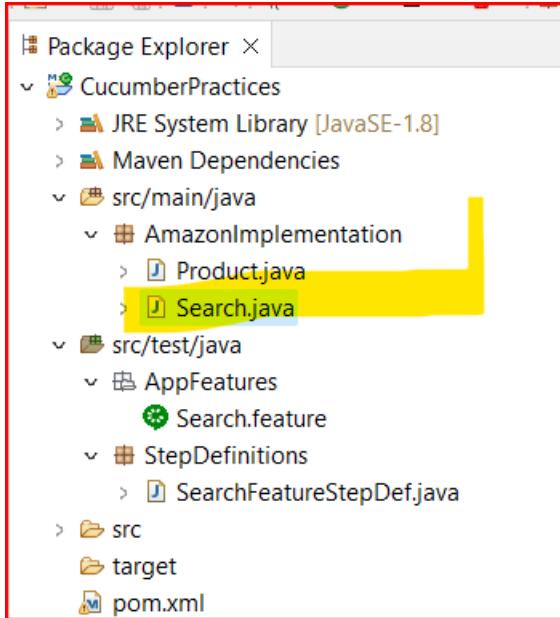
 public void setProductName(String productName) {
 this.productName = productName;
 }

 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 ArrayList 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.

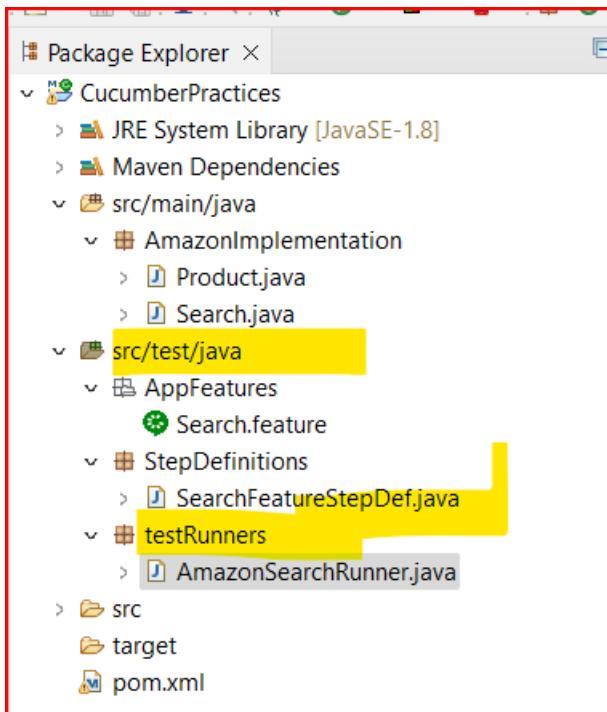
Note-

"Then" is used for assertions generally.

Runner.java:

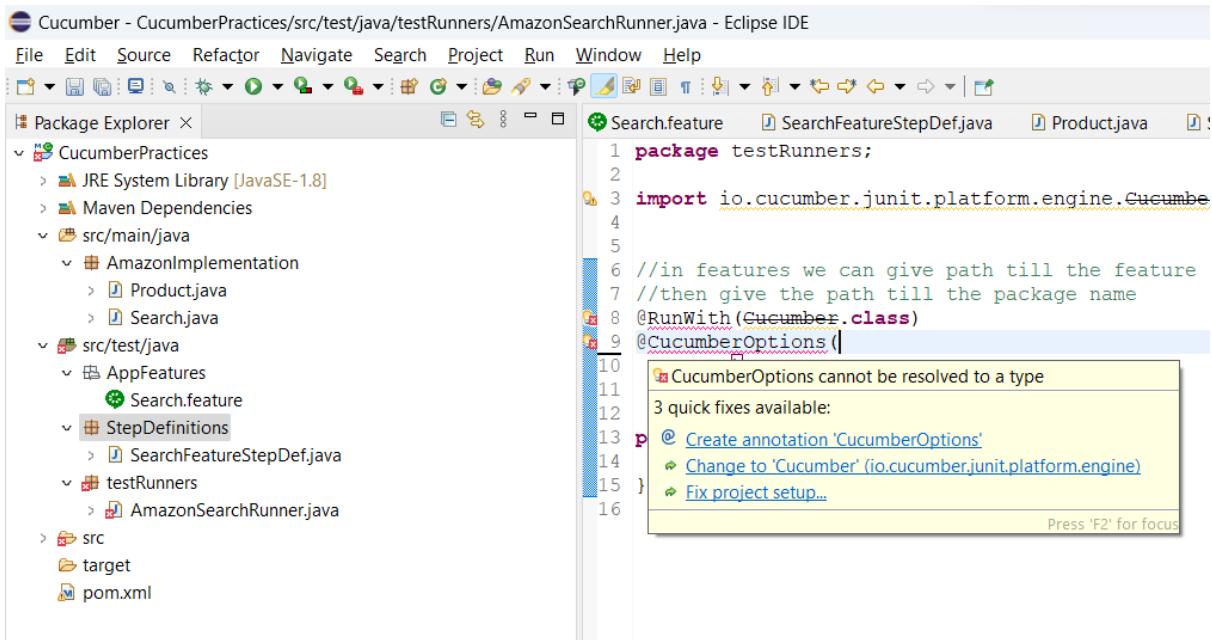
Every time we cannot run feature by feature.

Runner class location:



You will get this error in runner class:

the type cucumber is deprecated issue.



```

1 package testRunners;
2
3 import io.cucumber.junit.platform.engine.Cucumber;
4
5 //in features we can give path till the f
6 //then give the path till the package nam
7 @RunWith(Cucumber.class)
8 @CucumberOptions(
9
10
11

```

The type Cucumber is deprecated  
2 quick fixes available:  
Add @SuppressWarnings 'deprecation' to 'AmazonSearchRunner'  
Configure problem severity

Search for this in google:

https://www.google.com/search?q=the+type+cucumber+is+deprecated+issue&oq=the+&aqs=edge.0.69j59l2j69i57j69i

the type cucumber is deprecated issue

All Videos Images Books News More Tools

About 273,000 results (0.52 seconds)

**Stack Overflow**  
<https://stackoverflow.com/questions/57103333/java-cucumber-has-deprecated-giventhenwhen-in-4.7>

Jul 22, 2019 · 3 answers

**The tag itself is not deprecated.** The import is. You'll need to update your import statements to the new import.

cucumber - How to resolve deprecated @CucumberOptions? Dec 3, 2015  
no tests executed after migrating to cucumber 7.x Mar 10, 2022  
**The type CucumberOptions is deprecated** how can i resolve ... Sep 14, 2019  
Getting Error for @CucumberOptions and seems cucumber ... May 20, 2022  
More results from stackoverflow.com

You've visited this page 2 times. Last visit: 5/31/23

**GitHub**  
<https://github.com/blob/main/release-notes/v7...>

**cucumber-jvm/v7.0.0.md at main**  
Finally, the @Cucumber annotation has been deprecated in favour of @Suite . The @Cucumber annotated class was originally intended as a workaround for both JUnit ...  
You've visited this page 3 times. Last visit: 5/31/23

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

[cucumber-jvm/v7.0.0.md at main · cucumber/cucumber-jvm · GitHub](https://github.com/cucumber/cucumber-jvm/blob/main/release-notes/v7.0.0.md)

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

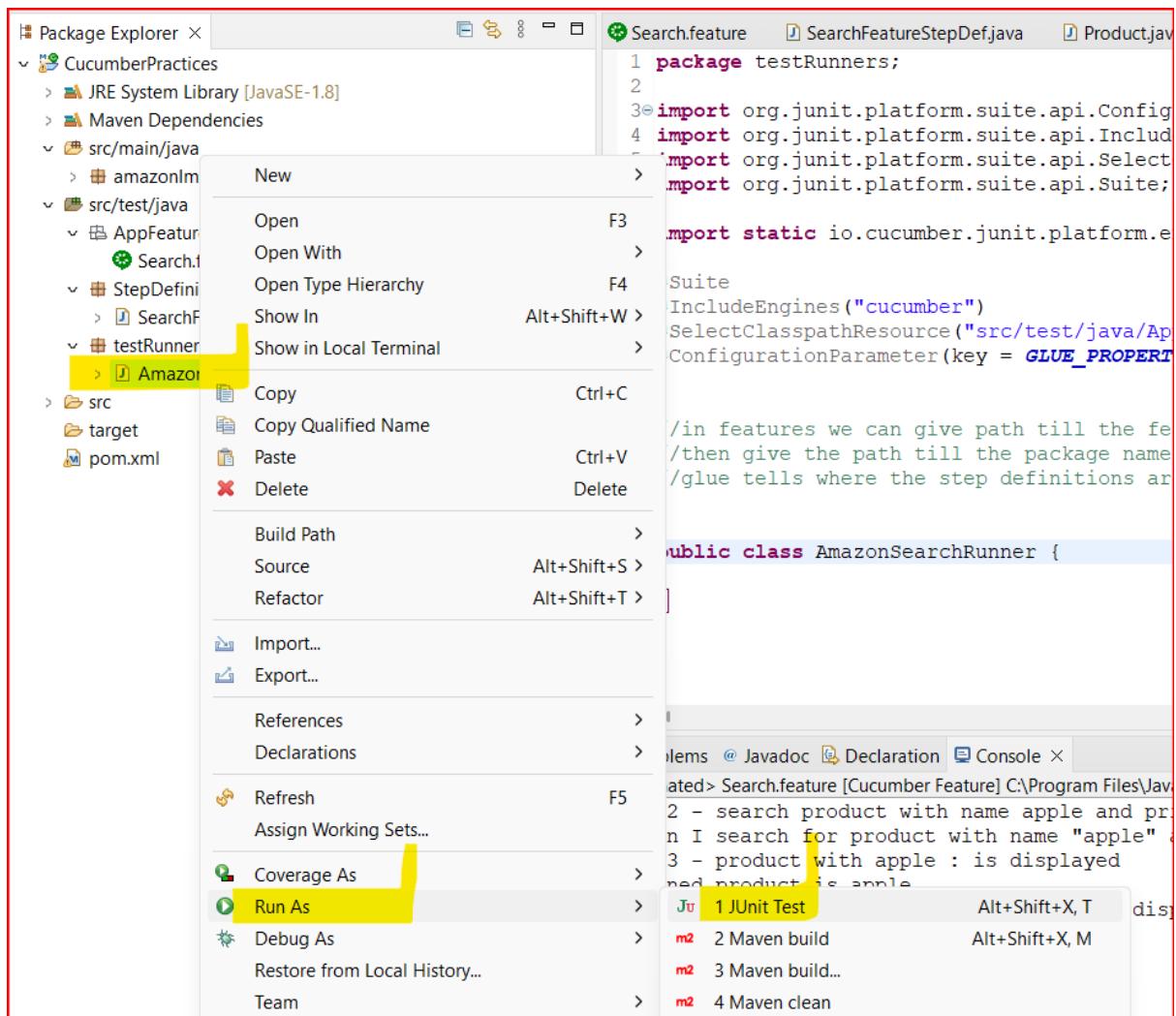
import static io.cucumber.junit.platform.engine.Constants.GLUE_PROPERTY_NAME;

@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
<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
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>(StackTrace  
PruningEngineExecutionListener.java:40) at  
org.junit.platform.launcher.core.EngineExecutionOrchestrator.selectExecutionListener(Engine  
ExecutionOrchestrator.java:184) at  
org.junit.platform.launcher.core.EngineExecutionOrchestrator.execute(EngineExecutionOrche  
strator.java:162) at  
org.junit.platform.launcher.core.EngineExecutionOrchestrator.execute(EngineExecutionOrche  
strator.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 `` section:

```

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

This is because see the seeding in mvnrepo.

This version has the highest seeds.

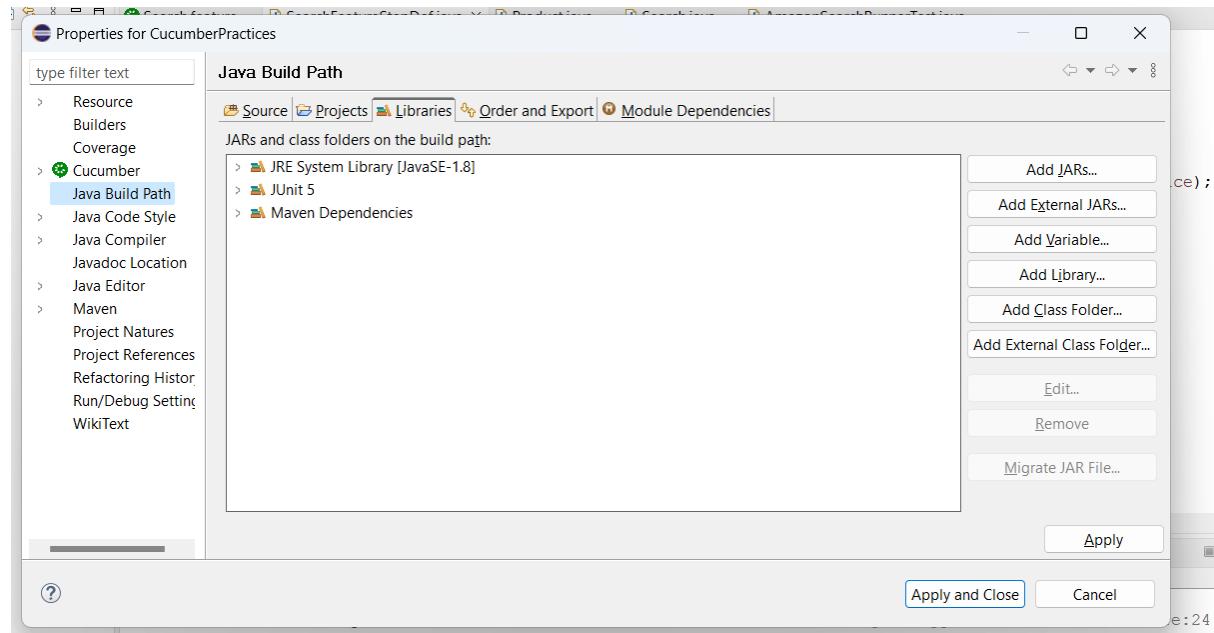
	Version	Vulnerabilities	Repository	Usages	Date
3.1.x	3.1.0		Central	1	May 07, 2023
	3.0.0		Central	2	Mar 14, 2023
	3.0.0-M9		Central	1	Feb 14, 2023
	3.0.0-M8		Central	3	Jan 12, 2023
	3.0.0-M7		Central	3	Jun 07, 2022
	3.0.0-M6		Central	2	Apr 04, 2022
3.0.x	3.0.0-M5		Central	14	Jun 17, 2020

Right click on runner class and run as junit test:

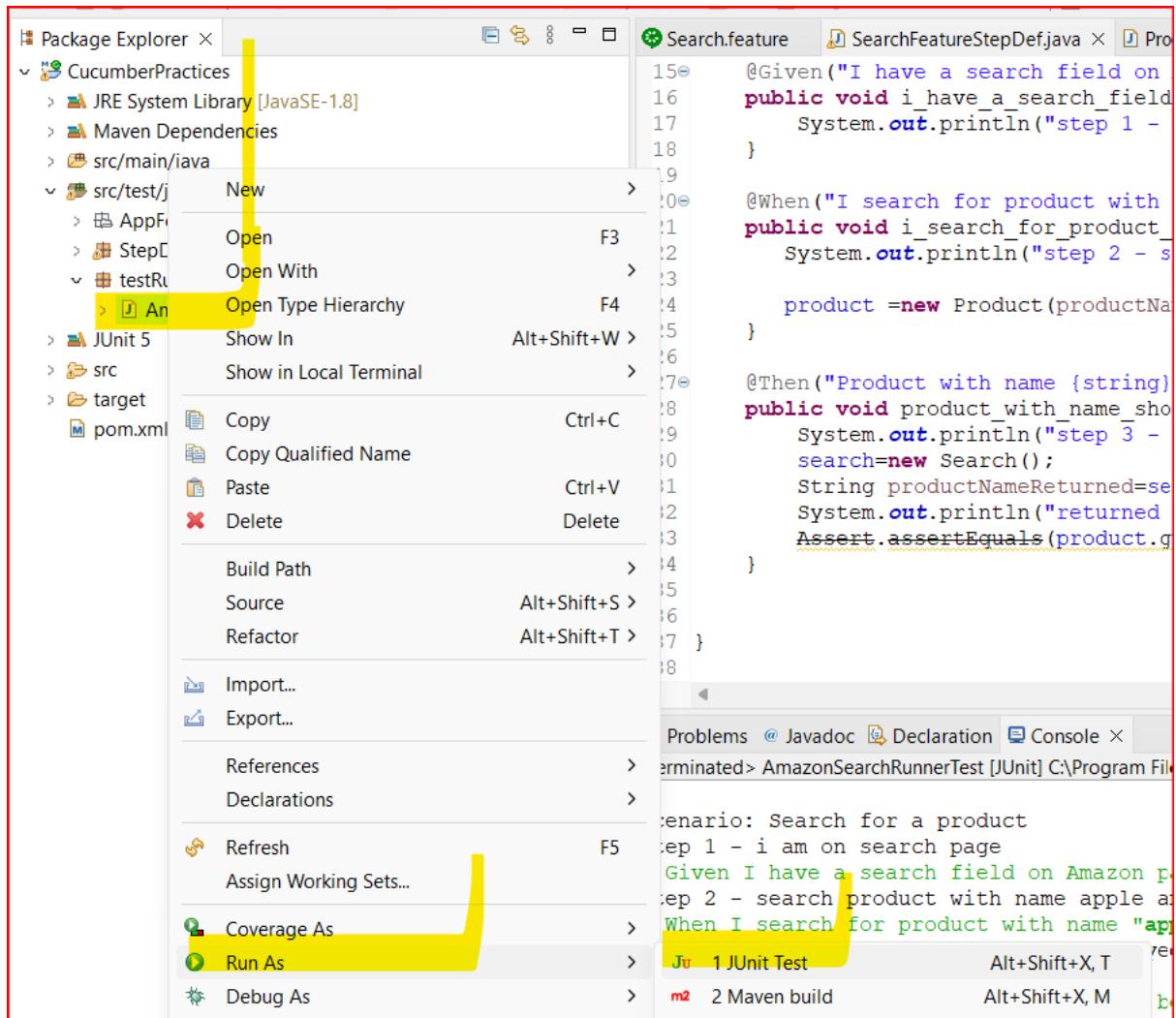
First time we get error as:

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:

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

The above can be ignored.

## Console output:



```

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 # StepDefinitions.SearchFeatureStepDef.i_search_for_product_with_name_and_price()
 When I search for product with name "apple" and price is 1000 # StepDefinitions.SearchFeatureStepDef.i_search_for_product_with_name_and_price()
step 3 - product with apple : is displayed
 Then Product with name "apple" should be displayed # StepDefinitions.SearchFeatureStepDef.product_with_name_should_be_displayed()

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

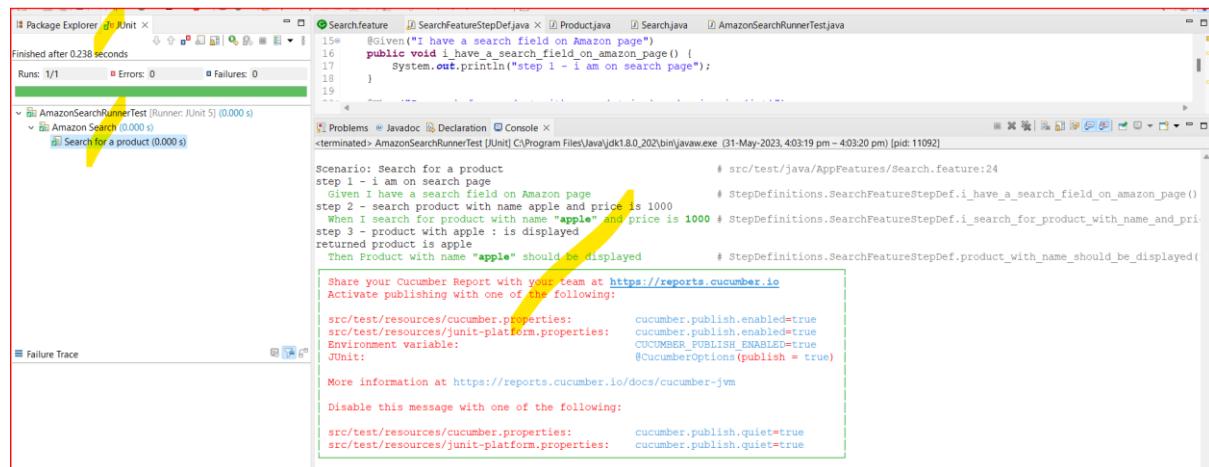
Disable this message with one of the following:

src/test/resources/cucumber.properties: cucumber.publish.quiet=true
src/test/resources/junit-platform.properties: cucumber.publish.quiet=true

```

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

Green means passed.



```

Package Explorer JUnit X
Finished after 0.238 seconds
Runs: 1/1 Errors: 0 Failures: 0

AmazonSearchRunnerTest [Runner: JUnit 5] (0.000 s)
 Amazon Search (0.000 s)
 Search for a product (0.000 s)

Search.feature JSearchFeatureStepDefJava X Product.java Searchjava AmazonSearchRunnerTest.java
15# @Given("I have a search field on Amazon page")
16 public void i_have_a_search_field_on_amazon_page() {
17 System.out.println("step 1 - i am on search page");
18 }
19

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 # StepDefinitions.SearchFeatureStepDef.i_search_for_product_with_name_and_price()
 When I search for product with name "apple" and price is 1000 # StepDefinitions.SearchFeatureStepDef.i_search_for_product_with_name_and_price()
step 3 - product with apple : is displayed
 Then Product with name "apple" should be displayed # StepDefinitions.SearchFeatureStepDef.product_with_name_should_be_displayed()

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:

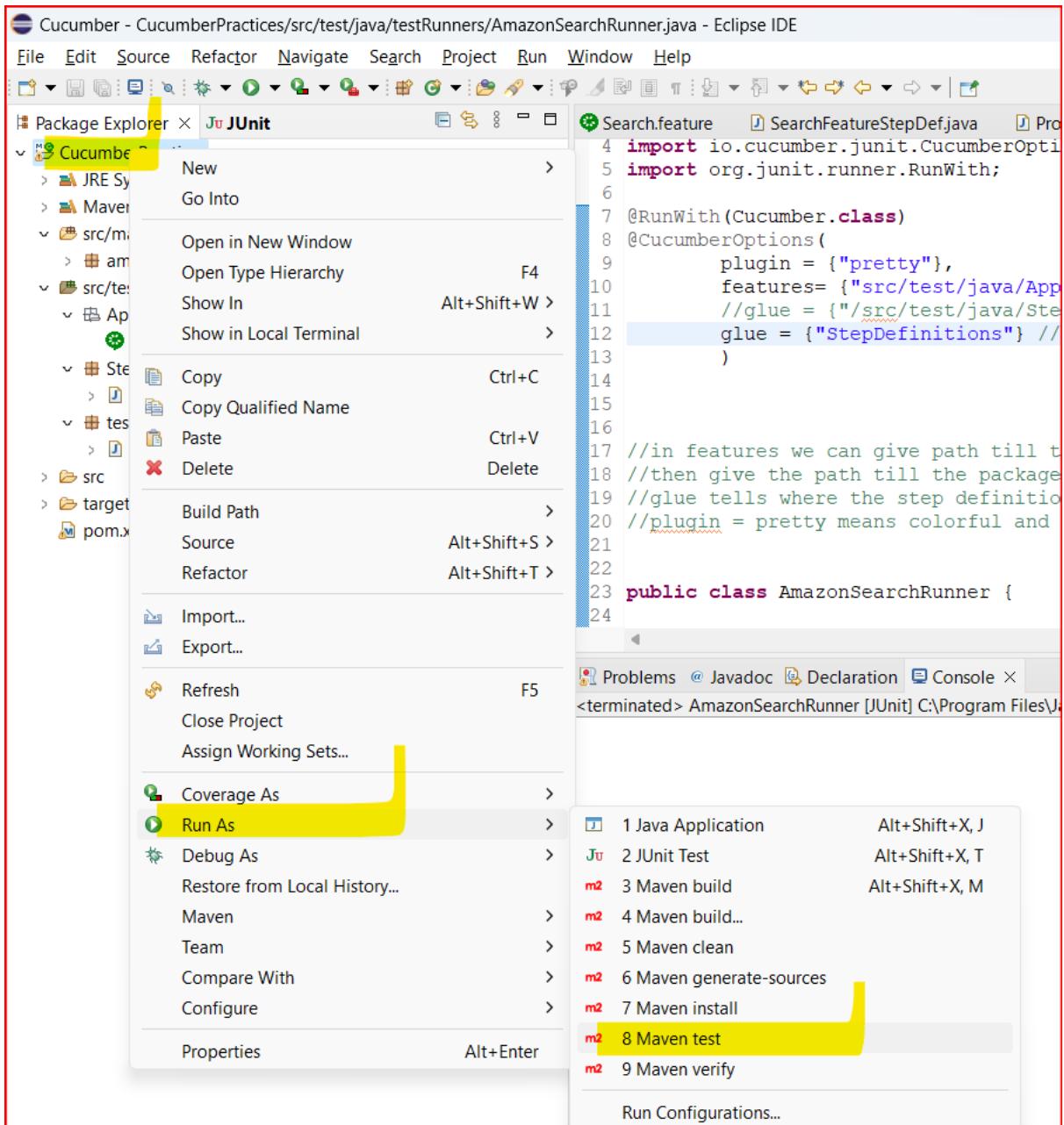
src/test/resources/cucumber.properties: cucumber.publish.quiet=true
src/test/resources/junit-platform.properties: cucumber.publish.quiet=true

```

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.

The screenshot shows the Eclipse IDE's 'Console' tab with a red border. It displays the output of a Maven build. The output includes several INFO messages from various Maven plugins: maven-compiler-plugin, maven-resources-plugin, maven-compiler-plugin again, and maven-surefire-plugin. The 'BUILD SUCCESS' message is highlighted with a yellow box. At the bottom of the console, it shows the total time taken and the finished date and time.

```

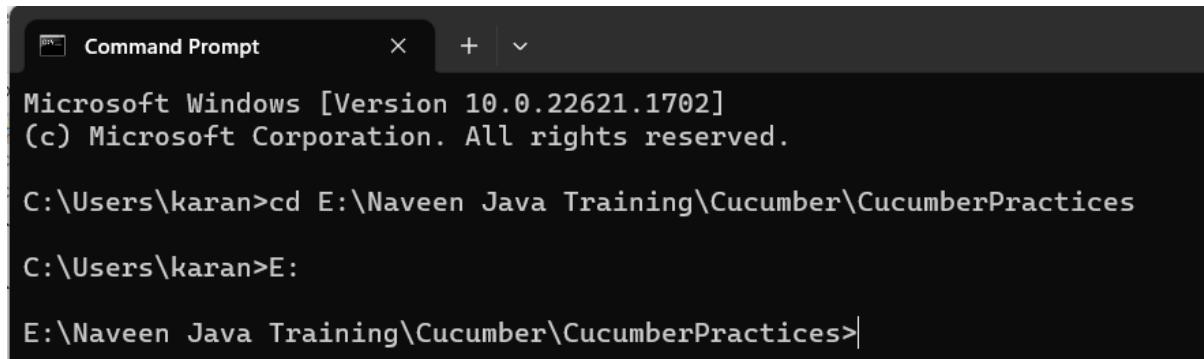
[maven-compiler-plugin:3.11.0:compile (default-compile) @ CucumberPractices]
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ CucumberPractices ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory E:\Naveen Java Training\Cucumber\CucumberPractices\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.11.0:testCompile (default-testCompile) @ CucumberPractices ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:3.0.0-M5:test (default-test) @ CucumberPractices ---
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 1.044 s
[INFO] Finished at: 2023-05-31T13:35:32+02:00
[INFO]

```

To run from command line using maven-

Open cmd.

Be in path of project.



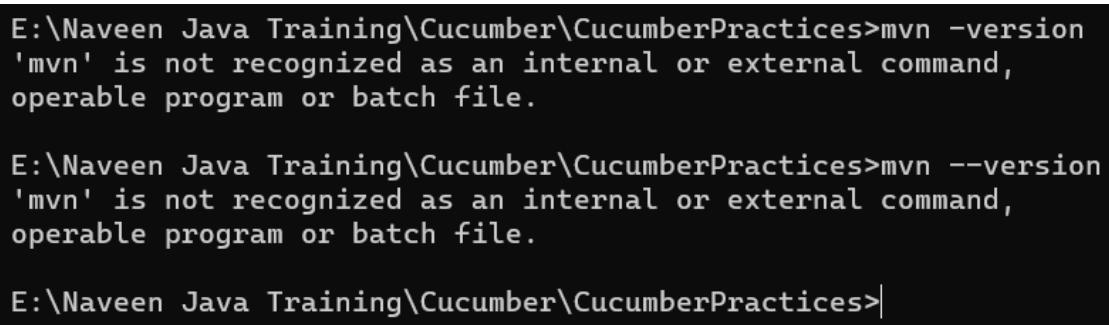
```
Command Prompt
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\karan>cd E:\Naveen Java Training\Cucumber\CucumberPractices

C:\Users\karan>E:

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

Check if maven is present.



```
E:\Naveen Java Training\Cucumber\CucumberPractices>mvn -version
'mvn' is not recognized as an internal or external command,
operable program or batch file.

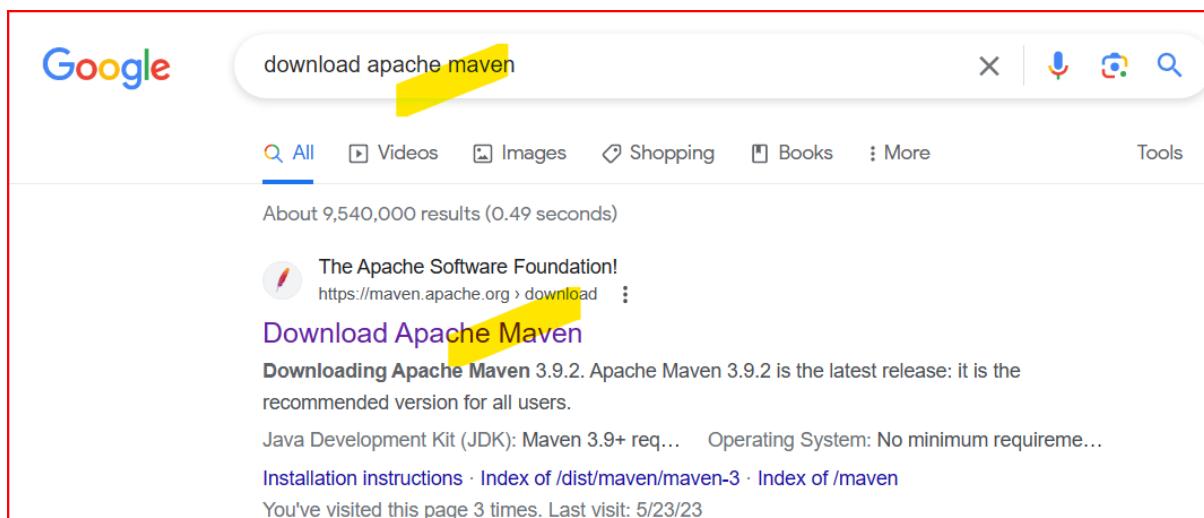
E:\Naveen Java Training\Cucumber\CucumberPractices>mvn --version
'mvn' is not recognized as an internal or external command,
operable program or batch file.

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

It is not present.

How to set it up.

Goto google and search for download maven.

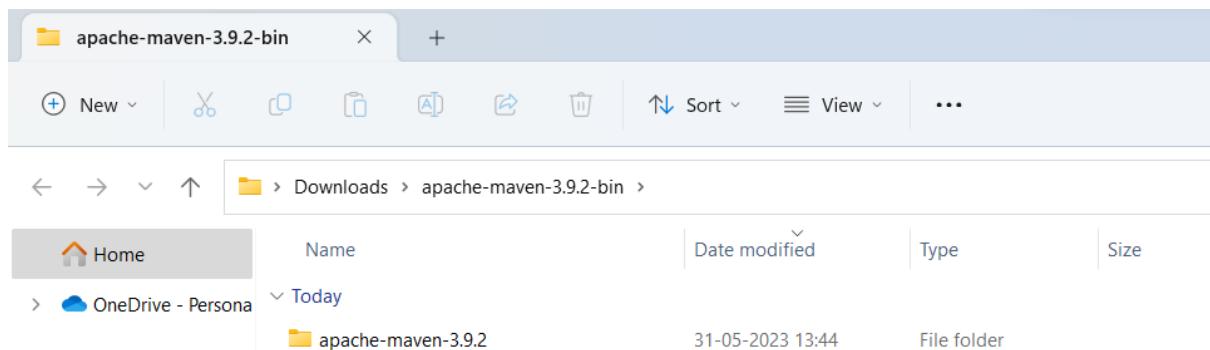


Once inside the link click on the highlighted zip file.

The screenshot shows the Maven download page at <https://maven.apache.org/download.cgi>. A red box highlights the left sidebar menu. The main content area is titled 'Files' and contains a table of Maven distributions. The table has columns for 'Link' and 'Checksums'. The 'Link' column lists the download URLs, and the 'Checksums' column lists the SHA-1 checksums for each file.

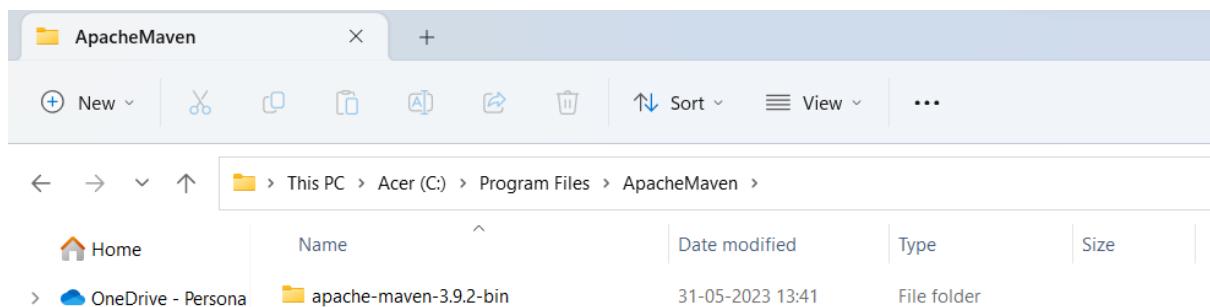
	Link	Checksums
Binary tar.gz archive	<a href="#">apache-maven-3.9.2-bin.tar.gz</a>	<a href="#">apache-maven-3.9.2-bin.tar.gz</a>
Binary zip archive	<a href="#">apache-maven-3.9.2-bin.zip</a>	<a href="#">apache-maven-3.9.2-bin.zip</a>
Source tar.gz archive	<a href="#">apache-maven-3.9.2-src.tar.gz</a>	<a href="#">apache-maven-3.9.2-src.tar.gz</a>
Source zip archive	<a href="#">apache-maven-3.9.2-src.zip</a>	<a href="#">apache-maven-3.9.2-src.zip</a>

Extract it.



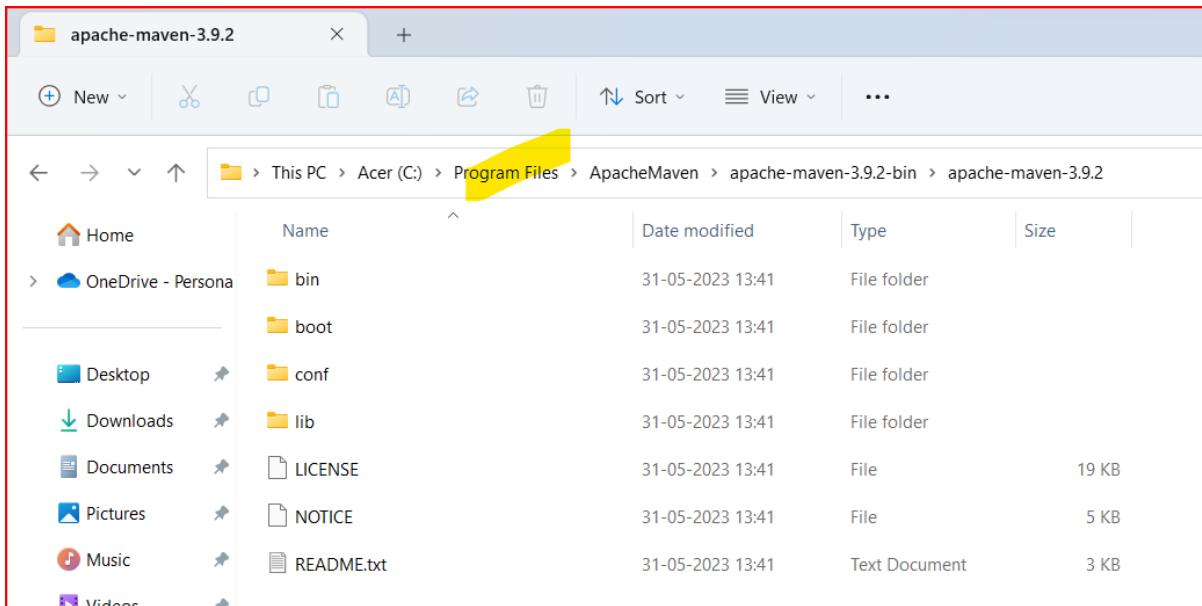
Copy this folder.

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 “MAVEN\_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:

```
Command Prompt
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\karan>mvn -version
Apache Maven 3.9.2 (c9616018c7a021c1c39be70fb2843d6f5f9b8a1c)
Maven home: C:\Program Files\ApacheMaven\apache-maven-3.9.2-bin\apache-maven-3.9.2
Java version: 1.8.0_202, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk1.8.0_202\jre
Default locale: en_IN, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

C:\Users\karan>mvn --version
Apache Maven 3.9.2 (c9616018c7a021c1c39be70fb2843d6f5f9b8a1c)
Maven home: C:\Program Files\ApacheMaven\apache-maven-3.9.2-bin\apache-maven-3.9.2
Java version: 1.8.0_202, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk1.8.0_202\jre
Default locale: en_IN, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

C:\Users\karan>
```

Maven is installed.

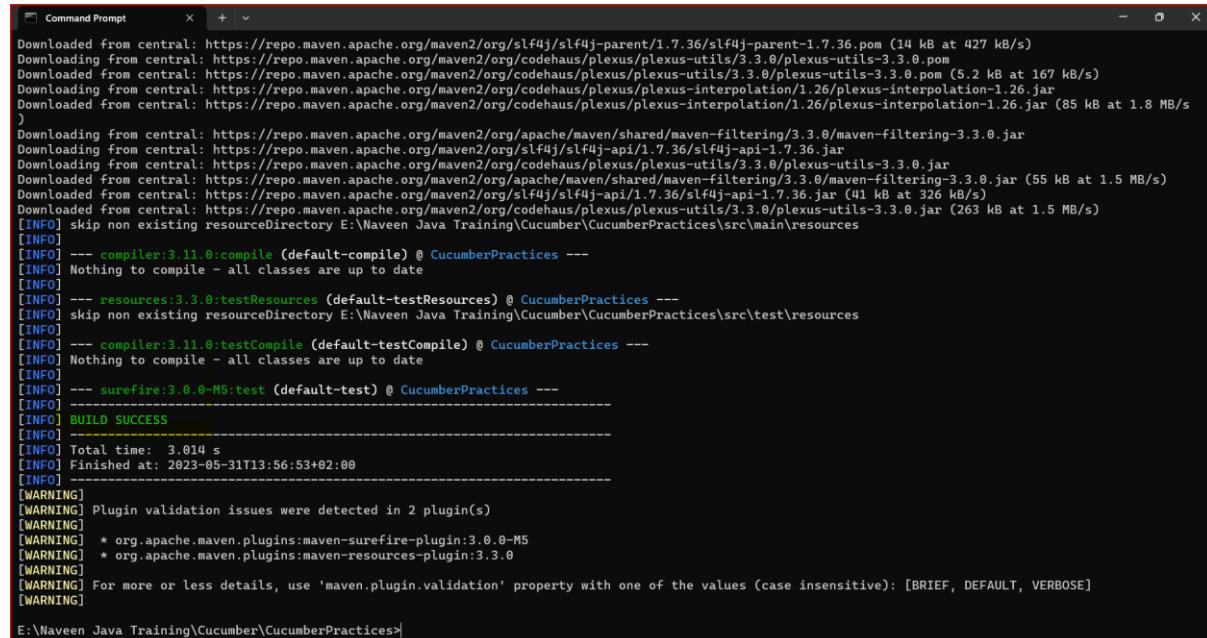
Now go to the path of the project.

Type this command.

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

Hit enter.

You should get build success.



```
Downloaded from central: https://repo.maven.apache.org/maven2/org/slf4j/slf4j-parent/1.7.36/slf4j-parent-1.7.36.pom (14 kB at 427 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.3.0/plexus-utils-3.3.0.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.3.0/plexus-utils-3.3.0.pom (5.2 kB at 167 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-interpolation/1.26/plexus-interpolation-1.26.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-interpolation/1.26/plexus-interpolation-1.26.jar (85 kB at 1.8 MB/s)
)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-filtering/3.3.0/maven-filtering-3.3.0.jar
Downloading from central: https://repo.maven.apache.org/maven2/org/slf4j/slf4j-api/1.7.36/slf4j-api-1.7.36.jar
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.3.0/plexus-utils-3.3.0.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-filtering/3.3.0/maven-filtering-3.3.0.jar (55 kB at 1.5 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/slf4j/slf4j-api/1.7.36/slf4j-api-1.7.36.jar (41 kB at 326 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.3.0/plexus-utils-3.3.0.jar (263 kB at 1.5 MB/s)
[INFO] skip non existing resourceDirectory E:\Naveen Java Training\Cucumber\CucumberPractices\src\main\resources
[INFO]
[INFO] --- compiler:3.11.0:compile (default-compile) @ CucumberPractices ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- resources:3.3.0:testResources (default-testResources) @ CucumberPractices ---
[INFO] skip non existing resourceDirectory E:\Naveen Java Training\Cucumber\CucumberPractices\src\test\resources
[INFO]
[INFO] --- compiler:3.11.0:testCompile (default-testCompile) @ CucumberPractices ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- surefire:3.0.0-M5:test (default-test) @ CucumberPractices ---
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.014 s
[INFO] Finished at: 2023-05-31T13:56:53+02:00
[INFO] -----
[WARNING]
[WARNING] Plugin validation issues were detected in 2 plugin(s)
[WARNING]
[WARNING] * org.apache.maven.plugins:maven-surefire-plugin:3.0.0-M5
[WARNING] * org.apache.maven.plugins:maven-resources-plugin:3.3.0
[WARNING]
[WARNING] For more or less details, use 'maven.plugin.validation' property with one of the values (case insensitive): [BRIEF, DEFAULT, VERBOSE]
[WARNING]
```

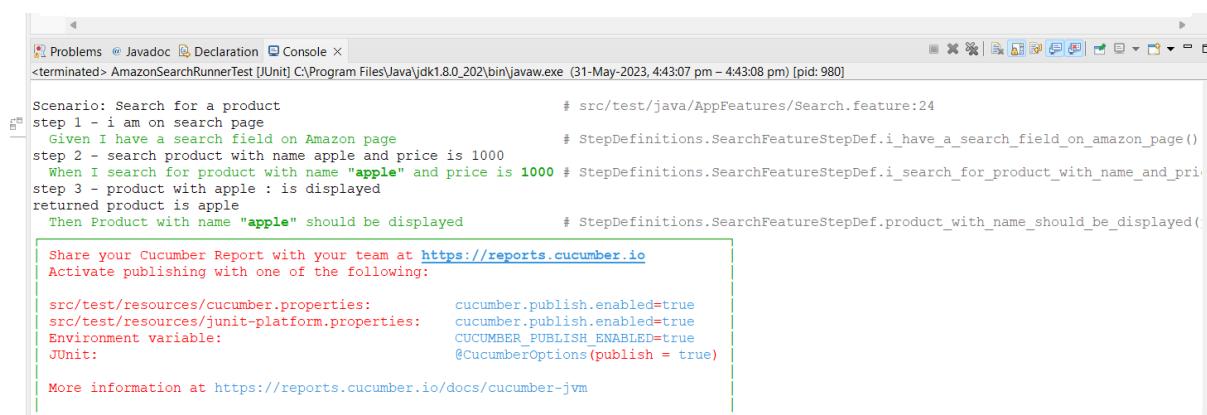
Now if we see we were not getting output when we ran using mvn we are only getting build success. No mention of steps passed/failed etc.

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:

```
21
22
23 public class AmazonSearchRunnerTest {
24
25 }
```

Now run using junit test:



The screenshot shows the Eclipse IDE interface with several open windows:

- Package Explorer**: Shows a single package named "Amazon Search" containing one class, "AmazonSearchRunnerTest". The status bar indicates "Finished after 0.254 seconds", "Runs: 1/1", "Errors: 0", and "Failures: 0".
- Search.feature**: The main feature file for the search functionality.
- AmazonSearchRunnerTest.java**: The runner class for the Cucumber test.
- Product.java**: A Java class representing a product.
- Search.java**: A Java class representing a search operation.
- AmazonSearchRunnerTest.java**: Another view of the runner class.
- Problems**: Shows no errors or warnings.
- Declaration**: Shows the declaration of the "Search" step definition.
- Console**: Displays the command-line output of the test run, including the scenario steps and their results.
- Failure Trace**: Shows the execution trace for the failed test.

The "Search.feature" file contains the following Cucumber steps:

```
Given I have a search field on Amazon page
When I search for product with name apple and price is 1000
Then Product with name "apple" should be displayed
```

The "AmazonSearchRunnerTest.java" file contains the following code:

```
@CucumberOptions(
 plugin = {"pretty"},
 features= {"src/test/java/AppFeatures"},
 glue = {"src/test/java/StepDefinitions"} //this will also work
)
public class AmazonSearchRunnerTest {
```

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.

The screenshot shows two terminal panes in an IDE. Both panes have tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The top pane displays the output of a Cucumber run for the 'AmazonSearchRunnerTest'. It includes step definitions and a summary of the test run. The bottom pane shows the Maven build output, indicating a successful build with no errors or warnings.

```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.699 s - in testRunners.AmazonSearchRunnerTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 2.908 s
[INFO] Finished at: 2023-05-31T16:44:16+02:00
[INFO]
```

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

Run from cmd:

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

```
? ? 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 ?
???
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.614 s - in testRunners.AmazonSearchRunnerTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.884 s
[INFO] Finished at: 2023-05-31T16:46:17+02:00
[INFO] -----
[WARNING]
[WARNING] Plugin validation issues were detected in 3 plugin(s)
[WARNING]
[WARNING] * org.apache.maven.plugins:maven-compiler-plugin:3.8.1
[WARNING] * org.apache.maven.plugins:maven-surefire-plugin:2.22.2
[WARNING] * org.apache.maven.plugins:maven-resources-plugin:3.3.0
[WARNING]
[WARNING] For more or less details, use 'maven.plugin.validation' property with one of the values (case insensitive): [BRIEF, DEFAULT, VERBOSE]
[WARNING]

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

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_price_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_displayed(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 |

---

