# 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.

A picture containing text, diagram, line, screenshot

Description automatically generated

# Let’s start cucumber project-

Maven project creation:

A screenshot of a project

Description automatically generated

Click next:

A screenshot of a computer

Description automatically generated

Search for maven plugin:

A screenshot of a computer

Description automatically generated with medium confidence

Give group id, artifact id:

A screenshot of a computer

Description automatically generated with medium confidence

Base project structure once created:

A screenshot of a computer program

Description automatically generated with medium confidence

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

A screenshot of a web page

Description automatically generated with medium confidence

Lots of good features

A screenshot of a computer

Description automatically generated

One way to install is drag and drop into eclipse.

But we will use the below link-

[Cucumber-Eclipse > Update Site](https://cucumber.github.io/cucumber-eclipse/update-site/)

Copy the browser url.

Open eclipse and go to the below location:

A screenshot of a computer

Description automatically generated

Enter the link.

Hit enter.

Select all the checkboxes inside Name field.

A screenshot of a computer

Description automatically generated with low confidence

Click next.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

You may have to restart eclipse for it to work.

A screenshot of a computer

Description automatically generated with medium confidence

Delete the dummy packages:

A screenshot of a computer

Description automatically generated

# Let’s add the required dependencies for project-

Search in google:

A screenshot of a computer

Description automatically generated with low confidence

Click maven:

A screenshot of a computer

Description automatically generated

Click pom.xml:

A screenshot of a computer

Description automatically generated with medium confidence

Copy the tags and values inside properties:

A picture containing text, font, line, screenshot

Description automatically generated

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:

A screen shot of a computer code

Description automatically generated with low confidence

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.

A screenshot of a computer

Description automatically generated with medium confidence

Copy them.

Remove existing dependency from pom and paste the copied one.

A screenshot of a computer program

Description automatically generated with low confidence

Copy the plugins from build section.

Taken screenshot of Naveen’s computer.

A picture containing text, screenshot, menu, font

Description automatically generated

Remove existing build from pom.

Paste the build here.

A screenshot of a computer program

Description automatically generated with medium confidence

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.

A screenshot of a computer program

Description automatically generated with low confidence

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,

A screenshot of a computer

Description automatically generated with medium confidence

And get the version number:

A screenshot of a computer

Description automatically generated with medium confidence

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

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer

Description automatically generated

See in pom if the version matches.

A picture containing text, font, screenshot, line

Description automatically generated

Remove the dummy packages:

A screenshot of a computer

Description automatically generated

These are created by default.

Expand maven dependencies and we can see cucumber present:

A screenshot of a computer

Description automatically generated

Build again the project:

Right click -> maven -> update project.

Select force update and click ok.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

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

A screenshot of a computer

Description automatically generated

# Where is .m2 located in eclipse-

Windows -> preferences -> maven -> user settings

A screenshot of a computer

Description automatically generated

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

Create a feature file.

Extension has to be given as “.feature”.

A screenshot of a computer

Description automatically generated

Sample feature file got created.

A screen shot of a computer

Description automatically generated with medium confidence

One feature file can have multiple scenarios.

This is the way to write multiple scenarios:

A picture containing text, screenshot, font, number

Description automatically generated

Getting this warning in console:



How to resolve it:

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

A screenshot of a computer

Description automatically generated with medium confidence

It will show building….

A screenshot of a computer

Description automatically generated with medium confidence

Now get this warning on the steps:

A screen shot of a computer

Description automatically generated with low confidence

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:

A screenshot of a computer

Description automatically generated

Console:

Nothing changed.

A screenshot of a computer

Description automatically generated with low confidence

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

Right click on feature.

Run as -> run configuration.

A screenshot of a computer

Description automatically generated

Select the “+” icon after clicking on cucumber.

A screenshot of a computer

Description automatically generated

Like this new window will come.

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

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

Click open.

Path has been updated.

A screenshot of a computer

Description automatically generated with medium confidence

Click apply and then run.

# For commenting in feature file use #-

A screenshot of a computer program

Description automatically generated with medium confidence

Feature file created:

A screenshot of a computer

Description automatically generated with medium confidence

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/java/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:

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer

Description automatically generated with medium confidence

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

A screenshot of a computer code

Description automatically generated with low confidence

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

Feature file:



Step definition file:



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

A picture containing text, font, line, screenshot

Description automatically generated

This exception is thrown because there is no java code:

A picture containing text, font, line, web page

Description automatically generated

If you run the feature file again:

A screenshot of a computer

Description automatically generated with medium confidence

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\_price\_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\_displayed(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:

A screenshot of a computer program

Description automatically generated with low confidence

|  |
| --- |
| #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:

A screenshot of a computer

Description automatically generated with medium confidence

|  |
| --- |
| **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:

A screenshot of a computer

Description automatically generated with medium confidence

|  |
| --- |
| **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:

A screenshot of a computer

Description automatically generated with medium confidence

|  |
| --- |
| **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:

A screenshot of a computer

Description automatically generated

You will get this error in runner class:

the type cucumber is deprecated issue.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

Search for this in google:

A screenshot of a computer

Description automatically generated with medium confidence

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:

A screenshot of a computer program

Description automatically generated with medium confidence

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

A screenshot of a computer

Description automatically generated

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.

A screen shot of a computer

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with low confidence

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)

A screenshot of a computer program

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated

This is because see the seeding in mvnrepo.

This version has the highest seeds.

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer

Description automatically generated

Run the runner class again as junit test:

A screenshot of a computer

Description automatically generated with medium confidence

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

A screen shot of a computer

Description automatically generated with low confidence

The above can be ignored.

Console output:

A screenshot of a computer

Description automatically generated with medium confidence

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

Green means passed.

A screenshot of a computer

Description automatically generated with medium confidence

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

Right click on project.

Run as maven test.

A screenshot of a computer

Description automatically generated

You should get build success.

A screenshot of a computer

Description automatically generated with medium confidence

To run from command line using maven-

Open cmd.

Be in path of project.

A screenshot of a computer

Description automatically generated

Check if maven is present.

A screen shot of a computer

Description automatically generated with medium confidence

It is not present.

How to set it up.

Goto google and search for download maven.

A screenshot of a computer

Description automatically generated with medium confidence

Once inside the link click on the highlighted zip file.

A screenshot of a computer

Description automatically generated with medium confidence

Extract it.

A screenshot of a computer

Description automatically generated with medium confidence

Copy this folder.

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

A screenshot of a computer

Description automatically generated with medium confidence

Go till this path mentioned in below image.

Copy the path.

A screenshot of a computer

Description automatically generated

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:

A screenshot of a computer program

Description automatically generated with medium confidence

Maven is installed.

Now go to the path of the project.

Type this command.



Hit enter.

You should get build success.

A screenshot of a computer

Description automatically generated

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:

A black text on a white background

Description automatically generated with medium confidence

Now run using junit test:

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

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.

A screenshot of a computer

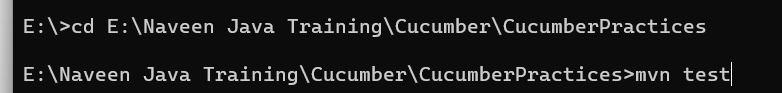
Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

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

Run from cmd:



A screenshot of a computer

Description automatically generated

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 │  └───────────────────────────────────────────────────────────────────────────────────┘ |