Most questions are asked from frame work, not from java, test ng or any other topic.

Most of 90% you have to work with frame works.

What is framework?

In java what still we are doing to create a project.

We created fist package and then we created classes

Then we created 2nd package and created class and so on

Package A

--- class a

---- class b

---- class c

Package B

Class --- x

Class --- y

Class --- z

This is what we did in java till now.

Here We mixed all folders and packages technical and non-technical type. If someone new person comes and see our coding who is non-technical, he will not understand our coding. Because we have written in our desired style. We make everything mess up.

Problems in above style to write code are

--- no proper structure, no technical files separate, non-technical files are not separate

--- no rules of code we are following

--- no standards to write package, class , this is your own desire create anywhere what you want.

--- no code reuse ability no protocol --- we are not following any protocol.

These are above mentioned problems are when we use a tool only.

But when we use frame work then --🡪 this provide us

--- organized structure

---proper folder hierarchy

---- separate place for separate code

----single execution point

Frame work gives us structure it says this is code – write in package

This is class—write there, this is technical --- write there, this is file – write there.

It is easily understood what to do, how to do, when to do inside a frame work.

If you join any industry or company in which there is already frame work you just will join company and will starting to use that existing frame work.

But if you join a company which did not has any frame work then there you have to create a frame work for all employees and company itself. You are now the driver of company.

If you create a frame work for your company then your worth will increase high. All company will be dependent on you.

Above mentioned this is the frame work.

In simple words; framework is pre-build set of tools whom you can use and it will be easy for you to build your software.

Cucumber is a testing framework with a focus on behaviour-driven development, and Maven is a build automation and project management tool. They serve different purposes in the software development lifecycle, and you can use them together by integrating Cucumber tests into a Maven-managed project.

**Class:01 Starting date: 21-Nov-23**

**what is framework?**

**in English frame work mean structure, base of something, dhaancha, pinjra,**

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

So for now what we have learnt ?

Programming language ------------------------------- java

Automated test software tool ------------------- selenium

Version control -------------------------------------- git

The tool which is used to build framework -------------------------------- maven

Sor for we have created java projects but now we will create maven projects

Now not ~~Java~~  , only maven projects

because In maven project We will having every single thing all things in just one maven project.

Now we are talking about management build tool with java and selenium

Now create first maven project.

Maven is building management tool.

Main folder: in this folder developers write their code, we don’t need it, delete this folder.

Test folder: in this folder tester write the code to test the application, so keep this folder.

Now create a folder inside test folder name this folder resources, this folder is used to save non technical code to write.

Now there are two folders one is java --- for technical files

Resources --- for non technical files

**What is pom xml file??**

This is configuration file which is considered as backbone of maven project.

In java we were downloaded many jar files but

Maven frame works says not to need to download any jar any more, its gone.

Now you have to work with selenium, test-Ng, cucumber in maven then you are not going to download any jar then what should you have to do. Then pom.xml file says that you just download the dependency I will automatically download all files.

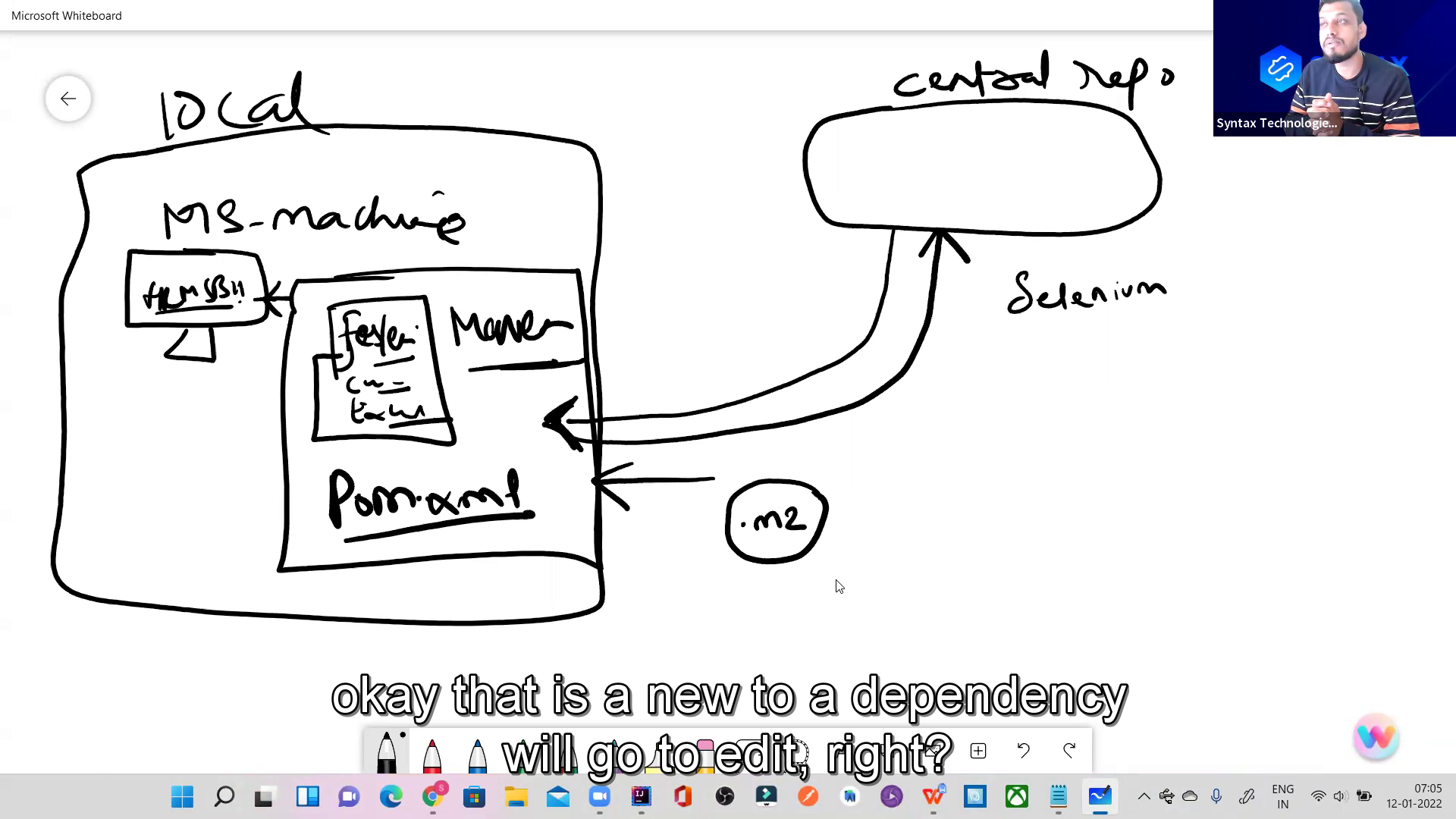
**How maven works, mean how it download jars and drivers automatically just by adding the dependency??**

When we put dependency in pom file this local pc is connected to central repository

Pom file send signal to repository is this dependency available here the central says yes and he transfer this file when we refresh our pom file. In our pc .m2 folder is created which store all jars and drivers in local pc.

**This is one scenario;**

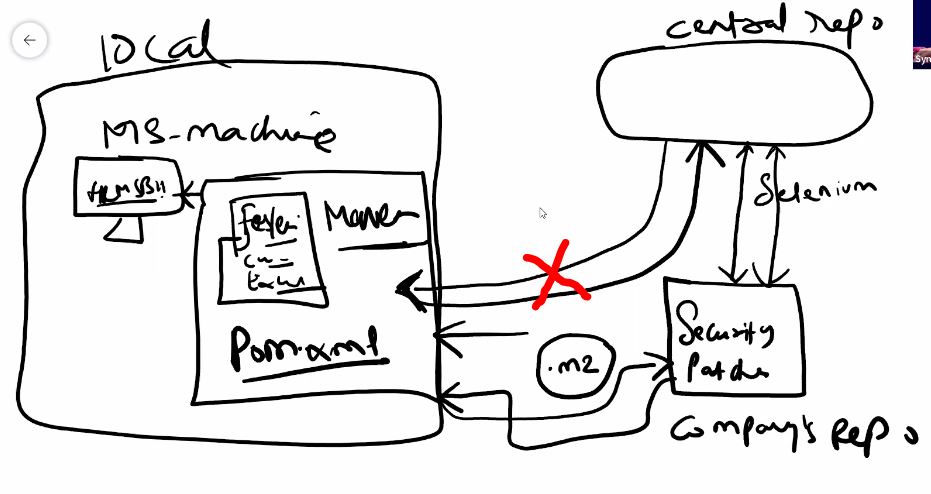
In this photo local machine or local repository is directly connected to central repository. On central repository is there are lot of adds, there is a highly security risk.



**2nd scenario;**

1. company repository

2.security patches.

****

**What is this mean?????**

<groupId>org.apache.poi</groupId>  
<artifactId>poi</artifactId>  
<version>4.1.2</version>

This dependency is present in pom file and this dependency pom file says to central repository , hey central do you have this group-id and this artifact-id and this version , if you have then please transfer this to me local machine, then in local machine .m2 folder is created and jars and libraries are downloaded in this folder.

Here are main two scenario or you say advantages and disadvantages of central repository.

Mariyam is project manager and five people are team member

Then everyone adding all jar files in his machine from central repository.

But here is disadvantage of this may be one member is downloading selenium version 4.2 of dependency or other person downloading other version if five people are downloading different, different version then some using updated version and some using older version then there might be problem or error, because someone using less version and some using new version then there may be error of whose code which is using older version. Because each version has different functionality.

What is solution of this case

In this case only Mariyam downloads jars and she will upload code on git hub and all other members will copy that code and will use same code and jars and drivers without no error.

2nd scenario if you are working on government project and they never allow you to direct link with central repository. Because of security issue. Then in this case company will add another repository inside company repository and this company repository will connect to central repository and this company repository will have security patches, and in this case your pc will connect indirectly with company repository to save from security crack issue.

Central repository mean maven repository maven repository website on internet or website of maven. It has lot of add on his page there may be security issue.

What is web Driver this is equal to Chrome driver this is an interface why this is interface because it has Own definition why it has own definition because it has Multiple situations why this has multiple situations because it has Multipl business cases

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

Example of Ammar who is business owner of automobile industry.

What is cucumber?

It is a software tool which test other software’s.

Why we need cucumber when there are lot of tools like tesNg, selenium, java, maven.

Explanation understands

Initially cucumber was just supporting ruby language. But now it supports java also.

**my-archetype/ mean template- archetype is blueprint and his structure is as follow.**

|-- src/

| |-- main/

| |-- resources/

| |-- archetype-resources/

| |-- src/

| |-- main/

| |-- java/

| |-- resources/

|-- pom.xml

|-- archetype-metadata.xml

**What does cucumber do ? benefits of cucumber.**

1. It increases the Transparency
2. Remove the communication gap
3. It engages more clients.
4. Cucumber solves the communication gap
5. More collaboration (kaam ma shirkat ) if ammar comes and see the project in gherkin language he will give suggestion or feed back and will collaborate more.
6. Focus on end user, it means the person who will use that software. A good tester consider himself as user of that software and use that software itself to check the functionality.
7. Business value, when you use BDD it mean this module provide the clients to tell their priority what page they need first and what things they need first to complete. When the priority pages are mage owner will start to do work will earn money and this will increase the business values.

Cucumber supports gherkin language

What is gherkin language?

Gherkin language is nothing just plane simple English language. Which we will write with some set of rules.

رویے پر مبنی ترقی

In very simple terms, BDD or behavior-driven development, is a technique where your specifications or test cases are written in plain English like sentences.

If we are going working with gherkin language there are set of protocols which we are follow.

What are those set of protocols? There are some keywords these keywords has specific meanings.

Whenever we deal with gherkin language when ever we deal with BDD part we have to keep these keywords in mind. Which keyword what mean.

1. Given - --- > precondition ------ Given user is navigated to the HRMs application
2. When -- -🡪 actions -🡪 when the user enter password and username

like send keys, click, select dropdown, double click etc.

1. And ---- -> supporting tag, this tag comes with every other keyword for support.
2. And user click on login button
3. Then ---🡪 expected output, result ---🡪 Then user should be able to see dash board.
4. Feature-----🡪 this keyword describe the functionality or user story.
5. Scenario-------🡪 this keyword describe how the functionality has to be performed.
6. Background--------🡪 this keyword takes all common steps present in all scenarios.

**There are three types of frame work**

1. Keyword driven framework ----------🡪 generally base on keyword, we pass the data that’s why we called KDD, keyword is going to drive
2. Data driven framework ---------------🡪 when your script execute across multiple sets of data it is called DDT, data is going to drive/ test next generation is DDT framework.
3. Behavioural driven development framework. -----------------> instead of writing showing or writing the code you write the behaviour of application, behaviour is going to drive. Now what does mean of behaviour, it is to define some one’s nature is called behaviour. What it do, what it say, what it eat, what it like.

Whose behaviour we are defining? we are defining the behaviour of test cases which we write multiple test case. It means We are writing in plain English language

BDD ---- defines the behaviour and to define the behaviour the language which we used is gherkin language.

BBD------ >> in simple words, instead of showing actual code to the clients, show him the behaviour of application/show him the functionality of the application/show him the user story. Behaviour is going to drive the framework.

First we will write test cases in gherkin language and then we see how to handle them.

What is hybrid frame work ? if frame work supports both DDT and BDD then it is called hybrid frame work.

**DDT :** in ddt mean there is data code, there is my code, if my execution automation scripts execute across multiple sets of data this is called data driven test.

**BDD:**  it simply means it define the behaviour of test case, here is plain simple English language you will find. If there is my plain English language,

the scenario in which I am writing these scenarios also executing across multiple sets of data then we say it is DDT and BDD both mean it is hybrid frame work.

BDD defines the behaviour and the language which defines the behaviour is gherkin language.



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

**Class 02 Date; 17-12-2023**

Feature --------one functionality is called one feature. describes the overall feature of functionality being tested. The file in which we write all the feature or test cases or scenario is called feature file.

Scenario--------test case is called scenario. One test case, one scenario. Describes a specific test scenario or case related to the feature.

Feature --------- > User Authentication

Scenario --------- > Successful Login

Given --------- > the user is on the login page

When --------- > the user enters valid credentials

Then --------- > the user should be logged in successfully

Scenario --------- > Unsuccessful Login

Given --------- >the user is on the login page

Then --------- >the user should see an error message

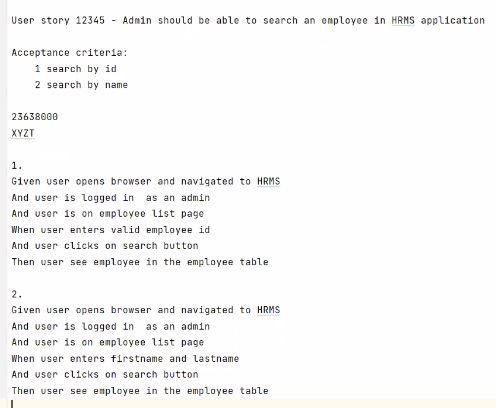
----------------------------------------------------------------------------------------------------

Note; don’t use keywords repeatedly just use and keyword to repeat every time.

The **.properties** file format is a simple key-value pair configuration text file.

Feature is what the functionality you want to test ????? example; user story 123 search employe.

Scenario ; how to test the functionality. Example ; search employee by id.



**In frame work we need**

1. java ---------- > jdk

2.cucumber --- > dependency

**i.**[**Cucumber JVM: Java**](https://mvnrepository.com/artifact/io.cucumber/cucumber-java)

[**io.cucumber**](https://mvnrepository.com/artifact/io.cucumber)**»**[**cucumber-java**](https://mvnrepository.com/artifact/io.cucumber/cucumber-java)

**ii.**[**Cucumber JVM: JUnit**](https://mvnrepository.com/artifact/io.cucumber/cucumber-junit)

[**io.cucumber**](https://mvnrepository.com/artifact/io.cucumber)**»**[**cucumber-junit**](https://mvnrepository.com/artifact/io.cucumber/cucumber-junit)

**3.selenium**

i**.**[**Selenium Java**](https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java)

[**org.seleniumhq.selenium**](https://mvnrepository.com/artifact/org.seleniumhq.selenium)**»**[**selenium-java**](https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java)

**ii.**[**Selenium Chrome Driver**](https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-chrome-driver)

[**org.seleniumhq.selenium**](https://mvnrepository.com/artifact/org.seleniumhq.selenium)**»**[**selenium-chrome-driver**](https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-chrome-driver)

**4.excel reader jars/or library**

i.**.**[**Apache POI Common**](https://mvnrepository.com/artifact/org.apache.poi/poi)

[**org.apache.poi**](https://mvnrepository.com/artifact/org.apache.poi)**»**[**poi**](https://mvnrepository.com/artifact/org.apache.poi/poi)

## . [Apache POI](https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml-schemas)

**5.webdrivermanager**

[WebDriverManager](https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager)

[io.github.bonigarcia](https://mvnrepository.com/artifact/io.github.bonigarcia) » [webdrivermanager](https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager)

[**org.apache.poi**](https://mvnrepository.com/artifact/org.apache.poi)**»**[**poi-ooxml-schemas**](https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml-schemas)

**in intellij download plugins**

**1. cucumber for java**

2.gherkin

3.properties

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

Step declaration; writing the cucumber scenario in feature file is in gherkin language is called step declaration. Only feature file you can not execute. Step declaration is only skeleton.

Scenario: User logs into HRMS

Given user navigated to hrms website

When user enters valid credentials

Then user should be logged into HRMS

Step definition; implementation of above scenario in java is called step definition. a step definition is a piece of code associated with a specific Gherkin step in a scenario.

**Annotation (@Then):** This indicates that the following method is a step definition for a Cucumber scenario's "Then" step.

This is a step definition of a cucumber’s scenario.

@Given ("user navigated to hrms website")  
public void user\_navigated\_to\_hrms\_website() {  
 openBrowser();  
}

Gherkin is a language used to write human-readable descriptions of software behaviours.

Class 03 date;12-20-2023

**Benefits of cucumber?**

Why we need cucumber if there is testNG??

1. increase the transparency, with clients and software team

2. reduce communication gap

3.cucumber supports BDD/gherkin language, it is easy to understand for non-technical people.

4.focus on end user – you always say user user mean you are using this app as user.

5. you don’t need to write manual test cases; you can link your jira with framework.

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

We need a feature file to write test scenario in our cucumber frame work.

In a feature file we can write lot of scenarios, like login with valid, login with invalid and other login

But in a feature file you can have only one feature. Because in one feature file you are describing one feature of any software. This feature may be very large. One file describes only one user story.

Every step declared in cucumber scenario generate the java code.

And keyword is supporting key word. In step definition And keyword is not annotation, the annotation is only showed the actual keyword for which this is using for supporting.

Multiple inheritance in not allowed in java.

1. login step definition are present in one class login class

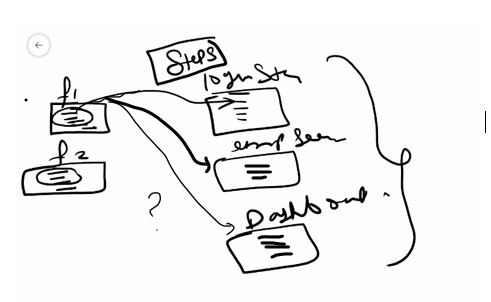
2. login step definition you don’t need to copy and paste in employee id search class. If you do this there will be multiple inheritance.

Suppose you have multiple classes in one java class, and if there comes error it is difficult to solve that error rather in if they are separately created.

Maven knows which class to use as the entry point.

**How multiple inheritance not allowed????????**





In above f1 feature file is confused that from which class I have to take step definition of java code. This is multiple inheritance not allowed in java.

Step definition; to implement the code for steps written in feature file. We use step definition.

**Runner class??**

in cucumber at till this point class 3 we can only run one feature file we can not run 2 or multiple feature files at same time.

Runner class is used to execute more then one feature files.

Runner class is **the configuration class** it is considered as back-bone .

Runner class only used to execute the feature files.

For link between feature files and steps classes/for link between step declaration and step definition cucumber is used.

Got to deceleration or usage option is basically showing link between step declaration and step definition.

**What is Junit???**

JUnit is a popular testing framework for Java that is used to write and run unit tests. Unit testing is a software testing technique in which individual units or components of a software application are tested in isolation to ensure they behave as expected.

**Annotations:**

**Is just like a label, which mean what test piece or which test part or which test step do you want to test.**

Think of annotations like special labels on ingredients. For JUnit, it's like putting a label on the cheese to say, "Let's test if this cheese tastes good."

**What is unit testing???????**

Unit testing is a software testing technique where individual units or components of a software application are tested in isolation. The term "unit" refers to the smallest testable part of an application, usually a function, method, or class.

**We are using cucumber with junit.**

import org.junit.runner.RunWith;  
  
@RunWith(Cucumber.class) // this is linking cucumber and junit.

this org.junit will execute my cucumber based on this particular class.

**@RunWith(Cucumber.class)**: This line is using the **@RunWith** annotation to tell JUnit to use the Cucumber test runner (**Cucumber.class**). This is a key part of integrating Cucumber with JUnit. It essentially says, "Hey JUnit, when you run tests in this class, use the Cucumber test runner to interpret and execute the Cucumber features."

In JUnit-based frameworks, the **@RunWith** annotation is used to specify the test runner class. In the provided code, **@RunWith(Cucumber.class)** indicates that the Cucumber class is the test runner.

**We are going to configure a runner class.**

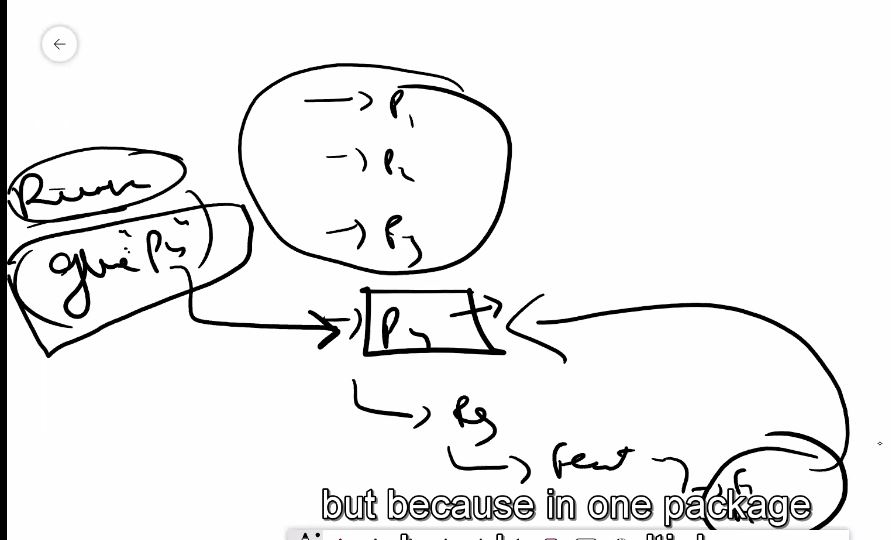
**The purpose of runner class is to execute all feature files not a single feature file from one point.**

@CucumberOptions ()

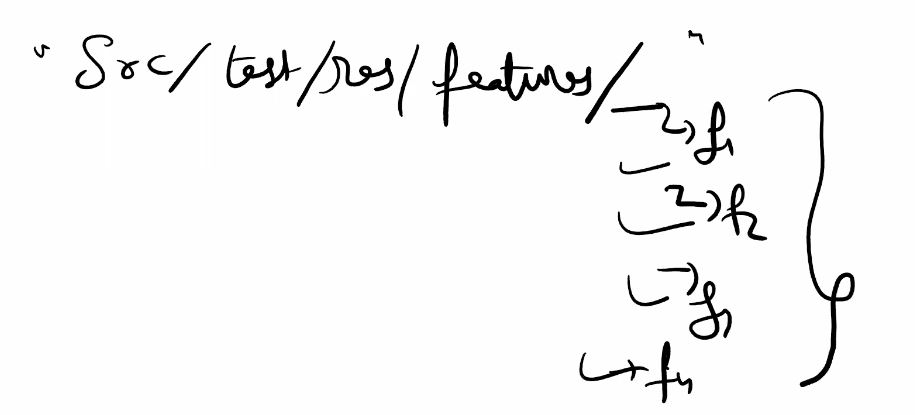
The **@CucumberOptions** annotation in Cucumber is used to provide various configuration options for your Cucumber tests. It allows you to specify settings such as the location of your feature files, the location of your step definitions, formatting options for the test results, and more.

**There are 6 cucumber options generally used**.

1. Feature ---path of feature files
2. Glue---- path of steps, codes generated against feature files
3. Dry-run ---------- to check unimplemented steps
4. Monochrome ---------- to remove unreadable code from console
5. Tags------------------to run smoke and regression test
6. Plugins---------------------to generate html file of your test cases.



If in feature you provide the path till feature folder then it execute all feature files.



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

**1. features**: Specifies the path to the directory or directories where your feature files are located. Feature files contain the descriptions of the behavior you are testing using Gherkin syntax.

**2. glue**: Specifies the package or packages where your step definitions are located. Step definitions contain the Java code that maps to the steps defined in your feature files.

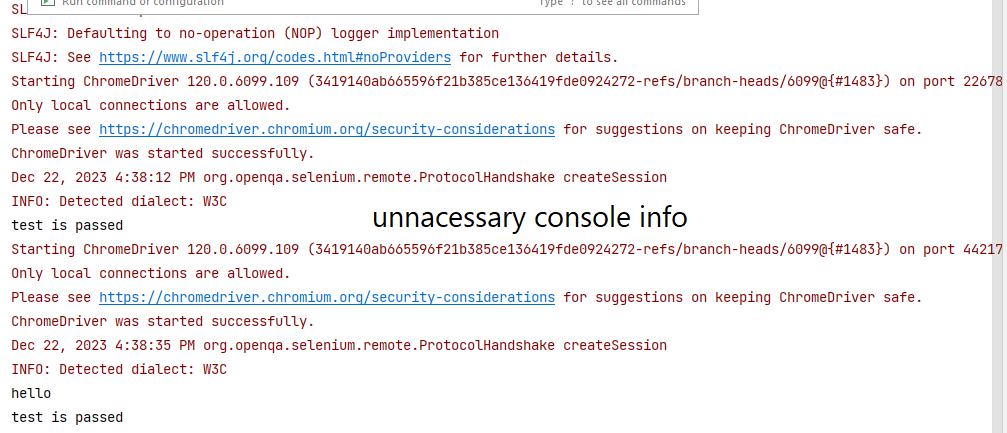
**3. Dry-run**= true; this command is used to tell the unimplemented steps in your feature files and give their undefined code also,

It also tells about implemented steps, implemented steps ticks green colour on side window.s

**Dry run** command does not execute the chrome browser, it just scans all the codes and give the code for undefined steps.

**Dry run** command second saves your time, which was wasting in running chrome and then getting code for undefined steps.

**Dry run=false**  this false command will execute your chrome whole execution.

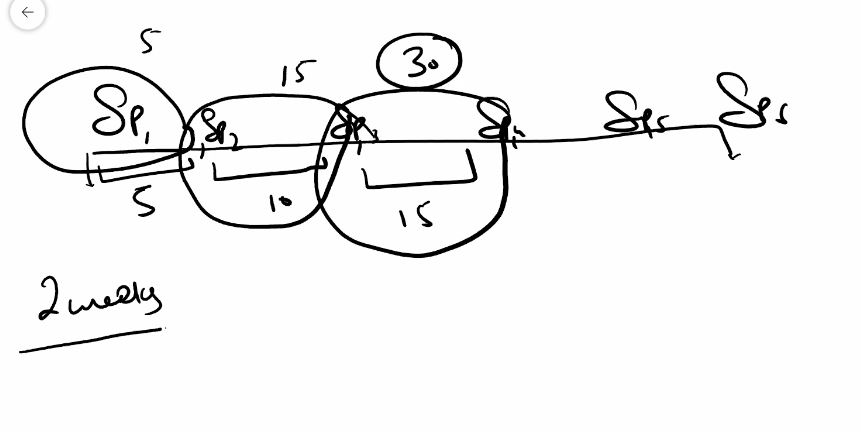
**4.Monochrome = true** this command is used to remove unnecessary unreadable irrelevant information from console and gives only useful information.

Feature = “src/test/feature”

Glue = “steps”

Feature and glue keywords, cucumber is glueing/sticking step definition with feature files/step declaration.

Sprint scenario;



**5. @tags**

Scenario is that; when your boss at the end of 3rd sprint says hey babar please run the smoke test then how you can do or boss says hey run the just one scenario that is search employee by name then what will you do. Then you will use tags in your feature file.

@Smoke test

@Regression test

In Cucumber, the **@tags** annotation is used to include or exclude specific scenarios or features during test execution.

Tags = “@smoke or @regression not @login” you can use combination of tags.

You can also use logical AND (**and**) and logical NOT (**not**) operators to create more complex tag expressions.

Tags = “@smoke” ------ are used to run the scenario of your own choice. You can add multiple tags to a single scenario.

**6. plugin = {"pretty", "html: target/cucumber.html"}**

if your boss says please share me your cucumber report then plugin is the command to generate html report.

And this report is helpful for your team members, for your proof and many other purpose are there.

**"pretty":** is format style. Is Outputs the results in a human-readable format to the console. It formats the output in a clean and readable way, making it easier for humans to understand the test results.

When you run Cucumber tests with the "pretty" format, the output will be displayed in a clear and structured manner on the console or in the specified output location. It typically includes colours, indentation, and other formatting styles to make it easier for developers and testers to interpret the results.

target/cucumber.html; this is the path where html file is generated other then this path html can not be generated. File can not be generated in excel, word, ppt or other format.

Target is the folder for java code for which is only machine readable code. But pretty command format style generate human readable file in this machine readable code.

**"html:target/cucumber.html":** Generates an HTML report and saves it in the specified directory (**target/cucumber.html**). The HTML report typically includes information about the executed scenarios, their status, and any additional details specified in the Gherkin feature files.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**cucumber background keyword**; In Cucumber, the **Background** keyword is used to define a set of steps that are common to all scenarios in a feature file. These steps are executed before each scenario in the feature file. The purpose of the **Background** section is to reduce redundancy in your feature files by allowing you to specify common steps once, rather than repeating them in every scenario.

Common steps mean common between 2 or more then 2 scenarios not in a single scenario.

-

-

-

-

-

**Class 04** Date;12-23-**2023**

**Scenario;** we know that about background keyword, what this keyword does, this keyword is used to write common steps from all scenario present in one feature file. You can say that this is precondition before run of every scenario.

Similarly you notice that some common steps are also common in many feature files or in all feature files. They steps are opening the browser and closing the browser hrm application.

Hooks ----- >> we provide hooks for precondition and post condition to write common steps of many feature files.

We provide these hooks under the step package where our java code is present.

Without java code feature files are useless.

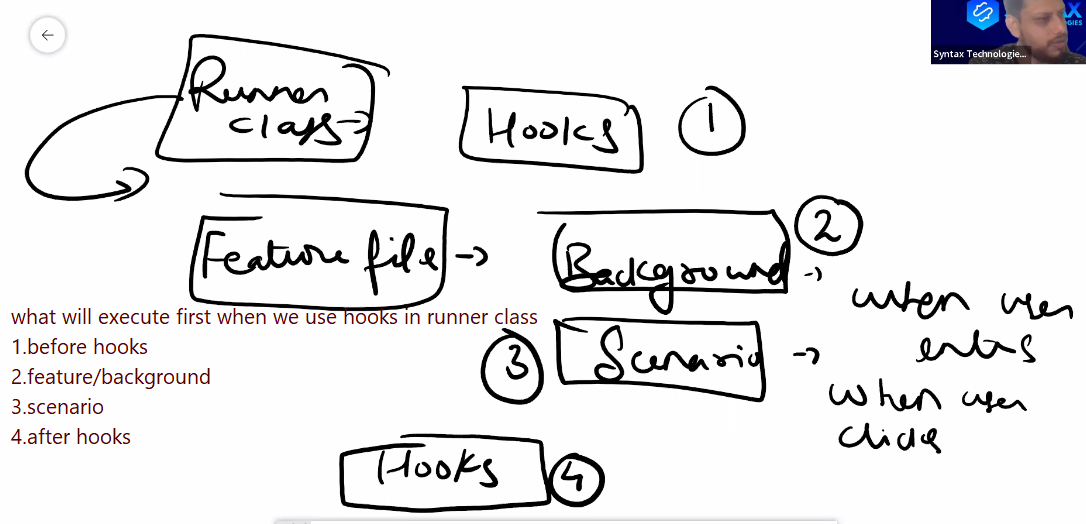
There are two hooks in cucumber .

**@Before** hooks will be run before the first step of each scenario. They will run in the same order of which they are registered.

**@After** hooks will be run after the last step of each scenario, even when there are failing, undefined, pending or skipped steps. They will run in the opposite order of which they are registered.

Cucumber supports only two hooks (Before & After) which works at the start and the end of the test scenario.

We just need to define hooks, no need to associate the hooks, cucumber takes care of associating.



**There could be many runners class,**

**1. smoke class**

**2. regression class**

