Chapter-1 : Maven.

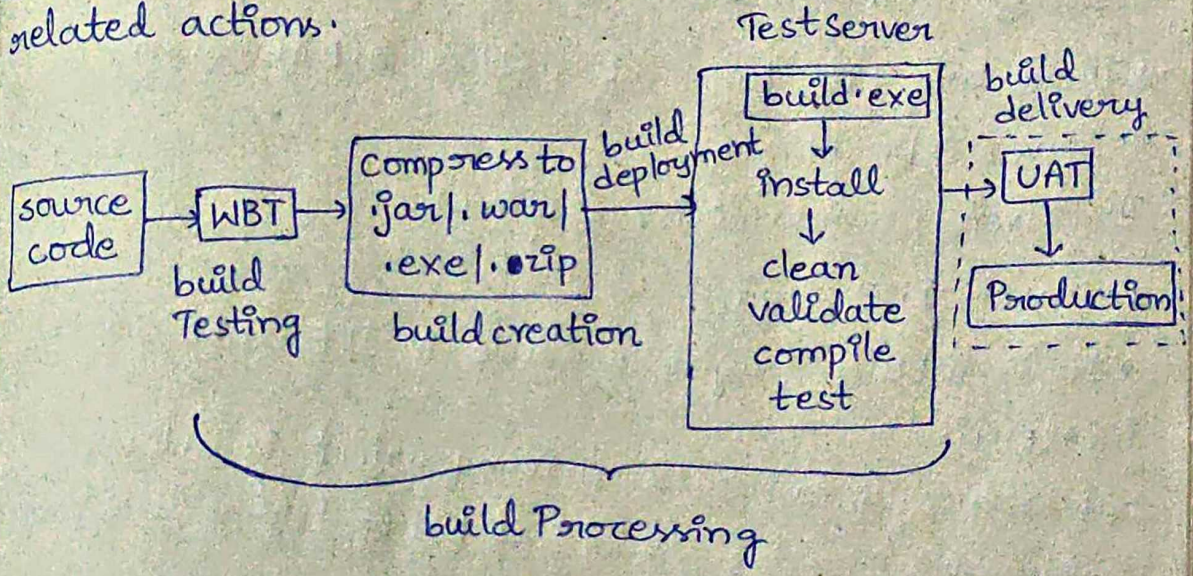
→ Maven is a build Management Tool. which involves build processing and build delivery.

→ Maven is used by developers, automation Test engineers and devops.

⇒ Developers Usage:-

→ Incase of development, maven is used as build process Tool (Build creation + Build Testing + Build deployment).

→ various maven commands are used for these build related actions.



⇒ Build Creation:-

→ The process of converting the source code into any executable format (.jar|.war|.exe).

⇒ **Build Testing:-** Testing the build to check if it is broken (or) not. by checking the status like pass/fail.

⇒ **Build Deployment:-** The process of installing the build into testing server (or) any other environment.

⇒ Testing usage:-

→ In automation testing, maven is used for testing the application with the current (or) recent frameworks by executing the test scripts.

→ It is also used to check integration issues b/w the framework components.
→ When multiple test engineers are working with the same framework there might be chances that changes done by one test engineer effects the entire framework.
→ In order to overcome this issue we use maven to build framework.

⇒ Devops usage:-

→ It is used for all the build processes.

⇒ There are two types of softwares in maven:-

1). Maven IDE plugin
2). Maven commandline plugin.

1). Maven IDE plugin:-

→ It is inbuilt plugin available in IDE's like eclipse, intelellij.
→ It allows us to create maven project in eclipse.
→ It also provides us with default folder structure.

→ Maven folder structure:-

→ src/main/java → genericUtilities, pomClasses.
→ src/main/resources → driver executable files
framework user guide
→ src/test/java → test scripts
→ src/test/resources → external files
* properties
* excel
→ target → results of execution
→ test-output → reports, failed xml
→ pom-xml → heart of Maven project
dependencies & plugins

.m2 → c:Users/——/

→ Dependency is the advanced feature in maven which downloads required jar files from cloud repository (MVN repository.com) to local reposito (.m2 folder) and it adds these jar files to the -ry maven project automatically.

2). Maven command line plugin :-

→ This plugin should be downloaded explicitly
→ It is used for command line executions of maven.

⇒ Advantages of Maven :-

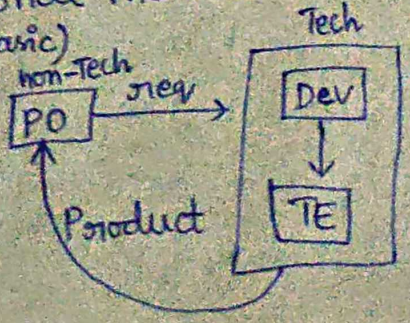
1). It checks integration and compilation issues b/w framework components.
2). It handles dependency jar files through pom.xml
3). It creates quick configuration setup for frame work.
4). It provides us with default folder structure.
5). It majorly supports jenkins.

⇒ Framework :-

→ It is well organized structured collection of reusable components such as generic libraries and object repository which facilitates test development modification and execution faster and easier.
→ There are two approaches to develop framework :-
1). TDD (Test driven development)
2). BDD (Behavior driven development)

1). TDD (Test driven development) :-

→ It is traditional method to develop a framework.



→ In this approach, the framework is developed considering all the technical aspects such as:

- → Resources
- → skills
- → architecture
- → programming language
- → Tools etc.

→ The communication b/w technical and non-technical teams is not enforced.

→ In the testing point of view manual test engineers will develop the test cases and store it in excel.

→ Automation test engineers develop test scripts looking into this excel.

→ Tools used here are : TestNG, JUnit.

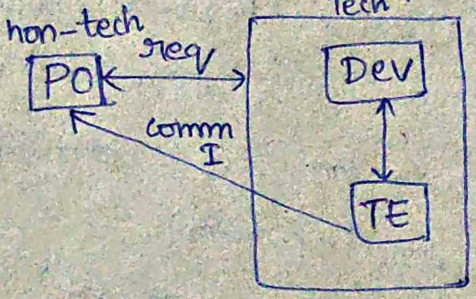
→ @Test is the driving factor here.

→ This approach has lot of reusable methods.

2). BDD (Behavior driven development):

→ In this approach, communication b/w technical and non-technical teams is given the 1st priority.

→ and then the technical aspects.



→ In this approach, feature files are developed which act like manual testcase documents / reference documents to the entire team.

→ These feature files contain scenarios that are developed using "Gherkin language".

→ Gherkin language is a normal english language with pref pre defined keywords.

→ step definitions (code) are developed for each and every step in feature files.
→ Even a non-technical person can perform automation testing by just executing feature files.
→ feature files are driving factor here.
→ Tools used here are : Cucumber, specflow, JBehave, Kahlan etc.
→ BDD framework is little complicated compare to TDD.
→ Less Reusable methods.

⇒ Types of Framework :

→ 1). Data Driven Framework
2). Method Driven Framework
3). Modular Driven Framework
4). Keyword Driven Framework
5). Hybrid Framework

1). Data Driven Framework :

→ whenever the application has huge test data then we prefer data driven framework.
→ Here, we store data in external resources like properties file (or) excel file (or) data providers and utilize it for test execution.
→ Examples of applications : e-commerce, banking, finance, travel etc.
→ Here, test data is the driving factor.

2). Method Driven framework :

→ whenever, the application contains more repeated functionalities like too many dropdowns, too many frames, more windows we prefer method driven framework.
→ Here, we develop generic methods and use them multiple times for test script development.

→ Examples of applications: e-commerce, CRM, healthca
→ Reusable methods are driving factors here.

3). Modular Driven Framework:

→ Whenever, the application is huge and has lot of modules we prefer modular driven framework
→ Here, test scripts, test data and suite files are maintained module wise to make debugging process easy.
→ Examples of applications: CRM, e-commerce.
→ modules are the driving factors here.

4). Keyword Driven Framework:

→ Whenever, Manual testers, freshers have to perform automation they would not be good at coding hence, we prefer keyword driven framework
→ Here, we create keyword library for the elements and the actions and utilize these keywords to develop the test scripts.
→ These keywords are stored in excel sheets.
→ Examples of applications: small applications like education, job portals.
→ Keywords are the driving factors here.

5). Hybrid Framework:

→ The combination of two (or) more frameworks is called as hybrid framework.
→ Examples of applications: all long term projects like e-commerce, CRM.
→ possible combinations: Datadriven + modular driven, datadriven + Method driven, datadriven + modular driven + Method driven, datadriven + Keyword driven, datadriven + Keyword driven + Method driven.

stages to develop Framework:

→ choice of framework → Hybrid (DataDriven + method driven)

1). Design:

→ create Maven project & add dependencies to pom.xml.

→ Add test data template (properties & excel file is added).

→ Develop generic libraries

1). IConstantPath (I)
2). PropertiesUtility (C)
3). ExcelUtility (C)
4). JavaUtility (C)
5). WebDriverUtility (C)
6). ListnerImplementation (C)
7). RetryImplementation (C)
8). BaseClass (C)

2). Development:

→ Develop POM classes.

→ Develop test scripts.

3). Execution:

→ convert test classes into suite files & execute suite file.

→ push the project to github, cloud repository.

→ Execute the project in Jenkins.

⇒ Login to application, create new user. validate it and delete the user. logout of application. (Scenario-1).

⇒ Scenario-2:-

→ Login to application, create new course. validate it and delete the course. logout of application.

⇒ Scenario-3:-

→ Login to application, create new category. Validate it and delete the category. logout of application.

⇒ Git:-

→ It is a version control tool (or) source control tool which is used to track and manage changes in the build (or) the software (or) source code.

→ It is usually used for co-ordinating work among engineers. who are collaboratively developing the software.

→ Git is used by developers, manual test engineers, automation test engineers and devops.

⇒ Developers usage:- They use it to store sourcecode.

Manual Test engineers usage:- They use it to store CRS (or) SRS (or) testcase documents.

Automation test engineers usage:- They use it to store automation framework.

Devops usage:- They use it to store build versions.

→ Git has two different softwares:-

1). Github
2). Git Client

1). GitHub:- It is cloud based distributed decentralized reporitory. used to store CRS (or) SRS (or) sourcecode (or) automation framework (or) build versions.

→ We can just create an account in Github.com and use it.

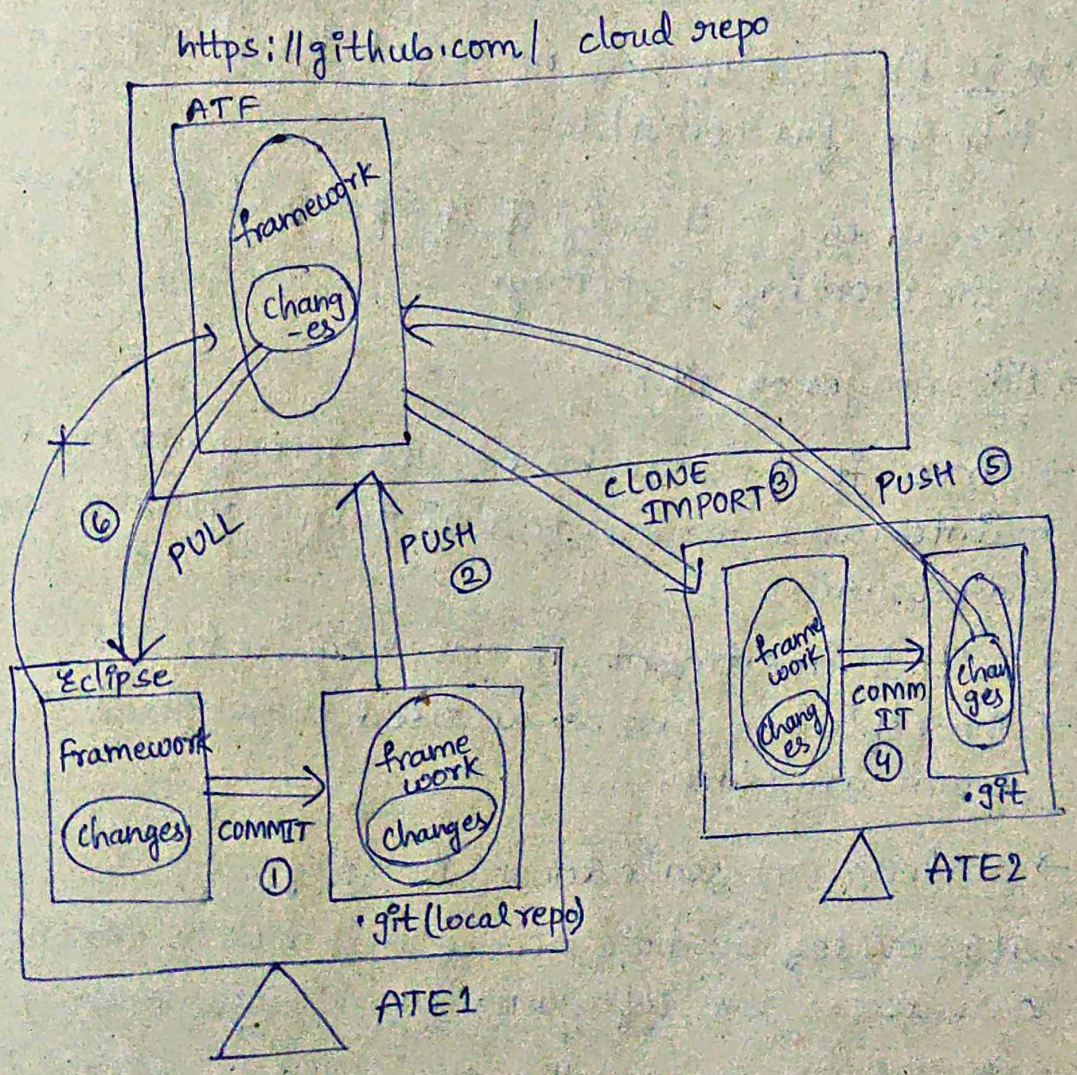
a) Git client:- This is the software which should be downloaded into our local devices.

1). Egit → It is inbuilt in IDE's which is responsible to perform push and pull operations of our source code to local (or) global repositories.

2). Git bash → It should be downloaded explicitly into our local devices.

→ It is used for command line execution of all git related operations.

⇒ Git Architecture:-



⇒ Why git is decentralized repository:-

→ Git is called decentralized repository because in git before pushing the code to github we have to COMMIT the code to local repository first and make sure the code is working in local repository.

then push the code to github (global repository).

⇒ Git commands:

1). COMMIT: It will copy the framework from working directory to local repository. (.git folder)

2). PUSH: It will copy the framework from local repository to global repository. (github)

3). IMPORT + CLONE: Import will get the framework from global repository to the local systems.

→ Clone will create a folder with the same name as the framework in local system.

4). PULL: It will help us to get the changes pushed into the framework.

⇒ We can use pull only if the project is existing in the working directory.

⇒ Advantages of git:

→ Since, it is cloud based repository, no need of maintenance team to maintain software (or) hardware.

→ Initial investment is not required.

→ The software can be accessed through internet from anyplace.

→ scaleup (or) scaledown is easy.

→ It provides remote access that means contributors can access the softwares from anywhere.

→ File share among the team members is easier.

→ It provides history for changes made by users.

→ Jenkins always gets the latest framework from git for batch Execution.

⇒ steps to create repository in github :

→ login to github.com
→ click on the [+] on the top right corner. click on New repository.
→ provide Repository name : SkillrayA2
→ click on create Repository.

⇒ steps to generate github token :

→ click on profile icon on top right corner.
→ click on settings.
→ scroll down and click on developer settings in the side menu.
→ click on personal access tokens and then click on tokens (classic).
→ click on generate new token and then click on generate new token (classic).
→ provide token name : Generic and set the expiration
→ click on repo ☑.
→ click on generate token.
→ copy the token and save it in a file.

⇒ steps to create local repository :-

→ Right click on the project → Team → share project.
→ select use (or) create repository in parent folder checkbox.
→ click on the project name and then click on create repository. click on finish.

⇒ steps to commit the project to local repository : & ~~pus~~ push it to global repo.

→ Right click on the project → Team → commit.
→ click on + icon. (All the files will be moved from Unstaged changes to staged changes.
→ Give commit message. (Framework A2 first Commit).
→ click on commit button. (The project will be pused to ⇒ local repository (or) .git folder).

→ login to github.com and open the repository.
→ Go back to eclipse, click on push Head button.
→ copy the url from global repository.
→ paste it in URI field (host and repository path are automatically filled).
→ Give github username and github token in password section.
→ click on preview, again click on preview and click on push.

⇒ jenkins installation and steps to download:

→ open the browser and type jenkins download.
→ click on the 1st link.
→ under Download jenkins 2.426.3 LTS and click on windows it will download.
→ double click on the jenkins file. click on next and again click on next.
→ select Run service as local system radio button.
→ click on next.
→ click on Test Port. click on next. set the path.
→ click on next twice and click on install.
→ click on yes and click on finish.
→ open the browser and type localhost:8080 press enter and copy that path.
→ open a new tab and paste it jenkins password will be displayed.
→ paste the password in administation password.
→ click on continue. click on install suggested plugins.

⇒ steps to install plugins: ① Maven integration plugin ② Github integration plugin

→ Manage Jenkins → Plugins → Available plugins.
→ Type maven integration plugin.
→ select the ☑ and click install.
→ should get all green ticks.

→ go back to manage jenkins → available plugins.
→ Type github integration. select the ☑ and click install.
→ should get all green ticks.

⇒ Path settings:-
① JDK path
② Git path
③ Maven path
} Manage jenkins → Tools → click on add JDK.
name: JAVA_HOME

→ open file explorer, Local Disk:C → program files → java → jdk-17. copy the path. (C:\Program Files\Java\jdk-17)
→ paste it in Java_HOME.
→ scroll till Git installations, change the name to GIT.
→ Go to file explorer, local Dis:C → program files → git → bin → copy the path (C:\Program Files\Git\bin) paste it before git.exe and paste it & give "\" in the path to Git executable.
→ scroll till maven installations, click on add maven. give the name as MAVEN_HOME.
→ click on apply then click on save.

→ How to create a job in jenkins:-
→ click on new item.
→ enter the name: SkillraryA2
→ select maven project and click on OK.
→ open new tab, and login to github account and open your reporitory. click on code.
→ In jenkins, go to source code management select the option git.
→ Go to github, copy the code url and paste it in Repository url.
→ Go to build in jenkins, give test in Goals and options.
→ click on apply and click on save.
→ click on build now. click on the link and console output.

⇒ Jenkins:

→ Jenkins is a CI/CD Tool used by developers, Devops and Automation Testers.

CI: continuous Integration.

CD: Continuous Development, continuous Deployment, continuous delivery.

Basically, Jenkins automates:

→ process of build creation: continuous development.

→ process of installing the build into testing environment: continuous deployment.

→ process of checking the integration issues between old feature and new features: continuous integration.

→ process of delivering the tested the build to the production environment: continuous delivery.

→ Why Jenkins is required in development?

→ continuous development: continuous monitor the git source code repository & create a new build if any changes happened in the git source code.

→ continuous deployment: get the latest build from git location & deploy the build into testing environment ~~UAT environment (or) production environment.~~

→ continuous delivery: get the latest build from git location & deploy the build into UAT environment (or) production environment.

→ Why Jenkins is required in Automation?

→ continuous Integration: continuous execution of the selenium test scripts in testing environment to check the integration issues.

→ Automation Engineers used Jenkins for continuous integration.

→ continuous Integration means checking the integration issues between the old build and the new feature by executing the old framework. If the test scripts get failed, then with the failure we will get to know the impact of new feature on the old build. Hence, we can analyze the failure, debug the failed test scripts, if any product issues / bugs are found we will raise the defect in Jira, if it's a test script issue we will collect it update the framework and re run the framework.

⇒ OOPs concepts in my Framework:

→ **Encapsulation:** Binding properties and behaviour of an object in a programming structure (or) class is called as Encapsulation.

→ **Rules to achieve fully encapsulated class:**

- → class must be declared with public keyword.
- → There must be at least one constructor in the class and must be declared with public keyword.
- → All the class members must be declared with private keyword.
- → We need to use getters() and setters() to access and initialize value in another class.

→ **Encapsulation in my Framework:**

- → I have used the Encapsulation in my framework in the PageObjectModel class.
- → In the pom class at first we will find the location of the virtual object by findBy annotations and after finding the location Virtual object becomes web element and we declare the web elements initially with private keyword.
- → In the pom class we use the constructor, and whenever we create the object of the pom class of particular webpage in our main test script then the constructor of that particular pom class will be executed in which we have PageFactory, initElement(driver, this)

from which 'this' keyword will hold the object address of current pom class and initializes to the driver.

→ Advantages of using pom class and Advantages of using Encapsulation rules:

→ Maintenance and Modification of all the web elements with respect to each and every webpage will be easy.

→ we can avoid StaleElementReferenceException (whenever we execute the constructor of pom class, the driver focusses exactly on the current webpage. by this we can avoid the exception).

→ we can restrict (or) limit the access scope of web elements of each and every webpage by using private keyword to the users (or) to a team.

→ we can achieve the code optimization by using business logics in POM class with respect to scripts.

→ Debugging is easy.

⇒ Polymorphism:

→ objects behaving differently based on the input and with same behaviour provides multiple outputs and it can happen during runtime (or) compile time is called as polymorphism.

a). compile time polymorphism: The method declaration binds with the method implementation during the compile time by the compiler based on the arguments is called as compile time polymorphism.

Example: Method overloading and constructor overloading.

Method overloading: Declaring multiple methods in a same class with same method name but with different arguments is called as method overloading.

→ I have used method overloading in my framework for the dropdowns handling to select the required

dropdown with the help of Select class.

→ public void handlingDropdown (WebElement ele, string visText)

```
{
  Select s= new Select(ele);
   s. SelectByVisibleText(visText);
}
public void handlingDropdown (WebElement ele,
                              int indexNo)
  {
    Select s = new Select(ele);
     s. SelectByIndex(indexNo);
  }
```

b). Run Time Polymorphism: The method declaration binds with the method implementation during the execution time by the jvm based on the object creation is called as Run Time polymorphism.

Method Overriding: Giving the new implementation from subclass to the superclass method is called as method overriding.

→ To achieve method overriding we have a criteria to follow:

- → we should follow Inheritance relationship.
- → Method name, parameters and return type of both super class and sub class method should be same.
- → visibility of subclass method should be equal (or) higher than the super class method.
- → To achieve method overriding, super class method should not be declared as final.

→ I have used method overriding in my framework where the all the abstract methods present in WebDriver & (interface) are given implementation in the RemoteWebDriver (implementation class) and further for their usage we have inherited all implemented methods to classes like ChromeDriver (class), FireFox Driver (class) etc.

→ I have used method overriding in my framework for launching the browser where driver object behaves differently with respect to particular class.

Example: ChromeDriver driver = new ChromeDriver();

WebDriver driver = new ChromeDriver();

→ I have used method overriding in my framework to get the screenshot of failed test script by providing the implementation to abstract method present in ITestListener interface in my implementation class.

→ I have used method overriding in my framework to re-run the failed test script multiple times with RetryAnalyzer.

→ Inheritance:

→ Acquiring the properties i.e., state and behaviour of a class from one class to another class is called as Inheritance.

→ I have used hierarchial inheritance in my framework to inherit the pre-similar conditions and post-similar which are same for the entire framework by storing all those pre-similar conditions and post-similar conditions with TestNG annotations in a Baseclass which need to be present in generic utility so that all team members can access it and through inheritance only test scripts will be executed from Test classes and rest all will be executed from Baseclass through inheritance.

Ex: Database connection, launching the browser, login to application and logout from the application closing the browser and closing database connection are stored with TestNG annotations and whenever we execute the main test scripts since, it is extended

to Baseclass, all the codes present in Baseclass and main test script will also be executed.

⇒ Abstraction:-

→ The process of hiding the complexity of the application and providing only the functionalities to the end user is called as abstraction.

→ To achieve Abstraction we need,

a). Abstract class (which cannot be 100% abstract but 0 to 100%).

b). Interface (partial and it consists of only abstract methods).

→ Disadvantages of Abstract class:-

→ We cannot achieve 100% abstraction but 0 to 100%

→ Multiple Inheritance is not possible.

→ We can create a constructor of abstract class but we cannot create the objects as abstract keyword present in abstract class restricts the object creation.

→ We cannot make abstract class as final (or) private but it must be always public.

→ We cannot create a object in Abstract class so no one can access data by creating the object.

→ Tight coupling is possible but where as loose coupling is not possible in Abstract class so we go for ~~Inh~~ interface.

→ I have used abstraction in my framework where all the abstract methods present in WebDriver interface were provided with new implementation in RemoteWebDriver class without creating the Object of RemoteWebDriver but we only create ChromeDriver object of chromedriver class. We do Upcasting.

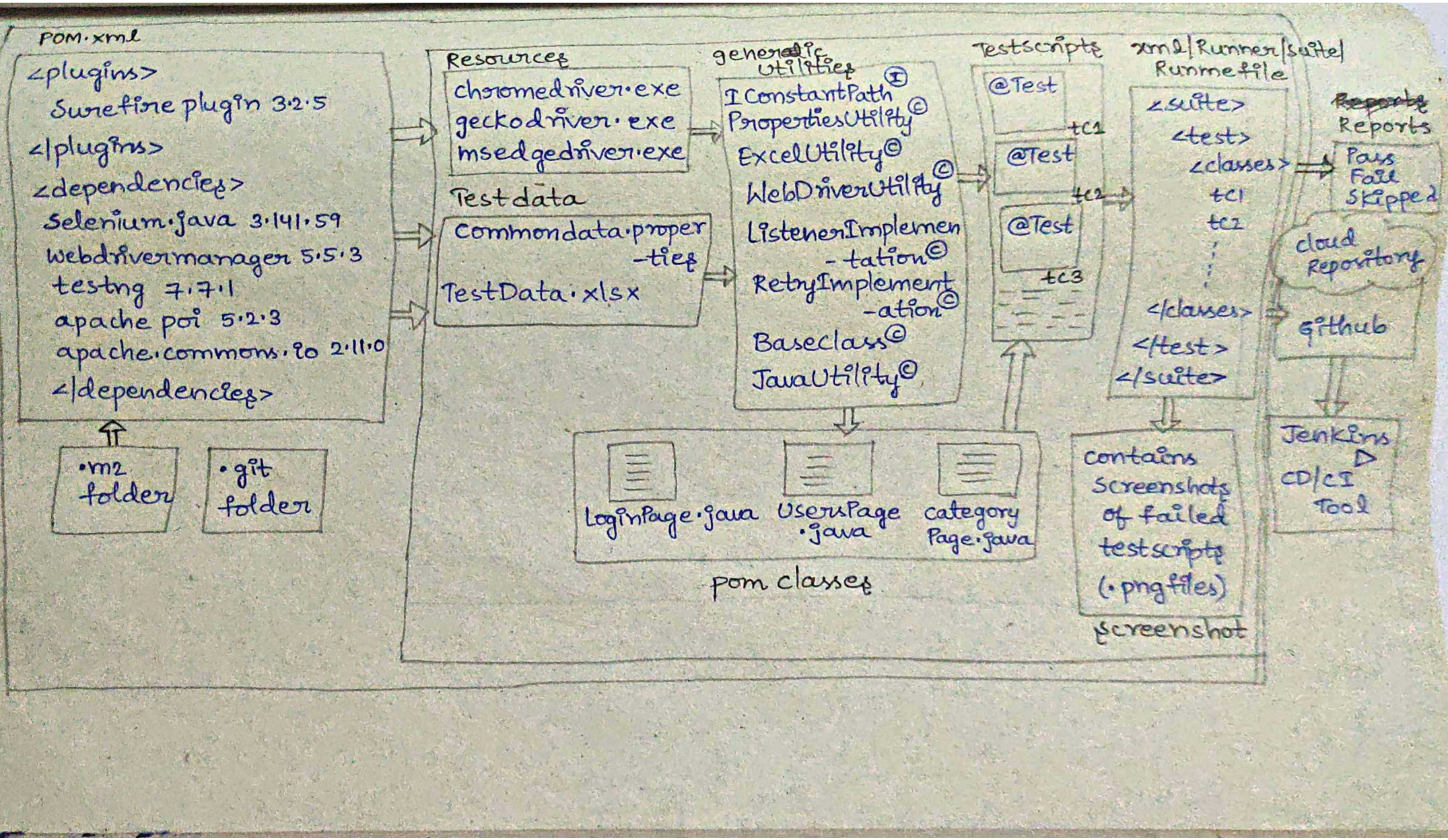
ChromeDriver driver = new ChromeDriver();
WebDriver driver = new ChromeDriver();

→ So, now all the abstract methods present in (driver) we are getting only the functionality to use in main test scripts but we exactly don't know where implementation is happening.

⇒ Framework Architecture:

⇒ Explain your framework?

Domain
Type of framework
client binding
Tools ~~are~~ used
Modules worked on
framework components
- → testscripts
- → generic utilities
- → pom classes
- → xml file
- → github
- → jenkins


POM.xml
<plugins>
Surefire plugin 3.2.5
</plugins>
<dependencies>
Selenium.java 3.141.59
webdrivermanager 5.5.3
testng 7.7.1
apache poi 5.2.3
apache.commons.io 2.11.0
</dependencies>
.m2 folder
.git folder
Resources
chromedriver.exe
geckodriver.exe
msedgedriver.exe
Test data
Commondata.properties
TestData.xlsx
generic Utilities
IConstantPath I
PropertiesUtility C
ExcelUtility C
WebDriverUtility C
ListenerImplementation C
RetryImplementation C
Baseclass C
JavaUtility C
Testscripts
@Test
tc1
@Test
tc2
@Test
tc3
xml/Runner/suite/Runnerfile
<suite>
<test>
<classes>
tc1
tc2
</classes>
</test>
</suite>
Reports
Pass
Fail
Skipped
cloud Repository
github
Jenkins
CD/CI Tool
LoginPage.java
UsersPage.java
category Page.java
pom classes
contains Screenshots of failed testscripts (.png files)
screenshot