**Cucumber framework set up Installation:**

1. Install Java
2. Install JDK
3. Install JRE
4. Install Eclipse
5. Install Cucumber plug-in and Natural Plug-in through eclipse market
6. Install Maven
7. Set Environment path

* Java Home
* Maven Home
* M2home

1. Add dependencies

* Selenium –java
* Cucumber –java
* Cucumber –junit
* Cucumber –jvm –deps
* Cucumber –reporting
* Cucumber –core
* Junit
* Gherkin
* Selenium-server-standalone(<https://www.selenium.dev/downloads/>)

Optional

* Coberta
* Mockito-all

**FOLDERS:**

Feature (Folder)

|

Drivers(Folder)

|

StepDefinitions

|

Pom.xml

|

Test Runner

**Coding explanation**  is used to run the Junit test

Create 'TestRunner' package and then 'Runner.java' class file under it.

package TestRunner;

import org.junit.runner.RunWith;

import cucumber.api.CucumberOptions;

import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)

@CucumberOptions(features="Features",glue={"StepDefinition"})

public class Runner

{

}

**@RunWith()** annotation - test runner class to start executing our tests.

**@CucmberOptions()** annotation - set some properties for our cucumber test like feature file, step definition, etc.

**Feature folder:** Feature Files where the test scripts are written in plain English.

Keywords:

Feature: High level business functionality

Scenario: What is the test scenario

Given: Prerequisite before the test steps get executed

When: Specific condition which should match in order to execute the next step.

Then: What should happen if the condition mentioned in WHEN is satisfied.

AND: keyword is used to show conjunction between two conditions

Scenario Outline: when the same test has to be performed with different data set

Examples: Examples are used to pass different arguments in the tabular format. Vertical pipes are used to separate two different columns. An example can contain many different columns

Background: Whenever any step is required to perform in each scenario then those steps need to be placed in Background

**Feature** − Scenario Outline

**Scenario Outline** − Login functionality for a social networking site.

Given user navigates to Facebook

When I enter Username as "<username>" and Password as "<password>"

Then login should be unsuccessful

**Example** −

| Username | password |

| username1 | password1 |

| username2 | password2 |

**CREATE POJO CLASS:**

**Public cass demo**

**{**

private String name;

public String getName() {

return name;

}

public void setName(String name) {

this= name;

}

**}**

**POM Using Pagefactory:**

**@FindBy** is used in Pagefactory to identify an element while POM without Pagefactory uses the **driver.findElement()** method to locate an element.

@FindBy(Name= "sname”)

WebElement txtbox;

public PagefactoryClass(WebDriver driver)

{

this.driver = driver;

PageFactory.initElements(driver, this);

}

static initElements() method of PageFactory class is used to initialize all the UI elements on the page as soon as the page loads.

**Step Definitions:**

Step Definitions java class files where code resides

Public class steps{

//Create object

Public Webdriver driver;

Public Loginpage lp;

@Given(“Login {string name} and {string password}”)

Public void login(string email, string password){

Lp=new Loginpage(driver);

Lp.setname(email);  
}