**1. Feature file**

A ***Feature File***is an entry point to the Cucumber tests. This is a file where you will describe your tests in Descriptive language (Like English). It is an essential part of Cucumber, as it serves as an automation test script as well as live documents. A feature file can contain a scenario or can contain many scenarios in a single feature file but it usually contains a list of scenarios.

**2. writing test cases in BDD format (given, when, and then)**

* The **given** part describes the state of the world before you begin the behavior you're specifying in this scenario. You can think of it as the pre-conditions to the test.
* The **when** section is that behavior that you're specifying.
* Finally the **then** section describes the changes you expect due to the specified behavior.

Scenario: I am a employee and his attitude

Given : An account with zero balance

When: I navigate to Credit card Payment Section and click on submit bu giving amount

Then : It should throw error message ñfund

Given : An account with sufficent balance who does not have credit card

When: I navigate to Credit card Payment Section

Then :You don't have to access as your are error message

Given : An account with sufficent balance who does not have credit card

When: I navigate to Credit card Payment Section and amount

Then :You don't have to access /error message

**3. scenario**

**Scenario** is one of the core Gherkin structures. Every scenario starts with the keyword “Scenario:” (or localized one) and is followed by an optional scenario title. Each feature can have one or more scenarios and every scenario consists of one or more steps

**4. Scenario outline**

Copying and pasting scenarios to use different values quickly  
becomes tedious and repetitive. Scenario outlines allow us to more  
concisely express these examples through the use of a template with  
placeholders, using Scenario Outline, Examples with tables and < >  
delimited parameters.

**|User Name| Password|**

**|User 1 | Password1|**

**|User 2 | Password2|**

Feature: Attitude Matters

In order to understand Attitude

As a It working guy

I Want to know whom to wishing

@tester

Scenario Outline: Employee and his attitude

Given I work in late nights

When I meet watchman

Then I greet him

Given I work in mornings

When I meet newspaper

Then I skipGreeting him

Given we work in "<weekday>"

When we meet "<guy>"

Then I skipGreeting him

Examples:

|weekday | guy |

|tenpm | watchmen |

|nineam | cabdriver|

**5. back ground**

Background allows you to add some context to the scenarios in a single feature. A Background is much like a scenario containing a number of steps. The difference is when it is run. The background is run before each of your scenarios but after any of your Before [Hooks](https://github.com/cucumber/cucumber/wiki/Hooks).

Feature: Attitude Matters

In order to understand Attitude

As a It working guy

I Want to know whom to wishing

@tester

Scenario: Employee and his attitude

Given I work in late nights

When I meet watchman

Then I greet him

Given I work in mornings

When I meet newspaper

Then I skipGreeting him

**6. create step definition file**

New > Other > Cucumber > Step-Definition class > Next > : - Specify: Source Folder: [browse your project directory] Package Name:[browse package name] Class Name:[Step-Def(java) file name] - Select any/all : Cucumber Annotations : Given/When/Then/And/But - Result : A sample Step definition(java) file is created for selected Annotations

**7. tags**

cucumber --tags @billing # Runs both scenarios

cucumber --tags @important # Runs the first scenario

cucumber --tags ~@important # Runs the second scenario (Scenarios without @important)

cucumber --tags @billing --tags @important # Runs the first scenario (Scenarios with @important AND @billing)

cucumber --tags @billing,@important # Runs both scenarios (Scenarios with @important OR @billing)

**8. web hooks**

Cucumber **hook** allows us to better manage the code workflow and helps us to reduce the code redundancy. We can say that it is an unseen step, which allows us to perform our scenarios or tests.

package hookTest;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import cucumber.annotation.en.Given;

import cucumber.annotation.en.Then;

import cucumber.annotation.en.When;

public class hookTest {

WebDriver driver = null;

@Before public void setUp(){

driver = new FirefoxDriver();

}

@Given("^user navigates to facebook$")

public void goToFacebook() {

driver.navigate().to("https://www.facebook.com/");

}

@When("^I enter Username as \"([^\"]\*)\" and Password as \"([^\"]\*)\"$")

public void I\_enter\_Username\_as\_and\_Password\_as(String arg1, String arg2) {

driver.findElement(By.id("email")).sendKeys(arg1);

driver.findElement(By.id("pass")).sendKeys(arg2);

driver.findElement(By.id("u\_0\_v")).click();

}

@Then("^login should be unsuccessful$")

public void validateRelogin() {

if(driver.getCurrentUrl().equalsIgnoreCase(

"https://www.facebook.com/login.php?login\_attempt=1&lwv=110")){

System.out.println("Test Pass");

} else {

System.out.println("Test Failed");

}

driver.close();

}

@After public void cleanUp(){

driver.close();

}

}

**9. how to connect feature and step definition file**

Feature: Attitude Matters

In order to understand Attitude

As a It working guy

I Want to know whom to wishing

@tester

Scenario Outline: Employee and his attitude

Given I work in late nights

When I meet watchman

Then I greet him

Given I work in mornings

When I meet newspaper

Then I skipGreeting him

Given we work in "<weekday>"

When we meet "<guy>"

Then I skipGreeting him

Examples:

|weekday | guy |

|tenpm | watchmen |

|nineam | cabdriver|

**Step definition File**

**package** practicecucu;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** cucumber.api.java.en.Given;

**import** cucumber.api.java.en.Then;

**import** cucumber.api.java.en.When;

**public** **class** Attitudetests {

@Given("^I work in late nights$")

**public** **void** i\_work\_in\_late\_nights() **throws** Throwable {

}

@When("^I meet watchman$")

**public** **void** i\_meet\_watchman() **throws** Throwable {

}

@Then("^I greet him$")

**public** **void** i\_greet\_him() **throws** Throwable {

}

@Given("^I work in mornings$")

**public** **void** i\_work\_in\_mornings() **throws** Throwable {

}

@When("^I meet newspaper$")

**public** **void** i\_meet\_newspaper() **throws** Throwable {

}

@Then("^I skipGreeting him$")

**public** **void** i\_skipGreeting\_him() **throws** Throwable {

}

@Given("^we work in \"([^\"]\*)\"$")

**public** **void** we\_work\_in(String arg1) **throws** Throwable {

}

@When("^we meet \"([^\"]\*)\"$")

**public** **void** we\_meet(String arg1) **throws** Throwable {

}

}

**11. generate reports**

Step 1 − Create a Maven project named cucumberReport in Eclipse.

Step 2 − Create a package named CucumberReport under src/test/java

Step 3 − Create a feature file named cucumberReport.feature

Step 4 − Create a step definition file.

Step 5 − Create a runner class file.

Select runTest.java file from package explorer.

Right-click and select the option, Run as.

Select JUnit test.