

Java Selenium – Cucumber Framework Notes

1. Test Driven Development (TDD) vs Behavior Driven Development (BDD)

Aspect	TDD	BDD
Definition	Development approach where tests are written before code implementation.	Extension of TDD focusing on business behavior, using natural language.
Focus	Tests technical correctness of code.	Tests business requirements and user behavior.
Language Used	Programming language (Java, Python, etc.).	Gherkin syntax (plain English).
Stakeholders	Developers & Testers.	Developers, Testers, Business Analysts, Product Owners.
Tools	JUnit, TestNG, NUnit, etc.	Cucumber, SpecFlow, Behave, JBehave, etc.
Example	Write unit test → Write code to pass test → Refactor.	Write feature file in Gherkin → Implement step definitions → Execute.

2. BDD Tools

- **Cucumber** → Most popular BDD tool, supports Java, Ruby, etc.
 - **SpecFlow** → .NET-based BDD framework.
 - **JBehave** → Java-based BDD framework.
 - **Behave** → Python-based BDD tool.
-

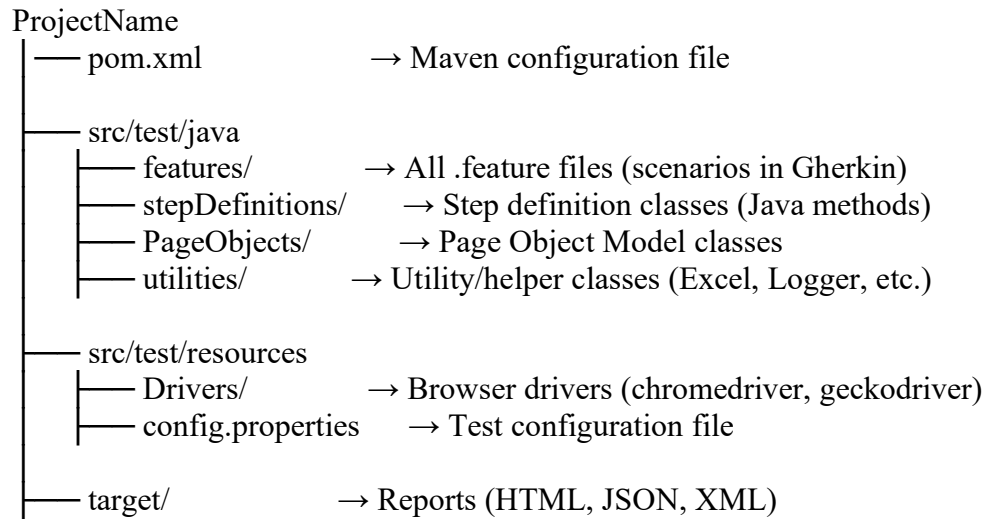
3. Cucumber Introduction

- Open-source tool supporting **BDD (Behavior Driven Development)**.
- Allows writing test scenarios in **Gherkin language** (plain English).
- Bridges the gap between **Business (non-technical) and Development/QA teams**.
- Cucumber integrates with **Selenium WebDriver** to automate UI testing.

Key Components:

- **Feature File** → Contains scenarios written in Gherkin.
 - **Step Definition** → Java methods mapping Gherkin steps.
 - **Runner Class** → Executes feature files using JUnit/TestNG.
-

4. Maven Project Folder Structure for Cucumber Framework



5. Cucumber Feature File

- Written in **Gherkin syntax**.
- File extension: .feature.

Example:

Feature: Login functionality

Scenario: Successful login with valid credentials

Given User is on the login page

When User enters valid username and password

And Clicks on login button

Then User should be navigated to the homepage

6. Step Definition File

- Java methods that implement the steps defined in feature files.
- Each step is linked using **annotations**: @Given, @When, @Then.

Example:

```
public class LoginSteps {  
  
    WebDriver driver;  
  
    @Given("User is on the login page")
```

```

public void user_is_on_login_page() {
    driver = new ChromeDriver();
    driver.get("https://example.com/login");
}

@When("User enters valid username and password")
public void user_enters_credentials() {
    driver.findElement(By.id("username")).sendKeys("admin");
    driver.findElement(By.id("password")).sendKeys("admin123");
}

@When("Clicks on login button")
public void click_login_button() {
    driver.findElement(By.id("loginBtn")).click();
}

@Then("User should be navigated to the homepage")
public void verify_homepage() {
    Assert.assertTrue(driver.getTitle().contains("Home"));
}
}

```

7. JUnit Test Runner Class File

- Used to configure and run cucumber tests.
- Defines location of feature files and step definitions.

Example:

```

import org.junit.runner.RunWith;
import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)
@CucumberOptions(
    features = "src/test/java/features",
    glue = {"stepDefinitions"},
    plugin = {"pretty", "html:target/CucumberReport.html", "json:target/cucumber.json"},
    monochrome = true,
    dryRun = false
)
public class TestRunner {
}

```

8. Gherkin Keywords

- **Feature** → Describes the functionality being tested.
- **Scenario** → Represents a test case.
- **Given** → Describes the initial context (preconditions).
- **When** → Describes an action performed by the user.
- **Then** → Describes the expected outcome.
- **And** → Used to combine multiple Given/When/Then.
- **Background** → Common precondition steps for all scenarios in a feature.
- **Scenario Outline** → Used for data-driven testing.
- **Examples** → Provides test data for Scenario Outline.

Example with Background & Scenario Outline:

Feature: Login functionality

Background:

Given User is on the login page

Scenario Outline: Login with multiple users

When User enters "<username>" and "<password>"

And Clicks on login button

Then User should see "<result>"

Examples:

username	password	result	
admin	admin123	Homepage displayed	
user1	pass123	Invalid login	