**Q. What is cucumber?**

Cucumber is a testing tool that supports Behavior-Driven-Development (BDD).

 In Behavior Driven Development, the tests are designed in plain English and are created from a more end-user perspective and describe the application characteristics.

Behavior Driven Development framework increases collaboration and coordination among members in an agile team of developers, testers, business analysts, product owners, customers, and other stakeholders. It does not require any technical knowledge and is written in plain text and requires no technical clarity. **Behavior Driven Development framework like Cucumber helps to bridge any understanding gap between business stakeholders and the developers.**

Cucumber reads executable specifications written in plain text and validates that the software does what those specifications say. The specifications consist of multiple examples, or scenarios.

In order for Cucumber to understand the scenarios, they must follow some basic syntax rules, called [Gherkin](https://cucumber.io/docs/gherkin/).

**Advantage:**

* Cucumber is an open-source and free-to-use tool.
* Cucumber tests are written in plain-text English called Gherkin, so people with no or less technical skills can also write scenarios.
* It allows us to involve business stakeholders who can’t easily read a code
* High reusability of code in the tests
* It can be integrated with Selenium and other testing frameworks like JUnit & TestNG
* End-user experience is a priority.

**Q. What is Gherkin?**

**Gherkin is a business readable language which helps you to describe business behavior without going into details of implementation. Gherkin documents are stored in .feature text files.**

**Gherkin Syntax:** Gherkin is line-oriented language just like YAML and Python. Each line called step and starts with keyword and end of the terminals with a stop. Tab or space are used for the indentation.

**Q. What is step definition?**

A Step Definition is a Java method with an [expression](https://cucumber.io/docs/cucumber/step-definitions/#expressions) that links it to one or more [Gherkin steps](https://cucumber.io/docs/gherkin/reference#steps). When Cucumber executes a [Gherkin step](https://cucumber.io/docs/gherkin/reference#steps) in a scenario, it will look for a matching step definition to execute. **A step definition is the actual code implementation of the feature mentioned in the feature file.**

**Q. What is snippet?**

When Cucumber encounters a [Gherkin step](https://cucumber.io/docs/gherkin/reference#steps) without a matching step definition, it will print a step definition snippet with a matching [Cucumber Expression](https://cucumber.io/docs/cucumber/cucumber-expressions). For missing step definition for corresponding feature file.

Consider this Gherkin step:

Given I have 3 red balls

If you don’t have a matching step definition, Cucumber will suggest the following snippet:

@Given("I have {int} red balls")

public void i\_have\_red\_balls(int int1) {

}

Suggested snippets will use your own [parameter types](https://cucumber.io/docs/cucumber/cucumber-expressions#parameter-types) if they match parts of your undefined step. If a [color](https://cucumber.io/docs/cucumber/cucumber-expressions#custom-parameter-types) parameter type exists, Cucumber would use that in the suggested expression:

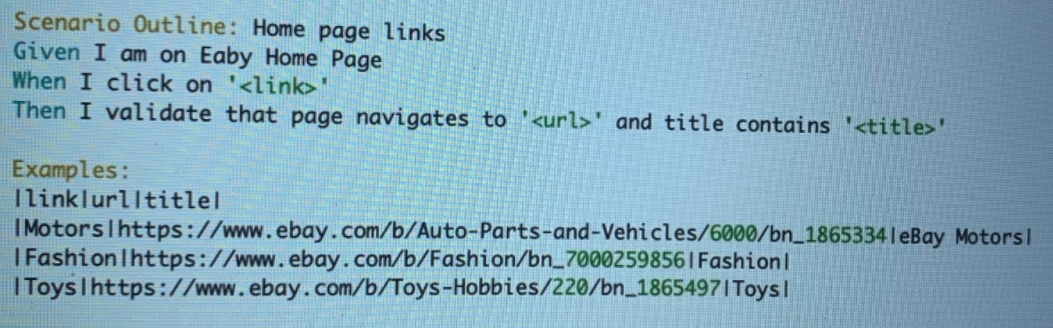
@Given("I have {int} {color} balls")

public void i\_have\_color\_balls(int int1, Color color) {

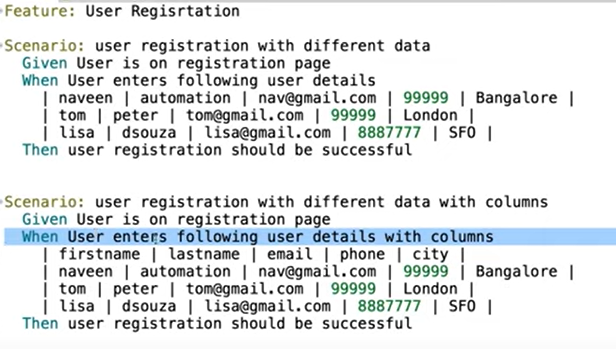
}

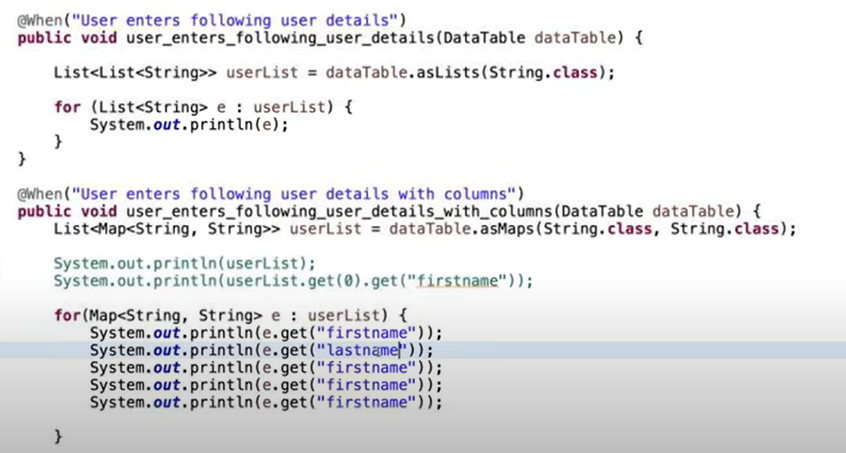
}

NOTE: Step definitions aren’t linked to a particular feature file or scenario. The file, class or package name of a step definition does not affect what Gherkin steps it will match. The only thing that matters is the step definition’s expression.

**SCENARIO OUTLINE: When particular scenario is run with more than one dataset in multiple combinations. It is like DATADRIVEN. **

**DATA TABLE:**

****

****

**🡺HOOK: Hooks are blocks of code that can run at various points in the Cucumber execution cycle. They are typically used for setup and teardown of the environment before and after each scenario. You can declare hooks in any class.** **Cucumber hooks** are blocks of code that runs before or after each scenario. It can be defined anywhere in project or step definition layers using methods @Before, @After.

**1. GLOBAL HOOKS will run once before any scenario is run or after all scenario have been run.**

@Before All run before any scenario is run and @ AfterAll run after all scenarios have been executed.

**2. SCENARIO HOOKS runs for every scenario.**

Before hooks run before the first step of each scenario.

After hooks run after the last step of each scenario, even when the step result is failed, undefined, pending, or skipped.

**3. STEP HOOKS Step hooks invoked before and after a step.  @BeforeStep @AfterStep**

**4. CONDITIONAL HOOKS with tags for more specifically execution.**

Associate a Before or After hook with a [tag expression](https://cucumber.io/docs/cucumber/api/#tag-expressions).

@After("@browser and not @headless")

**5. Ordering Prioritizing Hooks: @Before(order=0)**

**NOTE: An important thing to note about the after hook is that even in case of test fail, after hook will execute for sure. \*Make sure that the package import statement should be import cucumber.api.java.After; & import cucumber.api.java.Before;**

**🡺 TAGS are more effective way to organize your features and scenario. We can use tags for running subset of scenarios and restriction hooks to a subset of a scenario.**

**It is not possible to place tags above Background or steps (Given, When, Then, And and But).**

## **🡺 TAG INHERITANCE: Tags are inherited by child elements.**

Tags that are placed above a Feature will be inherited by Scenario, Scenario Outline, or Examples.

Tags that are placed above a Scenario Outline will be inherited by Examples.

**🡺PARALLEL EXECUTION:**

**By default, Cucumber runs tests sequentially in a single thread. Running tests in parallel is available as an opt-in feature. To enable parallel execution, set the cucumber.execution.parallel.enabled configuration parameter to true, e.g. in junit-platform.properties.**

**Cucumber supports JUnit's ParallelExecutionConfigurationStrategy; see the configuration options below.**

**Q. What is Scenario?**

Scenario **includes all the possible circumstances of the feature and test scripts for** these circumstances. The keyword "Scenario" represents a scenario in Gherkin language. One feature can have multiple scenarios, and each scenario consists of one or more steps.

### **Q. In which language is Cucumber software written?**

Cucumber software is written in a Ruby programming language.

### **Q. What language is used by the Cucumber tool?**

The Cucumber tool uses the Gherkin language, a simple English representation of the application behavior. The Gherkin language uses several keywords to describe the behavior of applications such as Feature, Scenario, Scenario Outline, Given, When, Then, etc.

### **Q. What are the two files required to execute a Cucumber test scenario?**

Following are the two files required to execute a Cucumber test scenario:

* Features
* Step Definition

**Q. What symbol is used for parameterization in Cucumber?**

**Answer:** Pipe symbol (|) is used to specify one or more parameter values in a feature file.

### **Q. What do you understand by a feature file?**

**A feature file is used to provide a high-level description of an Application Under Test (AUT).** The first line of the feature file must start with the keyword 'Feature' followed by the description of the application under test. A feature file may include multiple scenarios within the same file, and the extension of the feature file must be ".feature."

### **Q. What are the files needed in a Cucumber framework?**

**Feature file –** The feature file has an extension of .feature. It has single or multiple test scenarios described in plain text. The tests are written with keywords like Then, When, Background, Scenario Outline, Feature, And, But, and so on. Thus, it is a file that maintains the features and their descriptions.

**Step Definition file –** This file has an extension of .java. It basically provides the mapping of the test scenario steps described in the feature file to the automation code. Thus, when Cucumber runs a step described in the feature file, it searches the step definition file and executes the relevant functions that are mapped to that step.

**TestRunner file –** This file has an extension of .java. **It links the step definition file and the feature file.** It gives the user the option to execute one or more than one feature file. It has the path of the step definition file and the feature file.

### **Q. What are the different keywords used in feature file?**

Answer: The different keywords used in the feature file are

* Feature
* Background
* Scenario
* Scenario Outline
* Given
* When
* Then
* And
* But

### **Q. How to achieve parameterization in the Cucumber framework?**

We can achieve parameterization in Cucumber. This helps to pass multiple data sets at runtime in multiple combinations. We can perform data parameterization in the following ways:

* With the help of keyword Examples.
* Without the help of keyword Examples.

### **Q. What do you understand by the TestRunner class in the Cucumber testing approach? Explain with example.**

In the Cucumber testing approach, the TestRunner class provides the link between the feature file and the step definition file. **The TestRunner class is generally an empty class with no class definition.**

### **Q. Should we write code within the TestRunner class?**

It is advised not to write code under the TestRunner class. It should include the tags @RunWith and @CucumberOptions.

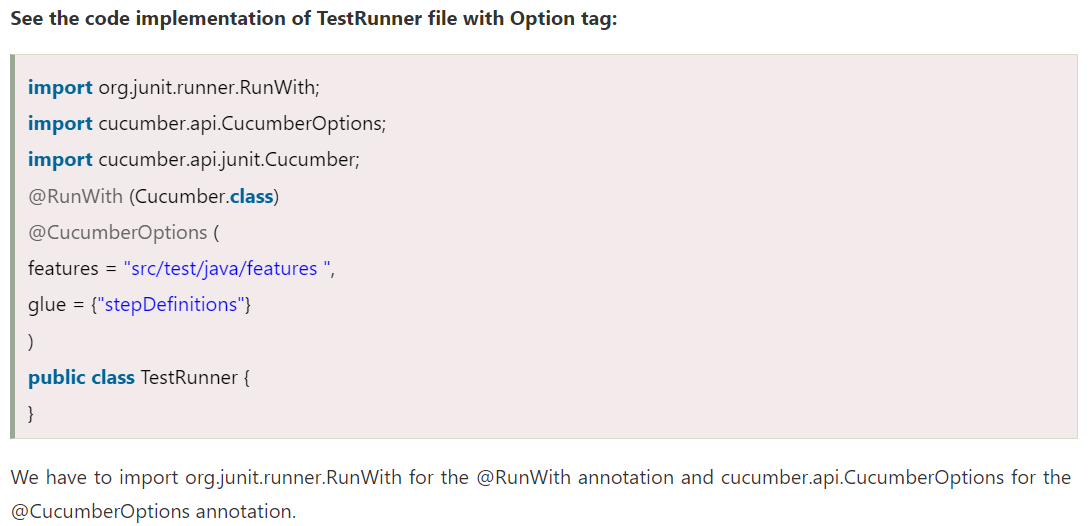
### **Q. What is the starting point of execution for feature files?**

When Cucumber is integrated with Selenium, the starting point of execution must be from the TestRunner class.

### **Q. How can you use the Options tag in the Cucumber framework?**

In the Cucumber framework, the Options tag is a part of the TestRunner file and comes in the form of an annotation called @CucumberOptions. It contains two parameters feature and glue.

* **Feature parameter:** The feature parameter is used to specify the path of the feature file.
* **Glue parameter:** The glue parameter is used to specify the path of the step definition file.



### **Q. What is the use of features property under the Cucumber Options tag?**

In the Cucumber framework, the features property is used to identify the location of the feature files.

### **Q. What is the use of glue property under the Cucumber Options tag?**

The Glue property is used to facilitate the Cucumber framework to identify the location of step definition files.

### **Q. What are the two build management tools that can be integrated with Cucumber?**

Following are the two build management tools that can be integrated with Cucumber:

* Gradle
* Maven

**Q. How can Cucumber be integrated with Selenium WebDriver?**

**Answer:** [Cucumber can be integrated with the Selenium Webdriver](https://www.softwaretestinghelp.com/selenium-webdriver-cucumber-selenium-tutorial-31/) by downloading the necessary JAR files.

**Given below are the list of JAR files that are to be downloaded for using Cucumber with Selenium web driver:**

* cucumber-core-1.2.2.jar
* cucumber-java-1.2.2.jar
* cucumber-junit-1.2.2.jar
* cucumber-jvm-deps-1.0.3.jar
* cucumber-reporting-0.1.0.jar
* gherkin-2.12.2.jar

**Q. When is Cucumber used in real-time?**

Cucumber tool is generally used in real-time to write acceptance tests for an application. It is generally used by non-technical people such as Business Analysts, Functional Testers, etc.

**Q. What is the name of the plugin that is used to integrate Eclipse with Cucumber?**

Cucumber Natural Plugin is the plugin that is used to integrate Eclipse with Cucumber.

**Q. What is the maximum number of steps that are to be written within a scenario?**

**Answer:** 3-4 steps.

### **Q. What are the differences between Jbehave and Cucumber?**

Jbehave is based on stories while Cucumber is based on features.

### **Q. What is the difference between Selenium and Cucumber?**

Selenium and Cucumber are both open-source testing tools, and both are used for functional testing. But there are some differences between them.

Following are some critical differences between Selenium and Cucumber:

* Selenium is used for only and only for browser automation i.e., web application (url based applications)., while Cucumber is an testing tool that support Behavior-Driven Development that can be used with Selenium (or Appium).
* Selenium is used for automated UI testing, while Cucumber is used for acceptance testing.
* Selenium is preferred by technical teams (SDETs/programmers), while Cucumber is typically preferred by non-technical teams (business stakeholders and testers).
* Selenium can work independently of Cucumber. Cucumber depends on Selenium or Appium for step-definition implementation.
* In Selenium, the script creation is complex, while Cucumber is simpler than Selenium.

### **Q. Why we have to use Cucumber with Selenium?**

Cucumber and Selenium are both testing frameworks and prevalent technologies. Many organizations use Selenium for functional testing. Along with Selenium, these organizations integrate Cucumber with Selenium as Cucumber makes it easy to read and understand the application flow. The most significant benefit of using Cucumber with Selenium is that it facilitates developers to write test cases in simple feature files easily understood by managers, non-technical stakeholders, and business analysts. It provides the facility to write tests in a human-readable language called Gherkin. The Selenium-Cucumber framework supports programming languages such as Java, .NET, PHP, Python, Perl, etc.

### **Q. What are the maximum numbers of scenarios that we can include in the feature file?**

In Cucumber, a feature file can contain a maximum of 10 scenarios. This number can vary from project to project and from one organization to another organization. It is the best practice to limit the number of scenarios included in the feature file.

**Q. Conclusion:**

* BDD is a methodology to understand the functionality of an application in the simple plain text representation.
* Cucumber is a tool that uses Behavior Driven Development to write acceptance tests of an application. It is used to bridge the communication gap between various project stakeholders.
* The main use of Cucumber lies in its simplicity to understand and usage of feature files by non-technical users.

