*Top 30 Cucumber Interview Questions & Answers*

***1. Full form of BDD***

BDD stands for Behavior Driven Development

**2.*Full form of TDD***

TDD stands for Test Driven Development

***3. What is the difference between BDD and TDD?***

| **BDD** | **TDD** |
| --- | --- |
| Behavior centered development process | Test centered development process |
| BDD tests are written in readable format using Given-When-Then steps | TDD tests are written using programming languages like Ruby, JAVA etc |
| BDD tests are readable by non-programmers | TDD tests are difficult to read by non-programmers |

***4. Name any 3 popular BDD testing tools***

* Cucumber
* JBehave
* Specflow

***5. What is the difference between cucumber, JBehave, and Specflow?***

* Cucumber is a Ruby based framework
* JBehave is a JAVA based framework
* Specflow is a .NET based framework

***6. What is Cucumber?***

Cucumber is a testing framework written in BDD style. Cucumber lets us define the system behavior from user’s perspective in plain English text called Gherkin. Cucumber by itself is written in Ruby programming language but it can be used to test the code written in other programming languages like JAVA, .NET, Groovy etc.

***7. What are the advantages of Cucumber?***

* Cucumber acceptance tests are written from user’s perspective
* Team Collaboration (Product owners, business analysts, developers, testers are all involved in writing scenarios)
* Code reusability
* Tests are written in plain-text English so people with less technical skills can also write scenarios
* Cucumber can be integrated with Selenium and other testing frameworks like JUnit & TestNG

***8. What are the two files required to run a cucumber test?***

* Feature file
* Step Definition file

***9. What is a feature file?***

Feature file is an entry point of Cucumber execution. Cucumber scenarios are written inside the feature file using a plain-text language called Gherkin. It can contain a scenario or multiple scenarios. All the acceptance tests are written in feature file using Given-When-Then statements.

***10. What is the file extension of feature file?***

.feature is the extension of feature file. Example, “Test.feature” is a feature file name.

***11. What is the language used to write a scenario in feature file?***

Gherkin is a plain-text language used to write cucumber scenario in the feature file. Gherkin is a business readable, domain-specific language used to describe the behavior of software application from user’s perspective so that its easy for non-programmers to read and collaborate.

***12. What is the two main purpose of using Gherkin?***

* Documentation
* Automated tests

***13. What are the keywords used in Feature file?***

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

***14. Is it mandatory to use Given, When, Then keywords while writing scenario?***

No.  \*  can also be used to write steps in the feature file.

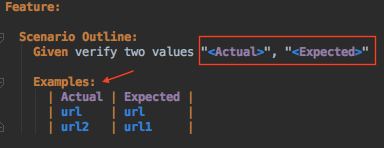
|  |  |
| --- | --- |
| 1  2  3  4  5  6 | Feature: Account Balance      Scenario: Verify Positive Balance      \* I have $100 in my account      \* I withdraw $50      \* I should have $50 balance |

***15. What is the difference between Given, When, Then steps in feature file?***

* Given defines the context of the scenario
* When defines the actions of the scenario
* Then defines the outcome of the scenario

***16. Explain Scenario Outline in feature file***

Scenario Outline keyword in feature file is used to execute scenarios multiple times using a different set of test data. Multiple sets of test data are provided by using ‘Examples’ in a tabular structure separated by pipes (| |)

*Example Scenario Outline below:* [[](http://letzdotesting.com/wp-content/uploads/2018/01/ScenarioOutline.png)](http://letzdotesting.com/wp-content/uploads/2018/01/ScenarioOutline.png)

***17. Explain background in feature file***

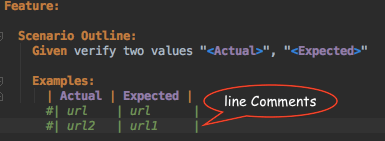
Steps written under the Background keyword are executed before every scenario.

For example: If you want to execute the same steps for every scenario like login to the website, you just write those common steps under the background keyword. While executing every scenario, steps written under background will be executed first.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | Feature: Account Balance      Background:      Given I login to the account      Scenario: Verify Positive Balance      Given I have $100 in my account      When I withdraw $50      Then I should have $50 balance      Scenario: Verify Zero Balance      Given I have $100 in my account      When I withdraw $100      Then I should have $0 balance |

***18. How to comment a line in Feature file?***

# is used to comment a line in feature file

[](https://i1.wp.com/letzdotesting.com/wp-content/uploads/2018/01/Comment.png)

***19. What is a Step definition file?***

Step definitions file has the actual code implementation of the scenario step.

*Feature file*

|  |  |
| --- | --- |
| 1  2  3  4 | Feature:      Scenario:      Given verify two values 'text', 'test' |

*Step definition file*

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | public class Test {  *@Given*("verify two values '(.\*)', '(.\*)'")      public void verify\_two\_values(String arg1, String arg2)  {         Assert.assertEquals(arg1,arg2);      }  } |

***20. Write an example BDD test scenario***

|  |  |
| --- | --- |
| 1  2  3  4  5 | Feature: Account Balance    Scenario: Verify Balance      Given I have $100 in my account      When I withdraw $50      Then I should have $50 balance |

***21. Explain Cucumber Tags***

Cucumber tags are used to organize scenarios in your feature file. You can have as many tags as you like before a scenario or feature. @ is used to represent tags. Example: @regression, @sprint5, @EndtoEnd

Tags are used to

* Group scenarios
* Ignore scenarios from execution

Logically group (OR & AND)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | Feature: Account Balance    *@regression* *@sprint2*    Scenario: Verify Zero Balance      Given I have $100 in my account      When I withdraw $100      Then I should have $0 balance |

***22. Cucumber Tags are case sensitive. True or False?****True*

***23. Name any two testing framework that can be integrated with Cucumber?***

* JUnit
* TestNG

***24. Name any two build management tools that can be integrated with Cucumber?***

* Gradle
* Maven

***25. What software do you need to run cucumber in JAVA?***

* Eclipse or IntelliJ IDE
* Gradle or Maven build tool
* Junit or TestNG testing framework
* Cucumber
* Selenium (To automate browser)

***26. How does a JUnit Test Runner class look like?***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | import cucumber.api.CucumberOptions;  import cucumber.api.junit.Cucumber;  import org.junit.runner.RunWith;    *@RunWith*(Cucumber.class)  *@CucumberOptions*(          features="src/test/resources/features",          glue= "ca.testing.stepdefinitions")    public class TestRunner{  } |

***27***.***What is @CucumberOptions in test runner? List the properties of @CucumberOptions***

@CucumberOptions are used to set specific properties for your cucumber test.

Properties are,

* Feature – path to feature file
* Glue – path to step definition
* dryRun – boolean value – check for missing step definition
* tags – used to group cucumber scenarios in the feature file
* strict – boolean value – fail the execution if there is a missing step

monochrome – boolean value – display console output in a readable way

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | import cucumber.api.CucumberOptions;  import cucumber.api.junit.Cucumber;  import org.junit.runner.RunWith;    *@RunWith*(Cucumber.class)  *@CucumberOptions*(          features="src/test/resources/features",          glue= "ca.testing.stepdefinitions",          tags = *@regression*,          dryRun = false,          strict = true,          monochrome = true)    public class Runfeatures {  } |

***28. Name any advanced framework design that can be used with Cucumber?***

* Page Object Model
* Log4j
* Extent Reporting
* Dependency Injection (Example: Pico Container)
* Object Repository

***29. Selenium can be integrated with Cucumber. True or False?****True*

***30. Explain Cucumber Hooks***

Cucumber Hooks are blocks of code that can be used to run before and after the scenarios using @before and @after methods. It helps us eliminates the redundant code steps that we write for every scenario and also manages our code workflow.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | public class Hooks {    *@Before*      public void before(){          System.out.println("This will execute before every Scenario");      }  *@After*      public void after(){          System.out.println("This will execute after every Scenario");      }  } |

***Note: @After will be executed even though the scenario is failed. @Before will be executed before ‘Background’ steps in the feature file.***