TDD
It is for UNIT Tests by Developers
-TEST => Write a test that fails
-CODE => Write just enough code to make the test fails
-REFACTOR => Refactor the code, fixing, avoiding duplications etc

— BDD advantages
Examples:
—> a method = test name “public void testSearchCustomer(){}” NOT
—> a understandable method “public void testFindCustomerByName(){}”
—> a understandable method “public void testFailsForCustomerEmptyFirstName(){}”
It is described BEHAVIORS NOT TESTS, KEYWORD==> USER SHOULD BE ABLE TO….

-Consistent vocabulary
-Eliminate ambiguity(belirsizlik)
-Eliminate miscommunication

So BDD is :
-Collaboration
-Build shared understanding
-Fast feedback
-Produce system documentation

Ubiquitous(her Verde bulunan) language==>Gherkin = fill the gaps between devs and non-technical people

Agile with BDD and TDD
- Both can be in Agile
-User Story ===> Discover<->Formulate<->Code. <—> Refactor -Test - Code

QUIZ-1
1)Which of the below is a Waterfall Model problem?
-Working in Silos(depo, ambarlama)
-Sequential process (sirali)
-Costly changes

2)Agile and Scrum are one and the same
-No

3)Which of the below frameworks is based on Agile principles
-Scrum
-Kanban
-Extreme Programming
-BDD (No -practice, not a framework)

4)BDD is a replacement of Agile/Scrum
-NO `BDD does not intend to replace your Agile process. It complements and makes it efficient.`

5)Test Driven Development (TDD) is —programming practice

6)A typical TDD process follows below order
Test—>Code—>Refactor

7)TDD deals with high level specifications while BDD deals with low level specifications
-TDD deals with **low level** specifications while BDD deals with **high level** specifications

8)BDD, TDD and Agile, all can happen together
YES

BDD PRACTICES
-Discovery —> Shared understanding of requirements
Formulation —> Writing clear, unambiguous scenarios like BDD Gherkin
Automation —> Automating scenarios using tools such as Cucumber, SpecFlow, Behave
Working Software —> Not a BDD practice:

QUIZ-2

1)A **feature file** can have multiple Feature keyword
No

2)A Feature can have more than one Background section
No

3)the steps in a Scenario are optional
Yes

4)A user story can be represented by one or more feature
False **A feature can be represented by one or more user stories and not vice-e-versa(tam tersi, tersine)**

5)Cucumber ignores the keyword (Given/When/Then) while matching a step with step definition
Yes true, step definition is important

6)Adding Examples is optional in a **Scenario Outline**
False- it is mandatory

7)A Rule can have it's own Background section
A rule can have one background section just like a feature

8)A Feature cannot have a Background section if a Rule under it has a Background section
A feature and a Rule , both can have on Background section each

9)It's ok to use Background to perform technical steps like WebDriver initialization or setting up DB or fetching a Token
No,Background section should be used to set the context or precondition that is common to multiple scenarios.These are essentially the steps a user would perform the system

QUIZ-3

1)Gherkin should be DECLARATIVE (beyan eden, bildiren)

2)What's the benefit of avoiding conjunction steps?
-We can reuse the steps

3)**It' s a good practice to have multiple intermediate "Then" steps in a single scenario as it helps validate many test cases**
FALSE

———————————————————-

Why is it important to write good Gherkin?

Gherkin serves as a Living documentation which is a single source of truth for all the stakeholders. Writing good Gherkin means a **readable** and **understandable** Living documentation. The Living documentation should clearly show the current state of the system, what behaviors are working, what are pending and what are not working. If we don't use **Gherkin to describe the behavior of the system**, the whole purpose of writing Gherkin gets defeated. Then it just becomes a UI action script written in plain language. Just an unnecessary wrapper on top of the underlying automation layers that can result into high maintenance.

What is the **best approach points for BDD test automation**
-Maintenance
-Reusability
-Readability
-Collaboration

*** RUNNERS:

1) **CLI io.cucumber.core.cli.Main**==>Maven exec==>scenarios in paralell
2) **Junit**: Runner class ==> supports annotated methods ==> feature in paralell
3) **TesNG**: Runner class ==> supports annotated methods ==> scenarios in paralell

Maven
Maven is needed to able to execute automation from **terminal**

QUIZ 4:

1)Feature file is located under this path: src/test/resources/awesomecucumber
Step definition file is located under this path: src/test/java/awesomecucumber/stepDef

If I right click on the Scenario and try to execute it, will it be able to find the feature file and step definition file?
-**YES** Cucumber can find both as long as i**t's invoked from a root directory that is common to both**. This **may not work in case of JUnit or TestNG runner though**

2)Which runner supports **parallel execution** at Example level
TestNG and **CLI**

3)JUnit supports parallel at **Scenario level**
-NO —> Junit supports parallel execution at **Feature level**

4)Select the false statement
-CLI runner runs “Cucumber @Before” as well as “JUnit @Before” -NO
CLI runs only Cucumber annotated methods.It can not see Junit annotated methods

5)Is it ok to have **different versions of cucumber-java and cucumber-junit dependencies?**
-NO, not recommended, **try to use same versions**

6)cucumber-junit dependency default scope is "test". Will it work in src/main/java path?
-NO, it works in “**test**” scope under the “test” file, not in java path
`If the scope is "test", the dependency will not resolve in src/main/java path. You can check this out by creating a simple class under src/main/java and using any JUnit annotation in it`

7)**@RunWith()** annotation is not needed in TestNG runner class
TRUE, it is for **jUnit**

8)**@BeforeClass** and **@AfterClass** annotations are from
Junit and TestNG

QUIZ 5:
1)What is the purpose of the `dryRun` option in Cucumber?
-To check the **mapping between feature and step definitions without executing the steps**

2)What does the `glue` option define in a Cucumber configuration?
-**To location of the step definitions**

3)How can you execute Cucumber scenarios with a **specific tag using Maven?**
- **mvn test -Dcucumber.filter.tags=“@tagname”**

4)What is the function of the `plugin` option in Cucumber?
- To specify the output format and reports should be generated and their location

5)Which of the following is a valid usage of the `plugin` option to**generate a JSON report**?
-@CucumberOptions(plugin = {“json:target/cucumber.json”})

6)What does setting `monochrome` to `true` in Cucumber do?
-Disabled colored outcome in the console

7)How do you **specify multiple tags** in the `tags` option to run scenarios that **match any of the tags?**
-@CucumberOptions(tags = "@Scenario1 or @Scenario2"

8)How do you specify the **path to step definitions** in the `glue` option?
-@CucumberOptions(glue = “com.example.steps”)
-@CucumberOptions(glue = {“com.example.steps”})
-@CucumberOptions(glue = {“com.example.steps”,”com.example.hooks”})

9)Which option in Cucumber allows you to ignore specific tags during execution?
—**@CucumberOptions(tags = “not@ignore”)**

QUIZ 6:
1)Below is an example of?

- `@When("I add a {product} to the cart”)`

-It is **cucumber expression**, if the expression is without special characters(start and end), it is cucumber expression

2)Below is an example of

- `@Given("I have {int} cucumber(s) in my belly")`

- This is an example of **optional text** where cucumber can be used in singular and plural form in Gherkin step

3)Below is an example of?

- `@Given("I'm on the Store/Product page”)`

-This is an example of **alternative text** where in we can use text Store or Product in Gherkin step and it will match the same step definition

4)Which of the below examples does Cucumber assume to be a Regular expression?
—regular expressions symbols

5)@ParameterType use Cucumber expressions to match the Gherkin parameter value
-False— regular expression is used (Custom Parameter)

QUIZ 7:

1)If a Data Table has 2 rows, the scenario will run 2 times
-NO

2)Data Table can be used only with "Given" step
-NO, Data Table can be used with any step

3)Data Table can be accessed as a last argument of the step definition
-TRUE

4)Cucumber supports below collection for automatic Data Table conversion
-List<List<String>>

5)Cucumber supports Data Table conversion to
-Custom Types
-DataTable object
-One of the supported collections

QUIZ 8:

1)**@After** hook in Cucumber will not execute if the scenario fails
-False== After hook executes irrespective of the status of the scenario

2)Which @After hook in Cucumber executes first?

- `@After(order = 1)`
- `@After(order = 2)`
- `@After(order = 2) executes first in reverse order`

3)Which **@After** hook in Cucumber executes first?

There are two Cucumber hooks:

- `@Before("@smoke")`
- `@Before()`

There are two scenarios:

- `@smoke`
- `Scenario: tagged as smoke`
- `Scenario: Not tagged`

What is the sequence of execution for hooks?

1—@Before
2—@Before(@smoke)
3—@Before()
-Non tagged @Before hook executes for both scenarios.Tagged @Before hook executes only for tagged scenario

4)Is this a valid statement?
@BeforeStep(Scenario scenario)
-YES

5)
We have two Cucumber hooks:

- @Before
- @BeforeStep

- We have a scenario:
- Given we have hooks
- When we execute scenario
- Then hook executes

What is the sequence of execution?
-
@Before
@BeforeStep
@BeforeStep
@BeforeStep

6)

QUIZ:9
1)In the runner, we have tags = "@stage or @featuretag". Which example will execute?

- @featuretag
- Feature: This is a feature
-
- Scenario Outline: This is a scenario outline
-
- @stage
- Examples:
- | product |
- | Blue Shoes |
-
- @prod
- Examples:
- | product |
- | Red Shoes |

—BOTH due to Tag inheritance. Hence both examples will execute irrespective of whether or not we add “or” condition with any other tag
Her ikisi de Tag inheritance nedeniyle. Bu nedenle, diğer etiketlere “veya” koşulu ekleyip eklememize bakılmaksızın her iki örnek de yürütülecektir.

2)Tags can be used to

-Execute a subset of scenarios
-Execute tagged hooks
Tags serve 2 purposes

3)In the runner, we have tags = "@featuretag and not @stage". Which example will execute?

```
@featuretag
Feature: This is a feature

Scenario Outline: This is a scenario outline

@stage
Examples:
| product |
| Blue Shoes |

@prod
Examples:
| product |
| Red Shoes |
```

Example tagged with @prod

4)In the runner, we have tags = "@prod". Which examples will execute?

```
@featuretag
Feature: This is a feature

@prod
@Rule: This is a rule

Scenario Outline: This is a scenario outline
```

```
•       @stage
•       Examples:
•       | product |
•       | Blue Shoes |
•
•       @prod
•       Examples:
•       | product |
•       | Red Shoes |
```

—BOTH

QUIZ:10

1) A scenario is truly independent if,
-it is using its own driven instance
-it is using its own test data
-it is NOT dependent on the application state
-it is NOT dependent on the user state

2)Why it is NOT a good practice to use WebDriver to prepare the test? They are:
-Ui Automation is flaky
-Slow
-May block validation
-Code duplication

3)What is the preferred way to synchronize WebDriver with the browser?
-Explicit wait

QUIZ;11
1)What is the primary purpose of using PicoContainer with Cucumber?

-To manage dependencies and share state between step definitions
Cucumber creates a new object of each step-definition class per scenario

Without DI, sharing data across steps requires static variables or other bad patterns

PicoContainer injects the same object instance into all step definitions of the same scenario

This allows you to share:

- **WebDriver**
- **Test data**
- **Scenario context**
- **Page objects**

- **API clients**
- **Database connections**

PicoContainer enables constructor injection, which allows Cucumber to automatically provide shared objects to all step classes in a scenario—clean, safe, and parallel-friendly.

2) How does PicoContainer inject dependencies into Cucumber step definition classes?

-By using constructor injection

Scans all step definition classes

Looks at their **constructors**

Identifies what dependencies they require (the constructor parameters)

Creates and injects those dependencies into each step class

Ensures all step classes in the **same scenario** receive the *same shared objects*

Why constructor injection?

Because with Cucumber:

- Step definition classes are recreated **per scenario**
- PicoContainer can easily manage object lifetimes
- It avoids static variables or singletons
- It makes sharing state across steps **clean & thread-safe**

This also enables parallel execution without race conditions.

3)What is a common practice when sharing state between Cucumber step definitions using PicoContainer?

-Using a shared state class injected into step definitions