**TDD (Test Driven Development) :**

**What ?**

* For Development purpose Developer use TDD
* Unit test (WBT)
* Developer write test before writing the code
* Test first approach

**Why we use ?** -> verify the code is working correctly or not ?

**Process :**

* **Write a test ->**  The first step in TDD is to write a test that specifies the desired behavior of the code. This test initially fails because the code hasn't been written yet.
* **Run the test ->** Execute the test for verify the code, it highlighted the missing functionality in code
* **Write the code ->** Developer write the code that fulfills the requirements of the test. For make test as pass
* **Run All test ->** The newly added code passes the new test and doesn't break any existing functionality
* **Refactor ->**  The code can be refactor to improve its design and maintainability
* **Repeat ->** This process is repeated, whenever the new feature is added

**Where ?**

Some popular TDD frameworks and tools that can be used for different programming languages include:

1. JUnit for Java
2. RSpec for Ruby
3. Pytest for Python

**BDD (Behaviour Driven Development):**

**What?**

* It is an extension of TDD
* It was written by BA, Developer ,QA
* It works on higher level of abstraction as compared to TDD
* It describe the behavior of a software system in a language that is easy to

understand by all parties

**Why?**

By using BDD we make feature file with multiple scenario in plain English language

It was used Gherkin Language, so even the non-technical people also understand easily

**Keywords on Gherkin Language:**

* **Feature** - Purpose of code
* **Background** - Collection of precondition for all scenario
* **Scenario -** Which kind of scenario
* **Scenario outline -** that scenario having multiple inputs
* **Given** - What we need to process
* **When** - Actual process
* **And** - add integration
* **Then -** result

**Cucumber options:**

* **Feature** : Use to set feature file path
* **Glue** : Use to set step definition file path
* **Dry run** : If (dry run = true) -> check step definition only, it doesn’t execute script. It allow only boolean values. (dry run = false) -> execution normally
* **Strict** :If (strict =true) -> It also check the step def and mapping the missing def . It execute script even the step def is missing . (strict = false) -> It pass the execution when the step def is missing .
* **Tags :** It is used to run the particular feature file and particular scenario.
* **Monochrome:** It is used to print the console message in human readable language.
* **Format (plugin)** : It is used to generate the report file in html, json, xml format.

**Where?**

Several BDD tools are use for different platform and the programming language are

1. Cucumber (Java/Ruby)
2. JBehave (Jave)
3. JBehave Web (Java with selenium integration)
4. Concordion (Java)
5. Cucumber -JS (Javascript)
6. Lettuce(Python)
7. Behave(Python)
8. SepcFlow(.Net)

**ATDD (Acceptance Test-Driven Development) :**

**What ?**

* It is extension of TDD .
* It involves collaboration between developers, testers, and business stakeholders

to define and agree upon acceptance criteria before the development process

begins.

**Why ?**

**Process :**

* **Collaborate on Acceptance Criteria:**

The whole team, collaborates to define acceptance criteria for the user stories or features to be developed. These acceptance criteria are based on the expected behavior of the system .

* **Write Acceptance Tests:**

Based on the acceptance criteria, the team writes acceptance tests that verify whether the system meets those criteria. Acceptance tests are usually written in a more natural language.

* **Automate Acceptance Tests:**

It was automated using testing frameworks or tools to facilitate automated testing. Automated acceptance tests serve as living documentation.

* **Implement Code to Fulfill Tests:**

Developer then start writing code to make the acceptance tests pass. The goal is to satisfy the specified acceptance criteria.

* **Run Tests Regularly:**

The automated acceptance tests are executed regularly, ideally after every code change or integration, to ensure that new code additions do not break existing functionality.

**JBehave**:

JBehave is a open source Java based framework supporting BDD framework.

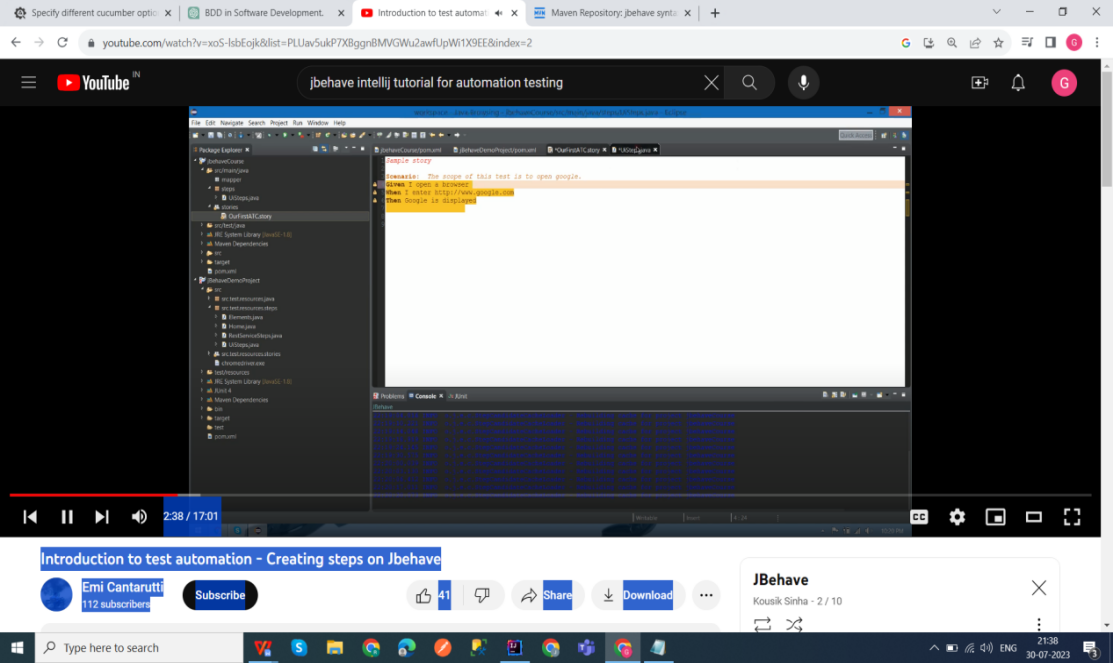
**JBehave Keywords:**

1. **Narrative**: The Narrative section is used to provide a high-level description of the user story or feature.
2. **Scenario -** Which kind of scenario
3. **Given** - What we need to process
4. **When** - Actual process
5. **Then -** result

**JBehave options:**

* Story Path: The location of the .story files
* Step Pattern: JBehave allows customization of step patterns using regular expressions, so you can match steps with different variations.
* Story Reporters: JBehave supports various types of reporters for generating output reports in different formats, such as HTML, XML, or plain text
* Failure Strategy: Defines how JBehave should handle failures during scenario execution. Options include 'Rethrowing', 'Silent', 'Pending', etc.
* Meta Filters: Allows filtering of scenarios based on metadata. You can use this to run specific scenarios or exclude certain scenarios based on tags or metadata.

JBehave code:



Step class:

