## **Test-Driven Development**

## **TDD** is an iterative development process. Each iteration starts with a set of tests written for a new piece of functionality.

These tests are supposed to fail during the start of iteration as there will be no application code corresponding to the tests. In the next phase of the iteration, Application code is written with an intention to pass all the tests written earlier in the iteration. Once the application code is ready tests are run.

Any failures in the test run are marked and more Application code is written/re-factored to make these tests pass. Once application code is added/re-factored the tests are run again. This cycle keeps on happening until all the tests pass. Once all the tests pass we can be sure that all the features for which tests were written have been developed.

## **Benefits of TDD**

## Unit test proves that the code actually works

## Can drive the design of the program

## Refactoring allows improving the design of the code

## Low-Level regression test suite

## Test first reduce the cost of the bugs

## **Drawbacks of TDD**

## Developer can consider it as a waste of time

## The test can be targeted on verification of classes and methods and not on what the code really should do

## Test become part of the maintenance overhead of a project

## Rewrite the test when requirements change

## **Behavior Driven Development**

Behavior Driven testing is an extension of TDD. Like in TDD, in BDD also we write tests first and the add application code. The major difference that we get to see here are

* Tests are written in plain descriptive English
* Tests are explained as behavior of application and are more user-focused
* Using examples to clarify requirements

## **Features of BDD**

1. Shifting from thinking in "**tests**" to thinking in "**behavior**"
2. Collaboration between Business stakeholders, Business Analysts, QA Team and developers
3. it is easy to describe
4. Driven by Business Value
5. Extends Test-Driven Development (TDD) by utilizing natural language that non-technical stakeholders can understand
6. BDD frameworks such as **Cucumber** or **JBehave** are an enabler, acting a "**bridge**" between Business & Technical Language

### **Selenium Test Script**

1. Launch the Browser
2. Navigate on the LogIn page
3. Enter UserName and Password
4. Click on login button
5. Print a successful message
6. Close the Browser

In it we are going to add

**It works in three layer**

1. Feature Layer
2. Step Defi*nitions*
3. Cucumber Options - Runner

Steps -

1. Create maven project
2. Install cucumber eclipse plugin from eclipse marketplace -
   1. **Cucumber Eclipse Plugin** helps eclipse to understand the basic Gherkin syntax and it works like a syntax highlighter.
   2. It highlights all the main syntax in the feature file which makes it more readable and clear.

### install the Natural Eclipse Editor for Gherkin for colour

### Delete all default packages

1. Convert your maven project in Cucumber project

Right click >> configure >> convert to Cucumber project

### Add All dependency

* 1. Selenium
  2. Cucumber Java
  3. Cucumber Testng
  4. Testng

1. Create ***Package***, by right click on the '*src/test/java*' folder and select *New > Package* and name it as ‘*features*’
2. Create ***Package*** by right click on the '*src/test/java*' folder and select *New > Package.* and name it as ‘*cucumberOptio*n*s*’
3. Create another ***Package***, by right click on the '*src/test/java*' folder and select *New > Package.* and name it as *‘stepDefinition’*
4. Create feature file
5. Create Test Runner Class in ‘*cucumberOptio*n*s*’
6. Create Step Definitions class to implement feature files
7. Run Test Runner file