**Difference Between For Loop and While Loop**

|  |  |
| --- | --- |
| **For Loop** | **While Loop** |
| It is used when the number of iterations is known. | It is used when the number of iterations is not known. |
| In case of no condition, the loop is repeated infinite times. | In case of no condition, an error will be shown. |
| Initialization is not repeated. | Initialization is repeated if carried out during the stage of checking. |
| Statement of iteration is written after running. | It can be written at any place. |
| Initialization can be in or out of the loop | Initialization is always out of the loop. |
| The nature of the increment is simple. | The nature of the increment is complex. |
| Used when initialization is simple. | Used when initialization is complex. |

**Difference Between HTML & XML**

|  |  |
| --- | --- |
| **HTML** | **XML** |
| HTML display data. | XML store and transfers data. |
| HTML is a predefined language. | XML is a standard language. |
| HTML stands for Hyper Text Markup language. | XML stands for extensible markup language. |
| HTML helps to develop the structure of web pages. | XML is used to exchange the data between the platforms. |
| HTML is less expressive. | XML is more expressive as compared to HTML. |
| HTML is not mandatory for a single root element. | XML documents must contain at least one root elements. |
| All content can be included in a body elements. | Attribute value are important in XML. |

**Defect Bug Lifecycle :-**

Deferred

Reopened

Rejected

Duplicate

Closed

Verified

Retest

Pending Retest

Fixed

Open

Assigned

New

**Testing :-**

**Software Testing :-**

Software testing is the process of assessing the functionality of a software program.

**Bug :-**

A software bug is an error, flaw or fault in the design, development, or operation of computer software that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.

**There are 3 types of bug**

Logic bugs

Algorithmic bugs

Resource bugs

**Error :-**

An error is a mistake made in the code; that’s why we cannot execute or compile code.

* Types of error
* Syntactic error
* User interface error
* Flow control error
* Error handling error
* Testing error
* Hardware error

**Fault :-**

The fault is a state that cause the software to fail to accomplish its essential function.

**Types of fault:-**

* Business logic faults
* Functional and logical faults
* Security fault
* Software/hardware fault

**Defect :-**

The defect is the different between the actual outcomes and expected outputs.

**Types of defect:-**

* High
* Medium
* Low

**Failure :-**

If the software has lots of defect, it leads to failure or causes failure.

**Mistake:-**

An error is a mistake made in the code; that’s why we cannot execute or compile code.

When each component or module works separately, we need to check the data flow between the dependent modules, and this process is known as integration testing.

**Testing Principles:-**

1.Testing shows presence of defects

2. Exhaustive testing is not possible

3. Early testing

4.Defect clustering

5. Pesticide paradox

6.Testing is context dependent

7.Absence of error fallacy

It determines the system’s software functional requirements.

**2) Non-functional Testing:-**

Non-functional testing is also known as NFT. This testing is not functional testing of software. It focuses on the software’s performance, usability, and scalability.

**3) Regression Testing:-**

It ensure that the newly added code is compatible with the existing code. In other words, a new software update has no impact on the functionality of the software. This is carried out after a system maintenance operation and upgrades.

**Statement Coverage :-**

**Test case 1** :-

x = 20 , y = 30

Input (int x, int y ) {

Int z = ((x + y) /200) \*100;

if (z > 50 )

printf (“PASS”);

else

printf (“FAIL”);

}

In test case 1 it is observed that the value of z will 25 which is less than 50 so as per the total condition so that result will be fail.

Therefore,

Total no. of statements = 7

Number of executed statements = 6

**Statement coverage** = No. of executed statements × 100

Total no. of statements

= 6/7 × 100

= 600/7

= 85%

**Test case no 2**

If x=100, y=75

Input (int x, int y) {

Int z = ((x + y) /200) \* 100;

if ( z > 50)

printf (“PASS”);

else printf (“FAIL”);

}

**In test case 2 :-**

It is observed that the value of z is 87.5 will be greater than 50 as per the condition so that the result may be pass

Total no of statement = 7

Total no of executed statement = 5

Statement coverage = Total no of executed statement × 100

Total no of statement

= 5/7×100

= 71.5%

**Branch Coverage :-**

Read A

Read B

If A + B>100 THEN

Print “ Larger than 100”

ENDIF

If A + B>50

Print “Larger than 50”

ENDIF

Read A

Read B



Larger than 50

A+B>50

Larger than 100

A+B>100

Branch Coverage = Number of executed branch × 100

Total no of branches

= 5/8 × 100

= 62.5 %

Branch Coverage % for result Yes is 62.5 %

**Branch coverage % for result No** = No. of executed branch × 100

Total no of branch

= 3/8 × 100

= 37.5%

Branch coverage % for result no is 37.5%

**Difference between White box Testing and Black Box testing**

|  |  |
| --- | --- |
| **Black Box** | **White Box** |
| It is a software testing technique that examines the functionality of software without knowing its internal structure or coding. | In white-box testing, the internal structure of the software is known to the tester. |
| Black Box Testing is also known as functional testing, data-driven testing, and closed-box testing. | It is also known as structural testing, clear box testing, code-based testing, and transparent testing |
| It is not well suitable for algorithm testing | It is well suitable and recommended for algorithm testing. |
| It is mainly performed by the software testers. | It is mainly performed by developers. |
| It is less exhaustive than White Box testing. | It is more exhaustive than Black Box testing. |
| It can be performed by trial and error technique. | It can test data domain and data boundaries in a better way. |
| It does not find the errors related to the code. | In white-box testing, there is the detection of hidden errors. It also helps to optimize the code. |

**Use Case Diagram**

Hotel Management System

Receptionist Administrator

**Introduction :-**

HOTEL MANAGEMENT SYSTEM is a hotel reservation site script where site users will be able to search rooms availability with an online booking reservation system. Site users can also browse hotels, view rooms inventory, check availability, and book reservations in real-time.

|  |  |
| --- | --- |
| **Actors** | **Administrator** |
| Pre-conditions | User must exist in the data base |
| Post-conditions | User is logged in |
| Main Scenario | 1- System display the login page  2- User enter his credential  3- System redirect the user to the dashboard |

**TDD :- Test-Driven Development**

TDD, also called test-driven design, is a method of implementing software programming that interlaces unit testing, programming and refactoring on source code.

TDD is a development practice while BDD is team methodology.

In TDD, the developers write the test while in BDD the automated specification are created by users or testers. (While developers writing them to the code under test).

**BDD :- Behavior-Driven Development**

**Define :-**

Behaviour Driven Development (BDD) is a synthesis and refinement of practices stemming from Test driven development. (TDD) and Acceptance test driven development (ATDD).

Behavior-Driven Development, or BDD, lately.

BDD is an extension of Test-Driven Development (TDD) that emphasizes developing features based on a user story and writing code that provides a solution to real problems.

Example :-

Cucumber

Spec-Flow

Quantum

J-Behave

Codeception.

**Benefits of BDD:-**

Strong collaboration. With BDD, all the involved parties have a strong understanding of the project and they can all have a role in communication and actually have constructive discussions. ...

High visibility. ...

The software design follows business value. ...

The ubiquitous language . ...

Software development meets user need. ...

More confidence from the developers' side. ...

Lower costs. ...

**Difference between TDD & BDD**

|  |  |
| --- | --- |
| **TDD** | **BDD** |
| This is driven by the developers. | This is driven by developers, QAs, product owners, customers and business analysts. |
| This is mostly focused on the coding implementation of the functionalities of the application. | This is mostly focused on the business scenarios of the product. |
| This is mainly used for unit testing. | This is mainly for making developers, testers, product owners, customers and business analysts agree on functional requirements of the application. |
| TDD requires team members having technical knowledge. | BDD does not require team members with technical knowledge. |
| TDD is known as Test Driven Development. | BDD is known as Behavior Driven Framework |
| The designing of test cases is the starting point of TDD. | The designing of scenarios is the starting point of BDD |
| TDD is used for projects involving third party tools and APIs. | BDD is used for projects involving the end users interaction |

**Different Key words inside the Gherkin format**

* Feature
* Rule
* Background
* Scenario
* Scenario Outline
* Given
* When
* Then
* And
* But
* ' \* '

**1) Feature :-**

**Feature**: Login Action Test

**Description:** This feature will test a Login and Logout functionality

**Scenario**: Successful Login with Valid Credentials

Given User is on Home Page

When User Navigate to Login Page

And User enters User Name and Password

Then Message displayed Login Successfully

**2) Rule :-**

**Feature:** OTP generation test

**Rule:** OTP is not generated if the account user has opted out of two-factor authentication for transactions below INR 2000

**Scenario:** For an amount lesser than INR 2000, if the account user has opted out of the two-factor authentication, an OTP must not be generated

**Scenario:** For an amount greater than INR 2000, if the account user has opted out of the two-factor authentication, an OTP must be generated

**Scenario Outline:** If the account user has not opted out of two-factor authentication, an OTP must be generated regardless of the amount.

**3) Background :-**

**Feature:** Add to Cart

This feature will test functionality of adding different products to the User basket from different flow

**Background:** User is Logged In

**Scenario:** Search a product and add the first result/product to the User basket

Given User searched for Lenovo Laptop

When Add the first laptop that appears in the search result to the basket

Then User basket should display with 1 item

**3) Scenario :-**

Feature: Delivery Application - delivery type selection

**Rule**: Only capital cities of a country are eligible for regular delivery of automobile spare parts. For all other cities, customers must select premium as the delivery type.

**Scenario Outline:** The delivery type of automobile spare parts is dependent on the destination city of delivery.

Given User is raising a delivery request for automobile spare parts

When User enters the <"city">, a <"delivery type"> is set as a system default

Then User submits delivery request

**Feature in cucumber format :-**

**1) Cucumber Tags -:**

These are normally used over the feature file to classify the scenarios over the feature files as per their given tag name. We can have multiple tags for a given scenario in the feature file.

Tags are user-defined and we can give any name to it such as @Smoke, @Regression, etc.

**Feature:** Title of your feature  
I want to use this template for my feature file

@tag1

**Scenario:** Title of your scenario  
Given I am on Github home page  
When I specify Username and Password  
And Click on Sign In button  
Then I should be able to see logout option

**2) Cucumber Background -:**

These are steps or series of steps that are common to all the scenarios in the feature file.

It allows us to add some context to the scenarios for a feature where it is defined. It runs before every scenario for a feature in which it is defined.

**Feature**: Title of your feature  
I want to use this template for my feature file

**Background:**  
Given I am on Gmail login page  
When I specify Username and Password  
And Click on Sign In button

**Scenario:** Create new message from data table  
When I am on New Email Page  
And I specify the following details  
| To1 | Subject |  
| Person1@email.com | Person1 subject |  
| Person2@email.com | Person2 subject |

**Scenario:** Create new message from transposed data table  
When I am on New Email Page  
And I specify following details from transpose table  
| To1 | Person1@email.com | Person2@email.com |  
| Subject | Person1 subject | Person2 subject |

**3) Cucumber Hooks -:**

These are the blocks of the code that runs before or after each scenario. So that we can define these, anywhere in our project.

**For example :-**

**Step Definition.**

As per the definition, these are just two annotation @After and @Before. In the console, we can see the blocks getting executed and giving clear output. We can also execute the hooks for specific Tags.

**Feature:**  
As a user  
I want to be able to add new clients in the system  
So that I can add accounting data for that client

**Background:**  
Given I am on Github home page  
When I specify Username as “Aishwarya123@gmail.com” and Password as “XYZ”  
And Click on Sign In button

@Smoke  
    **Scenario:** Editing the profile  
Given I click on Your Profile option  
When I click on edit profile button  
And Uploaded new picture  
Then I should be seeing new profile picture

@Regression @Everytime  
    **Scenario:** Create new gist  
Given I click on Your Gists option  
When I provide filename, description  
And click on Create public gist method  
Then I should be seeing the new gist

**JAVA**

**Data types are divided into two groups**:

* Primitive data types - includes byte, short, int, long, float, double, boolean and char
* Non-primitive data types - such as [String](https://www.w3schools.com/java/java_strings.asp), [Arrays](https://www.w3schools.com/java/java_arrays.asp) and [Classes](https://www.w3schools.com/java/java_classes.asp)

**Java divides the operators into the following groups:**

* Arithmetic operators
* Assignment operators
* Comparison operators
* Logical operators
* Bitwise operators

**Java Conditions and If Statements**

* + Less than: a < b
  + Less than or equal to: a <= b
  + Greater than: a > b
  + Greater than or equal to: a >= b
  + Equal to a == b
  + Not Equal to: a != b

**Java has the following conditional statements:**

* Use if to specify a block of code to be executed, if a specified condition is true
* Use else to specify a block of code to be executed, if the same condition is false
* Use else if to specify a new condition to test, if the first condition is false
* Use switch to specify many alternative blocks of code to be executed

**Java operators :-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Description** | **Example** |
| + | Addition | Adds together two values | x + y |
| - | Subtraction | Subtracts one value from another | x- y |
| \* | Multiplication | Multiplies two values | x \* y |
| / | Division | Divides one value by another | x / y |
| % | Modules | Returns the division remainder | x % y |
| ++ | Increment | Increases the value of a variable by 1 | ++ x |
| -- | Decrement | Decreases the value of a variable by 1 | --x |