**UNIT TESTING**

Unit testing focuses on individual units and components of a system.

Purpose : to check if the software works as intended and meets our requirements.

It Is generally done by developers, and is performed early in the development process, before the code is integrated and tested as a whole.

Unit tests are automated, run each time the code is changed to ensure that code doesn’t break down the existing functionality.

It is done on smallest possible unit, by isolating it from the entire system.

Objective of unit testing:

1. To isolate the section of code.
2. To verify correctness of code.
3. To test every function and procedure.
4. Fix bugs at early stage and reduce costs.

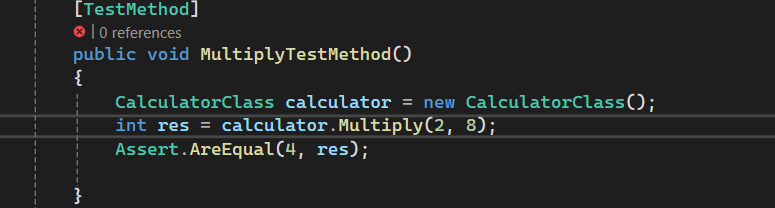
Advantages:

1. Allows developers to understand what a particular unit is providing a functionality and how to use it to gain a better understanding
2. Allow developers to refine code and make sure that code works properly.
3. Early detection of issues
4. Improved code quality
5. Faster development
6. Better documentation
7. Reduced time and cost, for later testing

**Manual testing vs automated testing**

|  |  |
| --- | --- |
| Manual testing | Automated testing |
| Humans execute the test cases on a piece of software | Makes use of various automated tools to test the cases defined. |
| Consumes human resources, time consuming | Processing time is faster which uses computation power |
| Allow exploratory analysis and random testing. | Doesnot allow exploratory analysis and random testing |
| Initial investment is less and return on investment is lower as well | Initial investment is more but return on investment is more |
| There can be human errors. | Extremely robust and reliable as we use variety of scripts, that a software goes. |
| Can work fine with changes compared to automated testing as it is done by humans. | Scripts need to be changed everytime whenever there is a change in UI |
| Resources are needed(investing in more humans) | Resources are needed |
| Will not be cost effective for testing a big software | If there is a huge software, then it is cost effective |
| Doesn’t offer feasibility when performance testing is the main point | Performance testing like load testing, stress testing etc, can be done easily. |
| No need of programming knowledge | Involves of requirement of programming knowledge |
| Lowers the efficiency in devops life cycle. | Integral in devops life cycle. |
| Still needed as it is still legacy | Brings efficiency like any other. |

Failed test case



Passed test case

A computer screen shot of a program

Description automatically generated

**NUnit framework**

NUnit is a popular open-source unit testing framework for .NET, similar to JUnit for Java or xUnit.net. It provides attributes and assertions to facilitate the writing and execution of unit tests for .NET applications.

[TestFixture]: Marks a class that contains tests. Each class with tests should be decorated with this attribute.

[Test]: Marks a method as a test case. NUnit will execute all methods marked with this attribute.

[SetUp]: Marks a method that should be run before each test method in the test fixture. It is used to set up the test environment or initialize resources.

[TearDown]: Marks a method that should be run after each test method in the test fixture. It is used to clean up resources or reset the test environment.

[TestCase]: Allows you to specify parameterized tests. You can provide multiple sets of arguments to a test method using this attribute.

[Ignore]: Marks a test method to be ignored. NUnit will skip the execution of the test method when running tests.

[Category]: Allows you to categorize tests. You can use this attribute to organize your tests into logical groups.

[Timeout]: Specifies a timeout for the test method. If the test method takes longer than the specified timeout, NUnit will consider it as a failure.

[TestCaseSource]: Allows you to specify a method that provides data for parameterized tests. The method specified by this attribute should return a collection of test cases.

[Explicit]: Marks a test method as explicit. NUnit will only run explicit tests when explicitly specified, ignoring other tests.

**MS Unit**

[TestClass] attribute marks the class CalculatorTests as a test class.

[TestInitialize] attribute marks a method to be run before each test method. In this case, TestInitialize() method initializes the calculator object before each test.

[TestMethod] attribute marks a method as a test method. MSTest will execute all methods marked with this attribute.

Assert.AreEqual() method is used to verify that the actual result matches the expected result for each operation.

**XUnit.Net**

[Fact]: Marks a method as a test method. xUnit.net will execute all methods marked with this attribute.

[Theory]: Marks a method as a parameterized test method. You can provide different sets of data using [InlineData] attribute to test different scenarios.

[InlineData]: Provides data for a parameterized test method. Used in conjunction with [Theory], it allows you to provide different sets of input data for parameterized tests.

using Xunit;

public class CalculatorTests

{

private Calculator calculator;

public CalculatorTests()

{

calculator = new Calculator();

}

[Theory]

[InlineData(1, 2, 3)] // First argument + second argument = expected result

[InlineData(0, 0, 0)]

[InlineData(10, -5, 5)]

[InlineData(-10, -20, -30)]

public void AddTestMethod(int a, int b, int expected)

{

// Arrange

// Act

int result = calculator.Add(a, b);

// Assert

Assert.Equal(expected, result);

}

}

**Test Doubles**

A diagram of a flowchart

Description automatically generated

Test doubles are objects that replace dependencies of the unit under test during unit testing.

They simulate the behavior of real objects but are designed to be controlled and manipulated in specific ways during testing.

Test doubles are used to isolate the unit under test and verify its behavior in isolation.

Dummy: A dummy object is a placeholder object that is passed to a method but is not used in the test. It simply satisfies the method's parameter requirements.

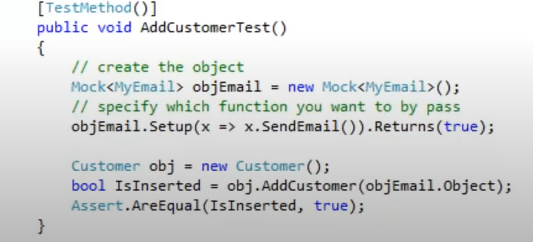
Stub: A stub is a simplified implementation of a dependency that provides predetermined responses to method calls. Stubs are used to control the behavior of the dependency and provide consistent results during testing.

Mock: A mock object is a dynamic object with pre-programmed expectations. It allows you to specify method calls that you expect to occur during the test and verify that they are called with the correct arguments and in the correct order.

Fake: A fake object is a simplified implementation of a dependency that behaves similarly to the real object but is designed for testing purposes. Fakes are often used when the real object is too complex or difficult to use in tests.

Spy: A spy is a type of mock object that records the interactions between the unit under test and its dependencies. It allows you to verify that certain methods are called on the dependency during the test.

**Mocking**



**Test driven development**

A diagram of a process

Description automatically generated

Step 1 :

First write the test case

public void TestMethodAdd()

{

Maths maths = new Maths();

int result = maths.add(1, 2);

Assert.AreEqual(3, result);

}

When we compile, the test case fails as no Maths class is there

Step 2 :

Create the class library maths

namespace MathsLibrary

{

public class Maths

{

public int Add(int a,int b)

{

return 0;

}

}

}

We write the least amount of code, so that the test case compiles, and we have the maths class.

A screenshot of a computer

Description automatically generated

Test case failed as the logic is not correct

Step 3 :

Now write the logic

public class Maths

{

public int Add(int a,int b)

{

return a+b;

}

}

A screenshot of a computer

Description automatically generated

Step 4 :

Iteratively we search for all the edge cases, and accordingly we implement the logic in the main maths class.

Advantages:

1. Understand the requirements well
2. Iterative development and testing
3. Catching the defects at the development stage( earlier the defects are catched the better it is)
4. You are forced to write test cases, so testing is implemented by default
5. Documentation is there