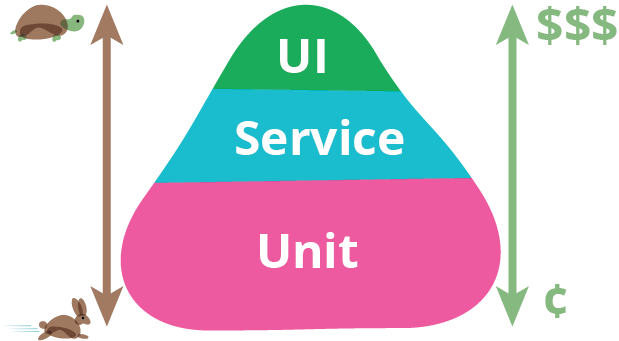
UNIT TESTING

DEFINITION: a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing can be done manually but is often automated.

# [TEST PYRAMID:](https://martinfowler.com/bliki/TestPyramid.html)



MASTERCARD REQUIREMENTS: *Locate documentation*

DEFINITIONS:

1. **Stubs**: a controllable replacement for an existing dependency (or collaborator) in the system. By using a stub, you can test your code without dealing with the dependency directly. An object that holds predefined data and uses it to answer calls during tests. It is used when we cannot or don’t want to involve objects that would answer with real data or have undesirable side effects.

An example can be an object that needs to grab some data from the database to respond to a method call. Instead of the real object, we introduced a stub and defined what data should be returned.

1. **Fakes**: objects that have working implementations, but not same as production one. Usually they take some shortcut and have simplified version of production code. An example of this shortcut, can be an in-memory implementation of Data Access Object or Repository. This fake implementation will not engage database, but will use a simple collection to store data.

1. **Mocks**: a fake object in the system that decides whether the unit test has passed or failed. It does so by verifying whether the object under test called the fake object as expected. There’s usually no more than one mock per test.

1. **External dependency:** is an object in your system that your code under test interacts with and over which you have no control

1. **Isolation**: being separate and apart from the application, and all other units. The whole point of unit tests is to reduce the scope of the system under test to a small subset that can be tested in isolation.

1. **Isolation (mocking) Framework**: a reusable library that can create and configure fake objects at *runtime*.

1. **SUT**: system under test

BEST PRACTICE:

1. Use the "Arrange, Act, Assert" pattern
2. Always Write Isolated Test Cases and avoid external dependencies
3. Ideally, One Assert Per Test Method, i.e.., Test One Thing Only in One Test Case
4. Avoid Test Interdependence
5. Recognize Test Setup Pain as a code Smell (i.e., when unit test becomes cumbersome, consider design flaw in app)
6. Limit Use of Mocks
7. Add tests to the Build (i.e., if a test fails, than the build fails)

TOOLS:

UNIT TESTING FRAMEWORKS:

1. **Visual Studio Test Tools**: simple built-in Visual Studio tool and uses an easy to understand method attribute structure (much like most testing frameworks) where you are able to add tags such as ‘[TestClass]’ and ‘[TestMethod]’ to your code in order to get testing.
2. **Nunit**: widely used tool for testing, and it serves as an excellent example of the open source unit testing frameworks.
3. **Xunit**: open source testing platform with a larger focus in extensibility and flexibility.

ISOLATION (MOCKING) FRAMEWORKS:

1. **Moq** : most popular and friendly mocking framework for .NET <https://www.nuget.org/packages/Moq/>
2. **Rhino Mocks:** will generate fake objects to replace the dependencies that you have, and then allow you to tell them, at runtime, how to behave.

<https://www.nuget.org/packages/rhinomocks/>

1. **FakeItEasy :** mocking framework for .NET that works great in C# and VB.NET alike. No need to know the difference between a stub, a mock or a spy, everything's a fake!

<https://www.nuget.org/packages/FakeItEasy/>

1. **Nsubstitute:** designed for Arrange-Act-Assert (AAA) testing and with Test Driven Development (TDD) in mind.

<https://www.nuget.org/packages/NSubstitute/>

HELPFUL TOOLS:

1. **Autofixture**: an open source library for .NET designed to minimize the 'Arrange' phase of your unit tests in order to maximize maintainability. Its primary goal is to allow developers to focus on what is being tested rather than how to setup the test scenario, by making it easier to create object graphs containing test data.

<https://www.nuget.org/packages/AutoFixture/>

1. **Entity Framework Effort**: Fake ObjectContext Realization Tool. It is a powerful tool that enables a convenient way to create automated tests for Entity Framework based applications.

<https://www.nuget.org/packages/Effort/>

# CONSIDERATIONS:

1. Not every test you could conceivably write qualifies as a unit test.
2. If you write code that stuffs things into a database or that reads a file from disk, you have not written a unit test. Unit tests don’t deal with their environment and with external systems to the code base. If it you’ve written something that can fail when run on a machine without the “proper setup,” you haven’t written a unit test.
3. Unit tests also don’t count as other sorts of tests. If you create some sort of test that throws thousands of requests for a service you’ve written, that qualifies as a smoke test and not a unit test. Unit tests don’t generate random data and pepper your application with it in unpredictable sequences. They’re not something that QA generally executes.
4. Unit tests don’t exercise multiple components of your system and how they act. If you have a console application and you pipe input to it from the command line and test for output, you’re executing an end-to-end system test — not a unit test.

# REFERENCES:

1. [Microsoft Guidelines for Test-Driven Development](https://msdn.microsoft.com/en-us/library/aa730844(v=vs.80).aspx)
2. [Considerations](https://stackify.com/unit-testing-basics-best-practices/?utm_referrer=https://www.google.com/https://www.safaribooksonline.com/library/view/practical-test-driven-development/9781788398787/)
3. [Best Practice](http://www.sw-engineering-candies.com/blog-1/unit-testing-best-practices)
4. [The Art of Unit Testing, 2nd Edition, by Roy Osherove](https://www.manning.com/books/the-art-of-unit-testing-second-edition)
5. [Mastercard Continuous Delivery](https://fusion.mastercard.int/confluence/display/CD/Continuous+Integration+and+Delivery)