WEEK 2 NUnit and Moq Hands On

**NUnit HandsOn:**

Q .Explain the meaning of Unit testing and its difference on comparison with Functional testing

* + Smallest unit to test mocking dependencies

**ANSWER:**

* Unit **Testing** focuses on testing **individual components (units)** of code (like a method or class) in isolation.
  + It **mocks dependencies** to isolate the behaviour of the unit.
  + Fast, done by developers.
* Functional **Testing** verifies that the **system works as a whole** based on requirements.
  + It does **not** mock dependencies.
  + Focuses on business logic, often done by QA teams.

Q .List various types of testing

* + Unit testing, Functional testing, Automated testing, Performance testing

**ANSWER:**

* **Unit Testing** – Test small units of code.
* **Functional Testing** – Validate user requirements.
* **Automated Testing** – Use tools/scripts to run tests without manual effort.
* **Performance Testing** – Check system under load, stress, and scalability.

Q. Understand the benefit of automated testing

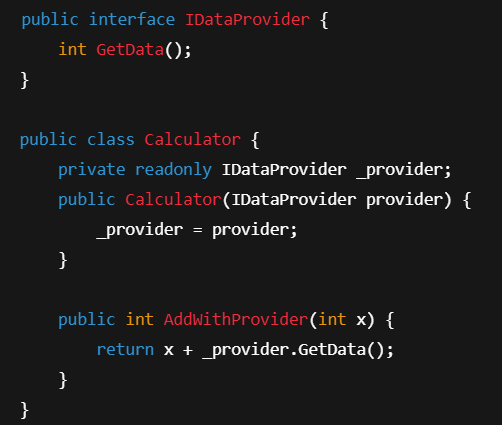
**ANSWER:**

* **Fast feedback loop** during development
* **Early bug detection**
* Reduces **manual testing effort**
* Enables **Continuous Integration/Delivery (CI/CD)**
* Ensures **code reliability and stability** over time

Q. Explain what is loosly coupled & testable design

* + Write code that is NOT dependent on the class for data.

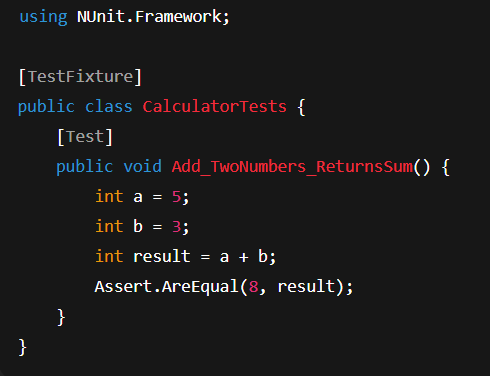
**ANSWER:**

* A **loosely coupled design** means components don't depend tightly on each other.
* Improves **reusability and testability**.
* Use interfaces and dependency injection: 

Q. Write your first testing program to validate a calculator addition operation

* + TestFixture, Test
* Understand the need of [SetUp], [TearDown] & [Ignore] attributes.
* Explain the benefit of writing parameterised test cases.
  + TestCase

**ANSWER:**

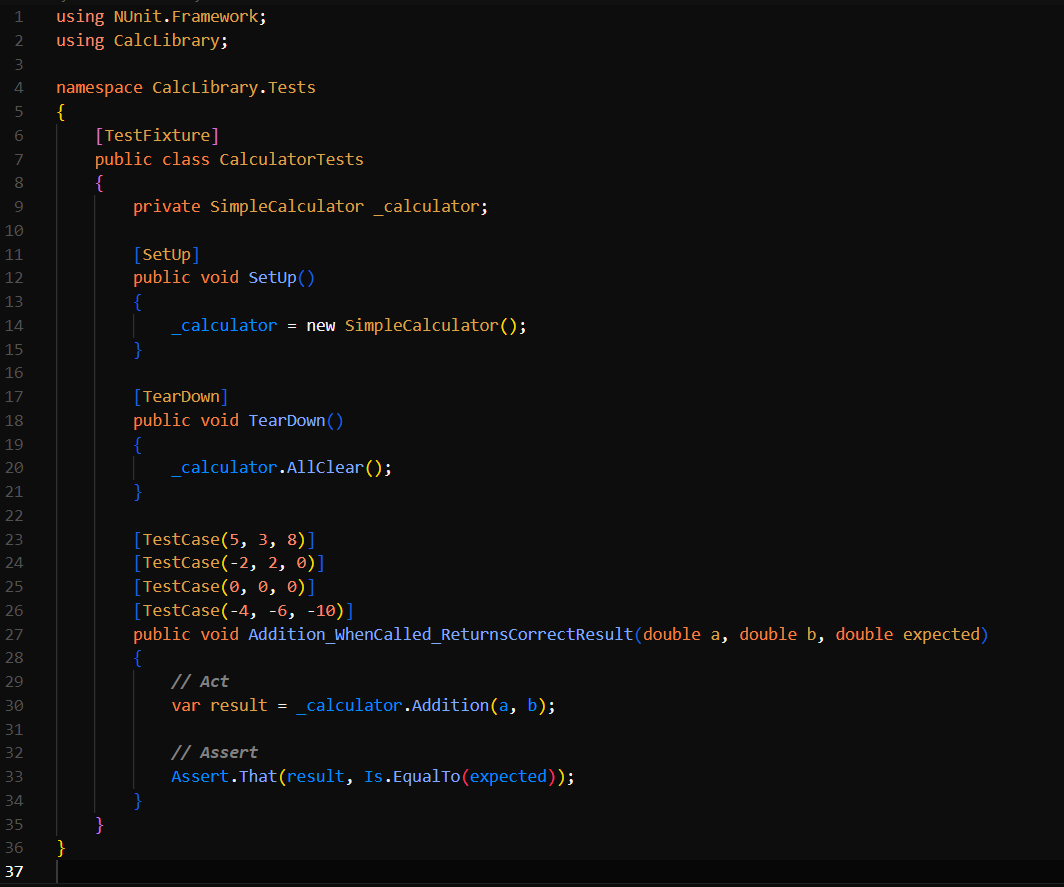


* **[SetUp]**: Runs **before each test**, useful for initializing objects.
* **[TearDown]**: Runs **after each test**, used for cleanup (e.g., closing DB connections).
* **[Ignore]**: Skips a test temporarily (e.g., due to known issues or incomplete implementation).

Benefit of writing parameterised test cases:

* Helps **test multiple inputs/outputs** with a single method.
* Reduces code duplication and improves coverage.

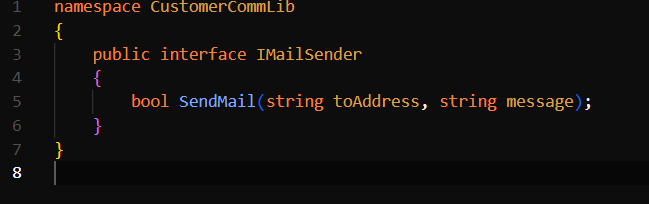
In the CalcLibrary, The CalculatorTests.cs will be:



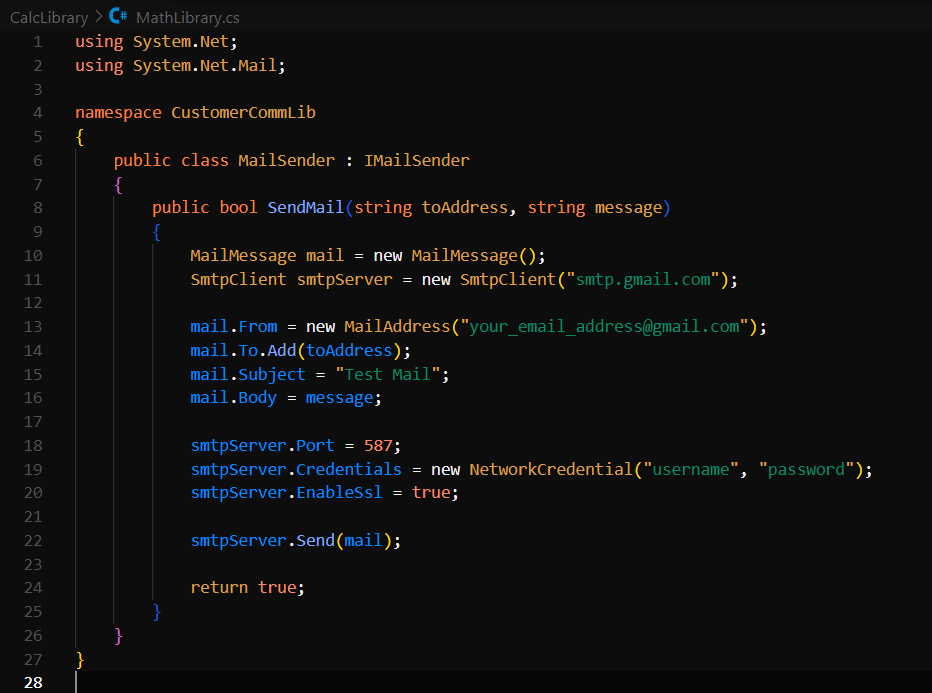
**Moq HandsOn:**

**Task 1:**

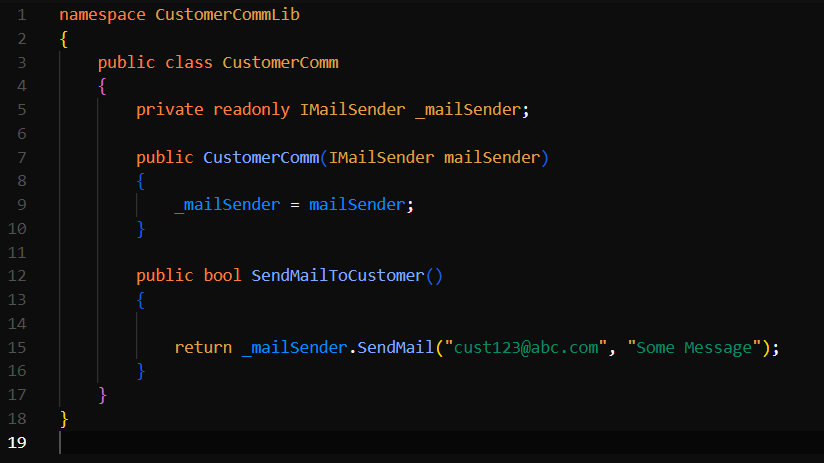
IMailSender.cs



MailSender.cs



CustomerCom.cs



**Task 2:**

CustomerCommTests.cs

