**C# .NET**

.NET is framework for building applications (web, desktop, machine learning, etc.)

\*\* C/C++: **compiles** into machine code that can be used for that OS only

\*\* Java, C#:

* **Compiles** to Intermediate Language code (independent of the OS)
* JIT (Just In-time Compilation) by a program called **CLR (Common Language Runtime)** to produce machine code that corresponds to that specific OS

1. **Architecture of .NET applications**

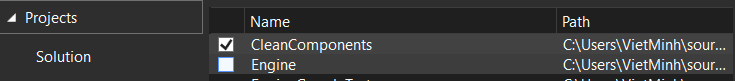
* Solutions 🡪 contains multiple projects
* Project 🡪 develops one or more Assemblies.
* Assembly (.dll or .exe) 🡪 container for related namespaces
  + .dll is a library, cannot be executed by itself
  + .exe is a executable that reserve threads and run
  + When compiling, the compiler builds one or more Assemblies.
* Namespace 🡪 Container for related classes

(in C++, we #include libraries containing multiple namespaces)

* Class: Data (Attributes) & Methods (Functions)

1. **Connecting Projects (References & Dependencies)**

Since every types developed are namespaces, classes, etc (OOP), we can easily “pass” them to other files

* Right click the “Reference” (or “Dependencies”) tab then select “Add Reference”
* Choose the project to be referenced from and check the box
* You can then use all types defined in that project with “using”

Difference between References and Dependencies: Reference is old framework, Dependency is new. However, they can be represented differently.

* If A **refers** to B, A “uses” a namespace/type from B.
* A screenshot of a computer

  Description automatically generated with medium confidenceIf A **depends** on B, B needs to be built before A (not necessarily a reference, e.g., artifacts produced by the build of another project)
* Sets up Build Order

A screenshot of a computer

Description automatically generated with medium confidence

1. **C# Program**

* **using** directive 🡪 includes other **namespaces/types**
  + using System (basic utility classes)
  + using System.Collections.Generic (lists & collections…)
  + A black background with white text

    Description automatically generated with low confidenceusing System.Linq (data)
  + using System.Text (Coding & texts)
  + using System.Threading.Tasks (multithreading)
* You can also create an alias for a namespace/type with a *using alias directive.*
* Everything in C# are classes, even including the program (*internal class Program*)
* main function is within the Program class:

static void Main (string[] args)

Graphical user interface, text, application, chat or text message

Description automatically generated(Intellisense: purple box stands for methods)

Run Application: Ctrl + F5

1. **\*\*\*\* Access Modifiers \*\*\*\***

The following seven accessibility levels can be specified using the access modifiers:

* public: Access isn't restricted.
* protected: Access is limited to the containing class or types derived from the containing class.
* internal: Access is limited to the current assembly.
* protected internal: Access is limited to the current assembly or types derived from the containing class.
* private: Access is limited to the containing type.
* private protected: Access is limited to the containing class or types derived from the containing class within the current assembly.
* file: The declared type is only visible in the current source file. File scoped types are generally used for source generators.