Standard .Net : → OS : windows.

32+ languages supported,
↳ 32+ language compilers are available.

↓

MSIL (C#), Vb.Net, F#, Small Talk

→ .Net language

→ Emits MSIL after compilation.

$$\begin{tikzcd}
\text{C\#} \arrow[d] & & \text{Vb.Net} \arrow[d] & & \text{F\#} \arrow[d] \\
\text{CSC} \arrow[d] & & \text{VBC} \arrow[d] & & \text{fSC} \arrow[d] \\
\text{MSIL .exe / .dll} & & \text{MSIL .exe / .dll} & & \text{MSIL .exe / .dll} \\
\text{MSIL .exe} \arrow[d] & & \text{MSIL.exe} \arrow[d] & & \text{MSIL.exe} \arrow[d] \\
\text{CLR} \arrow[d, no head] & & \text{CLR} \arrow[d] & & \text{CLR} \arrow[d] \\
\text{M/C} \arrow[r] & \text{CPU} & \text{M/C} \arrow[r] & \text{CPU} & \text{M/C} \arrow[r] & \text{CPU}
\end{tikzcd}$$

Common Language Specifications [CLS]

* Language should support / based on OOP

* new keyword → compulsory support to create new objects.

* After compilation using Common Type System [CTS] :- we need to generate MSIL.

* Allocation and memory deallocation should be done by CLR

* .exe / .dll execution → CLR target.

* de-allocation → CLR → target Garbage collector (GC)

* Exception Handling support → target → CLR

Common Type System (CTS)

$$\begin{tikzcd}
\text{C\#} & & & \text{vb.Net} & \\
\text{int x=10;} \arrow[d] & & & \text{Dim x as Intiger} \arrow[d] & \\
\text{CSC} \arrow[d] & \text{+ CTS} \arrow[dl] & & \text{VBC} \arrow[d] & \text{+ CTS} \arrow[dl] \\
\text{System.Int32} \arrow[dr, dash] & & & \text{System.Int32} \arrow[dll, dash] & \\
& \boxed{\text{MS2C}} & & &
\end{tikzcd}$$

CTS

code	MSIL
int	System.Int32
short	system.Int16
long	System.Int64
str	system.String
double	System.Double
bool	System.Boolean
⋮	⋮

.Net Core → open-source, microsoft f/w, → os: mac, linux, windows,

appⁿ → CLI, web-appⁿ mobile, web API,

Standard .Net	.Net core
mscorlib.dll	System.Private.CoreCLR.dll

.Net core

$$\begin{tikzcd}
\text{C\# | vb.Net} \arrow[d] & & \text{F\#} \arrow[d] \\
\text{Roslyn Compiler} \arrow[d] & & \text{F\# compil-} \\
\text{MSIL + code generation.} & &
\end{tikzcd}$$

* Features of CLR :—

1. Memory allocation
2. memory Deallocation via GC
3. Exception Handling
4. Second Time Compilation via JIT Compile
5. Loading dependencies
6. Security Checking.

* JIT Compilers :—

1) Standard / Normal JIT
2) Pre-JIT compiler

FCL → F/w class Library files

→ .Net F/w → SDK

BCL → Base Class Library files

System.dll

system.data.dll

system.Collection.dll

1. OS → windows

2. Sql Server → ms Sql Server.

3. IIS web server.

4. Visual Studio → 2022

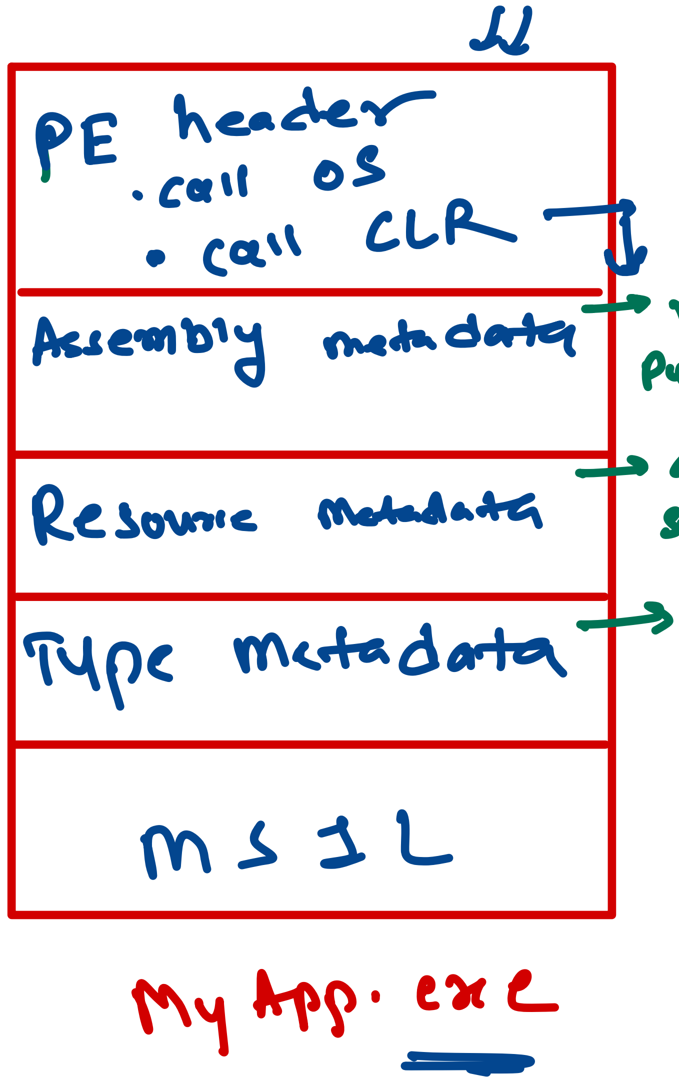
$\downarrow$

F/w

configuration with SQL server

IIS web server

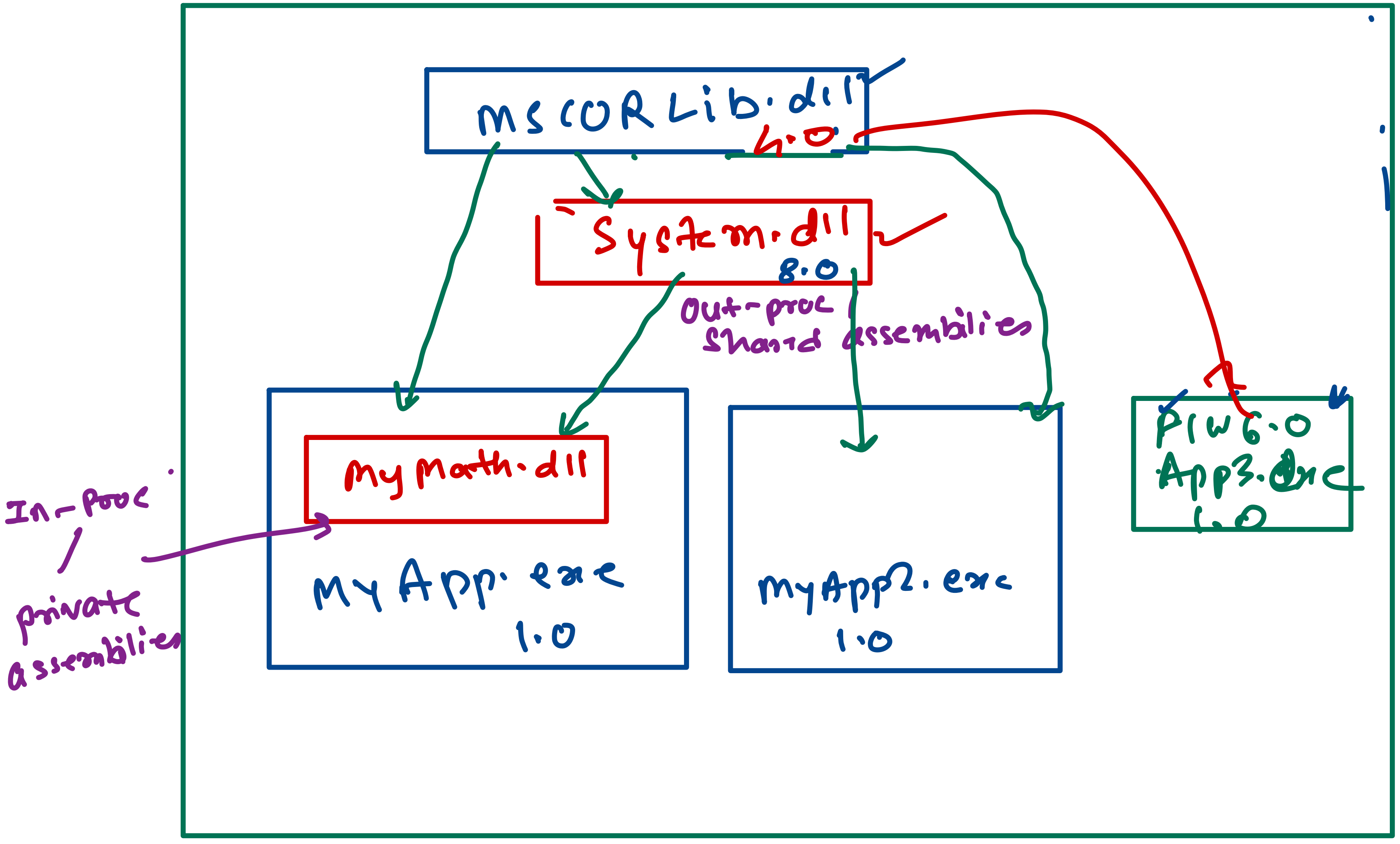
.Net Assembly . Structure.



→ version, company, security key
Public key.

→ audio, video, PDF, images
static resources info.

→ namespace, class, static, interface, abstract, virtual, override, events and delegates, constructors, properties, attributes etc---



MSCORLib.dll
4.0
System.dll
8.0
Out-proc /
Shared assemblies
MyMath.dll
MyApp.exe
1.0
myApp2.exe
1.0
P1W6.0
App3.exe
1.0
In-Proc
/
private
assemblies

Global Assembly Cache (GAC)

:- .Net registry

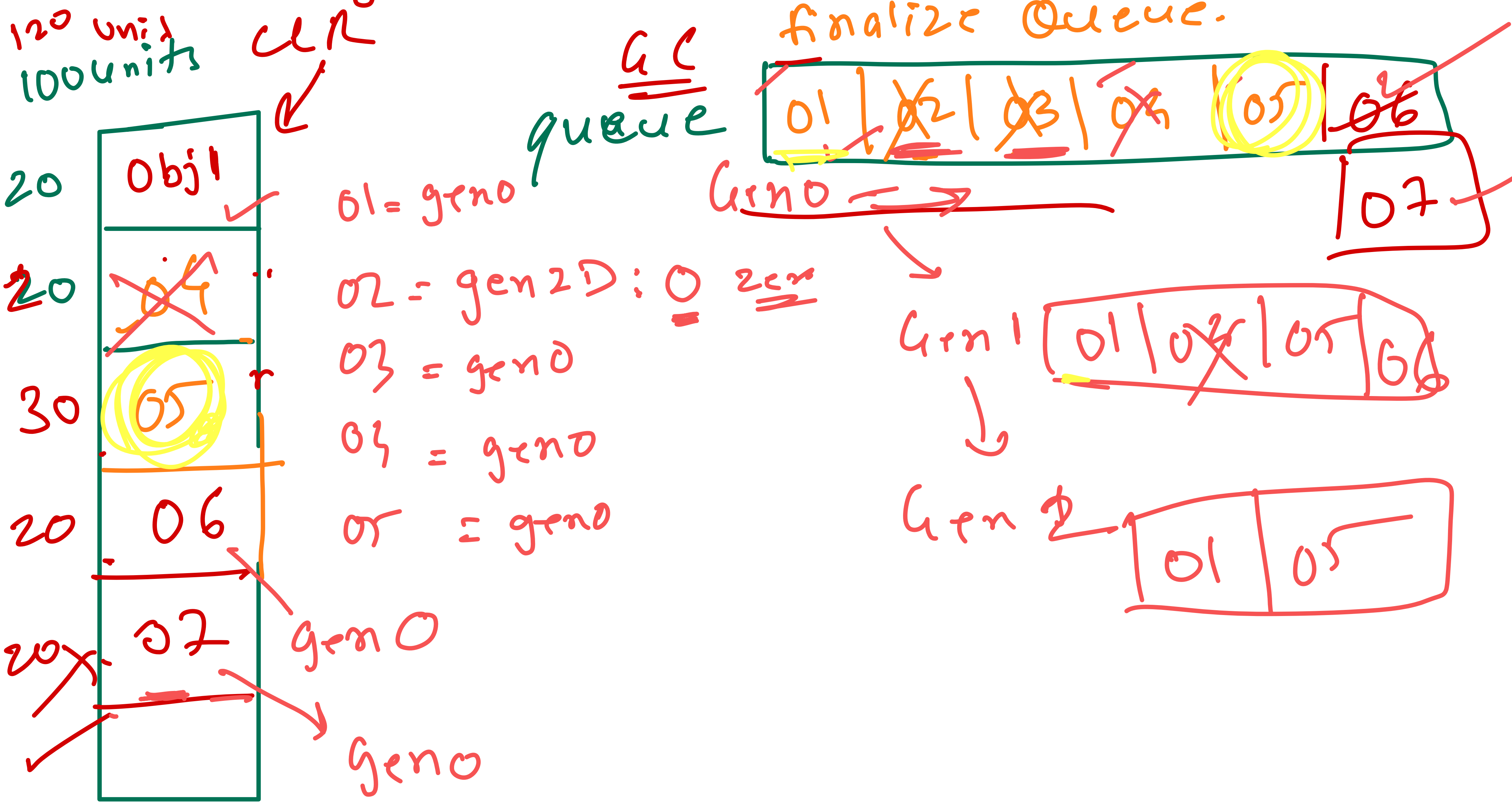
:- Dir → All flw → load.

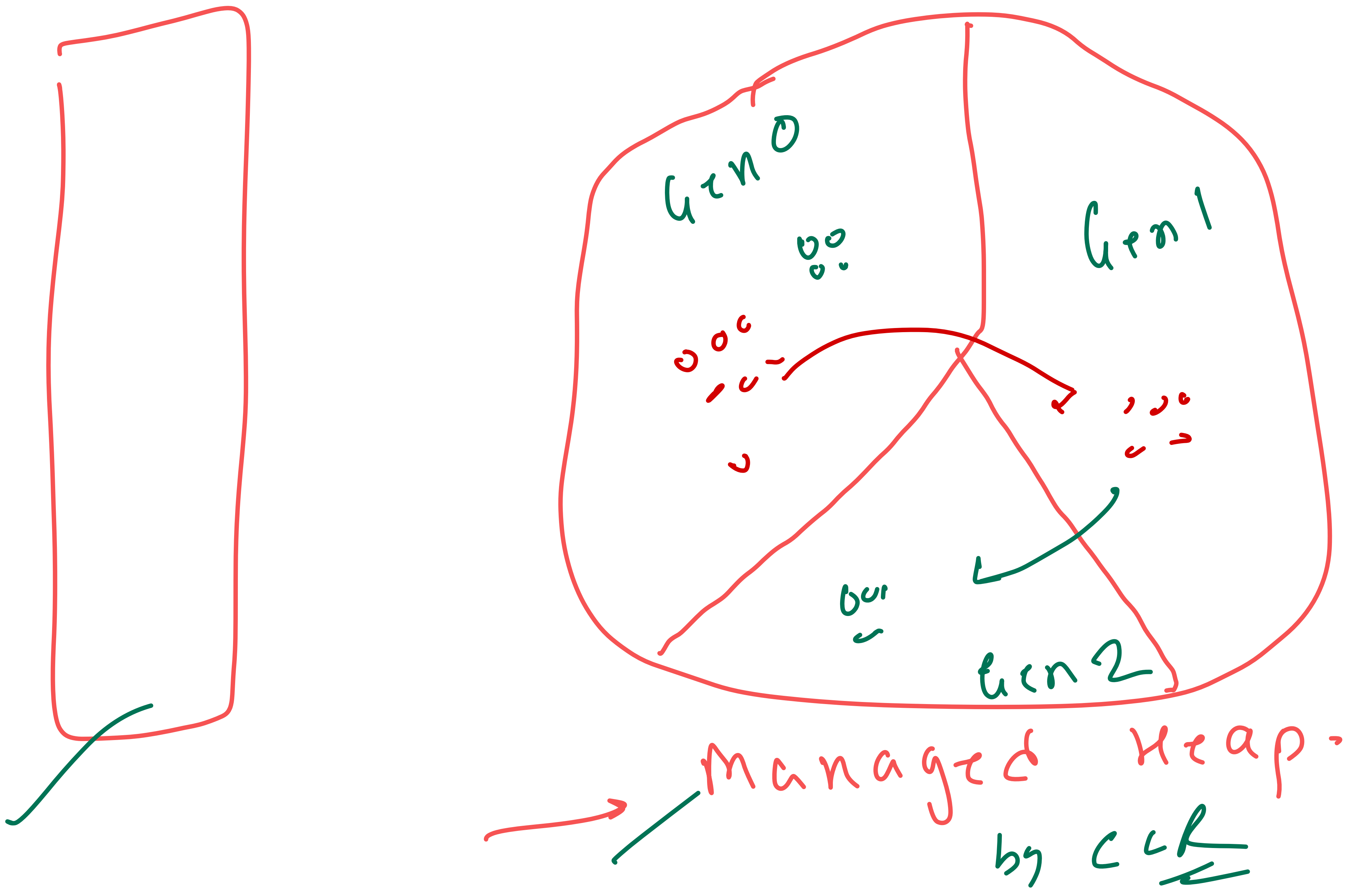
↓ path ≡ .

out-proc.

→ In-proc (dll) -
private copy is getting maintained / alongside / within same directory of your .exe file.

Garbage Collector : (GC)




Gen 0
Gen 1
Gen 2
Managed Heap.
by CLR

Do's and Don't about calling GC

1) Don't call GC by your own.

→ Generation promotions.

Gen0 → Gen1 → Gen 2

2) GC rarely cleans Gen1 and Gen2 sections of managed heap.

3) Donot write destructor (~class()) by your own.

→ GC ignores its deletion in the first iteration.

4) Do not declare large objects

→ greater than memory 85,000 + bytes

by birth these large objects

gets Id as gen 2.

Solution file

.sln → group of projects.

- → 1. csproj
  - ↳ program.cs
- → 2. csproj
  - ↳ program.cs
- → 3. csproj
  - ↳ program.cs

project. .net core

Output → projectDir/bin/debug/myapp.dll

/bin/debug/myapp.exe.