

C#.NET @ Sunbeam Infotech

Trainer: Nilesh Ghule

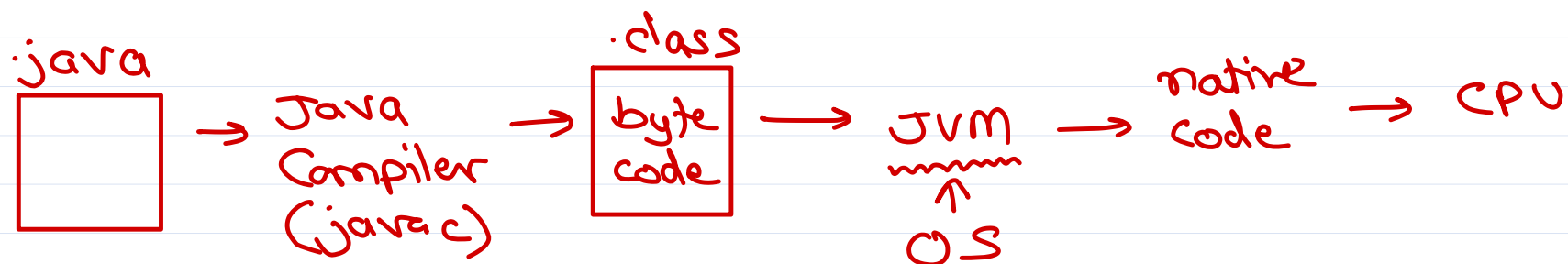


Agenda

- ① Java code compilation vs .Net lang code compilation
- ② C# code vs VB.Net code compilation
- ③ .Net framework & versions.
- ④ CTS, CLS, CLR (JIT, GC, ...)
- ⑤ Installations - on client machine vs on dev machine
- ⑥ Hello world appln (using Visual Studio) ↗ Console app
↘ Class Lib
- ⑦ .Net assembly (structure, exe execution) → ildasm,
SharpLab.io, ...
- ⑧ Class libs & Namespaces
- ⑨ C# classes (fields, methods, properties, constructor, destructor, ...)

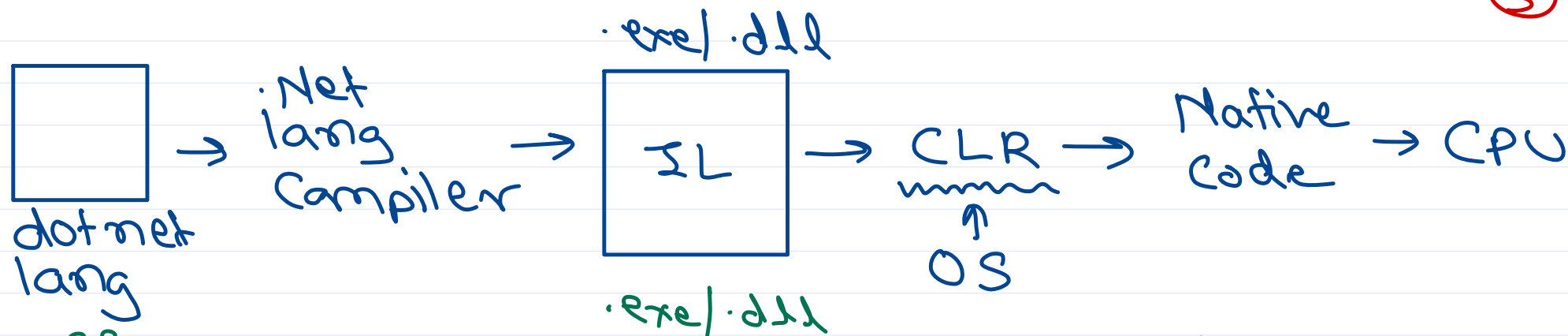


.Net - Compilation



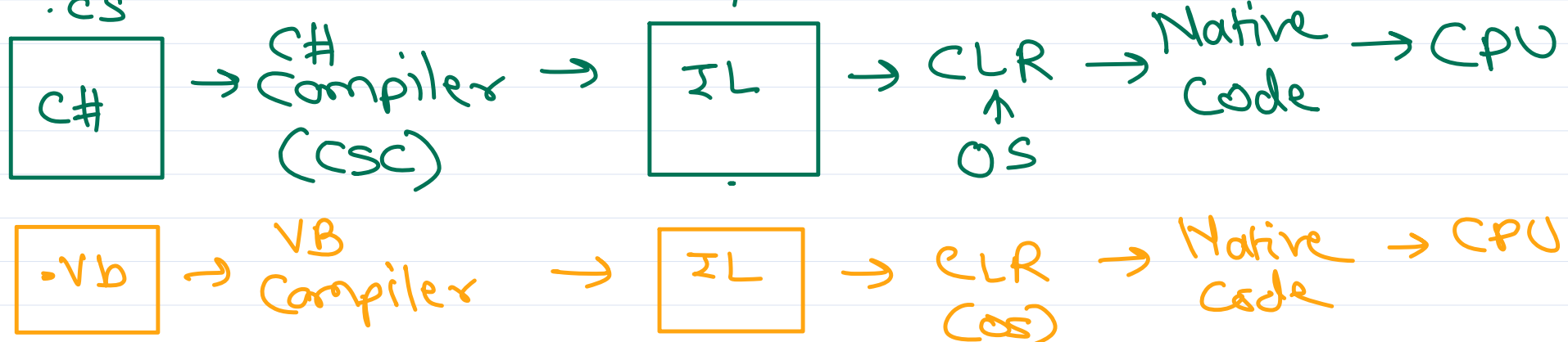
JVM based lang

- ① Java
- ② Kotlin
- ③ Scala

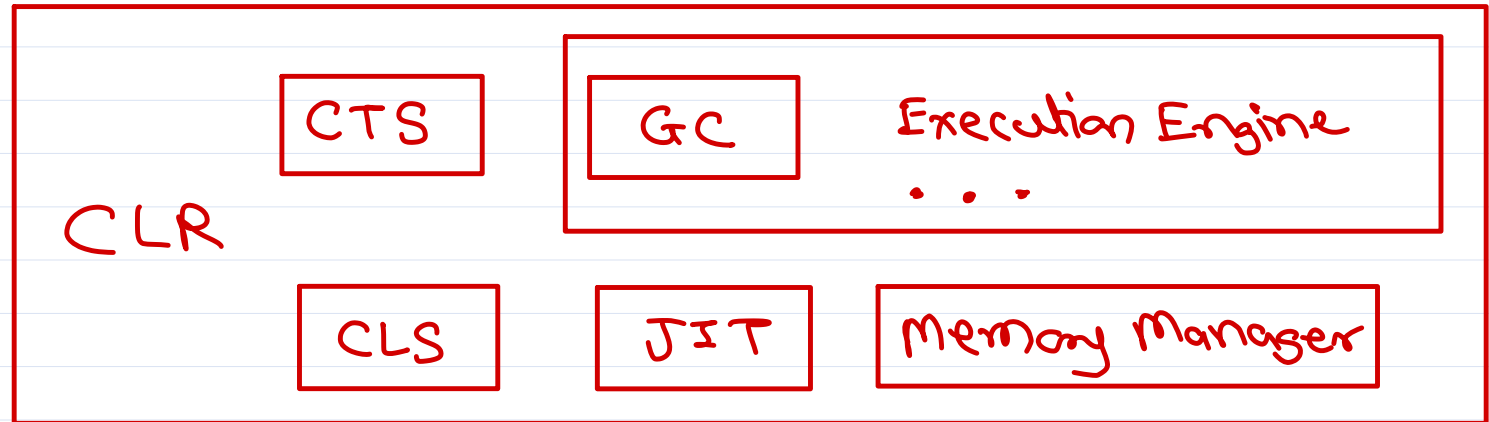
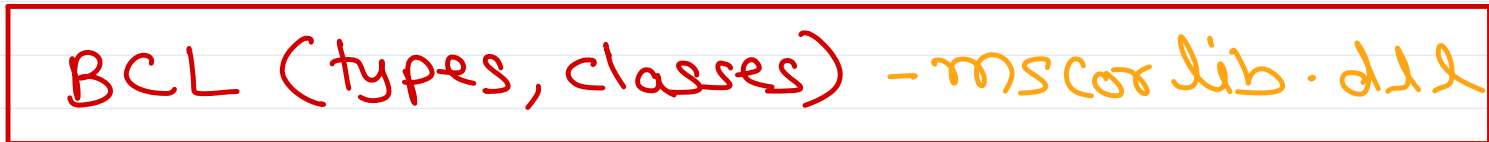
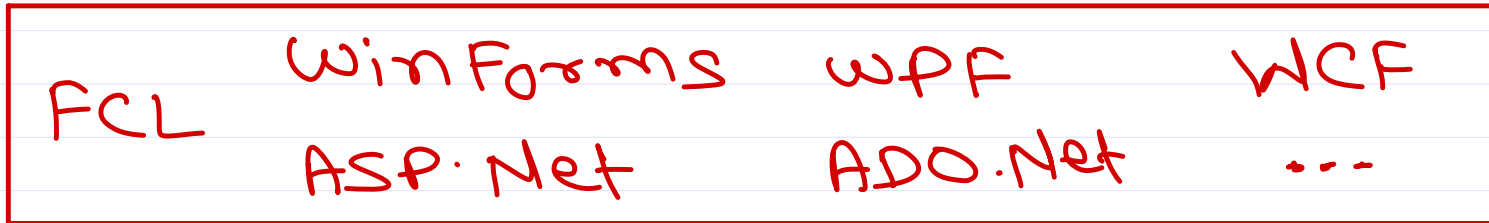
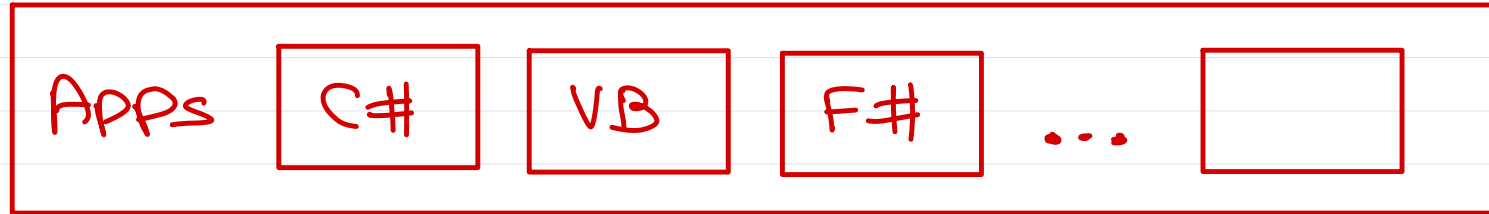


.Net lang

- ① C# (*)
- ② F#
- ③ VB.Net
- ④ J# x
- ⑤ Python
- ⑥ ...



.Net Framework



client machine Setup

- Net framework
 - ↳ CLR + BCL + FCL
- ## Dev machine Setup

.Net framework SDK

- Net framework
 - ↳ CLR + BCL + FCL
- + Dev tools
 - ↳ Lang Compilers + Debugger
 - + tools (ildasm, CAs, ...)
- + IDE (RAD tools) + ...

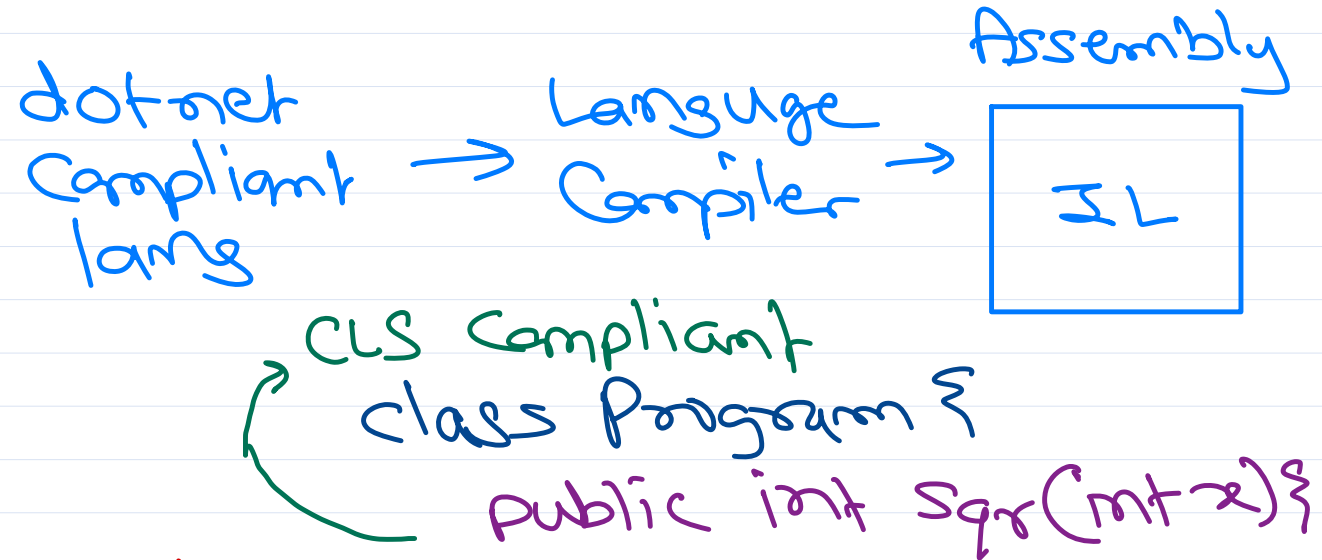
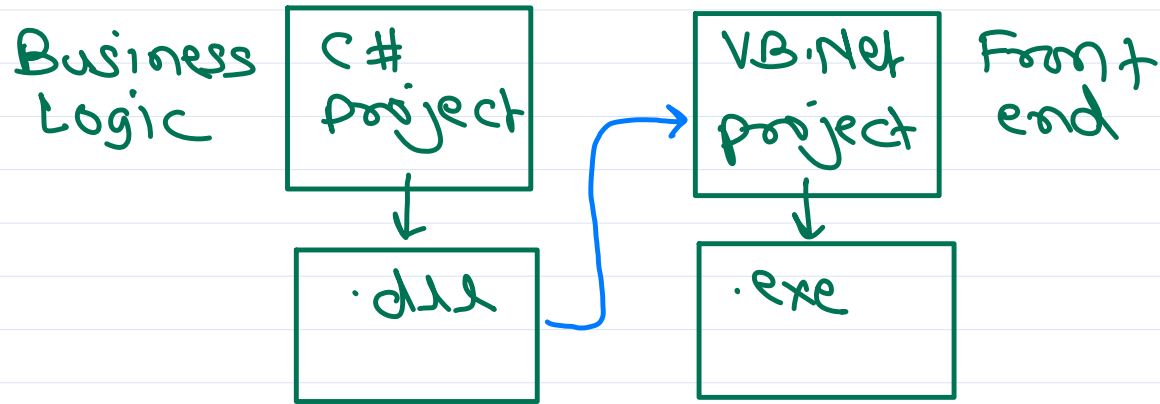
Visual Studio 2022

- Community
- Professional
- Enterprise



CLR — CTS & CLS

* Language interoperability



* Languages must follow certain rules.

→ .Net Compliant lang → Lang Compiler

→ lang rules → CLS

→ objects allocated using new

→ only signed types

→ stack based execution

→ ...

```
public int Sqr(int x) {  
return x * x;  
}
```

CLS Non-compliant



CLR — CTS & CLS

Common Type System → Types in IL → understood by CLR

CTS types

→ Value Types → represent value

Int32 a = 123; a 123

① Int16 ⑥ Decimal

② Int32 ⑦ Boolean

③ Int64 ⑧ Char

④ Single

⑤ Double

* User defined → Structure
→ Enum

* Typically on Stack (locals)

* If part/field in some obj
→ on Heap.

→ Reference Types → represent ref / addr.

① Class (e.g. String, ...)

② Interface

③ Array

④ delegate

⑤ object

String s = "DAC";

s
1000

"DAC"

1000

String obj



* JIT

- ✓ IL code → JIT → Native code
- ✓ Does conversion Method by Method.
- ✓ Frequently called methods - native code - Cached.
- ✓ JIT types
 - ① Standard JIT (Part of CLR) - all devices.
 - ② Economy JIT - on small devices (phones, embedded).
 - .Net Compact framework - Outdated.
 - ③ Pre-JIT (AOT - Ahead of Time):
 - all IL code → pre-JIT → Native code
 - ngen.exe ↓
 - Before Execution ↘



* GC → auto deallocate
unreferenced objects

- Mark-Sweep-Compact algorithm.

* Execution Engine

- ↳ GC
- ↳ Security

* Memory Manager
↳ allocates memory for

- class loading
- Stack
- Cache
- ...

.Net Assemblies

File Extensions

① .exe

- executable file
- OS creates process when end user starts execution.

② .dll

- lib of classes (reuse)
- dependent executable
- loaded in calling process memory, when its classes accessed in that process.

Assembly Structure

PE header
Manifest
Metadata
IL code
Resources

→ CLR info, referenced assms info, versions, ...

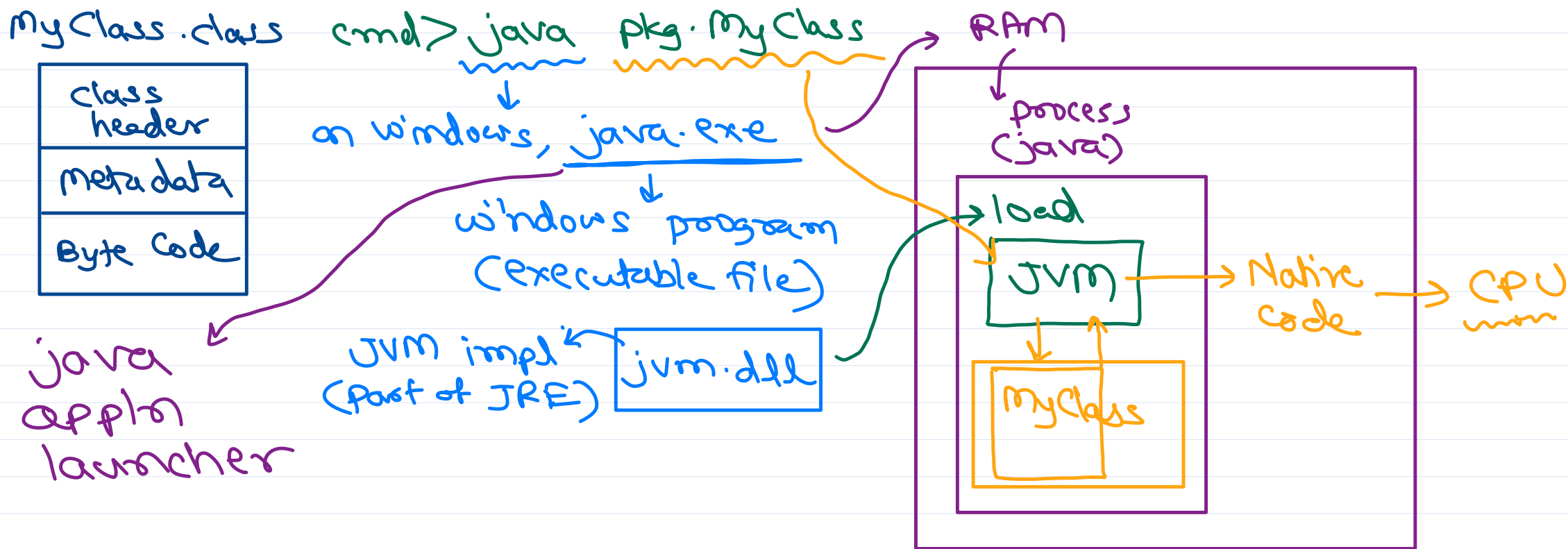
→ a.k.a. assembly metadata

→ type metadata - type name, fields, methods, ctors, ...

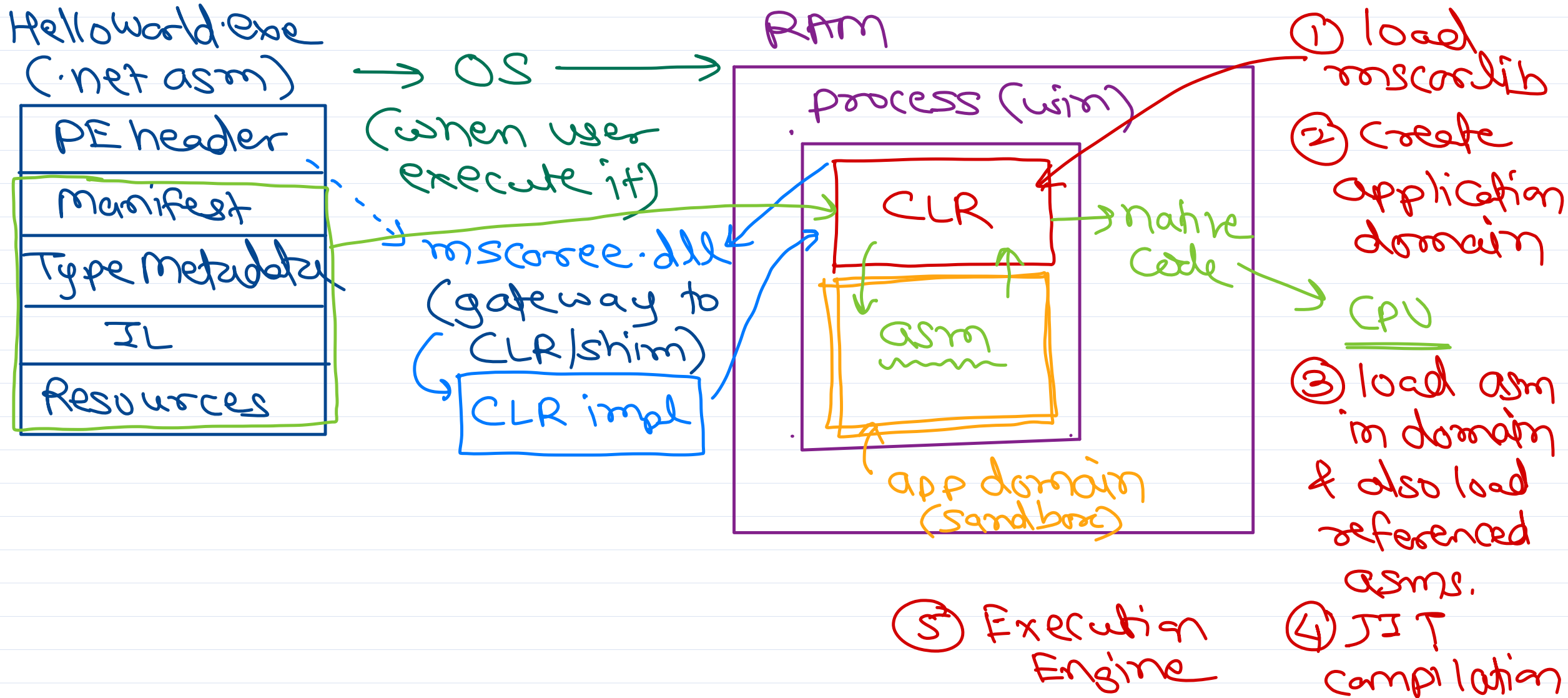
→ images, config files, ...



Java execution



.Net program execution





Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

