ISEL

Ambientes Virtuais de Execução

2021

Week 6 – meta-programming

Outline

- Remember – Reflection
- Meta-programming e.g. .net Emit
- Remember – Logger
- LoggerDynamic

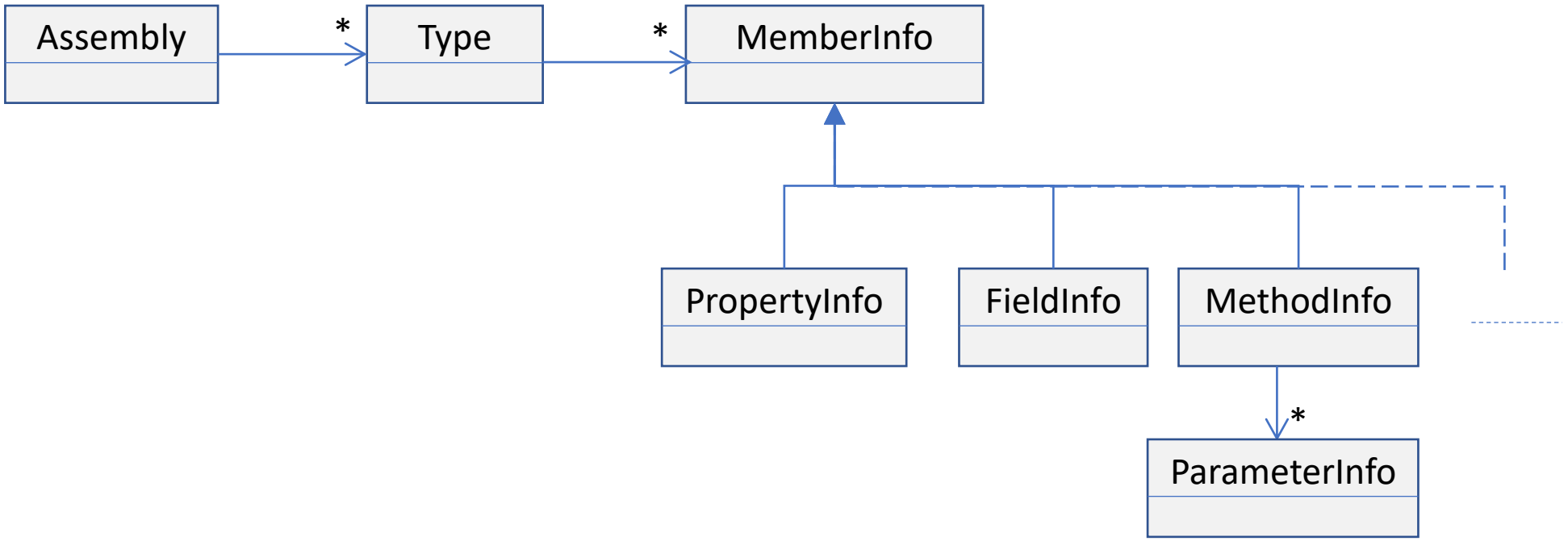
Remember - Reflection

Ability to **introspect**, the structure and behavior of a program at runtime.

Exemplos:

- .net - System.Reflection
  e.g. `foo.GetType().GetMethod("hello").Invoke(foo, null);`
- Java - java.lang.reflect
  e.g. `foo.getClass().getDeclaredMethod("hello").invoke(foo);`
- Javascript
  e.g. `foo['hello']()`
- ...

.net - System.Reflection



What can we do? E.g. Logger, log4net, ildasm, javap, intellisense, unit tests fw (xunit), ORM, dependency injection (Spring), etc

What can we not do ?

> Modify or **create new Types dynamically** (at runtime).

Outline

- Remember – Reflection
- Meta-programming e.g. .net System.Reflection.Emit
- Remember – Logger
- LoggerDynamic

Meta-programming

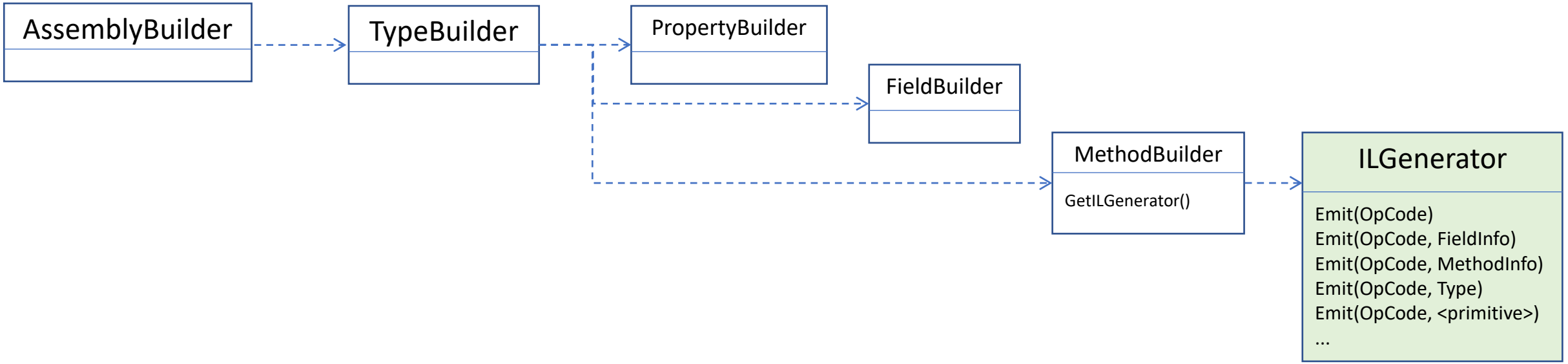
Ability to read, generate, transform or modify programs **dynamically** (at runtime).

Exemplos:

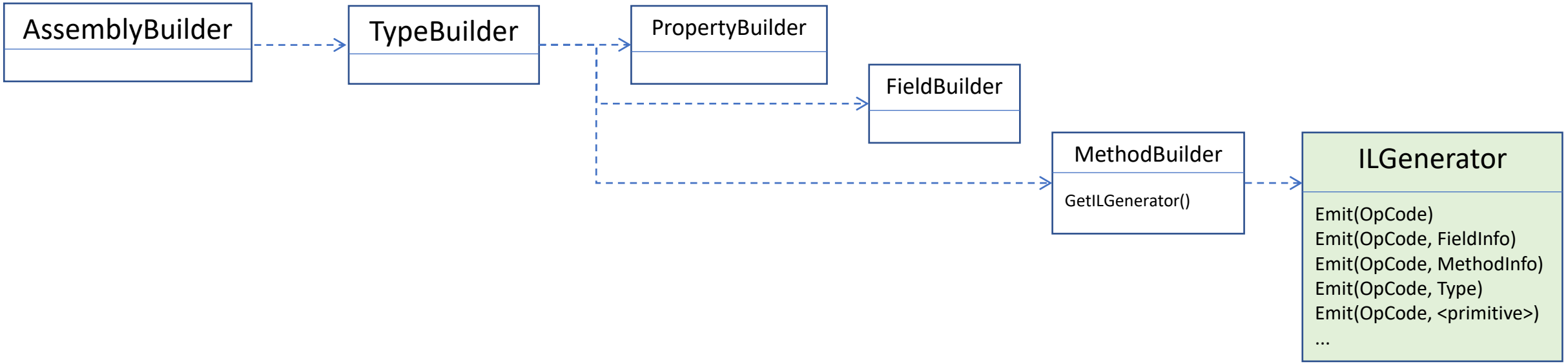
- .net - System.Reflection.Emit
- Java – third parties, e.g. Javassist (Java bytecode engineering toolkit), ASM (Java bytecode manipulation and analysis framework), and others.
- Javascript

  e.g. `foo.hello = function() { console.log('hello') }`

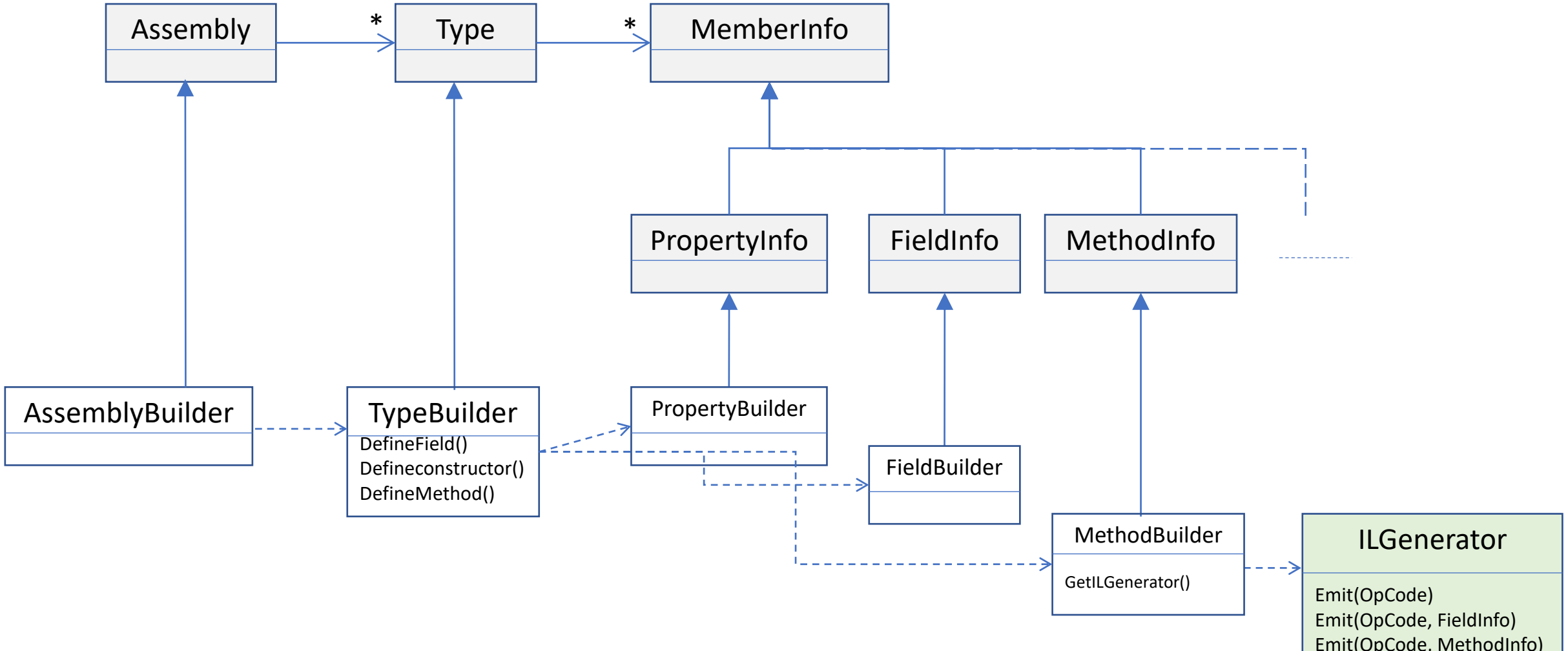
.net - System.Reflection.Emit



.net - System.Reflection.Emit



.net - System.Reflection and Emit



Using System.Reflection.Emit

- dotnet add package System.Reflection.Emit --version 4.3.0
- ~~AppDomain.CurrentDomain.DefineDynamicAssembly~~

```
    => AssemblyBuilder.DefineDynamicAssembly
```

- ~~ab.DefineDynamicModule(string, string)~~

```
    => ab.DefineDynamicModule(string)
```

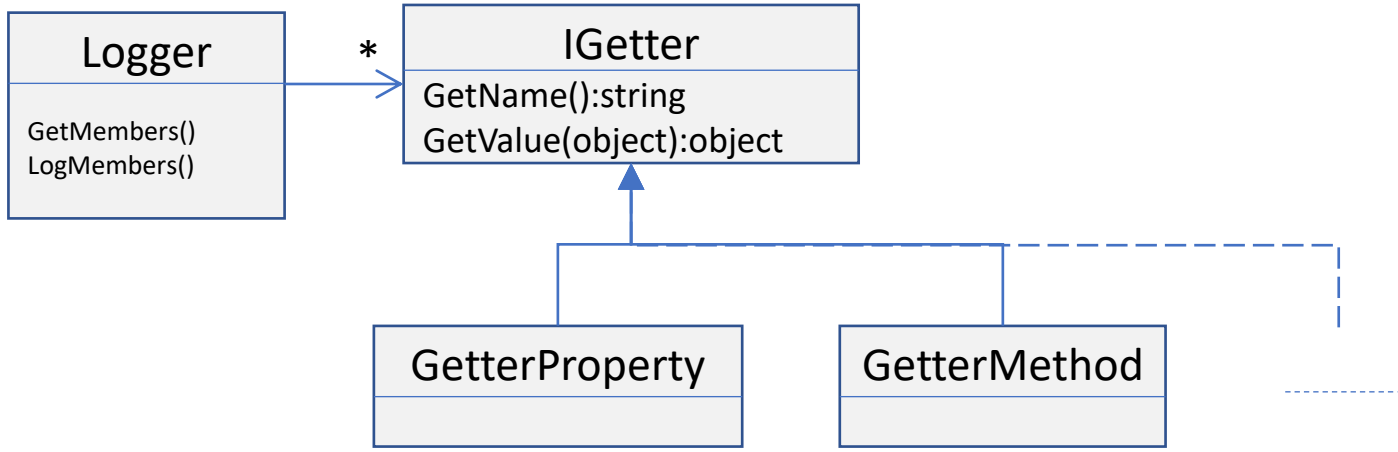
AddInterfaceImplementation

1. tb.AddInterfaceImplementation(typeof(IMultiplies));
2. MethodBuilder meth = tb.DefineMethod(
   "Mul",
   MethodAttributes.Public | MethodAttributes.Virtual,
3. tb.DefineMethodOverride(
   meth,
   typeof(*IMultiplies*).GetMethod("Mul"));

Outline

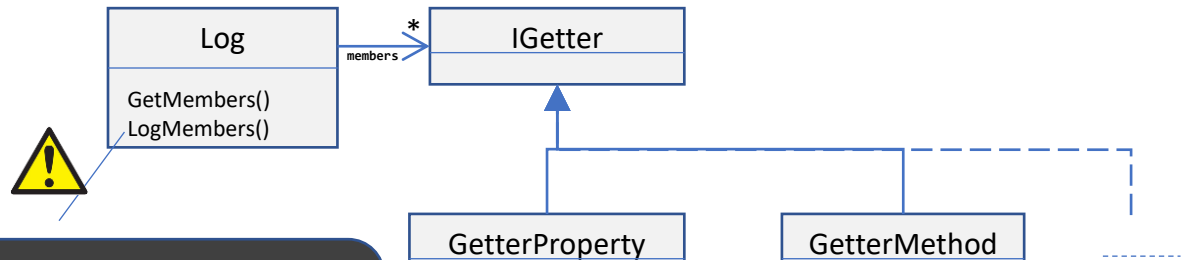
- Remember – Reflection
- Meta-programming e.g. .net Emit
- Remember – Logger via Reflect
- LoggerDynamic

Remember – Logger via Reflect

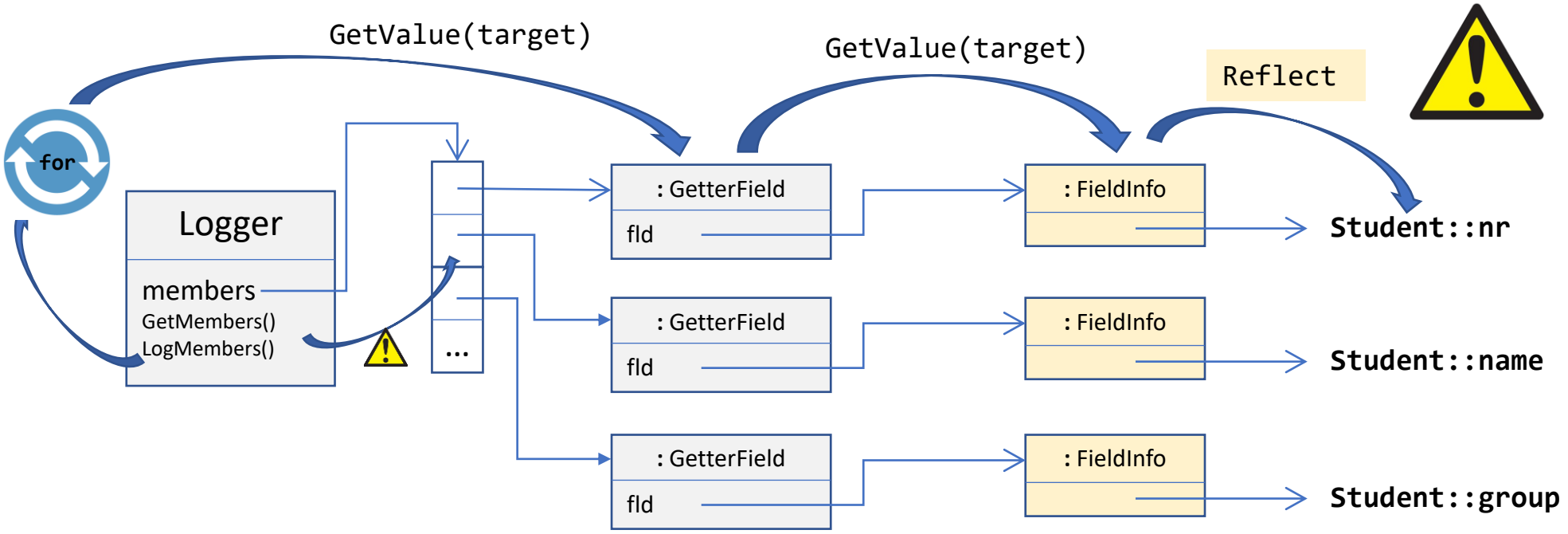


Logger e.g. for Student

```
public class Student {
    public readonly int nr;
    public readonly string name:
    public readonly int group;
    public readonly
string githubId;
}
```




```
foreach (IGetter m in ms) {
  ...
  str.Append(m.GetValue(target));
  ...
}
```

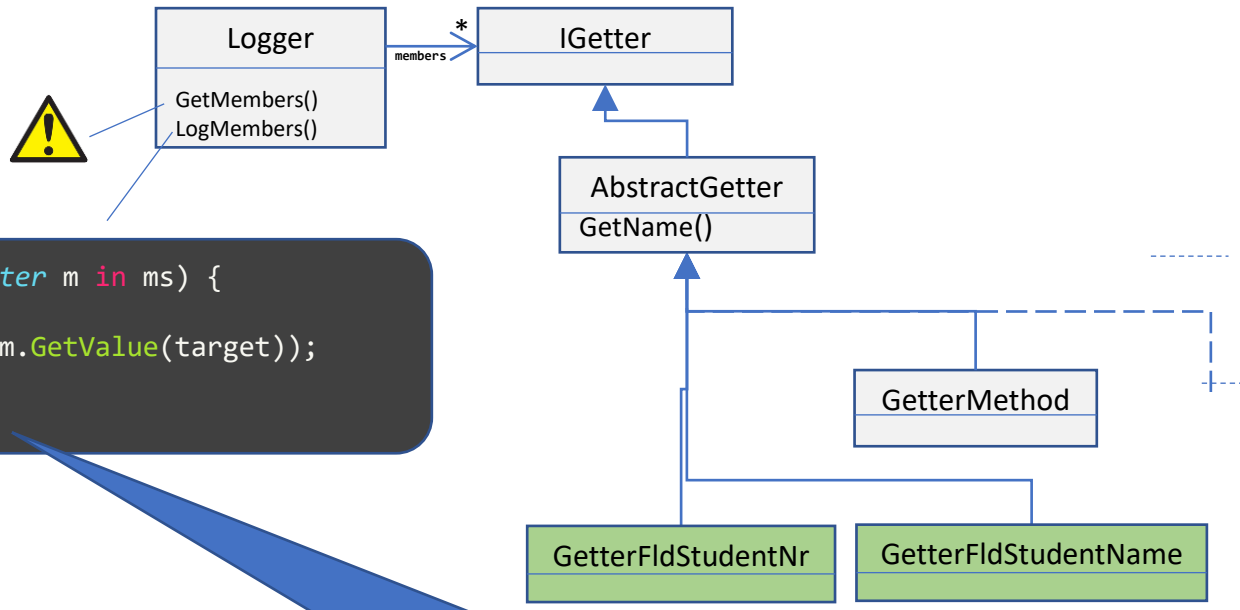


Outline

- Remember – Reflection
- Meta-programming e.g. .net Emit
- Remember – Logger
- LoggerDynamic

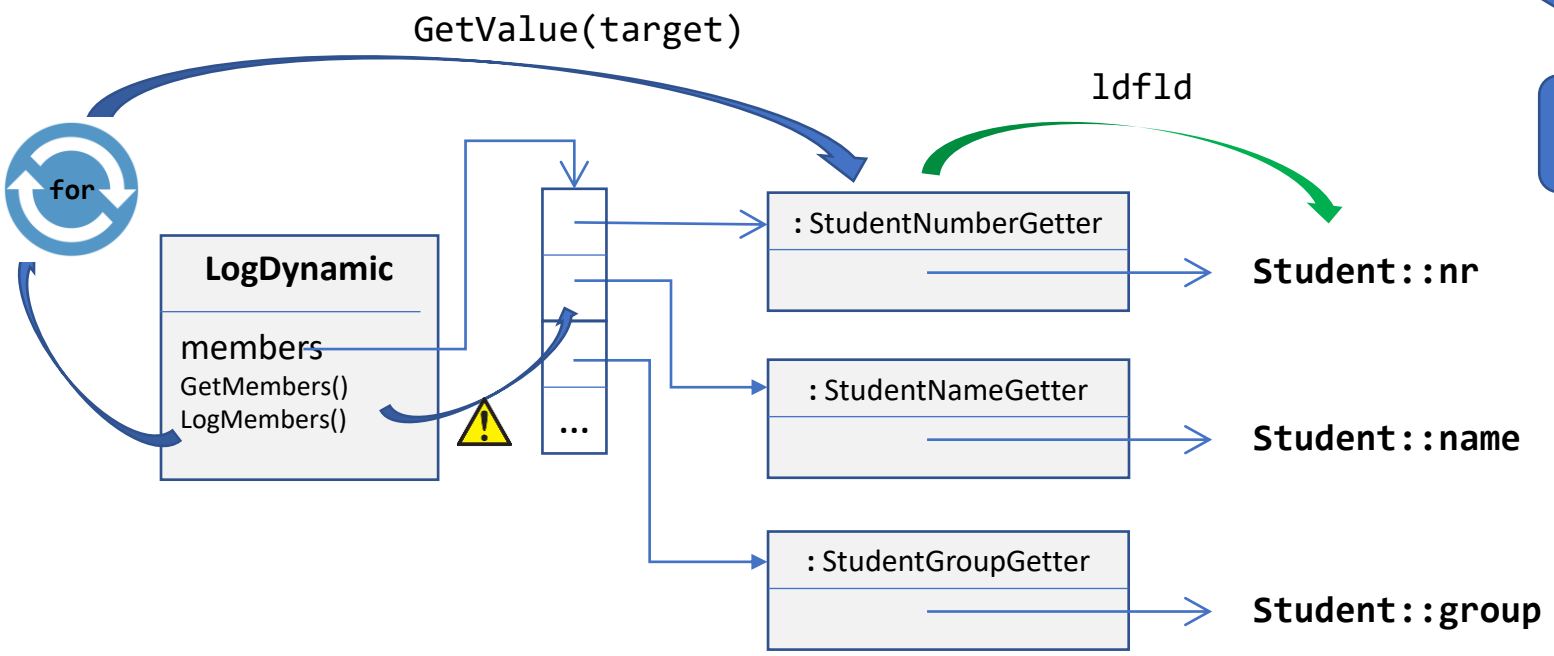
Logger e.g. for Student

```
public class Student {
    public readonly int nr;
    public readonly string name:
    public readonly int group;
    public readonly
string githubId;
}
```

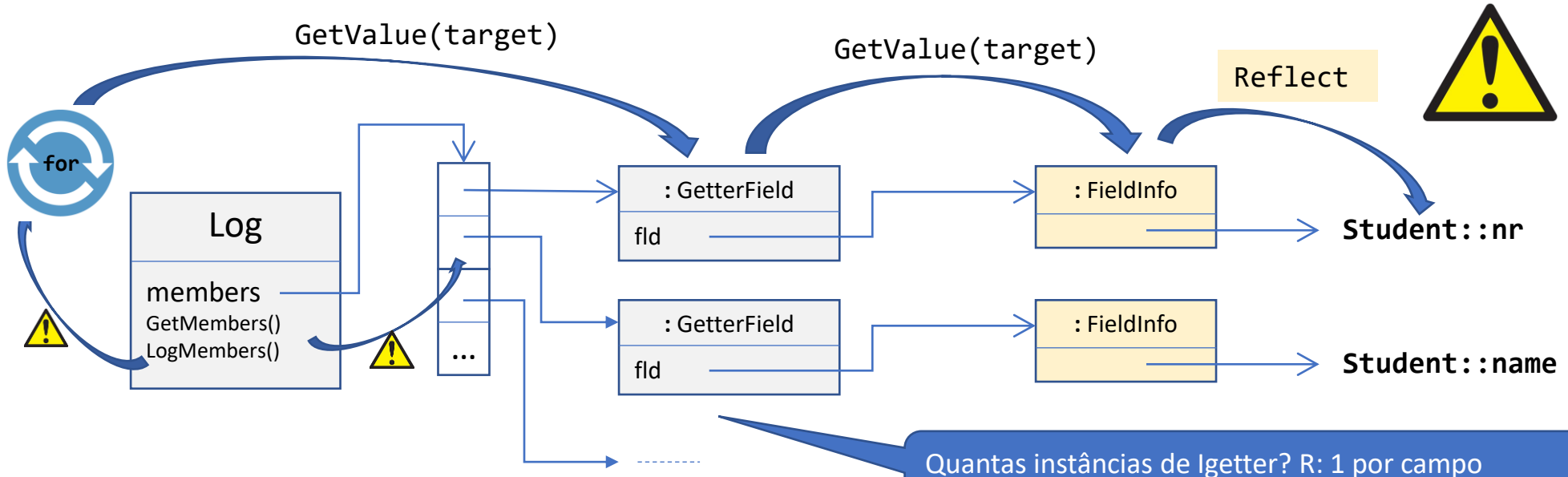



```
foreach (IGetter m in ms) {
  ...
  str.Append(m.GetValue(target));
  ...
}
```

Goal: Suppress the use of Reflection



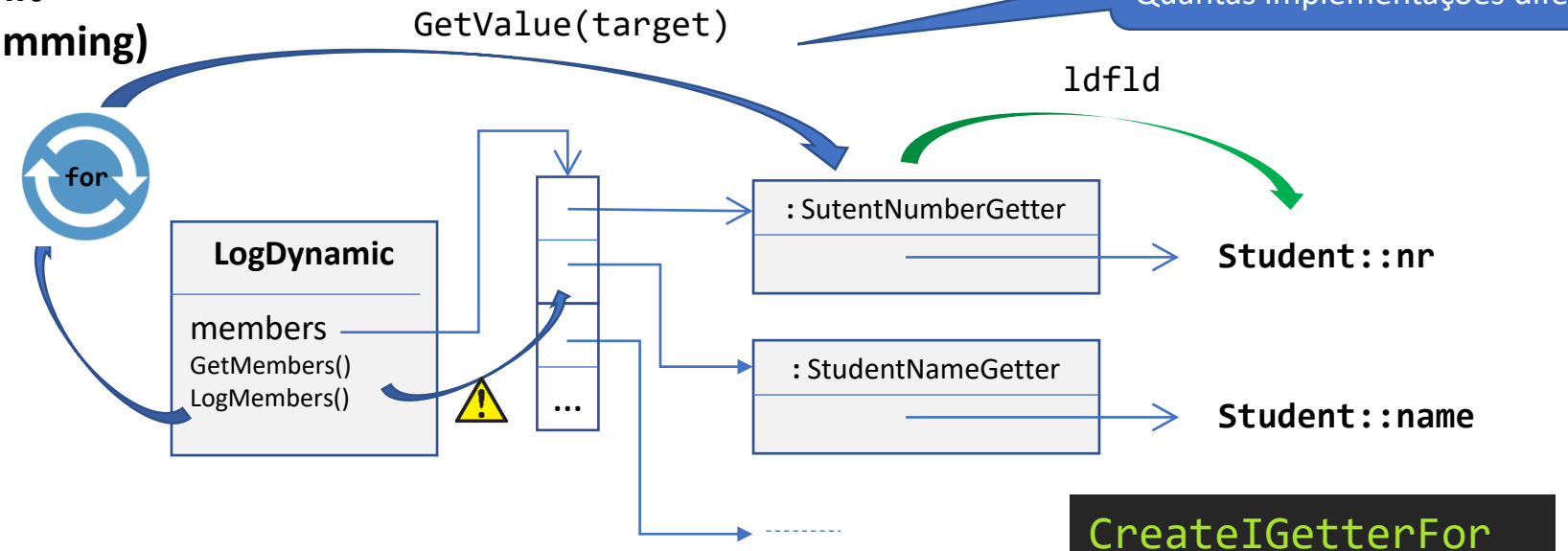
V1
Reflection



Quantas instâncias de Igetter? R: 1 por campo
Quantas implementações diferentes de Igetter? R: 1

V2
Reflection.Emit
(meta-programming)

Quantas instâncias de Igetter? R: 1 por campo
Quantas implementações diferentes de Igetter? R: 1 por **campo**




```
CreateIGetterFor
```

LogDynamic

```
foreach (IGetter m in ms) {
  ...
  str.Append(m.GetValue(target));
  ...
}
```