ISEL

Ambientes Virtuais de Execução

2020

Outline

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

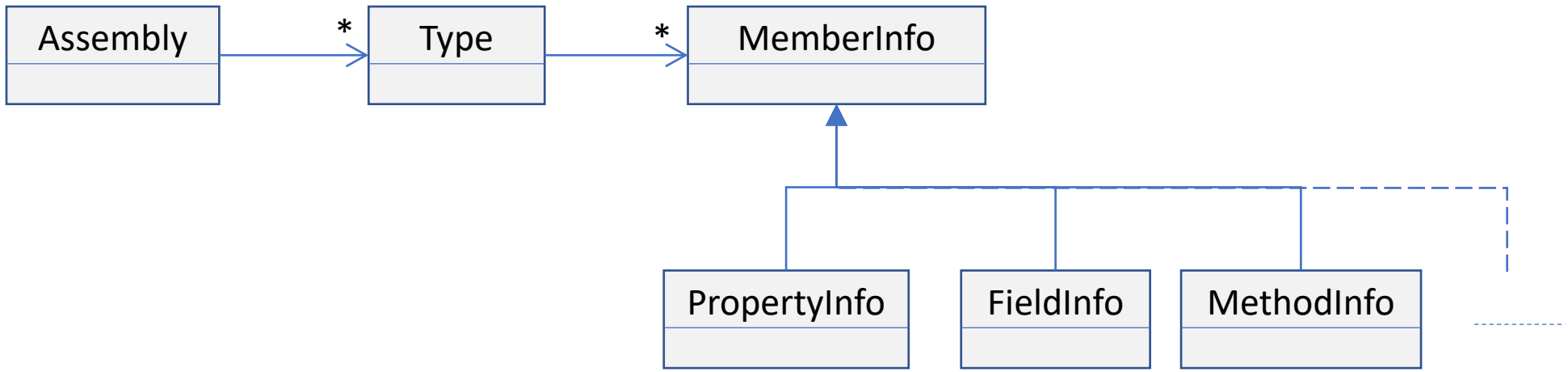
Remember - Reflection

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 we can not do ?

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

Outline

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

Meta-programming

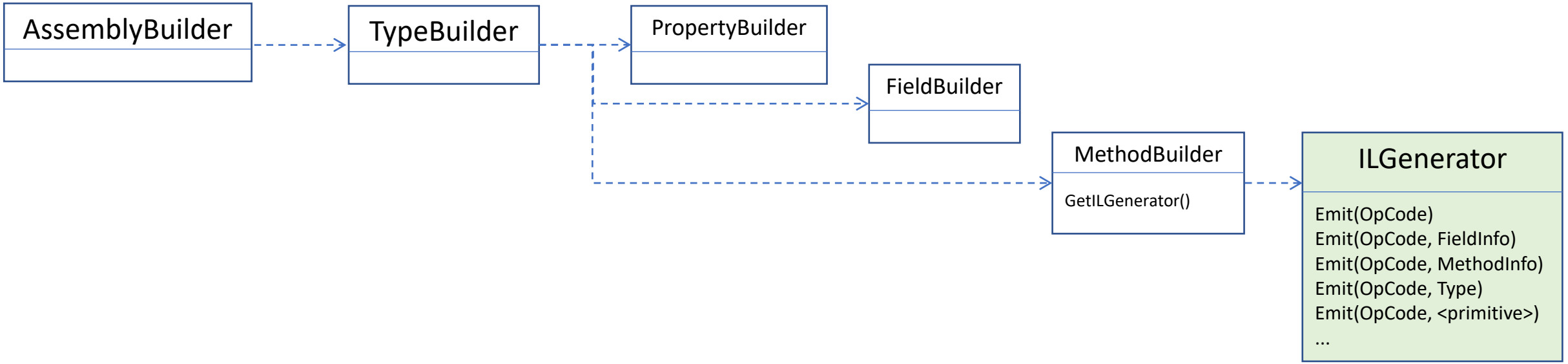
Ability to read, generate, transform or modify programs **dynamically** (at runtime).

Exemplos:

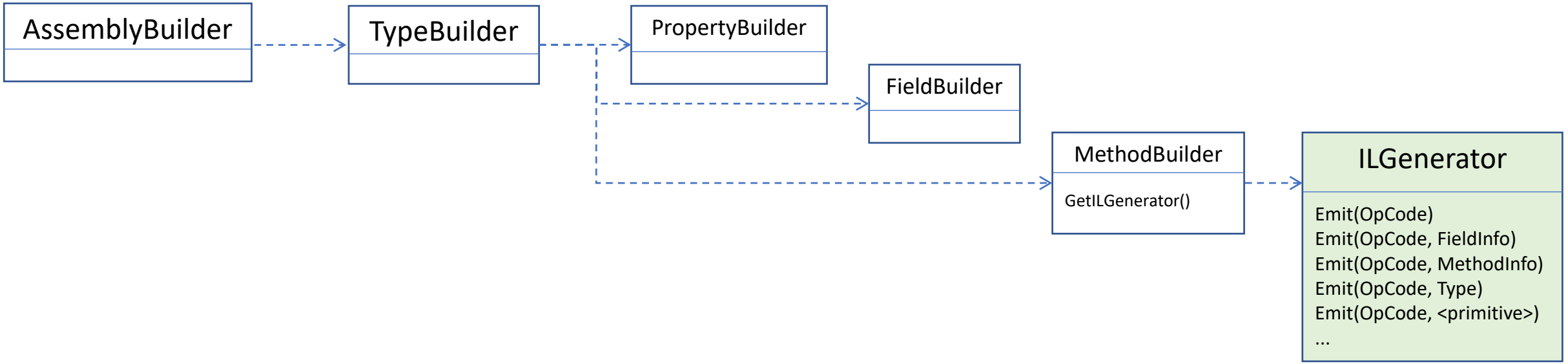
- .net - System.Reflection.Emit
- Java – third parties, e.g. Javassist (Java bytecode engineering toolkit), ASM (Java bytecode manipulation and analysis framework), and others.
- Javascript

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

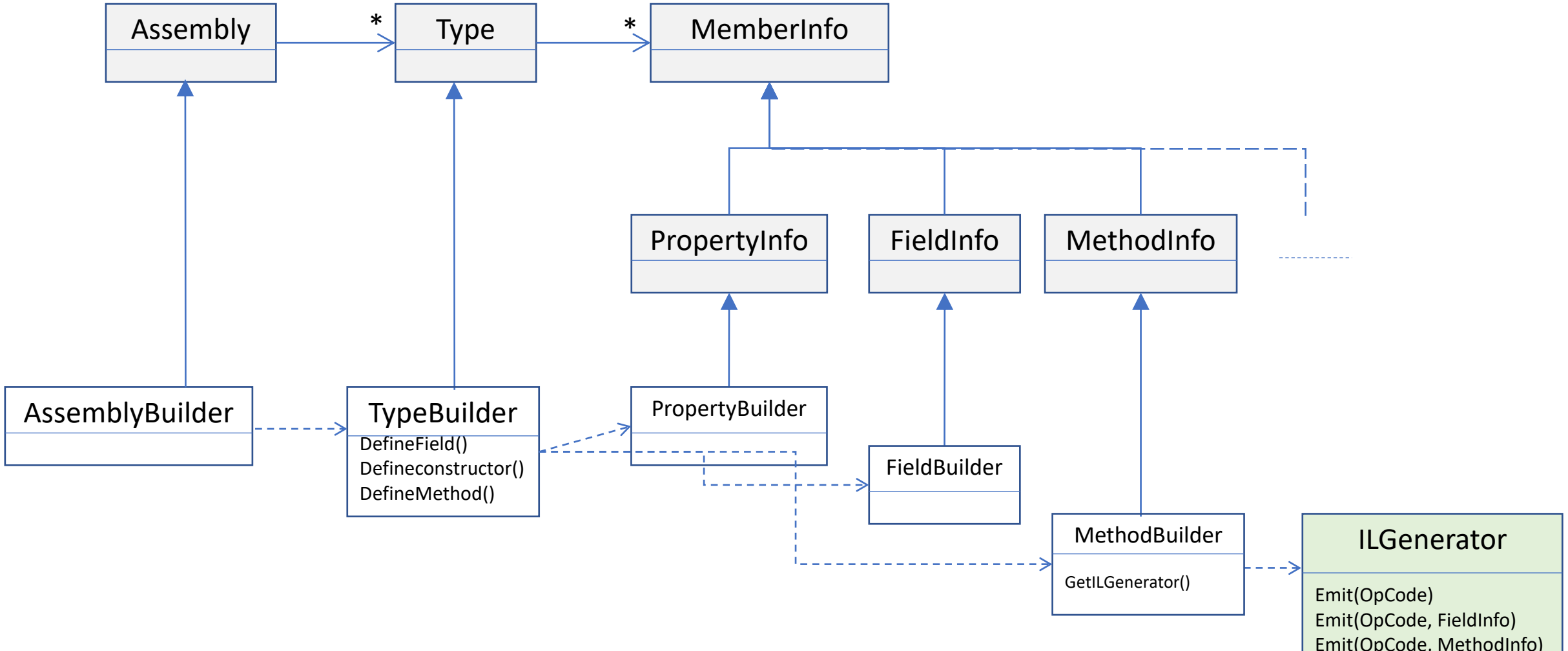
.net - System.Reflection.Emit



.net - System.Reflection.Emit



.net - System.Reflection and Emit



AddInterfaceImplementation

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

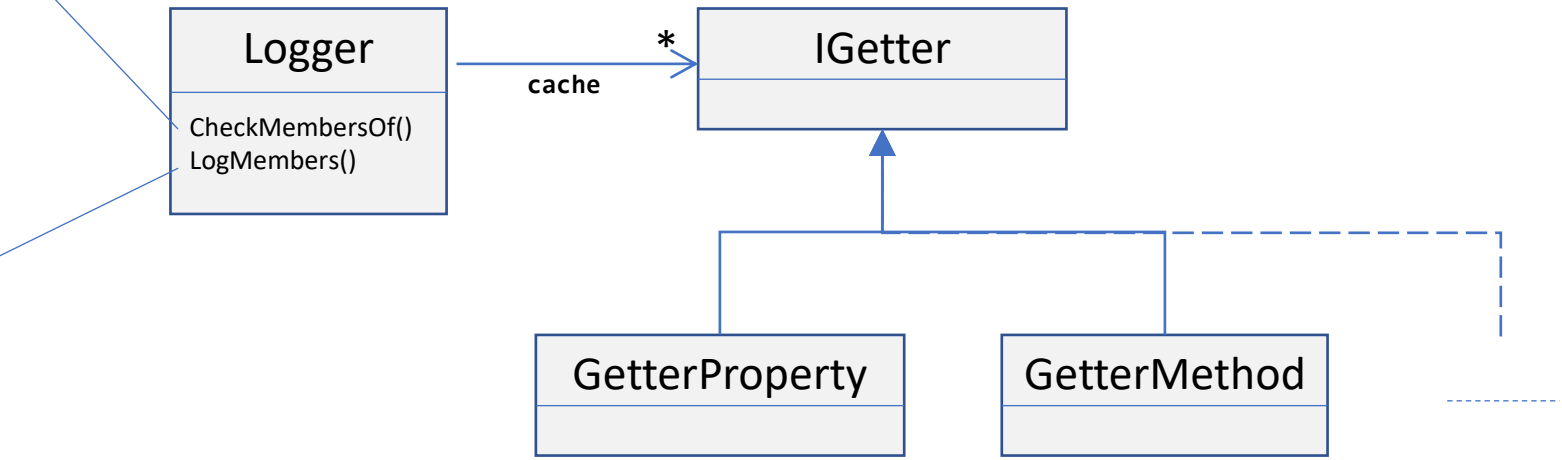
Outline

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

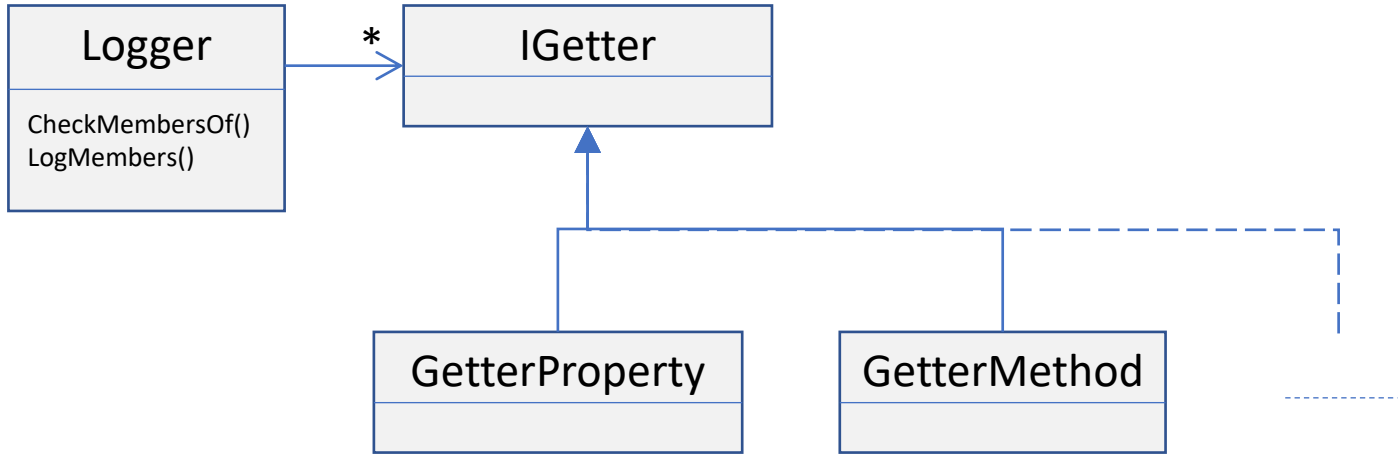
Logger

```
static List<IGetter> CheckMembersOf(Type klass) {
  ...
  foreach (FieldInfo f in klass.GetFields()) {
    if (CheckToLog(f)) {
      ...
  foreach (PropertyInfo p in klass.GetProperties()) {
    if (CheckToLog(p)) {
      ...
  ...
```

```
static void LogMembers(Type klass, object target) {
    List<IGetter> ms = CheckMembersOf(klass);
    foreach (IGetter m in ms) {
        Console.Write(" " + m.GetName());
        Console.Write(": ");
        Console.Write(m.Call(target));
        Console.Write(",");
    }
}
```

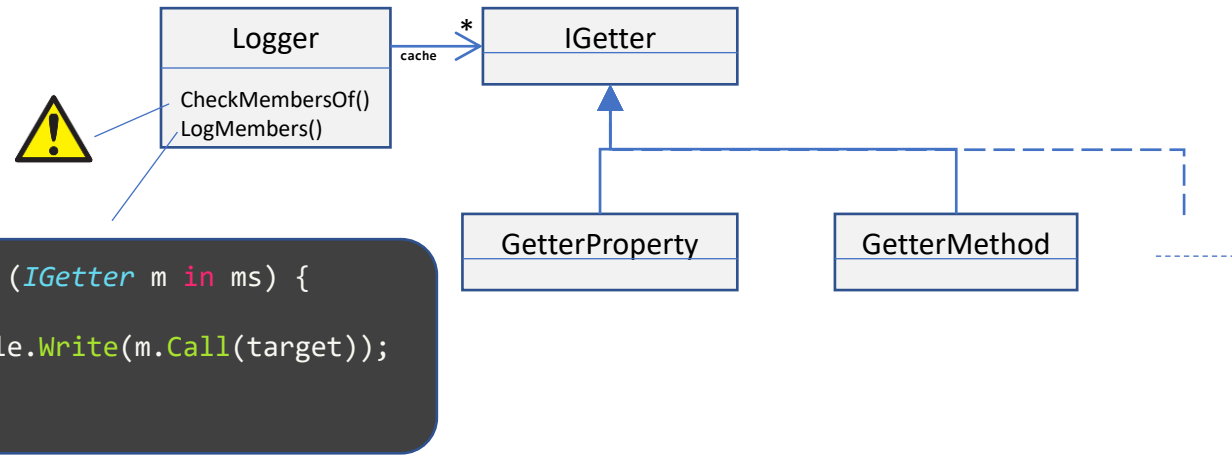


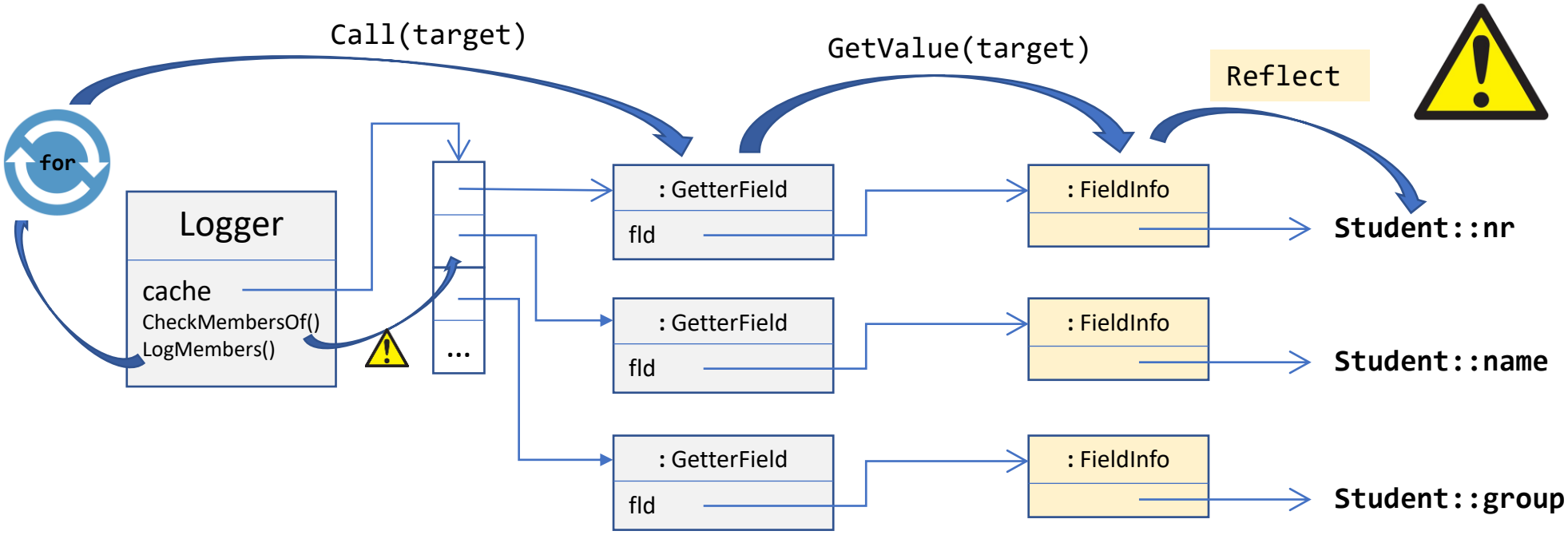
Logger


Logger
CheckMembersOf()
LogMembers()
*
IGetter
GetterProperty
GetterMethod

Logger e.g. for Student

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



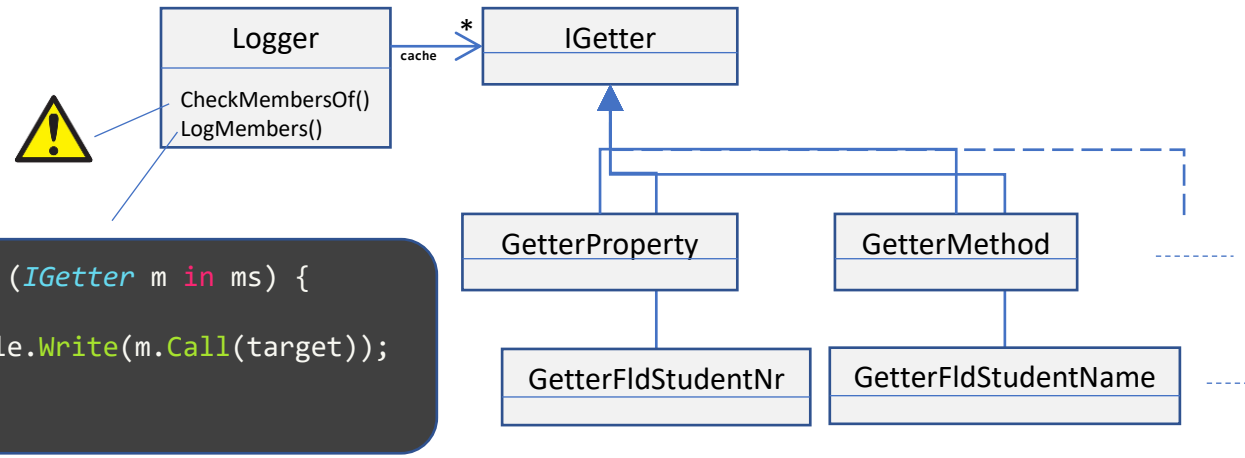


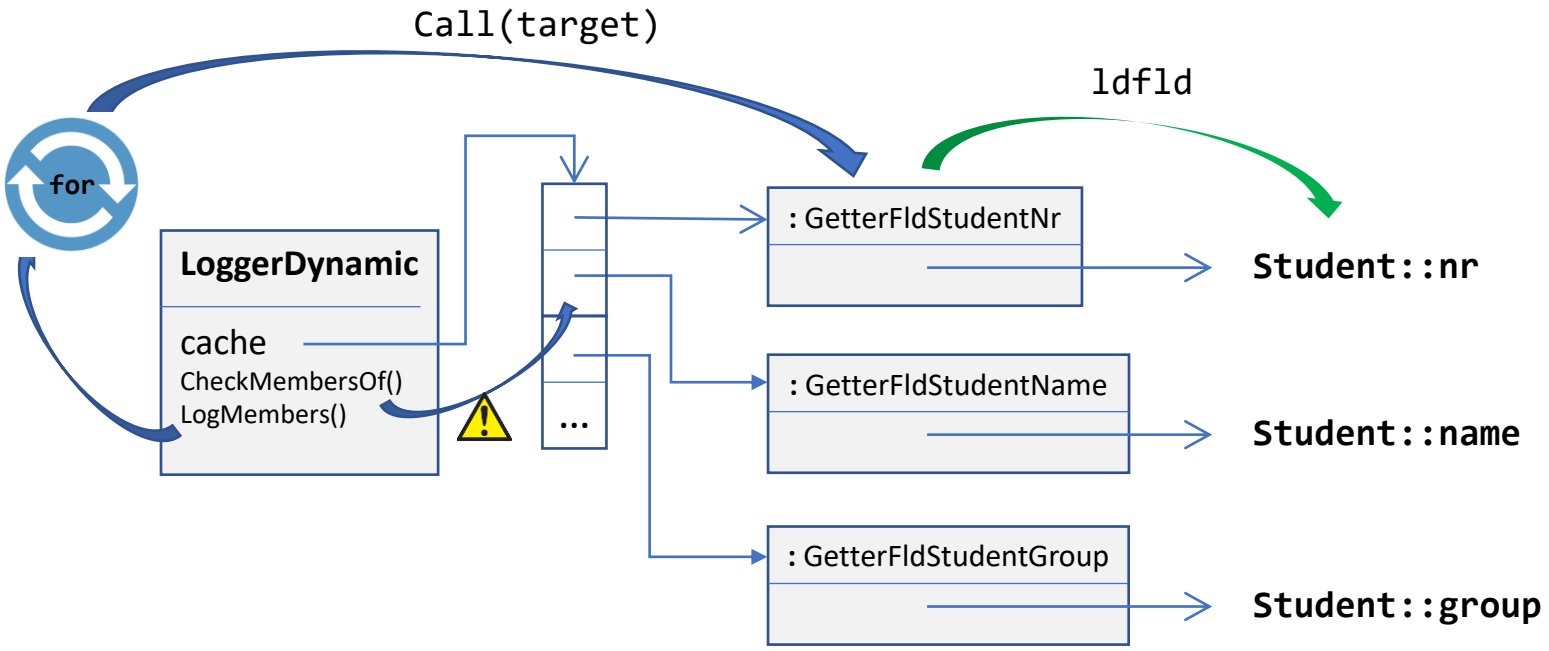
Outline

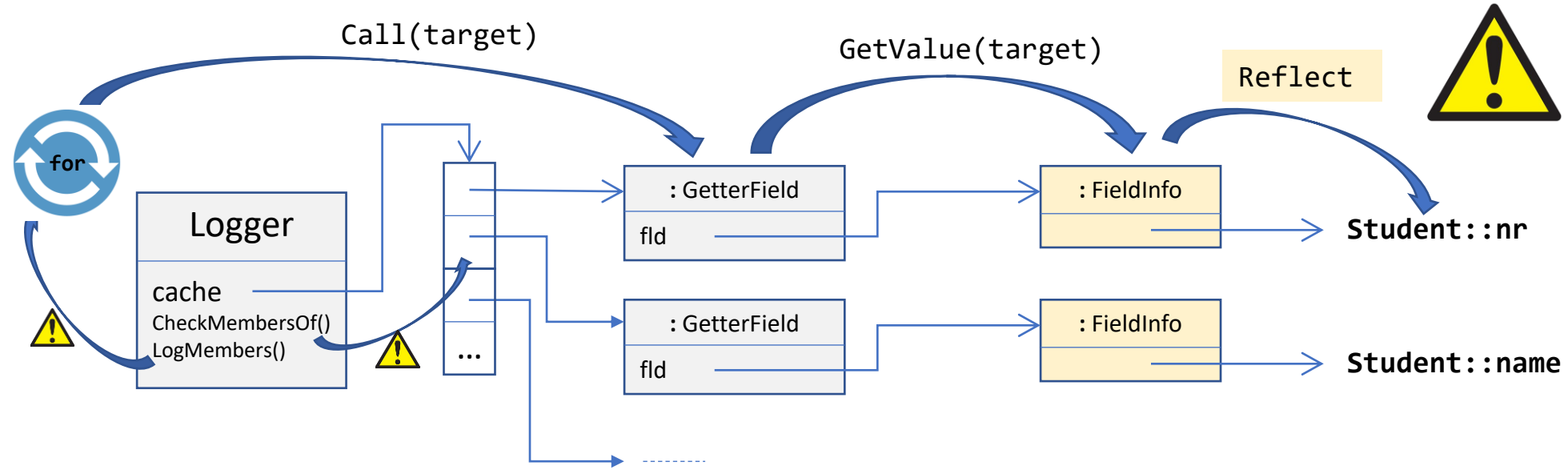
- Remember – Reflection
- Meta-programming e.g. .net Emit
- Remember – Logger
- LoggerDynamic

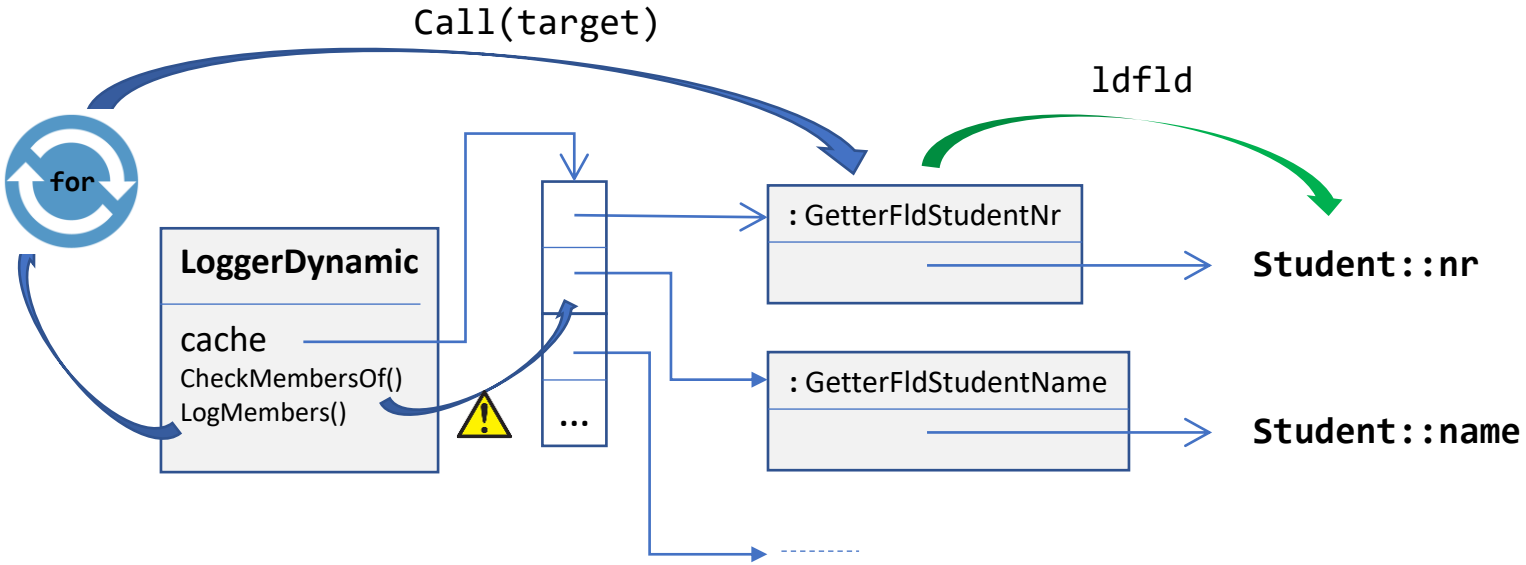
Logger e.g. for Student

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






Call(target)
GetValue(target)
Reflect
for
Logger
cache
CheckMembersOf()
LogMembers()
...
: GetterField
fld
: GetterField
fld
: FieldInfo
: FieldInfo
Student::nr
Student::name


Call(target)
ldfld
for
LoggerDynamic
cache
CheckMembersOf()
LogMembers()
...
: GetterFldStudentNr
: GetterFldStudentName
Student::nr
Student::name