Virtual Execution Environments

2021

Week 1

Reflection

Exemplos de utilização de metadata

- Intellisense no IDE

```
System.out.println("Module = " + p.
                    x : int                    java.awt.Point.x :
                    y : int
Debug               distance(Point2D pt) : double
```

- Compilador

```
C:\Windows\System32\cmd.exe

:\ISEL\pg5 ave - 2020-2021 - 2º sem\aulas-41d\aula01-jvm-i
pp.java:4: error: cannot find symbol
      System.out.println("Module = " + p.getModl());
                                         ^
 symbol:   method getModl()
 location: variable p of type Point
 error
```

- Visualizar no ILDASM
- Leitura em tempo de execução, i.e. Reflexão
- Testes unitários

```
C:\Windows\System32\cmd.exe

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.

Passed!  - Failed:     0, Passed:     1, Skipped:     0, Total:     1, Duration: 3 ms - Logger.Tests.dll (net5.0)

D:\ISEL\pg5 ave - 2020-2021 - 2º sem\aulas-41d\aula05-logger-with-annotations>
```

Reflection API

Object oriented API for metadata.

Enable programming with metadata.

Reflection API follows to **Type System**

Type System (e.g. ECMA 335) = Specification that describes how types are defined and behavior.

Exemplo: Um **tipo** pode ser definido por uma **classe** ou **interface**, que tem **membros** e esses membros podem ser ….

Reflection API

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

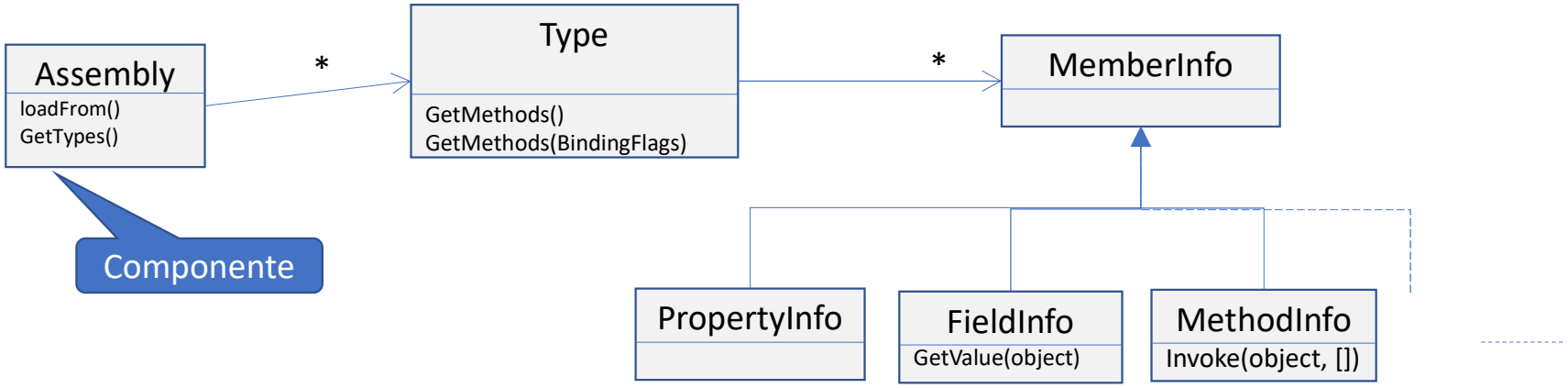
Exemplos:

- .net - System.Reflection
  e.g. `obj.GetType().GetMethod("hello").Invoke(foo, null);`

- Java - java.lang.reflect
  e.g. `obj.getClass().getDeclaredMethod("hello").invoke(foo);`

- Javascript
  e.g. `obj['hello’].call(obj)`

.net - System.Reflection



Solution --->* Project

aulas-41d > aula04-logger

Name

App

Logger

aula04-logger.sln

```
dotnet new console -o App
cd App
dotnet add reference ..\Logger\Logger.csproj
```

```
dotnet new classlib -o Logger
```

```
Point p = new Point(7, 9);
Student s = new Student(154134, "Ze Manel", 5243, "ze");
Account a = new Account(1300);
Log log = new Log();
log.Info(p);
log.Info(s);
log.Info(a);
```

Define o Domínio
e.g. Student,

Utilitário
Independente do domínio

```
namespace Logger
{
    public class Log
    {
        public void Info(object target) {
            string status = Inspect(target);
            printer.Print(status);
            printer.Print(Environment.NewLine);
        }
        private string Inspect(object target)
        {
            string fields = LogFields(target);
            …
```

GetType()

- **Duas instancias da mesma classe têm o mesmo `Type`.**
- **Duas instancias da mesma classe retornam o MESMO objecto Type**
- Exemplo:

```
Point p1 = new Point(2,3); Point p2 = new Point(9, 7);
Console.WriteLine(p1.GetType() == p2.GetType()); // true
```

Em Java:

```
out.println(p1.getClass() == p2.getClass()); // true
```

AVE mantém um único representante por cada tipo carregado

is ⇔ instanceof

- Duas instancias da mesma classe têm o mesmo `Type`.
- Exemplo:

```
Point p1 = new Point(2,3); Point p2 = new Point(9, 7);
Console.WriteLine(p1 is Point && p2 is Point); // true
```

Em Java:

```
out.println(p1 instanceof Point && p2 instanceof Point); // true
```

Diferenças ?

```
private string LogFields(object o) {
    Type t = o.GetType();

    StringBuilder str = new StringBuilder();
    FieldInfo[] fields = t.GetFields();
    foreach(FieldInfo field in fields) {
        str.Append(field.Name);
        str.Append(": ");
        str.Append(field.GetValue(o));
        str.Append(", ");
    }
    return str.ToString();

}
```

```
private string LogMethods(object o) {
    Type t = o.GetType();

    StringBuilder str = new StringBuilder();
    MethodInfo[] methods = t.GetMethods();
    foreach(MethodInfo method in methods) {
        str.Append(method.Name);
        str.Append(": ");
        if(method.GetParameters().Length == 0) {
            str.Append(method.Invoke(o, null));
        }
        str.Append(", ");
    }
    return str.ToString();

}
```

Testes Unitários

- Sistematizar um conjunto de testes
- Forma determinística de avaliar se o resultado é esperado (e.g. Assert)

? Como distinguir métodos de testes de outros métodos?

- Por convencão e.g. prefixo Test
- **Anotação** e.g. [Test], @Test, [Fact]

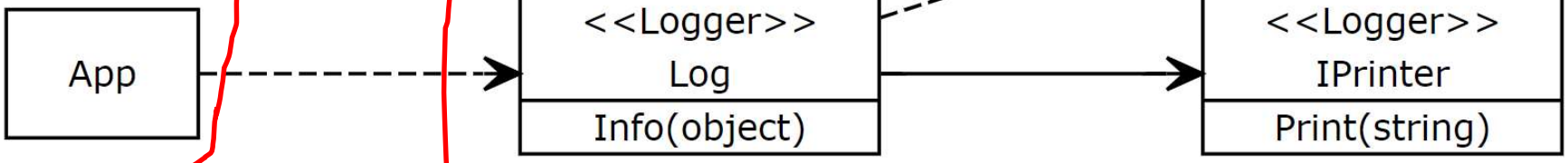
? Como é que a ferramenta de testes (e.g. dotnet test) identifica quais os métodos anotados ou com prefixo Test?

R: Utilização de API Reflection

aulas-41d > aula04-logger

Name

- App
- Logger
- aula04-logger.sln



Annotations in DotNet ⇔ *Custom Attributes*

Annotations:

- Let programmers express intentions on code, e.g. [Fact], [Test], [ToLog],
- In dotnet, annotations are classes that extend the class Attribute.

Example:

Logger developer

```
public class ToLogAttribute : Attribute { }
```

```
private bool ShouldLog(MemberInfo m) {
   /**
     * Check if it is annotated with ToLog
     */
var attr = (ToLogAttribute)Attribute.GetCustomAttribute(m,typeof(ToLogAttribute));
if(attr == null) return false;
```

Logger client developer

!!! We can suppress the Attribute suffix !!!

```
public class Point{
    [ToLog] public readonly int x;
    public readonly int y;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    [ToLog] public double GetModule() {
            return System.Math.Sqrt(x*x + y*y);
    }
…
```

Logger client developer

!!! We can suppress the Attribute suffix !!!

```
public class Point{
    [ToLog] public readonly int x;
    public readonly int y;
    …
```

(**Escreve**)
Serialize on metadata

Point::x : public initonly int32

Find Find Next

```
.field public initonly int32 x
.custom instance void [Logger]ToLogAttribute::.ctor() = ( 01 00 00 00 )
```

Point::y : public in...

Find Find Next

```
.field public initonly int32 y
```

(**Lê**)
Deserialize metadata to build a new instance

```
Attribute::GetCustomAttribute(MemberInfo, Type)
```

Dotnet Reflection API to get Custom Attributes

- `Attribute::IsDefined(MemberInfo, Type)`

Returns boolean

Returns Object ?!
Why not Attribute?

Builds an attribute instance from
the metadata information.

- `Attribute::GetCustomAttribute(MemberInfo, Type)`
- E.g. `ToLog attribute = (ToLog)Attribute.GetCustomAttribute(m,typeof(ToLog));`

typeof(ToLog) → Type
Java ⇔ ToLog.class

- Dotnet Type System does not require annotations to be Attribute.
  - **This is a C# requirement.**

How to get Type info

- Compile time:

```
typeof(TypeName) ⇔ TypeName.class      (java)
```

- Runtime:

```
obj.GetType() ⇔ obj.getClass()         (java)
```

C# naming conventions

- Method names begin with Capital letter, e.g. `Log`
- Interfaces begin with an I, e.g. `IPrinter`
- Attributes end with the suffix, e.g. `ToLogAttribute`