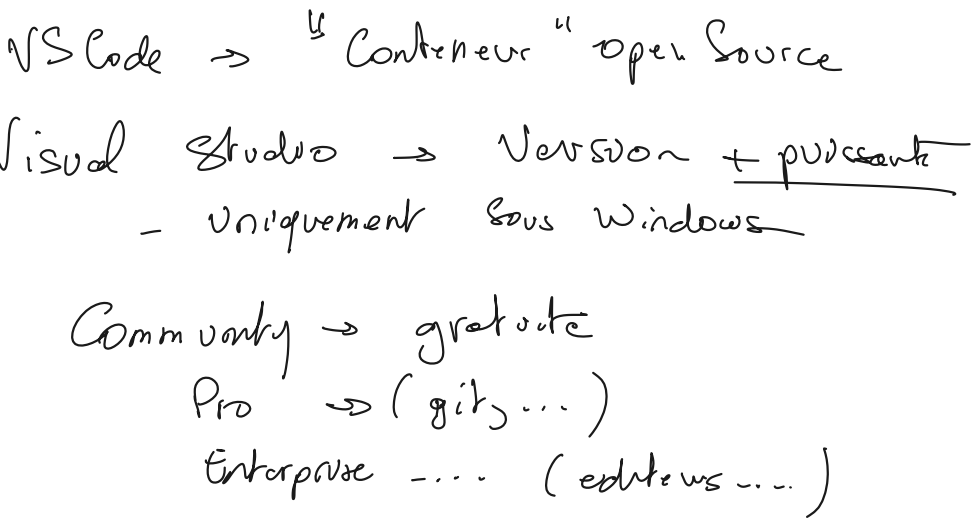
DEV C#

Installer Visual Studio Community :
https://visualstudio.microsoft.com/fr/downloads/

Charges de travail :
Dev Web et ASP.Net
Dev .Net Desktop



Le kit de fichiers pour le cours : https://we.tl/t-VslERGy8qR

9h - 17h

9h-12h30 13h30-17h
pause a 10h30 15h30

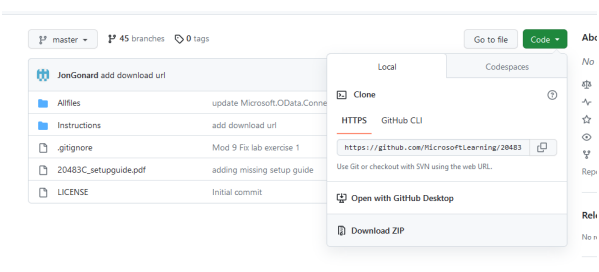
Récupération de la base de données
locale (sans MSSQLExpress) :
https://we.tl/t-FHt5rWFpbI

mdf **ET** ldf dans le dossier Data (absolument !)
Puis 'Ajouter un element existant' sous le dossier Data du projet.
modifiez la connexionString dans votre app.config selon ce qui est fourni.
ps : le contenu de la base sera recopié a chaque exécution du programme (et donc, réinitialisé)
Si le contenu de la base de données est vide, vous êtes alors connectés à master :
Ajoutez cet élément dans la connexionstring :
;Initial Catalog=SchoolDB
ex : *netFailover=False;Initial Catalog=SchoolDB""*
providerName="System.Data.EntityClient"/>

chaîne de connexion OK sous SQLExpress :
<add name="SchoolDBEntities"
connectionString="metadata=res://*/Data.SchoolData.csdl|res://*/Data.SchoolData.ssdl|res://*/Data.SchoolData.msl;provider=System.Data.SqlClient;provider connection string="Data Source=.\sqlexpress;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False;Initial Catalog=SchoolDB""
providerName="System.Data.EntityClient"/>

https://rug.gklearn.fr
identifiant et mot de passe : ~~d157921~~ d157922
Sous windows : P@ssw0rd

Code source des Labs + instructions :
https://github.com/MicrosoftLearning/20483-Programming-in-C-Sharp/tree/master



Etat de lieux.

historique

80 → C --- (C++) → jeux 3D

→ basic, pascal, Cobol, ...

C++ (problèmes :
- Gestion mémoire - Trop complexe
  - C++ "Moderne"

⇒ RUST → assez nouveau
  - App natives (performance, faible conso mémoire, ...)
  - (jeux)

C++ → Trop complexe

("Portable") → Compilable sur toute plateforme.

archi

	Win	Linux
- intel - x86		
ARM		

C++

(industrie

Dev Applicatif)

→ Très lourd.

→ Solutions

→ Design Pattern du GoF.

Exemple

→ - Singleton

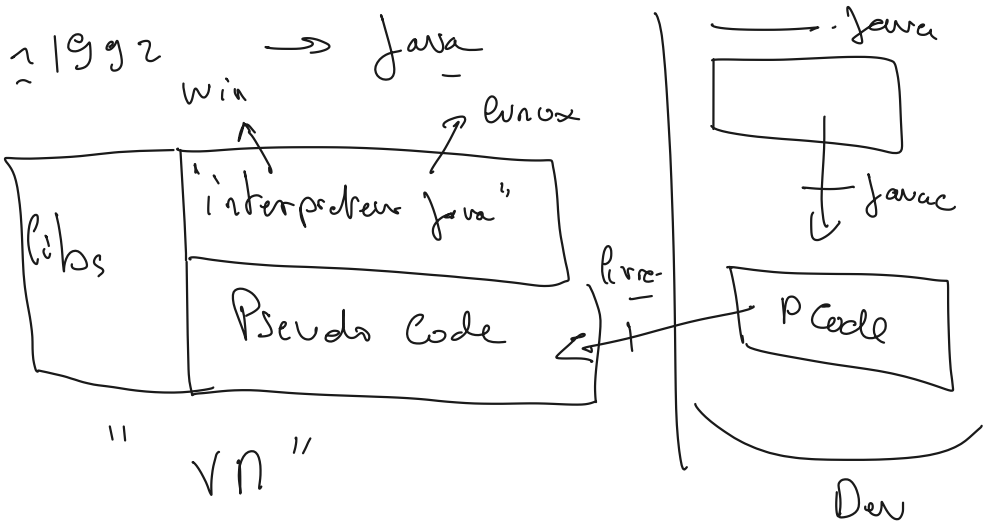
- Factory

VB(A)

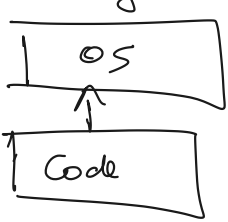
Delphi → RAD

→ Dev Productif

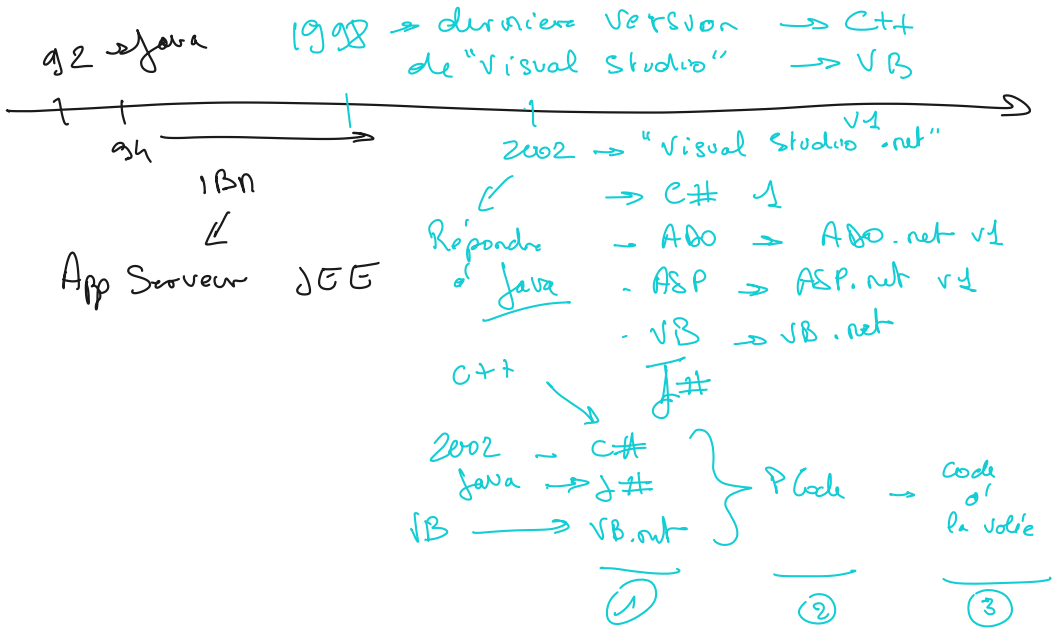
RAD → Conception graphique



Code natif

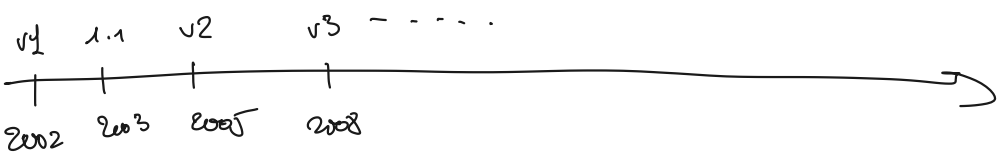


Code généré



Purement fonctionnel : F# (2023)

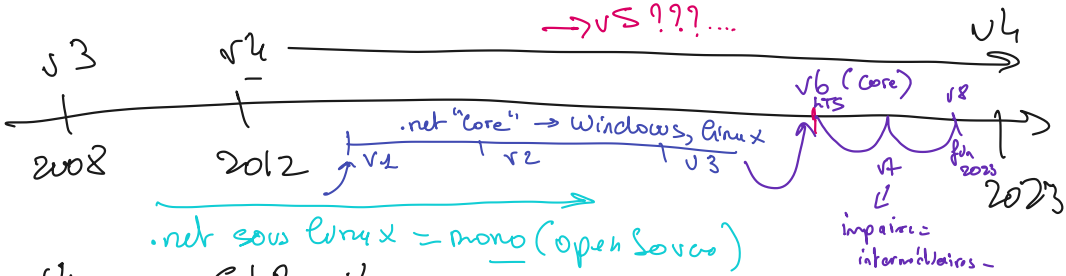
Scala



v1 → C# 1
ADO.net 1
ASP.net 1
} CLR v1

v2 - C# 2
ADO.net 2
} CLR v2

2008 Windows → XAML
.net 3 - WPF
- W(orkflow) F
- Wcf
} extensions
C# 3 → CLR v2



v4 → CLR v4

→ Axé sur le Prog. SMP (Multi Processing)

→ Threads

→ Multi coeurs

v4.1

.2

.3.

v4.7.1

v4.7.2

7. 10 . ...

v4.8

Avant:

Specs internes .net

Version du .net framework

v1 .. v4.

Maintenant

Specs (.net Standards)

Impl .net FW

Full, windows

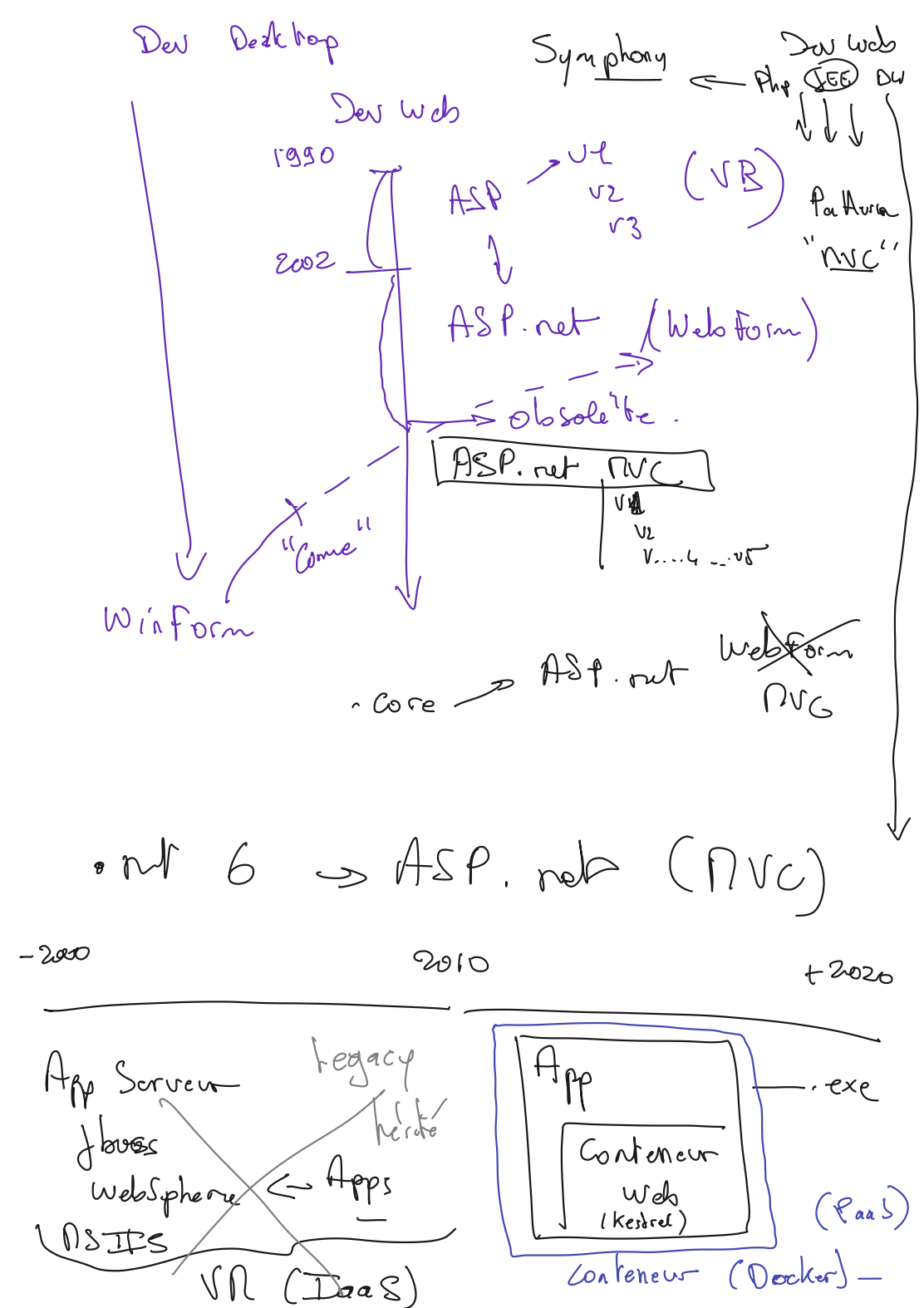
v4 +

Impl .net Core

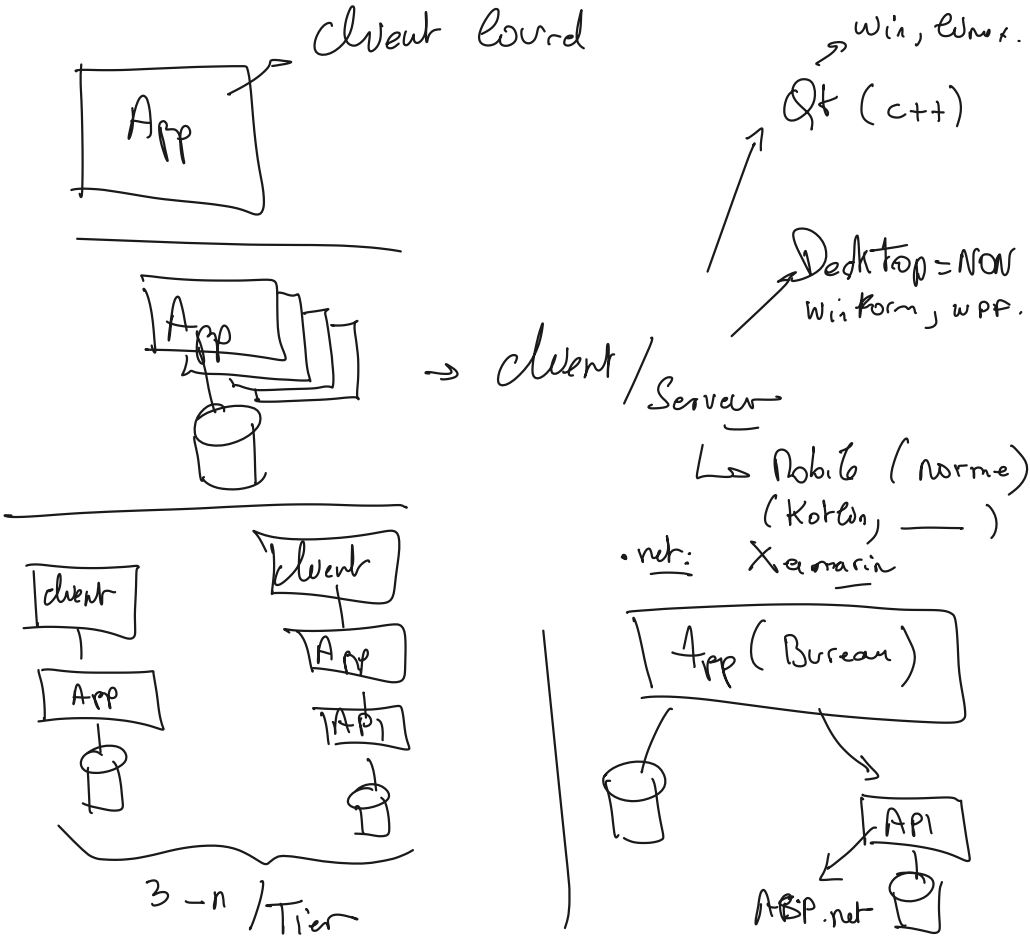
Open source

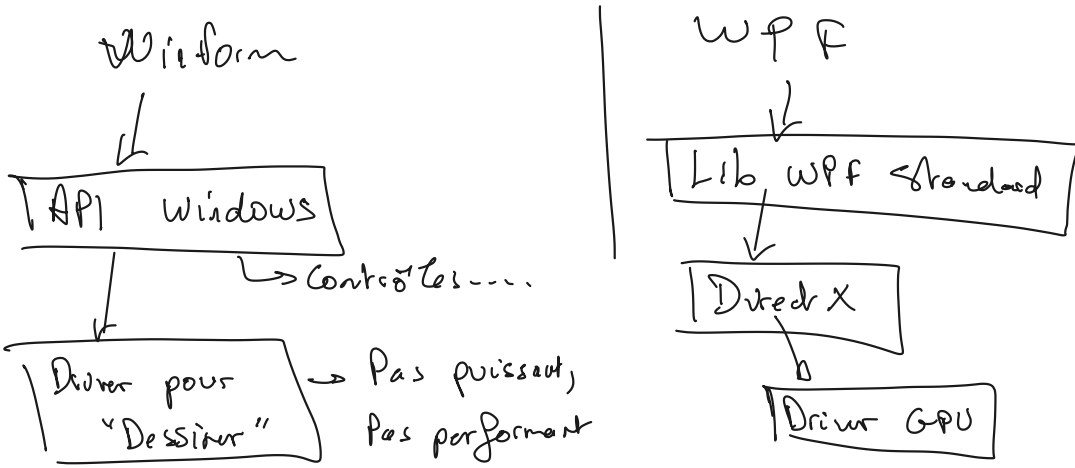
Win, Linux

v1 .. v3 . core


Dev Desktop
Dev Web
1990
2002
ASP
v1
v2
v3
(VB)
ASP.net (WebForm)
obsolète.
ASP.net MVC
v1
v2
v...4 ...v5
"Come"
Winform
Symphony
Dev Web
Php
JEE
DW
PalAura
"MVC"
.Core
ASP.net
WebForm
MVC
.net 6 → ASP.net (MVC)
-2000
2010
+2020
App Server
jboss
WebSphere
IIS
Legacy
Apps
VM (IaaS)
App
.exe
Conteneur
Web
(Kestrel)
(PaaS)
Conteneur (Docker)

Archi's


client lourd
App
Win, Linux.
Qt (c++)
Desktop = WPF
WinForm, WPF.
App
client / Server
Mobile (Xamarin)
(Kotlin, ___)
.net:
client
App
client
App
API
App (Bureau)
API
ASP.net
3 - n / Tier



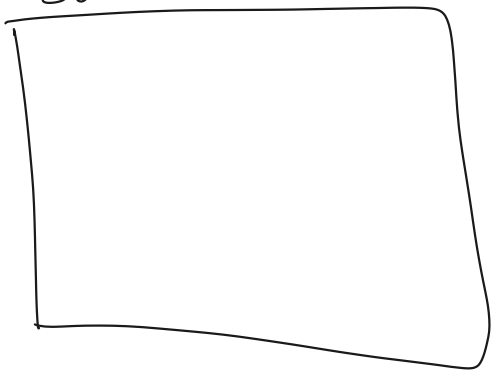
Console (Texte)

→ Simple

ASP.net (MVC) → App Console

Toute App en Conteneur (Docker) est une App Console

"Solution" ⟶ Conteneur de Projets

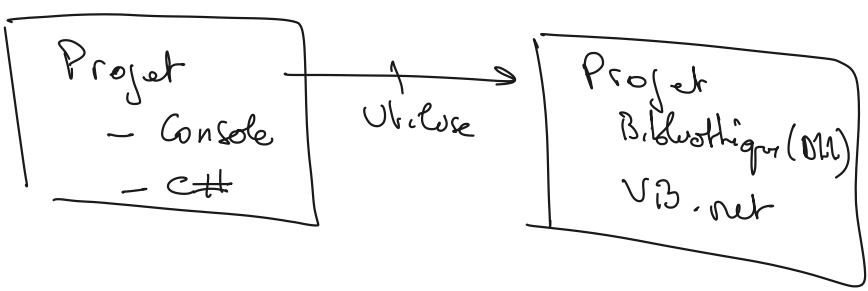


VStudio → instance → 1 instance = 1 Solution

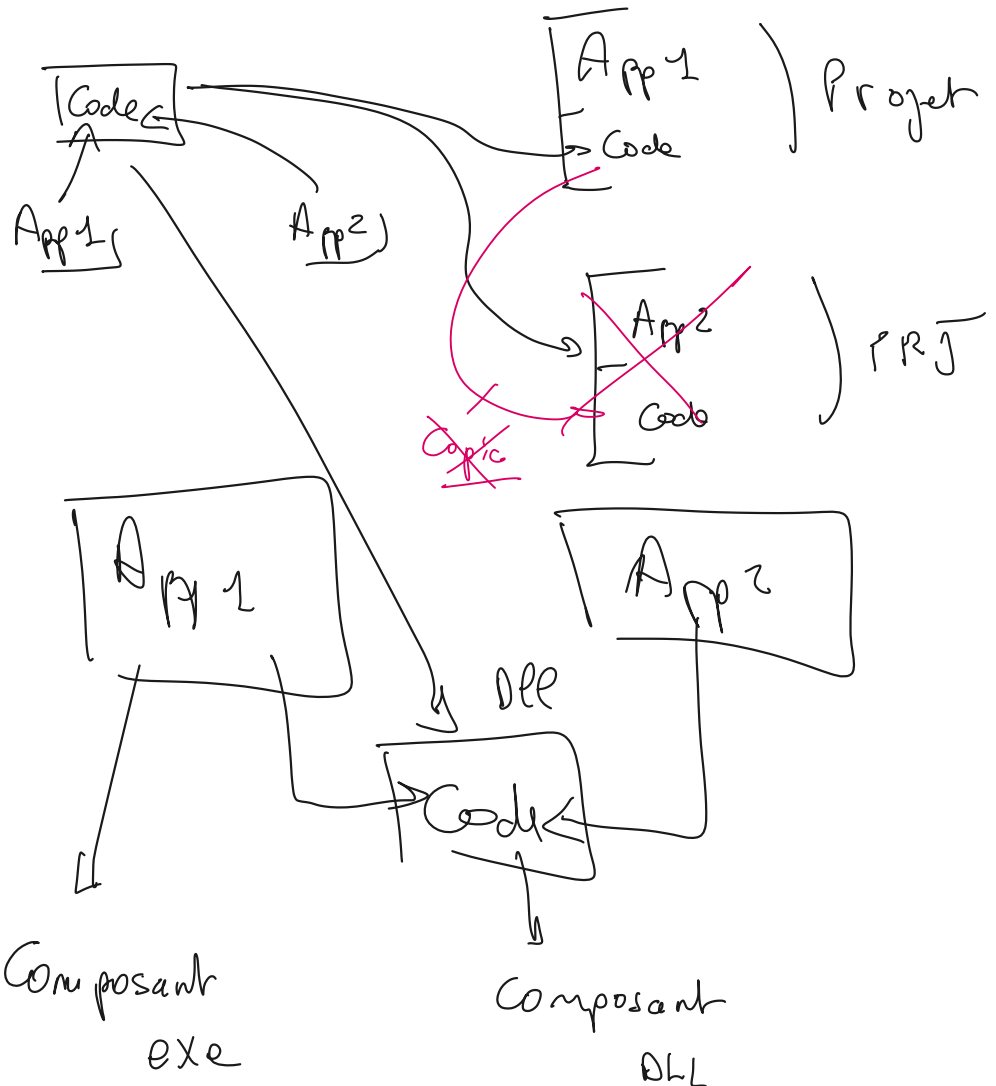
1 Projet = - 1 cible
- Console
- Winform
- WPF
- bibliothèque
- ASP.net
- etc....

- 1 Langage
- C#
- F#
- VB.net

- Des paramètres -



Réutilisabilité



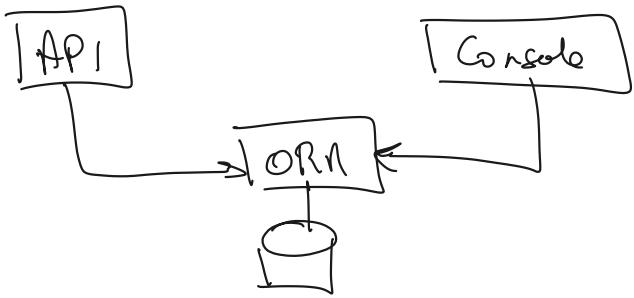
App front Web → ASP.net (MVC)

App Admin rapide avec → (WPF) / WinForm.

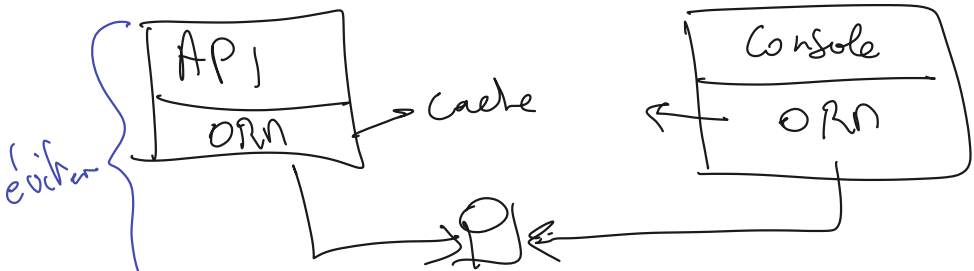
App scriptable → Console

API → ASP.net

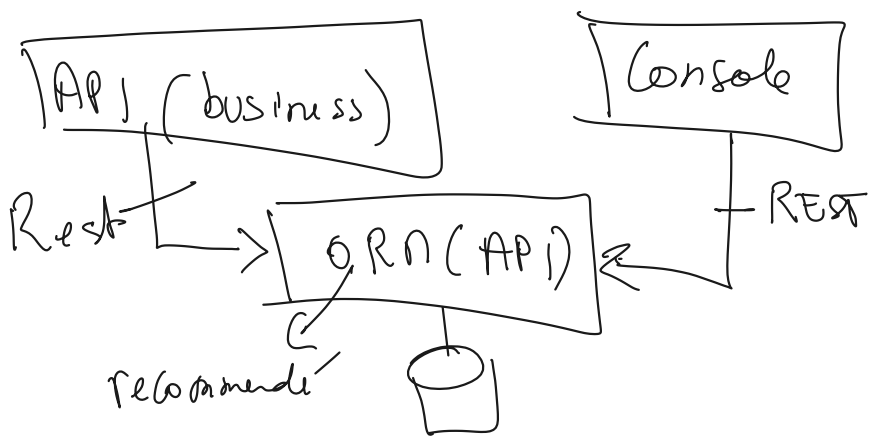
ORM → Lib / ASP.net


API
Console
ORM

Lib:


API
ORM
cache
Console
ORM
éviter

API

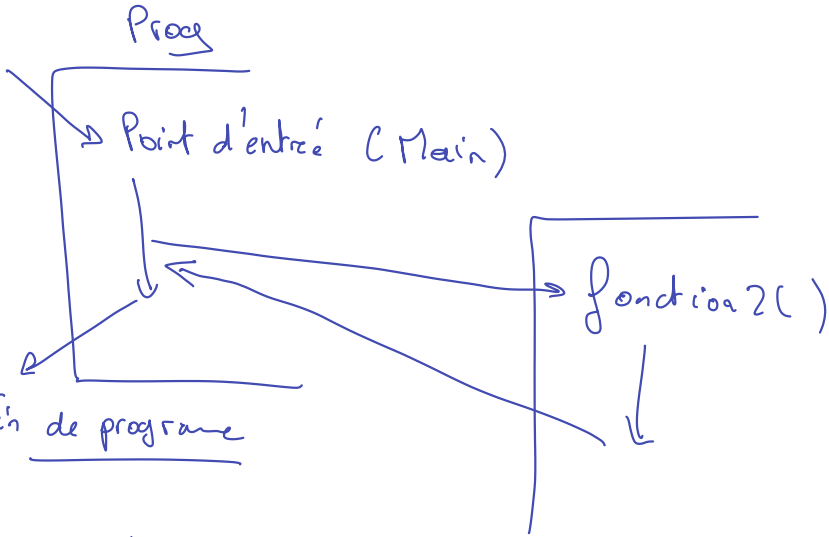

API (business)
Console
Rest
ORM (API)
REST
recommandé

C#

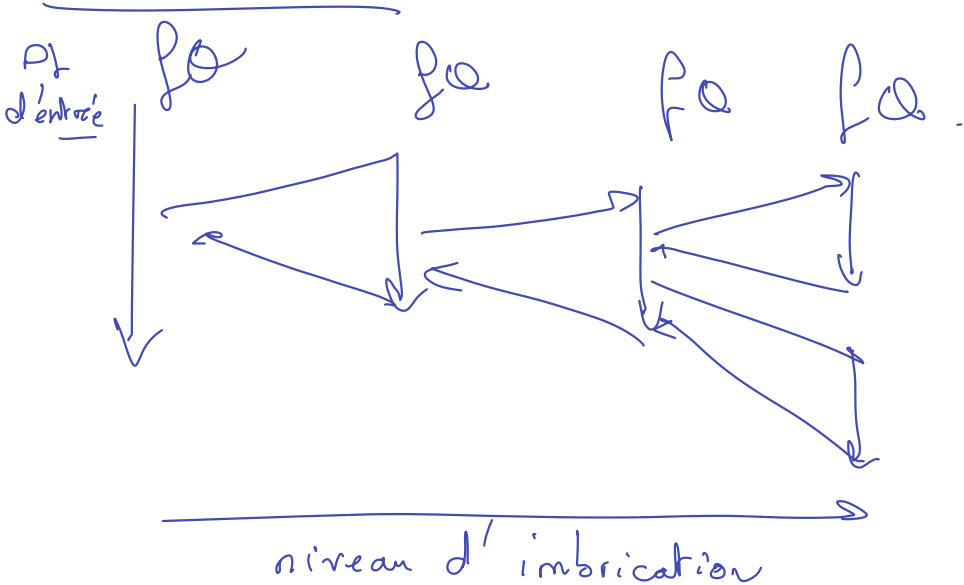
Générations

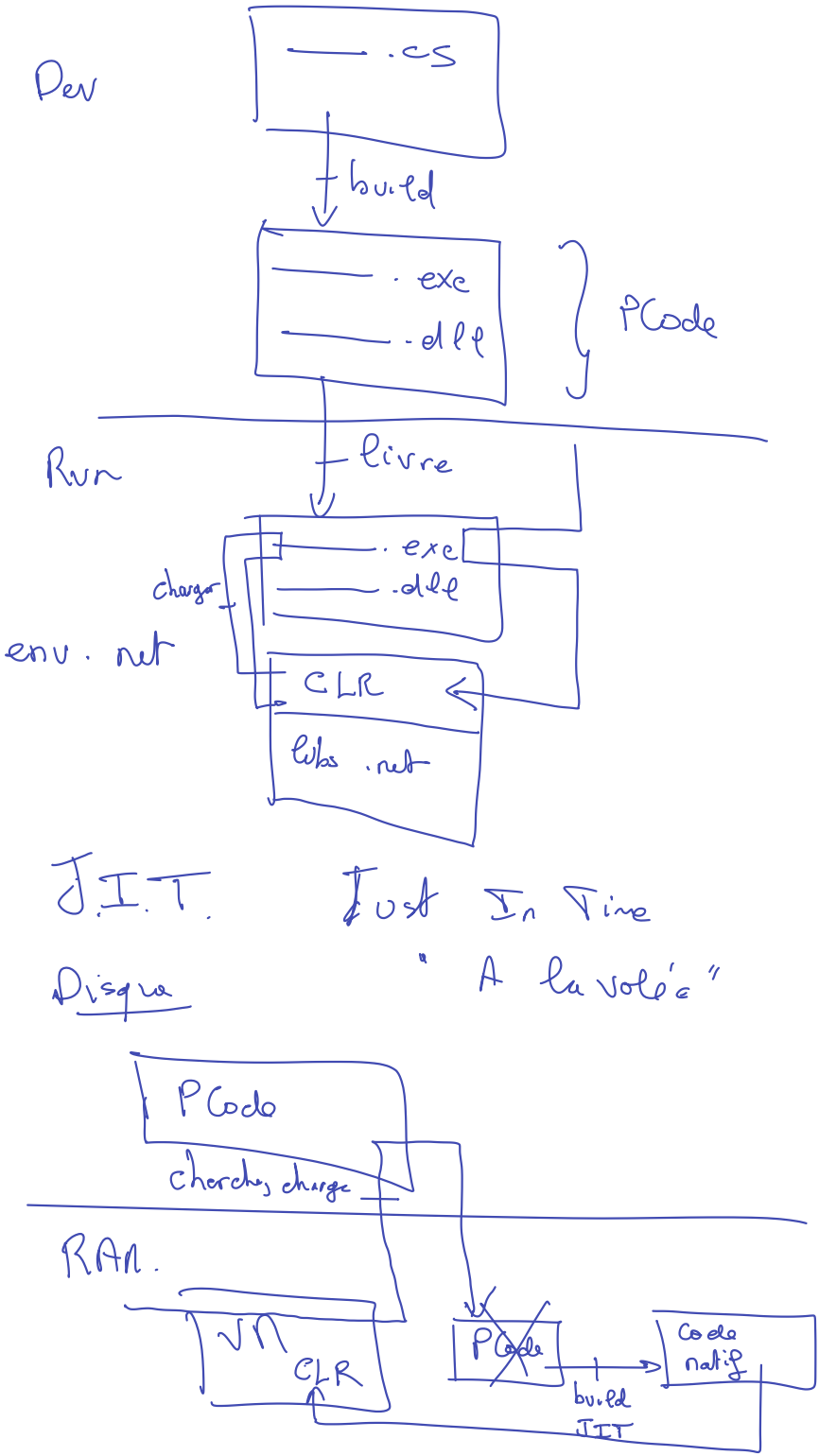
- Assembleur (1G)
- Prog naturelle (2G) → Basic, Cobol, php, shell, python
- Prog structuré (procédures, fonctions) (3G) → C, pascal, php, python, javascript, ...
- Prog. orienté objet (4G) → C++, Java, C#, TypeScript, Python, php.
  - Full objet → obligatoire
    - C#
    - Java
  - Semi objet
    - C++
    - Php
    - Python
  - Paramètres de fonctions → Réf / Valeur données
    - C++ → pointeur sur fonction
    - → fonctions
- Prog. fonctionnelle (5G) → Scala, F#
  - Possible Python, C++, C#, ...

Prog structuré



Pile d'appels :




Dev
.cs
build
.exe
.dll
PCode
Run
livre
.exe
.dll
charge
env. net
CLR
libs .net
J.I.T.
Just In Time
" A la volée "
Risque
PCode
cherche, charge
RAM.
VM
CLR
PCode
Code natif
build
JIT

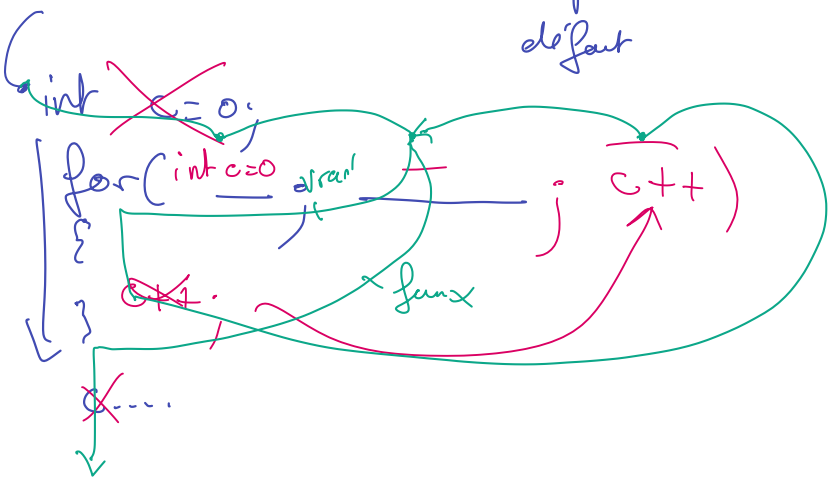
```
for(;;);
```

option

option

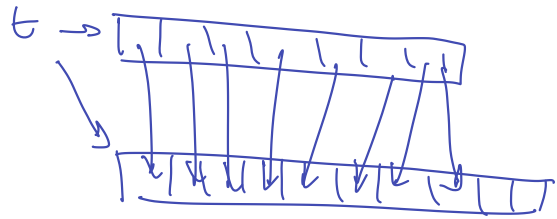
```
for ( <initialisation> ; <condition> ; <itération> )
```

True par défaut



Dim t(10) → tableau de 10

Redim t(20)

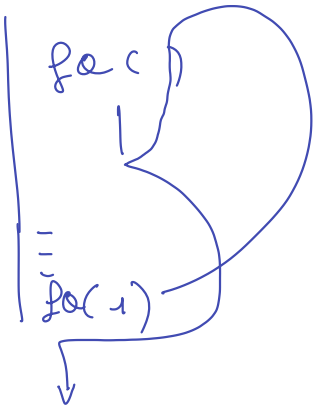


Connaissez-vous d'avance le nombre d'éléments ?

oui → Tableau

non → Liste

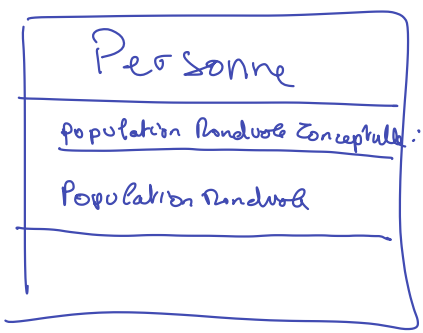
Script



Objet

```
class MC
foo()

MC mc = new MC();
mc.foo() ?
```

Personne.pm - capitale
↑
5.850.230.212 (exact)



Xavier.pm → 5
Bryan.pm → 7

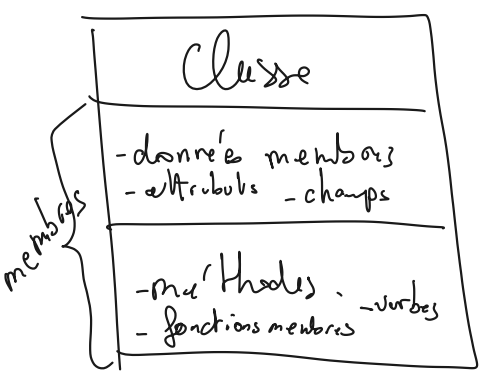
Accesseur : → Getter/Setter Java
→ Propriétés .net

private age

return age;

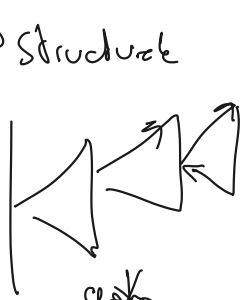
Java : ok
```
getAge() : int
setAge(int age);
this.age = age;
```

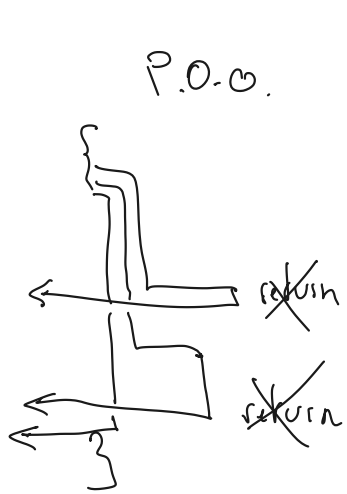
Concepts objets

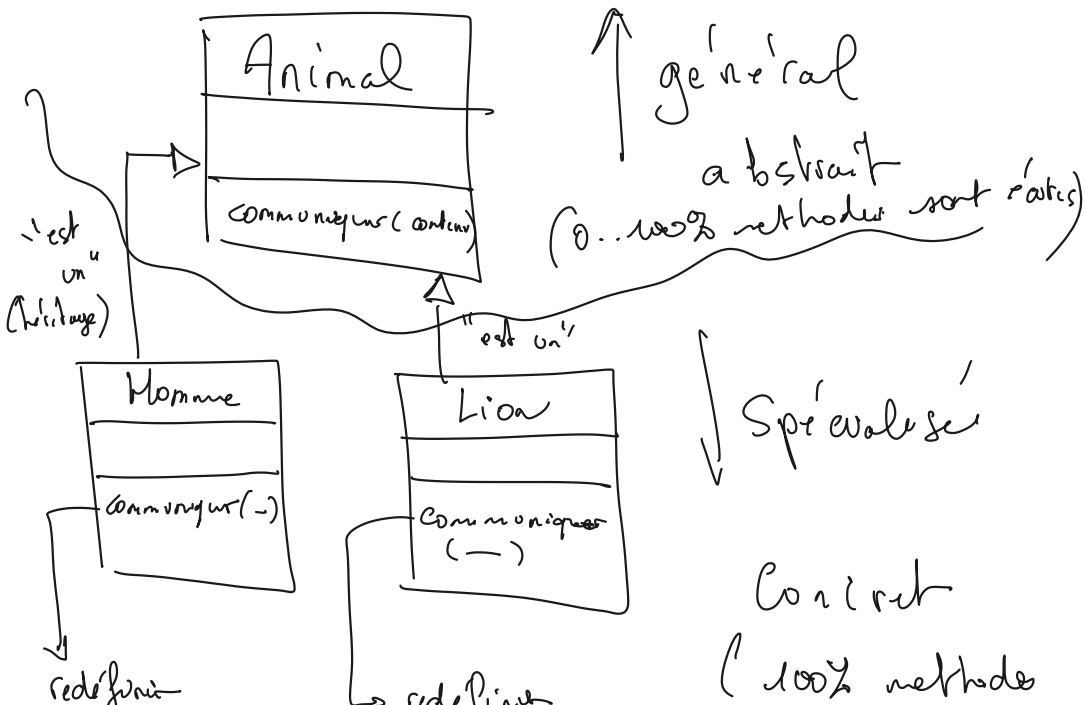


Événements

↳ délégués (ferme.net)
. (pointeur sur fonction)
- Deviendra une Lambda (Prog fonctionnelle)







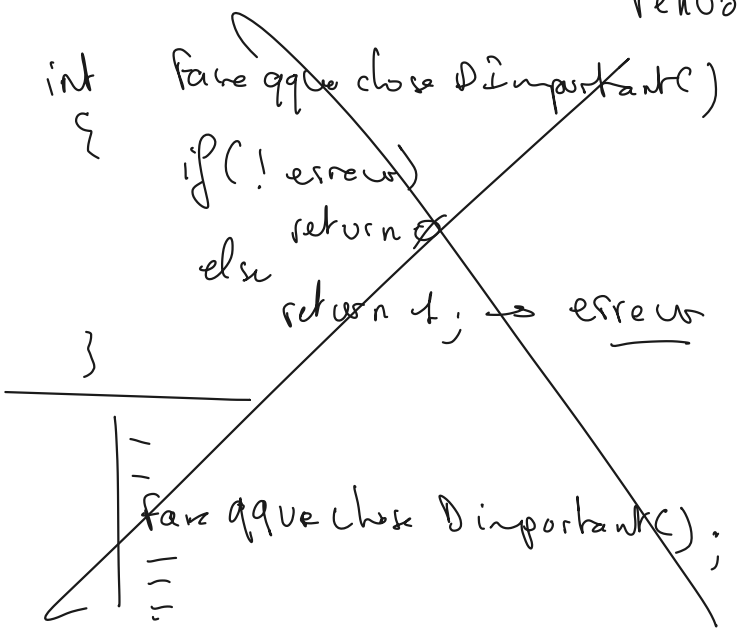
- Surcharge (Verticale) / (Polymorphisme)
  - overrides - -
  - redéfinition / extension de méthode

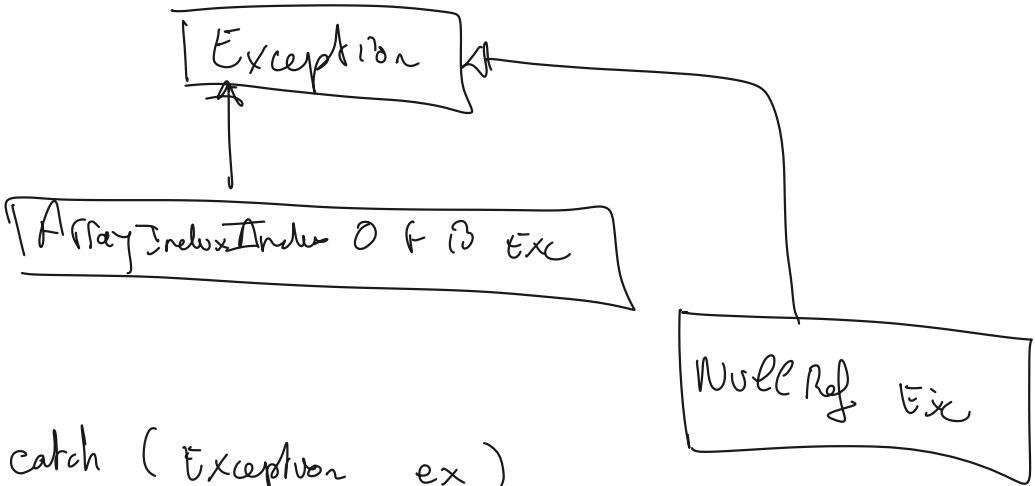
Signature = nom de méthode + Types de paramètres -

"La signature doit être unique par classe"

"Une signature peut être surchargée"

Anciens langages = sur erreur = renvoi d'un code





catch (Exception ex)

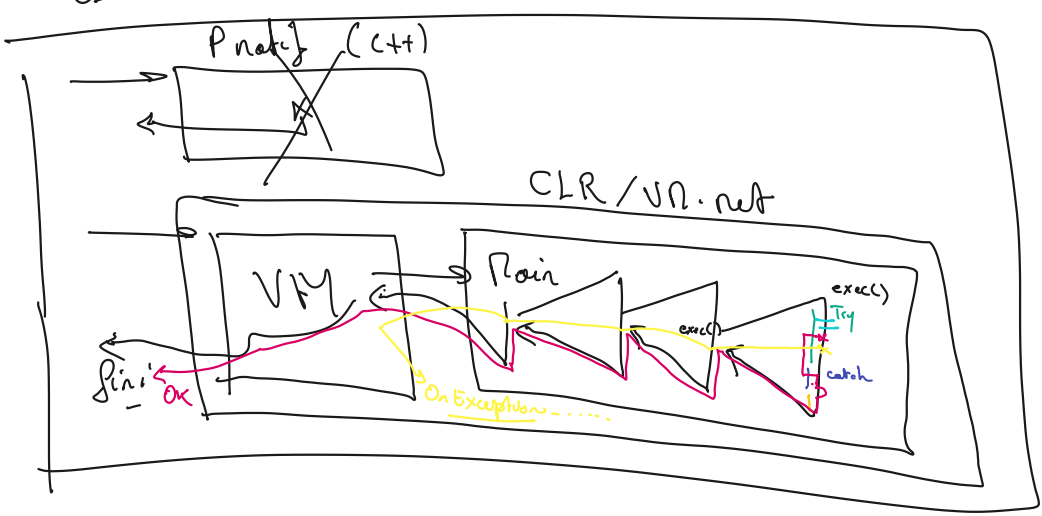
→ Toutes les erreurs

Try → bloc à exécuter

catch → selon l'erreur

finally → "Dans tous les cas"

OS



Consultez la fin de ce message pour plus de détails sur l'appel du débogage
juste-à-temps (JIT) à la place de cette boîte de dialogue.

********** Texte de l'exception **********
System.DivideByZeroException: Attempted to divide by zero.
  at WinFormsApp1.Class1.SePlanter() in C:\Users\PhilippePecqueur\source\repos\ConsoleApp1\WinFormsApp1\Class1.cs:line 14
  at WinFormsApp1.Form1.button1_Click(Object sender, EventArgs e) in C:\Users\PhilippePecqueur\source\repos\ConsoleApp1\WinFormsApp1\Form1.cs:line 12
  at System.Windows.Forms.Control.OnClick(EventArgs e)
  at System.Windows.Forms.Button.OnClick(EventArgs e)
  at System.Windows.Forms.Button.OnMouseUp(MouseEventArgs mevent)
  at System.Windows.Forms.Control.WmMouseUp(Message& m, MouseButtons button, Int32 clicks)
  at System.Windows.Forms.Control.WndProc(Message& m)
  at System.Windows.Forms.ButtonBase.WndProc(Message& m)
  at System.Windows.Forms.Control.ControlNativeWindow.WndProc(Message& m)
  at System.Windows.Forms.NativeWindow.Callback(IntPtr hWnd, WM msg, IntPtr wparam, IntPtr lparam)

********** Assemblys chargés **********
System.Private.CoreLib
  Version de l'assembly : 6.0.0.0
  Version Win32 : 6.0.2023.32017
  CodeBase : file:///C:/Program%20Files/dotnet/shared/Microsoft.NETCore.App/6.0.20/System.Private.CoreLib.dll
----
WinFormsApp1
  Version de l'assembly : 1.0.0.0
  Version Win32 : 1.0.0.0
  CodeBase : file:///C:/Users/PhilippePecqueur/source/repos/ConsoleApp1/WinFormsApp1/bin/Debug/net6.0-windows/WinFormsApp1.dll
----
System.Runtime
  Version de l'assembly : 6.0.0.0
  Version Win32 : 6.0.2023.32017
  CodeBase : file:///C:/Program%20Files/dotnet/shared/Microsoft.NETCore.App/6.0.20/System.Runtime.dll
----
Microsoft.Extensions.DotNetDeltaApplier
  Version de l'assembly : 17.0.0.0
  Version Win32 : 17.6.326.62524
  CodeBase : file:///c:/program%20files/microsoft%20visual%20studio/2022/enterprise/common7/ide/commonextensions/microsoft/hotreload/Microsoft.Extensions.DotNetDeltaApplier.dll
----
System.IO.Pipes
  Version de l'assembly : 6.0.0.0
  Version Win32 : 6.0.2023.32017
  CodeBase : file:///C:/Program%20Files/dotnet/shared/Microsoft.NETCore.App/6.0.20/System.IO.Pipes.dll
----
System.Linq
  Version de l'assembly : 6.0.0.0
  Version Win32 : 6.0.2023.32017
  CodeBase : file:///C:/Program%20Files/dotnet/shared/Microsoft.NETCore.App/6.0.20/System.Linq.dll
----
System.Collections
  Version de l'assembly : 6.0.0.0
  Version Win32 : 6.0.2023.32017
  CodeBase : file:///C:/Program%20Files/dotnet/shared/Microsoft.NETCore.App/6.0.20/System.Collections.dll
----
System.Windows.Forms
  Version de l'assembly : 6.0.2.0
  Version Win32 : 6.0.2023.32104
  CodeBase : file:///C:/Program%20Files/dotnet/shared/Microsoft.WindowsDesktop.App/6.0.20/System.Windows.Forms.dll
----
System.ComponentModel.Primitives
  Version de l'assembly : 6.0.0.0
  Version Win32 : 6.0.2023.32017
  CodeBase : file:///C:/Program%20Files/dotnet/shared/Microsoft.NETCore.App/6.0.20/System.ComponentModel.Primitives.dll
----
System.Windows.Forms.Primitives
  Version de l'assembly : 6.0.2.0
  Version Win32 : 6.0.2023.32104
  CodeBase : file:///C:/Program%20Files/dotnet/shared/Microsoft.WindowsDesktop.App/6.0.20/System.Windows.Forms.Primitives.dll
----
System.Collections.Concurrent
  Version de l'assembly : 6.0.0.0
  Version Win32 : 6.0.2023.32017
  CodeBase : file:///C:/Program%20Files/dotnet/shared/Microsoft.NETCore.App/6.0.20/System.Collections.Concurrent.dll
----
System.Runtime.InteropServices
  Version de l'assembly : 6.0.0.0
  Version Win32 : 6.0.2023.32017
  CodeBase : file:///C:/Program%20Files/dotnet/shared/Microsoft.NETCore.App/6.0.20/System.Runtime.InteropServices.dll
----
System.Drawing.Primitives
  Version de l'assembly : 6.0.0.0
  Version Win32 : 6.0.2023.32017
  CodeBase : file:///C:/Program%20Files/dotnet/shared/Microsoft.NETCore.App/6.0.20/System.Drawing.Primitives.dll
----
System.Collections.Specialized
  Version de l'assembly : 6.0.0.0
  Version Win32 : 6.0.2023.32017
  CodeBase : file:///C:/Program%20Files/dotnet/shared/Microsoft.NETCore.App/6.0.20/System.Collections.Specialized.dll
----
System.Console
  Version de l'assembly : 6.0.0.0
  Version Win32 : 6.0.2023.32017
  CodeBase : file:///C:/Program%20Files/dotnet/shared/Microsoft.NETCore.App/6.0.20/System.Console.dll
----
System.Threading
  Version de l'assembly : 6.0.0.0
  Version Win32 : 6.0.2023.32017
  CodeBase : file:///C:/Program%20Files/dotnet/sh

Fondamentaux .Net :
https://learn.microsoft.com/fr-fr/dotnet/csharp/fundamentals/types/

Variable → - portée
- étendue, visibilité.
-

Variables

- ~~Globales~~
- Par ~~Modules~~ -

→ donnée membres (POO)
→ static

- locales.
- paramètres -

C# → Pattern SingleTON -
+GC

- Type Valeur
- Type Référence

= Encapsulation

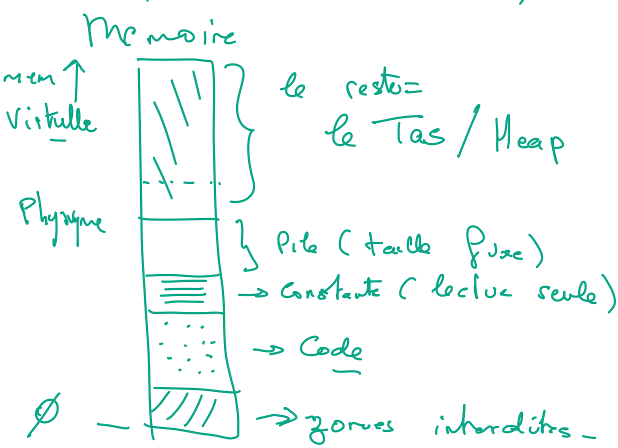
scope (portée) :

ouvre une portée
↳ {
int i; → i

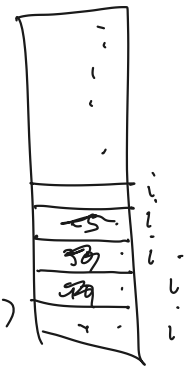
int j; → i, j

ferme → } → i

Pile (Stack) → zone mémoire.

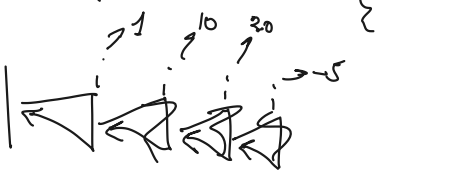


Pile




```
void m( )
{
   int i = 0;
   m( );
}
```

```
main( )
{
   m( );
}
```



CallStack = pile d'appel

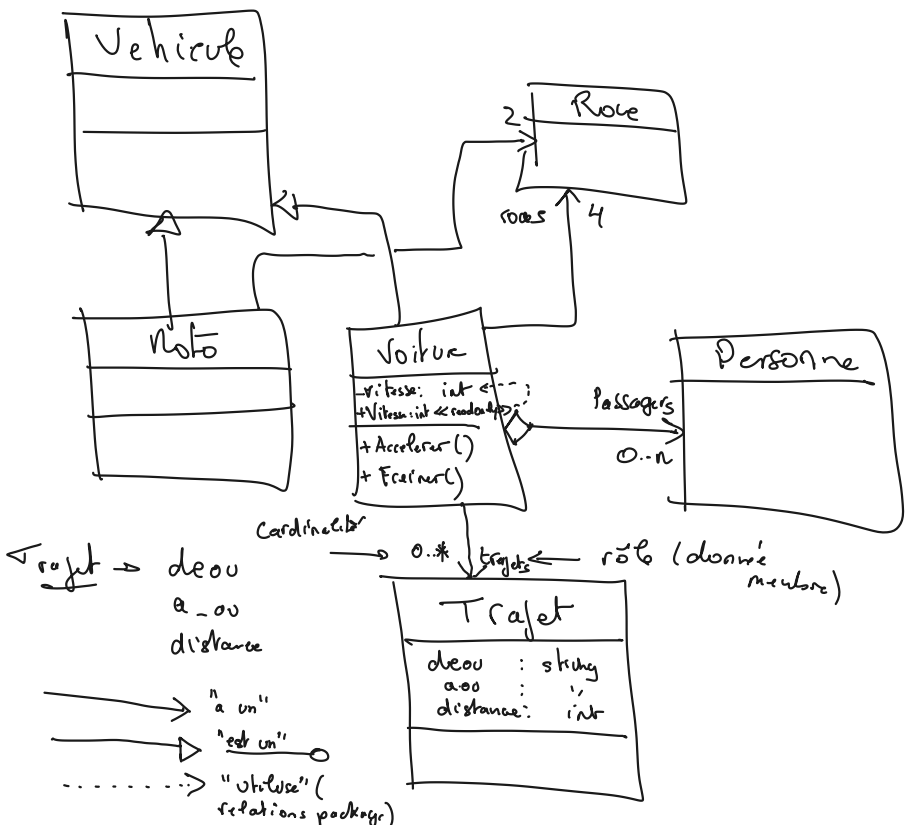
```
{
   if( . . . . )
   {
      int j;
   }
   {
      int i;
   }
}
```

→ Donnée membres -

UML → notation Unified Modeling language -

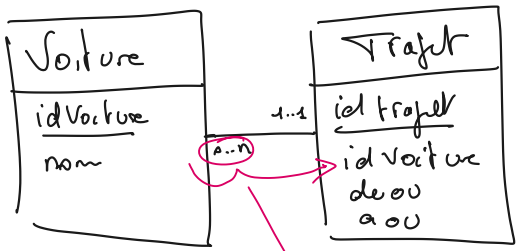
↳ UML 2 = 12 Types de diagrammes.

↳ - Diagramme de classe -

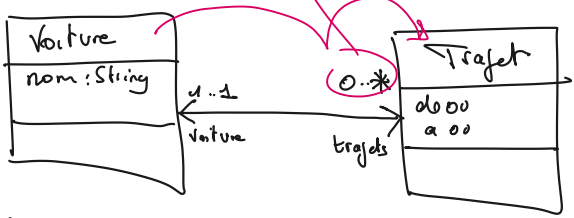


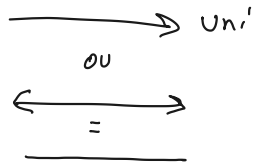
⚠ Merise et UML ne sont pas la même chose !!!

Merise :



UML





ex :

```
class Voiture {
    string nom ;
    List<Trajet> trajets = new List<Trajet>;
}
class Trajet {    string dd ou;
    Voiture voiture; }
```

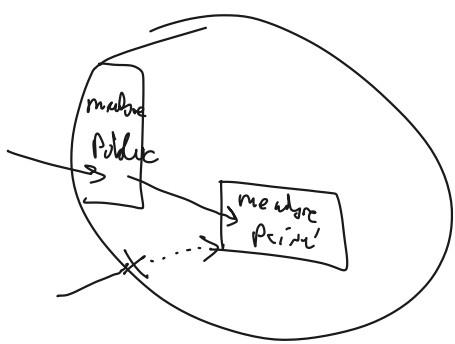
public : +

privé : -

protégé : #

```
class Voiture
{
    private int vitesse;
    public void Accelerer()
    {
        vitesse += 1;
    }
}
```

Capsule



Types Simples

- int, double, float, bool
- string ⇒ fonctionnent par copie

```
int i = 10;
int j = i;
j++
```

→ j = 11 / i = 10

Types Complexes

→ Tout le reste

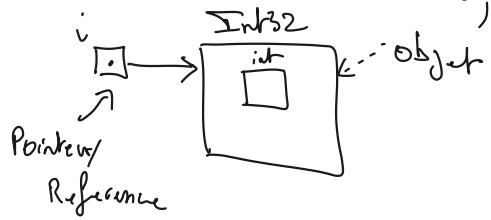
→ les objets — copie de référence

⇒ fonctionne par référence

```
object o;
o = null;
```

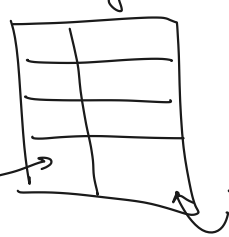
→ pas initialisé ?


```
Int32 i = new Int32();
```




référence (petit)

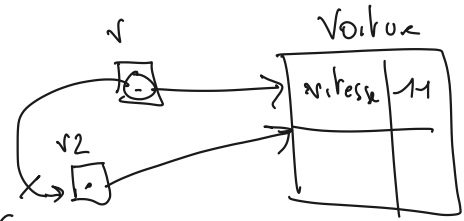

variable simple



ex :

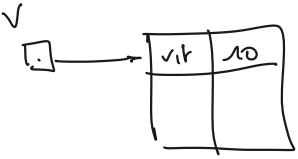
nouvelle instance

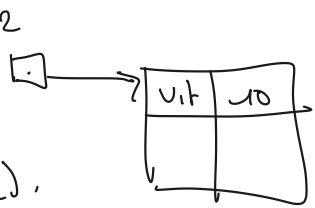
```
var v = new Voiture();
```




```
Voiture v2 = v;
v.vitesse++;
v2.vitesse = ??
```

```
v2 = (Voiture) v.clone();
```



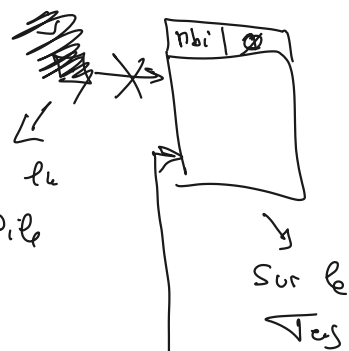



```
class Voiture {
    object clone() {
        Voiture temp = new Voiture();
        temp.vitesse = vitesse;
        return temp;
    }
```

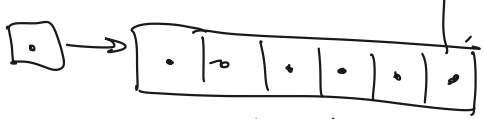
Garbage Collector (langage géré)

```
{
    Voiture v = new Voiture();
    v = null;
}
```

nombre d'instances



FReachable



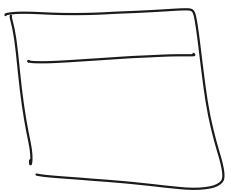
3 niveaux

- appel dtor() - membres
- appel dtor()
- vidage.

{

Personne p = new Personne();

}

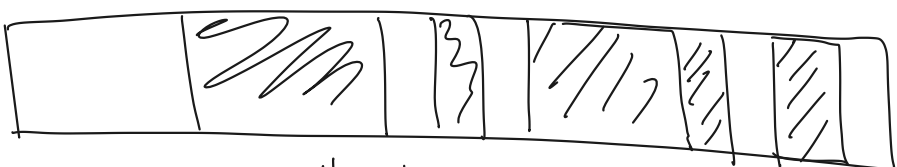


GC → Jean.dtor()

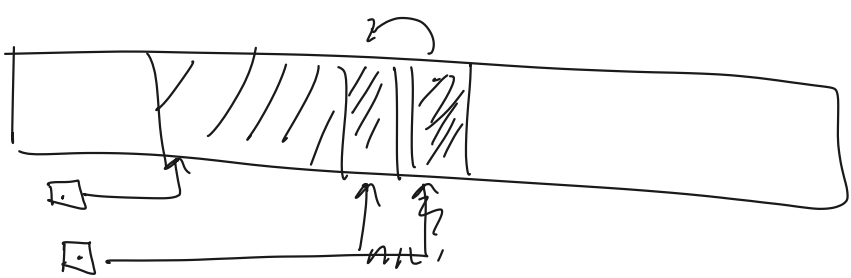
Zombies.zombies[0].
marcher();

class Personne
{ void marcher() appel par le GC avant destruction
~Personne()
{
Zombies.zombies.Add(this);
}
}
class Zombies
{
public static
var zombies = new
List<Personne>();
}

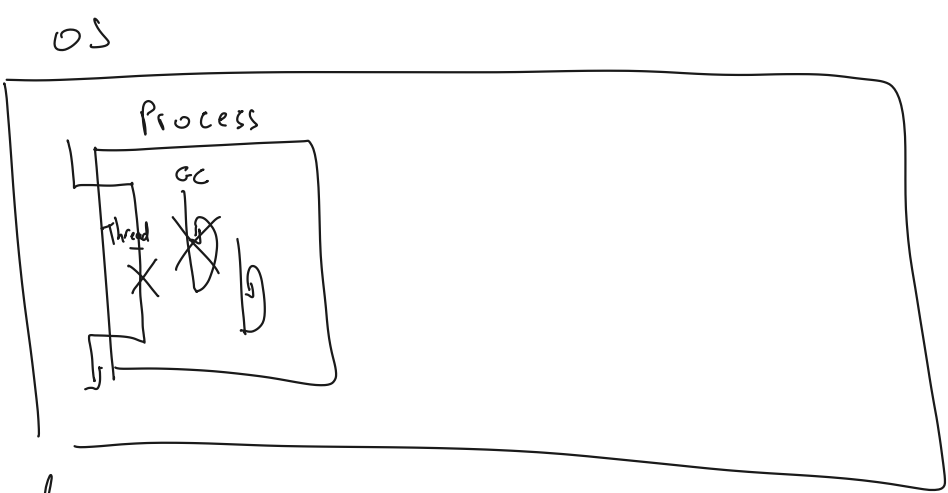
RAM



GC → - Appel dtor()
- Vidage Mémoire
- Compactage



Thread

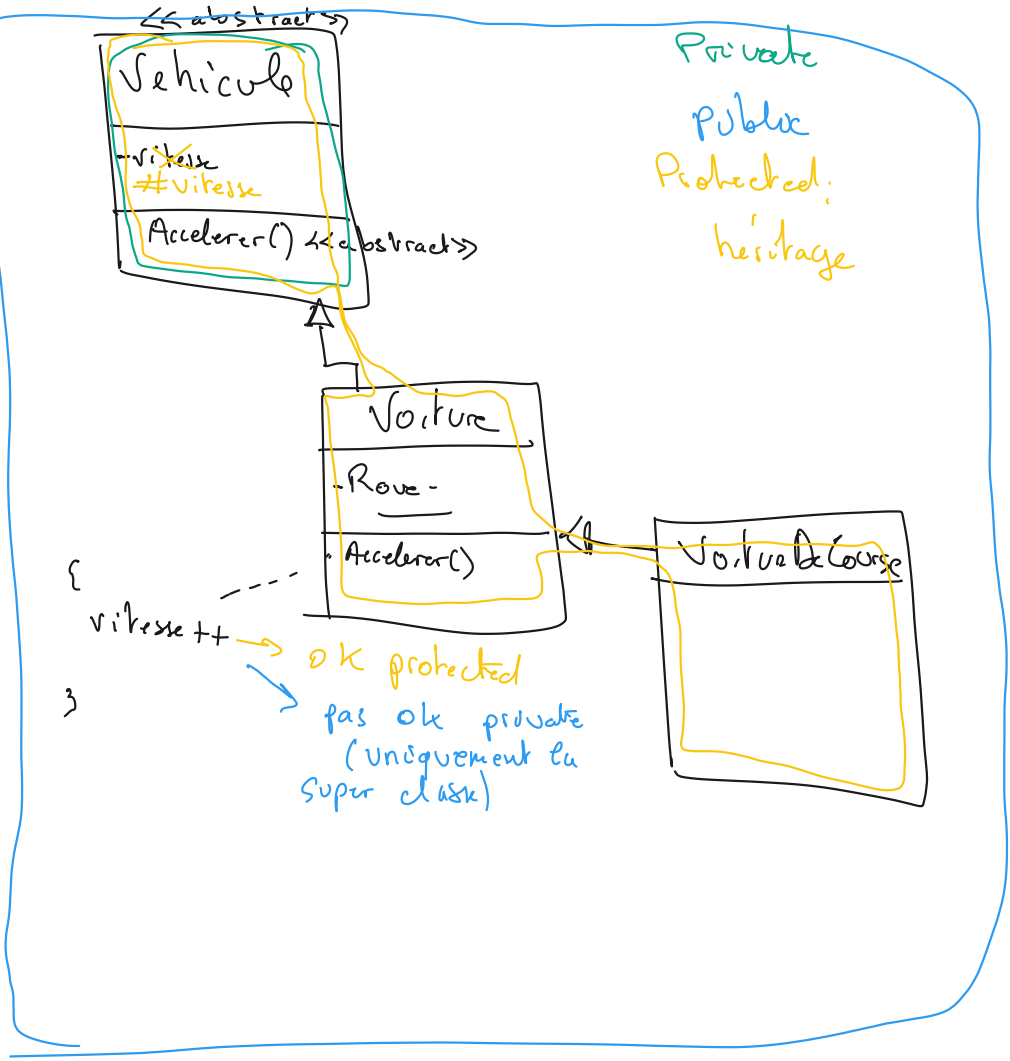


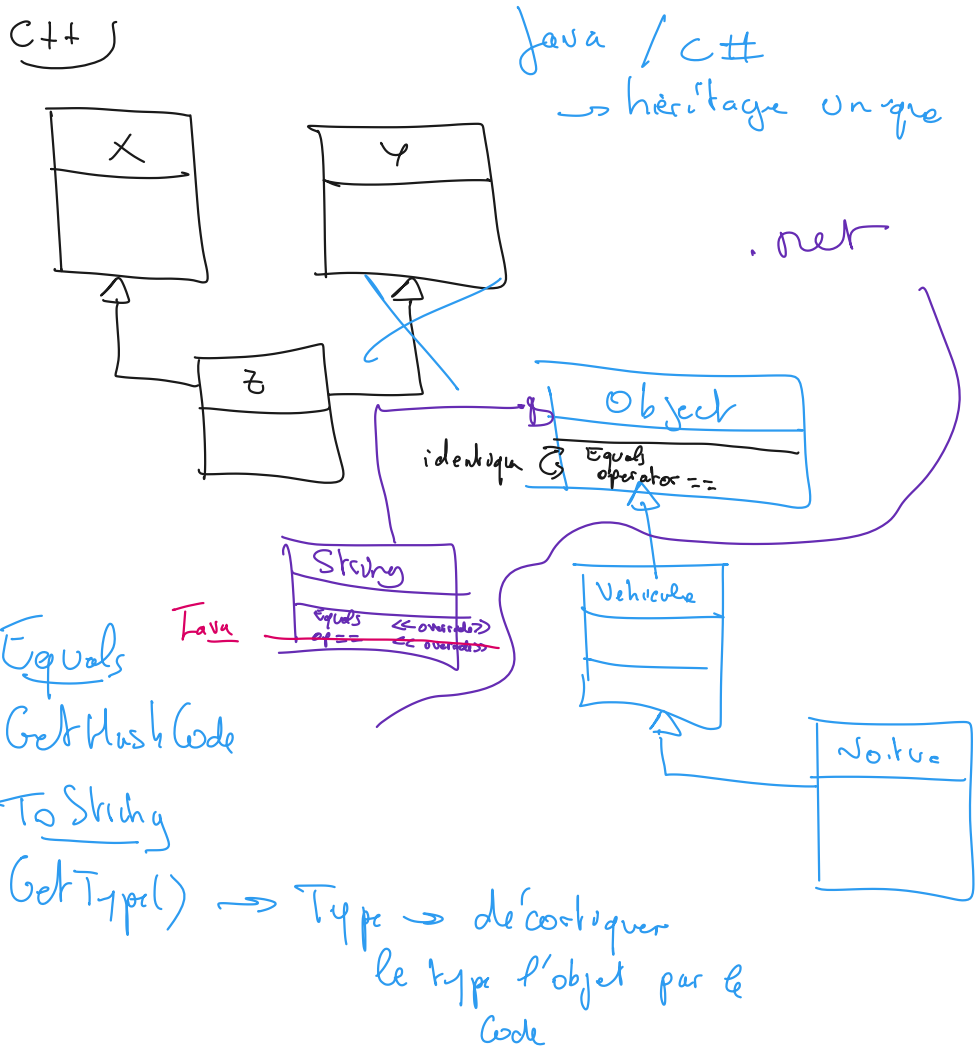
for(;;;) → 1.000.000

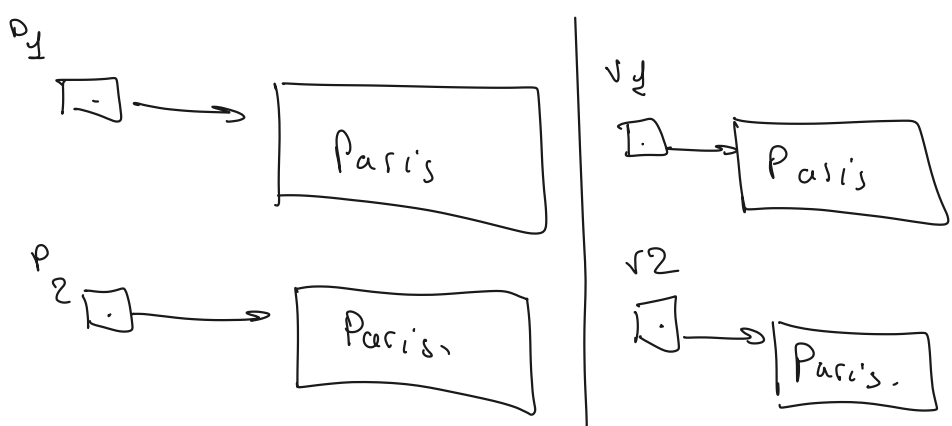
↳ 1 coeur / 1 Thread

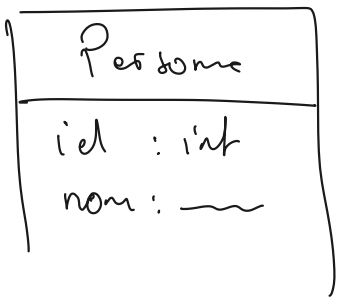
Encapsulation

"être dans une capsule"


Private
Public
Protected
héritage
<<abstract>>
Vehicule
-vitesse
#vitesse
Accelerer() <<abstract>>
Voiture
-Roue-
Accelerer()
VoitureDeCourse
{
vitesse ++
}
ok protected
pas ok private
(uniquement la
super classe)


C++
X
Y
Z
Java / C#
→ héritage unique
.net
Object
identique
Equals
operator ==
String
Equals
==
<<override>>
<<override>>
Java
Vehicule
Voiture
Equals
GetHashCode
ToString
GetType() → Type → décortiquer le type l'objet par le code


P1
Paris
P2
Paris
V1
Paris
V2
Paris.



Personne p1 → id: 1

p2 → id: 1

if (p1 == p2)

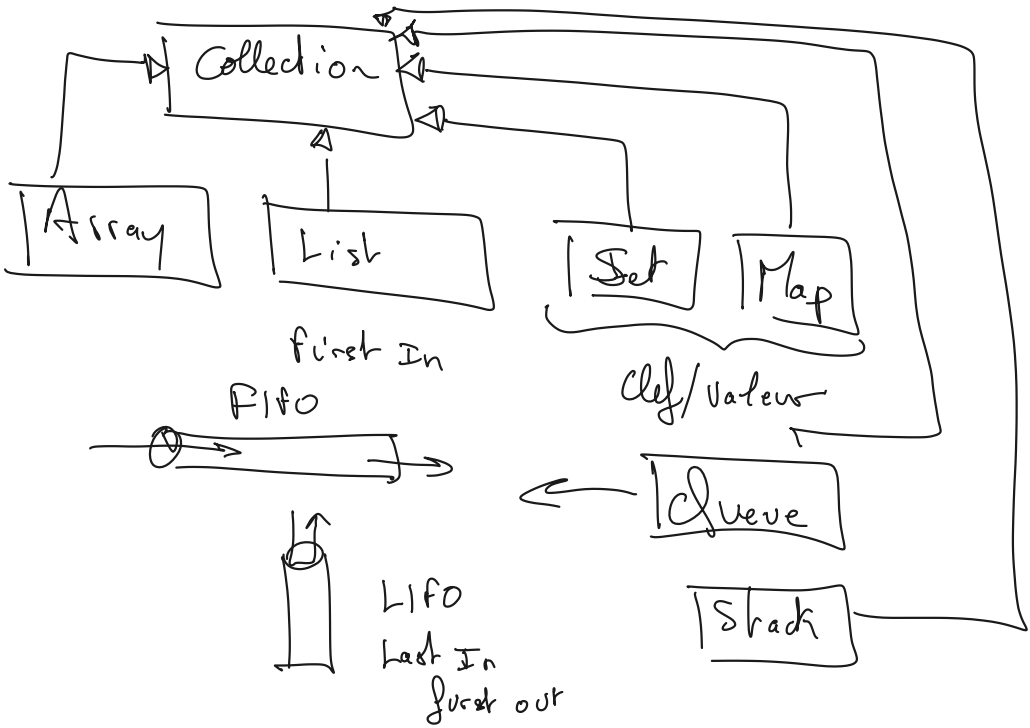
= surcharger — Equals

- operator==

ToString() = Comportement par

Default = FQDN

<NameSpace>.<Type>



C# 1 (2002, 2003) → obsolete

var list = new ArrayList();

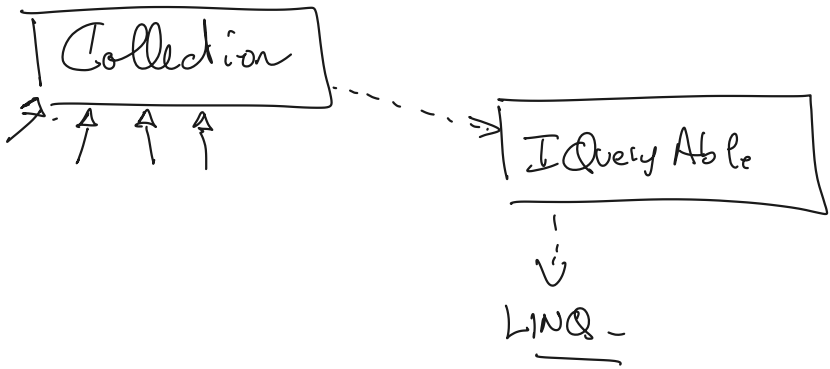
C# 2 > (2005...)

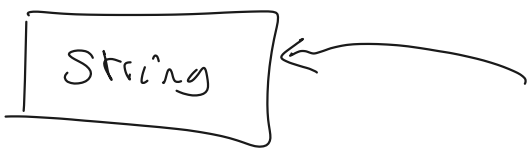
var s = new List<String>();

"List of string"

ok!

Language Integrated Native Query





avoir une méthode .md5()

```
string s = "ole"
s.md5();
```

"méthodes d'extension"
→ injection statique de code (à la compilation)

LINQ
↓
GroupBy (collections)

EF
↓
Entités (sans requêtes)

LINQ

→ "API Criteria"

Requêtes :

```
from _ in coll
select .....
```

```
coll.where( <predicat> )
  .Take(int).
  .orderby( <critères> )
  .groupby(  )
  .etc..... ();
```

```
void f()
void f2()        → signature : f
```

```
{
var vpf = f;
```

vpf : Type adresse vers la fonction

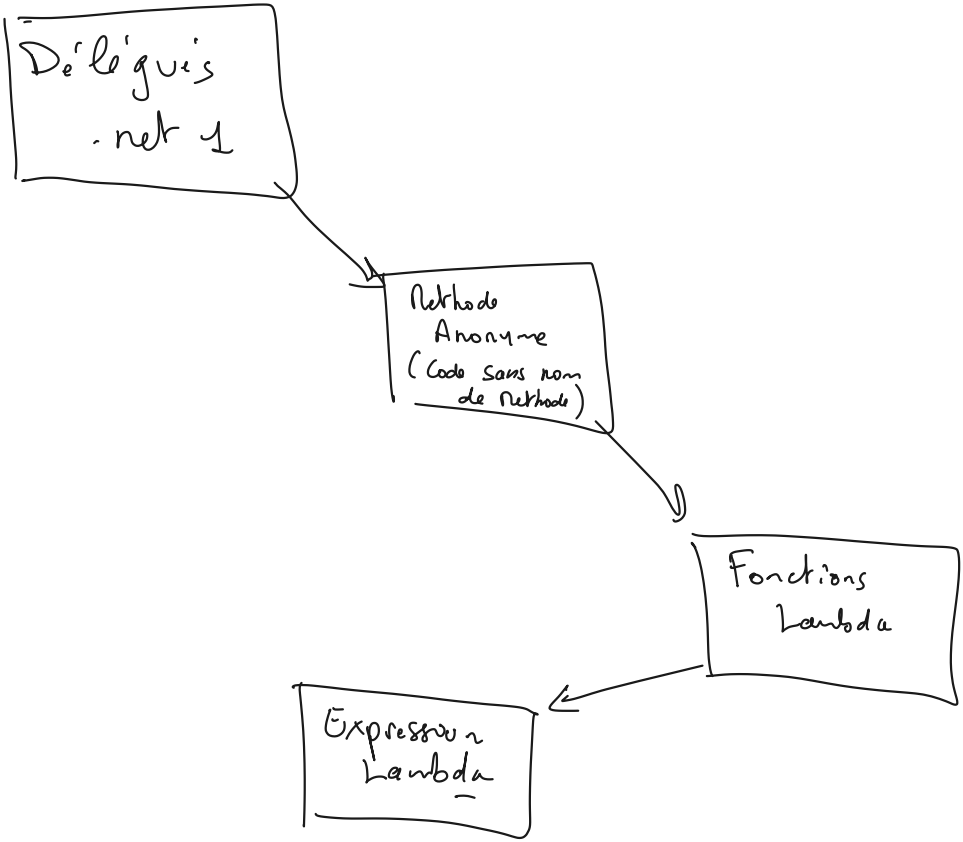
vpf() : opérateur d'appel de fonction

f() : adresse de la fonction

vpf = f2; → possible

vpf();

```
namespace ConsoleVariables
{

   class Voiture
   {

   }

   internal class Program
   {
      static void Main(string[] args)
      {
         var ints = new int[] { 1, 3, 4, 5 };

         var villes = new List<Ville>() {
            new Ville() { Nom="Paris"},
            new Ville() { Nom="Lille"},
            new Ville() { Nom="Rennes"},
            new Ville() { Nom="Valenciennes"}
         };
         villes[0].md5();    // utilisation de mon extension

         // modèle LINQ
         var villesdunord = from v in villes
                        where v.Nom == "Lille" || v.Nom == "Valenciennes"
                        orderby v.Nom descending
                        select new { VilleMajuscule = v.Nom.ToUpper() };

         // modèle CRITERIA
         var villesdunord2 = villes.Where( v => v.Nom == "Lille" || v.Nom == "Valenciennes")
            .OrderByDescending( v => v.Nom).
            Take(1);

         foreach (var v in villesdunord)
            Console.WriteLine(v);

         foreach (var v in villesdunord2)
            Console.WriteLine(v);
      }
   }

   static class MesExtensions
   {
      public static int md5(this Ville v ) // Définition d'une méthode d'extension
      {
         return 0;
      }
   }

   class Ville
   {
      public string Nom { get; set; }

      public override string ToString()
      {
         return Nom;
      }
   }
}
```


Délégués
.net 1
Méthode
Anonyme
(Code sans nom
de méthode)
Fonctions
Lambda
Expression
Lambda

```
using static ConsoleDelegues.Program;

namespace ConsoleDelegues
{
   class DemoLambda
   {
      // déclaration de la signature :
      public delegate int Calcul(int a, int b);
      Calcul tc = null;
      public void DefinirLeTypeDeCalcul()
      {
         // L'expression Lambda est une simplification de la syntaxe de la méthode Lambda
         // Sytaxe de méthode lambda :
         tc = (int a, int b) => { return a * b; };

         // Expression Lambda :
         tc = (a, b) => a + b;
      }
      public void Demo()
      {
         // je vais vouloir faire un calcul,
         // mais le type de calcul, est défini dynamiquement

         DefinirLeTypeDeCalcul();
         int resulat = tc(10, 2);    // ici prog fonctionnelle
         Console.WriteLine(resulat);
      }
   }

   class DemoAlarme
   {
      // 1 Déclarer la signature de l'évènement.
      public delegate void Alarme(object sender, EventArgs e);

      static void AlarmeSonore(object sender, EventArgs e)
      {
         Console.WriteLine("Beeeeeeeeep");
      }
      static void AlarmeVisuelle(object sender, EventArgs e)
      {
         Console.WriteLine("Flash visuel.....");
      }
      static void AlarmeDeMontre(object sender, EventArgs e)
      {
         Console.WriteLine("Bzzzzzzzzz");
      }

      // 2 déclarer un délégué ( pointeur sur... )
      public event Alarme alarmes = null;


   internal class Program
   {


      static void Main(string[] args)
      {
         new DemoAlarme().Demo();
         new DemoLambda().Demo();
      }
   }
}
```

```
public void Demo()
    {
        var adm = new Alarme(AlarmeDeMontre);

        string c;
        do
        {
            Console.WriteLine("1 : ajouter une alarme visuelle");
            Console.WriteLine("2 : ajouter une alarme sonore");
            Console.WriteLine("3 : ajouter une alarme de montre");
            Console.WriteLine("4 : ajouter une alarme Odorante par fonction anonyme");
            Console.WriteLine("5:  ajouter une alarme Flash par méthode Lambda");
            Console.WriteLine("9 : Supprimer une alarme de montre");
            Console.WriteLine("0 : sortir");
            Console.WriteLine("" : ne rien faire");
            c = Console.ReadLine();
            switch (c)
            {
                case "1":
                    // 3 AJouter des évènements au délégué
                    alarmes += new Alarme(AlarmeVisuelle);
                    break;
                case "2":
                    alarmes += new Alarme(AlarmeSonore);
                    break;
                case "3":
                    alarmes += adm;
                    break;

                case "4":
                    // Methode anonyme :
                    alarmes += delegate (object sender, EventArgs e) {
                        Console.WriteLine("Ca pue....");
                    };
                    break;

                case "5":
                    // Methode Lambda : ( retrait de delegate et ajout de =>   )
                    alarmes += (object sender, EventArgs e) => {
                        Console.WriteLine("Flash Flash Flash.....");
                    };
                    break;

                case "9":
                    alarmes -= adm;
                    break;
                case "":
                    {
                        Console.WriteLine("On déclenche l'alarme ? (y/n)");
                        string k = Console.ReadLine();
                        if (k == "y")
                        {
                            // 4 Déclencher l'évènement
                            if (alarmes != null)
                                alarmes(null, new EventArgs());
                        }
                    }
                    break;
            }
        } while (c != "0");
    }

}
```

Events

Dim 1

Syntaxe
- délégués
- méthanonym
- Lambdas

Use Cases
- Client Lourds
  Le Très fréquent
- ASP.net MVC
  Le Très Rare

ex: - WebSocket sur paquet entrant
- Plugin sur filtres, events du Serveur web
ex: Sur Exception (globale)

- 2 paramètres non générés
  - objet sender
  - "base de" EventArgs

Pattern ?

Pattern pour les Events

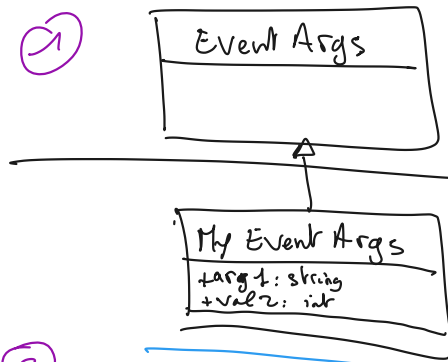
- event sans
  - objet sender
  - EventArgs
  } OUI

1er Cas: Pas de Params

↳ Pattern NS Event

1 arg = objet sender
2nd EventArgs e
} "EventHandler" est un délégué de ce type

2nd Cas : besoin de passer aux Events

① 

↔ postfixer par "EventArgs"

②

public delegate MyDelegate(object sender, MyEventArgs);

Postfixer par "Delegate" ou "Event"

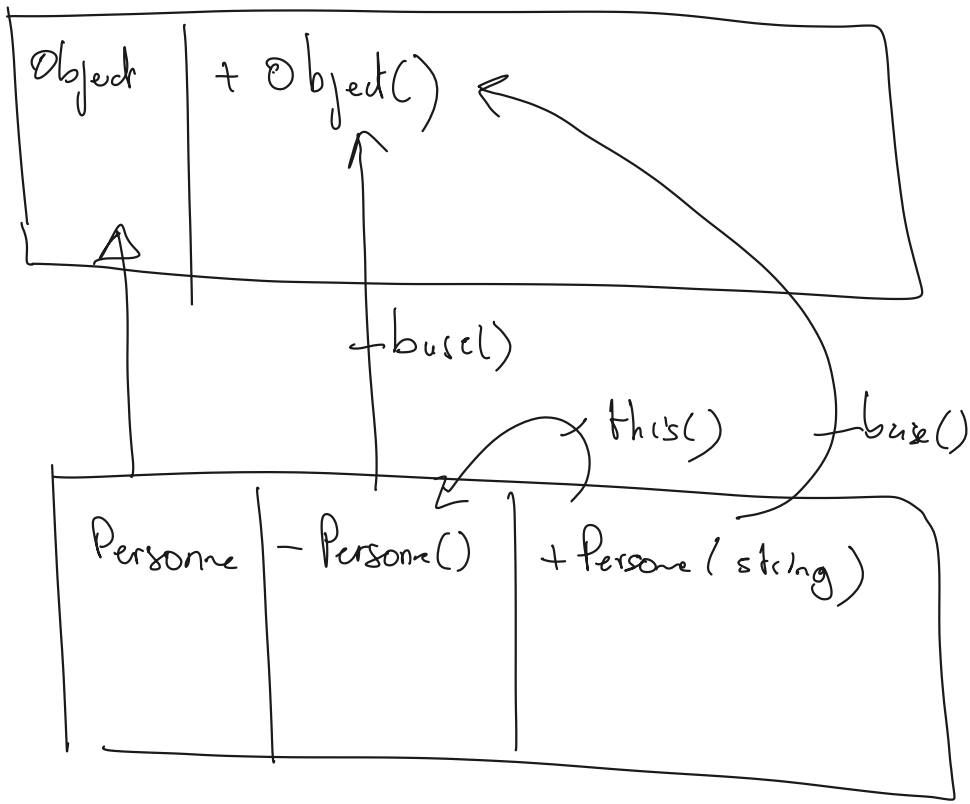
ex public delegate SurchauffeEvent(object sender, SurchauffeEventArgs e)

---

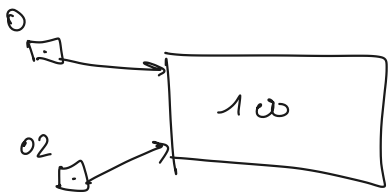
ctor()

- Par Défaut = 1 ctor par défaut (sans params), public

- Si Ajout d'un 1 ctor paramétré, le ctor par défaut passe privé




```
class PersonneException : Exception
  {
    public PersonneException( string message ) : base(message)
    {
    }
  }

  class Personne
  {
    static public int population;

    // mode Java :
    private int age;
    public int GetAge()
    { return age; }
    public void SetAge( int age )
    {
      if( age>=0 && age <=100)
        this.age = age;
      else
        throw new PersonneException("L'age fourni( "+age+" ) est incorrect.");
    }

    // Methode .Net recommandée :
    private string name;
    public string Name {
      get { return name; }
      set { name = value; }
    }

    // + simple ( champ implicite )
    public bool Cheveux { get; set; }

    public void direBonjour(string p)   // signature = direBonjourstring
    {
      string v;
      Console.WriteLine("Bonjour, je suis "+ name);
    }
  }
```



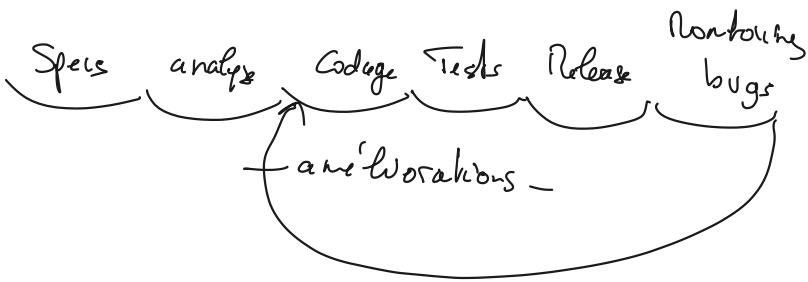
02 = 0;

int j = (int) 0.

Membres statiques :

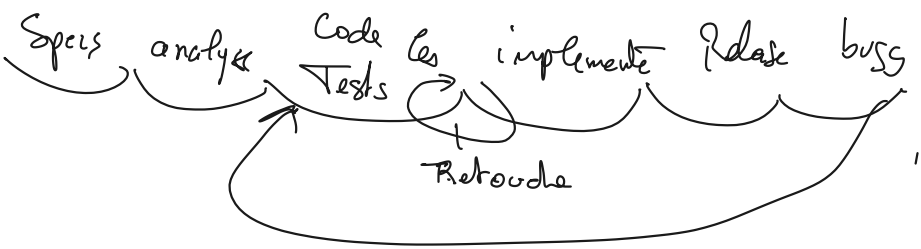
```
class Cercle
  {
     public float Rayon { get; set; }
     public static float PI { get; }

     static Cercle()
     {
        PI = (float)Math.PI;
     }

     public Cercle( float rayon )
     {
        Rayon = rayon;
     }

     public float Surface
     { get { return Rayon * Rayon * PI ;  } }
  }
```

Cycle de vie d'une App (Agile, Scrum, ...)



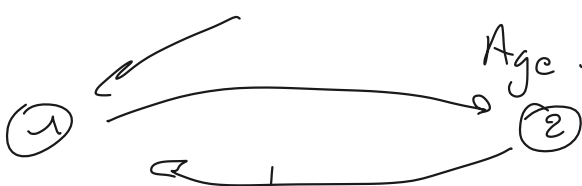
Methode TDD / BDD

Test Behaviour Driven Development



Analyse → class Personne

Age → 0..100



PersonneTest Tests

Personne
Age

TestSetAge()

→ Cas de Tests

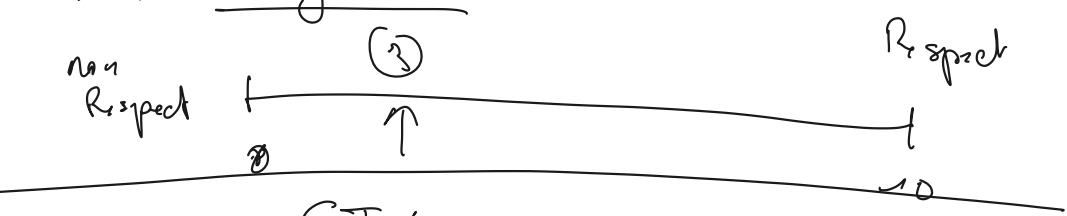
Age = 50

Age = 0

Age = 5

Age = 150

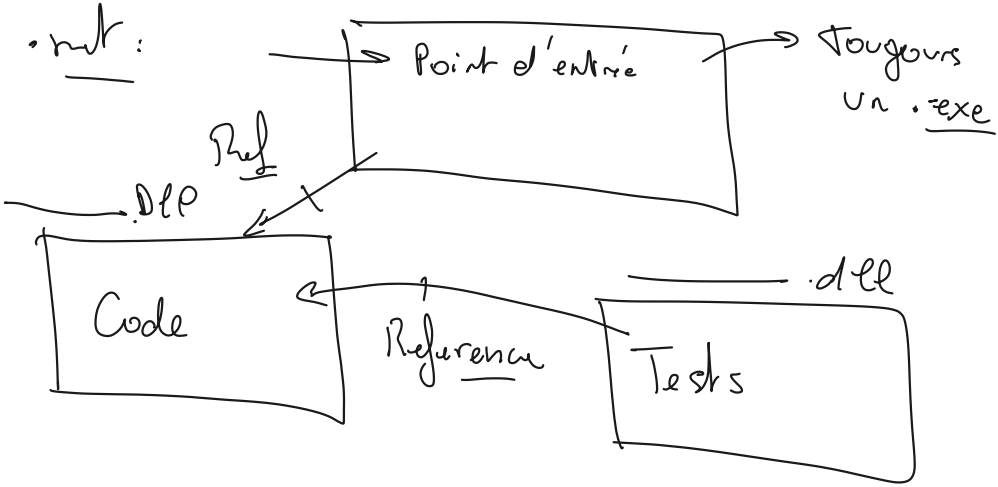
Test en général



CI/CD

build (côté SW)

Tests

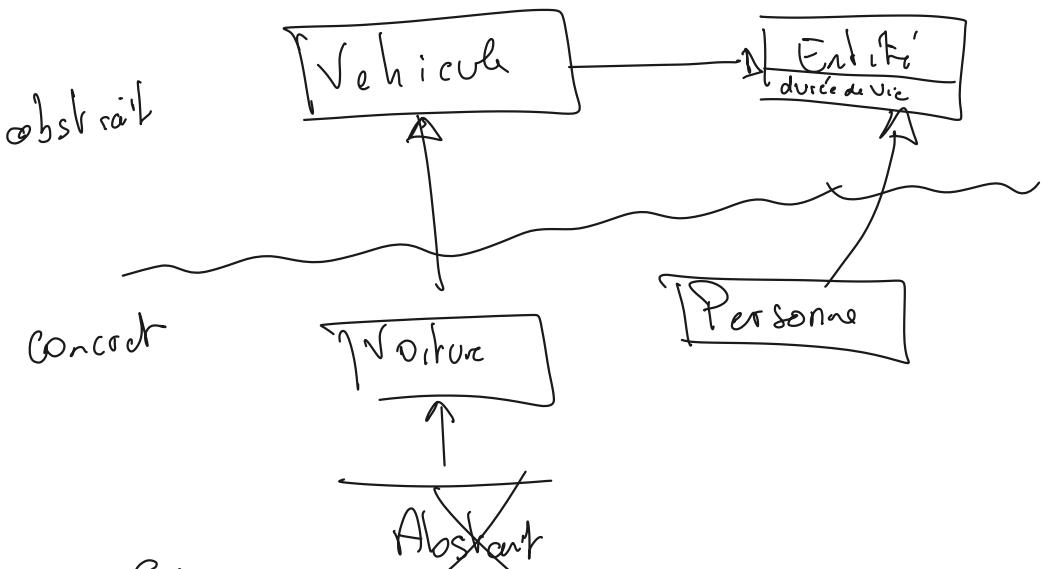


classe → Concrète ?

Object → Concret

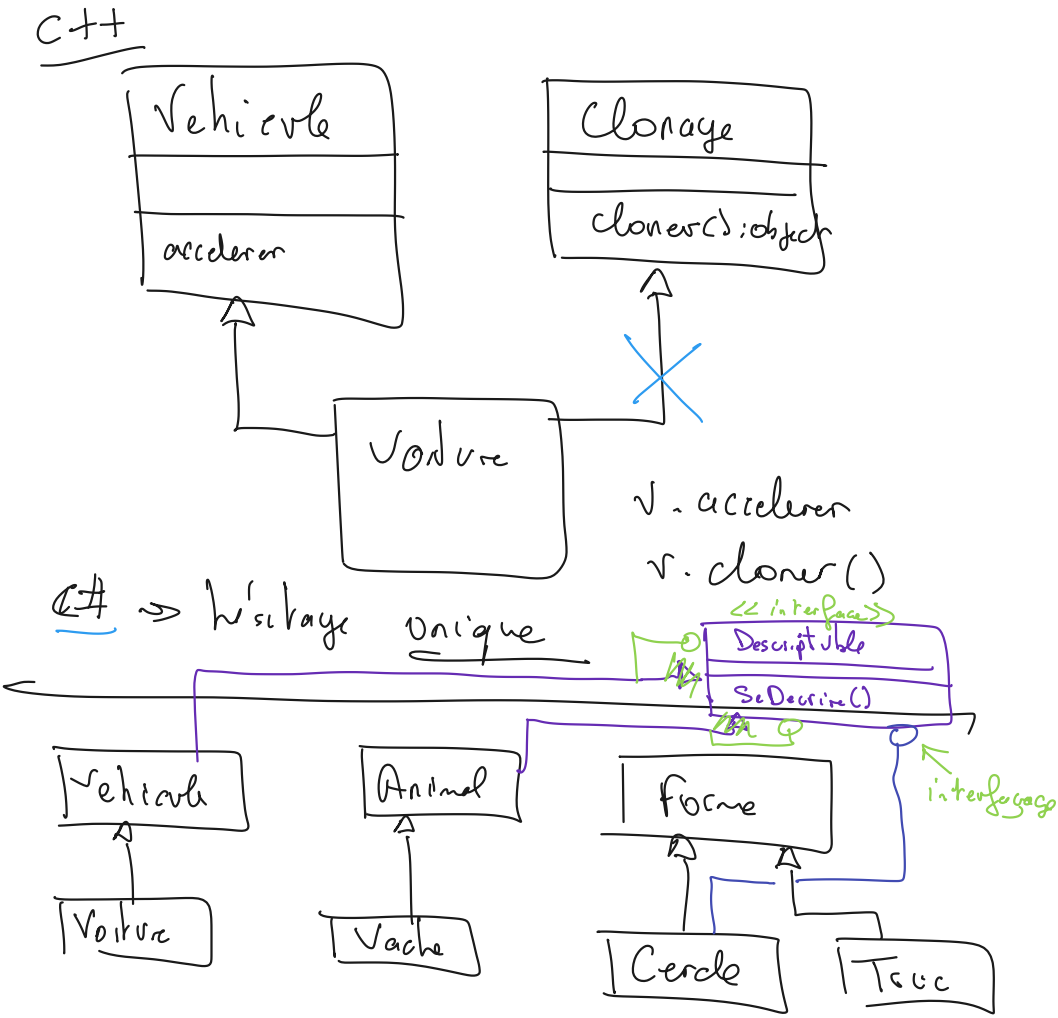
Sans specs → //

Si "abstract" → Abstrait



C++ → héritage multiple

C#
Java } // Unique


C++
Vehicule
accelerer
Clonage
cloner():object
Voiture
v.accelerer
v.cloner()
C# => héritage unique
<< interface >>
Decriptible
SeDecrire()
Vehicule
Animal
Forme
Voiture
Vache
Cercle
Truc
interfaçage
=> Ces classes doivent pouvoir se Décrire via l'usage :
for(var monobjet : mes objets)
monobjet.SeDecrire();
Liste de voiture, Lion, Cercle, ...

en C#:

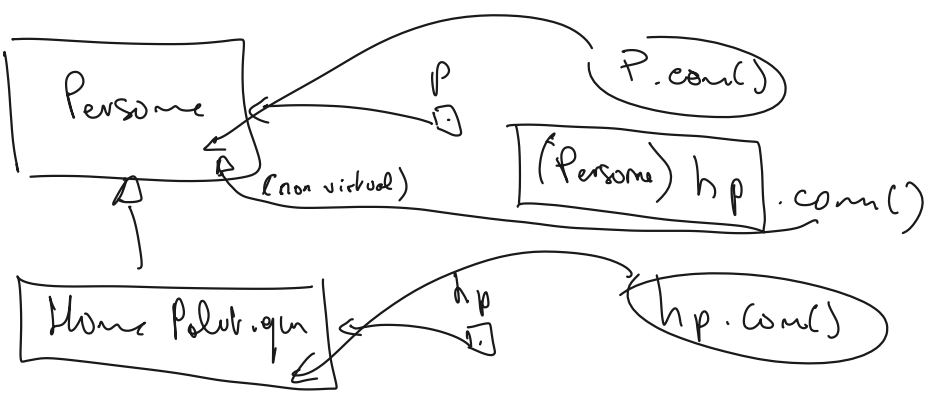
héritage unique,
interfaçage multiple ok

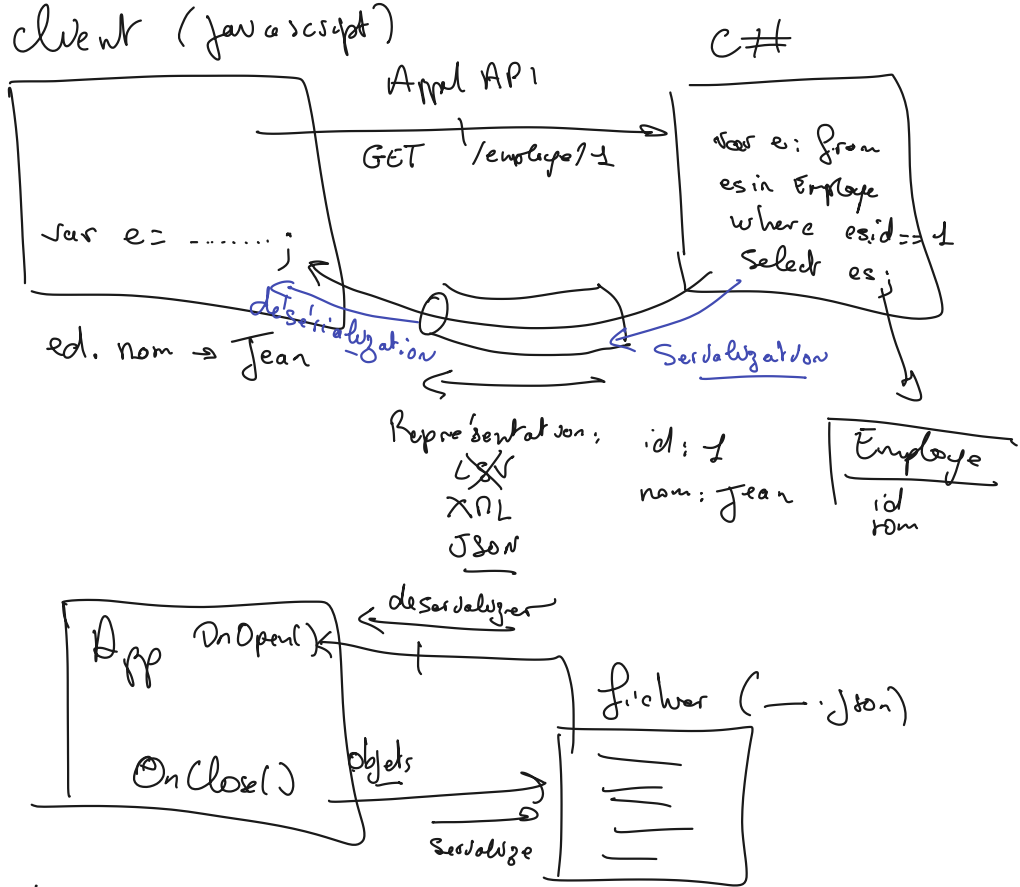
Une interface est "comme" une classe purement abstraite

Purement : 100% des membres sont abstraits

mot clef - class
- interface } différent

"on hérite d'une classe"
"on implémente une interface"





1) O.U ? → Fichier
Mémoire
Réseau
} flux (Stream)

2) Comment ? → - binaire
- JSON
- XML
} format

1998 2002 → .net

VB
C#
} ADO ActiveX Data Objects → ~~ADO~~.net

Forme
PrintFile
"DataReader"

→ Ouverture d'un curseur
→ lecture
→ ligne suivante
→ fermeture du curseur

DQL
ExecuteReader()

DML
ExecuteNonQuery
(PS, Insert, Update, Delete)

ADO.net DataSet (+ puissant)

avec
- WinForm
- WPF
- ASP.net (~~WebForm~~)
  - .net FW MVVM
  - obsolète

Formulaire

Produits

DataSet

en RAM

Produits

Fill "grille de données"

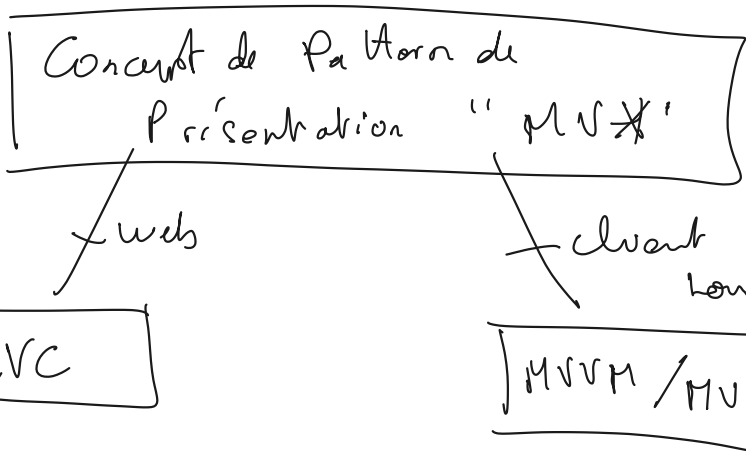
Contrôle de Type

Update

DataAdapter
SelectCommand
InsertCommand
DeleteCommand
UpdateCommand

Produits

(Jeux, buffer de données en mémoire)



ASP.net MVC

→ ADO.net DataReader → Ok

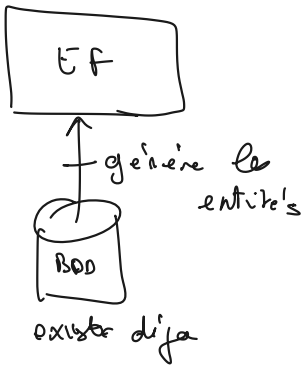
→ ADO.net DataSet → Trop lourd, Pas adapté

→ Recommandé : - ADO.net Entity Framework
+ LINQ

Approche "Bottom-up" → vous avez déjà un schéma de BDD

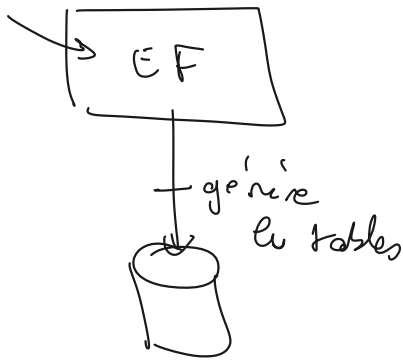
Approch "Top-down" → EF va créer le schéma de BDD.

Bottom Up



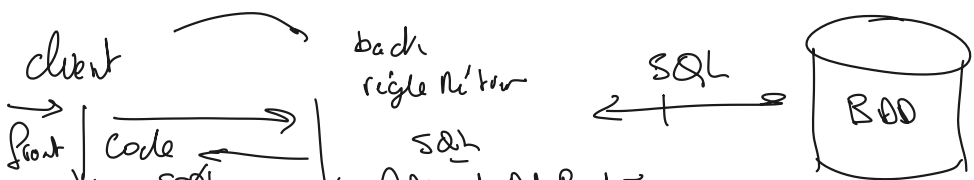
Top Down

Je crée les Entités



Une Entité est ..... une classe dont le rôle est de représenter une ligne de Table

---



→ OK si nb table ≤ 5

Design Pattern d'architecture Object Relationnel Mapping

client

1 Table = 2 Classes

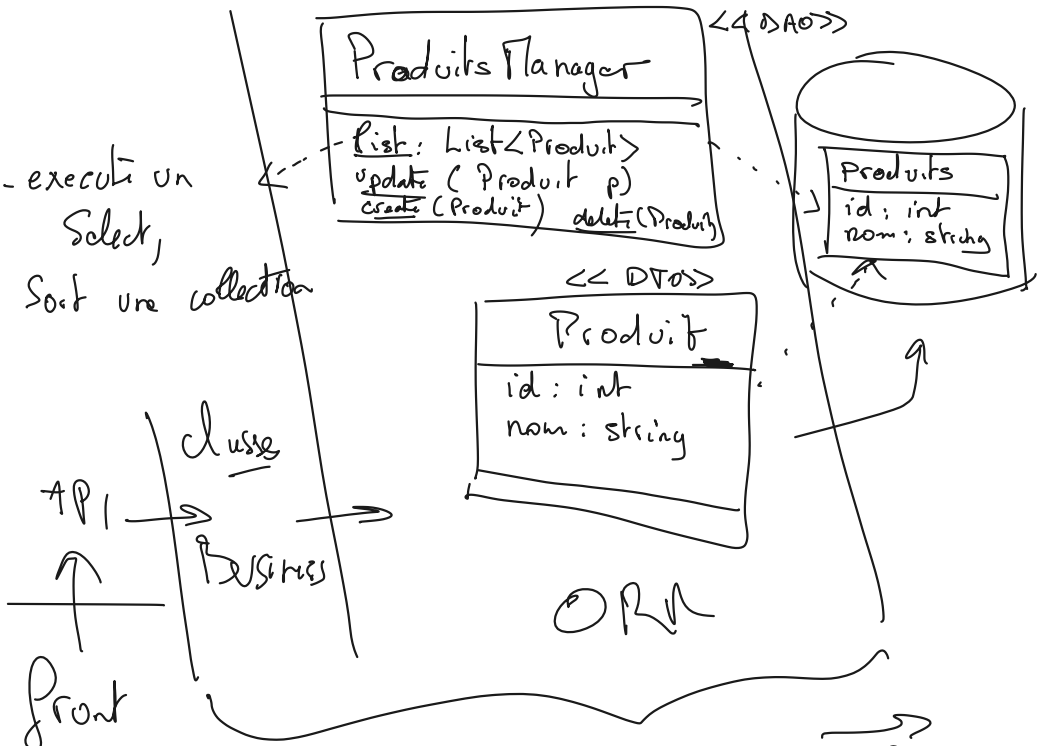
- 1 classe DAO pour l'accès à la table
- 1 classe DTO pour représenter les lignes (Entité)

Data Access Object

→ La classe gère les ordres d'accès

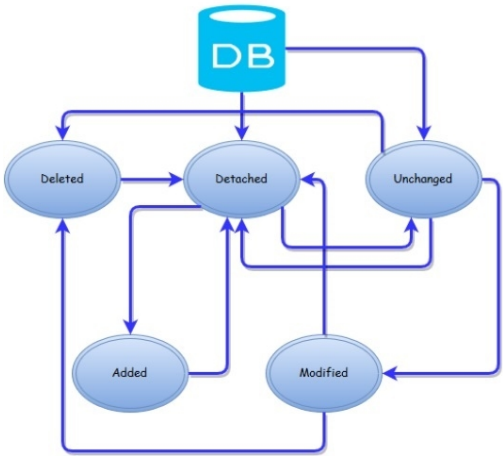
Data Transfer Object

→ La classe dont une instance représente une ligne



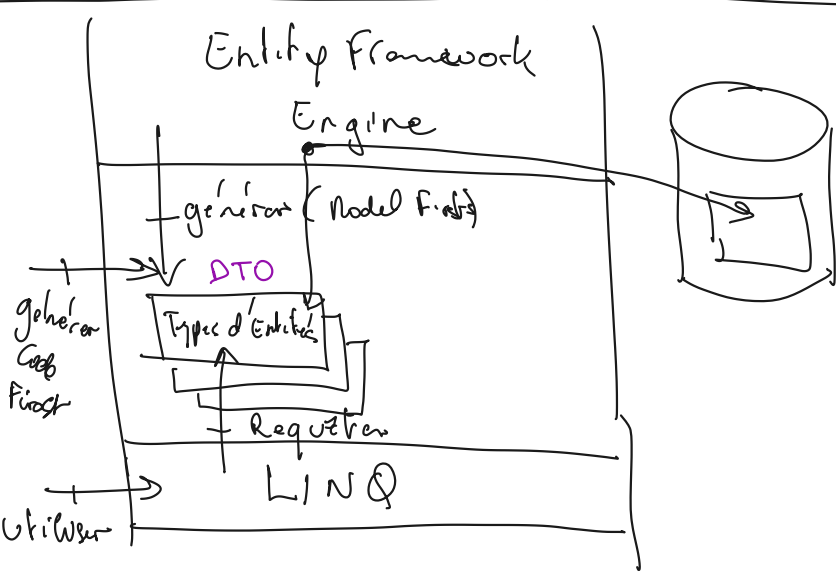
- Pas de PS
- Pas de Triggers

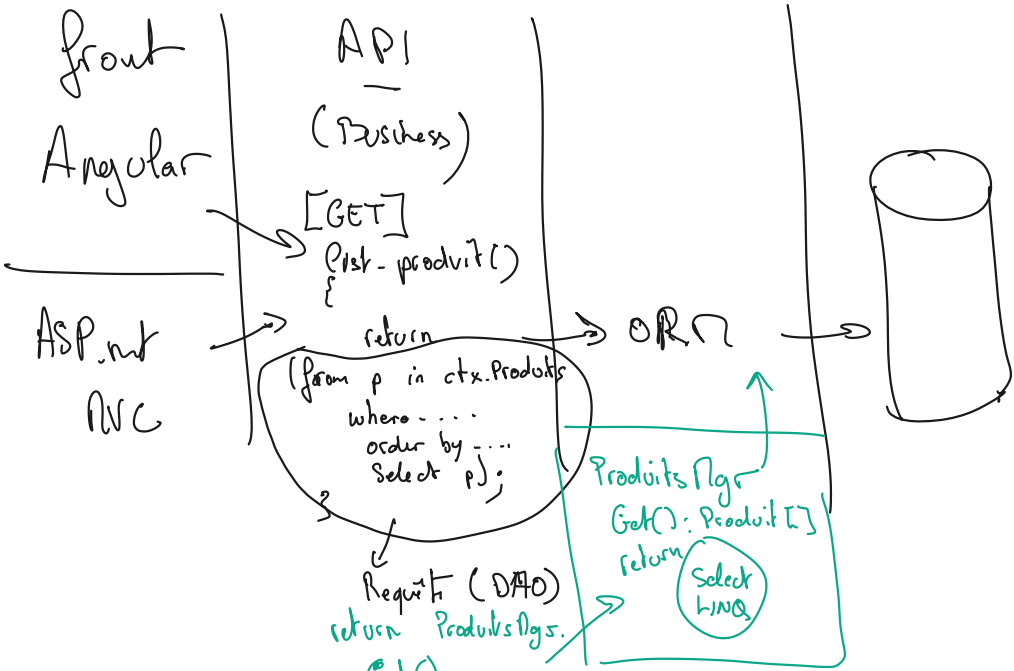
Une entité a un cycle de vie !



ex : Product p = new Product(); (Entité Poco)
p.id = 1 .....
ProductManager.create(p);

" Une entité est "sale" si elle a été changée mais pas synchronisée "

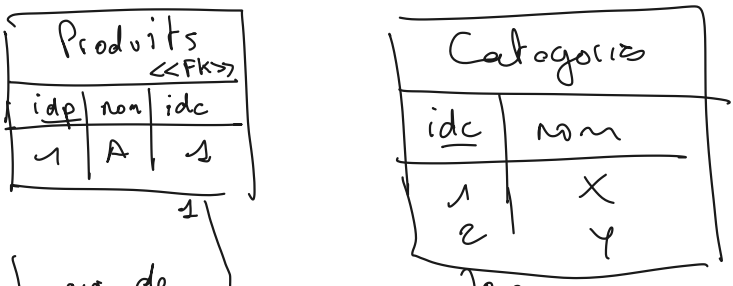




Q? Model First + bottom up
On a des DTO? → OUI

Q?: Code First sans BDD
doit on créer les DTO? → OUI
(Sauf s'il y a un assistant)

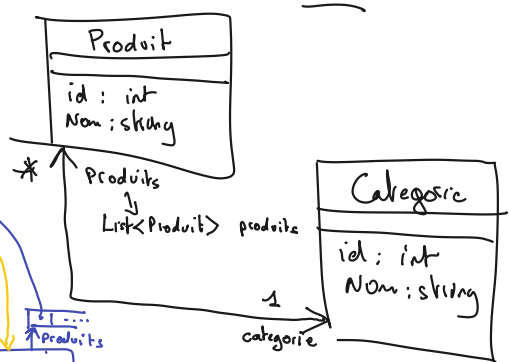
Q? : Est ce que EF peut
générer des DAO? → Jamais
(c'est une bonne pratique de le faire s'il est nécessaire)



changer de Catégorie : Update Produits set idc = 2 where idp = 1 } ok!

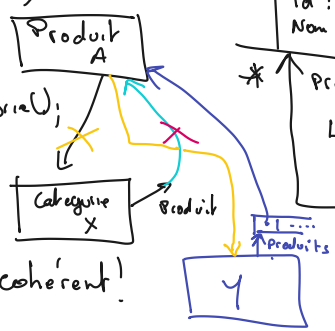
Je créé mon Moteur d'ORM — DTO
— DAO

UML :



Code client :

```
Produit p = new Produit();
p.id = 1;
p.nom = "A";
Categorie c = new Categorie();
c.id = 1;
c.nom = "X";
p.categorie = c;
```

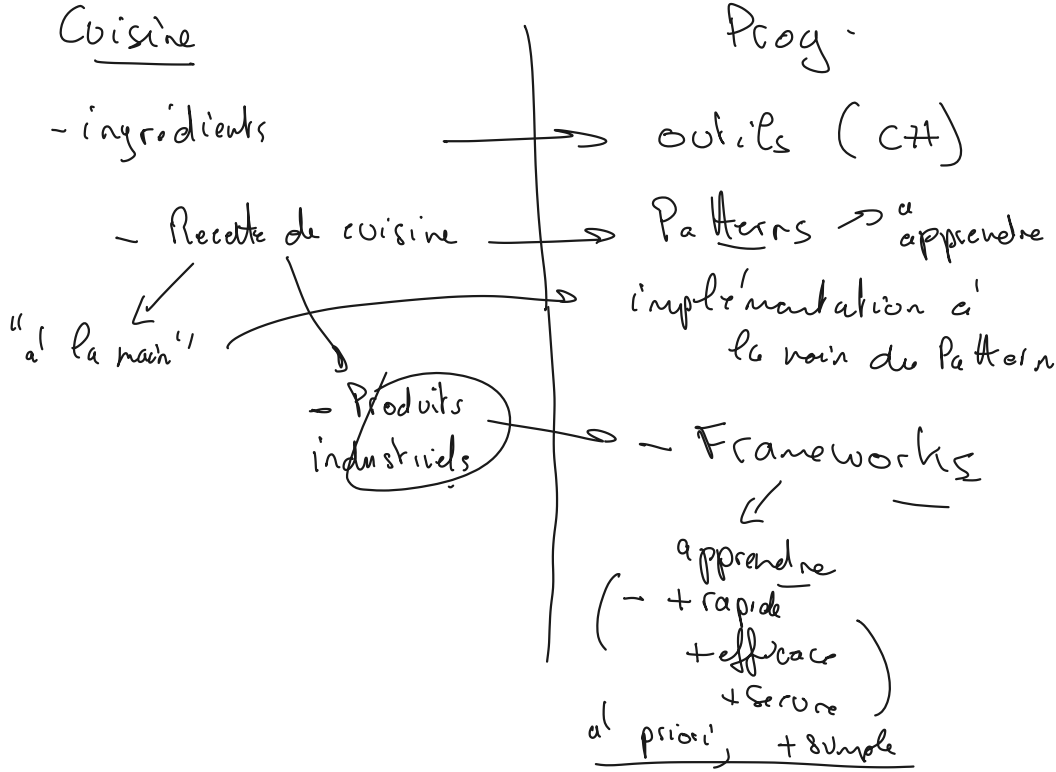


Pas cohérent !

```
c.produits.Add(p)
```

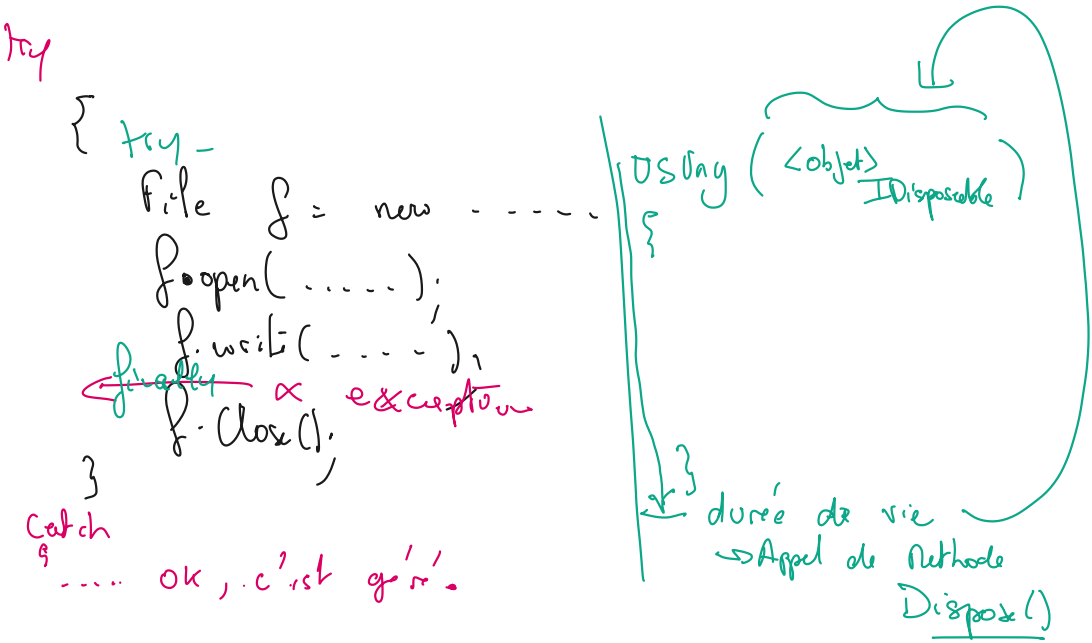
Code :

```
Categorie c2 = new Categorie(2, "y");
// objectif : changer la catégorie de A
.c2.produits.App(p)
.P.categorie = c2;
C.produits.Remove(p);
```



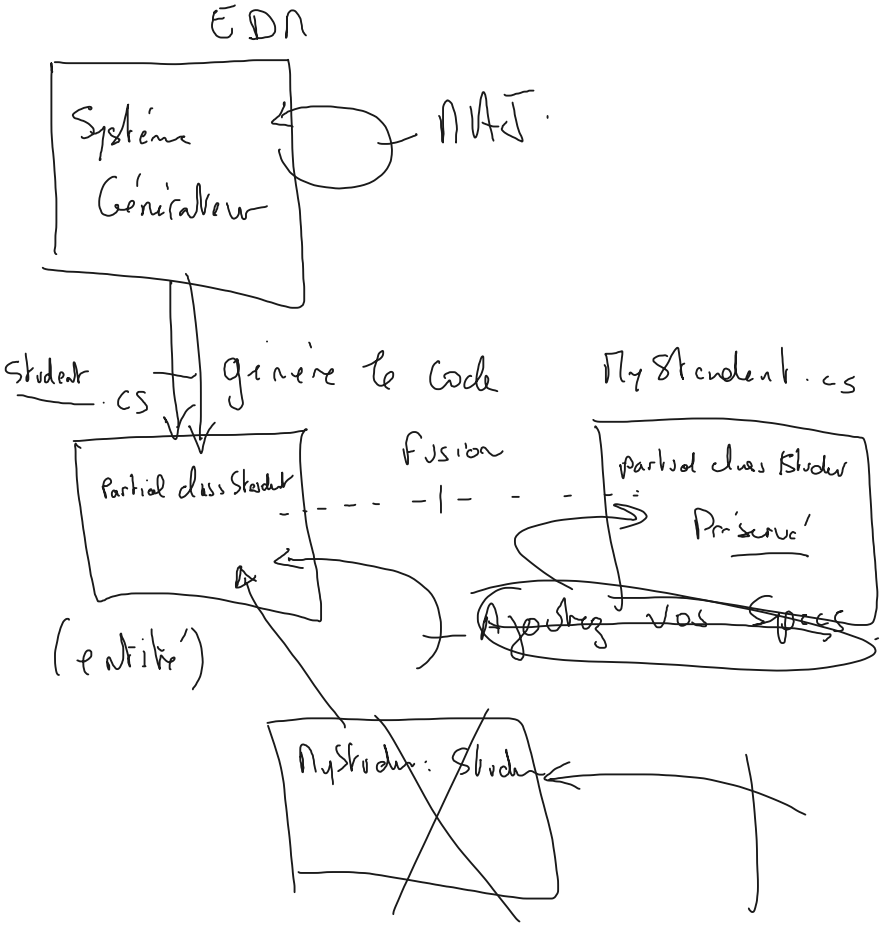

```
// Instancier le moteur d'entités
        var ctx = new AdventureWorksEntities();

        using (var tran = ctx.Database.BeginTransaction())
        {
            foreach (var psc in ctx.ProductSubcategories)
            {
                Console.WriteLine("Sous catégorie " + psc.Name + ", id : " +
psc.ProductSubcategoryID);
                psc.Name = psc.Name.ToUpper();
                Console.WriteLine("Sous catégorie " + psc.Name + ", id : " +
psc.ProductSubcategoryID);
                foreach (var p in psc.Products)
                    Console.WriteLine("  " + p.Name);
            }

            ctx.SaveChanges();
            tran.Commit(); //
        }
```

using (var tran = ctx.Database.BeginTransaction())



Avec Linq et transaction :

```
var ctx = new AdventureWorksEntities();

        using (var tran = ctx.Database.BeginTransaction())
        {
            foreach (var psc in (from psc in ctx.ProductSubcategories orderby psc.Name
select psc))
            {
                Console.WriteLine("Sous catégorie " + psc.Name + ", id : " +
psc.ProductSubcategoryID);
//                  psc.Name = psc.Name.ToUpper();
//                  Console.WriteLine("Sous catégorie " + psc.Name + ", id : " +
psc.ProductSubcategoryID);
                foreach (var p in psc.Products)
                    Console.WriteLine("  " + p.Name);
            }

            ctx.SaveChanges();
            tran.Commit();
        }
    }
```


EDM
Système Générateur
MAJ.
Student.cs
génère le Code
MyStudent.cs
Fusion
Partial class Student
Partial class Student
Privé
(entité)
Ajoutez vos Specs
MyStudent : Student

EF7 Sous ASP.Net MVC : (mode code first)
basé sur : https://learn.microsoft.com/fr-fr/aspnet/core/data/ef-mvc/intro?view=aspnetcore-7.0
Prérequis Packages NuGet :

Microsoft.EntityFrameworkCore
Microsoft.EntityFrameworkCore.SqlServer
Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore
Microsoft.Extensions.Configuration
EntityFramework.fr
EntityFramework
Microsoft.EntityFrameworkCore.SqlServer
MSTest.TestAdapter
MSTest.TestFramework
Microsoft.NET.Test.Sdk
coverlet.collector

Program.cs :

```
using Microsoft.Extensions.Configuration;
using WebApplicationEF.Data;
using Microsoft.EntityFrameworkCore;

using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

....
builder.Services.AddDbContext<SchoolContext>(options =>
            options.UseSqlServer(builder.Configuration.GetConnectionString(("DefaultConnection"))));
        // fourni par : Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore
        builder.Services.AddDatabaseDeveloperPageExceptionFilter();

var app = builder.Build();
        CreateDbIfNotExists(app);

private static void CreateDbIfNotExists(IHost host)
    {
        using (var scope = host.Services.CreateScope())
        {
            var services = scope.ServiceProvider;
            try
            {
                var context = services.GetRequiredService<SchoolContext>();
                DbInitializer.Initialize(context);
            }
            catch (Exception ex)
            {
                var logger = services.GetRequiredService<ILogger<Program>>();
                logger.LogError(ex, "An error occurred creating the DB.");
            }
        }
    }
```

appsettings.json (chaine de connexion) :

```
{
  "ConnectionStrings": {
    "DefaultConnection":
"Server=.\\sqlexpress;Database=ContosoUniversity1;Trusted_Connection=True;Encrypt=false;MultipleActiveResultSets=true"
  },
```

SchoolContext.cs :

```
public class SchoolContext : DbContext
  {
    public SchoolContext(DbContextOptions<SchoolContext> options) : base(options)
    {
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
      modelBuilder.Entity<Course>().ToTable("Courses");
      modelBuilder.Entity<Enrollment>().ToTable("Enrollments");
      modelBuilder.Entity<Student>().ToTable("Students");
    }

    public DbSet<Course> Courses { get; set; }
    public DbSet<Enrollment> Enrollments { get; set; }
    public DbSet<Student> Students { get; set; }
  }
```

Course.cs :

```
public class Course
  {
    [DatabaseGenerated(DatabaseGeneratedOption.None)]
    public int CourseID { get; set; }
    public string Title { get; set; }
    public int Credits { get; set; }

    public ICollection<Enrollment> Enrollments { get; set; }
  }
```

DbInitializer.cs :

```
public static class DbInitializer
  {
    public static void Initialize(SchoolContext context)
    {
      context.Database.EnsureCreated();

      // Look for any students.
      if (context.Students.Any())
      {
        return;   // DB has been seeded
      }

      var students = new Student[]
      {
      new Student{FirstMidName="Carson",LastName="Alexander",EnrollmentDate=DateTime.Parse("2005-09-01")},
      new Student{FirstMidName="Meredith",LastName="Alonso",EnrollmentDate=DateTime.Parse("2002-09-01")},
      new Student{FirstMidName="Arturo",LastName="Anand",EnrollmentDate=DateTime.Parse("2003-09-01")},
      new Student{FirstMidName="Gytis",LastName="Barzdukas",EnrollmentDate=DateTime.Parse("2002-09-01")},
      new Student{FirstMidName="Yan",LastName="Li",EnrollmentDate=DateTime.Parse("2002-09-01")},
      new Student{FirstMidName="Peggy",LastName="Justice",EnrollmentDate=DateTime.Parse("2001-09-01")},
      new Student{FirstMidName="Laura",LastName="Norman",EnrollmentDate=DateTime.Parse("2003-09-01")},
      new Student{FirstMidName="Nino",LastName="Olivetto",EnrollmentDate=DateTime.Parse("2005-09-01")}
      };
      foreach (Student s in students)
      {
        context.Students.Add(s);
      }
      context.SaveChanges();
```

Référence de Entity Framework moderne (core 7 et +) : https://learn.microsoft.com/fr-fr/ef/core/

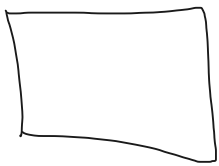
Process → Géré par l'OS

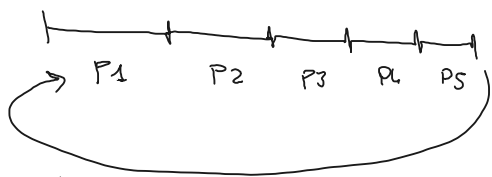
(Point d'entrée - Main)

(Mémoire associé)

1 Process = 1 Thread à la base

1 CPU - 1 Coeur





Priorité de Process / Thread

Windows

Niveaux de Priorité

CRITICAL

REALTIME

Above normal

Normal

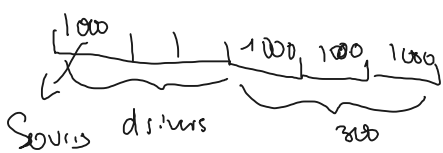
below

idle

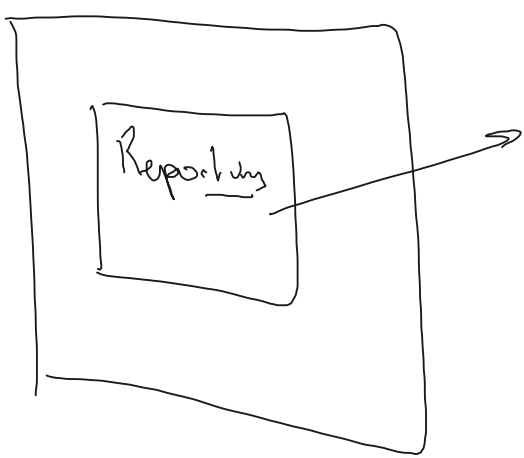
Linux

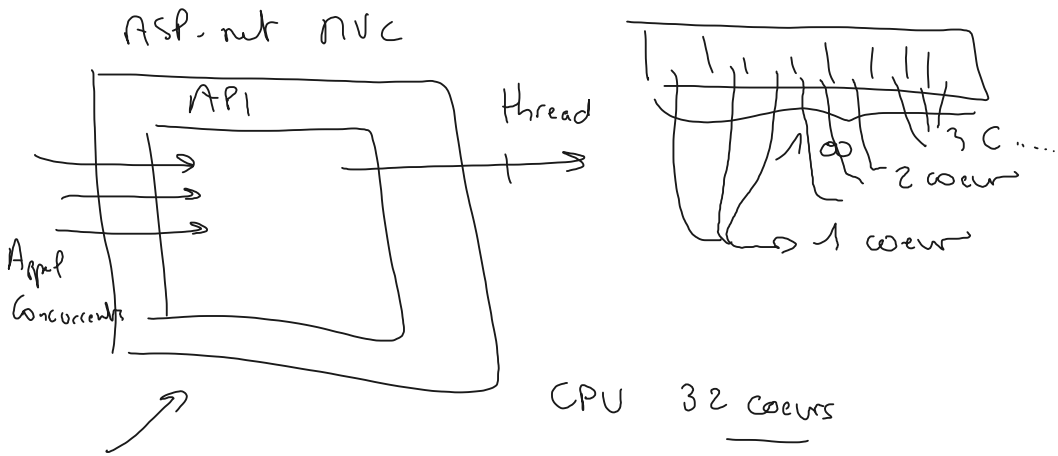
Poids

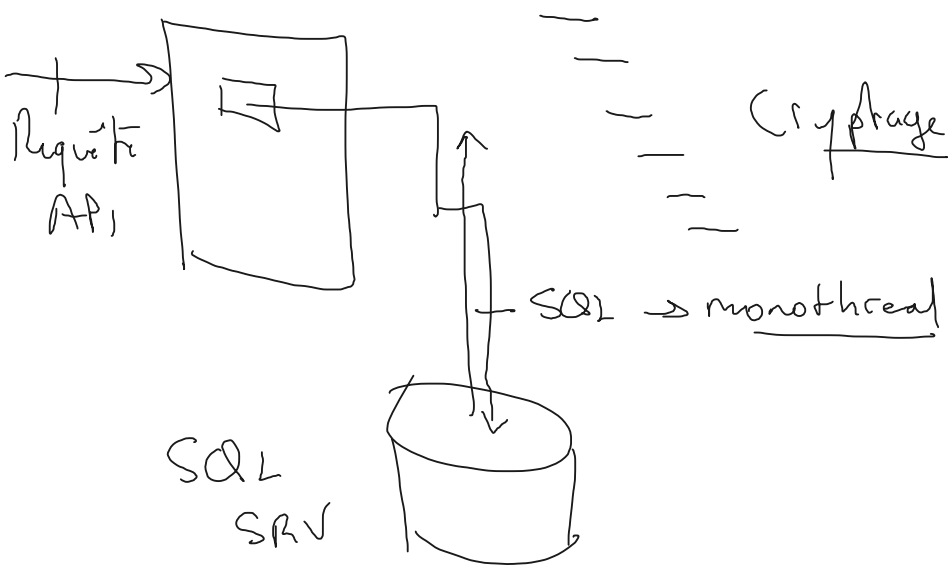
Process = 100



Server → CPU 32 Coeurs







- net 4 → SAP

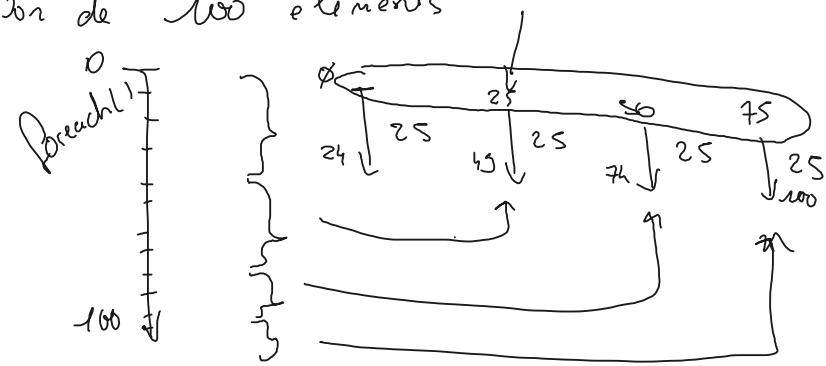
Moi → Core i7 2.5Ghz 8 coeurs

Participants → Core i5 3.2Ghz 4 coeurs

.net (Général) → primitives
. lock { }
Mutex, semaphore, ...

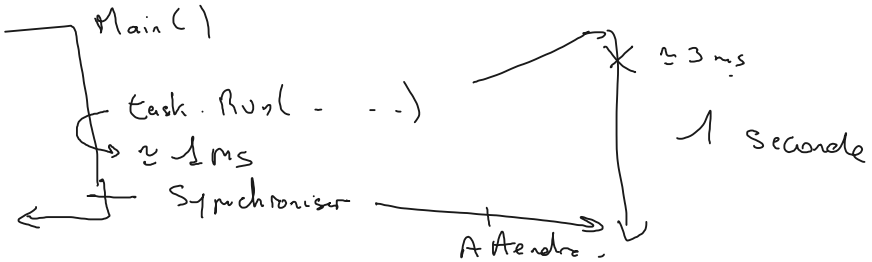
> 4 (inclus .net 6, 7)
→ Task
→ Parallel . Foreach ( )

Collection de 100 éléments



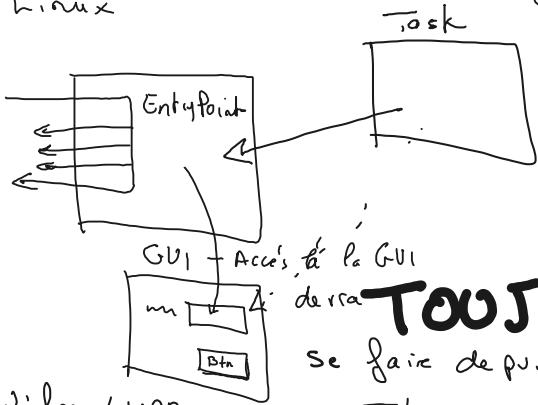
LINQ → Group by ( )
↓
1 ~~Thread~~ Task pour l'arbre qui suit.

Degré max de parallélisme est spécifiable.



- Windows
- Linux

) OS = Tous les Langages.

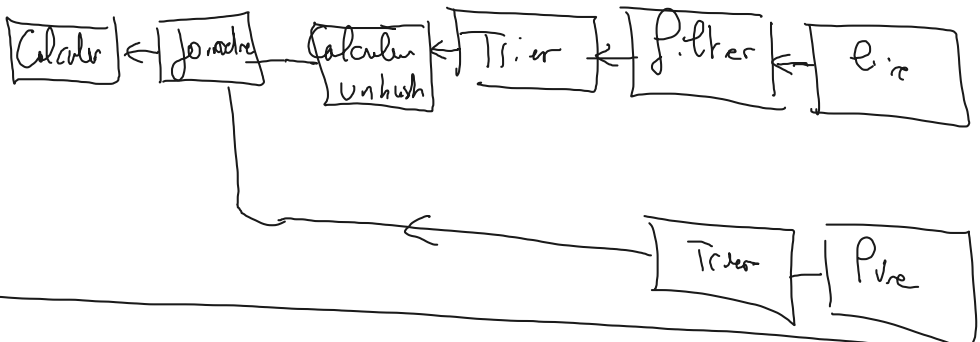


Winform / WPF

(OK : Console (à éviter))

de ça **TOUJOURS** se faire depuis le Thread principal

Plan d'exécution



class CalculMT

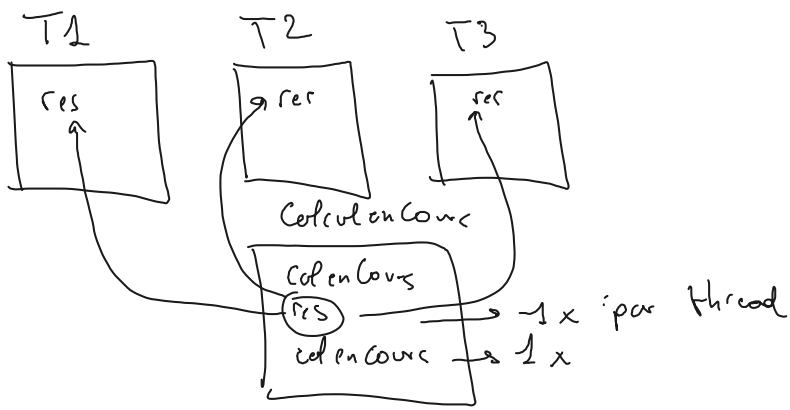
{

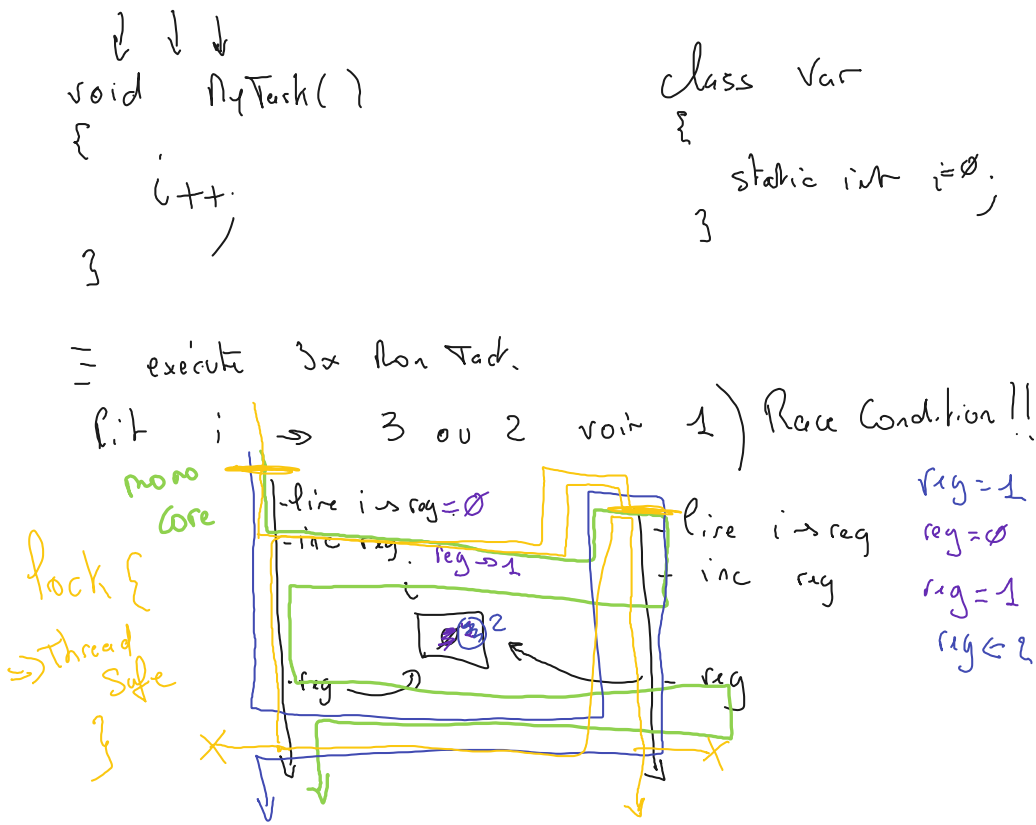
~~static~~ int nbcalculsencours

int resultat

→ lancer 10 calculs → 10 résultats

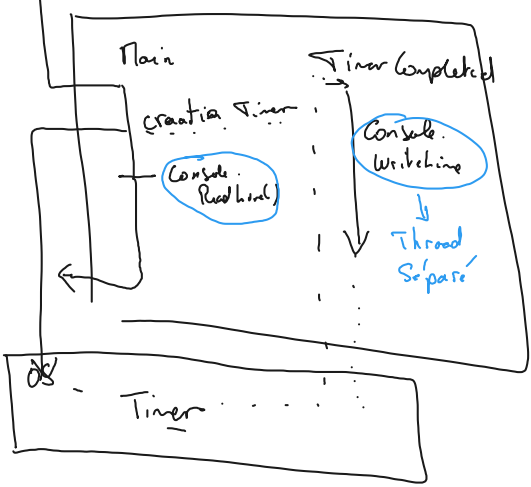
→ 1 instances de CalculMT





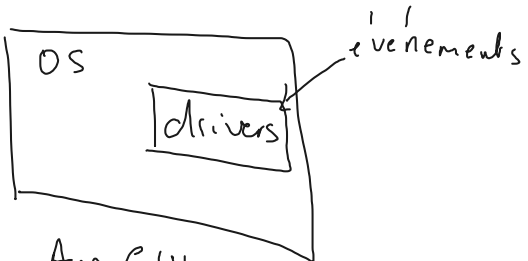

```
public static async Task ShowTodaysInfoAsync()
{
   string message =
      $"Today is {DateTime.Today:D}\n" +
      "Today's hours of leisure: " +
      $"{await GetLeisureHoursAsync()}";

   Console.WriteLine(message);
}

static async Task<int> GetLeisureHoursAsync()
{
   DayOfWeek today = await Task.FromResult(DateTime.Now.DayOfWeek);

   int leisureHours =
      today is DayOfWeek.Saturday || today is DayOfWeek.Sunday
      ? 16 : 5;

   return leisureHours;
}
// Example output:
//    Today is Wednesday, May 24, 2017
//    Today's hours of leisure: 5
```

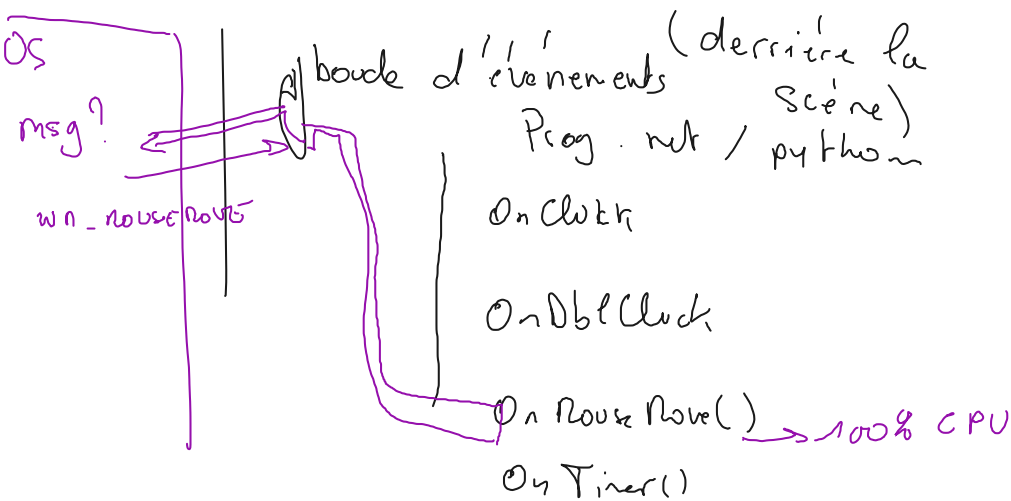
Thread Processus Callbacks (ex du Timer)

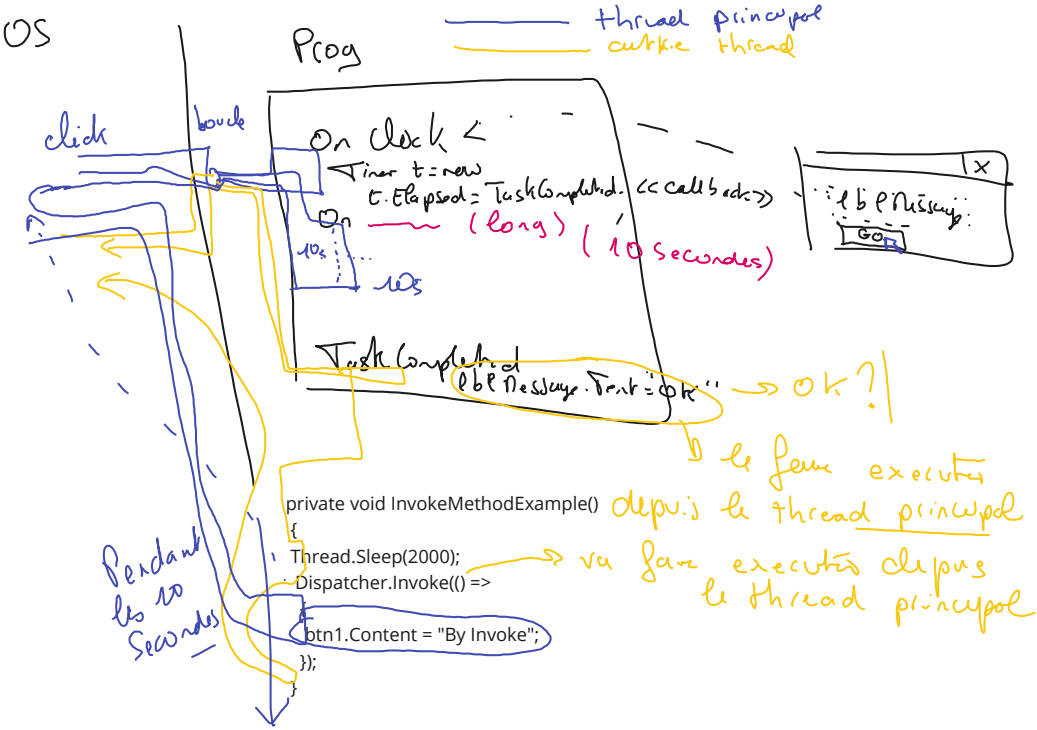


GUI (Windows / Linux)

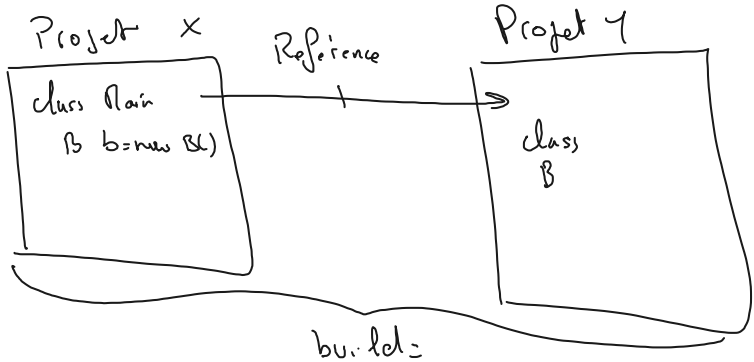
Systèmes d'événements

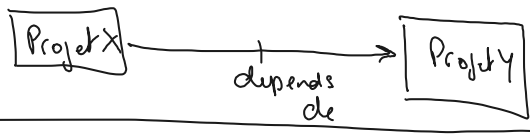




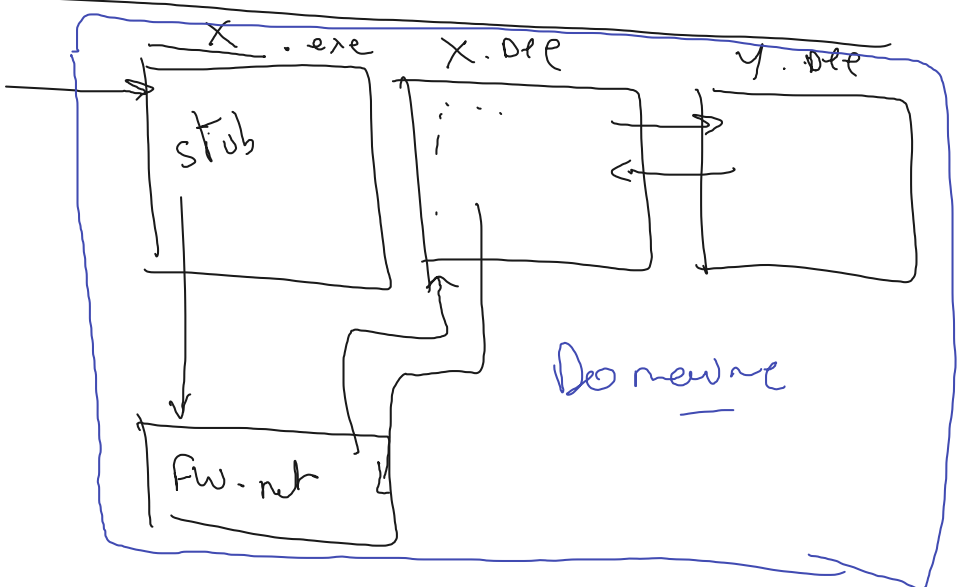


Lire l'appel de méthodes asynchrones en ASP.Net MVC : https://learn.microsoft.com/en-us/aspnet/mvc/overview/performance/using-asynchronous-methods-in-aspnet-mvc-4

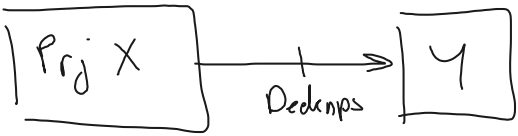




chargement dynamique de Y

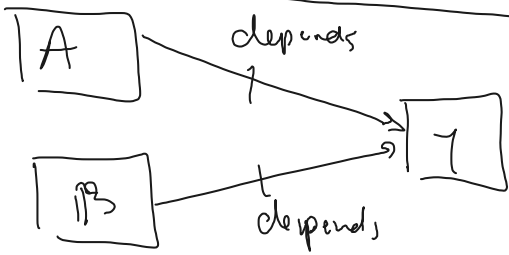


Compte utilisateur Jean



1 dossier ou

./X.exe
./Y/Y.dll
) alternatives (.net Fw 4)

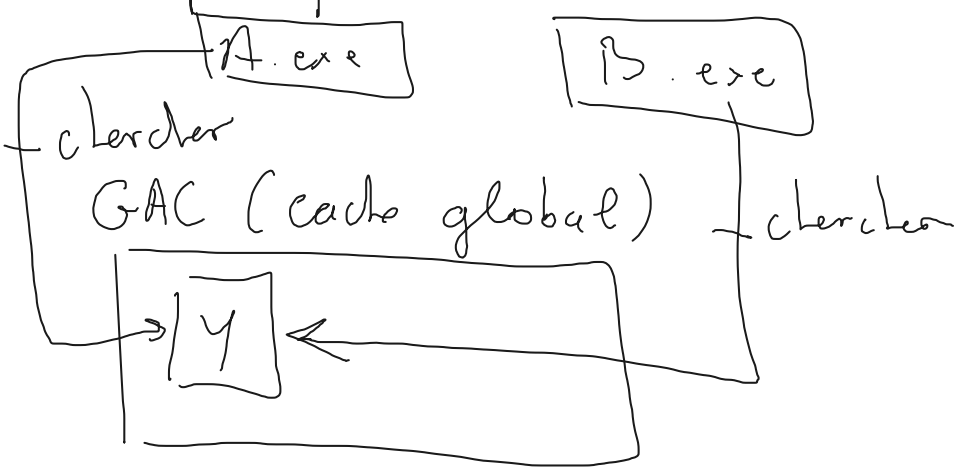


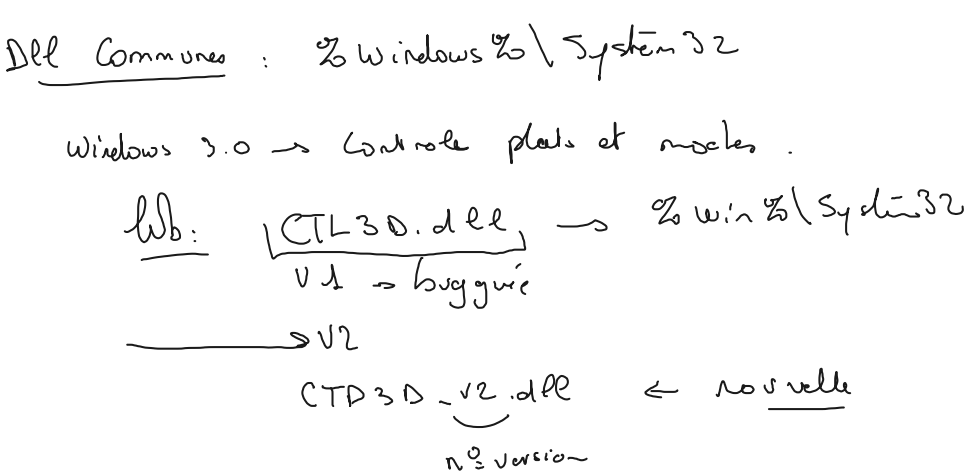
A\ A.exe
Y.dll

B\ B.exe
Y.dll

- Répétition =
  - 1) perte d'espace disque
  - 2) Pbm de maj

Bonne pratique :





Versionning sémantique:

<MAJEUR>.<mineur>.<release>.<build number>

(BONNES PRATIQUES :)
build number : toujours incrémenté, jamais réinitialisé
release : juste des patches, optimisations, aucune modif fonctionnelle
ex v1.1.0.4785 -> v1.1.1.4795 n'apporte rien de plus que moins de bugs et fonctionne "mieux"
mineur : ajout de fonctionnalités, et ne casse pas (no breaking changes) les versions précédentes :
Si AppA qui fonctionne avec B v1.0, alors AppA fonctionnera avec B v1.35 (la sous version 35 ajoute plein de fonctionnalités par rapport à la 0, et ne casse pas la compatibilité)
MAJEUR : si des évolutions sont si importants que cela, et cassent la rétrocompatibilité, alors, on "crée un nouveau produit, une nouvelle version"
AppA qui attends v1 de Y
On livre Y en v2, AppA ne trouvera pas Y.

API Rest : on créée une route supplémentaire pour chaque majeur
ex : /api/product : product en v1 (on ne connait pas le mineur)
/api/v2/product : product en v2

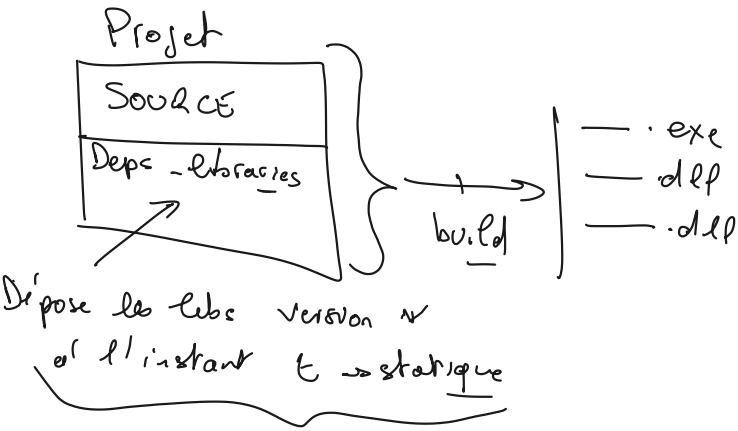
Parcours d'apprentissage C# MS Learn : https://learn.microsoft.com/fr-fr/training/browse/?roles=developer&products=dotnet&expanded=dotnet

- Présentation des Packages NuGet
- .Net en ligne de commande sous Linux
- Conteneurisation d'app

Package = Dépendances.

Principe

Avant



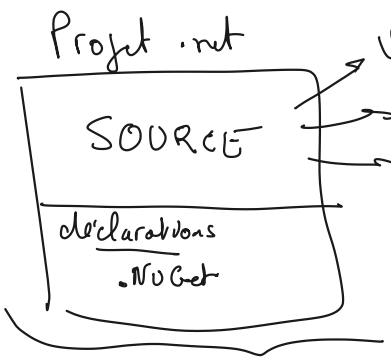
Source du Projet = Lourd

Source + binaires

Maintenant → utilisation des packages

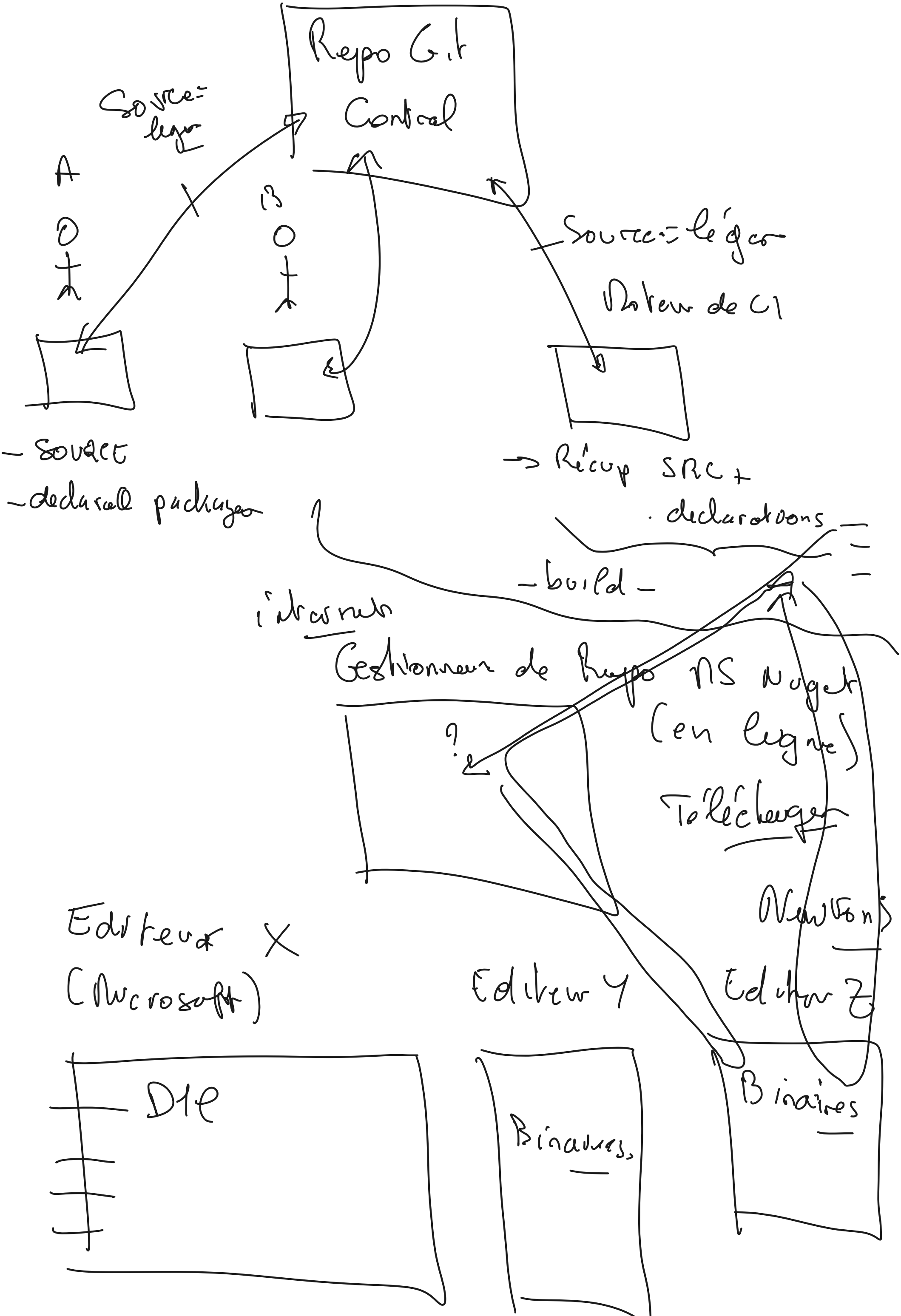
OS linux Redhat → yum
Ubuntu → aptitude
Alpine → Apk.

Dev - Java → ant (libs - obsolète)
→ maven (package)
- Python → flask. / pip ....
- Php → Composer
- .net → NuGet (Général)
↳ valable aussi en powershell



Taille du Projet
→ fichiers texte

Exe Newtonsoft.Json v3.x → dernière version


Repo Git
Central
A
B
Moteur de CI
- source
→ Récup SRC +
declarations
- build -
Gestionnaire de Repo MS Nuget
(en ligne)
Editeur X
(Microsoft)
Editeur Y
Editeur Z
DLL
Binaires
Binaires

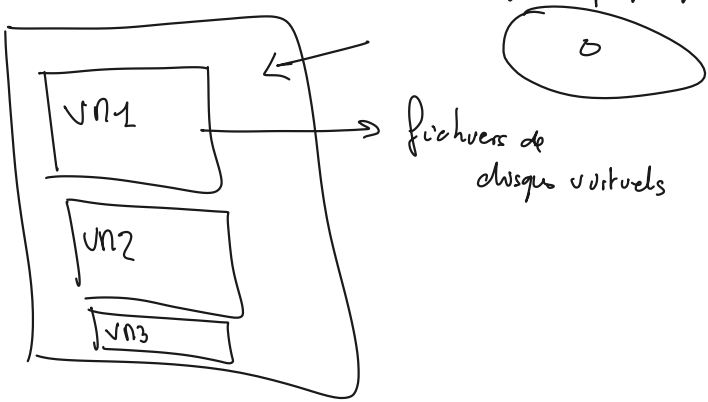
.Net en ligne de commande sous linux :
http://nbstagevm.francecentral.cloudapp.azure.com/guacamole
llogin : student / Pa$$w0rd
choisissez votre stxxx (mot de passe : password si éco d'écran)

Travailler en ligne de commande :
mkdir helloworld
cd helloworld

1. Lister les modèles disponibles :
    a. dotnet new list
2. Créer un projet de type Console dans le dossier actuel :
    a. **dotnet new console**
3. Compiler :
    a. dotnet build
4. Lancer :
    a. **dotnet run**

Nolwenn : st01
Victor : st02
Bryan : st03
Vincent : st04
Johann : st05
Manu : st06
Thibaut : st07
Xavier : st08
Nath : st09
Sasan : st10
Donovan : st11
Pierre : st12

Machine physique → disques physiques



Entreprise → plein d'App/Svc

→ machines Serveur

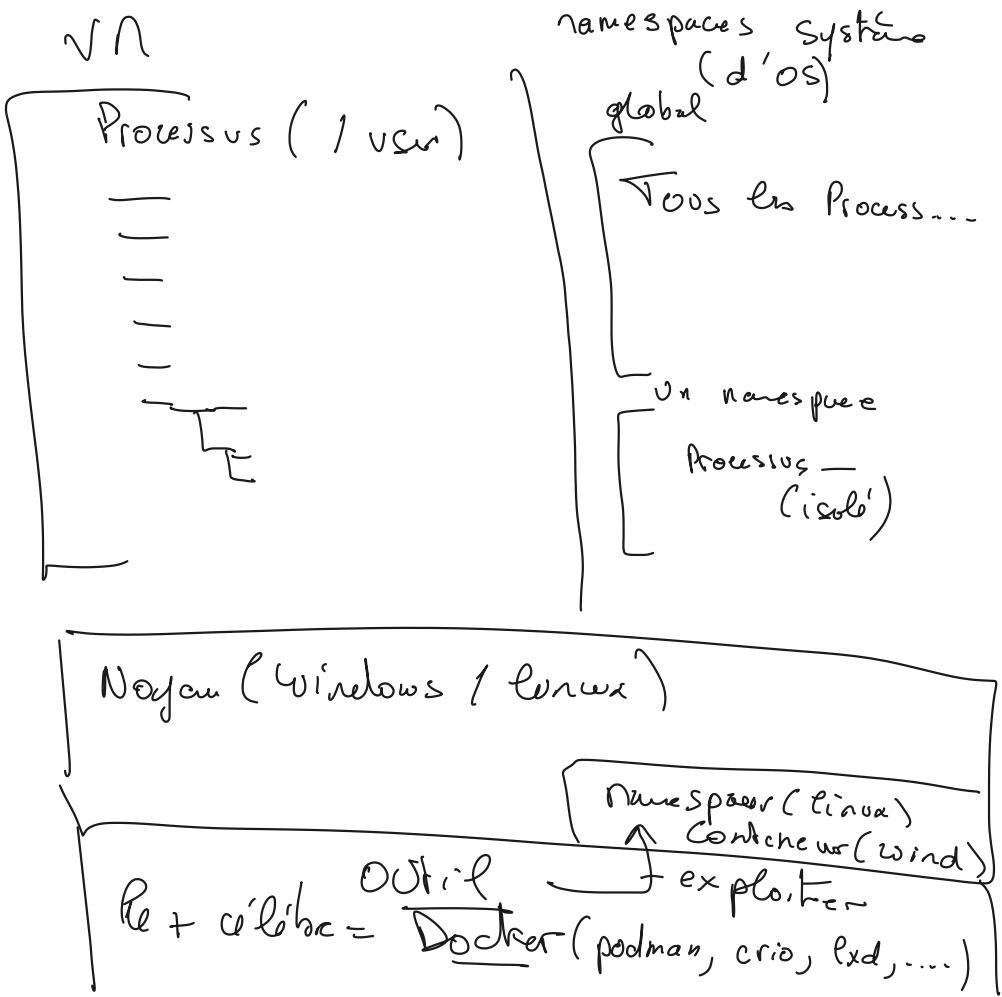
Objectif : Deployer un Serveur ASP.net NUC (Console)

→ Environnement (~~Windows~~ / Linux)
obsolète en Prod

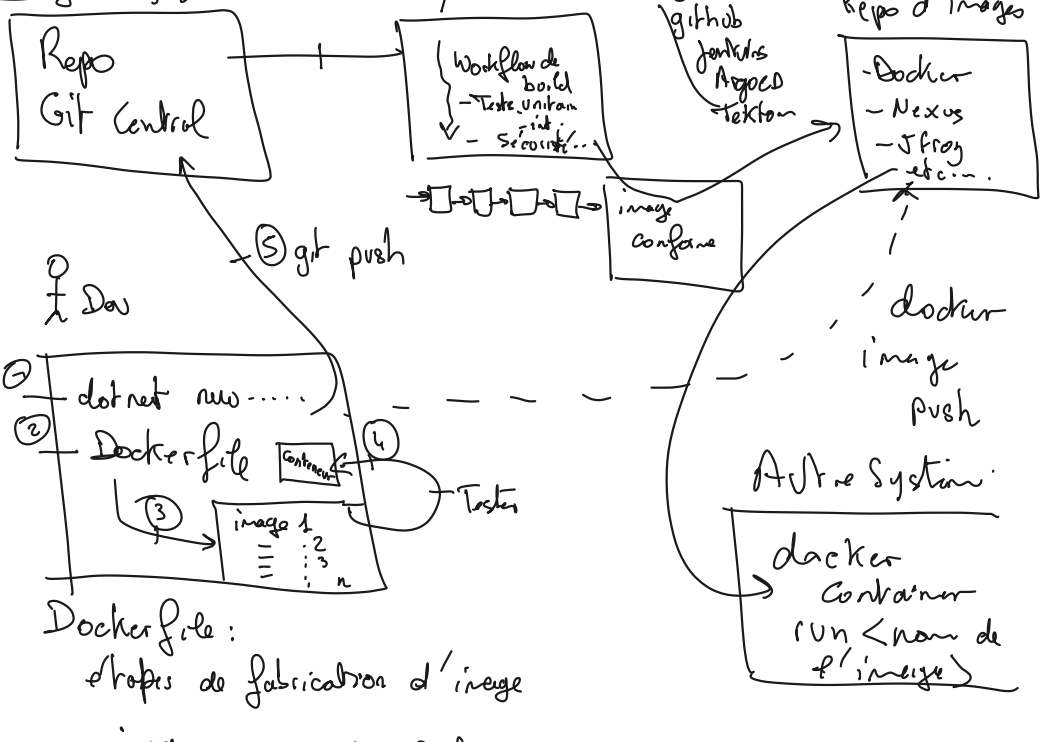
→ 1 VM Linux

→ ex : 8 Coeurs
16 Go RAM
120 Go espace disque

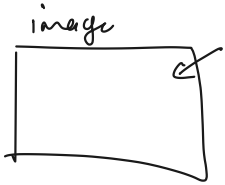
→ Démarrer une VM prends un certain temps
≈ 1 minute


VM
Processus (1 vCPU)
namespaces système (d'OS)
global
Tous les Process...
Un namespace
Processus — (isolé)
Noyau (Windows / Linux)
namespaces (Linux)
Conteneur (Wind)
Outil
+ exploiter
Le + célèbre = Docker (podman, crio, lxd, ...)

ex: github, gitlab .....



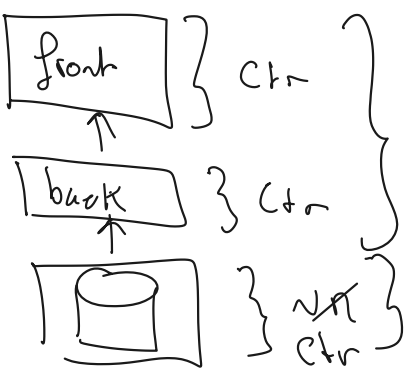
Dockerfile :
étapes de fabrication d'image



Kit de fichiers :
- OS (ubuntu, alpine, fedora, ...)
- votre App Compilée (java, net, ...)
- sources (php, python, ...)

autonome !!!

A Tier



Conteneur