
C#
Microsoft
.NET

Si tu étais…

Livre

Personnage de fiction

Film

Célébrité

Super pouvoir

Jeu vidéo

Chanson

Animal légendaire

Photo

Dessert

Sport

…

Etienne Divina

- Développement C# / .NET depuis 2003
- Testeur automaticien
- Software Craftmanship

- AXA France, WebCenter à Wasquehal

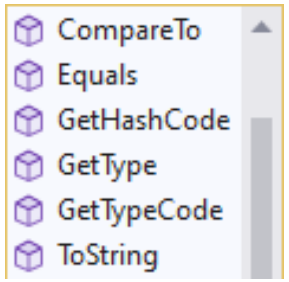
Quelques règles…

1. Respect avec l'intervenant
2. Respect avec mes camarades
3. Pas de téléphone
4. Je participe aux exercices
5. Si je connais déjà
    1. J'aide mes camarades
    2. J'approfondi
6. Pas d'IA générative pendant le cours : on a le droit d'utiliser la calculatrice quand on connait ses tables !

Langage C#

- Plateforme .NET
- Langage de programmation **orienté objet** et **fortement typé**
- En C# tout est objet !

```
int reponseSensDeLaVie = 42;
reponseSensDeLaVie.ToString();
```



- Le code C# est compilé en **MSIL**, traduit en langage machine au moment de l'exécution par la **CLR** (Common Langage Runtime)
- Les applications C# sont organisés en **assembly** (assemblage) dans lequel les classes sont organisés par **namespace** (espace de nom)

Avantages / inconvénients

Avantages

- Performant
- Refactorisation aisée
- IDE riche
- Très bien documenté, Open Source

Inconvénients

- Strict
- Concept objet difficile à maîtriser
- Plutôt réserver au Backend (quoique…)


.NET – A unified development platform
ECOSYSTEM
TOOLS
CLOUD
Azure
WEB
ASP.NET
Blazor
DESKTOP
.NET MAUI
WPF
WinForms
MOBILE
.NET MAUI
Xamarin
GAMING
Unity
IoT
ARM32
ARM64
AI
ML.NET
.NET for Apache Spark
NuGet
Visual Studio
GitHub
Visual Studio for Mac
Components, tools, library vendors
Visual Studio Code
CLI
.NET 6
COMMON BASE LIBARIES/APIs
INFRASTRUCTURE
RUNTIME COMPONENTS
COMPILERS
LANGUAGES

Ecosystème .NET

- Open source : .NET Platform · GitHub
- Possibilité de coder dans différents langages : C#, VB, Cobol, Python, …
- Framework répondant aux principaux besoins des développeurs :
  - Accès à la base données
  - Manipulation de fichiers
  - Gestion du multithreading
  - Etc.

Comment coder en C# ?

- Notepad
- Dotnet.exe

Konan

- VS Code
- + extensions

Hackerman

- Visual Studio
- + café

Minority Report

Notepad + dotnet.exe

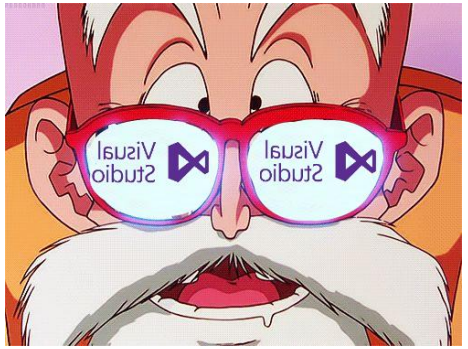
Setup

- Répertoire de projet C:\dev\csharp
- Dotnet new XXX
  - -> arborescence projet créée

Build & Run

- Dotnet build
  - -> dll / exe
- Dotnet run

Visual Studio



- Tout se dont le développeur a besoin sous la main
  - Navigation intelligente dans le code
    - Naviguer dans la définition
    - Naviguer dans les références d'appel ou d'utilisation
  - Renommage fiable
  - Débogage
    - point d'arrêts, avance ou retour
    - Pilotage de serveur en local (web)
  - Tests unitaires
  - Contrôle de code source (Git, etc.)
  - Base de données

Premiers pas

- Variables
  - Doit être déclaré avant utilisation
  - N'est visible que dans son « scope » : méthode, bloc « if »
  - Ne peut changer de type
- Même les types primitifs sont des objets
  - Méthodes associées
- Certaines conversions sont implicites
  - Int -> string implicite, string -> int explicite
  - Int -> double implicite

Exercice 1

- Faire une console qui demande la date de naissance, et afficher l'âge en année
  - Notions abordées :
    - Méthodes et paramètres
    - Manipulation des dates
    - Bonus : indiquer si la personne est un Boomer, génération X, Millennials, génération Z ou Alpha

Exercice 2

- Coding Game : Températures

[Coding Games and Programming Challenges to Code Better (codingame.com)]

Exercice 3 : Roulette SN2

- Faire une console qui ajoute un nom dans un fichier
  - Le(s) nom(s) sera(ont) passé(s) en paramètre(s)
  - Faire un fichier avec un nom par ligne

- La console doit pouvoir retourner un nom de manière aléatoire
  - Gérer un paramètre pour discriminer les actions
  - La console lit le fichier et prend un nom au hasard

  - Notions abordées
    - Manipulation de fichier
    - Manipulation de tableau
    - Bonus : le dernier nom sorti est enregistré et ne sera pas retourné la prochaine fois

Programmation orienté objet

- Définition de classes permettant d'instancier des Objets avec :
  - Propriétés ~ état
  - Méthodes ~ actions : interagissent avec l'état de l'instance
    - Constructeur(0..n) : appelé une fois à la création d'une instance.
      Permet d'initialiser les propriétés
    - Destructeur (0..1) : appelé lorsque le « garbage collector » libère la mémoire de l'objet
      Utile pour la libération explicite de ressource externe coûteuse

- Une classe peut être vu comme un moule, un gabarit
- Les instances comme les objets résultants
  - Chaque objet a sa vie propre, il peut être peint, être rayé, détruit…

Les 3 Principes de la POO

Encapsulation / Abstraction

- Fonctionnement interne n'est pas exposé
- Modifation d'état par méthodes de l'objet

Héritage

- Relation hiérarchique pour définir la réutilisation

Polymorphisme

- Traiter de la même façon (interface, héritage) des objets différents

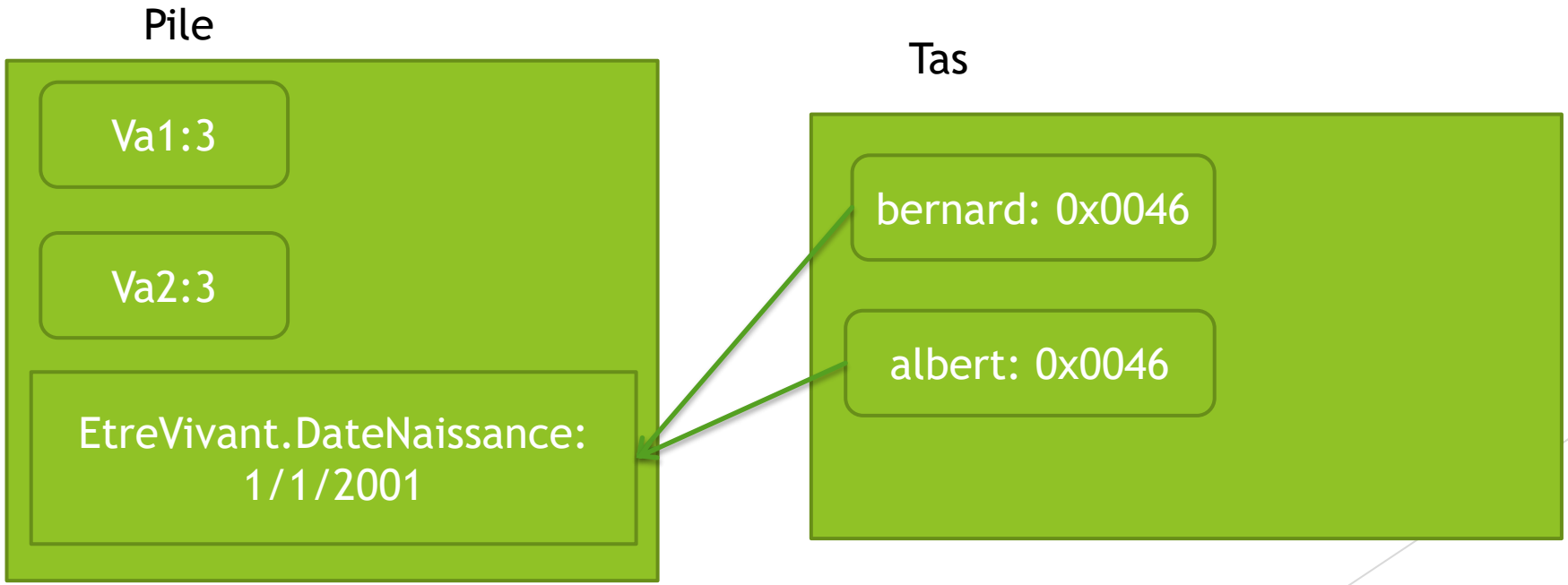
Exercice :

- Créer une classe EtreVivant avec
  - une propriété DateDeNaissance
  - 2 constructeurs
    - Un constructeur avec un paramètre permettant d'initialiser la propriété DateDeNaissance
    - Un constructeur sans paramètre qui initialise la date de naissance à la date courante
  - Une méthode GetAge() qui calcul l'âge de l'être vivant en année pleine

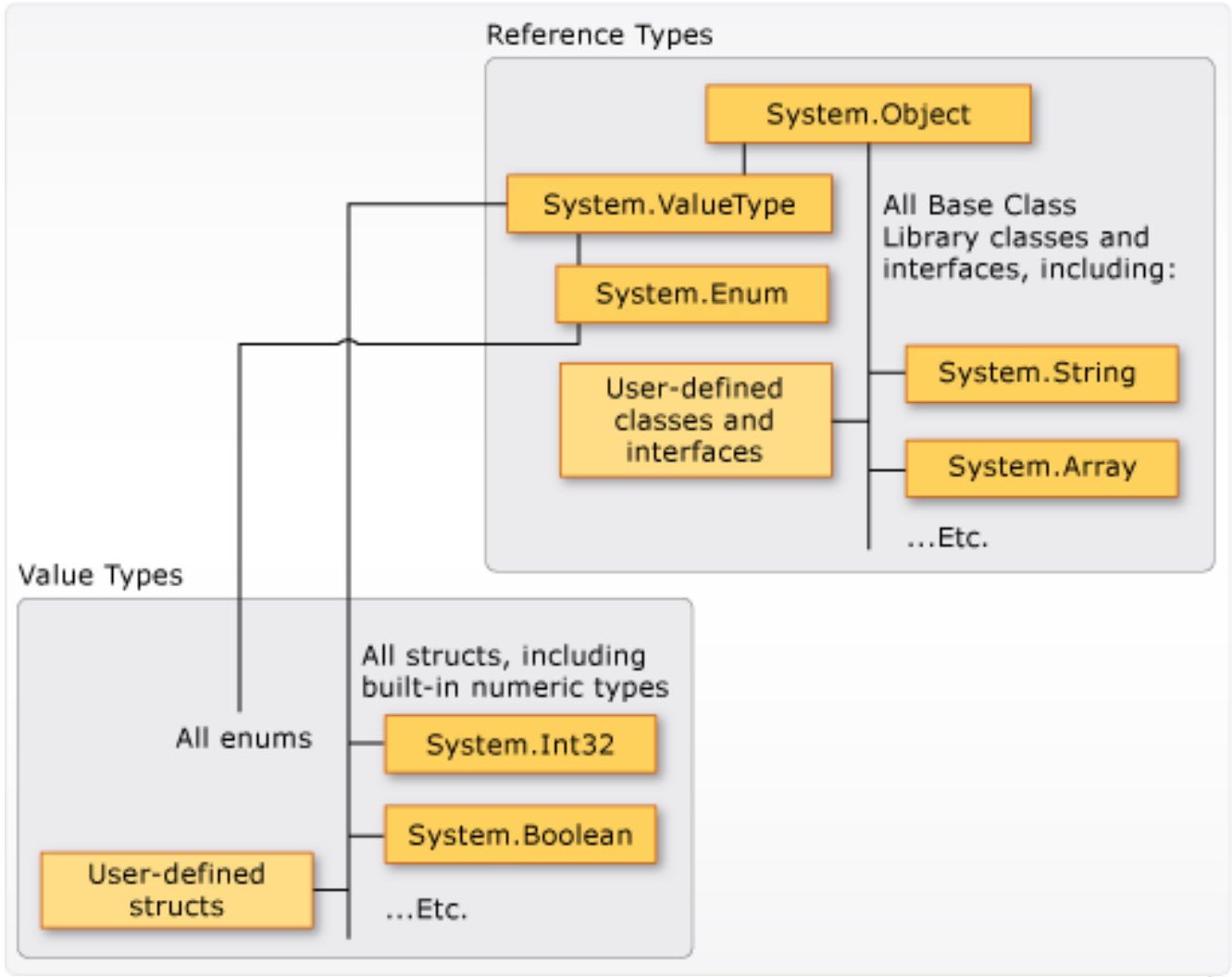
Type valeur / type référence

- Int val1 = 3;
- Int val2 = val1;

- Var bernard = new EtreVivant(1/1/2001);
- Var albert = bernard;



Type valeur / type référence



Exercice

- Déterminez la sortie de la console :

```
1 référence
public static void Oupsy(DateTime nouvelleDate, EtreVivant etreVivant)
{
    nouvelleDate += TimeSpan.FromDays(5);
    etreVivant.DateNaissance = nouvelleDate;
}

0 références
public static void Main(string[] args)
{
    EtreVivant alain = new EtreVivant(new DateTime(2001,1,1));
    DateTime nouvelleDate = new DateTime(2002, 2, 2);
    Oupsy(nouvelleDate, alain);
    Console.WriteLine($"ces deux dates sont elles identique ?  {alain.DateNaissance:D}  // {nouvelleDate:D}");
}
```

Static

- Le mot clé « static » s'applique au niveau de la classe, des propriétés ou des méthodes.
  Static veut dire indépendant d'une instance.

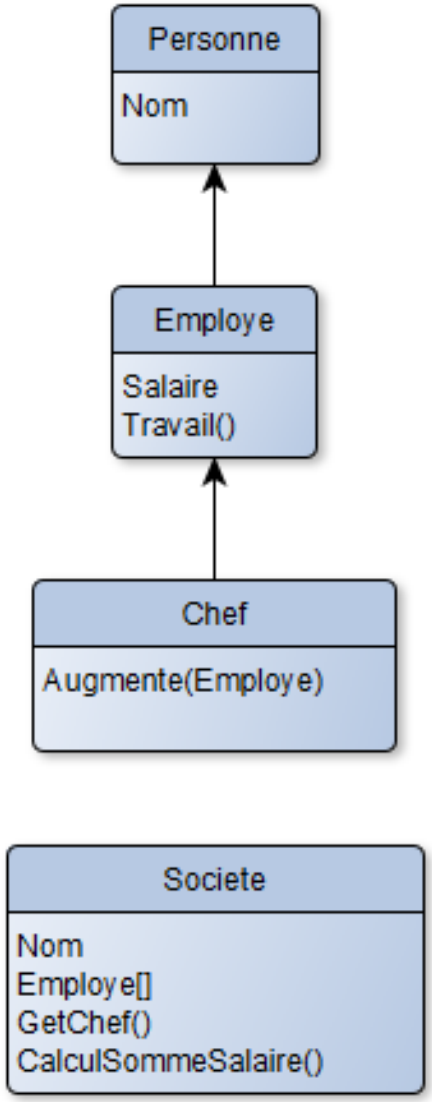
- Une méthode / méthode « static » n'est liée à aucune instance d'objet, elle n'a pas accès aux méthodes / propriétés de la classe, sauf si elles sont « static »
  L'accès se fait par le nom de la classe directement

- Une classe « static » ne peut pas être instanciée, elle ne peut contenir que des propriétés ou méthodes « static »
  ~Helper

Héritage

- La dérivation permet de bénéficier de l'implémentation de la classe parente
- Chaque classe / propriété / méthode a un niveau d'accès
  - Private : visible uniquement dans la classe
  - Protected : visible dans la classe et les dérivés
  - Public : visible pour tout le monde
  - Internal : public pour l'assembly uniquement
- Polymorphisme
  - Une classe dérivée peut être manipulée comme si elle était du type de la classe de base
  - Une classe dérivée peut substituer l'implémentation d'une méthode « virtual » avec le mot clef « override »
  - L'accès aux méthodes et propriétés de la classe parente s'effectue avec le mot clé « base »

Exercice : The Office



- Travail() écrit « [Nom] travaille » dans la console
- Le constructeur par défaut prend en paramètre le salaire

- Augmente() : augmente l'employé en paramètre d'une somme fixe
- Travail() écrit « [Nom] zzz $$$ zzz » dans la console

- GetChef(): renvoie le chef de la liste des employés
- Travail() : fait travailler tous les employés

Exercice : Vroom !

- Une classe véhicule
  - Propriété Vitesse
  - Méthode Avance() renvoie une chaine avec autant de « - » que de vitesse
- Une classe Vélo
  - Vitesse à 5
  - Avance() ajoute « 0-0 » aux tirets
- Une classe Voiture
  - Vitesse à 15
  - Avance() ajoute « [0]=[0]> » aux tirets
- Une classe Garage qui contient **une** liste de **véhicules**
  - Garage a une méthode Sortir() qui fait avancer tous les véhicules