informatique

ICT320

POO

PROGRAMMATION ORIENTEE OBJET

ANASS BENFARES &
XAVIER CARREL

ANASS BENFARES

Déroulement

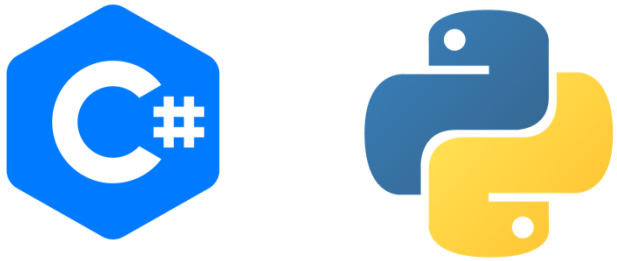
- Accès à la Roadmap du cours : https://roadmap.sh/r/embed?id=66714f98c0f2325c34220bba

- 5 périodes de cours avec moi et des périodes de projet avec M.Melly.
- Théorie puis exercices pratiques.
- Informations concernant les évaluations au prochain cours.
- Pas de questions bêtes !

Semaine 1 - Programmation orientée objet

- Programmation orientée objet c’est quoi ?
- C’est un paradigme de programmation !
- Organisation de notre code autour d’Objets !
- On a donc besoin de :

- Mais surtout de:



Du Procédural à la POO

- Vous allez devoir passer d'un code qui inclut tout à un code ou nous utiliserons **des classes** !

- Une classe qu'est-ce que c'est ?

Classes et Objets

- Vous allez devoir passer d'un code qui inclut tout (Procédural) à un code ou nous utiliserons **des classes** (POO)!

- Une classe qu'est-ce que c'est ?

Nos Objets

Classes et Objets

- Les gâteaux, c’est bien mais on utilise ça comment ?

- Prenons l’exemple d’une manufacture:
  - Exemple usine Tesla !
  - On veut créer plusieurs types de voitures, mais on dispose de spécificités communes !
  - On va décrire ensemble les spécificités d’une voiture

Attributs

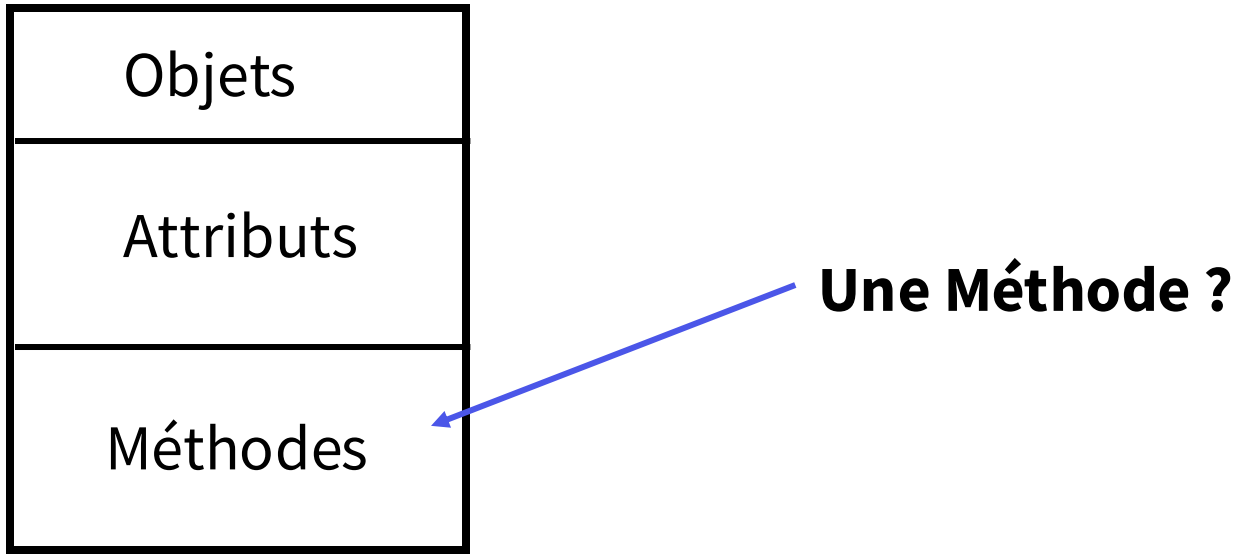
Une voiture dispose d’attributs -> Pourquoi et comment sait-on qu’une voiture est une voiture ?

Une voiture dispose également de fonctionnalités !

On a comme attributs : Couleur, Nombre de portes,.. ?

Modélisation

Pour modéliser de la Programmation Orientée Objets on va utiliser les schémas UML.



Méthodes

Une méthode est une **Feature** de votre objet.

Par exemple pour la voiture elle peut :

- Démarrer
- Arrêter la voiture
- Allumer les phares

Possiblement aussi :

- Accélérer
- Freiner

Ces méthodes peuvent s'appeler entre elles.

Par exemple quand on démarre une voiture, l'allumage des phares est automatique !

Passons au code !

Pour créer un Objet, nous avons besoin d’un Constructeur dans notre classe !

C’est le point de départ de chaque objet.

On peut envoyer des arguments pour définir les attributs liés aux spécificités.

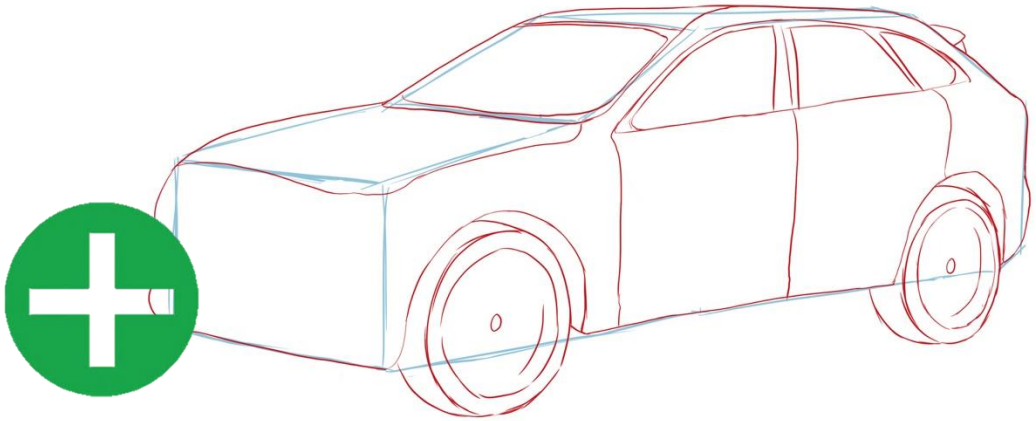
Par exemple la couleur de ma voiture !

Ecrire une Class ?

```
class Car
{
    // Attributs
    private string _color;
    private int _doorNumber;


    /// <summary>
    /// Start a car
    /// </summary>
    public void start()
    {
        // TODO : create code
    }

    /// <summary>
    /// Stop a car
    /// </summary>
    public void stop()
    {
        // TODO : create code
    }

    /// <summary>
    /// Turn on or turn off headlights for a car
    /// </summary>
    /// <param name="toogle">Turn on or turn off headlights</param>
    public void toggleHeadlights(bool toogle)
    {
        // TODO : create code
    }
}
```

Instancier un objet / Constructeur

```
Car porsche = new Car();
```



Instancier un objet / Constructeur

L'intérieur de la parenthèse, permet d'insérer des arguments

```
Car porsche = new Car();
```

porsche, est le nom de la variable de type objet

Le **new**, va appeler le **constructeur**

Le **constructeur** est un **méthode spéciale** qui va permettre **d'instancier** un objet.

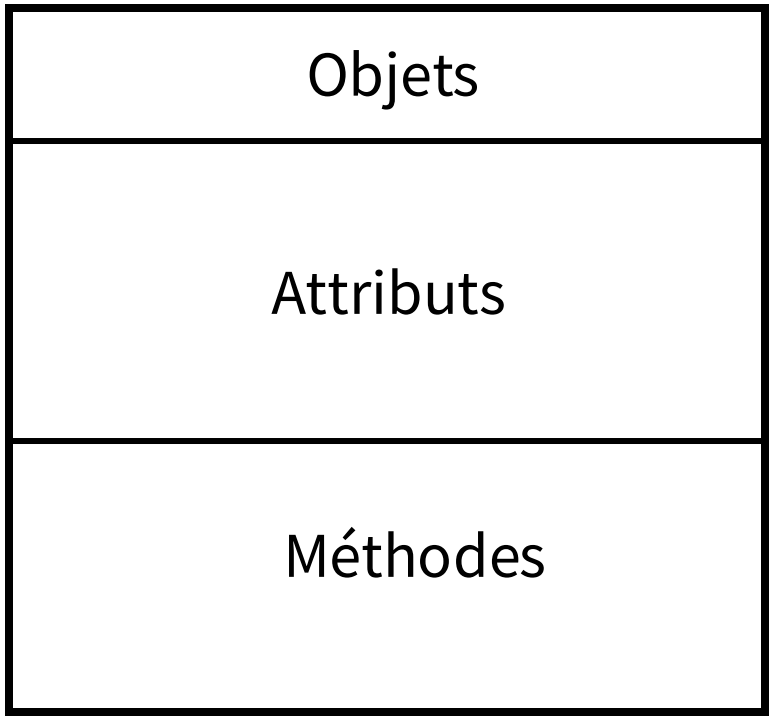
Donc il initie les valeurs initiales.

Un class définit une structure, mais c'est un objet qui est créer !

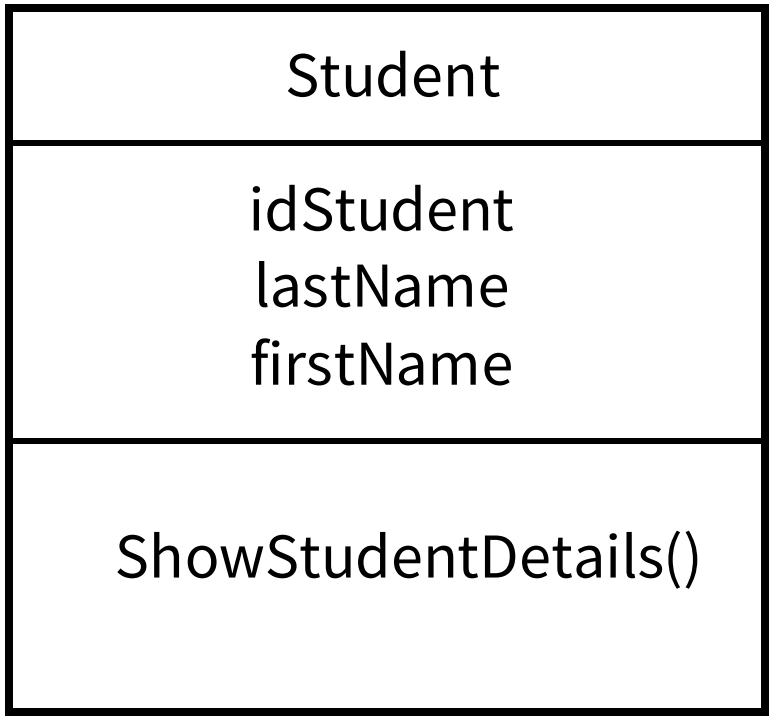
Class Eleve / Schéma UML

On aimerait créer des objets **Eleve**, qui vont permettre d'entrer un numéro d'étudiant, nom, prénom et d'afficher ces informations.

On aimerait que les informations sur l'élève ne puissent pas être accessible/modifier en dehors de la class !



Class Eleve / Schéma UML



Création d'une class Eleve

```
class Student
{
    // Attributs / Proprietes
    2 références
    private int _idStudent;
    2 références
    private string _firstName;
    2 références
    private string _lastName;
    //Static attribut
    1 référence
    public static int _studentTotal = 0;

    /// <summary>
   /// Constructeur pour initialiser un nouvel élève
   /// <param name=idEleve'>Student ID</param>
   /// </summary>
    2 références
    public Student(int id,string prenom, string nom)
    {
        this._idStudent = id;
        this._lastName = nom;
        this._firstName = prenom;
        _studentTotal++;
    }

    // Surcharge de la méthode ToString()
    2 références
    public override string ToString()
    {
    return $"ID:{_idStudent} Nom: {_lastName}, Âge: {_firstName}";
    }

}
```

```
static void Main()
{
    // Création de deux élèves
    Student eleve1 = new Student(1,"John","Doe");
    Student eleve2 = new Student(2,"Jane", "Darc");

    Console.WriteLine(eleve1.ToString());
    Console.WriteLine(eleve2.ToString());
```

Création des objets dans le Main

Collections de données

Dans de nombreux langages de programmation, il est possible de créer des collections de données.

C’est-à-dire une variable qui contiendrait plusieurs variables.

```
List<object> liste1 = new List<object>();

// Ajout de différents types d'éléments
liste1.Add(42);              // int
liste1.Add("Hello World");   // string
liste1.Add(3.14);            // double
liste1.Add(true);            // bool

// Affichage des éléments et de leur type
foreach (var variable in liste1)
{
    Console.WriteLine($"Valeur : {variable}, Type : {variable.GetType()}");
}
```

Collections de données

On pourrait par exemple créer une collection de données comme attribut d’une class et ajouter des objets d’une autre class dedans.

Essayons ensemble !

Collections de données

Nous avons également les tableaux en C# !

Contrairement aux listes, leur tailles sont fixes et ils consomment donc moins de mémoire.

Exercices pratiques

- Snail
- Parachutes