Les modificateurs d'accès

420-EBA-LI -PROGRAMMATION ORIENTÉE OBJET I

Public, private, etc

- L'un des principe fondamentaux de la **POO** est l'**encapsulation** qui permet de restreindre la manipulation des données,
- L'objectif est la *protection* de ces données afin qu'elles ne soient pas altérées ou modifiée sans permission,
- Les restrictions d'accès s'appliquent aux *classes*, *méthodes* et *propriétés*,
- On parle souvent de **visibilité**
- En **C#**, il y a quatre différents niveaux d'accès :
  - **`public`**,
  - **`private`**,
  - *`protected`*,
  - *`internal`*

public

- Permet d'avoir accès aux *propriétés* et *méthodes* en lecture/écriture de l'intérieur et de l'extérieur de la **Classe**,
- Afin de pouvoir définir ce niveau d'accès, il faut utiliser ***public*** comme suit :

```
class MaClasse
{
    public int monAttribut;

    public void maMethode() { }
}
```

```
static void Main(string[] args)
{
    MaClasse monInstance = new MaClasse();

    monInstance.monAttribut = 2;
    monInstance.maMethode();
}
```

private

- Permet d'avoir accès aux *propriétés* et *méthodes* en lecture/écriture uniquement de l'intérieur de la **Classe**,
- Afin de pouvoir définir ce niveau d'accès, il faut utiliser ***`private`*** comme suit :

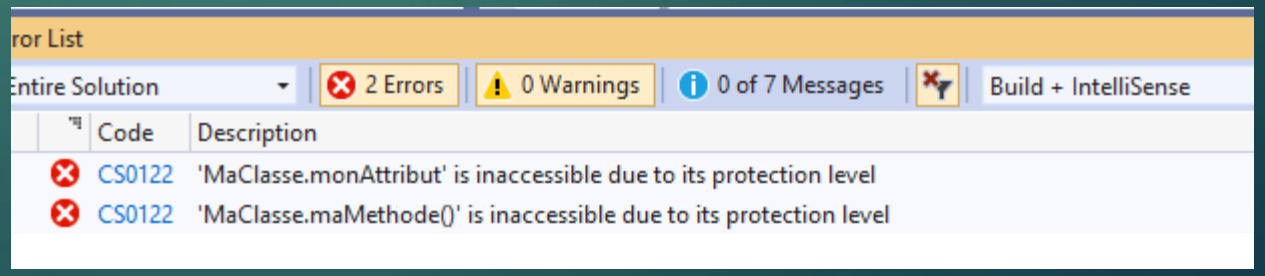
```
class MaClasse
{
    private int monAttribut;

    private void maMethode() { }

}
```

```
static void Main(string[] args)
{
    MaClasse monInstance = new MaClasse();

    monInstance.monAttribut = 2;
    monInstance.maMethode();
}
```



Comment choisir?

- Il faut se poser la question de l'encapsulation. Dois-je protéger de mauvaises manipulation l'accès à une méthode ou une propriété de l'extérieur ou au contraire souhaite-je l'exposer pour utilisation
- De manière générale tout ce qui n'a pas besoin d'être `public` ne devrait pas l'être!
- On va rapidement constater que le paradigme de la POO nous pousse à considérer tous les attributs comme `private`, en introduisant la notion d'accesseur et de mutateur (patience..)

Exemple

```
class Voiture
{
    private const int NB_KM_REF = 100;
    //L'essance dans le reservoir, invisible de l'exterieur!
    private double essence;

    //La consommation moyenne, invisible de l,exterieur!
    private double consommationMoyenne;

    public Voiture( double consommationMoyenne)
    {
        this.consommationMoyenne = consommationMoyenne;
    }

    //Fais partie des interactions de la voiture avec l'exterieur
    public void Remplir(double quantite)
    {
        essence += quantite;
    }

    //Fais partie des interactions de la voiture avec l'exterieur
    public void Rouler(int distance)
    {
        double essenceNecessaire = QuantiteNecessaire(distance);
        if(essenceNecessaire <= essence)
        {
            essence -= essenceNecessaire;
            Console.WriteLine($"On a roulé  {distance} km, il reste {essence} litres");
        }
        else
        {
            Console.WriteLine($"On n'a pas assez d'essence pour rouler {distance} km, il reste {essence} litres");
        }
    }

    //Aucun besoin d,exposer cette methode a l,exterieru
    private double QuantiteNecessaire(int distance) {

        return distance * consommationMoyenne / NB_KM_REF;
    }


}
```