Modificateur static

420-EBA-LI -PROGRAMMATION ORIENTÉE OBJET I

Le modificateur static

- On l'a souvent rencontré devant la méthode main par exemple
- On peut aussi le mettre devant une classe, une propriété ou un méthode, c'est l'objet des trois prochaines diapos

Méthode statique

- Une méthode statique est une méthode qui ne peut être appelée **que sur la classe elle-même** et pas sur une instance d'objet de cette classe
- Une méthode déclarée `static` sera donc appelée selon le modèle: `NomDeLaClasse.Methode()` (on parle parfois de **méthode de classe**)
- Par opposition, une méthode non statique (on parle alors parfois de **méthode d'instance) doit être appelée sur un objet préalablement instancié**

```
class Classe1
{
    public static void MethodeStatique()
    {
        Console.WriteLine("Appel d'une méthode statique (ou methode de classe)");
    }

    public void Methode()
    {
        Console.WriteLine("Appel d'une méthode non statique (ou methode d'instance)");
    }

}
```

```
static void Main(string[] args)
{
    Classe1.MethodeStatique();
    Classe1.Methode();

    Classe1 instanceDeClasse1 = new Classe1();
    instanceDeClasse1.Methode();
    instanceDeClasse1.MethodeStatique();
}
```

Propriété ou variable statique

- Une propriété statique est une propriété qui ne peut être appelée **que sur la classe elle-même** et pas sur une instance d'objet de cette classe.
- Une propriété déclarée `static` sera donc appelée selon le modèle: `NomDeLaClasse. Propriété` (on parle parfois de **variable de classe**)
- Par opposition, une propriété non statique (on parle alors parfois de **variable d'instance) doit être appelée sur un objet préalablement instancié**

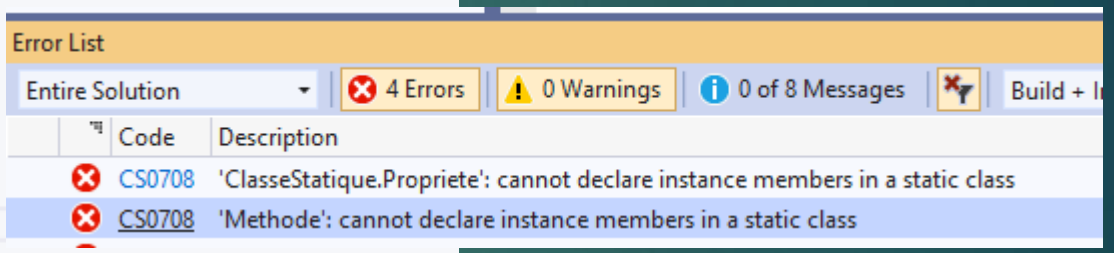
```
class Classe1
{

    public static string ProprieteStatique;

    public string Propriete;
```

```
    static void Main(string[] args)
    {
        Console.WriteLine(Classe1.ProprieteStatique);
        Console.WriteLine(Classe1.Propriete);

        Classe1 instanceDeClasse1 = new Classe1();
        Console.WriteLine(instanceDeClasse1.ProprieteStatique);
        Console.WriteLine(instanceDeClasse1.Propriete);
    }
```

Classe statique

- Une classe statique est une classe qui ne peut pas être instanciée
- Une classe déclarée `static` ne peut donc contenir que des propriétés et des méthodes statiques

```
static class ClasseStatique
{

    public static string ProprieteStatique;

    public string Propriete;

    public static void MethodeStatique()
    {
        Console.WriteLine("Appel d'une méthode statique (ou methode de classe)");
    }

    public void Methode()
    {
        Console.WriteLine("Appel d'une méthode non statique (ou methode d'instance)");
    }

}
```



Quelques constats

- Une méthode de classe (méthode statique) ne peut pas accéder à une variable d'instance (propriété non statique)
- L 'inverse n'est pas vrai! Une méthode d'instance (méthode non statique) **peut accéder** à une variable de classe (propriété statique)
- Lorsqu'une valeur doit être partagée entre toutes les instances d'un objet, on doit la déclarer `static`
- L'utilisation de variables statiques dans des classes instanciables est souvent considérée comme dangereuse, gardons simplement en tête pour le moment que sans nécessité réelle, on les évitera.

Exemple de classe statique

- Math est une classe statique:
  - https://docs.microsoft.com/en-us/dotnet/api/system.math?view=netcore-3.1
  - C'est souvent le cas des classes Utilitaires qui regroupent des fonctionnalités de même nature
- DateTime contient une variable statique:
  - Now qui est clairement partagée par toutes les instances
  - https://docs.microsoft.com/en-us/dotnet/api/system.datetime.now?view=netcore-3.1