Interface

Définition et exemple

Une interface est un ensemble de signatures, de méthodes notamment. Les interfaces s'implémentent dans les classes et permettent de s'assurer que certaines méthodes seront dès lors implémentées.

Reprenons notre exemple avec notre voiture, on va partir du principe que cette dernière peut se déplacer. Pour ce faire, nous allons créer une interface qui aura une signature de méthode pour le déplacement.

```
interface IMovable
{
    void Move();
}
```

Maintenant, nous allons faire en sorte que notre classe "*Car*" puisse implémenter cette interface et ainsi définir la méthode Move.

Nota bene :
Le « I » de « IMovable » est un moyen d'indiquer que c'est une interface. Cette pratique comporte des avantages et des inconvénients, ainsi il est tout aussi correct de nommer cette interface « Movable ».

```
class Car : IMovable
{
    /// <summary>
    /// Color
    /// </summary>
    public string Color { get; set; }

    /// <summary>
    /// Constructor
    /// </summary>
    /// <param name="color"> color of car </param>
    public Car(string color)
    {
        this.Color = color;
    }

    /// <summary>
    /// Move the car
    /// </summary>
    public void Move()
    {
        Console.WriteLine("La voiture se déplace");
    }

}
```

Dès l'instant où l'interface est implémentée, les méthodes contenues dans cette dernière doivent également l'être.

Les caractéristiques des interfaces sont :

- Une interface contient uniquement les signatures de :
  - Méthodes ➔ `public void SampleMethod();`
  - Propriétés ➔ `public int X { get; set; }`
  - Index ➔ `public int this[int index] { get; set; }`
  - Evénements ➔ `event EventHandler ShapeChanged;`
- Une classe peut implémenter **plusieurs** interfaces.
- Implémenter une interface assure que les signatures de l'interface seront définies dans la classe.

Et à quoi ça sert ?

Le but d'une interface est de s'assurer qu'une ou plusieurs méthodes soient implémentées dans les classes qui l'implémenteront. Si nous reprenons notre interface "*IMovable*" pour s'assurer du déplacement. Cette dernière a été implémentée par la classe "*Car*", donc une méthode supplémentaire a été ajoutée concernant le déplacement. (La méthode "*Move*").

Une autre valeur ajoutée des interfaces et qu'elles peuvent se cumuler dans les classes qui les implémentent. Par exemple : `class Car : IMovable, IFlyable, ITransformable ...`

Imaginons maintenant un tout autre contexte, par exemple la classe "*Person*". Elle est caractérisée par ses attributs, tels qu'un prénom, un nom, etc. Maintenant, faisons en sorte d'implémenter notre interface "*IMovable*" sur la classe "*Person*", que va-t-il se passer ?

La méthode "*Move*" doit être implémentée dans la classe "*Person*" et de ce fait, une personne peut maintenant se déplacer !