Classe et objet

Une classe est une structure permettant de définir les ***attributs*** et ***méthodes*** qu'un objet peut contenir. C'est une sorte de moule qui propose des propriétés et des fonctionnalités pour tous les objets qui seront créés à partir de celle-ci.

Exemple

J'ai envie d'une nouvelle voiture. J'utilise le moule « Voiture ». Ce dernier propose de choisir la couleur, ainsi que le nombre de portes. Je choisis d'avoir une voiture verte à 5 portes. De plus, ma voiture pourra démarrer, s'arrêter, allumer et éteindre ses phares!

Ici, les ***attributs*** sont : la couleur et le nombre de portes.
Les ***méthodes*** sont : démarrer, s'arrêter, allumer et éteindre ses phares.

Concrètement, que dois-je coder pour obtenir une classe (une sorte de moule) ?

```
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
    }
}
```

Et maintenant que j'ai implémenté une classe, comment puis-je obtenir ma nouvelle voiture ?

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

C'est quoi ce new ?

Pour créer un nouvel objet ou une nouvelle voiture, un mot-clé est utilisé : ***new*** + le nom de la classe. En programmation, on parle ***d'instancier*** un nouvel objet.

new a également une autre fonctionnalité, il appelle le ***constructeur*** de la classe.

Un constructeur ?

Au départ, une classe est codée. Elle permet d'instancier plein d'objets. Mais naturellement, ce mécanisme ne se fait pas par magie. Comme dans la réalité, si on désire une nouvelle voiture, il faut un jour ou l'autre, mettre les mains dans le cambouis pour que la voiture finissent par ronronner.

C'est pareil pour un objet, il sera fonctionnel qu'une fois « construit ». Le ***new*** va permettre de construire l'objet (donc, de l'instancier). Concrètement, il va appeler une méthode particulière qui effectuer les premiers réglages et allouer la place mémoire nécessaire à l'objet.

Cette méthode est appelé constructeur de la classe. En C#, elle est facilement repérable, puisqu'elle a le même nom que la classe[1].

À noter qu'il est possible d'avoir plusieurs constructeurs pour la même classe. Puisqu'ils auront le même nom et le même type, ce sont les paramètres qui vont les différencier. De plus, il y a toujours un constructeur par défaut (sans paramètre) si aucun n'est codé et que celui-ci disparaît du moment qu'on ajoute un constructeur personnalisé.

[1] Tout dépend du langage utilisé, en C#, c'est ainsi.

Voici le code du constructeur pour cette classe :

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

    /// <summary>
    /// Custom constructor
    /// </summary>
    /// <param name="color">color of car</param>
    /// <param name="doorNumber">number of door</param>
    public Car(string color, int doorNumber)
    {
        this._color = color;
        this._doorNumber = doorNumber;
    }

    /// <summary>
    /// Default constructor
    /// </summary>
    public Car()
    {
        this._color = "red"
        this._doorNumber = 5;
    }

    /// <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
    }
}
```

Dorénavant, pour créer un objet, il faudra lui transmettre ce que le constructeur attend.

```
Car porsche = new Car("green", 5);
```

Ou alors en utilisant le constructeur par défaut :

```
Car default = new Car();
```

Et le code dans cette méthode, il sert à quoi ? C'est quoi ces ***this*** ?

Encore un mot-clé : this

Une ou plusieurs propriétés sont contenues dans une classe (ex : color, doorNumber). Elles doivent pouvoir être utilisées et même réutilisées si nécessaire.

Au moment de l'instanciation de l'objet, le constructeur permet de lier la couleur choisie à l'intérieur de l'objet.

Pour stocker la couleur de la propriété dans l'objet et pas dans une variable quelconque, le mot clé ***this*** est utilisé pour préciser que la variable à utiliser est celle contenue dans la classe. Le mot-clé ***this*** correspond en fait à un objet instancié, présent en mémoire, c'est une référence sur celui-ci à partir d'une définition de classe.
Une classe nous permet d'instancier plusieurs objets, le mot-clé ***this*** correspondra donc à l'instance de l'objet en cours.

D'ailleurs, ce mot-clé est également utilisé pour appeler une méthode interne à la classe et doit être explicitement ajouté lorsqu'il peut y avoir confusion :

```
private string _color;

public Car(string color)
{
        this._color = color;
}
```

Vie et mort d'un objet

En résumé, un objet prend vie quand une instance de sa classe est créée. Ce mécanisme prend effet grâce au mot-clé ***new*** qui appelle une méthode spéciale nommée ***constructeur***. Ceci alloue directement de la mémoire RAM pour stocker les valeurs de l'instance de l'objet (pour une classe qui n'aurait qu'un attribut int, cela correspondrait à 32/64 bits selon l'architecture)

Donc, ma voiture est créée, mais maintenant elle est vieille et doit passer à la casse, je fais comment pour la détruire ?

Quand l'objet n'est plus utilisé, la mémoire qu'il utilise pour son stockage ne peut pas servir à autre chose tant que l'objet est toujours présent (référence utilisée dans le programme). Tout comme, il est impossible de parquer une voiture sur une place où une voiture est déjà parquée.

Dans les langages de programmation modernes tels que C#, la destruction d'un objet se fait de manière automatique via le garbage collector ou ramasse-miettes géré par la machine virtuelle du langage (CLR pour C#). L'avantage de détruire un objet qui n'est plus utilisé est de libérer des ressources mémoires. En effet, tout objet créé utilise de la mémoire pour être stocké (tout comme une voiture utilise un espace pour être parquée) qu'il est bon de remettre à disposition une fois qu'elle n'est plus utilisée.