Constructeurs

420-EBA-LI -PROGRAMMATION ORIENTÉE OBJET I

Pourquoi un constructeur

- **Problématique :** lors de l'*instanciation* d'un nouvel objet, on a besoin d'initialiser certaines de ses *propriétés*. Jusqu'à maintenant ce n'était pas possible, car il faut passer par un ***constructeur***,
- **Définition :** *«C'est une **méthode** particulière, portant le même nom que la classe, et qui est définie **sans aucun retour**. Il a pour mission d'**initialiser** les attributs d'un objet dès sa création. À la différence des autres méthodes qui s'exécutent alors qu'un objet est déjà créé et sur celui-ci, il n'est appelé que lors de la **construction** de l'objet ».*
- On peut avoir un **nombre quelconque** de constructeurs qui doivent différer par leurs paramètres.
- **Constructeur par défaut***:* Un **constructeur sans** aucun **paramètre** est appelé un **constructeur** par défaut. Appel : Un constructeur est appelé par l'instruction que l'on connait déjà

```
new NomDeLaClasse();
```

Exemples

- Voir ExempleCours / Constructeurs (Taxis)

Quelques points à retenir

- On peut définir autant de constructeur que l'on souhaite en fonction des besoins
- Il est grandement recommandé de faire appel au mécanisme d'appel imbriquée de constructeur pour éviter de dupliquer du code et enlever le risque d'incohérence entre les constructeurs
- Tout le code qui doit absolument être exécuté avant l'utilisation d'un objet va en général se trouver dans le constructeur. Autant que possible il doit rendre l'objet utilisable
- Si nous ne créons pas de constructeur explicitement, la classe appellera automatiquement le constructeur par défaut créé par le compilateur lorsqu'un objet est créé
- Meme si un constructeur par default est disponible, on encourage a toujours écrire explicitement au moins un constructeur
- La déclaration d'un constructeur supprime le constructeur par défaut crée par le compilateur

Le mot clé `this`

- Il permet d'accéder dans une classe donnée à l'instance de l'objet lui-même

```
public void Appeler()
{
    Console.WriteLine("Calling: " );
    AfficherInfo();
    //Equivalent a
    this.AfficherInfo();
}
```

- Il permet par exemple de lever une ambiguïté entre une variable locale et une propriété de la classe

```
class Taxi
{
    public bool isInitialized;

    public String name;

    public int seatCount;

    public Taxi(String name, int seatCount)
    {
        isInitialized = true;
        name = name;
        seatCount = seatCount;
    }
```

```
class Taxi
{
    public bool isInitialized;

    public String name;

    public int seatCount;

    public Taxi(String name, int seatCount)
    {
        isInitialized = true;
        this.name = name;
        this.seatCount = seatCount;
    }
```