



Cégep de Saint-Hyacinthe
Département d'informatique

Programmation orientée objet

420-2DP-HY

(3-3-3)

Notions de base sur les objets - Constructeurs et surcharge

(Version 1.0)

1,5 heures

Préparé par
Martin Lalancette

Comprendre les éléments suivants :

- Constructeurs
- Signature
- Surcharge
- Initialiseur

Table des matières

Introduction.....	3
Comprendre les constructeurs.....	3
La signature d'un membre	5
La surcharge d'un membre.....	6
La surcharge d'une méthode ou constructeur	6
Utiliser les initialiseurs.....	7
Bibliographie.....	9

Introduction

Cette séquence a pour but de vous initier aux notions de base nécessaires à l'introduction de la programmation. **Cette séquence traitera des notions reliées aux objets axées principalement sur la surcharge, la signature, les constructeurs/initialiseurs.**

Pour bien suivre les instructions qui vont être mentionnées tout au long des séquences d'apprentissage, une préparation de base s'impose. Il est important de créer un répertoire de travail (sur votre C : ou clé USB). Voici une suggestion d'arborescence :



Exercice 1. : S'assurer d'avoir créé l'arborescence ici haut mentionnée sur votre C ou votre clé USB. Copier ce fichier dans le répertoire rouge ici haut mentionné.

Comprendre les constructeurs

Les constructeurs :

1. Sont utilisés pour **initialiser les membres** (données) de l'objet.
2. Sont des méthodes spéciales qui sont **exécutées lors de la création d'une nouvelle instance de l'objet** (avec l'opérateur *new*).
3. Doivent avoir exactement le **même nom** que celle de la classe.
4. N'ont pas de type de retour (donc pas de *return*).
5. Peuvent être nombreux, mais doivent avoir une signature différente (c.-à-d. avoir de paramètres différents).

À savoir :

- Un constructeur qui n'a **pas d'arguments** (ou paramètres) est appelé le **constructeur par défaut** (default constructor).
- Si une classe est définie sans constructeur, un constructeur par défaut invisible est généré. Celui-ci ne fera rien cependant.
- Il est très utile de définir plusieurs constructeurs afin de fournir plusieurs façons d'initialiser l'objet.

Exemple #1 : Le constructeur par défaut.

```
/// <summary>
/// Le constructeur par défaut (facultatif)
/// </summary>
public Rectangle()
{
    // Ajouter les instructions nécessaires.
    // Pratique pour instancier les objets à l'intérieur de la classe.
}
```

Exemple #2 : Un constructeur copiant un objet Rectangle (*Copy Constructor*).

```
/// <summary>
/// Un constructeur copiant l'objet rectangle.
/// </summary>
/// <param name="r">Objet Rectangle</param>
public Rectangle(Rectangle r)
{
    _iHauteur = r._iHauteur;
    _iLargeur = r._iLargeur;
}
```

En règle générale, la déclaration des constructeurs doit se faire après la déclaration de variables et avant les méthodes. Il faut les regrouper ensemble.

Exercice 2. : Ouvrir la solution de la séquence qui a servi à l'initiation aux classes. Ajouter à la classe **Rectangle**, un constructeur sans paramètre qui initialise les champs Hauteur et Largeur à 5 par défaut. Faire un exemple d'appel.

Exercice 3. : Ajouter un autre constructeur qui permet d'initialiser la hauteur et la largeur de la classe **Rectangle** via les paramètres. Faire un exemple d'appel.

Exercice 4. : Forcer le développeur à utiliser le constructeur de **Rectangle** avec les paramètres. Ne pas permettre le `Rectangle()`. Faire un exemple d'appel.

Exercice 5. : Ajouter un constructeur qui permet d'initialiser la hauteur et la largeur de la classe **Rectangle** en sachant que les paramètres seront de type *string*. Faire un exemple d'appel.

Exercice 6. : Ajouter un constructeur à la classe **Étudiant** qui prend en paramètre le nom, prénom, numéro DA et la date de naissance pour initialiser les champs internes. De plus, le tableau servant à contenir les notes doit être initialisé avec la valeur -1. Faire un exemple d'appel.

La signature d'un membre

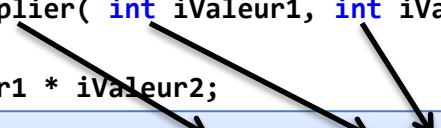
Pour permettre à C#, de déterminer quel membre exécuter, il doit se fier à sa signature. Donc chaque membre doit être conçu avec sa propre signature. Qu'est-ce que la signature d'une méthode? Elle est l'assemblage de :

- L'**identificateur** de la méthode (son nom)
- La **séquence des types** de la liste des paramètres

Ce qui est exclu de la signature :

- Modificateur d'accès (public, private, ...)
- Les identificateurs de paramètres (c.-à-d. leurs noms)
- Le type de retour

```
public int Multiplier( int iValeur1, int iValeur2 )  
{  
    return iValeur1 * iValeur2;  
}
```



La signature est : Multiplier(int, int)

La surcharge d'un membre


Lorsque deux ou plusieurs membres d'un type représentent le même type de membre (méthode, constructeur, etc.), qu'ils possèdent le même nom, mais des listes de paramètres différentes, le membre est dit **surchargé**.

La surcharge d'une méthode ou constructeur

La surcharge d'une méthode est la possibilité de déclarer **plusieurs méthodes/constructeurs avec le même identificateur**. Cependant, chaque signature doit demeurer unique.

Exemple :

```
public int Multiplier( int f1, int f2 ) { /*code...*/ }  
public float Multiplier( float f1, float f2 ) { /*code...*/ }  
public double Multiplier( double f1, double f2 ) { /*code...*/ }
```



Même identificateur, mais signature différente

Exercice 7. : Ajouter à la classe **Étudiant**, une méthode nommée **DéfinirInfos** ayant comme paramètre le nom, prénom et numéro DA. Elle devra affecter les champs correspondants. Concernant la date de naissance, elle devra être initialisée à vide (si string) ou la plus petite valeur du type. De plus, le tableau servant à contenir les notes doit être initialisé avec la valeur -1.

Exercice 8. : Ajouter au projet une nouvelle classe appelée **Montre**. Voici les membres à déclarer :

- 1) **Affichage** : Analogue, numérique ou hybride
- 2) **Énergie** : Mécanique, pile, solaire, kinetic
- 3) **Régulation** : Ressort_spiral, sonique ou quartz
- 4) **Fixation** : Bracelet, Chaîne ou poche
- 5) **DateAchat** : la date que la montre a été achetée. La date ne doit pas être dans le futur.
- 6) **Marque** : nom de compagnie de la montre
- 7) **Modèle** : modèle de la montre.
- 8) **ResistantChocs** : la montre résiste aux chocs (O ou N)
- 9) **ResistantEau** : la montre résiste à l'eau (O ou N)

Ajouter un constructeur par défaut qui initialise la date d'achat avec la date d'aujourd'hui. Déclarer un objet de type Montre nommé « **montre1** ». Afficher la date d'achat de la montre.

Exercice 9. : Ajouter un constructeur qui va prendre en paramètre le **modèle**, la **marque** et le **type d'affichage**. De plus, le constructeur devra initialiser la **date d'achat** avec la date du jour. Dans le même bouton, déclarer un objet de type Montre nommé « **montre2** ». Utiliser les valeurs suivantes : Marque=Pulsar, Modèle=Racing, Affichage=Analogue. Afficher le contenu des propriétés initialisées plus haut.

Utiliser les initialiseurs

Les [initialiseurs](#) d'objet vous permettent d'affecter des valeurs aux champs ou propriétés accessibles d'un objet, au moment de sa création, sans devoir appeler un constructeur suivi de ses lignes d'instructions d'assignation. La syntaxe de l'initialiseur de l'objet vous permet de spécifier les arguments

d'un constructeur ou de les omettre (et la syntaxe de parenthèses).
Exemple avec Rectangle :

```
class Rectangle
{
    // Champs
    private int _iHauteur;
    private int _iLargeur;

    // Propriétés
    public int Hauteur
    {
        get { return _iHauteur; }
        set { _iHauteur = value; }
    }

    public int Largeur
    {
        get { return _iLargeur; }
        set { _iLargeur = value; }
    }

    // Constructeur(s)
    public Rectangle()
    {
        _iHauteur = 10; // Valeur par défaut.
        _iLargeur = 12; // Valeur par défaut.
    }

    public Rectangle(int iHauteur,
                     int iLargeur)
    {
        _iHauteur = iHauteur;
        _iLargeur = iLargeur;
    }
}

static void Main(string[] args)
{
    Rectangle r1 = new Rectangle()
    {
        Hauteur = 10,
        Largeur = 5
    };

    Rectangle r2 = new Rectangle()
    {
        Hauteur = 13,
        Largeur = 24
    };
    ...
}
```

Exercice 10. : Déclarer un objet « **montre4** » de type **Montre** en utilisant les initialiseurs. Utiliser les valeurs suivantes : Marque=Casio, Modèle=YB568O, Énergie=pile , Régulation=sonique et fixation=Chaine. Afficher le contenu des propriétés initialisées plus haut.

Exercice 11. : Déclarer un objet de type **Montre** nommé « **montre5** » en utilisant les initialiseurs. Utiliser les valeurs suivantes : Marque=Bulova, Modèle=96B154, Régulation=quartz et fixation=bracelet. Afficher le contenu des propriétés initialisées plus haut.

Bibliographie

Aucune source spécifiée dans le document actif.