

**Programmation, bases de données et
serveurs-AEC(LEA.D4)
Technique de l'informatique DEC
Intensif(420.B0)**

**Introduction à la programmation
Orientée Objet
420-W20-SF
Pondération: 3-3-3
Hiver 2020**

**Professeur :
André Boumso**

Introduction aux principes de la POO

La redéfinition et la surcharge des méthodes

Objectifs

- À la fin de cette leçon, l'étudiant sera en mesure de:
 - Comprendre la surcharge d'une méthode
 - Utiliser la surcharge de méthode
 - Comprendre la définition d'une méthode
 - Utiliser la définition de méthode

Surcharge d'une méthode

- On surcharge des méthodes **dans une classe** en définissant plusieurs méthodes qui portent le **même nom** mais contiennent des **signatures différentes**.
- La signature d'une méthode consiste en:
 - Son nom
 - Le type et/ou le nombre des arguments

Surcharge d'une méthode

- Le passage de paramètre par valeur ou par référence ne change pas la signature de la méthode.
- Le changement de type de retour ne change pas non plus la signature d'une méthode.

Example

```
public int Somme(int val1, int val2)
{
    return val1 + val2
}
public double Somme(double val1, double
val2)
{
    return val1 + val2
}
```

Exemple

```
public int Somme(int val1, int val2)
{
    return val1 + val2
}
```

//Type de retour différent

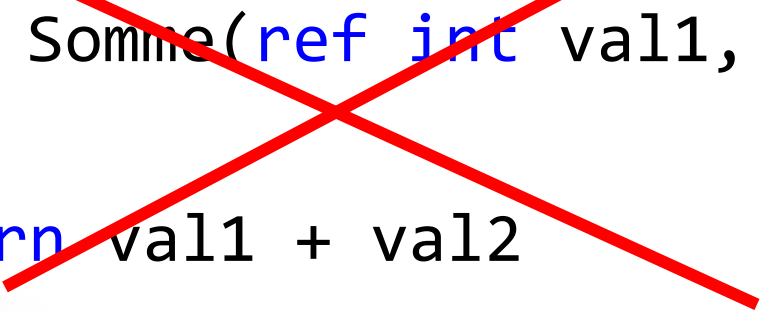
```
public long Somme(int val1, int val2)
{
    return val1 + val2
}
```


Exemple

```
public int Somme(int val1, int val2)
{
    return val1 + val2
}
```

// passage par référence

```
public int Somme(ref int val1, ref int val2)
{
    return val1 + val2
}
```



Exemple

```
public void AfficherDonnee(int val1, int  
val2)  
{  
    Console.Out.WriteLine((val1 +  
val2).ToString)  
}  
//Nombre de paramètres différents  
public void AfficherDonnee(string nom)  
{  
    Console.Out.WriteLine(nom)  
}
```

Redéfinition d'une méthode

- La redéfinition d'une méthode intervient quand on utilise une sous-classe d'une classe de base(parent). (Voir Héritage)
- La fonction redéfinie dans la sous classe doit avoir le même nom et le même nombre d'arguments que la méthode de la classe parent.
- Pour redéfinir une méthode, il faut utiliser le mot clé **override** dans sa déclaration.
 - `public override int Somme(int val1, int val2)`

Redéfinition d'une méthode

- Les méthodes **ToString** et **Equals** de la classe de base **Object** sont parmi les plus redéfinies dans les programmes C#.
 - `public virtual string ToString();`
 - `public virtual bool Equals(object obj);`

Exemple avec ToString

```
int[] valeurs = { 1, 2, 4, 8, 16, 32, 64, 128 };
```

```
Console.WriteLine(valeurs.ToString());
```

```
List<int> liste = new List<int>(valeurs);
```

```
Console.WriteLine(liste.ToString());
```

```
// Le résultat affiché est:
```

```
// System.Int32[]
```

```
//
```

```
System.Collections.Generic.List`1[System.Int32]
```

Exemple avec ToString

```
public override string ToString()
{
    string valRetour = string.Empty;
    foreach (int item in liste)
    {
        if (string.IsNullOrEmpty(valRetour))
            valRetour += item.ToString();
        else
            valRetour += string.Format("{0}", item);
    }
    return valRetour;
}
```

Exemple avec Equals

```
public override bool Equals(Object p_obj)
{
    //Vérifie si l'objet est null et compare les
types à l'exécution .
    if ((p_obj == null) ||
!this.GetType().Equals(p_obj.GetType()))
    {
        return false;
    }
    else
    {
        Point p = p_obj as Point;
        return (x == p.x) && (y == p.y);
    }
}
```

Redéfinition d'une méthode

- Lors de la création des classes, il est recommandé de:
 - Redéfinir les méthodes ToString et Equals


```

class Point
{
    protected int x, y;

    public Point() : this(0, 0)
    { }

    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }

    public override bool Equals(Object p_obj)
    {
        //Vérifie si l'objet est null et compare les types à l'exécution .
        if ((p_obj == null) || !this.GetType().Equals(p_obj.GetType()))
        {
            return false;
        }
        else
        {
            Point p = p_obj as Point;
            return (x == p.x) && (y == p.y);
        }
    }

    public override string ToString()
    {
        return "x: " + this.x + "y: " + this.y;
    }
}

```

Tableau de comparaison

	Surcharge	Redéfinition
1)	La surcharge de méthode est utilisée pour améliorer la lisibilité du programme.	La redéfinition de méthode est utilisée pour fournir l'implémentation spécifique de la méthode qui est déjà fournie par sa super classe.
2)	La surcharge de méthode est effectuée dans la classe elle-même.	La redéfinition de méthode se produit dans deux classes ayant une relation d'héritage.
3)	En cas de surcharge de méthode, les paramètres doivent être différent.	En cas de redéfinition de méthode, les paramètres doivent être identique.
4)	La surcharge de méthode est l'exemple du polymorphisme au moment de la compilation.	La redéfinition de méthode est l'exemple du polymorphisme au moment de l'exécution.
5)	En C#, la surcharge de méthode ne peut pas être effectuée en modifiant uniquement le type de retour de la méthode. Le type de retour peut être identique ou différent dans la surcharge de méthode. Mais il changer le paramètre.	Le type de retour doit être identique lors de la redéfinition de méthode.