

POO : Séance 3

Concepts de base de la POO

(Suite)

Ce qu'on a vu...

- POO
- Programmation Procédurale
- Objet
- Classe
- Attributs
- Méthodes
- Instanciation
- Constructeur
- Surcharge

Ce qu'on n'a pas vu...

- Private
- Public
- Encapsulation
- Etc.

Rappel : Exercice (I)

Class Rectangle

```
{  
    //Attributs  
    Int longueurRectangle ;  
    Int LargeurRectangle ;  
  
    //Méthodes  
    Float surface ()  
    {  
        Return (LargeurRectangle* longueurRectangle)  
    }  
}
```

Class MonRectangle

```
{  
  
    Public static void main (string [] args )  
    {  
        Rectangle Rect = new Rectangle ();  
        System.out.print ('La surface du rectangle est',  
        Rect.Surface ()) ;  
    }  
}
```

Appel du constructeur
par défaut

Appel de la méthode
surface de la classe
Rectangle

Message affiché : Le surface du rectangle est 0

Rappel : Exercice (2)

```
Class Rectangle
{
```

```
    Int longueurRectangle ;
    Int LargeurRectangle ;
    Rectangle (int lon, int lar)
    {
        longueurRectangle = lon ;
        LargeurRectangle = lar ;
    }
    Rectangle()
    {
        LongueurRectangle = 0 ;
        LargeurRectangle = 0 ;
    }
}
```

Comment Instancier la classe Rectangle en utilisant ce constructeur?

Comment Instancier la classe Rectangle en utilisant ce constructeur?

```
Class MaBanque
{
```

```
    public void main (String [] args)
    {
```

```
        Rectangle Rect = New Rectangle ();
```

```
        System.out.println ('La surface du rectangle est', Rect.surface());
```

```
        Rectangle Rect1 = New Rectangle (5,6);
```

```
        System.out.println ('La surface du rectangle est', Rect.surface());
```

```
    }
```

```
}
```

Appel du deuxième constructeur

La surface du rectangle est 0

Appel du premier constructeur

La surface du rectangle est 30

Règles de visibilité (I)

- Les attributs définies dans une classe sont accessibles dans toutes les méthodes de cette même classe.

Class CompteBancaire

```
{  
    //Attributs  
    int numCompte;  
    String titulaireCompte;  
    float soldeCompte;  
  
    //Méthodes  
    void Depot (float Somme)  
    {soldeCompte += Somme;}  
  
    void Retrait (float Somme)  
    {soldeCompte -=Somme;}  
  
    void EditerSolde ()  
    {system.out.println (soldeCompte );}  
}
```

Règles de visibilité (2)

- Les variables définies dans une méthode et ses paramètres ne sont visibles qu'à l'intérieur de la méthode

Class CompteBancaire

```
{  
    //Attributs  
    int numCompte;  
    String titulaireCompte;  
    float soldeCompte;  
  
    //Méthodes  
    void Depot (float Somme)  
    {soldeCompte +=Somme ;}  
  
    void Retrait (float Somme)  
    {soldeCompte -=Somme;}  
  
    void EditerSolde ()  
    {system.out.println (soldeCompte );}  
}
```

Règles de visibilité (3)

- Les variables locales peuvent masquer des attributs de la classe


Class CompteBancaire

```
{  
    //Attributs  
    int numCompte;  
    String titulaireCompte;  
    float soldeCompte;  
  
    //Constructeur  
    CompteBancaire (int numCompte, String titulaireCompte, float soldeCompte)  
    {  
        this.numCompte = numCompte;  
        this.titulaireCompte = titulaireCompte;  
        this.soldeCompte = soldeCompte;  
    }  
}
```


Règles de visibilité (4)

- Les méthodes définies dans une classe sont accessibles dans toutes les méthodes de cette même classe

```
Class Rectangle
{
    Int LongueurRectangle ;
    Int LargeurRectangle ;
    Rectangle(int lon, int lar)
    {
        LongueurRectangle = lon ;
        LargeurRectangle = lar ;
    }
    Rectangle ()
    {
        LongueurRectangle =0;
        LargeurRectangle =0;
    }
    Float surface ()
    {
        Return (LongueurRectangle* LargeurRectangle) ;
    }
    Void afficher_Surface ()
    {
        System.out.println (' La surface du rectangle est', surface ())
    }
}
```



**Et les autres classes, peuvent-elles
accéder à mes attributs? Et mes méthodes?**

Encapsulation

- L'accès direct aux variables d'un objet est possible en JAVA mais ... il n'est pas recommandé car contraire au principe d'encapsulation.
- Les données d'un objet doivent être privées et protégées. Elles sont protégées et ne sont accessibles (et surtout modifiables) qu'à travers de méthodes prévues à cet effet.
- En JAVA, il est possible d'agir sur la visibilité (accessibilité) des membres (attributs et méthodes) d'une classe vis-à-vis des autres classes.
- Plusieurs niveaux de visibilité peuvent être définis en précédant d'un modificateur la déclaration d'un attribut, méthode ou constructeur.

◆ Public

◆ Private

◆ Protected

A revoir dans la partie Héritage

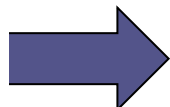
Encapsulation : Modificateurs de visibilité

Membres \ Visibilité	Public	Private
Classe	La classe peut être utilisée par n'importe quelle classe	
Attribut	Attribut accessible partout où sa classe est accessible. N'est pas recommandé du point de vue encapsulation	Attribut accessible uniquement dans le code de la classe qui le définit
Méthodes	Méthode pouvant être invoquée depuis le code de n'importe quelle classe	Méthode utilisable uniquement dans le code de la classe qui le définit

Classe : Publique

Méthodes : Publiques

Attributs : Privés



Encapsulation : Exemple (I)

```
public Class Rectangle
```

```
{
```

```
    private Int LongueurRectangle ;
```

```
    private Int LargeurRectangle ;
```

```
    public Rectangle (int lon, int lar)
```

```
    {
```

```
        LongueurRectangle = lon ;
```

```
        LargeurRectangle = lar ;
```

```
    }
```

```
    public Float surface ()
```

```
    {
```

```
        Return (LongueurRectangle* LargeurRectangle) ;
```

```
    }
```

```
}
```

Les attributs doivent être
privés

Les méthodes doivent être
public

La classe
est aussi
publique

Encapsulation : Exemple (2)

```
public Class MonRectangle  
{
```

```
    public static void main (string [] args)  
    {
```

```
        Rectangle Rect = new Rectangle (4,8);
```

Appel du constructeur :
une méthode publique

```
        System.out.println ('la surface du rectangle est',
```

```
        Rect.surface ());
```

La surface du rectangle est 32

```
        System.out.println (Rect.LargeurRectangle);
```

```
        System.out.println (Rect.LongueurRectangle);
```

Erreur ! Les attributs
sont privés. Aucun accès
n'est possible en dehors
de la classe concernée

```
    }  
}
```

Appel de la méthode surface : une méthode publique

Comment faire alors pour
consulter ou modifier les attributs
d'un objet quelconque ?

Les méthodes Get (accesseur) et Set (mutateur)

- Afin d'accéder et modifier les attributs d'une classe, on définit deux méthodes publiques : **Set** et **Get**.

```
Public class Rectangle
{
    int LongueurRectangle ;
    int LargeurRectangle;

    Public Rectangle (int lon, int lar)
    {
        LongueurRectangle = lon;
        LargeurRectangle = lar;
    }
    Float surface ()
    {
        Return (LongueurRectangle * LargeurRectangle);
    }
    Int GetLongueur ()
    {
        Return (LongueurRectangle);
    }
    Int GetLargeur ()
    {
        Return (LargeurRectangle);
    }
    Void SetLongueur (Long)
    {
        LongueurRectangle = long;
    }
    Int SetLargeur(Larg)
    {
        LargeurRectangle= Larg;
    }
}
```

Méthodes permettant
d'accéder aux attributs
privés de la classe

Méthodes permettant
de modifier les attributs
privés de la classe

Les méthodes Set et Get (2)

```
Public class MonRectangle
```

```
{
```

```
    Public static void main (string [] args)
```

```
    {
```

```
        Rectangle Rect = new Rectangle (5,8);
```

```
        System.out.println (Rect. LongueurRectangle );
```

```
        System.out.println (Rect. LargeurRectangle );
```

```
        System.out.println(Rect. GetLargeur ());
```

```
        System.out.println(Rect. GetLongueur ());
```

```
        System.out.println(Rect. Surface());
```

```
        Rect.SetLargeur(18);
```

```
        System.out.println(Rect.GetLargeur ());
```

```
        System.out.println(Rect.GetLongueur ());
```

```
        System.out.println(Rect. Surface());
```

```
        Rect.SetLongueur(9);
```

```
        System.out.println(Rect.GetLargeur ());
```

```
        System.out.println(Rect.GetLongueur ());
```

```
        System.out.println(Rect. Surface());
```

```
    }
```

```
}
```

Erreur! Les attributs sont
privés

8

5

40

Largeur = 18

18

5

90

Longueur = 9

18

9

162

Concepts vus

Encapsulation

Private

Public

Visibilité

Modificateurs de visibilité

Fin de la troisième séance

Questions ?

Questions (I)

❖ Lorsqu'on choisit le modificateur private pour une méthode, cette dernière sera accessible partout.

Non

❖ L'encapsulation consiste à protéger les attributs d'une classe A à l'aide du modificateur d'accès private.

Oui

❖ L'encapsulation consiste aussi à créer des méthodes permettant de manipuler les attributs privés d'une classe.

Oui

❖ L'accessesseur (get) permet d'accéder aux attributs privés d'une classe.

Oui

Questions (2)

❖ Le modificateur (set) permet de modifier l'attribut d'une classe.

Oui

❖ Les attributs d'une classe doivent être publics

Non

❖ Le constructeur d'une classe doit être privé

Non

❖ Les méthodes d'une classe doivent être privés

Non