



SUPPORT DE COURS

INITIATION A LA PROGRAMMATION ORIENTEE

OBJET

Niveau : Première année Licence Informatique

Option : Tronc Commun

Dispensé par : AMEVOR Kossi A.

Année académique 2023-2024

Chapitre 1: Classes et Objets

Classes et Objets

I - OBJET**1- Définition**

Un objet représente une entité du monde réel qui se caractérisent par une identité, des états significatifs et par un comportement.

1 - OBJET

2- Caractéristiques

□ Identité:

L'objet possède une identité, qui permet de le distinguer des autres objets, indépendamment de son état.

C'est le nom de l'objet.

Exemple : **rectangle1, rectangle2, unRectangle.**

1 - OBJET**2- Caractéristiques****□ Etat:**

L'état d'un objet est sa forme à un instant donné, telle que décrite par les valeurs de l'ensemble de ses variables propres, qui sont appelées des attributs.

Exemple:

Rectangle1 :
Longueur = 6
Largeur = 4

Rectangle2 :
Longueur = 5
Largeur = 5

1 - OBJET

2- Caractéristiques

□ Comportement:

Le comportement d'un objet se traduit par l'ensemble des actions (appelées opérations) que l'objet est à même de réaliser.

Ces opérations permettent de faire réagir l'objet aux sollicitations extérieures ou d'agir sur les autres objets. Ces opérations sont appelées des méthodes.

Exemple: Périmètre de rectangle1 = 20, Aire de rectangle1 = 24
Périmètre de rectangle2 = 20, Aire de rectangle2 = 25

I - OBJET

3- En résumé

Un objet est une entité logicielle :

- qui a une identité ;
- qui est capable de sauvegarder un état, c'est-à-dire un ensemble d'information dans des variables internes ;
- qui répond à des messages précis en déclenchant des activations internes appropriées qui changent l'état de l'objet. Ce sont des fonctions liées à des objets et qui précisent le comportement de ces objets.

Objet = Identité + état + Comportement

II - CLASSE

1- Définition

Une classe est une description abstraite d'un objet.

Elle est une entité regroupant des variables et des fonctions. Chacune de ces fonctions aura accès aux variables de cette entité.

Elle apparaît comme un moule ou une usine à partir de laquelle il est possible de créer des objets ; c'est en quelque sorte une « boîte à outils » qui permet de fabriquer un objet.

II - CLASSE

1- Définition

Entre classe et objet il y a, en quelque sorte, le même rapport qu'entre type et variable.

Une instance, c'est tout simplement le résultat d'une instantiation.

Une instantiation, c'est le fait d'instancier une classe.

Instancier une classe, c'est se servir d'une classe afin qu'elle nous crée un objet.

II - CLASSE

2- Exemple : Les rectangles

On désire gérer la catégorie de quadrilatères constituer de quatre cotés (Rectangles). Ces polygones ont :

- une longueur ;
- une largeur .

Pour chaque polygone on pourra calculer :

- le demi périmètre ;
- le périmètre ;
- l'aire.

II - CLASSE

3- Structure

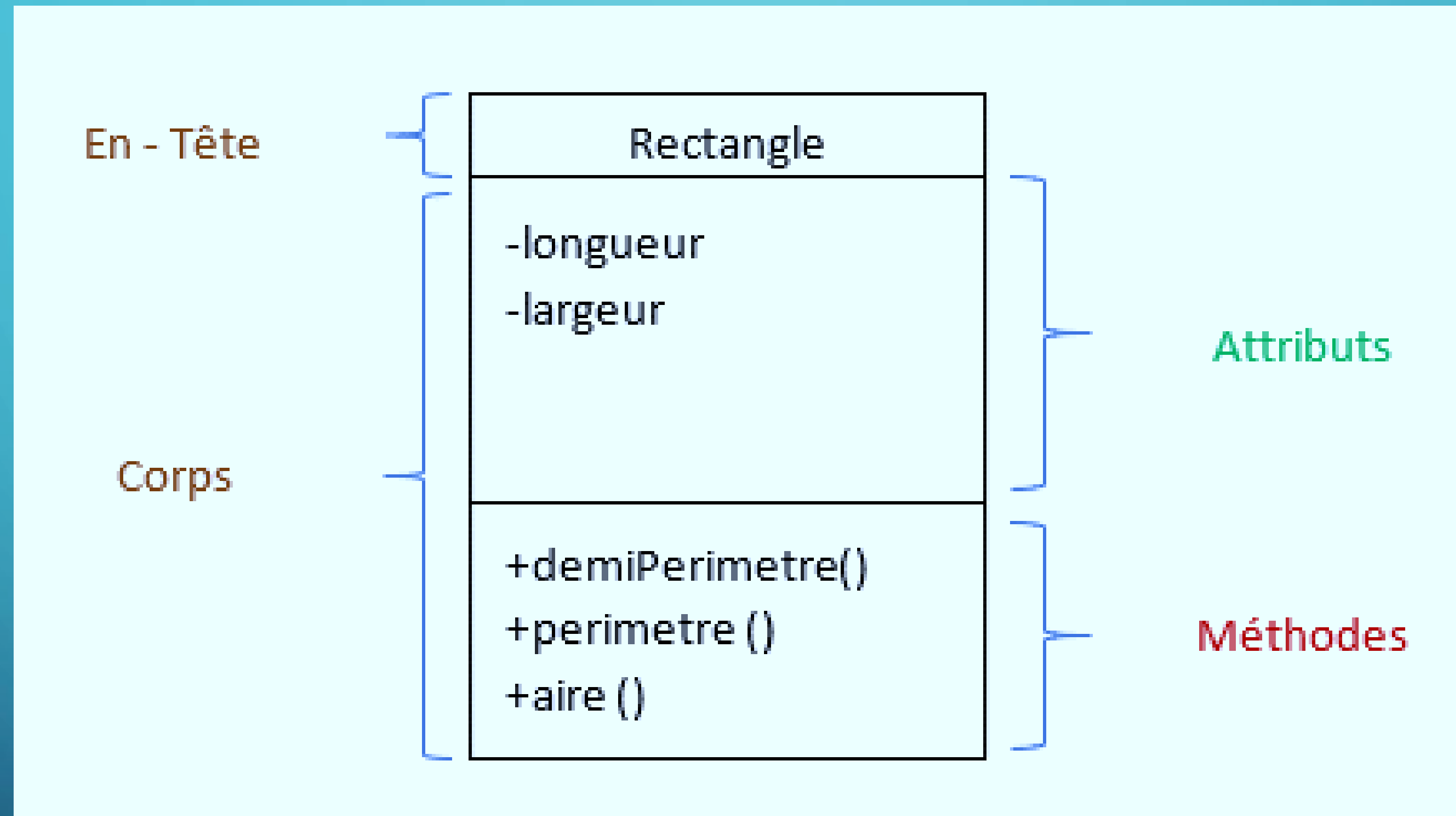
Une classe comporte sa déclaration, des variables et les définitions de ses fonctions.

Une classe se compose de deux parties : un en-tête et un corps.

Le corps peut être divisé en 2 sections : la déclaration des variables et des constantes (attributs) et la définition des fonctions (méthodes).

II - CLASSE

3- Structure



II - CLASSE

3- Structure

➤ Syntaxe de déclaration d'une classe

```
modificateurs class nom_de_classe [extends classe_mere] [implements interfaces]  
{  
    //Corps  
}
```

II - CLASSE

3- Structure

➤ Syntaxe de déclaration d'une classe

Les modificateurs de classe sont :

- **abstract** : la classe contient une ou des méthodes abstraites, qui n'ont pas de définition explicite. Une classe déclarée abstract ne peut pas être instanciée.
- **final** : la classe ne peut pas être modifiée, sa redéfinition grâce à l'héritage est interdite. Les classes déclarées final ne peuvent donc pas avoir de classes filles.
- **private** : la classe n'est accessible qu'à partir du fichier où elle est définie.
- **public** : La classe est accessible partout

II - CLASSE

3- Structure

➤ Syntaxe de déclaration d'une classe

Les modificateurs `abstract` et `final` ainsi que `public` et `private` sont mutuellement exclusifs.

Le mot clé **`extends`** permet de spécifier une superclasse éventuelle : ce mot clé permet de préciser la classe mère dans une relation d'héritage.

Le mot clé **`implements`** permet de spécifier une ou des interfaces que la classe implémente. Cela permet de récupérer quelques avantages de l'héritage multiple.

Exemple: **`class Rectangle { }`**

II - CLASSE

4- Attributs

Un attribut est une caractéristique d'une classe désignée par un nom permettant de mémoriser une ou plusieurs valeur.

Ce sont des variables stockant des informations d'état de l'objet.

Chaque attribut a un type de données. C'est ce dernier qui détermine l'ensemble de valeur possible à mémoriser.

II - CLASSE

4- Attributs

➤ Syntaxe de déclaration d'un attribut

modificateurs **type_attribut** nom_attribut [*expression*]

Le type d'un attribut peut être une classe.

II - CLASSE

4- Attributs

➤ Syntaxe de déclaration d'un attribut

Les modificateurs d'attribut sont :

- **public** : la variable est accessible partout.
- **protected** : la variable est accessible dans le même package et les sous-classes.
- **private** : la variable est accessible uniquement à l'intérieur de la classe.
- **final** : la valeur de la variable ne peut plus être modifiée une fois que celle-ci est initialisée. Elle permet de définir des constantes.
- **static** : la variable est partagée entre toutes les instances d'une même classe. On parle de variable de classe. Elle est directement accessible par la classe.

II - CLASSE

4- Attributs

➤ Exemple

private double **longueur**;

private double **largeur**;

private double **longueur, largeur**;

II - CLASSE

5- Méthodes

Une méthode (opération) est une modélisation d'une action applicable sur un objet, caractérisée par un en-tête appelé signature définissant son nom et ses paramètres.

C'est est une suite d'instructions qui manipule les caractéristiques et l'état d'un objet.

❖ Un **message** est une demande d'activation envoyé à un objet d'une classe.

II - CLASSE

5- Méthodes

➤ Syntaxe de déclaration d'une méthode

```
modificateurs type_retourné nom_méthode ( arg1, ..., argn )  
{  
    // Définition des variables locales et du bloc d'instructions  
}
```

Le type retourné peut être élémentaire ou correspondre à une classe (pour un objet).

II - CLASSE

5- Méthodes

➤ Syntaxe de déclaration d'une méthode

Les modificateurs de méthodes sont :

- **abstract** : la méthode n'est pas implémentée et devra être redéfinie par dans les sous-classes.
- **public** : la méthode est accessible aux méthodes des autres classes.
- **protected** : la méthode ne peut être invoquée que par des méthodes de la classe ou de ses sous-classes.
- **private** : la méthode est réservée aux autres méthodes de la même classe.
- **final** : la méthode ne peut être modifiée. La redéfinition lors de l'héritage interdite.
- **static** : la méthode appartient simultanément à tous les objets de la classe. Il est inutile d'instancier la classe pour appeler la méthode mais la méthode ne peut pas manipuler de variable d'instance. Elle ne peut utiliser que des variables de classes.

II - CLASSE

5- Méthodes

➤ Exemple

```
public double demiPerimetre ()  
{  
    // Définition des variables locales et du bloc d'instructions  
}  
  
double Perimetre ()  
{  
    // Définition des variables locales et du bloc d'instructions  
}
```

N.B : *Si toutes les méthodes d'une classe sont abstraits on considère que la classe est une interface.*

II - CLASSE

5- Méthodes

1- Constructeur

Un constructeur est une méthode spécifique de la classe qui est appelée implicitement à l'instanciation de l'objet.

Elle permet de créer un objet de cette classe et d'initialiser ses attributs.

Il porte le même nom que la classe et ne retourne pas de valeur.

II - CLASSE

5- Méthodes

1- Constructeur

On distingue 3 types de constructeurs :

- constructeur par défaut : il n'a pas de paramètre.
- constructeur paramétré : il a comme paramètre les attributs de la classe.
- constructeur par copie : il a un unique paramètre du même type que l'objet à créer et il recopie les attributs depuis l'objet passé en paramètre sur l'objet à créer.

II - CLASSE

5- Méthodes

1- Constructeur

Exemples:

```
public Rectangle() { }  
public Rectangle(double uneLongueur, double uneLargeur) {  
    longueur = uneLongueur ;  
    largeur = uneLargeur ;  
}  
public Rectangle(Rectangle unRectangle) {  
    longueur = unRectangle.longueur ;  
    largeur = unRectangle.largeur ;  
}
```


II - CLASSE

5- Méthodes

2- Destructeur

Un destructeur est une méthode spécifique de la classe appelée implicitement à la destruction de l'objet.

Il porte le même nom que la classe précédé de ~ (**tilde**), ne retourne pas de valeur et n'accepte aucun paramètre.

II - CLASSE

5- Méthodes

3- Sélecteur

Encore appelé les accesseurs, les sélecteurs sont des méthodes qui permettent de récupérer la valeur d'un attribut (caché) sans y accéder directement de l'extérieur ; ils sécurisent donc l'attribut en restreignant sa modification.

Exemples:

```
public double getLongueur () {  
    return longueur;  
}
```

II - CLASSE

5- Méthodes

4- Modificateur

Encore appelé les mutateurs, les modificateurs permettent de modifier l'état des attributs tout en vérifiant si la valeur que l'on veut donner à l'attribut respecte les normes de celle-ci ou diverses règles de cohérence.

Exemples:

```
public void setLongueur (double uneLongueur) {  
    longueur = uneLongueur ;  
}
```


III – LE MOT CLE THIS

Le mot-clé **this** désigne, dans une classe, l'instance courante de la classe elle-même.

Il est utilisable seulement dans les méthodes non statiques d'une classe.

III – LE MOT CLE THIS

Il est utilisé pour :

- Rendre le code explicite et éviter le masquage : Si, dans une méthode, un attribut est masqué alors la valeur de l'attribut peut être référencée à l'aide du mot réservé **this**.
- S'auto-désigner comme référence : Le mot clé **this** peut être utilisé pour passer une référence à l'instance elle-même dans une méthode.

Exemples:

```
public void setLongueur (double longueur) {  
    this.longueur = longueur ;  
}
```