

Programmation Orientée Objet : Introduction générale

I. Alaya ¹

¹ines.alaya@parisnanterre.fr

Licence MIASHS Miage DL CMI - 2025/2026

Grandes lignes du cours

Problématique de la programmation.

Conception par traitements.

Conception par objets.

La philosophie de la programmation orientée objet.

Le concept d'objet.

Le concept d'encapsulation.

Le concept de polymorphisme.

Le concept d'héritage.

Problématique de la programmation.

Problématique de la programmation.

Conception par traitements.

Conception par objets.

La philosophie de la programmation orientée objet.

Le concept d'objet.

Le concept d'encapsulation.

Le concept de polymorphisme.

Le concept d'héritage.

Problématique de la programmation

Le schéma simplifié d'un système informatique peut se résumer par la formule :

Système informatique = Structures de données + Traitements

Problématique de la programmation

Le schéma simplifié d'un système informatique peut se résumer par la formule :

Système informatique = Structures de données + Traitements

Le cycle de vie d'un système peut être décomposé en deux grandes phases :

- ▶ Une phase de **production** qui consiste à réaliser le logiciel.
- ▶ Une phase de **maintenance** qui consiste à corriger et à faire évoluer le logiciel.

Problématique de la programmation

Le schéma simplifié d'un système informatique peut se résumer par la formule :

Système informatique = Structures de données + Traitements

Le cycle de vie d'un système peut être décomposé en deux grandes phases :

- ▶ Une phase de **production** qui consiste à réaliser le logiciel.
- ▶ Une phase de **maintenance** qui consiste à corriger et à faire évoluer le logiciel.

Lors de la production du système (au sens industriel du terme), le concepteur a deux grandes options :

- ☞ soit orienter sa conception sur **les traitements**,
- ☞ soit orienter sa conception sur **les données**.

Outline

Problématique de la programmation.

Conception par traitements.

Conception par objets.

La philosophie de la programmation orientée objet.

Le concept d'objet.

Le concept d'encapsulation.

Le concept de polymorphisme.

Le concept d'héritage.

Conception par traitements.

Problématique de la programmation.

Conception par traitements.

Conception par objets.

La philosophie de la programmation orientée objet.

Le concept d'objet.

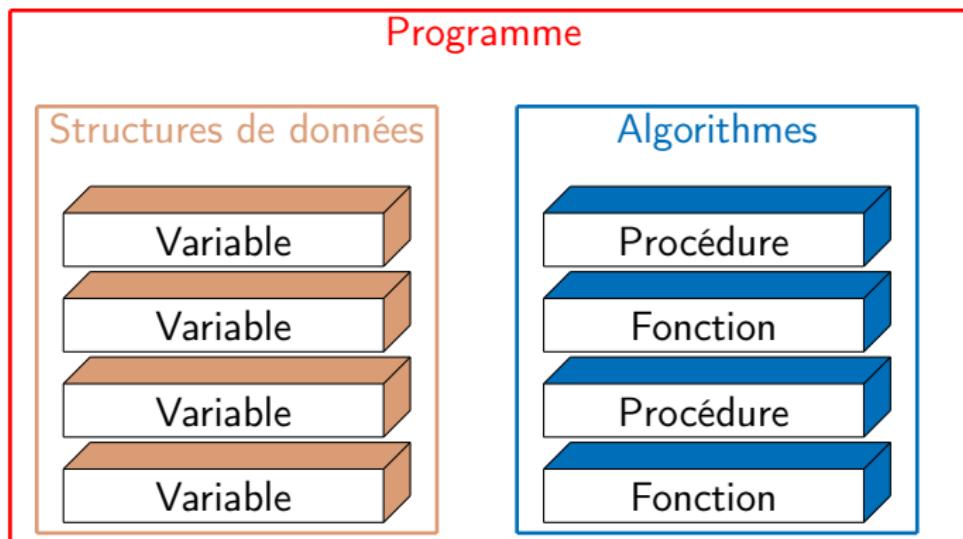
Le concept d'encapsulation.

Le concept de polymorphisme.

Le concept d'héritage.

Conception par traitements : principe

Principe : On sépare les données des moyens de traitement de ces données.



Conception par traitements : +/-

- ▶ Les premiers concepteurs de système informatique ont adopté cette approche : systèmes d'exp., gestionnaires de fenêtres, logiciels de gestion, logiciels de bureautique, logiciels de calcul scientifique, etc.
- ▶ De nombreux systèmes informatiques sont encore développés selon cette approche.
- ☞ Systèmes *ad-hoc*, i.e., adaptés au problème de départ, mais dont la **maintenance est difficile**.
- ☞ Les traitements sont généralement beaucoup **moins stables** que les données : changement de spécification, ajout de nouvelles fonctionnalités, etc.
- ☞ Les structures de données sous-jacentes sont choisies en **relation étroite** avec les traitements à effectuer.

Outline

Problématique de la programmation.

Conception par traitements.

Conception par objets.

La philosophie de la programmation orientée objet.

Le concept d'objet.

Le concept d'encapsulation.

Le concept de polymorphisme.

Le concept d'héritage.

Conception par objets.

Problématique de la programmation.

Conception par traitements.

Conception par objets.

La philosophie de la programmation orientée objet.

Le concept d'objet.

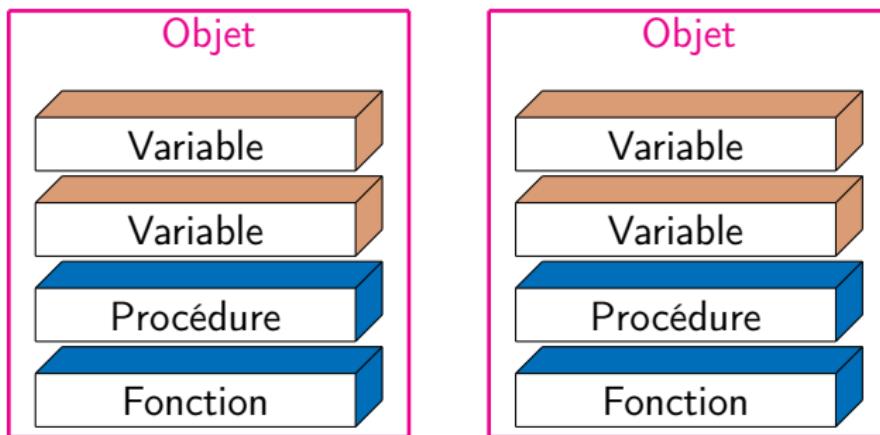
Le concept d'encapsulation.

Le concept de polymorphisme.

Le concept d'héritage.

Conception par objets : principe

Principe : afin d'établir de façon stable et robuste l'architecture d'un système, il semble raisonnable de s'organiser autour des données manipulées.



Conception par objets : points clés

- ▶ La construction d'un système va s'axer principalement sur la détermination des données dans un premier temps et la réalisation des traitements (de haut-niveau) agissant sur ces données dans un second temps.
- ▶ Cette approche permet de bâtir des systèmes plus simples à maintenir et à faire évoluer.
- ▶ On regroupe dans une même entité informatique, appelé **objet**, les structures de données et les moyens de traitement de ces données.

La philosophie de la programmation orientée objet.

Problématique de la programmation.

Conception par traitements.

Conception par objets.

La philosophie de la programmation orientée objet.

Le concept d'objet.

Le concept d'encapsulation.

Le concept de polymorphisme.

Le concept d'héritage.

Les 4 fondamentaux de la Programmation Orientée Objet

La conception orientée objet repose sur les quatre notions fondamentales suivantes :

- ▶ Le concept d'objet,
- ▶ le principe d'encapsulation,
- ▶ le polymorphisme,
- ▶ et l'héritage.

Outline

Problématique de la programmation.

Conception par traitements.

Conception par objets.

La philosophie de la programmation orientée objet.

Le concept d'objet.

Le concept d'encapsulation.

Le concept de polymorphisme.

Le concept d'héritage.

Le concept d'objet.

Problématique de la programmation.

Conception par traitements.

Conception par objets.

La philosophie de la programmation orientée objet.

Le concept d'objet.

Le concept d'encapsulation.

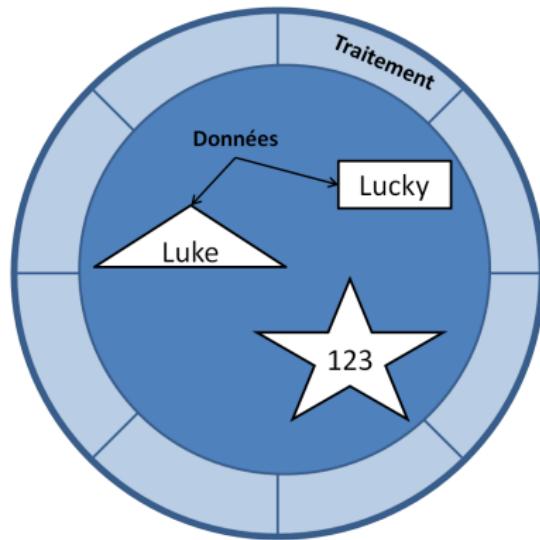
Le concept de polymorphisme.

Le concept d'héritage.

Le concept d'objet : définition

Définition

*Un **objet** est une entité autonome, qui regroupe un ensemble de propriétés (données) cohérentes et de traitements associés.*



Le concept d'objet : vocabulaire

- ▶ Les structures de données définies dans l'objet sont appelés ses **attributs (propriétés)**.
- ▶ Les procédures et fonctions définies dans l'objet sont appelés ses **méthodes (opérations)**.
- ▶ Les attributs et méthodes d'un objet sont appelés ses **membres**.
- ▶ L'ensemble des valeurs des attributs d'un objet à un instant donné est appelé **état interne**.

Le concept d'objet : vocabulaire (suite)

- ▶ L'ensemble des méthodes d'un objet accessibles de l'extérieur (depuis un autre objet) est appelé **interface**. Elle caractérise le comportement de l'objet.
- ▶ L'invocation d'une méthode d'interface est appelé **appel de méthode**. Elle peut être vu comme un envoi de message entre objet.

Le concept d'objet : notion de classe

Pour être véritablement intéressante, la notion d'objet doit permettre un certain **degré d'abstraction** ⇒ **notion de classe**.

Le concept d'objet : notion de classe

Pour être véritablement intéressante, la notion d'objet doit permettre un certain **degré d'abstraction** ⇒ **notion de classe**.

Définition

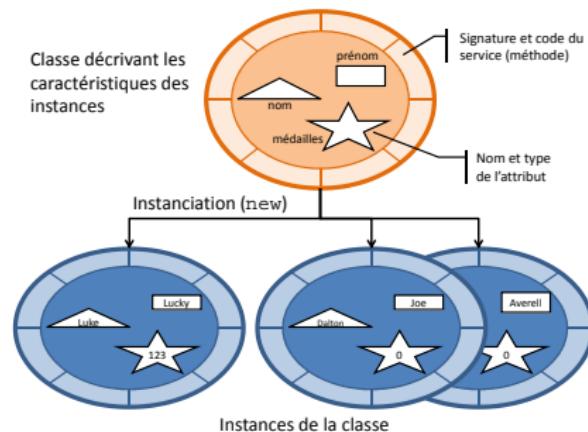
*On appelle **classe** la structure (modèle) d'un objet, i.e., la déclaration de l'ensemble membres qui composeront un objet.*

Le concept d'objet : notion de classe

Pour être véritablement intéressante, la notion d'objet doit permettre un certain **degré d'abstraction** ⇒ **notion de classe**.

Définition

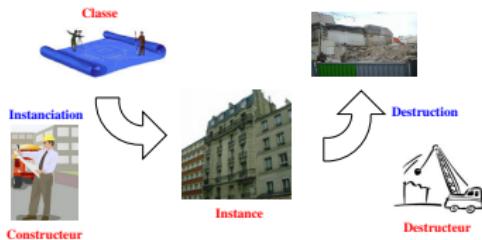
*On appelle **classe** la structure (modèle) d'un objet, i.e., la déclaration de l'ensemble membres qui composeront un objet.*



Le concept d'objet : création/destruction

Instants particulièrement importants,

- ▶ naissance d'un objet : appelée **instanciation (construction)**,
- ▶ et mort d'un objet : appelée **destruction**.



Le **constructeur** est la méthode qui permet de créer des instances dont les traits sont décrits par la classe.

Le **destructeur** est la méthode qui permet de détruire une instance de la classe.

Le concept d'objet : classe *versus* objet

Il est important de saisir la différence entre les notions de **classe** et **instance de la classe** :

classe = attributs + méthodes + mécanismes d'instanciation +
mécanismes de destruction

Le concept d'objet : classe *versus* objet

Il est important de saisir la différence entre les notions de **classe** et **instance de la classe** :

classe = attributs + méthodes + mécanismes d'instanciation +
 mécanismes de destruction

instance de la classe = valeurs des attributs + accès aux méthodes

Le concept d'objet : classe *versus* objet

Il est important de saisir la différence entre les notions de **classe** et **instance de la classe** :

instance de la classe = valeurs des attributs + accès aux méthodes

L'**instanciation** est le mécanisme qui permet de créer des instances dont les traits sont décrits par la classe.

La **destruction** est le mécanisme qui permet de détruire une instance de la classe.

L'ensemble des instances d'une classe constitue l'**extension de la classe**.

Outline

Problématique de la programmation.

Conception par traitements.

Conception par objets.

La philosophie de la programmation orientée objet.

Le concept d'objet.

Le concept d'encapsulation.

Le concept de polymorphisme.

Le concept d'héritage.

Le concept d'encapsulation.

Problématique de la programmation.

Conception par traitements.

Conception par objets.

La philosophie de la programmation orientée objet.

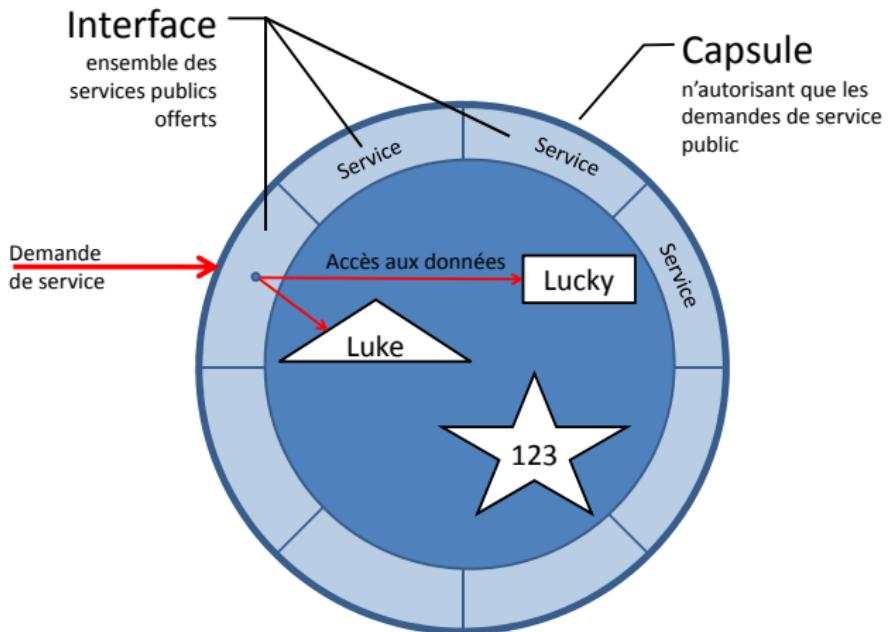
Le concept d'objet.

Le concept d'encapsulation.

Le concept de polymorphisme.

Le concept d'héritage.

Encapsulation : schématisation



Encapsulation : définition

Définition

*Le terme **encapsulation** désigne le principe consistant à cacher l'information contenue dans un objet et de ne proposer que des méthodes de modification/accès à ces propriétés (attributs).*

Encapsulation : définition

Définition

*Le terme **encapsulation** désigne le principe consistant à cacher l'information contenue dans un objet et de ne proposer que des méthodes de modification/accès à ces propriétés (attributs).*

- ▶ L'objet est vu de l'extérieur comme une **boîte noire** ayant certaines propriétés et ayant un comportement spécifié.
- ▶ La manière dont le comportement est implémenté reste cachée aux utilisateurs de l'objet.

Intérêt

Protéger la structure interne de l'objet contre toute manipulation non contrôlée, produisant une incohérence.

Encapsulation : en pratique

L'encapsulation nécessite la spécification de **parties publics et privées** de l'objets.

Éléments publics : correspond à la partie visible de l'objet depuis l'extérieur. c'est un ensemble de méthodes utilisables par d'autres objets (environnement).

Éléments privées : correspond à la partie non visible de l'objet. Il est constitué des éléments de l'objet visibles uniquement de l'intérieur de l'objet et de la définition des méthodes.

Outline

Problématique de la programmation.

Conception par traitements.

Conception par objets.

La philosophie de la programmation orientée objet.

Le concept d'objet.

Le concept d'encapsulation.

Le concept de polymorphisme.

Le concept d'héritage.

Le concept de polymorphisme.

Problématique de la programmation.

Conception par traitements.

Conception par objets.

La philosophie de la programmation orientée objet.

Le concept d'objet.

Le concept d'encapsulation.

Le concept de polymorphisme.

Le concept d'héritage.

Le terme **polymorphisme** désigne à la fois,

1. la possibilité de spécifier le comportement *ad-hoc* d'une méthode selon le type d'objets l'invoquant (**redéfinition** ; en anglais « *overriding* »).
2. la possibilité de définir des comportements différents pour la même méthode selon les arguments passés en paramètres (**surcharge** ; en anglais « *overloading* »).

Polymorphisme : définition

Définition

Le **polymorphisme** peut être vu comme la capacité de choisir dynamiquement la méthode qui correspond au type réel de l'objet.

Polymorphisme : exemple

Exemple

Si l'on considère deux types d'objets Ordinateur et FourElectrique possédant tous les deux une méthode allumer. Cette méthode est implémentée différemment selon le type d'appareil.

Exemple

Si l'on considère une classe Rectangle possédant une méthode nommée redimensionner, on peut spécifier un comportement différent selon qu'elle est invoquée avec un argument de type entier ou deux arguments de type flottant :

- ▶ redimensionner(5) multiplie la longueur de chacun des côtés par 5.
- ▶ redimensionner(1.5,2.5) ajoute 1.5cm à la largeur et 2.5cm à la longueur.

Outline

Problématique de la programmation.

Conception par traitements.

Conception par objets.

La philosophie de la programmation orientée objet.

Le concept d'objet.

Le concept d'encapsulation.

Le concept de polymorphisme.

Le concept d'héritage.

Le concept d'héritage.

Problématique de la programmation.

Conception par traitements.

Conception par objets.

La philosophie de la programmation orientée objet.

Le concept d'objet.

Le concept d'encapsulation.

Le concept de polymorphisme.

Le concept d'héritage.

Héritage : définition

Définition

*Le terme **Héritage** désigne le principe selon lequel une classe peut hériter de caractéristiques (c'est-à-dire, d'attribut et de méthodes) d'une ou plusieurs autres classes.*

Héritage : définition

Définition

*Le terme **Héritage** désigne le principe selon lequel une classe peut hériter de caractéristiques (c'est-à-dire, d'attribut et de méthodes) d'une ou plusieurs autres classes.*

Vocabulaire

Une classe *A* héritant d'une classe *B* est appelée **classe dérivée** de *B* et la classe *B* est appelée la **superclasse** de *A* (la classe dont elle dérive).

Héritage : définition

Définition

*Le terme **Héritage** désigne le principe selon lequel une classe peut hériter de caractéristiques (c'est-à-dire, d'attribut et de méthodes) d'une ou plusieurs autres classes.*

Vocabulaire

Une classe *A* héritant d'une classe *B* est appelée **classe dérivée** de *B* et la classe *B* est appelée la **superclasse** de *A* (la classe dont elle dérive).

Remarque(s)

Un des intérêts de l'héritage est de pouvoir définir de nouveaux attributs et de nouvelles méthodes pour la classe dérivée, qui viennent s'ajouter à ceux et celles héritées.

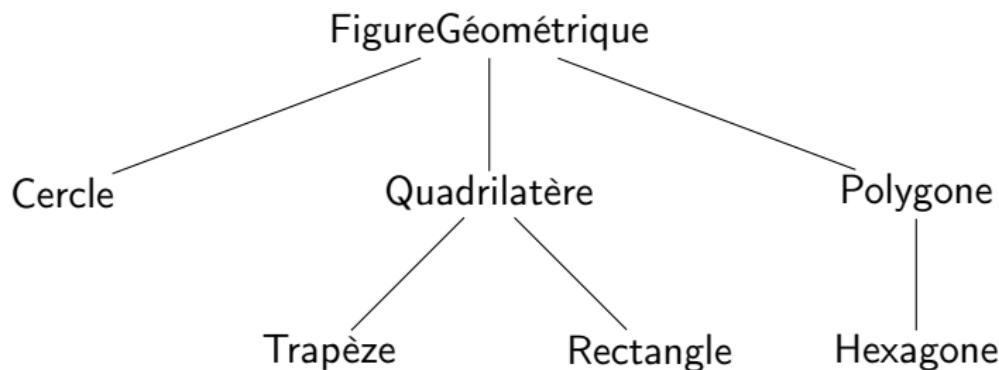
Héritage : Hiérarchie des classes

L'héritage entre les classes peut être répété : une classe *A* ayant hérité d'une classe *B* peut-elle même transmettre certaines de ses caractéristiques à une classe dérivée *C*.

Héritage : Hiérarchie des classes

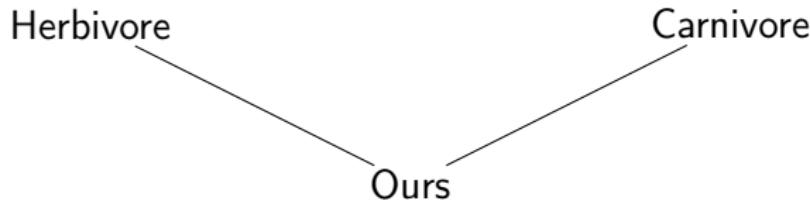
L'héritage entre les classes peut être répété : une classe *A* ayant hérité d'une classe *B* peut-elle même transmettre certaines de ses caractéristiques à une classe dérivée *C*.

On représente généralement sous forme de diagramme les relations de parenté qui existent entre différentes classes.



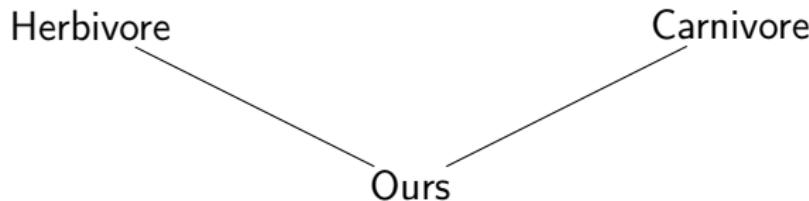
Héritage : héritage multiple

Une classe peut hériter de caractéristiques de plusieurs classes, on parle alors d'**héritage multiple**.



Héritage : héritage multiple

Une classe peut hériter de caractéristiques de plusieurs classes, on parle alors d'**héritage multiple**.



Remarque(s)

Par opposition, lorsqu'une classe hérite de caractéristiques d'une seule classe, on parle d'**héritage simple**.

Dans de nombreux langages orientés objets (tels que C++ et Java), le concept d'héritage a amené à introduire une catégorie de membres intermédiaire entre les membres publics et les membres privés :

- ▶ les membres **protégés**. Ce sont des les membres visibles par les instances de la classe et des ses dérivées.