Les concepts de l'approche objet

Conception du nouveau SI avec UML

Table des matières

Introduction

Ce cours a pour objectif de vous faire découvrir les différents concepts et principes de l'approche objet qui sont à la base d'UML. Leur connaissance est indispensable pour comprendre les éléments utilisés dans la panoplie des diagrammes d'UML qui seront abordés dans les vidéos qui suivent.

Dans un premier temps, nous aborderons le concept d'objet, puis nous verrons, par abstraction, comment le modéliser en UML.

La notion de classe, représentation commune d'un ensemble d'objets similaires, sera introduite. Nous évoquerons ensuite le principe d'encapsulation, masquage d'information internes et propres au fonctionnement de l'objet.

Les relations se spécialisation et de généralisation introduisant les hiérarchies de classes seront décrites, ainsi que l'héritage, les classes concrètes et abstraites puis nous aborderons le polymorphisme, conséquence directe de la spécialisation.

Enfin, nous évoquerons la composition d'objets avant de terminer sur une notion plus spécifique à UML, la spécialisation des éléments du diagramme par les stéréotypes.

Ou

Notre objectif est donc de présenter l'essentiel des concepts objet qui nous paraissent nécessaires à une bonne compréhension d'UML. Les concepts qui nous semblent importants à bien maîtriser sont les suivants :

- ✓ objet et classe,
- ✓ encapsulation et interface,
- ✓ association et agrégation de classes,
- ✓ généralisation et spécialisation de classe,
- ✓ polymorphisme,
- ✓ persistance.

Objet

Qu'est-ce qu'un objet ?

Un objet est une entité identifiable du monde réel. Il peut avoir une existence physique (un livre, un acteur) ou ne pas en avoir (Une commande client). Identifiable signifie que l'objet peut être désigné.

En UML, tout objet possède un ensemble d'attributs appelés aussi des propriétés et un ensemble de méthodes appelés aussi des opérations. Un attribut est une variable destinée à recevoir une valeur. Une méthode est une opération qui prend des valeurs en entrée et modifiant les valeurs des attributs ou produisant un résultat.

On peut dire aussi, qu'objet possède une structure, un état et un comportement. La structure est constituée par des attributs, les valeurs des attributs constitue l'état et le comportement est constitué de la liste des méthodes.

Exemple

Considérons l'étudiant Ali Baba, n° 1234, inscrit à la filière Informatique le 13/09/2021 appartenant au Groupe B

Cet objet est caractérisé par la liste de ses attributs :

- ✓ N° étudiant : 1234
- ✓ Nom et prénom : Ali Baba
- ✓ Date inscription : 13/09/2021
- ✓ Filière : Informatique
- ✓ Groupe B

Et des méthodes qu'il peut exécuter. Dans notre cas, voici les la liste des méthodes :

- ✓ S'inscrire
- ✓ Modifier informations
- ✓ Changer filière
- ✓ Changer groupe

Tout système conçu en UML est composé d'objets interagissant entre eux et effectuant les opérations propres à leur comportement.

L'abstraction

L'abstraction est un principe très important en modélisation. Elle consiste à retenir uniquement les propriétés pertinentes d'un objet pour un problème précis. Les objets utilisés en UML sont des abstractions d'objets du monde réel.

D'une autre façon, l'abstraction est le processus qui consiste à représenter des objets appartiennent au monde réel en garant que les propriétés pertinentes, attachées aux objets que l'on souhaite manipuler.

Supposons que l'on veuille écrire un programme qui gère des maisons. Si l'on se place du point de vue de celui qui l'habite, on s'intéressera probablement au nombre de pièces, à leurs surfaces, si elle est proche des commerces, si elle est accessible par les transports en commun.

Celui qui la construit devra connaître les délais de construction, avoir le permis de construire et les plans, avoir des détails sur la façon de la connecter aux différents réseaux : égouts, eau, électricité, téléphone, etc…

Du point de vue de la municipalité qui a besoin de calculer ses impôts locaux, il faudra connaître la surface, la date de construction, l'implantation locale, etc…

Les classes d'objets

Un ensemble d'objet similaire, c'est-à-dire possédant la même structure et le même comportement forme une classe d'objets. Ne pas oublier que la structure est constituée des attributs et le comportement est constitué des méthodes.

Chaque objet d'une classe, appelé aussi une instance de classe, se distingue par son identité propre et possède des valeurs spécifiques pour ses attributs.

Exemple :

Soit les deux objets suivants de deux étudiants

Etudiant 1
✓ N° étudiant : 1234 ✓ Nom et prénom : Ali Baba ✓ Date inscription : 13/09/2021

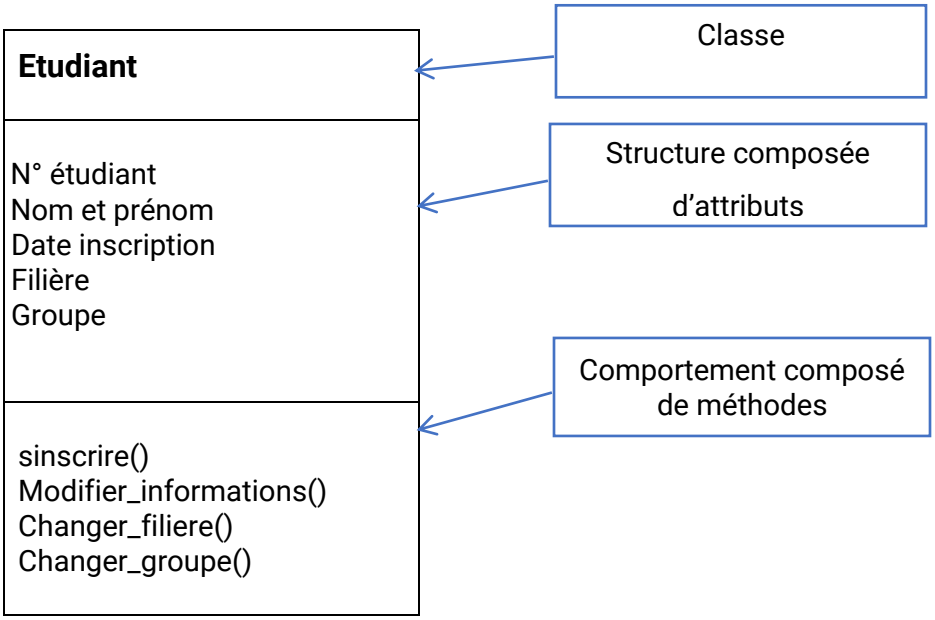
Etudiant 2
✓ N° étudiant : 5678 ✓ Nom et prénom : Salhi Ahmed ✓ Date inscription : 03/09/2021

✓ Filière : Informatique ✓ Groupe : B

✓ Filière : Mathématique ✓ Groupe : A

Nous constatons que tous les étudiants partagent la même structure et le même comportement et on peut les regrouper dans un seul moule appelé Classe. Il s'agit de la classe **Etudiant**.

Voici une représentation de la classe étudiant :



L'étudiant Ali Baba est une instance de la classe étudiant dont les attributs et les valeurs sont attribués sont illustrés come ci-dessous

Ali Baba : Etudiant
N° étudiant = 1234 Nom et prénom = Ali Baba Date inscription = 13/09/2021 Filière = Informatique Groupe = B

Un objet représente une classe dans le monde réel. Un objet est une instanciation d'une classe, on parle d'instance ou d'objet. L'étudiant Ali Baba et l'étudiant Salhi Ahmed sont deux objets ou deux instances de la classe étudiant.

Une classe représente une abstraction générique d'un objet. On dit la classe étudiant, la classe vendeur, la classe commande.

Attributs

Les attributs d'un objet représentent les propriétés caractérisant cet objet. Les attributs constitue la structure d'un objet. Ce sont les données spécifiques à cet objet.

Dans l'exemple ci-dessous, la classe Employé a quatre attributs (numéro Employé, nomEmployé, prénomEmployé, adresse); l'objet Hind a également quatre attributs dont les valeurs sont connues (son numéro d'employé est 0103, son nom est Sabri, son prénom est Hind, son adresse est le 15 avenue Hassan 2).

Hind : Employé
numéro Employé : 0103 nomEmployé :Sabri prénomEmployé : Hind adresse : 15 avenue Hassan 2

Les méthodes ou les opérations

Les opérations ou les méthodes caractérisent le comportement de l'objet. Il s'agit de l'ensemble des actions qui peuvent modifier les attributs de l'objet lui-même ou d'autres objets. Ces actions peuvent être faites par l'objet sur un ou plusieurs autres objets ou sur lui par un ou plusieurs autres objets

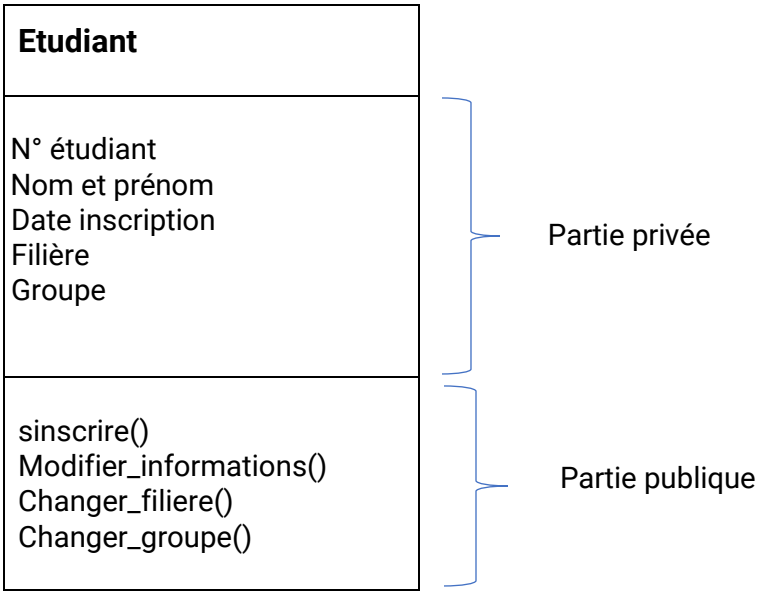
Compte
NuméroCompte SoldeCompte
retirer() déposer calculerSoldeCompte()

Client
nomClient prénomClient codeClient adresseClient
modifierInformation()

L'encapsulation

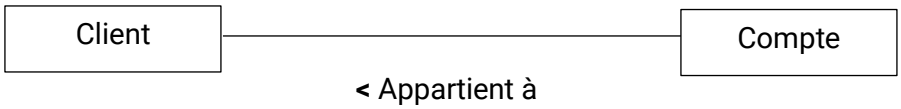
L'encapsulation consiste à masquer des attributs et des méthodes de l'objet vis-à-vis des autres objets. En effet, certains attributs et méthodes ont pour seul objectif des traitements interne à l'objet et ne doivent pas être accessible aux objets extérieurs. Les attributs et les méthodes encapsulés sont appelés privés, et s'ils sont accessible de l'extérieur sont déclarés public.

Plus précisément, les attributs ne sont accessibles qu'à partir de méthodes définies dans la classe. L'ensemble des méthodes d'une classe rendu visible aux autres classes porte le nom d'interface.



Les associations

Les associations représentent les liens ou les relations entre deux ou plusieurs classes. Elles sont représentées par une ligne. Le nom de l'association est écrit au-dessus ou au-dessous de la ligne. Cependant, quand l'association est évidente son nom peut être omis.



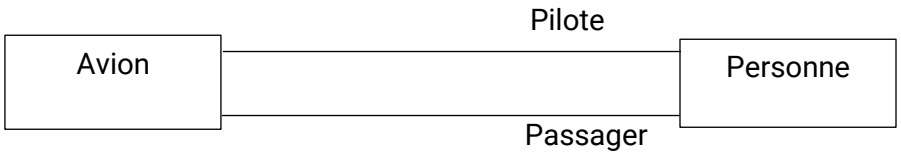
Le sens de l'association peut être précisé par les signes inférieur < ou supérieur > si cela est nécessaire. Dans l'exemple ci-dessus l'association Appartient à pointe vers Client pour traduire le fait que le compte appartient au client.

L'extrémité d'une association est appelée rôle. Chaque association binaire (entre deux classes) possède deux rôles. Le rôle décrit comment une classe voit une autre classe au travers d'une association. Le nom du rôle se distingue du nom d'une association par son placement de l'extrémité de l'association.

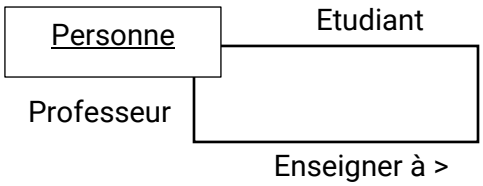
Ainsi, dans l'exemple suivant, le rôle du client est d'être détenteur du compte et celui du compte est d'être possédé par le client.

Lorsque deux classes sont reliées par une seule association, le nom des classes suffit souvent à caractériser le rôle. Le nommage des rôles prend tout son intérêt lorsque plusieurs associations relient deux classes.

Dans l'exemple suivant, les noms des rôles permettent de distinguer deux objets : pilote et passager appartenant à la classe Personne.



Aussi, dans l'exemple suivant, les noms des rôles permettent de distinguer deux objets : professeur et étudiant appartenant à la classe Personne et reliés par l'association *Enseigner à*.



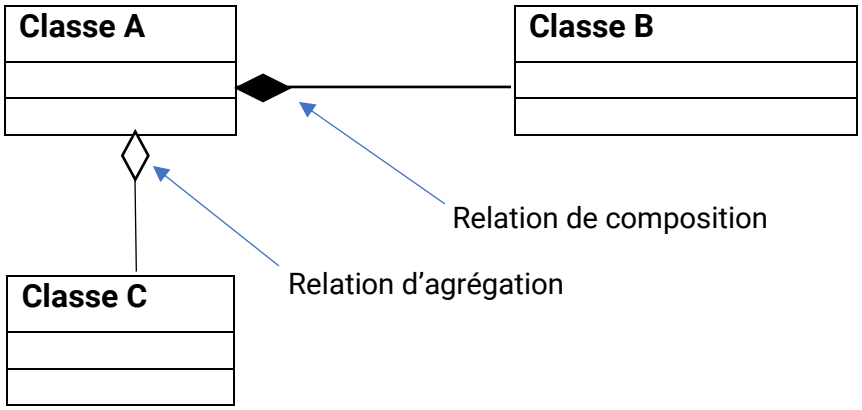
L'agrégation et la composition

L'agrégation et la composition sont deux types de relations particulières d'association entre classes. L'agrégation et la composition, les deux, expriment le fait qu'une classe est composée d'une ou plusieurs autres classes.

La composition peut être vue comme une relation **"faire partie de"**, c'est à dire que si un objet B fait partie d'un objet A alors **B ne peut pas exister sans A**. Ainsi si A disparaît alors B également.

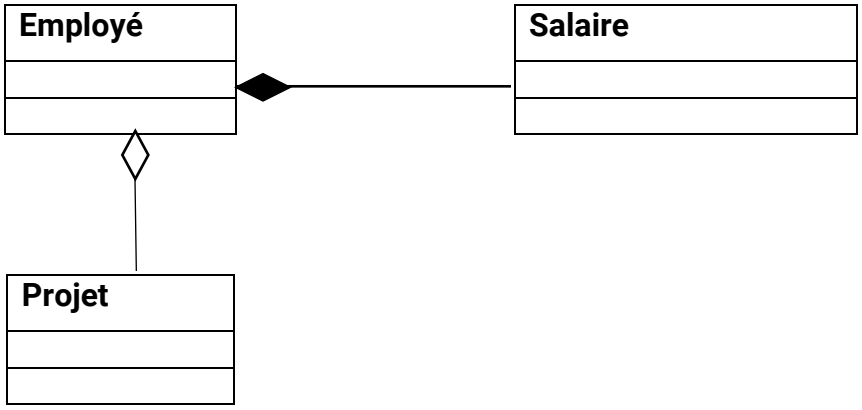
L'agrégation quant à elle est vue comme une relation de type **"avoir un",** c'est à dire que si un objet A a un objet B alors B peut vivre sans A.

En UML la représentation d'une relation de **composition** est faite avec un losange plein. La représentation d'une relation d'**agrégation** est représentée par un losange vide.



L'exemple suivant montre une agrégation et une composition. Un objet de Salaire **fait partie de** l'objet employé, si on supprime l'employé alors son salaire aussi est supprimé,

c'est une relation de **composition**. En revanche l'employé **a un** projet courant. Si le salarié part de l'entreprise le projet perdurera, c'est une relation d'**agrégation**.



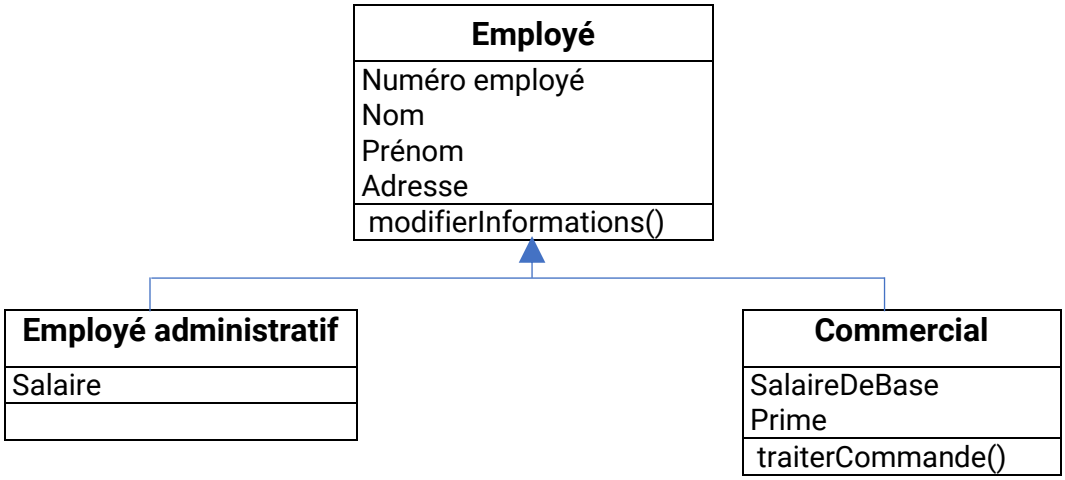
La spécialisation et généralisation

La généralisation permet de regrouper des classes ayant des caractéristiques communes, attributs et opérations, en une superclasse qui aura comme attributs et opérations ces caractéristiques communes. La généralisation permet donc de créer une classe plus générale.

La spécialisation représente la démarche inverse de la généralisation puisqu'elle consiste à créer à partir d'une classe, plusieurs classes spécialisées. Les principes de généralisation et de spécialisation vont permettre d'établir une hiérarchie entre les classes puisque certaines classes vont dériver d'autres classes.

Cette hiérarchisation des classes entre une superclasse, classe parent, et des classes dérivées est exprimé graphiquement par une flèche qui pointe vers la classe la plus générale.

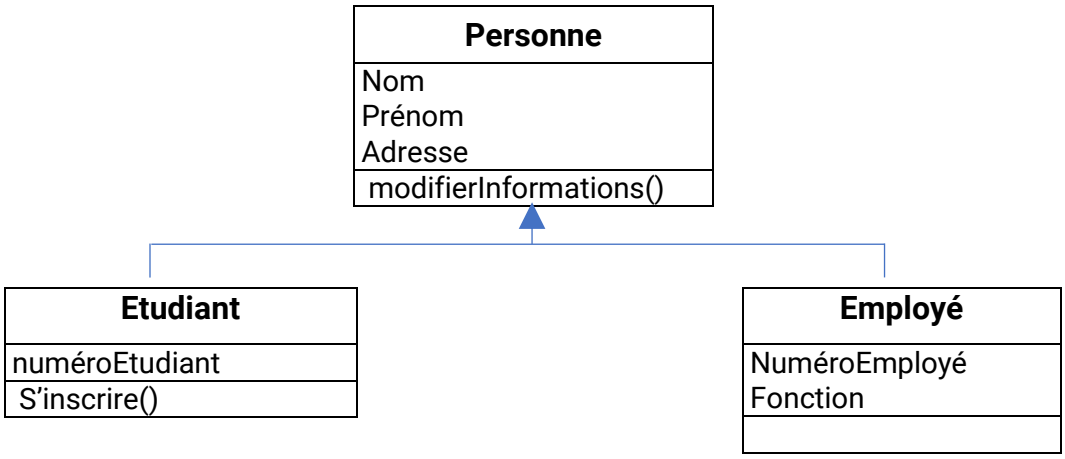
Dans l'exemple suivant, les classes Commercial et Employé administratif partagent les mêmes attributs : numéro employé, le nom, le prénom et ladresse et la même opération : modeifierInformations(). Au lieu des les répéter pour chacune, les attributs et l'opération communs sont regroupés dans une superclasse Employée. Dans les classes Employé administratif et Commercial, nous plaçons dans chaque classe, les attributs et les opérations spécifiques à chacune des deux classes. Dans la classe Employé administratif, on trouve l'attribut Salaire, alors que dans la classe Commercial, il y a les attributs salaireDeBase et prime



Héritage

Alors qu'est-ce que l'héritage ?

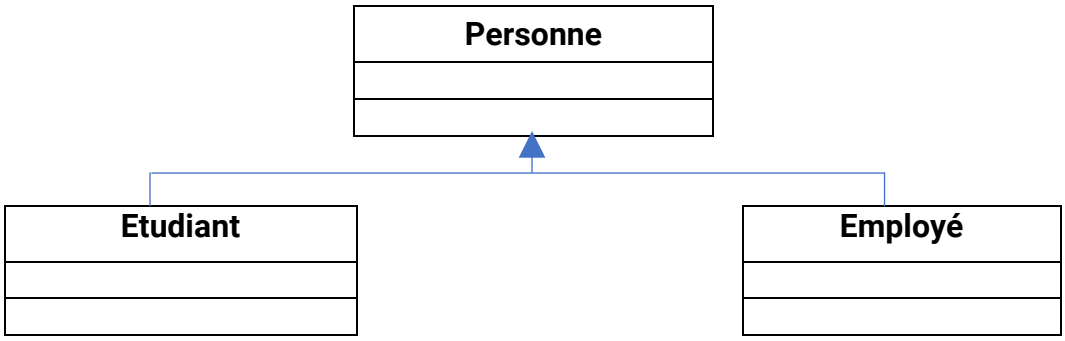
L'héritage est une autre façon d'exprimer la généralisation et la spécialisation. L'héritage est la propriété qui permet à une sous-classe d'hériter la structure, c'est-à-dire les attributs, et le comportement, c'est-à-dire les méthodes, de sa superclasse, c'est-à-dire la classe parent auxquelles viendront s'ajouter ses propres attributs et méthodes. C'est-à-dire, un objet A créé d'une sous-classe, peut hériter les attributs et les méthodes de sa superclasse auquel s'ajoutent ses propres attributs et ses propres méthodes. Ainsi, en plus des attributs et des méthodes de l'objet A, ce dernier dispose des attributs des méthodes de superclasse. Ces derniers sont considérés comme les siens.



Dans cet exemple, les classes Etudiant et Employé sont des sous-classes de la classe Personne, la flèche exprime la hiérarchie entre sous-classe et la superclasse. Chacune des sous-classes, hérite des attributs de la superclasse Personne : nom, prénom adresse. En outre, les classes Etudiant et Employé ont des attributs spécifiques : numéroEtudiant, numéroEmployé, fonction.

Les classes abstraites et concrètes

On parle de classe abstraite dans le cas de l'héritage. Soit l'exemple suivant :

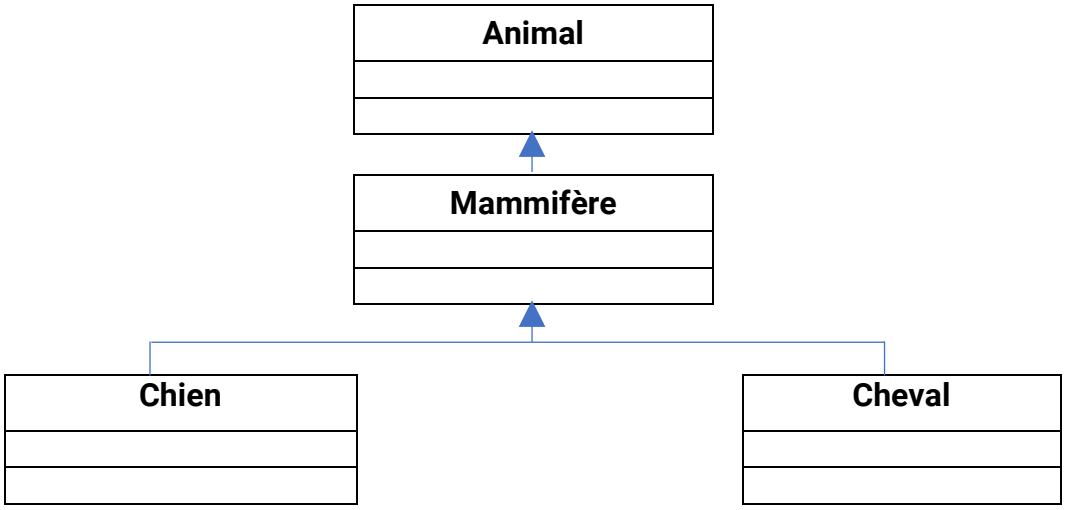


La classe Personne est une classe abstraite. Les classes Etudiant et Employé sont deux classes concrètes.

En fait la classe Personne a été créée pour factoriser les attributs et les méthodes communs aux classes Etudiant et Employé. Dans le monde réel, il y a soit un étudiant soit un employé. Une personne est abstraite, one sait pas de qui s'agit-il. On parle ainsi de classe abstraite.

Une classe abstraite a pour rôle d'être une superclasse et de factoriser les attributs et les méthodes communs des sous sous-classes.

Dans l'exemple suivant, les deux classes Animal et Mammifère sont des classes abstraites. Les classes Chien et classe Cheval sont deux classes concrètes.



Le polymorphisme

Le polymorphisme est la capacité donnée à une même opération de s'exécuter différemment suivant le contexte de la classe où elle se trouve.

Ainsi une opération définie dans une superclasse peut s'exécutera de manière différente selon la sous-classe où elle est héritée.

Exemple :

Soit la classe Employé et ses deux sous-classes Cadre et NonCadre.

1. **Nom de classe : Employé.**
    a. Attributs :
        i. numéro,
        ii. nom,
        iii. salaire de base.
    b. Opérations : calculSalaire( ).
2. **Nom de la sous-classe : Cadre**.
    a. Attributs :
        i. niveau d'encadrement.

b. Opérations : calculSalaire( ).

3. **Nom de la sous-classe : NonCadre**.
    a. Attributs :
        i. niveau de spécialisation.
        ii. Opérations : calculSalaire( ).

Le principe de calcul du salaire étant de calculer pour chaque type d'employé une prime spécifique en fonction soit du niveau d'encadrement, soit du niveau de spécialisation selon le type d'employé.

Voyons maintenant comment se réalise l'application du polymorphisme lors de l'exécution des opérations. Dans cet exemple, lorsque l'on appelle l'opération calculSalaire( ), c'est l'opération de sous-classe Cadre ou celle de la sous-classe NonCadre qui est en fait activée selon l'objet concerné. L'opération de la sous-classe fait en général appel explicitement à l'opération calculSalaire( ) de la super-classe pour bénéficier des traitements communs aux cadres et non cadres et ensuite il y aura poursuite du traitement spécifique à la sous-classe.