CHAPITRE II:
INGENIERIE DIRIGEE PAR DES MODELES (IDM)

Dr S.Fellah

Ingénierie Dirigée par des Modèles (IDM)

- Introduction à l'IDM
- Qu'est ce qu'un modèle
- Modélisation orientée objets

Introduction à l'IDM

- Suite à **l'approche objet** des années 80 et de son principe du « **tout est objet** », l'ingénierie du logiciel s'oriente vers **l'ingénierie dirigée par les modèles** (IDM) et le principe du « **tout est modèle** ».

- L'ingénierie dirigée par les modèles (IDM), ou «Model Driven Engineering» (MDE) permis plusieurs **améliorations** dans le **développement** des **systèmes complexes.**

- Il s'agit d'une forme **d'ingénierie générative** dans laquelle **tout ou** une **partie** d'une **application** est **engendrée** à partir de **modèles**.

Qu’est ce qu’un modèle

- Un modèle est une **abstraction**, une **simplification** d’un système qui est suffisante pour **comprendre** le système modélisé et répondre aux questions que l’on se pose sur lui.

- Un système peut être décrit par **différents modèles** liés les uns aux autres.

Modélisation orientée objets

- Le but de la **modélisation objet** est de **décrire les objets**.
- La **description d'un objet** est une **abstraction** ayant des **limites claires** et un **sens précis** dans le contexte du problème étudié.
- Un **objet possède** une **identité** et peut être distingué des autres.

Modélisation orientée objets

❖ L'approche objet : s'inspirer du monde réel.

❖ Un ensemble **d'Objets, communiquant entre eux,** possédant **:**

- des **propriétés**
- des **comportements**

Un **Objet =**
- des **DONNÉES (propriétés)**
- des **MÉTHODES manipulant ces données (comportements)**
- des échanges de **MESSAGES** avec d'autres objets.

Un Objet peut correspondre à :

- Un **objet concret** du monde réel, ayant une réalité **physique**
(une personne, une voiture, un outil, . . .)
- Un **concept abstrait**
(un compte bancaire, . . .)

Modélisation orientée objets

Objet et classe

Objet

Un **objet** représente une **entité** du monde **réel** ou **virtuel** qui se caractérise par un ensemble de **propriétés** (attributs), des **états** significatifs et un **comportement**.

- L'**état** d'un objet correspond aux **valeurs de tous ses attributs à un instant donné.**

- **Les propriétés** sont définies **dans** la **classe** d'appartenance de l'objet.
- **Le comportement** d'un objet est caractérisé par **l'ensemble des opérations** qu'**il** peut **exécuter** en réaction aux messages provenant des autres objets.

- **Les opérations** sont définies **dans** la **classe** d'appartenance de l'objet.

Modélisation orientée objets

Exemple: Considérons l'employé Ahmed, n°30, embauché en tant qu'ingénieur travaillant sur le site N.

- *Cet objet est **caractérisé** par la **liste** de ses **attributs** et son **état** est représenté par les **valeurs** de ses attributs :*
  - n°_employé : **30**,
  - nom : **Ahmed**,
  - qualification : **ingénieur**,
  - lieu_de_travail : **site N**.

***Son comportement** est **caractérisé** par les **opérations qu'il peut exécuter**.*

- entrer dans l'organisme,
- changer de qualification,
- changer de lieu de travail,
- sortir de l'organisme.

Modélisation orientée objets

Classe

Une **classe est l'abstraction d'un ensemble d'objets qui possèdent une structure identique** (liste des attributs) et un même comportement (liste des opérations).

- Un **objet** est une **instance** d'une et une **seule classe.**
- En **groupant** les instances en classes, on **décrit** les instances par leurs **propriétés générales**, de manière **abstraite**.
- Dans une **classe**, on ne **décrit qu'une fois** la structure et le **comportement communs** d'un **ensemble d'objets.**
- Une **classe précise** :
  - les **propriétés** (attributs) des **instances**
  - le **comportement** (méthodes) des **instances**

Modélisation orientée objets

Exemple: Considérons la **classe Employé** qui représente l'ensemble des employés d'une entreprise.

- Nom de classe : Employé.

Attributs :

- **numéro**,
- **nom**,
- **qualification**,
- **site_de_travail**.

Opérations :

- engager un employé,
- consulter un employé,
- modifier un employé,
- départ d'un employé.

Modélisation orientée objets

Analyse/Conception "Orienté Objet" d'un Système :

❖ Le **comportement** global du **Système** **repose** sur :

- **l'ensemble** des **objets** identifiés
- les **relations** et les **communications** possibles entre ces objets.

❖ L'aspect **dynamique** du Système correspond à des **envois** de **messages** entre objets qui **déclenchent** des **traitements** divers ...

Système vu par l'approche 'Orienté Objet' :

Ensemble d'objets interagissant entre eux pour réaliser les fonctions du Système

Modélisation orientée objets

Encapsulation et interface

❖ L'approche objet se caractérise par le **regroupement dans une même classe** de la **description** de la structure des **attributs** et de la **description** des **opérations**.

❖ Ce regroupement des deux descriptions porte le nom d'**encapsulation données-traitements.**

❖ **L'encapsulation** (des attributs et des méthodes) permet de **dissimuler** à l'utilisateur d'un objet des détails susceptibles d'évoluer avec le temps, ou ne présentant pas d'intérêt pour l'utilisation externe de l'objet.

❖ **L'ensemble** des **opérations** d'une classe **rendu visible** aux autres classes porte le nom d'**interface.**

Modélisation orientée objets

Encapsulation et interface

Le principe d'Encapsulation permet aux objets de se présenter sous deux vues possibles

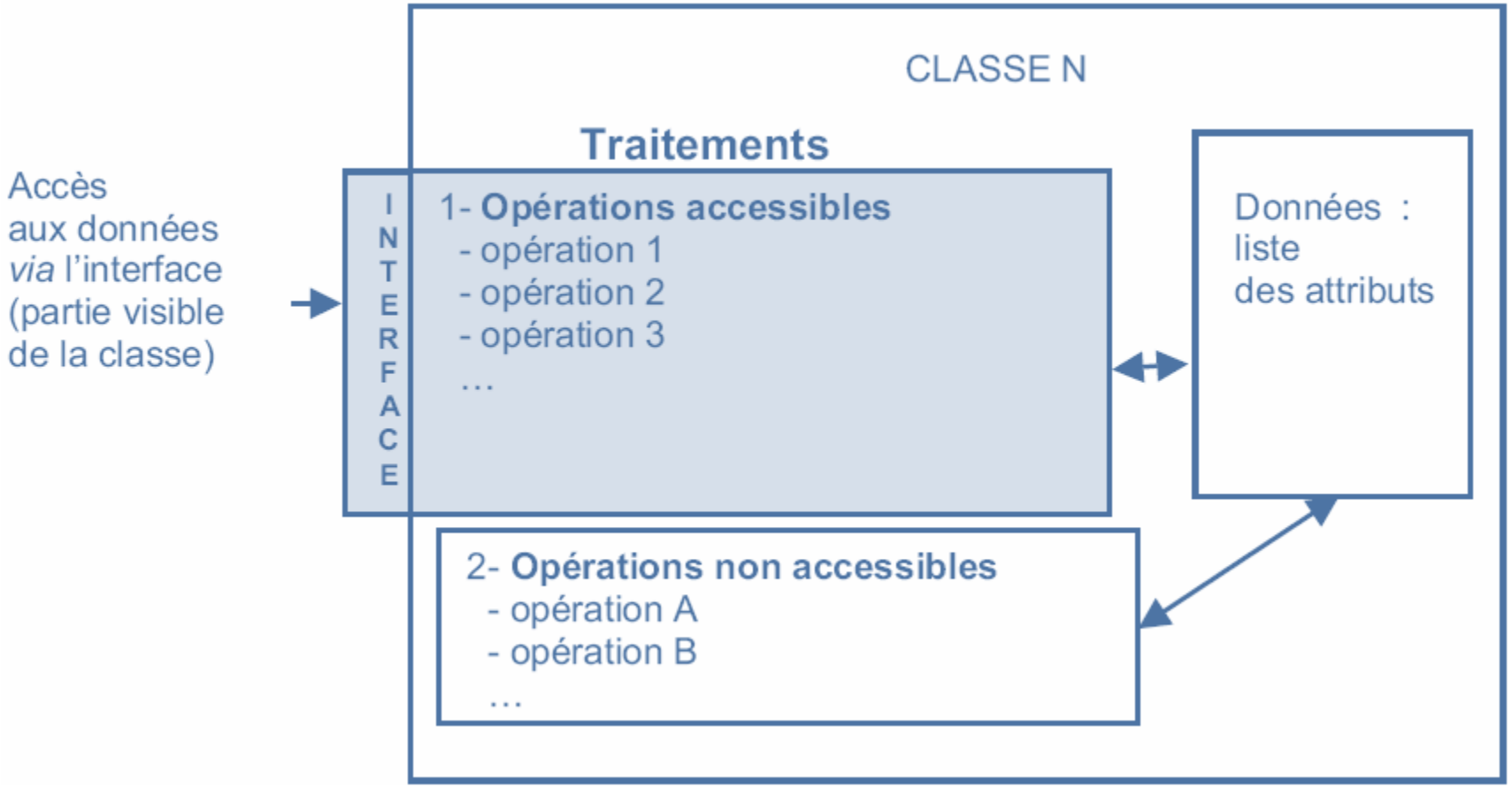
- **la vue externe (celle de l'*utilisateur de l'objet)***

- Définit l'**interface de l'objet**
- Fournit la **liste** des **services accessibles** aux **utilisateurs** de l'objet
- UML/C++ : correspond aux déclarations qualifiées **public.**

- **la vue interne (celle du *concepteur de l'objet)***

- Donne les détails de constitution interne de l'objet (comment il est construit)
- UML/C++ : correspond aux déclarations qualifiées **protected ou private.**

Modélisation orientée objets



Le schéma de principe de l'encapsulation

Modélisation orientée objets

Association entre les classes

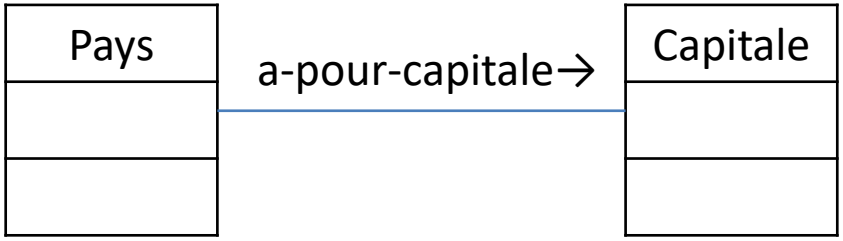
❖ L'**association** **représente une relation entre plusieurs classes.**

❖ Elle correspond à **l'abstraction** des **liens** qui existent **entre** les **objets** dans le monde réel.

Modélisation orientée objets

Association entre les classes

Les associations peuvent être nommées :
"a-pour-capitale" nomme l'association dans le sens "Pays" vers "Capitale"

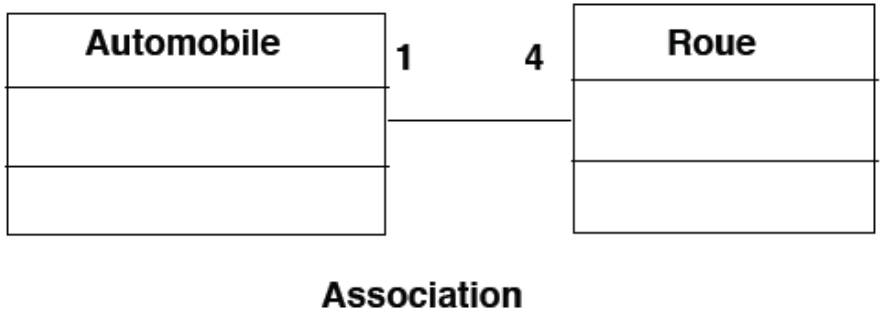


Modélisation orientée objets

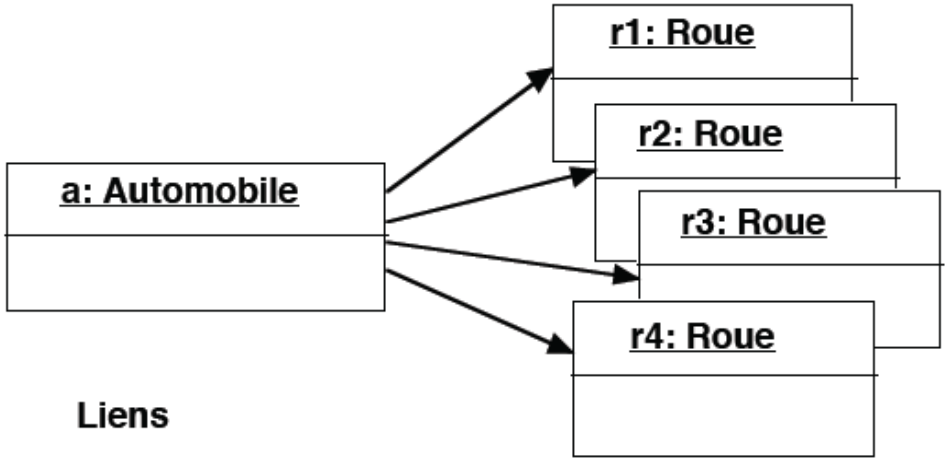
Association entre les classes

La multiplicité (ou **cardinalités**) **d'une association** exprime **le nombre de liens entre les instances** de chaque classe de l'association.

Exemple: dans l'association "Automobile" à "Roue", une instance d'Automobile est liée à 4 instances de Roues, et, une instance de Roue est liée à une seule instance d'Automobile



Association

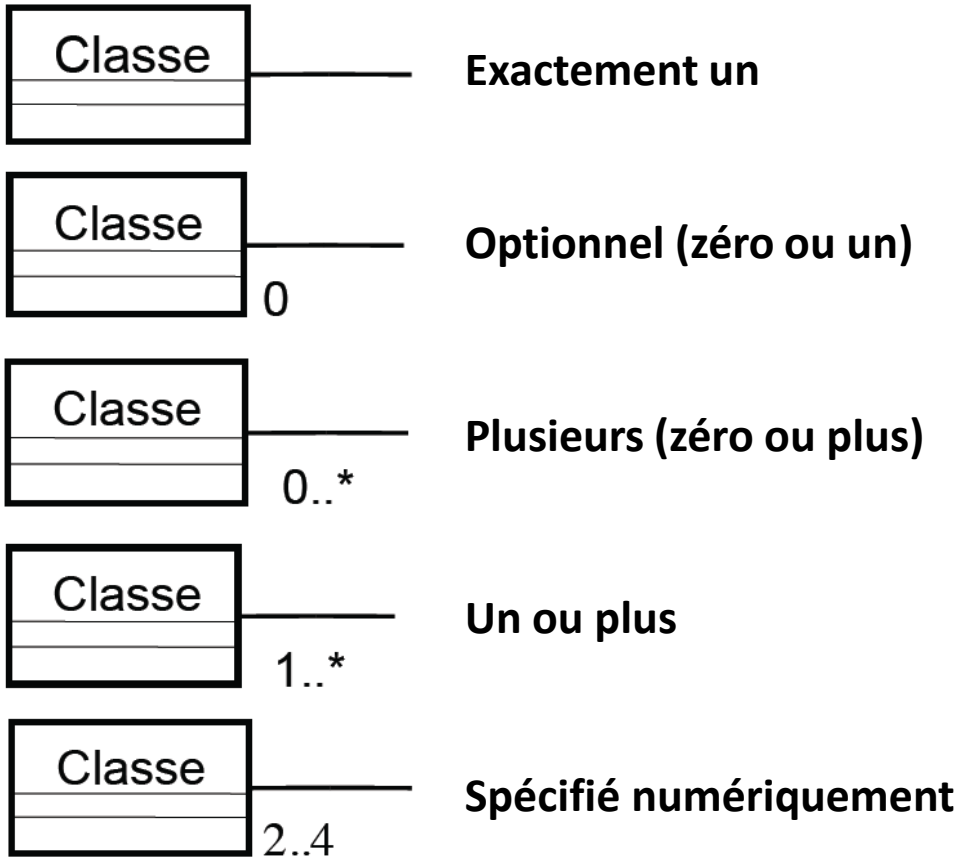


Liens

Modélisation orientée objets

Association entre les classes

La multiplicité d'une association

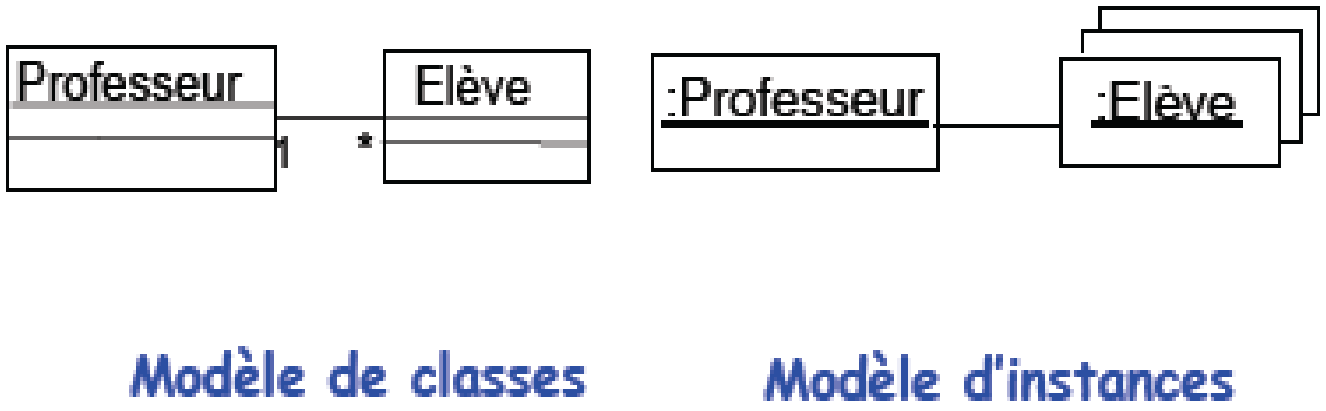


Modélisation orientée objets

Association entre les classes

Liens et Associations

Les **liens correspondants** à des **relations** de cardinalité **N** entre les classes sont notés :



Modélisation orientée objets

Agrégation entre les classes

L'agrégation est une forme particulière d'association entre plusieurs classes**.**

Elle exprime le fait qu'une classe est **composée** d'une ou plusieurs autres classes.

L'agrégation indique que **des instances** d'une classe sont **contenues** (ou agrégées) dans **une instance** d'une autre classe.

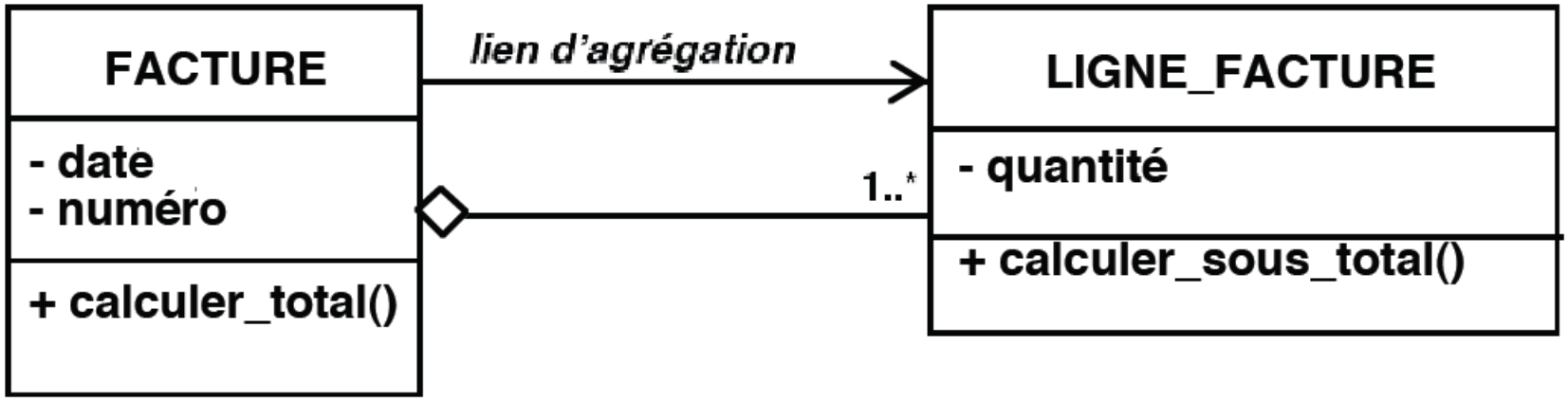
Une agrégation est une relation "composé-composant" ou "partie-de" Un des objets participant à l'agrégation est un composé, un assemblage de composants ou de parties

Exemple, une facture est un composé de lignes factures

Modélisation orientée objets

Agrégation entre les classes

L'agrégation est orientée du composé vers le composant:



Modélisation orientée objets

Héritage

❖ **L'héritage** est un concept OO qui consiste à **spécialiser** des classes en **réutilisant** les **attributs** et le **comportement** d'une **autre** classe.

- Vocabulaire : Une classe (*sous-classe, classe fille, classe dérivée) peut être dérivée d'une autre classe (super-classe, classe mère).*

▪ La classe B **hérite** de la classe A.

- C-à-d que *B est une **spécialisation** de la classe A.*
- La classe B hérite des méthodes et propriétés de la classe A.

❖ On peut y ajouter des propriétés et méthodes et redéfinir des méthodes.

Modélisation orientée objets

Généralisation et spécialisation de classe

- La **généralisation** de classes consiste à **factoriser** dans une classe, appelée **superclasse,** les attributs et/ou opérations des classes considérées.

- Appliquée à l'ensemble des classes, elle permet de réaliser une **hiérarchie** des classes.

Modélisation orientée objets

Généralisation et spécialisation de classe

❖ La **spécialisation** représente la démarche **inverse** de la **généralisation** puisqu'elle consiste à **créer** à **partir** d'une **classe**, **plusieurs** classes **spécialisées**.

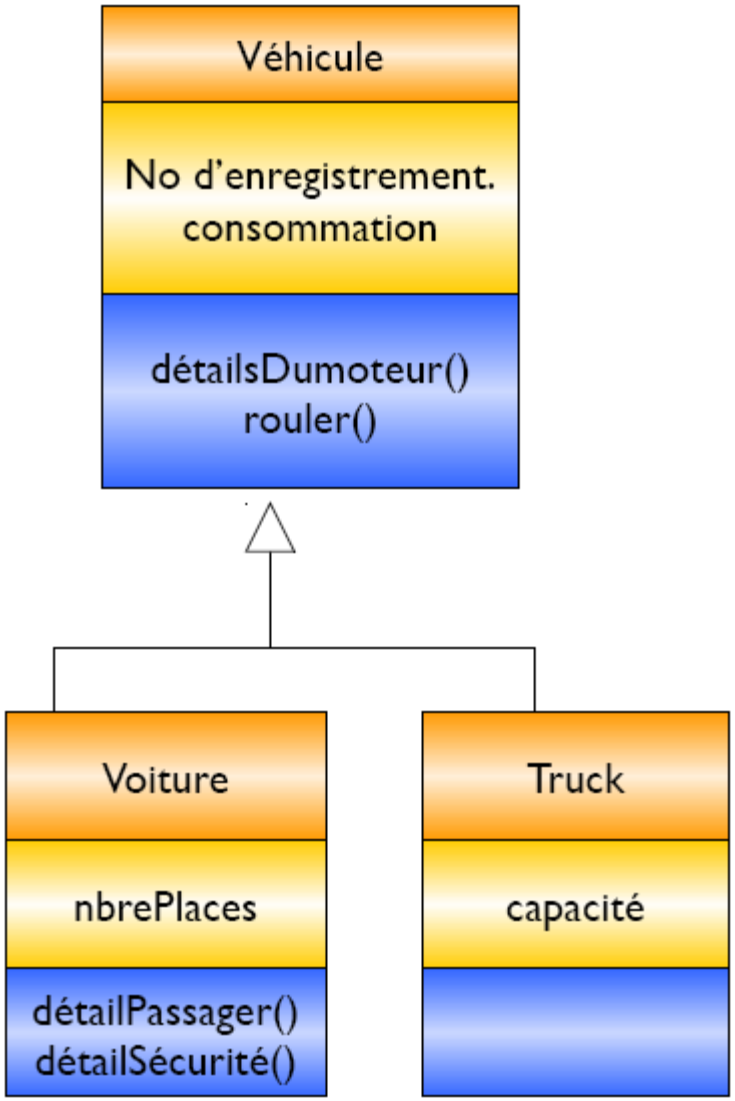
❖ Chaque nouvelle classe créée est dite **spécialisée** puisqu'elle comporte **en plus** des attributs ou opérations de la super-classe ***des attributs ou opérations qui lui sont propres.***

➢ Une classe spécialisée porte aussi le nom de **sous-classe.**

✓ La **généralisation-spécialisation** est un des mécanismes les plus **importants** de l'approche objet **qui facilite la réutilisation des classes**.

Modélisation orientée objets

Généralisation et spécialisation de classe

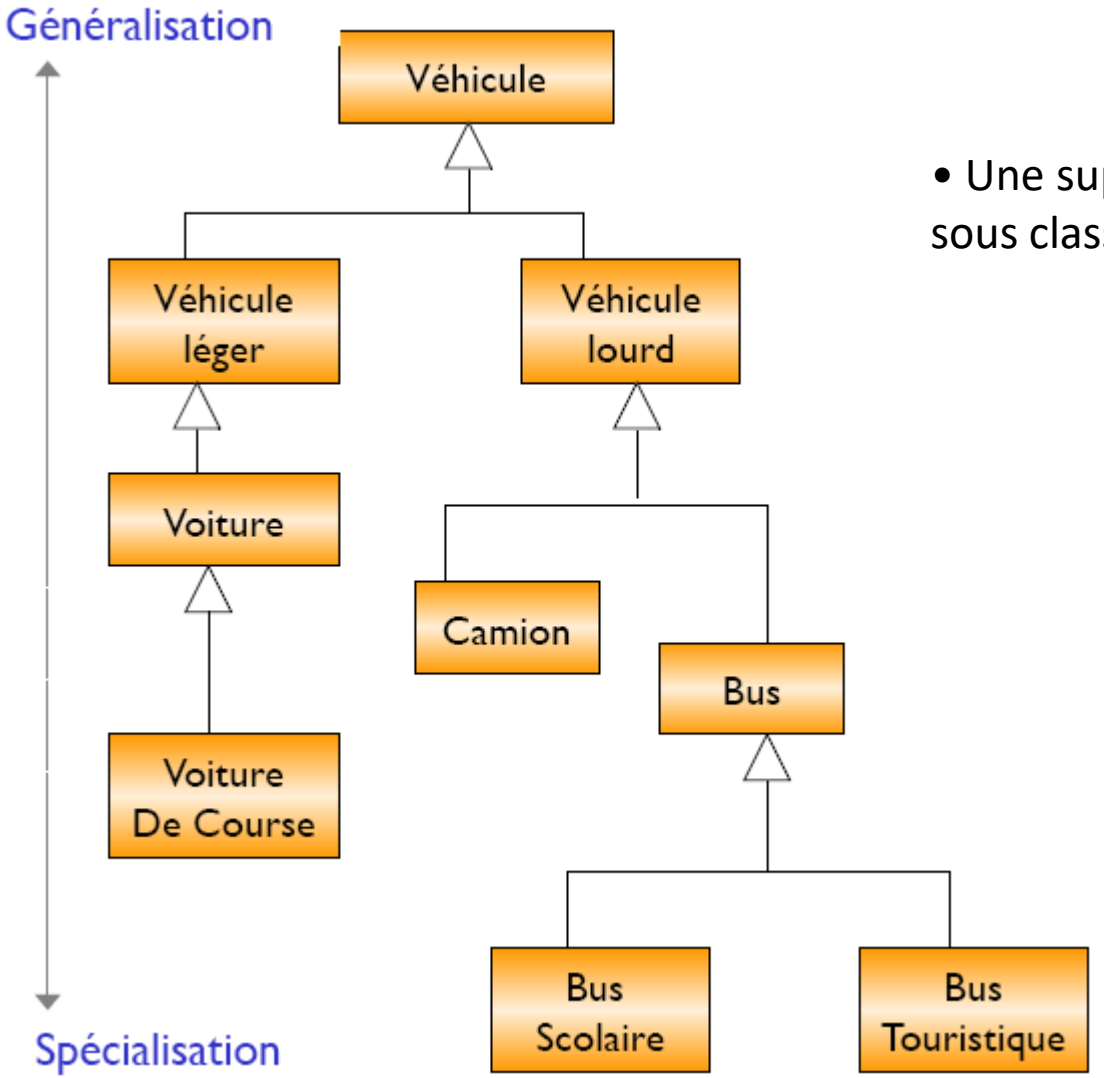


- **Super classe**
  – reprend les attributs et les méthodes communes aux sous Classes

- **Sous classe**
  – héritent de tout ce qui constitue la super classe
  – peut ajouter de nouveaux attributs
  – peut ajouter ou redéfinir des méthodes

Modélisation orientée objets

Héritage à plusieurs étages

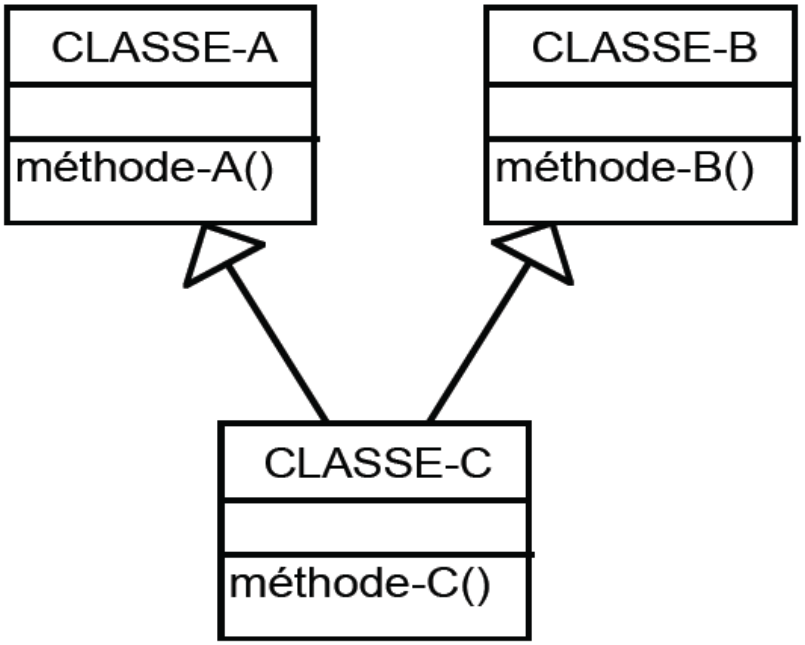


- Une super classe peut à son tour devenir une sous classe …

Modélisation orientée objets

Héritage multiple

C'est la capacité, pour une classe dérivée, **d'hériter** de **plusieurs** classes de base;



- La classe CLASSE-C hérite des méthodes méthode-A() et méthode-B() des CLASSE-A et CLASSE-B

Modélisation orientée objets

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 super-classe peut s'exécuter de manière différente selon la sous-classe où elle est héritée.

Modélisation orientée objets

Exemple

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

- **Nom de classe : Employé.**

Attributs :

– numéro,

– nom,

– salaire de base.

– Opérations : calculSalaire( ).

- **Nom de la sous-classe : Cadre.**

Attributs : niveau d'encadrement.

– Opérations : calculSalaire( ).

- **Nom de la sous-classe : NonCadre.**

Attributs : niveau de spécialisation.

– Opérations : calculSalaire( ).

Modélisation orientée objets

Persistance

La **persistance est la propriété donnée à un objet de continuer à exister après la fin** de l'exécution du programme qui l'a créé.

C'est la possibilité de stocker les valeurs des attributs d'un objet sur un support externe pour pouvoir les recharger ultérieurement

- les objets deviennent immortels !

Modélisation orientée objets

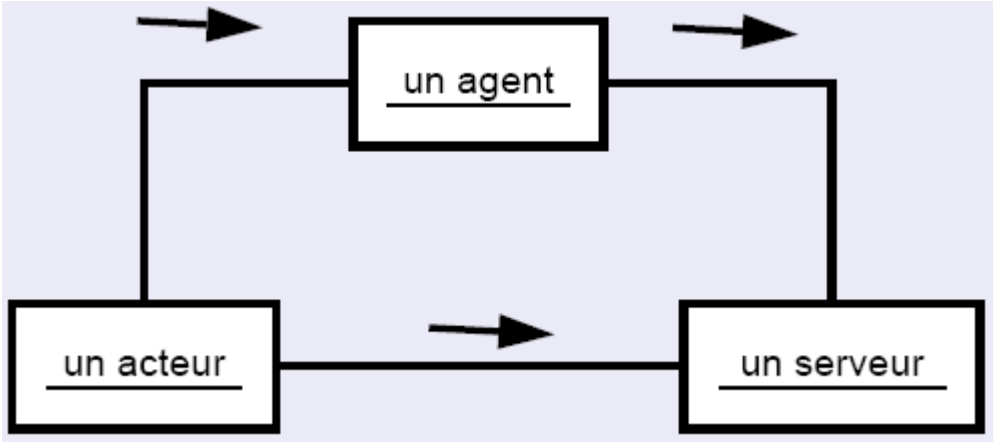
Objets et Messages

Dans les échanges des messages entre objets, on distingue :

Acteur : objet **actif**, à **l'origine** de l'envoi de messages

Serveur : objet **passif** **destinataire** des messages (jamais à l'origine d'un échange)

Agent : objet à la fois Acteur et Serveur peut interagir avec les autres objets, de sa propre initiative ou suite à un message.

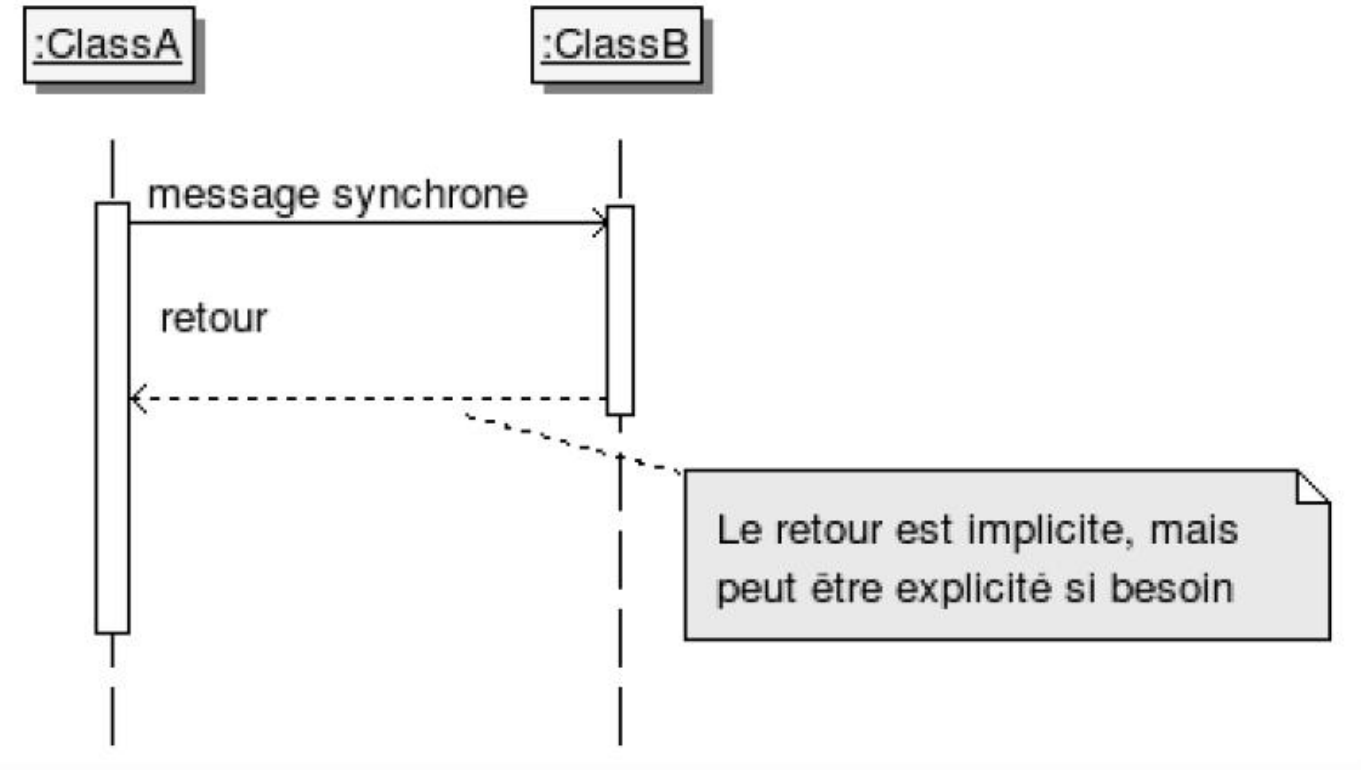


Modélisation orientée objets

Message **Synchrone**

Lorsqu'un objet A envoie un **message synchrone à l'objet B :**

- **A attend que B accepte** de prendre le message
- Une fois le message accepté par **B**, **A** est bloqué tant que **B** n'a pas fini le traitement correspondant.

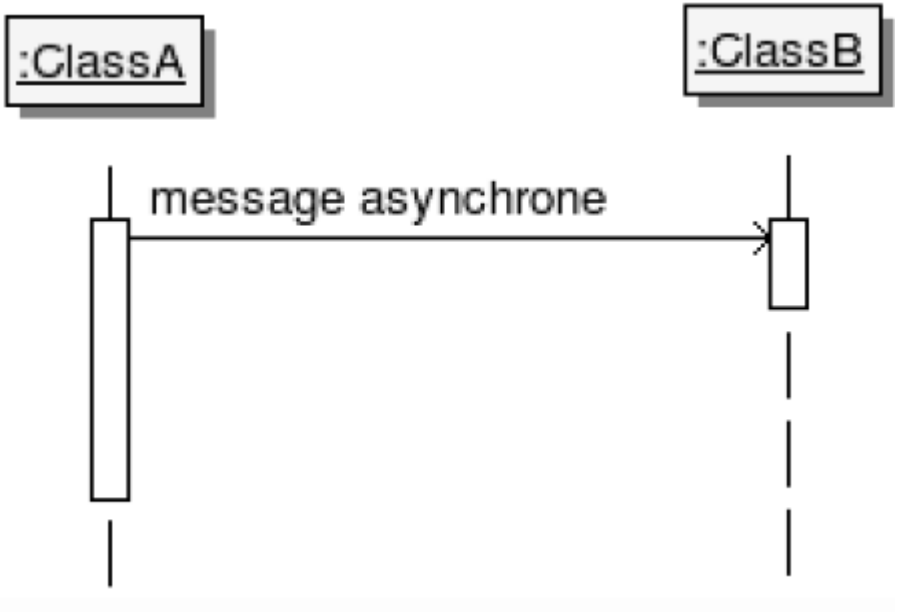


Modélisation orientée objets

Message **Asynchrone**

Un objet A envoie un **message asynchrone à l'objet B :**

- **A n'attend pas que B accepte** le message
- **A** n'attend pas que **B** ait fini le traitement correspondant
- **A** passe directement à autre chose (sans savoir si **B** a accepté le message, ni si **B** a fini le traitement correspondant).



Modélisation orientée objets

Récapitulons :

- ✓ Un **Objet possède une identité**
- ✓ Un ensemble d'**attributs** (données) **caractérise l'état** de l'objet
- ✓ Un ensemble d'**opérations** (méthodes) définit le **comportement** de l'objet
- ✓ L'**encapsulation** permet aux objets de fournir une **interface** aux autres objets, tout en dissimulant ses détails internes
- ✓ Un objet peut être en **relation** avec d'autre(s) objet(s)
- ✓ Un objet peut échanger des **messages** avec d'autre(s) objet(s)
- ✓ Un objet possède un **cycle de vie (construction . . . destruction)**
- ✓ Un objet peut être **persistant.**