Conception Orientée Objet
UML 2

SMI-5

Année 2016 - 2017

Plan

- Notions sur le Génie Logiciel (GL)
- Concepts de l'approche objet
- UML
  - **Diagrammes statiques (structurels)**
  - Diagrammes comportementaux

References

- UML 2 Analyse et conception; Joseph Gabay, David Gabay.
- UML 2 de l'apprentissage à la pratique. Laurent AUDIBERT.
- Développement de logiciel à objets avec UML. Jean-Marc Jézéquel, Noël Plouzeau, Yves Le Traon.
- UML, Introduction au génie logiciel et à la modélisation. Delphine Longuet.
- Méthodes de Conception Orientés Objet (MCOO). Verlain FOUNDIKOU.

UML - Règles générales

- Les principaux éléments généraux d'UML que nous présentons:
  - le stéréotype,
  - la valeur marquée,
  - la note,
  - La contrainte (~ OCL),
  - la relation de dépendance.

UML - RÈGLES GÉNÉRALES

STÉRÉOTYPAGE

- Un **stéréotype (الصورة النمطية) constitue un moyen de classer les éléments de la modélisation.**

- Permet aux concepteurs d'étendre le vocabulaire de l'UML et créer des éléments du nouveau modèle (W):
  - Dérivés de ceux qui existent déjà,
  - Ont des propriétés spécifiques qui sont adaptées à un domaine de problème particulier ou autre usage spécialisé

- Le nom d'un stéréotype nouvellement défini ne doit pas être déjà attribué à un autre élément du métamodèle.

- Un certain nombre de stéréotypes sont déjà définis dans UML,

- D'autres valeurs de stéréotypes pourraient être ajoutées:
  - À l'évolution générale d'UML,
  - À la prise en compte de situations particulières propres aux entreprises.

UML - Règles générales

Stéréotypage (suite)

- Peut s'appliquer à n'importe quel concept d'UML.
- Nous nous intéresserons à quelques uns que nous présenterons au niveau des diagrammes.
- Dans le diagramme de classe, le stéréotype permet de considérer de nouveaux types de classe.
- **Formalisme et exemple**
  - Le nom du stéréotype est indiqué entre guillemets.
  - L'exemple ci-dessous montre une classe Client stéréotypée comme « acteur ».

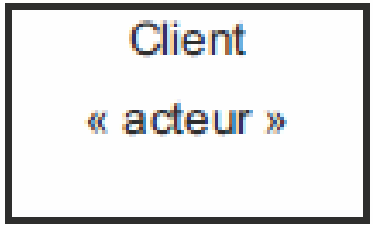


Figure. Exemple d'une classe stéréotypée

UML - Règles générales

Valeur marquée

- UML permet d'indiquer des valeurs particulières au niveau des éléments de modélisation et en particulier pour les attributs de classe.

- **Formalisme et exemple**
  - La valeur marquée est mise entre accolades avec indication du nom et de la valeur :

    {persistance : string} si l'on veut ajouter ce type d'attribut dans une classe.

Profil d'utilisation (modèle d'utilisation)

- Permet de donner la possibilité de spécialiser chaque application d'UML à son propre contexte,

- Caractérisé principalement par la liste des stéréotypes, la liste des valeurs marquées et les contraintes spécifiées pour un projet donné.

UML - Règles générales

Note

- Une **note correspond à un commentaire explicatif d'un élément d'UML.**
- Symbole graphique qui contient des informations.
- En général, ces informations sont sous forme textuelle.

- **Formalisme et exemple**

  La figure ci-dessous montre le formalisme et un exemple de la représentation d'une note.

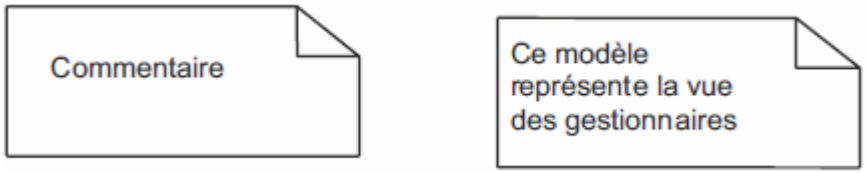


Figure. Formalisme et exemple d'utilisation d'une note

UML - Règles générales

Note (suite)

- L'utilisation d'une note permet de présenter des commentaires, des contraintes, le contenu de méthodes ou des valeurs marquées
- Un commentaire fournit:
  - des explications utiles,
  - des observations de diverses natures
  - des renvois vers des documents de description plus complets. (Par défaut, ils ne véhiculent pas de contenu sémantique).

Contrainte

- Une contrainte est une note ayant une valeur sémantique particulière pour un élément de la modélisation.
- S'écrit entre accolades {}
- Dans le cas où la contrainte concerne deux classes ou plus, celle-ci s'inscrit à l'intérieur d'une note.

UML - Règles générales

Contrainte (suite)

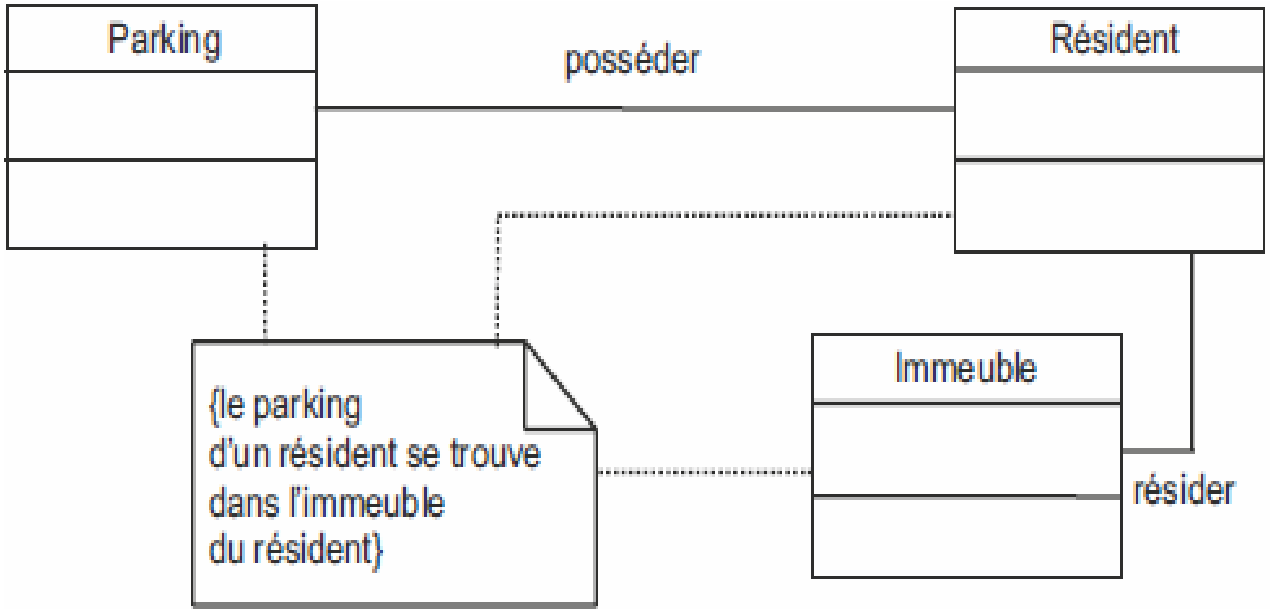
- **Formalisme et exemple**
  - 1ère forme d'écriture d'une contrainte :
    {ceci est une contrainte}
  - 2ème forme d'écriture : à l'intérieur d'une note

- Dans UML, un langage spécifique d'expression de contraintes est disponible ;

- C'est le langage OCL (*Object Contraint Language*).

UML - RÈGLES GÉNÉRALES

CONTRAINTE (SUITE)



Exemple d'utilisation d'une contrainte
(sans représentation des multiplicités)

Diagrammes structurels (statiques)

Les diagrammes structurels (statiques)

Diagramme de classe (DCL) et diagramme d'objet (DOB) [1]

- Le diagramme de classe constitue l'un des pivots essentiels de la modélisation avec UML.
- Il permet de donner la représentation statique du système à développer.
- Cette représentation est centrée sur les concepts de classe et d'association.
- Chaque classe se décrit par les données et les traitements dont elle est responsable pour elle-même et vis-à-vis des autres classes.

Les diagrammes structurels (statiques)

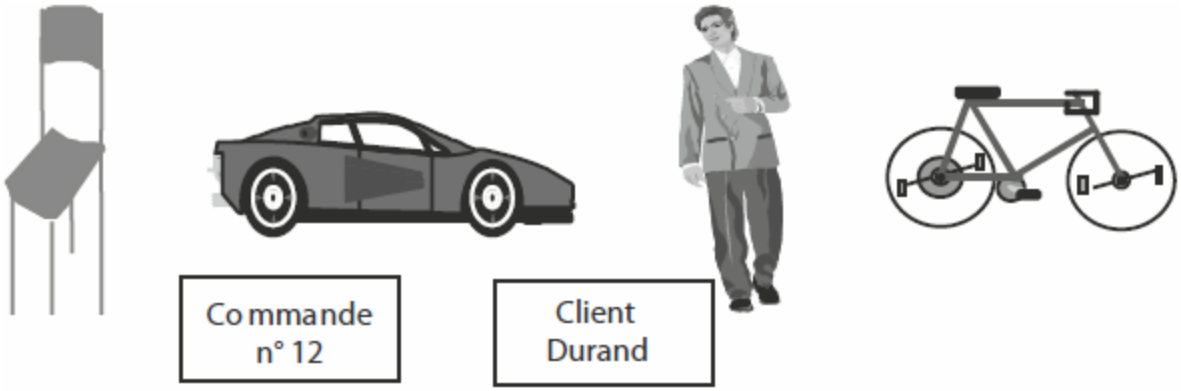
Diagramme de classe (DCL) et diagramme d'objet (DOB) [2]

- Les traitements sont matérialisés par des opérations.
- Le détail des traitements n'est pas représenté directement dans le diagramme de classe ;
- Seul l'algorithme général et le pseudo-code correspondant peuvent être associés à la modélisation.
- La description du diagramme de classe est fondée sur :
  - le concept d'objet,
  - le concept de classe comprenant les attributs et les opérations,
  - les différents types d'association entre classes.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

OBJET

- Un objet est un concept, une abstraction ou une chose qui a un sens dans le contexte du système à modéliser.
- Chaque objet a une identité
- Peut être distingué des autres, sans considérer a priori les valeurs de ses propriétés.



Exemples d'objets physiques et d'objets de gestion

Diagramme de classe (DCL) & diagramme d'objet (DOB)

Classe

- Une classe décrit un groupe d'objets ayant:
  - les mêmes propriétés (attributs),
  - un même comportement (opérations),
  - une sémantique commune (domaine de définition).

- Un objet est une instance d'une classe.

- La classe représente l'abstraction de ses objets.

- Au niveau de l'implémentation, (au cours de l'exécution d'un programme), à l'identificateur d'un objet correspond une adresse mémoire.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

CLASSE : FORMALISME GÉNÉRAL

- Une classe se représente à l'aide d'un rectangle comportant plusieurs compartiments.

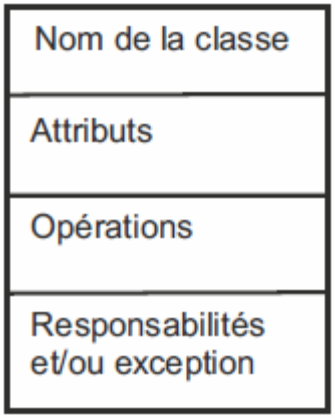
- Les trois compartiments de base sont :
  - la désignation de la classe,
  - la description des attributs,
  - la description des opérations.

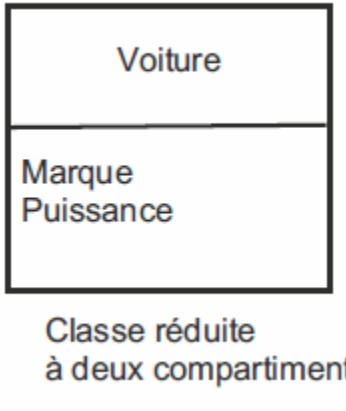
- Deux autres compartiments peuvent être aussi indiqués :
  - la description des responsabilités de la classe,
  - la description des exceptions traitées par la classe.

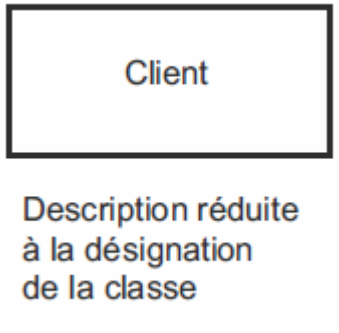
- Selon les objectifs poursuivis par le modélisateur , il est possible de manipuler les classes en limitant le niveau de description à un nombre réduit de compartiments.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

CLASSE : FORMALISME GÉNÉRAL ET EXEMPLES





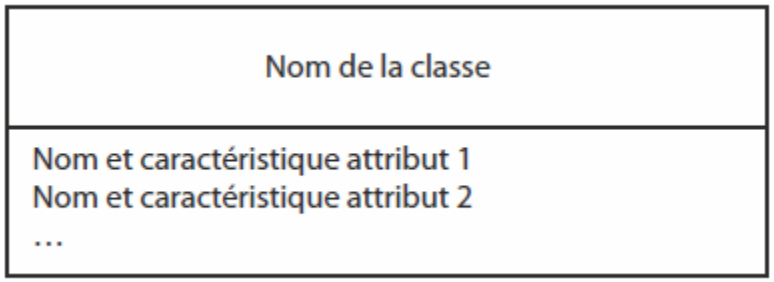


Formalisme général d'une classe et exemples

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

ATTRIBUT :

- Un attribut est une propriété élémentaire d'une classe.
- Pour chaque objet d'une classe, l'attribut prend une valeur (sauf cas d'attributs multivalués).
- Formalisme



- Exemple:

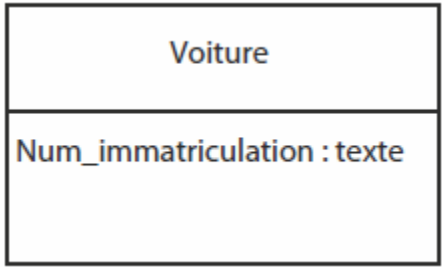


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

ATTRIBUT : CARACTÉRISTIQUES

- Le nom de la classe peut être qualifié par un « stéréotype ».

- La description complète des attributs d'une classe comporte un certain nombre de caractéristiques qui doivent respecter le formalisme suivant :

Visibilité/Nom attribut : type [= valeur initiale {propriétés}]

  - *Visibilité : cf plus loin.*
  - *Nom d'attribut : nom unique dans sa classe.*
  - *Type : type primitif (entier, chaîne de caractères…) dépendant des types* disponibles dans le langage d'implémentation ou type classe matérialisant un lien avec une autre classe.
  - *Valeur initiale (par défaut): valeur facultative donnée à l'initialisation d'un objet de la* classe.
  - *{propriétés} : valeurs marquées facultatives (ex : «interdit» pour mise à jour* interdite).

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

ATTRIBUT : CARACTÉRISTIQUES

- Un attribut peut avoir des valeurs multiples.

- Cette caractéristique est indiquée après le nom de l'attribut
  Ex: prénom [3] pour une personne qui peut avoir trois prénoms).

- Un attribut dérivé est un attribut dont la valeur peut être calculée à partir d'autres attributs de la classe.

  Se note « /nom de l'attribut dérivé ».
  Exemple d'attribut dérivé:

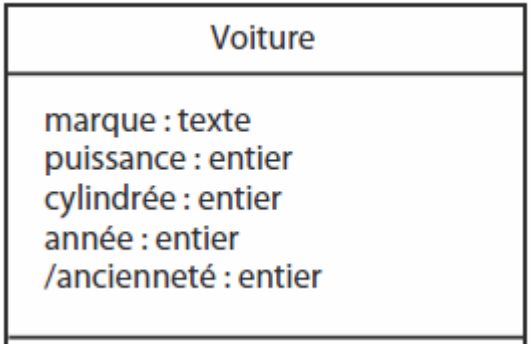
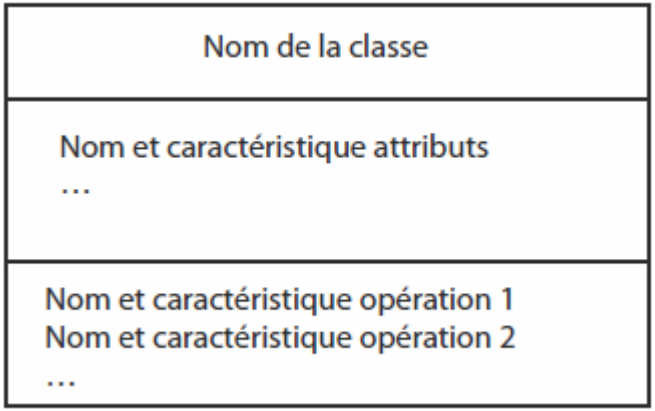


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

OPÉRATION

- Une opération est une fonction applicable aux objets d'une classe.
- Permet de décrire le comportement d'un objet.
- Une méthode est l'implémentation d'une opération.
- **Formalisme:**
  - Chaque opération est désignée soit seulement par son nom soit par son nom, sa liste de paramètres et son type de résultat.
  - La signature d'une méthode correspond au nom de la méthode et la liste des paramètres en entrée.



- Exemple:

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

OPÉRATION : CARACTÉRISTIQUES

- La description complète des opérations d'une classe comporte un certain nombre de caractéristiques qui doivent respecter le formalisme suivant :

Visibilité Nom d'opération (paramètres) [:[type résultat] {propriétés}]

- *Visibilité : cf plus loin*
- *Nom d'opération : utiliser un verbe représentant l'action à réaliser.*
- *Paramètres : liste de paramètres (chaque paramètre peut être décrit, en plus* de son nom, par son type et sa valeur par défaut).

  L'absence de paramètre est indiquée par ( ).
- *Type résultat : type de (s) valeur(s) retourné(s) dépendant des types disponibles* dans le langage d'implémentation.

  Par défaut, une opération ne retourne pas de valeur, ceci est indiqué par exemple par le mot réservé «void» dans le langage C++ ou Java.
- *{propriétés} : valeurs facultatives applicables (ex. : {query} pour un comportement* sans influence sur l'état du système).

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

EXEMPLE DE REPRÉSENTATION DE CLASSE

- Exemple de représentation d'une classe Voiture

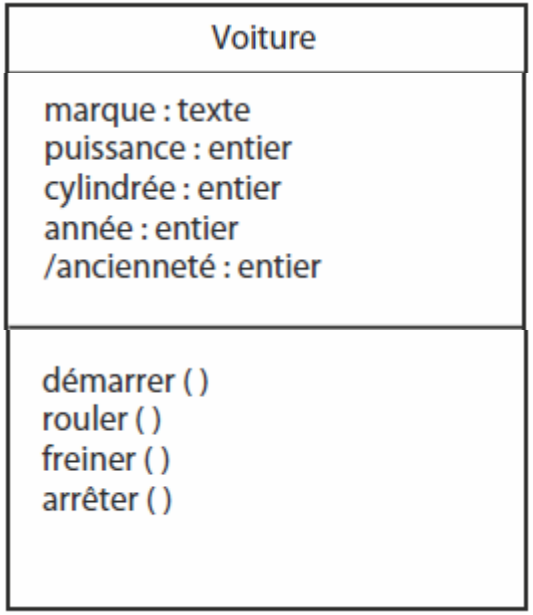
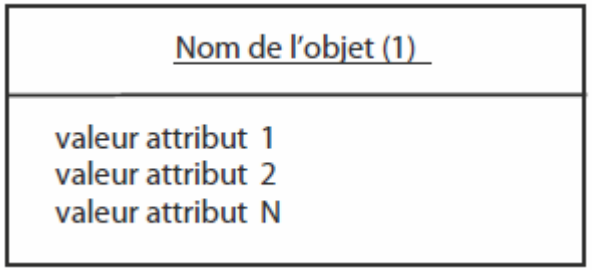
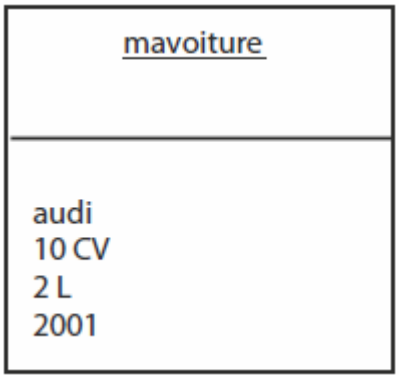


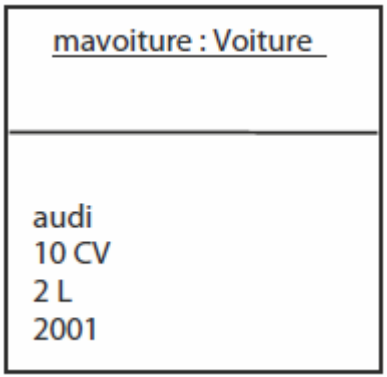
DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

FORMALISME DE REPRÉSENTATION D'OBJET



- (1) Le nom d'un objet peut être désigné sous trois formes :
  - *nom de l'objet, désignation directe et* explicite d'un objet ;
  - *nom de l'objet : nom de la classe, désignation incluant le nom de la classe ;*
  - *:nom de la classe, désignation anonyme d'un objet d'une classe donnée.*
- ***Exemples***





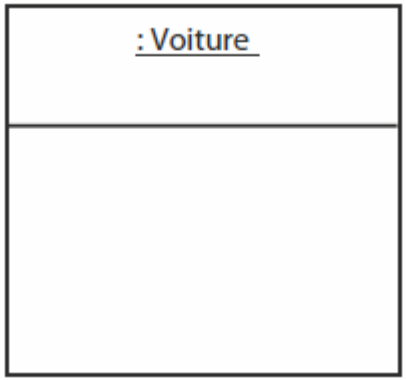


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

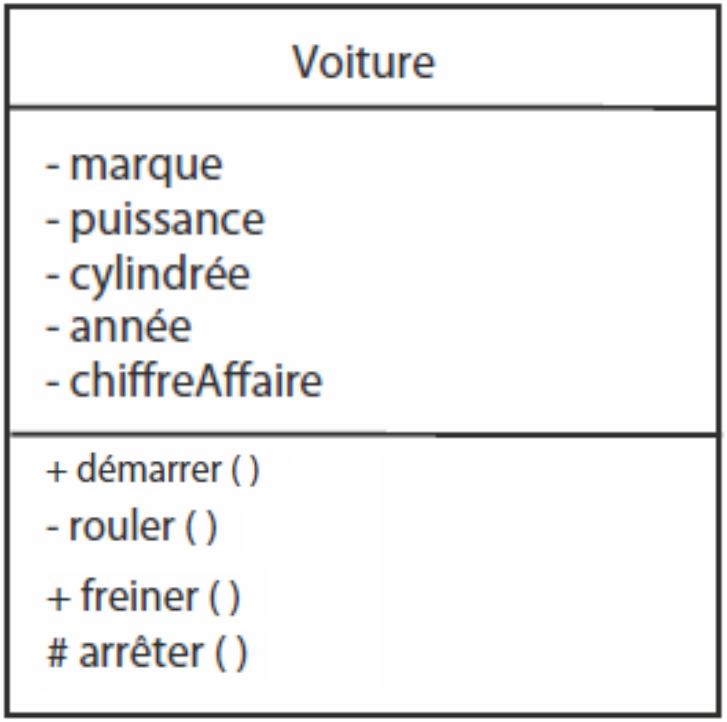
VISIBILITÉ DES ATTRIBUTS ET OPÉRATIONS

- Chaque attribut ou opération d'une classe peut être de type:
  - **Public (+)** : Attribut ou opération visible par tous.
  - **Protégé (#)** : Attribut ou opération visible seulement:
    - à l'intérieur de la classe et
    - pour toutes les sous-classes de la classe.
  - **Privé (-)** : Attribut ou opération seulement visible à l'intérieur de la classe.
  - **Paquetage (~)** : Attribut ou opération ou classe seulement visible à l'intérieur du paquetage où se trouve la classe.

Diagramme de classe (DCL) & diagramme d'objet (DOB)

Visibilité des attributs et opérations

- Exemple d'utilisation des symboles de la visibilité des éléments d'une classe.



- Dans cet exemple, tous les attributs sont déclarés de type privé,
- les opérations « démarrer » et « freiner » sont de type public,
- l'opération « rouler » est de type privé
- l'opération « arrêter » est de type protégé.

Diagramme de classe (DCL) & diagramme d'objet (DOB)

Attribut ou opération de niveau classe

- Un attribut ou une opération peut être défini non pas au niveau des instances d'une classe, mais au niveau de la classe.
- Il s'agit :
  - soit d'un attribut qui est une constante pour toutes les instances d'une classe
  - soit d'une opération d'une classe abstraite (cf plus loin)
  - soit par exemple d'une opération « créer » qui peut être définie au niveau de la classe et applicable à la classe elle-même.

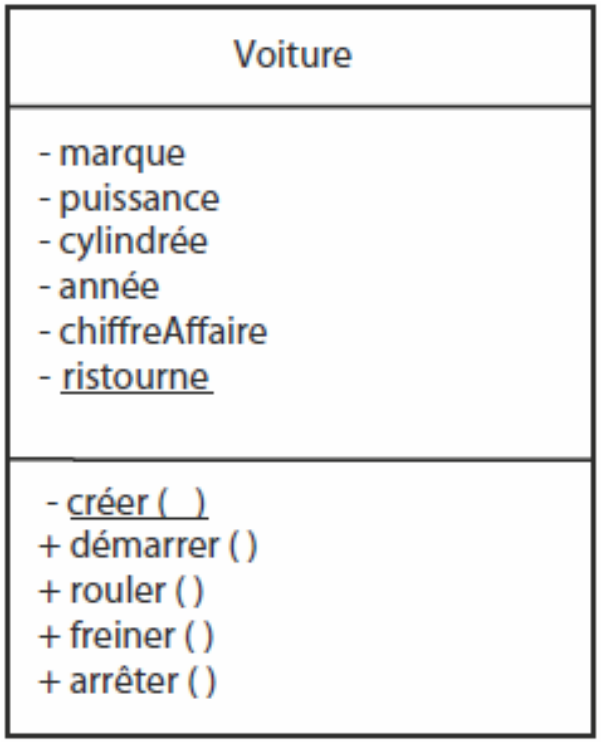
- Formalisme:

  C'est le soulignement de l'attribut ou de l'opération qui caractérise cette propriété.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

ATTRIBUT OU OPÉRATION DE NIVEAU CLASSE

- Exemple:

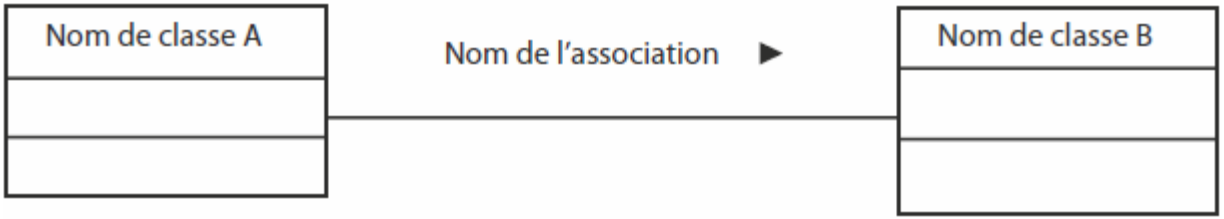


- l'attribut « ristourne » (réduction) est de type classe
- l'opération « créer » est une opération exécutable au niveau de la classe.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

LIEN ET ASSOCIATION

- Un lien est une connexion physique ou conceptuelle entre instances de classes (entre objets).
- Une association décrit un groupe de liens ayant une même structure et une même sémantique.
- Un lien est une instance d'une association.
- Chaque association peut être identifiée par son nom.
- Une association entre classes représente les liens qui existent entre les instances de ces classes.
- Formalisme d'association:



Le symbole (facultatif) indique le sens de lecture de l'association.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

LIEN ET ASSOCIATION

- **Exemple:**

RÔLE DANS UNE ASSOCIATION

- Le rôle tenu par une classe vis-à-vis d'une association peut être précisé sur l'association.
- Exemple:

Diagramme de classe (DCL) & diagramme d'objet (DOB)

Multiplicité

- La multiplicité indique un domaine de valeurs pour préciser le nombre d'instance d'une classe vis-à-vis d'une autre classe pour une association donnée.

- La multiplicité peut aussi être utilisée pour d'autres usages comme par exemple un attribut multivalué.

- Le domaine de valeurs est décrit selon plusieurs formes :
  - Intervalle fermé – Exemple : 3..15
  - Valeurs exactes – Exemple : 3, 5, 8
  - Valeur indéterminée notée * – Exemple : 1..*
    - Dans le cas où l'on utilise seulement *, cela traduit une multiplicité 0..*

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

MULTIPLICITÉ : FORMALISME ET EXEMPLE

- À une instance de A correspond 0 ou 1 instance de B.
- À une instance de B correspond 0 à nombre non déterminé d'instances de A.

- À une instance de A correspond 1 à un nombre non déterminé d'instances de B.
- À une instance de B correspond 2 à 10 instances de A.

- À une instance de A correspond 2 à 4 instances de B.
- À une instance de B correspond 1 ou 3 instances de A.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

NAVIGABILITÉ D'ASSOCIATION

- La navigabilité indique si l'association fonctionne de manière unidirectionnelle ou bidirectionnelle,
- Elle est matérialisée par une ou deux extrémités fléchées.
- La non navigabilité se représente par un « X »

- Situation Possible:

- Navigabilité unidirectionnelle de A vers B.
- Pas de navigabilité de B vers A (non définie explicitement).

- Navigabilité unidirectionnelle de B vers A.
- Navigabilité de A vers B (non définie explicitement).

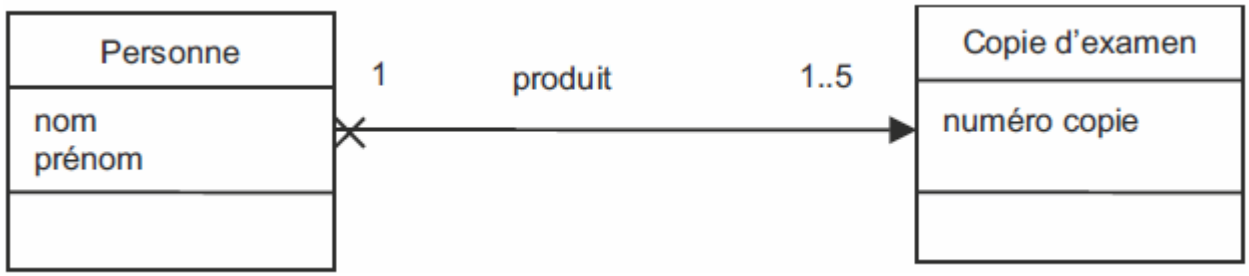
- Navigabilité bidirectionnelle entre A et B. Habituellement représentée sans flèche.

- Par défaut, on admet qu'une navigabilité non définie correspond à une navigabilité implicite.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

NAVIGABILITÉ D'ASSOCIATION

- Exemple:



- à une personne sont associées ses copies d'examen
- mais l'inverse n'est pas possible (retrouver directement l'auteur de la copie d'examen, notamment avant la correction de la copie).

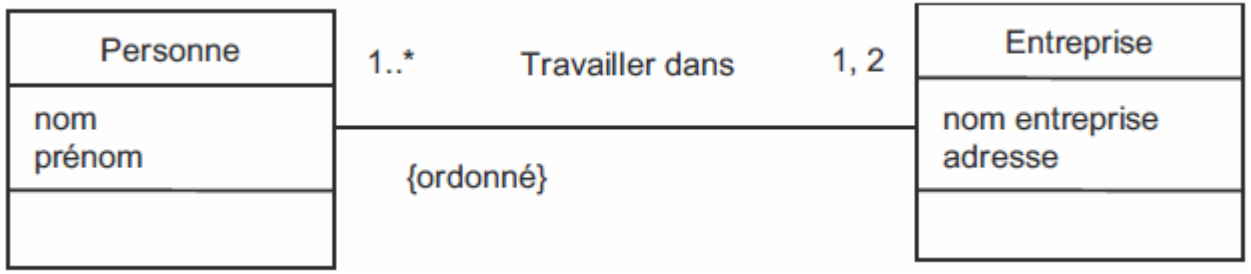
DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

CONTRAINTES

- Propriétés particulières pour préciser la sémantique d'une association.
- Exemple de contraintes: **Ordre de tri**

  Pour une association de multiplicité supérieure à 1, les liens peuvent être :
  - non ordonnés (valeur par défaut),
  - ordonnés ou triés lorsque l'on est au niveau de l'implémentation (tri sur une valeur interne).

- Ex:



- pour une entreprise donnée, les personnes seront enregistrées suivant un ordre qui correspondra à un des attributs de Personne.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

CONTRAINTES

- Exemple de contraintes: **Propriétés de mise à jour de liens**

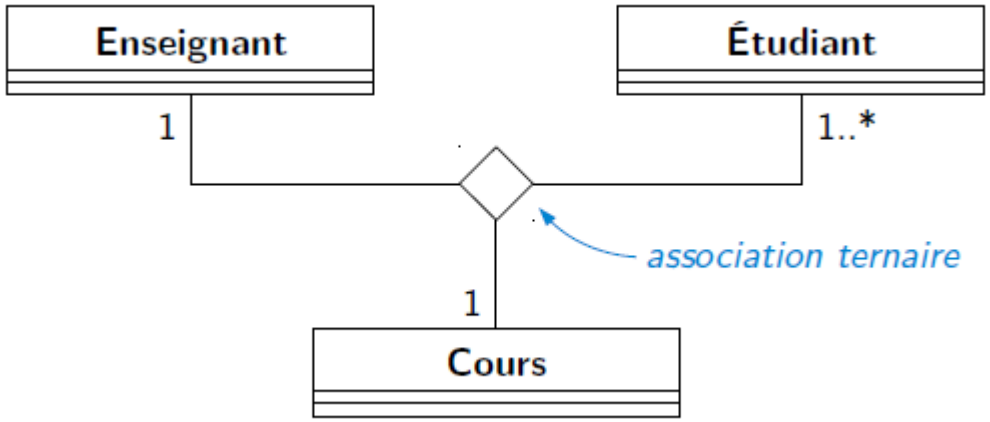
- Il est possible d'indiquer des contraintes particulières relatives aux conditions de mise à jour des liens:
  - {interdit} : interdit l'ajout, la suppression ou la mise à jour des liens.
  - {ajout seul} : n'autorise que l'ajout de liens.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

ASSOCIATION N-AIRE

- Association reliant plus de deux classes
- se représente en utilisant un **losange** permettant de relier toutes les classes concernées.

- Exemple



- Instance d'une association n-aire = lien entre n objets

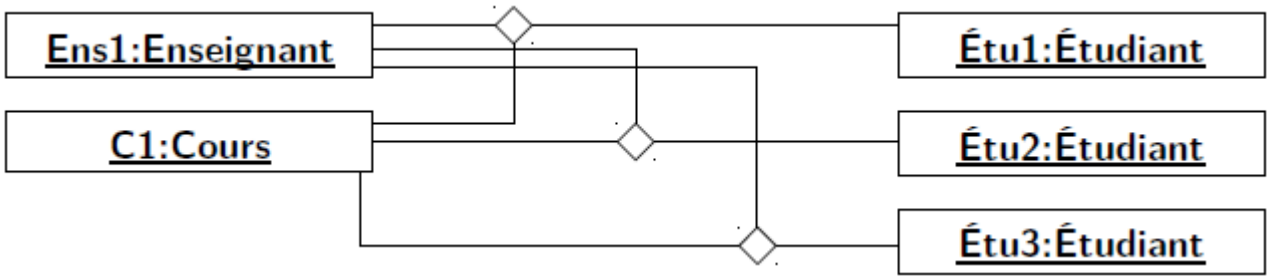


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

ASSOCIATION N-AIRE

- Multiplicités : pour chaque *(n-1)-uplet d'objets, contraint le nombre* d'objets qui lui sont associés

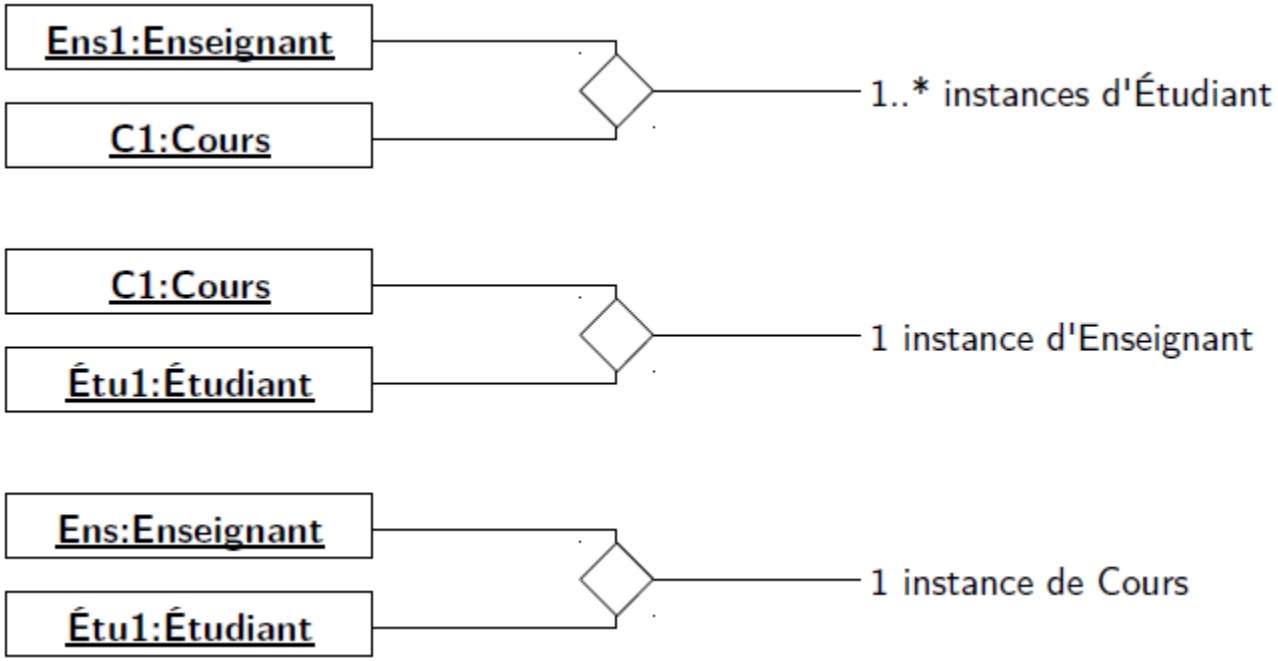


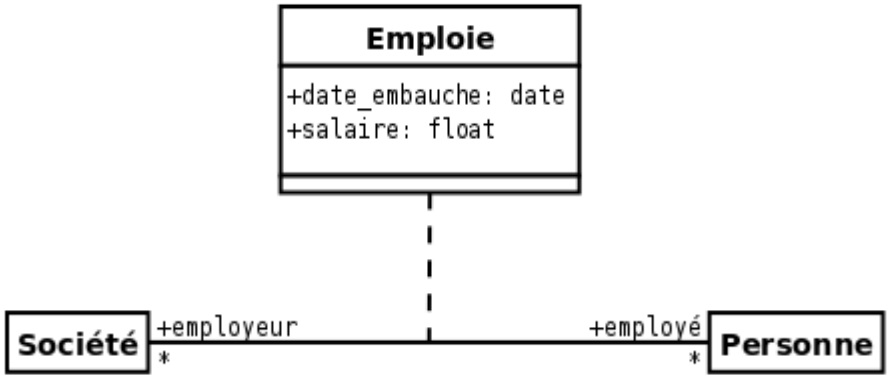
DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

CLASSE-ASSOCIATION

- Une **classe-association permet:**
  - paramétrer une association entre deux classes (ou plus) par une classe
  - **décrire soit des attributs soit des opérations** propres à l'association.

- la classe-association est reliée au lien (ou au losange N>2)par un trait en pointillé.

- Une classe-association peut être reliée à d'autres classes d'un diagramme de classes.
- Exemple:

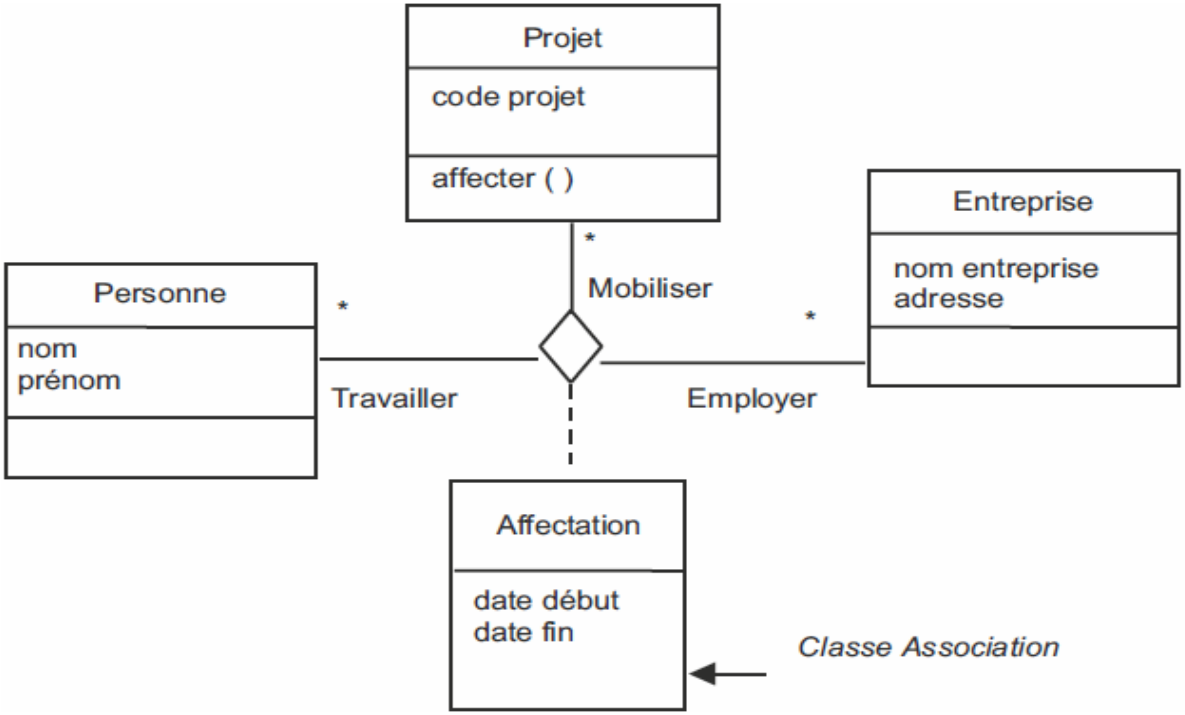


- Instance unique de la classe-association pour chaque lien entre objets

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

ASSOCIATION N-AIRE ET CLASSE-ASSOCIATION

- **Exemple:**



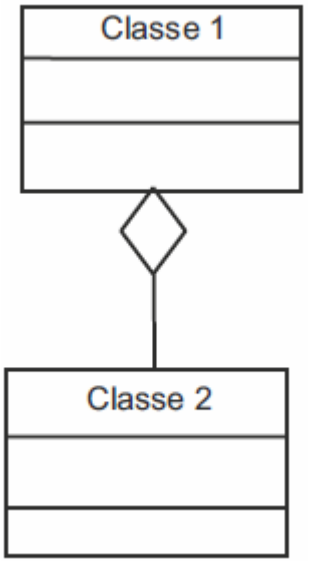
- Association de dimension 3 comprenant une classe-association «Affectation».
- La classe-association 'Affectation' permet de décrire les attributs propres à l'association de dimension 3 représentée.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

AGRÉGATION

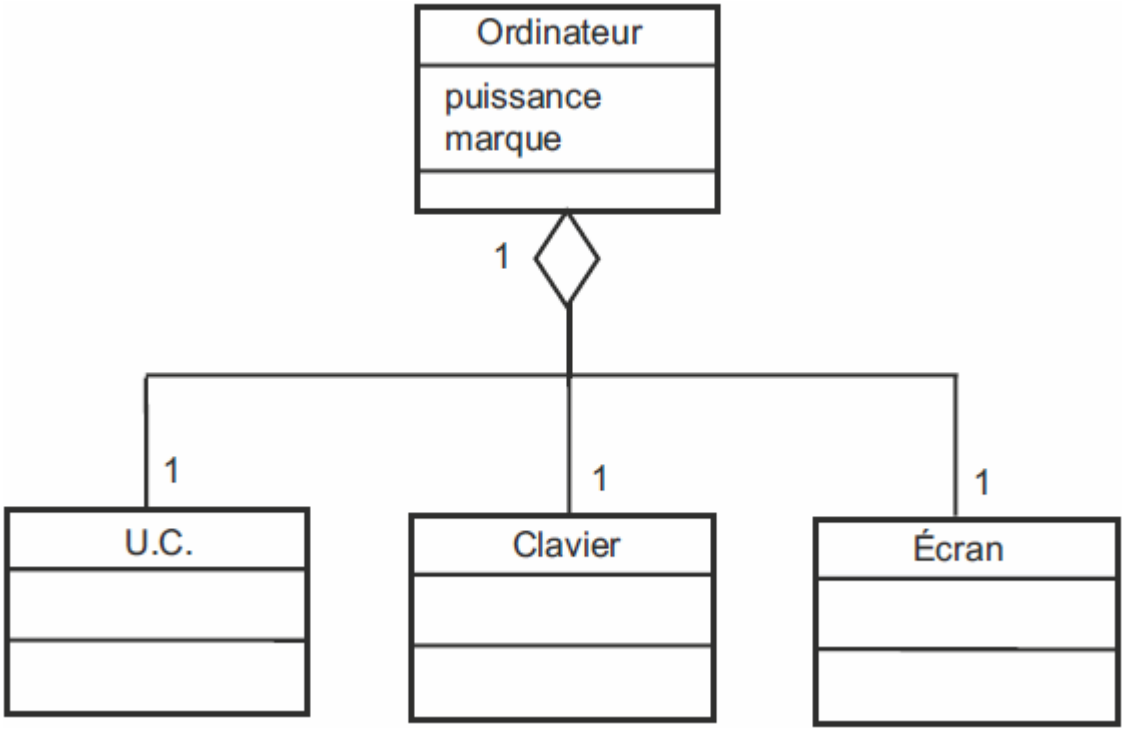
- L'agrégation est une association qui permet de représenter un lien de type « ensemble » comprenant des « éléments ».
- Il s'agit d'une relation entre une classe représentant le niveau « ensemble » et 1 à *n classes de niveau « éléments ».*
- *L'agrégation* représente un lien structurel entre une classe et une ou plusieurs autres classes.

- Formalisme



AGRÉGATION

- Exemple



Dans cet exemple, un ordinateur est modélisé par le fait qu'il comprend une UC, un clavier et un écran.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

COMPOSITION

- La composition est une relation d'agrégation dans laquelle il existe une contrainte de durée de vie entre la classe «composant» et la ou les classes « composé ».

- Contenance structurelle entre instances.

- La suppression de la classe « composé » implique la suppression de la ou des classes « composant ».

- Formalisme (1):

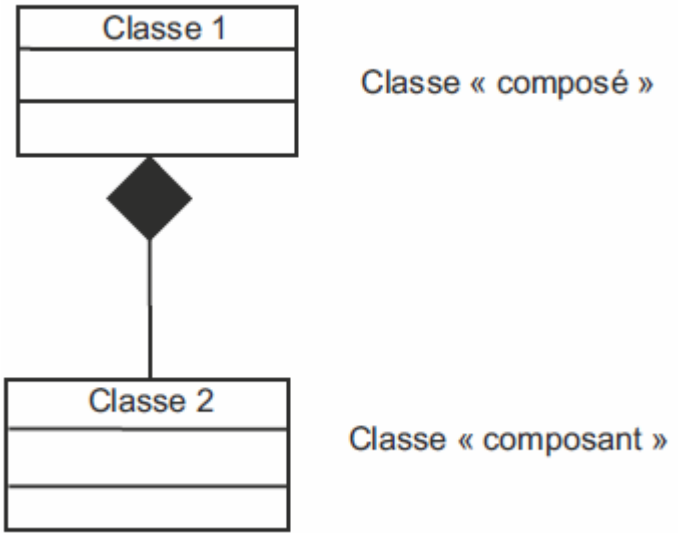
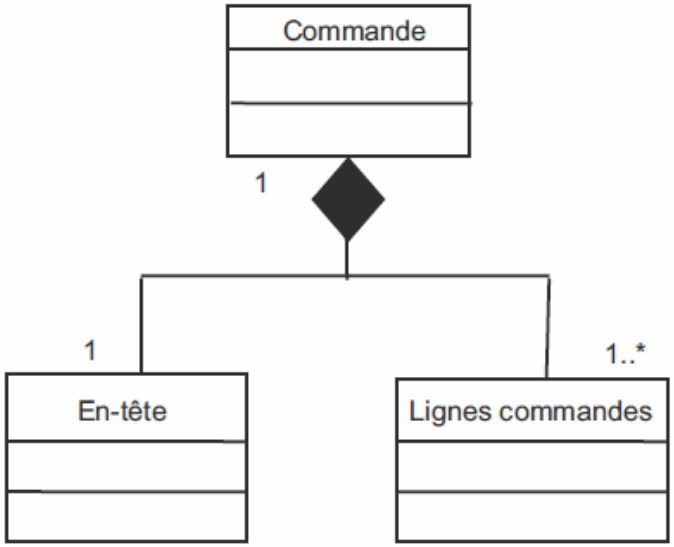


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

COMPOSITION

- Exemple:



- Formalisme (2):

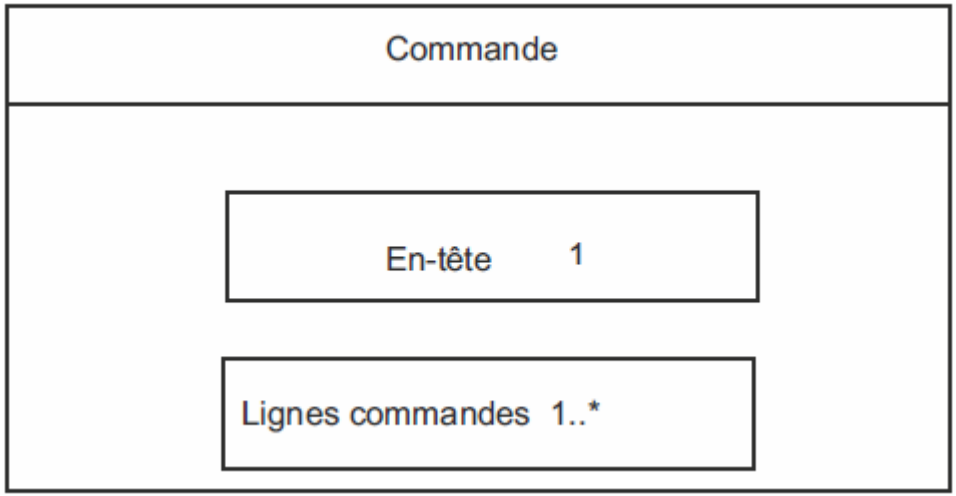


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

AGRÉGATION & COMPOSITION

- Exemple:

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

DÉPENDANCE

- La dépendance entre deux classes permet de représenter l'existence d'un lien sémantique.
- Une classe B est en dépendance de la classe A si des éléments de la classe A sont nécessaires pour construire la classe B.

- **Formalisme:**

  La relation de dépendance se représente par une flèche en pointillé entre deux classes.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

CLASSE INTERFACE

- Une classe d'interface permet de décrire la vue externe d'une classe.
- La classe d'interface, identifiée par un nom, comporte la liste des opérations accessibles par les autres classes.
- Le compartiment des attributs ne fait pas partie de la description d'une interface.
- L'interface peut être aussi matérialisée plus globalement par un petit cercle associé à la classe source.
- La classe utilisatrice de l'interface est reliée au symbole de l'interface par une flèche en pointillé.
- La classe d'interface est une spécification et non une classe réelle (ne peut pas servir à créer des objets).
- Une classe d'interface peut s'assimiler à une classe abstraite.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

INTERFACE

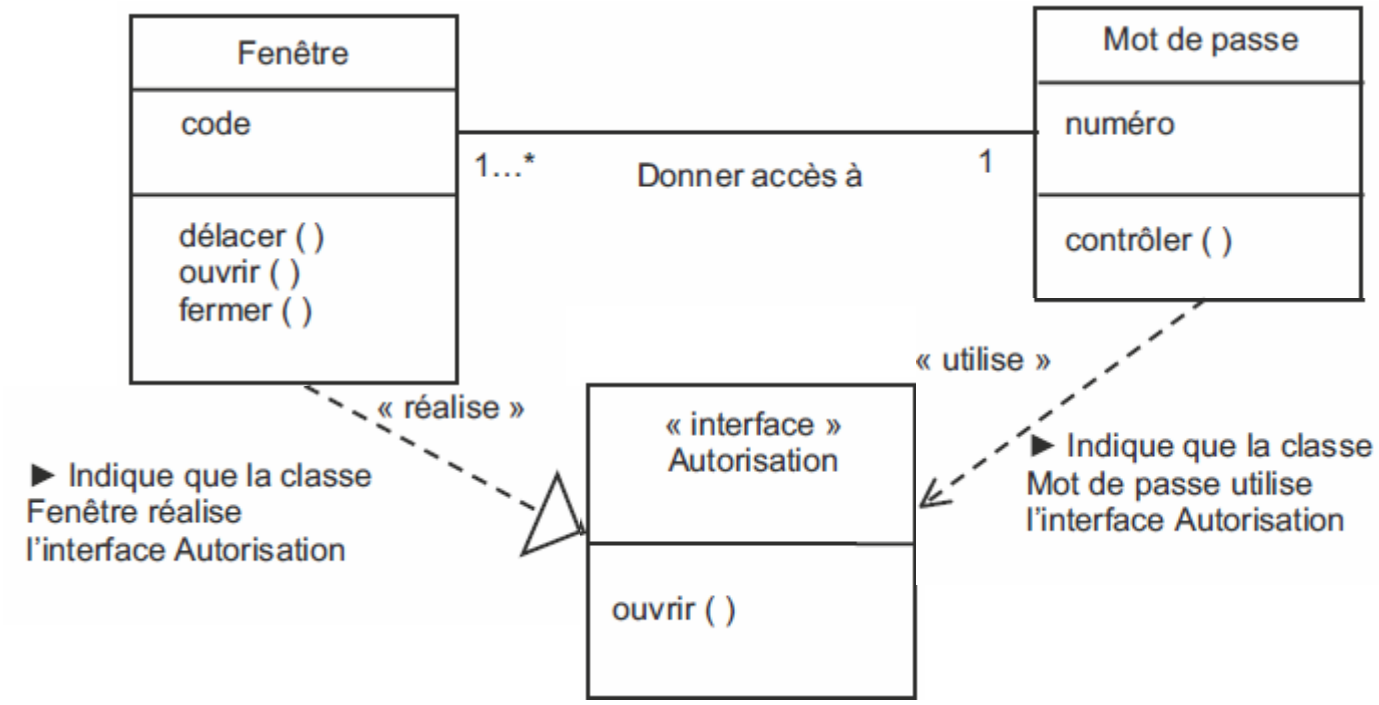


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

INTERFACE

- Exemple

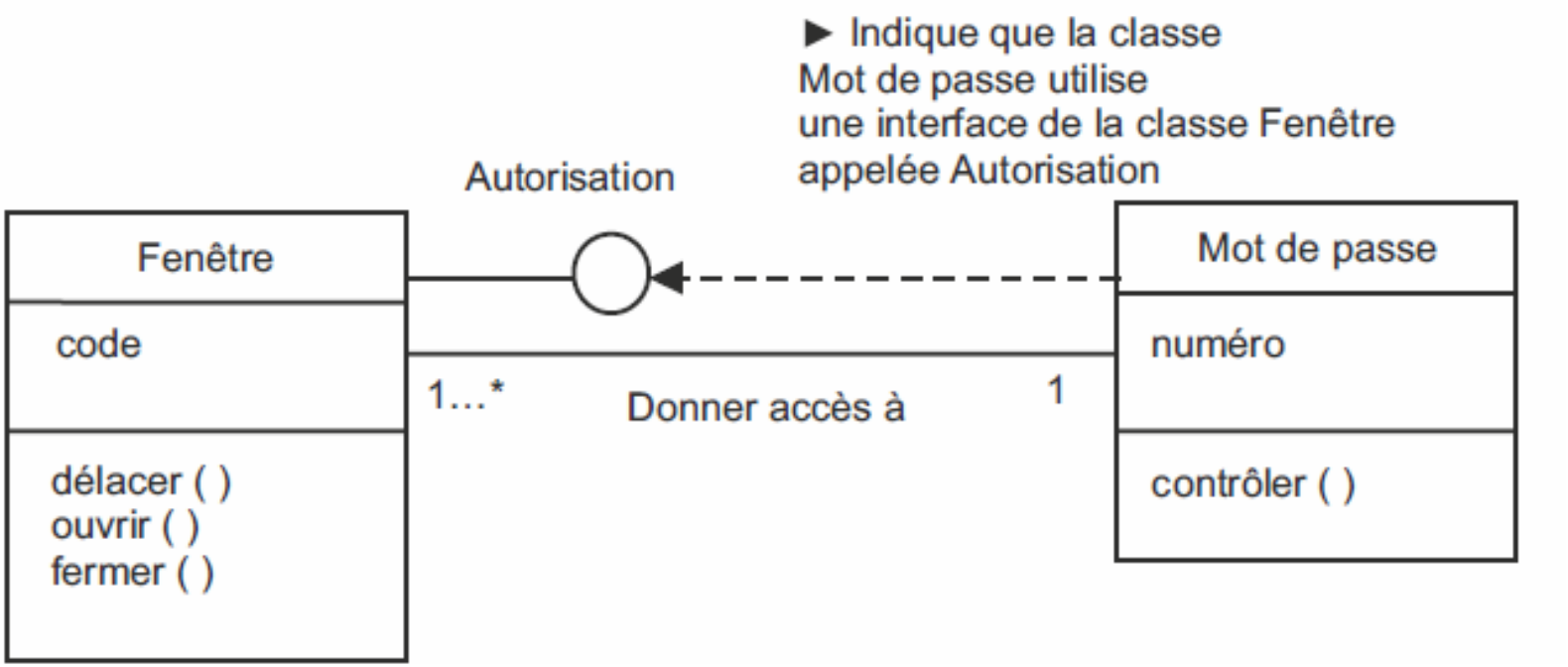


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

GÉNÉRALISATION & SPÉCIFICATION

- La généralisation est la relation entre une classe et deux autres classes ou plus, partageant un sous-ensemble commun d'attributs et/ou d'opérations.
- La classe généralisée s'appelle **super-classe**, les classes affinées s'appellent **sous-classes**.
- L'opération qui consiste à créer une super-classe à partir de classes s'appelle la **généralisation**.

- Inversement la **spécialisation** **(héritage simple)** consiste à créer des sous-classes à partir d'une classe.
- L'héritage permet à une sous-classe de disposer des attributs et opérations de la classe dont elle dépend.

- Le symbole utilisé pour la relation d'héritage ou de généralisation est une flèche avec un trait plein dont la pointe est un triangle fermé désignant le cas le plus général

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

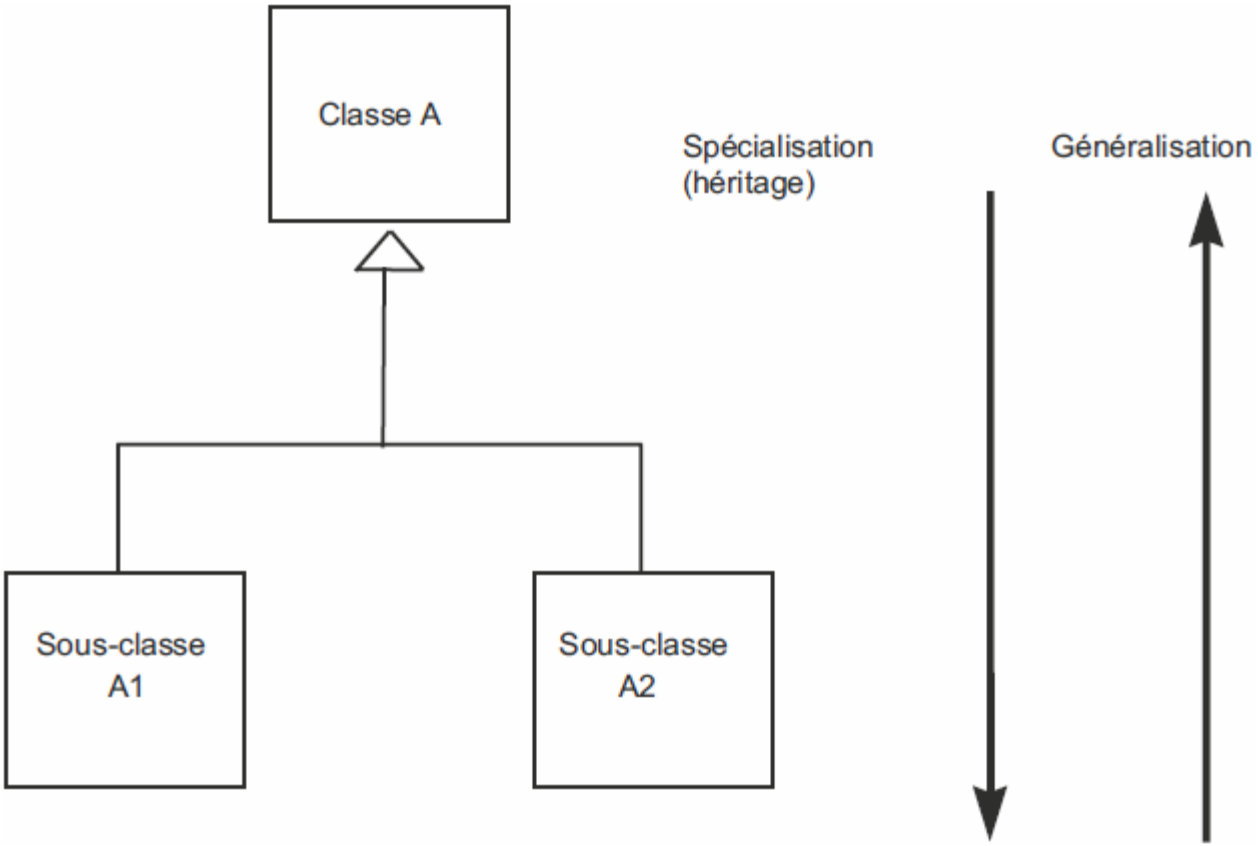
GÉNÉRALISATION & SPÉCIFICATION

- Les propriétés principales de l'héritage sont :
  - la classe enfant possède toutes les caractéristiques de ses classes parents, mais elle ne peut accéder aux caractéristiques privées de cette dernière ;
  - une classe enfant peut redéfinir (même signature) une ou plusieurs méthodes de la classe parent. Sauf indication contraire, un objet utilise les opérations les plus spécialisées dans la hiérarchie des classes ;
  - toutes les associations de la classe parent s'appliquent aux classes dérivées ;
  - une instance d'une classe peut être utilisée partout où une instance de sa classe parent est attendue.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

GÉNÉRALISATION & SPÉCIFICATION

- Formalisme:

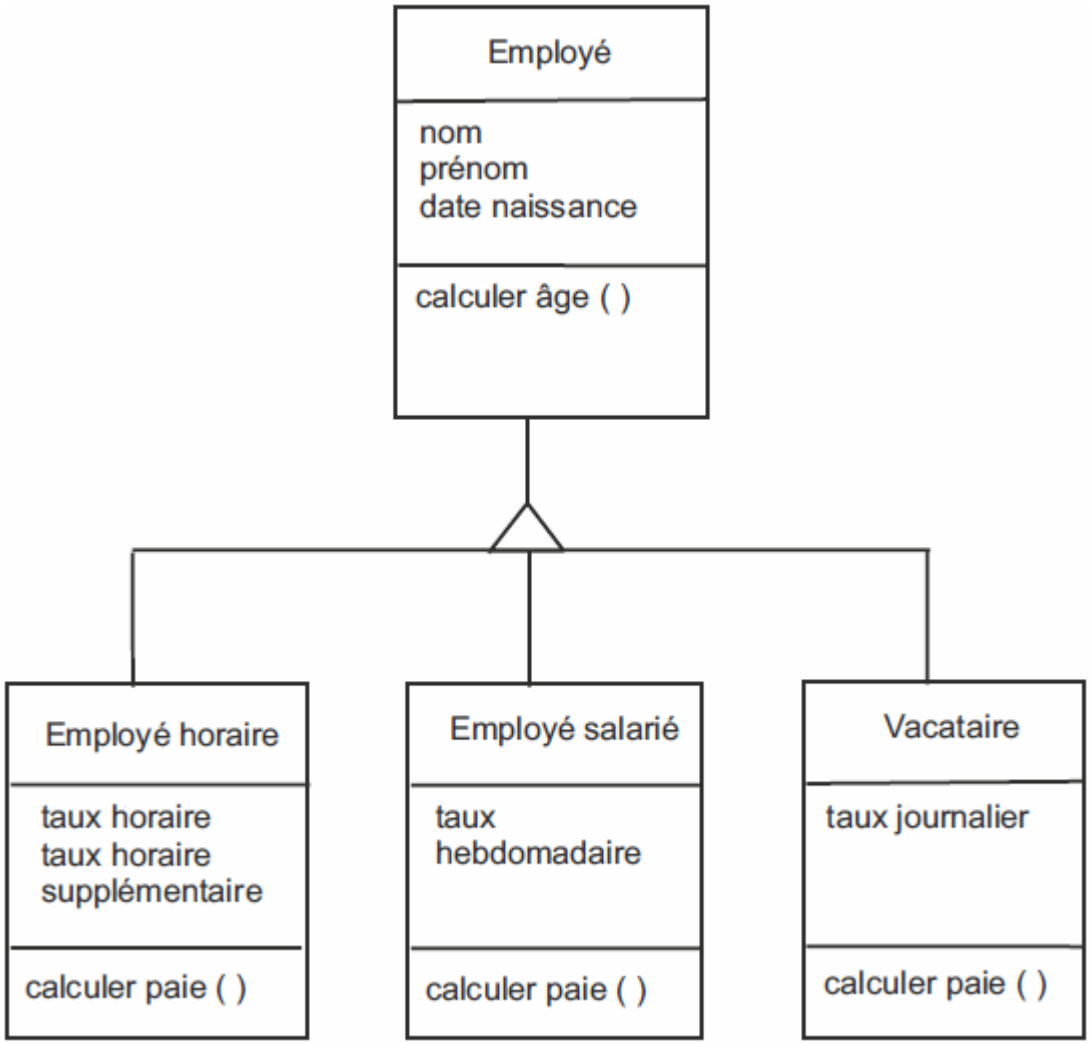


- la sous-classe A1 hérite de A, c'est une spécialisation de A
- la sous-classe A2 hérite de A, c'est une spécialisation de A.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

GÉNÉRALISATION & SPÉCIFICATION

- Exemple



- les attributs nom, prénom et date de naissance et l'opération « calculer âge » de « Employé » sont hérités par les trois sous-classes : Employé horaire, Employé salarié, Vacataire.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

HÉRITAGE D'OPÉRATION

- Opération commune aux sous-classes : Définition dans la super-classe

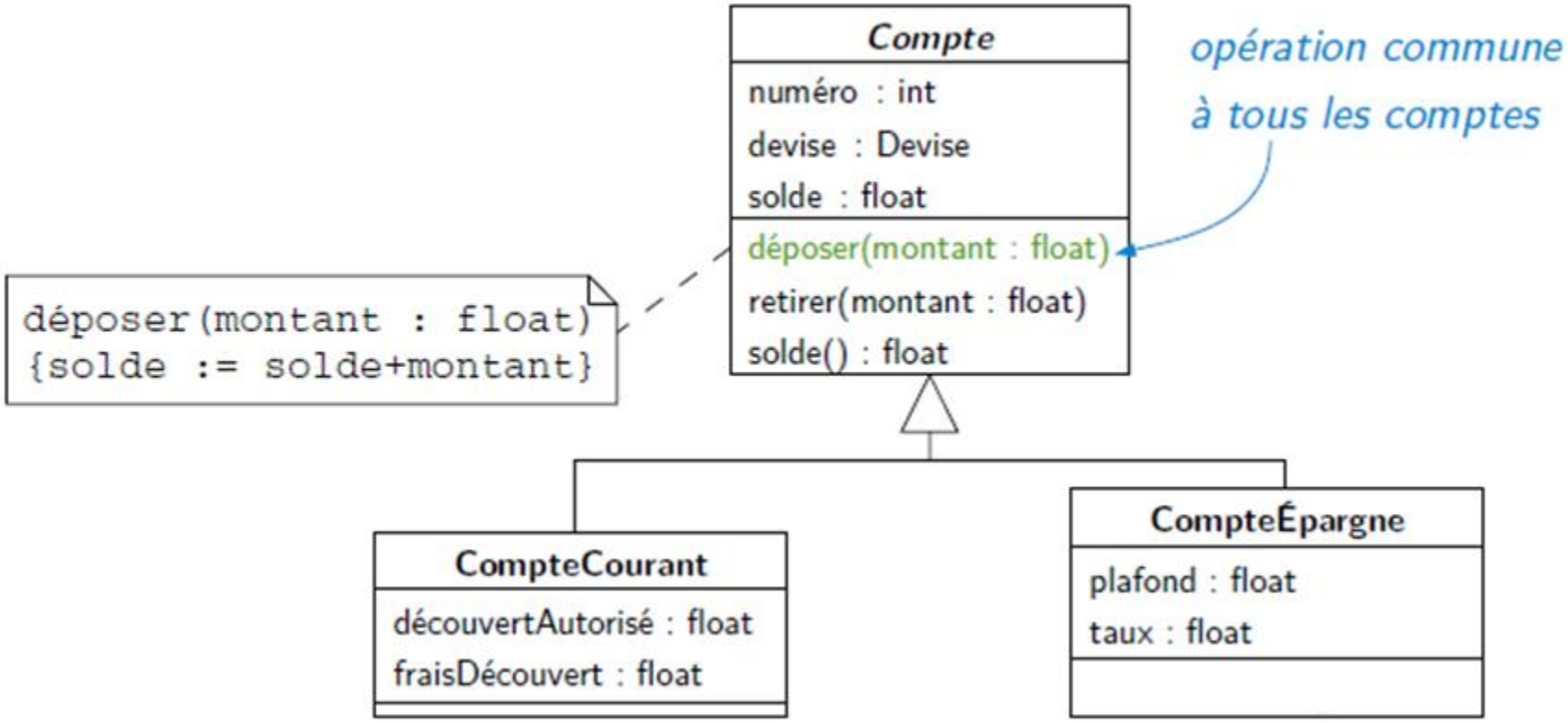


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

REDÉFINITION D'OPÉRATION

- Opération commune aux sous-classes : Définition dans la super-classe
- Possibilité de redéfinition locale de l'opération dans une sous-classe pour prendre en compte un cas particulier

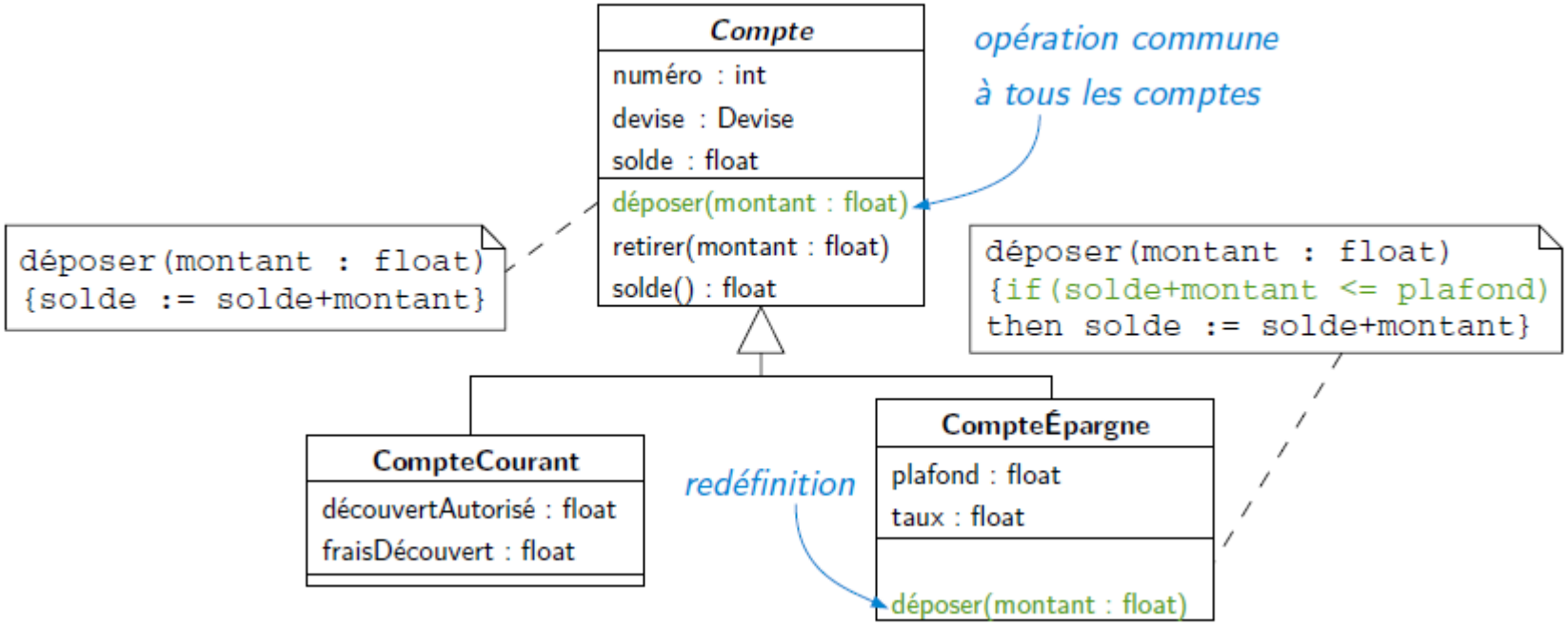


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

CLASSE ABSTRAITE

- Une classe abstraite est une classe qui n'a pas d'instance car certaines opérations sont non définies.

- Écriture :
  - Opération non définie en italique
  - Nom de la classe en italique (ou stéréotype « abstract »)

- Exemple: On ne peut pas calculer la surface d'une forme sans savoir de quelle forme il s'agit

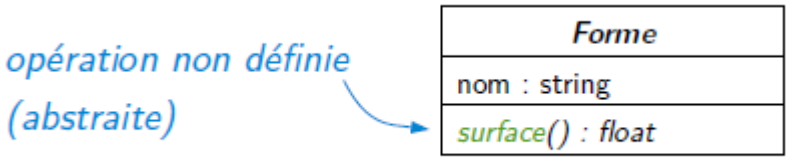


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

EXTENSION ET RESTRICTION DE CLASSE

- L'ajout de propriétés dans une sous-classe correspond à une extension de classe.
- Le masquage de propriétés dans une sous-classe correspond à une restriction de classe.
- Exemple:

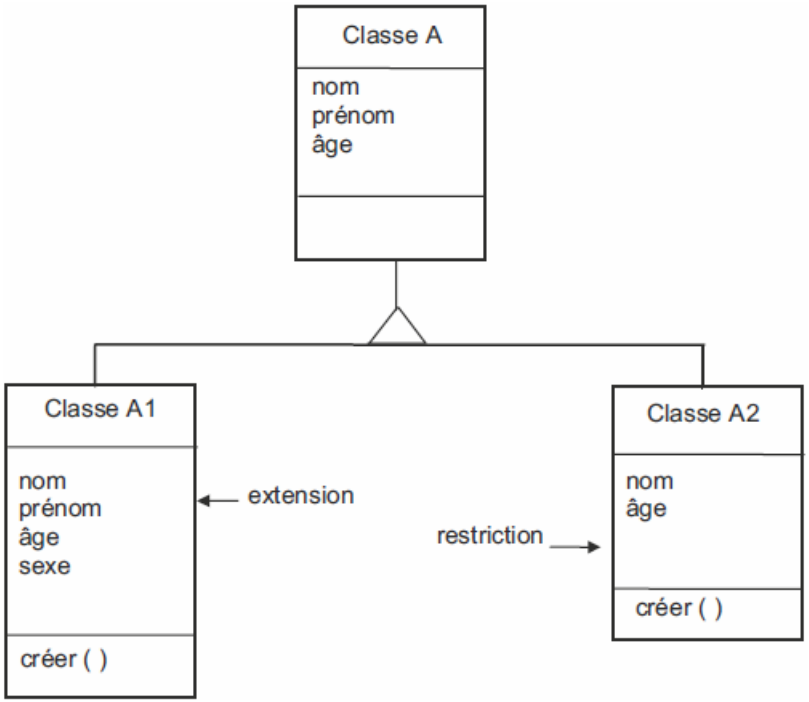


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

POLYMORPHISME

- Le polymorphisme représente la faculté d'une méthode à pouvoir s'appliquer à des objets de classes différentes.

- Une opération définie dans une superClasse peut s'exécuter de manière différente selon la sous-classe où elle est héritée.

- Lors de l'exécution, l'appel de l'opération va automatiquement déclencher l'exécution de l'opération de la sous-classe concernée.

- Lors de l'exécution de l'opération de la sous-classe, il est possible de faire appel à l'opération de la super-classe qui contient en général la partie **commune** applicable aux sousClasses.
- Sinon (pas de partie commune entre sous-classes): ne pas définir l'opération à polymorpher → super-classe abstraite

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

POLYMORPHISME

- Augmente la généricité du code.
- Permet la définition d'une opération abstraite dans les classes héritant d'une classe abstraite
- Opération polymorphe : Opération définie dans différentes sous-classes mais opération spécifique à la sous-classe

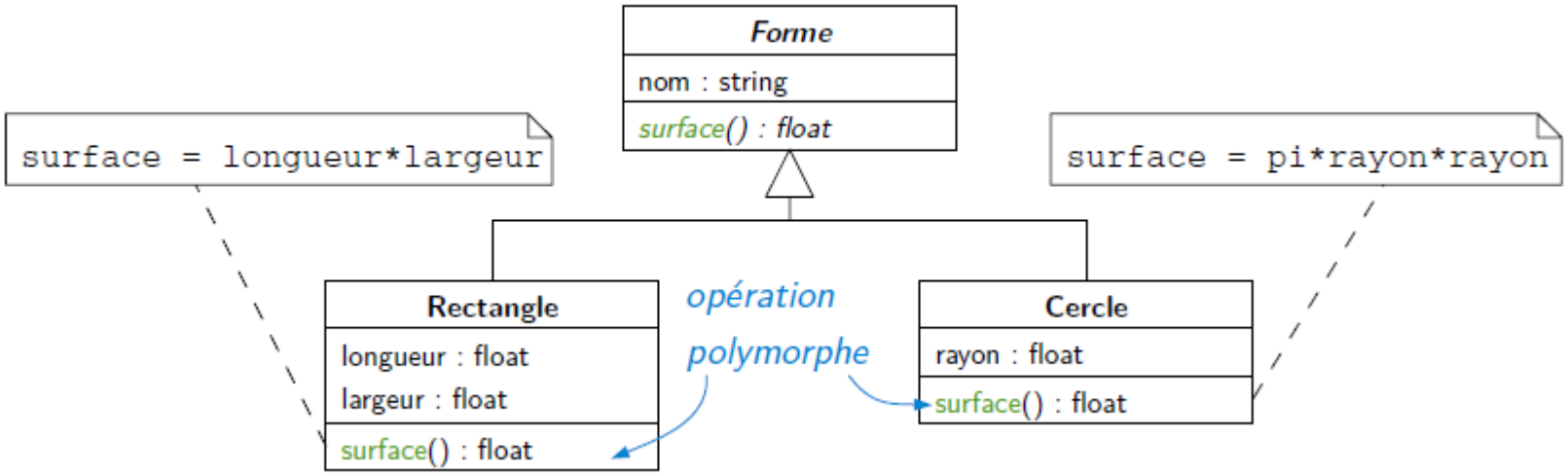


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

HÉRITAGE MULTIPLE

- Cas où une classe hérite de deux ou plusieurs classes «parentes» distinctes.

- Exemple: la classe « Véhicule amphibie » hérite des classes « Véhicule terrestre » et « Véhicule marin »

– Le C++ permet son implémentation ≠ Java

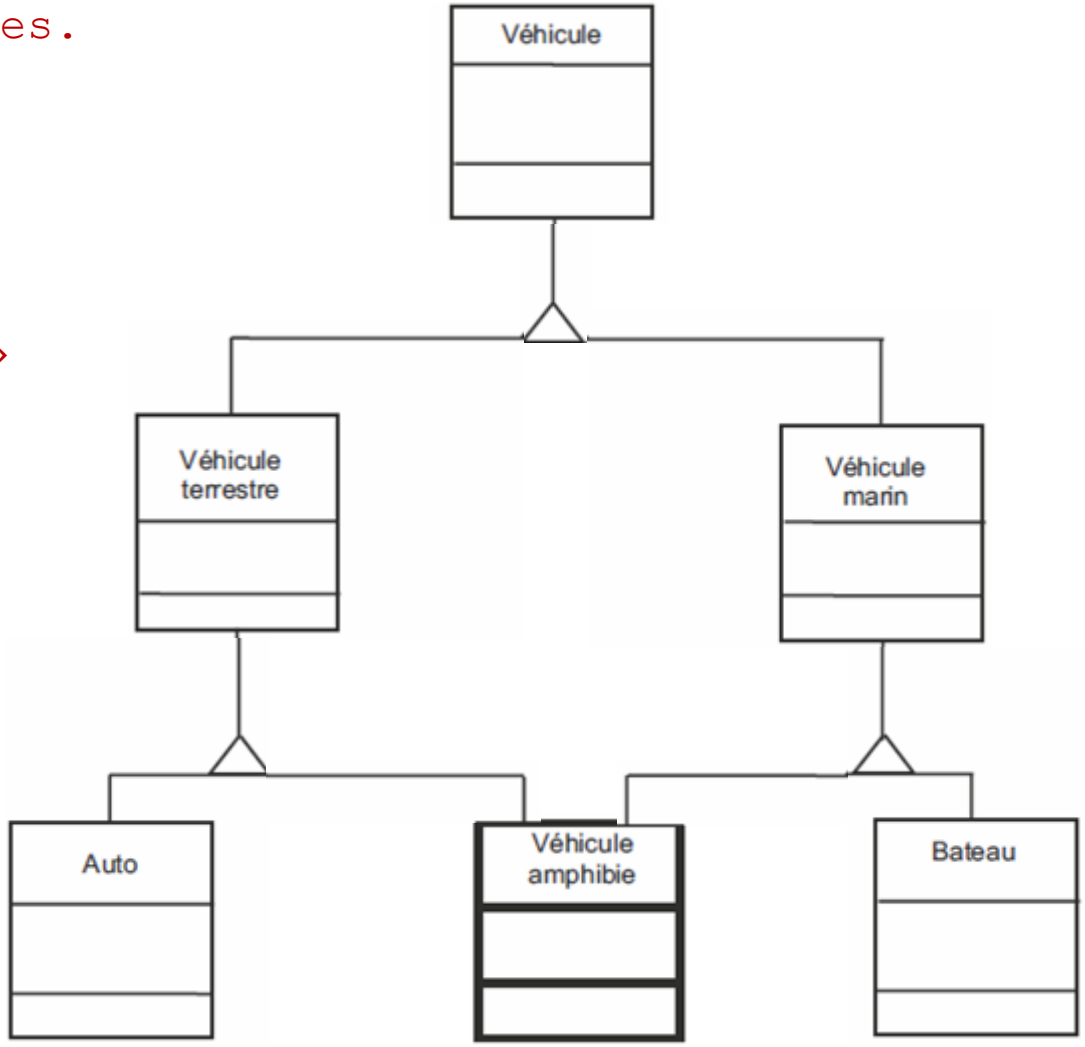


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

DIAGRAMME D'OBJET (DOB)

- Un diagramme d'objets donne une vue **figée** de l'état d'un système **à un instant donné (snapshot)**.
- Le DOB Représente des objets (*i.e.* instances de classes) et leurs liens (*i.e.* instances de relations).
- DOB, caractéristiques:
  - illustrer le modèle de classes en montrant un exemple qui explique le modèle ;
  - Le DOB utilise les mêmes concepts que le diagramme de classes
  - Le diagramme de classes modélise les règles et le diagramme d'objets modélise des faits.
  - préciser certains aspects du système en mettant en évidence des détails imperceptibles dans le diagramme de classes ;
  - exprimer une exception en modélisant des cas particuliers ou des connaissances non généralisables qui ne sont pas modélisés dans un diagramme de classe ;
  - Un DOB ne montre pas l'évolution du système dans le temps.

    Pour représenter une interaction, il faut utiliser un diagramme de communication ou de séquence (cf. plus loin).

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

EXERCICE 1

- **Énoncé:**

  Une entreprise nationale de vente d'appareil électroménager souhaite réaliser une première expérience d'analyse objet avec la méthode UML sur un petit sousensemble de son SI. Ce sous-ensemble concerne le suivi des personnels des agences locales implantées dans les régions. Chaque région est pilotée par une direction régionale qui a en charge un certain nombre d'agences locales. Une direction régionale est caractérisée par un code et un libellé. Chaque agence est caractérisée par un code, un intitulé, une date de création et une date de fermeture. À une agence sont rattachées une à plusieurs personnes. Chaque personne est caractérisée par les données : numéro, qualité (M., Mme), nom, prénom, date de naissance, date prévisionnelle d'arrivée, date d'arrivée et date de départ. Il est demandé d'élaborer le diagramme de classe de ce premier sous-ensemble du SI de cette entreprise.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

EXERCICE 1

- **Solution: diagramme de classe**

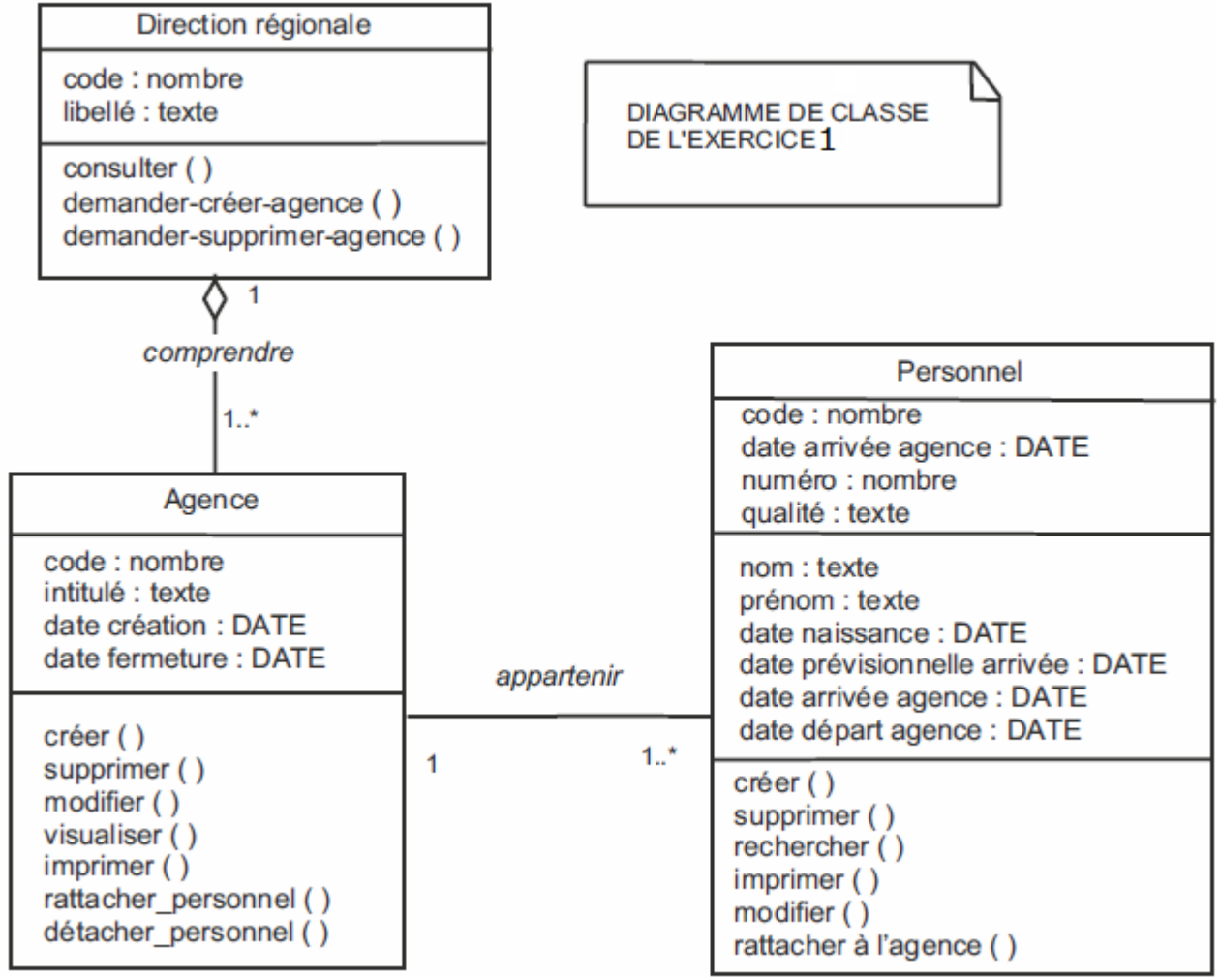


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

EXERCICE 1

- **Solution: diagramme d'objet**

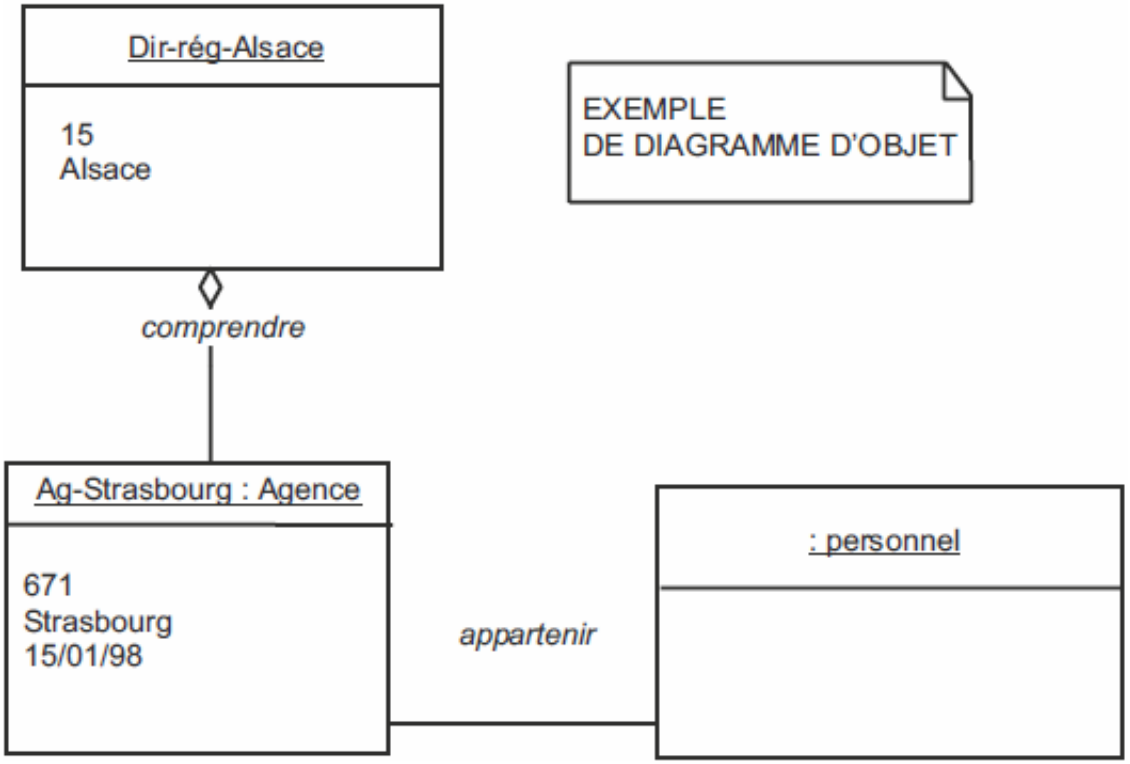


DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

EXERCICE DE SYNTHÈSE: RENTCOT

- **Gîte(W) (n.m):**

  Lieu où l'on peut se loger. Lieu d'hébergement meublé aménagé et destiné à être loué à des fins touristiques, pour une courte durée (week-end, quelques jours, voire semaines).

- **Ennoncé (1):**

« RentCot » est une association qui permet à divers propriétaires ruraux de mettre en location, à la semaine, des gîtes meublés.

Elle publie annuellement un catalogue contenant les gîtes proposés par les propriétaires. Les gîtes doivent répondre à un certain nombre de critères qualité, correspondant à un nombre d'étoiles, qui sont vérifiées lors de l'adhésion du gîte et une fois tous les trois ans lors d'une visite de contrôle.

Le propriétaire reçoit tous les ans un catalogue des gîtes, et peut modifier les informations qui le concernent (prix par saison, photo du gîte, nombre de personnes, de chambres, terrain...).

« RentCot » regroupe 450 gîtes, pour une moyenne de 12 semaines de réservation par gîte et par an.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

EXERCICE DE SYNTHÈSE: RENTCOT

- **Ennoncé (2):**

« RentCot » propose aux propriétaires qui le souhaitent, un service central de réservation.

Tous les ans, les propriétaires qui veulent utiliser ce service signent un contrat avec «RentCot», qui spécifie les périodes ouvertes à la location et la rémunération de la centrale de réservation en pourcentage de chaque location, ce dernier taux étant valable pour l'année et pour l'ensemble des gîtes.

Le propriétaire, en signant le contrat, joint un relevé d'identité bancaire.

Le propriétaire ayant signé le contrat de la réservation centrale reçoit chaque mois un état des réservations fermes.

Il reçoit aussi tous les mois un état des sommes encaissées par la centrale de réservation.

Le virement bancaire des sommes dues, correspondant à l'état précédent, est envoyé en milieu du mois suivant.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

EXERCICE DE SYNTHÈSE: RENTCOT

- **Ennoncé (3):**

Un client potentiel (que l'on peut appeler client réservataire) téléphone à la centrale de réservation pour réserver un gîte sur la base du catalogue.

La centrale de réservation prend en compte la demande, et lui envoie un contrat de location ainsi qu'une demande d'acompte si un accord a été trouvé sur les dates de réservation.

Le client réservataire renvoie le contrat signé accompagné de l'acompte : la réservation devient ferme.

Un mois avant le séjour, le client locataire envoie le solde du paiement; il reçoit alors une confirmation de séjour lui donnant les coordonnées de la personne à contacter pour convenir de son arrivée.

Le client peut à tout moment annuler son séjour, 30% des sommes versées ne sont pas remboursées.

En cas de non-retour du contrat signé après 15 jours, la pré-réservation est automatiquement annulée.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

EXERCICE DE SYNTHÈSE: RENTCOT

- **Corrigé (1):**

  l'analyse de l'ensemble des données disponibles nous permet de mettre en évidence les classes suivantes :

  - Propriétaire,
  - Gîte,
  - Gîtes gérés,
  - Catalogue,
  - Activité,
  - Période Location,
  - Réservation,
  - Client
  - Contrat de Location.

DIAGRAMME DE CLASSE (DCL) & DIAGRAMME D'OBJET (DOB)

EXERCICE DE SYNTHÈSE: RENTCOT

- **Corrigé (2):** Principales informations portées par les documents échangés

DIAGRAMME DE COMPOSANT (DCP)

- Permet de représenter les composants **logiciels** d'un système ainsi que les liens existant entre ces composants.
- Vue statique de l'implémentation du système illustrant les choix de réalisation.
- Les diagrammes de composants sont composés :
  - **des descriptions des implémentations du système (les composants),**
  - **des groupes d'implémentations (les modules),**
  - **des relations entre les diverses implémentations (les dépendances).**
- *Ils sont généralement utilisés pour décrire les choix d'implémentation et les* dépendances de compilation et d'implémentation entre les composants du système.
- Les composants logiciels peuvent être de deux origines :
  - Composants métiers propres à une entreprise.
  - Composants disponibles sur le marché comme par exemple les composants EJB, CORBA, .NET, WSDL.

DIAGRAMME DE COMPOSANT (DCP)

- Composant:
  - Elément physique représentant une partie de l'implémentation du système (assimilé à un élément exécutable du système):
    - **code (source, binaire ou exécutable),**
    - **script, fichier de commande,**
    - **fichier de données, table, ...**
  - *implante des services utilisables par d'autres composants.*
  - **Caractérisé** par :
    - un nom
    - une spécification externe sous forme:
      - soit une ou plusieurs interfaces requises: interface(s) nécessaire(s) au bon fonctionnement du composant.
      - soit d'une ou plusieurs interfaces fournies: interface(s) proposée(s) par le composant aux autres composants.
    - un port de connexion: point de connexion entre le composant et une interface.
      L'identification d'un port permet d'assurer une certaine indépendance entre le composant et son environnement extérieur.

DIAGRAMME DE COMPOSANT (DCP)

- *Formalisme général:* Un composant est représenté par:
  - un classeur avec le mot-clé «composant» ou bien par
  - un classeur comportant une icône représentant un module.

Figure. Formalisme général d'un composant

- UML propose des stéréotypes de composants :
  - *<<document>>: un document quelconque;*
  - *<<exécutable>>: un programme qui peut s'exécuter sur un* nœud (cf. diagrammes de déploiement);
  - *<<fichier>>: un document contenant du code source ou des* données;
  - *<<bibliothèque>>: une bibliothèque statique ou dynamique;*
  - *<<table>>: une table d'une base de données.*
- *En général, un composant représente l'implantation d'une classe.*
- Les instances de composants sont surtout utilisées dans les **diagrammes de déploiement (cf. plus loin).**

Diagramme de composant (DCP)

Les deux types de représentation

- Deux types de représentation sont disponibles pour modéliser les composants :
  - une représentation « boîte noire »
  - une représentation « boîte blanche »

- Pour chaque représentation, plusieurs modélisations des composants sont proposées.

Diagramme de composant (DCP)

Représentation "Boite noire"

- **Boîte noire (W): la représentation d'un système sans considérer son fonctionnement interne (que ce soit un objet mécanique ou électronique, un organisme, une personne, un mode d'organisation sociale, ou tout autre système);**
- C'est une vue externe du composant qui présente ses interfaces fournies et requises sans entrer dans le détail de l'implémentation du composant.
- Une boîte noire peut se représenter de différentes manières:
  - **Connecteur d'assemblage**
  - **Connecteur d'interfaces**
  - **Compartiment**

Représentation "Boite blanche"

- **Dans la théorie des systèmes, une boîte blanche (white box [En]), ou boîte transparente, est un module d'un système dont on peut prévoir le fonctionnement interne car on connaît les caractéristiques de fonctionnement de l'ensemble des éléments qui le composent**
- C'est une vue interne du composant qui décrit son implémentation à l'aide de classificateurs (classes, autres composants) qui le composent.
- Pour la représentation, plusieurs modélisations sont possibles:
  - **Dépendance**
  - **Compartiment**
  - **Ports et connecteurs**

DIAGRAMME DE COMPOSANT (DCP)

REPRÉSENTATION "BOITE NOIRE" (1)

- **Connecteur d'assemblage**
  - Une interface fournie se représente à l'aide d'un trait et d'un petit cercle et une interface requise à l'aide d'un trait et d'un demi-cercle.
  - Ce sont les **connecteurs d'assemblage.**

- **Exemple:**

Figure. Représentation d'un connecteur d'assemblage

  - le composant Commande possède deux interfaces fournies et deux interfaces requises.

DIAGRAMME DE COMPOSANT (DCP)

REPRÉSENTATION "BOITE NOIRE" (2)

- **Connecteur d'interfaces:**
  - Représentation qui fait recours aux **dépendances d'interfaces *utilise et réalise:***
    - Pour une interface fournie, c'est une relation de **réalisation** partant du composant et allant vers l'interface ;
    - Pour une interface requise, c'est une dépendance avec le mot-clé « **utilise** » partant du composant et allant vers l'interface.
- **Exemple:**

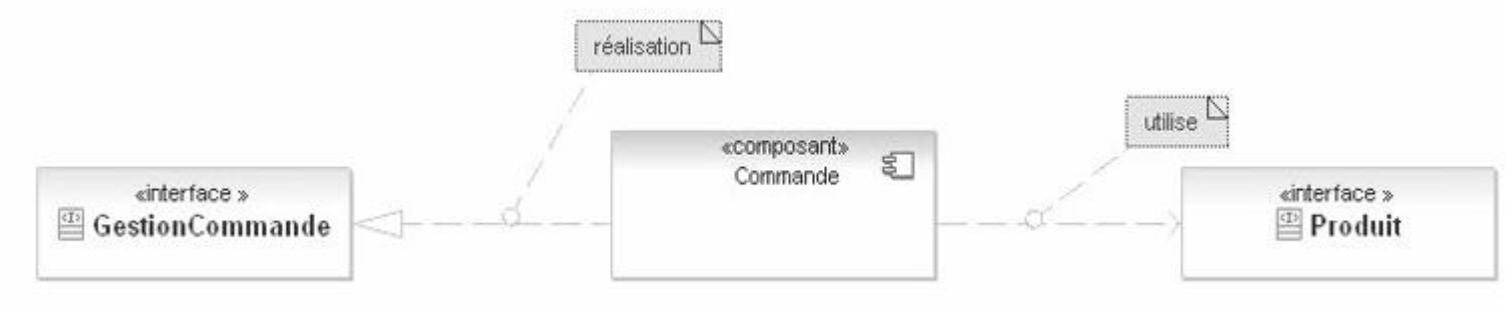


Figure. Représentation d'un composant avec connecteur d'interfaces

– Le composant Commande possède une interface fournie «GestionCommande» et une interface requise « Produit ».

DIAGRAMME DE COMPOSANT (DCP)

REPRÉSENTATION "BOITE NOIRE" (3)

- **Compartiment**
  - Décrire sous forme textuelle les interfaces fournies et requises à l'intérieur d'un **second compartiment**

- **Exemple:**

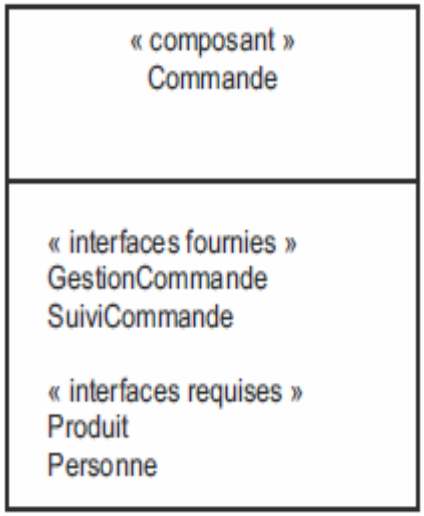


Figure.Représentation d'un composant avec compartiments

DIAGRAMME DE COMPOSANT (DCP)

REPRÉSENTATION "BOITE BLANCHE" (1)

- **Dépendance**
  - Les classificateurs qui composent le composant sont reliés à celui-ci par une relation de dépendance.
  - Les relations entre les classificateurs (association, composition, agrégation) sont aussi modélisées.
  - Néanmoins, si elles sont trop complexes, elles peuvent être représentées sur un diagramme de classe relié au composant par une note.

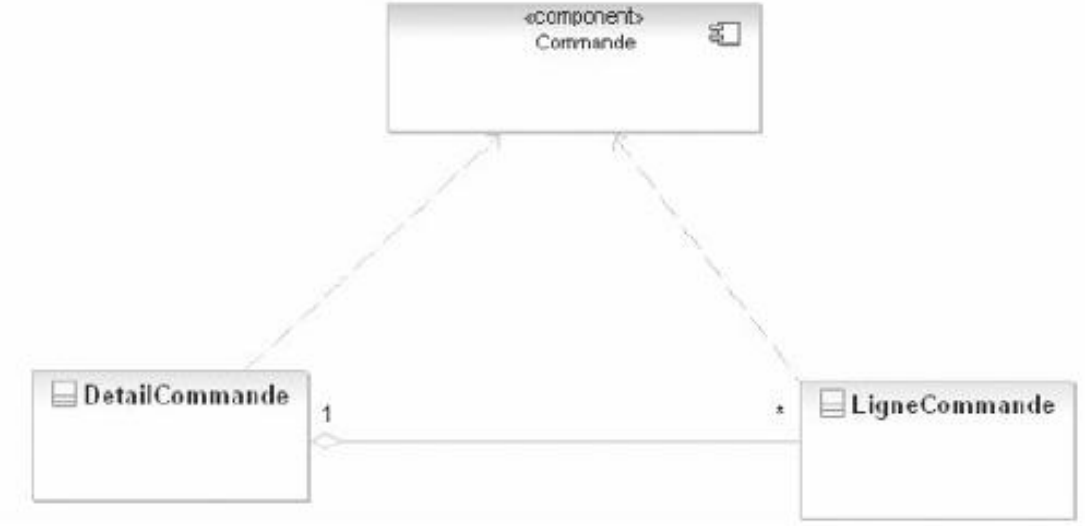


Figure. Représentation boîte blanche avec dépendances

DIAGRAMME DE COMPOSANT (DCP)

REPRÉSENTATION "BOITE BLANCHE" (2)

- **Compartiment**
  - Décrire sous forme textuelle les interfaces fournies et requises à l'intérieur d'un compartiment,
  - Les classificateurs (classes, autres composants) dans un autre compartiment,
  - Les artefacts (élément logiciel : jar, war, ear, dll) qui représentent physiquement le composant dans un dernier compartiment.
- **Exemple:**

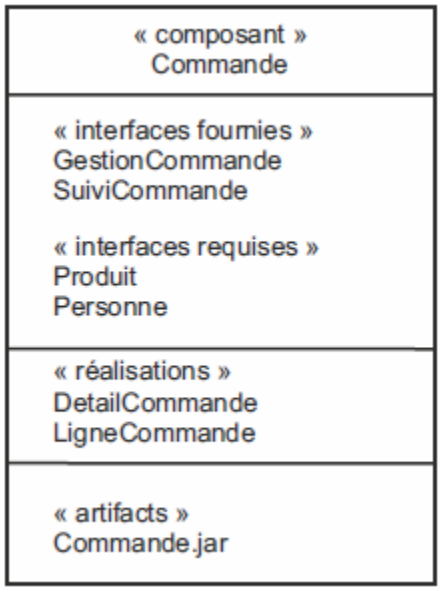


Figure. Représentation boîte blanche avec compartiments

DIAGRAMME DE COMPOSANT (DCP)

REPRÉSENTATION "BOITE BLANCHE" (3)

- **Ports et connecteurs (1)**
  - Le port est représenté par un petit carré sur le composant.
  - Les connecteurs permettent de relier les ports aux classificateurs.
  - Ils sont représentés par une association navigable et indiquent que toute information arrivée au port est transmise au classificateur.
- **Exemple:**

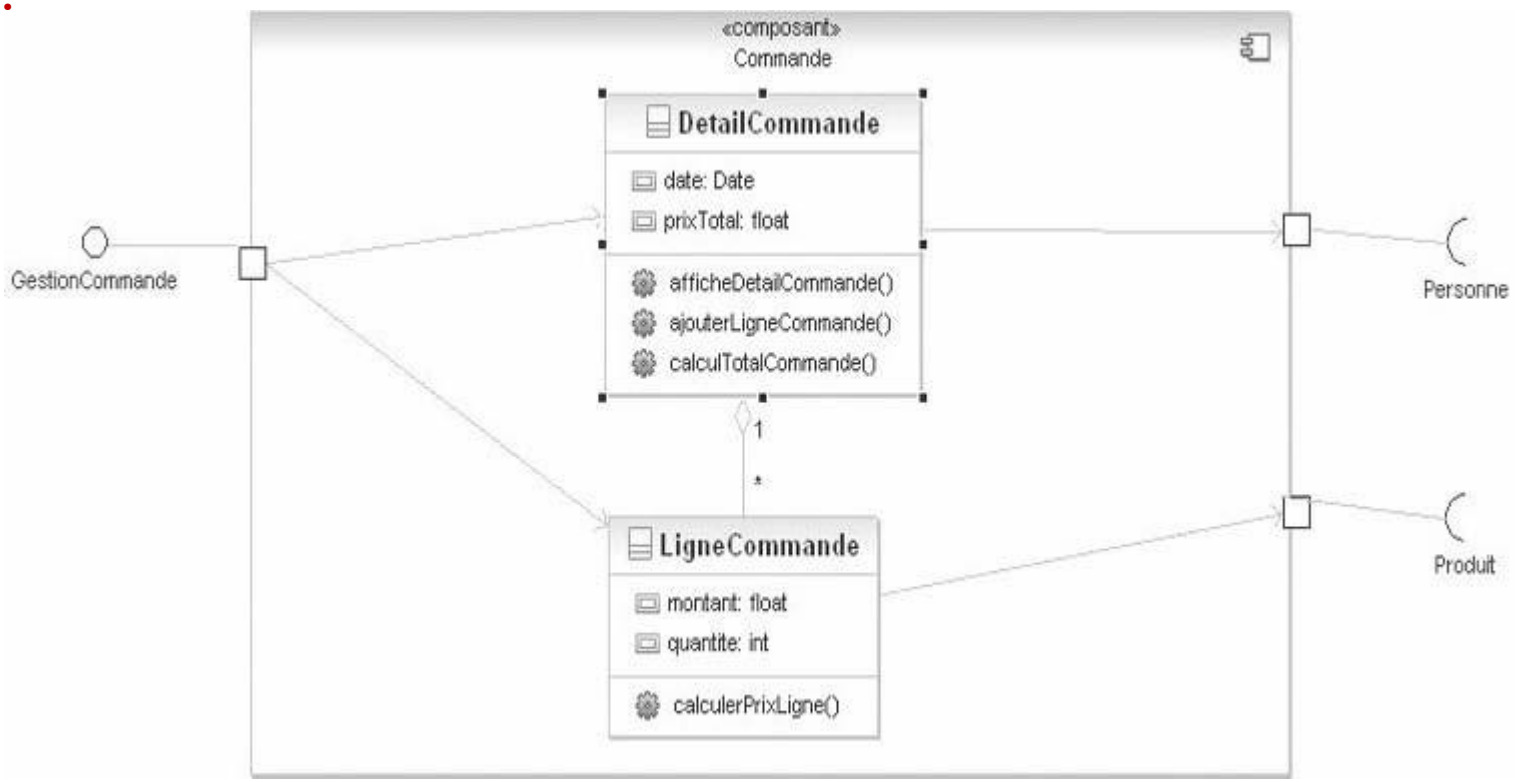


DIAGRAMME DE COMPOSANT (DCP)

REPRÉSENTATION "BOITE BLANCHE" (4)

- **Ports et connecteurs (2)**
  - le composant Commande est constitué de deux classes (classificateur) reliées par une agrégation: DetailCommande et LigneCommande.

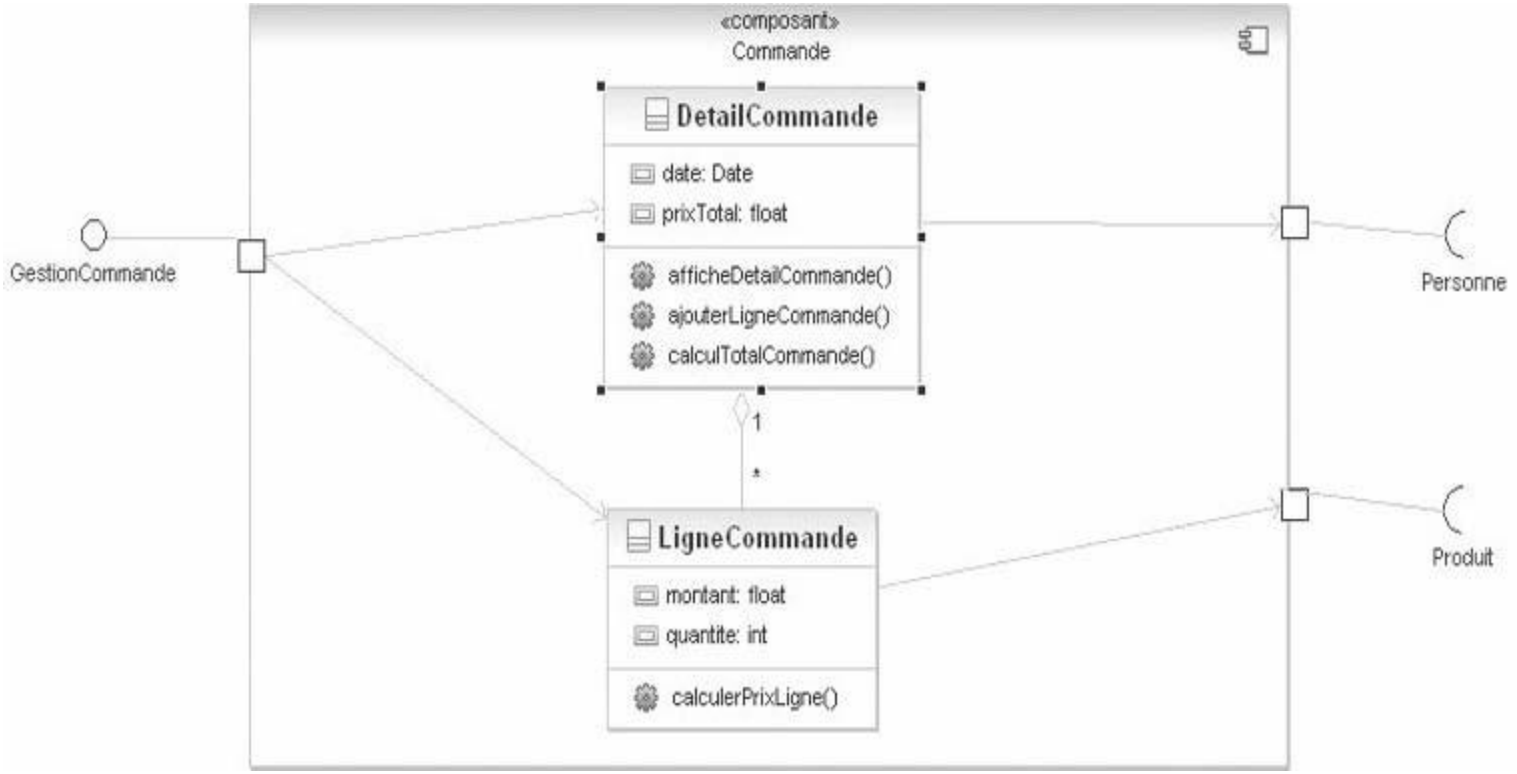


DIAGRAMME DE COMPOSANT (DCP)

REPRÉSENTATION "BOITE BLANCHE" (5)

- **Ports et connecteurs (3)**
  - L'interface fournie GestionCommande est accessible de l'extérieur *via un port et* permet d'accéder *via les connecteurs aux opérations des deux classes DetailCommande* et LigneCommande (ajouterLigneCommande, calculTotal-Commande, calculPrixLigne).

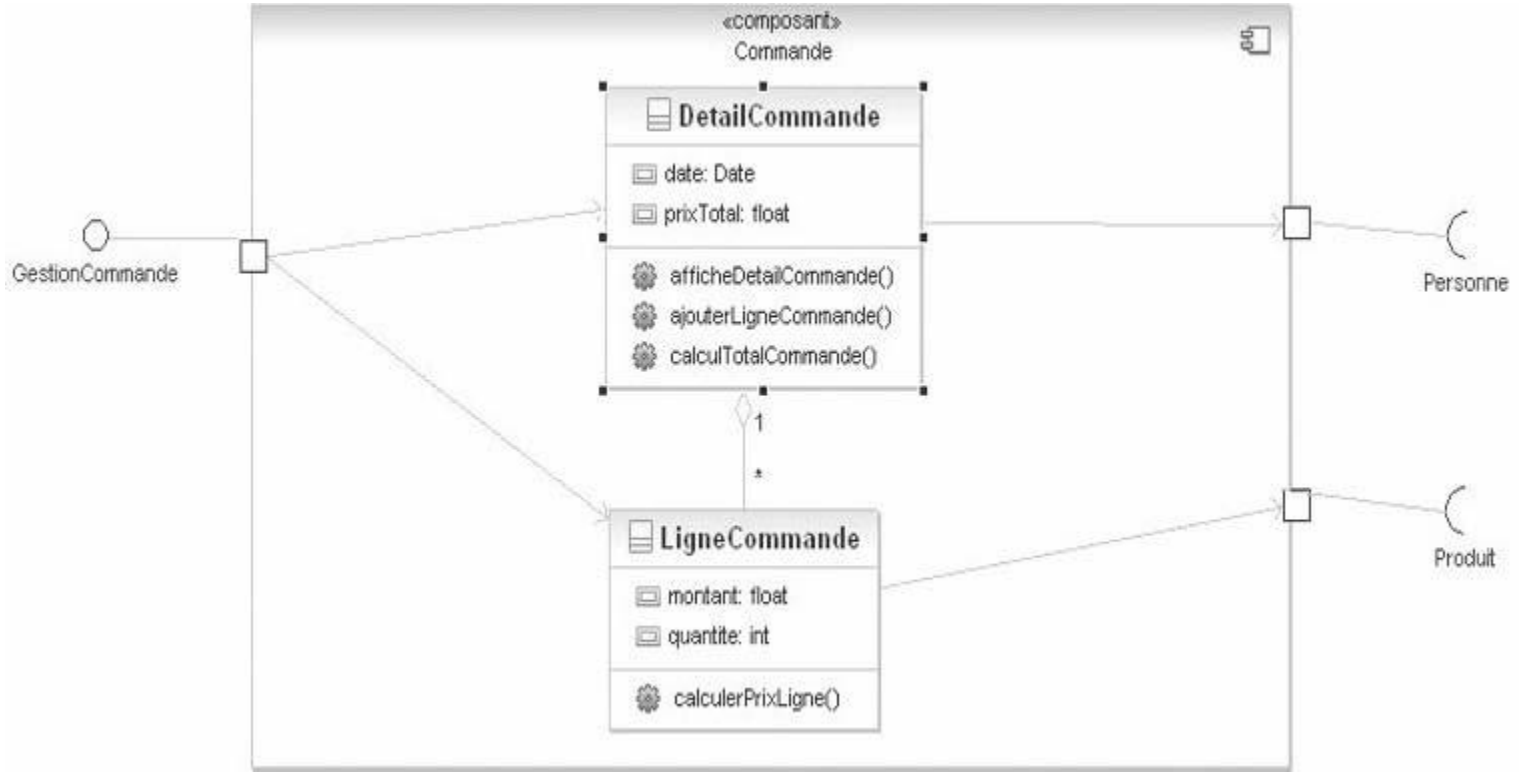


DIAGRAMME DE COMPOSANT (DCP)

REPRÉSENTATION "BOITE BLANCHE" (6)

- **Ports et connecteurs (4)**
  - L'interface requise Personne est nécessaire pour l'affichage du détail de la commande et est accessible *via un port du composant Commande.*
  - L'interface requise Produit est nécessaire pour le calcul du prix de la ligne de commande et est accessible *via un port du composant Commande.*

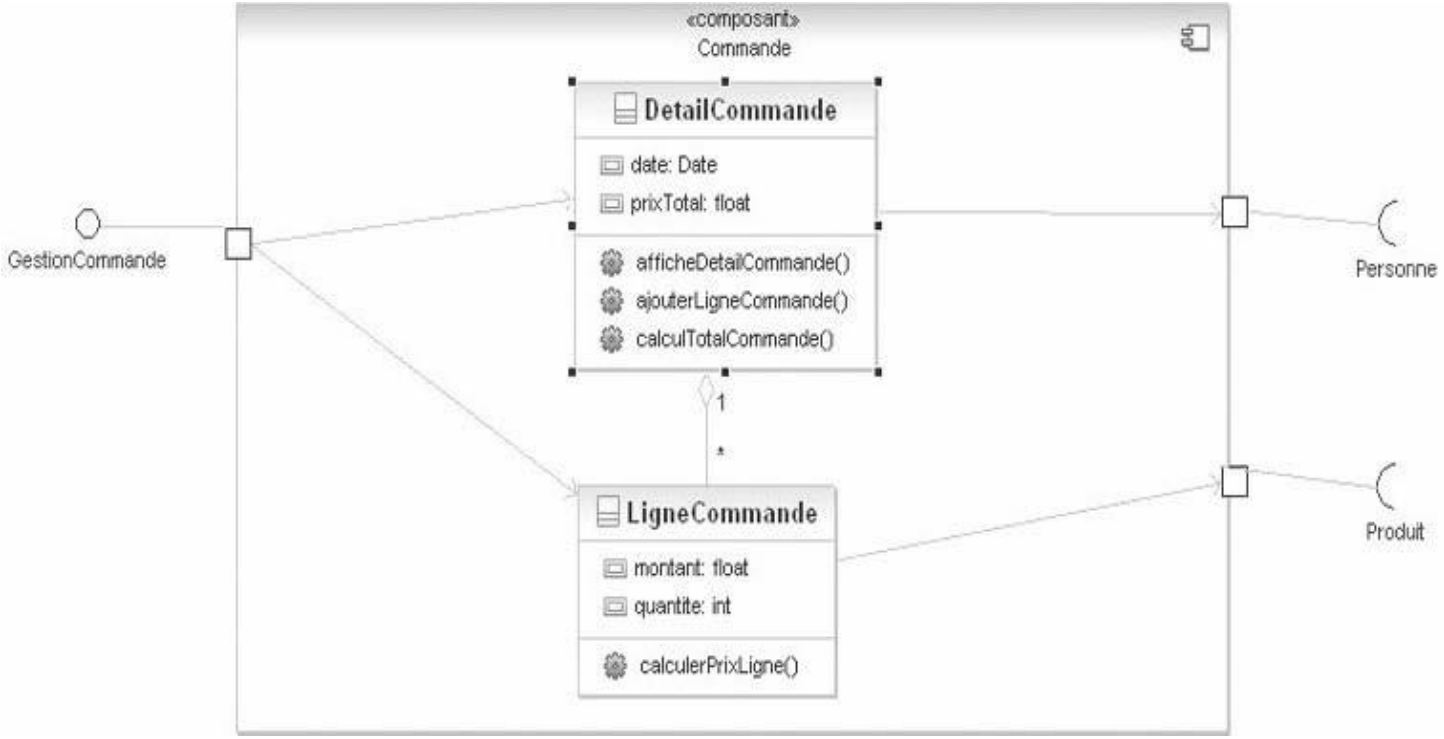


Diagramme de Déploiment (DPL)

- Permet de représenter l'architecture physique supportant l'exploitation du système.
- Cette architecture comprend:
  - des noeuds correspondant aux supports physiques (serveurs, routeurs…)
  - la répartition des artefacts logiciels (bibliothèques, exécutables…) sur ces noeuds.
- C'est un véritable réseau constitué de noeuds et de connexions entre ces noeuds qui modélise cette architecture.
- Les diagrammes de déploiement existent sous deux formes : spécification et instance.

Diagramme de Déploiment (DPL)

Noeud

- Ressource matérielle de traitement sur laquelle des artefacts seront mis en oeuvre pour l'exploitation du système.

- Les noeuds peuvent être interconnectés pour former un réseau d'éléments physiques.

- *Formalisme:*

  Un noeud ou une instance de noeud se représente par un cube ou parallélépipède

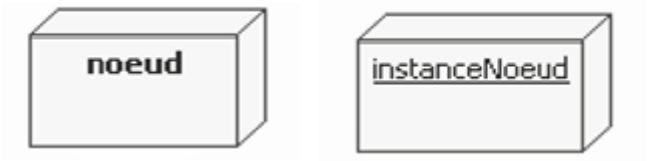


Figure – Représentation de nœud ou d'instance de nœud

- Ex:

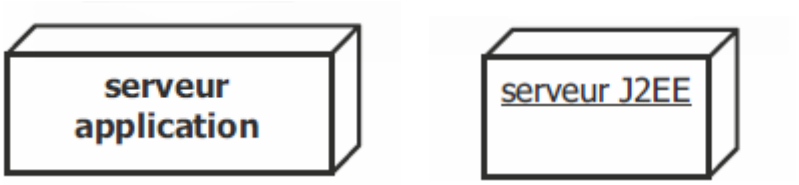


Figure – Exemple de représentation de nœud ou d'instance de nœud

Diagramme de Déploiment (DPL)

Noeud

- Chaque ressource matérielle est représentée par un nœud.
- *En général, cette ressource possède ses propres attributs* (capacité mémoire, capacité calculatoire, ...).
- *Exemple : calculateur, imprimante,…*

Diagramme de Déploiment (DPL)

Noeud

- Il est possible de représenter des nœuds spécialisés:
  - Unité de traitement:
    - Unité physique disposant de capacité de traitement sur laquelle des artefacts peuvent être déployés.
    - Caractérisé par le mot-clé « device ».
  - Environnement d'exécution:
    - Représente un environnement d'exécution particulier sur lequel certains artefacts peuvent être exécutés.
    - Caractérisé par le mot-clé « executionEnvironment ».

Diagramme de Déploiment (DPL)

Artefact

- Un artefact correspond à un élément **concret** existant dans le monde réel:
  - Fichier,
  - Exécutable,
  - Script,
  - table d'une base de données…

- Spécification d'un élément physique qui est utilisé ou produit par le processus de développement du logiciel ou par le déploiement du système.

- Un artefact peut être relié à d'autres artefacts par des liens de dépendance.

Diagramme de Déploiment (DPL)

Artefact

- *Formalisme et exemple:*
  - Représenté par un rectangle caractérisé par le mot-clé «artifact» et/ou une icône particulière dans le coin droit du rectangle.

Représentation d'un artefact

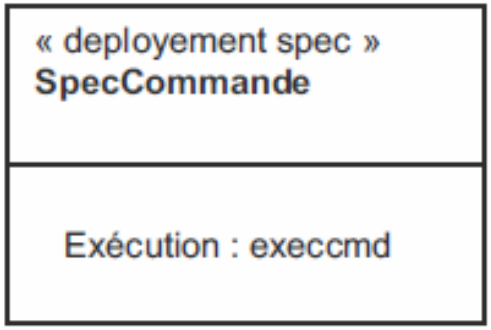
Diagramme de Déploiment (DPL)

Spécification de déploiement

- Elle permet de préciser les conditions de déploiement de l'artefact sur le noeud sur lequel il va être implanté.

- Une spécification de déploiement peut être associée à chaque artefact.

- *Formalisme et exemple:*
  - Une spécification de déploiement se représente par un rectangle avec le mot-clé « deployment spec ».
  - Exemple:

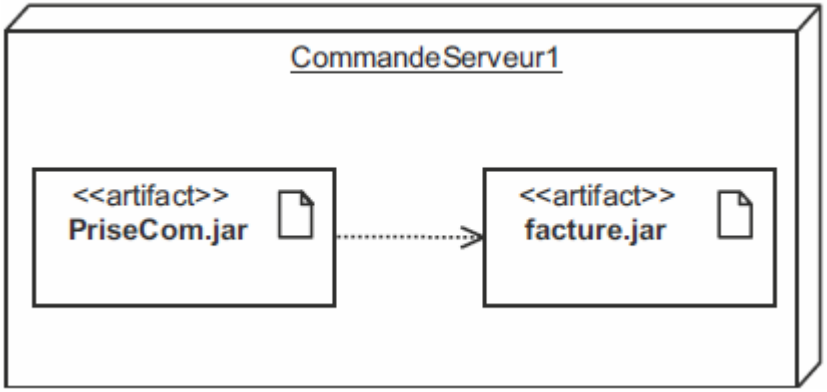


Exemple d'un artefact avec spécification de déploiement

Diagramme de Déploiment (DPL)

Liens entre un artefact et les autres éléments du diagramme

- Deux manières de représenter le lien entre un artefact et son nœud d'appartenance :
  1. Représentation inclusive: un artefact est représenté à l'intérieur du nœud auquel il se situe physiquement.



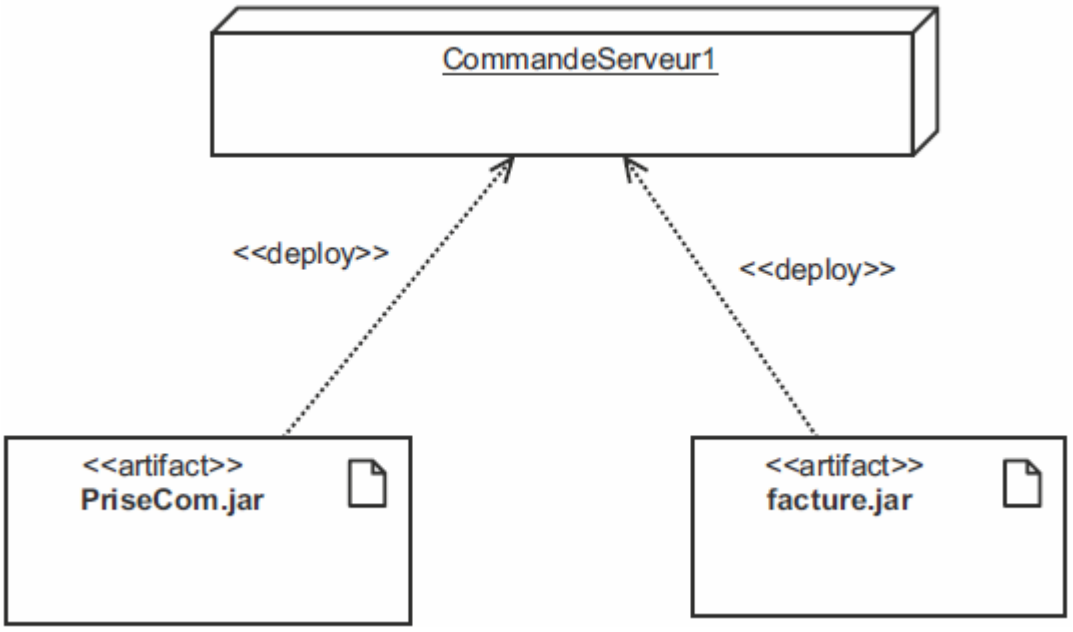
Exemple de représentation inclusive d'artefact

Diagramme de Déploiment (DPL)

Liens entre un artefact et les autres éléments du diagramme

- Deux manières de représenter le lien entre un artefact et son nœud d'appartenance :

  2. Représentation avec un lien de dépendance typé « deploy » : l'artefact est représenté à l'extérieur du noeud auquel il appartient avec un lien de dépendance entre l'artefact et le noeud typé avec le mot-clé « deploy ».

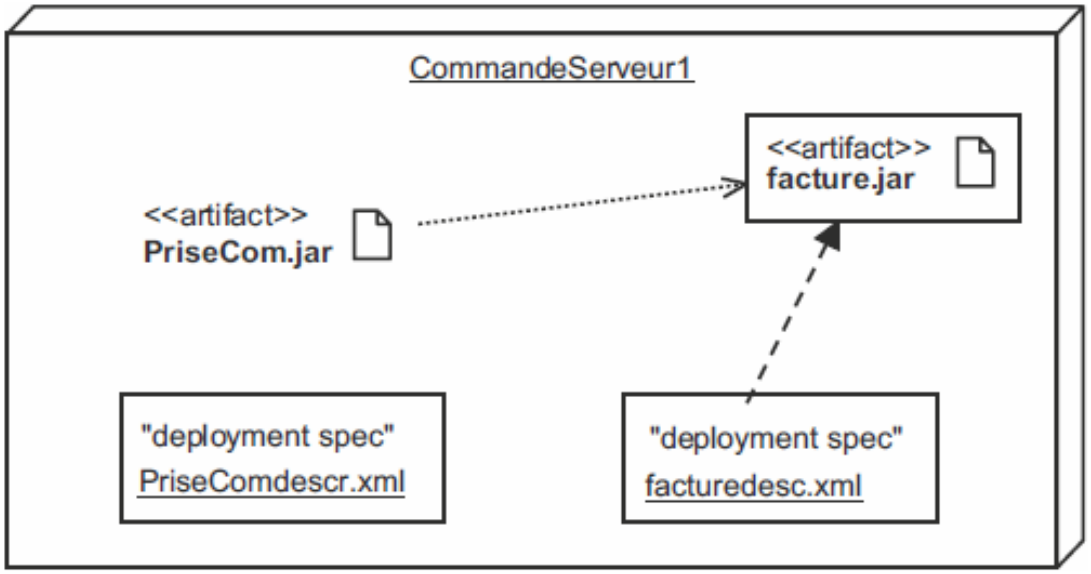


Exemple de représentation d'artefact avec lien de dépendance

Diagramme de Déploiment (DPL)

Représentation et exemples

- Exemple 1:



Exemple de représentation d'un diagramme de déploiement

Diagramme de Déploiment (DPL)

- Exemple de diagramme de déploiement comportant quatre nœuds (1)

Implémentation d'une architecture J2EE

Plusieurs composants sont déployés.

- Un serveur web où se trouvent les éléments statiques du site dans une archive : images, feuilles de style, pages html (static.zip).

- Un serveur d'application « front » sur le lequel est déployée l'archive « front.ear » composée de:
  - l'application web « front.war »
  - «clientejb.jar »
  - « commun.jar » (classes communes aux deux serveurs d'application).

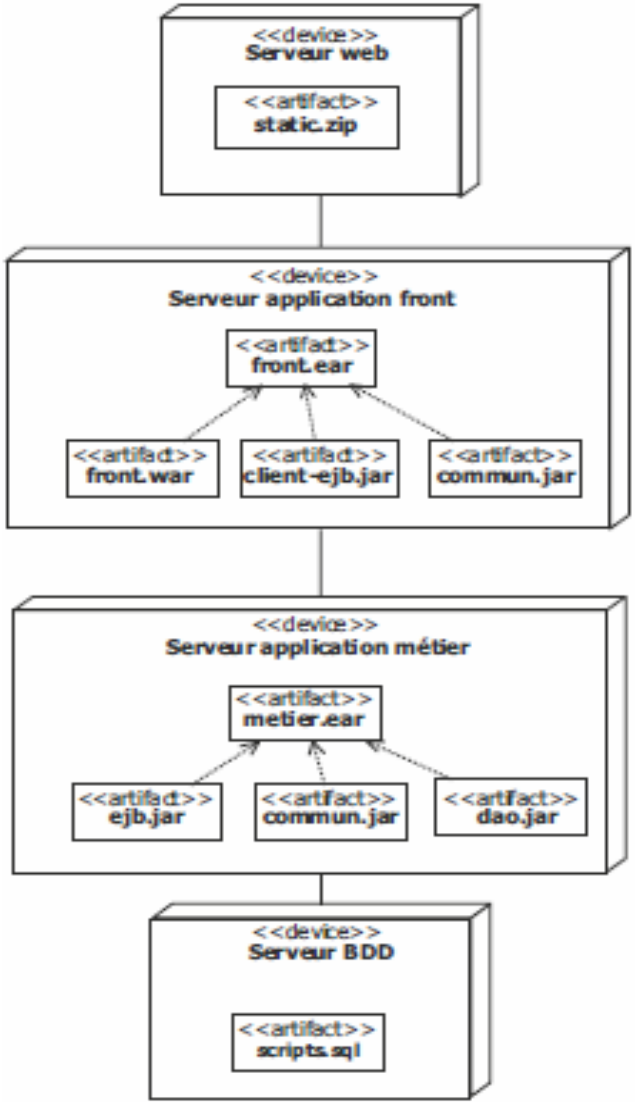


Diagramme de Déploiment (DPL)

- Exemple de diagramme de déploiement comportant quatre nœuds (2)

Implémentation d'une architecture J2EE

Plusieurs composants sont déployés.

- Un serveur d'application métier sur lequel sont déployés les composants : «ejb.jar».

Ils sont packagés dans l'archive « metier.ear ».

Deux autres archives sont nécessaires au fonctionnement des EJB :

- « dao.jar » (classes qui permettent l'accès à la base de données)
- « commun.jar » (classes communes aux deux serveurs d'application).

- Un serveur BDD (base de données) sur lequel sont stockées des procédures stockées PL/SQL : « scripts.sql ».

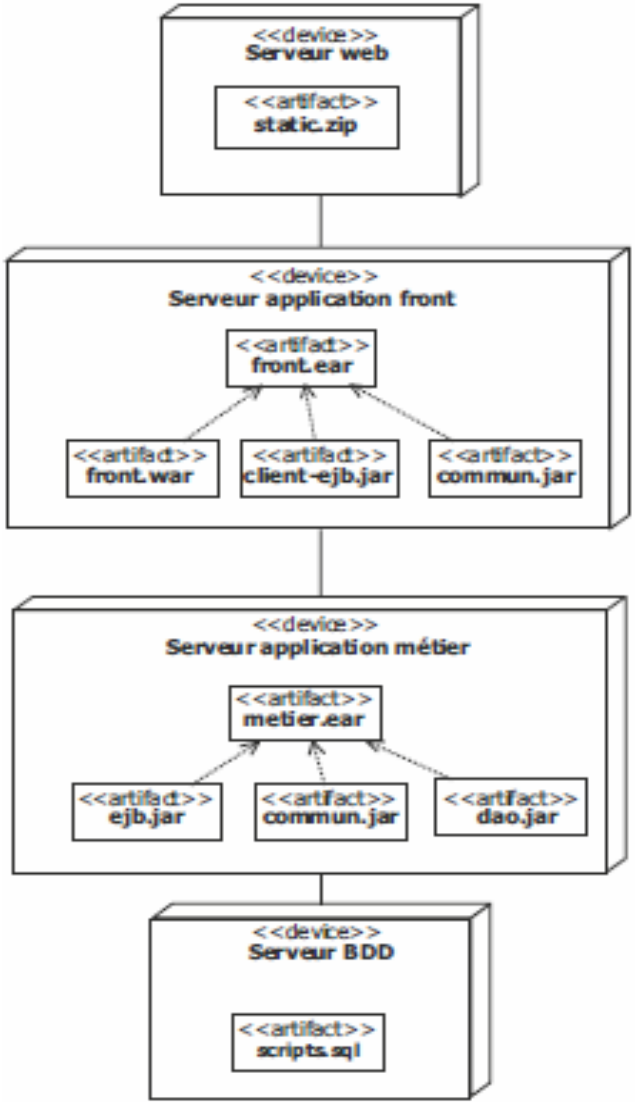


DIAGRAMME DE PAQUETAGE (DPA)

PAQUETAGE

- Un paquetage regroupe des éléments de la modélisation appelés aussi membres, portant sur un sous-ensemble du système.

- Le découpage en paquet doit traduire une logique du système à construire qui corresponde à des espaces de nommage **homogènes**

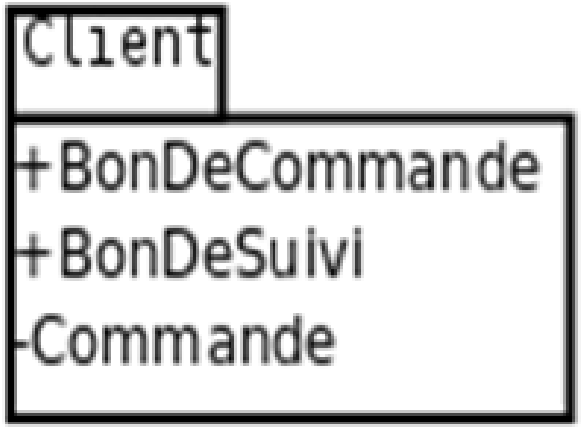
- Il peut contenir tout type d'élément de modèle :
  - classes,
  - cas d'utilisation,
  - interfaces,
  - diagrammes…
  - même des paquetages imbriqués (décomposition hiérarchique)

Diagramme de paquetage (DPA)

Paquetage

- Tout élément n'appartient qu'à un seul paquetage (cf. Méta-Modèl)

- Les éléments d'un paquetage peuvent avoir une visibilité déclarée soit de type public (+) soit privé (-).

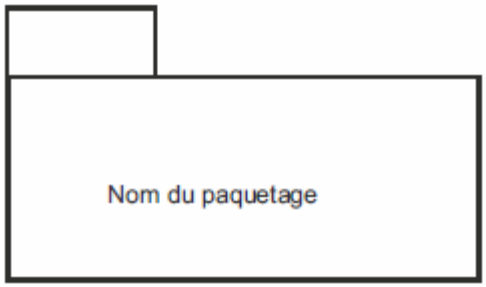


- Les paquetages constituent un mécanisme de gestion important des problèmes de **grande taille**.

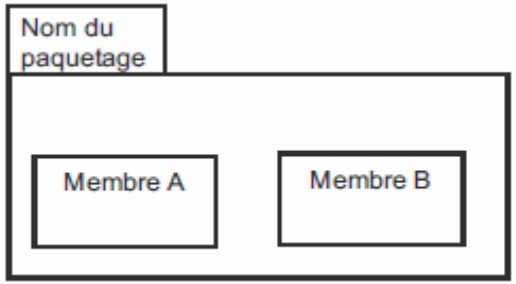
DIAGRAMME DE PAQUETAGE (DPA)

PAQUETAGE

- Formalisme:
  - **Représentation globale:** Le nom du paquetage se trouve à l'intérieur du grand rectangle.



  - **Représentation détaillée:** Les membres du paquetage sont représentés et le nom du paquetage d'ensemble s'inscrit dans le petit rectangle.



  - **Représentation éclatée:** Les membres du paquetage sont reliés par un lien connecté au paquetage par le symbole ⊕

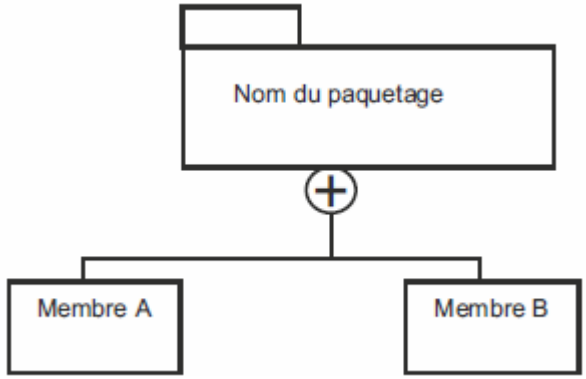


DIAGRAMME DE PAQUETAGE (DPA)

PAQUETAGE

- Exemple de représentation éclatée:

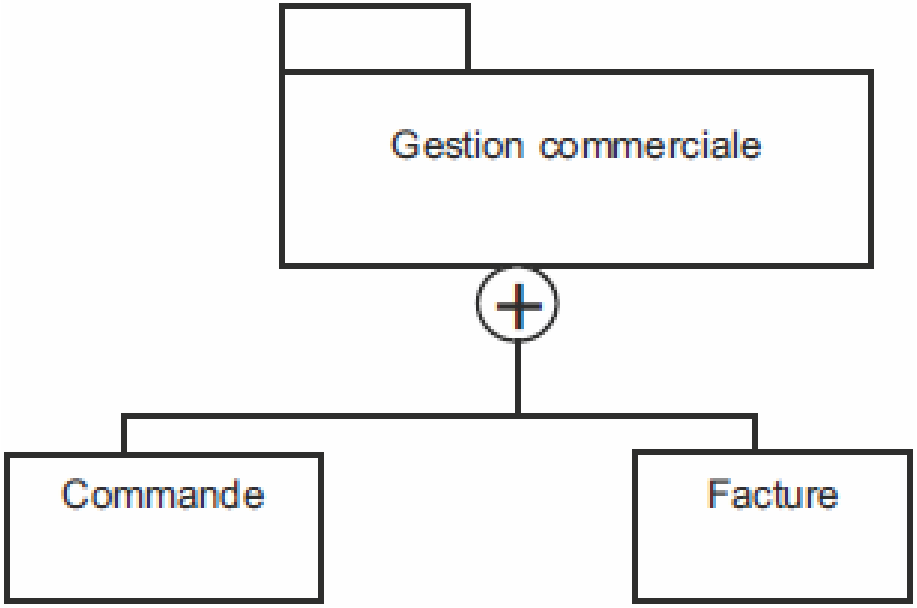


Diagramme de paquetage (DPA)

Dépendance entre Paquetages (1)

- Peut être qualifiée par un niveau de visibilité qui est soit public soit privé.
- Par défaut le type de visibilité est public.
- À chaque type de visibilité est associé un lien de dépendance.
- Les deux types de dépendances entre paquetages sont :
  1. « **import** » :
     - permet pour un paquetage donné, d'importer l'espace de nommage d'un autre paquetage.
     - Tous les membres du paquetage donné ont accès à tous les noms des membres du paquetage importé sans avoir à utiliser explicitement le nom du paquetage concerné.
     - Ce type de dépendance correspond à un lien ayant une visibilité « public ».

Diagramme de paquetage (DPA)

Dépendance entre Paquetages (2)

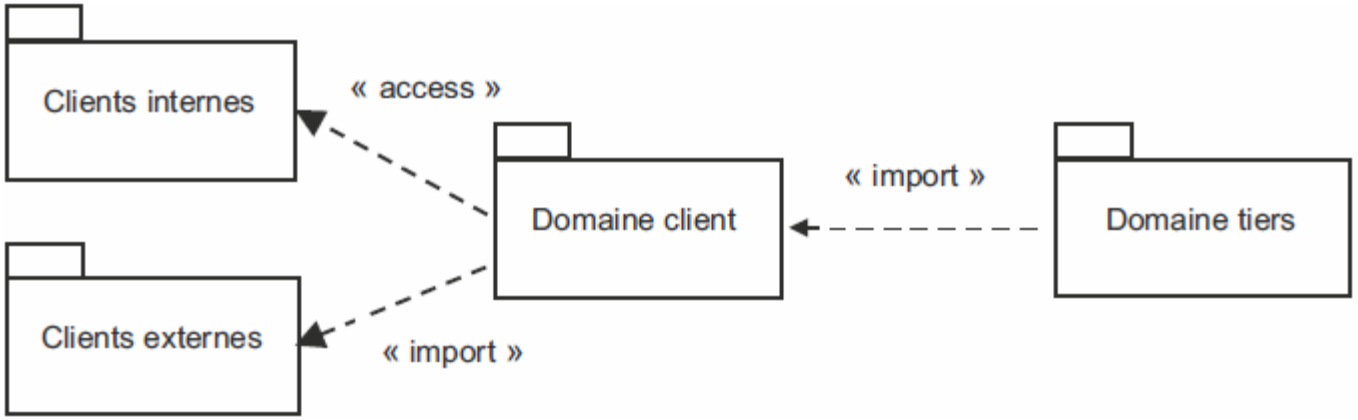
- Peut être qualifiée par un niveau de visibilité qui est soit public soit privé.
- Par défaut le type de visibilité est public.
- À chaque type de visibilité est associé un lien de dépendance.
- Les deux types de dépendances entre paquetages sont :

2. « **access** » :
   - Permet pour un paquetage donné, d'avoir accès à l'espace de nommage d'un paquetage cible.
   - L'espace de nommage n'est pas importé et ne peut être transmis à d'autres paquetages par **transitivité**.
   - Ce type de dépendance correspond à un lien ayant une visibilité « privé ».

Diagramme de paquetage (DPA)

Dépendance entre Paquetages (3)

- Exemple:



- Dans cet exemple, les éléments de 'Clients externes' sont importés dans 'Domaine client' et ensuite dans 'Domaine tiers'.

- Les éléments de 'Clients internes' sont seulement accessibles par le paquetage 'Domaine client' et pas à partir du paquetage 'Domaine tiers'.

DIAGRAMME DE PAQUETAGE (DPA)

FUSION ENTRE 2 PAQUETAGES :

- Permet de fusionner 2 paquetages et obtenir un paquetage contenant la fusion des 2 paquetages d'origine.

- Trois rôles à distinguer :
  - le paquetage à fusionner (entrant dans la fusion, mais préservé après la fusion) ;
  - le paquetage recevant (paquetage d'origine avant la fusion, mais non conservé après la fusion) ;
  - le paquetage résultat (paquetage contenant le résultat de la fusion et écrasant le contenu du paquetage d'origine).

- Le lien de dépendance comporte dans ce cas le mot-clé 'merge'

DIAGRAMME DE PAQUETAGE (DPA)

FUSION ENTRE 2 PAQUETAGES :

- Un exemple type de fusion entre deux paquetages

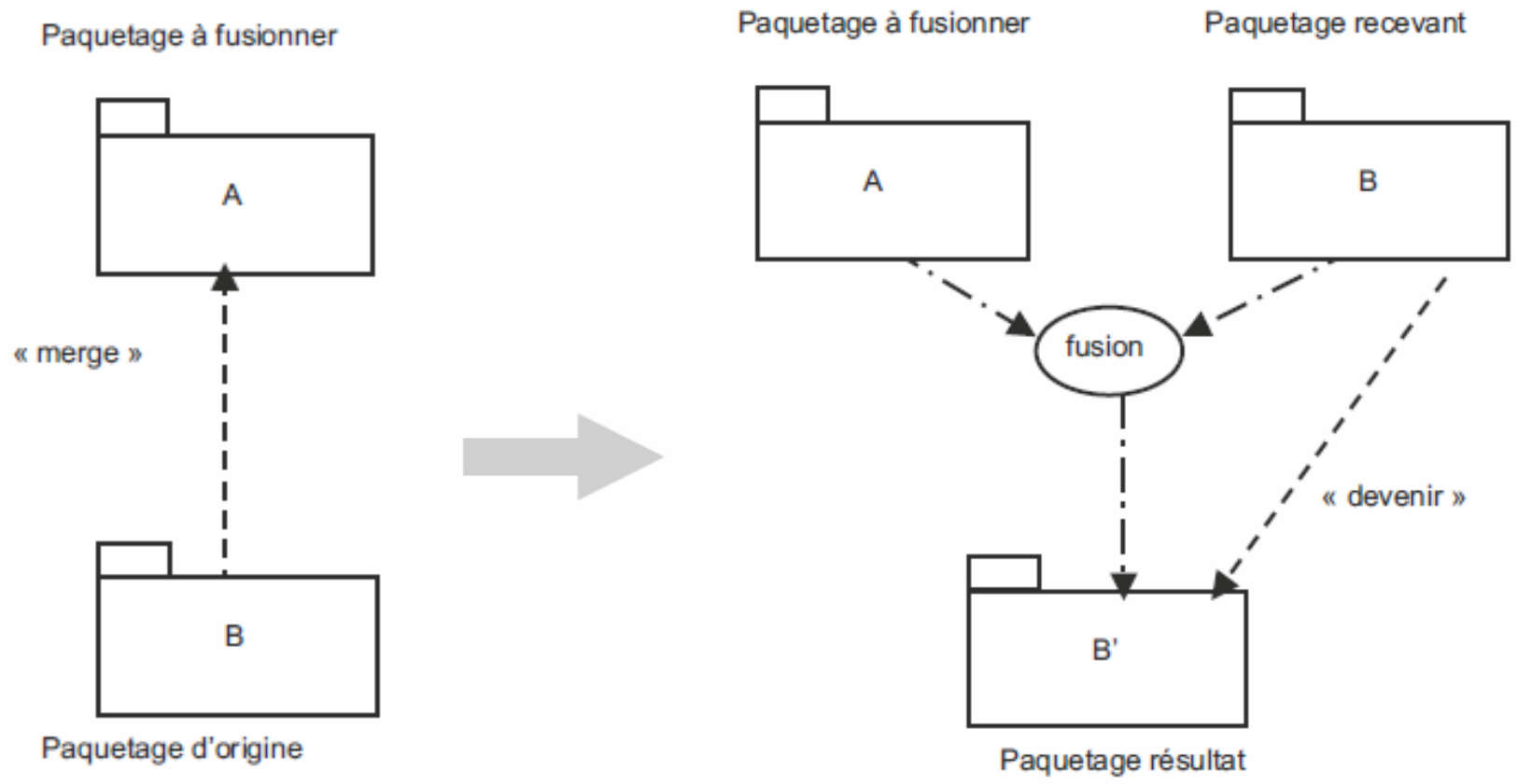


DIAGRAMME DE STRUCTURE COMPOSITE (DSC)

- Diagramme qui affiche la structure interne d'un classificateur, y compris ses points d'interaction avec d'autres parties du système.

- Permet de décrire des collaborations d'instances (de classes, de composants…) constituant des fonctions particulières du système à développer.

COLLABORATION

- Représente un assemblage de rôles d'éléments qui interagissent en vue de réaliser une fonction donnée.

- Deux manières de représenter une collaboration :
  - représentation par une collaboration de rôles,
  - représentation par une structure composite.

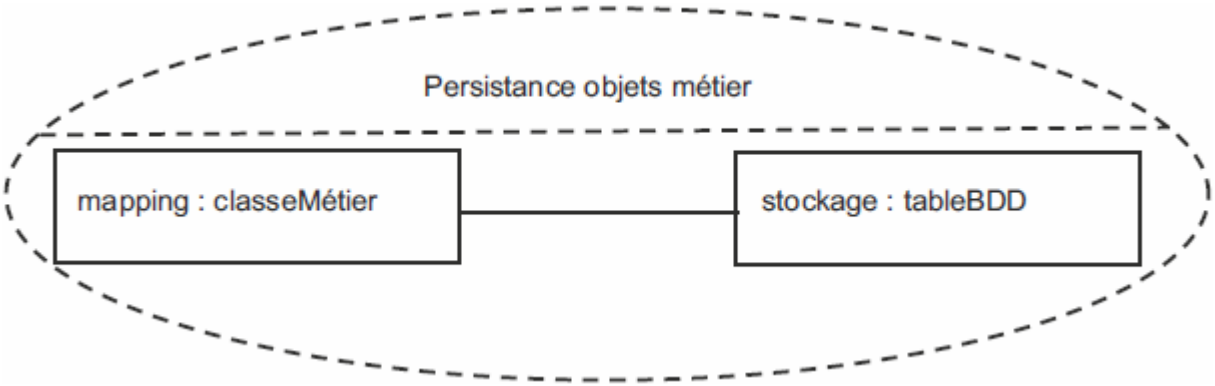
DIAGRAMME DE STRUCTURE COMPOSITE (DSC)

COLLABORATION

- Représentation par une collaboration de rôles:

  Une collaboration est formalisée par une ellipse en pointillé dans laquelle on fait figurer les rôles des éléments qui interagissent en vue de réaliser la fonction souhaitée.

- Exemple:



- La fonction 'Persistance objets métier' résulte d'une collaboration entre deux rôles d'éléments :
  - mapping : classeMétier,
  - stockage : tableBDD.

DIAGRAMME DE STRUCTURE COMPOSITE (DSC)

COLLABORATION

- Représentation par un *diagramme de structure composite*:
  - Permet de montrer plus explicitement les éléments de la collaboration :
    - la collaboration représentée par une ellipse en pointillé ;
    - les éléments participant à la collaboration (classe, composant…) représentés à l'extérieur de la collaboration ;
    - les rôles considérés dans chaque participation représentés sur les liens entre les éléments participants et la collaboration.

DIAGRAMME DE STRUCTURE COMPOSITE (DSC)

COLLABORATION

- Représentation par un diagramme de structure composite:
  - Exemple:

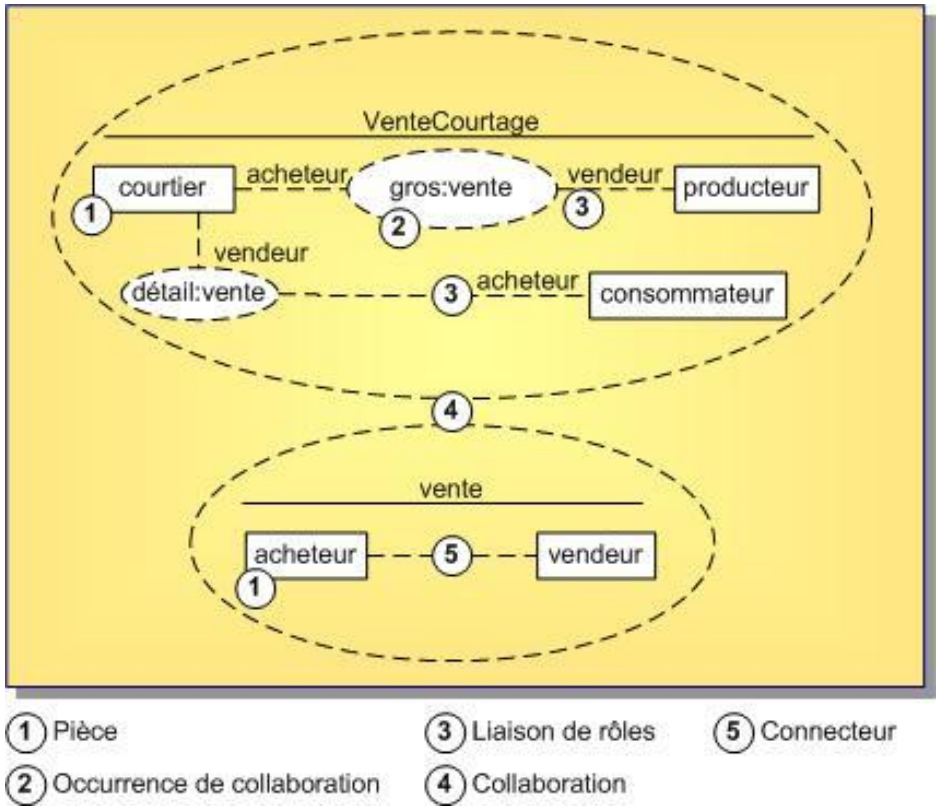
  - la fonction 'Persistance objets métier' résulte d'une collaboration entre la classe 'ClasseMétier' considérée suivant le rôle mapping et la classe 'TableBDD' considérée suivant le rôle stockage.

  - Cette représentation permet aussi de préciser les seuls attributs des classes participantes qui sont considérés suivant les rôles pris en compte.

DIAGRAMME DE STRUCTURE COMPOSITE (DSC)

COLLABORATION

- Exemple



- Les diagrammes de structures composites prennent en charge la notation prenant en charge les interfaces fournies et requises.
- Vous pouvez masquer ou afficher les interfaces sur le diagramme, selon vos besoins.
-