Module : Méthodes d'analyse et de conception
Partie I : UML
Chapitre 3 : La notation UML des concepts OO

Pr. ABDELMAJID DARGHAM

ENSAK

Génie Informatique

2ème Année Cycle Ingénieur / S3

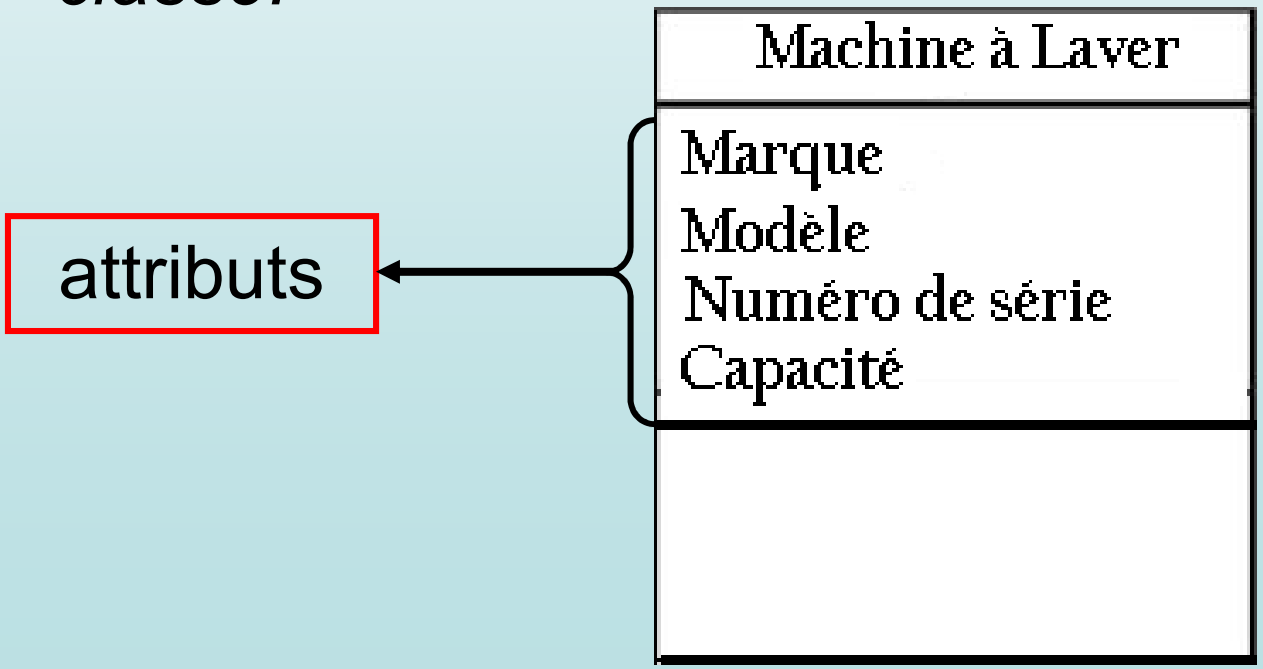
Année universitaire : 2018/2019

Sommaire

- Les attributs
- Les méthodes
- Responsabilités, contraintes et notes
- Les associations
- L’agrégation
- L’héritage

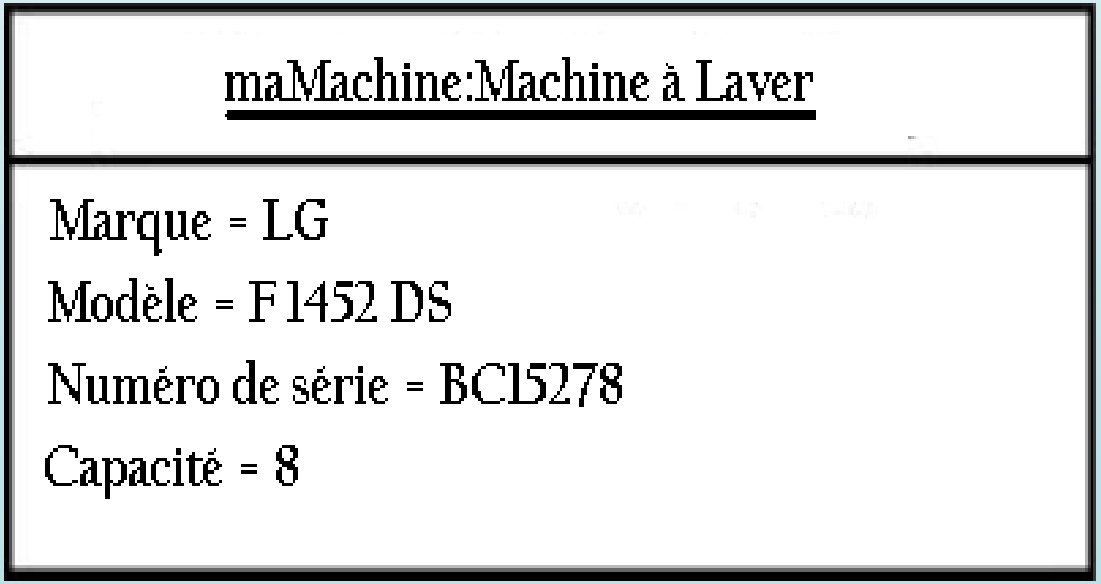
Les attributs

- Attributs d'un objet :
  - *Un attribut d'un objet est une propriété qui caractérise l'état d'un objet.*
  - *Les attributs d'un objet sont les attributs de sa classe.*



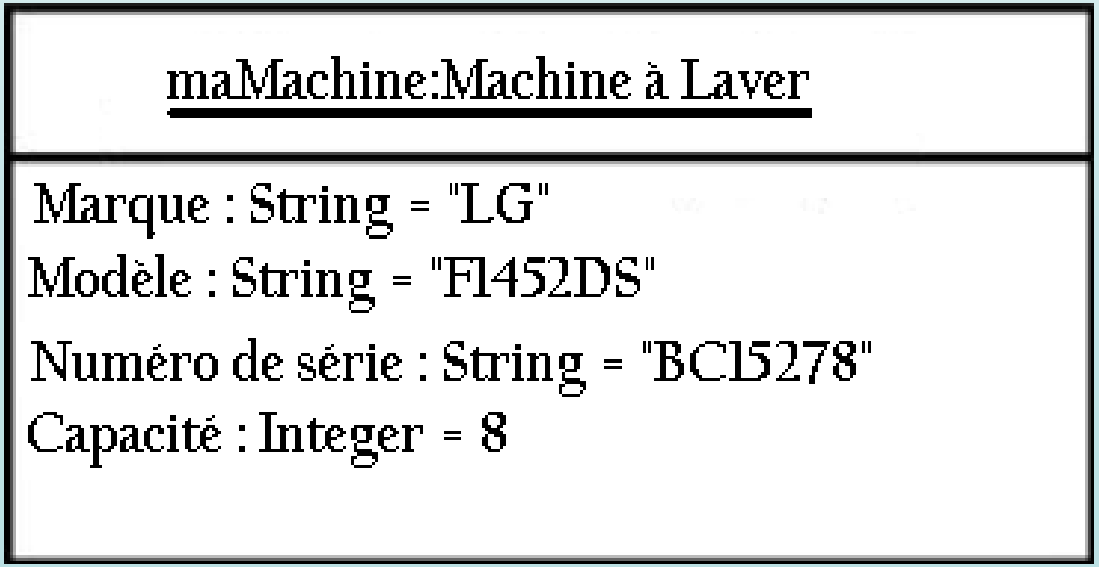
Les attributs

- Attributs d'un objet :
  - *Un objet possède une valeur spécifique pour chacun des attributs de sa classe.*
  - *L'attribut d'un objet a un nom, un type et une valeur initiale.*



Les attributs

- Quelques types possibles d'attributs :
  - *Boolean : Vrai / Faux.*
  - *String : chaîne de caractères entre "...".*
  - *Integer : valeur numérique entière.*
  - *Floating-point : valeur numérique réelle.*

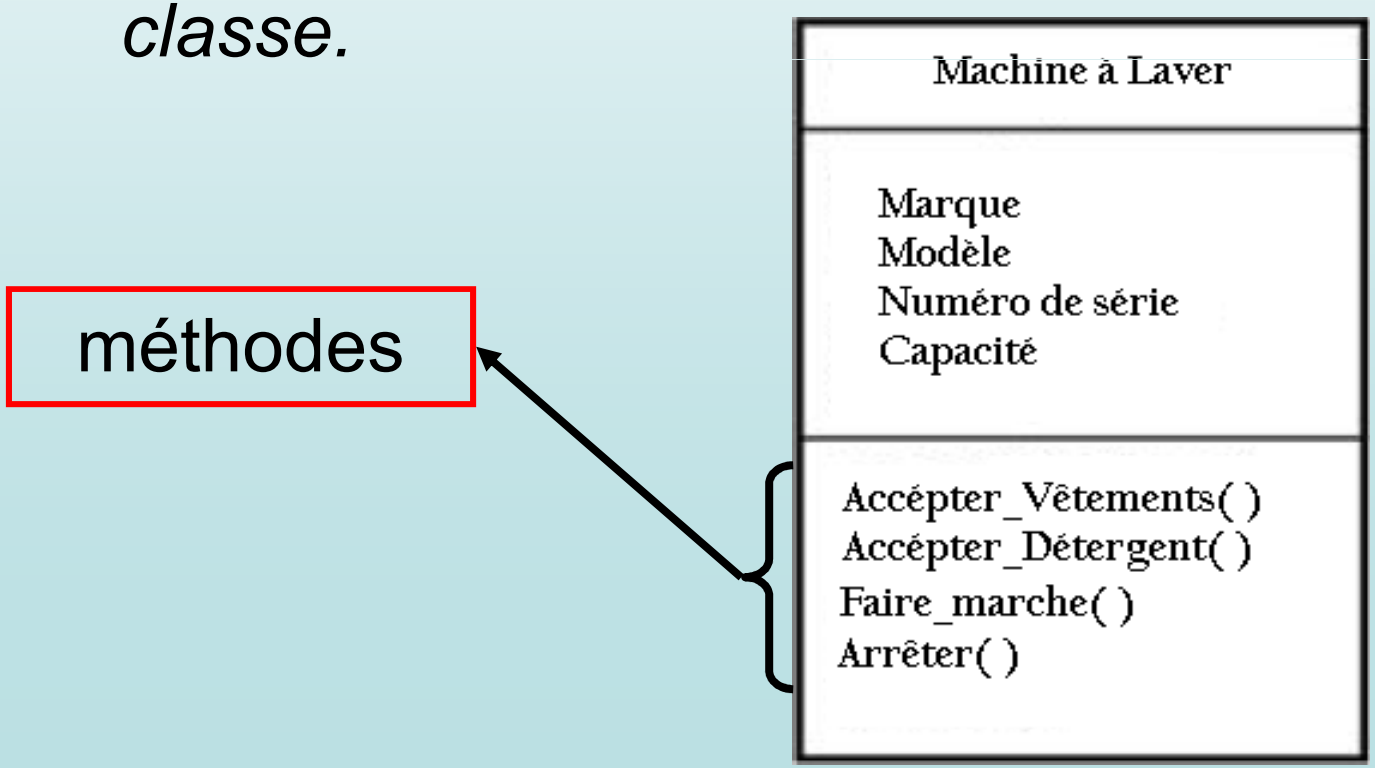


Les attributs

- Syntaxe de déclaration d'un attribut :
  - *Visibilité Nom-attribut : Type = Valeur-initiale*
- Visibilité des attributs :
  - *Un attribut peut être :*
    - *Privé : - (moins accessible)*
    - *Protégé : # (un peu accessible)*
    - *Publique : + (plus accessible)*
  - *Par défaut, un attribut est privé (pour être conforme avec les concepts objet).*

Les méthodes

- Méthodes d'un objet :
  - *Une méthode d'un objet est une propriété qui caractérise le comportement d'un objet.*
  - *Les méthodes d'un objet sont celles de sa classe.*



Les méthodes

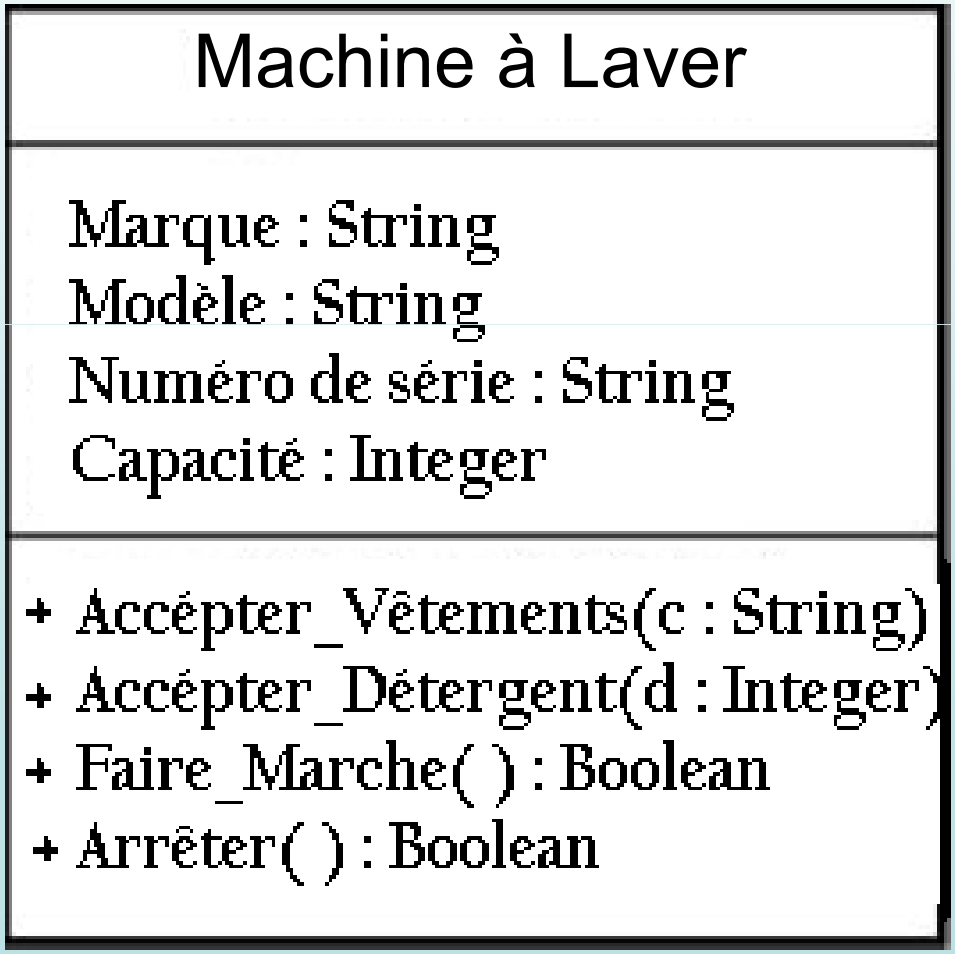
- Méthodes d'un objet :
  - *Les méthodes d'un objet se rattachent à la classe de cet objet.*
  - *Les méthodes d'une classe sont partagées par tous les objets instances de cette classe.*
  - *Une méthode est une fonction ayant :*
    - *Un nom;*
    - *Un type de retour;*
    - *Une liste d'arguments;*
    - *Une visibilité.*

Les méthodes

- Syntaxe de déclaration d'une méthode :
  - *Visibilité Nom-méthode(Liste-d-arguments) : Type-retour*
  - *La liste des arguments d'une méthode peut être vide.*
  - *Un argument d'une méthode se déclare selon la syntaxe suivante :*
    - *Nom-argument : Type-argument = Valeur-par-défaut*

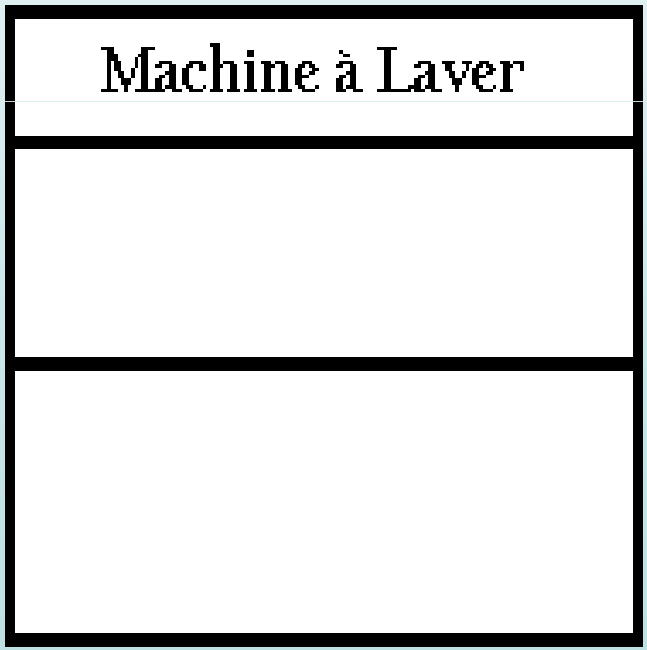
Les méthodes

- Exemple :



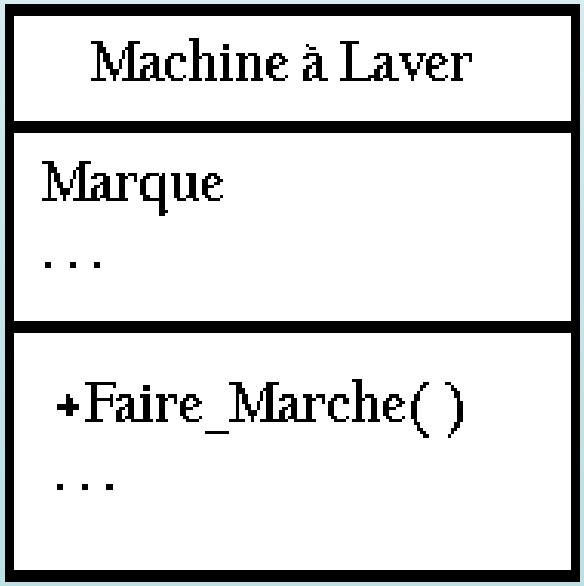
Les méthodes

- Visualisation des attributs et des méthodes :
  - *En pratique, on ne montre pas toujours tous les attributs et les méthodes d'une classe.*



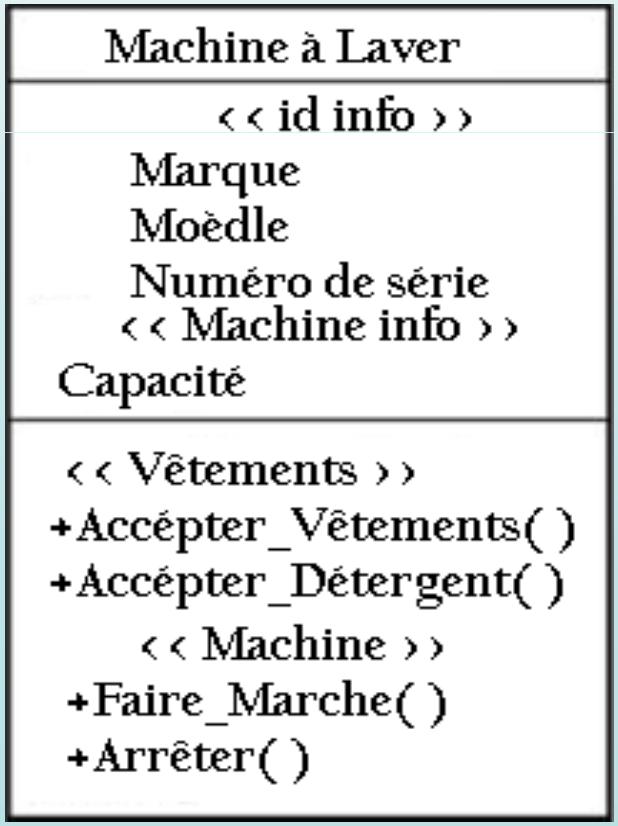
Les méthodes

- Visualisation des attributs et des méthodes :
  - *Il est possible d'utiliser les **points de suspension** "..." pour indiquer qu'il reste encore des attributs (et/ou méthodes) qui ne sont pas affichés.*



Les méthodes

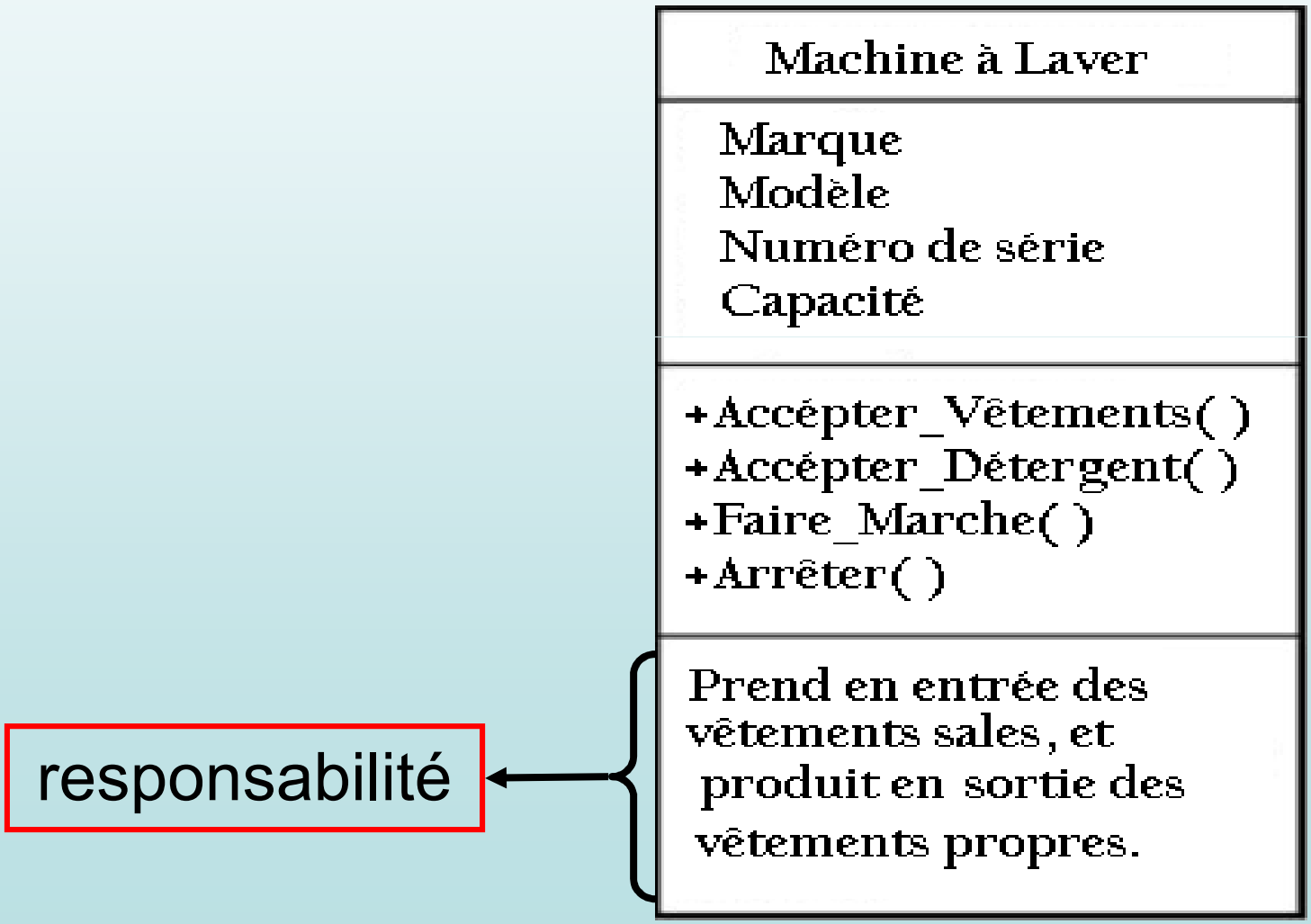
- Visualisation des attributs et des méthodes :
  - *On peut utiliser des **stéréotypes** pour organiser et documenter les attributs et les méthodes d'une classe.*



Responsabilités, contraintes et notes

- Responsabilité :
  - *La **responsabilité** est une description de ce que la classe peut faire.*
  - ***Avantages :***
    - ***Enrichir** la classe par plus d'informations.*
    - ***Éviter** les **ambiguïtés**.*
  - ***Notation :***
    - *La **responsabilité** s'ajoute dans un **4 ème compartiment** dans le diagramme de classe.*

Responsabilités, contraintes et notes

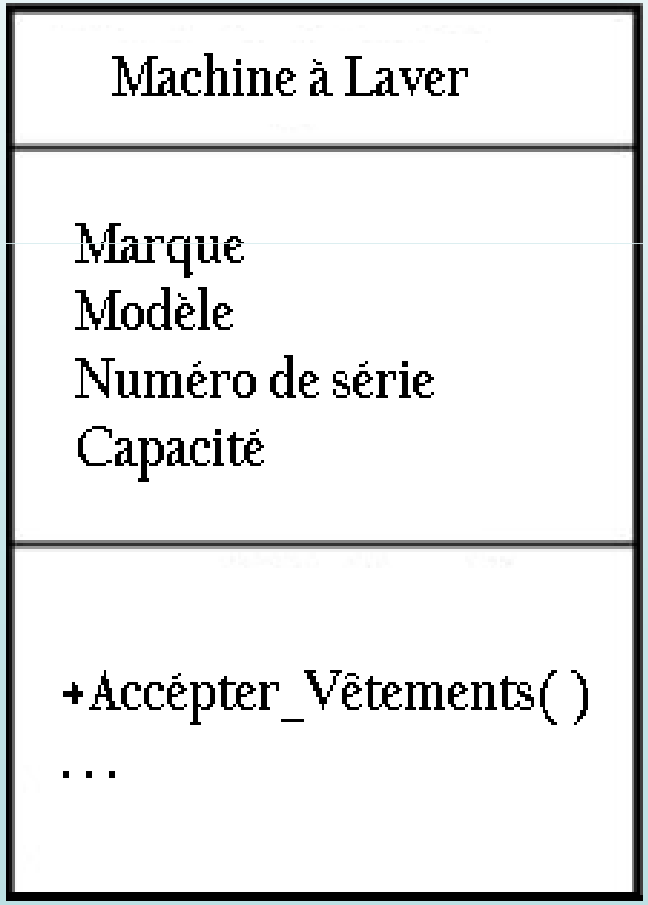


Responsabilités, contraintes et notes

- **Contraintes** :
  - *Une **contrainte** est une **condition** (ou une **règle**) que doit respecter un attribut d'une classe.*
  - *Elle se présente sous la forme d'un **texte libre** placé entre "**{...}**".*
  - *Les contraintes sont également exprimées en utilisant un langage spécial : **OCL** (**Object Constraint Language**).*

Responsabilités, contraintes et notes

- **Exemple :**



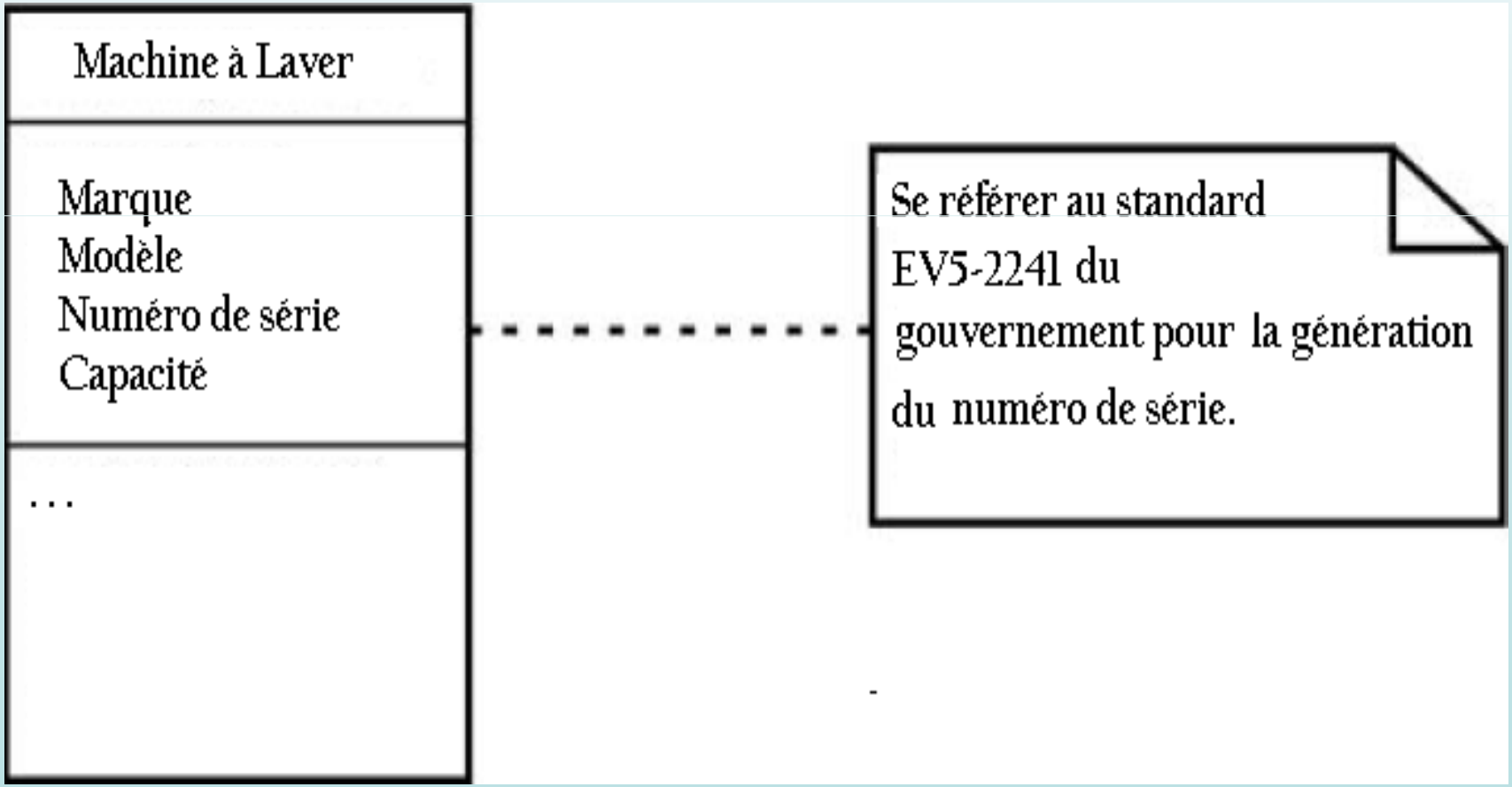
{Capacité = 8 or 12 or 16 Kg}

Responsabilités, contraintes et notes

- **Notes** :
  - *En plus des **attributs**, des **méthodes**, des **responsabilités** et des **contraintes**, on peut encore **ajouter plus d'informations** sur une classe sous la forme de **notes** qui sont jointes à celle-ci.*
  - *Une note se représente à l'aide d'un **rectangle vertical** dont le **coin supérieur droit** est **plié**.*
  - *Elle peut contenir aussi bien de **graphique** que de **texte**.*

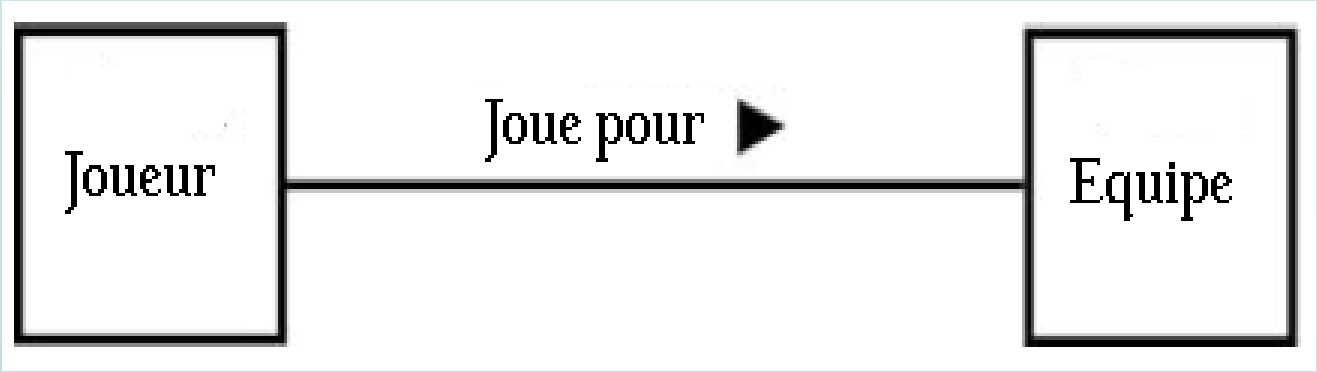
Responsabilités, contraintes et notes

- Exemple :



Les associations

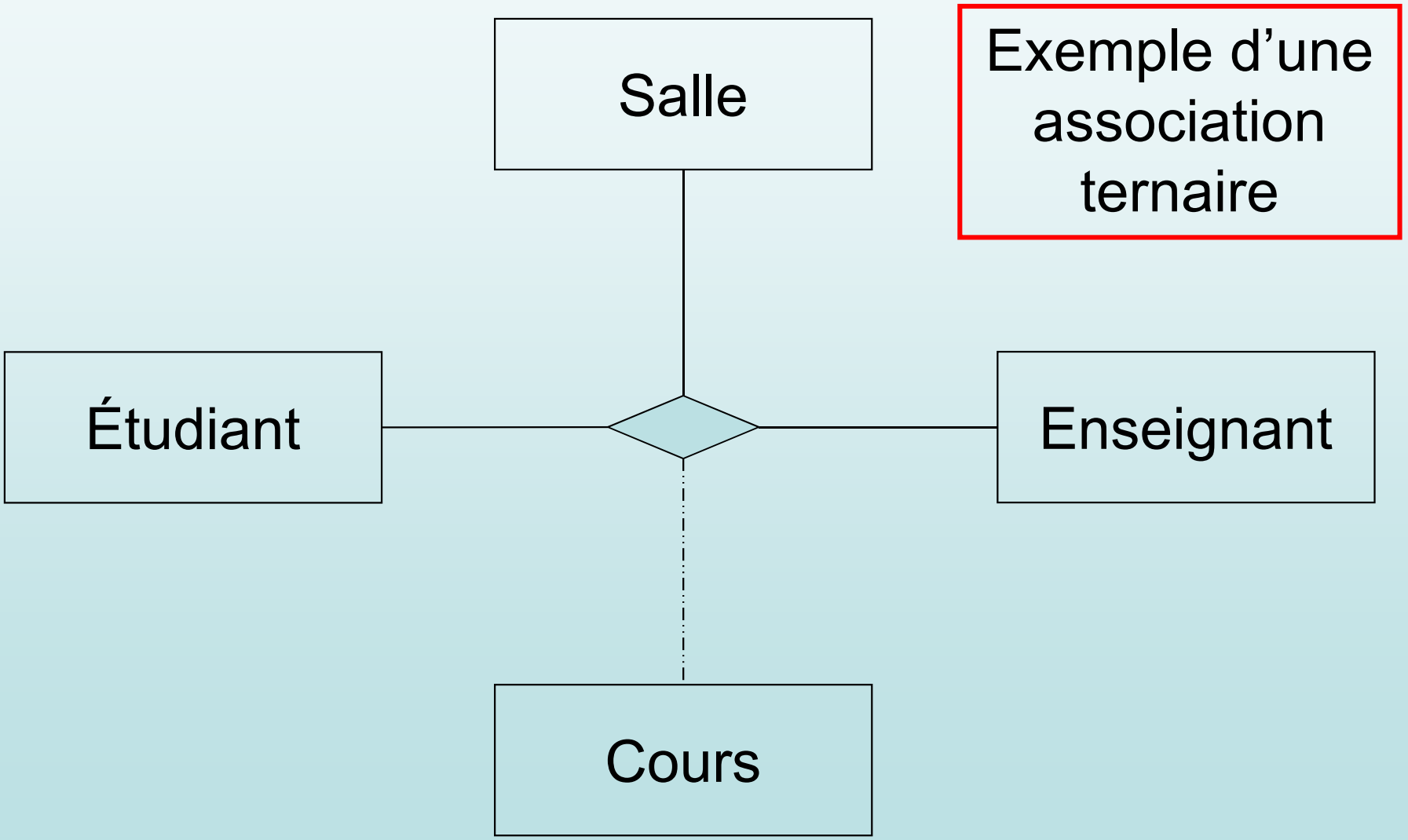
- Définition *:*
  - *Une association exprime une relation entre les objets de deux ou plusieurs classes.*
- Notation UML :
  - *Une association se représente à l'aide d'une ligne qui joigne les classes qui contribuent dans l'association.*



Les associations

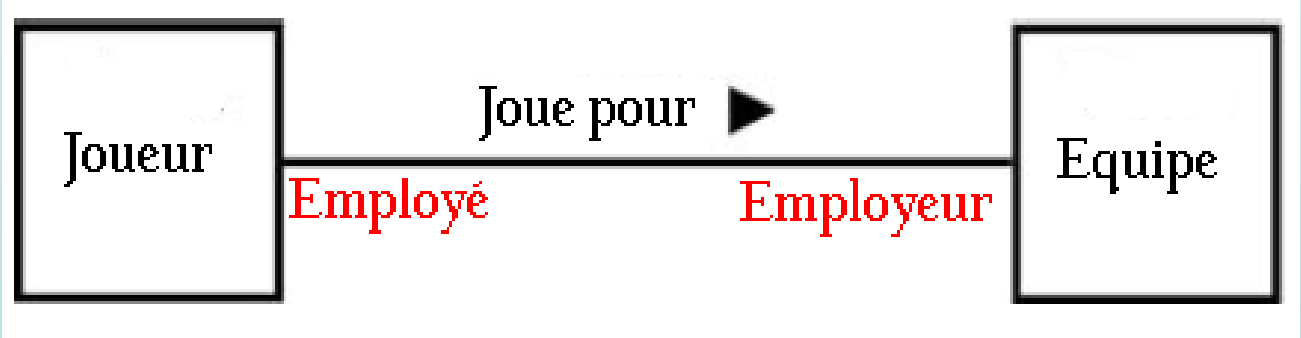
- Arité des associations *:*
  - *La plupart des associations sont binaires, car elles relient deux classes.*
  - *Il peut exister des associations ternaires, quaternaires, …etc.*
  - *Pour représenter ces associations, on utilise un losange sur lequel arrivent les différentes classes en association.*
- Exemple :
  - *Un cours est assuré par un enseignant, poursuivi par des étudiants et tiens lieu dans une salle.*

Les associations



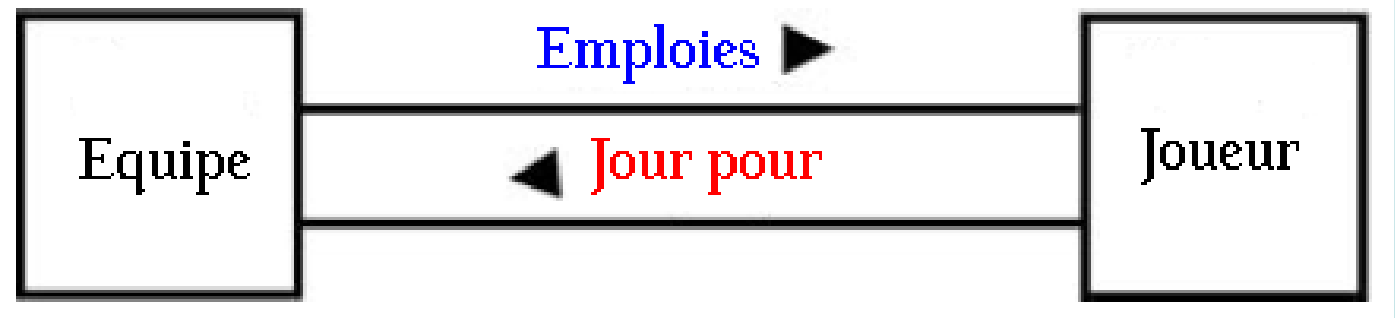
Les associations

- Rôles *:*
  - *Dans une association, chaque classe joue un rôle déterminé.*
  - *On peut visualiser les rôles dans le diagramme de classes en les écrivant près des extrémités de la ligne d'association.*



Les associations

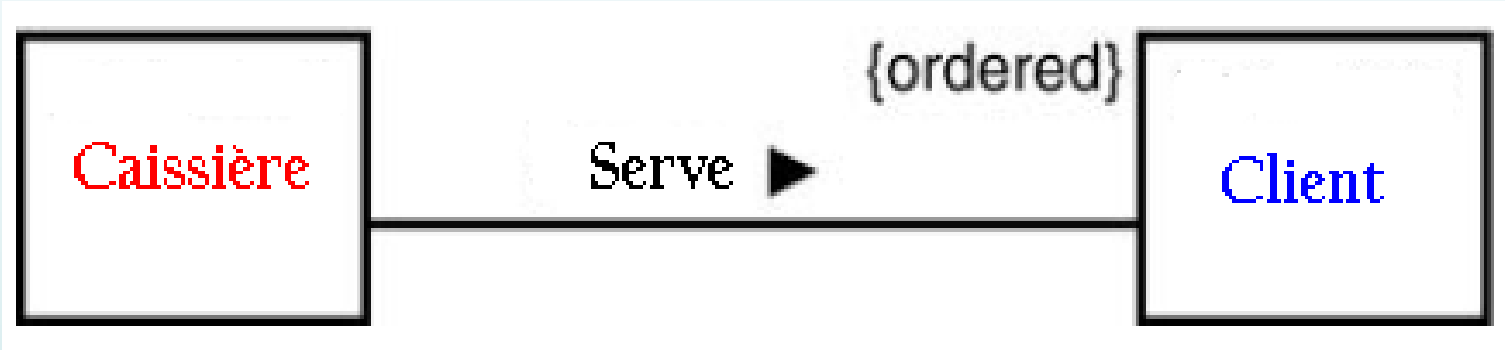
- Sens de lecture d'un rôle *:*
  - *On met un petit triangle pour préciser le sens de lecture du rôle dans une association.*

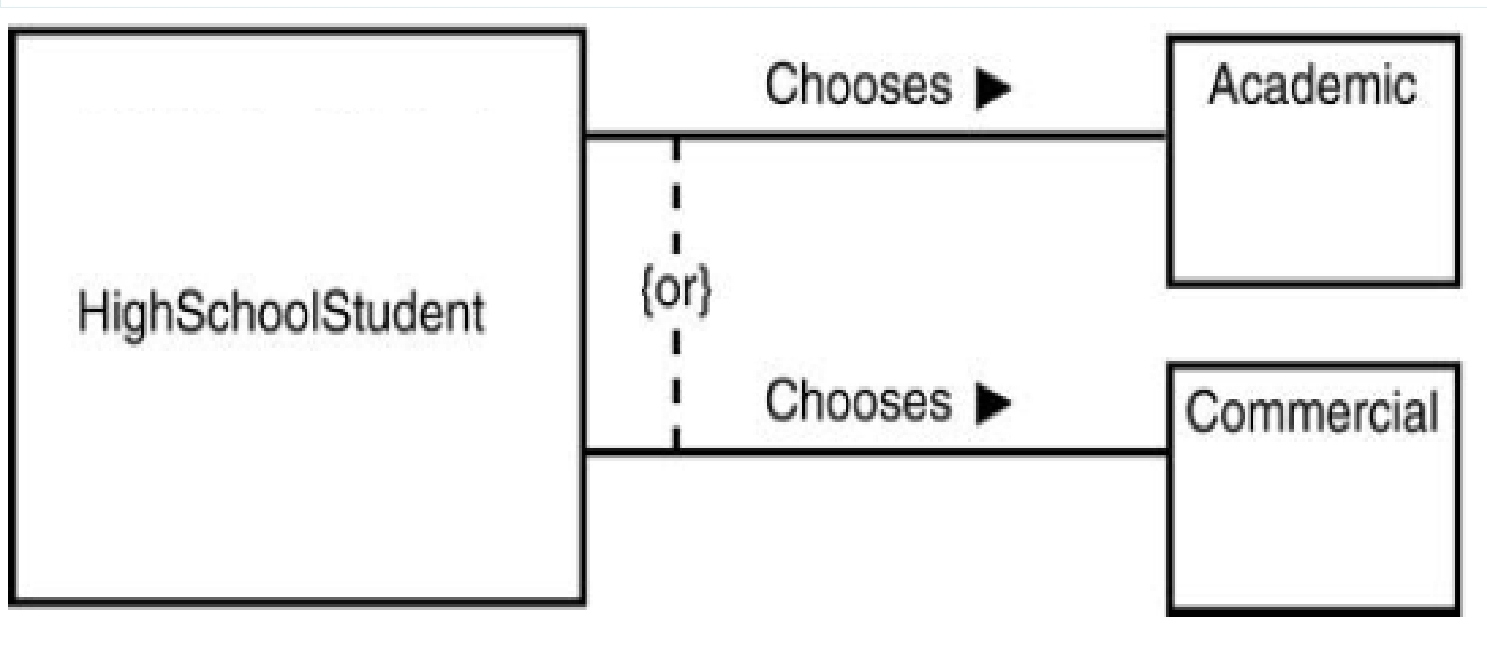


Les associations

- Contraintes sur les associations *:*
  - *Certaines associations sont soumises à des contraintes, c'est-à-dire qu'elles doivent respecter des règles.*
- Exemples :
  - *Un étudiant choisi des études académiques ou commerciales.*
  - *Une caissière de banque serve les clients dans l'ordre où ils apparaissent.*

Les associations

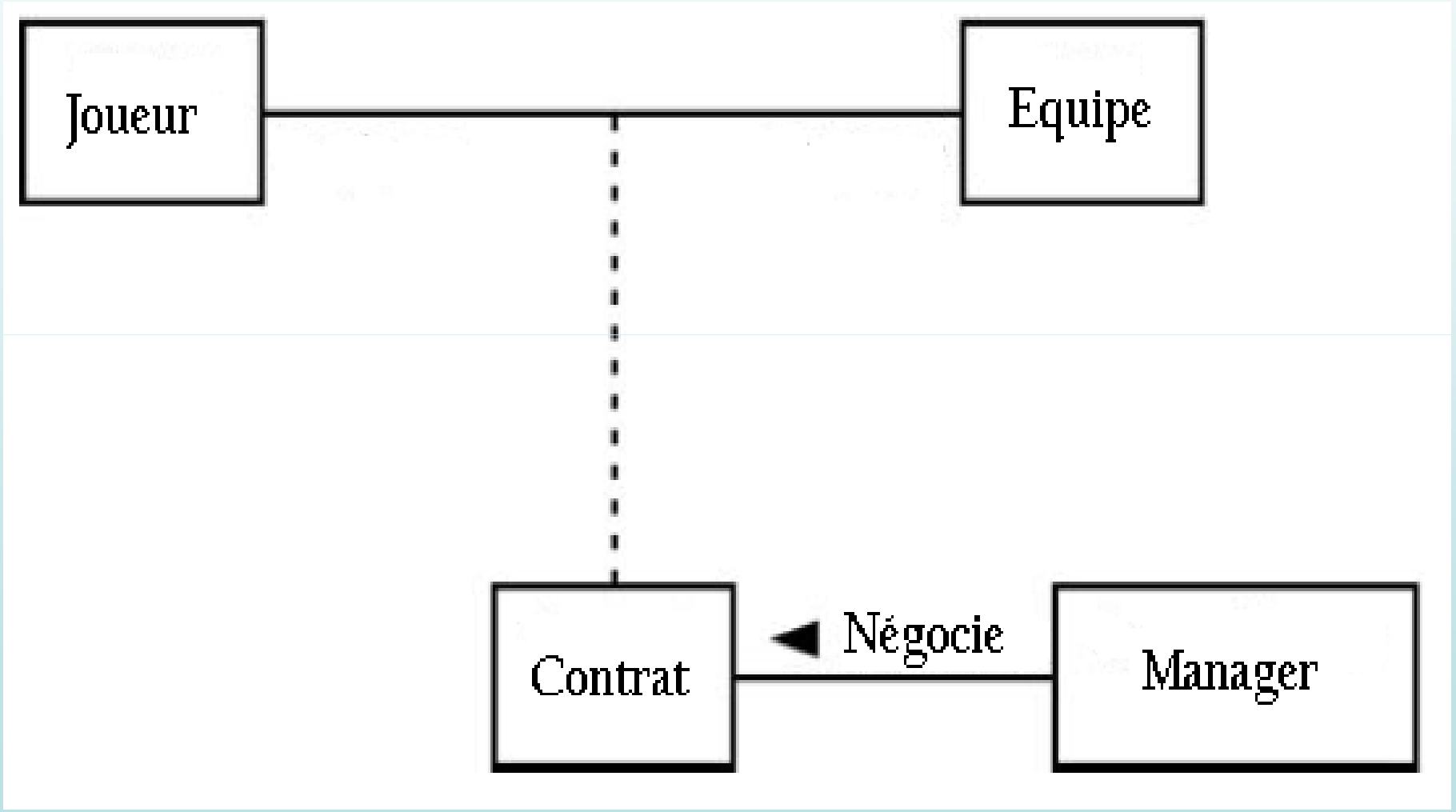




Les associations

- Les classes-associations *:*
  - *Tout comme une classe, une association peut avoir des attributs et des méthodes.*
  - *Dans ce cas, il s'agit en fait d'une classe-association.*
  - *La notation UML utilise une ligne pointillée pour attacher une classe à une association.*
  - *Une classe-association peut avoir des associations avec d'autres classes.*

Les associations



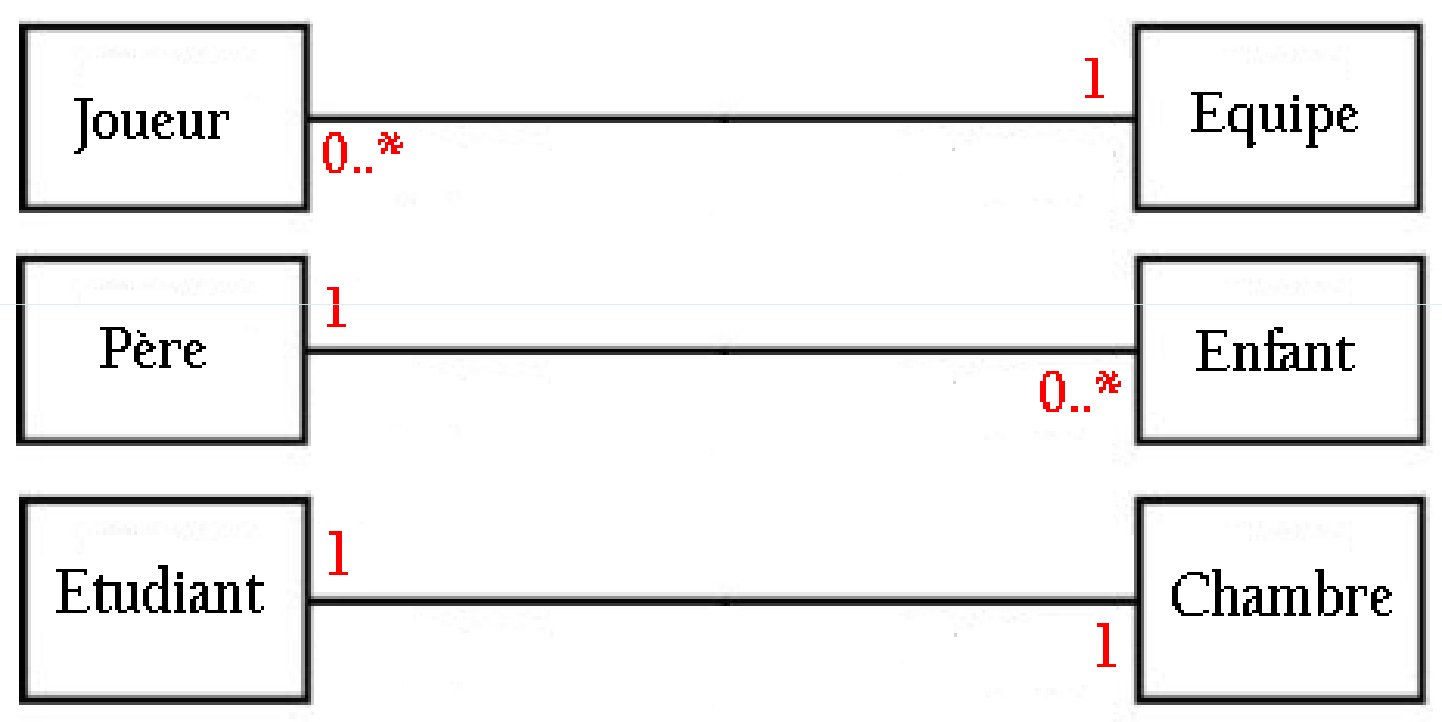
Les associations

- Multiplicité des associations *:*
  - *Chaque rôle d'une association porte une indication de multiplicité qui montre combien d'objets de la classe considérée peuvent être liés à un objet de l'autre classe.*
  - *La multiplicité est une information portée par le rôle, sous la forme d'une expression entière bornée.*
  - *Par exemple, un joueur ne peut joué que pour le compte d'une seule équipe, alors qu'une équipe emploies plusieurs joueurs.*

Les associations

- Multiplicité conventionnelles dans UML *:*
  - *1 : Un et un seul.*
  - *0..1 : zéro ou un.*
  - *M..N : de M à N.*
  - ** : zéro ou plusieurs.*
  - *0..* : zéro ou plusieurs.*
  - *1..* : un ou plusieurs.*

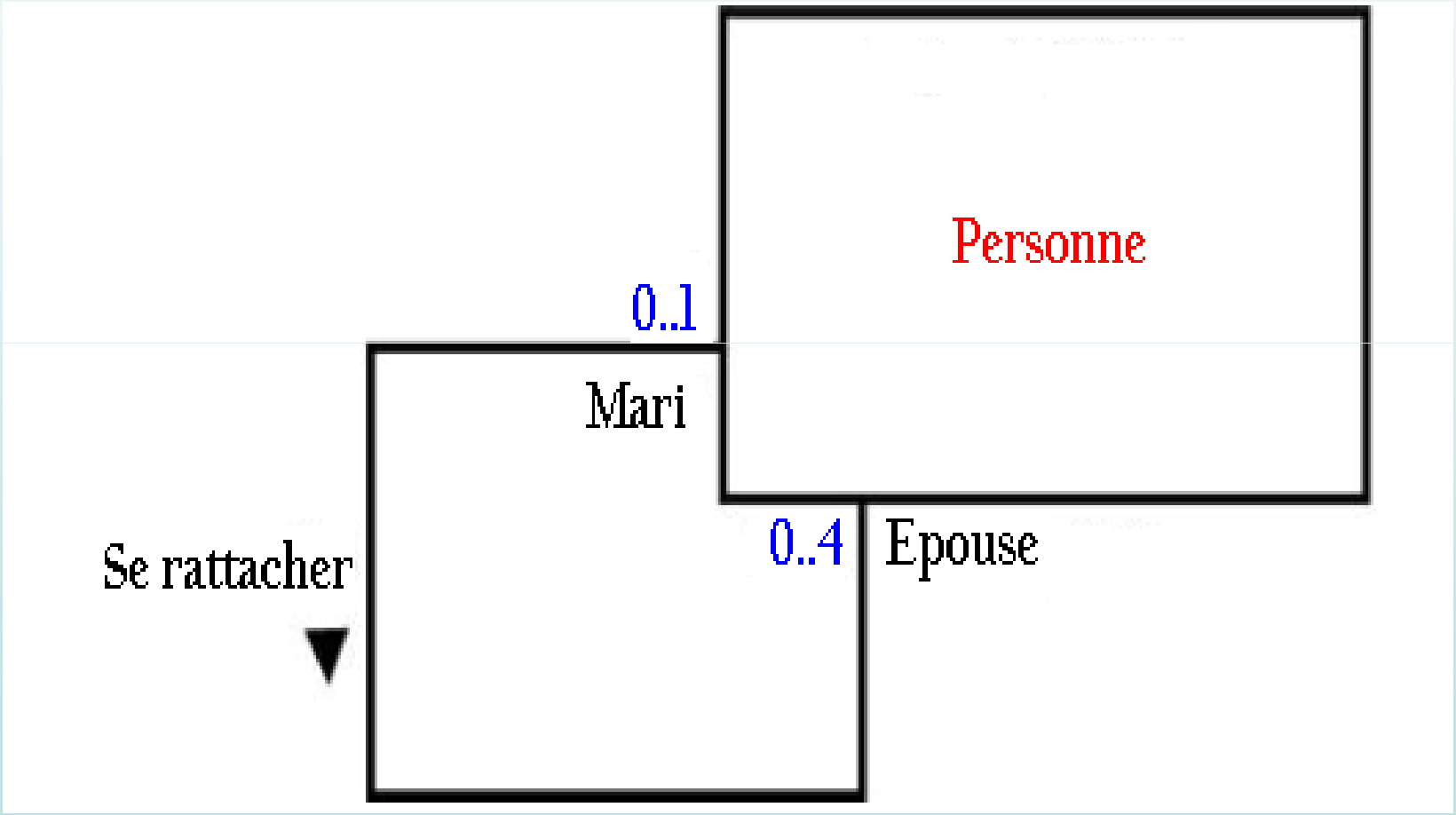
Les associations



Les associations

- Associations réflexives *:*
  - *Une association réflexives relie une classe à elle-même.*
  - *Dans ce cas, les objets d'une même classe sont en relation entre eux.*
  - *L'indication des rôles est important pour distinguer les instances qui participent à la relation.*
  - *Par exemple, un homme (le marie) peut avoir 0, 1, 2, 3 ou 4 femmes (l'épouse). Les hommes et les femmes sont des personnes.*

Les associations



L'agrégation

- Définition *:*
  - *L''agrégation est une association non symétrique qui exprime un couplage fort et une relation de subordination.*
  - *Elle représente des relations de type :*
    - *Tout / partie.*
    - *Ensemble / élément.*
    - *Maître / esclave.*
  - *Il y a deux types d'agrégation : l'agrégation faible et l'agrégation forte ou la composition.*

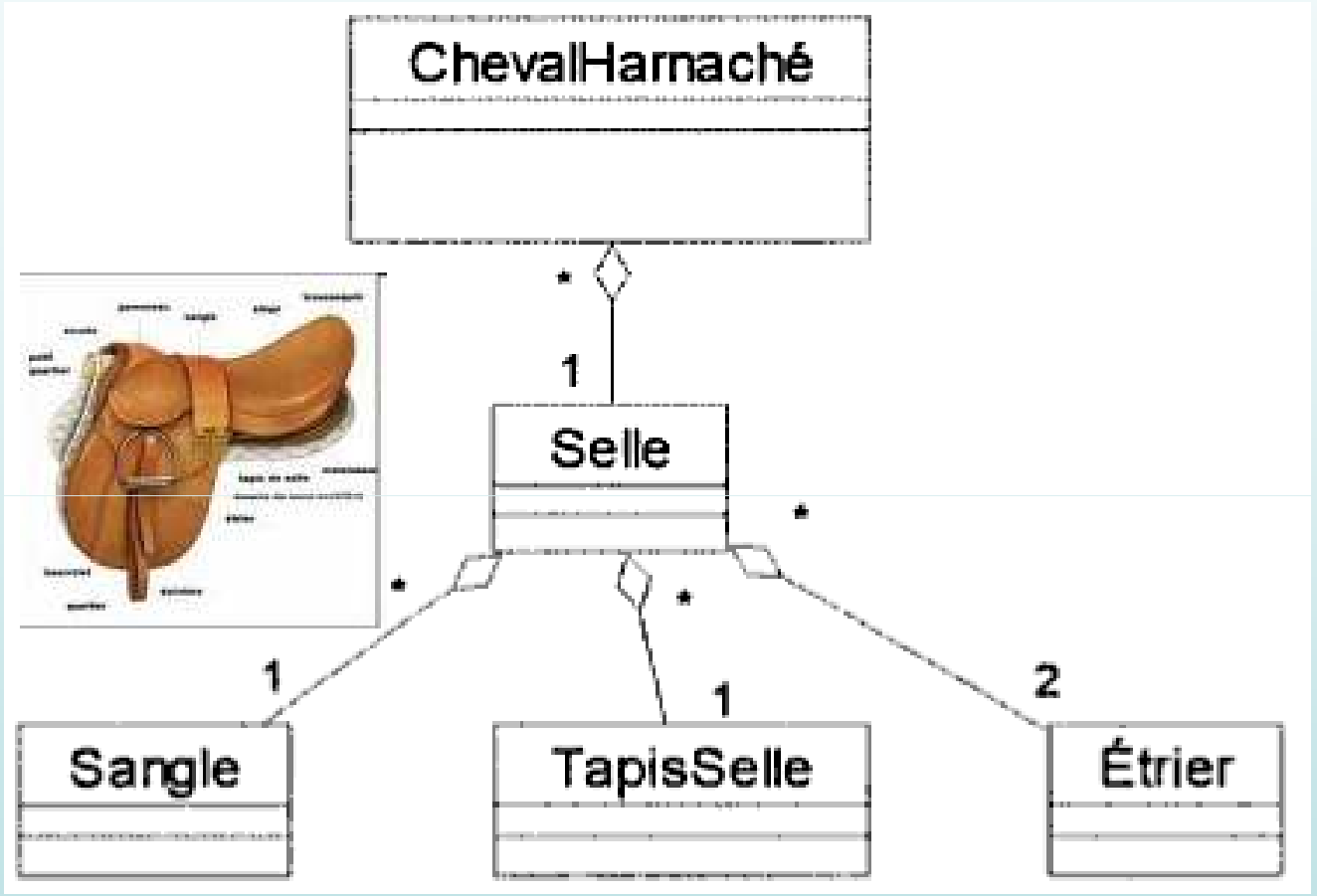
L'agrégation

- Agrégation faible *:*
  - *Une agrégation faible est caractérisée par :*
    - *Les composants (agrégés) peuvent partagés plusieurs composés (agrégats).*
    - *La destruction du composé ne conduit pas à la destruction des composants.*
  - *L'agrégation faible est schématisé en UML par un petit losange (sans remplissage) du côté de l'agrégé.*

L'agrégation

- Exemple *:*
  - *Un cheval harnaché est composé d'une selle. Une selle est elle-même composée d'une sangle, d'étriers et d'un tapis de selle.*
  - *C'est une agrégation faible :*
    - *La perte du cheval n'implique pas la perte de ces objets.*
    - *La perte de la selle n'entraîne pas la perte de ses composants.*
    - *Les objets peuvent être partagés par plusieurs chevaux.*

L’agrégation



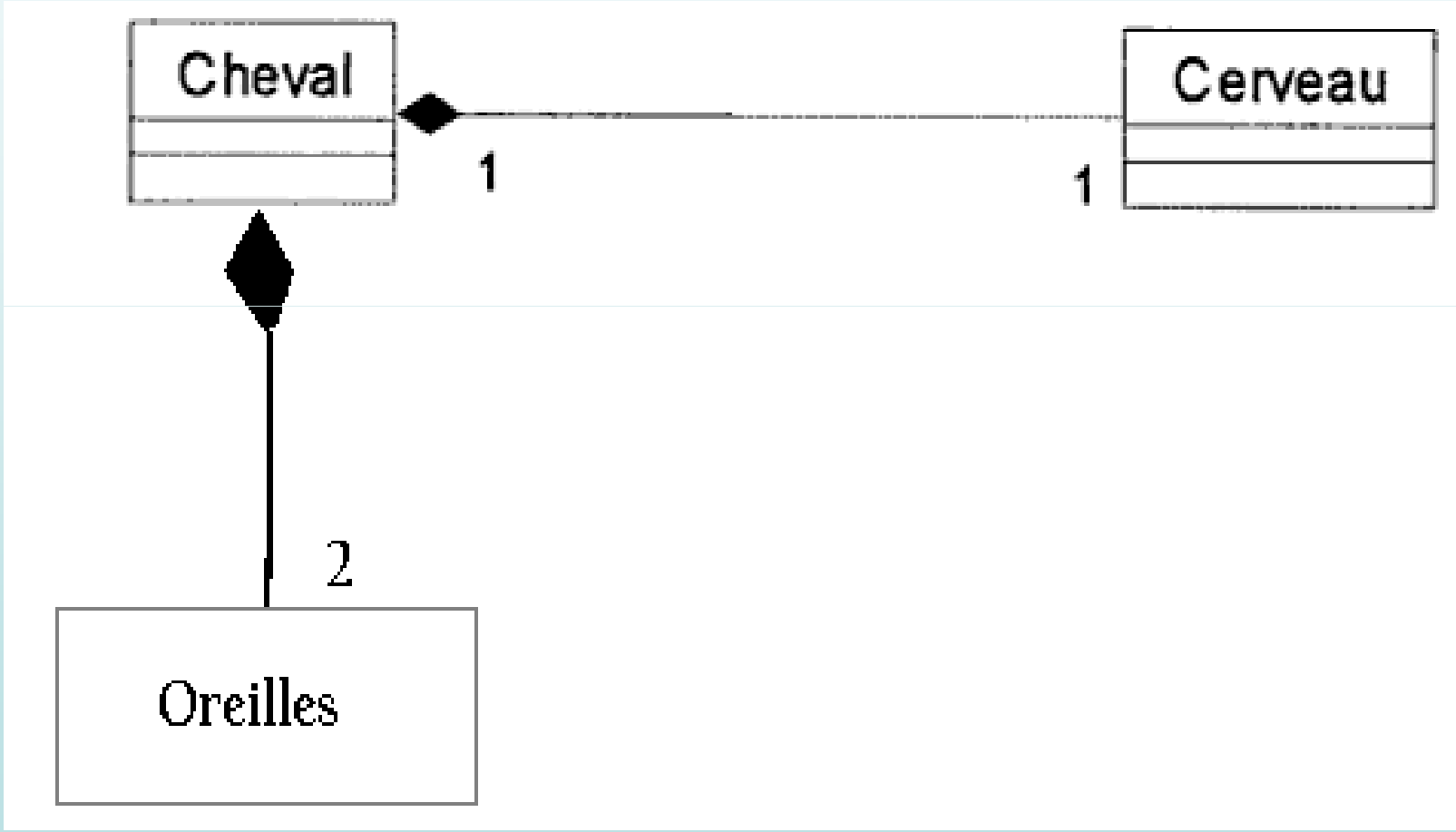
L'agrégation

- Agrégation forte (composition) *:*
  - *Une agrégation forte est caractérisée par :*
    - *Les composants (agrégés) ne sont pas partagés par les composés (agrégats).*
    - *La destruction du composé conduit à la destruction des composants.*
  - *La première contrainte entraîne que la valeur de la multiplicité du côté de l'agrégat ne peut être que 0 ou 1.*
  - *On représente une composition par un losange avec un remplissage noir du côté de l'agrégat.*

L’agrégation

- Exemple *:*
  - *Un cheval est composé d’un cerveau et deux oreilles.*
  - *C’est une composition :*
    - *Le cerveau et les oreilles ne sont pas partagés.*
    - *La mort du cheval implique la mort du cerveau et des oreilles.*

L’agrégation



L'agrégation

Différence entre agrégation et composition :

	Agrégation	Composition
Représentation	losange transparent	losange noir
Partage des composants par plusieurs associations	oui	non
Destruction des composants lors de la destruction du composé	non	oui
Cardinalité au niveau du composé	quelconque	0..1 ou 1

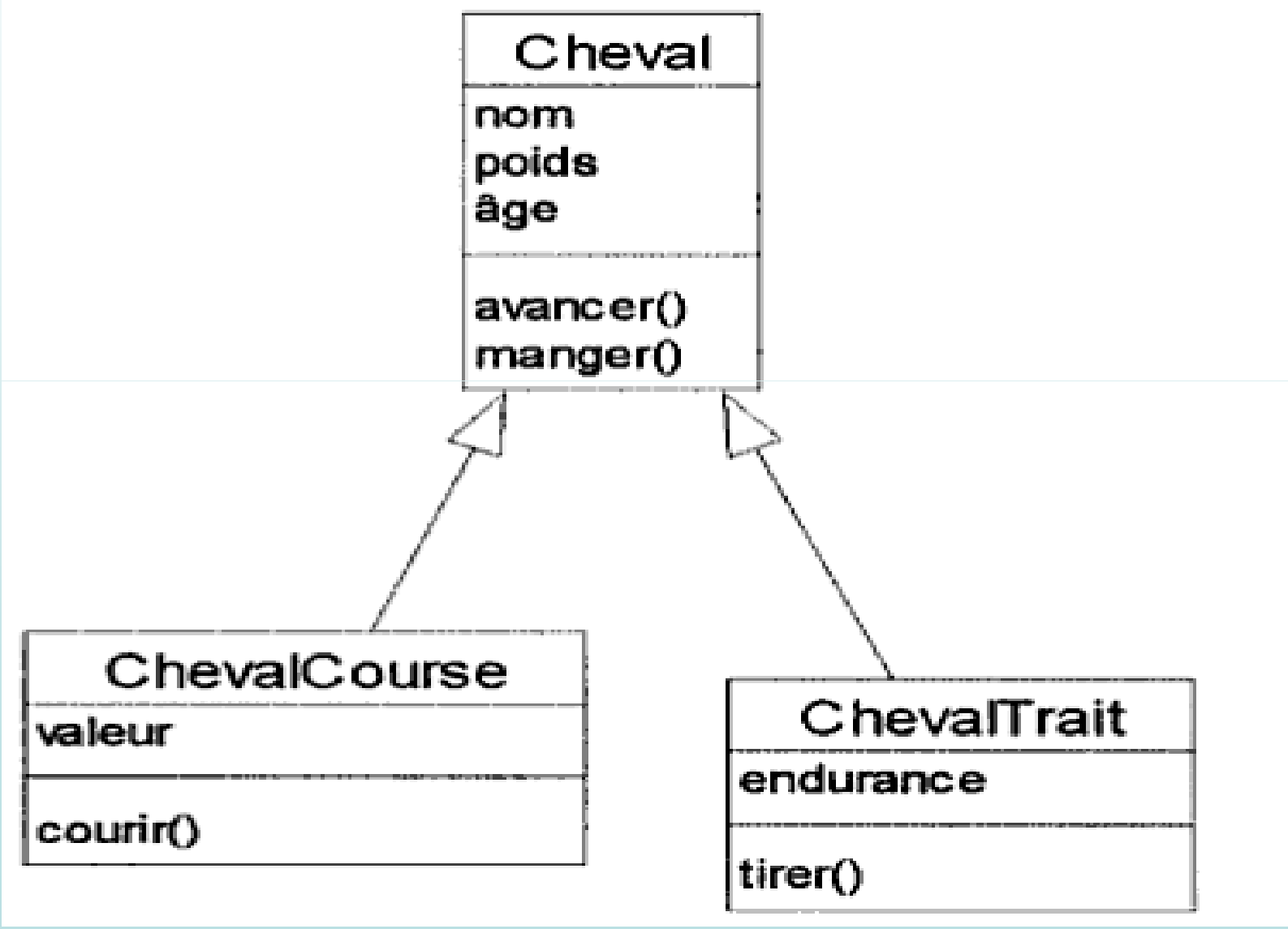
L'héritage

- Classes spécifiques et classes générales *:*
  - *Une classe est plus spécifique qu'une autre si toutes ses instances sont également instances de cette autre classe.*
  - *La classe plus spécifique est dite sous-classe ou classe dérivée de l'autre classe.*
  - *Cette dernière, plus générale, est dite super-classe.*
  - *La relation qui existe entre une sous-classe et sa super-classe s'appelle l'héritage.*

L’héritage

- Exemple *:*
  - *Un cheval a un nom, un poids et un âge.*
  - *Il peut avancer et manger.*
  - *Un cheval de trait est aussi un cheval, mais il est caractérisé par son endurance et par le fait qu’il tire les objets très lourds.*
  - *Un cheval de course, qui est également un cheval, est caractérisé par sa valeur et par le fait qu’il court très vite.*

L’héritage


Cheval
nom
poids
âge
avancer()
manger()
ChevalCourse
valeur
courir()
ChevalTrait
endurance
tirer()

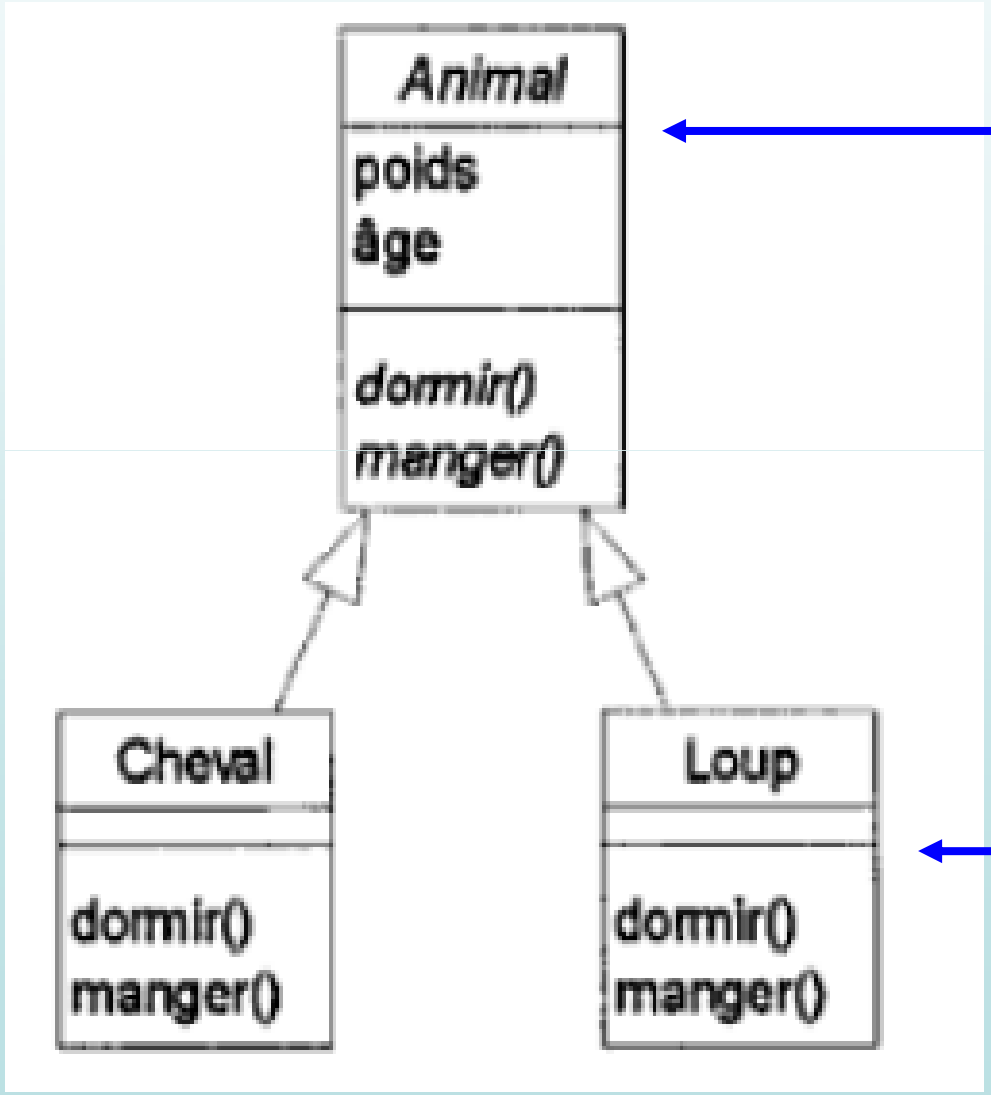
L'héritage

- Classes abstraites et concrètes *:*
  - *Une classe concrète est instanciable, c'est-à-dire qu'elle possède des instances.*
  - *Par contre, une classe abstraite est une classe non instanciable d'une manière directe, car elle ne fournit pas une description complète.*
  - *Une classe abstraite possède des sous-classes concrètes et sert à factoriser des attributs et des méthodes communs à ses sous-classes.*

L'héritage

- Classes abstraites et concrètes *:*
  - *Une méthode introduite dans une classe abstraite et avec seulement sa signature et sans son code est dite abstraite.*
  - *En UML, on met le nom d'une classe abstraite en italique, ou bien on l'identifié par le stéréotype* **<<abstract>>**.
  - *Par exemple, un animal a un poids et un âge. Il peut dormir et manger de façon distincte, selon la nature de l'animal. Un cheval et un loup sont des animaux.*

L'héritage



Une classe abstraite

Deux classes concrètes

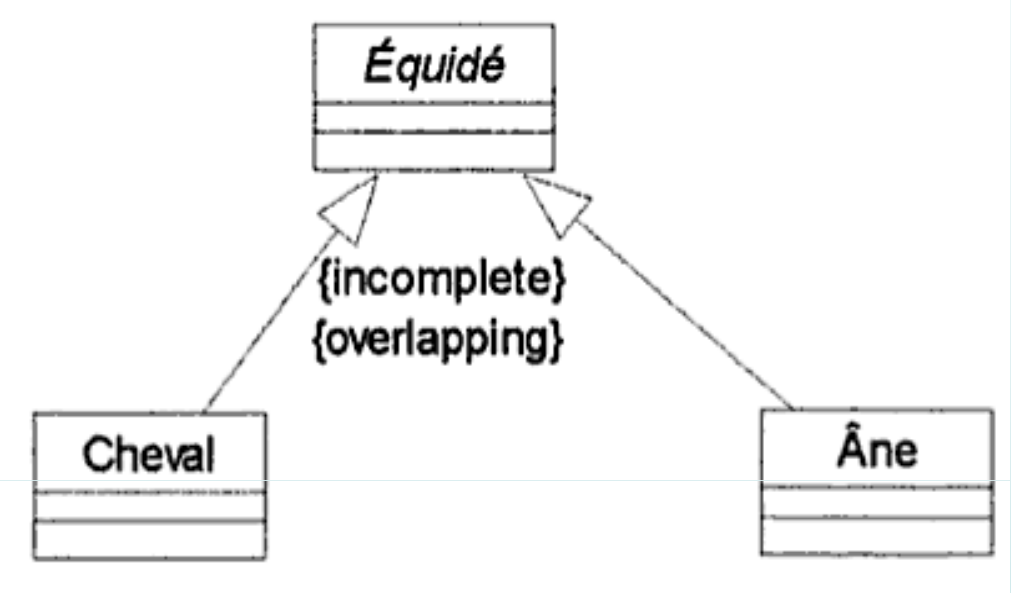
L'héritage

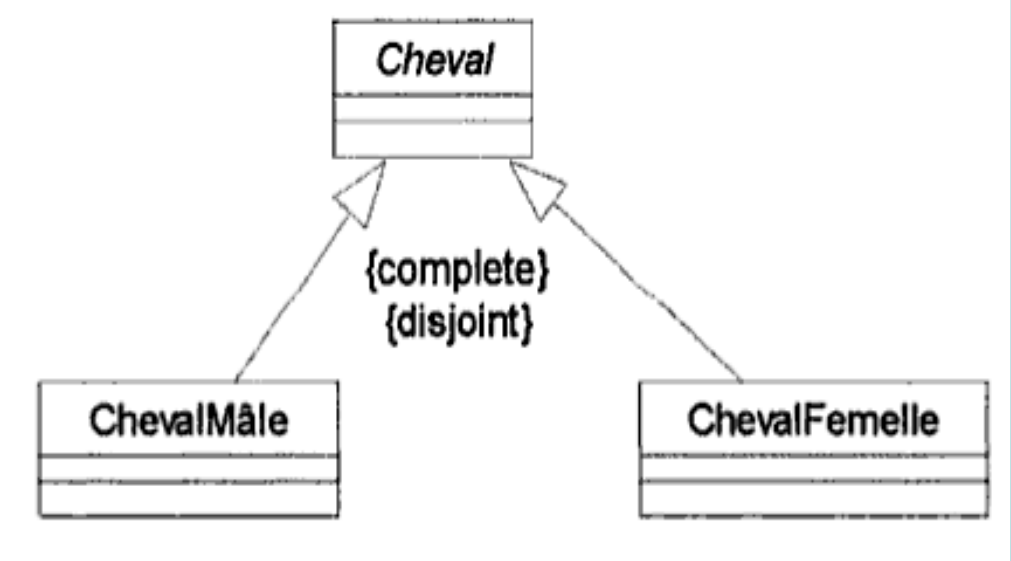
- Contraintes sur la relation d'héritage *:*
  - *UML offre 4 contraintes sur la relation d'héritage :*
    - *La contrainte* **{incomplete}** *signifie que l'ensemble des sous-classes est incomplet et qu'il ne couvre pas la surclasse.*
    - *La contrainte* **{complete}** *signifie au contraire que l'ensemble des sous-classes est complet et qu'il couvre la surclasse.*
    - *La contrainte* **{disjoint}** *signifie que les sous-classes n'ont aucune instance commune.*

L'héritage

- Contraintes sur la relation d'héritage *:*
  - *UML offre 4 contraintes sur la relation d'héritage :*
    - *La contrainte* **{overlapping}** *signifie que les sous-classes peuvent avoir une ou plusieurs instances communes.*
  - *Exemples :*
    - *Les chevaux et les ânes appartiennent à la famille des équidés. Mais ils ne couvrent pas cette famille (il y a aussi les zèbres). De plus, il existe les mulets qui sont issus d'un croisement entre cheval et âne.*

L'héritage


Équidé
{incomplete}
{overlapping}
Cheval
Âne


Cheval
{complete}
{disjoint}
ChevalMâle
ChevalFemelle

L'héritage

- L'héritage multiple *:*
  - *En UML, l'héritage multiple consiste à ce qu'une sous-classe peut avoir plus qu'une super-classe.*
  - *L'héritage multiple pose un problème : il engendre un conflit lorsqu'une même méthode est héritée plusieurs fois dans la sous-classe.*
  - *En effet, lors de la réception d'un message appelant cette méthode, il faut définir un critère pour en choisir une parmi toutes qui sont héritées.*

L'héritage

- L'héritage multiple *:*
  - *Une solution à ce problème consiste à redéfinir la méthode dans la sous-classe afin de supprimer le conflit.*
  - *Remarques :*
    - *Rares sont les langages de programmation qui supportent l'héritage multiple.*
    - *Le langage C++ supporte l'héritage multiple.*

L’héritage