UML : Les concepts de l'approche par objets.

BAKAYOKO Mekossou

Table des matières

Objectifs

A la fin de cette leçon vous serez capable de :

- *Définir* les notions objets, classes, spécialisation, composition
- *Maîtriser* les principes abstraction, encapsulation

Introduction

Cette Leçon a pour objectif de vous faire découvrir les différents concepts et principes de l'approche objet qui sont à la base d'UML.

Étudier l'approche *Objet* est indispensable pour comprendre les éléments utilisés dans la collection des diagrammes d'UML qui seront abordés dans les Leçons suivants

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

La notion de *classes*, représentation commune d'un ensemble d'objets similaires, sera introduite.

Nous évoquerons ensuite le principe d'encapsulation, masquage d'informations internes et propres au fonctionnement de l'objet.

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

Enfin, nous évoquerons la composition d'objets.

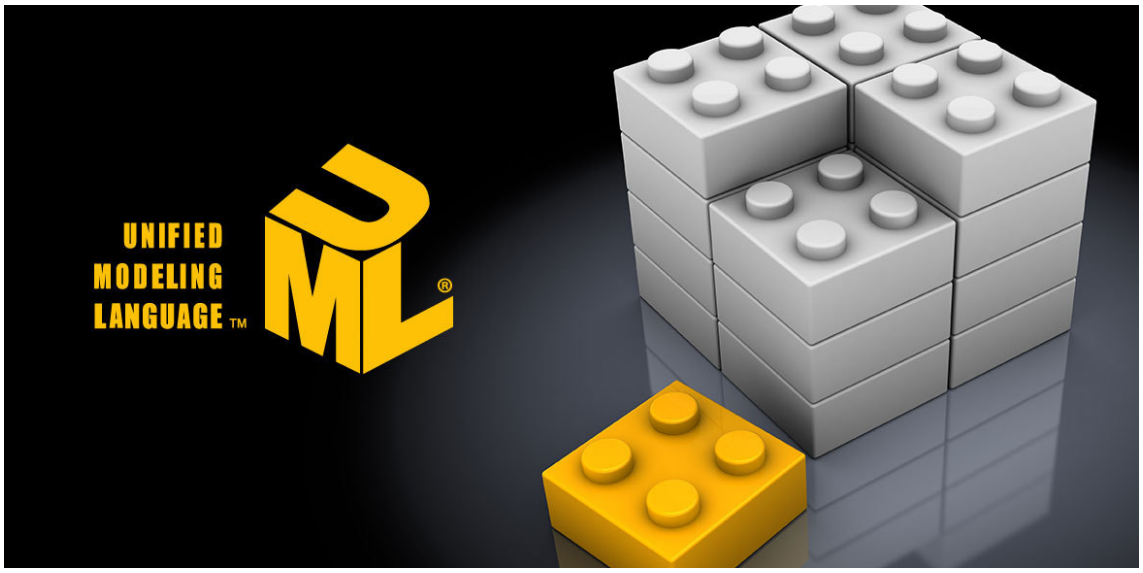
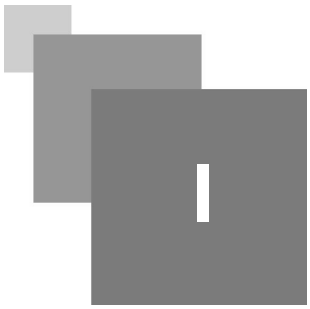


Figure 1 : UML et L'approche Objet

Grands principes de la notion objet



Objectifs

A la fin de cette section vous serez capable de définir et comprendre les notions ci dessous:

1. La notion d'objet

Définition

Un objet est une entité identifiable du monde réel. Il peut avoir une existence physique (un cheval, un livre) ou ne pas en avoir (un texte de loi). Identifiable signifie que l'objet peut être désigné.

Un objet est une représentation de quelque chose (nous verrons plus tard qu'il s'agit en fait d'un modèle).

Quelque chose, c'est à dire à peu près tout ce que l'on veut : *« une maison, un micro-processeur, une personne ou encore une instruction en langage de programmation. »*

Exemple

- Ma tablette UVCI
- Mon sac au doc

1.1. Les attributs et les méthodes d'un objet

Définition

En UML, tout objet possède un *ensemble d'attributs* (sa structure) et un *ensemble de méthodes* (son comportement).

- *Un attribut* est une variable destinée à recevoir une valeur.
- *Une méthode* est un ensemble d'instructions prenant des valeurs en entrée et modifiant les valeurs des attributs ou produisant un résultat.

Remarque

- Un objet statique du monde réel est toujours perçu comme dynamique. Ainsi en UML, un livre est perçu comme un objet capable de s'ouvrir lui-même à la énième page.
- Tout système conçu en UML est composé d'objets interagissant entre eux et effectuant les opérations propres à leur comportement.
- Le comportement global d'un système est ainsi réparti entre les différents objets. Dans notre exemple, il suffit de faire le parallèle avec le monde réel pour le comprendre.

Exemple

« Un troupeau de chevaux est un système d'objets interagissant entre eux, chaque objet possédant son propre comportement. »

2. L'abstraction

Définition

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

Exemple

- *« On s'intéresse aux chevaux pour l'activité de course. Les propriétés d'aptitude de vitesse, d'âge et d'équilibre mental ainsi que l'élevage d'origine sont pertinentes pour cette activité et sont retenues. »*
- *« On s'intéresse aux chevaux pour l'activité de trait. Les propriétés d'âge, de taille, de force et de corpulence sont pertinentes pour cette activité et sont retenues. »*

Remarque

L'abstraction est une simplification indispensable au processus de modélisation.

Un objet UML est donc une abstraction de l'objet du monde réel par rapport aux besoins du système, dont on ne retient que les éléments essentiels.

3. Les classes d'objets

Définition

Un ensemble d'objets similaires, c'est-à-dire possédant la même *structure* et le même *comportement* et constitués des mêmes *attributs* et *« méthodes »*, forme une *classe d'objets*.

La structure et le comportement peuvent alors être définis en commun au niveau de la classe.

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

Exemple

L'ensemble des chevaux constitue la classe Cheval qui possède la structure et le comportement décrits à la figure ci-dessous

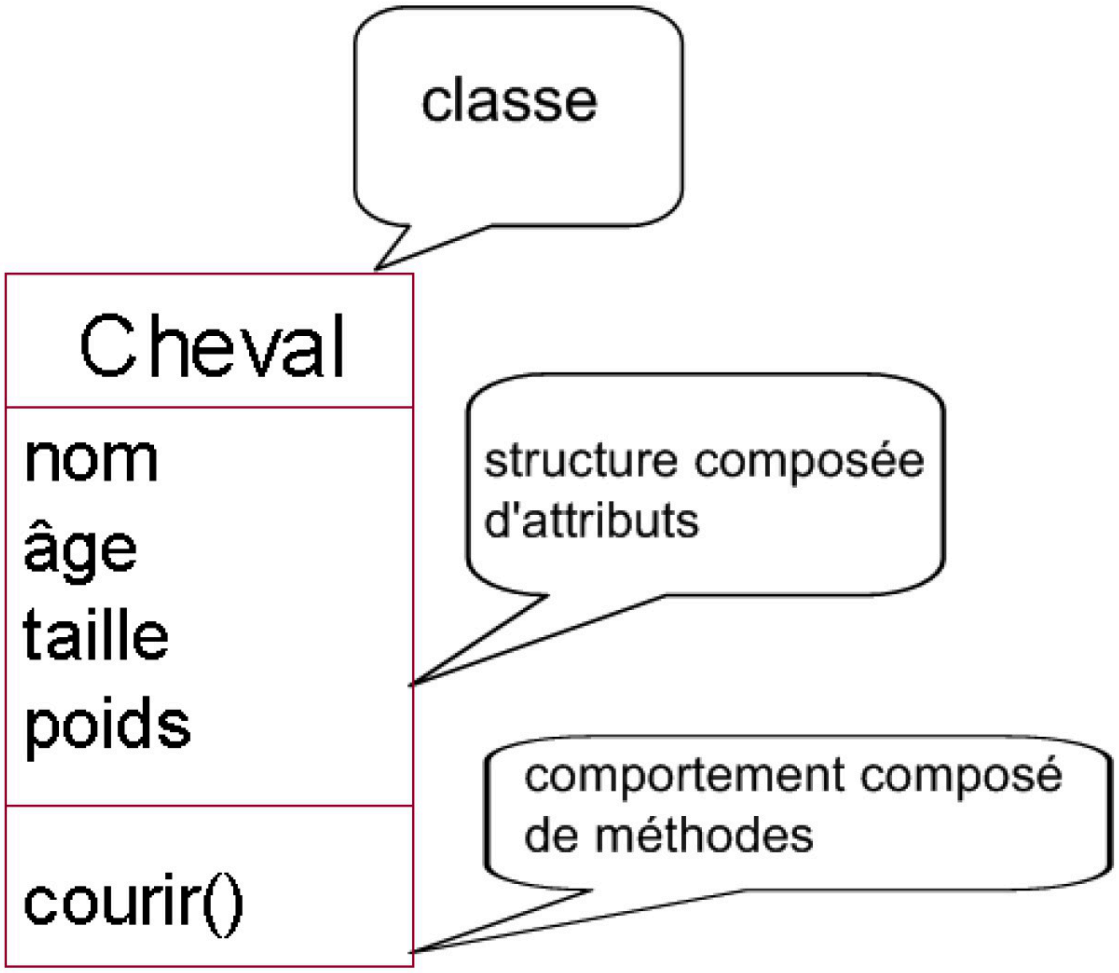


Figure 1 : La classe cheval

Le cheval Jorphée est une instance de la classe Cheval dont les attributs et leurs valeurs sont illustrés à la figure 2

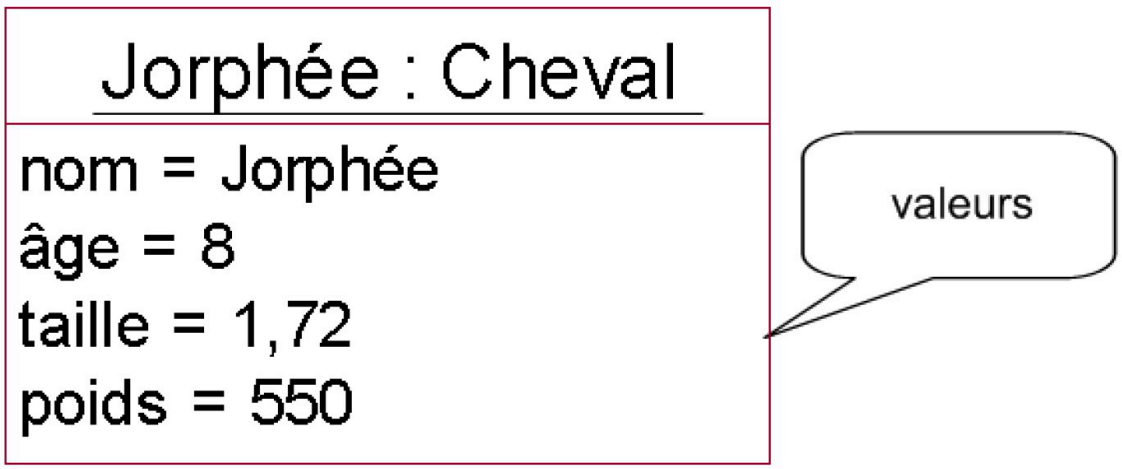


Figure 2 : Instance de la classe cheval : le cheval Jorphée

Remarque

Le nom d'une classe apparaît au singulier. Il est toujours constitué d'un nom commun précédé ou suivi d'un ou plusieurs adjectifs qualifiant le nom. Ce nom est significatif de l'ensemble des objets constituant la classe.

4. L'encapsulation

Définition

- L'encapsulation consiste à *masquer des attributs et des méthodes* de l'objet vis-à-vis des autres objets. En effet, certains attributs et méthodes ont pour seul objectif des *traitements internes* à l'objet et ne doivent pas être exposés aux *objets extérieurs*.
  Encapsulés, ils sont appelés les *attributs et méthodes privés* de l'objet.
- L'encapsulation est une abstraction puisque l'on simplifie la représentation de l'objet vis-à-vis des objets extérieurs.
  Cette représentation simplifiée est constituée des *attributs et méthodes publiques* de l'objet.
- La définition de l'encapsulation se fait au niveau de la classe. Les objets extérieurs à un objet sont donc les instances des autres classes.

Exemple

Lorsqu'il court, un cheval va effectuer différents mouvements comme lever les jambes, lever la tête, lever la queue. Ces mouvements sont internes au fonctionnement de l'animal et n'ont pas à être connus à l'extérieur. Ce sont des *méthodes privées*.

Ces opérations accèdent à une partie interne du cheval : ses muscles, son cerveau et sa vue. Cette partie interne est représentée sous la forme d'*attributs privés*. L'ensemble de ces attributs et méthodes est illustré à la figure 3

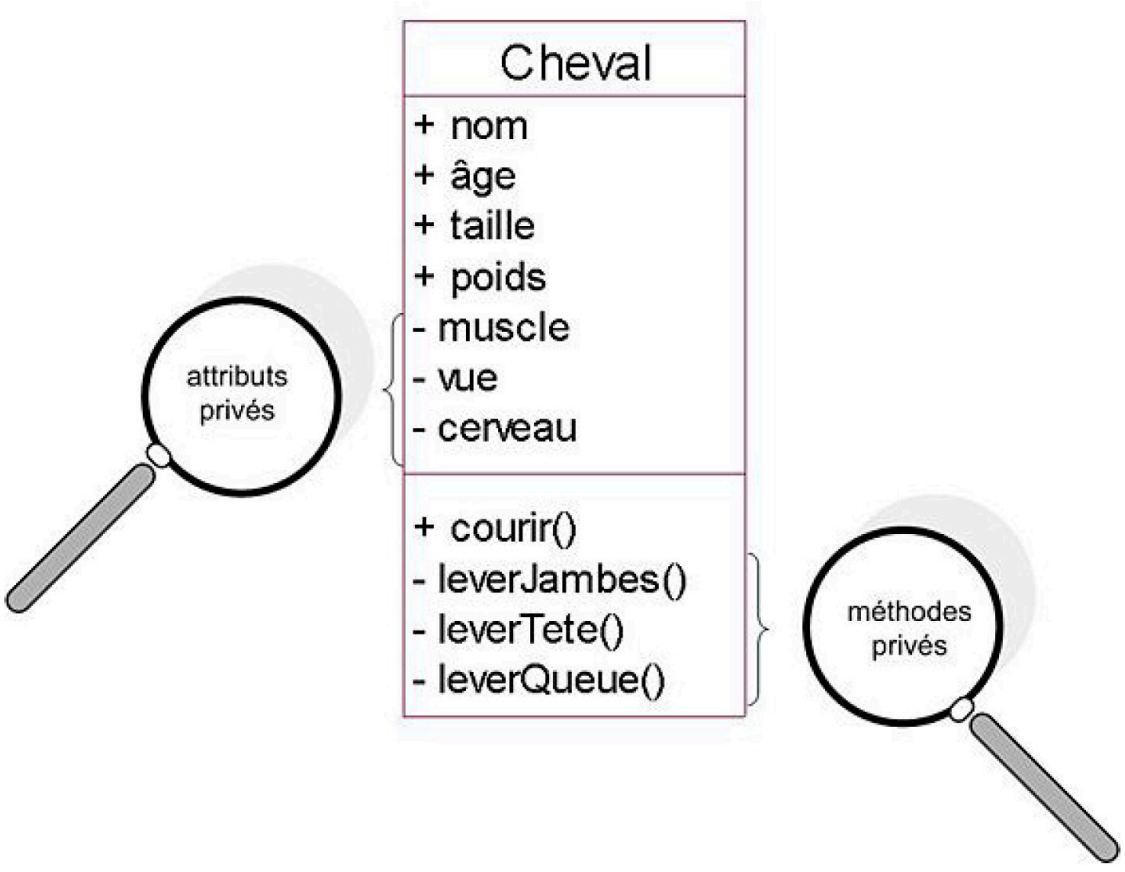


Figure 3 : La classe cheval détaillée

Attention

Dans la notation UML :

- Les *attributs et méthodes publics* sont précédés du *signe plus (+)*
- Les *attributs et méthodes privés* (*« encapsulés »*) sont précédés du *signe moins (-)*.

5. La spécialisation et la généralisation

Définition

Jusqu'à présent, chaque classe d'objets est introduite séparément des autres classes. Une classe peut également être définie comme un sous-ensemble d'une autre classe, ce sous-ensemble devant toujours constituer un ensemble d'objets similaires.

Il s'agit alors d'une sous-classe d'une autre classe. Elle constitue ainsi une *spécialisation* de cette autre classe.

Exemple

La classe des chevaux est une *sous-classe* de la classe des mammifères.

Définition

La *généralisation* est la relation inverse de la *spécialisation*.

Si une classe est une *spécialisation* d'une autre classe, cette dernière est une *généralisation* de la première. Elle en est sa *surclasse*.

Exemple

La classe des mammifères est une *sur-classe* de la classe des chevaux.

Définition : Hiérarchie de classes

La relation de spécialisation peut s'appliquer à plusieurs niveaux, donnant lieu à une hiérarchie de classes.

Exemple

La classe des chevaux est une sous-classe de la classe des mammifères, elle-même sous-classe de la classe des animaux. La classe des chiens est une autre sous-classe de la classe des mammifères. La hiérarchie correspondante des classes est représentée à la figure 4

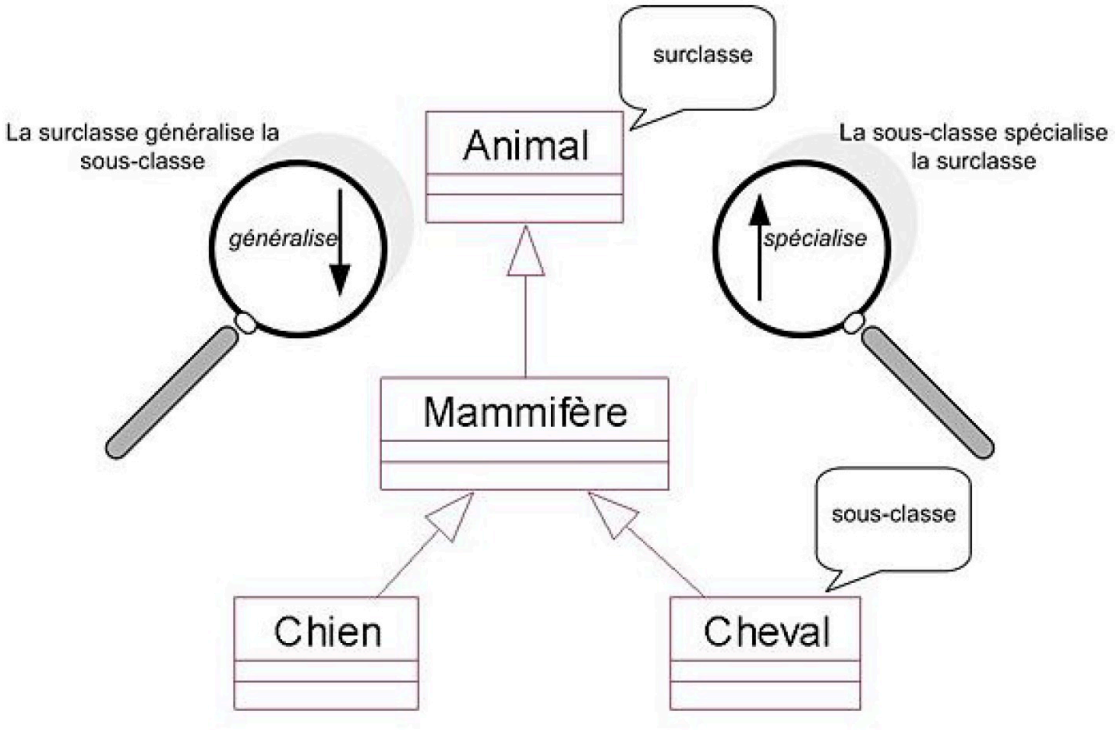


Figure 4 : Hiérarchie de classes

6. L'héritage

Définition

L'héritage est la propriété qui fait bénéficier à une sous-classe de la structure et du comportement de sa surclasse.

L'héritage provient du fait qu'une sous-classe est un sous-ensemble de sa sur-classe. Ses instances sont également instances de sa sur-classe.

En conséquence, elles bénéficient de la structure et du comportement définis dans cette sur-classe, en plus de la structure et du comportement introduits au niveau de la sous-classe.

Exemple

Soit un système où la classe Cheval est une sous-classe directe de la classe Animal, un cheval est alors décrit par la combinaison de la structure et du comportement issus des classes Cheval et Animal, c'est-à-dire avec les *attributs « âge, taille, poids, nom et élevage »* ainsi que les *méthodes « manger »* et *« courir »*. Cet héritage est illustré à la figure 5

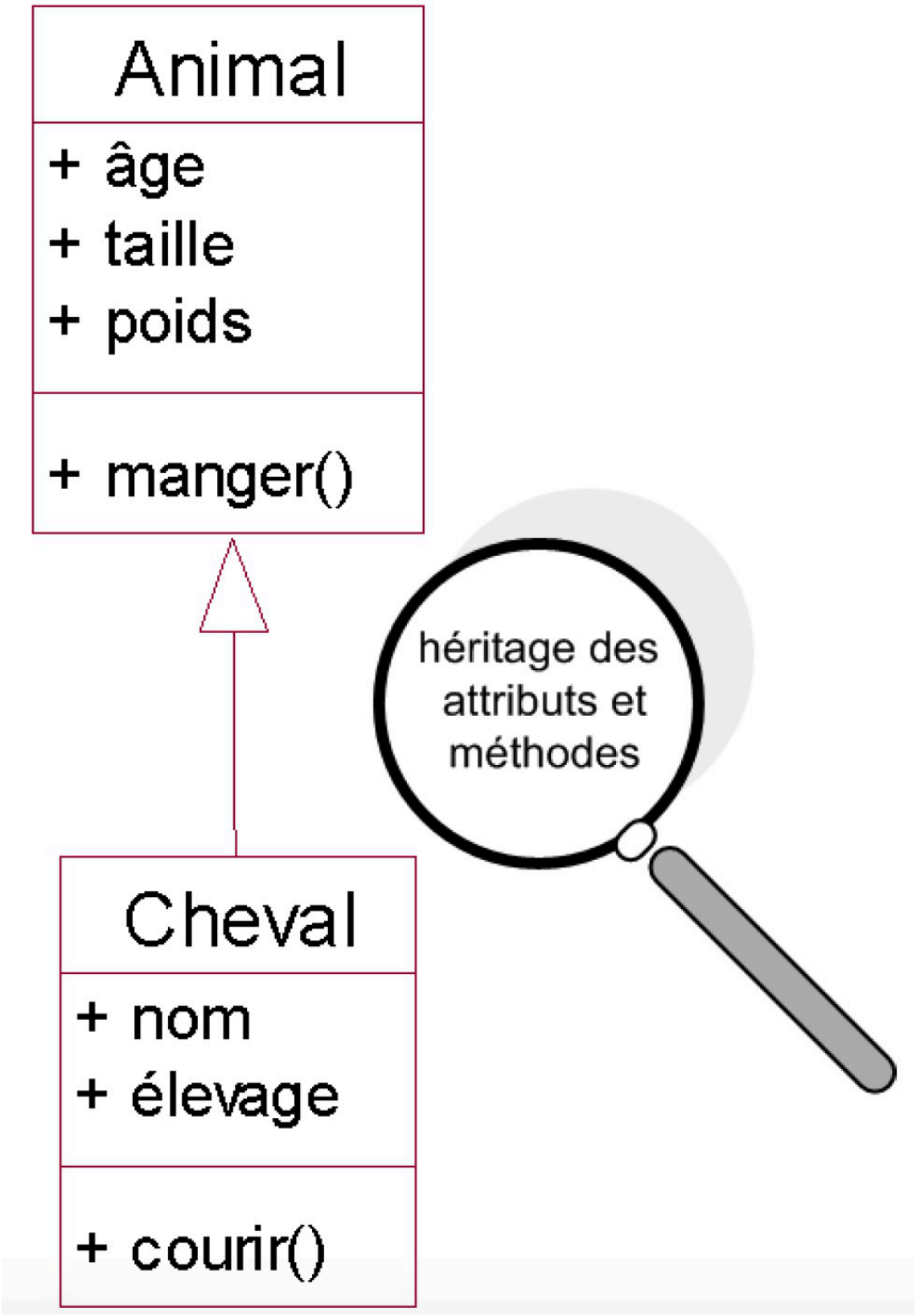


Figure 5 : héritage

Remarque

L'héritage est une conséquence de la spécialisation. Cependant, les informaticiens emploient beaucoup plus souvent le terme hérite que spécialise pour désigner la relation entre une sous-classe et sa sur-classe.

7. Les classes abstraites et concrètes

La hiérarchie présentée à la figure 4 montre qu'il existe deux types de classes dans la hiérarchie :

- *Classes concrètes* : les classes qui possèdent des *instances*, à savoir les classes Cheval et Chien.
- *Classes abstraites* : les classes qui ne possèdent pas directement des *instances*, comme la classe Animal.
  En effet, si dans le monde réel, il existe des chevaux, des chiens, le concept d'animal reste, quant à lui, abstrait. Il ne suffit pas à définir complètement un animal. La classe Animal est appelée une *classe abstraite*

Remarque

Une classe abstraite a pour vocation de posséder des sous-classes concrètes. Elle sert à factoriser des attributs et des méthodes communs à ses sous-classes.

Exemple

La figure 6 reprend la hiérarchie en indiquant précisément les *classes abstraites* et les *classes concrètes*.

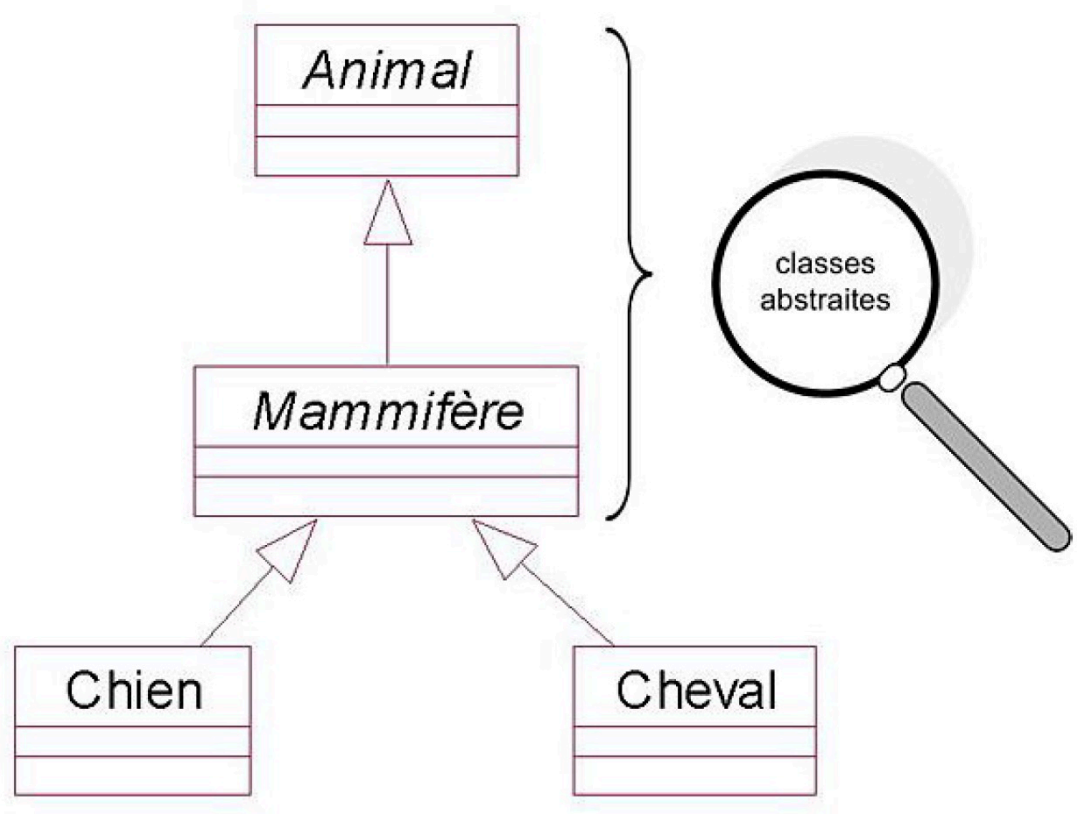


Figure 6 : Classes abstraites et classes concrètes

Attention

En UML, le nom des classes abstraites apparaît en caractères italiques.

8. Le polymorphisme

Définition

Le polymorphisme signifie qu'une classe (très généralement abstraite) représente un ensemble constitué d'objets différents car ils sont instances de sous-classes distinctes.

Lors de l'appel d'une méthode de même nom, cette différence se traduit par des comportements différents (sauf dans le cas où la méthode est commune et héritée de la surclasse dans les sous-classes).

Exemple

Soit la hiérarchie de classes illustrée à la figure 7. La méthode *« caresser »* a un comportement différent selon que le cheval est instance de *ChevalSauvage* ou de *ChevalDomestiqué*. Dans le premier cas, le comportement sera un refus (se traduisant par un cabrement) alors que dans le second, le comportement sera une acceptation.

Si l'on considère la classe Cheval dans son intégralité, on a donc un ensemble de chevaux qui ne réagissent pas de la même façon lors de l'activation de la méthode caresser.

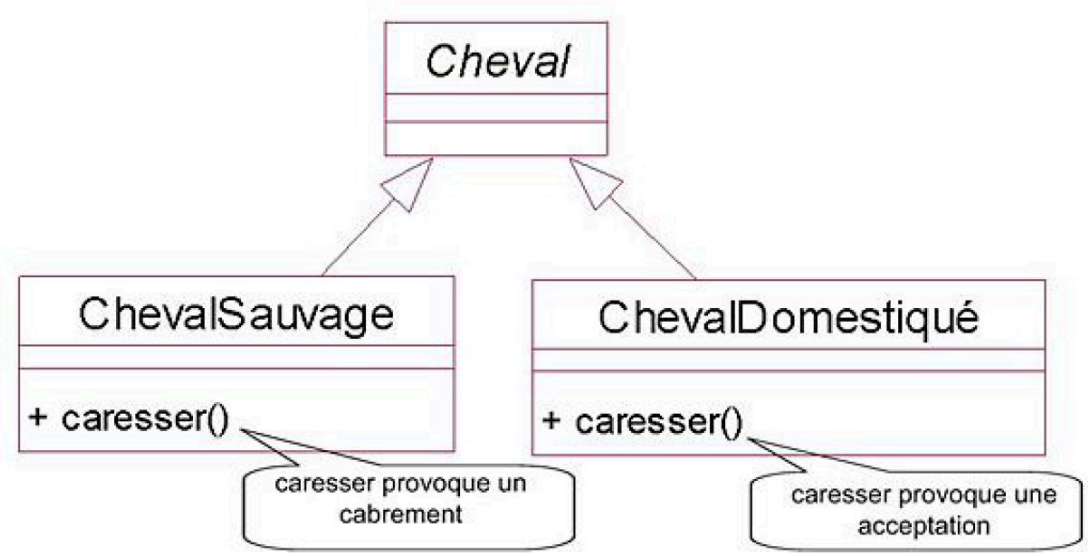


Figure 7 : Le polymorphisme

9. La composition

Définition

Un objet peut être complexe et composé d'autres objets. L'association qui unit alors ces objets est la *composition*. Elle se définit au niveau de leurs classes mais les liens sont bâtis entre les instances des classes. Les objets formant l'objet composé sont appelés composants.

Exemple

Un cheval est un exemple d'objet complexe. Il est constitué de ses différents organes (jambes, tête, etc.). La représentation graphique de cette composition se trouve à la figure 8.

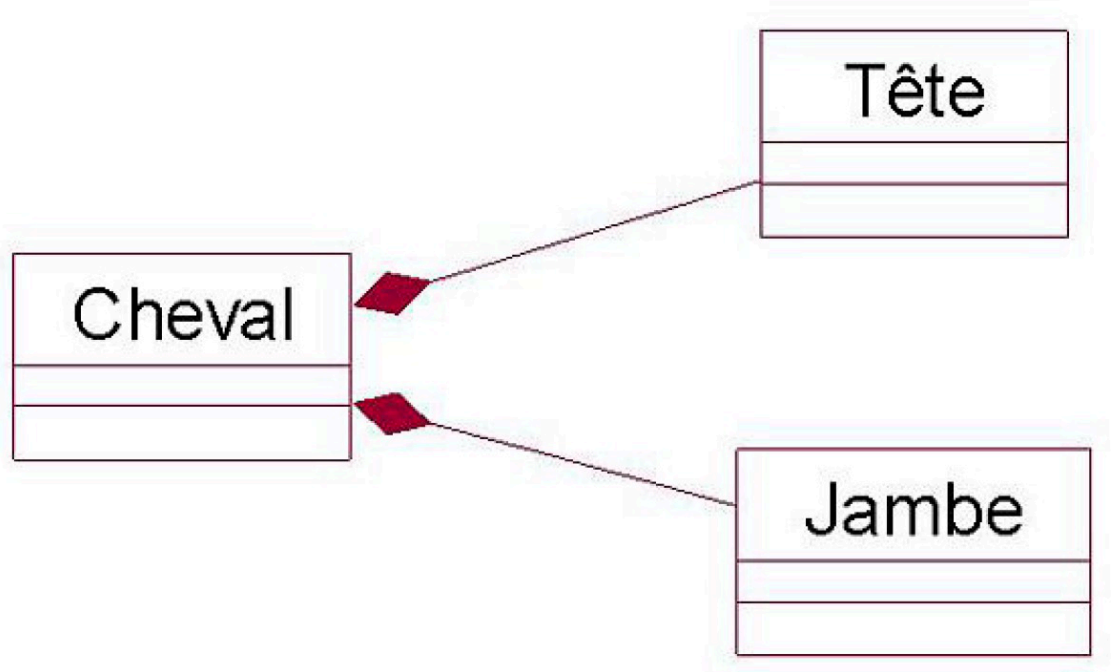


Figure 8 : Composition

9.1. La composition forte et la composition faible

Définition

La composition peut prendre deux formes :

- *la composition faible ou agrégation* : la composition faible, les composants peuvent être partagés entre plusieurs objets complexes
- *la composition forte* : les composants ne peuvent être partagés et la destruction de l'objet composé entraîne la destruction de ses composants

Exemple

Si l'on reprend l'exemple précédent dans le cas d'un cheval de course harnaché et si l'on ajoute la selle dans les composants, on

obtient :

- Une composition forte pour les jambes et la tête ; en effet, jambes et tête ne peuvent pas être partagées et la disparition du cheval entraîne la disparition de ses organes.
- Une agrégation ou composition faible pour la selle.

L'ensemble est illustré à la figure 9

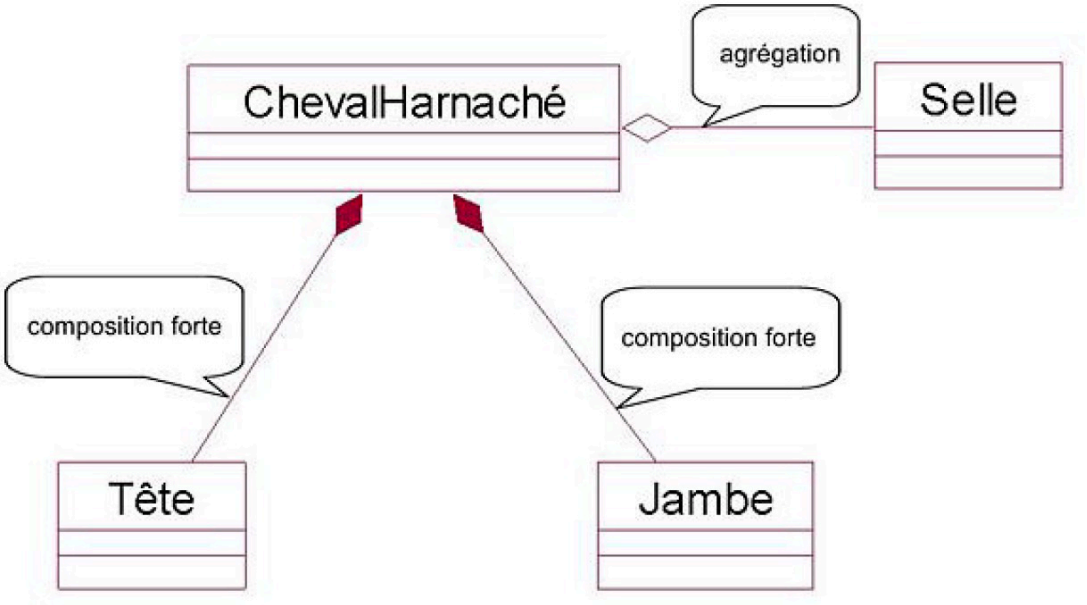


Figure 9 : Composition et agrégation

10. Conclusion

L'approche par objets forme la base d'UML. Elle est constituée de concepts (*objets, classes, spécialisation, composition*) et de principes (*abstraction, encapsulation*).

Nous verrons dans les chapitres suivants comment les différents diagrammes d'UML s'appuient sur les concepts et principes de l'approche par objets

11. Exercice : Exercices

[Solution n°1 p 17]

Exercice : Exercice 01

B hérite de A. Qui est la classe mère de B ?

- [ ] object
- [ ] A
- [ ] void
- [ ] nil
- [ ] B

Exercice : Exercice 02

Pour respecter le principe d'encapsulation, comment différents objets de différentes classes interagissent ?

- ◯ À l'aide de méthodes
- ◯ En utilisant des références croisées
- ◯ À l'aide de pointeurs

Exercice : Exercice 03

Le principe d'abstraction est permis par quel mécanisme ?

- ◯ Héritage
- ◯ Encapsulation
- ◯ Prototypage dynamique

Solutions des exercices

> Solution n°1

Exercice p. 16

Exercice 01

- ☐ object
- ☑ A
- ☐ void
- ☐ nil
- ☐ B

Exercice 02

- ◉ À l'aide de méthodes
- ○ En utilisant des références croisées
- ○ À l'aide de pointeurs

Exercice 03

- ◉ Héritage
- ○ Encapsulation
- ○ Prototypage dynamique