



Concepts objets par l'exemple

Stéphane Ducasse

Stephane.Ducasse@univ-savoie.fr

<http://www.iam.unibe.ch/~ducasse/>

Plan

- Inspiré d'un cours de l'Université de Laval.
- Parallèle entre une situation réelle et la modélisation objet
- Objet, requêtes et messages
- Instance, classes et héritage
- Autre Exemple
- Composition d'objets



Problème

- Supposons que Pierre veuille envoyer des fleurs à sa grand-mère Béatrice à l'occasion de son anniversaire.



Dans la vie...

- Pierre va alors chez son fleuriste Gaston et lui dit la variété et le nombre de fleurs qu'il désire expédier à sa grand-mère.
- Il lui donne l'adresse de sa grand-mère Béatrice.

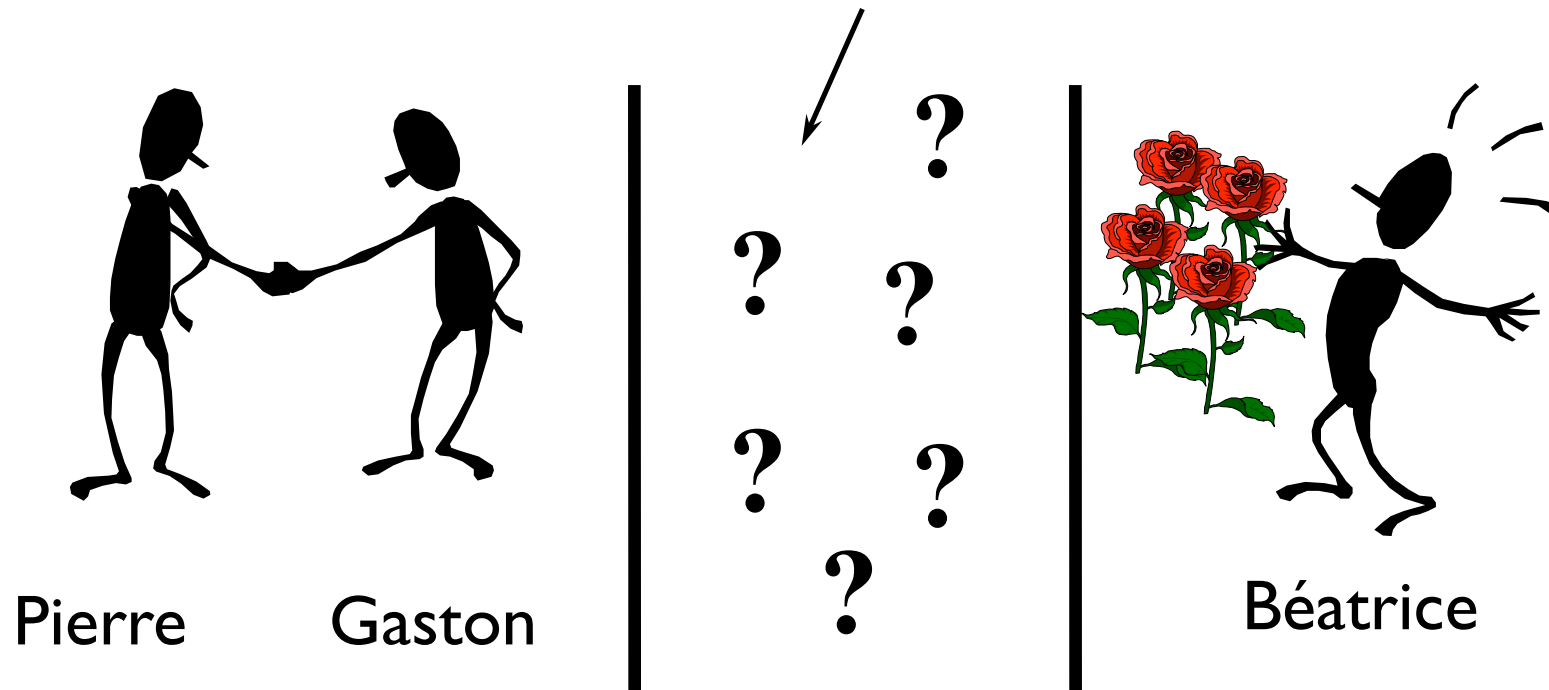


Objet, responsabilité, messages et méthodes

- En termes techniques, Pierre a résolu son problème en choisissant un *objet* [Gaston, le fleuriste] et lui a passé un *message* contenant sa requête [livrer des roses à sa grand-mère Béatrice].
- C'est la *responsabilité* de Gaston de répondre à la requête de Pierre. Il connaît et utilise des *mécanismes* [il exécute une *méthode*] pour bien le faire.

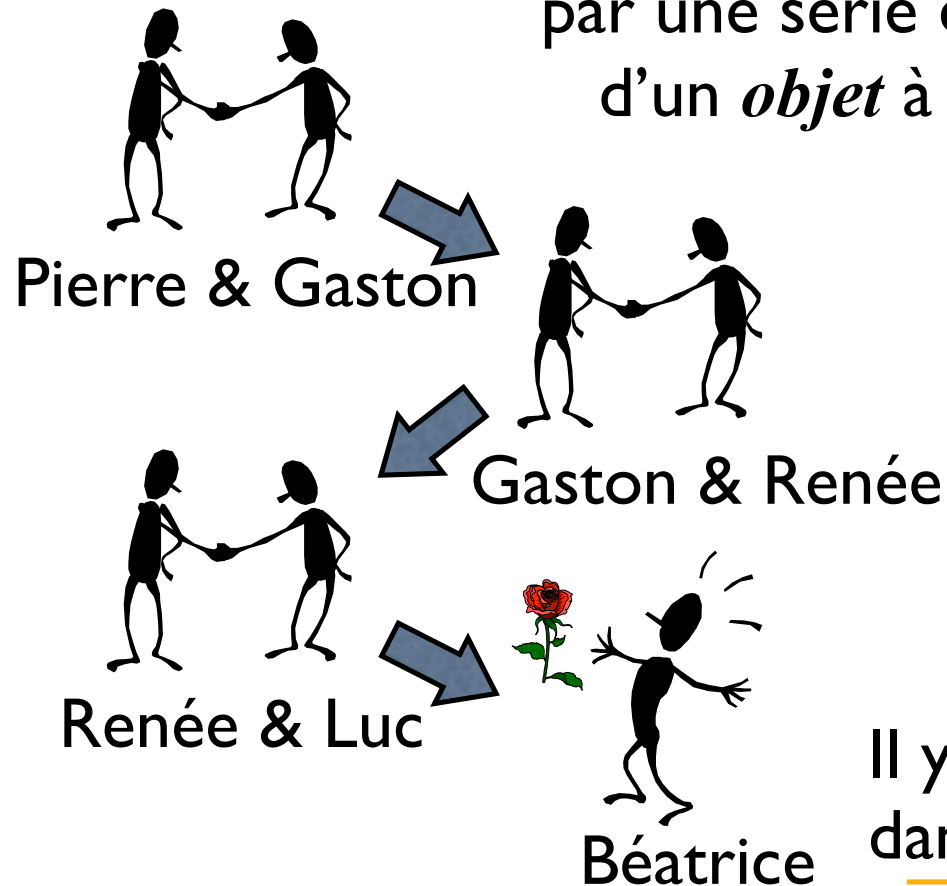
Encapsulation de l'information

Les mécanismes et les moyens employés
pour
livrer les fleurs sont inconnus de Pierre.



Délégation de la tâche à un tiers

Une *requête* est satisfaite
par une série de *requêtes*
d'un *objet* à un *autre*.



Pierre *n'a pas* à connaître
la *façon exacte* dont sera
exécutée sa requête,
l'important étant que sa
grand-mère Béatrice
reçoive des fleurs.

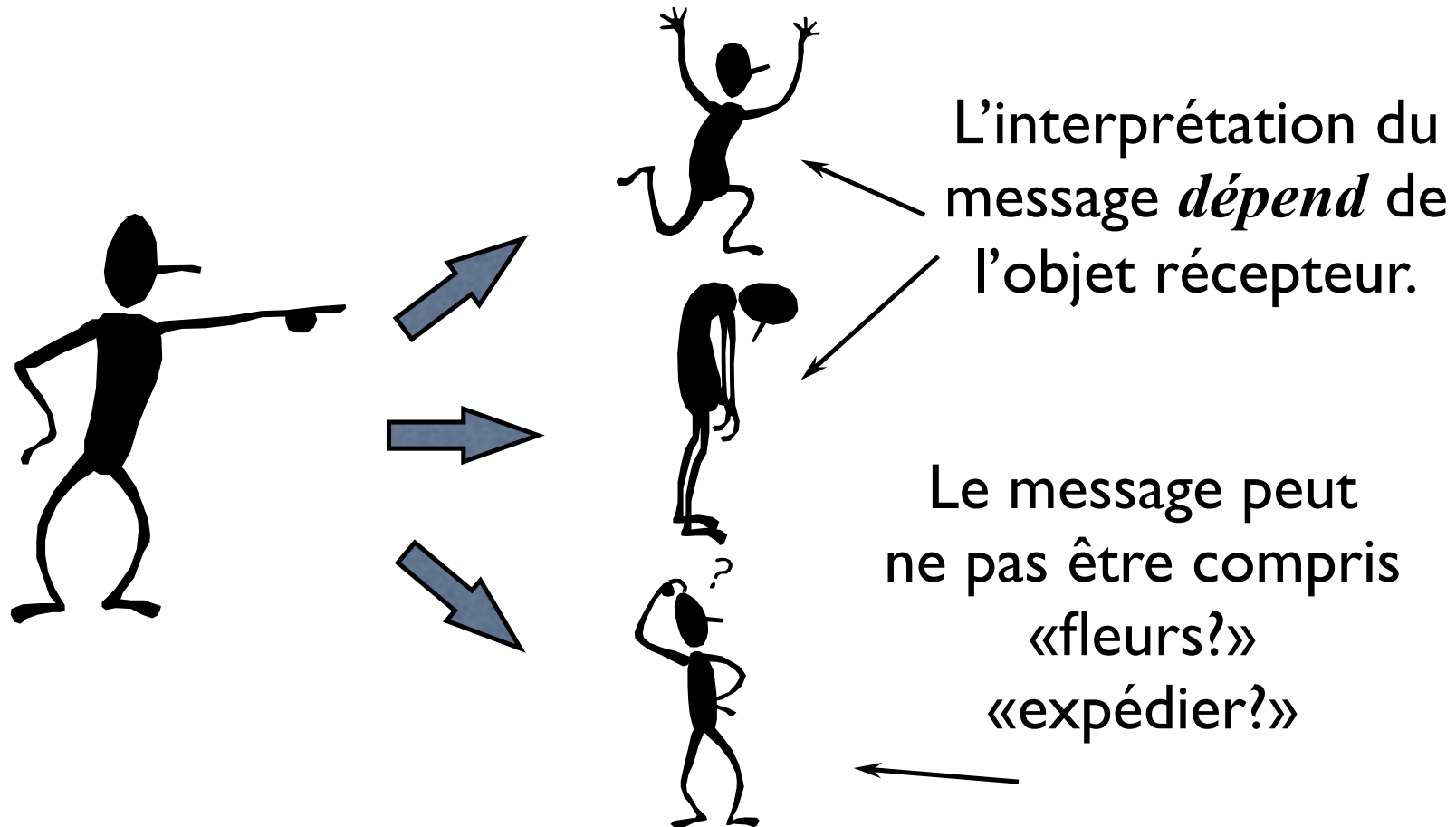
Il y a la notion de confiance
dans le travail des autres.

L'envoi d'un message

L'objet *émetteur* [Pierre] demande a gaston
(**récepteur** du message) d'envoyer des roses

```
gaston envoyerN: 4 fleur: #rose a:  
beatrice
```


L'interprétation du message



La liaison tardive

- L'exécution de la tâche dépend du receveur
- Différents receveurs d'un même message peuvent réagir différemment
- L'émetteur n'est pas concerné, le receveur décide
- Le choix de la réaction est *dynamique*, il dépend du *receveur*



Responsabilités d'un objet

- Un objet a un ensemble de *responsabilités* qu'il peut satisfaire ou garantir. Ces responsabilités sont réalisées par l'exécution des requêtes.
- Un protocol est un ensemble de messages compris par un objet.
- *interface* est un synonyme de *protocole*.



Plan

- Objet, requêtes et messages
- ***Instance, classes et héritage***
- Autre Exemple
- Composition d'objets



Classes et instances (i)

- Lorsqu'on se rend chez *Gaston*, on s'attend à obtenir certains *services* parce qu'on connaît ce qu'un *fleuriste* en général peut offrir comme *services*.
 - **Gaston** est une *instance* de la *classe* **Fleuriste**.
 - Tous les fleuristes instances de la classe Fleuriste partagent le même comportement.
-
- Gaston est une sorte de marchand: un fleuriste

Classes et instances

- *tous* les objets sont des instances d'une classe.
- objet == instance
- C'est la classe du *receveur* qui détermine quelle méthode sera exécutée en réponse à un message.
- Tous les objets d'une classe donnée utilisent la **même** méthode en réponse a un **même** message.



Hiérarchie de classes - Héritage

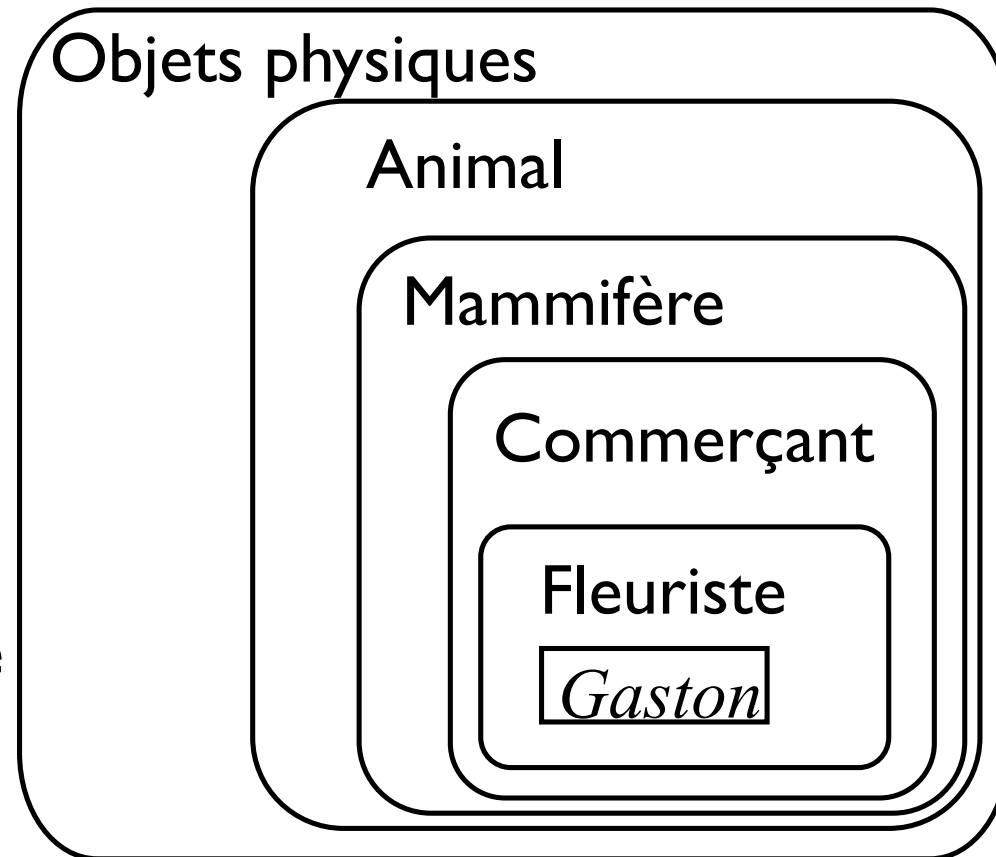
- Je connais bon nombre de *comportements* de Gaston, le fleuriste, non pas parce que je sais qu'il est *fleuriste*, mais aussi parce qu'il est un *commerçant*.
- De ce fait, je m'attends à ce qu'il me demande de l'argent pour ses services et qu'il me donne un reçu en échange.
- Ces *comportements* sont aussi vrai pour l'épicier, le coiffeur ou le préposé au club vidéo.

Hiérarchie de classes - Héritage

- Étant donné que **Fleuriste** est plus *spécialisé* que **Commerçant**, ce qui est vrai pour ce dernier est vrai aussi pour **Fleuriste**, et donc pour *Gaston*.
- En organisant nos connaissances en terme de catégories ou hiérarchie, on obtient une structure qui classifie et qui permet une compréhension rapide du problème.

Les catégories englobant Gaston

Difference entre
Commerçant/Fleuriste
et
Fleuriste/Gaston

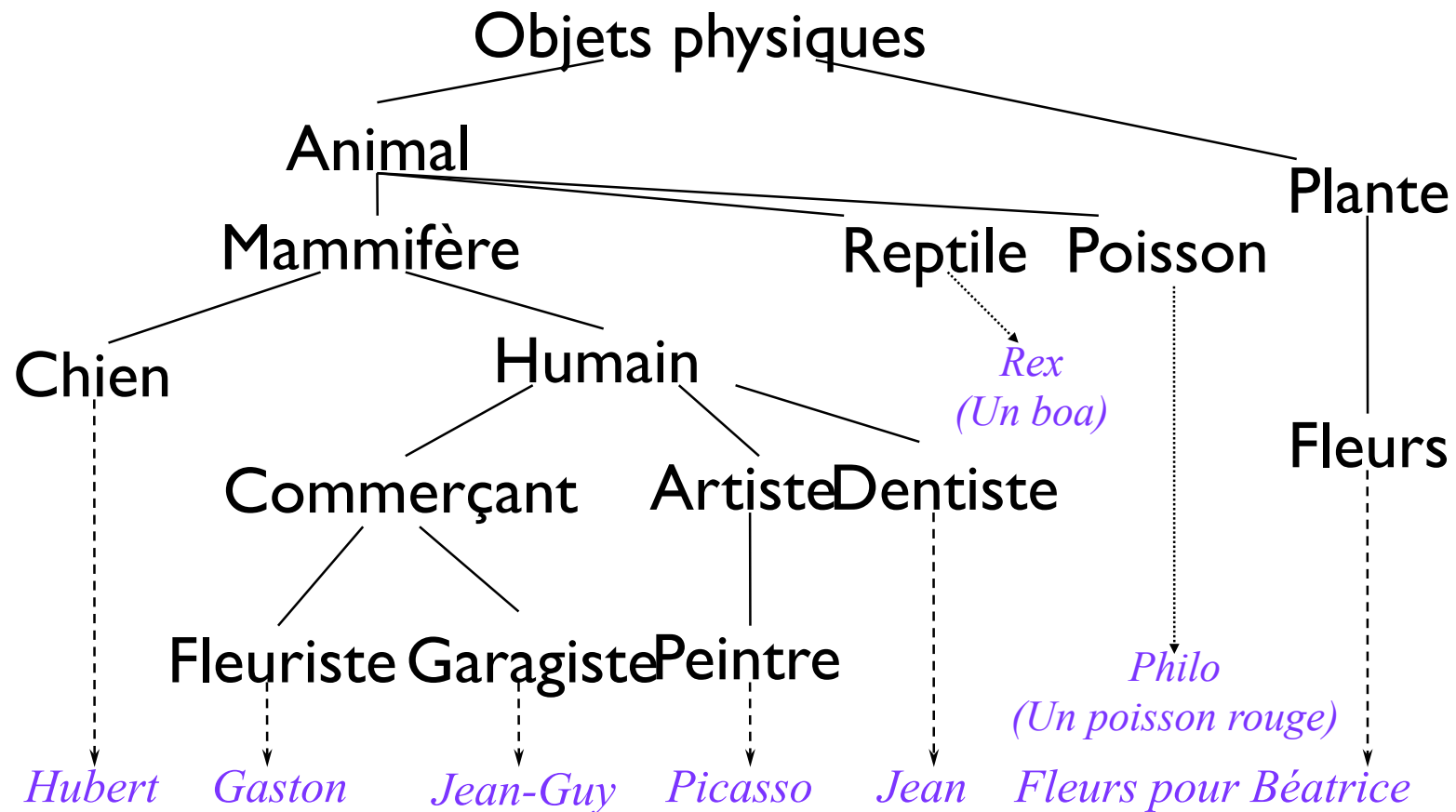


L'héritage

- Les classes peuvent être organisées dans des structures hiérarchiques d'héritage.
- Une *classe enfant* hérite des caractéristiques de la *classe parent* plus haut dans l'arbre hiérarchique.
- *Fleuriste* a les caractéristiques de *Commerçant*, *Commerçant* a les caractéristiques d'*Humain*, etc...



Hiérarchie de classes



Un objet

- Un objet possède 2 types de propriétés :
 - Statiques : état, attributs.
 - Dynamiques : comportements, services.
- Le chat de ma voisine :
 - Statiques :
 - son nom (Phillipon)
 - son poids (18 livres – je sais il est gros...)
 - sa couleur (noir et blanc)
 - Dynamiques :
 - il miaule quand il a faim
 - il se couche sur son cousin près du sofa
 - il mange des mouches

Héritage: Réutilisation

- On ne veut pas réécrire tout le code
- On veut juste décrire la différence ou particulariser du code déjà existant
- Une sousclasse est presque comme sa superclasse
 - ajoute de l'état
 - ajoute du comportement
 - spécialise le comportement



Plan

- Objet, requêtes et messages
- Instance, classes et héritage
- ***Autre exemple***
- Composition d'objets



Une classe

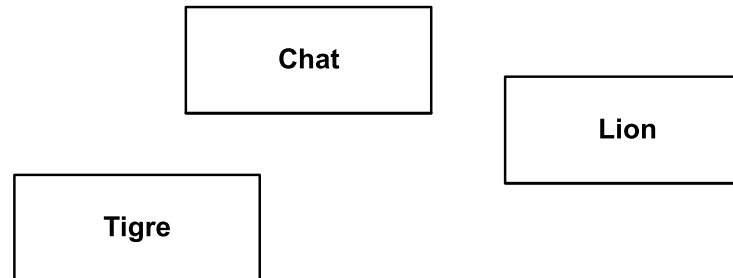
- Phillipon est un digne représentant de la race des chats. Autrement dit, Phillipon est une instance de la classe des chats.
- Tous les chats ont des propriétés communes :
 - ils miaulent (propriété dynamique)
 - ils ont une moustache (propriété statique)
 - etc.
- Chat est ici une classe représentant un ensemble d'objets ayant des attributs et des comportements semblables.

Une classe (2)

- La classe Chat
 - On ne conserve que les caractéristiques qui nous intéressent :
 - Les attributs : couleurs, race, nom, etc.
 - Les comportements : manger, dormir, miauler, etc.
 - Offre des services (miauler, manger, etc.)
 - Protège les données (encapsulation des propriétés)
 - Peut créer des instances (objets).
 -

Vers une hiérarchie de classes

- Si on poursuit l'analogie, il existe :



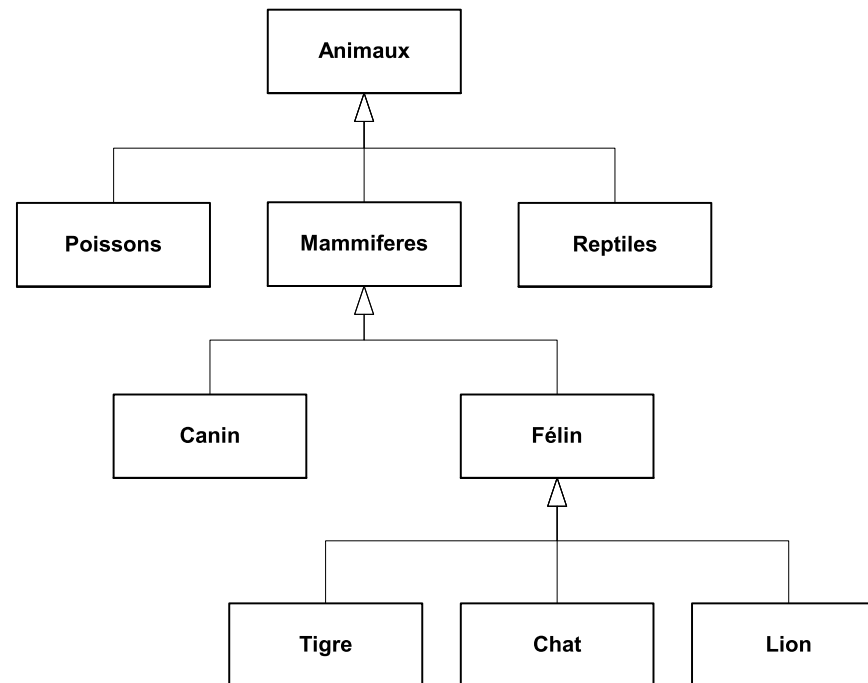
- Pas pareils, mais tous partagent des caractéristiques :
 - ils mangent de la viande
 - leurs griffes peuvent se rétracter
 - leurs molaires sont coupantes
 - ils ont une moustache

Vers une hiérarchie de classes

- Nous pouvons donc généraliser les races chat, lion et tigre en une super-race nommée félins.
- La super-race définit des propriétés et des comportements qui sont valides autant pour les chats que pour les lions et les tigres.
- En termes informatiques, la classe Félin correspond à une super-classe des classes Chat, Lion et Tigre.
- Ces dernières sont des classes dérivées de la classe Félin et qu'elles héritent des propriétés et des comportements de leur super-classe. C'est le concept d'héritage.

Vers une hiérarchie de classes (3)

- Le concept d'héritage peut être étendu pour donner un arbre appelé **arbre d'héritage**.



Vers une hiérarchie de classes (4)

- On s'attend à certaines caractéristiques de Phillipon parce qu'il est un **instance** (objet) de la classe **Chat** (miauler, dormir, moustaches).
- On s'attend à ce qu'il puisse se mouvoir non pas parce qu'il est un chat, mais parce qu'il **appartient** à la classe **Animal**, donc par héritage.

Liaison tardive

- Les chats, les serpents et les poissons partagent tous un comportement commun hérité de la classe des animaux : ils ont la capacité de se mouvoir.
- Tous le font, mais chacun à sa manière :
 - le chat marche
 - le serpent rampe
 - le poisson nage
- Si on donne l'ordre de se mouvoir, chaque animal va pouvoir répondre à cet ordre, à sa manière propre.




Polymorphisme

- Littéralement plusieurs formes (plusieurs réactions possibles...)
- souvent confondu avec la liaison tardive...
- Polymorphisme décrit le fait qu'une variable peut contenir des valeurs (objets) étant différentes.
- Par exemple une variable de type Animal peut contenir des objets instances de souclasse d'Animal (Chat, Serpent....)
- Par abus de langage, on accepte de confondre liaison tardive et polymorphisme

Encapsulation

- Phillipon miaule, marche, mange :
 - Nous connaissons les manifestations extérieures du miaulement, mais la façon dont le son est produit nous est masqué.
- Le chat devient alors une abstraction dont on ne montre seulement que les quelques comportements.
- Une interface représente cette abstraction.
 - L'interface énonce les services offerts et les détails de fonctionnement sont cachés.
- Emphase sur la compréhension des éléments importants, sans se soucier de l'implantation.



- 
- Objet, requêtes et messages
 - ***Instance, classes et héritage***
 - Autre exemple
 - Composition d'objets



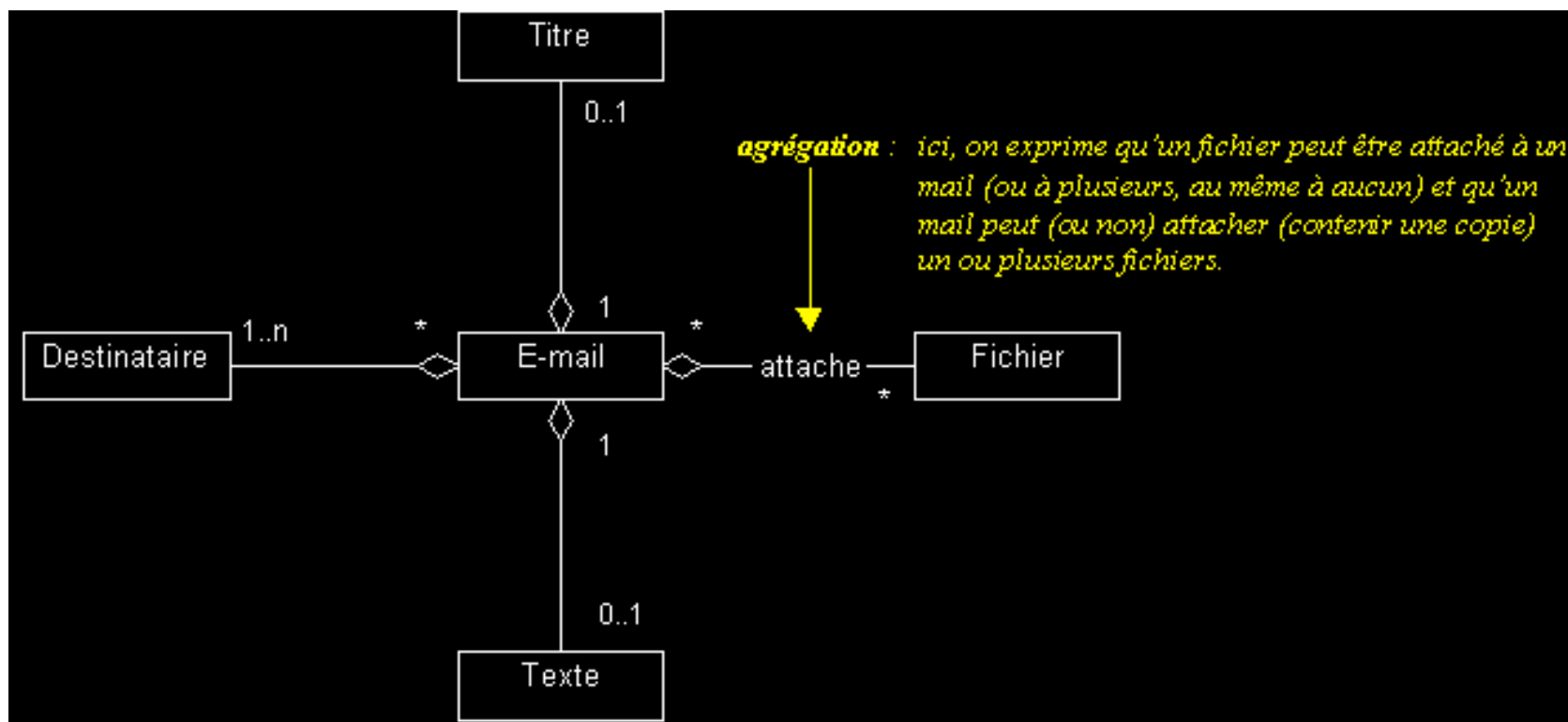
Agrégation

- L'agrégation est une association non symétrique, qui exprime un couplage fort et une relation de subordination.
Elle représente une relation de type "ensemble / élément".
- Peut (mais pas nécessairement) exprimer :
 - qu'une classe (un "élément") fait partie d'une autre ("l'agrégat"),
 - qu'un changement d'état d'une classe, entraîne un changement d'état d'une autre,
 - qu'une action sur une classe, entraîne une action sur une autre.

Aggrégation d'objets

- Un objet a un état composé par d'autres objets
- Une voiture est composée
 - de 3,4 ou 5 portes
 - d'un moteur
 - de roues
- Chaque voiture instance de la class voiture a un nombre particulier de portes, moteur...roues.
- Une classe spécifie la **structure** de ses instances
- Ses instances spécifient les **valeurs**





- Objet, requêtes et messages
- ***Instance, classes et héritage***
- Autre Exemple
- Composition d'objets

Résumé

Les **objets** répondent à des **messages** en exécutant des **méthodes**

Un objet peut **déléguer** à d'autres objets pour accomplir ses tâches

Le récepteur du message **détermine** l'exécution de la méthode associée

Une classe décrit l'état et spécifie le comportement de toutes ses instances

Une classe crée des **instances** qui partagent toutes le même comportement

Une classe hérite d'autres classes dans des hiérarchies de spécialisation