Généralisation et spécialisation

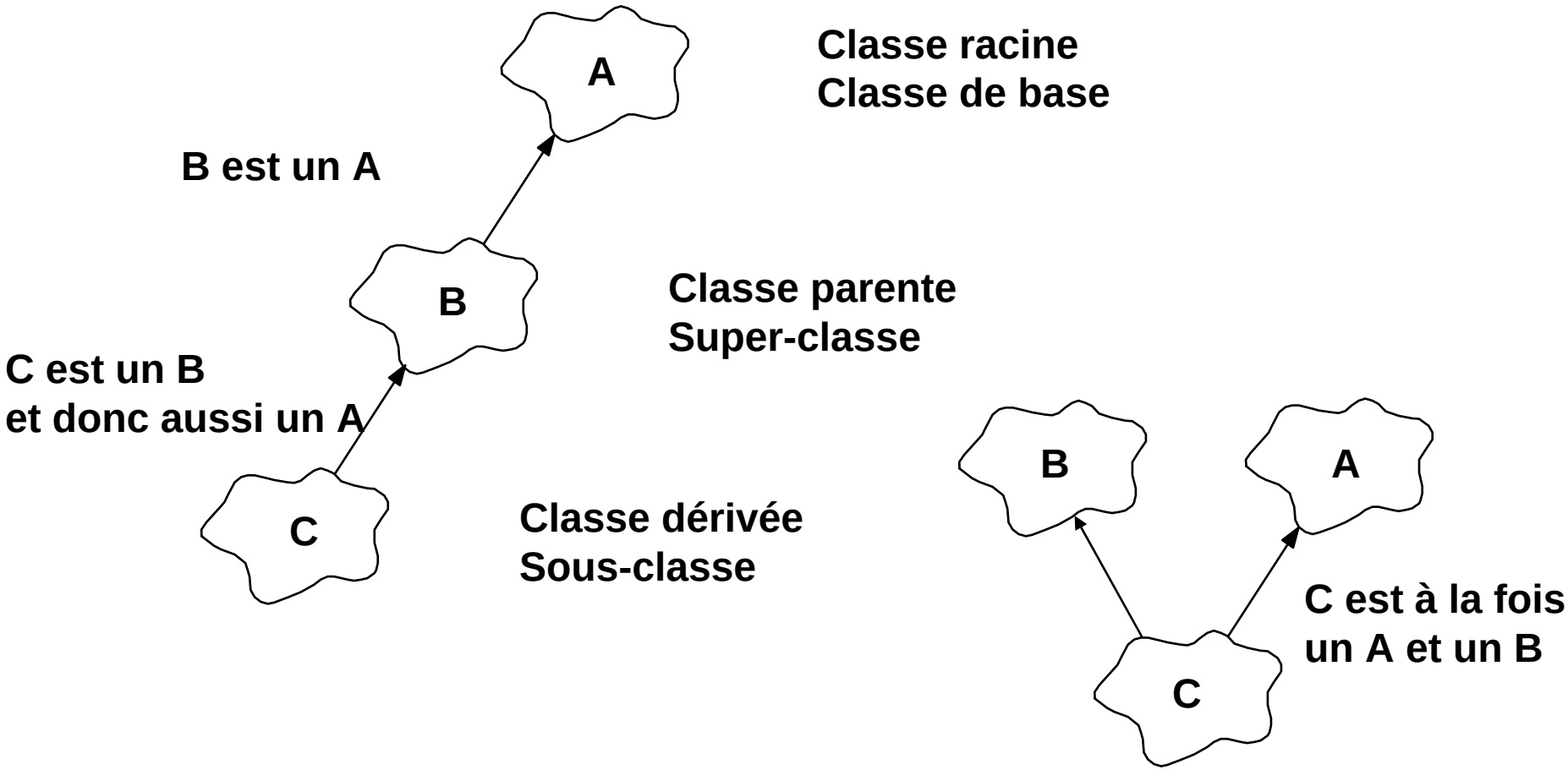
Rappels sur l'héritage

- Relation entre classes qui exprime qu'une classe est conforme au contrat d'une autre classe
- Une relation d'héritage se lit : "est un" ou "est une sorte de"
- L'héritage permet de construire des hiérarchies de classes
- L'héritage est une relation non réflexive, transitive et anti-symétrique
- L'héritage peut être simple ou multiple

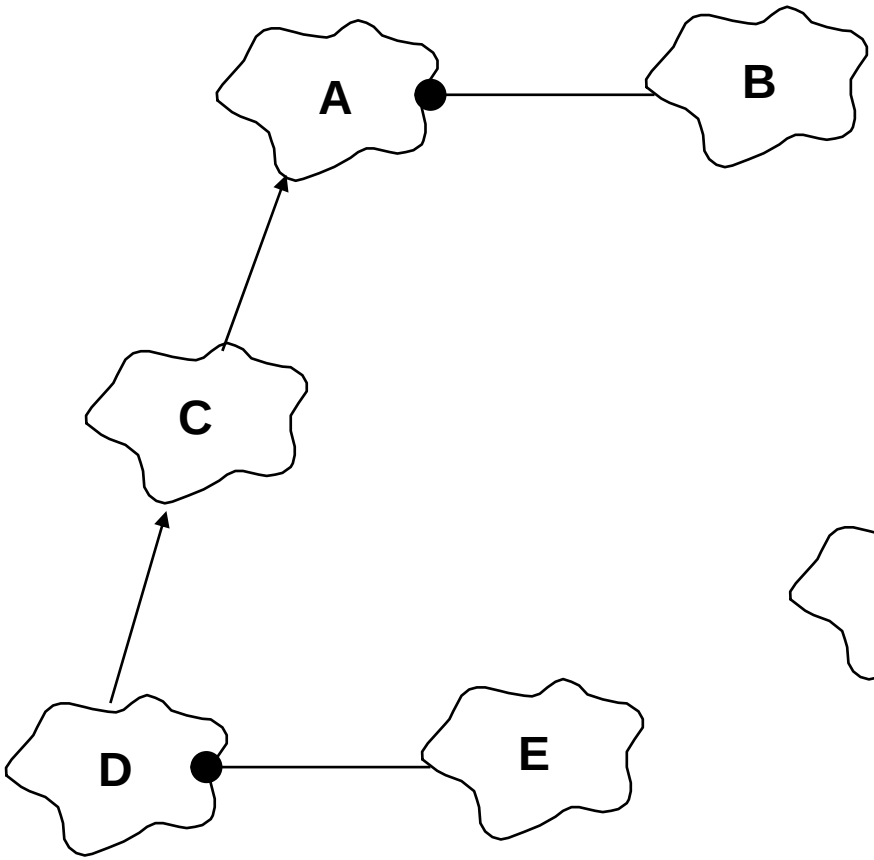
Rappels sur l'héritage

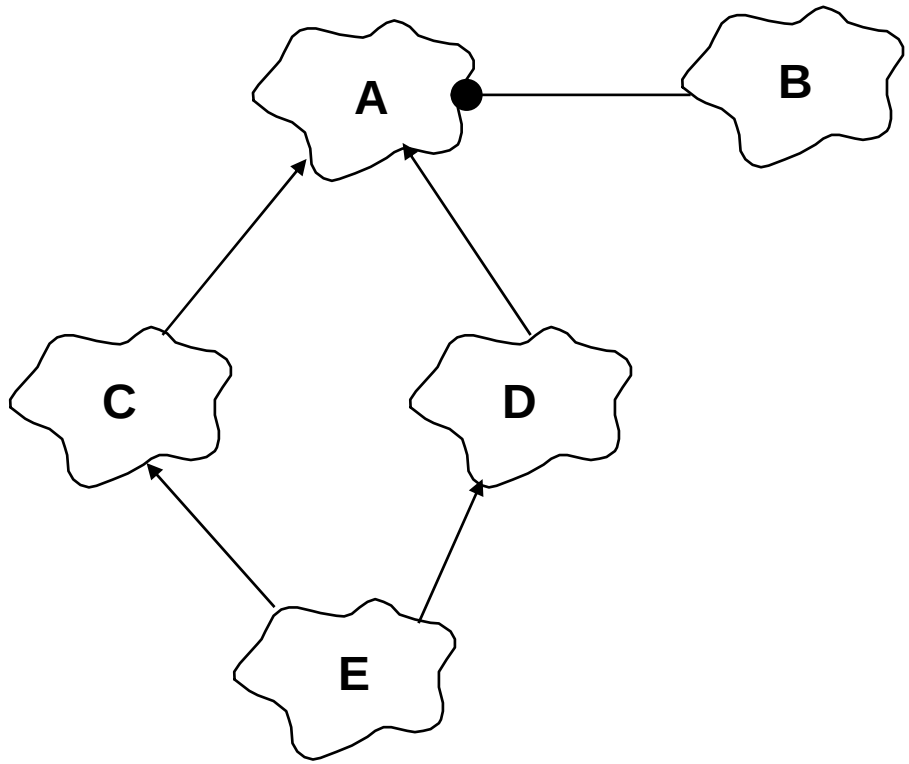
- Il n'y a pas d'héritage partiel, toutes les propriétés de la classe parentes sont héritées intégralement dans la classe fille
- L'héritage concerne l'état, le comportement et les contraintes
- Le besoin d'hériter partiellement est le signe que la relation considérée n'est pas vraiment une relation d'héritage

Exemples d'héritage



Exemples d'héritage

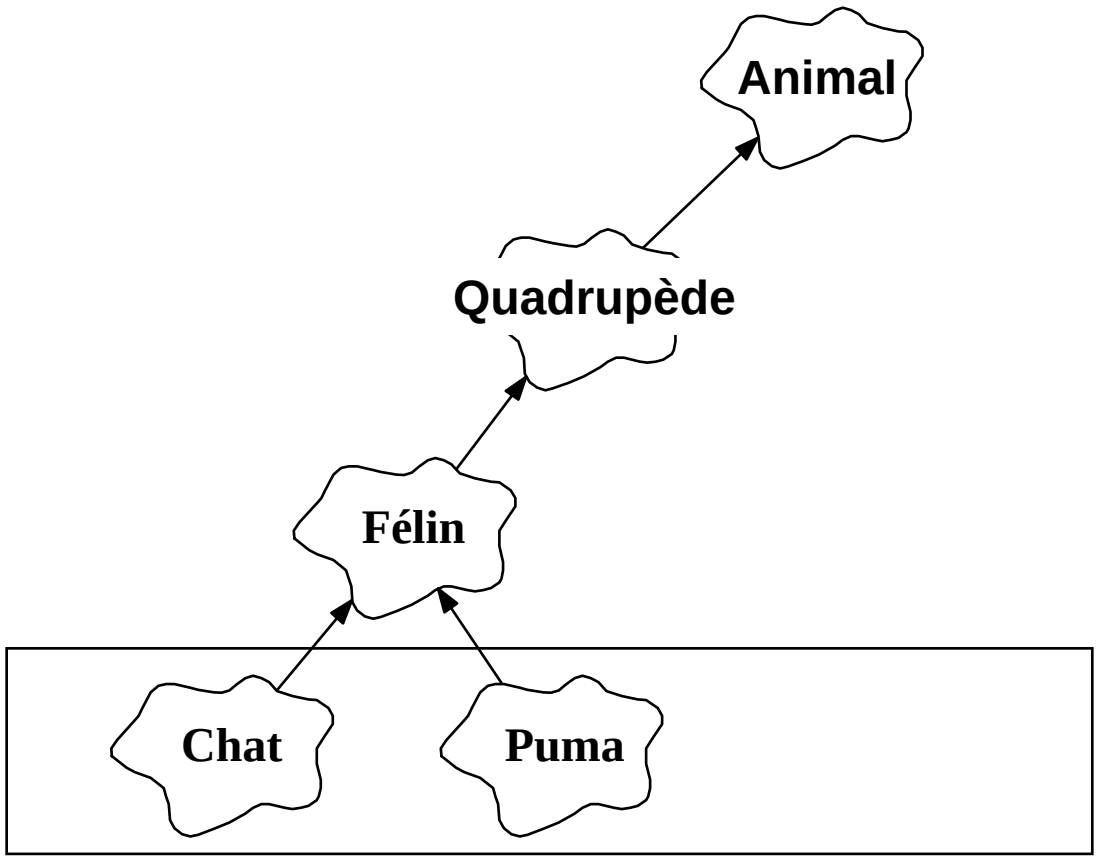


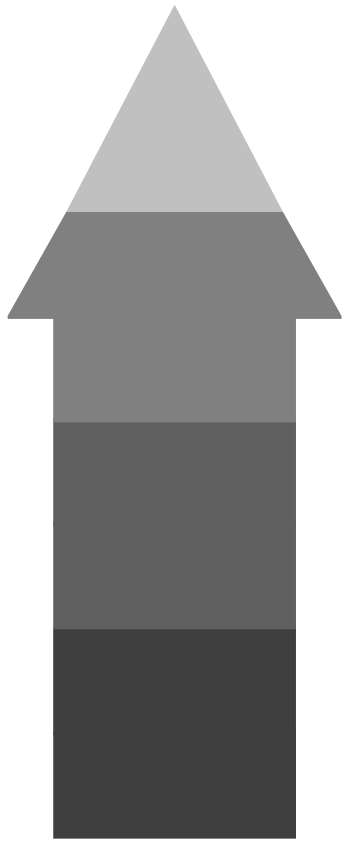


La généralisation

- Abstractions d'abstractions
- Hiérarchie d'abstractions
- Factorisation des éléments communs à un ensemble de classes
- Chaque niveau d'abstraction devient de plus en plus général
- Démarche difficile

Exemple de généralisation

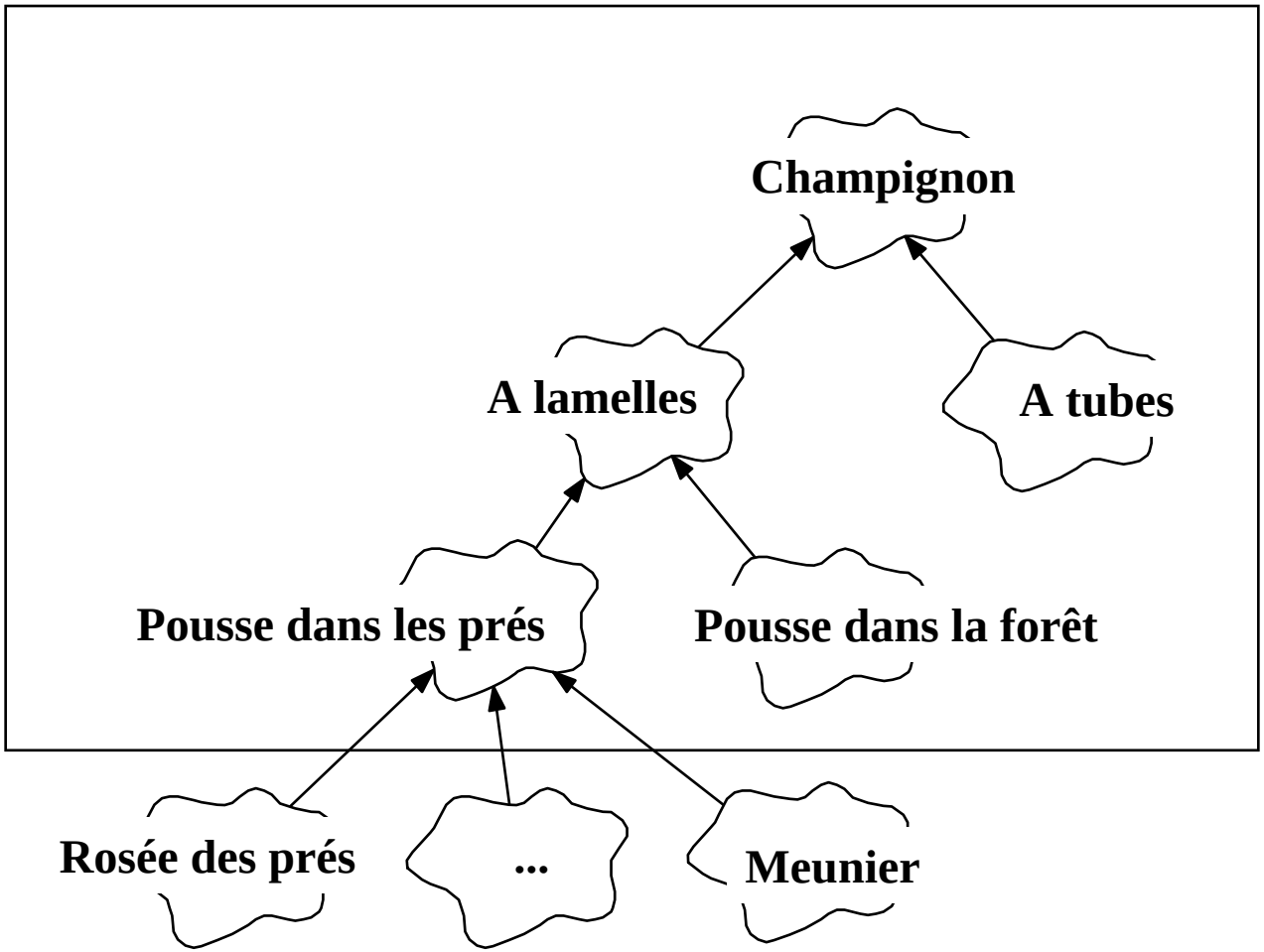


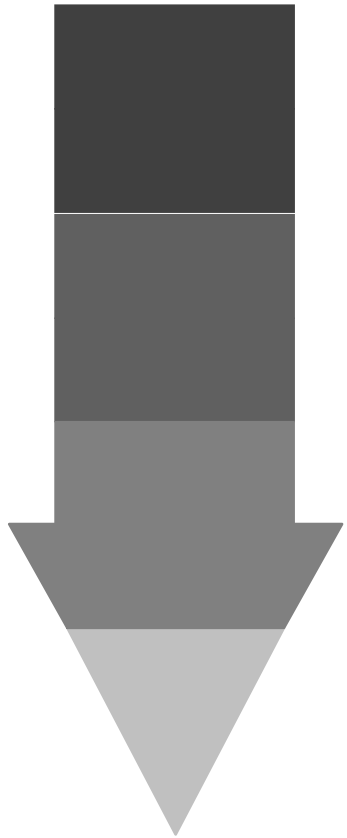


La spécialisation

- Opération inverse de la généralisation
- Extension cohérente d'un ensemble de classes
- La spécialisation permet de capturer les spécificités d'un ensemble d'objets non discriminés par la structure de classe existante
- Démarche plus simple que la généralisation car basée sur l'extension d'un ensemble pré-existant

Exemple de spécialisation



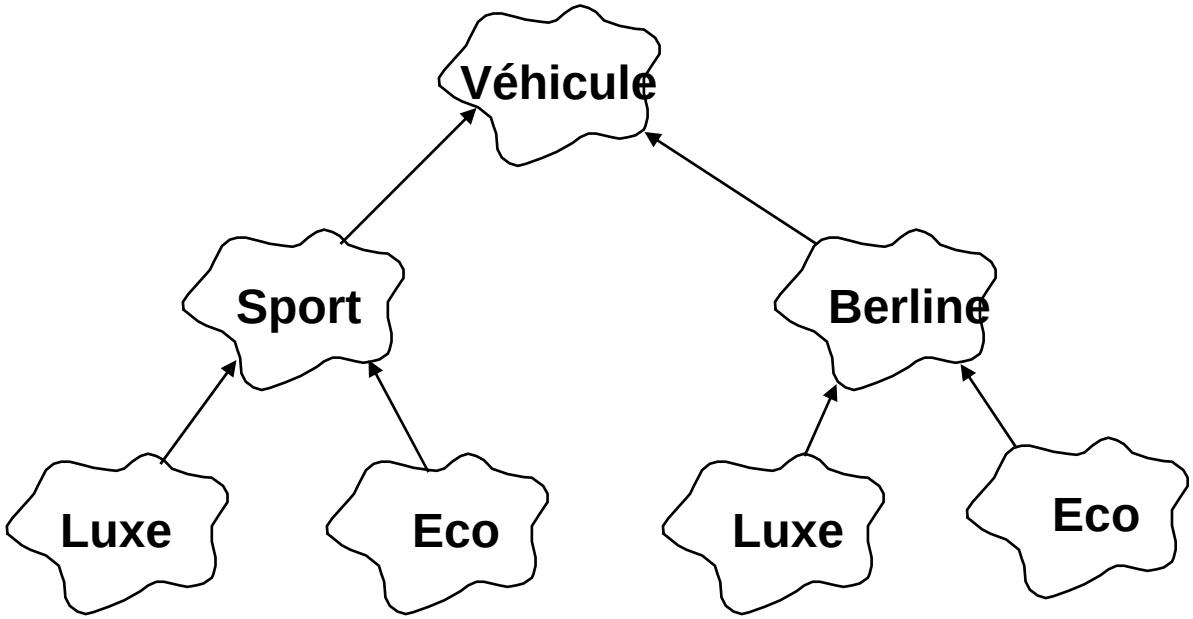


Caractéristiques des classifications

- Les classifications s'effectuent suivant des critères dépendant du point de vue
- Ainsi pour les animaux :
  - le nombre de pattes
  - le type de nourriture
  - le pelage, les plumes ou les écailles
  - le mode de reproduction
- Les critères une fois déterminés doivent être appliqués de manière cohérente et uniforme pour toute la classification

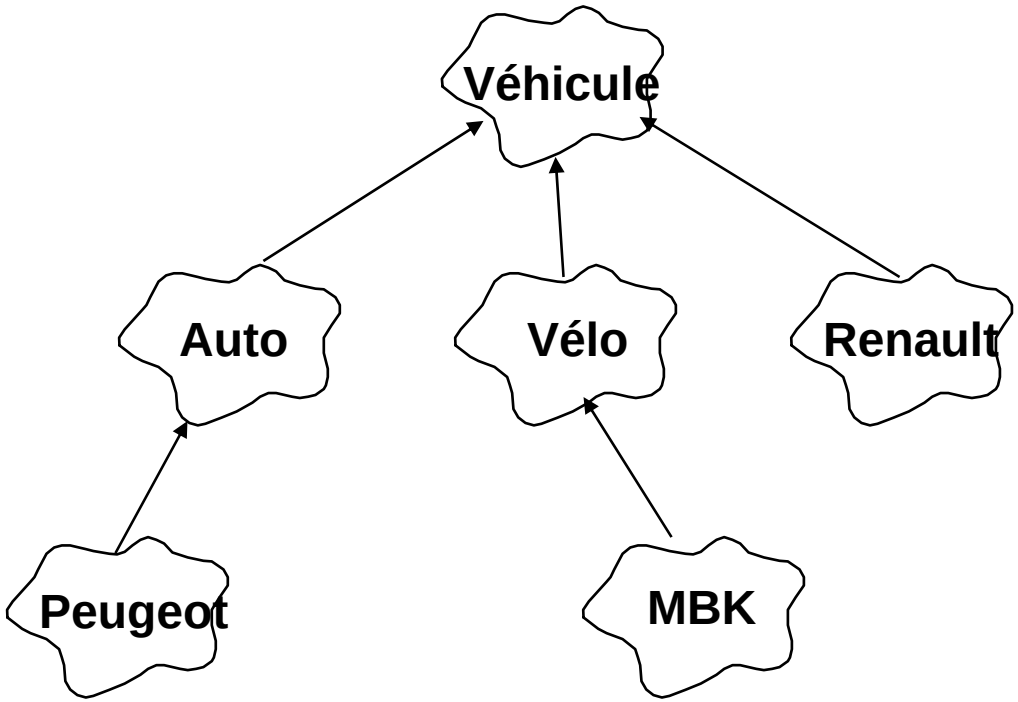
Caractéristiques des classifications

- **L'ordre d'application de critères orthogonaux peut être difficile à déterminer au sein d'une décomposition unique**



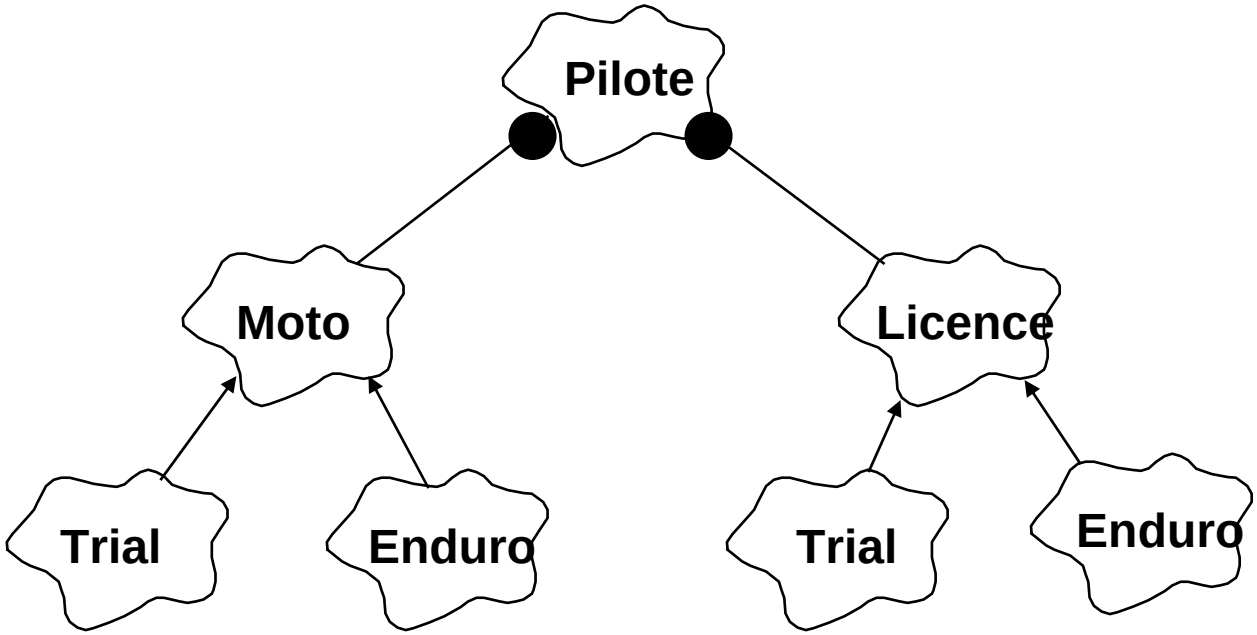
Caractéristiques des classifications

- **Les classifications doivent comporter des niveaux d'abstractions équilibrés**



Caractéristiques des classifications

- **Les critères partagés tendent à induire des décompositions covariantes**



Caractéristiques des classifications

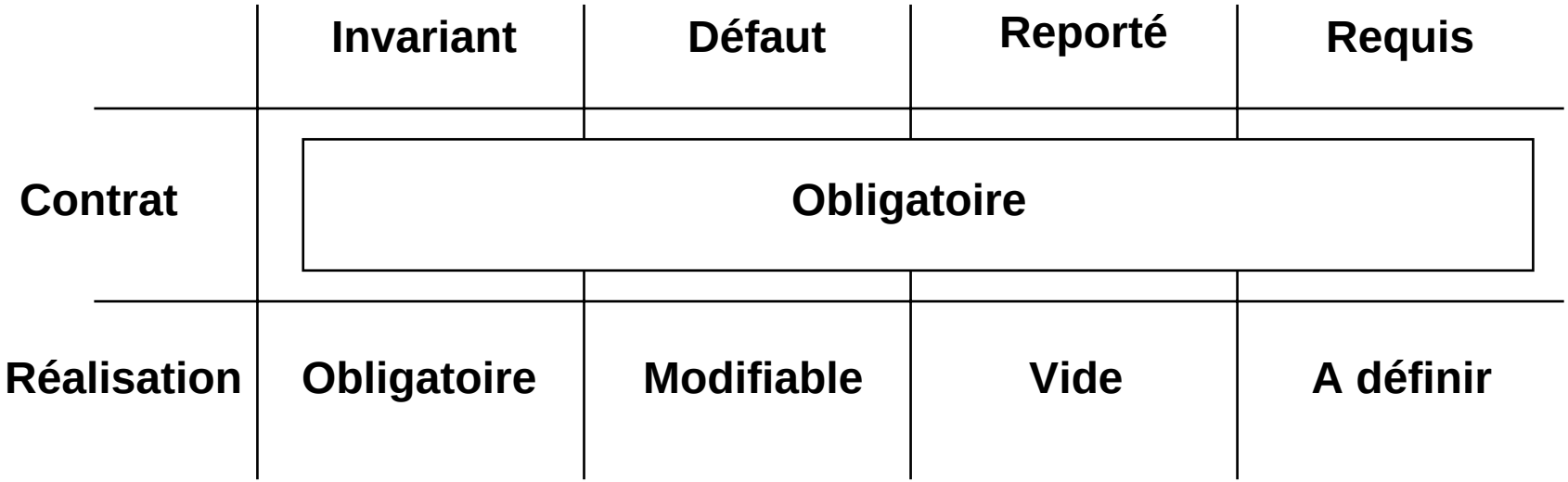
- Les bonnes classifications sont stables et extensibles
- Les bonnes classifications peuvent comporter des exceptions inclassables suivant les critères retenues
  - Le grand panda est proche des ursidés
  - Le petit panda est proche des ratons-laveurs
- Les bonnes classifications permettent de bien discriminer les objets du domaine de l'application, tout en factorisant la description de leurs caractéristiques communes

Héritage multiple

- Une classe peut dériver de plusieurs super-classes
- Un objet peut cumulerdes caractéristiques (indépendantes) décrites dans deux classes différentes
- Un moyen pour mélanger des caractéristiques et éviter la duplication
- Induit des problèmes de nommage
- Sujet à controverse, à utiliser prudemment

Héritage du contrat et héritage de la réalisation

- Le contrat du comportement est toujours hérité dans son intégralité
- La réalisation du comportement n'est pas toujours héritée



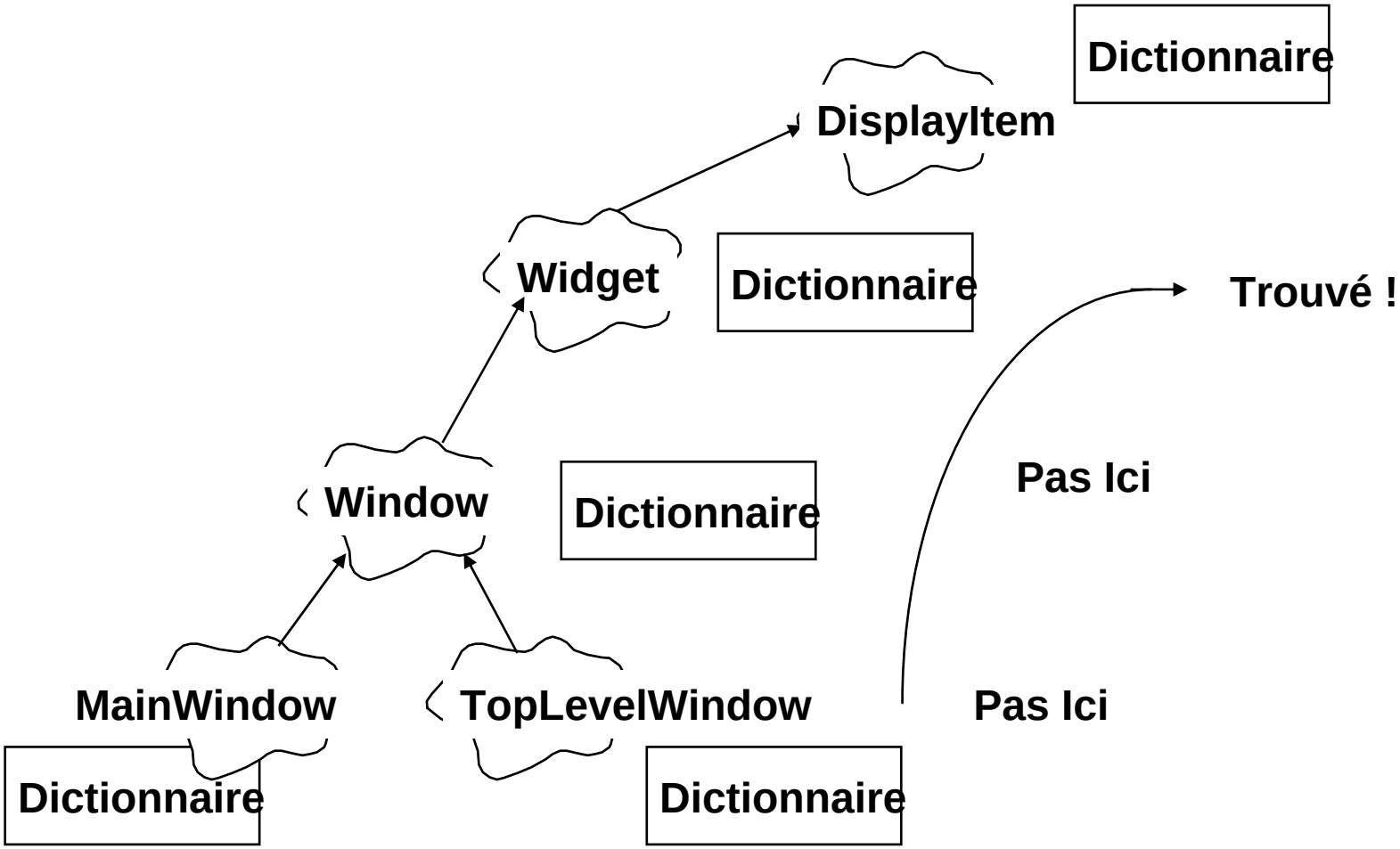
La liaison dynamique

- La liaison statique est une édition de liens dans laquelle l'association nom-classe est réalisée lorsque le nom est déclarée (à la compilation), c'est-à-dire avant la création de l'objet que le nom désigne
- La liaison dynamique est une édition de liens dans laquelle l'association nom-classe n'est réalisée que lorsque l'objet désigné par le nom est créé
- La liaison statique force la définition précise du type des objets dès la compilation
- La liaison dynamique permet de déterminer le type précis d'un objet durant l'exécution

Le déclenchement des opérations

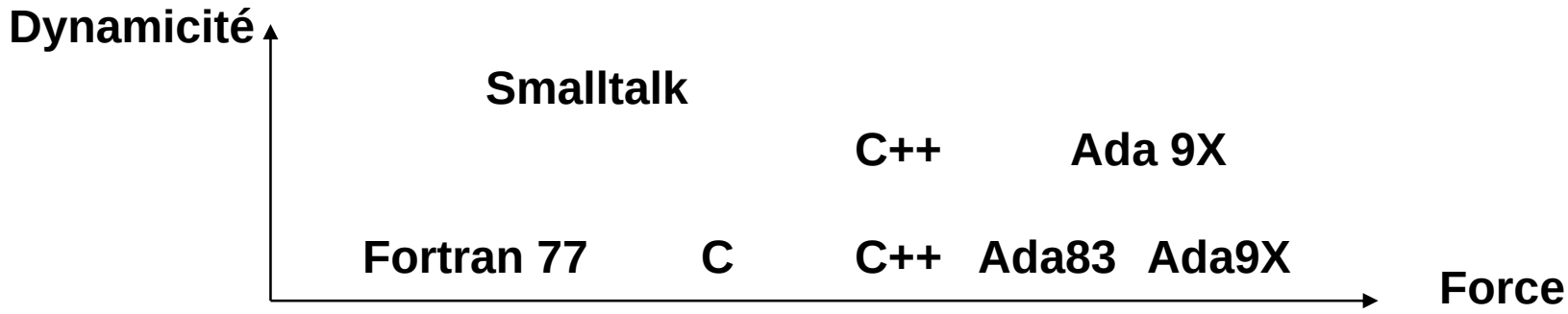
- Dans les langages traditionnels du type Pascal, le déclenchement d'une opération est une activité complètement statique
- Dans les langages à liaison dynamique le déclenchement d'une opération requiert une activité durant l'exécution pour retrouver le code de la méthode qui correspond au message reçu
- La situation se complique encore en présence d'héritage, car dans ce cas les opérations doivent être recherchées dynamiquement tout le long de l'arbre d'héritage

Exemple de Smalltalk



Influence du typage

- Lorsque la liaison dynamique n'est pas requise les passages de messages peuvent être avantageusement remplacés par de simples appels de procédures
- Dans le cas d'un langage non typé tel Smalltalk ceci n'est pas réalisable
- Dans le cas de langage plus fortement typés, comme C++ ou Ada 9X, le développeur peut choisir le type de liaison méthode par méthode



Exemple de C++

- C++ est un langage typé
- Le mot clé `virtual` permet de désigner les opérations sujettes à liaison dynamique
- Les compilateurs associent à chaque classe une table qui contient des pointeurs vers la plus proche implémentation des opérations virtuelles
- La recherche durant l'exécution est éliminée, le coût du déclenchement est proche du coût d'un appel de procédure liée statiquement

Le polymorphisme

- **Littéralement : qui peut prendre plusieurs formes**
- **Un concept de la théorie des types, selon lequel un nom peut désigner des objets instances de classes différentes issues d'une même arborescence**
- **La possibilité de déclencher des comportements différents en réponse à un ensemble commun d'opérations**
- **La possibilité de manipuler des objets par l'intermédiaire d'un contrat de niveau d'abstraction supérieur**

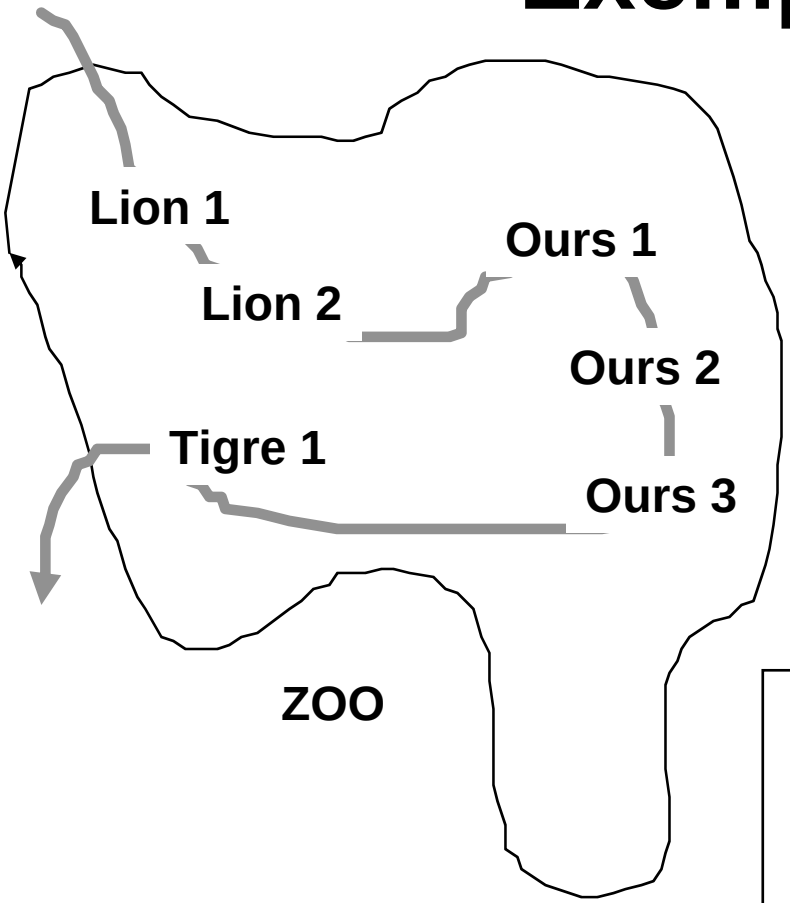
Le polymorphisme

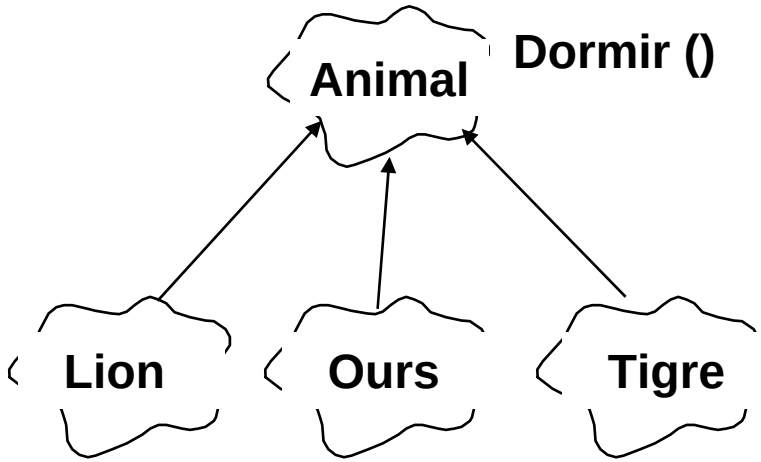
- L'interaction entre l'héritage et la liaison dynamique
- Le typage permet au compilateur de garantir que le message sera compris par le destinataire
- Ne pas confondre avec la surcharge des opérations, qui peut être résolue statiquement en fonction de la signature
- Un moyen d'écrire du code plus abstrait, défini en fonction des termes du contrat d'une des super-classes

L’extensibilité

- **L’héritage et la liaison dynamique apportent une grande souplesse pour l’extension de code existant**
- **La généricité permet l’extension uniquement dans les limites connues à la compilation du module parent**
- **L’héritage permet une extension plus large, à la fois de l’état et du comportement**
- **La polymorphisme simplifie la maintenance du code et tend à supprimer l’usage d’instructions à branchements multiples (*case*)**

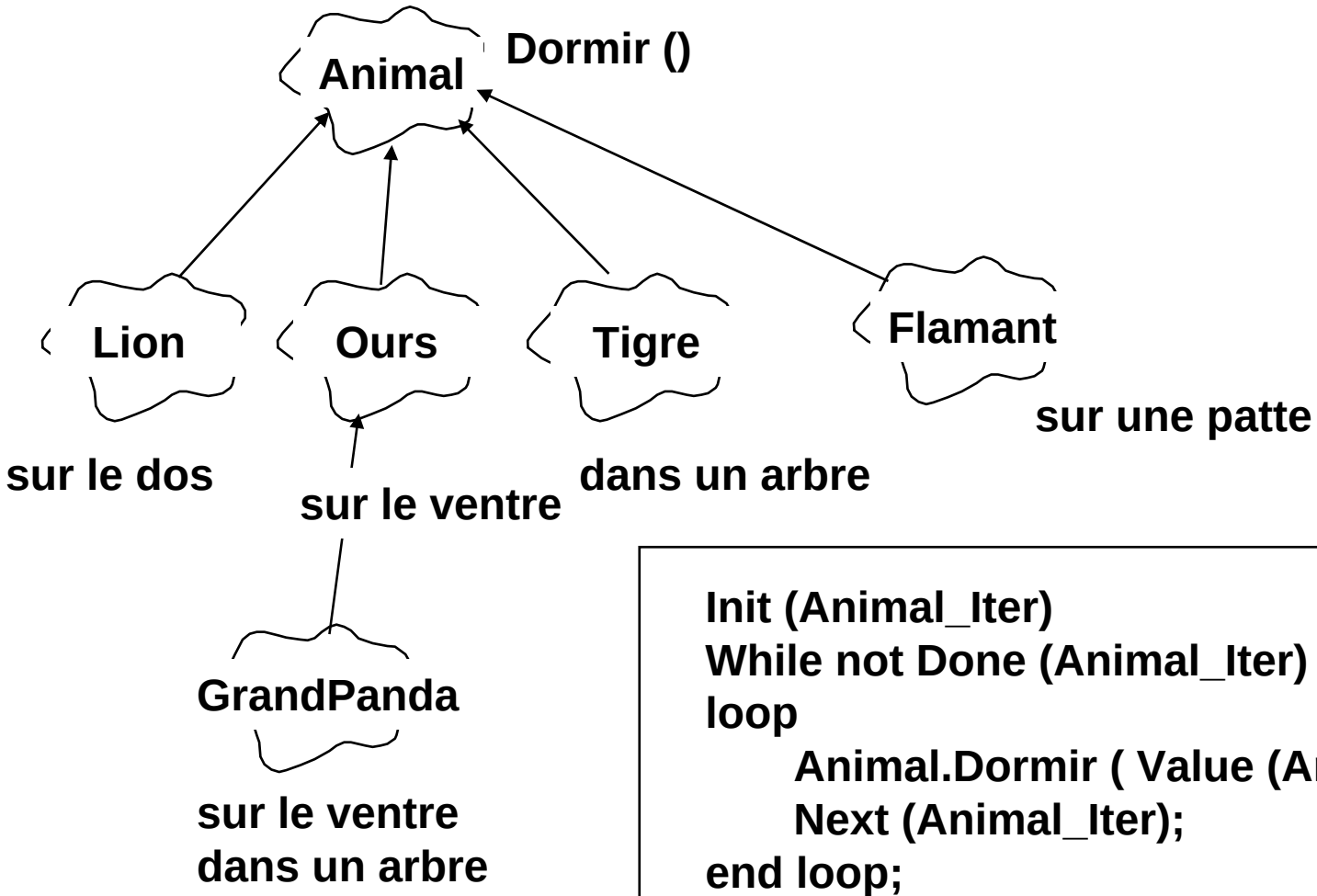
Exemple du ZOO






```
Init (Animal_Iter)
While not Done (Animal_Iter)
loop
      Animal.Dormir ( Value (Animal_Iter) );
      Next (Animal_Iter);
end loop;
```

Extension du ZOO




```
Init (Animal_Iter)
While not Done (Animal_Iter)
loop
      Animal.Dormir ( Value (Animal_Iter) );
      Next (Animal_Iter);
end loop;
```

Suppression des instructions *case*

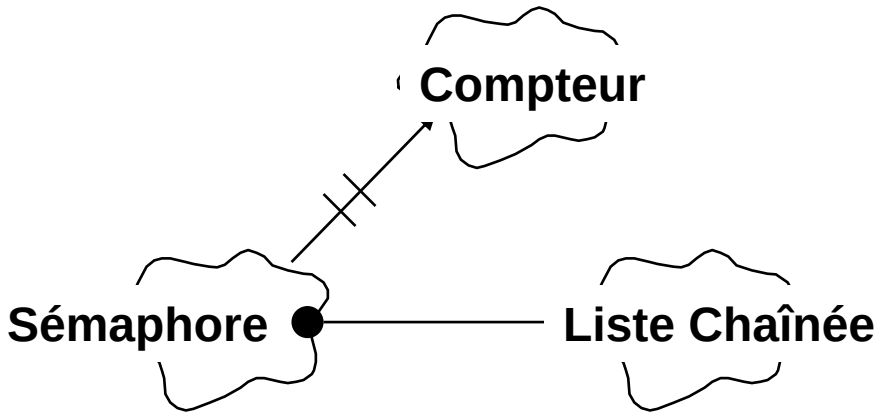
```
Init ( Animal_Iter )
While not Done ( Animal_Iter );
loop
    Current_Animal : Animal := Value ( Animal_Iter );
    case Current_Animal.kind
        when Lion
            Lion.Dormir ( Current_Animal );
        when Tigre
            Tigre.Dormir ( Current_Animal );
        when Ours
            Ours.Dormir ( Current_Animal );
        when Flamant
            Flamant.Dormir ( Current_Animal );
    end case;
    Next ( Animal_Iter );
end loop;
```

Le principe de substitution

- Confusion entre héritage (is) et aggrégation (has)
- L'aggrégation peut remplacer une relation d'héritage, l'inverse n'est pas vrai
- Puisque l'héritage décrit une relation "être une sorte de", un objet instance d'une classe dérivée doit pouvoir remplacer un objet instance de la classe parente, de façon totalement transparente
- La sémantique d'un programme ne doit pas être modifiée lorsque n'importe quel objet d'une classe C quelconque est remplacé par un objet instance d'une des sous-classes de C

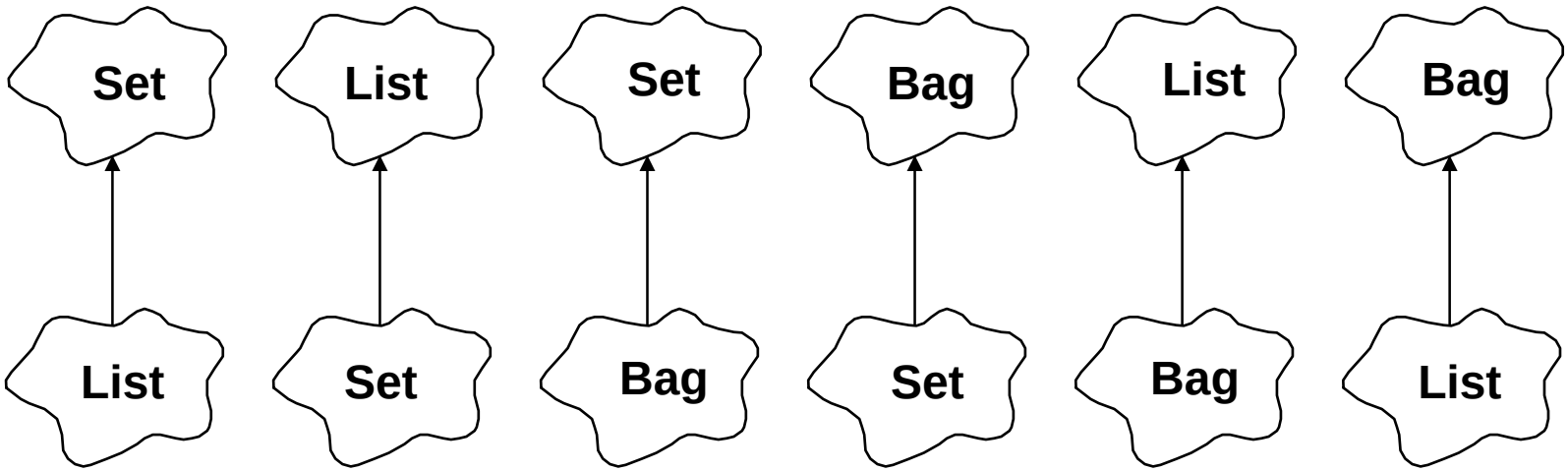
Le principe de substitution

- A la base du bon fonctionnement du polymorphisme
- Ne pas confondre héritage visible avec héritage de l'implémentation



Application du principe de substitution

- Quelles relations d'héritage sont correctes ?



Conclusion

- L’héritage permet de décrire des structures entre classes basées sur les notions de spécialisation et de généralisation
- L’héritage combiné à la liaison dynamique (polymorphisme) s’implifie l’écriture et la maintenance du code
- L’héritage ne remplace pas l’aggrégation, il convient de distinguer l’héritage visible de l’héritage pour l’implémentation