

# Rappels sur l'objet

M1 MIAGE - SIMA - 2008-2009

Yannick Prié

UFR Informatique - Université Claude Bernard Lyon 1



# Objectifs de ce cours

- Rappels sur les concepts fondamentaux liés à la programmation orientée objet
- TP noté pour remettre en action tous ces concepts



# Objets et catégories d'objets

- Objets du monde
  - objets « concrets »
    - cette pierre, ma télévision, ta voiture
    - plus ou moins coopératifs
  - objets « abstraits », « conceptuels »
    - mon compte bancaire, le langage de programmation que j'utilise, la liberté
- Inséparables des *catégories* d'objets
  - les pierres, les télévisions, les langages de programmation, les comptes bancaires, *etc.*
- Toujours relatifs à un certain contexte
  - choix de découpage du réel lié à une pratique
  - exemple : le professionnel du bâtiment et le néophyte ne voient pas la même chose en regardant un immeuble



# Objets et abstraction

## ■ Objets

- tout ce qui nous permet de réfléchir, parler, manipuler des concepts, avec
  - un certain nombre de propriétés les caractérisant
  - un certain nombre de comportements connus

## ■ Abstraction

- passage du particulier au général
- « abstraire » des propriétés, des comportements

## ■ Classes d'objets

- regroupements d'objets ayant des propriétés et comportements similaires



# En informatique

- Programme « classique » (ex. C)
  - structures de données
    - tableau, arbre, *etc.*
  - opérations sur ces structures de données
    - fonctions
- Difficultés
  - *penser* ensemble structures de données et fonctions
  - faire *évoluer* en même temps structures de données et fonctions
  - *réutiliser* des structures/fonctions en les spécialisant
  - ...



# Idée objet en informatique

- Regrouper dans un composant
  - des caractéristiques qui concernent une entité informatique
    - structure de données
    - ensemble d'attributs
      - variables avec nom, type, valeur
  - les opérations liées à cette entité
    - ensemble de fonctions
    - appelées *méthodes*
      - avec : nom, valeur de retour, paramètres



# Objet informatique

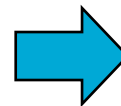
- Etat  
Ce qu'est l'objet à un instant donné
- + Comportement  
Comment l'objet réagit aux sollicitations
- + ...

# Etat d'un objet

- Ensemble des valeurs des attributs de l'objet à un instant donné
- L'état d'un objet change pendant sa vie

*Etat n*

<u>Ma voiture</u>
Marque : « Fiat »
Couleur : bleu
Masse : 943 kg
Volume essence : 32 l



*Etat n+1*

<u>Ma voiture</u>
Marque : « Fiat »
Couleur : bleu
Masse : 943 kg
Volume essence : 31 l

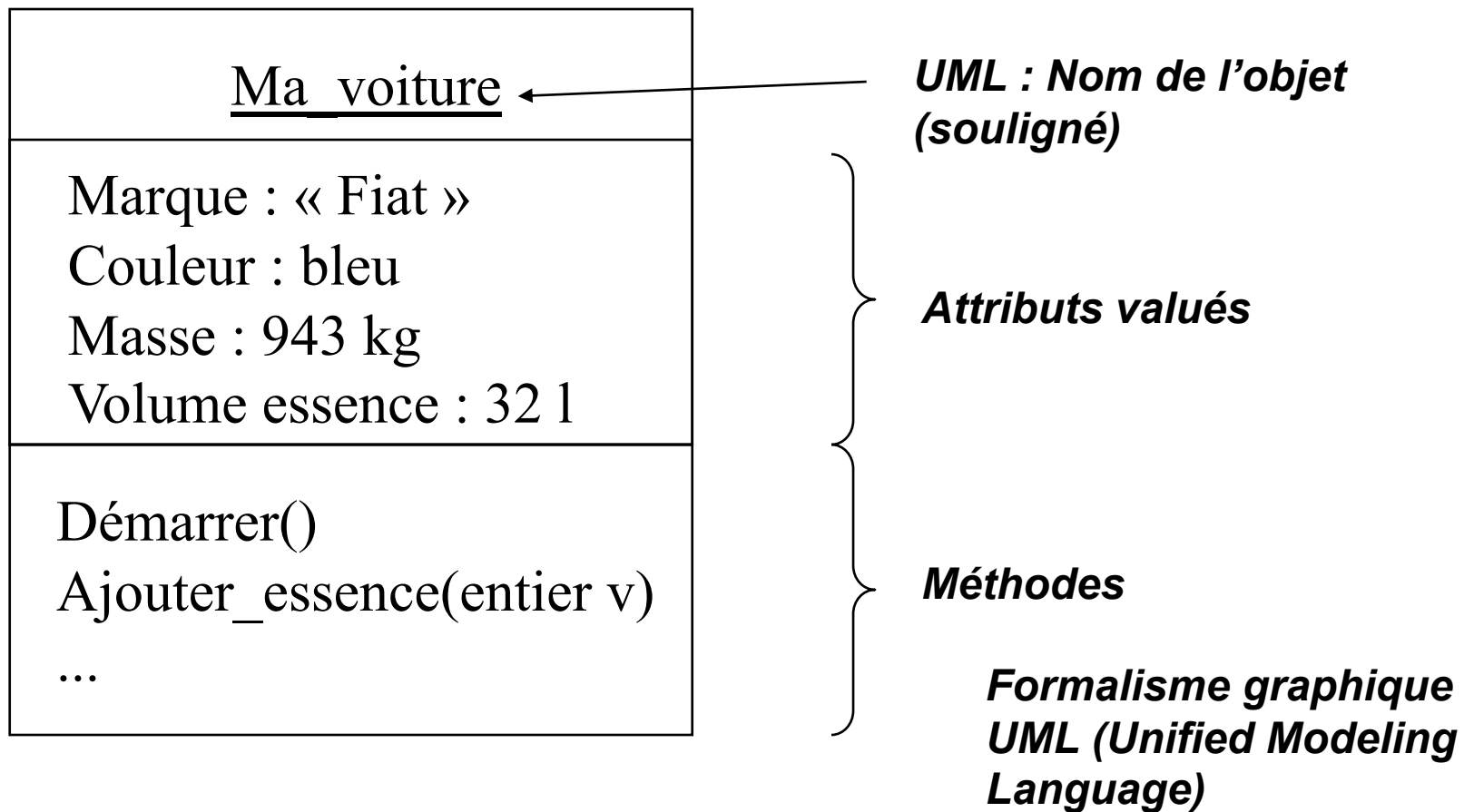




# Comportement d'un objet

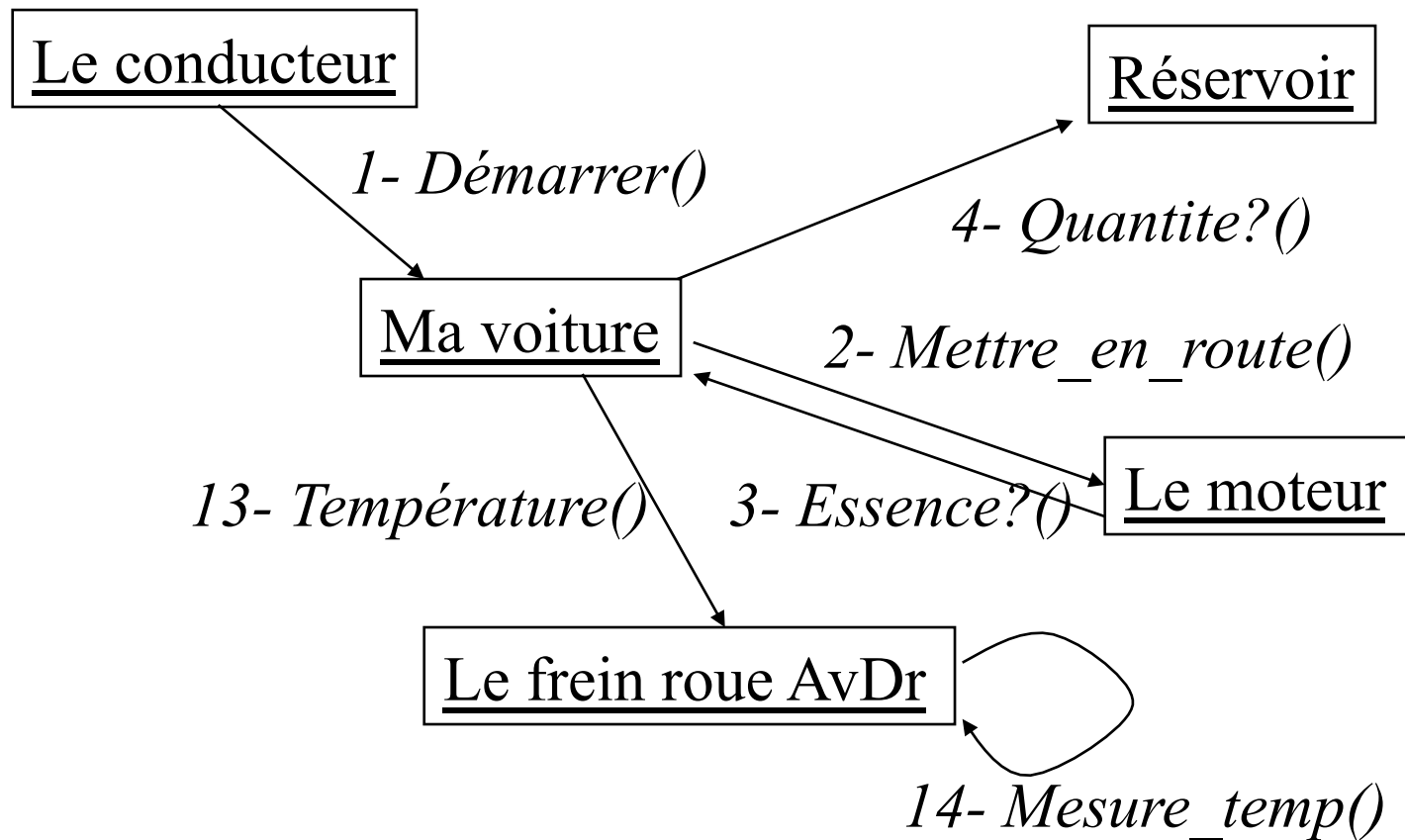
- Actions et réactions possibles
  - ensemble d'*opérations / méthodes*
  - exemple automobile
    - *démarrer, rouler, stopper, ajouter\_essence*
- Stimulation
  - demander à un objet d'effectuer une méthode = lui envoyer un message
  - Par exemple dans un programme
    - `ok = ma_voiture.démarrer()`
    - `vol = ma_voiture.ajouter_essence(15)`
- L'état dépend des opérations effectuées
  - Ex. `ma_voiture.volume_essence` diminue si `ma_voiture.rouler()` a été appelée
- Les opérations dépendent de l'état courant
  - Ex. `ma_voiture.démarrer()` ne marchera pas si `ma_voiture.volume_essence == 0`

# Représentation d'un objet



# Messages et collaboration d'objets

## Passage du flux de contrôle





# Accès aux attributs/méthodes

- Accès depuis un autre objet
  - Attribut/méthode public
    - tout objet peut y accéder
  - Attribut/méthode privé
    - aucun autre objet ne peut y accéder
    - seul l'objet lui-même peut utiliser ses attributs et méthodes
      - comme un programme « indépendant »
  - Attribut/méthode protégé
    - accès limité

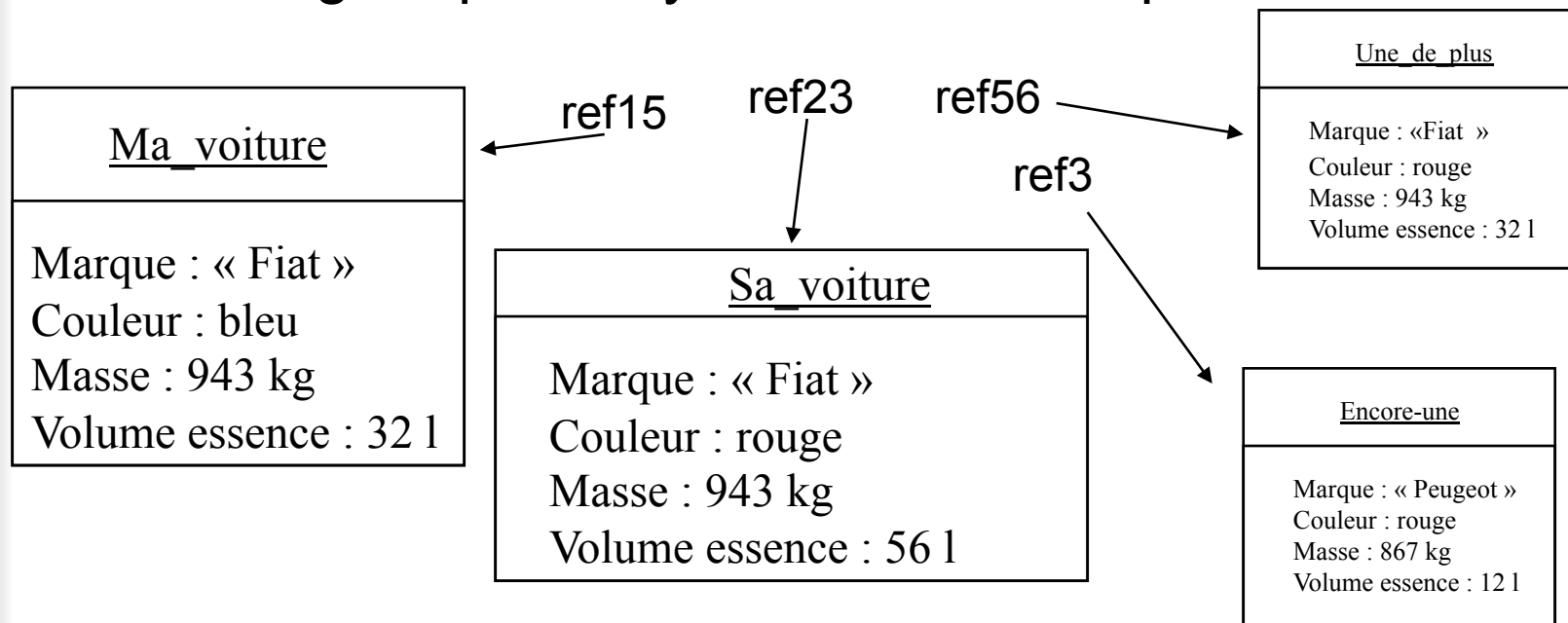


# Objet informatique

- **Etat**  
Ce qu'est l'objet à un instant donné
- **+ Comportement**  
Comment l'objet réagit aux sollicitations
- **+ Identité**  
Ce qui identifie l'objet

# Identité d'un objet

- Existence propre de l'objet
  - identification non ambiguë
  - indépendante de l'état
  - géré par le système informatique



# Liens entre objets

- Pour pouvoir envoyer un message à un objet, il faut le « connaître »
  - Ex. l'objet *Le\_conducteur* « connaît » l'objet *Ma\_voiture*
- Connaître un objet revient à avoir une référence qui lui corresponde
  - attributs
  - paramètres de méthodes

<u>Ma_voiture</u>	ref15
Marque : « Fiat » Couleur : bleu Masse : 943 kg Volume essence : 32 l Conducteur : ref67	

<u>Encore-une</u>	ref3
Marque : « Peugeot » Couleur : rouge Masse : 867 kg Volume essence : 12 l	

<u>Le_conducteur</u>	ref67
Sexe : M Couleur_yeux : bleu Age : 45 Voitures : (ref15, ref3)	



# En bref

- **Cohérence interne des objets**
  - données + traitements
- **Faible couplage entre l'objet et l'environnement**
  - envoi de messages uniquement entre objets qui se connaissent
- **Insertion dans un scénario de communication par envoi de messages**
  - objets clients : à l'origine d'une interaction
  - objets serveurs : répondent à la sollicitation
  - en général : objets clients et serveurs



# Que nous manque-t-il ?

- Soient 2 objets :
  - même structures de données (attributs)
  - même comportement (opérations)
- Il faut les décrire abstraitement de la même manière

<u>Ma R12</u>
Marque : Renault Couleur : bleu
Démarrer()

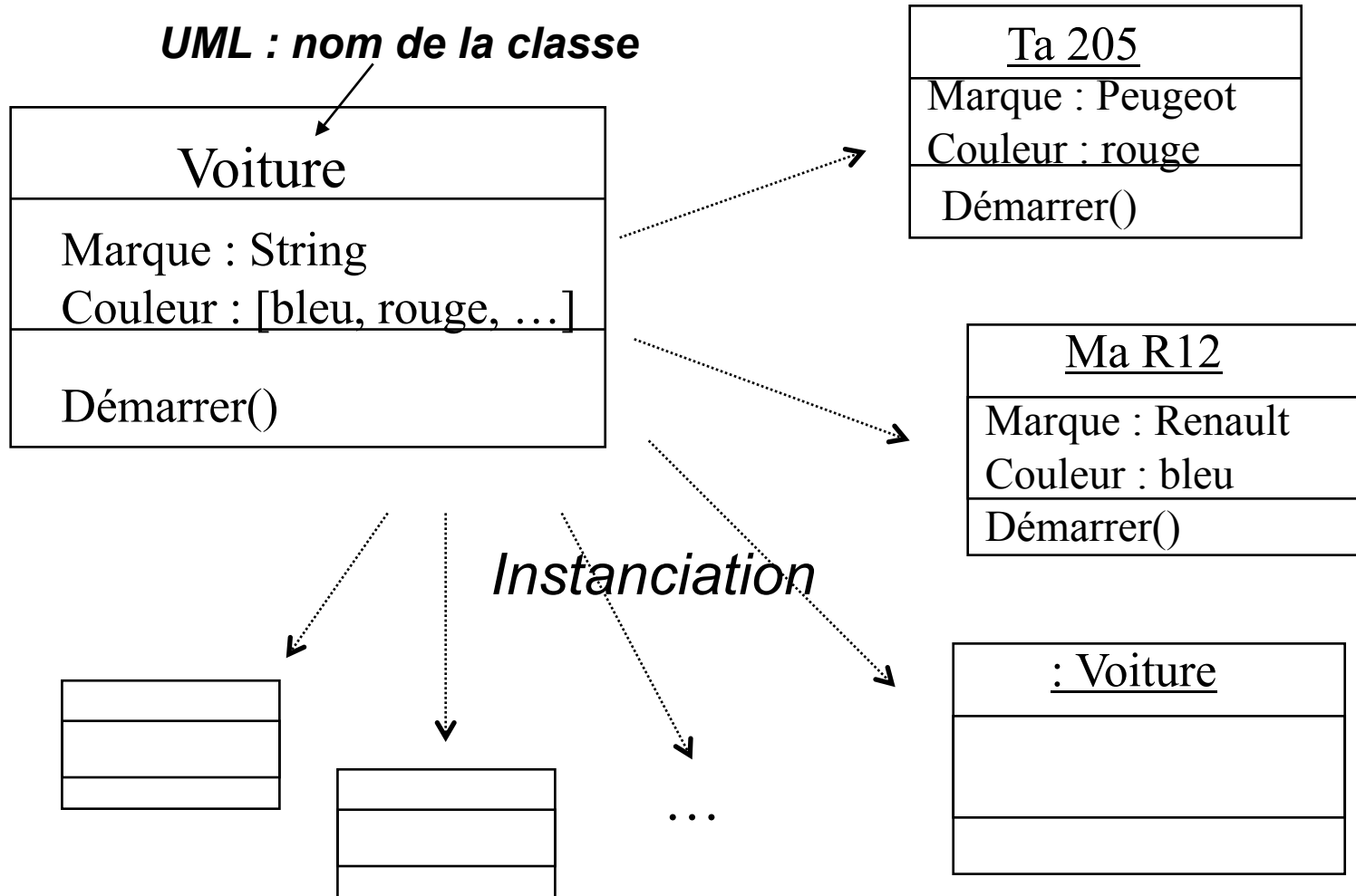
<u>Ta 205</u>
Marque : Peugeot Couleur : rouge
Démarrer()



# Notion de classe

- Les objets sont regroupés à l'aide grâce aux classes
- Une classe est une abstraction décrivant les propriétés communes des objets qui en sont des instances
- Une classe décrit une infinité d'instances
- Un objet sait toujours à quelle classe il appartient

# Classification





# Dans un programme OO

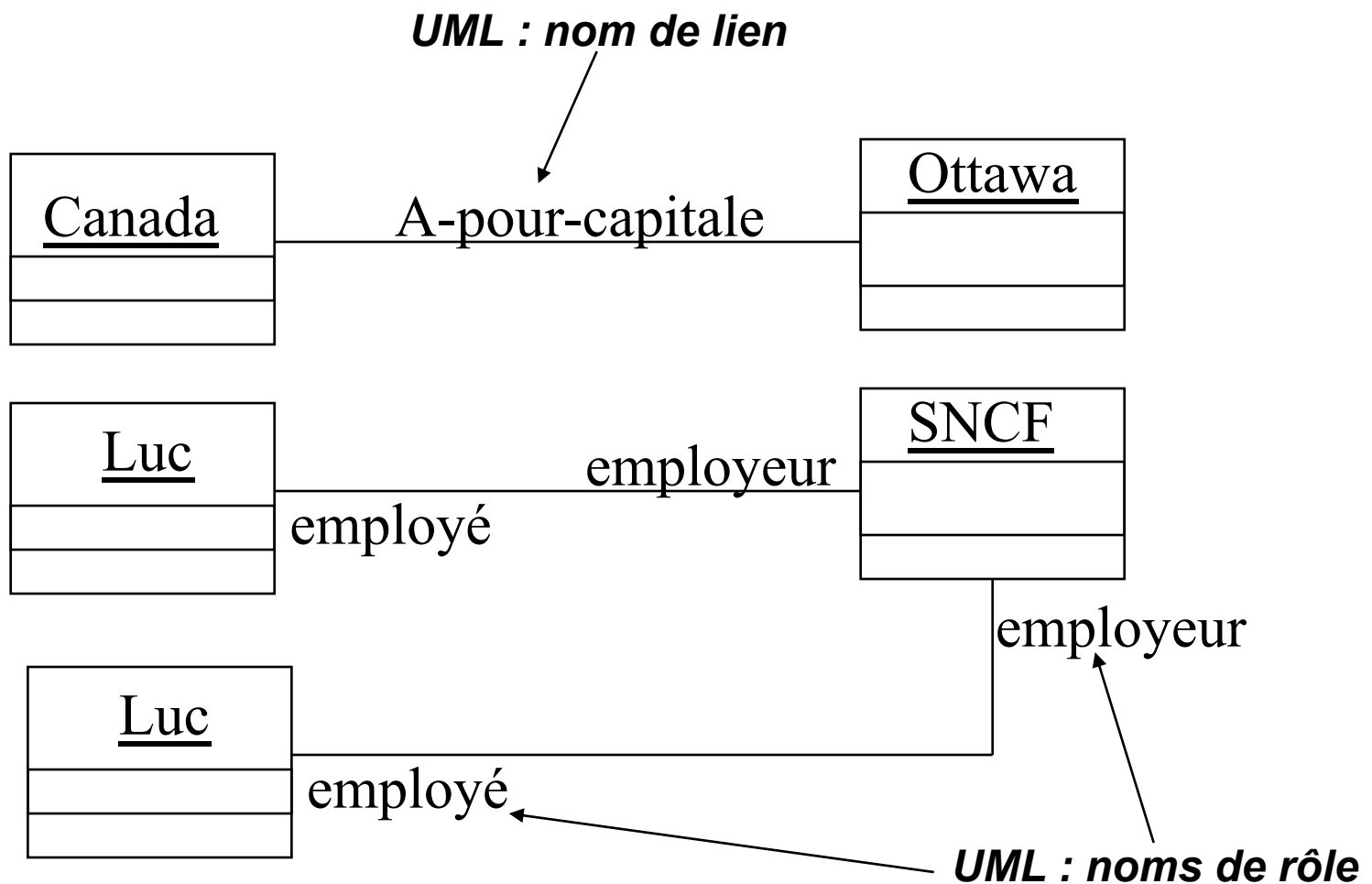
- On définit des classes
  - leur attributs, privés et publics
  - leurs méthodes, privées et publiques
- On instancie des objets à partir des classes
- On lance/gère la collaboration
  - envoi de messages à des objets
- Exécution du programme = des objets
  - qui s'envoient des messages
  - qui changent d'état



# Résumé

- **Objet = état + comportement + identité**
  - Attributs
  - Méthodes
  - (référence)
- **Classe**
  - Abstraction
  - Définit une infinité d'objets instances

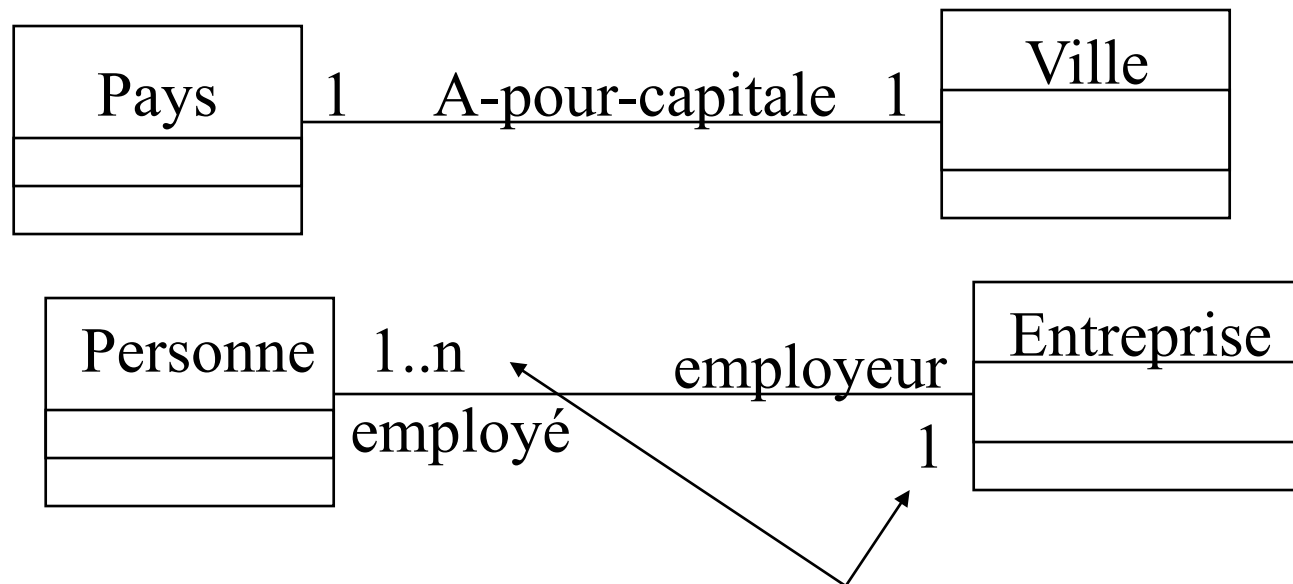
# Liens entre objets



# Associations entre classes

- Associations simples

- Liens entre objets → associations entre classes



**UML : cardinalités  
de l'association**

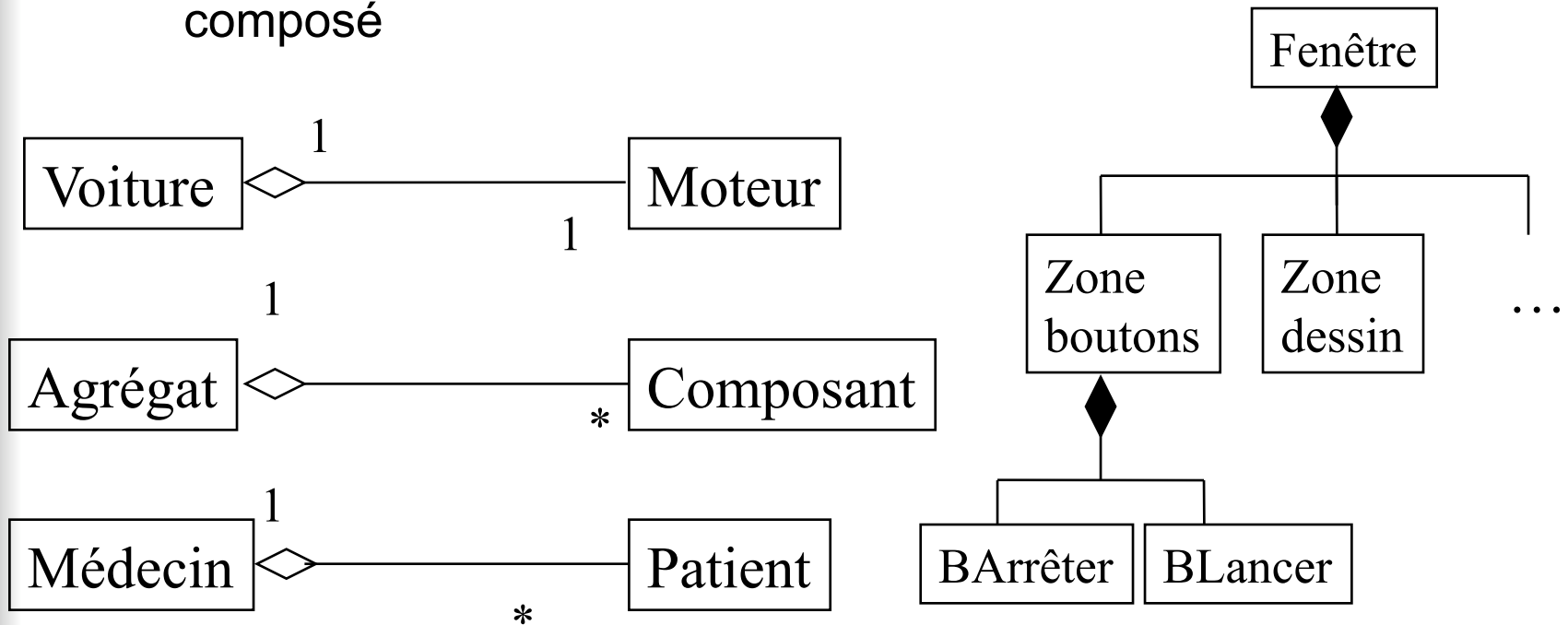
# Associations entre classes

## ■ Agrégation

- Une association particulière, dissymétrique, non nommée

## ■ Composition

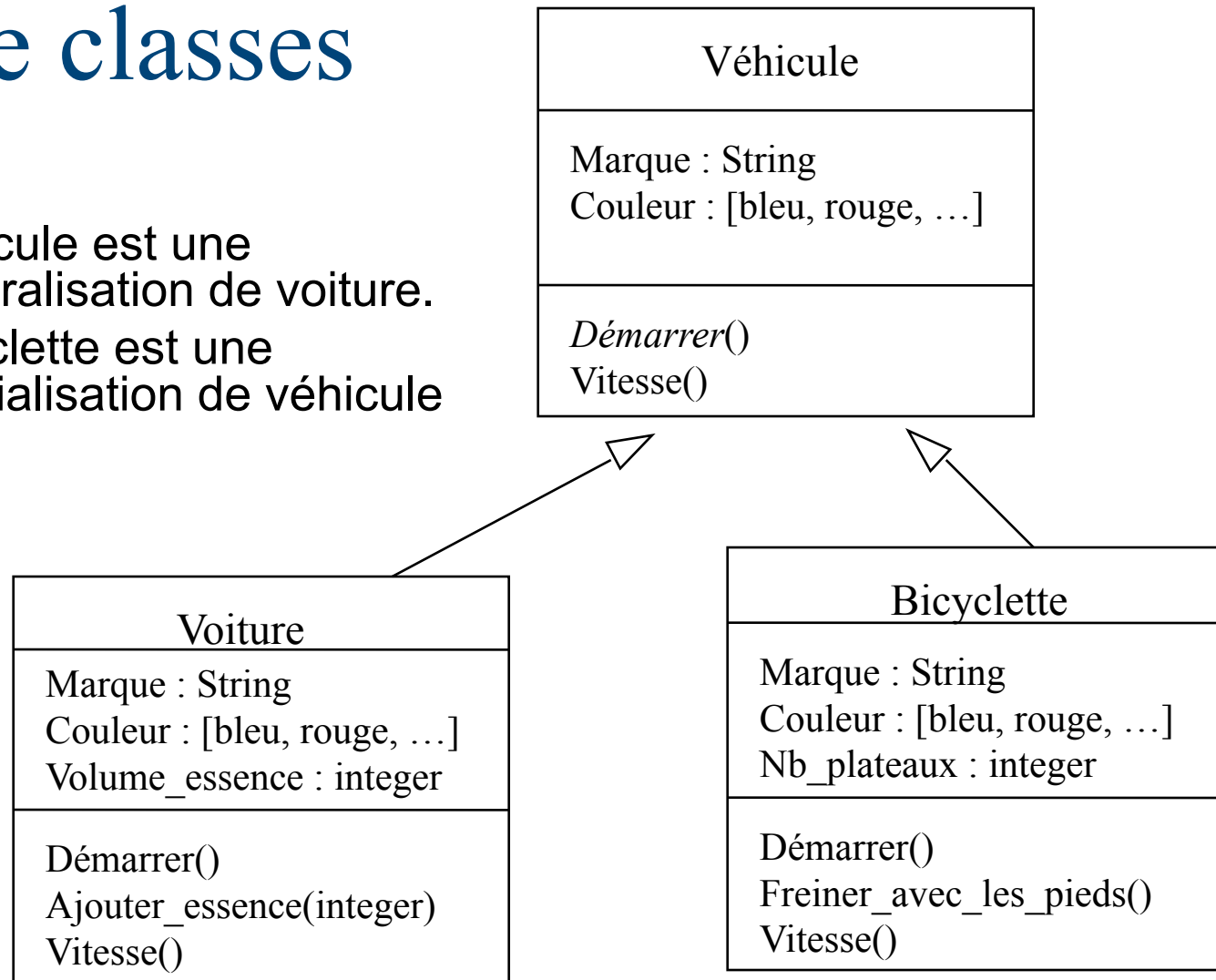
- Les composants n'ont aucune indépendance par rapport au composé





# Spécialisation / généralisation entre classes

- Véhicule est une généralisation de voiture.
- Bicyclette est une spécialisation de véhicule

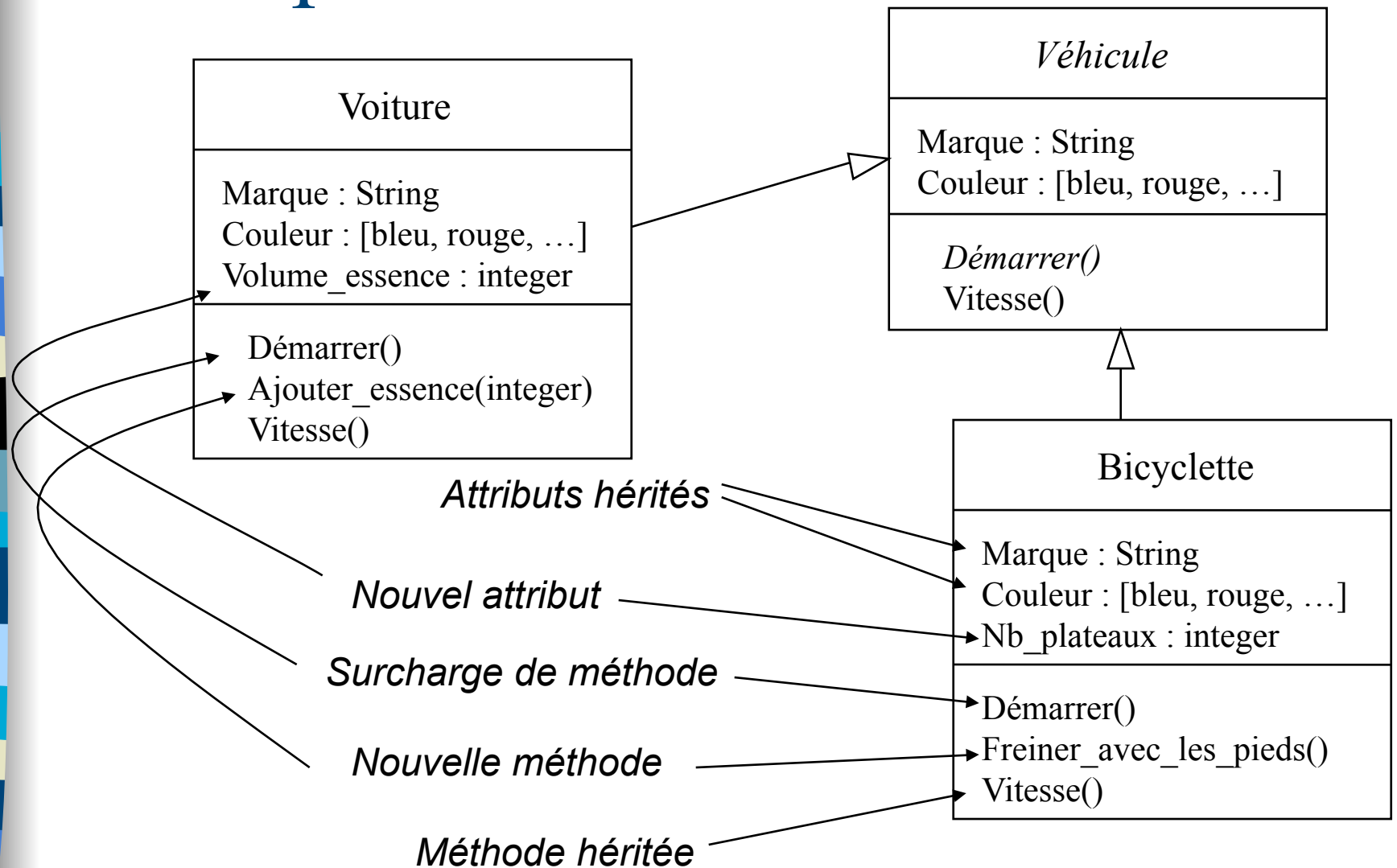




# Généralisation / spécialisation

- Mise en place d'une *hiérarchie* de classes
  - Voiture est une sous-classe de Véhicule
- Implémentation : partage d'attributs et héritage
  - Une sous-classe hérite des attributs et des méthodes de sa super-classe
  - Héritage multiple : plusieurs super-classes
    - à manipuler avec beaucoup de précautions
    - possibilité d'implémenter avec des interfaces
- Ajout d'éléments propres
  - Une sous-classe peut ajouter des attributs et méthodes à ceux qu'elle possède par héritage
- Surcharge
  - Une sous-classe peut redéfinir les attributs et méthodes de sa super-classe

# Exemple



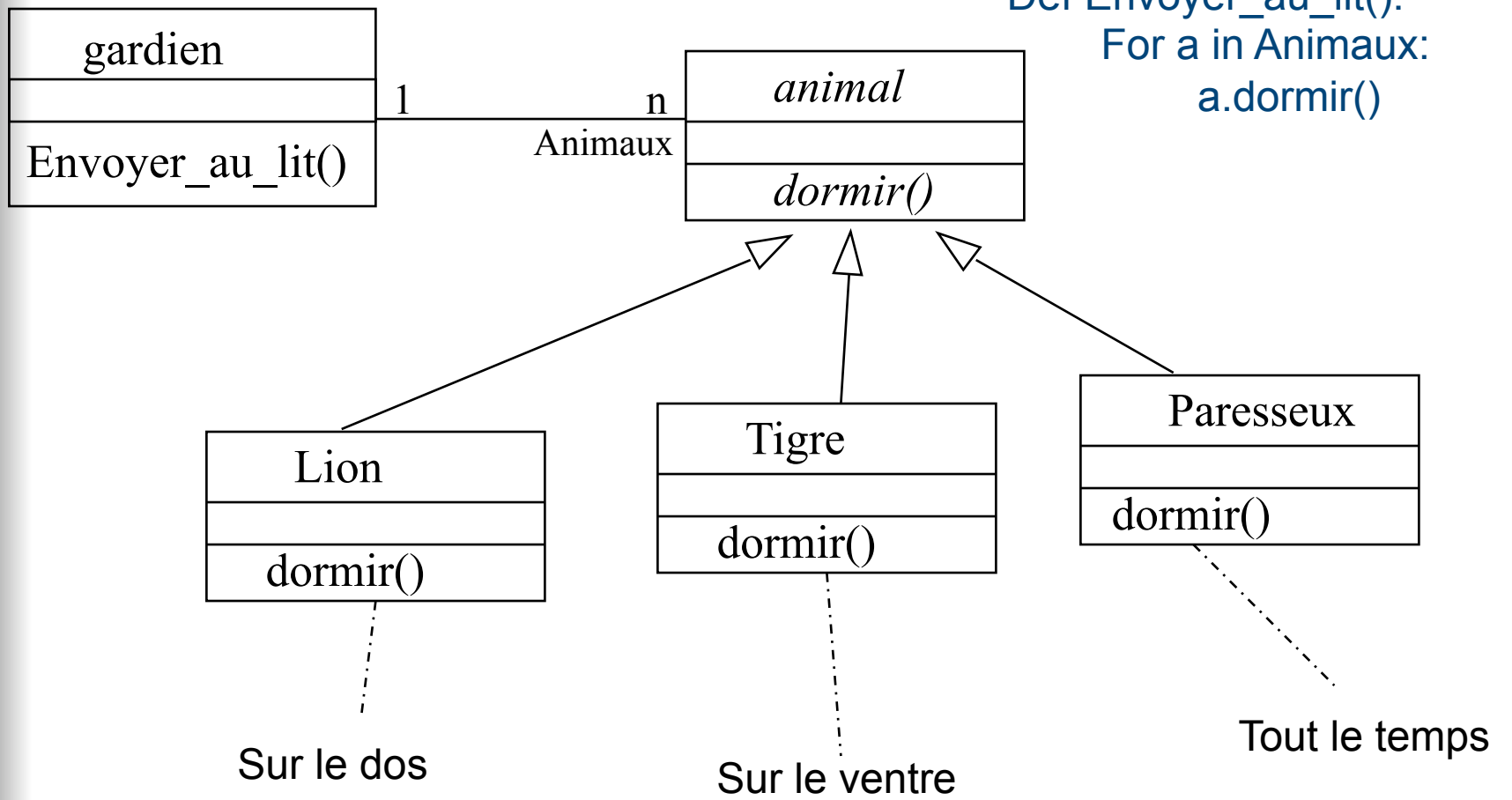
# Polymorphisme

- Une même opération peut se comporter différemment pour différentes classes / objets
  - Suivant l'objet, le langage sélectionne la méthode à utiliser pour la classe en cours
  - Il n'y a pas besoin de connaître toutes les méthodes existantes pour en implanter une nouvelle

<u>Ma R12</u>
Marque : Renault Couleur : bleu
Démarrer()

<u>Mon vélo</u>
Marque : Peugeot Couleur : bleu Nb_vitesse : 21
Démarrer()

# Exemple polymorphisme



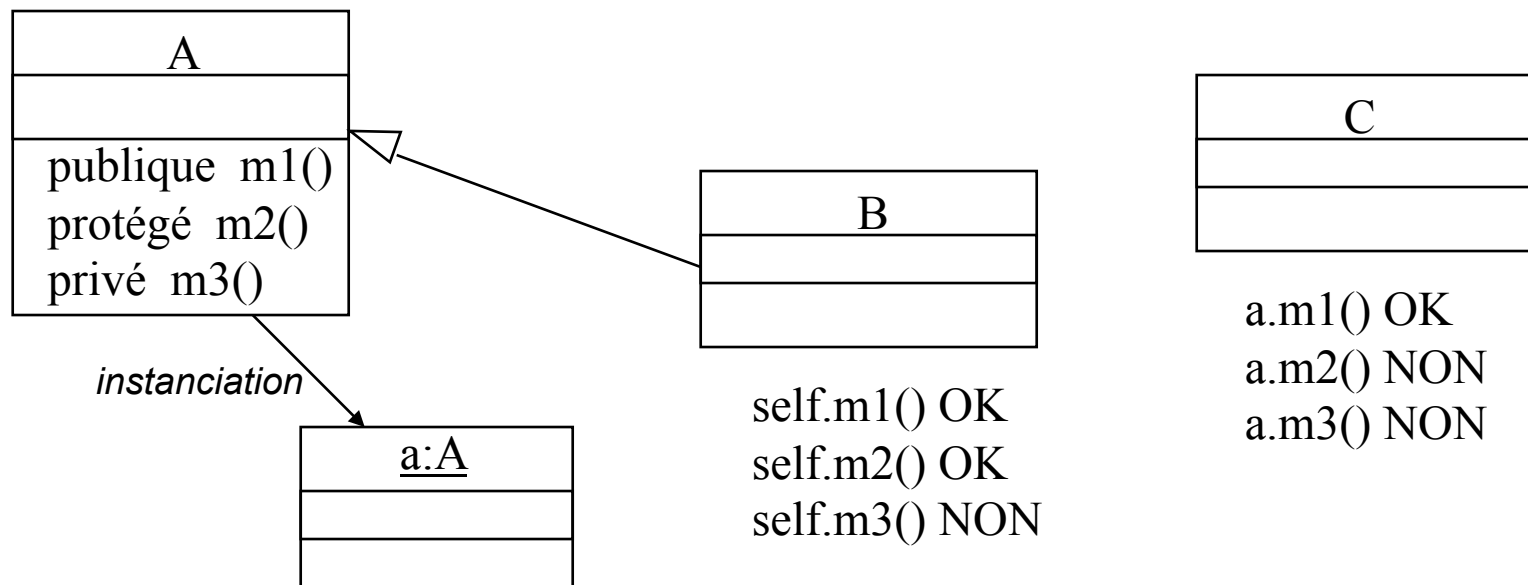


# Classe abstraite

- Classe qui n'est pas utilisée pour l'instanciation, et regroupe des propriétés et comportements
- Une classe dont certaines méthodes seront obligatoirement redéfinies dans les classes utilisées
  - Exemple : animal, vehicule
    - pas d'instances, mais des instances de sous-classes

# Contrôle d'accès des attributs et méthodes

- 3 types :
  - privé : limitation à la classe
  - public : accès pour toute classe
  - protégé : accès limité aux sous-classes





# Hiérarchie de classes

- Regroupement/organisation de l'ensemble des classes de l'application
  - hiérarchie de classe
  - + associations entre ces classes
- Provenance
  - certaines classes sont livrées avec le système
  - certaines proviennent de paquetages additionnels, récupérés ou achetés
  - certaines sont fabriquées par le programmeur
- Organisation en paquetages
  - ensemble de classes fortement liées (collaboration)
  - qui rendent des services (interfaces)





# Définition d'une classe

- Déclaration
  - éventuellement sous-classe d'une ou plusieurs autres classes
- Attributs
  - types simples
  - autres objets
- Méthodes
  - constructeur utilisé à l'instanciation
    - initialiser les attributs
    - réserver de la mémoire
    - appeler le constructeur de la super-classe si besoin
  - destructeur : utilisé à la destruction
    - libération de la mémoire
  - autres
    - sélecteurs : renvoient une partie de l'état de l'objet
    - modificateurs : modifient l'état
    - calcul



# Instances objets

- Création de l'objet
  - avec des paramètres ou non
  - appel du constructeur adapté
    - allocation mémoire
- Vie de l'objet
  - réception et traitement de messages
  - envoi de messages à d'autres objets
- Mort de l'objet
  - appel du destructeur



# Critères caractéristiques de l'OO

- Encapsulation données / traitements
- Identité
- Abstraction / classification
- Polymorphisme
- Généralisation / héritage



# TP noté

- Des tortues qui jouent au foot