Introduction à la programmation orientée-objet

Yannick Prié
UFR Informatique – Université Lyon 1

Master SIB M1 – 2005-2006
UE2.2 *Organisation de l'information documentaire et programmation*

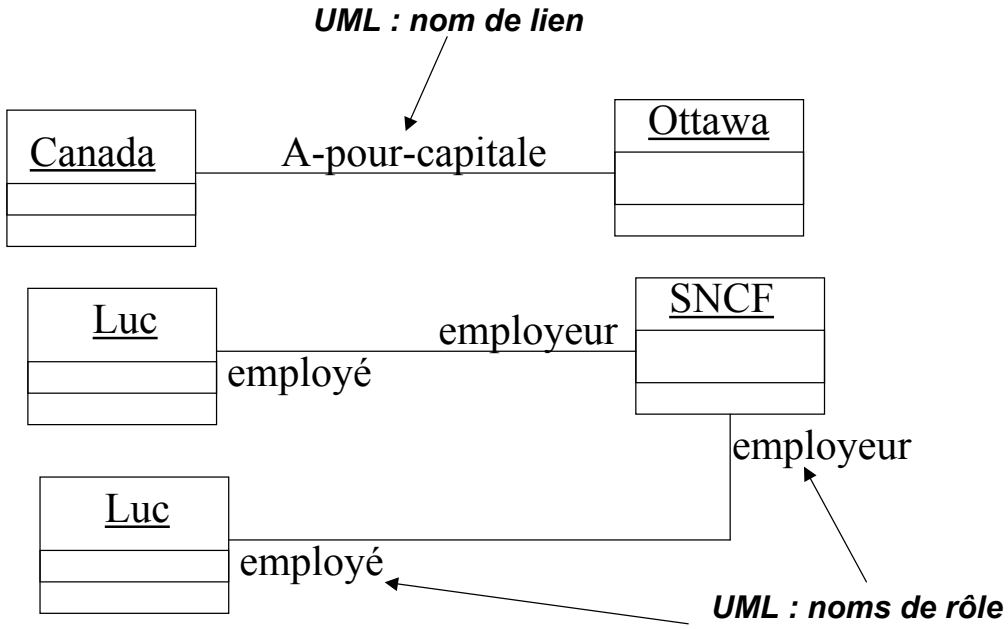
Résumé des épisodes précédents

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

Plan

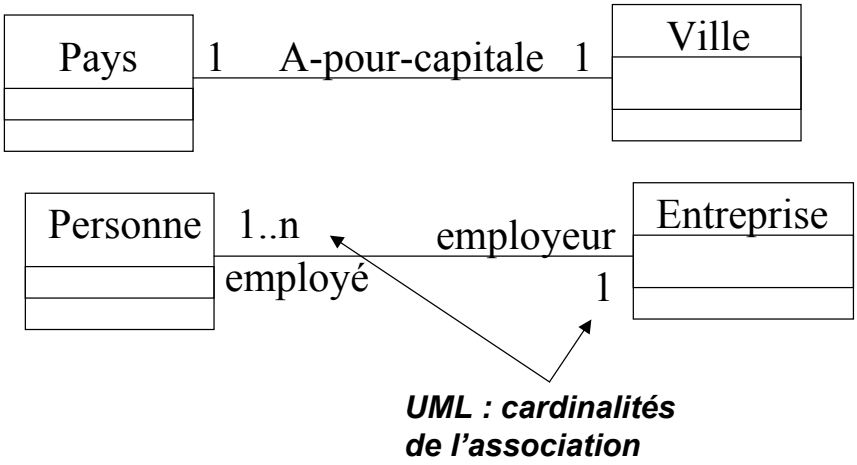
- Relations entre classes
- Hiérarchies de classes
- Classes et objets
- Initiation à la conception objet
- Eléments objets de Python

Liens entre objets



Associations entre classes

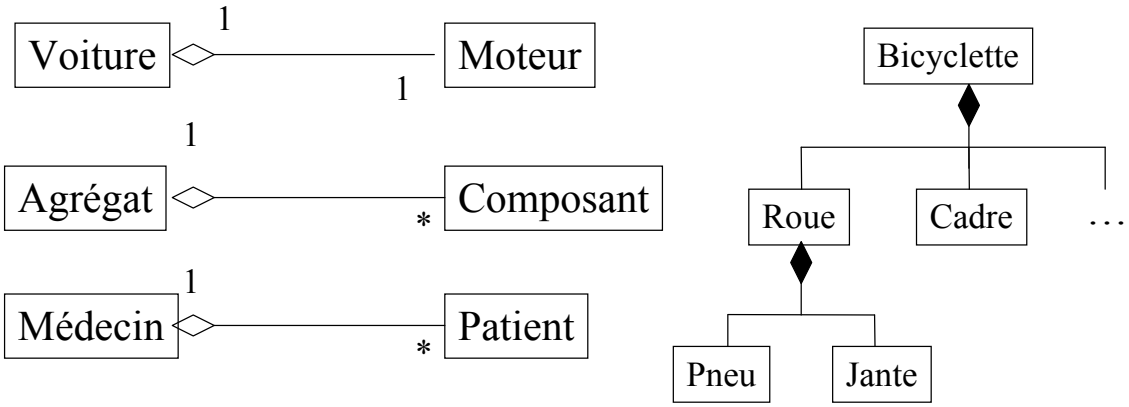
- Associations simples
  - Liens entre objets → associations entre classes



UML : cardinalités de l'association

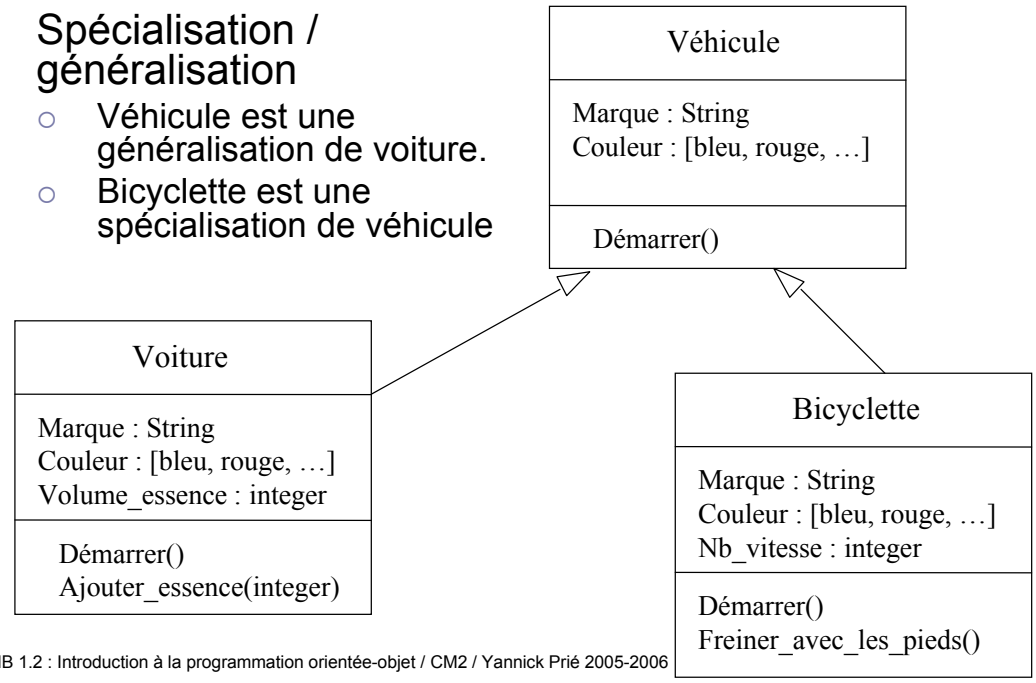
Associations entre classe

- Aggrégation
  - Une association particulière, dissymétrique
  - tout/partie - composant/composé - maître/esclave



Associations entre classes

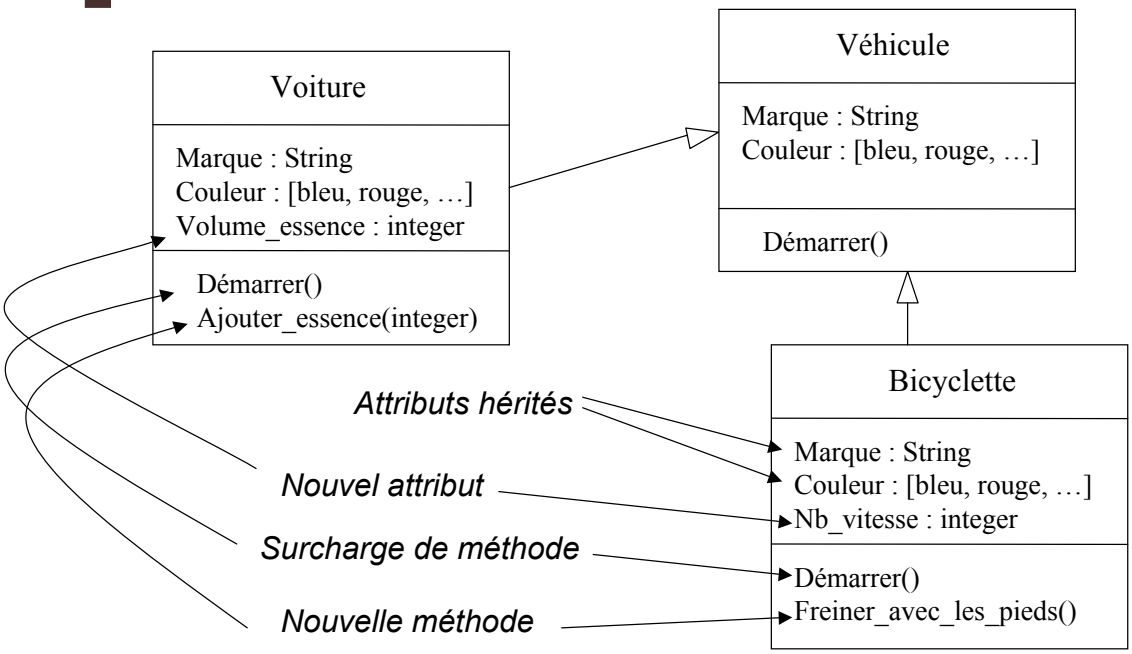
- Spécialisation / généralisation
  - Véhicule est une généralisation de voiture.
  - Bicyclette est une spécialisation de véhicule



Généralisation / spécialisation

- Mise en place d'une *hiérarchie de classes*
  - Voiture est une sous-classe de Véhicule
- 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
- 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 sur-classe

Exemple



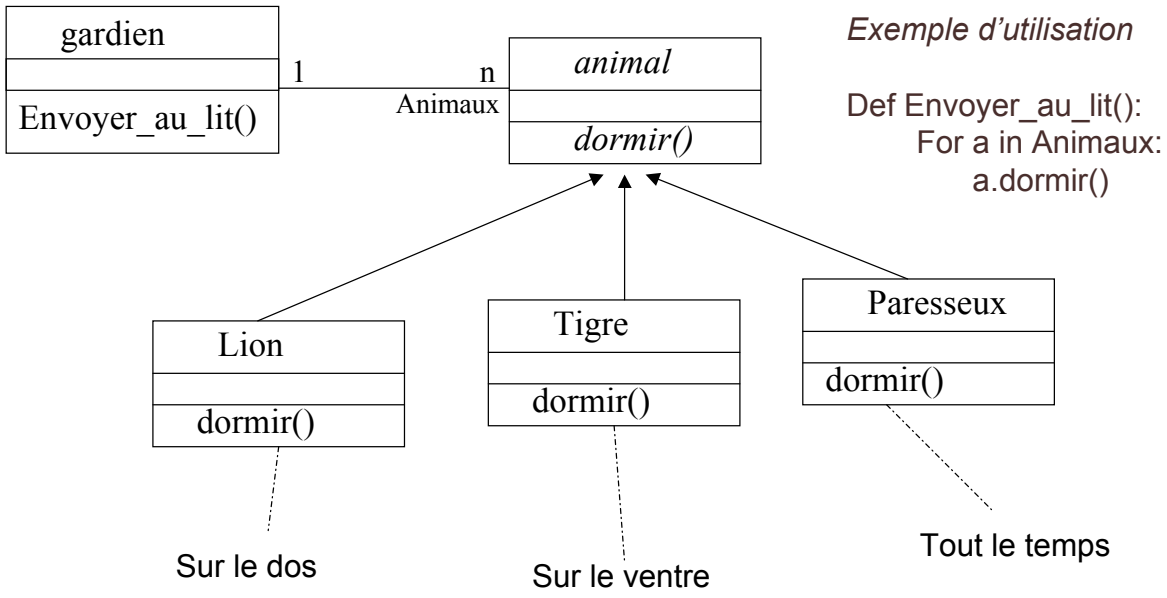
Polymorphisme

- 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

Ma R12
Marque : Renault Couleur : bleu
Démarrer()

Mon vélo
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
    - 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

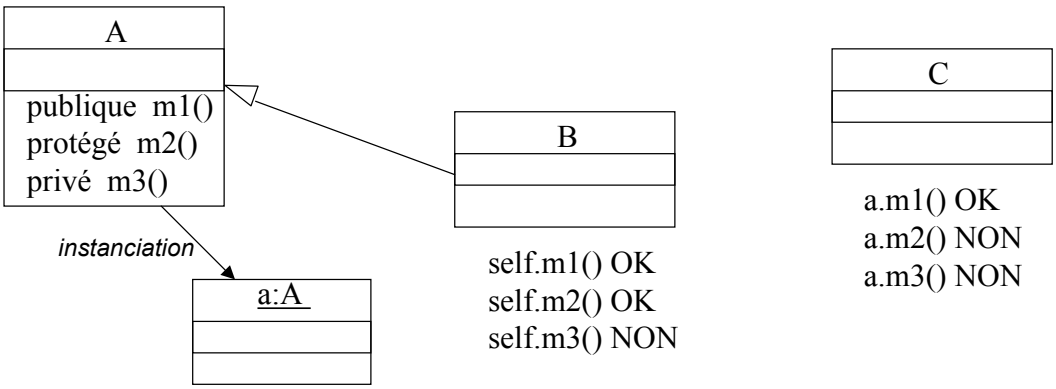


Diagramme de classes

- 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 additionels, récupérés ou achetés
  - Certaines sont fabriquées par le programmeur
- Organisation en paquetages
  - Ensemble de classes utiles

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
  - 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
    - ...

Exemple python

Constructeur

Attribut privé

```
class etudiant:
   "Classe representant un etudiant"

   def __init__(self,n,a):
      self.nom = n
      self.__age = a

   def get_age(self):
      return __age

   def affiche(self):
      print "Nom : %s - age %i " % ( self.nom ,
       self.__age )

   def __repr__(self):
     print "Nom : %s (%i ans)" % ( self.nom ,
       self.__age )
```

```
class cours:
   "Classe representant un cours"

   def __init__(self,m):
      self.matiere = m
      self.etudiants = []

   def ajout_etudiant(self,e):
      self.etudiants.append(e)

   def liste_etudiants(self):
      print self.matiere + " :"
      for e in self.etudiants:
         e.affiche()
```

Une sous-classe

```
class etudiant_sib(etudiant):
   "Classe representant un étudiant du Mastere SIB"

   def __init__(self,n,a,p):
      etudiant.__init__(self,n,a)
      if p :
         self.prog = 1

   def affiche(self):
      print self.nom
      if self.prog:
         print ("(prend des cours de programmation)")
```

Appel constructeur de etudiant

Redéfinition de méthode

Héritage d'attribut

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

Exemple python

```
>>> e1 = etudiant("Antoine",20)
>>> e2 = etudiant("Antoine",20)
>>> e3 = etudiant("Lydie",24)
>>> e4 = etudiant_sib("Jerome", 22, 1)
>>> c = cours("Systemes
    d'informations")
>>> c.ajout_etudiant(e1)
>>> c.ajout_etudiant(e2)
>>> c.ajout_etudiant(e3)
>>> c.liste_etudiants()
Systemes d'informations :
Nom : Antoine - age 20
Nom : Antoine - age 20
Nom : Lydie - age 24
>>> c.ajout_etudiant(e4)
>>> c.liste_etudiants()
Systemes d'informations :
Nom : Antoine - age 20
Nom : Antoine - age 20
Nom : Lydie - age 24
Jerome
(prend des cours de programmation)
```


Critères caractéristiques de l'OO

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

Langages orientés-objet

- Plus ou moins récents, objets (différents héritages), lisibles, efficaces, simples, expressifs, modulaires, résistants aux erreurs de codages, interfaçables avec d'autres langages, portables, interprétables/compilables, etc.
- Quelques exemples
  - Smalltalk
    - tout objet
  - C++
    - extension du C
  - JAVA
    - plus haut niveau, très utilisé
  - C# (Microsoft)
    - Clone de Java
  - Python
    - Simple, lisible, très haut niveau
  - ...

Initiation à la conception OO

- Quelques transparents
  - Pour donner une idée des grandes lignes de la conception orientée objet de SI
    - Expression des besoins
    - Analyse et conception objet
    - Développement incrémental
    - Gestion de projet
- Mots-clés
  - Réutilisation, abstraction,documentation
  - Attention à l'utilisateur

Objets du monde et objets informatiques

- Objectif :
  - objet informatique de première classe représente un objet du monde réel
    - stabilité au changement
  - langage commun entre
    - l'utilisateur du système
    - le concepteur
    - l'informaticien

→ fonder l'analyse du problème (et de la solution) sur des objets partagés

Des besoins au classes

- Cas d'utilisation
  - classes d'interactions entre système et utilisateur
    - Ex. se connecter au système
  - description par des scénarios = interactions particulières qui décrivent le maximum du fonctionnement du système
    - Ex.
      - sc1 = taper login, mdp, appui « entrée », connexion
      - sc2 = taper login, pas de mdp, appui « entrée », fenêtre erreur, appui « ok »
      - sc3 = taper login, mdp faux trois fois de suite, fenêtre erreur2
      - sc4 = taper login, annulation
      - ...
- Réalisation des scénarios avec des « collaborations d'objet »
  - un ensemble d'objets interagit pour mettre en œuvre le scénario
  - plusieurs scénarios → plusieurs types d'objets qui rendent des services dans divers contextes
- Déduction du diagramme de classes
  - Abstraction à partir des objets nécessaires

Gestion de projet

- Importance du développement incrémental
  - plusieurs cycles pour arriver au système complet
  - prototype qui fonctionne à chaque fin de cycle
    → satisfaction utilisateur / concepteur / développeur
- Attention portée au risque
  - évaluer en permanence ce qu'on sait faire, ce qu'on ne sait pas faire, et le danger associé à chaque risque
  - s'attaquer toujours au plus risqué

Python : le retour

- Déclaration de classes
- Méthodes et attributs privés
- Méthodes de classe
- ...