# UML

Pier Donini (Pier.Donini@heig-vd.ch)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

Hes·so
Haute Ecole Spécialisée de Suisse Occidentale

## Modélisation

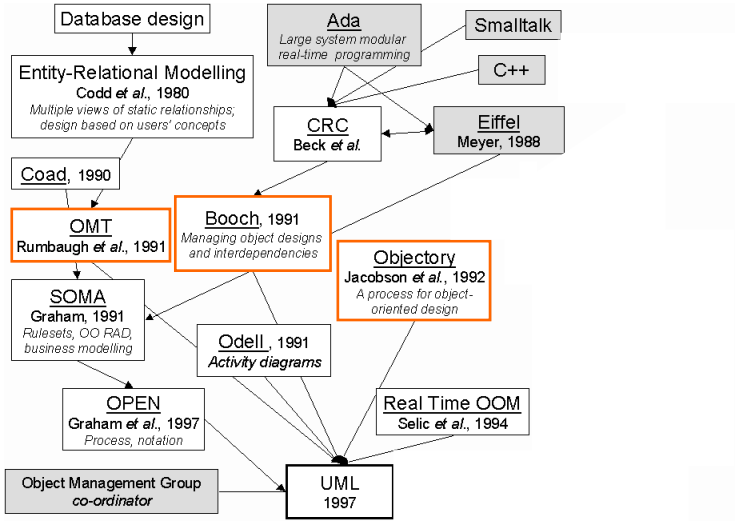
- La modélisation est un processus d'abstraction permettant de représenter un problème:
  - Identifier les entités du monde réel et leurs traitements.
- Une composante essentielle du développement d'applications:
  - Conception sans être contraint par des considérations d'implémentation,
  - Assure que l'application représentera les objets nécessaires et remplira les fonctionnalités désirées,
  - Tout en satisfaisant aux exigences de documentation, sécurité, extensibilité, réutilisabilité, maintenance…
  - ***Avant*** qu'une implémentation ne rende des modifications très difficiles.

# UML

- Unified Modeling Language (langage de modélisation unifié).
- Permet de spécifier et de visualiser les composants d'une application.
- Basé sur une sémantique précise et une notation graphique expressive.
- Repose sur plusieurs types de diagrammes (classe, cas d'utilisation, etc).
- Est devenu ***la*** référence pour la modélisation objet.
- Né de la fusion de plusieurs méthodes (par Booch, Jacobson et Rumbaugh).
- Défini par le consortium OMG (*Object Management Group*, http://www.omg.org).
- UML n'est pas associé à une méthodologie particulière (pas de description du processus d'élaboration des modèles). Cf., génie logiciel…



# Influences



# Références UML

- Manuels
  - Le guide de l'utilisateur UML, *Booch, Rumbaugh, Jacobson*
  - Modélisation objet avec UML, *Muller, Gaertner*, Eyrolles
  - UML distilled, *Fowler, Scott*
- Outils
  - Payants
    - Together http://www.borland.com/en-GB/Products/Requirements-Management/Together
    - Rational Rose, http://www-03.ibm.com/software/products/en/rosemod
    - Visual paradigm, www.visual-paradigm.com
    - Poseidon, www.gentleware.com/se.html



# Références UML (2)

- Outils UML (suite)
  - Gratuits
    - Together CE (plus distribué)
    - BOUML, www.bouml.fr
    - Poséidon CE, www.gentleware.com/ce.html
    - JDeveloppper (orienté Java), www.oracle.com/technetwork/developer-tools/jdev/overview
    - Plugins pour Eclipse et Netbeans
  - Open source
    - ArgoUML, argouml.tigris.org
    - Slyum (heig-vd), https://github.com/HEIG-GAPS/slyum/tree/master
    - StarUML, staruml.io

# Diagrammes UML

- Diagrammes de cas d'utilisation (use case)
  - Partition des fonctionnalités du système selon les besoins *acteurs* (*quoi*, pas *comment*). Scénarii non techniques…
- Diagrammes de **classes**
  - Cœur du langage.
  - Description statique des classes et des liens entre elles.
- Diagrammes d'objets
  - Description d'instanciations particulières du diagramme de classes.
- Diagrammes de séquences
  - Description dynamique des opérations (quels objets, quels messages).
  - Organisés de manière temporelle.
- Diagrammes de collaborations, d'états, d'activités, de composants, de déploiement…

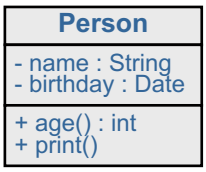


# Diagramme de classes

- Modélisation statique de la réalité.
- Classes, représentant d'un ensemble d'entités du monde réel perçues dans un contexte donné.
- Propriétés des classes:
  - Attributs,
  - Opérations.
- Liens entre les classes:
  - Héritage,
  - Associations.

# Classes

- Une classe est décrite par:
  - Son nom,
  - Ses attributs et leur visibilité,
  - Ses méthodes et leur visibilité.
  - Les propriétés de classe (attributs et méthodes non liées à un objet, mais à la classe elle-même) sont soulignées.
  - Attributs et méthodes ne sont pas nécessairement représentés.



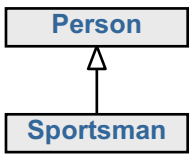


- Visibilité:
  - + publique
  - - privée
  - # protégée
  - ~ (ou rien) paquetage



# Classes (2)

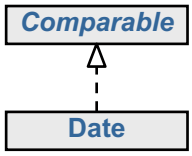
- Héritage: dénoté par une flèche en trait plein.




- Classe abstraite:
  - Représentation d'entités abstraites (n'existant pas en tant que telles): non instantiable directement, elle nécessite la définition de sous-classes.
  - Exemple: une classe abstraite Animal et ses sous-classes Chat, Chien…
  - Permet de définir des propriétés communes à une hiérarchie.
  - Les classes abstraites sont représentées en *italique.*
- Méthode abstraite:
  - Seule sa signature est définie, pas son implémentation.
  - Une classe déclarant une méthode abstraite est forcément abstraite et l'implémentation de la méthode doit être fournie dans les sous-classes.

# Classes (3)

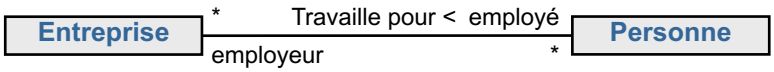
- Interface:
  - Ensemble de méthodes abstraites que peuvent implanter des classes (pour garantir un comportement).
  - Une interface est représentée en *Italique*.
- L'implémentation d'une interface par une classe est représentée par une flèche en pointillés.





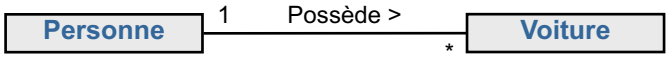

# Associations

- Une association représente une relation qui existe entre plusieurs objets où chaque objet joue un *rôle* déterminé.
- Représentée par un lien reliant deux classes.
- Une association est caractérisée par:
  - Un nom (généralement un verbe conjugué) éventuellement suivi d'une flèche précisant le sens de lecture.
  - Un rôle pour chacune des classes participantes. Le rôle d'une classe précise comment cette classe est *vue* par l'autre classe.
  - Le nom ou les rôles d'une association doivent être spécifiés.
  - Des cardinalités (ou multiplicité) indiquant le nombre d'objets liés par une association.



# Cardinalités

- Les cardinalités précisent combien d'objets de chaque classe peuvent être liés à un objet de l'autre classe par l'association.
  - « Une personne peut posséder plusieurs voitures, mais une voiture n'a qu'un seul propriétaire »



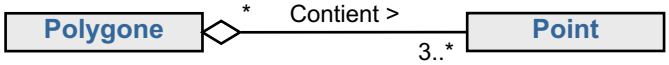

  - Populations:


- Notation:
  - 1 — 1 et un seul
  - 0..1 — zéro ou 1
  - 0..* ou * — de zéro à plusieurs
  - 1..* — au moins 1
  - n — exactement n (n entier)
  - n..m — de n à m (entiers), m > n
  - n..* — n (entier) ou plus

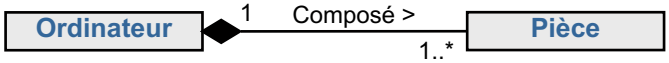


# Agrégations

- Agrégation: association spécialisée indiquant qu'un *tout* « *est composé de* » *parties*.
  - « Un polygone est composé de plusieurs points et un point peut appartenir à plus d'un polygone. »

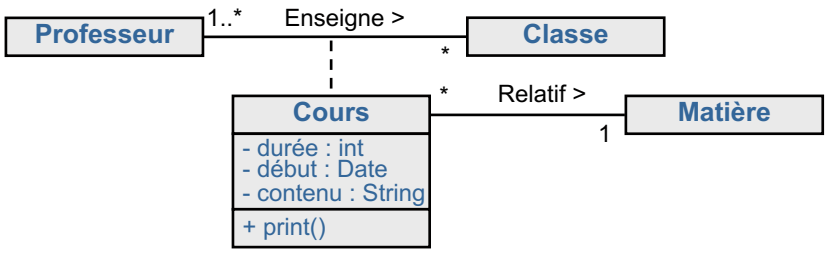



- Composition: agrégation spécialisée décrivant une contenance structurelle.
  - La destruction d'un objet composite (le *tout*) implique la destruction de ses objets composants (les *parties*).
  - Un objet composant ne peut faire partie que d'un seul objet composite. La cardinalité du côté du composite est au maximum 1 (1 ou 0..1).
  - « Un ordinateur est composé de ses pièces. »



# Classe d'association

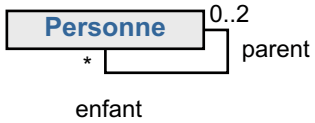
- Classe pour représenter les propriétés propres à une association.
  - Reliée à l'association par une ligne en traitillé.
  - Cette classe n'est pas nécessairement nommée (elle est identifiable par l'association concernée).
  - Elle peut être liée à d'autres classes.
- P. ex: à chaque couple d'objets (professeur, classe) d'une occurrence de l'association Enseigne correspond un unique objet cours (et réciproquement).



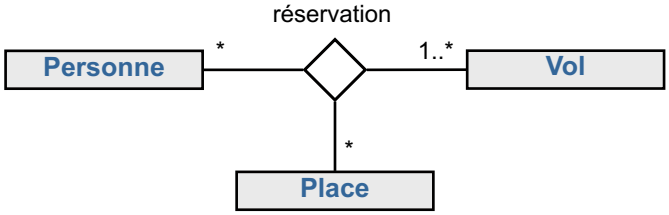



# Arité des associations

- Généralement lien binaire entre classes différentes (mais il peut exister plus d'une association entre deux classes données),
- Il peut exister des associations cycliques,

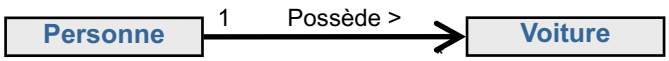



- Ou n-aires.
  - Non triviales à définir (dans le doute préférer des associations binaires).
  - « Une place est réservable dans un vol pour une personne. »



# Navigabilité

- Par défaut les associations peuvent être parcourues dans tous les sens.
- Afin de réduire le couplage entre classes, une propriété de navigabilité, dénotée par une flèche, peut être attachée à une des extrémités d'une association.

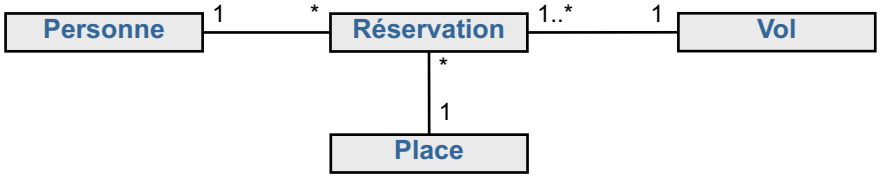



- Elle indique dans quel sens il est possible de traverser l'association lors de l'implémentation du diagramme de classe.
- Les objets de la classe *cible* participant à l'association doivent pouvoir être atteints directement depuis la classe *source*, mais non l'inverse.
- Ici, un objet Personne référence directement l'ensemble de ses objets Voiture qui, eux, ne référencent pas directement leur propriétaire.



# Implémentation des associations

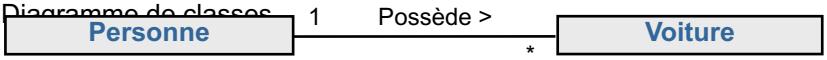
- Par des classes spécifiques:
  - pour implémenter des associations n-aires
    (attention à l'inversion des cardinalités par rapport au schéma original)




  - pour implémenter des classes d'associations
    (attention au déplacement des cardinalités sur la classe d'association)

# Implémentation des associations (2)

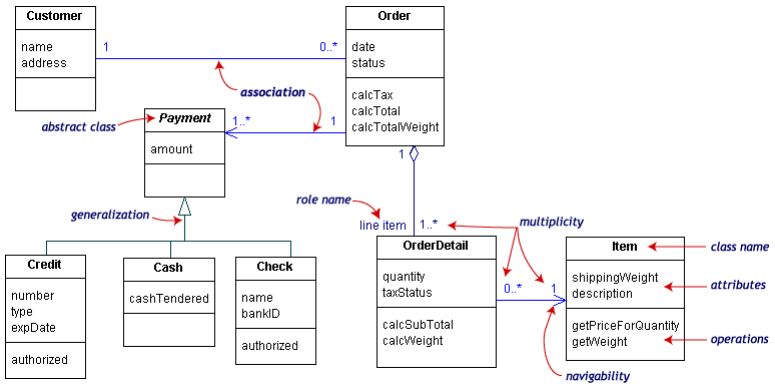
- Dans les classes liées, selon les cardinalités et la navigabilité, définir:
  - Des attributs références (pour les cardinalité 0..1 ou 1) ou
  - Des collections (tableau, liste…) de références (pour les cardinalités > 1).
- Ne jamais indiquer dans un diagramme de classes les références ou les collections de références utilisées pour implémenter les associations.
- Une implémentation de l'association Possède (navigable dans les deux sens):
  - Diagramme de classes




  - Implémentation

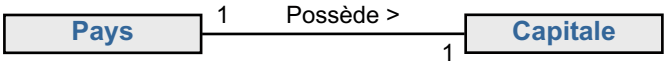



# Exemples (1)

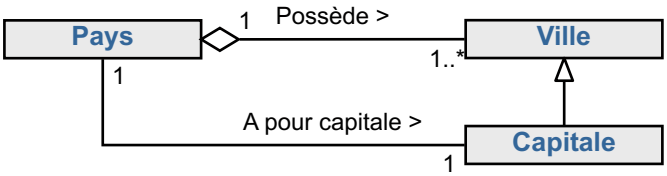


# Exemples (2)

- « Un pays possède une capitale. »




- « Un pays possède des villes et une capitale. »




  - Contrainte d'intégrité (à implanter): *la capitale est bien une ville du pays*.
  - Sans l'association « à pour capitale », CI: *une seule capitale par pays*.



# Exemples (3)

- « Un joueur de rugby est un avant, un demi ou un arrière. »
- « Une équipe de rugby est composée de 8 avants, 2 demis et 5 arrières. »