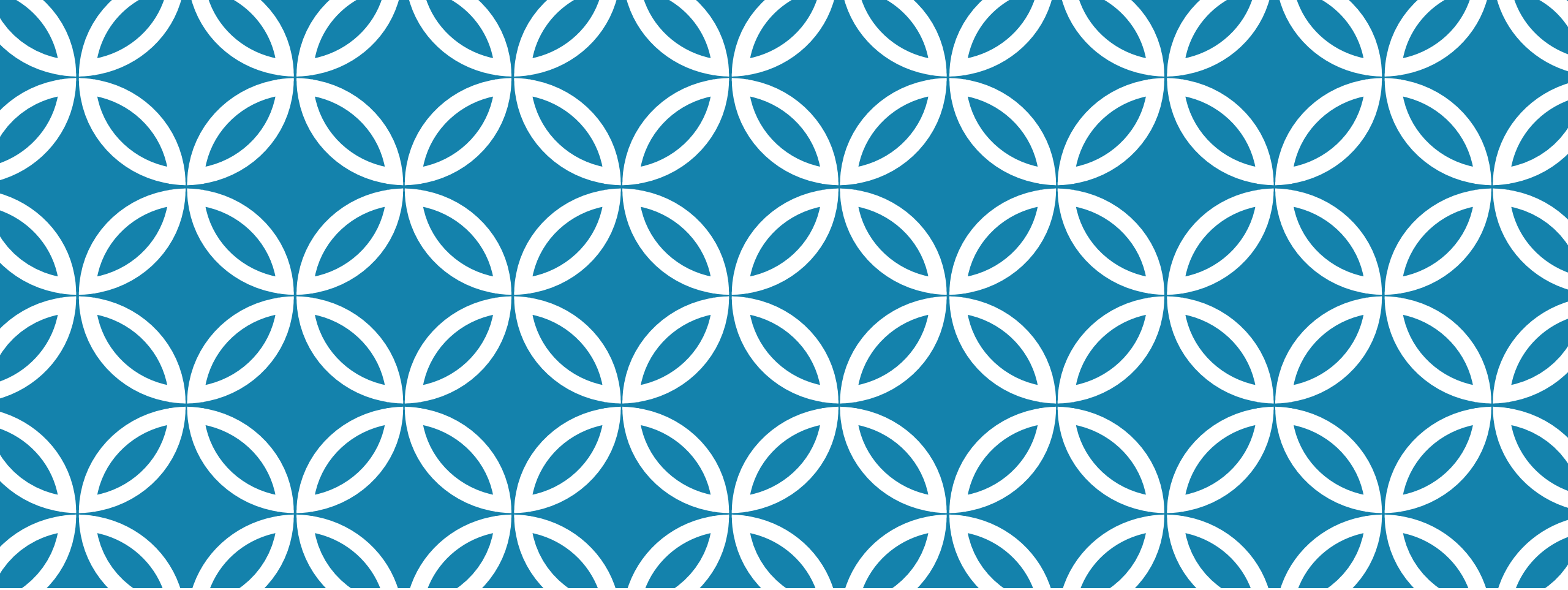


CONCEPTION ORIENTÉE OBJET

Présentation

Amosse Edouard

Inspiré du cours de F. Mallet

ORGANISATION DU COURS

1. Volume Horaire et EDT http://unice.fr/faculte-des-sciences/departements/informatique/contenus-riches/documents-telechargeables/documentsl3i/calendrier-des-cours-tp-et-td-de-projet-informatique-et-de-coo

2. Evaluation
   1. Examen théorique : 1h30 : 50%
   2. Travaux Pratiques (en TD) : 50%

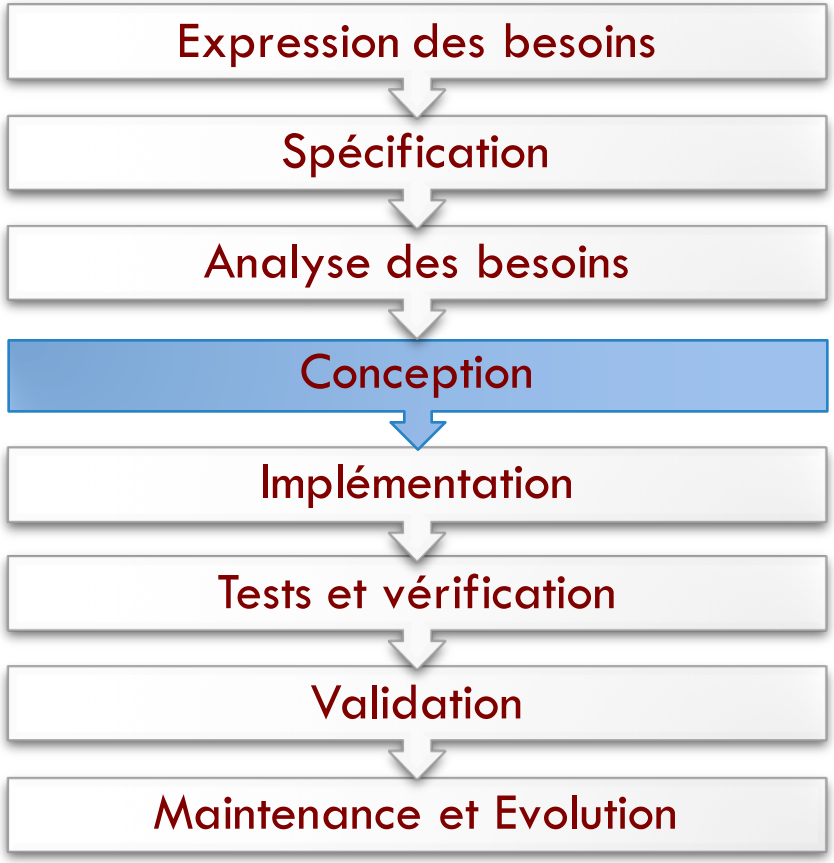
PLAN DU COURS

- Introduction aux méthodologies de conception
- Introduction à UML
- Objets et Classes
- Les diagrammes
  - Les cas d'utilisation
  - Les classes et leurs instances
  - Les machines à états (et transitions)
  - Les activités
  - Les interactions
  - Le mécanisme de profilage

OBJECTIFS

- Montrer les forces de la COO
- Décrire l'histoire de la POO
- Commenter l'utilisation actuelle de la POO

CYCLE DE VIE D'UN LOGICIEL



CYCLE DE VIE D'UN LOGICIEL

- Expression des besoins
  - Définition d'un cahier des charges
- Spécification
  - Ce que le système doit être et comment il peut être utilisé
- Analyse
  - Éléments intervenant dans le SI, leurs structures et relations
  - A définir sur 3 axes
    - Savoir-faire de l'objet → *axe fonctionnel*
    - Structure de l'objet → *axe statique*
    - Cycle de vie de l'objet → *axe dynamique*
- Conception
  - Apport de solutions techniques: architecture, performance et optimisation
  - Définition des structures et des algorithmes

CYCLE DE VIE D'UN LOGICIEL

- Implémentation
  - Réalisation et programmation
- Tests et Vérification
  - Contrôles de qualité
- Vérification de la correspondance avec le cahier des charges

- Maintenance et Evolution
  - Maintenance corrective : traiter les erreurs (bugs)
  - Maintenance évolutive : intégration de nouveaux changements

CONCEPTION

- Processus créatif qui consiste à représenter les diverses fonctions du système permettant d'obtenir rapidement un ou plusieurs programmes réalisant ses fonctions.
- Une « bonne » conception se définit en termes de la satisfaction des besoins et des spécifications.
- Une bonne Conception participe largement à la production d'un logiciel qui répond aux **facteurs de qualité**.
- Elle se base sur la **Modularité**.

CONCEPTION – CRITÈRES DE QUALITÉ

- **Cohésion** : Se définit comme étant le caractère de ce qui forme un tout, dont les parties sont difficilement séparables.
- **Couplage** : Relatif à la cohésion. Il exprime le degré d'interconnexion des differents composants d'un système.
- **Compréhensibilité** : La compréhensibilité d'un module dépend de :
  - Sa cohésion
  - L'appelation : Utilisation de noms significatifs
  - La documentation : Lien entre le monde réel et le composant
  - La complexité
- **Adaptabilité** : Dépend du couplage et de la documentation. Un logiciel adaptable doit avoir un haut degré de lisibilité.

MÉTHODES DE CONCEPTION

On distingue principalement de trois de méthodes de conception:

- Méthodes fonctionnelles
- Méthodes systémiques
- Méthodes orientées objets

MÉTHODES FONCTIONNELLES

Les méthodes finctionnelles ou cartésiennes consistent à décomposer hiérarchiquement une application en un ensemble de sous applications.

Ces méthodes utilisent les raffinements successifs pour produire des spécifications don't l'essentiel est sous forme de notation graphique en diagramme de flots de données.

MÉTHODES FONCTIONNELLES

Points forts

- Simplicité du processus de conception
- Capacité à répondre rapidement aux besoins ponctuels des utilisateurs

Points faibles:

- Fixer les limites pour les décompositions hiérarchiques
- Rédondance (éventuelle) des données

MÉTHODES SYSTÉMIQUES

Les méthodes systéliques sont influencés par les systèmes de Gestion de bases de données en proposant une double démarche de modélisation:

- La modélisation des données
- La modélisation des traitements

- **Points forts :**
  - Approche globale prenant en compte la modélisation des données et des traitements
  - Niveaux d'abstraction dans le processus de conception
  - Bonne adaptation à la modélisation des données et à la conceptiond es BDs

- **Points faibles**
  - Double démarche de conception : données et traitements
  - Pas de fusion possible des deux aspects (données et traitements)

MÉTHODES FONCTIONNELLES ET SYSTÉMIQUES

Les méthodes foncionnelles et systémiques sont de type **descendant (approache Top-Down)**.

Inconvénients

Réutilisabilité : Modules non généraux mais adaptés aux sous problèmes pour lesquels ils ont été concus

Extensibilité: L'architecture du logiciel est fondée sur les traitements qui sont moins stables que les données; par conséquent cette approche est inadaptée à la conception de gros logiciel.

MÉTHODES ORIENTÉES OBJETS

- Dans une approche Orientée Objet (OO), le logiciel est considéré comme une collection d'objets dissociés définis par des propriétés.

- Une propriété est soit un attribut de l'objet ou une opération sur l'objet.

- Un objet comprend donc à la fois une structure de données et une collection d'opérations (son comportement).
- Contrairement aux méthodes fonctionnelles et systémiques, les méthodes orientées objets sont **ascendantes**.

MÉTHODES ORIENTÉES OBJETS

La technologie Orientée Objet

- Guide la conception par
  - Un ensemble de concepts
    - Abstraction, modularité, Encapsulation, Polymorphisme
  - Des langages et outils qui supportent ces concepts
    - Classification vs. prototype
    - Héritage (Simple, Multiple)
    - Typage (Fort, Faible)
- Avantages
  - Reflète plus finement les objets du monde réel
    - Du code : facile à maintenir
    - Plus stable: Isolation des changements
    - Réutilisation des composantes
    - Faciliter le prototypage

MÉTHODES ORIENTÉES OBJETS – EXEMPLES

Système de gestion d'un lycée

Objets

- Personnes
  - Etudiant, enseignant, principal, secrétaire
- Diplôme
  - Année, matière, parcours
- Notes
  - Coefficients

Fonctions

- Calculer la moyenne
- Calculer les taux d'encadrement
- Calculer le nombre de redoublants
- Calculer le taux de réussite au baccalauréat

OBJECTIFS DES TECHNOLOGIES À OBJETS

- Utiliser le langage du domaine
  - Modèle et vocabulaire métier
- Construire des modèles faciles à:
  - Etendre, modifier, valider, vérifier
- Faciliter l'implantation
  - Génération facilitée vers les langages à objets
- Nécessite une méthode et des outils
  - Rational Unified Process, Agile, … (cf. semestre 2)
  - UML est seulement un langage

LES OBJETS…

Définitions :

- Entité cohérente rassemblant des données et du code travaillant sur ces données
- Structure de données valuées qui répond à un ensemble de messages

Caractérisé par :

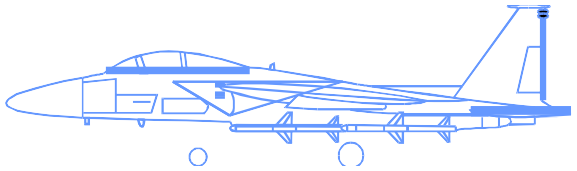
- son comportement : que peut-on faire avec cet objet?
  - Méthodes
- son état : comment réagit l'objet quand on applique ces méthodes?
  - Attributs (Champs)
- son identité : comment distinguer les objets qui ont le même état et le même comportement?
  - Identifiant

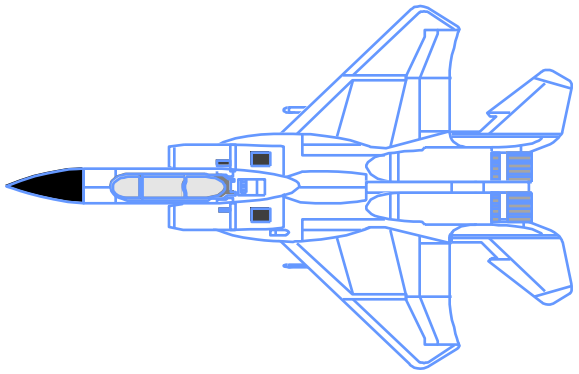
A les mêmes réactions et la même modularité que le monde réel

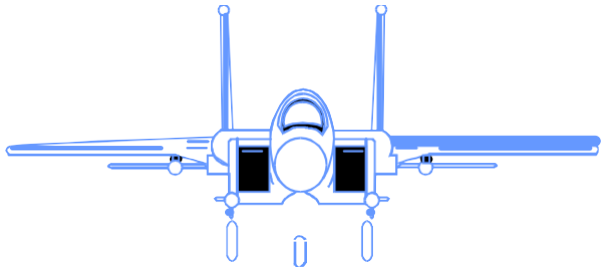
- L'objet informatique est une projection de l'objet du monde réel

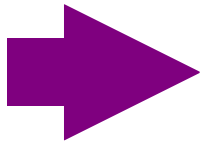
UN MODÈLE

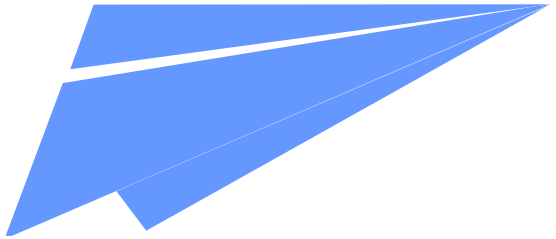
- Une simplification de la réalité











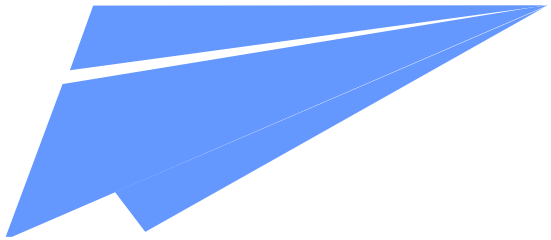
POURQUOI UN MODÈLE?

- Quatre objectifs à la modélisation
  - Aider à **visualiser** le système
  - **Spécifier** la structure et le comportement
  - **Servir de plan** pour la construction effective
  - Permettre de **documenter** les choix
- Quatre avantages
  - Abstraction : diviser pour régner
  - Compréhension : mises au point avec le client
  - L'énergie déployée pour modéliser révèle les difficultés
  - Les erreurs sur les modèles **coûtent bien moins cher**

IMPORTANCE DES MODÈLES

moins important

Plus important



Avion papier

Avion militaire

Le développement logiciel AUSSI nécessite des modèles bien pensés !

OBJETS – CLASSESS (EXEMPLES)

Classe

Figure
longueur largeur origine
périmètre **surface** **transposer**

Objet

rectangle: Figure
longueur: 24 largeur: 20 Origine : (12, 20)

```
Figure rectangle= new Figure( );
rectangle.surface();
```

MODEL DRIVEN ARCHITECTURE (MDA)

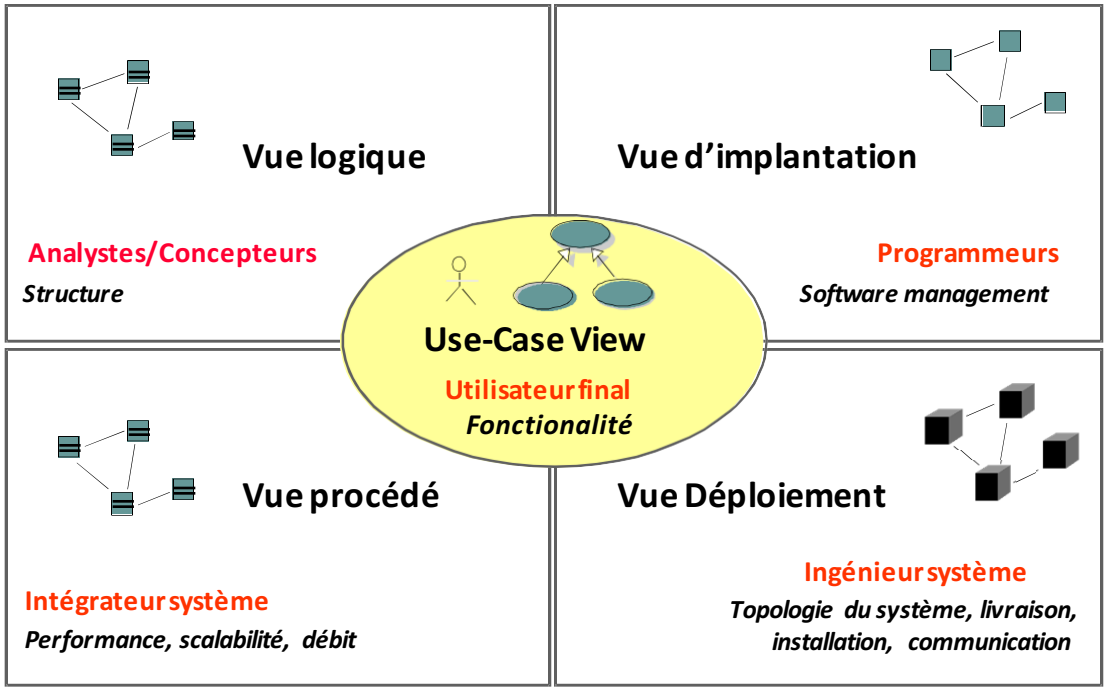
- Développement orienté modèles
  - Spécifier un modèle indépendant de la plateforme sur laquelle il sera déployé
  - Spécifier la ou les plateformes
  - Choisir une plateforme adaptée
  - Transformer le modèle de spécification en un modèle spécifique pour la plateforme

LES 4 PRINCIPES POUR CRÉER UN MODÈLE

- Un modèle influence énormément la façon d'aborder le problème et la solution
  - Vue du concepteur BD # vue du programmeur OO
- Chaque modèle peut être exprimer à différents niveaux de précision
  - Les meilleurs modèles permettent de choisir le niveau de détail en fonction de qui regarde et pourquoi il le regarde
- Les meilleurs modèles sont liés à la réalité
- Un seul modèle n'est jamais suffisant
  - Tous les systèmes gagnent à être décrits avec plusieurs petits modèles relativement indépendant => comment assurer la cohérence entre les modèles

UN SEUL MODÈLE NE SUFFIT PAS!

- Créer un ou plusieurs modèles différents mais avec un point commun



UNIFIED MODELING LANGUAGE (UML)

- Langage visuel unifié
  - Tout le monde doit parler le même langage
- Langage pour spécifier
  - Executable-UML
  - Supposé précis et non ambigu
- Des liens vers +s langages de prog.
  - Java, C++, VB
  - RDMS ou OODMS
  - Génération de code et *reverse engineering*.

UML - HISTORIQUE

Années 80:

- Méthodes pour organiser la programmation fonctionnelle (Merise)
- Séparation des données et des traitements

Début des années 90:

- Apparition de la programmation objet: nécessite d'une méthodologie adaptée
- Apparition de plus de 50 méthodes entre 1990 et 1995

1994

- Consensus sur 3 méthodes
  - **OMT** de James Rumbaugh : représentation graphique des aspects statiques, dynamiques et fonctionnels d'un système
  - **OOD** de Grady Booch: concept de package
  - **OOSE** de Ivar Jacobson: description des besoins de l'utilisateur

UML

Consensus entre OMT, OOD, OOSE pour créer une méthode commune:

- **UML** : Unified Modeling Language (Langage de Modélisation Unifié)

1997: Définition de la norme UML comme standard de modélisation des systèmes d'information objet par l'**OMG** (*Object Management Group*)

UML est employé dans l'ensemble des secteurs du développement informatique

- Systèmes d'information
- Télécommunication, défense
- Transport, aéronautique, aérospatial
- Domaines scientifiques

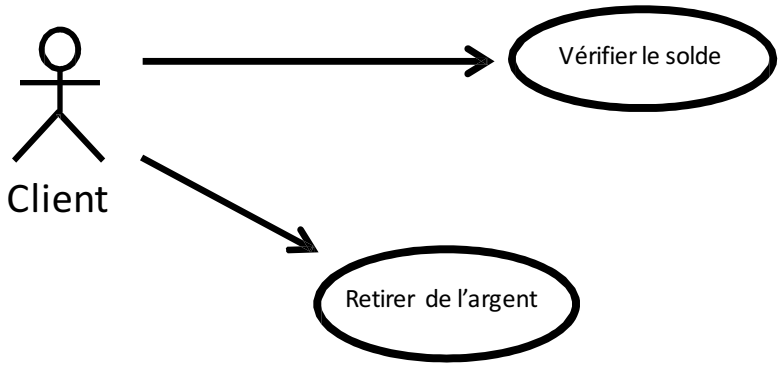
Mais pas seulement : on peut modéliser des comportement mécaniques, humain, etc.

UML ET RUP

- Un langage n'est pas suffisant, il faut aussi une méthode
- Les méthodes (*process*) qui fonctionnent le mieux avec UML sont :
  - Orienté par les *Use-case* ;
  - Centré sur l'architecture ;
  - Itératif et incrémental.
    - Les utilisateurs réagissent au fur et à mesure.
- *Rational Unified Process*

MODELE ORIENTÉ “USE CASE”

- Les *use-case* sont la base
  - Ils doivent être précis et concis
  - Ils sont compréhensibles par la majorité
  - Ils permettent de synchroniser les différents modèles
  - Ils décrivent l’ensemble des fonctions du système et les acteurs concernés



LES BASES D'UML

Les *éléments*

- Ce sont les abstractions essentielles au modèle.

Les *relations*

- Les relations expriment les liens existants entre les différents éléments.

Les *diagrammes*

- Un diagramme est une représentation visuelle de l'ensemble des éléments qui constituent le système
- Ils servent à visualiser un système sous différents angles (utilisateur, administrateur par ex.)
- Dans les systèmes complexes, un diagramme ne fournit qu'une vue partielle du système
  - L'ensemble des diagrammes réunis permet d'obtenir une vue globale du système à concevoir
  - Chaque diagramme va permettre de modéliser ou spécifier une vue (spécificité) du système à concevoir

UML - LES VUES

Vue des cas d'utilisation

- Description du modèle vu par les acteurs du système
- Besoins attendus pour chaque acteur
- Le **QUOI** et le **QUI**

Vue logique

- Définition du système vu de l'intérieur
- **COMMENT** satisfaire les besoins des acteurs

Vue d'implémentation

- Dépendances entre les modules

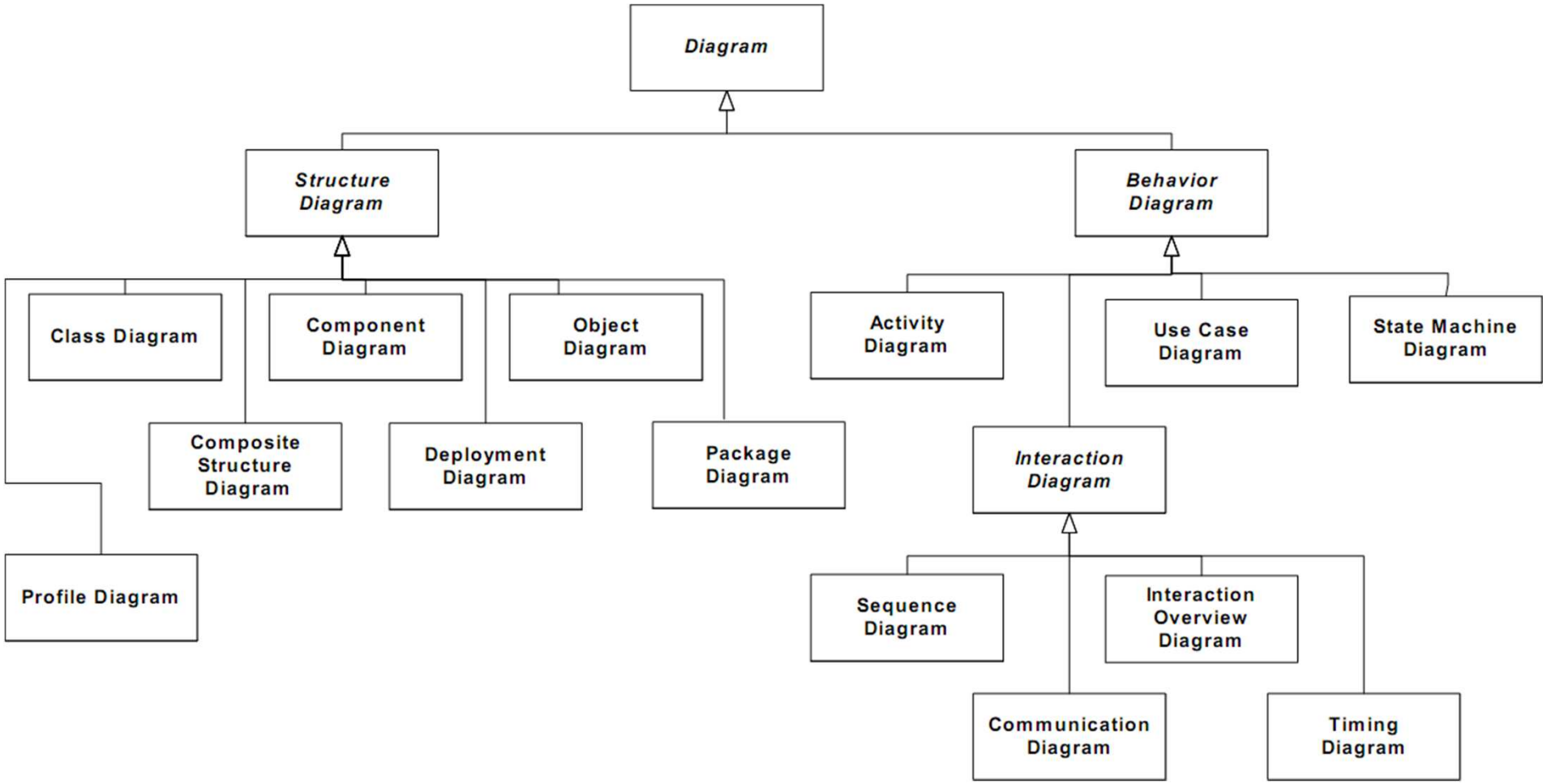
Vue des processus

- Vue temporelle et technique
- Mise en œuvre des notions de tâches concurrentes, synchronisation…

Vue de déploiement

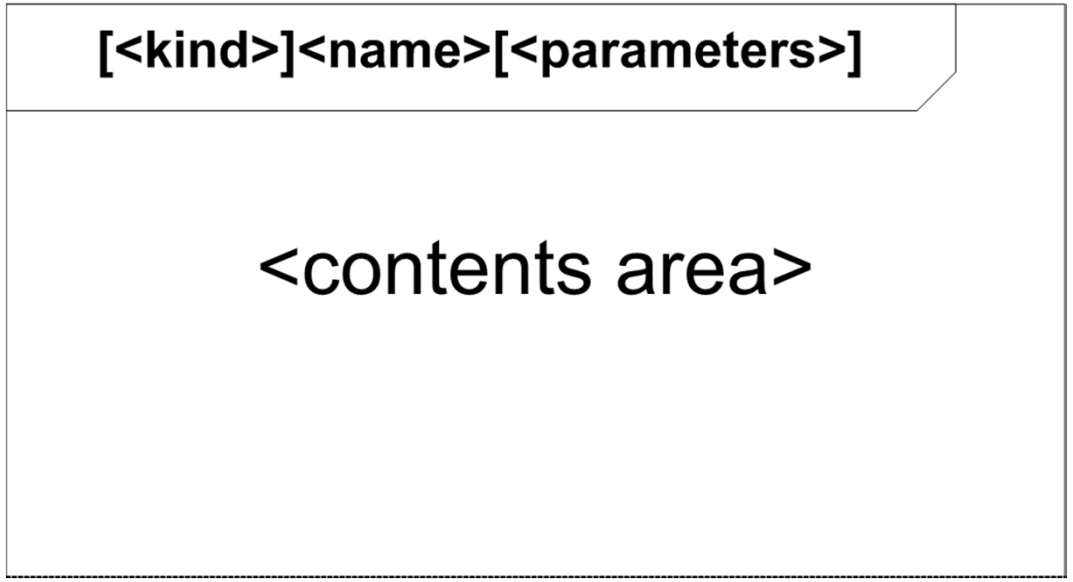
- Position géographique et architecture physique de chaque élément
- Le **OÙ**

UML – LES DIAGRAMMES



LES DIAGRAMMES

- Les diagrammes sont représentés dans des cadres(frames)
  - kind ∈ { activity, class, component, deployment, interaction, package, state machine, use case }
  - Forme simplifiée {act, class, cmp, dep, sd, pkg, stm, uc }



UML - OBJETS

- Objectifs
  - Encapsulation, abstraction, modularité, hiérarchie
  - Structure d'une classe
  - Relations entre une classe et un objet
  - Polymorphisme et généralisation
  - Les interfaces

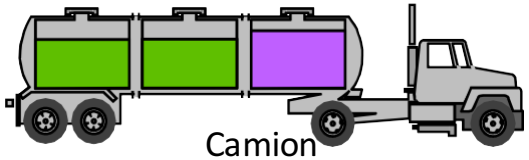
RETOUR SUR LES OBJETS ET LES MODELES

- Qu'est-ce qu'un objet ?
- Quatre concepts au centre de la COO
- Qu'est-ce qu'une classe ?
- Généralisation et polymorphisme
- Organisation des éléments modèles

QU'EST-CE QU'UN OBJET ?

- Un objet représente une entité physique, conceptuelle ou logicielle du monde réel.

  - Entité physique

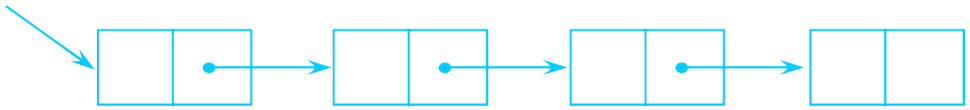


Camion

  - Entité conceptuelle

Procédé chimique

  - Entité logicielle



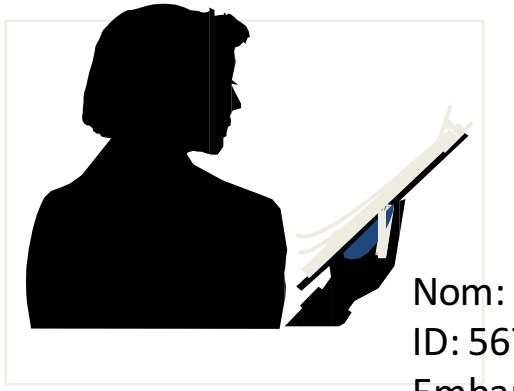
Liste chainée

QU'EST-CE QU'UN OBJET ?

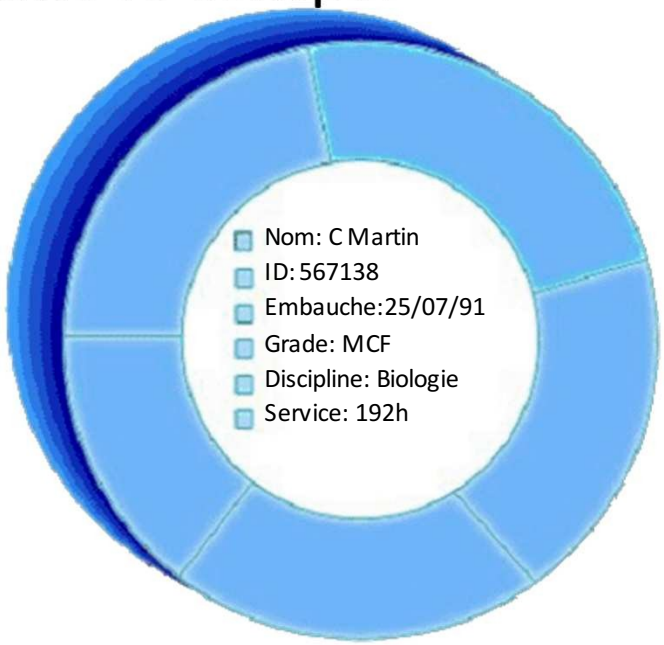
- Un objet a une frontière bien définie, une identité : **état** et ***comportement***.
  - L'état est représenté par des slots et des références
  - Le comportement est représenté par les opérations et les machines à états

ETAT D'UN OBJET

- L'état est une condition ou situation pendant la vie d'un objet qui satisfait une condition, effectue une activité ou attend pour un événement.
- L'état d'un objet peut changer dans le temps.



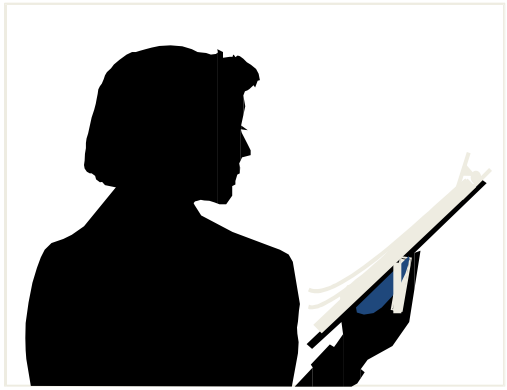
Nom: C Martin
ID: 567138
Embauche: 25/07/1991
Grade: Maitre de conférence
Discipline: Biologie
Service dû : 192h



Professeur Martin

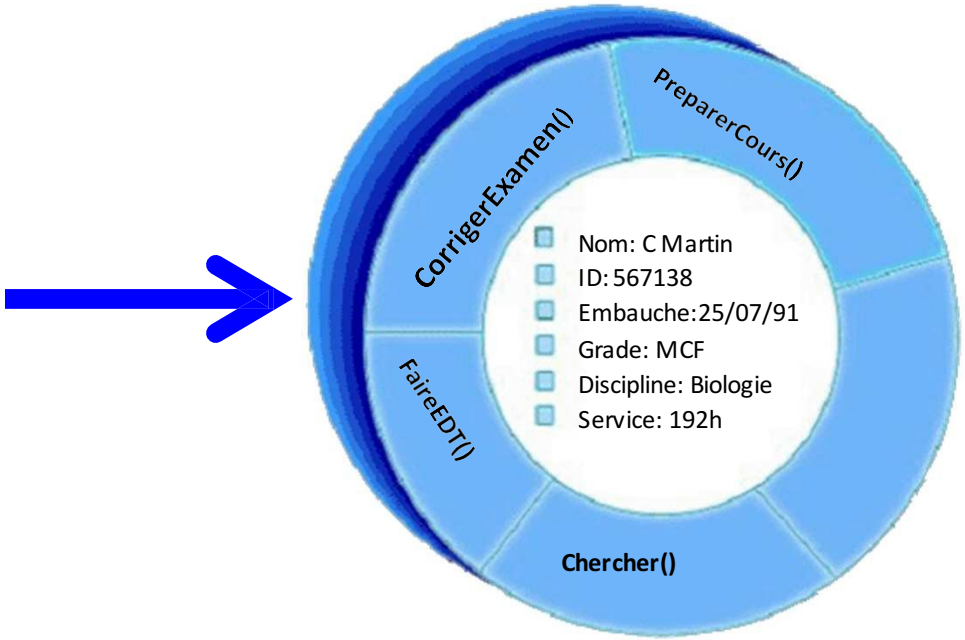
COMPORTEMENT D'UN OBJET

- Le comportement détermine comment l'objet agit ou réagit
- Le comportement visible d'un objet est son interface (ensemble d'opérations).



Comportement du professeur Martin

- Corriger les examens
- Préparer un nouveau cours
- Chercher
- Faire l'emploi du temps



IDENTITÉ DES OBJETS

✈L'identité d'un objet est unique même si l'état est le même que celui d'un autre objet

Professeur "C Martin"
enseigne la biologie

Professeur "C Martin"
enseigne la biologie

CONCEPTS FONDAMENTAUX DE LA COO

- Abstraction
- Encapsulation
- Modularité
- Hiérarchie (Héritage)
- Polymorphisme

ABSTRACTION

- Caractéristiques qui differencie une entité (objet) des autres
- Dépend de la perspective et di contexte
- N'est pas une manifestation concrète, dénote l'essentiel

ENCAPSULATION

Mécanisme consistant à rassembler, au sein d'une même structure, les données et les traitements

- Définition des attributs et méthodes au niveau de la classe

L'implémentation de la classe est cachée pour l'utilisateur

- Définition d'une interface : vue externe de l'objet

Possibilité de modifier l'implémentation sans modifier l'interface

- Facilité de l'évolution de l'objet

Préservation de l'intégrité des données

- L'accès direct aux attributs est interdit
- L'interaction entre les objets se fait uniquement grâce aux méthodes

ENCAPSULATION

Concepteur

Voiture
marque couleur immatriculation
démarrer **conduire** **arrêter**

Affiche :
La voiture est démarrée

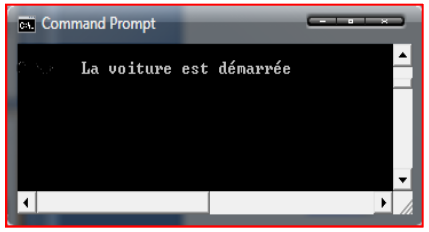
Utilisateur

J'aimerais créer une nouvelle twingo

```
Voiture twingo = new Voiture( );
```

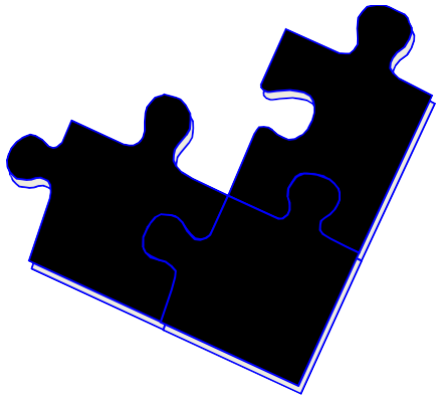
Que se passe-t-il si je démarre ma twingo?

```
twingo.démarrer( );
```

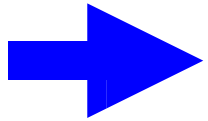


MODULARITÉ

- Casser un système en sous modules

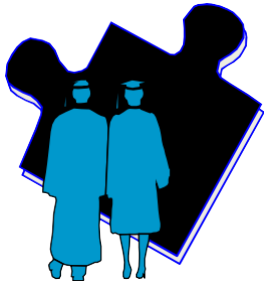


Système de gestion de l'université



Inscription administrative

Inscription pédagogique



Gestion des parcours

HIÉRARCHIE (HÉRITAGE)

Un objet spécialisé bénéficie ou hérite des caractéristiques de l'objet le plus général, auquel il rajoute ses éléments propres

- Création de nouvelles classes basées sur des classes existantes
- Transmission des propriétés (attributs et méthodes) de la classe mère vers la classe fille

Traduit la relation « est un … »

Deux orientations possibles

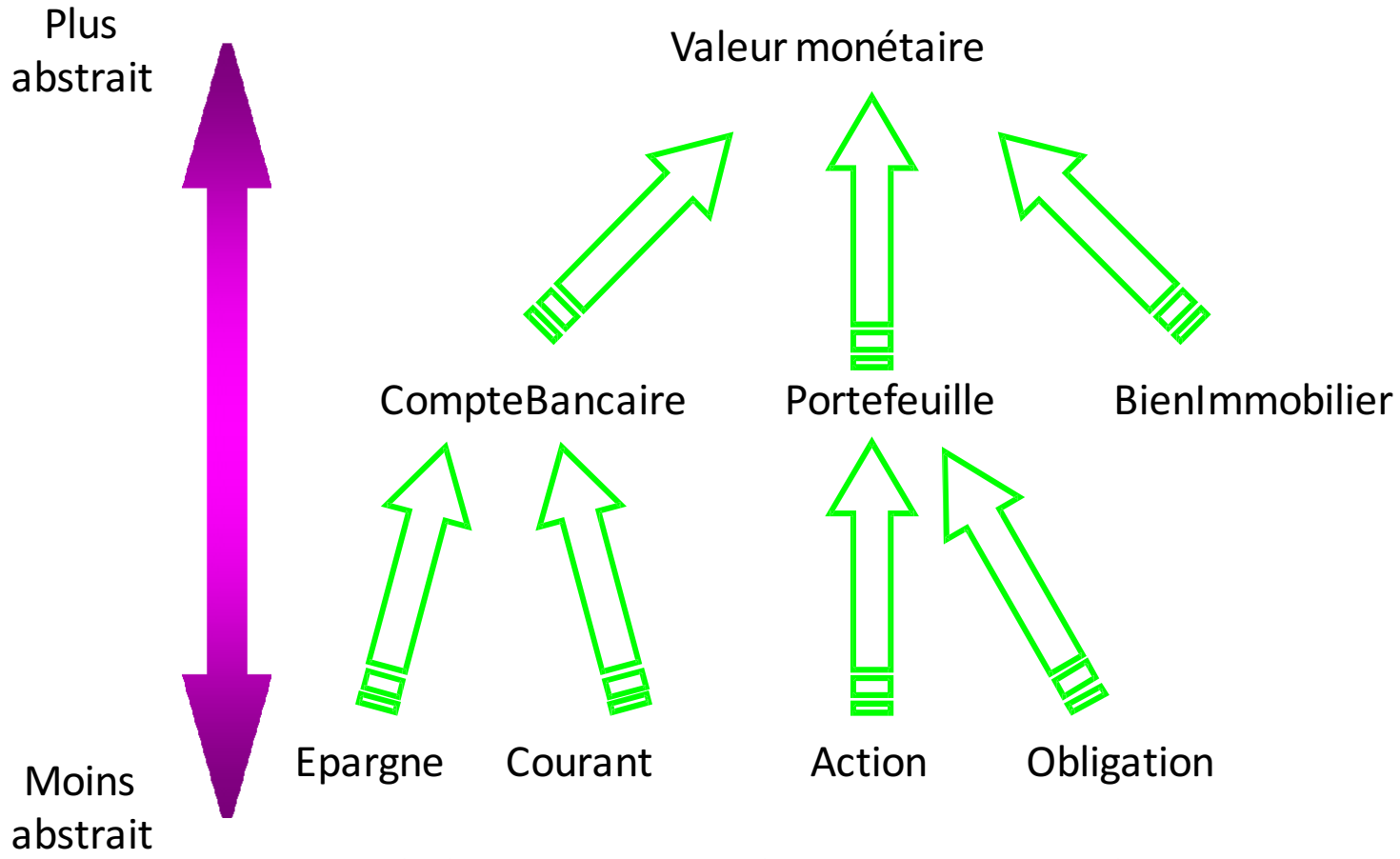
- Spécialisation : Ajout / adaptation des caractéristiques
- Généralisation : Regroupement des caractéristiques communes

Possibilité d'héritage multiple

Avantages

- Éviter la duplication du code
- Encourager la réutilisation du code

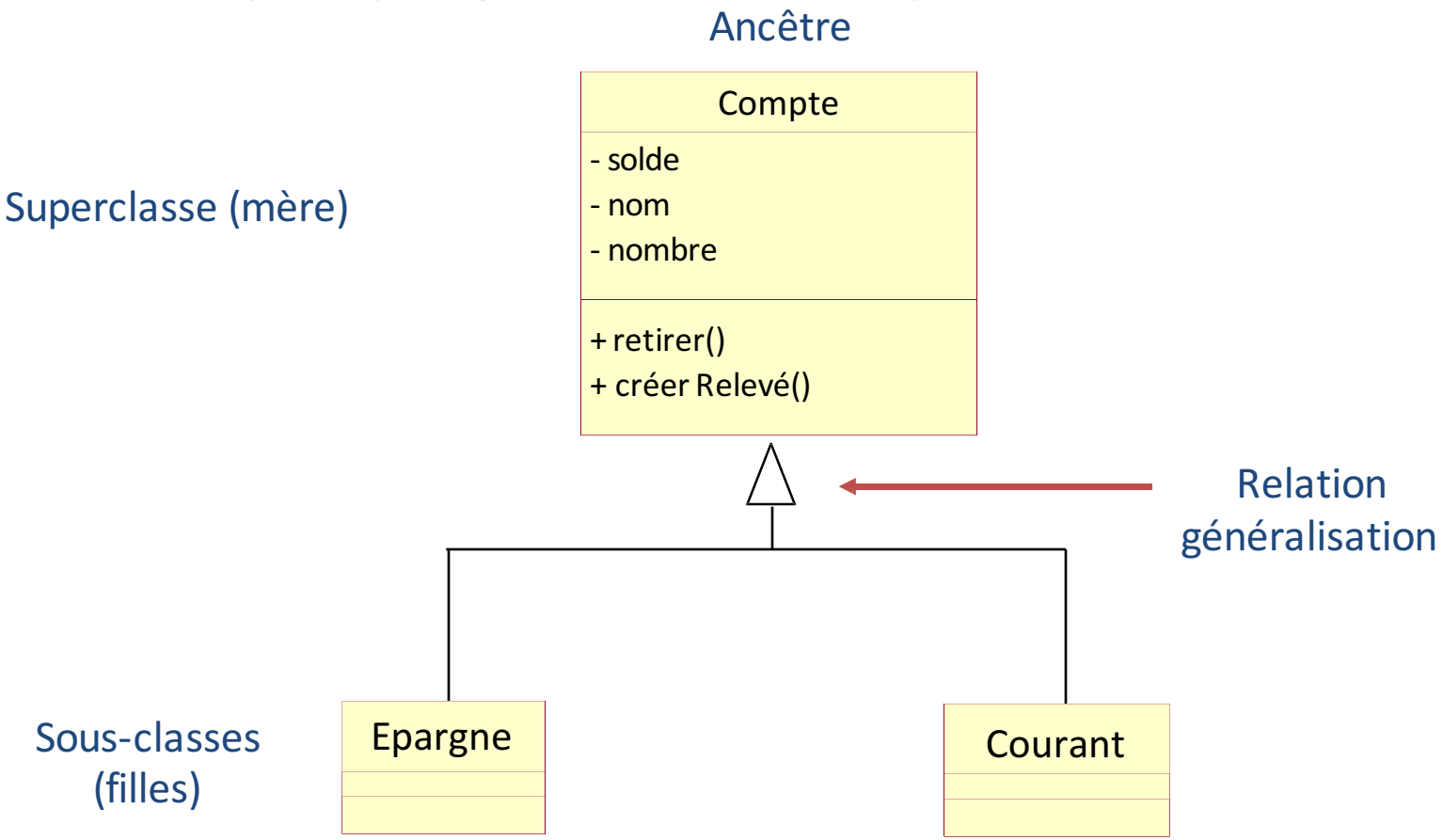
HÉRITAGE SIMPLE



Les éléments au même niveau hiérarchique devraient être au même niveau d'abstraction

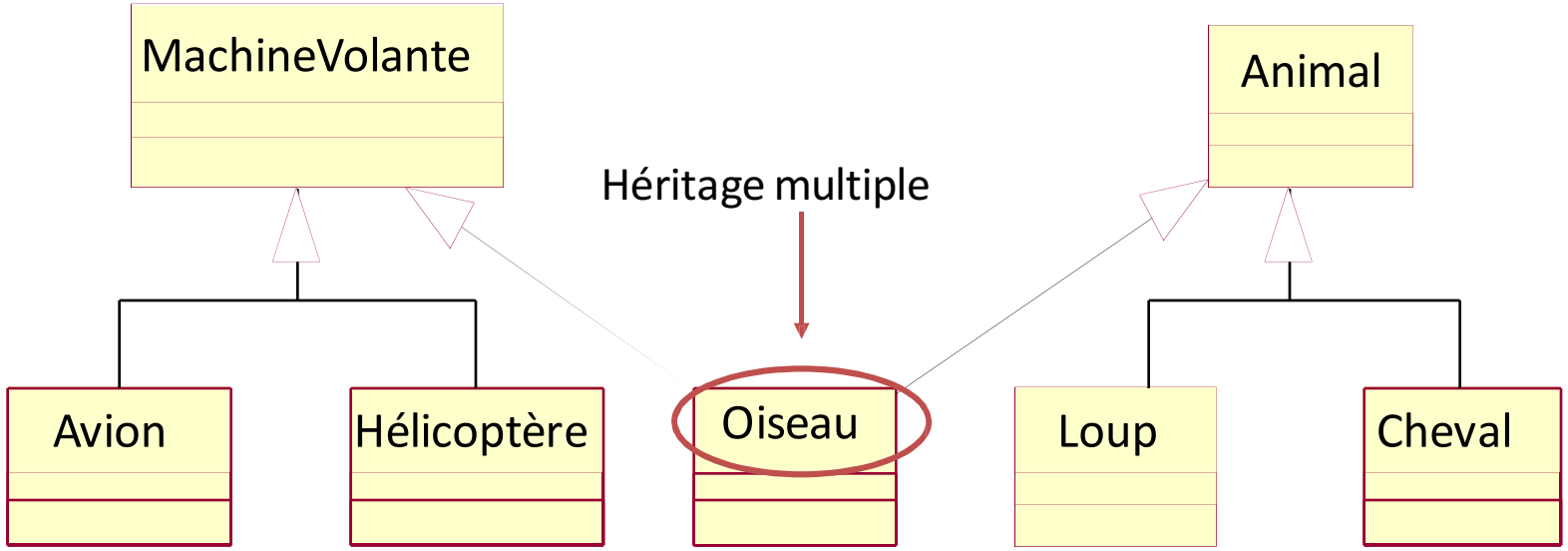
HÉRITAGE SIMPLE

- Un *CompteEpargne* **est-un** *Compte*



HERITAGE MULTIPLE

- Une classe peut hériter de plusieurs

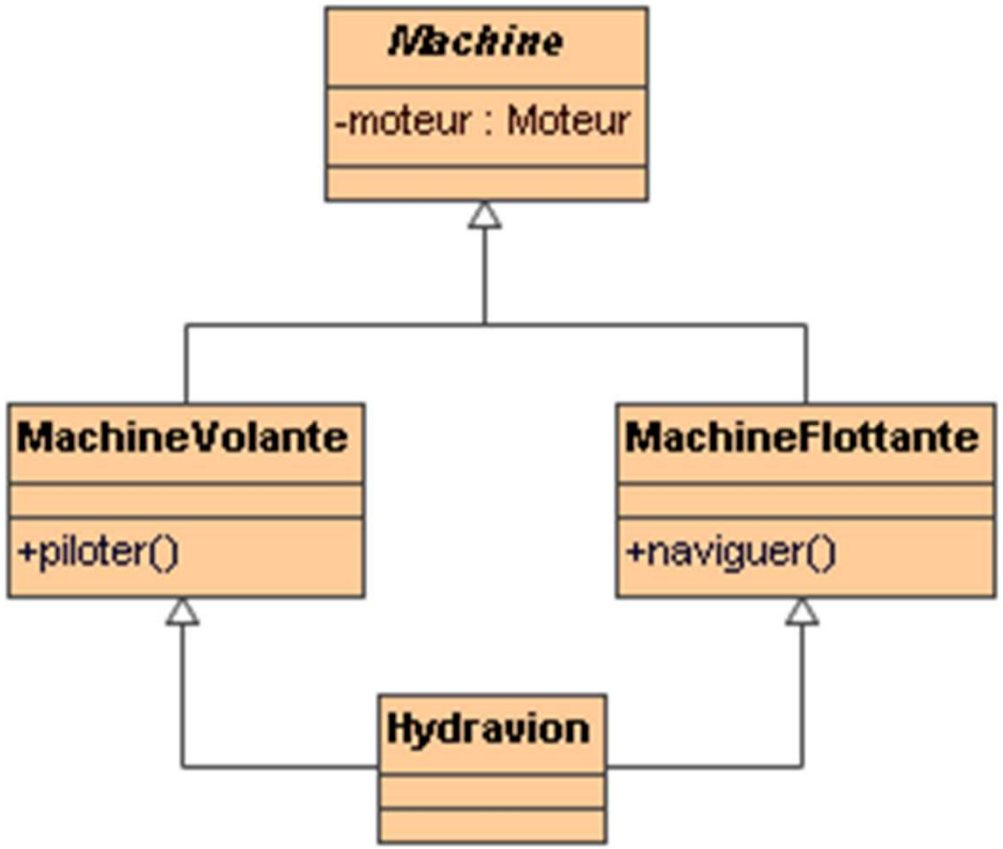


Utiliser l'héritage multiple avec prudence et seulement si indispensable!

Non supporté par la plupart des langages de POO (ex: Java, C#)

PROBLÈME AVEC L'HÉRITAGE MULTIPLE

- Combien de moteurs à l'hydravion ?



DE QUOI HÉRITE-T-ON?

- Une sous-classe hérite les attributs, les opérations et les références de ses parents.
- Une sous-classe peut:
  - Ajouter des attributs, des opérations, des références.
  - Redéfinir des opérations héritées.
- Les catégories communes sont montrées dans la classe mère la plus haute possible

POLYMORPHISME

Définition :

- *Poly* : plusieurs
- *Morphisme* : Forme

Faculté d'une méthode à pouvoir s'appliquer à des objets de classes différentes

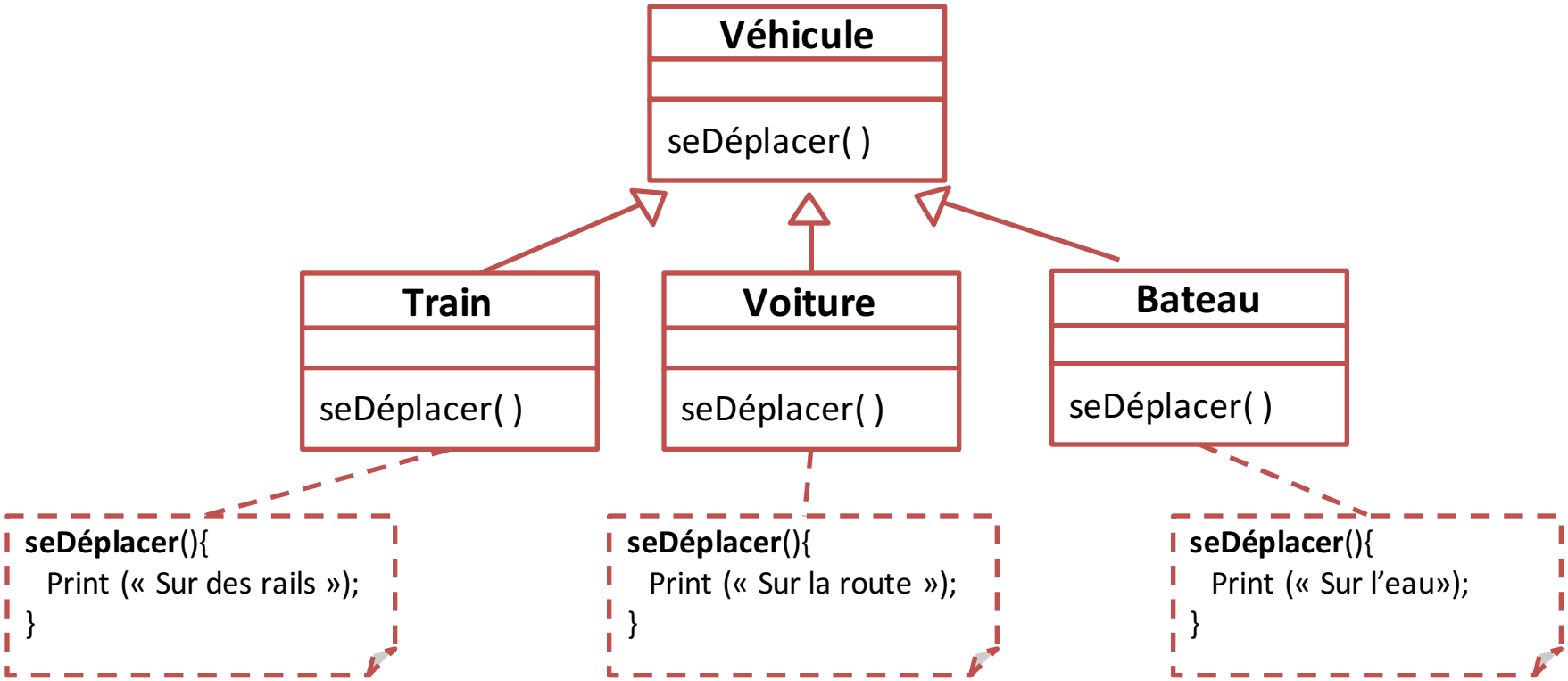
Capacité d'une classe à redéfinir une méthode héritée à partir d'une classe mère

- Surcharge

Avantages

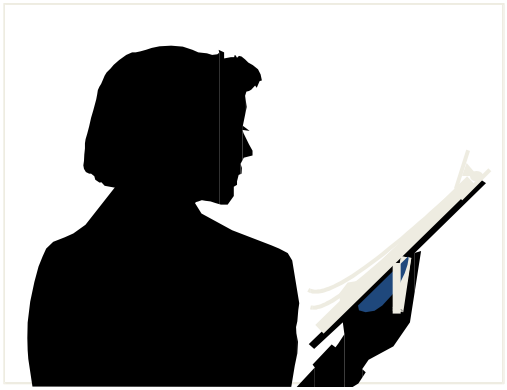
- Lisibilité du code
- Généricité du code

POLYMORPHISME



REPRESENTATION DES OBJETS EN UML

- Un objet (*InstanceSpecification*) est représenté par un rectangle.
- Le nom est souligné



Professeur C Martin

C Martin :
Professeur

Objet nommé

: Professeur

Objet anonyme

MODÈLES OU CLASSES

Contient la description d'un objet

- Modèle de l'objet effectif

Correspond à l'«*idée*» du monde réel de l'objet

- Analogie avec la philosophie platonnienne idéaliste :
  - « *Vous vous promenez dans la campagne, vous croyez avoir rencontré des troupeaux de chevaux. Quelle erreur! (…) Car le Cheval-Modèle, le Cheval-Idée, n'est ni noir ni blanc, il n'est d'aucune race chevaline. Il est cheval pur et vos sens ne vous le montreront jamais…* » *[Civilisation Grecque – A.Bonnard]*
- La classe ➔ l'« idée » du cheval
- Un pur sang arabe de couleur noire, dont le nom est ASWAD et qui boîte légèrement, est un **objet** instancié à partir de cette classe! ➔ ça c'est un cheval

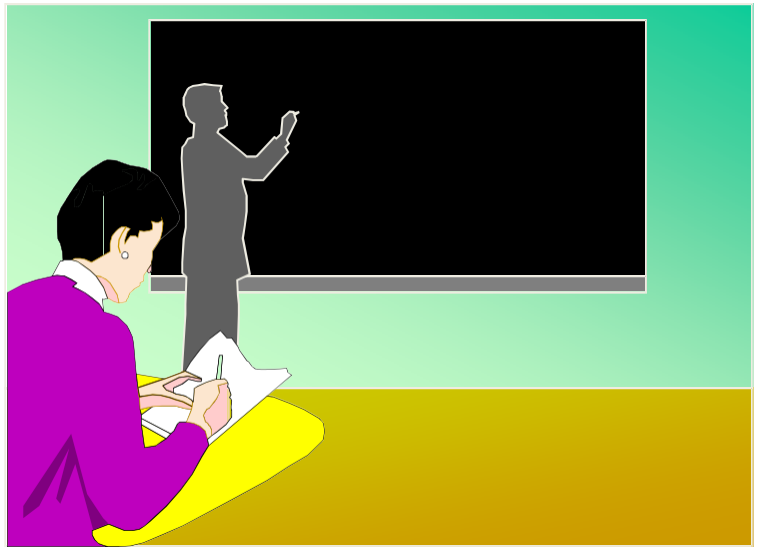
C'EST QUOI UNE CLASSE?

- Une classe décrit un ensemble d'objets qui partagent les mêmes *attributs*, *opérations*, *références*, et *sémantique*.
  - Un objet est l'instance d'une classe.
- Une classe est une abstraction car elle
  - Met en évidence certaines caractéristiques
  - Supprime d'autres caractéristiques

LA CLASSE COURS

Classe
Cours

Attributs
Nom
Salle
Durée
Crédits
Semestre



Comportement
Ajouter un étudiant
Enlever un étudiant

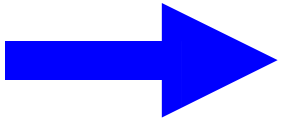
CLASSES VS. OBJETS

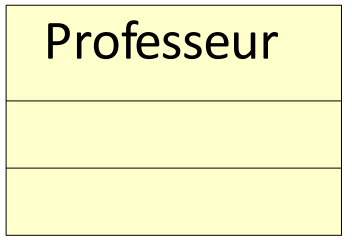
- La classe est une définition abstraite
  - Elle définit la structure et le comportement de chaque objet issue de cette classe
  - Sert de modèle pour la création d'instances

Professeur T

Professeur M

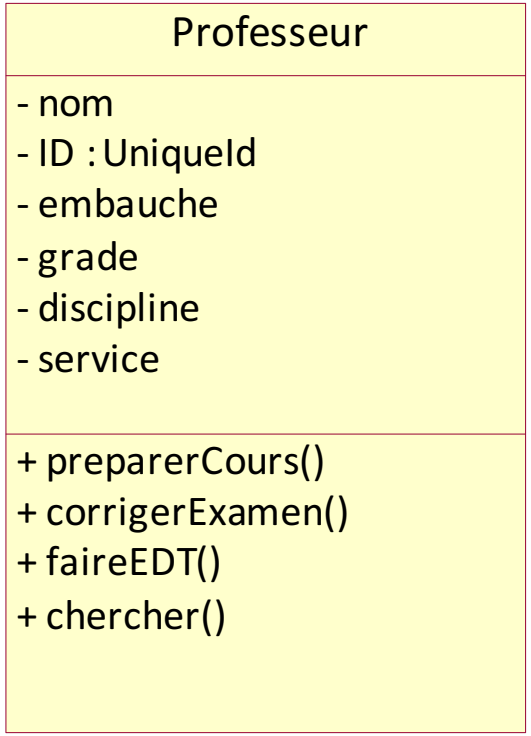
Professeur A





LES CLASSES EN UML

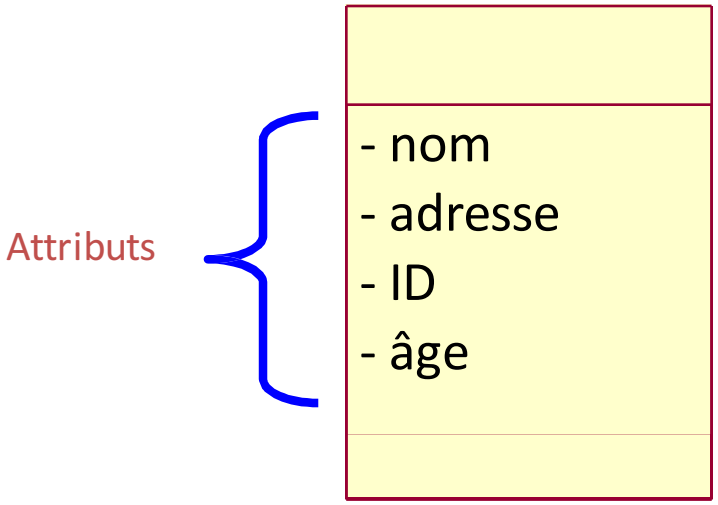
- Une classe est représentée par un rectangle avec 3 compartiments
  - Le nom de la classe
  - La structure (les attributs)
  - Le comportement (opérations)



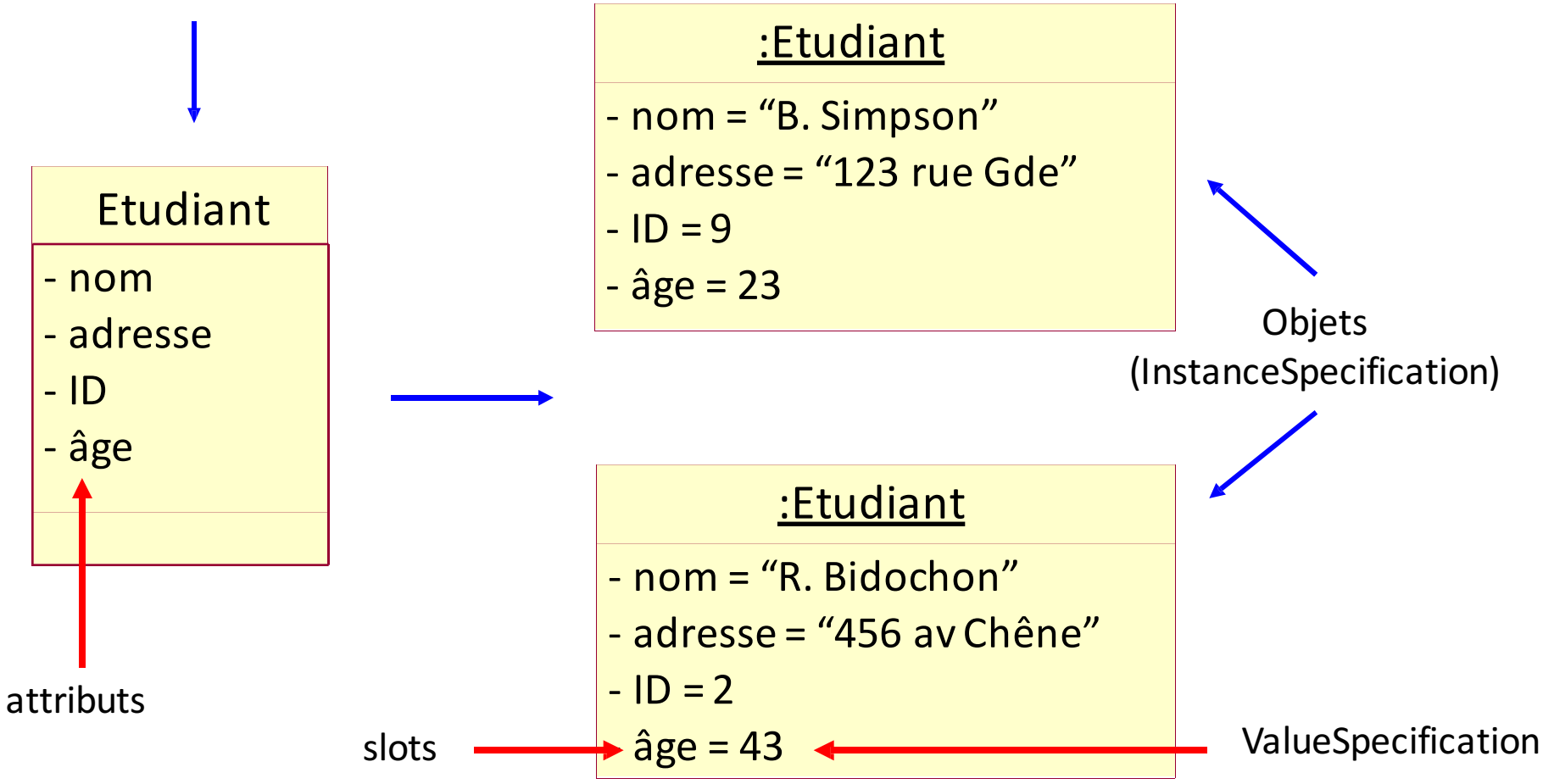
ATTRIBUTS DES CLASSES

✈ Un attribut est une propriété structurelle nommée dont le type décrit le domaine des valeurs que l'instance peut prendre.

- Une classe peut avoir un nombre quelconque d'attributs y compris 0.



ATTRIBUTS ET VALEURS



CLASSES ET OPÉRATIONS

- Un service qui peut être invoqué par un objet pour effectuer un comportement. Une opération a une signature, qui définit les paramètres formels possibles
- Une classe peut avoir un nombre quelconque d'opérations