Modélisation des Systèmes d'Information

Jean-Yves Antoine

http://www.info.univ-tours.fr/~antoine

Modèle objet

Jean-Yves Antoine

U. Bretagne Sud - UFR SSI - IUP Vannes

année 2001-2002

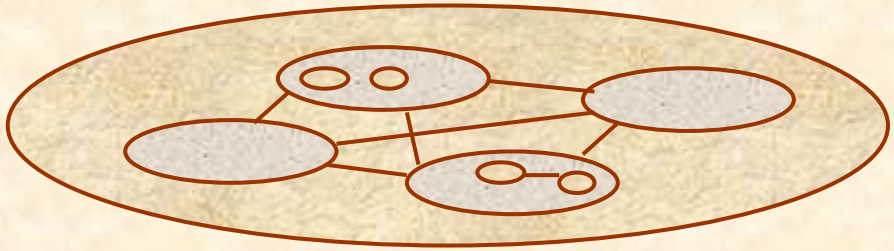
Plan du chapitre

❶ Pourquoi la modélisation objet

❷ Concepts principaux en modélisation objet

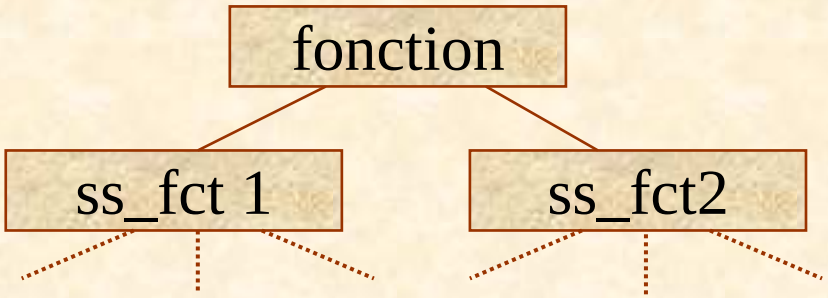
Pourquoi l'objet : quelle architecture logicielle ?

- Développement logiciel = décomposer / réunir
  - Diviser pour comprendre : décomposer le système en sous-parties pour maîtriser la complexité
  - Réunir pour construire : assurer la cohérence globale
- Architecture logicielle : comment décomposer ?



Programmation classique

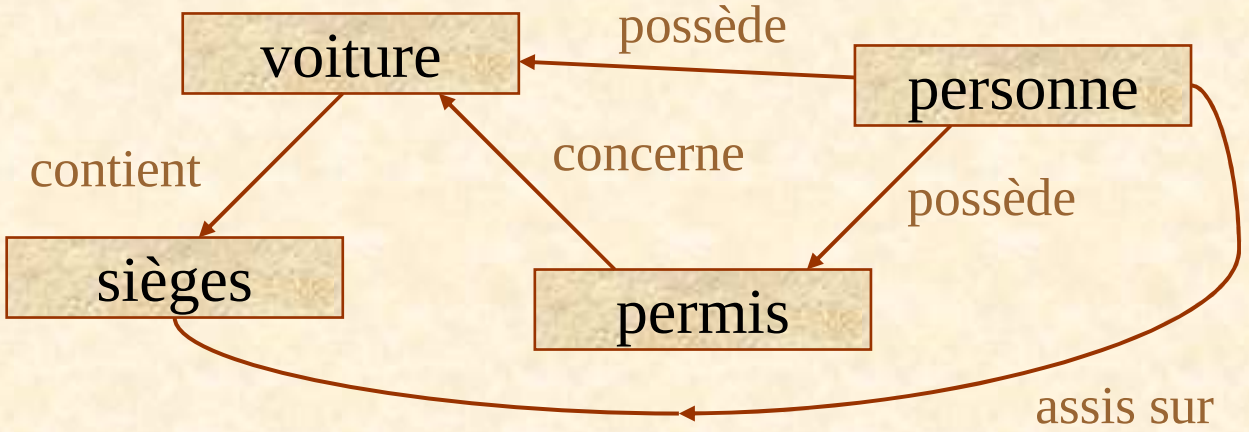
- Décomposition fonctionnelle
  - Décomposition suivant des critères fonctionnels
  - Décomposition hiérarchique :



- Limitations
  - Hiérarchie fonctionnelle : remise en cause difficile
  - Adaptabilité limitée et coûts des erreurs important

Programmation objet

- Décomposition « systémique »
  - Ensemble hétérarchique de composants (les objets) indépendants (encapsulation) dont la collaboration dynamique fonde les fonctionnalités du système.
  - Objets défini comme une abstraction du monde réel



Programmation objet : avantages

- Objets abstractions du domaine
  - Analyse / conception facilitée
- Couplage hétérarchique faibles entre objets
  - Adaptabilité
  - Réutilisabilité / portabilité
  - Adéquation avec un cycle itératif de développement
- Equilibre traitements / données : cf. supra…

Approche objet : historique

Une nouvelle approche … qui ne date pas d' hier !

- **1967 — Simula** (Dahl, Myhrhaug et Nygard)
  - Algol + encapsulation + héritage
- **1972 — SmallTalk** (Xerox Palo Alto, A. Kay)
  - IHM graphiques — Premier langage réellement orienté objet
- **1980s**
  - C++ : sur-ensemble de C (B. Stroustrup, AT&T)
  - Eiffel : LOO + génie logiciel (B. Meyer)
  - Ada : pas un LOO ; gros projets (Dod) (J. Ichbiah, Bull)
- **1990s — Java** (Sun)
  - Applications réseau : Toile

Concepts objets

- Objets
  - état
  - comportement
  - identité
- Relations entre objets
  - message
  - interface
  - abstraction
- Classe / Instance
- Hiérarchie
- Polymorphisme

Objet

- Unité atomique d'un programme objet
  - représente un élément du domaine
  - abstraction : ce qui concerne spécifiquement le domaine
- Objet = Etat + Comportement + Identité
  - Etat : décrit les propriétés de l 'objet à un moment donné
  - Comportement : définit les propriétés dynamiques de l'objet : comment il agit et comment il réagit aux informations qui lui parviennent de son environnement.
  - Identité : caractérise de manière univoque l'existence propre de l'objet

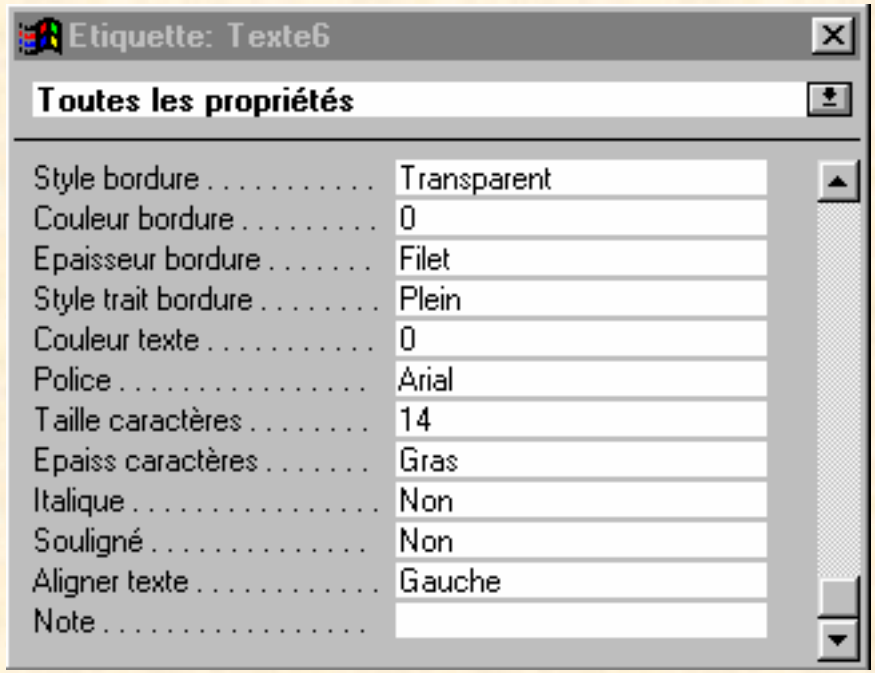
Objet : état

- Etat = ensemble d'attributs
  - Chaque attribut caractérise une propriété précise
  - Les attributs peuvent prendre à un moment donné toute valeur d'un domaine de définition donné.
- Exemple : Base de Données « UBS »

Une personne
nom = Antoine age = 35 métier = prof

Objet : état

- Exemple : OLE dans Microsoft™ Access



attributs de
l'objet graphique

Objet : comportement

- Comportement = ensemble de méthodes
  - Méthode = opération atomique qu'un objet réalise soit de son propre chef, soit en réaction à une stimulation de l'environnement (envoi de message d'un autre objet).
  - L'action réalisée dépend de l'état de l'objet
- Exemple : BD «UBS»

Une personne
nom age métier
ChangerAge

Objet : comportement

- Types de méthodes
  - constructeur : crée et initialise l'objet

    ⇨ CreerPersonne
  - modificateur : modifie l'état de l'objet

    ⇨ ChangerAge
  - observateur : donne une information sur l'état

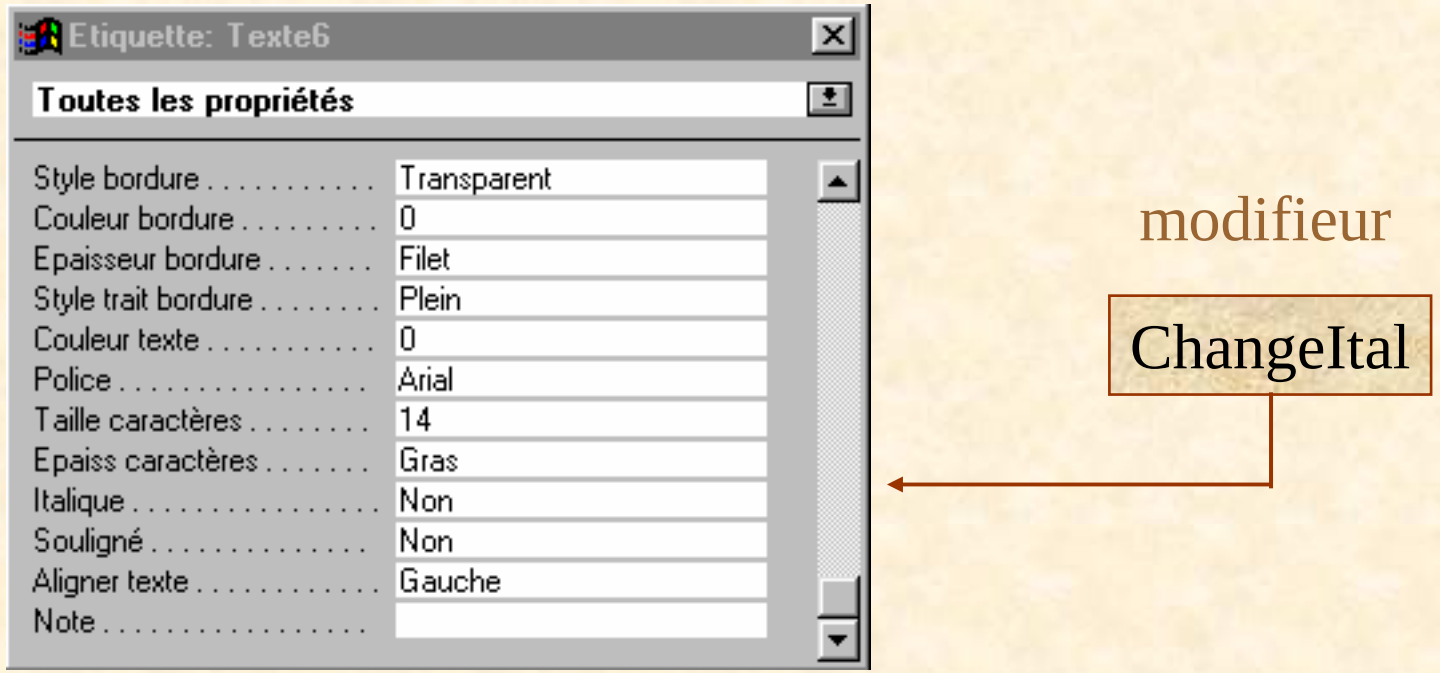
    ⇨ DonnerAge
  - destructeur : détruit l'objet

    ⇨ OterPersonne
- **Cycle de vie d 'un objet**

Tout objet est créé, évolue et est détruit ...

Objet : comportement

- Exemple : OLE dans Microsoft™ Access



Objet : identité

- Existence propre d'un objet
  - Distinguer des objets ayant le même état
  - Gérée implicitement : pas d'attribut correspondant
  - Si on le souhaite, on peut cependant rajouter un identifiant dans l'état de l'objet (clé naturelle) :

    - numéro INSEE
    - numéro étudiant
    - numéro

  ↳ Notion de clé primaire en base de données

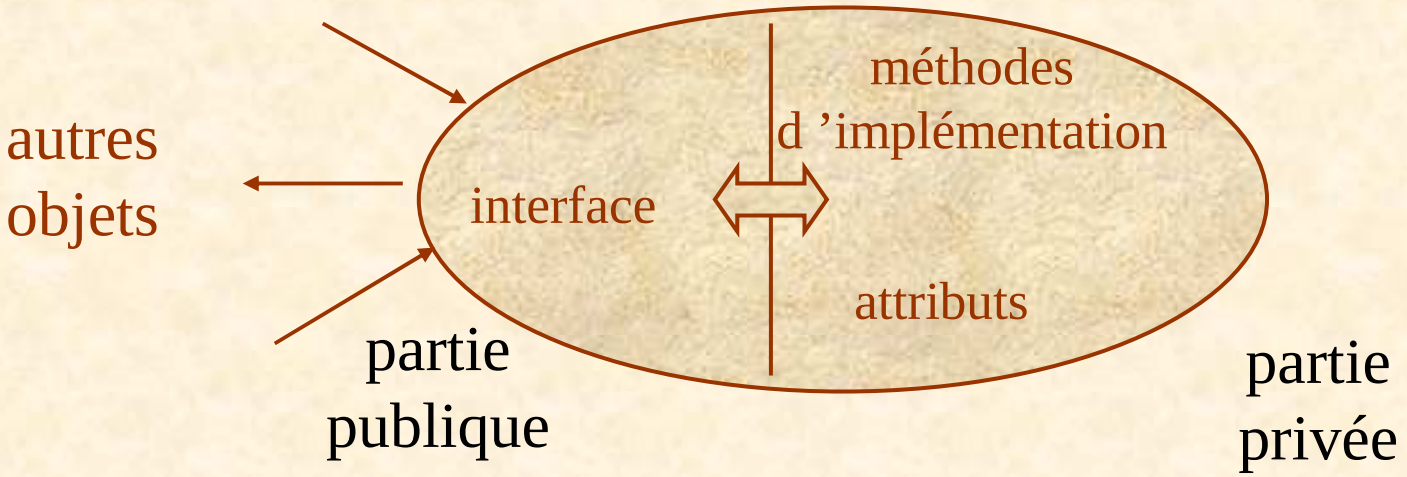
Relations entre objets : messages

- Système : société d'objets en relation
  - Dynamique du système : collaborations entre objets
  - Interaction non structurée par passage de messages
- Interface : contrôle de la modularité
  - Interface = méthodes invocables par l'extérieur
  - L'état de l'objet ne peut être modifié par l'extérieur qu'en invoquant une méthode de l'interface
  - L'objet peut refuser de répondre à une invocation externe

L'objet garde toujours la maîtrise de son état

Relations entre objets : interface

- Abstraction de données
  - Un objet est complètement défini par son interface
  - Les autres objets n'ont pas à connaître l'implémentation interne de l'objet pour communiquer avec lui.

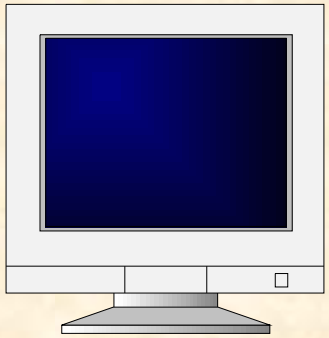


Relations entre objets : exemple BD «UBS»

Une personne
nom = Antoine age = 35 métier = prof

- Méthodes publiques : interface
  - ChangerNom, ChangerAge, ChangerMetier
  - DonnerNom, DonnerAge, DonnerMetier
- Méthodes d ’implémentation
  - VerifMajuscule, VerifSupZero

Relations entre objets : exemple BD «UBS»



objet IHM
« saisie âge »

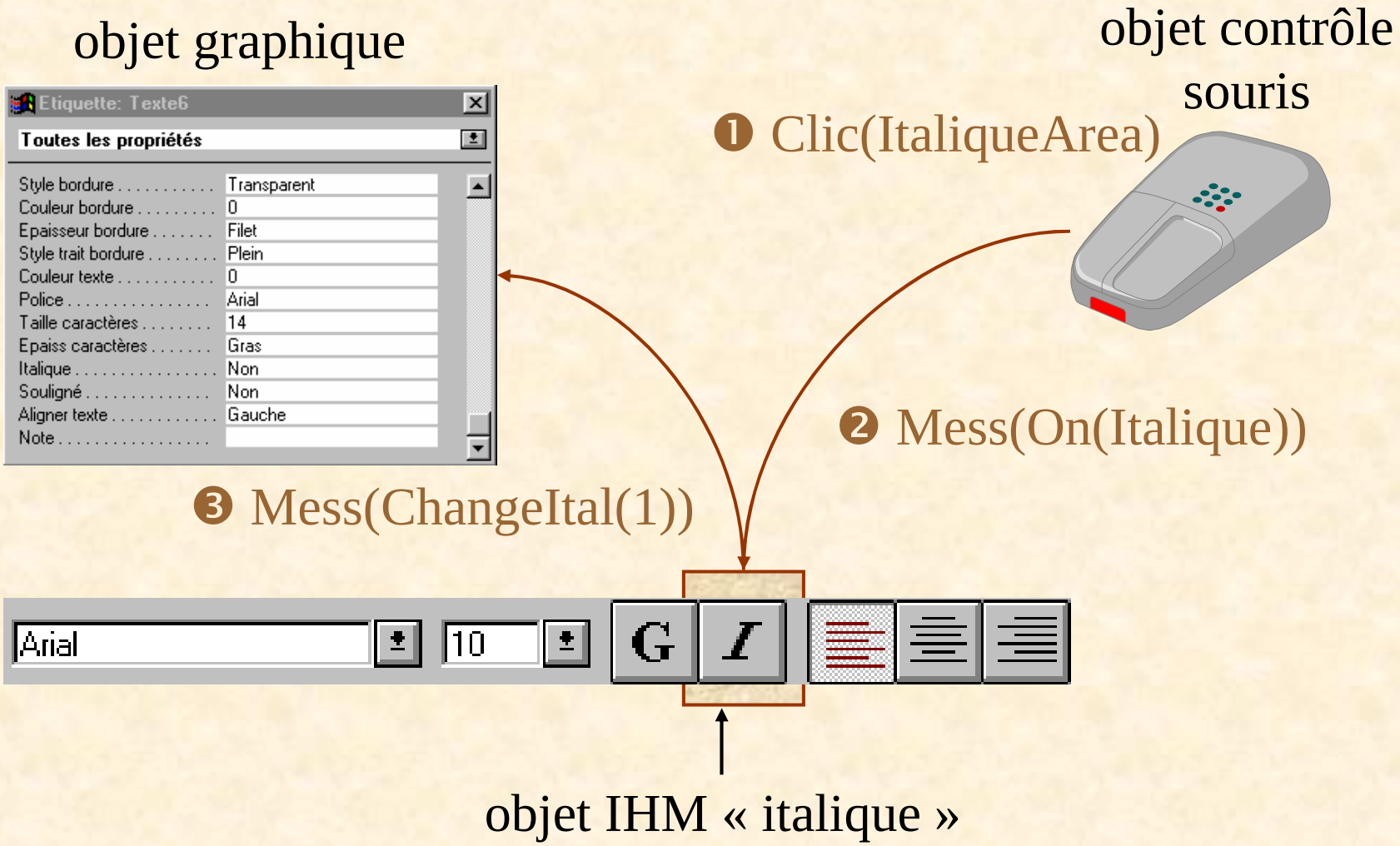
❶ Mess(ChangerAge(-36))

❷ VerifPos(-36)

❸ Mess(ERR)

objet
« personne : Antoine»

Relations entre objets : exemple «Access»

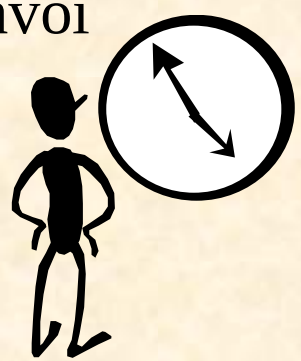


Relations entre objets : synchronisation

- Message simple
  - L'expéditeur reste bloqué jusqu'à ce que le destinataire réponde au message
  - Cas classique d'appel d'une méthode.
- Message synchrone
  - L'expéditeur reste bloqué jusqu'à ce que le destinataire accepte le message (réponse par un autre message).
- Message asynchrone
  - L'envoi du message ne bloque pas l'expéditeur. Celui ne sait même pas si le message sera traité…

Relations entre objets : synchronisation

- Autres types de synchronisation
  - Message minuté (ou borné) — Expéditeur bloqué jusqu'à acceptation du message par le distinataire à concurrence d'une durée donnée.
  - Message dérobant — Déclenchement d'une opération à la réception uniquement si le destinataire s'est mis en attente de message. Expéditeur libéré dès l'envoi



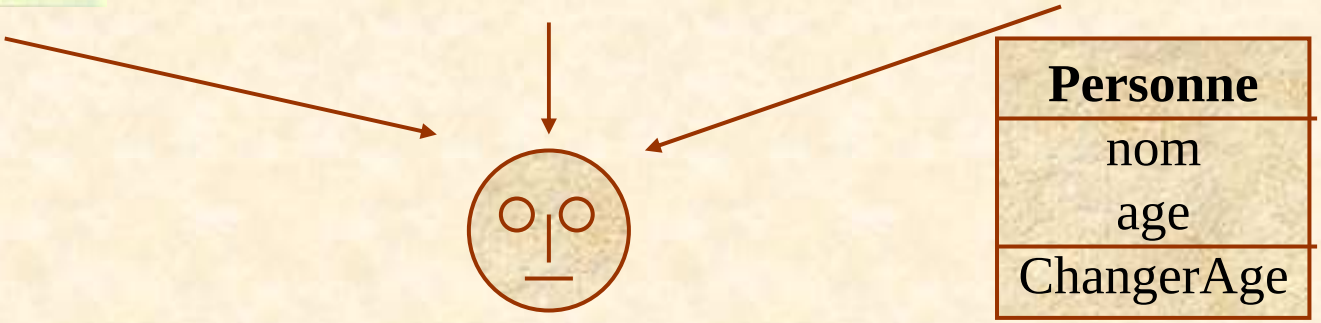
1°bilan : approche objet et qualité logicielle

- Encapsulation
  - Equilibre traitements / données
  - Modularité gérée harmonieusement
- Abstraction
  - Adaptabilité
  - Réutilisabilité / portabilité
  - Tracabilité
- Un «problème» en BD : persistance des objets
  - Le plus souvent, mécanismes ad hoc de sauvegarde

Classe

- La classe, une nouvelle étape dans l'abstraction
  - Rechercher les similitudes et ignorer les différences



  - Abstraction pour réduire la complexité

(cf. mémoire)

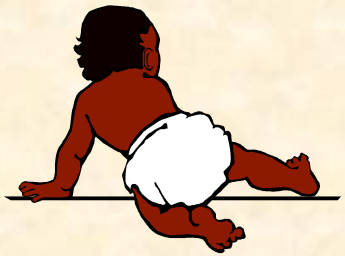
Classe

- Classe : vision intensionnelle
  - Classe = objet prototypique décrivant l'ensemble des propriétés (structure d'attribut, comportements) communs aux éléments de la classe
- Classe : vision extensionnelle
  - Classe = ensemble des objets réunis
- Classe et objet
  - Toute classe est un objet : état + comportement + identité

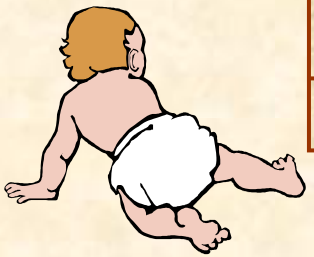
Classe : instanciation

- Instance
  - Tout objet d'une classe est appelé instance de la classe
  - La classe décrit la structure de ses instances : elles auront les même attributs et méthodes que la classe.
- Cycle de vie d'une instance
  - Création d'instance : constructeur ⇨ méthode de classe qui ne se sera pas dans le comportement de l'instance
  - L'état courant d'une instance est défini en contexte et en toute indépendance par l'objet créé.

Classe : exemple



Personne
nom = Parker age = 1
ChangerAge



Personne
nom = Bidochon age = 1
ChangerAge

instance de

instance de

Personne
nom = Perez age = 5
ChangerAge

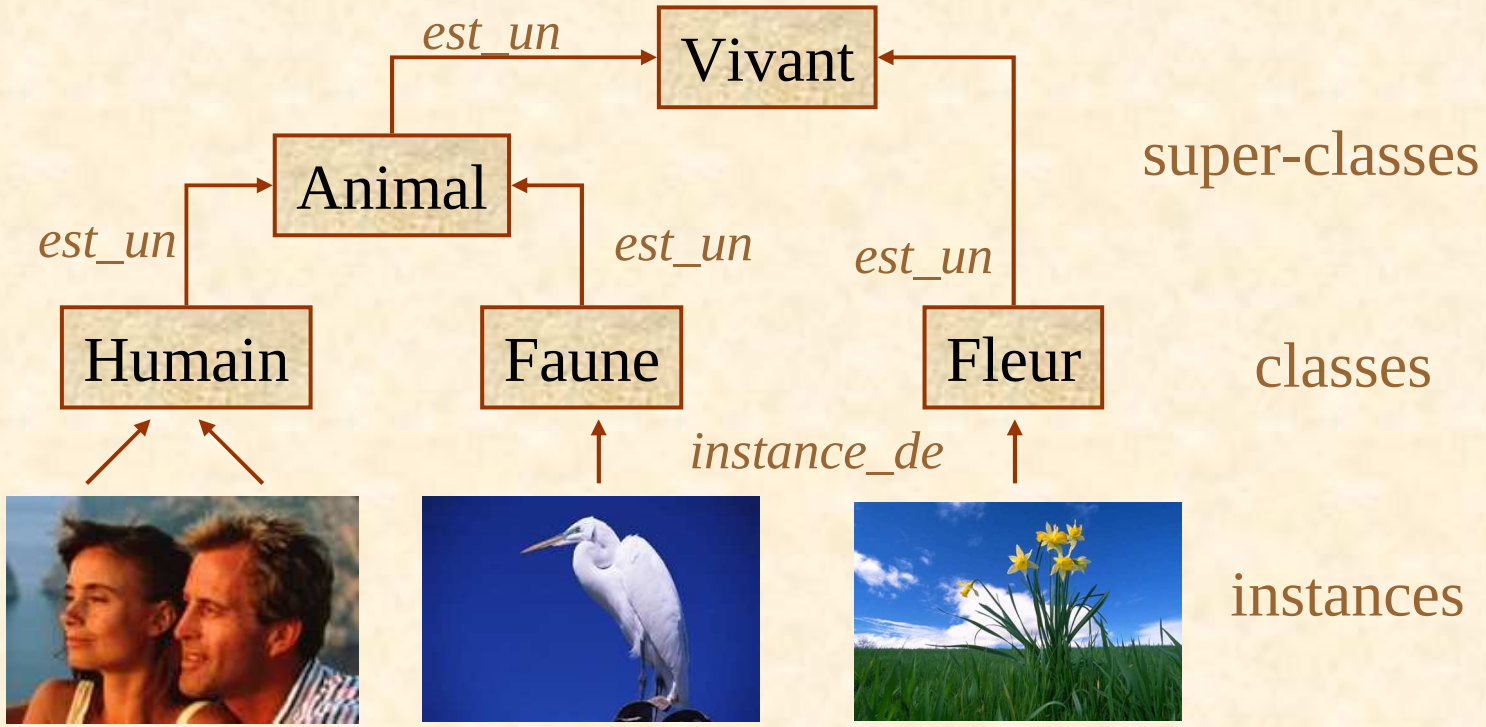
instance de

Personne
nom age
ChangerAge CreerPers

classe

Hiérarchie de classes

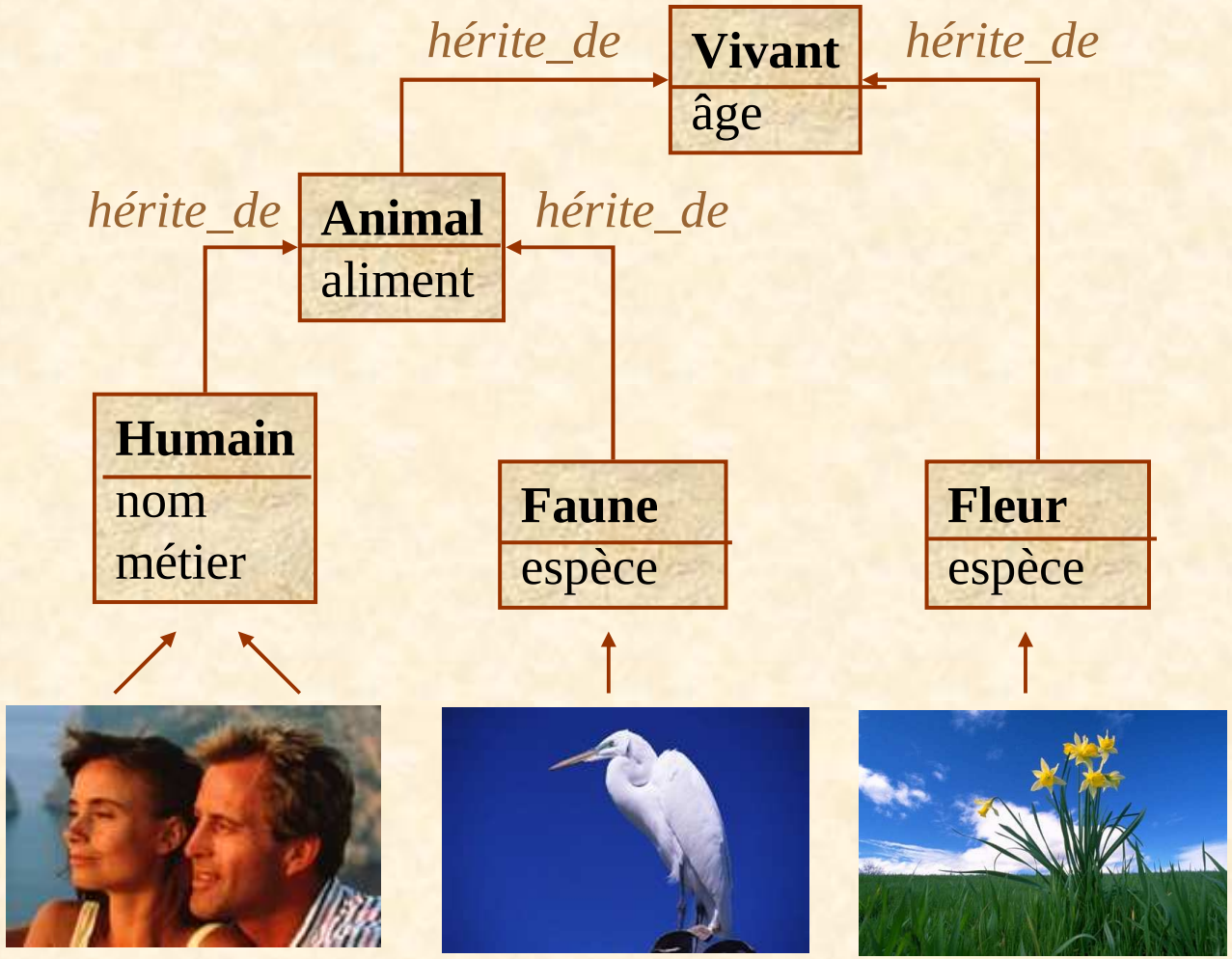
- La super-classe, ou l'abstraction récursive
  - Puisqu'une classe est un objet, elle peut être elle aussi considérée comme l'instance d'une classe supérieure



Héritage

- Héritage
  - Mécanisme de transmission des propriétés (attributs, comportement) d'une classe vers ses sous-classes
  - Ce qui est générique est défini au niveau de la super-classe, ce qui est spécifique est défini au niveau de la sous-classe
- Généralisation / spécialisation
  - Généralisation : mise en commun de propriétés communes entre différentes classes dans une super-classe
  - Spécialisation : création d'une sous-classe par mise en avant de propriétés spécifiques non communes à toutes les (classes) instances de la super-classe

Héritage : exemple



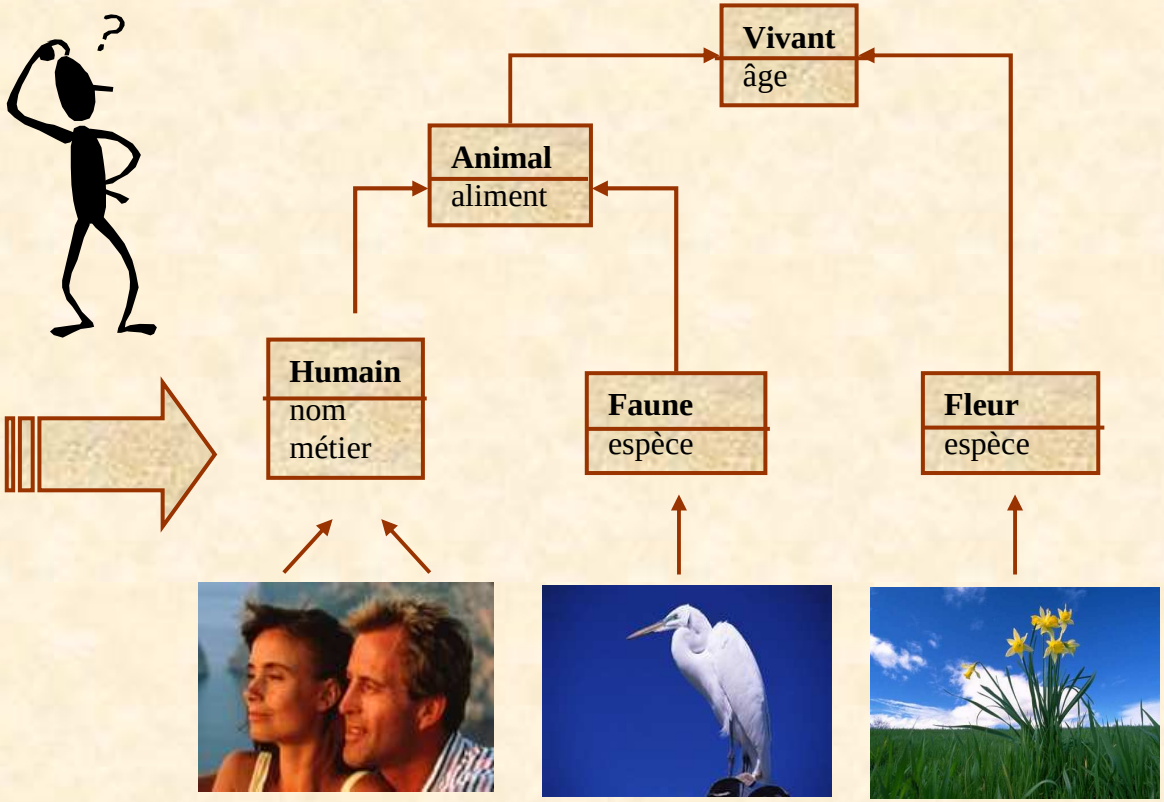
Classe abstraite

- Classe ne permettant pas d'instancier d'objets
- Exemple
  - classe Vivant
  - classe Animal

- Utilité
  - Représentation de concepts centraux pour l'application

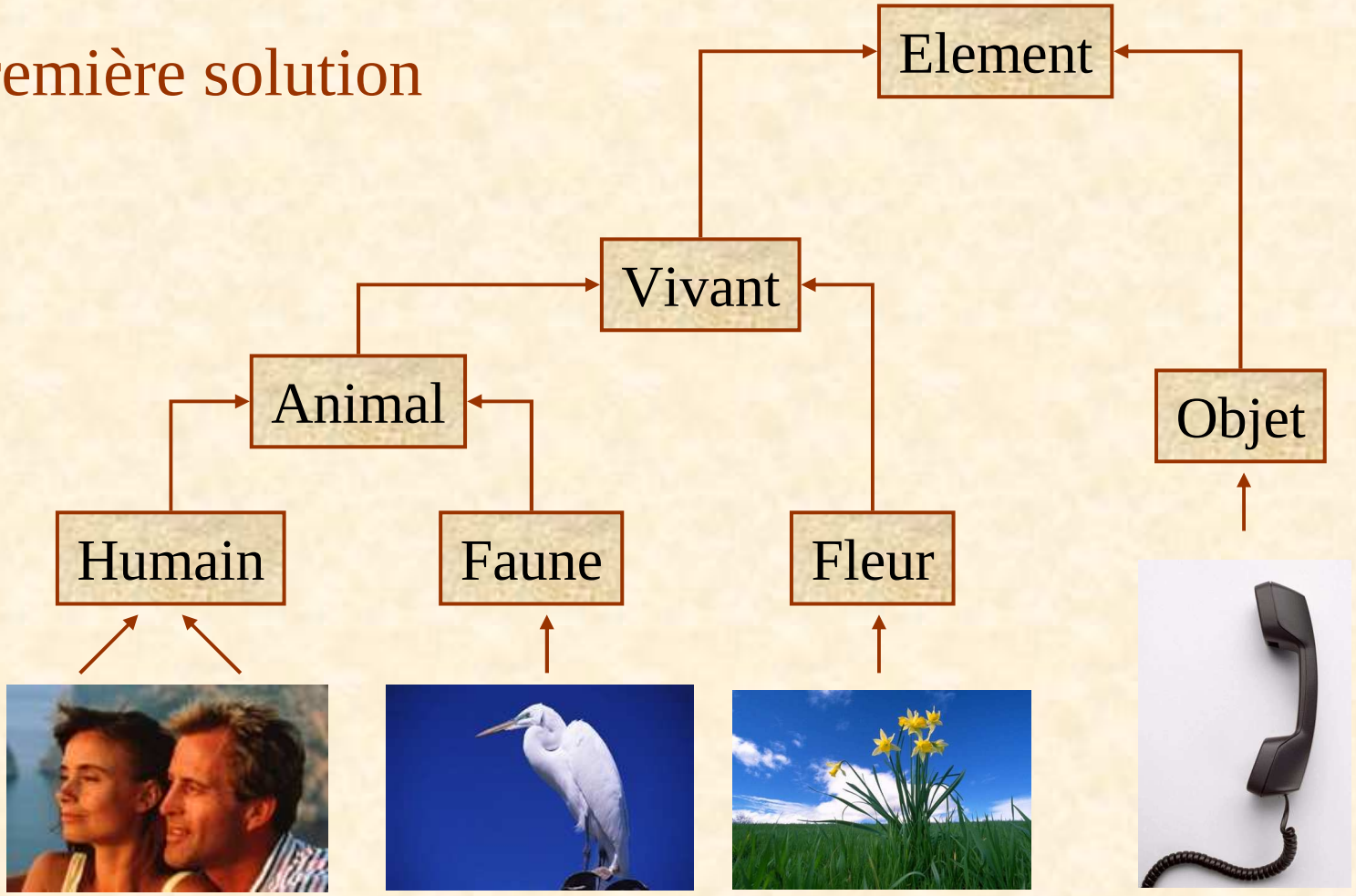
Héritage

- Difficultés de la classification ⇨ analyse/conception



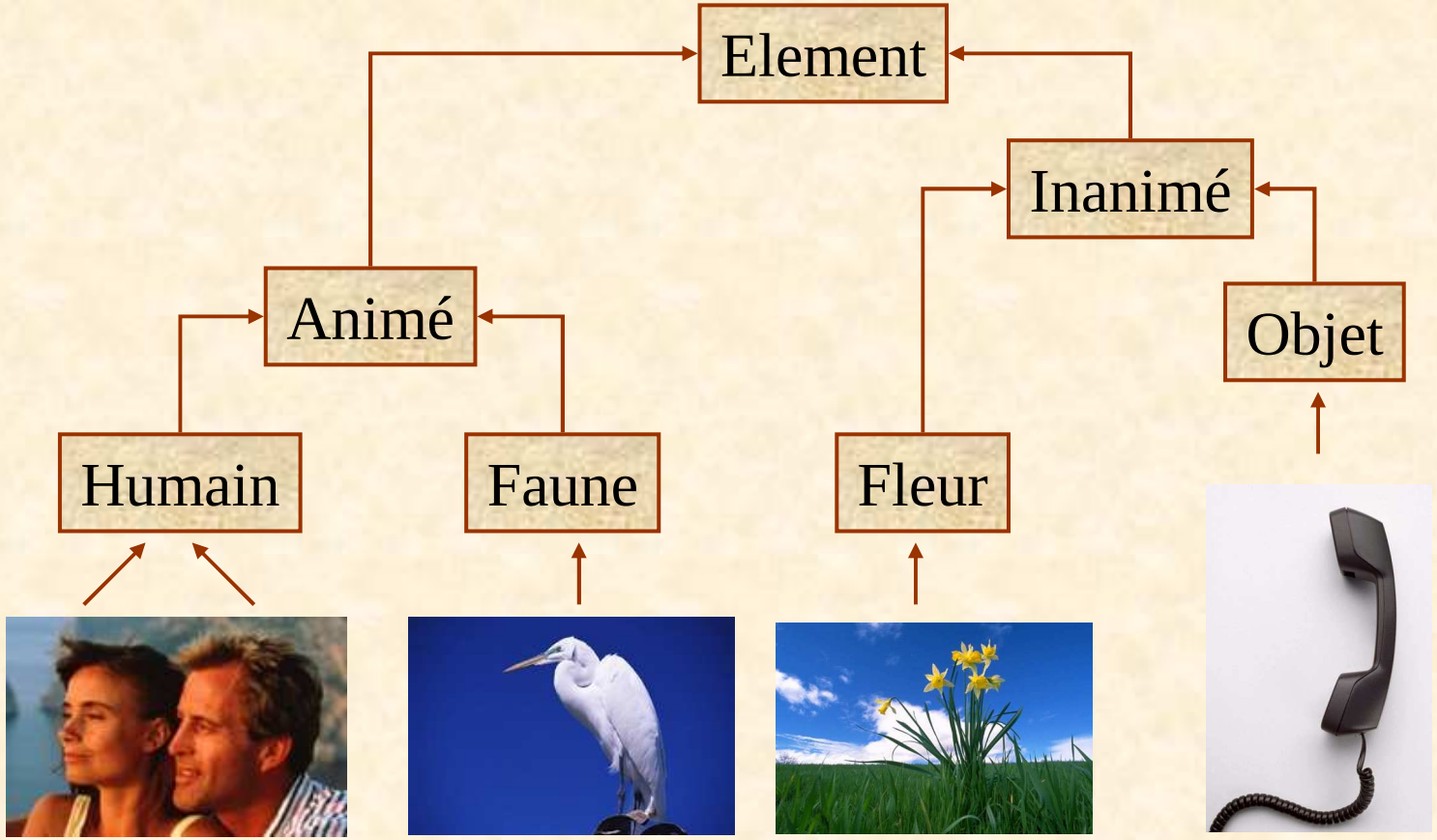
Héritage

- Première solution



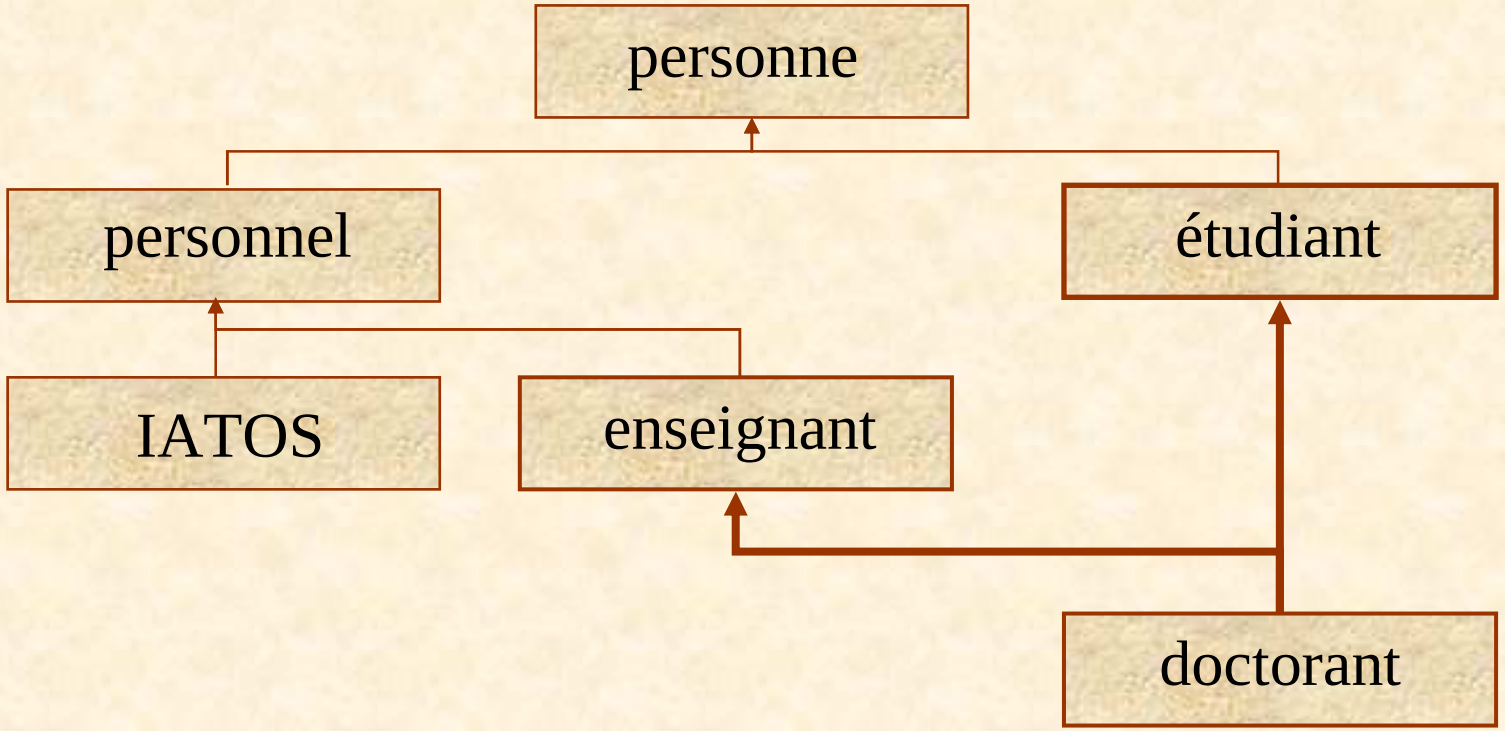
Héritage

- Seconde solution



Héritage multiple

- Définition — Lorsqu'une classe hérite des propriétés de plusieurs super-classes.
- Exemple — BD « UBS »



Héritage multiple

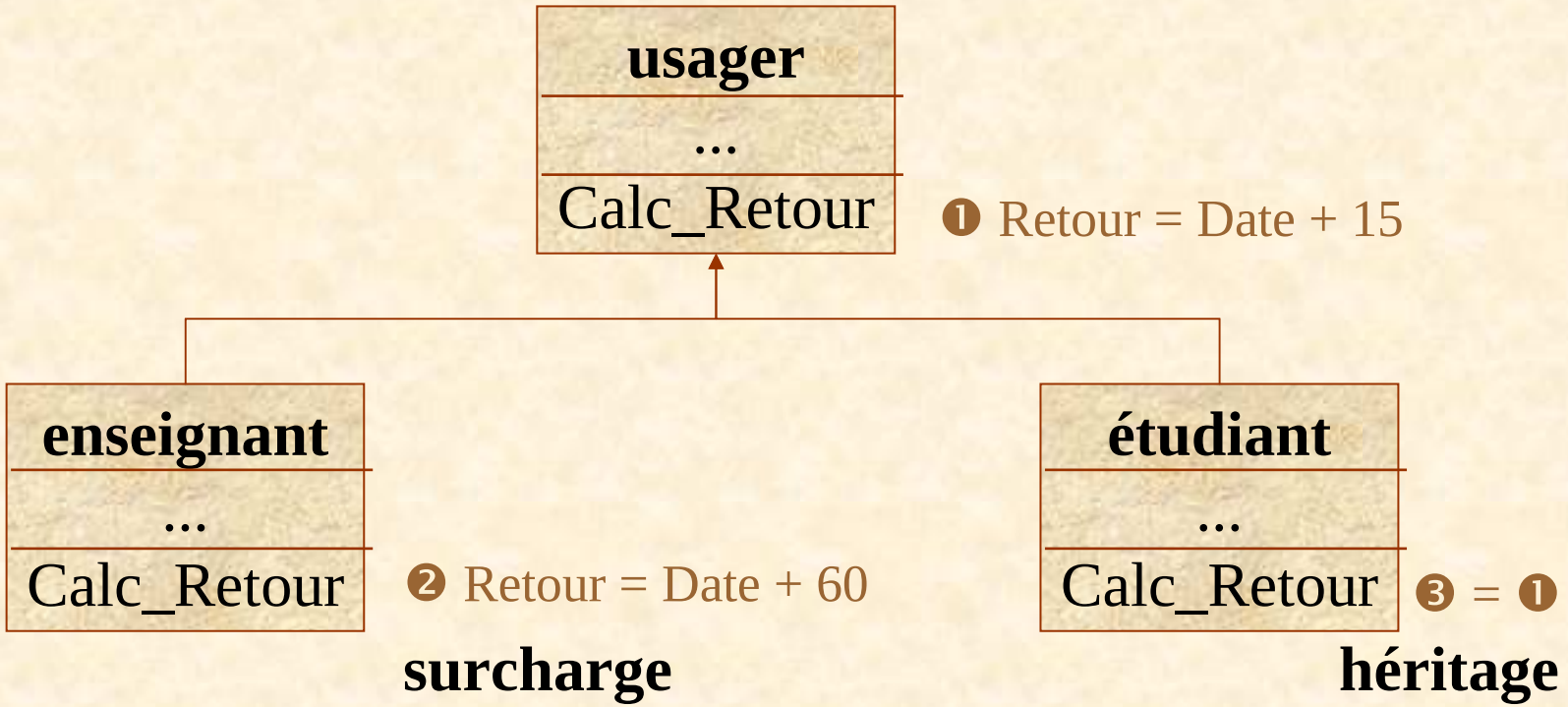
- Conflits entre propriétés héritées
  - Exemple — Filières d'inscription et d'enseignement
  - Solutions — Interdiction de conflits de noms d 'attributs, préfixage de la propriété (analogue préfixage attributs des BD relationnelles), priorités entre classes…
- Héritage multiple inconnu de certains L.O.O.
  - Java...

Polymorphisme

- Surcharge
  - Définition — Redéfinition d ’une méthode héritée pour pouvoir lui donner une implémentation différente
  - Utilité — La méthode garde la même sémantique : spécialisation tout en gardant le même comportement vis à vis de l’extérieur : méthode polymorphe
- Polymorphisme : liaison dynamique
  - Définition — Faculté d’une même méthode à pouvoir s’appliquer différemment suivant la classe
  - Conséquences — Déclenchement de traitements différents suivant le contexte (i.e. classe réceptrice)

Polymorphisme

- Exemple — BD « B.U. UBS »



Polymorphisme

- Avantages
  - Conception — Un seul nom de méthode pour des traitements équivalents mais spécifiques
  - Adaptabilité — Nouveau besoin = nouvelle sous-classe avec surcharge pour adaptation à ce besoin spécifique
- « Limitations »
  - Liaison dynamique — Mise en œuvre plus facile avec un langage interprété (Java, Smalltalk) qu'avec un langage compilé (C++) … mais exécution plus lente.