Processus Systèmes d'Information - UML

➔ Enseignant : Marie Aimar,
aimar@emi.u-bordeaux.fr,

Bibliographie

Les best-of :

Conception orientée objet et applications,
G Booch (Addison-Wesley, 1992).

Le génie logiciel orienté objet,I. Jacobson,
M. Christerson, P. Jonsson, G. Overgaard (Addison-Wesley, 1993).

Les outsiders :

Méthodes orientées objet, 2nd édition,
I Graham (thompson Publisher, 1997).

Analyse des systèmes : de l'approche fonctionnelle à l'approche objet,
Ph. Larvet (InterEditions, 1994).

Spécial UML :

UML La notation unifiée de modélisation objet,
M. Lai (InterEditions, 1997).

Modélisation objet avec UML,
P.A. Muller (Eyrolles, 1998).

Les sites internet

Index to Object-Oriented Information Sources
http: //iamwww.unibe.ch/ scg/OOinfo/index.html

UML Resources
http: //www.rational.com/uml/adobe.html

Aspect-Oriented Programming Home Page
http: //www.parc.xerox.com/csl/projects/aop/

Objectifs du Cours

➔ **Architecture Logicielle** - Approcher les problèmes de l'analyse et de conception des systèmes informatiques (des programmes).

➔ Au travers du **filtre** des méthodes orientées objet - **UML**.

Ce que nous ne ferons pas !

- Traiter des problèmes de l'organisation d'un développement informatique :
  - gestion du temps,
  - gestion des équipes,
  - communication avec les autres entités de l'entreprise.
- Tous les concepts seront abordés uniquement au niveau du *Design* d'une application.

Combattre quelques idées reçues

- Il existe une étape **Analyse** - **Conception** du logiciel différente de l'analyse des besoins d'une entreprise (ou d'un service).

- Ne pas tenter de solutionner tous les problèmes d'un service avec un seul logiciel. Apprendre à cerner précisément le problème à traiter.

- Les méthodes orientées objet ne sont pas faites (et n'ont pas prétendue être faites) pour traiter les problèmes organisationels de management d'une entreprise ou d'un service.

Analyse des systèmes

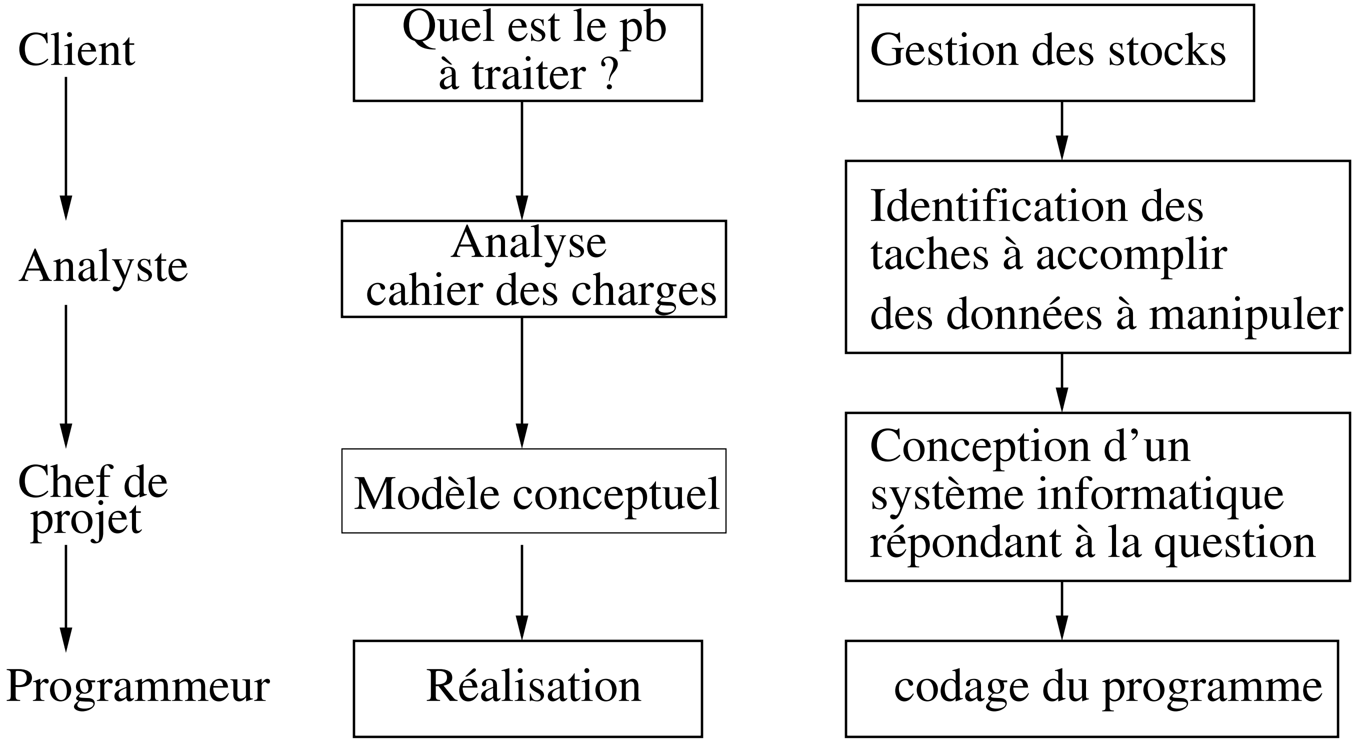
1. Analyser un problème.

2. Proposer un modèle.

3. Concevoir un solution.

4. Réaliser (programmer) la solution.

Schema classique - caricatural



Les erreurs de cette vision des choses

➔ Vous n'interviendrez *"jamais"* sur un projet totalement nouveau

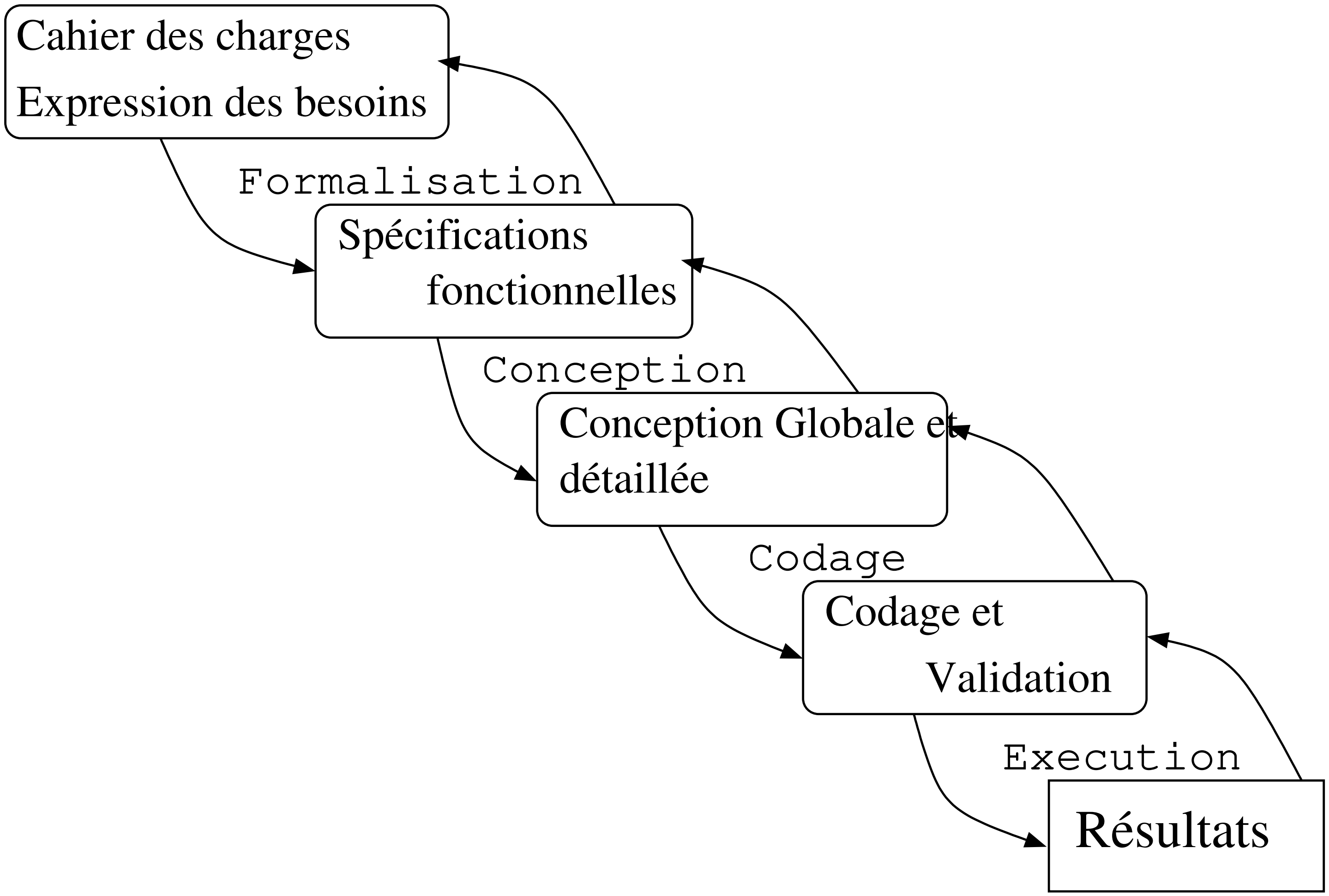
➔ Ignore les relations avec les autres applications de l'entreprise : la gestion des stocks peut communiquer avec la fabrication des marchandises, les commandes des clients . . .

➔ cloisonne les activités : organisation hiérarchique de l'analyste au programmeur, le client n'a pas de contact avec le réalisateur final.

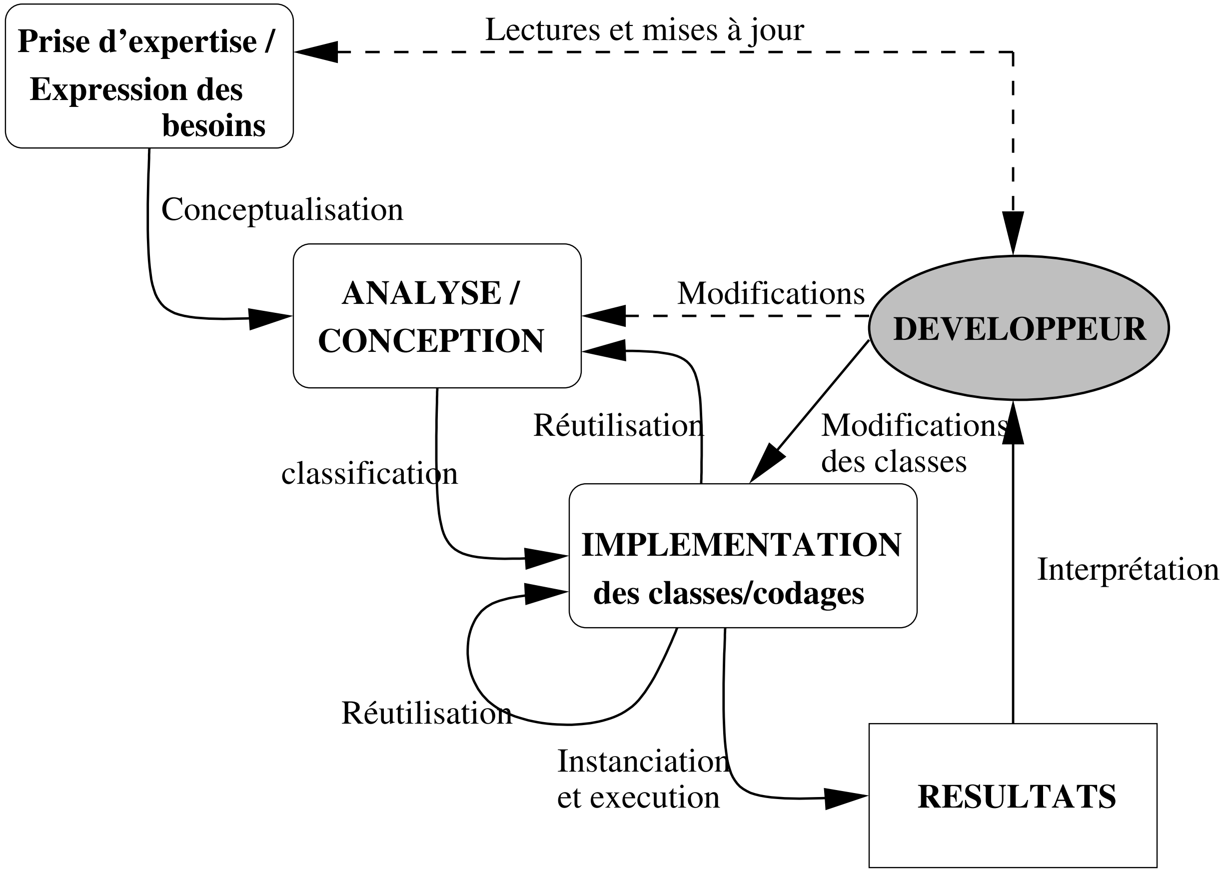
➔ Un client *connait pas* ses besoins.

Un système doit être facile à modifier, à enrichir de nouvelles fonctionnalités, au pire changer d'objectif. L'architecture doit donc être modulaire et facile à maintenir.

Développement en cascade



Développement Orienté Objet



Développement Orienté Objet

➔ Conséquences de l'application des méthodes OO : les phases d'analyse, de conception et de programmation sont très liées.

➔ Historique des méthodes orientées objet :

1. langages de programmation,
2. méthodes de conception,
3. méthode d'analyse.

Quelques repères

- Age de l'invention :
  1967 - le langage de programmation SIMULA.
  1970 - SMALLTALK (Palo Alto).

- Age de la confusion :
  1980 - les langages ++.
  Les méthodes de conception se multiplient

- Age de la maturité:
  1990 - Object Management Group : standardisation.
  Unification des méthodes OMT (Booch) OOSE (Jacobson) et Rumbaugh : Unified Modeling Language (version 1.0 1997).

Les concepts de base

➜ **Objets** : unités de base organisées en classes et partageant des traits communs (attributs ou procédures). Peuvent être des entités du monde réel, des concepts de l'application ou du domaine traité.

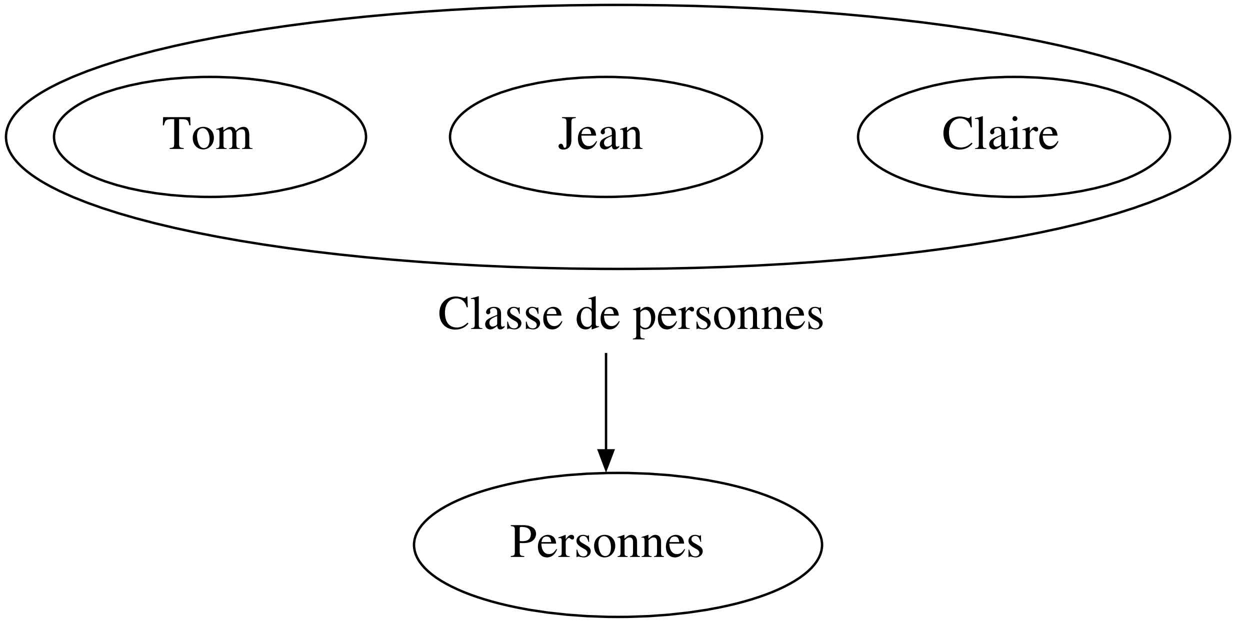
➜ **Encapsulation** : les structures de données et les détails de l'implémentation sont cachés aux autres objets du système. La seule façon d'accéder à l'état d'un objet est de lui envoyer un message qui déclenche l'exécution de l'une de ses méthodes. Les types d'objets peuvent être assimilés aux types de données abstraites en programmation. Abstraction et encapsulation sont complémentaires, l'encapsulation dessant des barrières entre les différentes abstractions.

- ➔ **Héritage** : chaque instance d'une classe d'objet hérite des caractéristiques (attributs et méthodes) de sa classe mais aussi d'une éventuelle super-classe. L'héritage est un des moyens d'organiser le monde c.-à-d. de décrire les liens qui unissent les différents objets.

- ➔ **Polymorphisme** : possibilité de recourir à la même expression pour dénoter différentes opérations. L'héritage est une forme particulière du polymorphisme caractéristique des systèmes orientés objet.

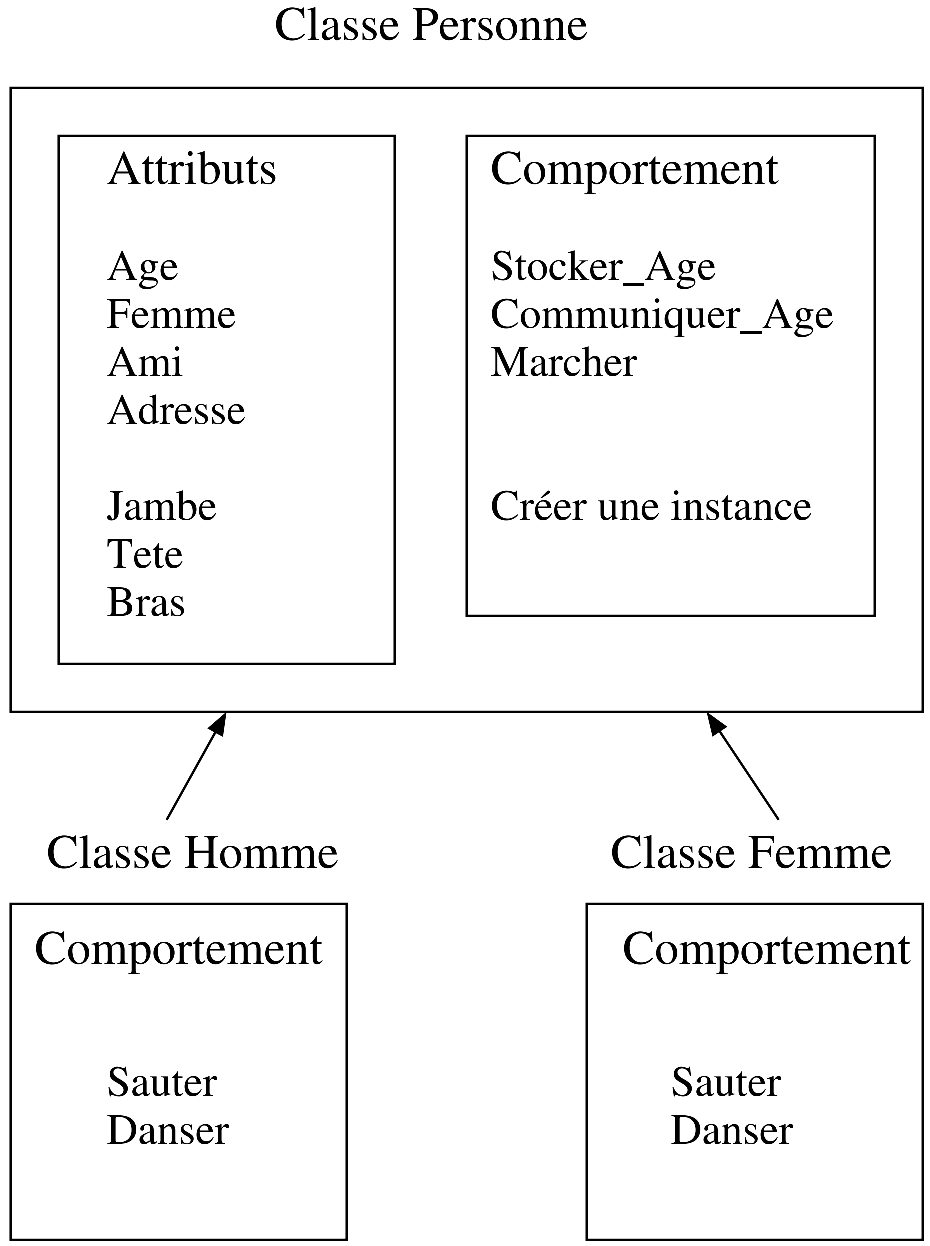
- ➔ **Modularité** : partition du programme qui crée des frontières bien définies (et documentées) à l'intérieur du programme dans l'objectif d'en réduire la complexité (Myers). Le choix d'un bon ensemble de modules pour un problème donné, est presque aussi dificile que le choix d'un bon ensemble d'abstractions.

Faire des choix

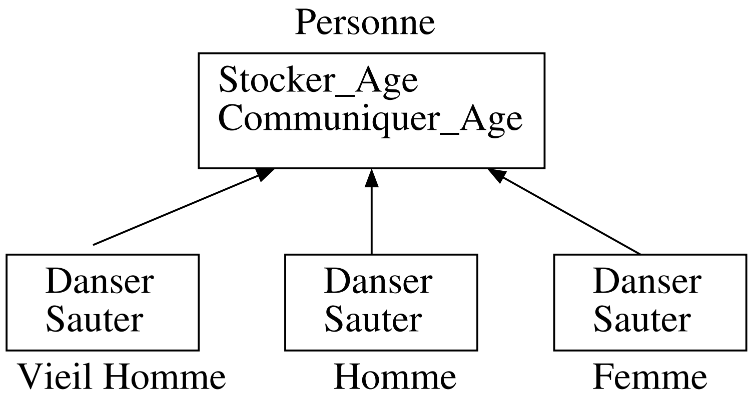


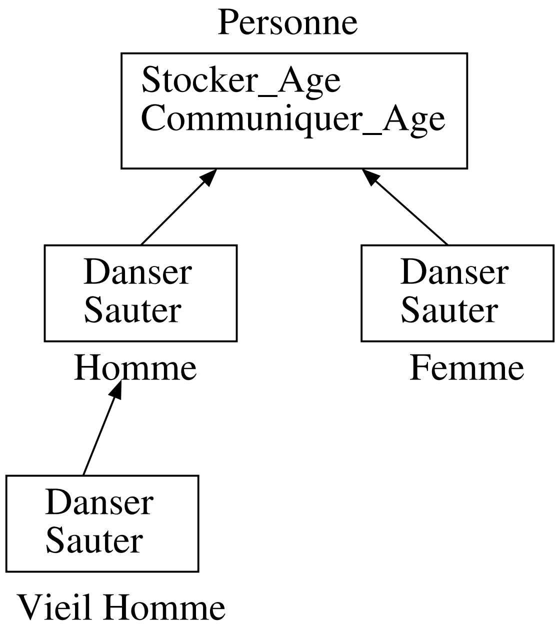
➔ Quelles sont les caractéristiques - attributs - d'une personne?

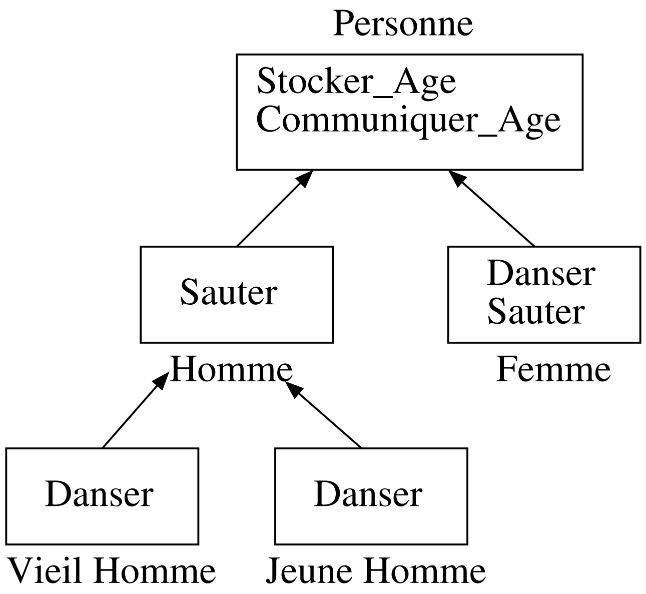
➔ Quels sont les comportements génériques - fonctions - d'une personne?


Classe Personne
Attributs
Age
Femme
Ami
Adresse
Jambe
Tete
Bras
Comportement
Stocker_Age
Communiquer_Age
Marcher
Créer une instance
Classe Homme
Comportement
Sauter
Danser
Classe Femme
Comportement
Sauter
Danser

3 manières d'ajouter une classe Vieil Homme







Trouver les bons objets

- Méthode de désagrégation / agrégation :
  - désagréger un module $\Rightarrow$ une suite de modules,
  - agréger une suite de modules $\Rightarrow$ un module.

Désagrégation

On part d'un tout que l'on éclate en plusieurs parties. Chaque partie, formant à son tour un tout, est susceptible d'être à nouveau éclatée en parties plus petites.

Il est difficile d'exprimer en décomposition logicielle ce qu'est une partie.

La conception fait l'hypothèse que le système est un tout. Pour détailler et exprimer la solution, on postule que ce tout est composé de parties cohérentes séparables.

Dans un premier temps, la décomposition est basée sur les entités du domaine du problème.
La désagrégation est très différente de la décomposition fonctionnelle puisqu'une fonctionnalité n'est pas une entité du monde concret.
La granularité de la taille des entités à utiliser est un facteur important de l'effort d'abstraction à réaliser.

Comment faire trouver les bons Objets?

c.-à-d. : 1) comment trouver un objet?
2) comment distinguer un bon objet d'un mauvais ?

Quelques règles d´ecriture d'un module

- ➔ Un module représente un concept et tout le concept.
- ➔ Pour représenter une idée, il faut que cette idée existe.
- ➔ Ne pas regrouper dans un module des opérations qui n'ont pas de raisons particulières d'être ensemble (écriture de modules fourre-tout).
- ➔ Pour concrétiser une idée le choix du nom du module est un élément puissant d'expression (exemple les design patterns).
- ➔ Dans une première phase "*simpliste*" le choix des méthodes correspond aux verbes.

Un exemple.. enfin !