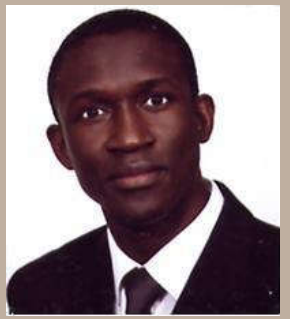
UVS
UNIVERSITE VIRTUELLE DU SENEGAL

ANALYSE ET CONCEPTION ORIENTÉE OBJET

IGL 2331

Baïdy THIONGANE



Présentation de l'enseignant

Baïdy THIONGANE

- Ingénieur Informaticien Diplômé de l'Ecole Polytechnique Fédérale de Lausanne
- Enseignant – Chercheur UVS
- Oracle Certified Master Java EE Enterprise Architect
- Oracle Certified Professional Java SE Programmer

Plan du cours

1. Introduction
2. Le langage de modélisation graphique UML
3. Qu'est-ce que l'Analyse Orientée Objet (AOO)
4. Qu'est-ce que la Conception Orientée Objet (COO)

Objectifs spécifiques

- **Séquence 1** : **Introduction**
  - Objectifs spécifiques :
    1. Expliquer les enjeux de l'Analyse et de la Conception Orientée Objet
    2. Etablir la frontière entre l'Analyse et la Conception Orientée Objet
    3. Expliquer un processus de développement logiciel
    4. Examiner les concepts de base de l'Orientée Objet

- **Séquence 2** : **Le langage de modélisation UML**
  - Objectifs spécifiques :
    1. Expliquer le langage de modélisation graphique UML
    2. Décrire les diagrammes d'UML
    3. Utiliser le langage de modélisation graphique UML

Objectifs spécifiques

- **Séquence 3 : Qu'est-ce que l'Analyse Orientée Objet (AOO)?**
  - Objectifs spécifiques :
    1. Expliquer ce que recouvre le terme Analyse Orientée Objet
    2. Identifier les classes d'analyse avec leurs attributs et associations
    3. Développer des cas d'utilisation complets en utilisant la notation UML
    4. Mettre en œuvre les concepts objets pour l'analyse d'un problème

- **Séquence 4 : Qu'est-ce que la Conception Orientée Objet (COO)?**
  - Objectifs spécifiques :
    1. Décrire les objectifs de la COO
    2. Définir les concepts fondamentaux et donner la terminologie de la technologie OO
    3. Produire des modèles de conception en utilisant la notation UML
    4. Définir les contrats des opérations système

Références

- Object-Oriented Analysis And Design – Second Edition
  - *Auteur*: Booch, Grady
  - *Editeur*: Addison-Wesley, Santa Clara, California
  - *Publication*: 1994

- UML 2 par la pratique : Etudes de cas et exercices corrigés – Edition 8
  - *Auteur*: Roques, Pascal
  - *Editeur*: Eyrolles
  - *Publication*: 2018
  - *Disponible*: ScholarVox

Consigne de travail

Consigne

- Lire et comprendre les séquences du cours
- Reprendre les exemples du cours
- Faire les tests de connaissances
- Faire les travaux dirigés et les rendre dans les délais
- Participer dans les forums
- N'hésiter pas à solliciter les tuteurs

UVS
UNIVERSITE VIRTUELLE DU SENEGAL

Analyse et Conception Orientée Objet

INTRODUCTION

La crise du logiciel

*« Alors que le matériel informatique a fait, et continue de faire, des progrès très rapides, le logiciel, l'autre ingrédient de l'informatique, traverse une véritable crise. Etant donné que cette crise a été décelée en 1969 déjà et qu'elle dure toujours, il serait plus approprié de parler d'**une maladie chronique**.»*

Le non-respect des coûts et des délais

Le coût de développement d'un logiciel est presque impossible à prévoir et le délai de livraison n'est que rarement respecté. On cite ainsi des **dépassements moyens du coût budgété et du délai prévu respectivement de 70% et 50%.**

Une qualité déficiente

La qualité du logiciel livré est souvent déficiente. Le produit ne satisfait pas les besoins de l'utilisateur, **il consomme plus de ressources que prévu et il est à l'origine de pannes**.

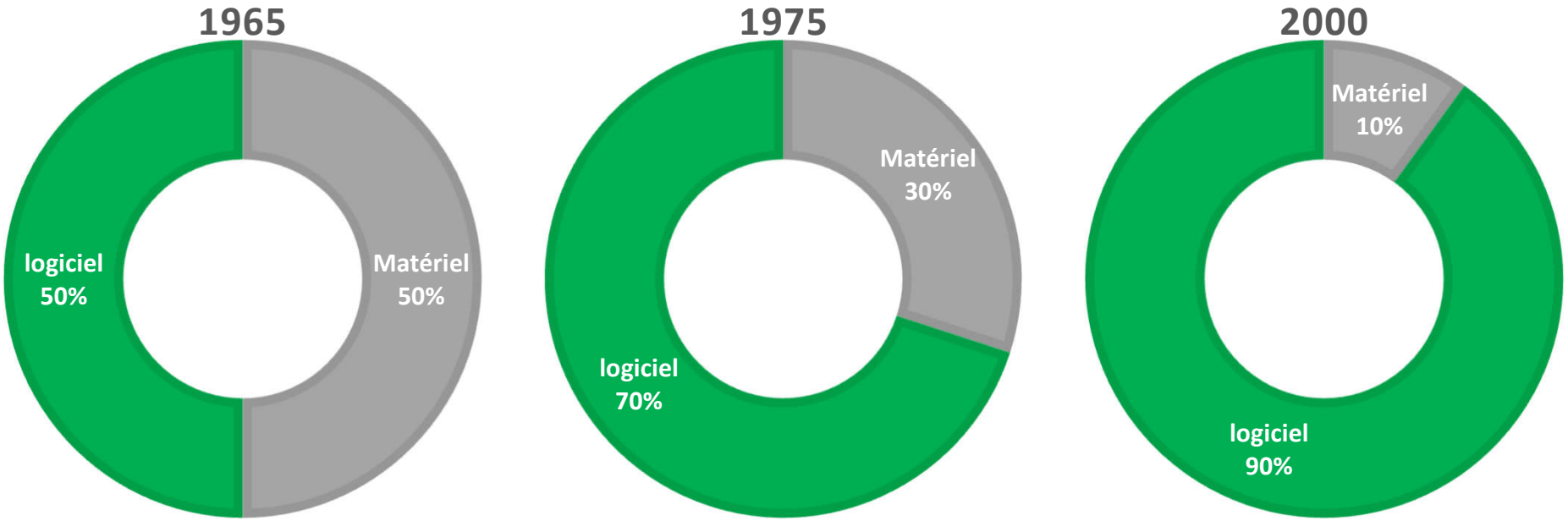
Une maintenance difficile

La maintenance du logiciel est difficile, coûteuse et souvent à l'origine de nouvelles erreurs. Mais en pratique, il est indispensable d'adapter les logiciels car leurs environnements d'utilisation changent et les besoins des utilisateurs évoluent.

Une réutilisation quasi-impossible

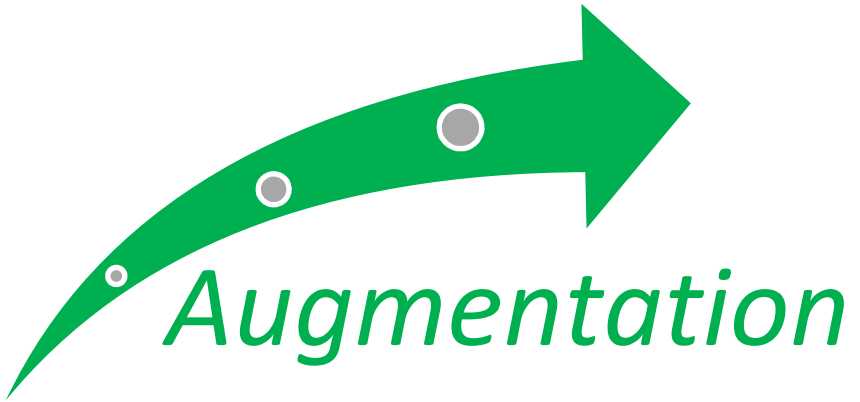
Il est rare qu'on puisse réutiliser un logiciel existant ou un de ses composants pour confectionner un nouveau système, même si celui-ci comporte des fonctions similaires. **Tout amortissement sur plusieurs projets est ainsi rendu impossible**.

Coût d'un système informatique



Augmentation des Coût d'un SI

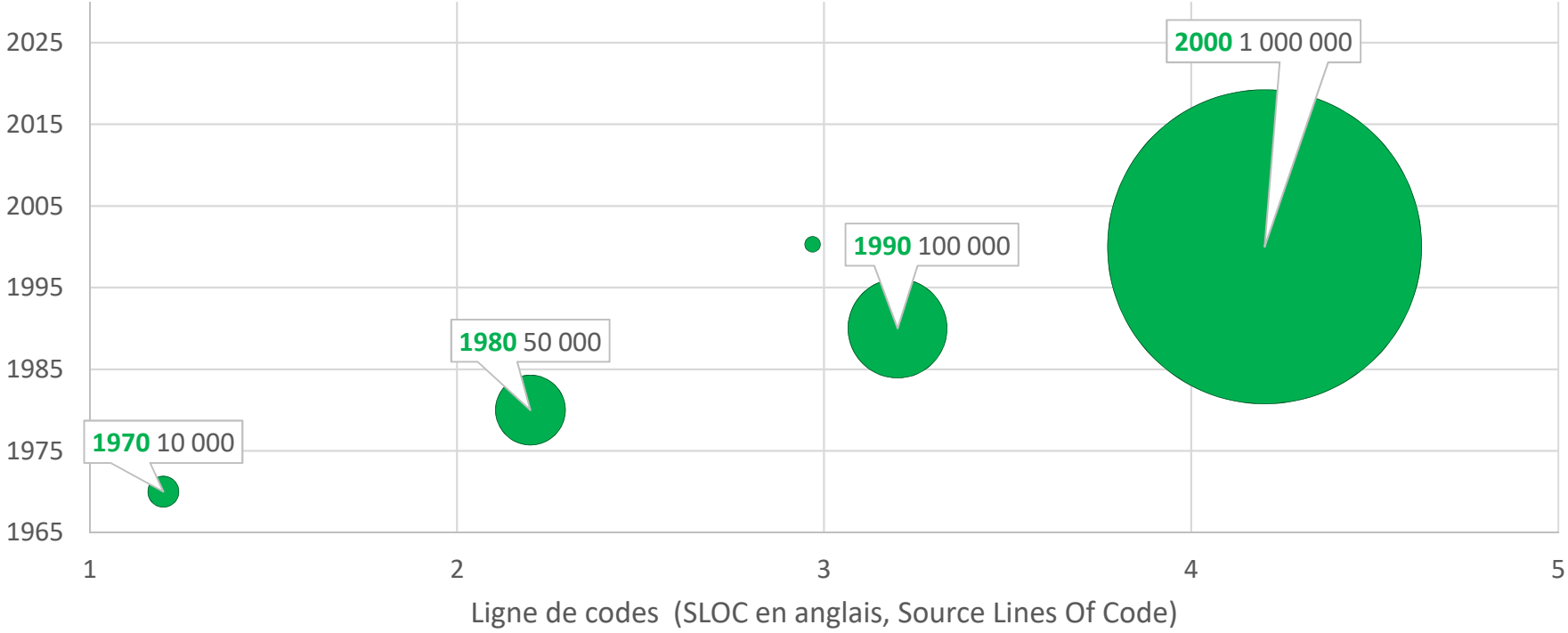
cette augmentation des coûts d'un Système Informatique (SI) est surtout due aux **frais de maintenance**, situés actuellement entre **40% et 75% du coût total d'un système informatique (matériel et logiciel).** Or ces frais de maintenance résultent avant tout de la **qualité déficiente** du logiciel, au moment de sa livraison déjà ou après qu'il ait évolué.



La raison de fond

Le fait étant qu'il est beaucoup plus difficile de créer des logiciels que le suggère **notre intuition**. Comme les solutions informatiques sont essentiellement constituées de composants immatériels, tels des programmes, des données à traiter, des procédures et de la documentation, on sous-estime facilement leur complexité.

La complexité



Evolution du nombre de ligne de code en moyenne dans les logiciels

La réalité

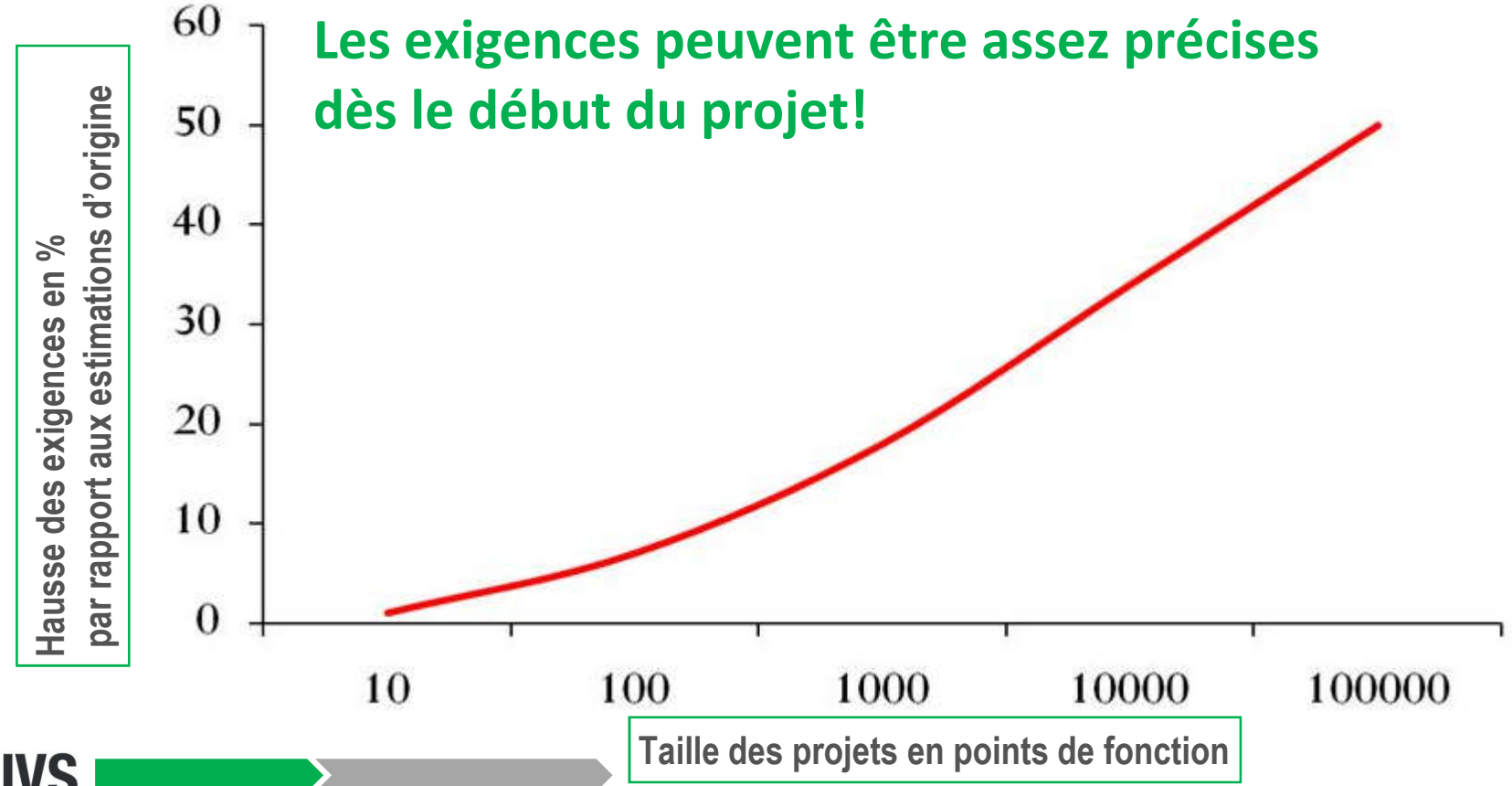
Logiciel	Lignes de codes
Word, Excel, Navigateur	5.000.000
Windows 7 ou Windows XP	40.000.000
Facebook	50.000.000
Voiture	100.000.000
Services Google	2.000.000.000
Génome humaine	3.300.000.000

La solution

Pour maîtriser la complexité des systèmes logiciels, il convient de procéder selon **une démarche** bien définie, de se baser sur **des principes et méthodes**, et d'utiliser **des outils performants**. On arrive ainsi **à gérer des projets complexes et à élaborer scientifiquement des solutions**.

Affirmation erronée n°1

Les exigences peuvent être assez précises dès le début du projet!



Source: *Applied Software Measurement, Capers Jones, 1997.*

Affirmation erronée n°2

- **Les exigences sont stables!**
  - Le marché bouge en permanence
  - La technologie évolue
  - Les besoins des clients et utilisateurs changent
  - Les objectifs des intervenants changent

Affirmation erronée n°3

- **La conception peut être terminée…avant que ne débute la programmation**
  - Demandez à un programmeur…
  - Encore trop de variables, d'inconnues et de nouveautés
  - Une spécification complète doit être aussi détaillée que le code lui-même
  - Il est beaucoup plus difficile de créer des logiciels que le suggère notre intuition.

En conséquence…

Processus de développement logiciel

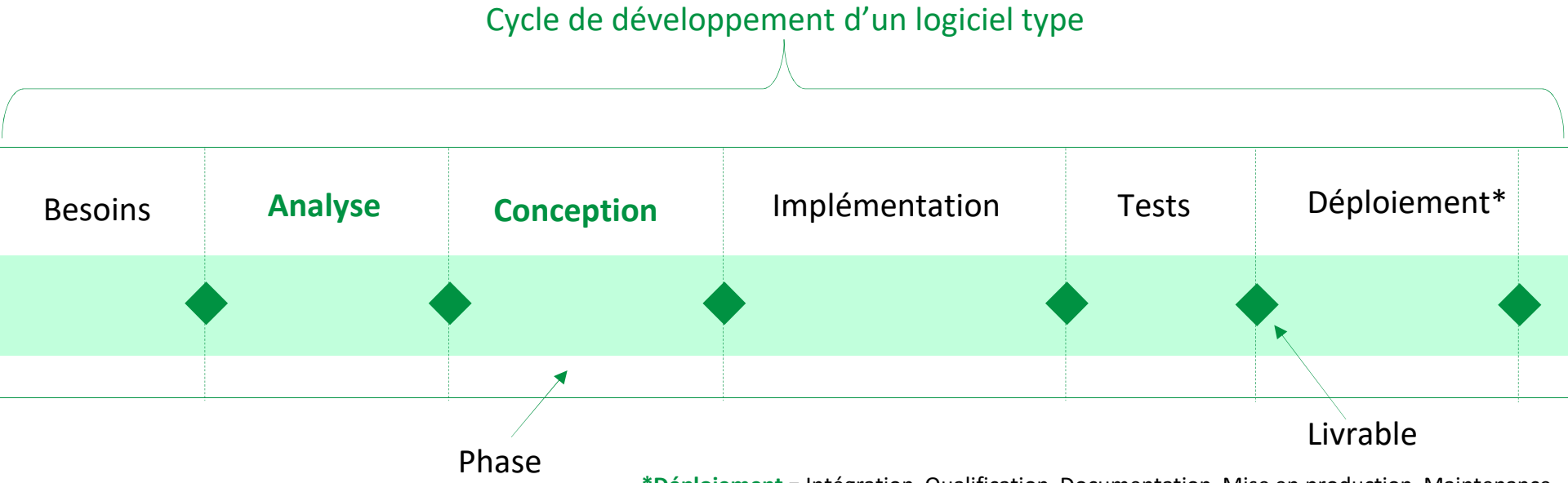
- **Ceci n'est pas un cours sur un processus de développement logiciel!** Toutefois, il est difficile d'explorer les notions d'analyse et de conception sans évoquer la façon dont leurs différentes étapes s'articulent entre elles.

- Un processus de développement logiciel sont les étapes de l'organisation et de la gestion du développement logiciel.

- Un processus de développement logiciel est constitué par un ensemble de cycles de développement.

Cycle de développement

Ainsi il est possible de schématiser le cycle de développement d'un logiciel type selon la structure de cycle de développement suivante:



***Déploiement** = Intégration, Qualification, Documentation, Mise en production, Maintenance

Modèle de cycle de développement

Afin d'être en mesure d'avoir une méthodologie commune entre le client et la société de service réalisant le développement, des modèles de cycles de vie ont été mis au point définissant les étapes du développement ainsi que les documents à produire (livrables) permettant de valider chacune des étapes avant de passer à la suivante. A la fin de chaque phase, des revues sont organisées.

L'Orientée Objet - OO

L'**Orientée Objet** cherche ainsi donc à établir et à utiliser des principes sains d'**ingénierie** dans le but de **développer économiquement du logiciel** qui est **fiable** et qui **fonctionne** efficacement sur des machines réelles.

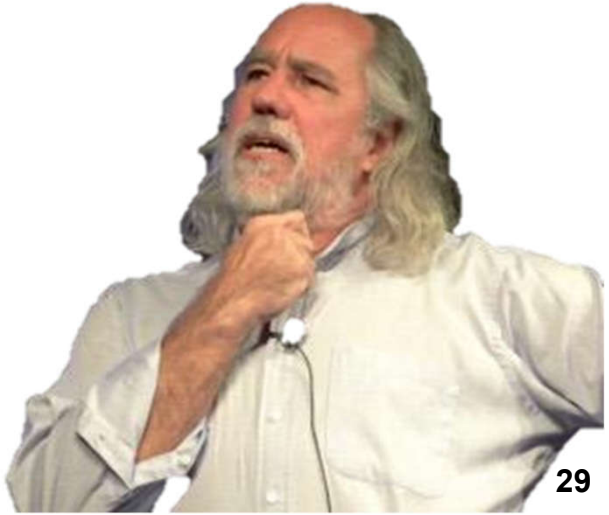
Les objectifs de l'Orienée Objet

- Faciliter le **développement**, la **maintenance** et l'**évolution** des applications ;
- Permettre la **modularité** du code ;
- Permettre la **réutilisation** du code;
- Augmenter la **qualité** des logiciels;
- Diminuer le **coût** de développement d'un logiciel;
- Respecter les **délais** de livraison.

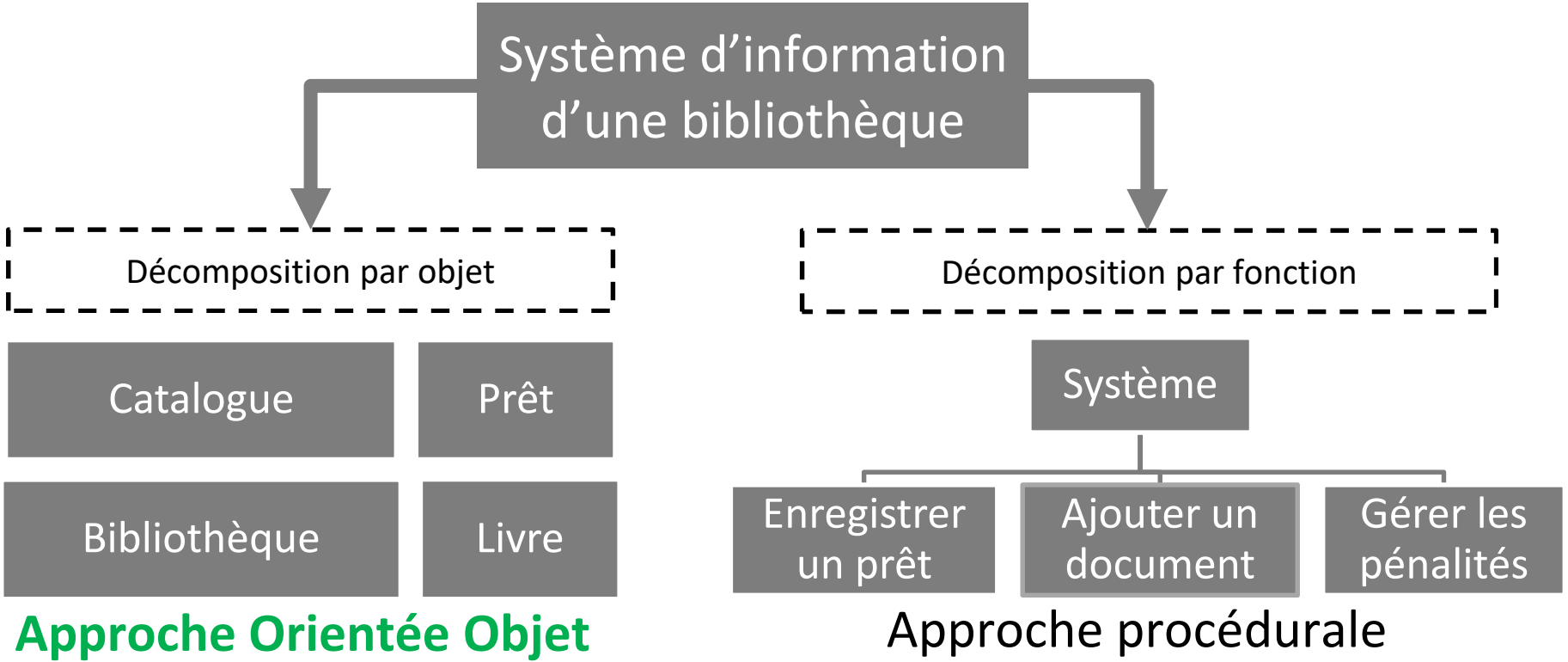
Fondements de l'Orientée Objet

«L'ingénieur comme l'artiste doit avoir une connaissance approfondie des matériaux de sa discipline. Quand on utilise des méthodes orientées objets pour analyser ou concevoir un système logiciel complexe, les éléments de construction fondamentaux sont les **classes** et les **objets**.»

Grady BOOCH
Co-créateur d'UML



L'approche orientée objet

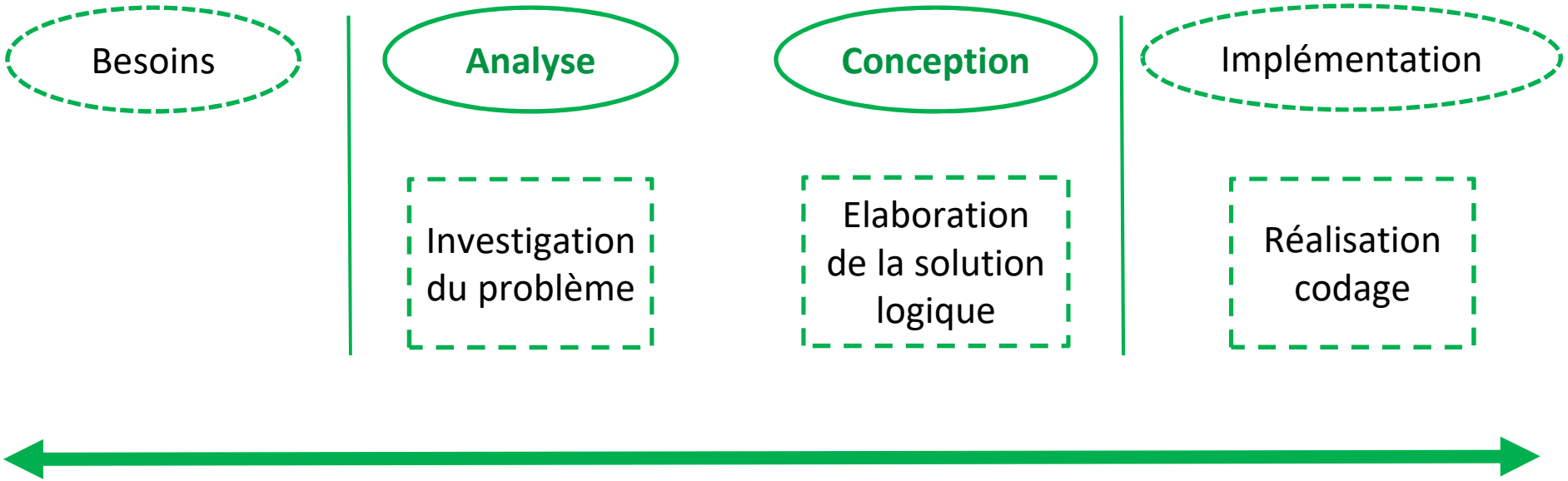


Orientation Objet et Orientation Procédurale

PROCÉDURALE	OBJET
Que doit faire mon programme ?	De quoi doit être composé mon programme ?
Le programme informatique est divisé en petites parties appelées **fonctions**	Le programme informatique est divisé en petites parties appelées **objets**
Conception du haut vers le bas	Conception des interactions entre les objets
Réutilisation limitée du code	Réutilisation du code
Code complexe	Conception complexe
Centrée sur le partage des données	Centrée sur la protection des données
Fondée sur la notion de traitement	Fondée sur la notion de service
Langages: C, VB, Pascal, Fortan	Langages: C++, Java, VB.NET, C#.NET
Moins lisible à la longue	Devient plus lisible
Pas besoin d'aborder le polymorphisme, l'héritage, etc	Nouvelles notions et utilisation du procédural
Le code sera moins bien compris par d'autres	Séparations évidentes

L'Analyse et la Conception Orientée Objet…

Phases clés du cycle de développement d'un logiciel



L'Analyse et la Conception Orientée Objet

- **Analyse orientée-objet**
  - Investigation du problème en termes d'objets du domaine
- **Conception orientée-objet**
  - Élaboration d'une solution en termes d'objets logiciels inter-opérants
- **Programmation orientée-objet**
  - Codage avec un langage de programmation orienté objet

L'Analyse et la Conception Orientée Objet

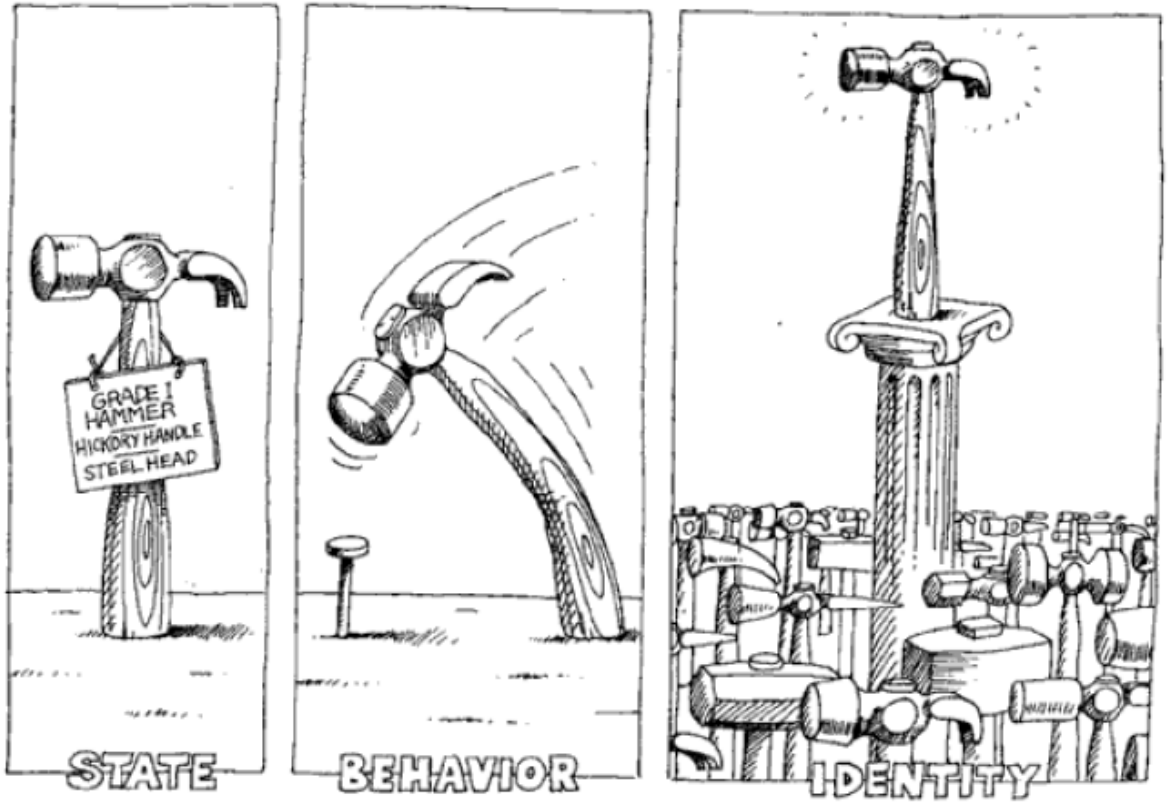
- Historiquement, la programmation orientée-objet est apparue en premier (Simula : 1968, Smalltalk : 1970,etc.). Puis les concepts objets ont remonté les niveaux d'abstraction, d'abord vers la conception, puis l'analyse au début des années 90.

- Dans le cadre de ce cours nous mettrons un focus sur l'Analyse Orientée Objet (AOO) et la Conception Orientée Objet (COO) en partant des besoins du client.

Les 7 concepts de base de l'Orienté Objet

1. Objet
2. Abstraction
3. Composition et réutilisation des objets
4. Classe
5. Classification et héritage
6. Encapsulation
7. Polymorphisme

An object has state, exhibits some well-defined behavior, and has a unique identity.



1. Objet

- Objet = entité aux frontières bien définies, possédant une **identité**, encapsulant un **état** et un **comportement**.

- Objet = entité logicielle regroupant un ensemble des données et de méthodes de traitements

- Objet = unité logique de la programmation orientée objet.

1. Objet - Etat

Il est défini à l'aide d'**attributs** (appelés aussi *données membres* ou *propriétés*). Ce sont des variables associées à l'objet et qui stockent des valeurs (des informations sur l'état de l'objet).

Attributs :
marque = Nissan
modèle = Qashqai
Couleur = grise
vitesse = 0 km/h

Etat : A l'arrêt

Attributs :
marque = Airbus
modèle = A380
Altitude = 8000 m
vitesse = 900 km/h

Etat : En vol

1. Objet - Comportement

- il est caractérisée par des méthodes (appelées parfois fonctions membres ou opérations) qui permettent de modifier l'état de l'objet.

- Une **méthode** rattachée à l'objet permet de déclencher un des comportements associés à l'objet.

- Il n'est pas possible d'agir directement sur les données d'un objet pour modifier son état; il est nécessaire de passer par ses méthodes. On dit qu'on « envoie un message » (faire une requête) à un objet.

- Un objet peut recevoir un message qui déclenche
  - une méthode qui modifie son état et/ou …..
  - une méthode qui envoie un message à un autre objet

1. Objet - Comportement

Attributs :
marque = Nissan
modèle = Qashqai
couleur = grise
vitesse = 95 km/h

Méthodes:
démarrer
accélérer
freiner
éteindre

Etat : Roule

Attributs :
marque = Airbus
modèle = A380
altitude = 8000 m
vitesse = 900 km/h

Méthodes:
décoller
atterrir
virer

Etat : En vol

1. Objet - Identité

L'objet possède une identité, qui permet de le distinguer des autres objets, indépendamment de son état.

Exemple:
Numéro de châssis
Plaque d'immatriculation
Numéro de la Carte Nationale d'Identité

1. Objet - Résumé

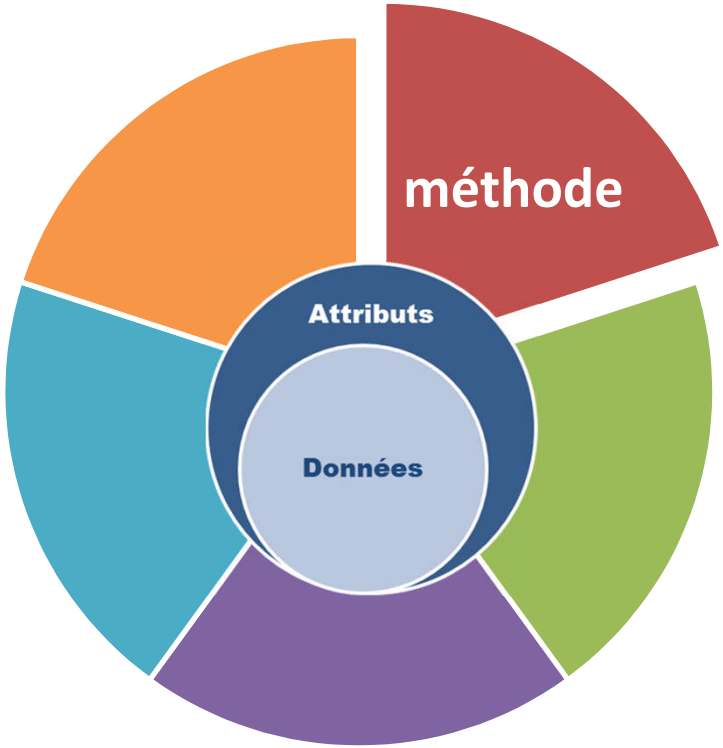
- Un **objet** est une variable améliorée: il stocke les données, mais on peut effectuer des requêtes sur cet objet (on demande à l'objet de faire des opérations sur lui-même), en envoyant un message à cet objet. Ceci est équivalent à un appel de fonction.

- Il possède une structure interne et un comportement.

- Méthode =
  - Un comportement d'un objet invoqué par un message
  - Semblable à une fonction, mais appartenant à un objet

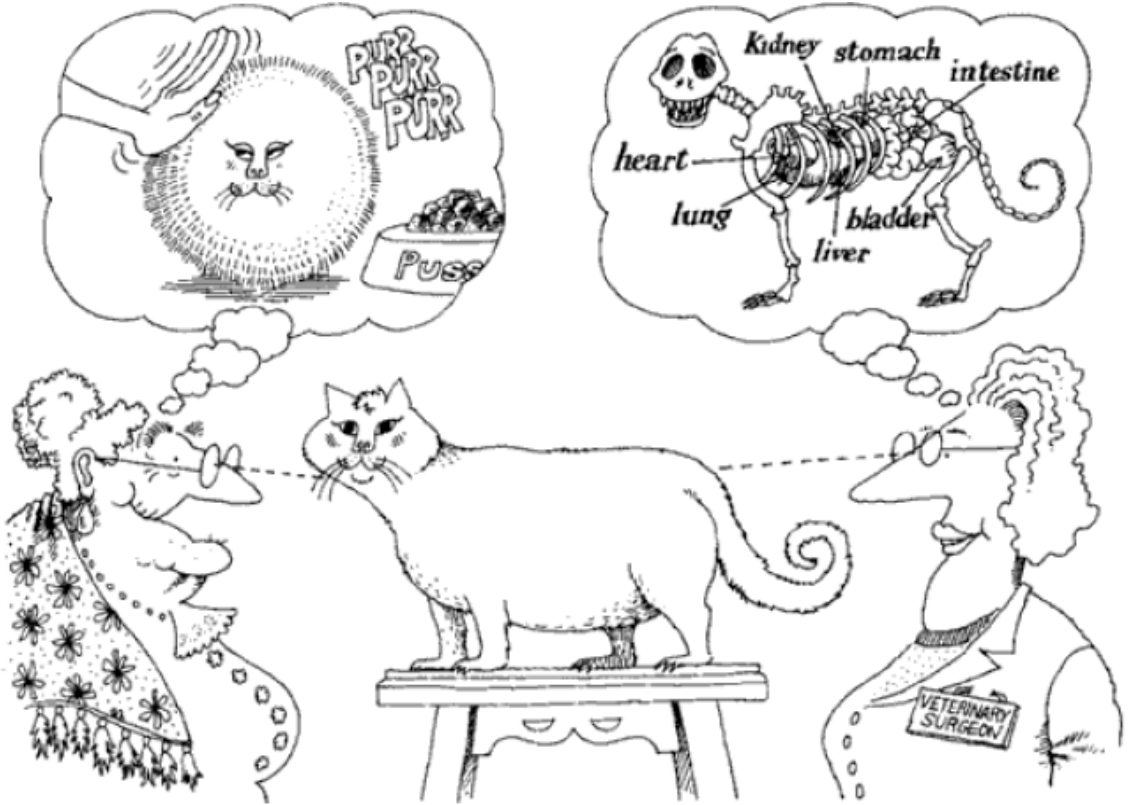
- L'exécution d'un système objet repose sur les objets et les liens entre objets.

1. Objet



- Etat = Attributs
- Comportement = Méthodes
- Identité = Données

Abstraction focuses upon the essential characteristics of some object, relative to the perspective of the viewer.



2. Abstraction

Un objet est une abstraction

- Seules les propriétés pertinentes pour le modèle sont représentées
- Un objet peut être une abstraction pour une entité :
  - matérielle - un avion, une voiture, une personne
  - immatérielle - une commande, un compte
  - interface homme-machine - le bouton « OK »
  - informatique - une liste chaînée
  - réseau – router, adresse IP

Modularity packages abstractions into discrete units.

3. Composition et réutilisation des objets

Composer ou créer un objet à partir d'autres objets est le cœur de la programmation orientée objet. Cela permet:

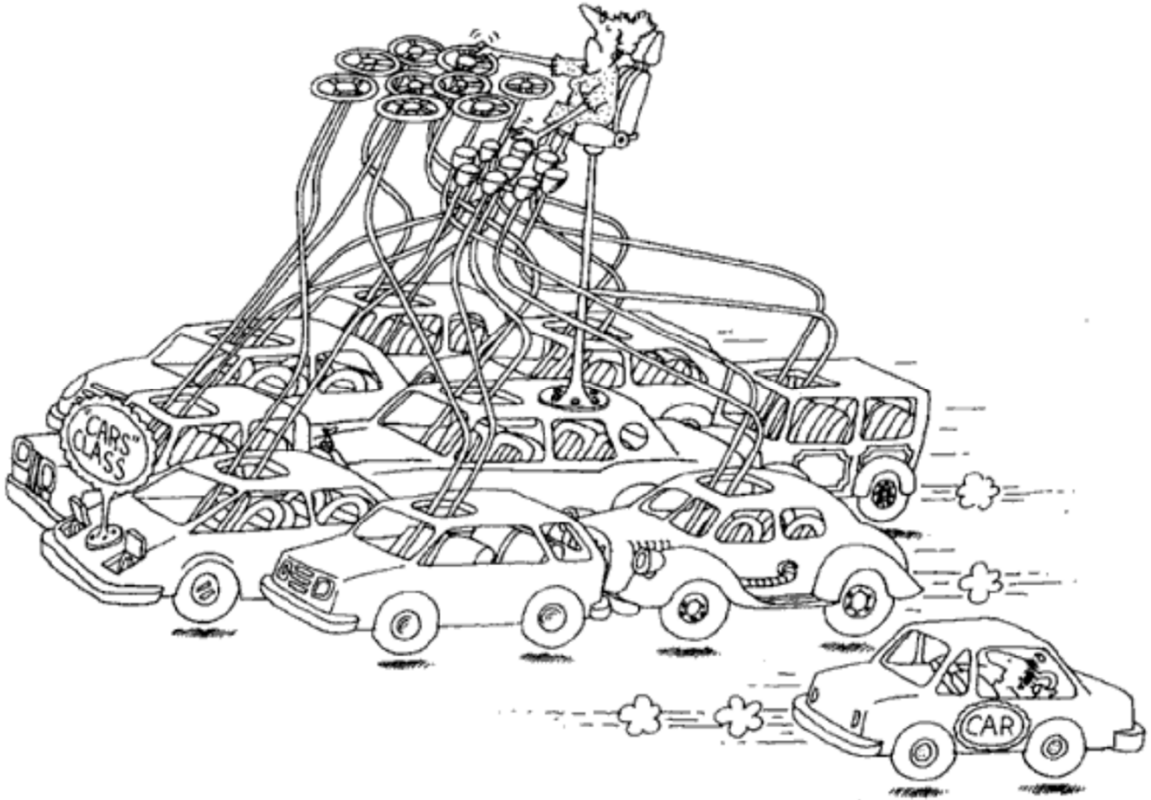
- La **construction** d'objets plus complexes en partant de blocs de constructions plus simples
    - Par exemple: Un programme de dessin; il se composera d'objets tels que des fenêtres, des menus, un cadre de dessin, une palette d'outils, une palette de couleur, etc…L'application de dessin est construite en ressemblant et en créant ses classes de composants et en les regroupant en un ensemble intégré

3. Composition et réutilisation des objets

- La **simplification** de l'organisation des programmes
  - Par exemple on peut utiliser la bibliothèque de base de java (Java JDK) pour assembler rapidement et simplement les pièces d'un programme.

- La **réutilisation** des logiciels développés
  - Par exemple: on peut utiliser les classes de l'application de dessin pour une application de publication.
- L'agrégation spécifie une relation « tout - partie »
- La composition est une agrégation forte
  - Une partie n'appartient qu'à un seul composite à un moment donné
  - Si l'agrégat composite est détruit, alors toutes ses parties le sont aussi

A class represents a set of objects that share a common structure and a common behavior.



4. Classe

- Une classe est un descripteur d'objets
  - Un objet est une instance de classe
- Une classe (ou type d'objets) représente une famille d'objets qui partagent des propriétés communes:
  - Les objets qui ont les mêmes états et les mêmes comportements.
- Une classe regroupe les objets qui ont :
  - La même structure (même ensemble d'attributs)
  - Le même comportement (même méthodes)
- Les classes servent pour la création des objets
  - Un objet est une instance d'une classe

4. Classe

- Un **programme** OO est constitué de classes qui permettent de créer des objets qui s'envoient des messages.
- L'ensemble des **interactions** entre les objets défini un algorithme.
- Les **relations** entre les classes reflètent la décomposition du programme.

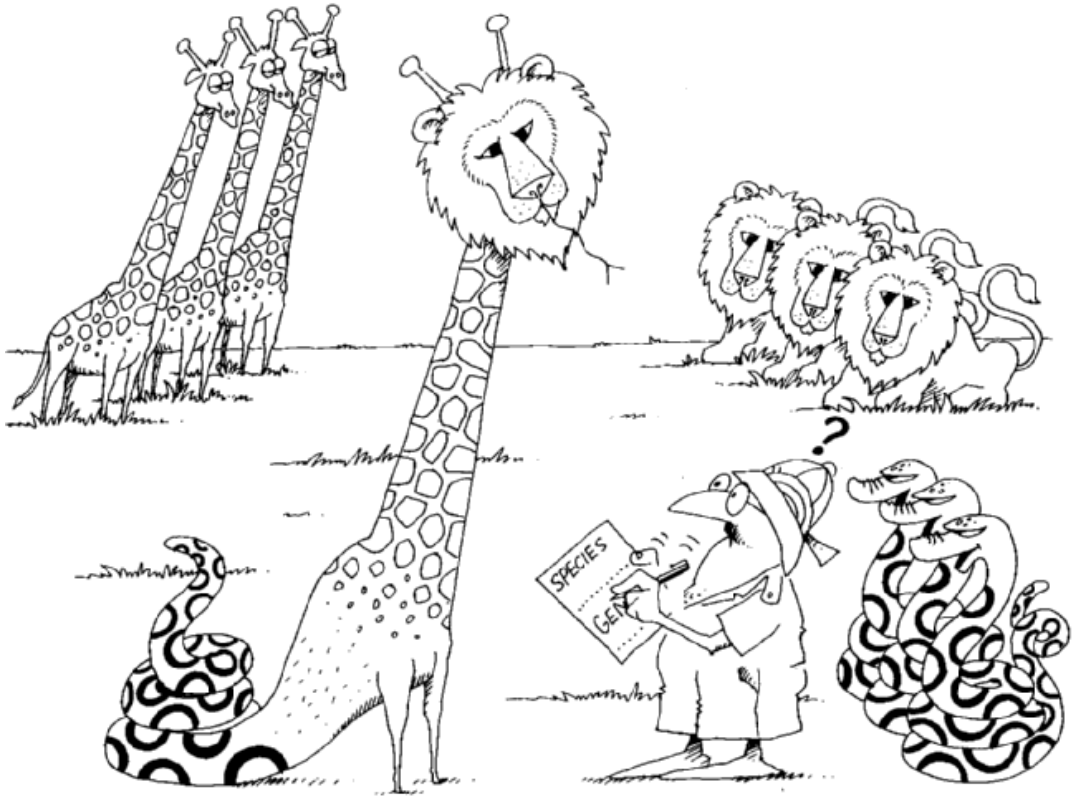
4. Classe

- **Champs** (appelés aussi attributs ou données membres ou propriétés). L'ensemble des champs définissent l'état d'un objet (chaque objet a ses données propres).

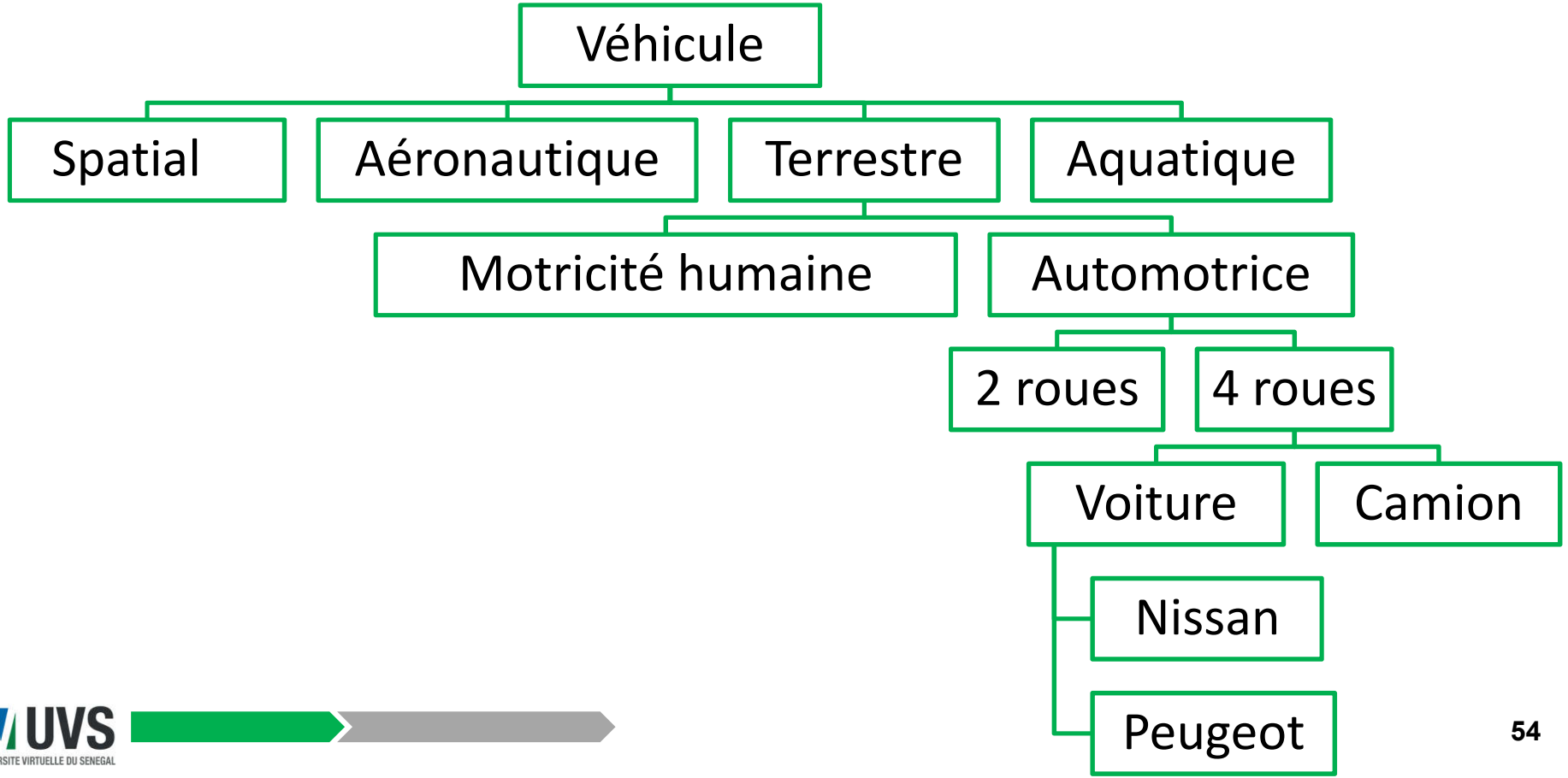
- **Méthodes** (appelés aussi fonctions membres ou comportement). Les méthodes définissent un ensemble d'opérations applicables à l'objet (On manipule les objets par des appels de ses méthodes).

- L'ensemble des méthodes est appelé l'**interface** de l'objet.
  - Une interface définit toutes les opérations qu'on peut appliquer à l'objet (définit tous ce qu'il est possible de "faire" avec un objet).

Classification is the means whereby we order knowledge.



4. Arbre de classification



4. Définition d'une classe

Une classe bien structurée:

- fournit une abstraction claire de quelque chose tiré du vocabulaire du domaine du problème ou de celui de la solution;
- comprend un petit ensemble de responsabilités bien définies et les réalise parfaitement;
- fournit une séparation nette entre les spécifications de l'abstraction et son implémentation;
- est compréhensible et simple tout en étant extensible et adaptable.

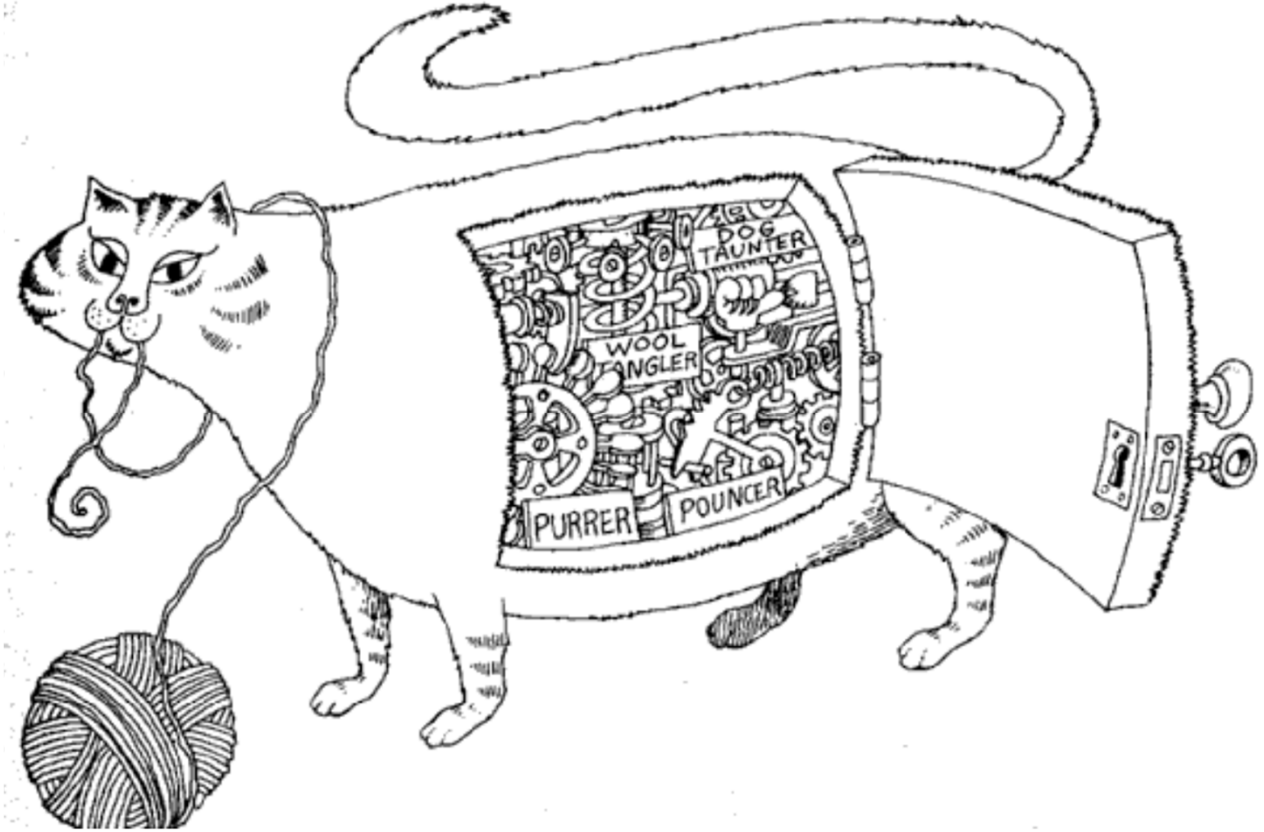
5. Classification et héritage

- La réutilisation des objets ne se limite pas à leur composition. Elle exploite aussi une capacité puissante de la programmation orientée objet connue sous le nom d'**héritage**. L'héritage permet non seulement aux objets d'être utilisés en tant que tels, mais il autorise aussi la création de nouveaux objets obtenus par extension et façonnage d'objets existants.
- La **classification** est une méthode générale d'organisation du savoir et elle est utilisé ailleurs que dans la programmation orientée objet.
- L'héritage est le mécanisme de transmission des caractéristiques d'une sur-classe vers ses sous-classes.
- Quand une classe en étend une autre, elle hérite de ses données et de ses méthodes. On parle d'héritage simple. Si elle étend plusieurs classes on parle d'**héritage multiple.**
- Une classe hérite des attributs, des méthodes et des associations de toutes ses sur-classes.
- L'héritage évite la duplication, facilite la réutilisation et l'évolutivité

5. Classification et héritage

- **Classification = généralisation + spécialisation**
  - **Généralisation** : regroupement des caractéristiques communes à un ensemble de classes au sein d'une surclasse plus générale.
  - **Spécialisation** : ajout de caractéristiques spécifiques dans une sous-classe, ou adaptation des caractéristiques transmises.
  - Une instance d'une sous-classe est également instance de toutes ses sur-classes.
  - La classification correspond à un "OU" logique.

Encapsulation hides the details of the implementation of an object.

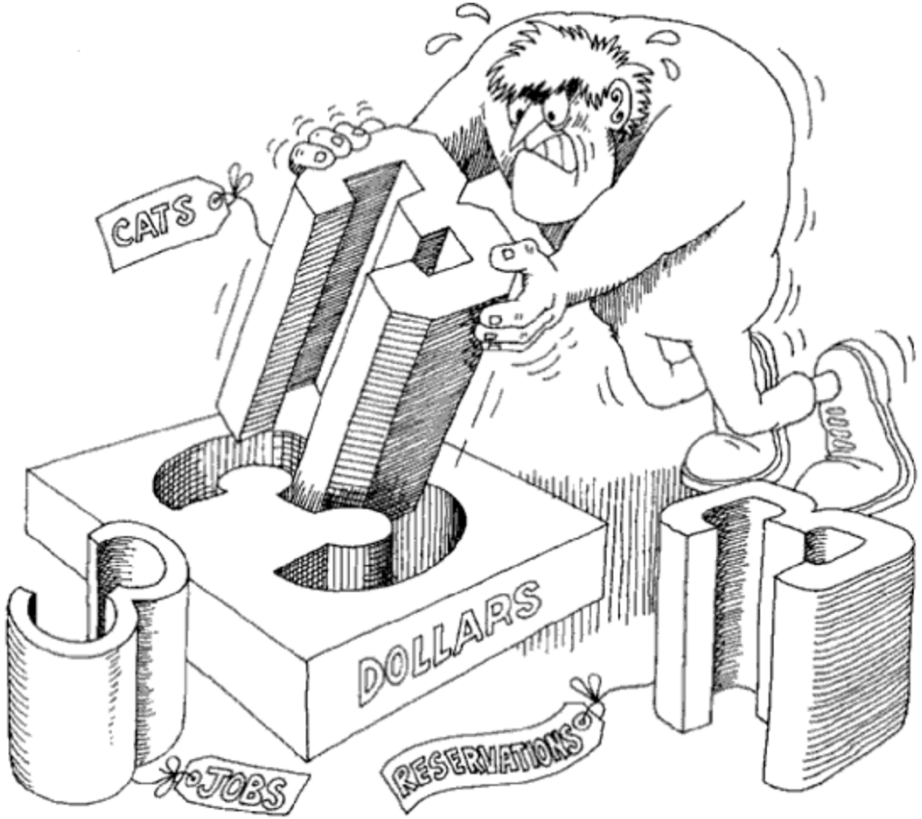


6. Encapsulation

- L'encapsulation est le faite qu'un objet renferme ses propres attributs et ses méthodes. Les détails de l'implémentation d'un objet sont masqués aux autres objets du système à objets. On dit qu'il y a encapsulation de données et du comportement des objets.

- On précise trois modes d'accès aux attributs d'un objet.
  - Le mode **public** avec lequel les attributs seront accessibles directement par l'objet lui-même ou par d'autres objets. Il s'agit du niveau le plus bas de protection.
  - Le mode **private** avec lequel les attributs de l'objet seront inaccessibles à partir d'autres objets : seules les méthodes de l'objet pourront y accéder. Il s'agit du niveau le plus fort de protection.
  - Le mode **protected** : cette technique de protection est étroitement associée à la notion d'héritage. Elle se comporte comme private avec moins de restriction. Une classe dérivée à un accès aux membres protected mais pas aux membres private.

With polymorphism, large case statements are unnecessary, because each object implicitly knows its own type.

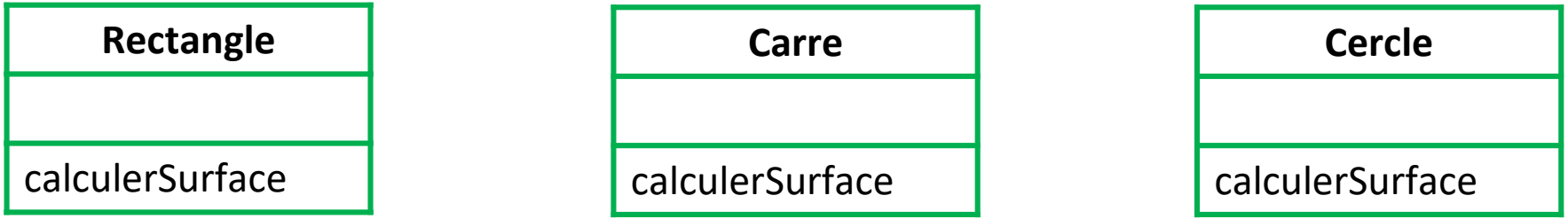


7. Polymorphisme

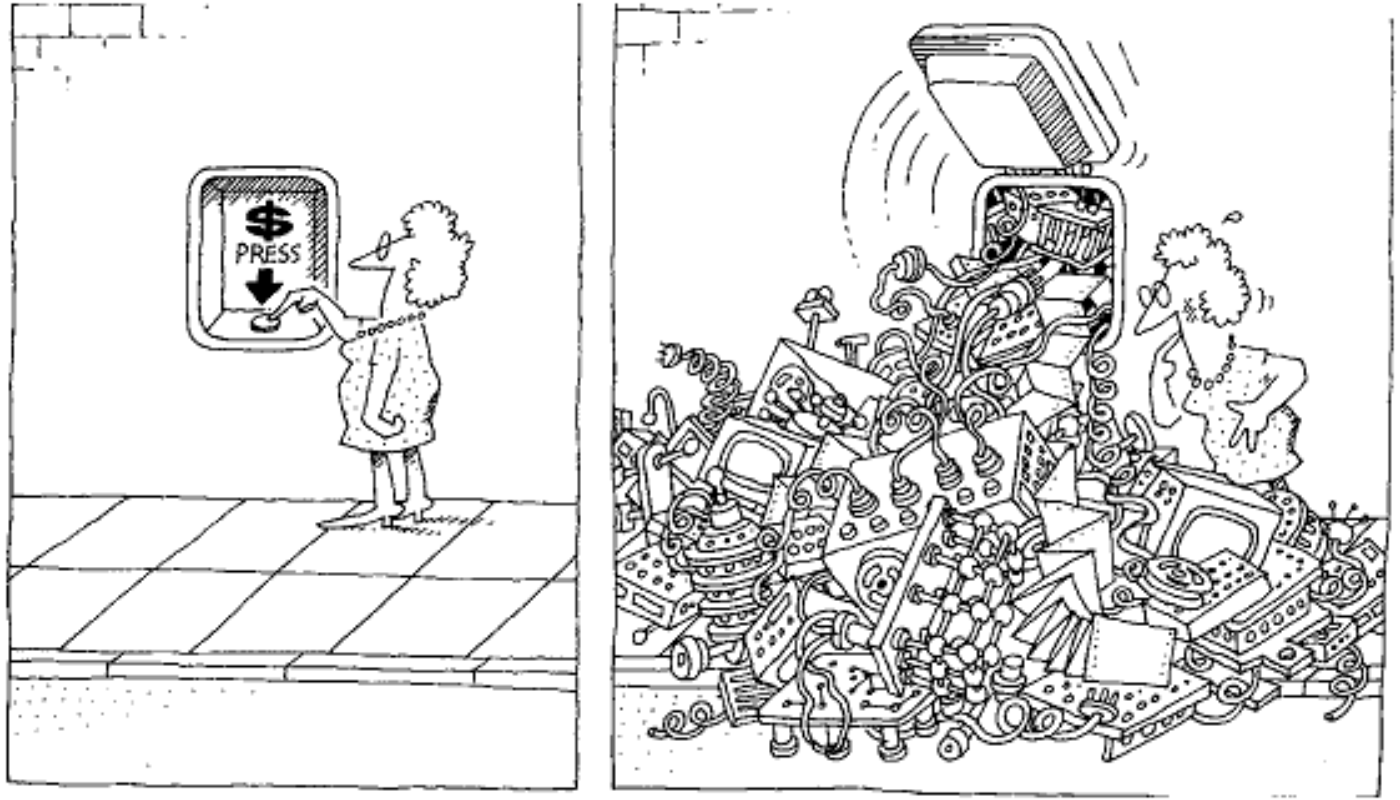
- Le terme **polymorphisme** issu du grec signifie la faculté de prendre plusieurs formes.
- Une entité est **polymorphe** si à l'exécution elle peut se référer à des instances de classe différentes.
- Le **Polymorphisme** veut dire que le même service peut avoir un comportement différent suivant la classe dans laquelle il est utilisé. C'est un concept fondamental de la programmation objet, indispensable pour une utilisation efficace de l'héritage.

7. Polymorphisme

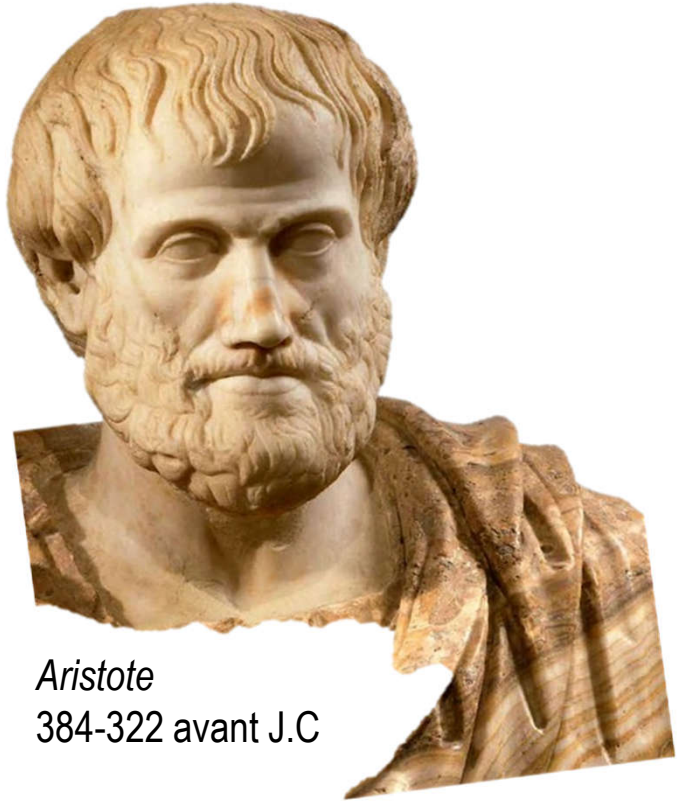
- Le polymorphisme permet d'éviter les codes qui comportent de nombreux embranchements et tests. En d'autres termes, le polymorphisme est un mécanisme qui permet à une sous classe de redéfinir une méthode dont elle a hérité tout en gardant la même signature de la méthode.



The task of the software development team is to engineer the illusion of simplicity.



L’excellence



Aristote
384-322 avant J.C

«L'excellence est un art que l'on n'atteint que par **l'exercice constant**. Nous sommes ce que nous faisons de manière répétée. L'excellence n'est donc pas une action mais une habitude.»

ANALYSE ET CONCEPTION ORIENTÉE OBJET

QCM – SÉQUENCE 1 – Introduction

Les exigences sont souvent très stables. Ils ne changent pas souvent .

- ◯ Vrai
- ◯ Faux

UVS
UNIVERSITE VIRTUELLE DU SENEGAL

Analyse et Conception Orientée Objet

UML