

Concept de la Programmation Orientée Objet

ESTM

L3 Téléinfo

2015-2016

Constats

- ❑ Il y a beaucoup de manières à écrire un programme qui effectue une tâche spécifiée.
- ❑ La manière de programmer dépende du langage utilisé.
- ❑ Le langage utilisé dépend de la manière de programmer.

Paradigmes de programmation

Un paradigme de programmation détermine la vue qu'à le développeur sur son programme.

On peut citer plusieurs :

- ❑ Programmation procédurale : P.P. (Pascal, C, etc.)
- ❑ Programmation orientée objet : P.O.O. (C++, Java, Python, Ruby, Delphi, etc.)
- ❑ Etc.

Différence dans les approches

PP vs POO

- ❑ La programmation procédurale (PP) repose sur l'équation de Niklaus Wirth :

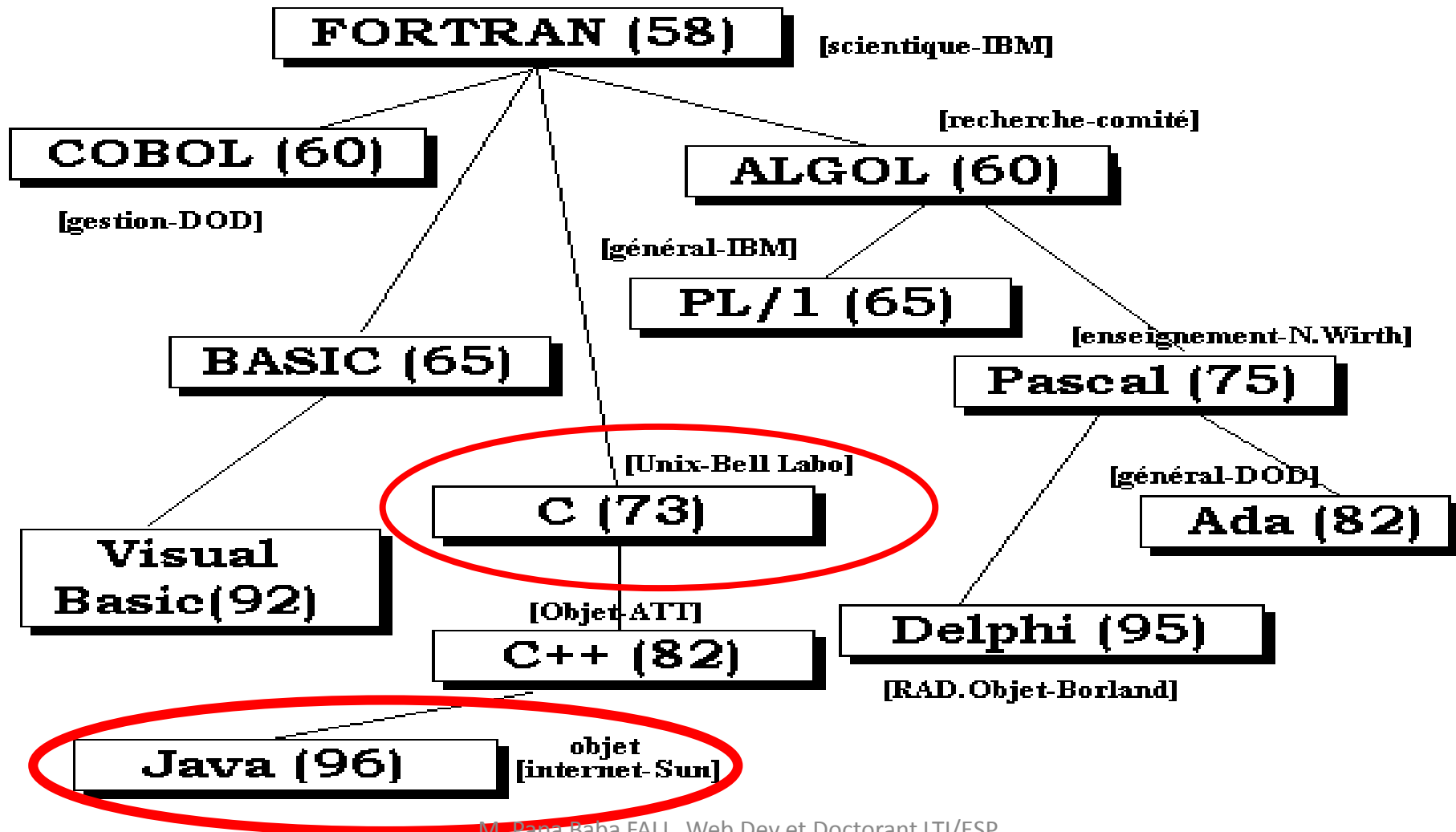
Algorithmes + Structures de données = Programme

- ❑ Apports de la POO

Algorithmes + Structures de données = Programme

- Concept de POO : tout est dirigé par les données;
- Conformité avec le milieu naturel

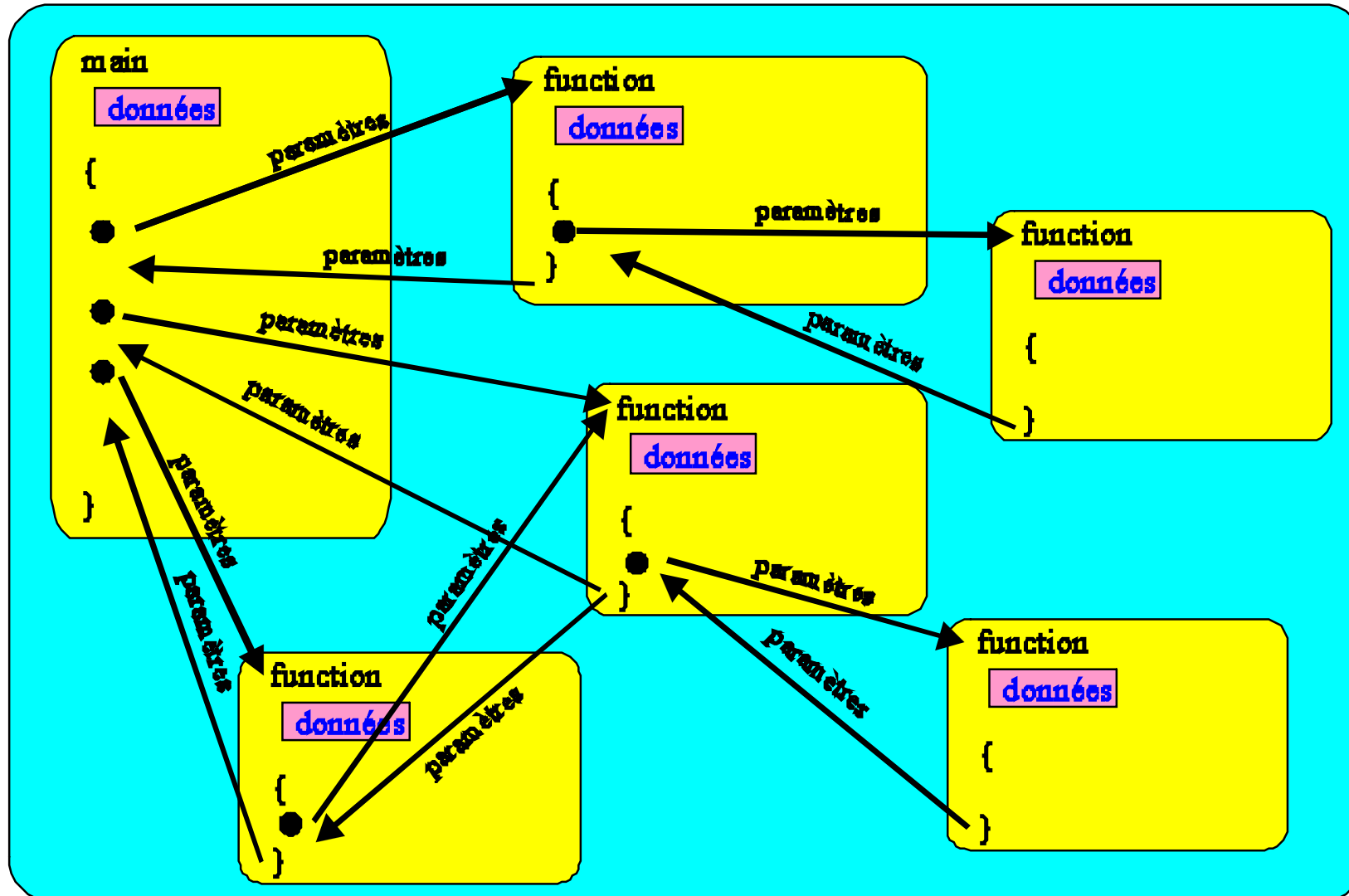
Evolution dans l'approche en programmation



Programmation procédurale (rappel de C)

- ❑ Le programme est composé des fonctions
- ❑ Les données (variables) sont créées à l'intérieure des fonctions ou bien passées comme paramètres
- ❑ Il y a un programme principal (**main**)

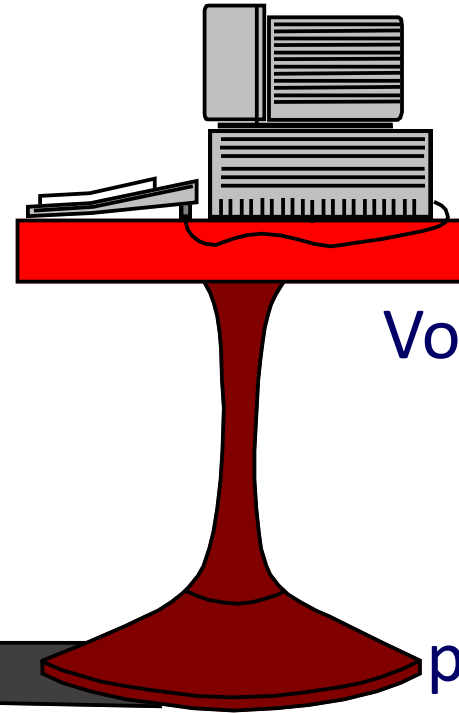
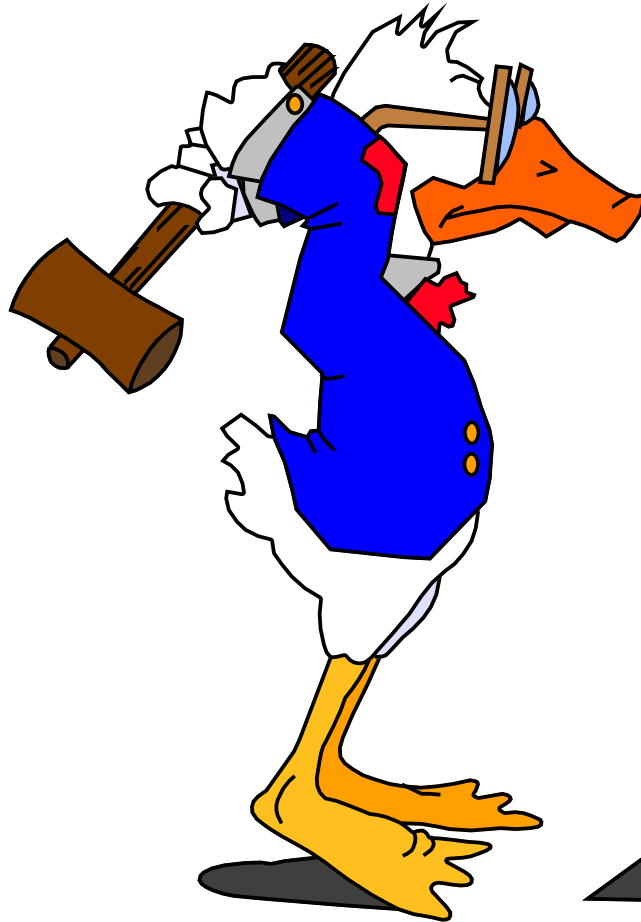
Un programme en C



Limitations

- ❑ Il n'y a pas de méthode ou de cadre pour bien organiser les fonctions.
- ❑ Les modifications d'une fonction entraînent d'autres modifications dans d'autres fonctions, etc. (la portée d'une modification est trop grande et difficile à gérer).
- ❑ Redondance dans le code (la même chose est codée plusieurs fois)
- ❑ Propagation des erreurs – débogage difficile

Est-ce qu'il faut oublier le C?



NON!

**Vous allez avoir le
choix parmi
plusieurs
méthodes de
programmation!**

Paradigme orienté objet

Comment peut on y arriver?

- ❑ Introduction des nouvelles (?) notions
 - objet
 - classe
 - instanciation
 - événement
 - hiérarchie des classes (héritage)
 - polymorphisme
- ❑ On va utiliser ces notions pour introduire le paradigme de programmation orientée objet.

Objet (exemple / fenêtre)

❑ propriétés d'une fenêtre

- ouverte/fermée
- cassée/intacte
- taille
- sens d'ouverture
- type de verre
- coefficient de réflexion de chaleur

❑ **Pour une fenêtre concrète, ces propriétés ont des valeurs.**

Objet (exemple / fenêtre)

❑ opérations avec une fenêtre donnée

- ouvrir
- fermer
- casser
- réparer
- changer la verre

Objet (exemple / livre)

❑ propriétés d'un livre dans une bibliothèque

- état (emprunté / disponible / perdu)
 - date de la fin de l'emprunt
 - titre
 - auteur
 - nombre de pages
- **Pour un livre donné, ces propriétés ont des valeurs!**

Objet (exemple / livre)

❑ opérations sur un livre d'une bibliothèque

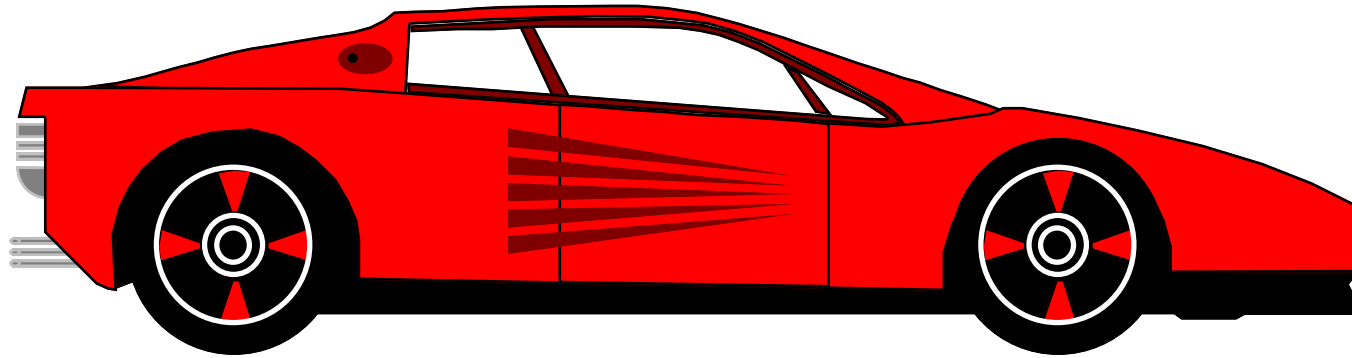
- emprunter
- rendre
- perdre
- voler

Objet (définition)

 Un objet se définit par un triplet :

- un **identificateur** (*OID ou Object Identifier*) qui permet de le référencer de façon unique dans le programme.
- des données (**attributs**) caractérisant l'objet. Ce sont des variables stockant des informations *d'état de l'objet*.
- et des procédures (**méthodes**) agissant sur ces données.
 - ✓ caractérisent son comportement, c.-à-d. l'ensemble des *opérations* que l'objet est à même de réaliser.
 - ✓ permettent de faire réagir l'objet aux sollicitations extérieures (ou d'agir sur les autres objets).
- **Remarque**: le contexte de l'application à réaliser dira quelles sont les propriétés et les méthodes indispensable pour les objets.

Objet (exercice)



☐ Propriétés intéressantes?

☐ Actions intéressantes?

(pour une compétition, pour une entreprise qui loue des voitures, etc...)

Classes

- ❑ Une **classe** est la description d'une famille d'objets ayant même structure et même comportement.
- ❑ Elle regroupe :
 - ❖ un ensemble d'**attributs** : données représentant l'état de l'objet;
 - ❖ et un ensemble de **méthodes** : opérations applicables aux objets.
- ❑ Objet = instantiation d'une classe.

Classe des objets

- Les objets ayant des mêmes propriétés et les mêmes méthodes peuvent être mis dans une classe.



 **Une classe sera définie par les propriétés et les méthodes sur ses éléments.**

Classe des livres

❖ propriétés

- état (emprunté / disponible / perdu)
- date de la fin de l'emprunt
- titre
- auteur
- nombre de pages

❖ méthodes

- emprunter
- rendre
- perdre
- voler

Classe des fenêtres

❖ propriétés

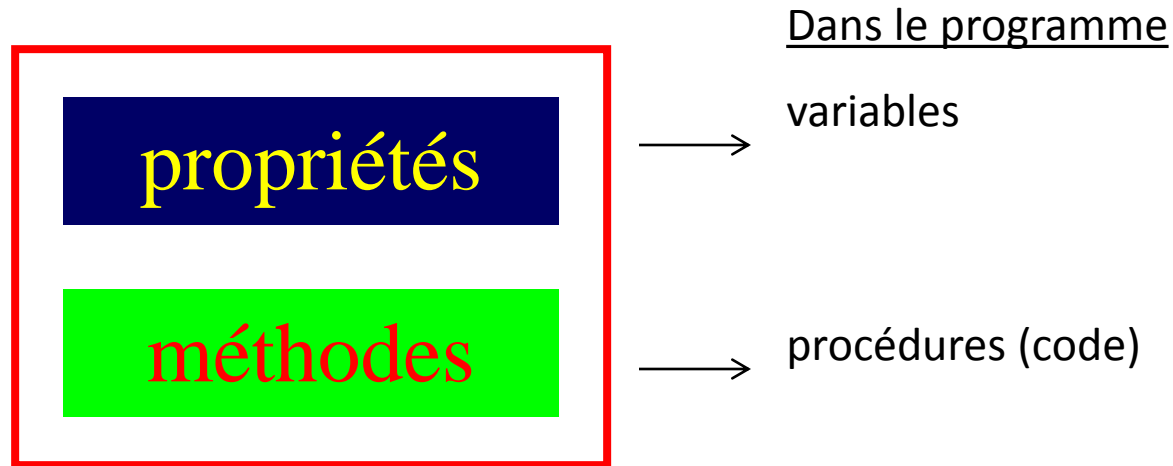
- état d'ouverture (ouverte/fermée)
- état (cassée/intacte)
- taille
- sens d'ouverture
- type de verre
- coef de réflexion de chaleur

❖ méthodes

- ouvrir
- fermer
- casser
- reparer
- Changer la verre

Classe

- ❑ Une classe est un ensemble des propriétés et des méthodes.



ENCAPSULATION

Une classe dans un programme

- ❑ Une classe est une collection de données et des méthodes sur ces données.
- ❑ Une classe est une **encapsulation** de données et de traitements.
- ❑ La notion de l'encapsulation est la création des classes
 - donne une organisation naturelle des fonctions (méthodes)
 - permet d'effectuer la modification locale du traitement

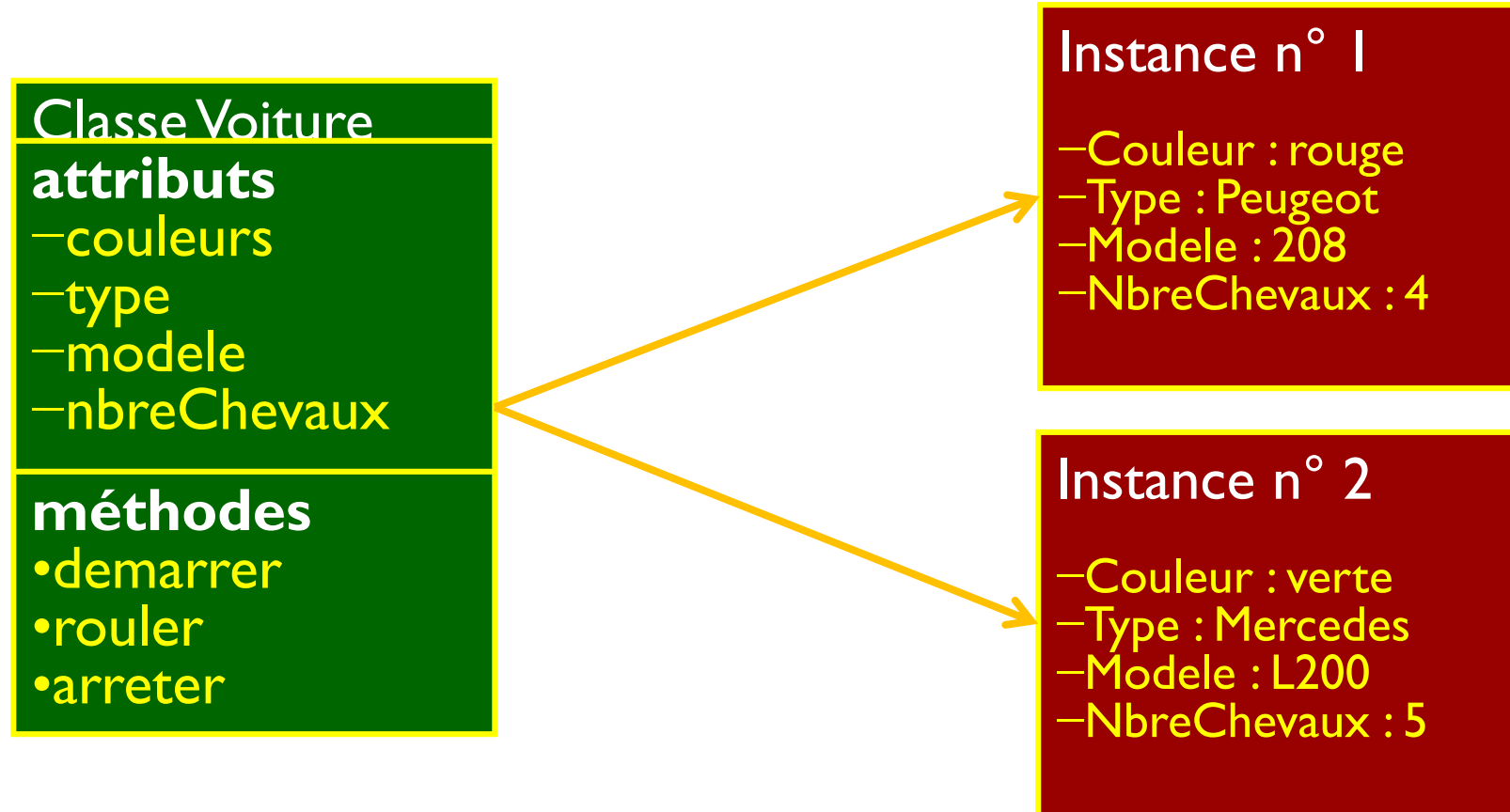
Exemple dans un langage de programmation (JAVA)

```
public class Cercle;  
{  
    double Rayon;  
    public double calculerAire()  
    {  
        return Rayon * Rayon * 3.1415;  
    }  
}
```

Instantiation

- ❑ Un objet est une instantiation d'une classe
- ❑ Les propriétés (i.e. des variables) de la classe ont des valeurs.
- ❑ Les méthodes de la classe fonctionnent sur l'objet.

Instanciación ejemplo



Une instance de la classe livre

propriétés

- état = emprunté
- date de la fin de l'emprunt = 2003/03/20
- titre = Concepts of object-oriented programming
- auteur = David N. Smith
- nombre de pages = 189


méthodes

- emprunter
- rendre
- perdre
- voler

Comment créer un objet?

- ❑ Dans chaque classe, il y a une méthode spéciale :

La méthode constructeur

-  Cette méthode permet de créer un nouveau objet de la classe en définissant les valeurs des propriétés et en permettant accès aux méthodes sur cet objet.

Une instance de la classe livre

propriétés

- état = emprunté
- date de la fin de l'emprunt = 2003/03/20
- titre = Concepts of object-oriented programming
- auteur = David N. Smith
- nombre de pages = 189

méthodes

- emprunter
- rendre
- perdre
- voler
- livre

méthode
constructeur

Exemple dans un langage de programmation (JAVA)

```
public class Cercle;  
{  
    double Rayon;  
    public double calculerAire()  
    {  
        return Rayon * Rayon * 3.1415;  
    }  
    public Cercle(double r) // constructeur  
    {  
        Rayon = r;  
    }  
}
```

Objet comme valeur d'une propriété

Un objet peut être la valeur d'une propriété

Exemple

- ❑ On définit la classe des roues
- ❑ Ensuite, on définit la classe des voitures
- ❑ Pour une voiture donnée, on a quatre roues qui sont de la classe des roues.

Exemple dans un langage de programmation

```
public class Dessin  
{  
    Cercle Composant1;  
    double surfaceDessin;  
}
```


Exemple (objet comme valeur d'une propriété)

Classe auteur

propriétés

- nom (String)
- prénom (String)
- état (mort/vivant)
- prix (liste)

méthodes

- obtenir prix
- auteur

Exemple (objet comme valeur d'une propriété)

Classe livre

propriétés

- état (emprunté / disponible / perdu)
- date de la fin de l'emprunt
- titre
- auteur
- nombre de pages

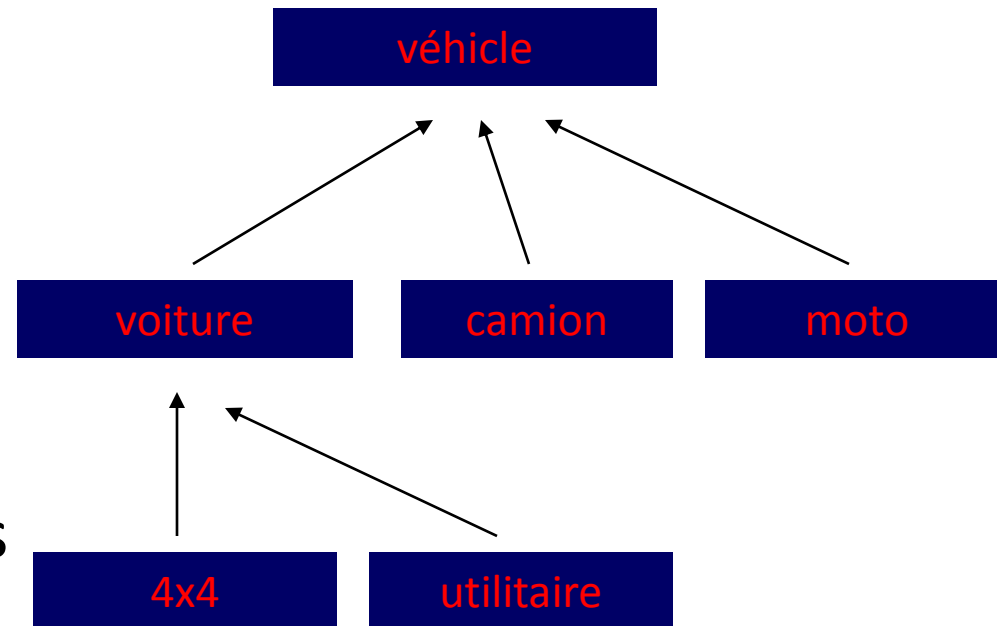
méthodes

- emprunter
- rendre
- perdre
- voler
- livre

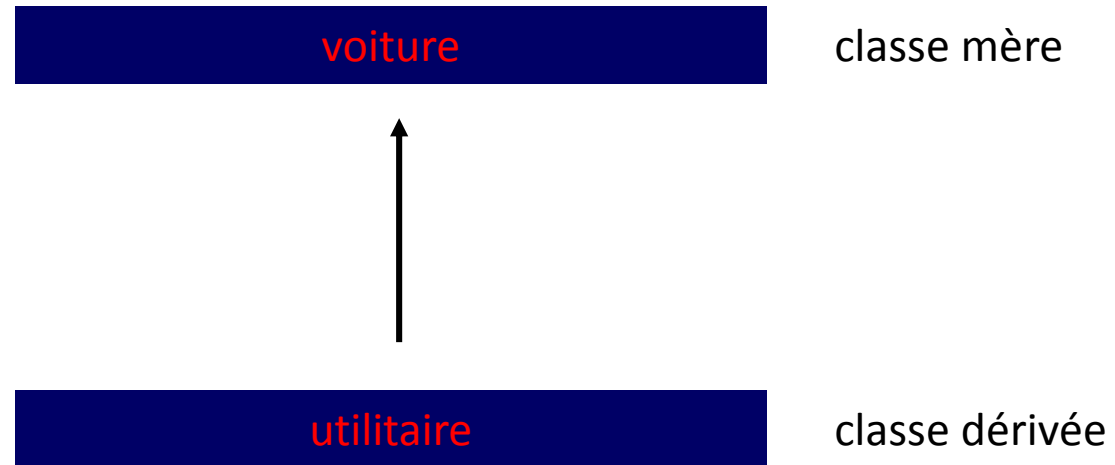
La valeur est une instantiation de la classe auteur

Hiérarchie des classes

- Exemple :
classification des
espèces (Darwin)
- La hiérarchie va
des classes
générales vers les
classes spécifiques



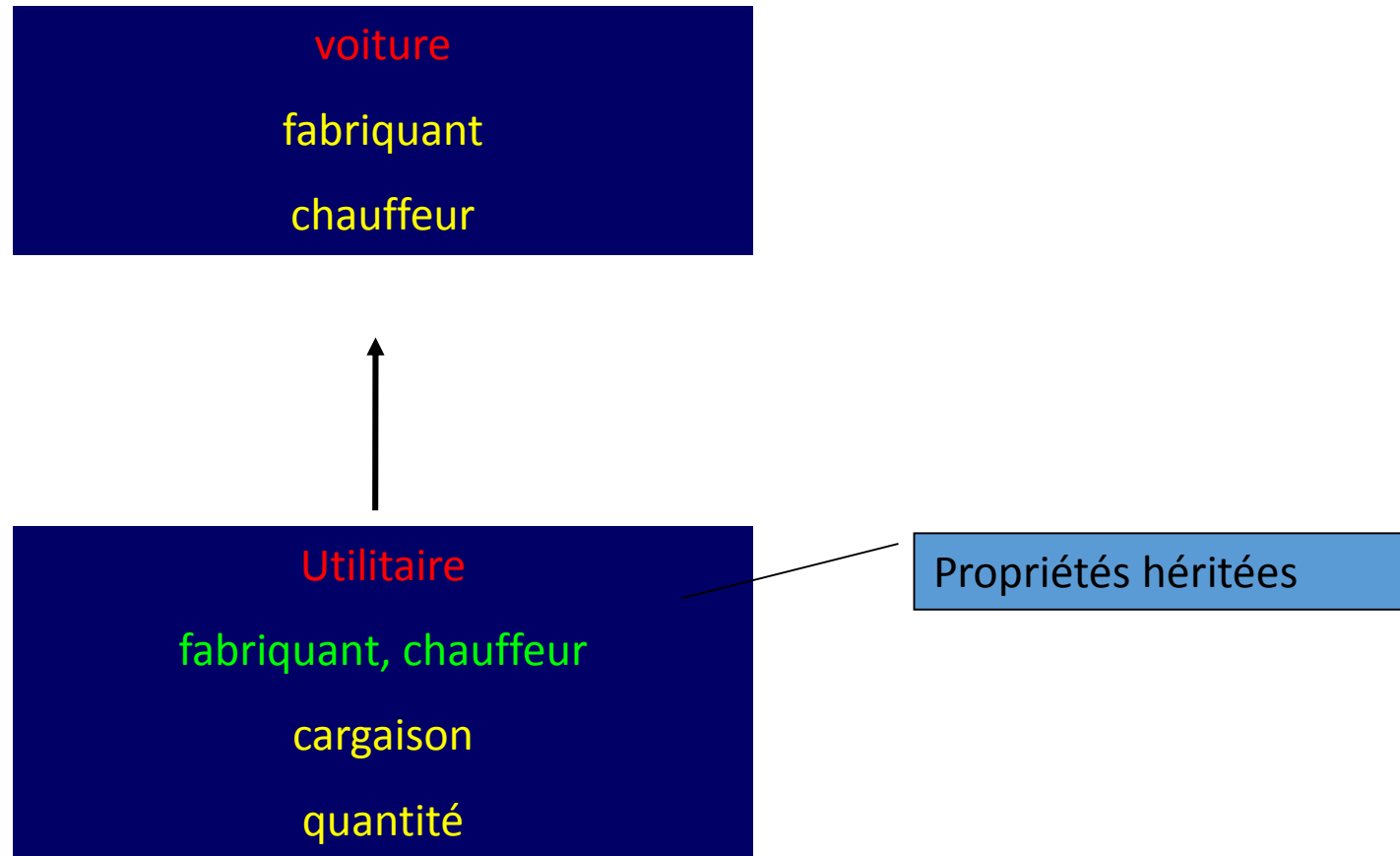
Hiérarchie des classes



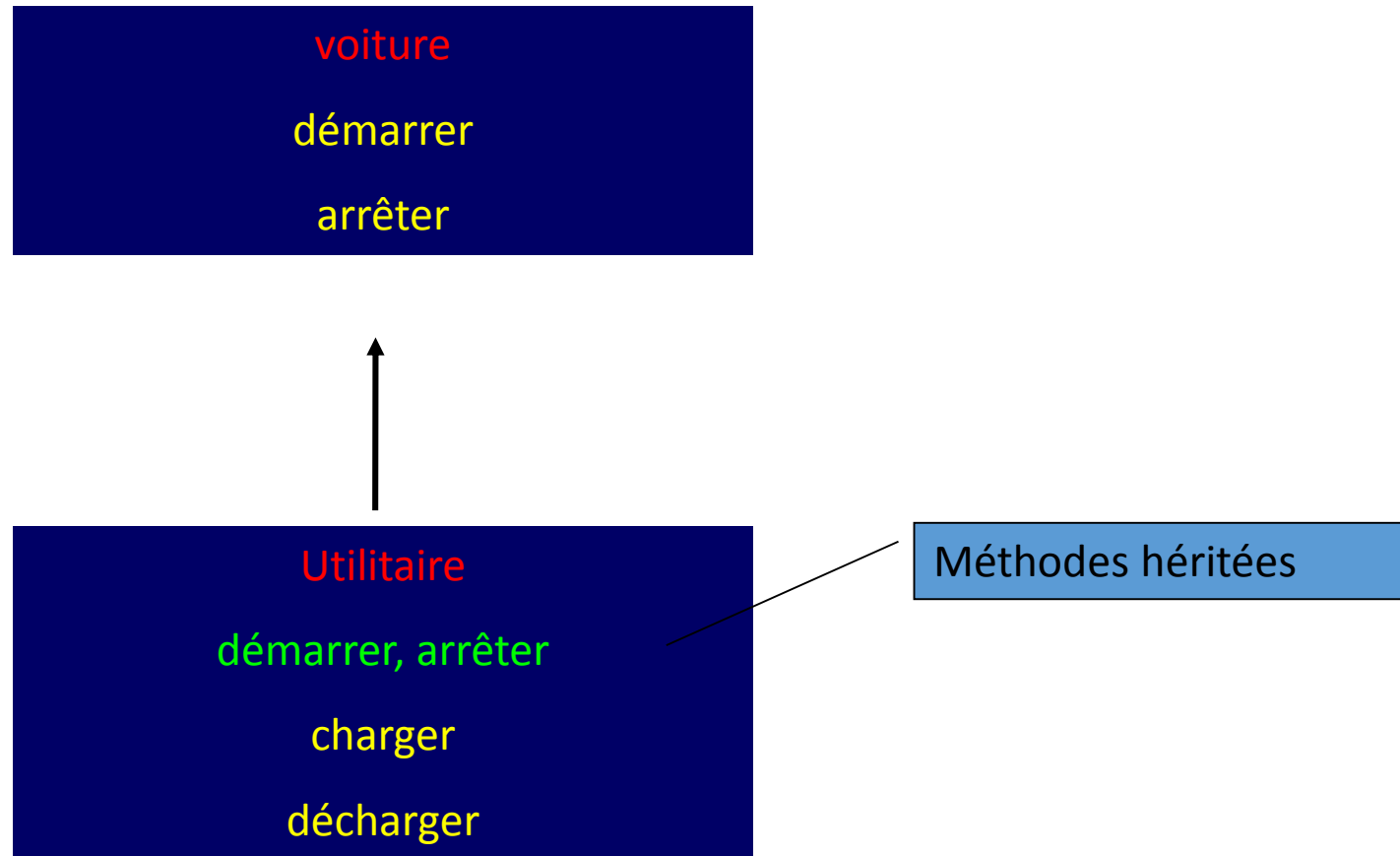
Héritage

- ❑ la classe dérivée possède
 - TOUTES LES PROPRIETES DE SA CLASSE MERE
 - TOUTES LES METHODES DE LA CLASSE MERE

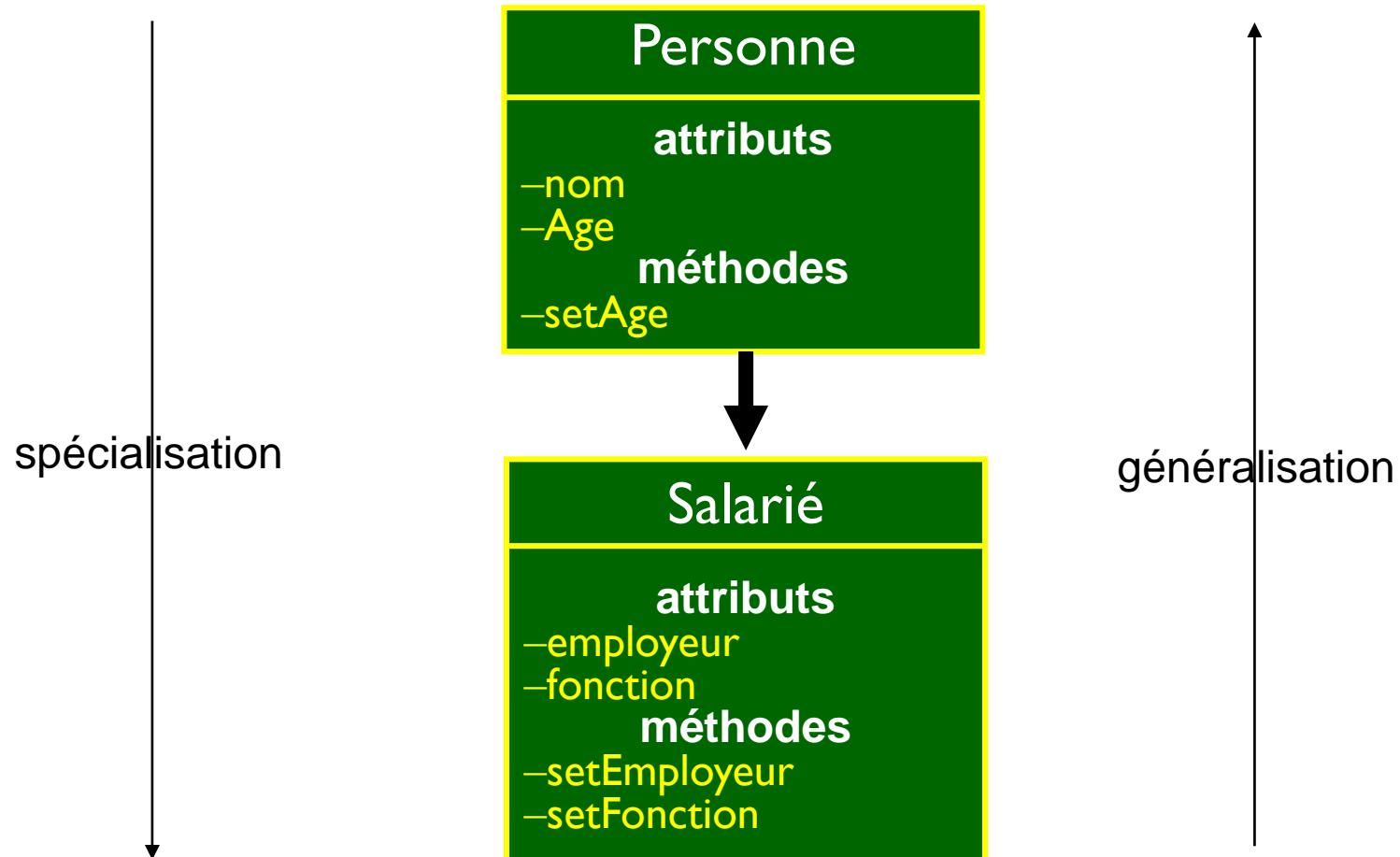
Héritage des propriétés (exemple)



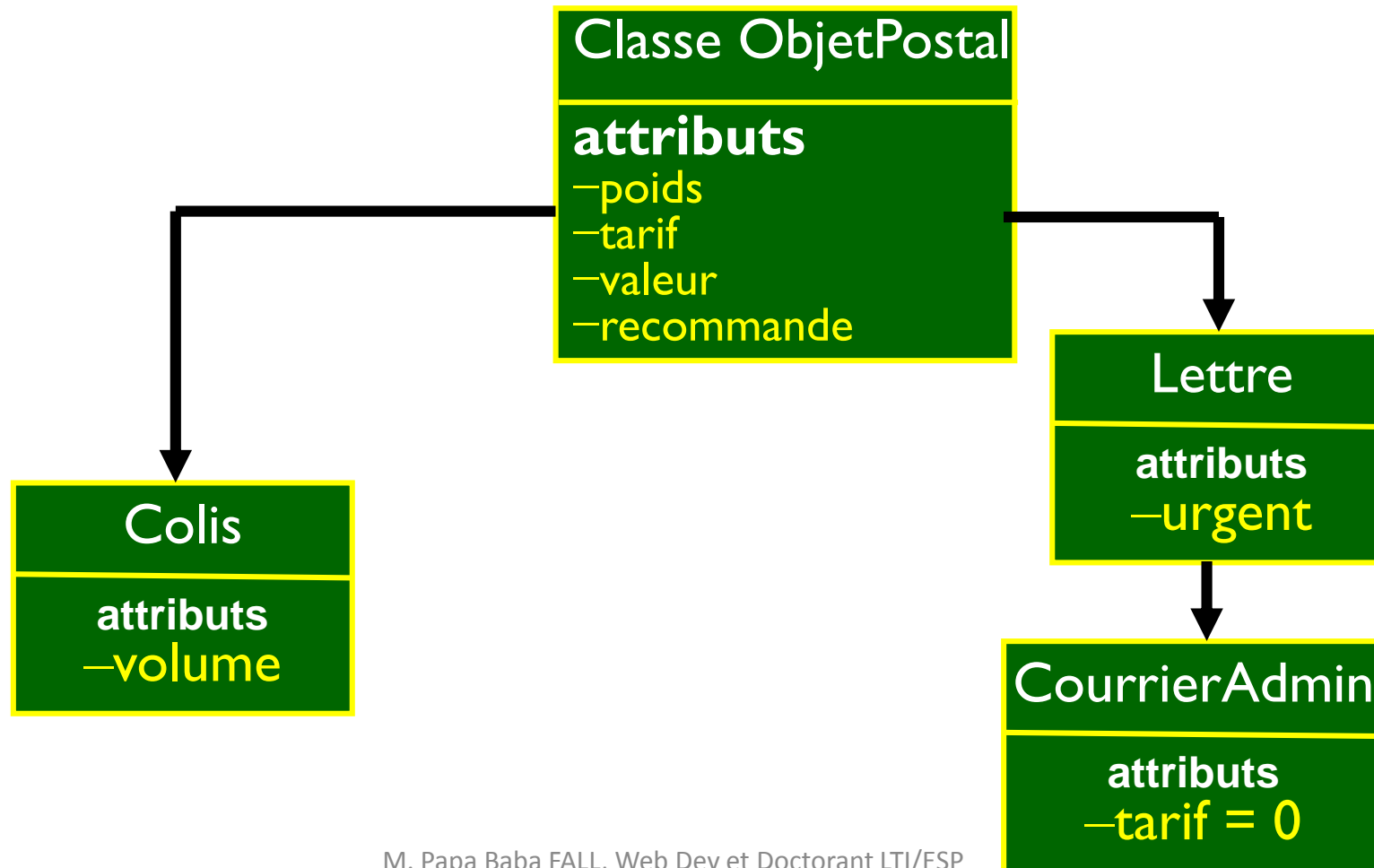
Héritage des méthodes (exemple)



L'héritage : exemple I



L'héritage : exemple 2



Héritage : Redéfinition de méthode

- Réécrire l'implémentation d'une méthode héritée sans modifier sa signature (seul l'objet récepteur diffère)
- Une méthode héritée peut être **redéfinie** dans la sous-classe: on conserve la même signature et le même type de retour.
- On peut étendre le comportement d'une méthode héritée: la méthode de la super-classe reste accessible pour les objets de la sous-classe (*mot-clé **super***).

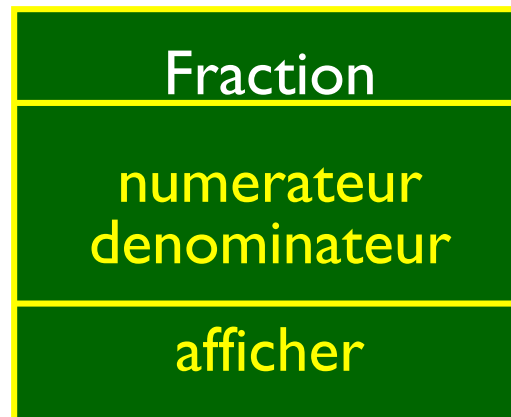
Héritage : Surcharge de méthodes

- Redéfinition de la signature et du code d'une méthode héritée
- La **surcharge** concerne 2 méthodes de même nom mais avec des signatures différentes.

✓ exemple : constructeurs de la classe *Fraction*

Fraction (int numérateur, int dénominateur)

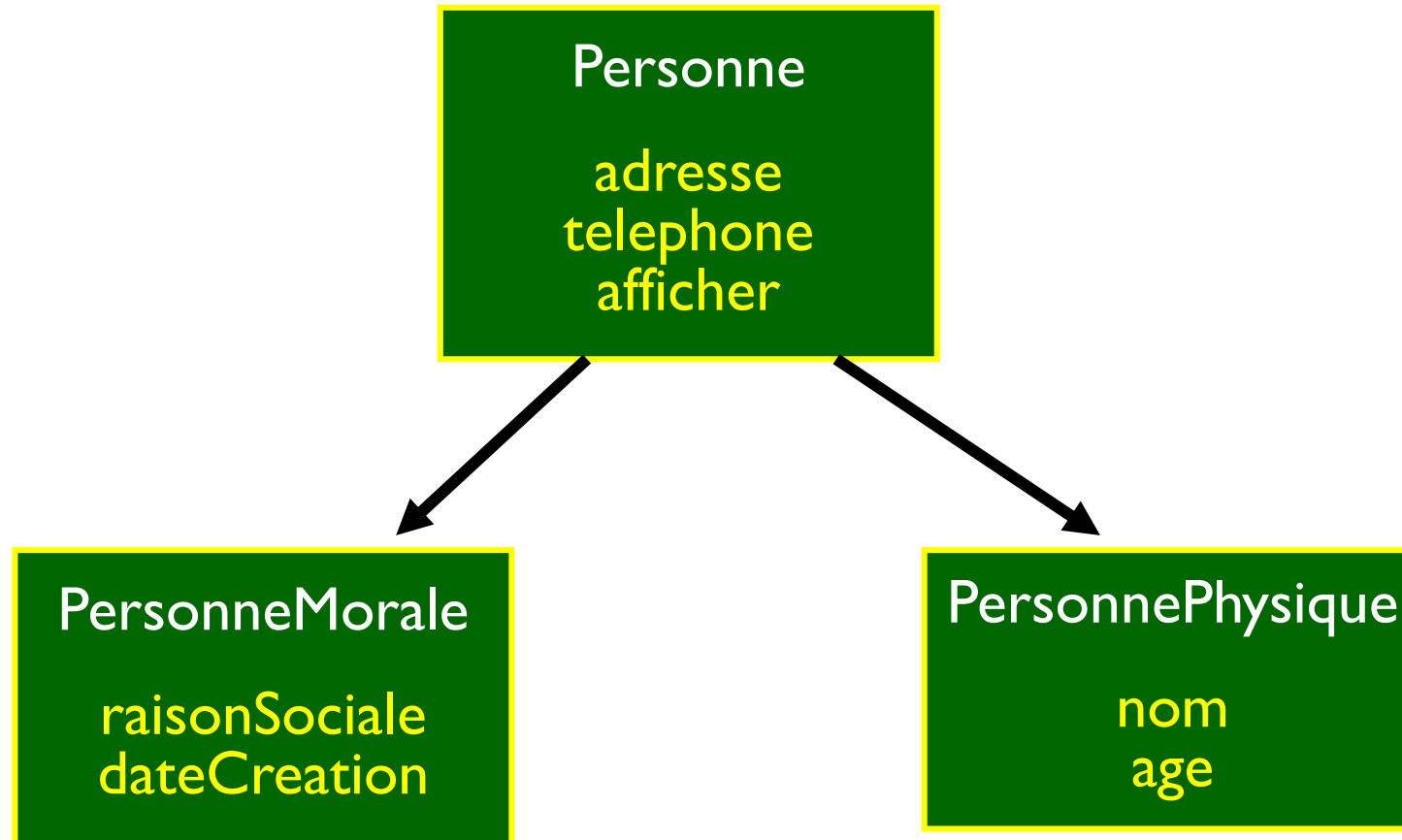
Fraction (int numérateur)



Classe abstraite

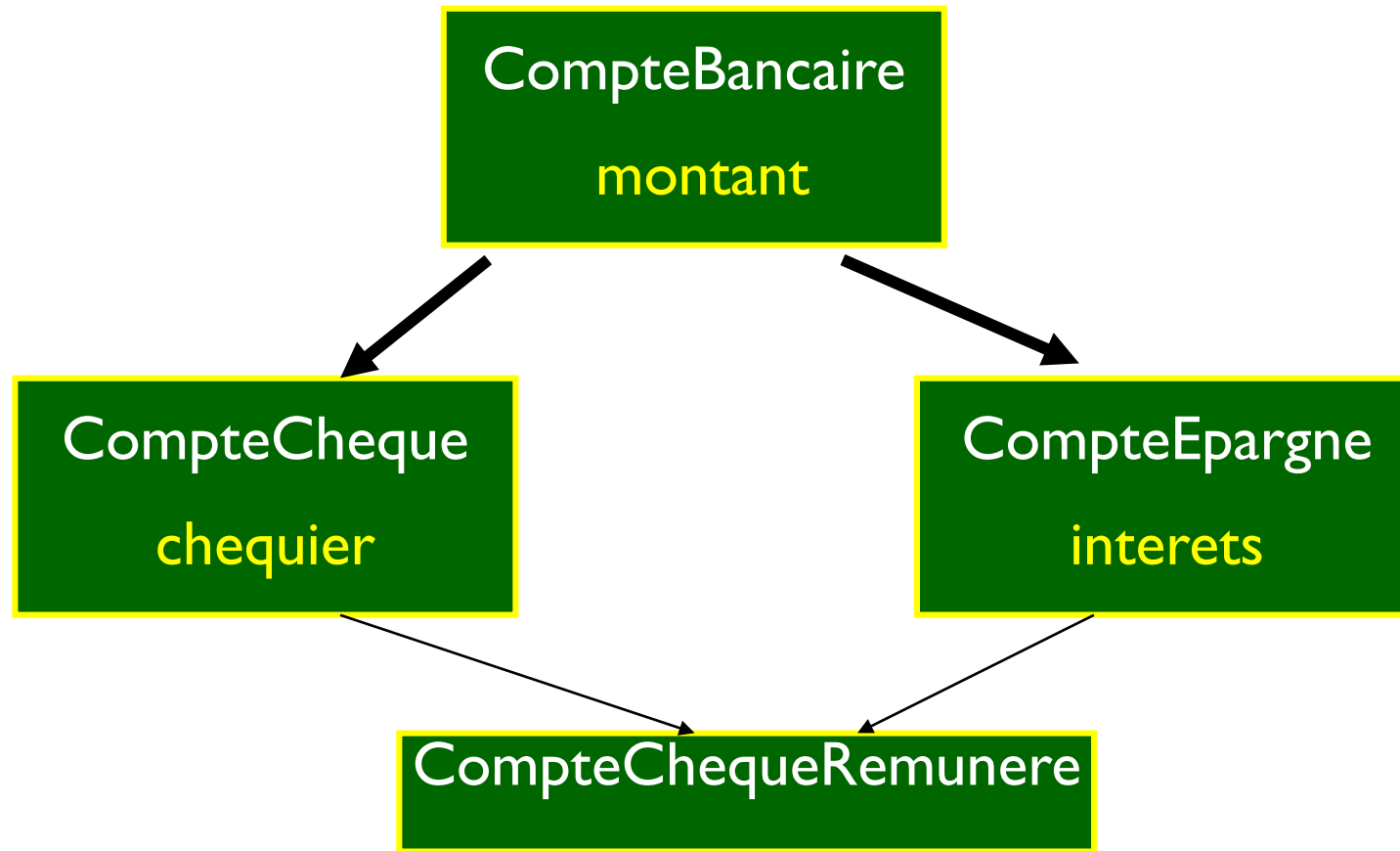
- ⑩ Une classe abstraite est une classe utilisée pour factoriser des propriétés communes à plusieurs classes mais qui est trop générique pour être instanciée.
- ⑩ Une classe abstraite est telle que l'on ne peut pas caractériser de manière précise ses instances.

Classe abstraite : exemple

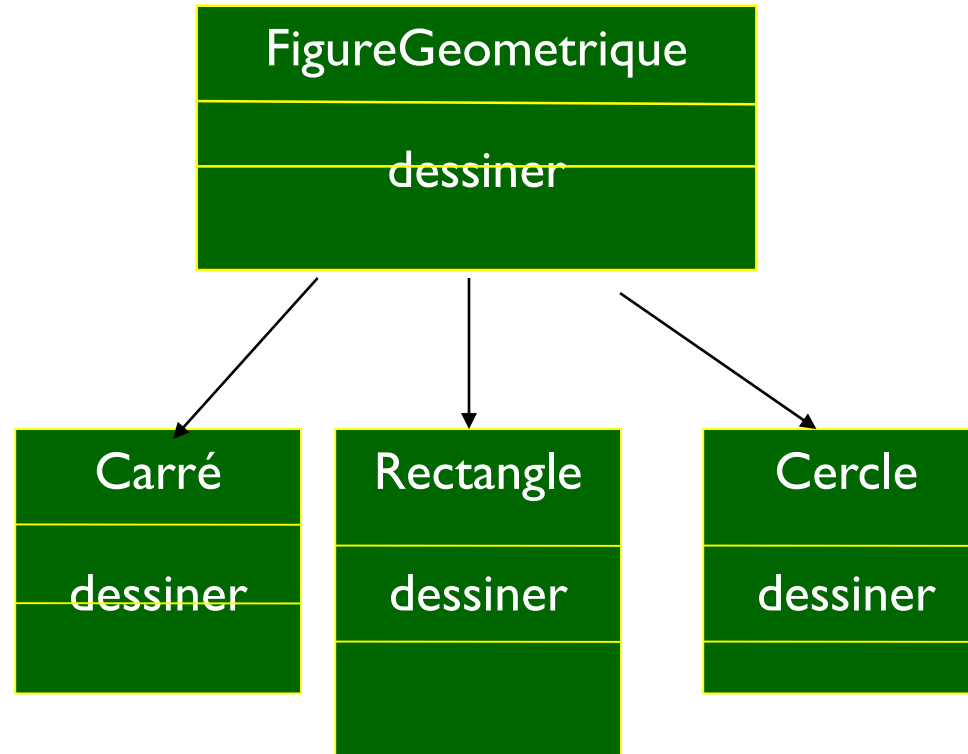


L'héritage multiple

- ⑩ Permet à classe d'hériter de plusieurs classes mères.



Le polymorphisme (exemple)



- La méthode « *dessiner* » est implémentée différemment dans les classes dérivées.
- Le système appellera la méthode « *dessiner* » spécifique selon la forme de l'objet concerné.

Exemple dans un langage de programmation

```
public class CercleUnitaire extends Cercle
{
    CercleUnitaire()
    {
        Rayon=1;
    }
}
```


Hiérarchie des classes

- ☐ Exercice : donnez une classification hiérarchique de quelques éléments de dessins (ligne, triangle, carré, losange cercle, etc.)
- ☐ Exercice : donnez une classification hiérarchique d'une base de données des livres dans une bibliothèque
- ☐ Exercice : donnez une classification hiérarchique de ce que vous voulez.

Le polymorphisme

- ⑩ Faculté d'une méthode donnée à s'exécuter différemment suivant le contexte de la classe où elle se trouve.
- ⑩ Une méthode définie dans une superclasse peut s'exécuter de manière différente selon la sous-classe où elle est héritée.
- ⑩ C'est le cas lorsqu'une sous-classe hérite d'une méthode sans la surcharger.

Le polymorphisme

⑩ La liaison dynamique consiste dans ce cas, lors de l'exécution, à :

- ❖ parcourir de façon ascendante le graphe d'héritage pour trouver l'objet récepteur.
- ❖ exécuter la première méthode dont la signature correspond à celle du message sera exécutée.

Evénements

Notion de message et de liaison

⑩ L'activation de méthode se fait par envoi d'un **message** à l'objet concerné (objet récepteur).

- ❖ Message = récepteur + signature de méthode

- ❖ Signature de méthode = (nom méthode + paramètres)

⑩ **Liaison** : mécanisme consistant à sélectionner le code de la méthode à activer lors de la réception d'un message par un objet récepteur :

- ❖ Liaison **statique** : réalisée à la compilation du code source

- ❖ Liaison **dynamique** : réalisée à l'exécution (cas de Java)

Un programme orienté objet

- ❑ modélisation du domaine à l'aide des classes
- ❑ définition des classes
- ❑ création des instances (peut être dynamique)
- ❑ messages entre les objets (appel des méthodes)