INTRODUCTION AUX PRINCIPES DE LA PROGRAMMATION ORIENTEE OBJET

Christophe GNAHO
christophe.gnaho@u-paris.fr

PLAN

- Objectifs
- Programmation O.O. versus Programmation Structurée
- Concept d'*objet* et concept de *classe*
- Les trois principes de la POO
- Synthèse

Objectifs

Tous les langages de Programmation Orientée Objet (comme le langage Java ou le C++, etc.) reposent sur les mêmes principes.

La compréhension de ces principes fondamentaux est **un prérequis nécessaire à une bonne utilisation des langages.**

En préambule à l'étude de la programmation en langage Java, nous présentons ces principes.

Nous abordons respectivement :

Le concept **d'Objet** et le concept de **Classe**

Les trois principes de la POO

- **Encapsulation**
- **Héritage**
- **Polymorphisme**

Programmation O.O. vs Programmation Structurée

Que doit faire le programme ?

Programmation Structurée

Programme ?

Sur quoi porte le programme ?

Programmation Orientée Objet

Programmation O.O. vs Programmation Structurée

Créditer un compte,
débiter un compte ?

Programmation
Structurée

**Programme de gestion
de comptes bancaires?**

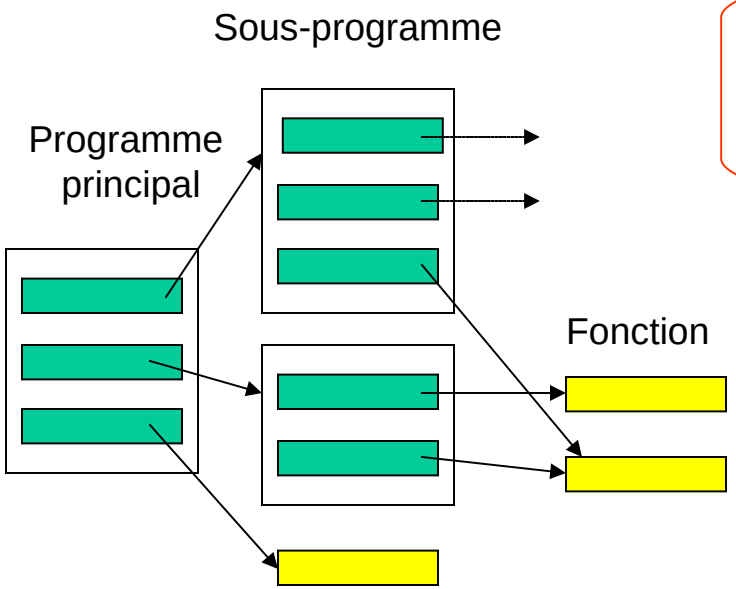
Comment se présente
(structure) un compte
bancaire ?

Programmation
Orientée Objet

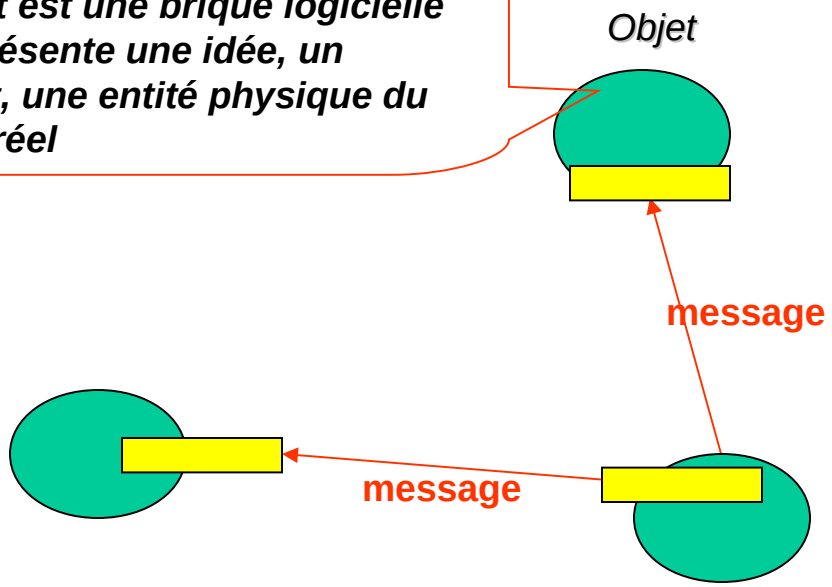
Programmation O.O. vs Programmation Structurée

Programmation Structurée

Programmation Orientée Objet



Un objet est une brique logicielle qui représente une idée, un concept, une entité physique du monde réel



Les objets communiquent entre eux par des *messages*

Lorsqu'un objet reçoit un *message,* il déclenche une de ses *fonctions* pour :

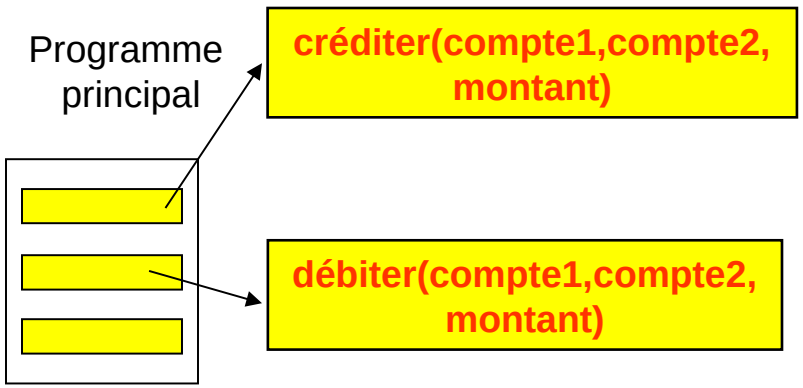
- modifier son état

et / ou

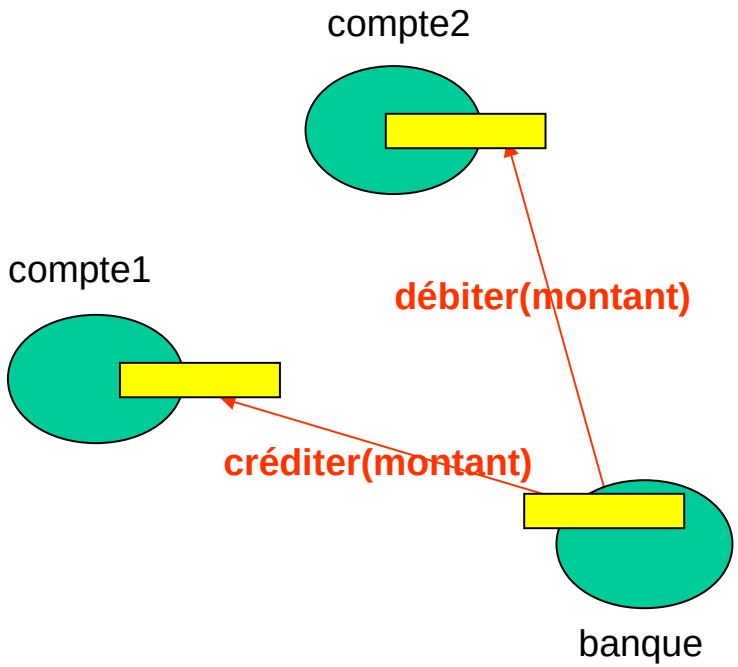
- envoyer un message à un autre objet

Programmation O.O. vs Programmation Structurée

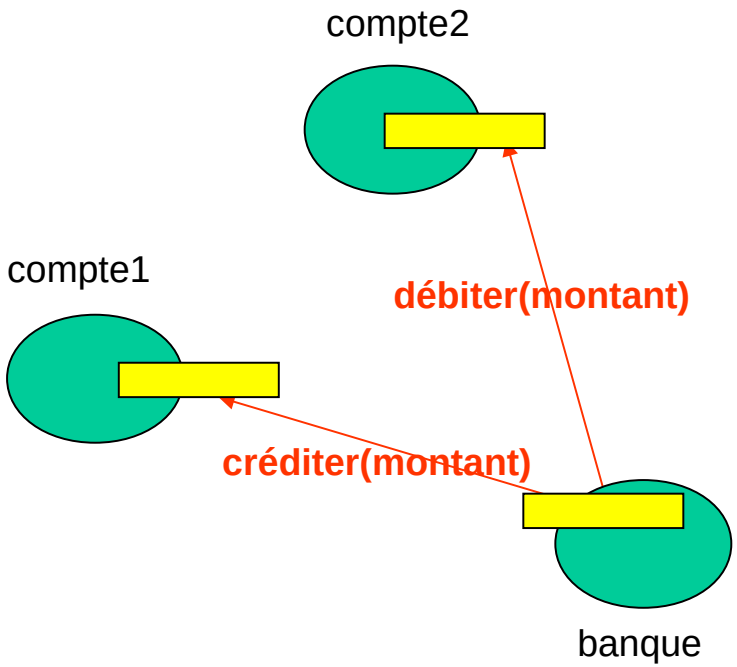
Programmation Structurée



Programmation Orientée Objet



Programmation O.O. vs Programmation Structurée



L'objet *banque* envoi un message (*créditer (montant)*) à l'objet *compte1*
L'objet *compte1* réagit à ce message en déclenchant sa fonction *créditer(montant)*

L'objet *banque* envoi un message (*débiter (montant)*) à l'objet *compte2*
L'objet *compte2* réagit à ce message en déclenchant sa fonction *débiter(montant)*

Programmation O.O. vs Programmation Structurée

Objectifs visés par la POO

- Permettre de développer une partie d'un programme sans qu'il soit nécessaire de connaître les détails internes aux autres parties
- Permettre d'apporter des modifications locales à un module, sans que cela affecte le reste du programme
- Permettre de réutiliser des fragments de code développés dans un cadre différent
- Faciliter l'évolution du code
- Améliorer la conception et la maintenance des grands systèmes
- Programmation par «composants». Conception d'un logiciel à la manière de la fabrication d'une voiture

CONCEPT D'OBJET

D'un point de vue Informatique, un objet est un ensemble de données structurées présent dans la mémoire de l'ordinateur au cours de l'exécution d'un programme.

Un objet combine des données et du code exécutable (travaillant sur ces données). Ce dernier correspond à un certain nombre d'opérations que l'objet est capable d'effectuer.

Les données gérées par un objet sont retenues dans **un ensemble de variables internes à cet objet**.
Les opérations réalisables par un objet sont représentées par un ensemble de méthodes (fonctions).

Un objet Personne

Données structurées

nom = Dulac
âge = 30
profession = Infirmière

Opérations

nomEnMajuscule()
changerAge()
changerProfession()

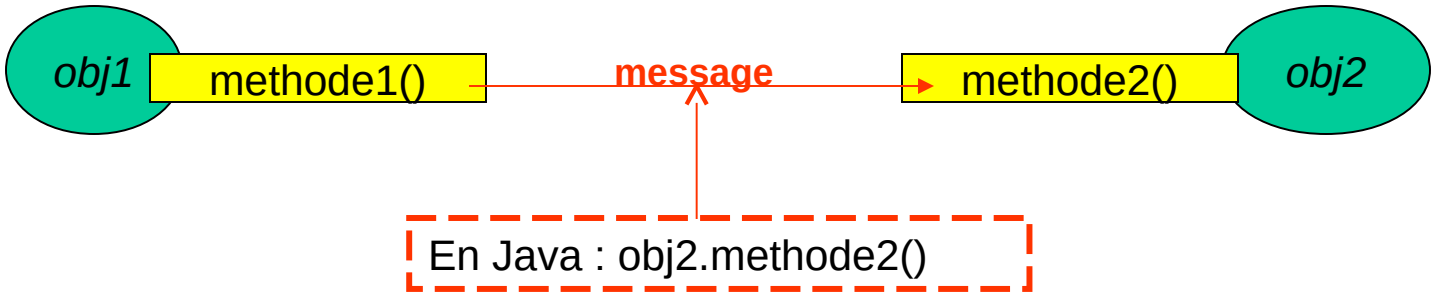
CONCEPT D'OBJET

L'exécution d'un programme est vue comme une suite d'interactions (d'envois de messages) entre objets.

Un message émis d'un objet *obj1* à un objet *obj2* représente une requête d'effectuer une opération. La réception d'un message par un objet provoque l'invocation d'une méthode (fonction).

Les instructions exécutées lors de l'invocation d'une méthode sont principalement de trois types :

- manipuler les données de l'objet
- envoyer des messages
- créer de nouveaux objets

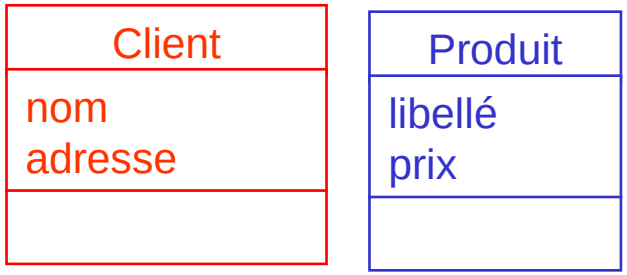


CONCEPT DE CLASSE

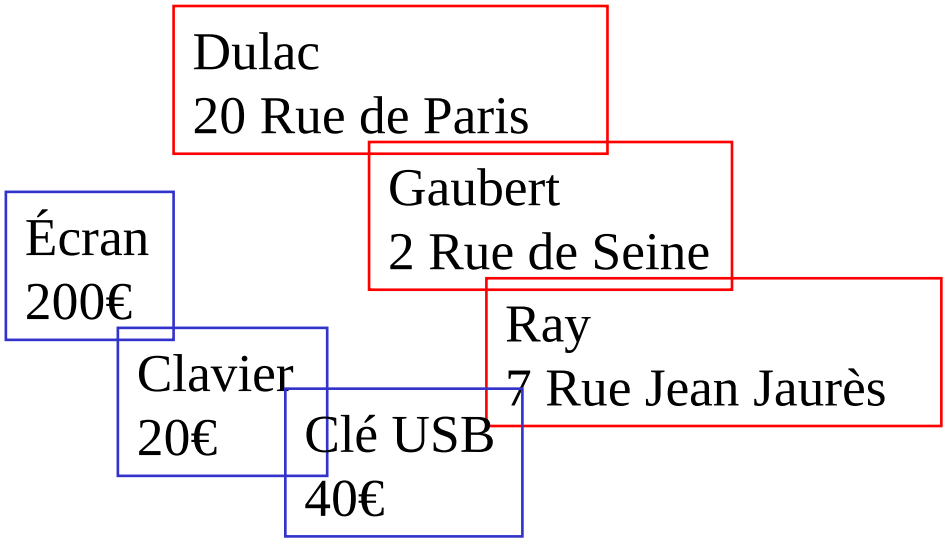
Tous les objets ayant les mêmes structures de données et les mêmes méthodes sont rassemblés dans une famille.

Une telle famille est appelée **classe**, les objets qu'elle rassemble sont des **instances.**

Classe

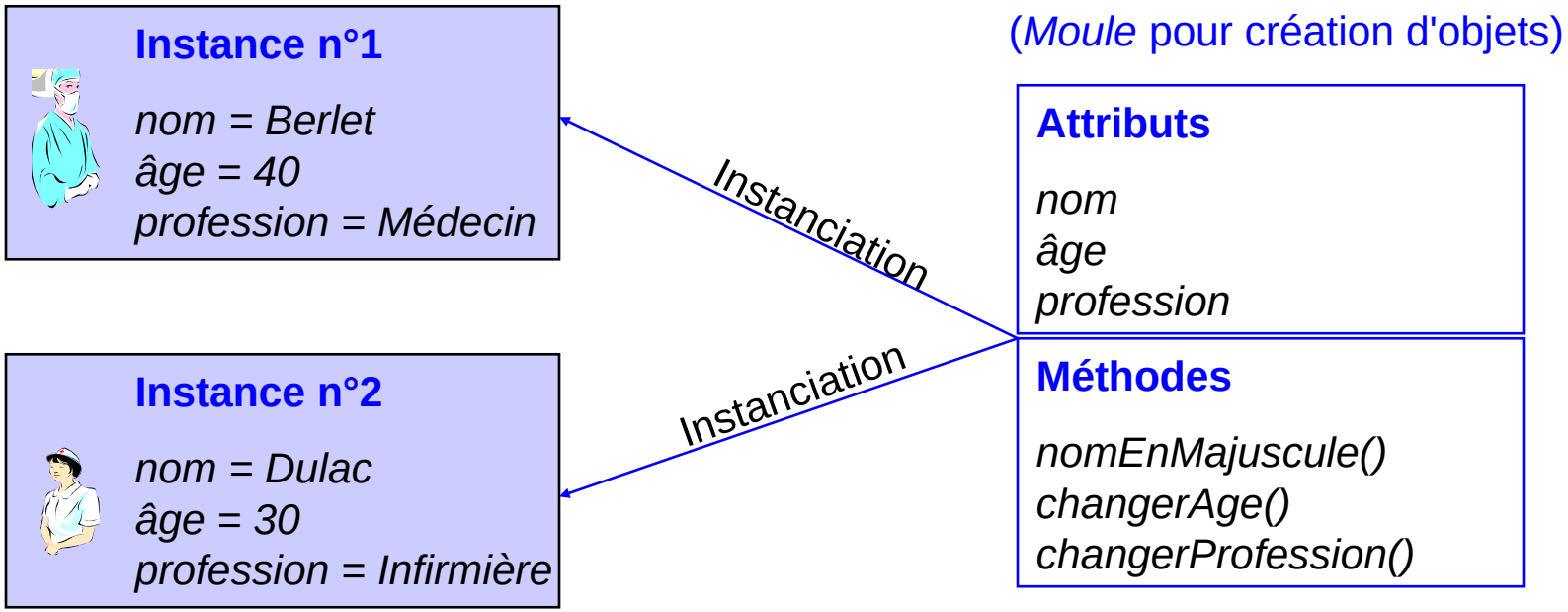


Objets



CONCEPT DE CLASSE

Tout objet est une instance d'une classe.



CONCEPT DE CLASSE

Une classe est donc une structure de données regroupant les données d'un programme et les moyens de traitement de ces données.

Une classe regroupe par conséquent deux éléments de la programmation structurée :

- les *attributs* (l'équivalent des variables en programmation structurée)

Tout comme n'importe quelle autre variable, un *attribut* peut posséder un type quelconque défini au préalable : nombre, caractère, ..., ou même un type Classe.

- les *méthodes* (l'équivalent des fonctions en programmation structurée)

Elles représentent les traitements applicables à l'objet ou à un de ses attributs ...

En conclusion

Une ***classe*** est donc un **type de données** servant à stocker des données dans des *attributs* et à les gérer au travers des *méthodes*.

Un **programme orienté objet** se présente sous la forme d'une **suite de définitions de classes**

PRINCIPES DE LA POO
(Encapsulation)

Trois principes :

- Encapsulation
- Héritage
- Polymorphisme

PRINCIPES DE LA POO
(Encapsulation)

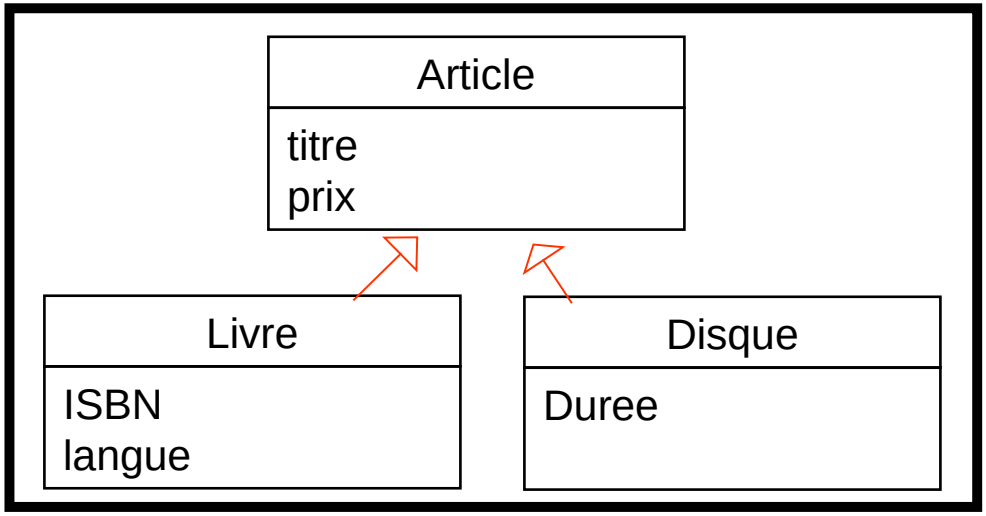
Grâce au concept d'objet, il ne s'agit plus de déclarer des variables générales puis un ensemble de fonctions destinées à les gérer de manière séparée, mais de regrouper le tout sous le couvert d'une et même entité. **Il s'agit d'une forme d'encapsulation**.

Cependant, on peut aller encore plus loin : on a la possibilité de **masquer aux yeux d'un programmeur extérieur les attributs et l'ensemble des méthodes destinées à la gestion *interne* de l'objet**, tout en laissant visibles d'autres attributs et méthodes.

L'encapsulation permet donc de garder une cohérence dans la gestion de l'objet, **tout en assurant l'intégrité des données qui ne pourront être accédées qu'au travers des méthodes visibles.**

PRINCIPES DE LA POO
(Héritage)

Le concept d'*héritage* permet de construire une classe à partir d'une (ou plusieurs) autre(s) classe(s) en partageant ses attributs et ses méthodes.



Grâce au concept *d'héritage* un objet *article* va pouvoir donner naissance à des *descendants* (par exemple *livre* et *disque*).

Tout comme un enfant *hérite* des caractéristiques de ses parents et développe les siennes, un objet (descendant) hérite des caractéristiques (attributs et méthodes) de son ancêtre, mais aussi peut en développer de nouvelles, ou bien encore se spécialiser. Ainsi, un *livre* est toujours considéré comme un *article*, il possède donc un *titre* et un *prix*, mais il dispose également de deux attributs spécifiques (*ISBN* et *langue*).

PRINCIPES DE LA POO
(Héritage)

Intérêts

Spécialisation, enrichissement :

une nouvelle classe réutilise les attributs et les méthodes d'une classe en y ajoutant des attributs et/ou des méthodes particulières

Réutilisation :

évite de réécrire du code existant ou de dupliquer du code.

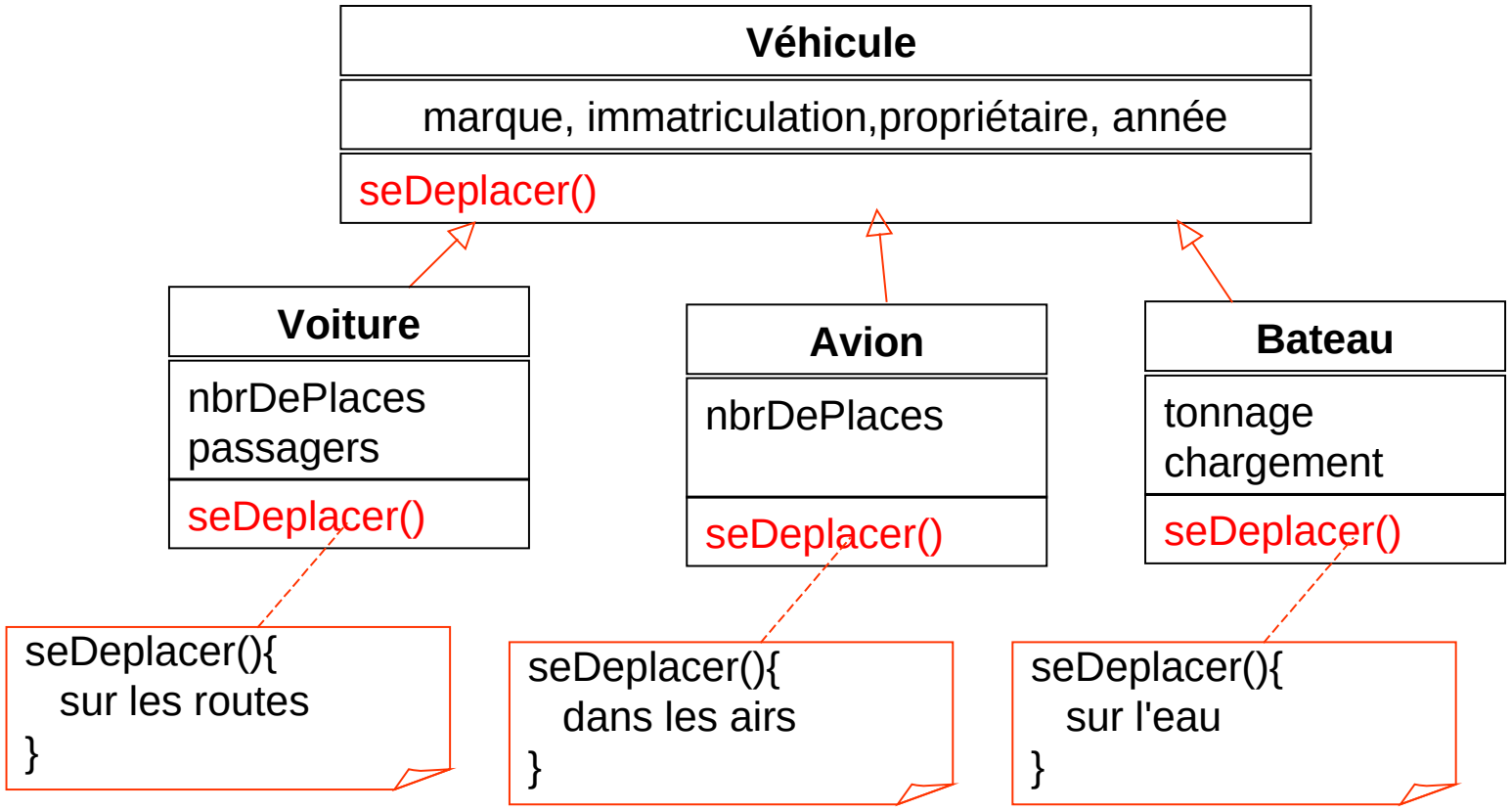
Redéfinition :

une nouvelle classe redéfinit les attributs et méthodes d'une classe de manière à en changer le sens et/ou le comportement pour le cas particulier défini par la nouvelle classe

Notez que bien souvent, on ne possède pas les sources de la classe à hériter

PRINCIPES DE LA POO
(Polymorphisme)

Commençons par analyser la structure du mot : *poly* comme *plusieurs* et *morphisme* comme *forme*. Le polymorphisme traite de la capacité de l'objet à posséder *plusieurs formes*. Cette capacité dérive directement du principe d'héritage vu précédemment (héritage des caractéristiques et redéfinition de méthodes).



PRINCIPES DE LA POO
(Polymorphisme)

Définitions

- Capacité d'une méthode à s'appliquer à des objets de classes différentes
- Capacité à choisir dynamiquement (en cours d'exécution) la méthode qui correspond au type (à la classe) réel de l'objet en cours

Intérêts

- Un code lisible
- Un code générique

Synthèse

Les concepts clés :

Objet

- Attribut
- Méthode

Classe

Les principes de la POO

- **Encapsulation**
- **Héritage**
- **Polymorphisme**