Ecole Supérieure
d'Informatique et du Numérique
COLLEGE OF ENGINEERING & ARCHITECTURE

Chap1: Introduction à la POO

POO: Concepts de base

Objectifs

- Ce chapitre présente les concepts de base de l'approche orientée objet.
- La maîtrise des notions présentées dans ce chapitre est essentielle pour une bonne compréhension de l'approche objet.
- Concepts théoriques indépendants de la technologie

Plan

I. L’approche orientée objet Vs approche fonctionnelle

II. Les classes, les objets et les messages

III. Abstraction

IV. Encapsulation

V. Héritage et Polymorphisme

I. Approche objet Vs Approche fonctionnelle

Approche fonctionnelle: diviser pour régner

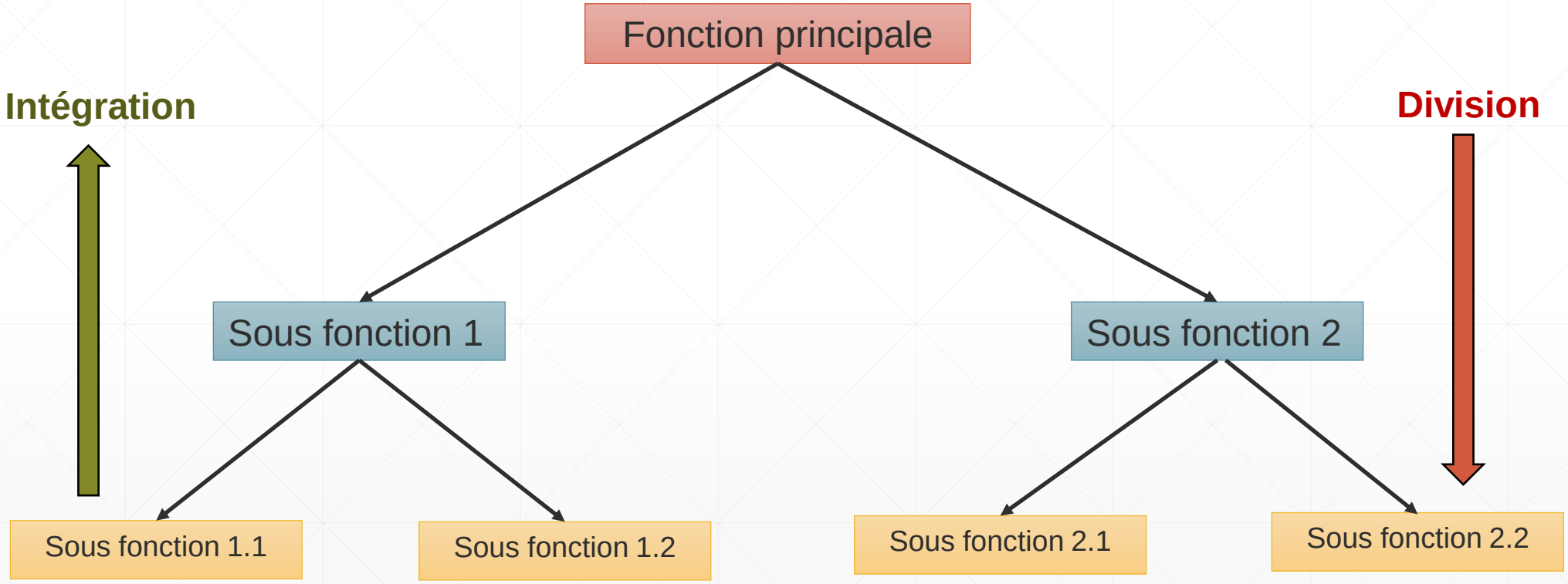
La construction du logiciel basée sur des itérations de type **division**/**réunion**:

- Diviser pour comprendre
- Réunir les différents composants pour construire le logiciel

Le processus de décomposition a été dirigé traditionnellement par **un critère fonctionnel**.

Les fonctions du système sont identifiées, puis décomposées en sous-fonctions, et cela **récursivement** jusqu'à l'obtention d'éléments simples, directement représentables dans les langages de programmation.

Approche fonctionnelle: diviser pour régner

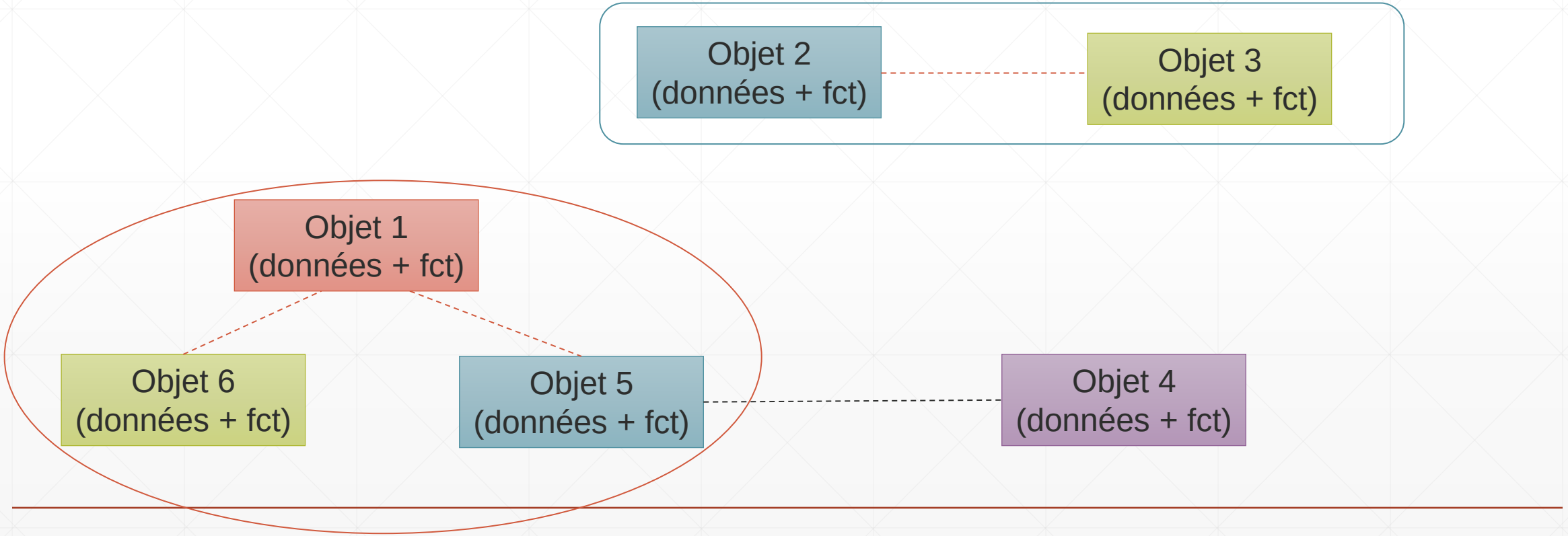


L’approche objet

- **Orienté objet** : organisation du logiciel comme une collection d’**objets** dissociés comprenant à la fois une **structure de données** et un **comportement (fonctions)**

L’approche objet

- Collaboration dynamique entre plusieurs types d’objets



Comparaison des deux approches

L’approche fonctionnelle:

- Mécanismes intégrateurs sont la fonction et la hiérarchie
- Apporte des résultats satisfaisants lorsque les fonctions sont bien identifiées dans le temps
- Implique des modifications **structurelles lourdes** suite à une évolution fonctionnelle à cause du couplage fort statique entre l’architecture et la fonction

L’approche objet:

- Propose une méthode de décomposition du système étudié, non pas basée uniquement sur ce que le système fait, mais plutôt sur l'intégration de ce que le système **est** et **fait**.

Pourquoi l’approche objet

- La **stabilité** de la modélisation par rapport aux entités du monde réel
- La **réutilisation** des éléments dans d’autres programmes

Pourquoi l’approche objet

La simplicité du modèle qui ne fait appel qu’à cinq concepts fondateurs :

- les classes,
- les objets
- l’abstraction,
- l’héritage,
- le polymorphisme.

Pour exprimer de manière uniforme l’analyse, la conception et la réalisation d’une application informatique.

II. Objet et Classe

Les objets

- Les objets définissent une représentation simplifiée des entités du monde réel.
- **Par exemple:** êtres vivants (Etudiants, Profs), des entités abstraites (Compte bancaire)

- **Les objets dans le monde réel Vs POO:**
  - Ils naissent, vivent et meurent;
  - Ils sont alloués, changent d'états et se comportent en conséquence, et sont désalloués.

Les objets: Définition

❖L’objet est une unité atomique formée de l'union d'un **état** et d'un **comportement**.

Définition :

Un objet est une structure de données valorisées et qui répond à un ensemble de messages. Cette structure de données définit son **état** tandis que les messages qu'il comprend décrit son **comportement**

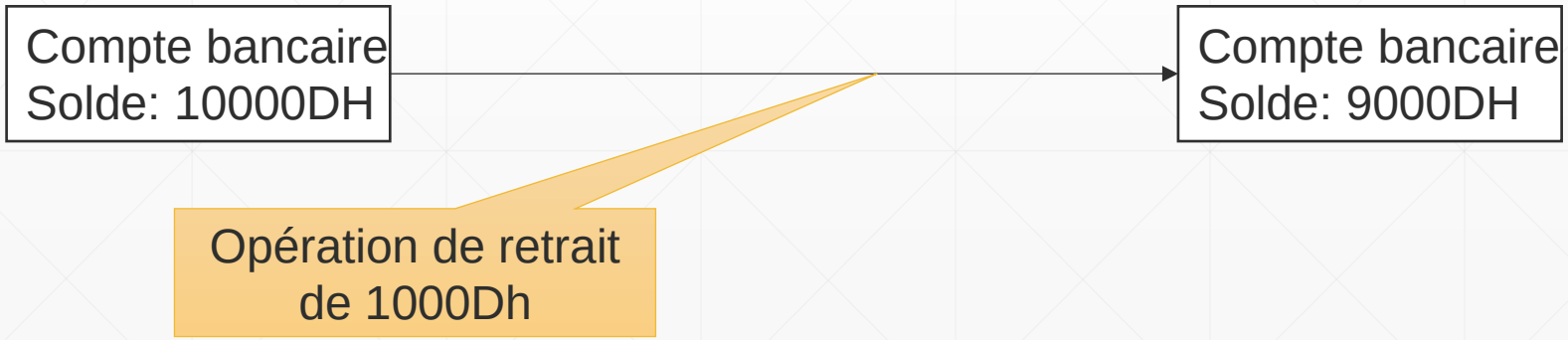
Un objet = état + comportement + identité(JAVA)

L’état d’un objet

Objet: etd1

- **CNE**: 200411234
- **Nom**: Durant
- **Age**:20
- **Filière**: ing3/GI

- L’état regroupe les valeurs instantanées de tous les attributs
- Chaque attribut peut prendre une valeur dans un domaine de définition donné.
- L'état évolue au cours du temps
- En règle générale, l'état d'un objet peut être vu comme la conséquence de ses comportements passés.



Relation entre l’état et le comportement d’un objet (Nouveau)

- L’état et le comportement sont liés; en effet, le comportement à un instant donné dépend de l'état courant, et l'état peut être modifié par le comportement.

- Par exemple Il n’est possible de faire atterrir un avion que s’il est en train de voler, c'est-à-dire que le comportement **Atterrir()** n’est valide que si l'information **En Vol** est valide.

Attention (Approche fonctionnelle)

Fct: avg (int nbr1, int br2)
int age = 20, int section=2
avg(age, section)

La classe

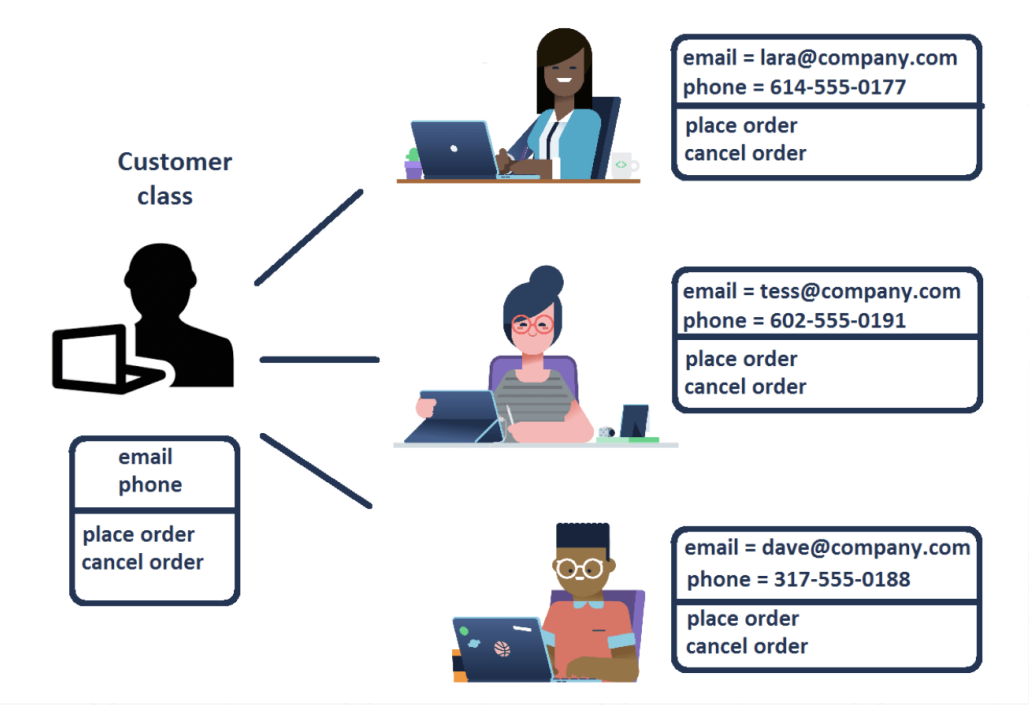
Une **classe** sert à regrouper sous un même terme générique les objets partageant la même structure de données et le même comportement. On dit qu'un objet est **une instance** d'une classe.

- **La classe est séparée en deux parties :**
  - la spécification de la classe qui décrit le domaine de définition et les propriétés des instances de la classe
  - la réalisation de la classe qui décrit comment la spécification est réalisée

La classe

- La classe est une description abstraite d’un ensemble d’objets
- La classe peut être vue comme la factorisation des éléments communs à un ensemble d’objets
- La classe décrit le domaine de définition d’un ensemble d’objets
- Du point de vue de la programmation objet, une classe est un type structuré de données.

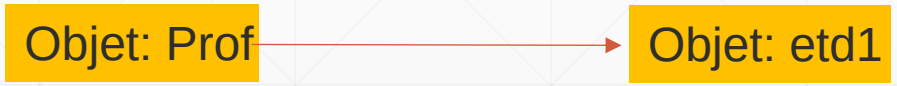
La classe

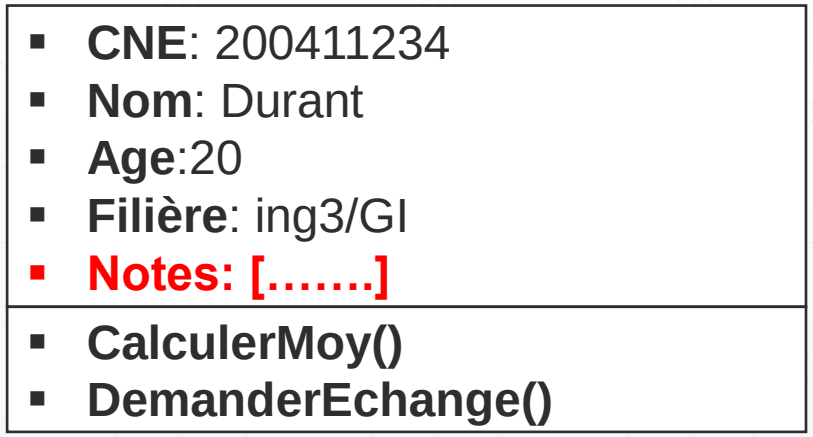

Customer class
email
phone
place order
cancel order
email = lara@company.com
phone = 614-555-0177
place order
cancel order
email = tess@company.com
phone = 602-555-0191
place order
cancel order
email = dave@company.com
phone = 317-555-0188
place order
cancel order


Class
Objects
Rotary Phone
Touch Tone Phone
Cellular Phone

Le comportement

- Le comportement regroupe toutes les compétences d'un objet et décrit les actions et les réactions de cet objet. Chaque atome de comportement est appelé méthode.
- Les méthodes d'un objet sont déclenchées suite à une stimulation externe, représentée sous la forme d'un message envoyé par un autre objet.





III. Abstraction

Abstraction

- L'abstraction consiste à **concentrer la réflexion** sur un élément d'une représentation ou d'une notion, en portant spécialement l'attention sur un ensemble de ses caractéristiques et en négligeant tous les autres.

- Elle se base sur l'identification des **caractéristiques communes** à un ensemble d'éléments, puis de la description condensée de ces caractéristiques dans ce qu'il est convenu d'appeler une classe.

- Elle est arbitraire et se définit par rapport à un point de vue. Ainsi, un objet du monde réel peut être vu au travers d'abstractions différentes, il faut déterminer que sont les critères importants

Abstraction par l'exemple

Par exemple, considérons l'abstraction d'une **personne**:

- Du point de vue **université**: le nom, l'adresse, N° de sécurité sociale et le cursus universitaire.

- Du point de vue **police**: le nom, l'adresse, la taille, le poids, couleur des cheveux, couleur des yeux.

Abstraction

- L’abstraction est une phase d’analyse qui se préoccupe de préciser qu’est-ce que une classe **est** et qu’est-ce qu’elle **sait faire** dans le cadre du domaine analysé.

- L’abstraction détermine les méthodes et les attributs qui sont d’intérêt pour l’application à développer et pour ignorer le reste.

- Les systèmes OO procèdent seulement à l’abstraction de ce qu’ils ont besoins pour résoudre le problème étudié.

IV. Encapsulation

Encapsulation

- L'encapsulation est une issue de conception qui traite la façon dont les fonctionnalités sont **compartimentées** dans un système.

- Vous ne devriez pas devoir savoir comment une chose est implémenté pour pouvoir l'utiliser.

- Vous pouvez construire n'importe quoi de la façon que vous voulez, et alors vous puissiez plus tard changer l'implémentation sans affecter d'autres composants dans le système

Encapsulation

L’encapsulation présente un double avantage:

- Les données encapsulées dans les objets sont **protégées** des accès intempestifs;
- Les utilisateurs d’une abstraction ne dépendent pas de la réalisation de l’abstraction mais seulement de sa spécification.

Encapsulation (Exemple 1)

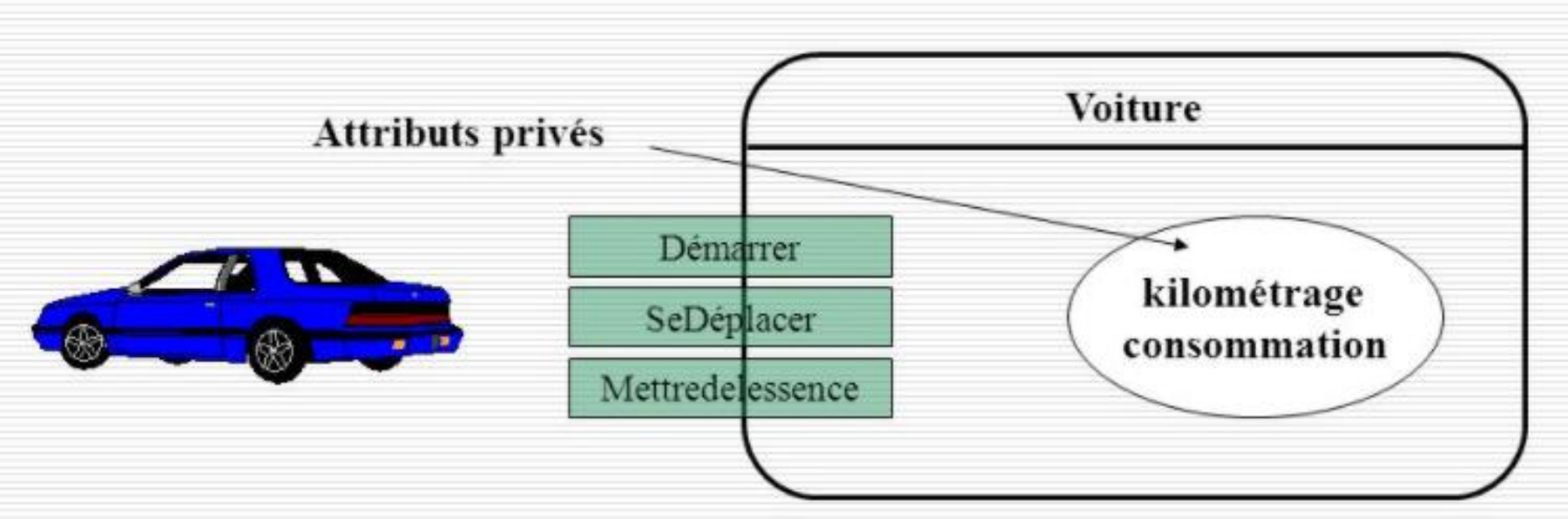
Considérez votre banque:

- Comment maintient-elle votre information de compte, sur une unité centrale, un mini-ordinateur, ou un PC?
- Quelle base de données utilise-t-elle?
- Quel système d'exploitation?

En cachant les détails de l'implémentation des comptes, votre banque est libre pour changer cette implémentation à tout moment, et cela ne devrait pas affecter la manière dont ses services vous sont fournis.

Encapsulation (Exemple 2)

- Un autre exemple, Objet voiture



V. Héritage et polymorphisme

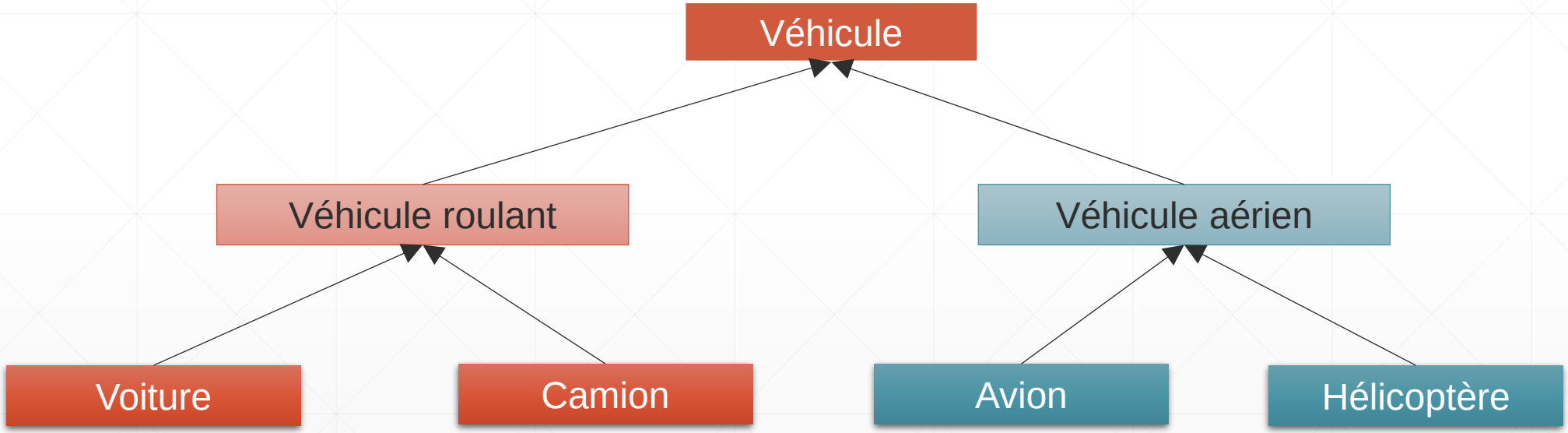
Héritage

- Les similitudes existent souvent entre différentes classes. Deux classes ou plus partagent souvent les mêmes attributs et/ou les mêmes méthodes.

- L’héritage modélise les associations de type "est", "est une sorte de", et " est comme " , il vous permet de réutiliser des données et le code existants facilement.

Héritage

- Exemple:



Polymorphisme

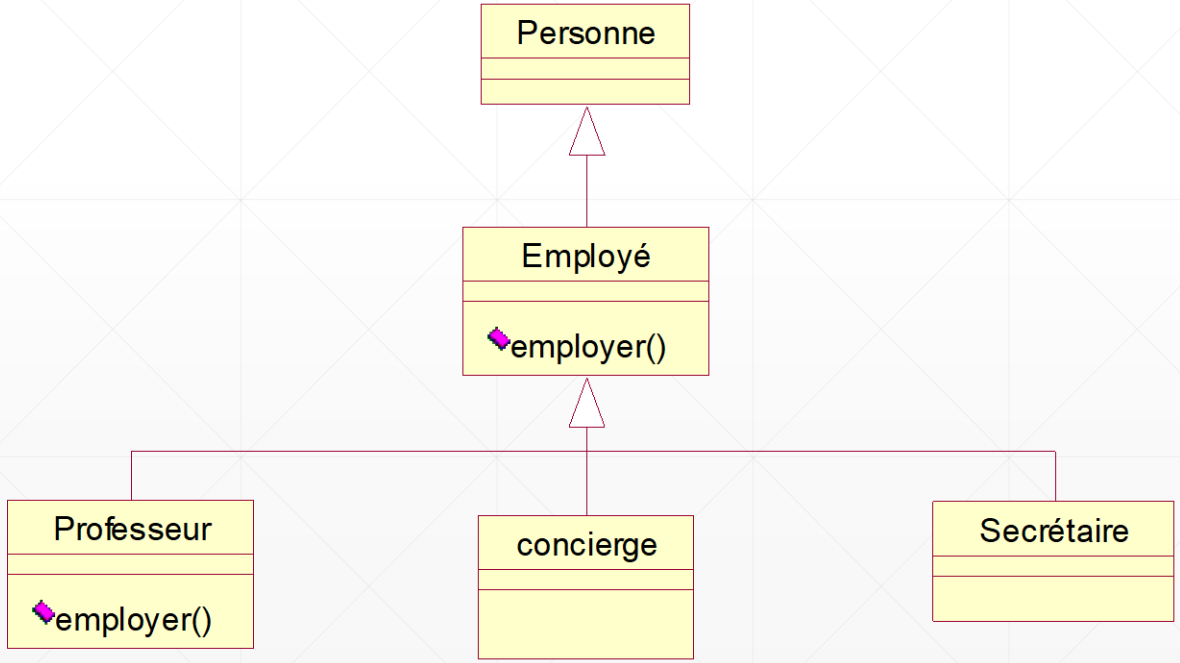
- Le terme polymorphisme décrit la caractéristique d'un élément qui peut prendre plusieurs formes.

- En informatique, le polymorphisme désigne un concept **de la théorie des types**, selon lequel un nom d'objet peut désigner des instances de classes différentes issues d'une même arborescence.

- Le terme polymorphisme désigne **le polymorphisme de méthode**, c'est-à-dire la possibilité de déclencher des méthodes différentes en réponse à un même message.

Polymorphisme

- Considérons un exemple plus réaliste du polymorphisme, en explorant comment l'université gère l'affectation de son personnel, modélisé dans la figure suivante



Polymorphisme

- Le polymorphisme vous permet d'envoyer le message employer() à n'importe qui (professeur concierge secrétaire). Vous n'avez pas à compliquer le code avec des bloc de if ou case.

- Comme résultat vous pouvez ajouter d'autre type d'employer par exemple administrateur et adapter la manière dont il est affecté à l'université.

- Le polymorphisme est donc un moyen pour **étendre** votre système facilement

Merci de votre attention