Programmation Orienté Objet (POO)

Présenté par : M. Bouderbala

Promotion : 2ème Année LMD Informatique / Semestre N°4

Etablissement : Université de Relizane

Année Universitaire : 2021/2022

A propos

- Ce cours est une **introduction** à la programmation orientée objet.
- **Pré-requis**: Le cours d'algorithmique et structures de données
- **Java** sera utilisé comme exemple de langage oriente objet pour illustrer les notions vues en cours.
- **Eclipse** sera utilisé comme environnement de développement.

Objectifs

Introduire l'approche orientée objet, A l'issue du cours :

- Vous aurez pris conscience de l'importance d'appliquer les principes de l'oriente objet (**Abstraction, encapsulation, héritage, polymorphisme..etc**.).
- Vous serez capables de concevoir une petite application en utilisant l'approche orientée objet.
- Vous serez capable de l'implémenter en java.

Contrôles & Evaluations

- Un examen final
- Un TP test

Contenu du cour

- **Chapitre 1 : Introduction à la Programmation Orienté Objet**
  - Notions de base
  - Historique
  - Utilisation des TAD
- **Chapitre 2 : Les classes**
  - Déclaration des classes
  - Les constructeurs et destructeurs
  - Les méthodes d'accès
  - Encapsulation

Contenu du cour

- **Chapitre 3 Héritage et polymorphisme**
  - Généralités
  - Surcharge et redéfinition
  - Héritage : Références
  - Polymorphisme
  - Les classes abstraites
- **Chapitre 4 Interface et implémentation**
  - Principe
  - Application

Contenu du cour

- **Chapitre 5 Interface graphique et Applet**
  - Composants, gestionnaire d'affichage
  - Mise en page
  - Gestion des événements et écouteur
  - Applet

Documentations

- **Il existe de nombreux livres sur java…**
  - Claude Delannoy, « Programmer en Java », Eyrolles
  http://java.sun.com/docs/books/jls/
  - *Thinking in Java by Bruce Eckel*
  http://www.ibiblio.org/pub/docs/books/eckel/
  - Lemay L, « Le Programmeur JAVA 2 », Campus Press.
  - Horstmann et Cornell, « Au coeur de Java 2 Volume I - Notions fondamentales »,
  - ***The Java Programming language fourth edition***
  AW Ken Arnold, James Gosling, David Holmes

CHAPITRE 1

- Introduction à la programmation orientée objet

Mise en œuvre d'une application (1)

- Plusieurs étapes sont nécessaires pour mettre en œuvre une application :

Etape 1 :Définition du problème
Etape 2 :Analyse du problème
Etape 3 :Concevoir une solution au problème
Etape 4 :Implémenter la solution (Programmer)
Etape 5 :Mise au point du programme (Tester la solution)

Mise en œuvre d'une application (2)

- Ceci doit être fait en considérant:
  - La maintenance du logiciel
  - l'évolution du logiciel
  - la réutilisation du logiciel

Paradigmes de Programmation (1)

- **Paradigme :**
  - "Modèle théorique de pensée qui oriente la recherche et la réflexion scientifiques" (Larousse).
  - Représentation, vision du monde, modèle, courant de pensées.
  - Manière d'utiliser des techniques et des exemples propres à chaque problème.

Paradigmes de Programmation (2)

- **Programmation:**

  "Ensemble des activités qui permettent l'écriture des programmes informatiques. Elle représente usuellement le codage, c'est à dire la rédaction du code source d'un logiciel."

- **Langage :**

  "Une abstraction des opérations réalisée par un ordinateur. Un langage est habituellement domine par un paradigme a partir duquel sa structure a été conçue. Souvent, des éléments issus de différents paradigmes y sont présents."

Paradigmes de Programmation (3)

- **Programmer:**

  "Utiliser un langage afin de se conformer de façon plus ou moins rigoureuse a un ou plusieurs paradigmes de programmation."

Ex:

- ✓ Le language **Java** supporte la programmation orientée objet,
- ✓ **C++** est un langage de programmation compilé, permettant la programmation sous de multiples paradigmes comme la programmation procédurale, la programmation orientée objet et la programmation générique.

Paradigmes de Programmation (4)

- Il n'existe pas de paradigmes uniques bon pour toutes les applications.
- Tout problème peut etre résolu en se basant sur l'un ou l'autre des paradigmes de programmation

Paradigme de programmation (5)

Un paradigme de programmation:

- Un paradigme de programmation est un style fondamental de programmation informatique qui traite de la manière dont les solutions aux problèmes doivent être formulées dans un langage de programmation (à comparer à la méthodologie, qui est une manière de résoudre des problèmes spécifiques de génie logiciel).

Paradigme de programmation (6)

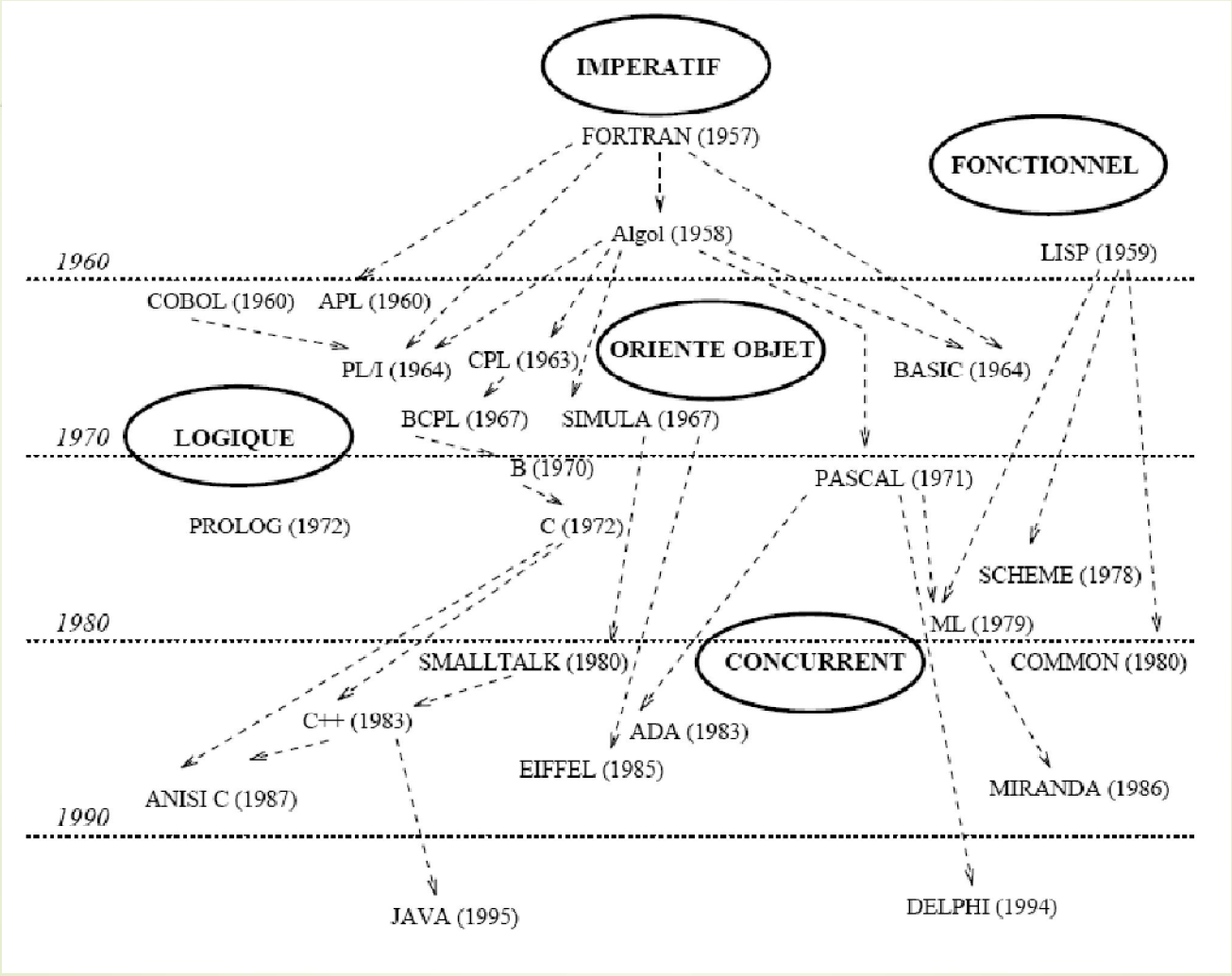
Un paradigme de programmation:

- Une façon d'aborder un problème.
- Une manière de penser.
- Concrétisation d'une philosophie de programmation.
- Un style de programmation.

Paradigmes des langages de programmation

- Il existe 4 principaux paradigmes de programmation:
- La programmation **impérative** (procédurale): Pascal, C, Fortran. (programme = algorithme + structure de données).
- La programmation **fonctionnelle** : (adopte une approche beaucoup plus mathématique de la programmation) Lisp, Scheme, Haskell.
- La programmation **logique** : (la description d'un programme est sous forme de prédicats) Prolog, GHC.
- La programmation **Orientée Objet** : Smalltalk,C++, Java, C#

Historique des Paradigmes des langages de programmation



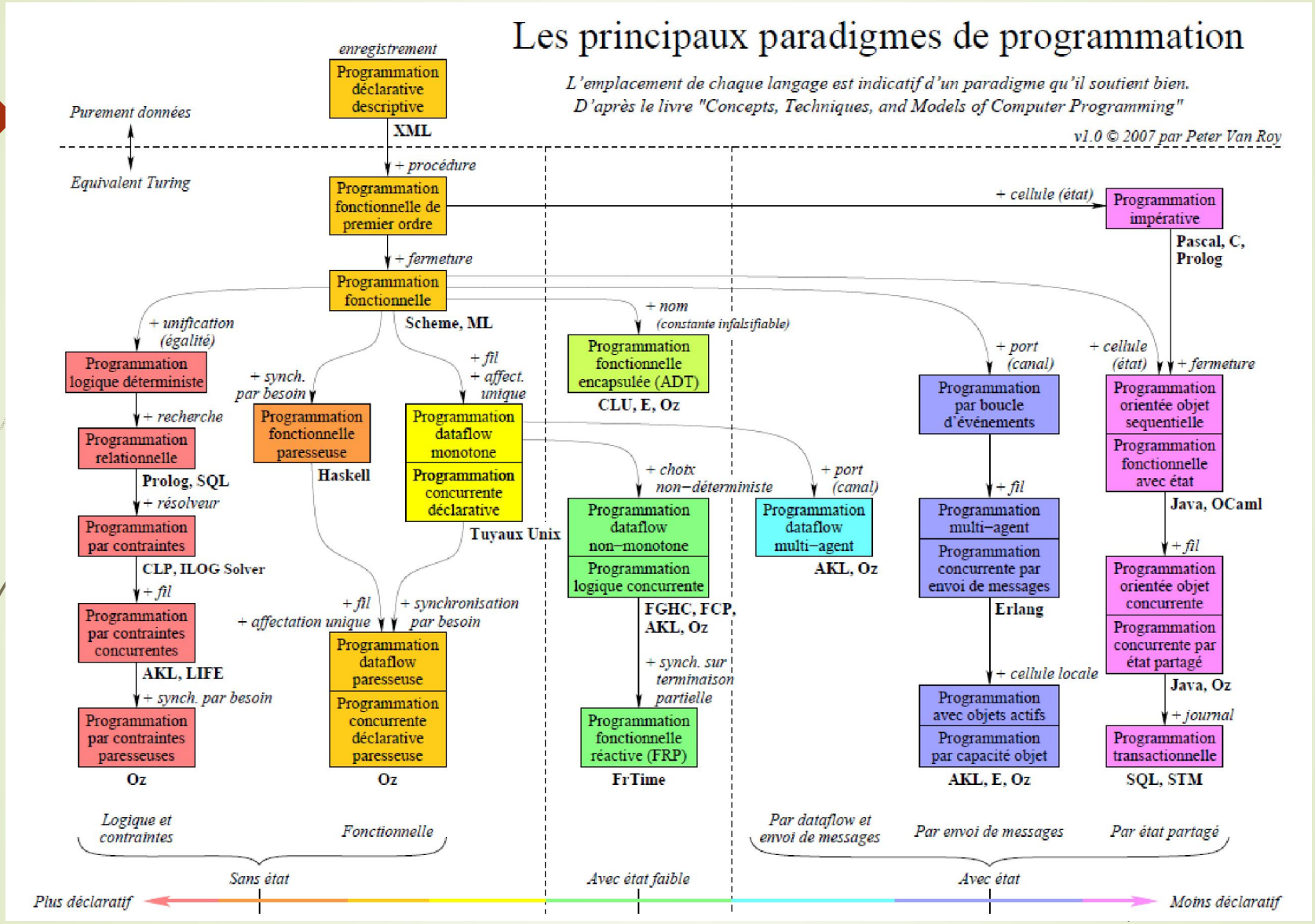
Autres Paradigmes

- programmation par scripting/dynamique
- programmation déclarative
- programmation orientée aspect
- Programmation concurrente
- et plus…

Les principaux paradigmes de programmation

L'emplacement de chaque langage est indicatif d'un paradigme qu'il soutient bien.
D'après le livre "Concepts, Techniques, and Models of Computer Programming"

v1.0 © 2007 par Peter Van Roy



Pourquoi étudier différents paradigmes?

- Afin de mieux exprimer des idées complexes
- Différentes tâches demandent des approches différentes
- Afin de faciliter l'apprentissage de nouveaux langages
- Afin de faire les bons choix
- Afin de suivre l'évolution de l'état de l'art en programmation

Programmation impérative VS POO (1)

Comment avez-vous l'habitude de programmer?
Quelle est la structure de votre code?

- ***Représentation procédurale:***
  Séparer les données des traitements.
  Subdivision basée sur les traitements.
- ***Représentation Objet:***
  Manipuler un ensemble d'objets qui interagissent entre eux.
  Subdivision basée sur les données.

Programmation impérative VS POO (2)

Dans la programmation procédurale

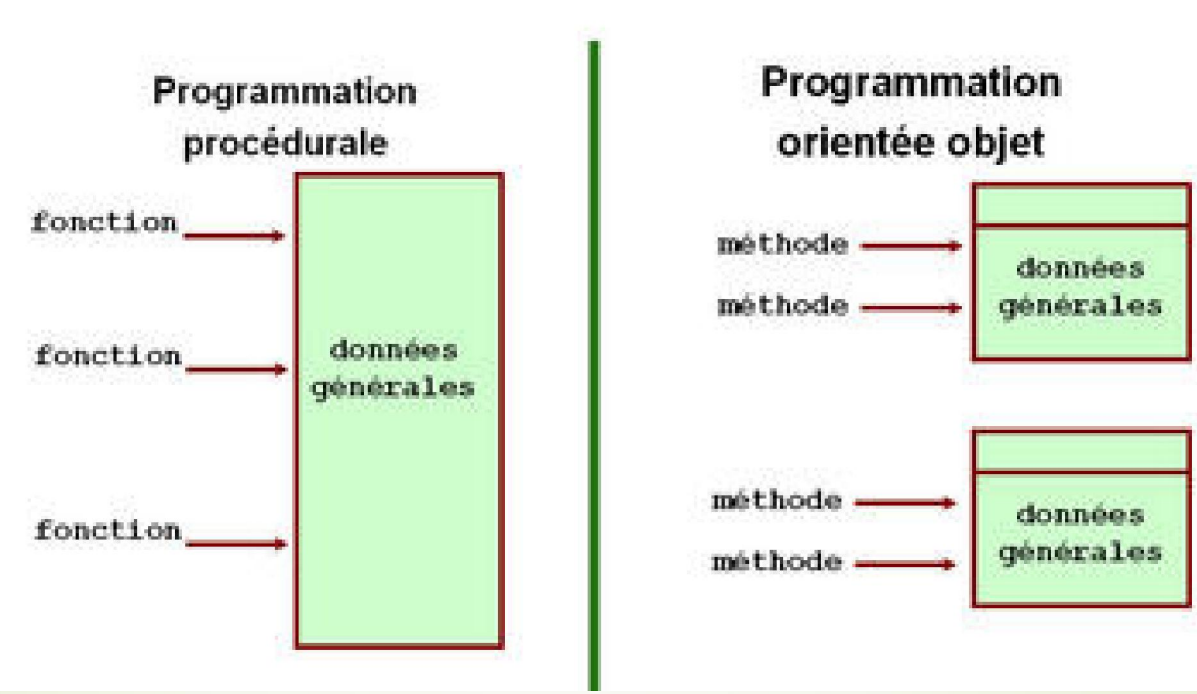
- On apprend à subdiviser un problème en une série d'étapes simples (des fonctions) permettant de résoudre un problème.
- D'abord, on décide de la manière dont on va manipuler les données, ensuite le type de structures de données les plus appropriées pour faciliter cette manipulation.
- Donne des logiciels difficiles à maintenir et à réutiliser

Programmation impérative VS POO (3)

Dans La programmation orientée objet

- Un programme informatique est considéré comme étant un ensemble d'objets fonctionnant ensemble pour effectuer une tâche.
- On s'intéresse d'abord aux données, avant de déterminer l'algorithme qui sera utilisé pour opérer sur ces données.
- Les objets coopèrent par envois de messages (Appel de méthodes)

Programmation impérative VS POO (4)



La programmation orienté objet

Définition:

- La **programmation orientée objet (POO)**, ou **programmation par objet**, est un paradigme de programmation informatique élaboré par les Norvégiens *Ole-Johan Dahl* et *Kristen Nygaard* au début des années 1960 et poursuivi par les travaux d'*Alan Kay* dans les années 1970.
- Il consiste en la définition et l'interaction de briques logicielles appelées *objets* ; un objet représente un concept, une idée ou toute entité du monde physique, comme une voiture.
- Il possède une structure interne et un comportement, et il sait interagir avec ses pairs. Il s'agit donc de représenter ces objets et leurs relations ;
- l'interaction entre les objets via leurs relations permet de concevoir et réaliser les fonctionnalités attendues, de mieux résoudre le ou les problèmes. Dès lors, l'étape de modélisation revêt une importance majeure et nécessaire pour la POO. C'est elle qui permet de transcrire les éléments du réel sous forme virtuelle.

La programmation orienté objet

- ***Définition:***

La POO est une méthode d'implémentation dans laquelle les programmes sont organisés sous formes de collections coopératives d'objets, dont chacun représente une instance d'une classe quelconque et dont toutes les classes sont membres d'une hiérarchie de classes unis à travers des relations d'héritage.

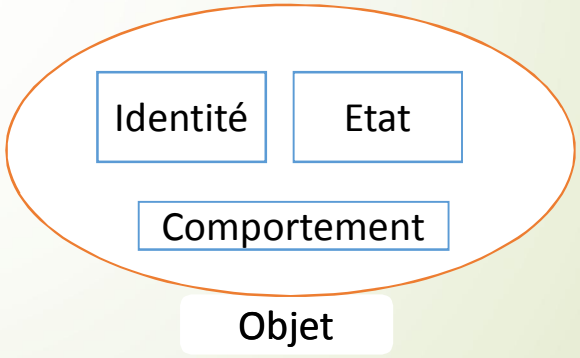
Algorithmique Objet

- Un algorithme dans l'approche orienté objet sera essentiellement vu comme un ensemble d'objets auxquels l'utilisateur envoie des messages et qui s'en envoient pendent le fonctionnement.
- Ces objets seront toujours pour l'utilisateur des boites noires et qui contiendront des variables locales, inconnues de l'environnement, et qui ne s'y intéressera d'ailleurs pas. Le seul moyen d'accéder à ces objets sera l'envoie des messages qu'ils sont capables de comprendre.

C'est quoi un Objet ? (1)

- L'objet est une entité logicielle qui représente la brique de base de la POO L'objet se caractérise par :

- ✓ Une identité (référence ou handle)
- ✓ Un état.
- ✓ Un comportement.



C’est quoi un Objet ? (2)

- Dans le monde réel, un objet peut être: un étudiant, un enseignant, un téléphone, une voiture, etc.

- **Exemple:** Une voiture
  • Identité = Nom, Marque, type, nombre de vitesses, Année, couleur,…
  • comportement = rouler, tourner, accélérer, changer de vitesse, freiner,…

C’est quoi un Objet ? (3)

Dans la POO un objet est un mélange de variables et de fonctions ou procédures.

- **L’objet :**
  • L’objet possède une **identité** unique qui le distingue des autres objets indépendamment de son **état** et de son **comportement**.
  • maintient son **état dans des variables appelées attributs ou champs.**
  • implémente son **comportement** à l'aide de fonctions ou procédures appelées **méthodes.**

- **Remarque:** Deux objets peuvent avoir le même état et le même comportement mais ont toujours des identités différentes, et réciproquement, le comportement et l’état d’un objet peuvent changer durant l’exécution sans que cela affecte son identité.

Cycle de vie d'un objet

Un objet en programmation orienté objet (POO) a un cycle de vie de 3 phases :

1. Construction (en mémoire).
2. Utilisation (changements d'état par affectations, comportements par exécution de méthodes)
3. Destruction

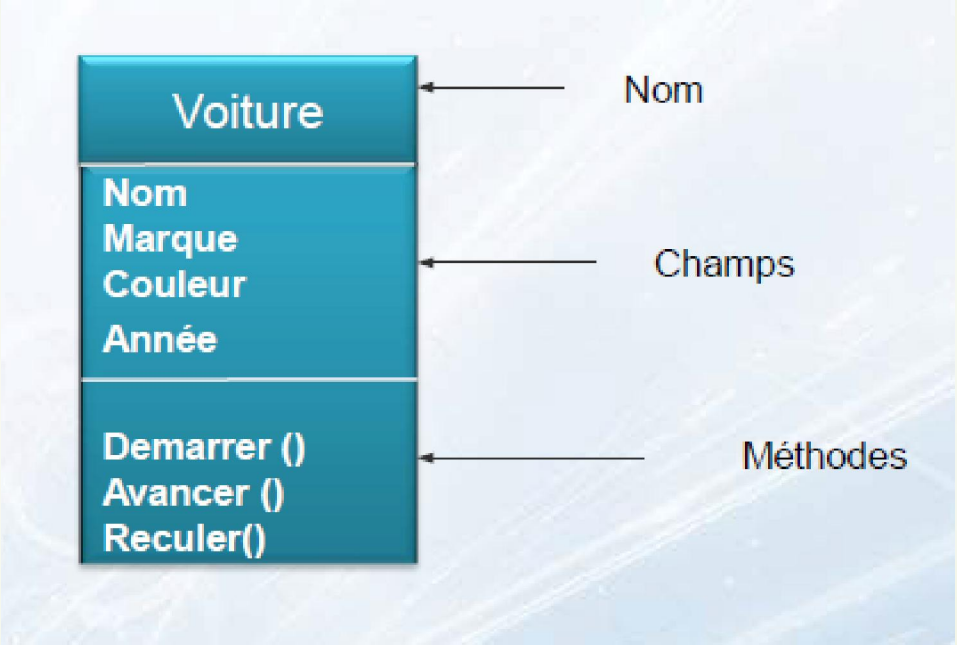
La Classe

La classe est une description d'une famille d'objets ayant une même structure et un même comportement.

Elle est caractérisée par :

- Un nom
- Une composante statique : des champs (ou attributs) nommés ayant une valeur. Ils caractérisent l'état des objets pendant l'exécution du programme.
- Une composante dynamique : des méthodes représentant le comportement des objets de cette classe. Elles manipulent les champs des objets et caractérisent les actions pouvant être effectuées par les objets.

La Classe : Représentation graphique



Classe et objet (1)

- La classe est la **machin**e qui fabrique les objets. Elle peut être considérée comme un **moule** ou un **modèle** à partir duquel on peut créer des objets.
- Une classe est un **modèle** de définition pour des objets partageant des **caractéristiques communes**, mais l'**état** de chaque objet est indépendant des autres
- Des objets créés à partir de la même classe auront des aspects semblables (mêmes attributs et même méthodes).
- Un **objet d'une classe** est aussi appelé ***instance*** de cette classe.

Classe et objet (2)

- Exemple:

La classe

Les objets (instances)

Classe et objet (3)

- Les objets V1,V2 ont les mêmes valeurs d'attributs, on dit qu'ils sont égaux car ils ont le même *état*. Ce sont néanmoins deux objets distincts car ils ont des *identités* différentes.
- Les objets d'une même classe ont **toujours** une identité distincte et **généralement** un état distinct.

Concepts de base de la POO

La Programmation orienté Objet (POO) est basée sur les concepts suivants:

- L'encapsulation
- L'héritage
- Le polymorphisme

L'encapsulation (1)

- L'encapsulation est un principe clé dans la POO.
- Elle consiste à regrouper des données (attributs) et un comportement (méthodes) dans une même classe et à réglementer l'accès aux données de l'extérieur (par d'autres objets).
- Cela signifie qu'il n'est pas possible d'agir directement sur les données d'un objet ; il est nécessaire de passer par ses méthodes.
- L'appel de la méthode d'un objet est aussi appelé *envoi de message* à l'objet.

L'encapsulation (2)

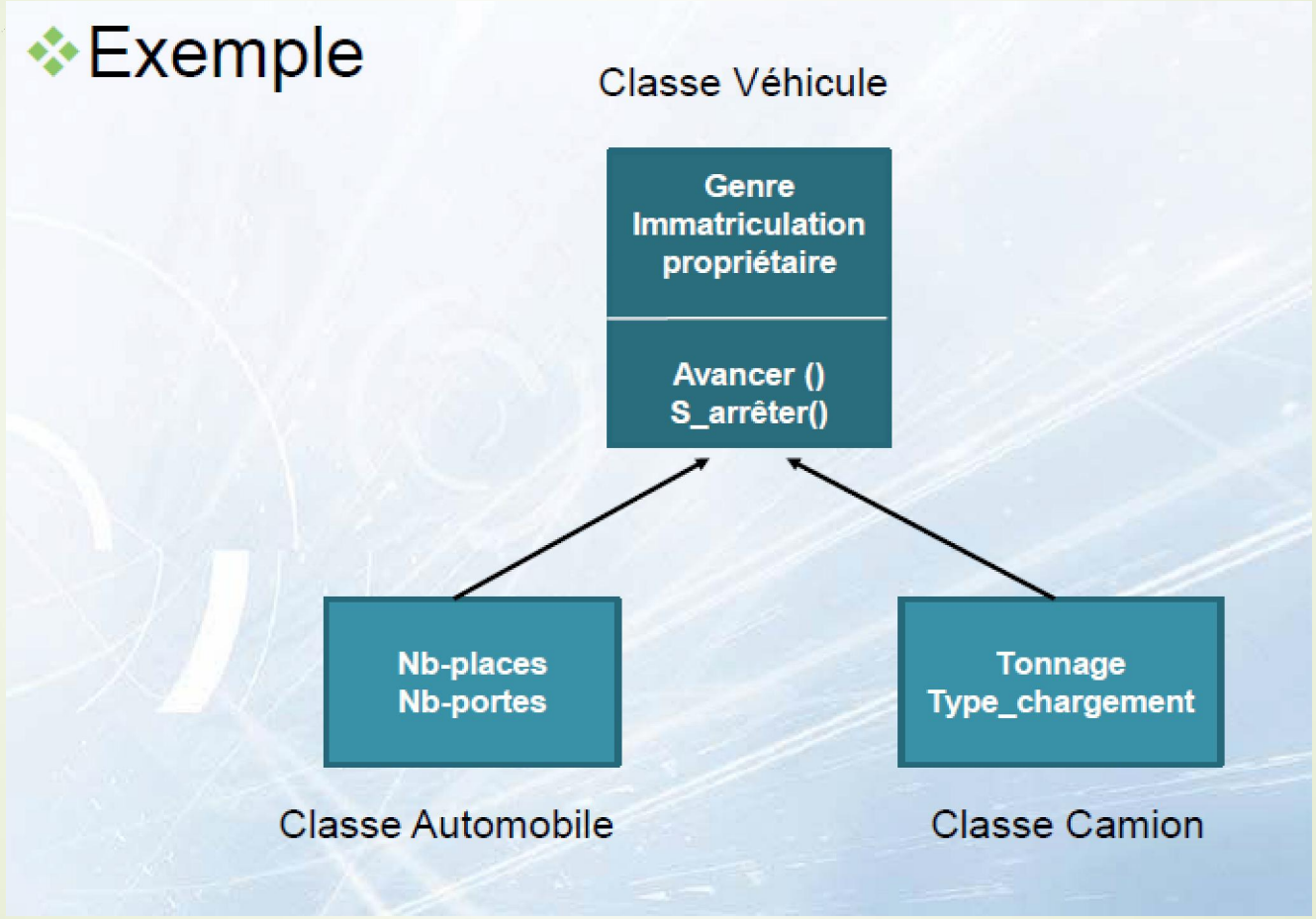
- Le principe de l'encapsulation est que, vu de l'extérieur, un objet se caractérise uniquement par ses méthodes visibles appelées interface. Les données, elles, restent invisibles et inaccessibles.
- Ce principe permet de concevoir des programmes (logiciels) plus sûrs, plus lisibles et plus faciles à maintenir, car une modification de la structure des données d'un objet ne se répercute pas sur les objets qui communiquent avec lui (invoquent ses méthodes).

L'héritage (1)

- L'héritage est un autre concept important en POO.
- Il permet de définir une (ou plusieurs) nouvelle classe (appelée *classe dérivée, classe fille* ou *sousclasse*) à partir d'une classe existante (appelée *classe de base, classe mère* ou *super-classe*), à laquelle on ajoute de nouvelles données et de nouvelles méthodes.
- L'héritage facilite largement la réutilisation de produits existants, ce qui est un avantage important de la POO.

L'héritage (1)

- Exemple



Le Polymorphisme (1)

- Le terme polymorphisme issu du grec signifie la faculté de prendre plusieurs formes.
- C'est un concept puissant de la POO qui vient compléter celui de l'héritage.
- En POO, cela peut s'appliquer aussi bien aux objets qu'aux méthodes.

Le Polymorphisme (2)

Polymorphisme d'objets:

- Il permet une certaine flexibilité dans la manipulation des objets en exploitant la relation d'héritage entre les classes.
- Il applique la règle suivante :

Si la classe A est dérivée de la classe B
alors un objet de la classe A peut aussi être considéré comme étant un objet de la classe B.

Le Polymorphisme (3)

Polymorphisme de méthodes

Il exprime le fait que :

- Des méthodes d'une même classe puissent avoir le même nom et des signatures différentes (ceci est appelé ***surcharge*** ou ***surdéfinition*** de méthodes).
- La même méthode peut se comporter différemment sur différentes classes de la hiérarchie. Autrement dit, des méthodes peuvent avoir des noms similaires dans une hiérarchie de classes et être différentes. Ceci est appelé ***redéfinition*** ou ***spécialisation*** de méthodes.

Le Polymorphisme (4)

- Exemple

Quel type de polymorphisme?