Méthodes d'analyse et de conception : Partie I : UML

Pr. A. DARGHAM
ENSAK
GI / S3 / Année 2018/2019

Chapitre 2 : Les concepts de l'OO

Sommaire

Le concept d'objet

- **La notion d'objet :**
  - *Un objet est une **entité** possédant certaines **propriétés** qui la caractérisent.*
  - *Il peut être **concret** (une personne, un livre, un ordinateur, une voiture, une machine à lavé, …etc) ou **abstrait** (une facture, une équation mathématique, un logiciel, …etc).*
  - *Certaines propriétés d'un objet caractérisent son **état** : ce sont des **attributs**.*
  - *Certaines propriétés d'un objet caractérisent son **comportement** : ce sont des **méthodes**.*

Le concept d'objet

- **La notion d'attribut :**
  - *Un **attribut** d'un objet indique l'état de ce dernier. Il représente un **élément de structure** d'un objet.*
  - *Un attribut possède un **nom** et une **valeur**.*
- **Exemples d'attributs :**
  - *Attributs possibles d'un objet « Personne » : le **poids**, la **taille** et **l'âge**.*
  - *Attributs possibles d'un objet « Voiture » : la **marque**, le **modèle**, la **puissance fiscale** et la **couleur**.*

Le concept d'objet

- **Domaines d'attributs :**
  - *L'**ensemble des valeurs possibles** d'un attribut est appelé le **domaine de cet attribut**.*
  - *À un moment donné, un attribut d'un objet ne peut avoir qu'une seule valeur appartenant au domaine de cet attribut.*
- **Exemples :**
  - *Un objet « Personne » de poids = **75Kg**, de taille = **1m65** et d'âge = **30 ans**.*
  - *Un objet « Voiture » de marque **Ford**, de modèle **Fiesta**, de PF **6ch** et de couleur **bleu**.*

Le concept d'objet

- **La notion de méthode :**
  - *Une **méthode** d'un objet indique le comportement de ce dernier.*
  - *Une **méthode** est une **opération** qu'un objet peut **effectuer** ou **subir**.*
  - *Une **méthode** est décrite par un **verbe**.*
- **Exemples de méthodes :**
  - *Si l'on prend un objet « Personne », moi ou vous par exemple, nous pouvons effectuer ces opérations telles que : **manger**, **dormir**, **lire**, **écrire**, **parler**, **travailler**, …etc.*

Le concept d'objet

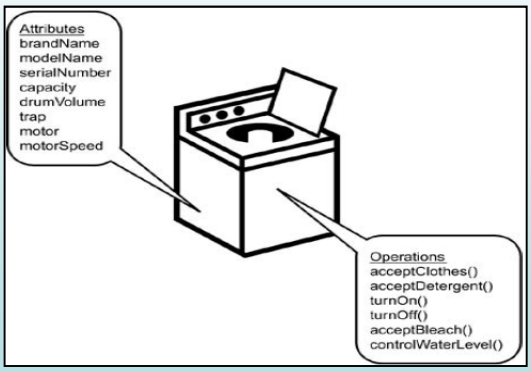
- **La notion de l'identité de l'objet :**
  - *En plus de son état, et de son comportement, un objet possède une **identité**.*
  - *L'**identité** d'un objet **caractérise son existence propre** et permet de le **distinguer de façon non ambiguë** indépendamment de son état.*
  - *L'**identité** permet de distinguer entre deux objets se trouvant dans un même état.*
  - *L'**identité** est un **concept**, elle ne se représente pas de manière spécifique en modélisation.*

Le concept d'objet

- **La notion de l'identité de l'objet :**
  - *Chaque objet possède une identité de manière **implicite**.*
  - *En phase de réalisation, l'**identité** est souvent construite à partir d'un **identifiant** issu d'une façon naturelle du domaine du problème.*
  - *Ainsi, nos voitures possèdent toutes un **numéro d'immatriculation**, nos téléphones un **numéro d'appel**, et nous-mêmes sommes identifiés par notre **numéro de SS** (sécurité sociale), ou par notre **numéro de CIN**.*

Le concept d'objet

- **Schéma illustrant la notion d'objet :**



Le concept de classe

- **La notion de classe :**
  - *Une **classe** regroupe un **ensemble d'objets** ayant **même structure** (attributs) et **même comportement** (méthodes).*
  - *Une classe représente le **modèle** (catégorie) pour ses objets.*
  - *En programmation, une classe nous sert pour créer des objets.*
  - *Il y a une relation entre un objet et sa classe : on dit qu'un **objet** est un **instance** d'une **classe** (**Mécanisme d'instanciation**).*

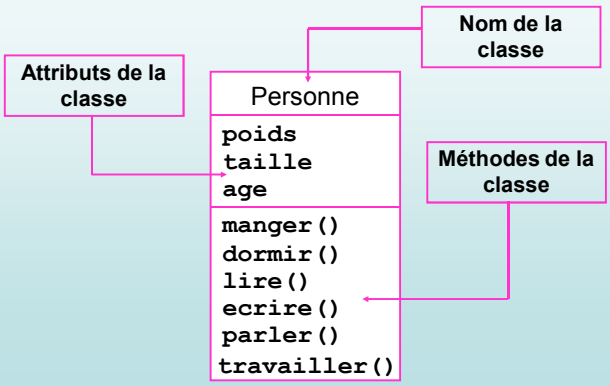
Le concept de classe

- **Exemples de classe :**
  - *La classe « Personne » : moi et vous, nous sommes des instances de cette classe.*
  - *La classe « Stylo » : le stylo de Fatima est une instance de cette classe.*
  - *La classe « Voiture » : la voiture du directeur en est une instance.*
- **Description d'une classe :**
  - *Une **classe** est décrite par un **nom** (première lettre en majuscule), la **liste de ses attributs** et la **liste de ses méthodes**.*

Le concept de classe

- **Représentation UML d'une classe :**
  - *En UML, une classe est représentée par un **rectangle** ayant **3 compartiments**.*
  - *Le **1er compartiment** est réservé au **nom de la classe**.*
  - *Le **2ème compartiment** est réservé aux **attributs de la classe**.*
  - *Le **3ème compartiment** est réservé aux **méthodes de la classe**.*
- **Exemple :**
  - *Considérons la classe « Personne ». Sa représentation UML est la suivante :*

Le concept de classe



Le concept de classe

- **Représentation UML d'un objet :**
  - *On représente un objet par un simple **rectangle** qui contient le nom de l'objet.*
  - *Le nom d'un objet est **souligné**.*
  - *On peut citer le nom de la classe de l'objet.*
  - *On peut omettre le nom de l'objet et garder celui de sa classe précédé de « : ».*
- **Exemple :**
  - *Un objet « Personne » sera désigné par exemple comme « Fatima », « :Personne » ou « Fatima:Personne ».*

Le concept de classe

Fatima

Fatima : Personne

: Personne

Représentation graphique d'un objet en UML

Le concept d'abstraction

- **L'abstraction en général :**
  - ***Abstraire**, c'est **soustraire** : soustraire du monde concret qui nous entoure toutes ses particularités pour en extraire des généralités appelées concepts.*
- **Exemple :**
  - *De tous les individus semblables à moi-même, j'extrais le concept d'homme.*
- **L'abstraction en POO :**
  - *L'abstraction est l'implémentation d'un code pour imiter les caractéristiques et les actions d'une entité du monde réel (un objet).*

Le concept d'abstraction

- **Exemple en C++ d'une abstraction :**

```
class Personne {
private:
  float poids, taille, age;
public:
  Personne(float, float, int);
  void manger( );
  void dormir( );
  void lire( );
  void ecrire( );
};
```

Le concept d'abstraction

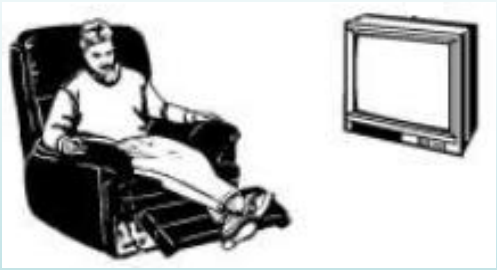
- **Instanciation :**
  - *Mécanisme par lequel on peut **créer un nouvel objet** d'une **certaine classe**.*
  - *Un tel objet est donc une **instance** de cette classe.*
- **Exemple en C++ d'une instanciation :**

```
class Personne { … };
…
Personne lui(80, 1.50, 29);
Personne elle(60.95, 1.45, 25);
```

Le concept d'encapsulation

- **La notion d'encapsulation :**
  - *L'**encapsulation** est l'équivalent **POO** de la notion de **boîte noire**.*
  - *L'encapsulation est le **mécanisme** qui consiste à **cacher les détails de l'implémentation d'un objet**.*
  - *Le mécanisme d'encapsulation assure à la fois une **cohésion interne** très forte et un **faible couplage** avec l'extérieur.*
  - *Le monde extérieur ne voit que l'**interface** (méthodes) de l'objet et pas son **corps** (code).*

Le concept d'encapsulation



La télévision cache le détail de ses opérations à la personne qui la regarde

Le concept d'encapsulation

- **Avantages de l'encapsulation :**
  - *Exposer l'interface d'un objet et cacher le détail de son implémentation permet d'atteindre l'objectif de **modularité**.*
  - *En POO, un système logiciel est formé de plusieurs objets qui interagissent entre eux de plusieurs façons.*
  - *Si l'un de ces objets ne fonctionne pas correctement et qu'il est nécessaire de le changer, le fait de cacher son implémentation aux autres objets, signifie que probablement ces derniers ne seront pas changés.*

Le concept d'encapsulation

- **Exemple montrant l'avantage de l'encapsulation:**
  - *Par exemple, votre ordinateur est formé d'un CPU, d'un moniteur, d'un clavier, …etc.*
  - *Le moniteur cache ses opérations au CPU.*
  - *Si le moniteur ne fonctionne pas correctement ou il tombe en panne, vous êtes obligés alors de le réparer ou même de le changer.*
  - *Mais, il est fort probable que vous n'allez pas réparer ou changer votre CPU.*
  - *La même chose survient dans un système logiciel. L'encapsulation permet donc d'avoir une **architecture modulaire** d'un logiciel.*

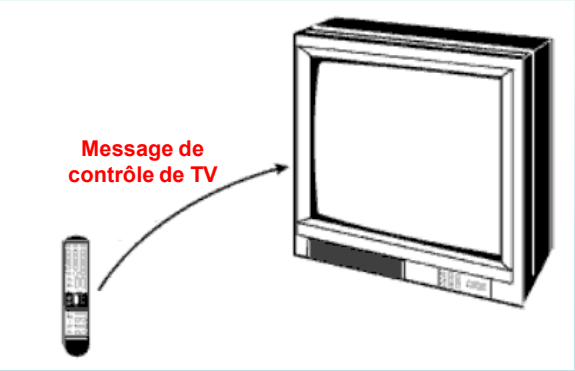
Le concept d'envoi de messages

- **La notion d'envoi de messages :**
  - *Dans un système logiciel orienté objet, un ensemble d'objets **collaborent** pour réaliser une tâche spécifique.*
  - *Cette collaboration entre objets est effectuée par un mécanisme unique : l'**envoi de messages**.*
  - *Un objet envois un message à un autre pour lui demander d'exécuter une opération.*
  - *L'objet qui reçoit un message répondra en exécutant l'opération demandée.*

Le concept d'envoi de messages

- **Exemple concret :**
  - *La TV et son télécommande présentent un bon exemple de mécanisme d'envoi de messages.*
  - *Lorsque vous voudriez regarder la TV, vous appuierez sur le bouton « ON » de votre télécommande.*
  - *La télécommande envois un message à la TV. Celle-ci le reçois, le comprends puis y réponds et vous, vous commencez à s'amuser de regarder la TV.*

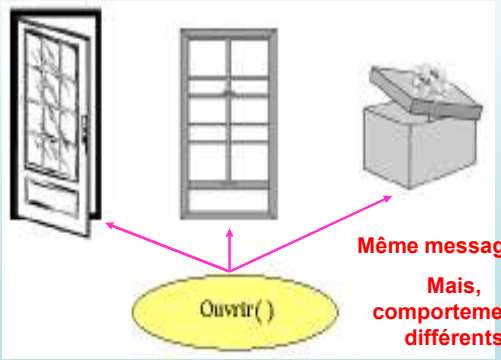
Le concept d'envoi de messages



Le concept de polymorphisme

- **Le polymorphisme en général :**
  - ***Polymorphe** : qui prend plusieurs formes.*
- **Le polymorphisme en OO :**
  - *Mécanisme par lequel on peut donner un **même nom à une méthode** dans des classes distinctes.*
- **Exemple :**
  - *On peut ouvrir une porte, une fenêtre, une école, un journal, une boîte à cadeaux, …etc.*
  - *Le message « Ouvrir » est alors un **message polymorphe**, et chaque objet receveur de ce message s'adapte pour l'exécuter.*

Le concept de polymorphisme



Le concept d'héritage

- **La notion d'héritage :**
  - *Mécanisme par lequel on peut **dériver une classe à partir d'autres**.*
  - *La **classe mère** ou **spuer-classe** est la classe de laquelle on peut dériver d'autres classes par héritage.*
  - *La **classe fille**, **sous-classe** ou **classe dérivée** est la classe obtenue à partir de sa super-classe par héritage.*
  - *La classe dérivée hérite les propriétés de sa super-classe.*

Le concept d'héritage

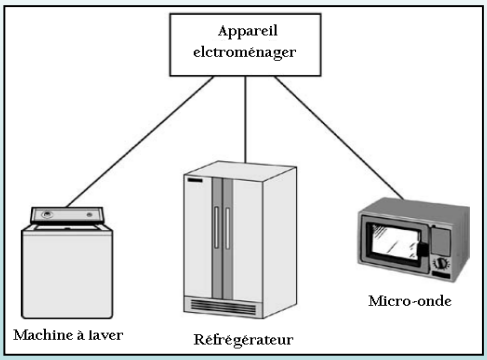
- **Héritage par généralisation :**
  - *La **généralisation** consiste à **factoriser les éléments communs** d'un ensemble de classes dans une classe plus générale.*
- **Exemples d'héritage par généralisation :**
  - *« Véhicule terrestre » est une généralisation de « Voiture » et « Camion ».*
  - *« Véhicule aérienne » est une généralisation de « Avion » et « hélicoptère ».*
  - *« Véhicule » est une généralisation de « Véhicule terrestre » et « Véhicule aérienne ».*

Le concept d'héritage

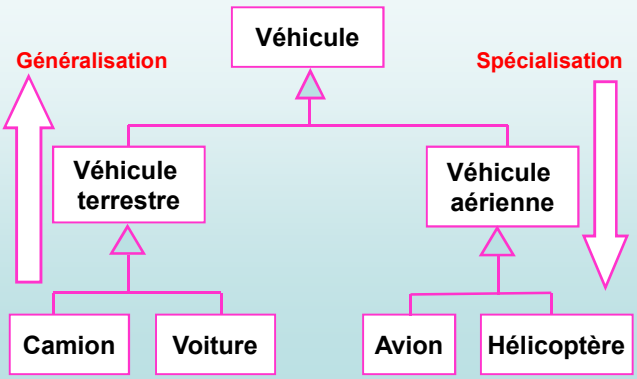
- **Héritage par spécialisation :**
  - *La **spécialisation** consiste à **capturer les particularités** d'un ensemble d'objets non discriminés par les classes déjà identifiées.*
- **Exemples d'héritage par spécialisation :**
  - *« Véhicule terrestre » et « Véhicule aérienne » sont des spécialisations de « Véhicule ».*
  - *« Camion » et « Voiture » sont des spécialisations de « Véhicule terrestre ».*
  - *« Avion » et « Hélicoptère » sont des spécialisations de « Véhicule aérienne ».*

Le concept d'héritage

- **Illustration de la notion d'héritage :**



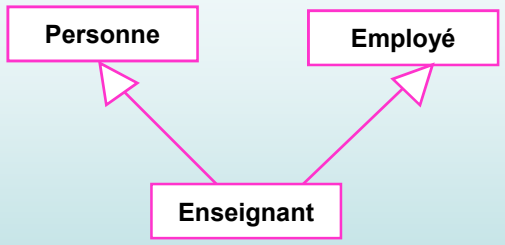
Le concept d'héritage



Le concept d'héritage

- **Types d'héritage :**
  - ***Héritage simple** : une classe n'hérite que d'une seule classe.*
  - ***Héritage multiple** : une classe peut hériter d'au moins deux classes.*
- **Exemple d'héritage multiple :**
  - *Un « Enseignant » est à la fois une « Personne » et un « Employé ».*
  - *Par conséquent, une classe « Enseignant » hérite des deux classes « Personne » et « Employé ».*

Le concept d'héritage



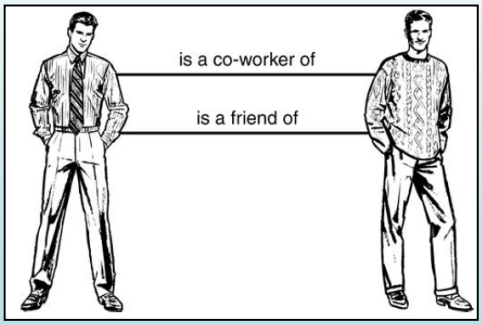
Héritage multiple

Les relations entre classes

- **L'association :**
  - *Relation qui exprime un **lien sémantique bidirectionnelle** entre classes.*
  - *Elle reflète une connexion qui existe dans le domaine d'application.*
- **Exemples d'association :**
  - *Une université héberge des étudiants, un étudiant étudie dans une université : il y a un lien sémantique entre un objet « Université » et un objet « Étudiant ».*
  - *Il s'agit d'une **association** entre ces deux objets.*

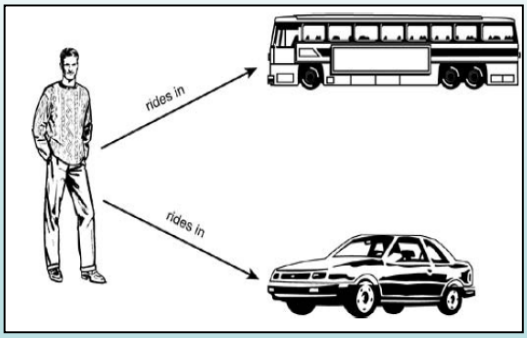
Les relations entre classes

➢ **Illustration de la notion d'association :**



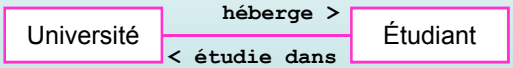
Les relations entre classes

➢ **Illustration de la notion d'association :**



Les relations entre classes

➢ **Représentation UML d'une association :**



Les relations entre classes

➢ **L'agrégation :**

- *Relation qui exprime un **lien structurel** entre classes.*
- *Elle permet de représenter des relations de type **maître/esclave**, **tout/partie** ou **composant/composé**.*

➢ **Exemples d'agrégation :**

- *Une équipe est formé de plusieurs joueurs.*
- *Un dossier comprend plusieurs fichiers et éventuellement d'autres dossiers (les sous-dossiers).*

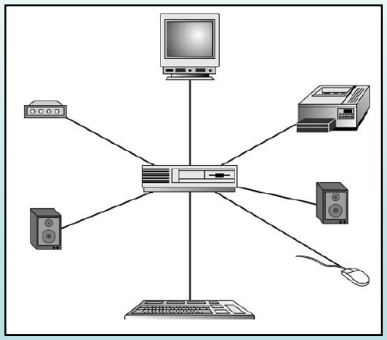
Les relations entre classes

➢ Types **d'agrégation** :

- ***Agrégation faible** : il n'y a pas une dépendance entre l'**agrégé** et l'**agrégat**.*
  - ***Exemple** : la relation entre une équipe et ses joueurs est une agrégation faible.*
- ***Agrégation forte** (**composition**) : le cycle de vie de l'agrégé (**composite**) dépend de celui de l'agrégat (**composant**).*
  - ***Exemple** : la relation entre un dossier et ses fichiers ou ses sous-dossiers est une agrégation forte.*

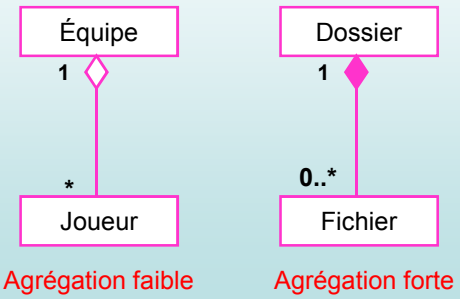
Les relations entre classes

➢ **Illustration de la notion d'agrégation :**



Les relations entre classes

- **Représentation UML de l'agrégation :**



Agrégation faible — Agrégation forte

Bibliographie

- ***Livres :***
  - *Modélisation objet avec UML,* *Pierre-Alain Muller.*
  - *À l'école de l'intelligence,* *Jean-Yves Fournier*
  - *Teach Yourself UML in 24 Hours,* *Joseph Schmuller.*
  - *The book of visual studio.net - A guie for developpers,* *Robert B. Dunaway.*