

The background is a light gray gradient. It is decorated with several realistic water droplets of various sizes, some with highlights and shadows, scattered across the surface. In the upper center, there is a faint, circular logo that appears to be the University of Michigan's 'M' emblem.

UML

INTRODUCTION



UML (Unified Modeling Language) est un langage de modélisation **orientée objet**.

Son but est d'aider à la conception d'applications, en spécifiant les différents éléments participant à celle-ci.

UML utilise une représentation graphique afin d'expliquer les différentes fonctionnalités du processus que l'on essaie de concevoir.

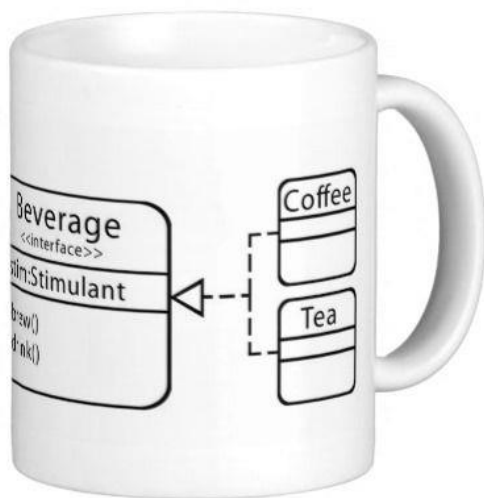
POURQUOI UML ?



UML permet dans un premier temps de bien comprendre les besoins du client, et ainsi éviter les surcoûts liés à la livraison d'un projet qui ne convient aux besoins du client, et ainsi un gain temps.

UML permet aussi d'aider les autres développeurs à mieux comprendre comment fonctionne un projet, au cas où il serait repris ou simplement pour effectuer des maintenances dessus.

MODELISATION ORIENTE OBJET



Le but est de représenter un objet de la vie courante, en objet « code », c'est-à-dire quelque chose que l'on pourra utiliser dans notre application.

Pour un objet, on va donc lui déclarer ses :

- Attributs : ce qui caractérise notre objet
- Méthodes : les actions réalisables par notre objet




EXEMPLE D'OBJET

Par exemple, prenons une voiture :

- On peut dire qu'une voiture est caractérisée, par sa couleur, sa marque, son modèle, etc. Ce sont ses attributs
- Une voiture peut avoir plusieurs comportements, comme par exemple rouler, ce sont des méthodes.

Mais on peut peut-être en ajouter d'autres ?



DIAGRAMMES UML

Il en existe plusieurs, tous ont un but précis :

- Diagramme cas d'utilisation
- Diagramme de classes
- Diagramme de séquence
- Diagramme d'activités
- Diagramme de séquences
- Etc...

Nous allons nous attarder sur les principaux : cas d'utilisations (Use case) et diagramme de classe.



CAS D'UTILISATION

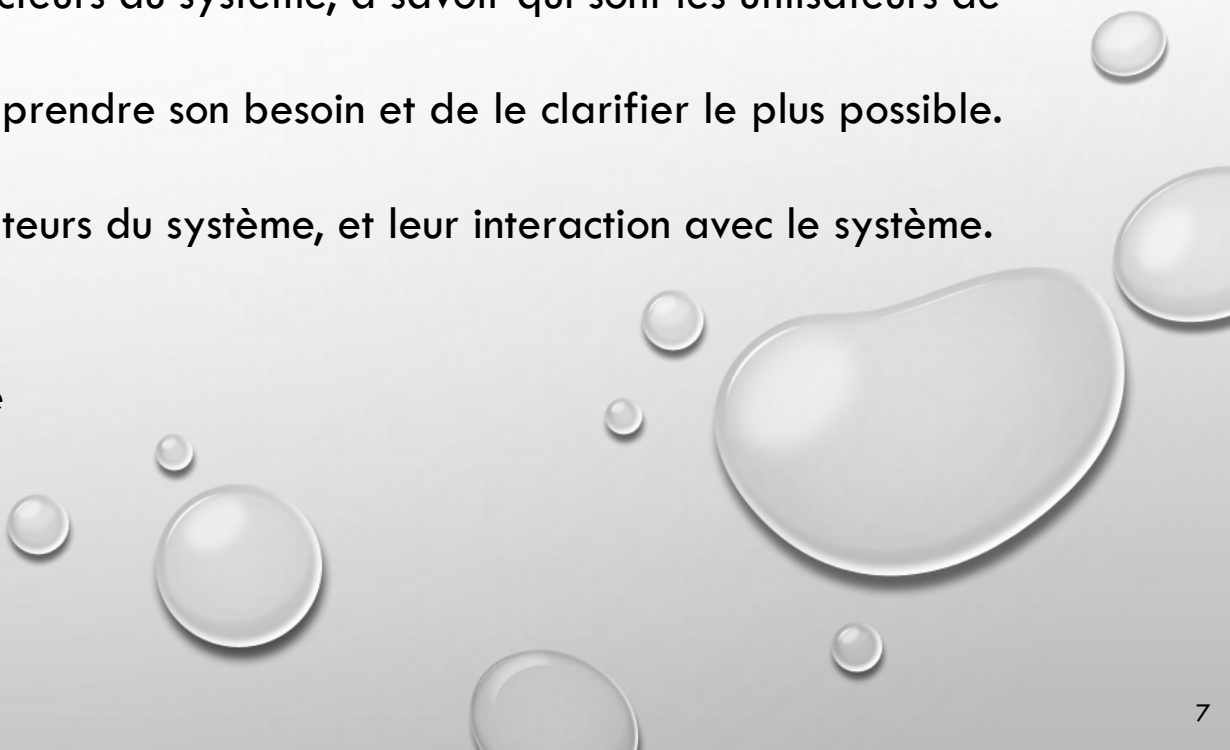
Il sert à structurer les différents besoins des utilisateurs et les objectifs correspondants d'un système.

Il faut donc dans un premier temps déterminer les différents acteurs du système, à savoir qui sont les utilisateurs de l'application, ou du site web.

Pour cela, il faut souvent questionner le client afin de bien comprendre son besoin et de le clarifier le plus possible.

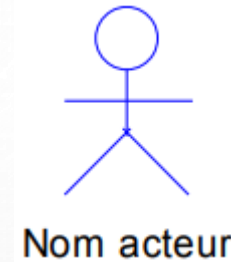
C'est justement le rôle du cas d'utilisation d'identifier les utilisateurs du système, et leur interaction avec le système.

Une fois ces besoins identifiés et structurés :

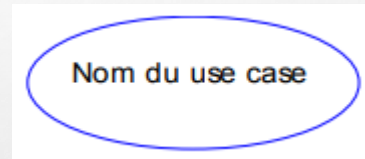
- On peut identifier les fonctionnalités principales du système
 - Le cheminement pour y accéder
- 

MODÉLISATION CAS D'UTILISATION

Afin de représenter un acteur du système on le fait de cette manière :



Afin de représenter une action, un objectif du système on le fait de cette manière :

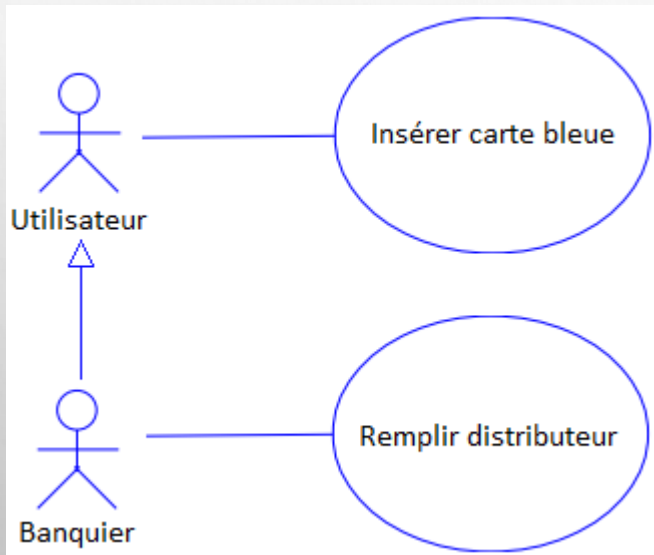


Pour indiquer qu'un acteur peut effectuer une action, on le représente par un simple trait :



MODÉLISATION CAS D'UTILISATION

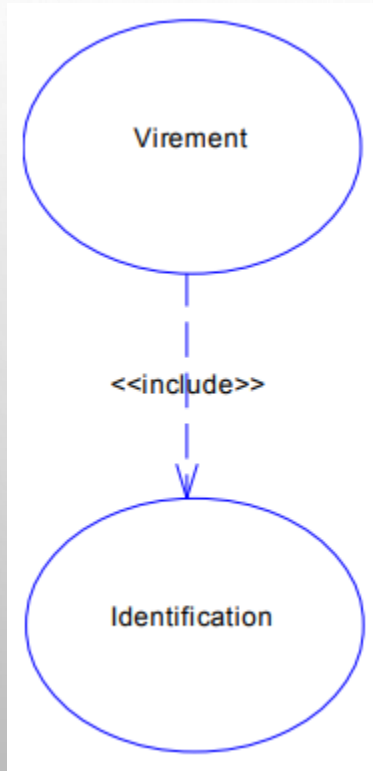
On peut indiquer qu'un acteur peut effectuer les mêmes actions qu'un autre acteur, avec des fonctionnalités supplémentaires :



Un banquier peut donc remplir le distributeur, mais aussi insérer sa carte bleue afin de l'utiliser.
On représente cela par une flèche allant du banquier vers l'utilisateur

MODÉLISATION CAS D'UTILISATION

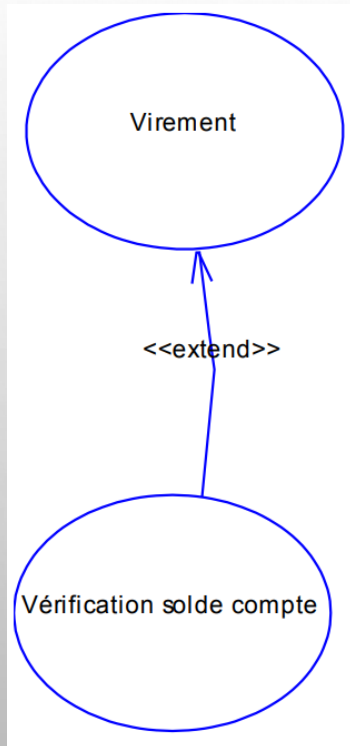
On peut indiquer que pour effectuer une action, on doit avoir au préalable fait une autre action :



L'action « **virement** » implique obligatoirement d'avoir fait l'action « **identification** » avant.

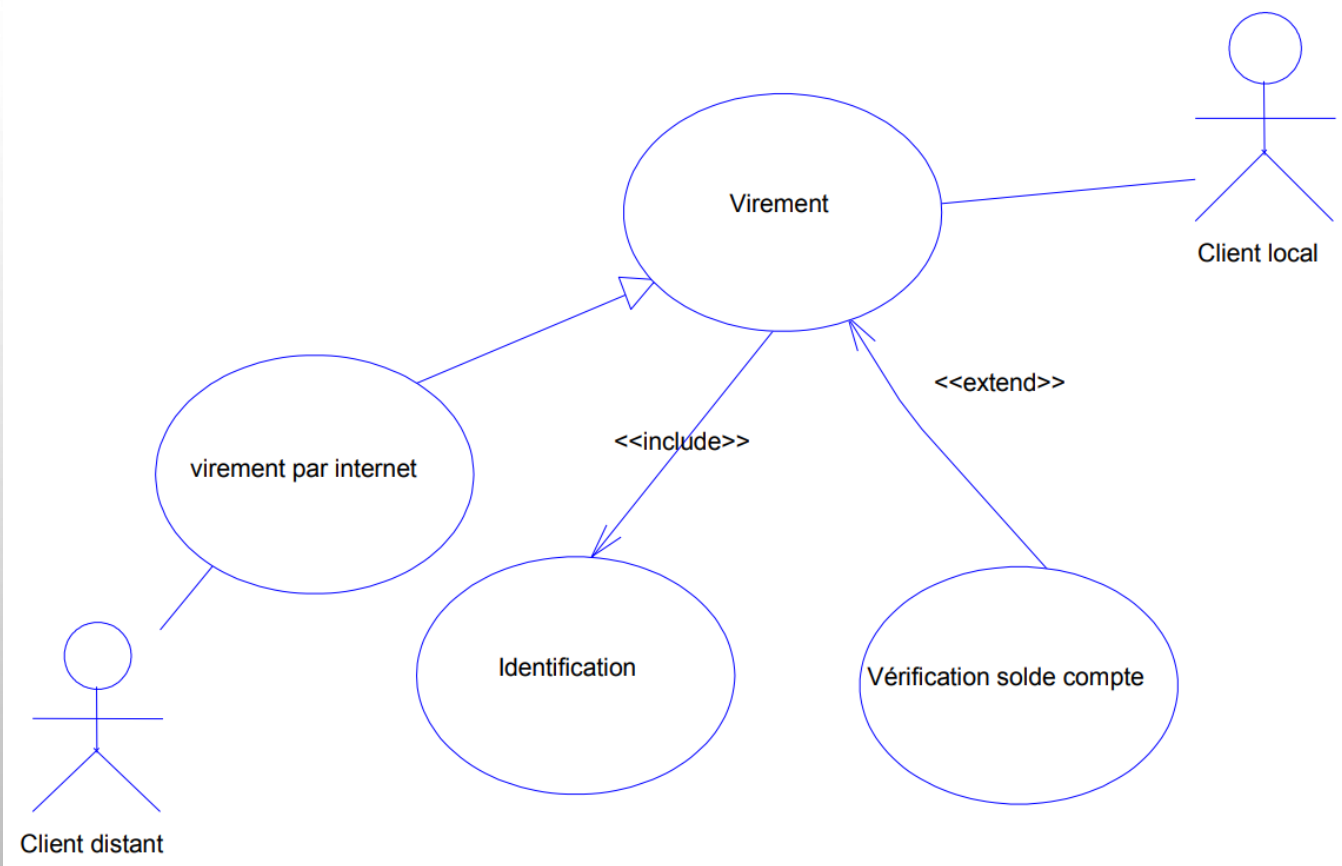
MODÉLISATION CAS D'UTILISATION

On peut indiquer qu'un cas d'utilisation ajoute son comportement à un autre cas d'utilisation :



Lorsque l'on fait l'action « **virement** » on a aussi l'action « **vérification solde compte** »

MODÉLISATION CAS D'UTILISATION



Exemple de cas d'utilisation complet



REALISATION D'UN CAS D'UTILISATION

Afin de réaliser un diagramme de cas d'utilisation, il est important de se poser plusieurs questions :

- Quels sont les acteurs ?
- Quelles sont les tâches réalisées par les acteurs ?
- Le système devra t'il informer l'acteur ?

Il y a plusieurs manières de voir un système :

- Un cas d'utilisation par acteur
- Un cas d'utilisation global

Tout dépend de la complexité de celui-ci.

The background of the slide is decorated with several realistic water droplets of various sizes, some with highlights and shadows, giving a fresh and clean aesthetic.

MODÉLISATION DIAGRAMME DE CLASSE

Le diagramme de classe permet de modéliser les différentes entités du système, c'est-à-dire les différents objets le composant et les relations entre eux.

Ce diagramme permet de faire ressortir plusieurs concepts de notre système :

- Les classes (objets) de l'application, avec leurs attributs et méthodes
- Les différentes relations entre elles
- Les généralisations

MODÉLISATION DIAGRAMME DE CLASSE

Qu'est-ce qu'une classe ?

Une classe est une description abstraite d'un objet en langage informatique (voir page 4 & 5).

Comment représente t'on une classe en UML ?

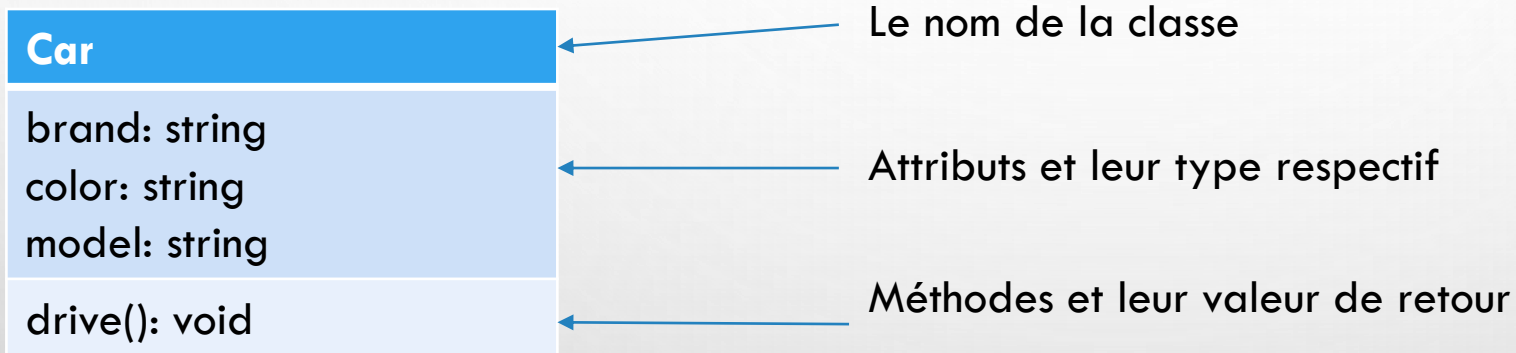
Il s'agit plusieurs rectangle représentant chacun un élément de la classe :

- Le nom de la classe
- Les attributs de la classe
- Les méthodes de la classe

Si leur contenu n'est pas pertinent, il est possible de ne pas les préciser.

MODÉLISATION DIAGRAMME DE CLASSE

Comment représenter une classe dans un diagramme de classe :



MODÉLISATION DIAGRAMME DE CLASSE

La notion d'attribut :

On a vu qu'un attribut est une caractéristique d'un objet, on doit lui donner un nom, mais il faut aussi lui donner un type, son type est un moyen de représenter la valeur attendue dans cet attribut, à savoir un entier, une chaîne de caractère ou encore une date.

Il est aussi important de préciser la « visibilité » des attributs d'une classe, à savoir si l'attribut peut-être utilisé en dehors de celle-ci ou non.

Pour cela, il existe 3 niveaux de visibilités :

- « public » (+) : élément visible en dehors de la classe
- « private » (-) : élément non-visible en dehors de la classe
- « protected » (#) : élément visible dans les classes filles (héritage)



MODÉLISATION DIAGRAMME DE CLASSE

Notion d'identifiant unique :

Il s'agit d'un attribut particulier, qui permet de repérer de façon unique chaque objet, instance de la classe. Il s'agit souvent :

- D'un entier incrémenté
- D'un numéro de facture
- D'un email

Il s'agit d'un moyen de représenter facilement notre objet, plutôt que de dire « la Renault Clio Rouge », on aura plutôt un numéro d'immatriculation qui est unique.



MODÉLISATION DIAGRAMME DE CLASSE

Notion de méthodes :

Il s'agit des actions réalisables par l'objet, comme « rouler » pour la voiture.

Il faut préciser les parenthèses après, qui correspondent au paramètre de la méthode, autrement dit ce que le comportement a besoin pour fonctionner.

Les 2 points après la fonction représente ce que va renvoyer la méthode, void est pour « vide », autrement rien, sinon on peut indiquer qu'il s'agit d'un entier, une chaîne de caractères.

Les méthodes sont soumis aux mêmes règles de visibilité que les attributs.

MODÉLISATION DIAGRAMME DE CLASSE

Récapitulatif de notre classe :

Car

```
-id: int(6)
-immatriculation: string
-brand: string
-color: string
-model: string

+drive(): void
+airConditionner(temp: int): void
```

Classe d'un diagramme, correctement rédigée.



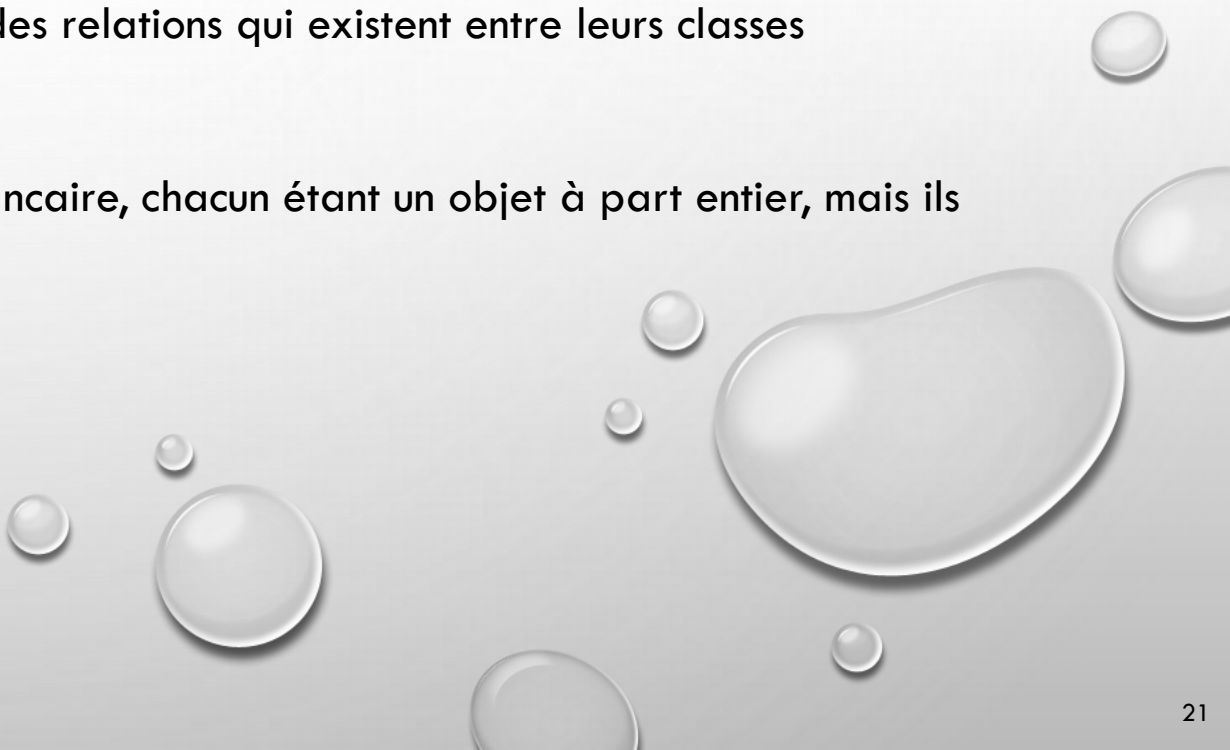
MODÉLISATION DIAGRAMME DE CLASSE

Notion de relations entre les classes :

Il existe plusieurs liens entre les objets, cela se traduit par des relations qui existent entre leurs classes respectives.

Par exemple, on peut dire qu'une personne a un compte bancaire, chacun étant un objet à part entier, mais ils sont liés entre eux.

Il existe plusieurs types de relations :

- L'association
 - La généralisation
- 

MODÉLISATION DIAGRAMME DE CLASSE

L'association :

Il s'agit de la relation la plus courante, elle permet de relier plusieurs classes entre elles.

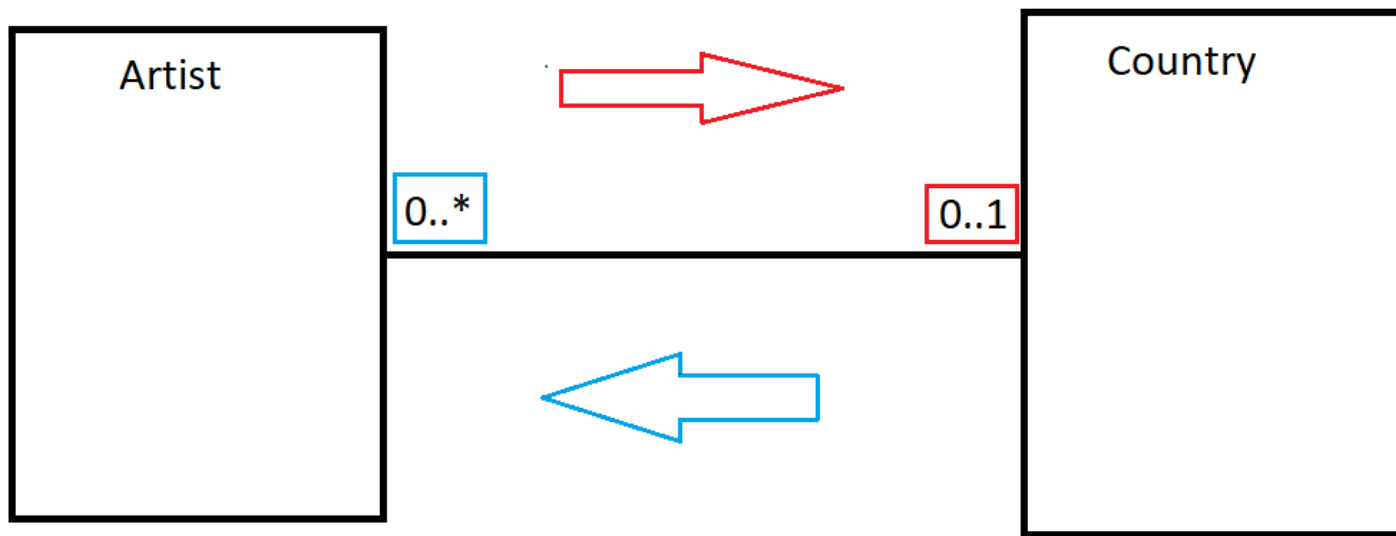
Lorsque l'on écrit une association entre deux classes, il est important de préciser plusieurs informations :

- La multiplicité : le nombre d'instances de l'association pour une instance de la classe. La multiplicité est définie par un nombre entier ou un intervalle de valeurs :

1	Un et un seul
0..1	Zéro ou un
0..*	Zéro à plusieurs
1..*	Un à plusieurs

MODÉLISATION DIAGRAMME DE CLASSE

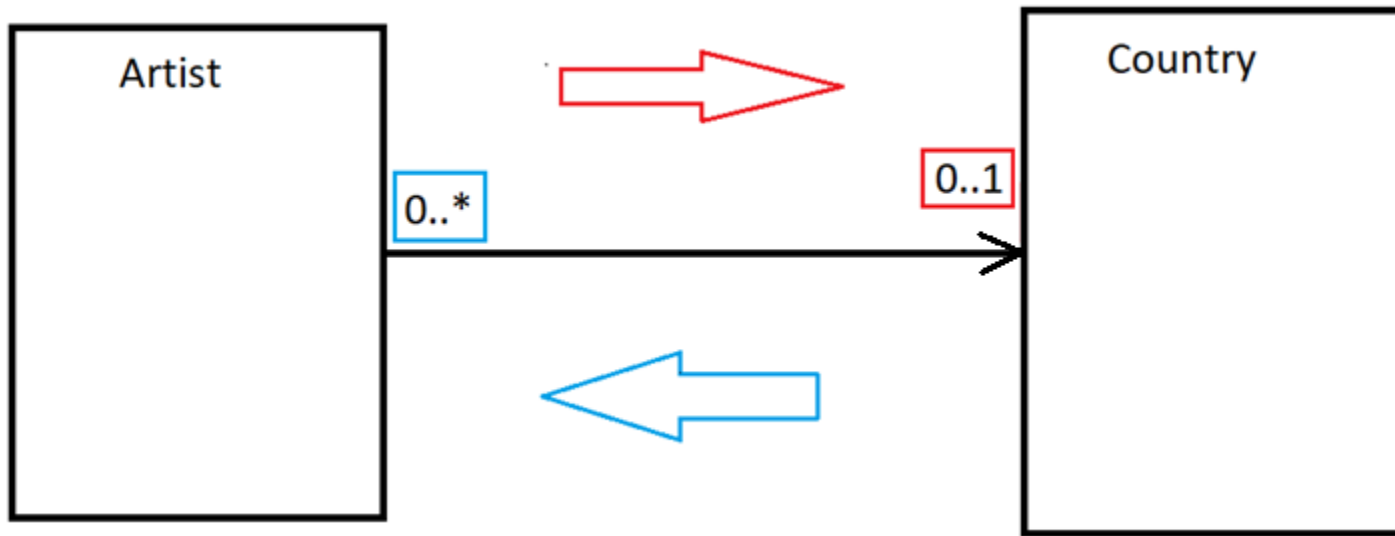
Exemple de relation « association » simple :



Un artiste a un ou aucun pays
Un pays a aucun ou plusieurs artistes

MODÉLISATION DIAGRAMME DE CLASSE

Exemple de relation « association » avec navigation :

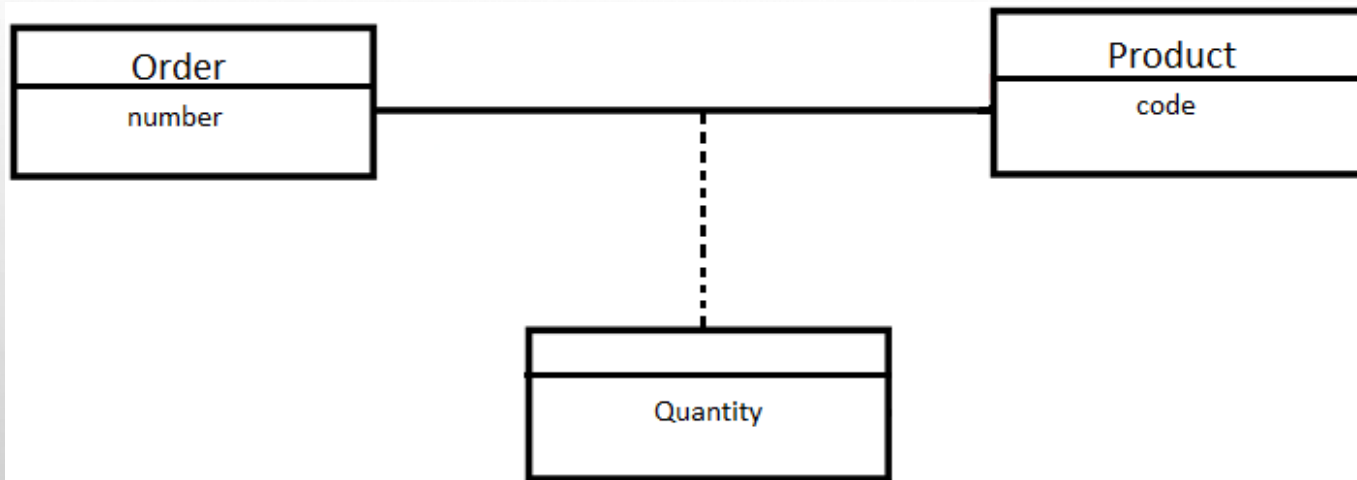


Vous noterez qu'il y a la présence d'une flèche partant de « **Artist** » vers « **Country** », ce qui signifie :

Les instances d'Artist connaissent les instances de Country, mais les instances de Country ne connaissent pas les instances d'Artist

MODÉLISATION DIAGRAMME DE CLASSE

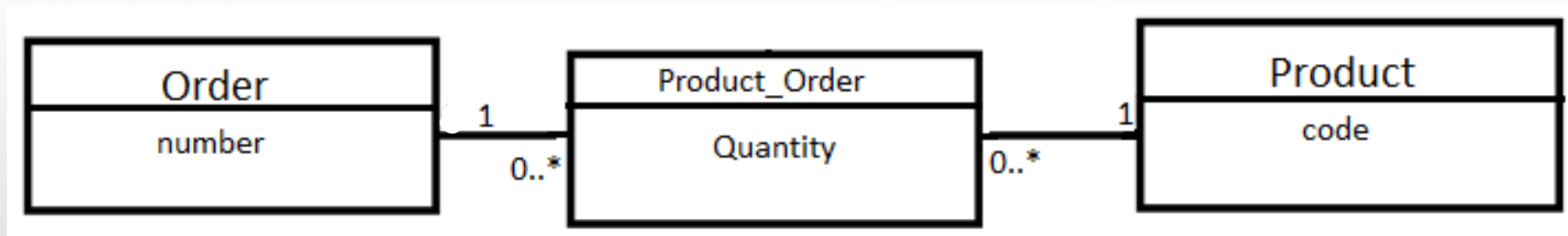
Il existe aussi les classes dites d'association, il s'agit d'une classe porteuse d'attributs entre deux autres classes :



Ce qui signifie qu'il y a une classe en relation entre **Order** et **Product** qui aura l'attribut « **quantity** »

MODÉLISATION DIAGRAMME DE CLASSE

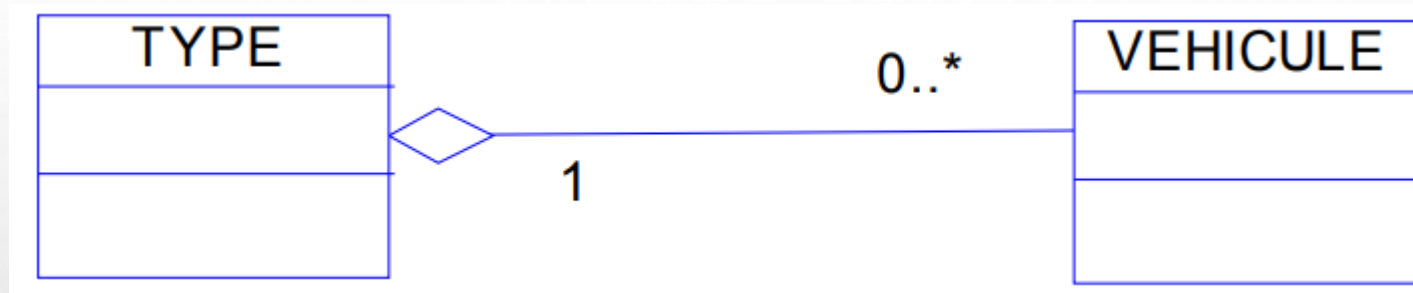
Il s'agit de la même représentation, sauf que cette fois on utilise une classe entre les deux, et non d'association :



Ainsi il y a une classe « Product_Order » qui est créée.

MODÉLISATION DIAGRAMME DE CLASSE

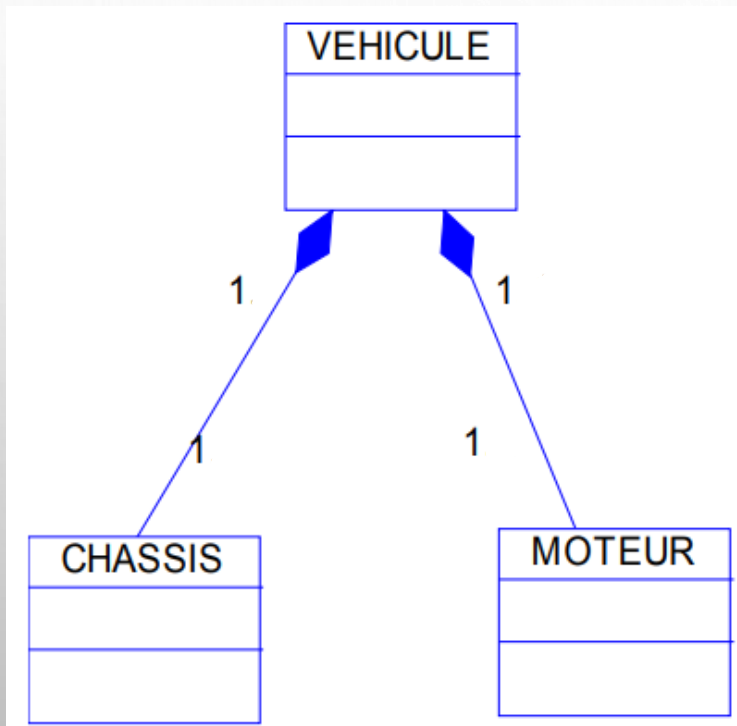
Une autre relation, c'est l'**agrégation**, une variante de l'association :



Elle se représente par un losange du côté de l'agregat. Cela signifie qu'un **véhicule** a les attributs d'un **type**, on aurait pu déclarer les attributs de types directement dans véhicule, mais on a souhaité le faire autrement.

MODÉLISATION DIAGRAMME DE CLASSE

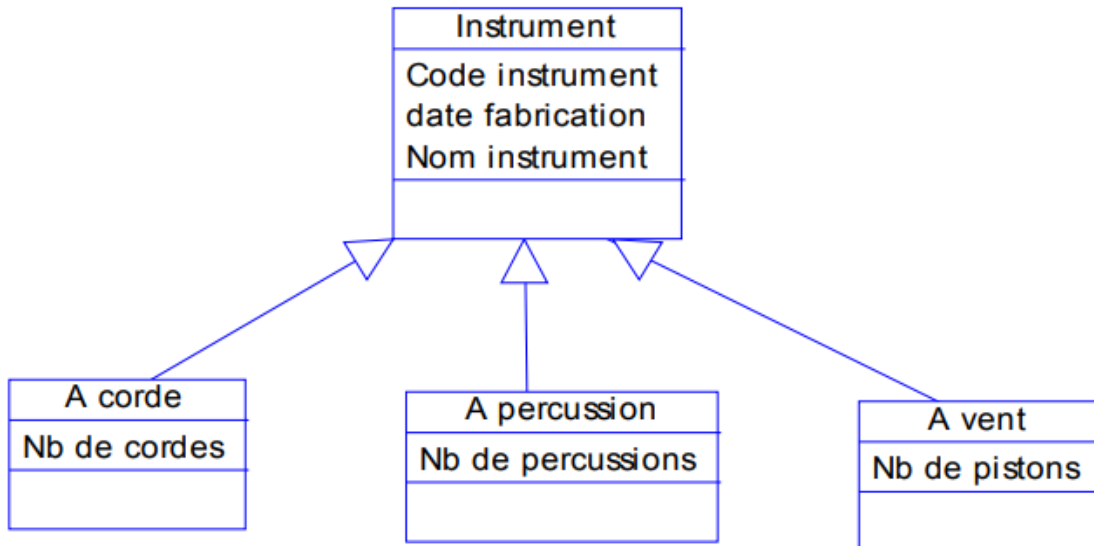
La **composition**, qui est un cas particulier de l'agrégation :



Elle se représente par un **losange plein** du côté de l'agregat. La composition permet la même chose que l'agrégation, sauf qu'elle a une coïncidence sur la durée de vies des objets, c'est-à-dire que si l'on supprime le véhicule, on supprime les châssis et les moteurs qui lui sont liés.

MODÉLISATION DIAGRAMME DE CLASSE

Enfin, il y a la **généralisation** :



La généralisation se caractérise de la même manière que pour que les cas d'utilisations : par une flèche.

On parle de généralisation lorsque l'on veut identifier plusieurs objets comme étant un sous-ensemble d'objets, ayant des spécificités communes.

La classe principale, ici **instrument**, est appelée **classe mère**, les autres sont ses classes filles, par héritage, elles contiennent tous les attributs, méthodes et relations de la classe mère.

On peut dire que « A Corde » est un instrument.