UML : CLASSE

UTILISATION

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

CLASSE EN UML

Comment représente-t-on une classe en UML ?

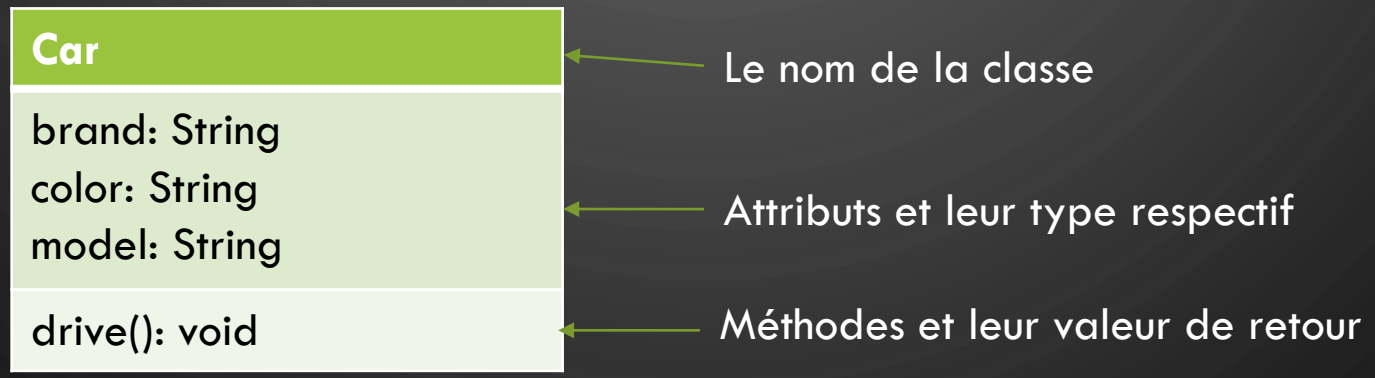
Il s'agit plusieurs rectangle représentant chacun un élément de la classe :

- Le **nom** de la classe (écrite en **PascalCase**)
- Les **attributs** de la classe (écrite en **camelCase**)
- Les **méthodes** de la classe (écrite en **camelCase**)

Notamment pour les méthodes, si leur contenu n'est pas pertinent, il est possible de ne pas les préciser.

EXEMPLE

Comment représenter une classe dans un diagramme de classe ?



VISIBILITÉ

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 (String, int, Date, float, boolean, etc)

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)

IDENTIFIANT

Un identifiant unique, est 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.

MÉTHODES

Pour les méthodes, il s'agit des actions réalisables par l'objet, comme « rouler » pour la voiture.

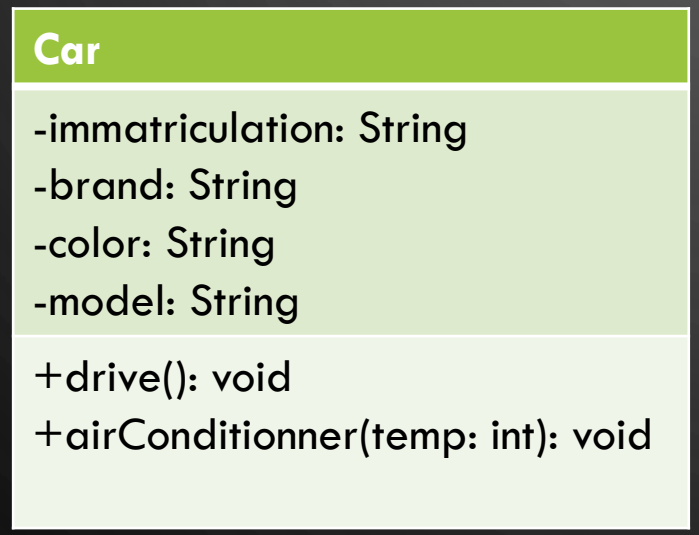
Il faut préciser les parenthèses après, qui correspondent au paramètre de la méthode, autrement dit ce dont le comportement a besoin pour fonctionner.

Les 2 points après la fonction représentent 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 soumises aux mêmes règles de visibilités que les attributs.

EXEMPLE COMPLET

Récapitulatif de notre classe :



Classe d'un diagramme, correctement rédigée.

RELATIONS

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
- L'agrégation
- La composition

MULTIPLICITÉ

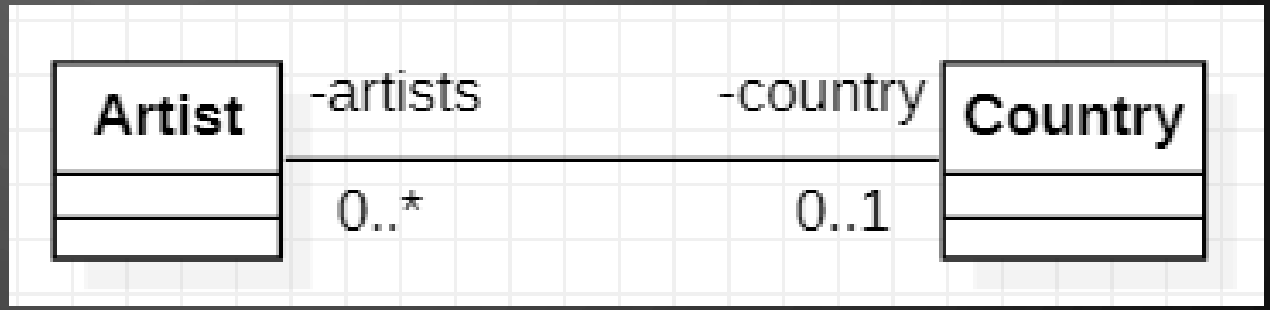
Lorsque l'on écrit une relation entre deux classes, il est important de préciser plusieurs informations :

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 :

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

ASSOCIATION

Exemple de relation « association » simple :

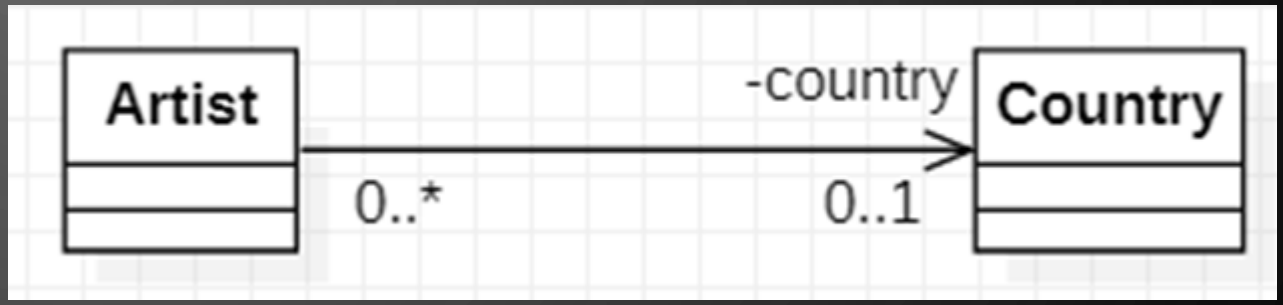


Une instance d'artiste n'a aucune ou une seule instance de pays
Une instance de pays n'a aucune ou plusieurs instances d'artistes

« -artists » et « -country » représente le nom des attributs qui seront créés dans chacune des classes respectives

NAVIGATION

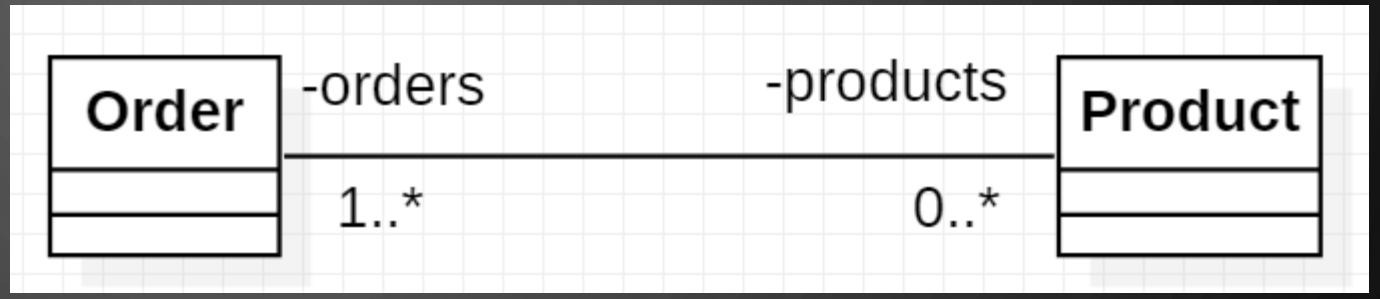
Exemple de relation « association » avec navigation :



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

RELATION *..*

Il existe des relation dites « *..* », c'est-à-dire avec une multiplicité impliquant « * » de chaque côté de la relation.



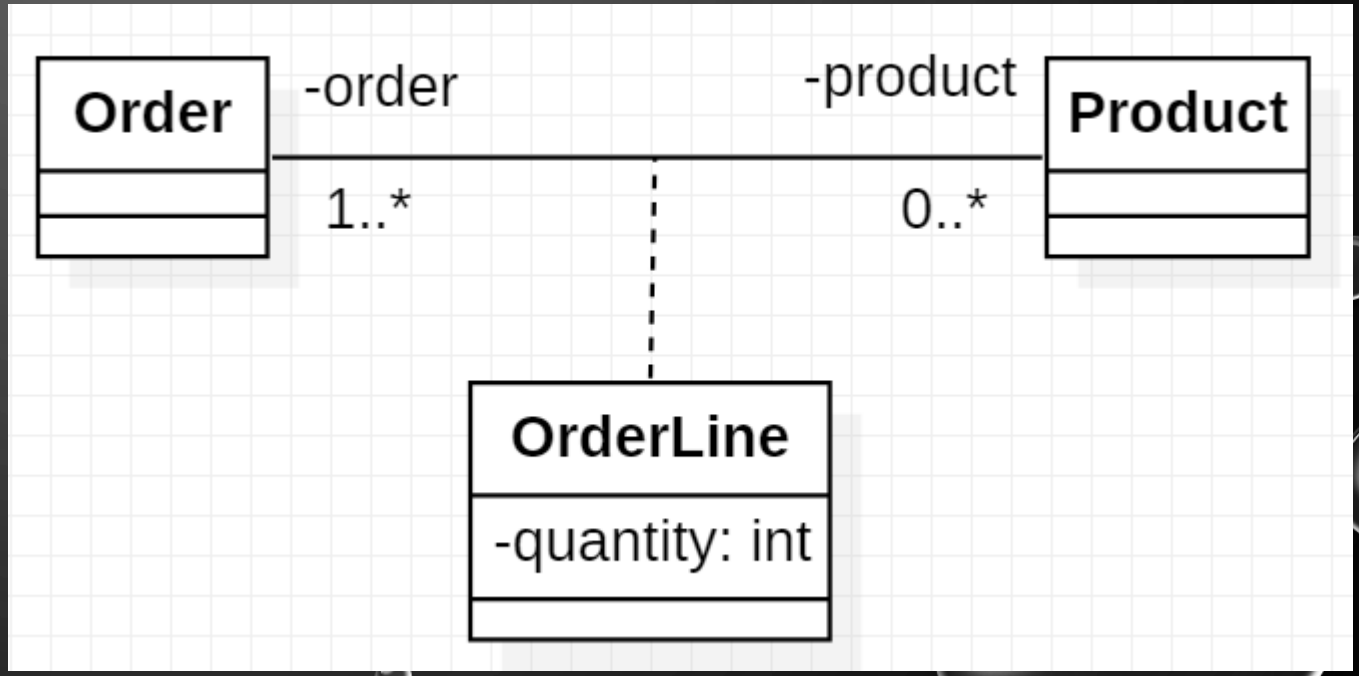
Pour faire le parallèle avec la base de données, une table associative sera créée entre les deux, en termes de classe, il n'y a pas de changements : on a bien nos deux classes avec les deux attributs créés de chaque côté.

CLASSE D'ASSOCIATION

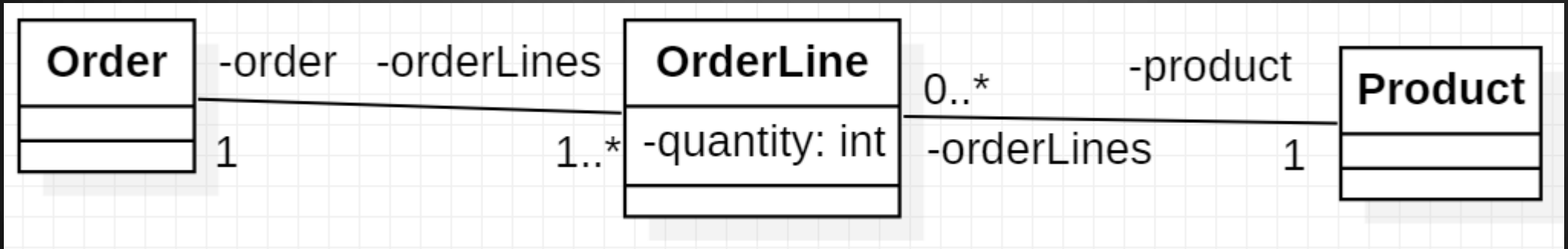
Il existe aussi les classes dites d'association, il s'agit d'une classe porteuse d'attributs entre deux autres classes, lors d'une relation « *..* »

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

Cette fois il faut bien la déclarer en tant que classe, la créer dans le code et elle sera présente en BDD (comme le cas précédent)

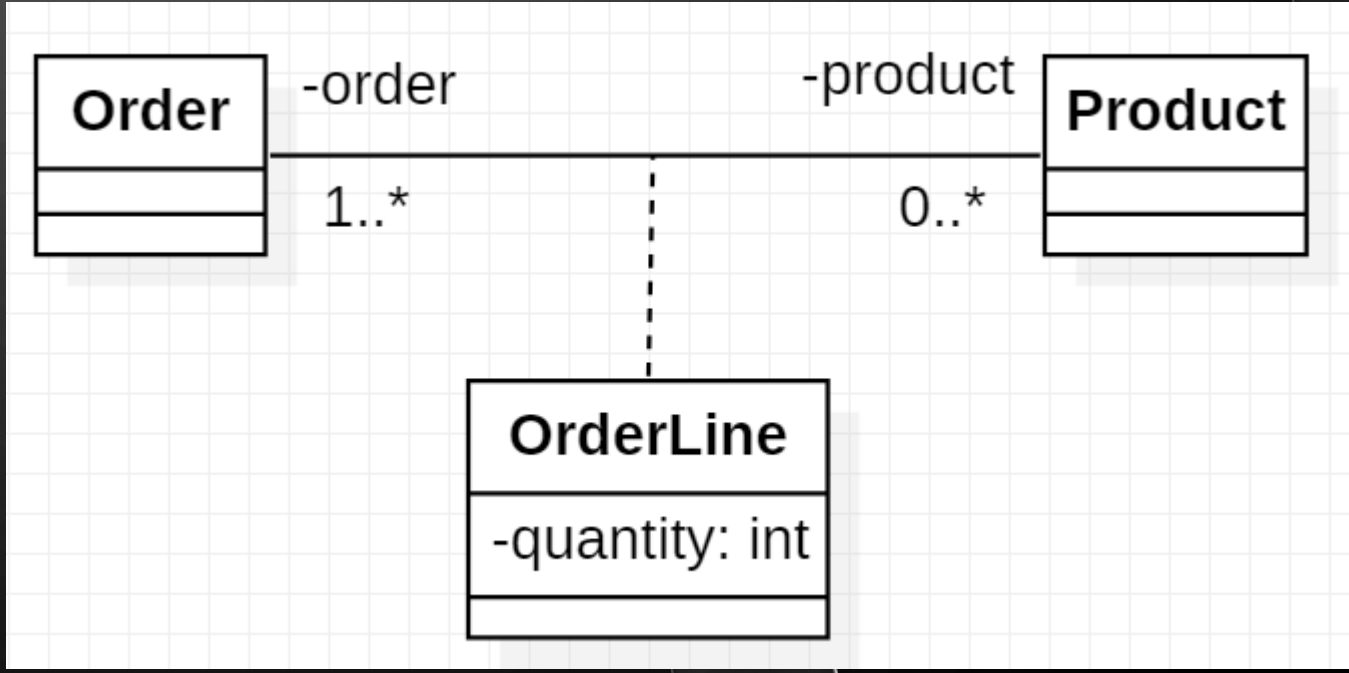


CLASSE D'ASSOCIATION



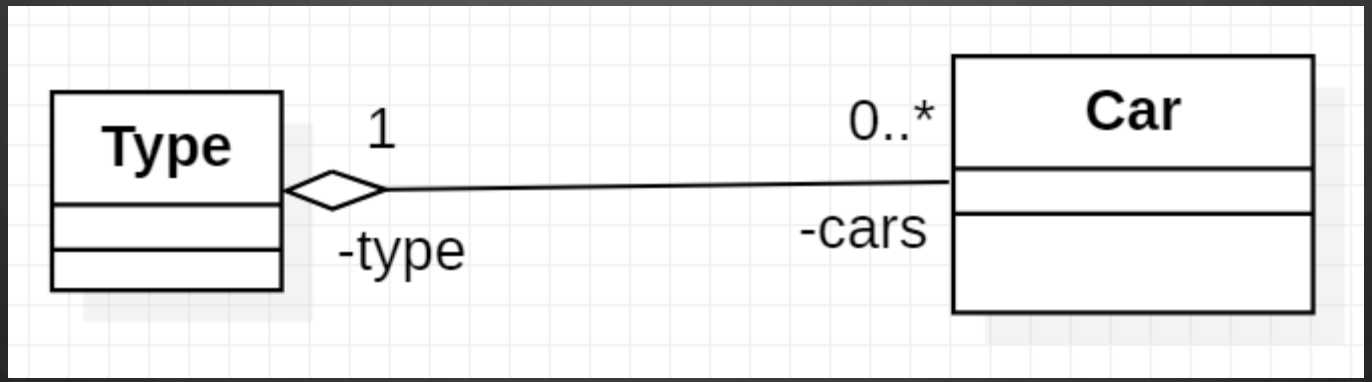
Il s'agit de la même représentation, sauf que cette fois on utilise une classe entre les deux, et non une classe d'association, mais la finalité est la même !

Pour comparer :



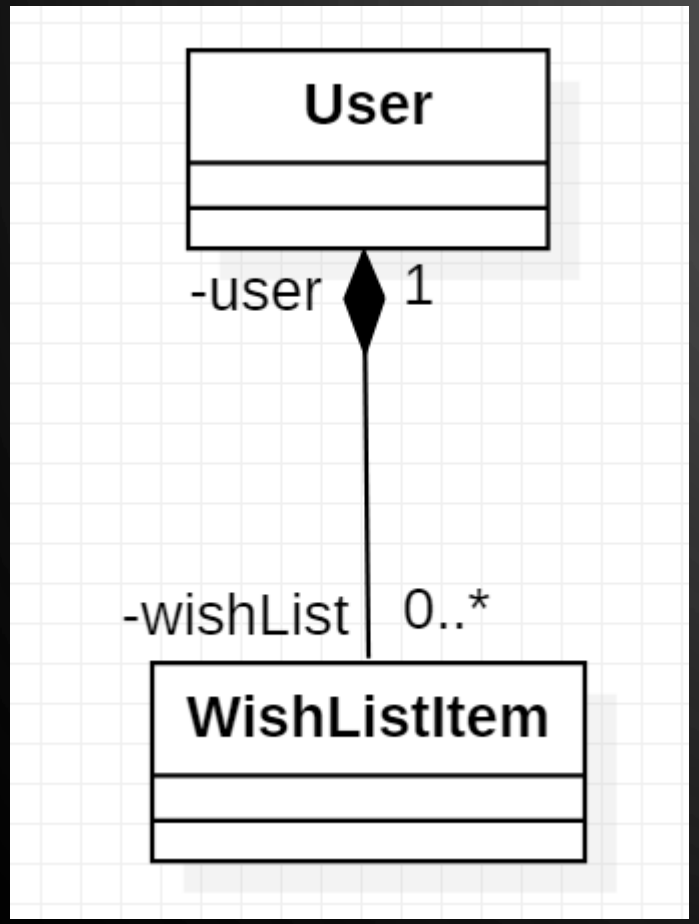
AGRÉGATION

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



Elle se représente par un losange du côté de l'agrégat. 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.

COMPOSITION



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

Elle se représente par un **losange plein** du côté de l'agrégat. 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.

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.