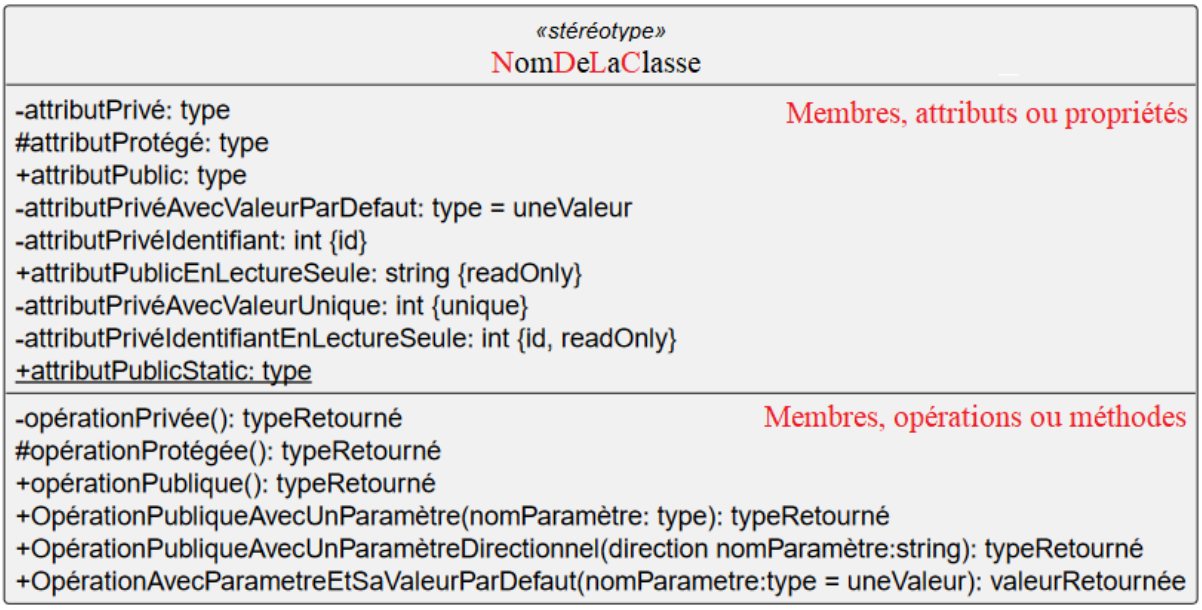
Les diagrammes de classes UML

I) Qu'est-ce qu'une classe en POO ?

Voir intro à la POO.

II) Représentation d'une classe avec un diagramme UML



Nommage

Le nom d'une classe, d'un attribut ou d'une méthode doit être explicite, en anglais et en camelCase (ou CamelCase pour les classes).
Classe : MyClass
Attribut : myAttribut
Methode : myMethode

Visibilité

- \- : visibilité privée. Le membre n'est accessible que depuis l'intérieur de la classe.
- # : visibilité protégée. Le membre n'est accessible que depuis la classe et ses héritières.
- \+ : visibilité publique. Le membre est accessible depuis la classe et en dehors.

Type

Exemples de type : int, bool, string, float, array, List, nom d'une classe, etc. (dépend du langage).

Stéréotype

- Le stéréotype d'une classe est un mot qui permet de faciliter la lecture du diagramme.
  « entity » : indique que la classe correspond à une table dans la base de données.
  « controller » : indique que la classe est un contrôleur.
  « repository » : indique que la classe contient des méthodes faisant appel à des requêtes.
  « singleton », « listener », « enum », etc.
- Le stéréotype d'une méthode permet de la classer dans une catégorie de comportement : « abstract », « getter », « setter », etc.

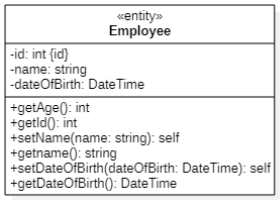
Informations à transmettre

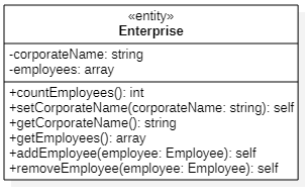
Un diagramme UML peut rapidement devenir complexe à lire. En fonction du destinataire de l'information, il est possible de ne pas faire apparaitre certaines informations.

- Par exemple, l'utilisateur d'une classe n'a besoin de connaître que ce qu'il peut utiliser, c'est à dire uniquement les membres publics.
- Par souci de simplification, on peut aussi décider de ne pas faire figurer les getters et les setters, voire même toutes les méthodes.
- On peut même n'afficher que le nom des classes.

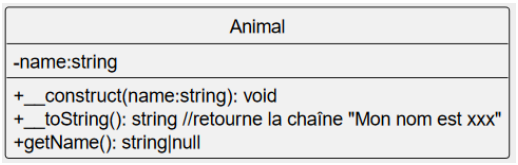
Exercice 1 : Modéliser deux classes, en respectant le principe d'encapsulation, dont les instances vont être persistées en base de données et qui permettent de créer les objets ci-dessous :

- Un employé caractérisé par un numéro unique, un nom et une date de naissance. L'objet doit pouvoir renvoyer son âge en années.
- Une entreprise caractérisée par sa dénomination sociale et les employés qu'elle fait travailler. L'objet doit pouvoir retourner le nombre d'employés qu'elle fait travailler.





Exercice 2 : Implémenter la classe ci-contre en PHP.
Tester cette classe en créant deux animaux et en affichant le nom du premier avec la méthode magique __toString() et le nom du second avec la méthode getName().

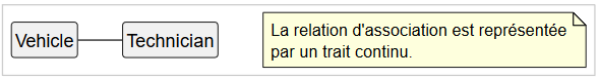


III) Représentation des liaisons entre les classes

Le diagramme de classes permet de mettre en évidence le rôle de chaque classe dans l'application et le ou les liens qu'elles ont entres elles. C'est très utile pour le développeur car il sait exactement ce qu'il doit "coder".

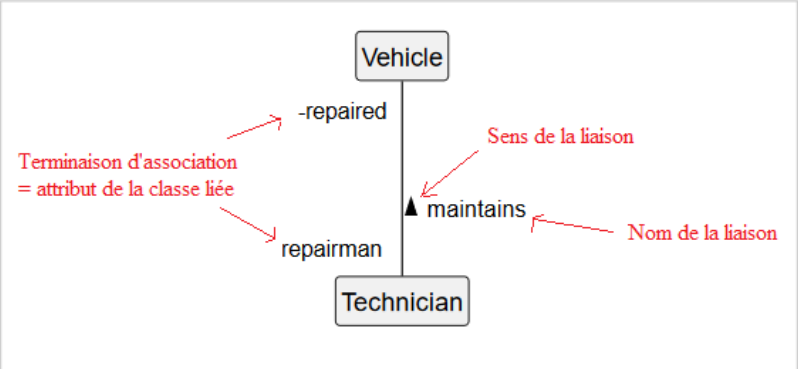
Le lien associatif

Une association entre A et B exprime une relation de contenance, c'est-à-dire qu'elle désigne un lien entre deux objets A et B sachant que A contient une ou plusieurs instances de B (et/ou inversement).
Dans l'objet A, il existe un attribut qui stock une ou plusieurs instances de B (et/ou vice-versa)



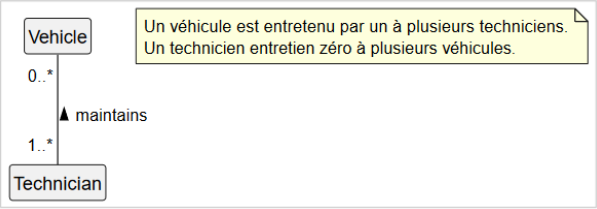
Grâce au lien, nous comprenons que *vehicle* est entretenu par *technician* et que *technician* entretient *vehicle*.

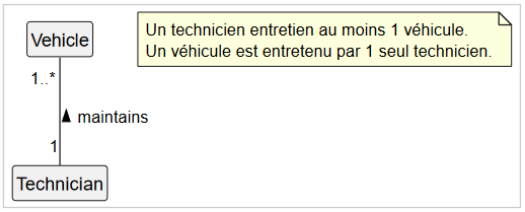
Si on le souhaite, on peut rajouter des informations supplémentaires sur la liaison (nom, sens, terminaisons d'association).



Les cardinalités d'une association

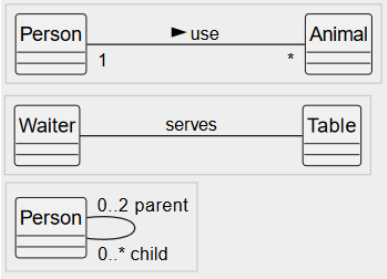
Des cardinalités peuvent être ajoutées afin d'exprimer une multiplicité du lien associatif entre deux classes.
Dans l'exemple précédent, on voit que *vehicle* est entretenu par *technician*, mais on n'a pas d'information sur le nombre de techniciens nécessaire pour entretenir le véhicule.





Cardinalité	Interprétation
1	Un et un seul
1..*	Un ou plusieurs
2-5	2 à 5 (maximum
1,5,7	1 ou 5 ou 7
0..*	0 ou plusieurs
*	0 ou plusieurs

Attention
Dans la méthode Merise, les cardinalités sont inversées par rapport à UML.

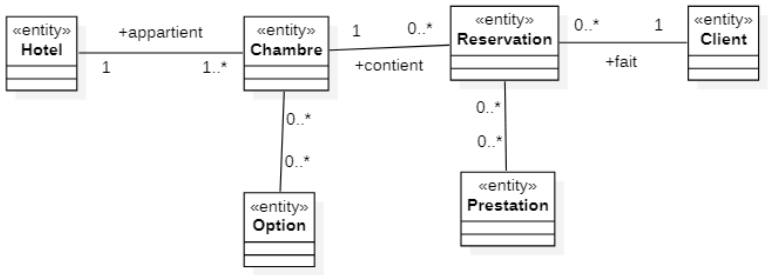


Exercice 3 :
Pour chaque diagramme ci-contre :
- Déterminer les cardinalités, lorsqu'elles ne sont pas notées.
- Exprimer la relation en prenant en compte les cardinalités.

Exercice 4 : Une entreprise gère des hôtels. Des clients peuvent réserver des chambres dans ces hôtels. Une réservation ne peut porter que sur une seule chambre.

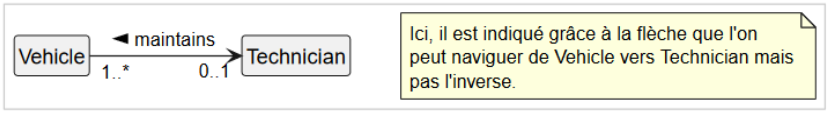
1) Réaliser le diagramme de classes correspondant au cas décrit ci-dessus sans les membres, en nommant les associations et en précisant les cardinalités.
2) Compléter votre diagramme en tenant compte des informations ci-dessous :

Des prestations supplémentaires (petit déjeuner, réveil par l'accueil, etc.) peuvent compléter la mise à disposition d'une chambre. Une chambre est équipée de différentes options (lit simple/ double, micro-onde, lit enfant, etc.).



Navigabilité d'une association

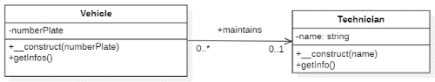
La navigabilité indique qu'un objet de la classe cible peut être atteint par un objet de la classe source au travers de l'association.



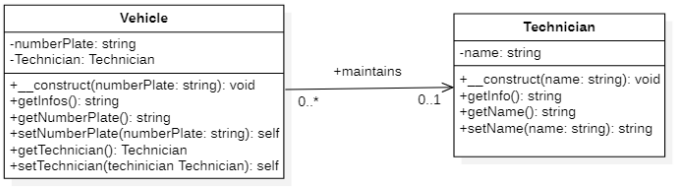
Concrètement cela veut dire que la classe *Vehicle* procédera une méthode getTechnician() qui permettra de récupérer le technicien s'occupant du véhicule.
La navigabilité étant unidirectionnelle, la classe *Technician* ne possédera pas de méthode getVehicles() qui permettrait de récupérer la liste de tous les véhicules réparés par le technicien.
Si on souhaite pouvoir accéder à cette information dans *Technician*, alors la navigabilité de la relation sera bidirectionnelle.

Il est essentiel pour le développeur de connaitre la navigabilité d'une relation. Cela lui permet de connaitre les méthodes à implémenter.

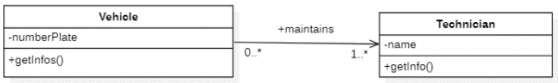
Exercice 5 : Implémentation d'une association unidirectionnelle simple



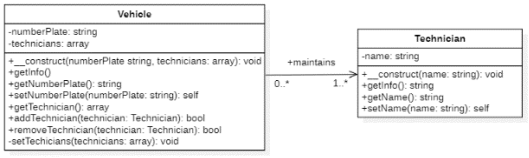
Tous les membres ne sont pas représentés dans ce diagramme. A l'aide des informations qu'il contient et afin de respecter le principe d'encapsulation et le principe de chainage des méthodes, compléter ce diagramme avec tous les membres et les types possibles.
La méthode getInfo() renvoie une chaine de caractères avec toutes les informations connues sur l'objet.
Puis coder ces deux classes.
Et vérifier votre code en instanciant un technicien et un véhicule en lui assignant un technicien et appeler getInfo() sur chaque objet.



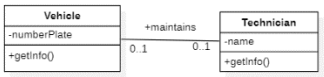
Exercice 6 : Implémentation d'une association unidirectionnelle multiple



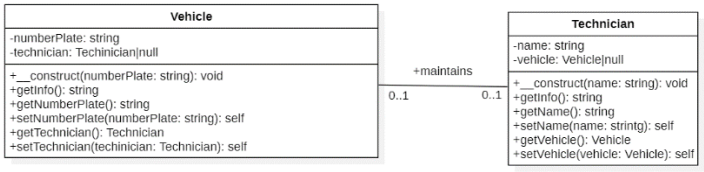
Même exercice que précédemment



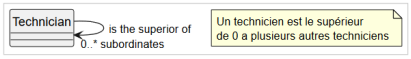
Exercice 7 : Implémentation d'une association bidirectionnelle simple



Même exercice que précédemment

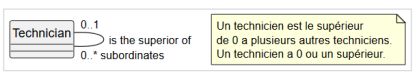


Association réflexive

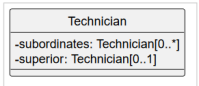


C'est équivalent à cette représentation :

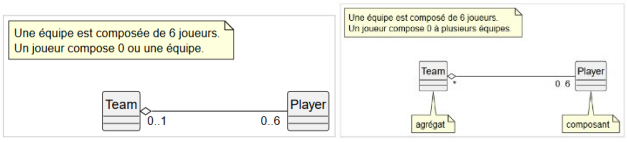
Voici la même modélisation mais avec une bidirectionnalité :



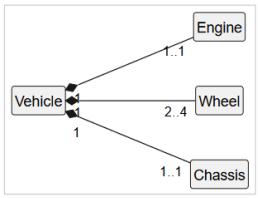
Ce qui est équivalent à :



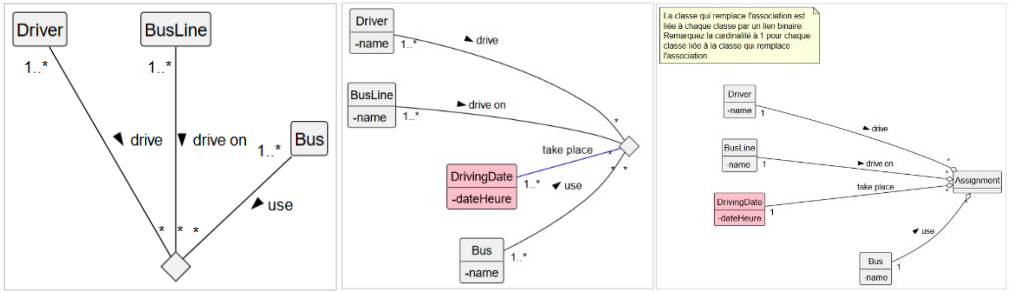
Agrégation



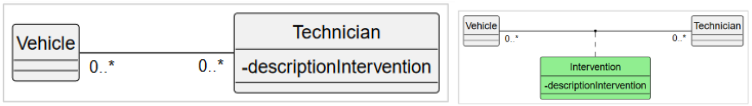
Composition



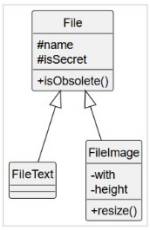
Association N-aire



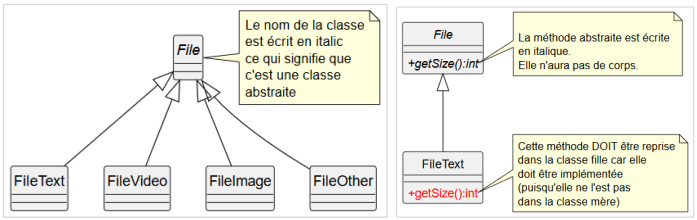
Association porteuse



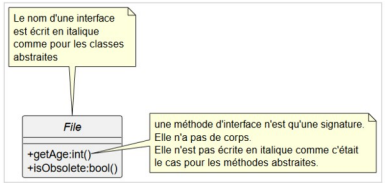
Relation d'héritage



Relation abstraite



Interface



Il est difficile de distinguer une interface d'une classe abstraite (dans les deux cas, le nom est écrit en italique). Il est possible d'utiliser le stéréotype pour une meilleure lisibilité :