LE DIAGRAMME DE CLASSES

Un diagramme de classes en UML (Unified Modeling Language) sert à représenter visuellement la structure d'un système informatique. Il montre les classes, leurs attributs (données ou propriétés), leurs méthodes (actions ou fonctions) et les relations entre les classes (héritage, association, agrégations, compositions). Il est utilisé pour mieux comprendre la programmation orientée objet, pour concevoir et analyser une application plus facilement et pour gagner du temps grâce à une vue claire du système à développer.

En programmation orientée objet (C++, Python, JAVA, etc.), une classe est un modèle qui sert à créer des objets. On peut la comparer à un moule pour fabriquer des objets ayant la même forme, mais chacun avec ses propres valeurs.

Par exemple, si on a une classe appelée « Voiture », elle contient :

- Des attributs (données) → couleur, marque, vitesseMax ;
- Des méthodes (actions) → démarrer (), freiner (), accélérer ().

Chaque objet créé à partir de cette classe sera une voiture spécifique (par exemple : une voiture rouge, de marque Renault, allant jusqu'à 180 km/h).

1. Les éléments principaux d'un diagramme de classes

Les diagrammes de classes sont composés principalement des éléments suivants.

Classes : cela définit le modèle à partir duquel sont créés les objets. Chaque objet est une instance d'une classe particulière, héritant des attributs et des méthodes définies dans la classe.

Attributs : cela fait référence à une variable ou à une donnée associée à une classe. Les attributs définissent les caractéristiques et les propriétés d'un objet qui sont utilisés pour décrire son état ou son comportement.

Méthodes : ce sont des fonctions associées à une classe. Elles définissent le comportement et les actions que les objets instanciés d'une classe peuvent effectuer. Elles sont utilisées pour manipuler les données de l'objet, effectuer des calculs, interagir avec d'autres objets, ou encore pour fournir des fonctionnalités spécifiques.

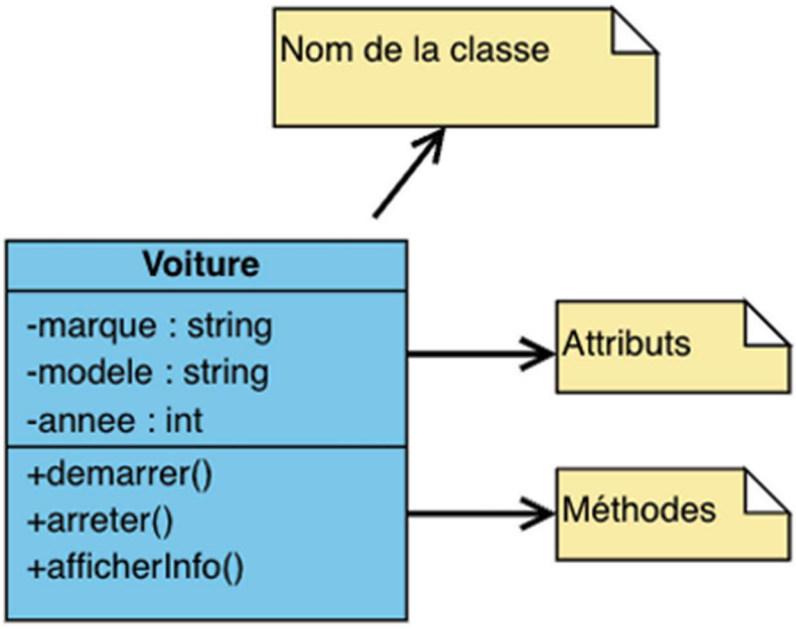
Relations : ce sont les liens entre les classes. Elles aident à modéliser les interactions entre les objets et à structurer efficacement les logiciels. Il en existe plusieurs types :

– **L'association** est une relation entre deux classes où chaque objet d'une classe est lié à des objets d'une autre classe par le biais d'une référence.

– **L’agrégation** – c’est une forme spécifique d’association dans laquelle un objet contient ou est composé d’autres objets, mais ces objets peuvent exister indépendamment de l’objet principal.

– **La composition** – c’est une relation plus forte que l’agrégation dans laquelle un objet est composé d’autres objets qui n’existent que dans le contexte de l’objet principal.

– **L’héritage** – c’est une relation dans laquelle une classe hérite des attributs et méthodes d’une autre classe (appelée classe parent).

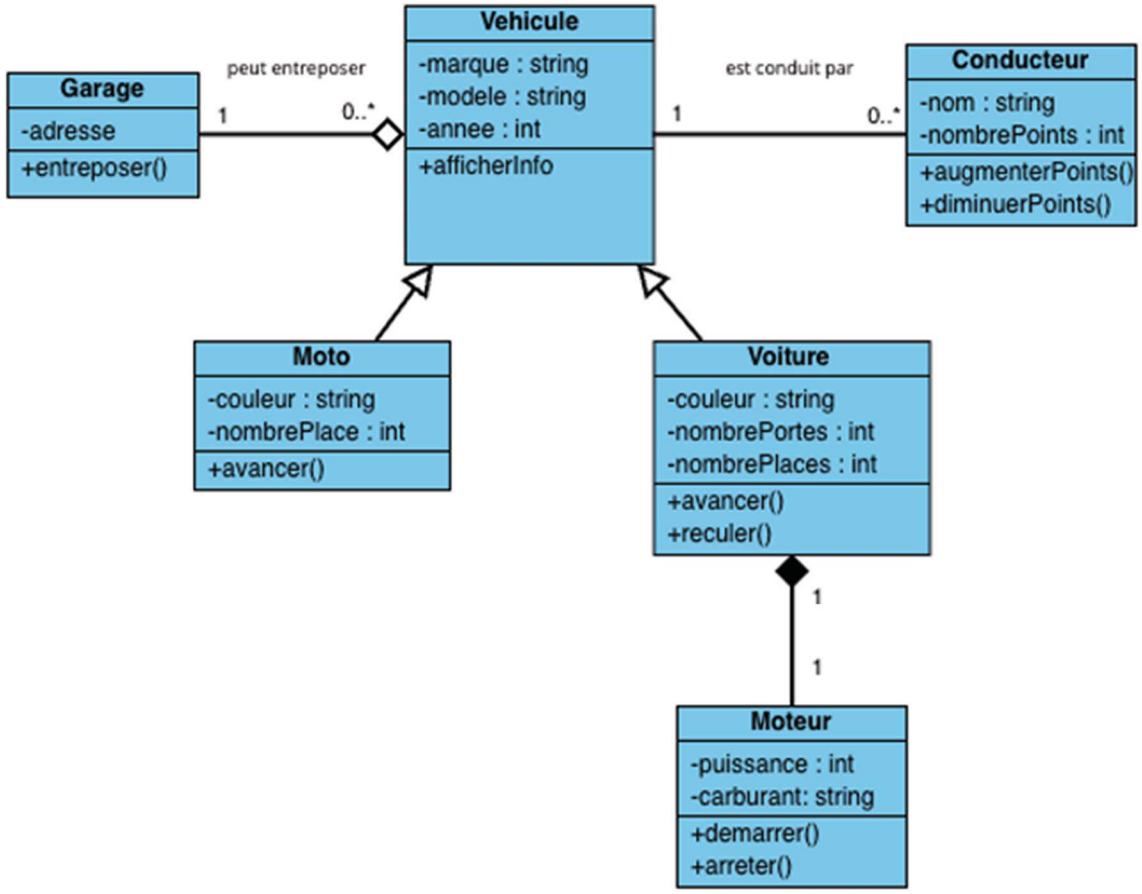


Les attributs et les méthodes peuvent être définis avec différents niveaux de visibilité pour en contrôler l'accès depuis l'extérieur de la classe.

- Public (+) : l'attribut ou la méthode est accessible partout, depuis n'importe quelle autre classe.
- Privé (-) : l'attribut ou la méthode est accessible seulement depuis la classe elle-même.
- Protégé (#) : l'attribut ou la méthode est accessible uniquement depuis la classe et ses classes dérivées (héritage).

2. Exemple de diagramme de classes

Vous trouverez ci-dessous le diagramme de classes d'un logiciel de gestion de véhicules. Cet exemple montre différentes classes ainsi que les relations entre celles-ci et les cardinalités.



Les cardinalités décrivent combien d'instances participent à une relation entre deux classes.

Cardinalité	Signification	Exemple
1	exactement 1	Une voiture a exactement 1 moteur.
0..1	0 ou 1 (optionnel)	Un véhicule peut ou non être conduit.
0..* ou *	0 à l'infini (aucune limite)	Un garage peut entreposer plusieurs véhicules.
1..*	au moins 1	Une voiture doit avoir au moins 1 conducteur.
2..5	entre 2 et 5 inclus	Une équipe peut avoir entre 2 et 5 mécaniciens.

3. Application

Dans le cadre d'un programme en C++, si on veut instancier la classe ("créer un objet") Conducteur, que l'on veut appeler **Laurent**, on écrit :

Conducteur **Laurent**;

On peut aussi l'instancier via un constructeur personnalisé qu'il faudra déclarer dans le code :

Conducteur(string n, int pts) : nom(n), nombrePoints(pts) {}

Ainsi, l'instanciation sera : **Conducteur Laurent("Laurent", 12)**;

Dans notre programme, on aura une instance ("objet") qui s'appellera **Laurent** et qui aura comme attribut de nom : Laurent, et comme nombre de points : 12.