Fiche UML

# CM1 :

# CM2 :

Diagramme de classe : Le plus connu et le plus utilisé. Il décrit de manière graphique la structure du système. Il montre les les classes et leurs relations.

## Les différents types d’association

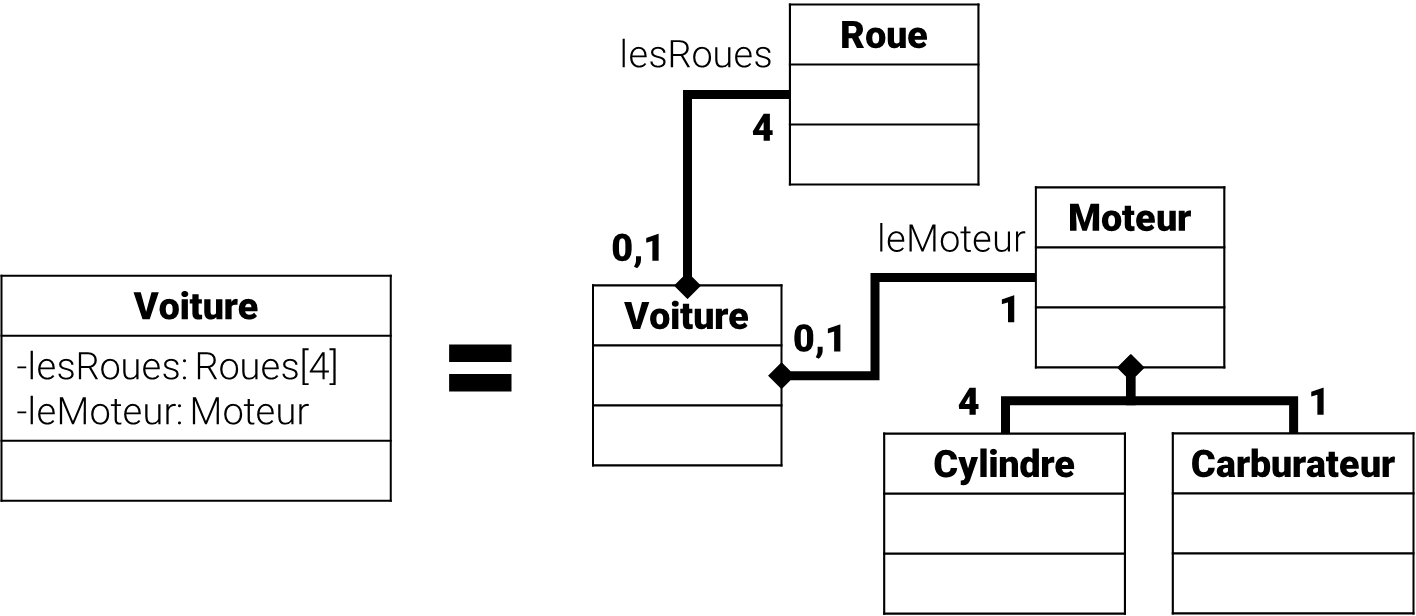
Association  (anonyme/nommée) = abstraction de liens entre objets. Elle permet de mettre en relations 2 classes ou plus. Par défaut, elle est bidirectionnelle c’est-à-dire quelle peut se lire dans les deux sens (il est possible de donner un sens de lecture) mais peut-être rendue unidirectionnelle si-nécessaire. En ce qui concerne la syntaxe, les noms et les rôles et de l’association doivent être écrit en minuscule, commencer par une lettre et peuvent potentiellement contenir des chiffres. Cependant si celui-ci est composé de plusieurs mots alors chaque mot devra commencer par une majuscule : mot1Mot2Mot3…

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Multiplicité / cardinalité   |  |  | | --- | --- | | 1 | Exactement 1 | | \* | Plusieurs (0 à n) | | 0,1 | Optionelle (0 ou 1) | | 1,2,4 | Spécifiée | | 1…10 | Intervalle | |

Agrégation (anonyme et unidirectionnelle) = association plus forte, puisqu’elle met en relation le tout et ses parties. Elle permet de définir des objets, en assemblant des objets plus élémentaires.

Composition (anonyme et unidirectionnelle) = agrégation plus forte puisque les parties du tout sont au plus dans un seul objet composite en même temps. Ainsi si l’objet composite est détruit, toutes les parties sont détruites avec lui.

Information complémentaire : Faire de l’agrégation par valeur équivaut à déclarer un attribut au sein d’une classe.



Héritage : Mécanisme de transition des caractéristiques d’une classe (attribut et méthodes) vers une sous classe, celles-ci peuvent être simple ou multiples. On peut d’ailleurs aussi trouver des hiérarchies d’abstraction (Transitif)  
Exemple : Un téléphone Un téléphone sans fil Un téléphone sans fil avec répondeur

## Démarches de classification

Les deux démarches suivantes de permettent de construire des hiérarchies de classe.

|  |  |
| --- | --- |
|  |  |

Démarche de spécialisation = capture de spécificités. Une classe peut être spécialisée en d’autres classe afin d’y ajouter des caractéristiques spécifiques ou en adapter certaines

Démarche de généralisation = factorisation (se base sur un ou plusieurs points communs entre les différents éléments). Plusieurs classes peuvent etre généralisées en une seule et meme classe qui les factorise, afin de regrouper les caractéristiques communes d’un ensemble de classe ( duplication, réutilisation.  
Exemple : Camion / Moto / Bus

* Attributs communs : plaqueImmatriculation
* Méthode communes : avancer, reculer,seGarer

Celles-ci peuvent être nommées afin de désigner le critère de classification auquel elles appartiennent.

## Contraintes

Plusieurs contraintes peuvent leurs être affectés :

|  |  |
| --- | --- |
|  | L’ensemble des sous-classes est complet |
|  | L’ensemble des sous-classes est incomplet |
|  | Les sous-classes n’ont aucune instance en commun |
|  | Les sous-classes peuvent avoir une ou plieurs instances en commun |

Exemple

|  |  |
| --- | --- |
|  | : Par exemple il manque la classe « Zebre »   : Par exemple un mulet est à la fois un cheval et un âne.   : Le cheval est soit un mâle, soit une femelle, il n’y a pas d’autre alternative possible. La classe « Cheval » est entièrement couverte.   : Il n’existe aucun cheval à la fois mâle et femelle |

## Les classes abstraites

Il s’agit d’une classe dont l’implémentation est incomplète et non directement instanciable. Elle permet notamment de factoriser des caractéristiques.

|  |  |
| --- | --- |
|  | Remarque : Si la fille d’une classe abstraite est elle-même abstraite le contrat d’implémentation d’une méthode abstraite sera transmis à ses filles |

## Les interfaces

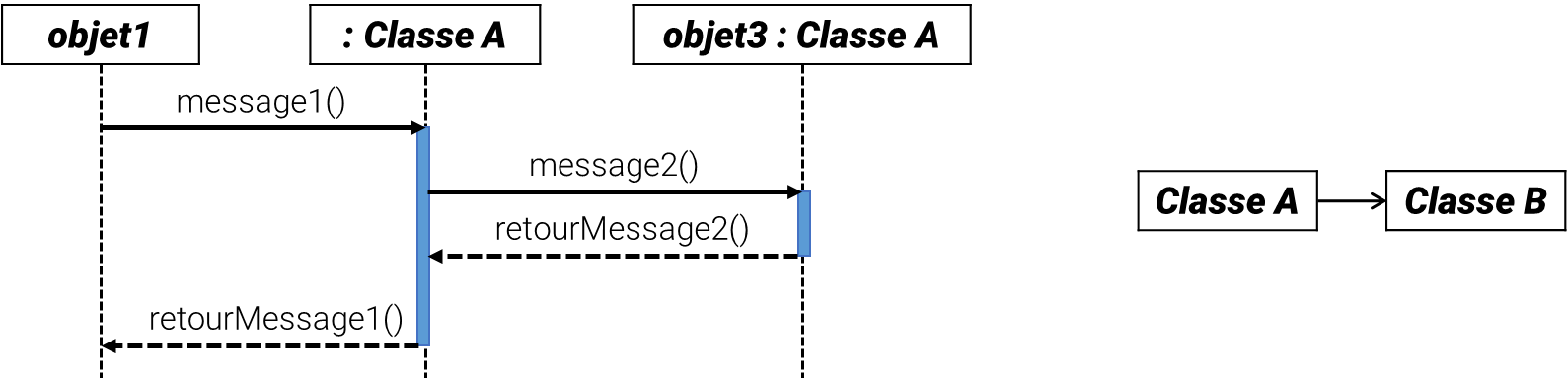
Réalisé par une ou plusieurs classes celle-ci se présente comme une classe totalement abstraite : elle n’a aucun attribut et toutes ses opérations sont abstraites.

…

## Le diagramme de séquence

Il montre une interaction (ensemble des messages échangés entre objets) entre objets, présenté selon l’axe du temps soit, en d’autres termes, les messages échangés et leur chronologie.

Attention : les liens structurels entre objets ne sont pas montrés, ils sont implicites.



Objet : Membre du système qui a une identité, un état et un comportement. Il possède une instance de classe associé avec un nom ou une classe ou les 2. Il possède aussi une ligne de vie qui matérialise son existence (création, utilisation, destruction) et une période d’activité.

Message : Elément de communication unidirectionnelle entre objet qui déclenche une activité chez le récepteur et qui peut transporter de l’information (un ou plusieurs paramètre(s)). Il est le seul moyen de communication entre objets. L’émission et la réception de message sont des évènements. Il peut prendre plusieurs formes :

|  |  |
| --- | --- |
|  | Appel d’opération |
|  | Retour d’opération |
|  | Création d’objet |
|  | Destruction d’objets |
|  | Stimulus/signal (l’émetteur n’attend pas de retour) |

# Vocabulaire

* Classes : modèle pour créer des objets et mettre en œuvre un comportement dans un système.
  + Nom : 1ère ligne d'une forme de classe.
  + Attributs : 2nde ligne d'une forme de classe. Chaque attribut de la classe apparaît sur une ligne distincte.
  + Méthodes : 3ème ligne d'une forme de classe. On les appelle aussi opérations ; elles apparaissent sous forme de liste, chaque opération occupant une ligne différente.
* Signaux : symboles qui représentent les communications à sens unique et asynchrones entre des objets actifs.
* Types de données : classifieurs qui définissent des valeurs de données. Les types de données peuvent modéliser les types primitifs et les énumérations.
* Paquetages : formes conçues pour organiser les classifieurs connexes d'un diagramme. On les symbolise par une grande forme rectangulaire à onglets.
* Interfaces : groupe de signatures d'opération et/ou de définitions d'attributs définissant un ensemble cohérent de comportements. Les interfaces sont semblables à des classes, sauf qu'une classe peut avoir une instance de son type et qu'une interface doit compter au moins une classe pour la mettre en œuvre.
* Énumérations : représentations de types de données définis par l'utilisateur. Une énumération comprend des groupes d'identificateurs qui représentent des valeurs de l'énumération.
* Objets : instances d'une ou plusieurs classes. On peut ajouter des objets à un diagramme de classes pour représenter des instances concrètes ou prototypiques.
* Artefacts : éléments du modèle qui représentent les entités concrètes d'un système logiciel, tels que des documents, des bases de données, des fichiers exécutables, des composants logiciels, etc.

# Diagramme de classe

## Avantages des diagrammes de classes

* Illustrer des modèles de données pour des systèmes.
* Mieux comprendre l’aperçu général des schémas d’une application.
* Exprimer visuellement les besoins d'un système et diffuser cette information dans toute l'entreprise.
* Créer des schémas détaillés qui mettent l'accent sur le code spécifique qui doit être programmé et mis en œuvre dans la structure décrite.
* Fournir une description indépendante de l'implémentation des types utilisés dans un système, qui sont ensuite transmis entre ses composants.

## Composants de base d’un diagramme de classes

Toutes les classes ont des niveaux d'accès différents, en fonction du modificateur d'accès (indicateur de visibilité).

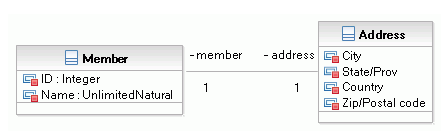
* Public (+)
* Privé (-)
* Protégé (#)
* Paquetage (~)
* Dérivé (/)
* Statique (souligné)

# Diagramme d’objet

Les diagrammes d'objet offrent une image instantanée des instances d'un système et des relations entre ces instances. L'instanciation des éléments de modèle dans un diagramme de classes vous permet d'explorer le comportement d'un système à un point de cohérence.

Les diagrammes d'objets sont des diagrammes structurels UML représentant les instances des discriminants dans les modèles. Ils utilisent une notation semblable à celle utilisée dans les diagrammes de classes. Cependant, les diagrammes de classe représentent les discriminants en cours et leurs relations au sein d'un système alors que les diagrammes d'objets représentent des instances spécifiques de ces discriminants et leurs liens à un point de cohérence donné. Vous pouvez créer des diagrammes d'objets en instanciant les discriminants dans des diagrammes de classes, de déploiement, de composants et de cas d'utilisation.

La figure suivante illustre un diagramme de classes simple qui consiste en en une association de deux classes.



Deux types d’utilisation :

* Diagramme de classes pour décrire la structure d'un système + diagrammes d'objets pour vérifier si le diagramme de classes est complet et exact.
* Diagramme d'objets pour prendre connaissance de certains faits concernant des éléments de modèles particuliers et leurs liens ou pour illustrer des exemples précis des discriminants requis.

## Spécification d’instance

Dans les modèles UML, les spécifications d'instances sont des éléments qui représentent une instance dans le système modélisé. Lorsque vous instanciez un discriminant dans un modèle, vous créez une spécification d'instance qui représente une entité du système modélisé à un point de cohérence, similaire à image instantanée de l'entité. Vous pouvez modéliser les changements apportés à l'entité en créant plusieurs spécifications d'instances, une pour chaque image instantanée.

Les spécifications d'instances peuvent comporter les informations suivantes concernant l'entité :

* La classification de l'entité : discriminants dont l'entité est l'instance
* Le type d'instance, basé sur ses discriminants. Par exemple, une spécification d'instance dont le discriminant est une classe décrit un objet de cette classe alors qu'une spécification d'instance dont le discriminant est une association décrit un lien de cette association
* Les valeurs spécifiques des fonctions structurelles, qui sont représentées par des emplacements

A l'instar des discriminants, les spécifications d'instances ont des attributs que l'on appelle emplacements. Une spécification d'instance peut avoir un emplacement pour chaque fonction structurelle de son discriminant, les fonctions héritées comprises. Vous pouvez définir des valeurs pour chaque emplacement dans une spécification d'instance à partir du moment où un type valide est défini pour cet emplacement.

* La figure de gauche illustre une classe appelée Member. Elle comporte deux attributs, ID et Name. L'attribut ID est de type Integer et l'attribut Name est de type UnlimitedNatural.
* La figure de droite illustre la spécification d'instance correspondante comportant deux emplacements et valeurs. La valeur de l'emplacement ID est 1 et la valeur de l'emplacement Name est John Q Public.

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement

Figure : A gauche la classe "Member", à droite l'instance "MemberInstance"