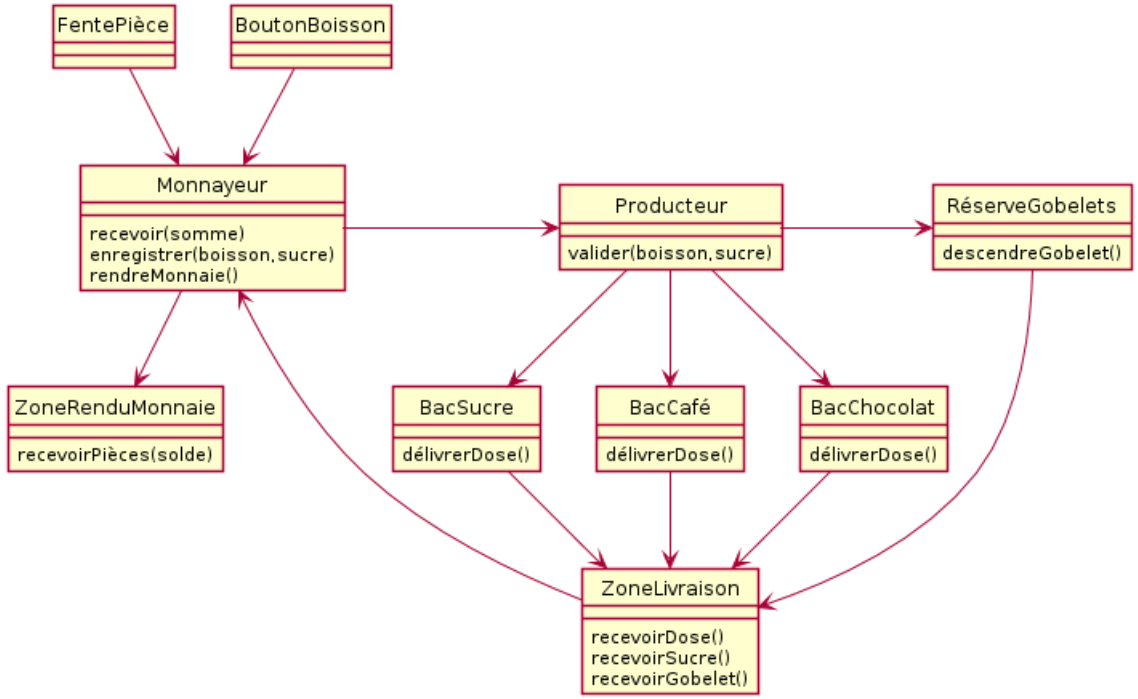
Diagrammes de classes

Les classes sont des *types* d'objets.



Objets

Objets du monde réel :

- Le stylo que l'un tient en main.
- Le stylo que quelqu'un d'autre a dans sa trousse.
- La facture papier que vous avez reçue à la livraison de votre dernière commande.

Objets informartiques

En informatique, les objets ne désignent pas de éléments matériels concrets.

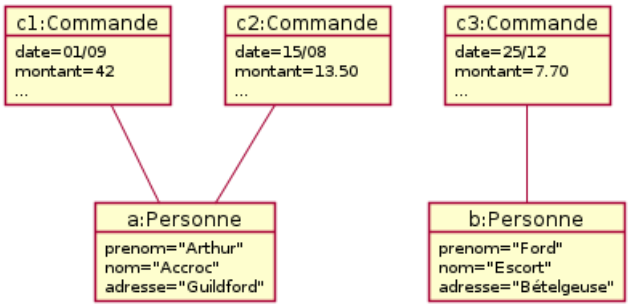
Un objet est un conteneur, qui possède sa propre existence et incorpore des données et des mécanismes en rapport avec une chose tangible. C'est le concept central de la programmation et de la conception orientée objet.

Objets informatiques

- Une structure de données rassemblant les informations disponibles sur une personne (adresse="Guildford", nom="Accroc", prénom="Arthur" etc).
- Une structure de données rassemblant les informations disponibles sur une autre personne (adresse="Bételgeuse", nom="Escort", prénom="Ford" etc).
- Une représentation informatique du stylo tenu en main (avec son propre niveau d'usure, sa propre couleur etc).

- L'enregistrement dans une base de données de votre dernière commande (avec sa propre date, son propre montant, ses propres articles etc).

Objets en UML



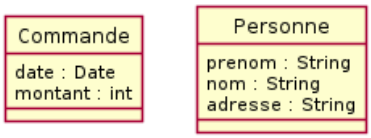
- c1, c2 et c3 sont des commandes
- a et b sont des personnes
- c1, c2 et c3 ont un montant, une date etc…
- a et b ont un nom, un prénom et une adresse

Syntaxe du nom des objets : `nom:Type`

- Le nom peut-être omis, ou le type, mais pas les deux
- Les deux points restent dans tous les cas de figure, même sans rien devant ou derrière

Classes

Une classe définit un type d'objet.



Une classe déclare donc des *propriétés* communes à un ensemble d'objets :

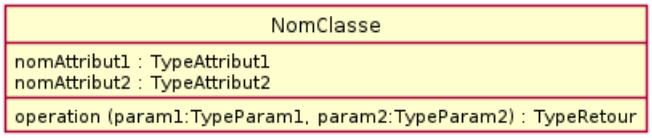
- Les *attributs* correspondant à des variables associées aux objets de la classe
- Les *opérations* correspondant à des opérations (fonctions) associées aux objets de la classe

Exemples de propriétés :

- Toutes les commandes ont une date (attribut).
- Toutes les personnes ont un nom (attribut).
- Toutes les personnes ont un numéro (attribut).
- Tous les articles peuvent être achetés (opération).

Représentation d'une classe en UML

Une classe est composée d'un nom, d'attributs et d'opérations.



- Le nom d'une classe commence par une majuscule
- Le nom d'une propriété commence par une minuscule
- Les types de base (int, long, float, double, boolean) sont en minuscules
- Il n'a pas d'espace dans les noms de classes ou de propriétés
- Pour les noms composés, on fait commencer chaque mot par une majuscule

Concepts et instance

Une instance est la concrétisation d'un concept abstrait.

Instanciation d'une classe :

- Une *classe* est la spécification d'un type
- Un *objet* est une instance avec ses propres attributs et son propre état

Instanciation d'une association :

- Une *association* entre deux classes est un concept : elle décrit quels types d'objets et combien peuvent être associés à d'autres
- Un *lien* est une instance : c'est un lien concret entre deux objets particuliers

Notation abrégée d'une classe

Si une classe est déjà définie, il est possible de la représenter simplement, sans ses propriétés.

Relations entre classes

- La relation d'*héritage* est une relation de généralisation/spécialisation permettant l'abstraction de concepts.
- Une *association* représente une relation possible entre les objets d'une classe.
- Une relation de *composition* décrit une relation de contenance et d'appartenance.
- Une *dépendance* est une relation unidirectionnelle exprimant une dépendance sémantique entre les éléments du modèle (flèche ouverte pointillée).

Association

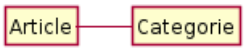
Une association est une relation structurelle entre objets.

- Une association est souvent utilisée pour représenter les liens possibles entre objets de classes données
- Elle est représentée par un trait entre classes
- Elle est souvent dirigée par une flèche

La flèche indique ici que la relation est *unidirectionnelle* : les objets de classe `Article` connaissent ceux de la classe `Commentaire` auxquels ils sont liés, mais pas l'inverse.

Association bidirectionnelle

Certaines associations sont bidirectionnelles :

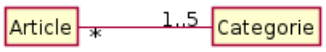


L'absence de flèche indique ici que l'on peut accéder aux catégories à partir des articles qui leur sont liés, et inversement.

Multiplicités

Les multiplicités permettent de contraindre le nombre d'objets intervenant dans les instanciations des associations. On en place de chaque côté des associations.

- Une multiplicité d'un côté spécifie combien d'objets de la classe du côté considéré sont associés à un objet donné de la classe de l'autre côté.
- Syntaxe : `min..max`, où `min` et `max` sont des nombres représentant respectivement les nombre minimaux et maximaux d'objets concernés par l'association.



Ici, le `1..5` s'interprète comme *à un objet donné de la classe `Article`, on doit associer au minimum 1 objet de la classe `Categorie` et on peut en associer au maximum 5*.

Ecriture des multiplicités

Certaines écritures possibles :

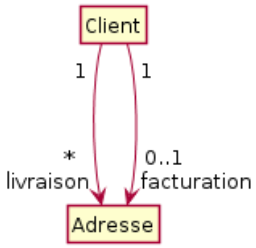
- `*` à la place de `max` signifie *plusieurs* sans préciser de nombre.
- `n..n` se note aussi `n` pour *exactement n*
- `0..*` se note aussi `*`

Exemples :

- `1..*` : au minimum 1 mais sans maximum
- `1..2` : entre un et deux (mais jamais 0, par exemple)
- `*` : autant qu'on le souhaite

Rôles

On peut donner à une classe un rôle dans une association. C'est surtout utile quand plusieurs associations concernent les mêmes classes en qu'en conséquence, de mêmes objets peuvent être liés par des modalités différentes.



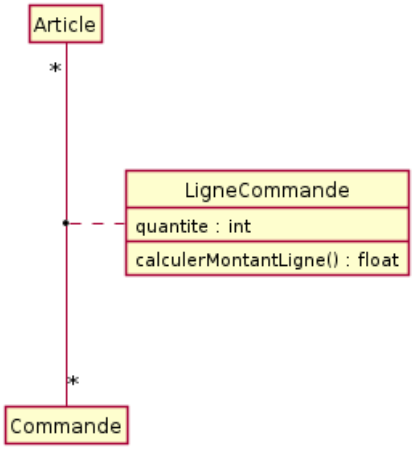
Ici, des adresses peuvent être liées aux clients :

- Les adresses jouent le rôle d'adresses de livraison ou d'adresses de facturation
- Rien n'empêche qu'une adresse soit à la fois une adresse de livraison et une adresse de facturation

Classe-association

Pour faire porter des informations par une association, on emploie une *classe-association*.

Graphiquement, on la relie à l'association avec des pointillés.

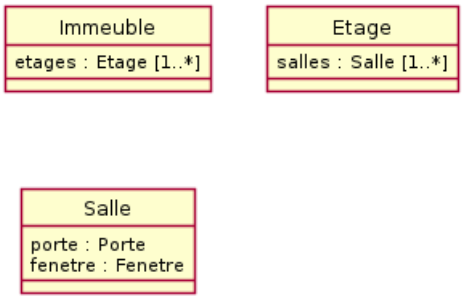


Relations *est une partie de*

Les objets correspondant aux attributs d'une classe *font partie* des objets de la classe en question :

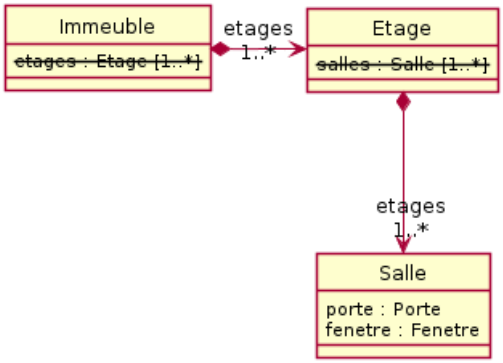
- Les attributs définissent des parties de la classe

- Ils la *composent*



Expression des compositions avec des relations

Il est possible de représenter plus explicitement les relations de *composition* entre classes.



De manière équivalente à la définition d'attributs, on peut utiliser des compositions *unidirectionnelles*

- Il ne faut pas représenter *à la fois* des attributs et des compositions pour la même relation
- Le nom de l'attribut devient le rôle de la classe composante
- La multiplicité de l'attribut devient le rôle de la classe composante (sans multiplicité explicite, `1`)