

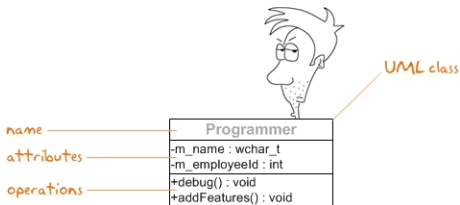
# Introduction à la modélisation UML

## Module RCPI01


Sebastien.Kramm@univ-rouen.fr

IUT R&T Rouen

2018-2019



- 1 Introduction : le probleme
- 2 Représentation d'une classe en UML
- 3 Diagrammes UML
  - Diagrammes de cas d'utilisation
  - Diagramme de classes
  - Diagrammes de séquence et de collaboration

- Systèmes informatiques : de plus en plus complexes, dépassent la compréhension et la maîtrise par un seul individu.  
⇒ Le recours à un modèle conceptuel s'avère indispensable.
- Un **modèle** est une représentation abstraite d'un système, qui facilite l'étude et la communication entre intervenants au sein d'un projet.
- Utilisé et progressivement enrichi dans toutes les étapes d'un projet : spécification, analyse, conception, test, intégration et rétro-ingénierie.
- UML (Unified Modeling Language) est le **standard industriel** de modélisation orientée objet.
- Objectifs :
  - ① **Représenter** des systèmes entiers (au-delà du seul logiciel) par des concepts objets,
  - ② Créer un langage de modélisation utilisable par les humains **et** les machines,
  - ③ Établir un **couplage** explicite entre les concepts et le produit  UNIVERSITÉ DE ROUEN

- Un objet est une entité aux frontières précises
  - Il est identifié (avec un nom)
  - Il est insécable (il doit être complet)
- Un objet est modélisé par une classe, qui est dotée :
  - d'un ensemble d'attributs qui caractérisent son état,
  - d'un ensemble de méthodes (d'opérations) qui définissent son comportement.
- Un objet est une instance de classe (une occurrence d'un type abstrait).

Modéliser : comprendre et représenter

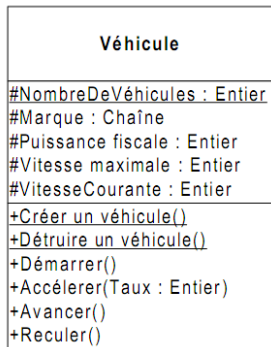
- Un modèle est une abstraction de la réalité.
- Abstraction : ensemble des caractéristiques essentielles d'une entité, retenues par un observateur.
- Un modèle est une vue **subjective** mais **pertinente** de la réalité.
- Un modèle ne représente pas une réalité absolue mais reflète les aspects importants de la réalité, il en donne donc une vue juste et pertinente.

- UML est un langage graphique, formel et normalisé
  - gain de précision
  - gage de stabilité
  - encourage l'utilisation d'outils
- UML est un support de communication performant
  - Cadre l'analyse
  - Facilite la compréhension de représentations abstraites complexes
  - Son caractère polyvalent et sa souplesse en font un langage universel

- 1 Introduction : le probleme
- 2 Représentation d'une classe en UML
- 3 Diagrammes UML
  - Diagrammes de cas d'utilisation
  - Diagramme de classes
  - Diagrammes de séquence et de collaboration

# Représentation d'une classe

- Une **classe** représente le modèle de l'objet ("moule").
  - la liste de ses caractéristiques et de leur type ⇒ **attributs**,
  - les choses qu'il peut faire ⇒ **méthodes** (fonctions associées à l'objet).
- Est représentée par un rectangle découpé en trois parties



Nom de la classe

Description des attributs  
ou données membres

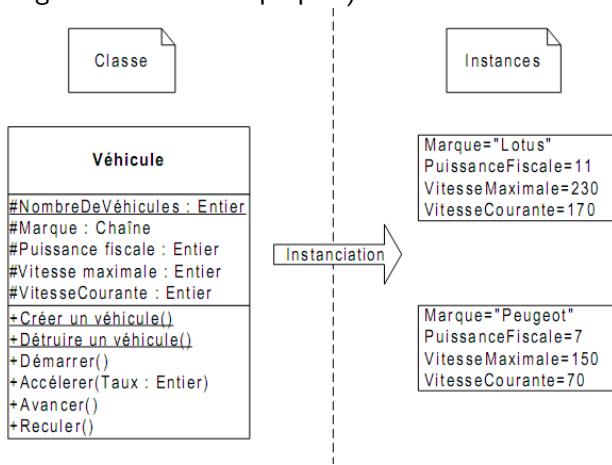
Description des méthodes  
= code associé au données

src : B. Garcia



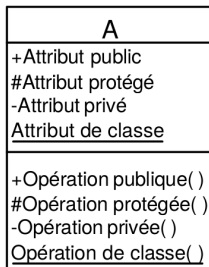
# Classe $\neq$ Instance

Un **objet** est une **instance** de la classe (création en mémoire d'une zone pour le stockage de ses données propres).



src : B. Garcia

- Symbole de visibilité des attributs et des opérations
  - + : public, rend l'élément visible à tous les clients de la classe
  - # : protégé, qui rend l'élément visible aux classes dérivées
  - - : privé, rend l'élément visible à la classe seule



src : PA Muller

- 1 Introduction : le probleme
- 2 Représentation d'une classe en UML
- 3 Diagrammes UML
  - Diagrammes de cas d'utilisation
  - Diagramme de classes
  - Diagrammes de séquence et de collaboration

- 5 diagrammes statiques (ou structurels)
  - diagrammes de cas d'utilisation ("*Use Cases* ")
  - diagrammes d'objets
  - diagrammes de classes
  - diagrammes de composants
  - diagrammes de déploiement
- 4 diagrammes dynamiques (ou comportementaux)
  - diagrammes de collaboration
  - diagrammes de séquence
  - diagrammes d'états-transitions
  - diagrammes d'activités

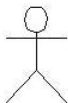
(Ce cours n'en présente que quatre.)

- 1 Introduction : le probleme
- 2 Représentation d'une classe en UML
- 3 Diagrammes UML**
  - Diagrammes de cas d'utilisation
  - Diagramme de classes
  - Diagrammes de séquence et de collaboration

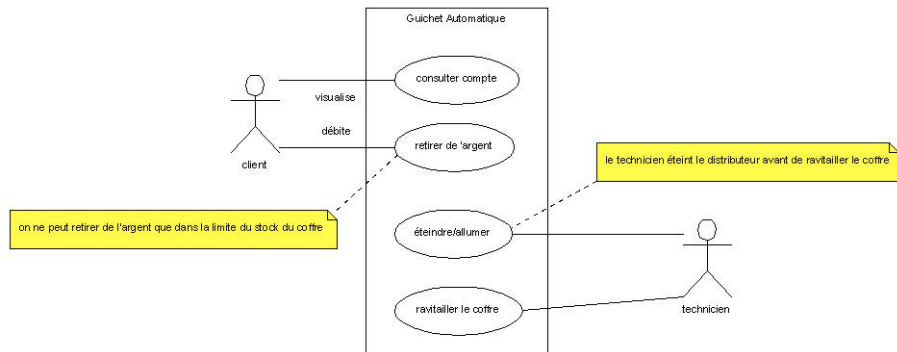
# Diagrammes de cas d'utilisation ("use cases" : UC)

- Expression du comportement du système (actions et réactions), selon le point de vue de l'utilisateur.
- Décrivent le système et les relations entre le système et l'environnement.
- Intérêts :
  - Permettent de délimiter les frontières du système
  - Constituent un moyen d'exprimer les besoins d'un système
  - Utilisés par les utilisateurs finaux pour exprimer leurs attentes et leurs besoins
  - Permettent d'impliquer les utilisateurs dès les premiers stades du développement
  - Constituent une base pour les tests fonctionnels

- Acteur : entité (personne ou système) externe qui échange de l'information (entrée/sortie)
  - L'acteur peut consulter ou modifier l'état du système.
  - En réponse à l'action d'un acteur, le système fournit un service qui correspond à son besoin.
  - Les acteurs peuvent être classés (hiérarchisés) via le concept d'héritage.
- Use case : ensemble d'actions réalisées par le système, en réponse à une action d'un acteur
  - Les uses cases peuvent être structurés.
  - Les uses cases peuvent être organisés en paquetages (packages).
  - L'ensemble des use cases décrit les objectifs (le but) du système.



# Exemple 1 : automate bancaire (DAB)

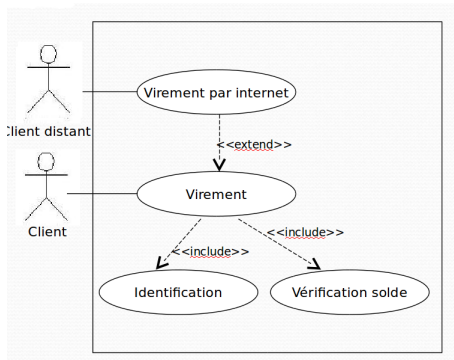




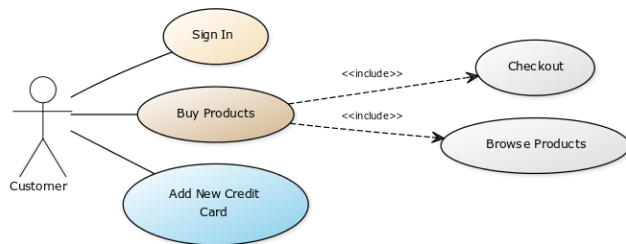
# Relations entre UC

Ce mécanisme est extensible via les stereotypes (`<< ... >>`) d'UML :

- Relation d'utilisation `<< include >>` : Le cas d'utilisation en contient un autre
- Relation d'extension `<< extend >>` : Le cas d'utilisation étend et précise les objectifs (le comportement) d'un autre cas

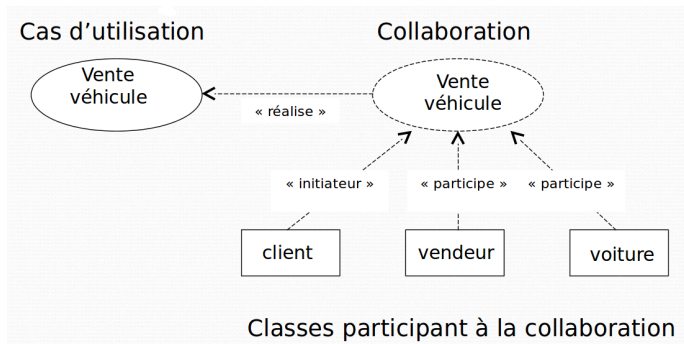


## Exemple 2 : site de vente en ligne



# Collaborations entre UC

- Interaction entre objets dont le but est de répondre à un besoin d'un utilisateur (réaliser un objectif du système).
- Représente les classes qui participent à la réalisation d'un cas d'utilisation.

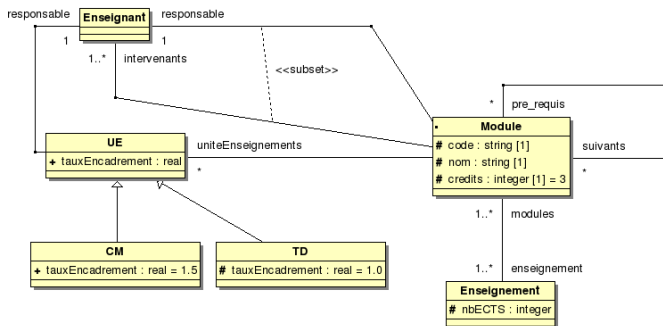


- ① Recenser les acteurs et les hiérarchiser.
- ② Pour chaque acteur, identifier les UC principaux qu'il déclenche.
- ③ Organiser les UC grâce aux relations de généralisation, d'inclusion et d'extension.
- ④ Construire le diagramme des UC.

- 1 Introduction : le probleme
- 2 Représentation d'une classe en UML
- 3 Diagrammes UML**
  - Diagrammes de cas d'utilisation
  - Diagramme de classes**
  - Diagrammes de séquence et de collaboration

# Diagramme de classe

- Montre une collection de classes et montre les relations entre eux.
- Aucun aspect dynamique ou temporel.
- Pour un modèle complexe, on en donne plusieurs, selon les cas d'utilisation.
- Montre les associations et aggregations entre classes.



- Plusieurs types d'associations
  - Association simple ("navigabilité")
  - Agregation
  - Composition
  - Héritage
- Cardinalité des associations : chaque rôle d'une association porte une indication de multiplicité qui montre combien d'objets de la classe considérée peuvent être liés à un objet de l'autre classe.

|        |                             |
|--------|-----------------------------|
| 1      | Un et un seul               |
| 0..1   | Zéro ou un                  |
| M .. N | De M à N (entiers naturels) |
| 0 .. * | De zéro à plusieurs         |
| 1 .. * | D'un à plusieurs            |

# Association : cas général

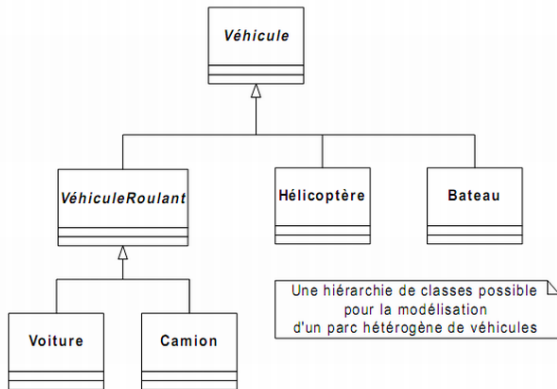
- Les associations représentent des relations structurelles entre classes d'objets, sans préjuger de l'implémentation de cette relation.
- La plupart des associations sont binaires : elles connectent deux classes.
- On peut nommer les rôles que jouent les deux classes l'une pour l'autre.





# Cas particulier : héritage

- L'héritage permet de factoriser des comportements et des attributs
- Correspond à une relation de type "...est un..."



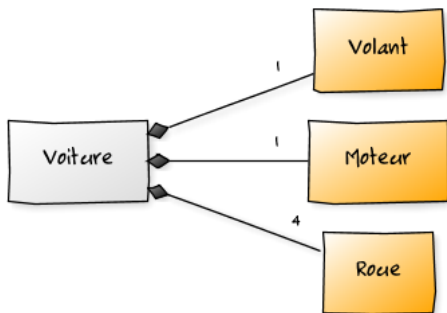
src : B. Garcia

- L'héritage reste un cas particulier :
  - Une classe "voiture" peut dériver (hériter) d'une classe "véhicule".
  - Mais une classe "roue" ne peut pas dériver d'une classe "voiture" ! (l'inverse non plus...)
- Par contre une classe "voiture" pourra **contenir** d'autres objets :
  - 4 objets de la classe "roue",
  - 1 objet de la classe "moteur",
  - 1 objet de la classe "volant".

On parle de **relation d'association**, et plus précisément :

- de **composition** si la destruction de l'objet entraîne la destruction des éléments contenus,
- d'**agregation** si les objets contenus ont une existence indépendante.

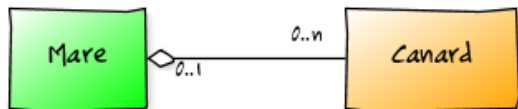
- Si l'objet "Voiture" est détruit, les objets contenus le seront aussi.



- La composition est notée par un losange plein du côté de la classe "conteneur".
- Sens de lecture : on dira : une "voiture" est composée de 1 "volant", 1 "moteur, 4 "roues".



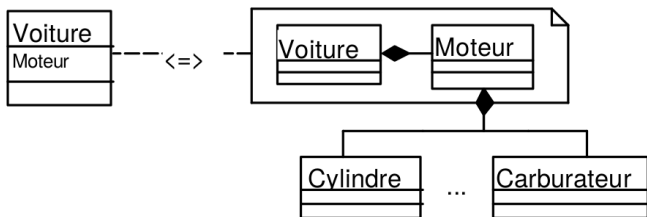
- Les objets ont une existence indépendante :
  - la mare "contient" des canards,
  - la mare existe sans canards,
  - les canards existent sans la mare.



- L'agregation est notée par un losange vide du côté de la classe "conteneur".

# Composition ou attributs ?

- La relation de composition est sémantiquement identiques aux attributs
- La notation par composition s'emploie dans un diagramme de classes lorsqu'un attribut participe à d'autres relations dans le modèle.

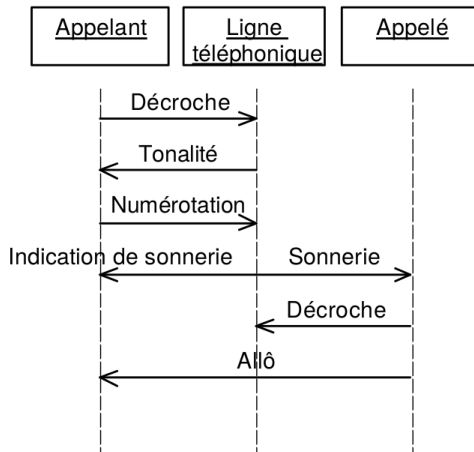


src : P.A. Muller

- 1 Introduction : le probleme
- 2 Représentation d'une classe en UML
- 3 Diagrammes UML**
  - Diagrammes de cas d'utilisation
  - Diagramme de classes
  - Diagrammes de séquence et de collaboration

- Ces deux notations sont duales : on se contente souvent de l'une des deux
  - Diagrammes de séquence : mettent l'accent sur le classement chronologique des messages de collaboration d'instance
  - Diagrammes de collaboration : mettent l'accent sur l'organisation structurelle des éléments qui envoient et reçoivent des messages

# Diagrammes de séquence

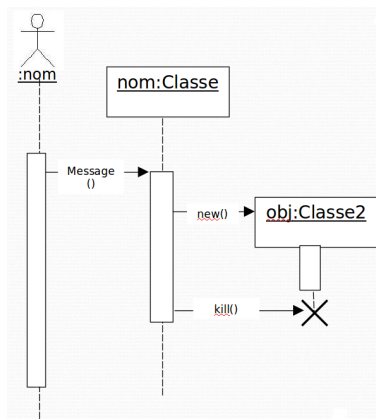




- Représente des collaborations entre objets selon un point de vue temporel, montre la chronologie des envois de messages.
- Contrairement au diagramme de collaboration, on n'y décrit pas le contexte ou l'état des objets, la représentation se concentre sur l'expression des interactions.
- Les diagrammes de séquences peuvent servir à illustrer **un** cas d'utilisation.
- L'ordre d'envoi d'un message est déterminé par sa position sur l'axe vertical du diagramme ; le temps s'écoule "de haut en bas" de cet axe.

# Diagrammes de séquence : convention graphique

- Acteur
- Objet
- Ligne de vie
- Bande d'activation
- Envoi de message
- Création dynamique
- Suppression d'un objet



# Diagrammes de séquence : messages

Les messages peuvent être asynchrones (non-bloquants) ou synchrones (bloquants)

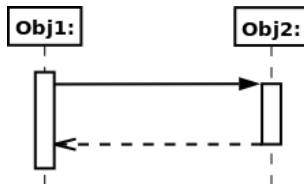
## Messages asynchrones

- n'attendent pas de réponses,
- ne bloquent pas l'émetteur,
- aucune information si le message arrivera, quand, et si il sera traité.



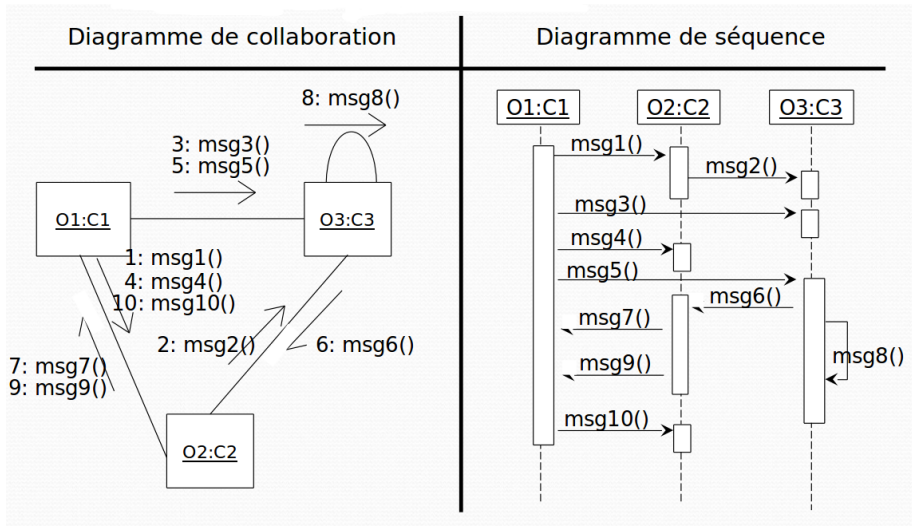
## Messages synchrones

- L'émetteur reste bloqué, tant que le récepteur n'a pas acquitté ou répondu.



- On peut dessiner une barre oblique quand le temps de transmission n'est pas négligeable devant la dynamique du système.

# Équivalence diagramme de séquence / de collaboration



src : T. Lelore

