TCH099 – CHAPITRE 2
LA MODÉLISATION

Anis Boubaker, Ph.D.
Maître d'enseignement
École de Technologie Supérieure

PLAN DU CHAPITRE

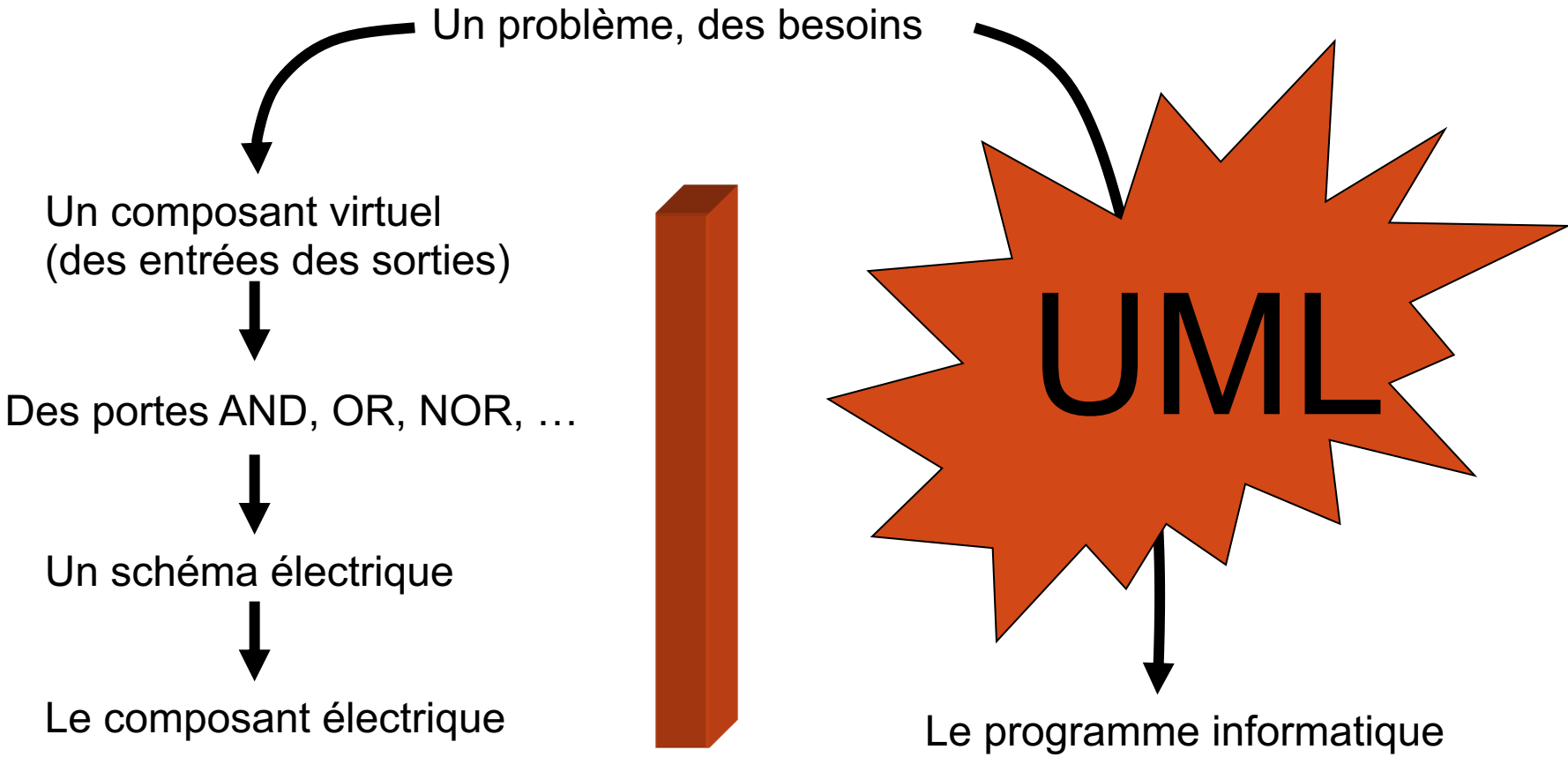
1. Pourquoi modéliser?
2. Diagramme de cas d'utilisation
3. Diagramme de séquences
4. Diagramme de classes

POURQUOI MODÉLISER?

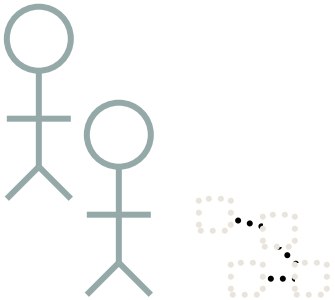
Présentation rapide

ÉTS

L'ELECTRICIEN ET L'INFORMATICIEN



L'INDUSTRIE LOGICIELLE AUJOURD'HUI



architects

tool vendors

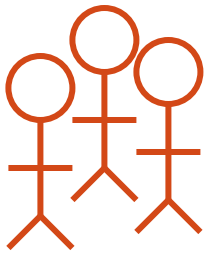
middleware providers

standardization organization

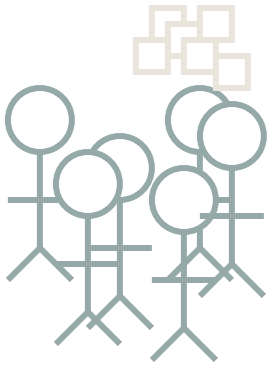
De nombreux acteurs

- des préoccupations différentes
- des métiers différents
- des compétences variées
- des outils variés

Séparation des préoccupations

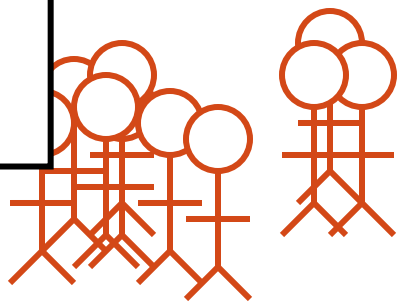


site administrators



developers

application testers



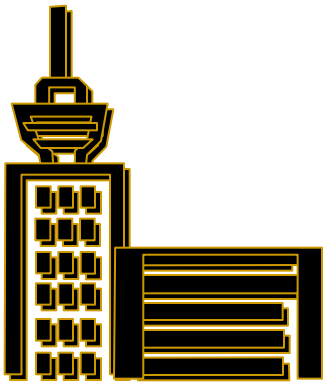
end users

Séparations des préoccupations

Point de vue des promoteurs

Point de vue des locataires

Point de vue des pompiers



système

Point de vue des plombiers

Point de vue du cadastre

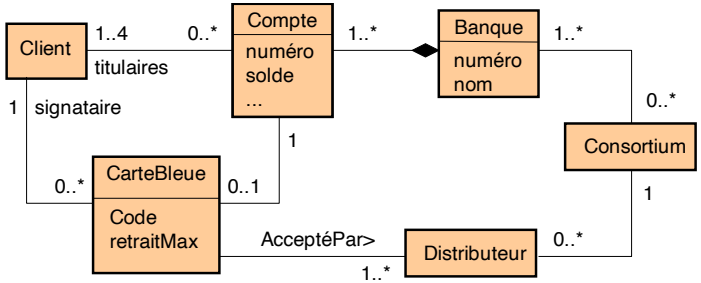
Point de vue des électriciens

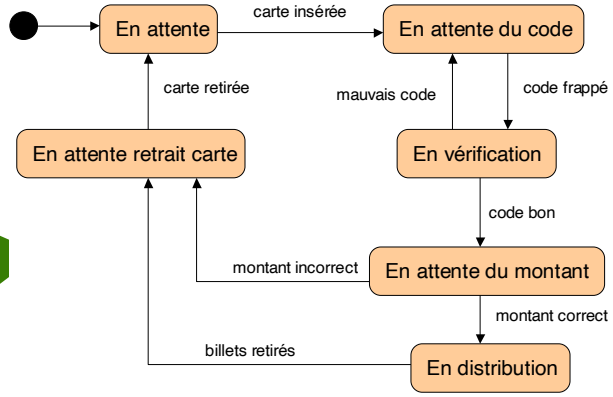
Point de vue des assureurs

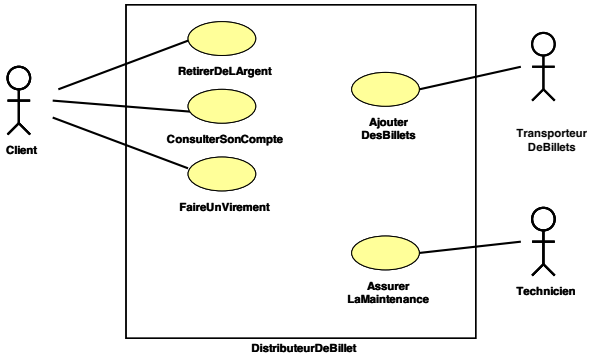
Point de vue des paysagistes

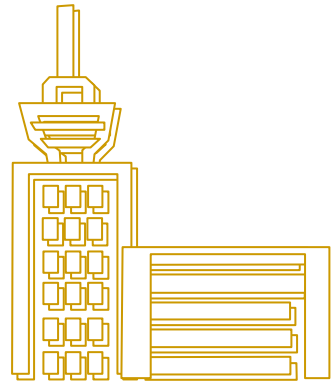
Point de vue des notaires

Point de vue des architectes

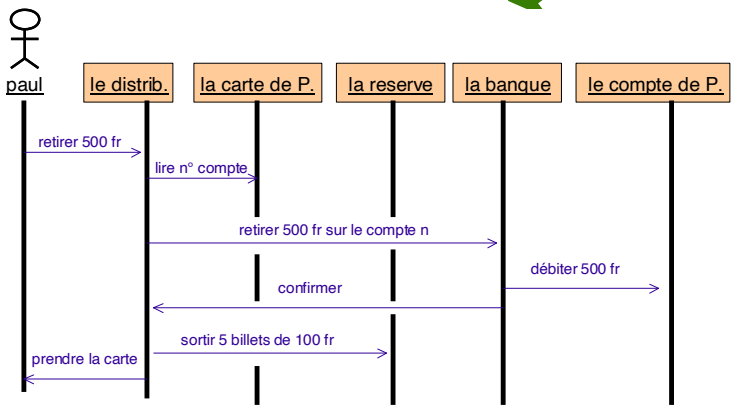

Client
1..4
0..*
titulaires
Compte
numéro
solde
...
1..*
Banque
numéro
nom
1..*
0..*
Consortium
1
1
signataire
0..*
CarteBleue
Code
retraitMax
0..1
AcceptéPar>
Distributeur
0..*
1..*

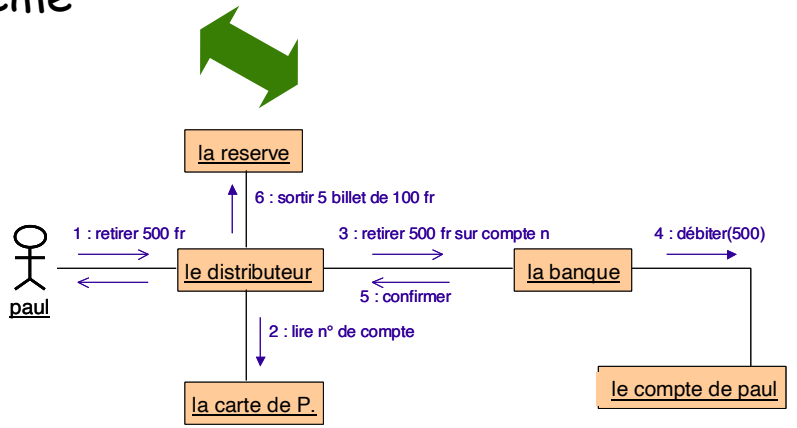

En attente
carte insérée
En attente du code
carte retirée
mauvais code
code frappé
En attente retrait carte
En vérification
code bon
montant incorrect
En attente du montant
montant correct
billets retirés
En distribution

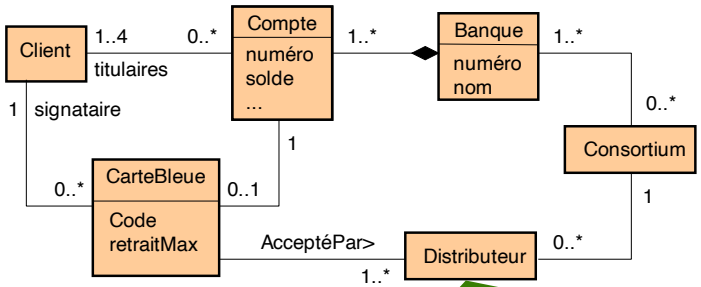

Client
RetirerDeLArgent
ConsulterSonCompte
FaireUnVirement
Ajouter
DesBillets
Transporteur
DeBillets
Assurer
LaMaintenance
Technicien
DistributeurDeBillet

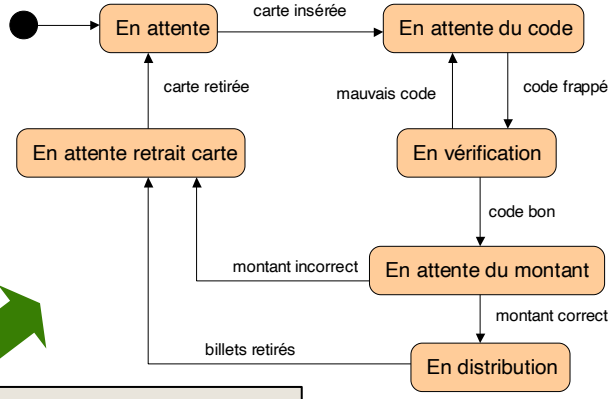


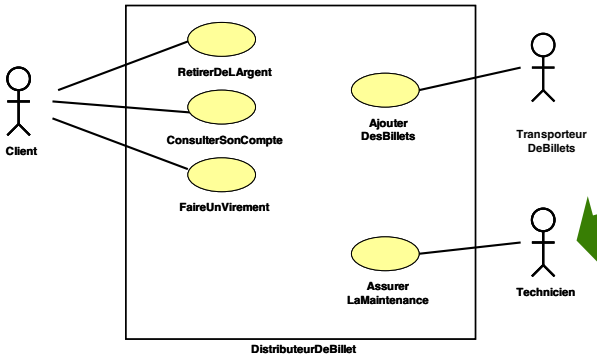
système


paul
le distrib.
la carte de P.
la reserve
la banque
le compte de P.
retirer 500 fr
lire n° compte
retirer 500 fr sur le compte n
débiter 500 fr
confirmer
sortir 5 billets de 100 fr
prendre la carte


la reserve
6 : sortir 5 billet de 100 fr
1 : retirer 500 fr
3 : retirer 500 fr sur compte n
4 : débiter(500)
paul
le distributeur
la banque
5 : confirmer
2 : lire n° de compte
la carte de P.
le compte de paul





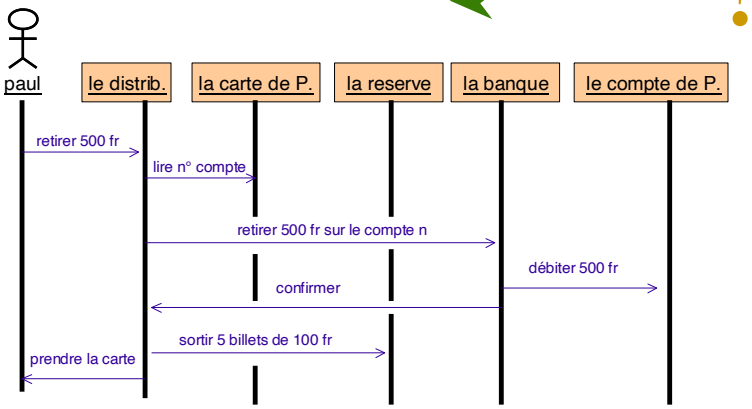


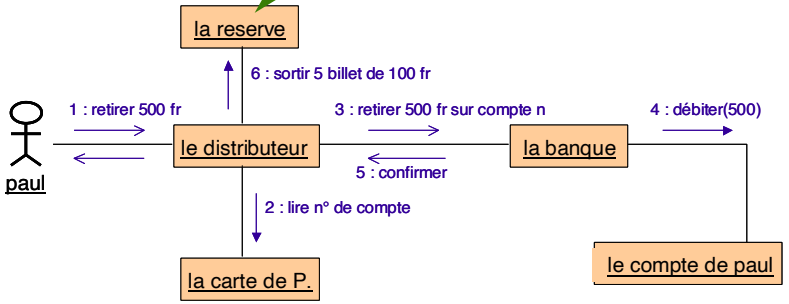
Oui, Oui, Oui, mais ...

pour "l'informaticien »,

la seule chose importante dans le logiciel c'est ...

Le code





UML

- UML (*Unified Modeling Language*) est le langage de modélisation OO le plus connu et le plus utilisé au monde

- UML est l'accomplissement de la fusion des trois langages (+ méthodes)
  - Booch,
  - OMT (*Object Modeling Technique*)
  - et OOSE (Object Oriented Software Engineering)

UML = 14 DIAGRAMMES

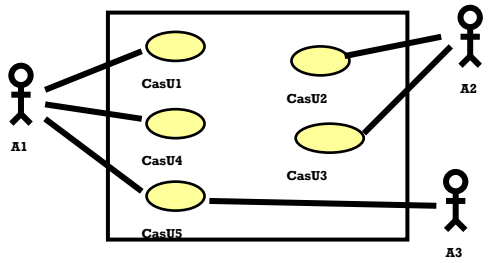
- Diagrammes structurels:
  - Diagramme de classes et d'objets
  - Diagramme de paquetages
  - Diagramme de composants et de déploiement
  - Diagramme de structure composite (structure interne d'une classe + collaboration)
  - Diagramme de profil (spécialiser un MM pour un domaine)
- Diagrammes comportementaux :
  - Diagramme de cas d'utilisation
    - Diagramme d'états-transitions (Automate à états finis)
    - Diagramme d'activités
- Diagrammes d'interaction:
  - Diagramme de séquences
  - Diagramme de communication (DC)(collaboration en UML 1)
  - Diagramme global d'interaction (variante du diagramme d'activité pour les DC)
  - Diagramme de temps (décrire les variations des objets au cours du temps)

DIAGRAMME DE CAS D'UTILISATION

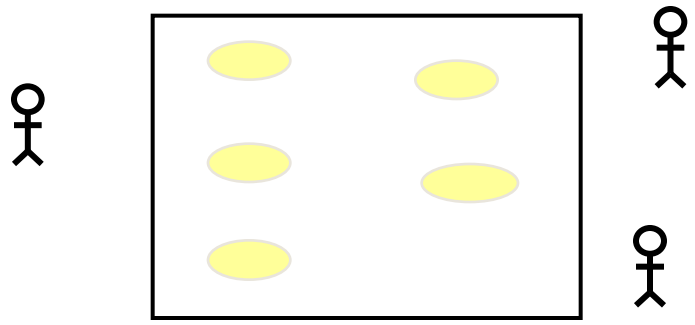
MODÈLE DES CAS D'UTILISATION

- Buts :
  - modéliser le point de vue des **utilisateurs**
  - définir les limites précises du **système**
  - définir le système par rapport à son **environnement**,

- Notation très simple, compréhensible par tous, y compris le client



ELEMENTS DE BASE



Acteurs

Cas d'utilisation

Système

Important : Il ne faut pas ajouter d'association entre acteurs

EXEMPLE

Client

Transporteur DeBillets

Technicien

Distributeur De Billet

CAS D'UTILISATION (CU)

- Cas d'utilisation (CU)
  - une manière d'utiliser le système
  - une suite d'interactions entre un acteur et le système

- Correspond à une fonction du système **visible par l'acteur**
- Permet à un acteur d 'atteindre **un but fonctionnel**
- Doit être **utile en soi**

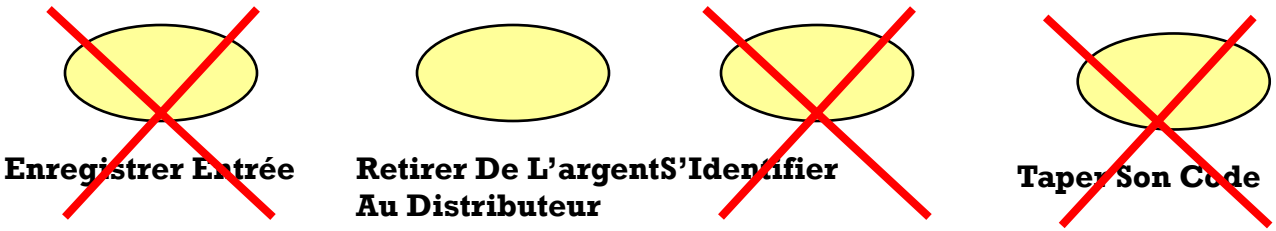


DIAGRAMME DE CAS D 'UTILISATION

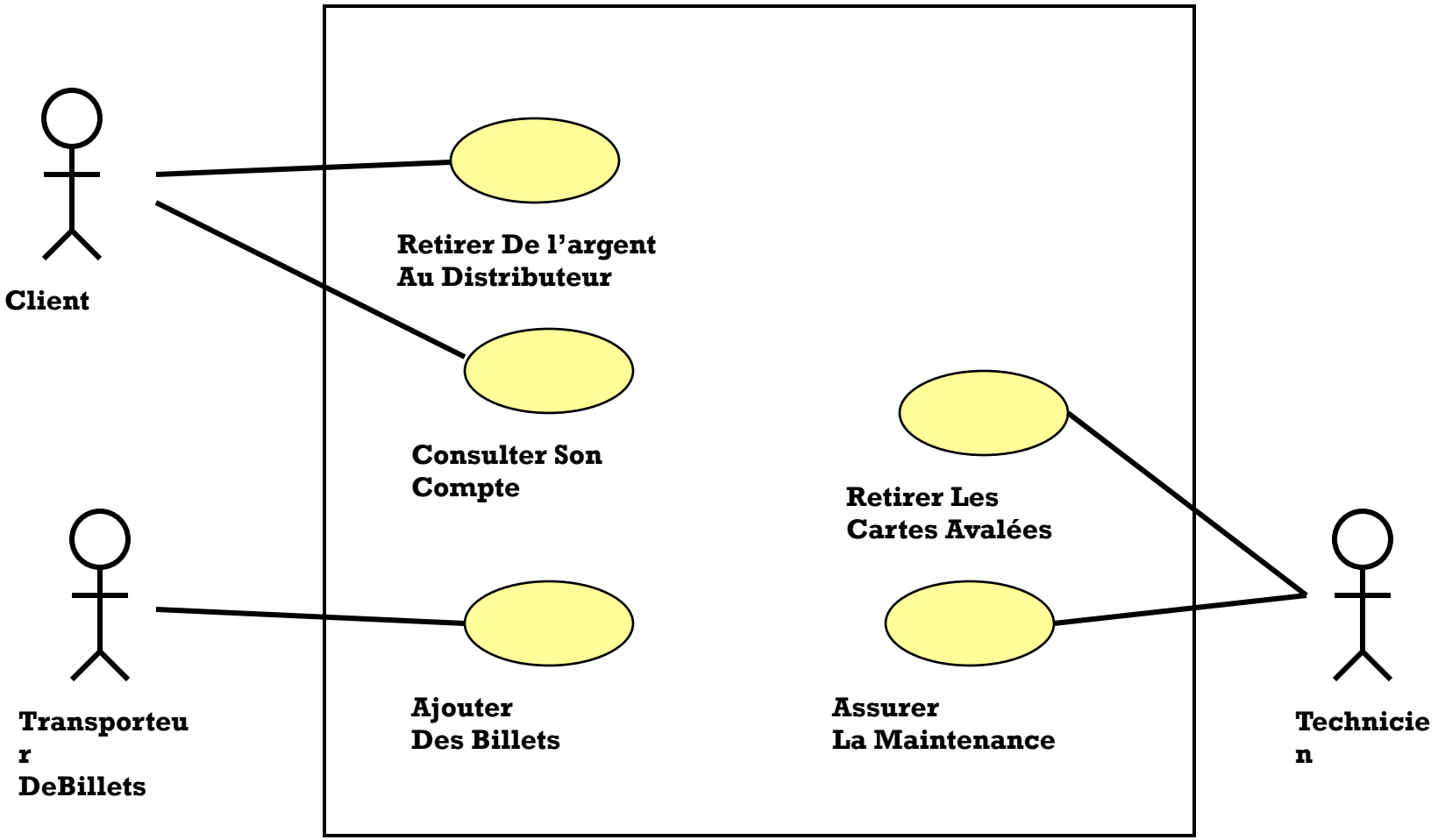
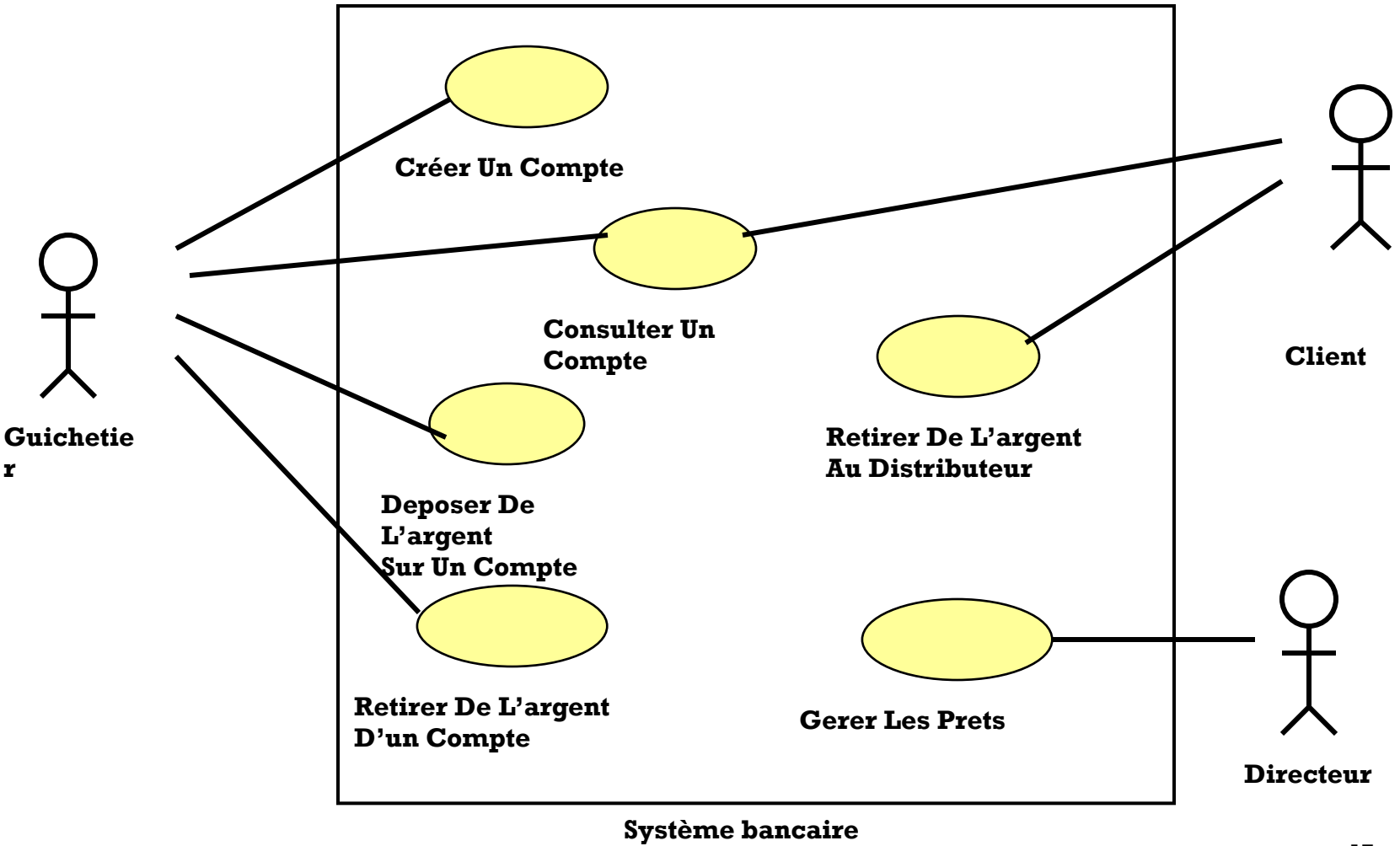


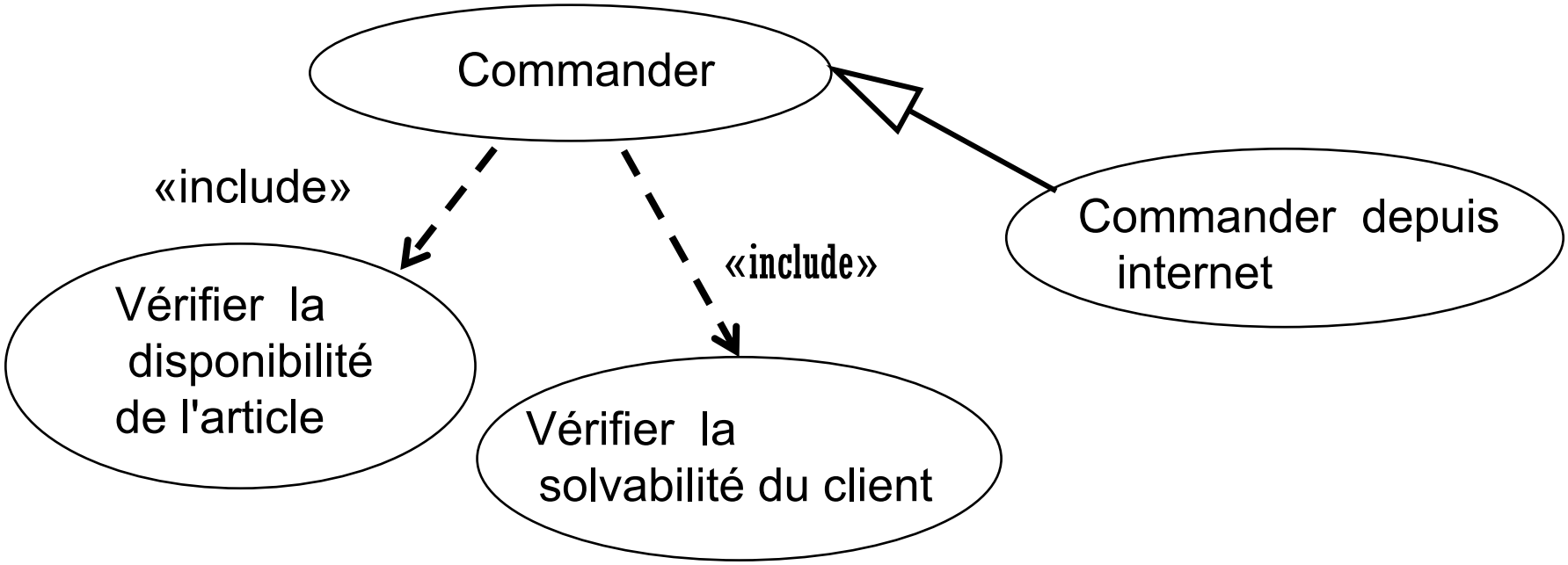
DIAGRAMME DE CAS D 'UTILISATION



RELATIONS ENTRE LES CU

- *«include»*
  - Un cas d'utilisation inclut un autre cas d'utilisation
- *«extend»*
  - Un cas d'utilisation peut déclarer des points d'extension (e.g. prolongements logiques)
  - On dit qu'un cas d'utilisation **A** étend un cas d'utilisation **B** lorsque le cas d'utilisation **A** peut être appelé au cours de l'exécution du cas d'utilisation **B**.
- *Généralisation*
  - Une relation de généralisation d'un cas d'utilisation **B** vers un cas d'utilisation **A** signifie que **B** est une spécialisation de **A**

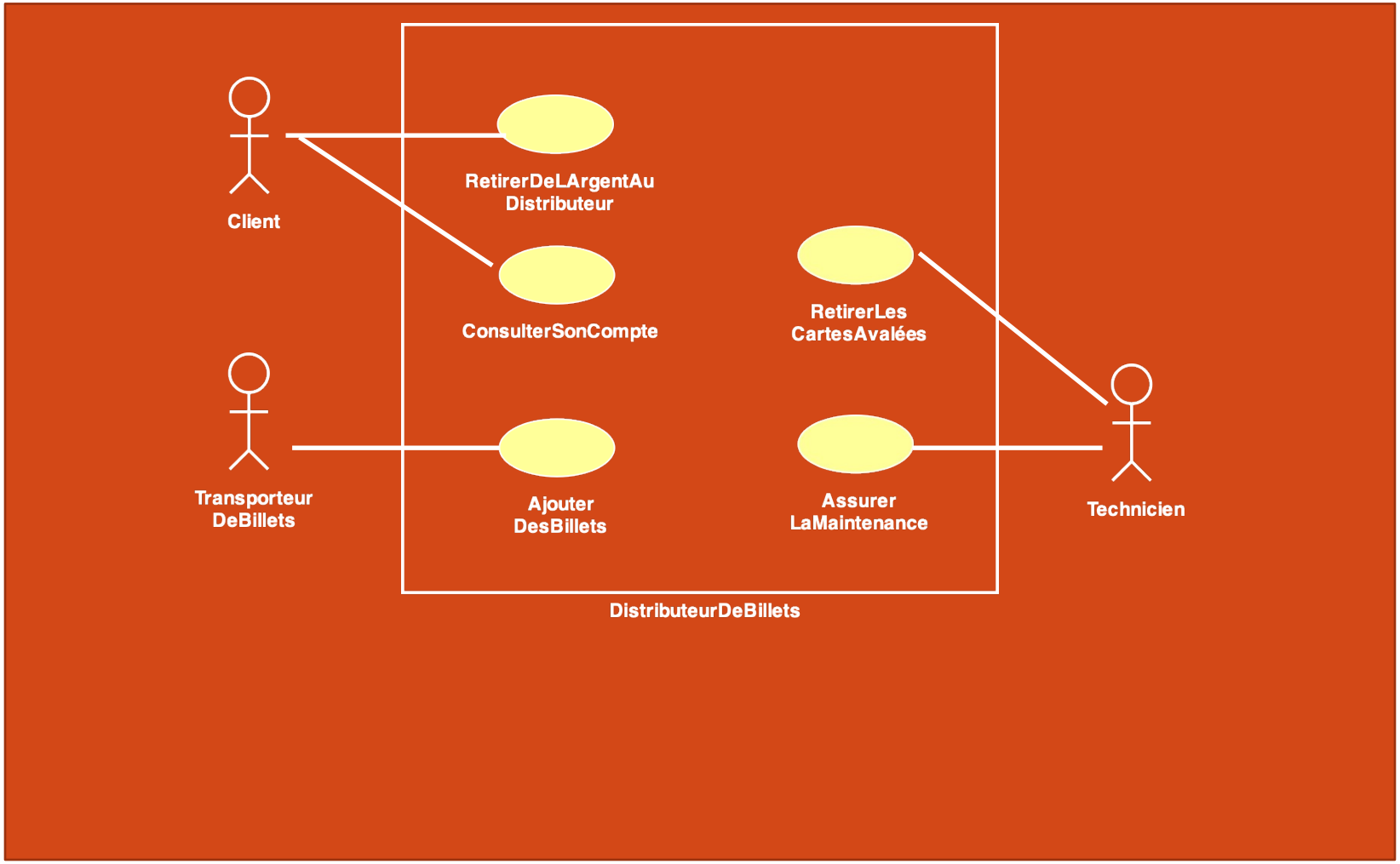
RELATIONS ENTRE LES CU: NOTATION



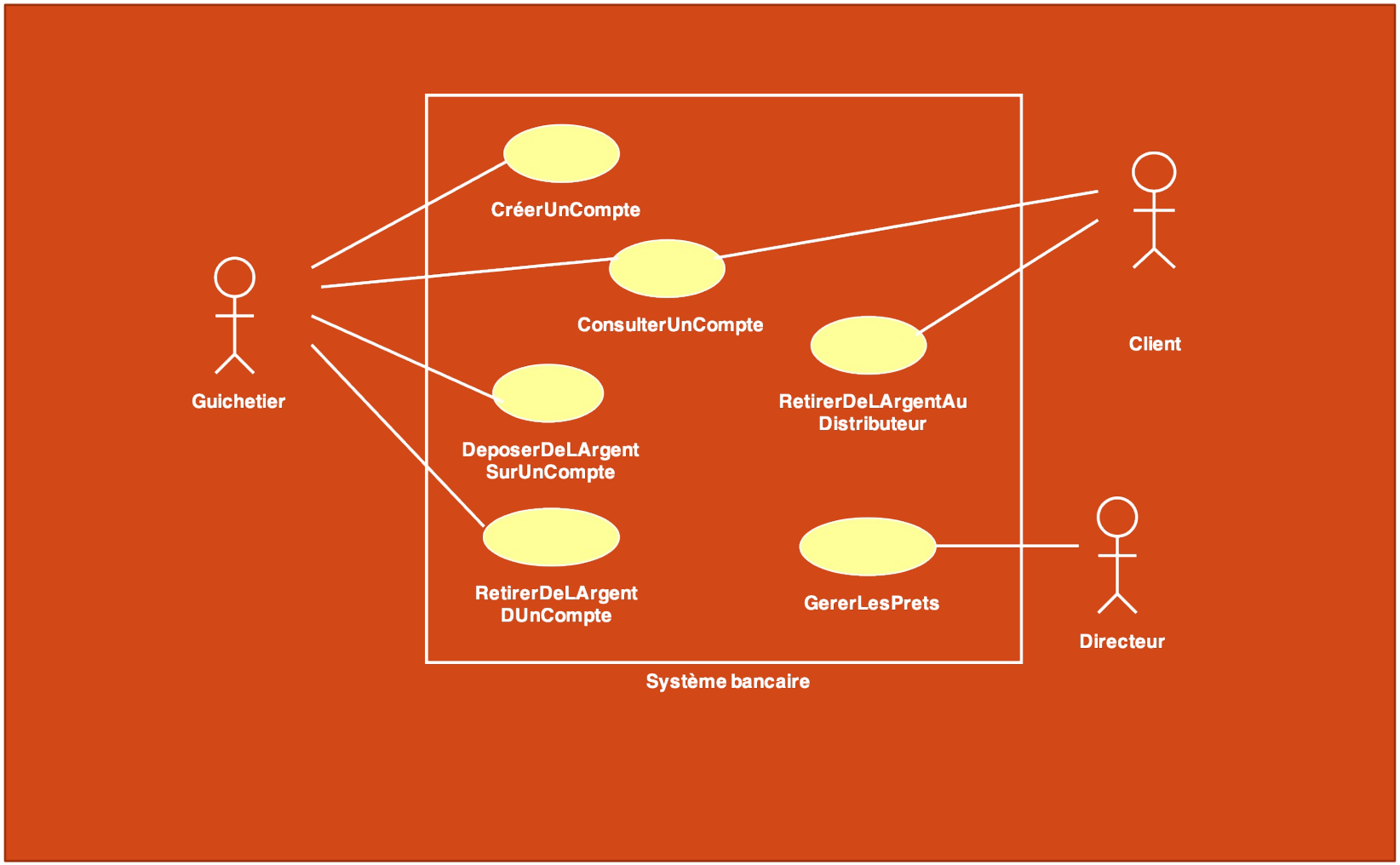
EXEMPLE DE CAS D'UTILISATION

EXEMPLE DE CAS D'UTILISATION

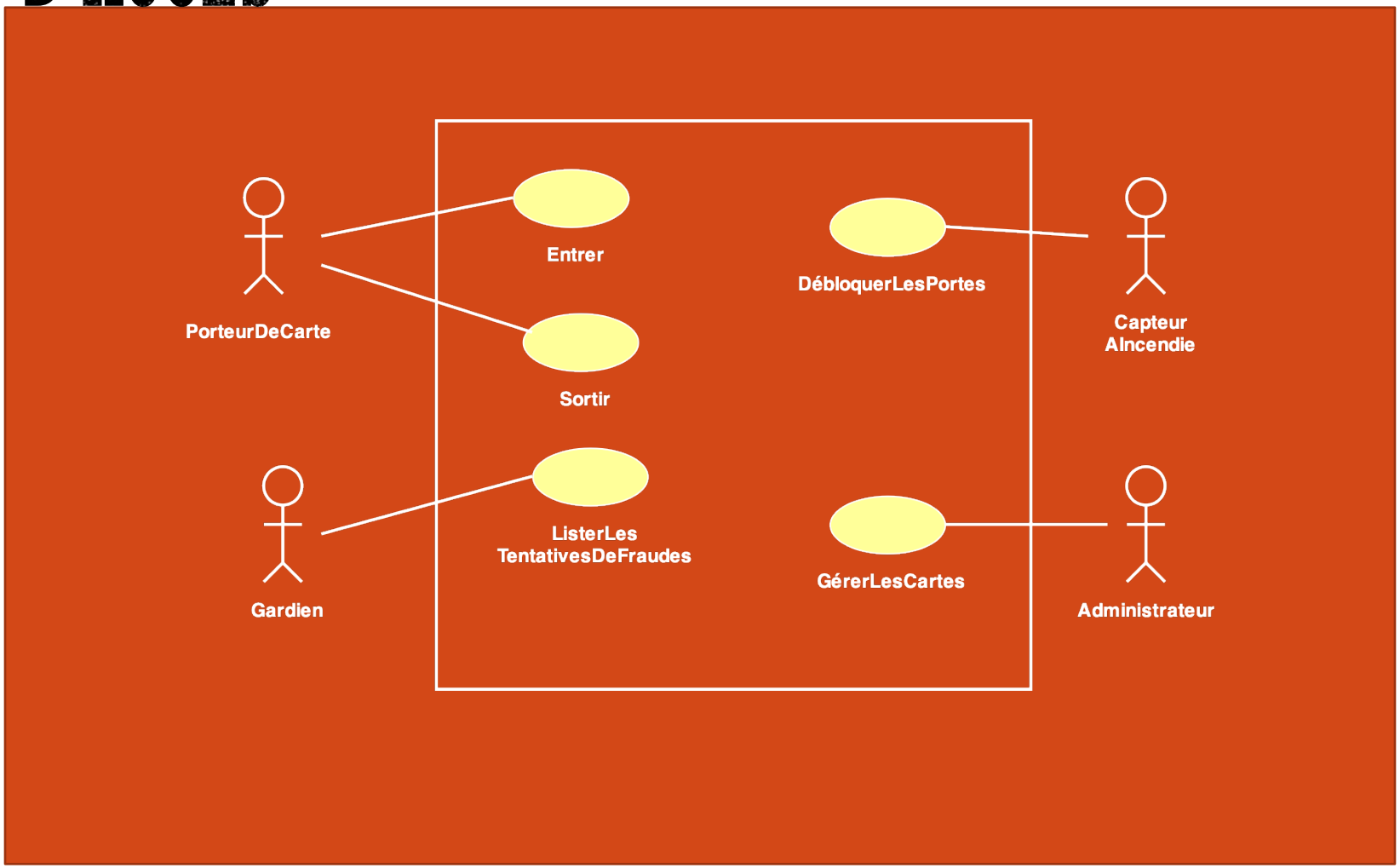
EXEMPLE DE CAS D'UTILISATION

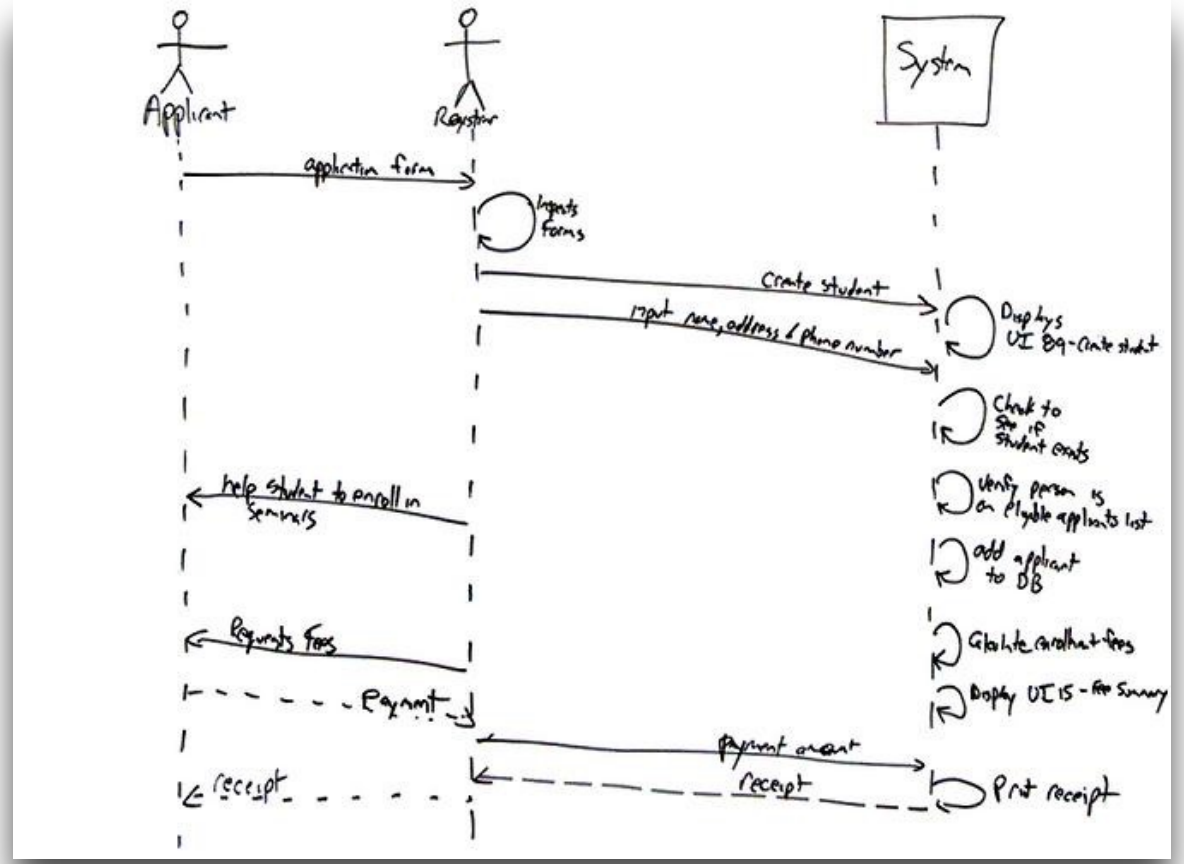


EXEMPLE: SYSTÈME BANCAIRE



EXEMPLE: SYSTÈME DE CONTRÔLE D'ACCÈS





LE DIAGRAMME DE SÉQUENCES

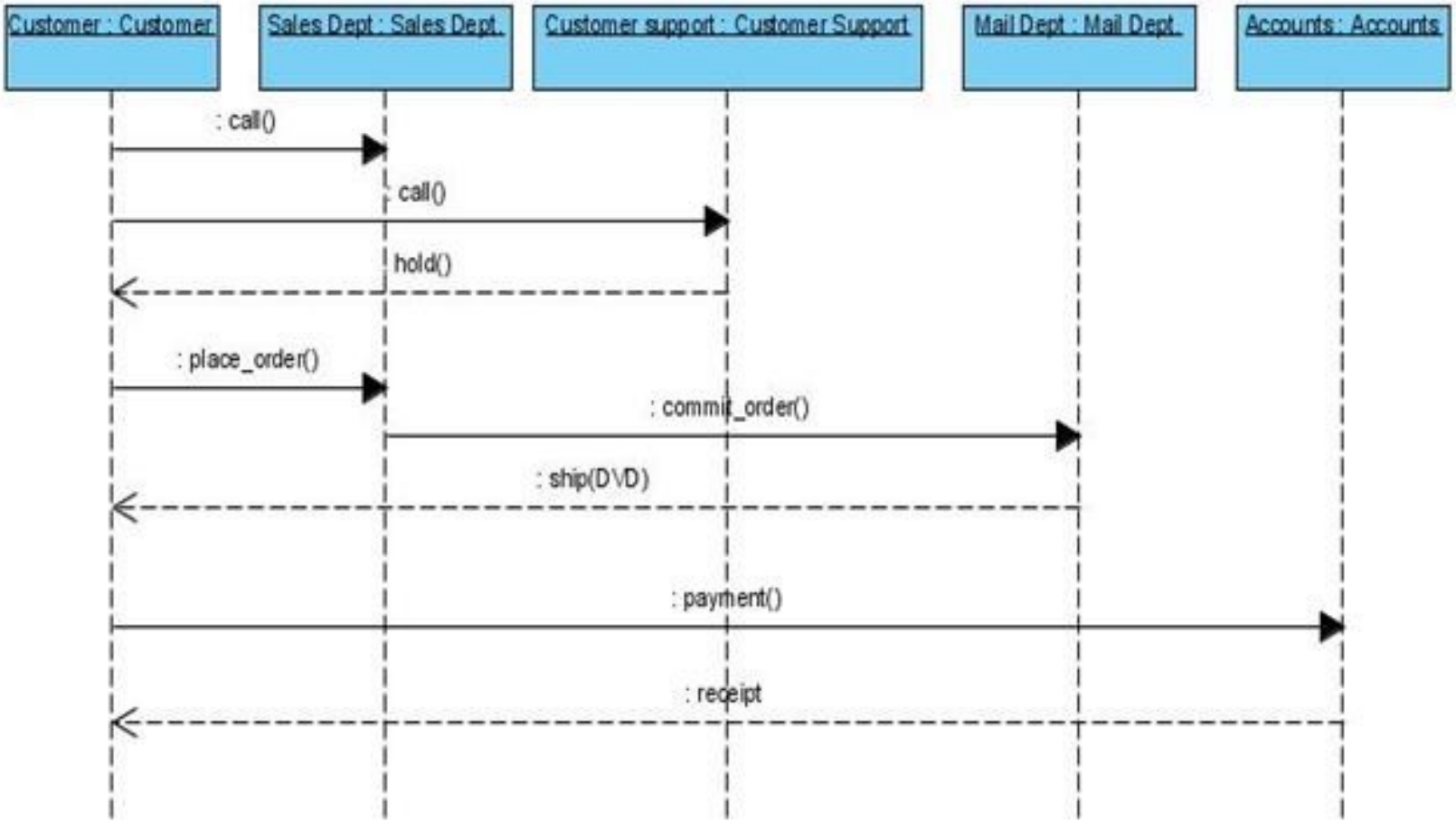
DIAGRAMME DE SÉQUENCES

- Fait partie de la catégorie des diagrammes dynamiques (comportementaux)
- Permet de représenter les interactions entre participants sous forme de messages.

DEUX TYPES (GRANULARITÉS) DE DS

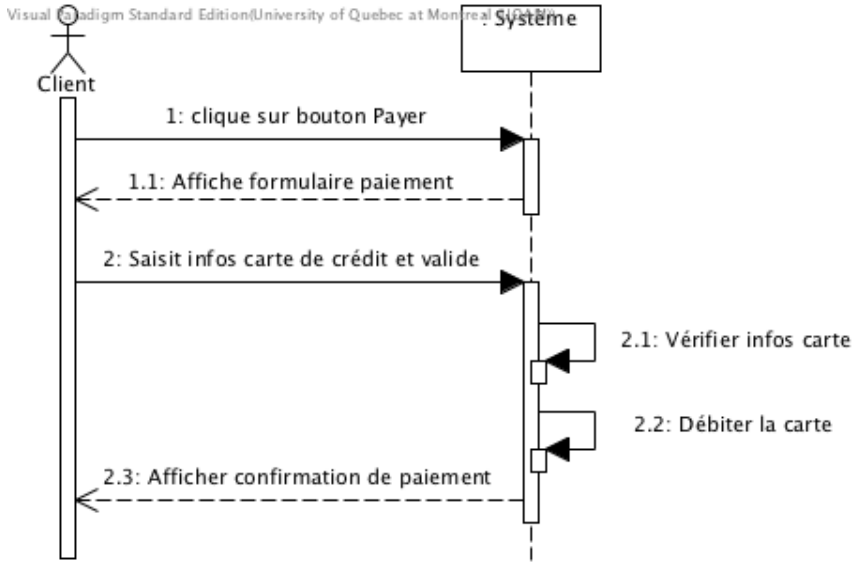
- Diagramme de séquences système: lors de la phase d'analyse
- Diagramme de séquences de conception:
  - Les participants sont des objets
  - Les messages sont des appels de méthodes munis de leurs paramètres
  - Les réponses sont les données de retour

DS DE CONCEPTION (EXEMPLE)



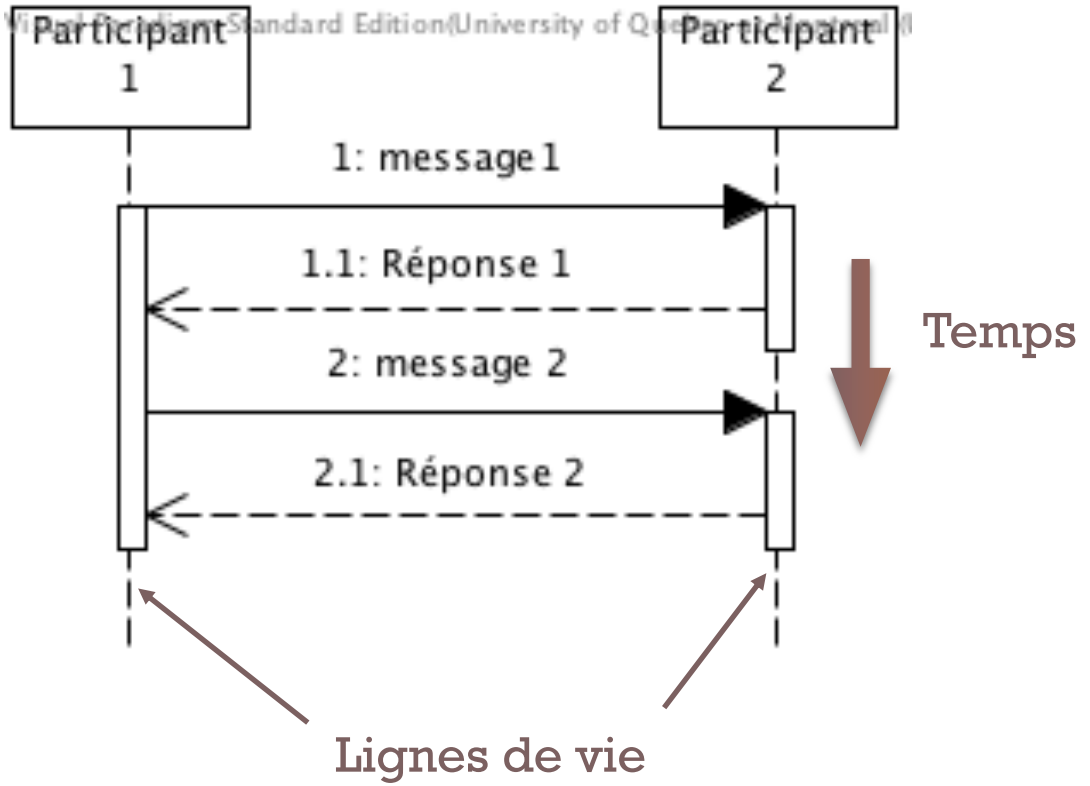
DIAGRAMMES DE SÉQUENCE SYSTÈME

- Utilisés lors de la phase d'analyse pour modéliser graphiquement les scénarios;
- Même niveau de détail que la description textuelle du scénario;
- Modélise les interactions entre les acteurs et le système;
- Le système est une boite noire.



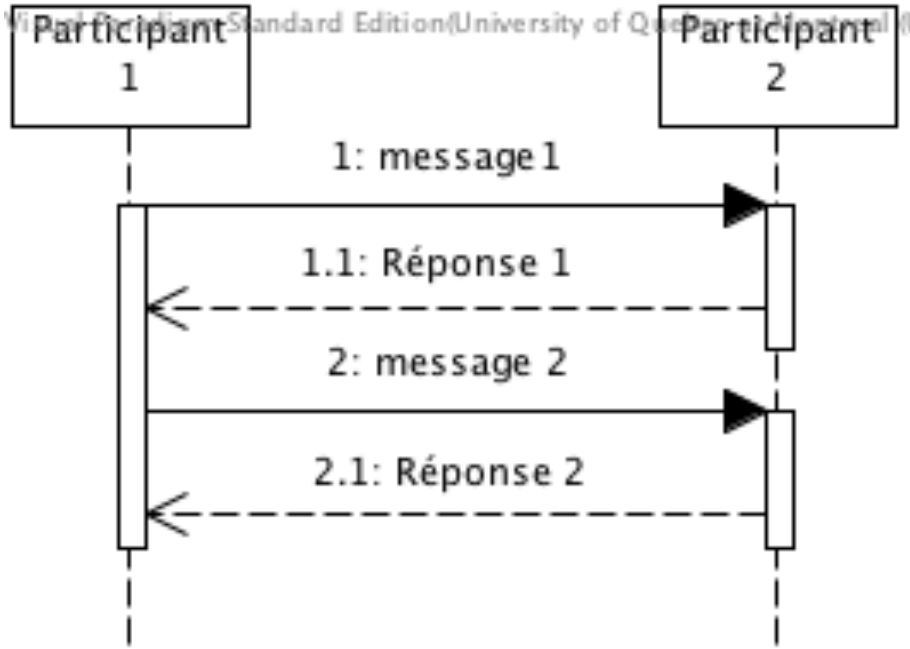
SYNTAXE: LIGNES DE VIE

- Chaque participant a une ligne de vie.
- Les messages transitent horizontalement.
- Le temps s'écoule verticalement, de haut en bas.



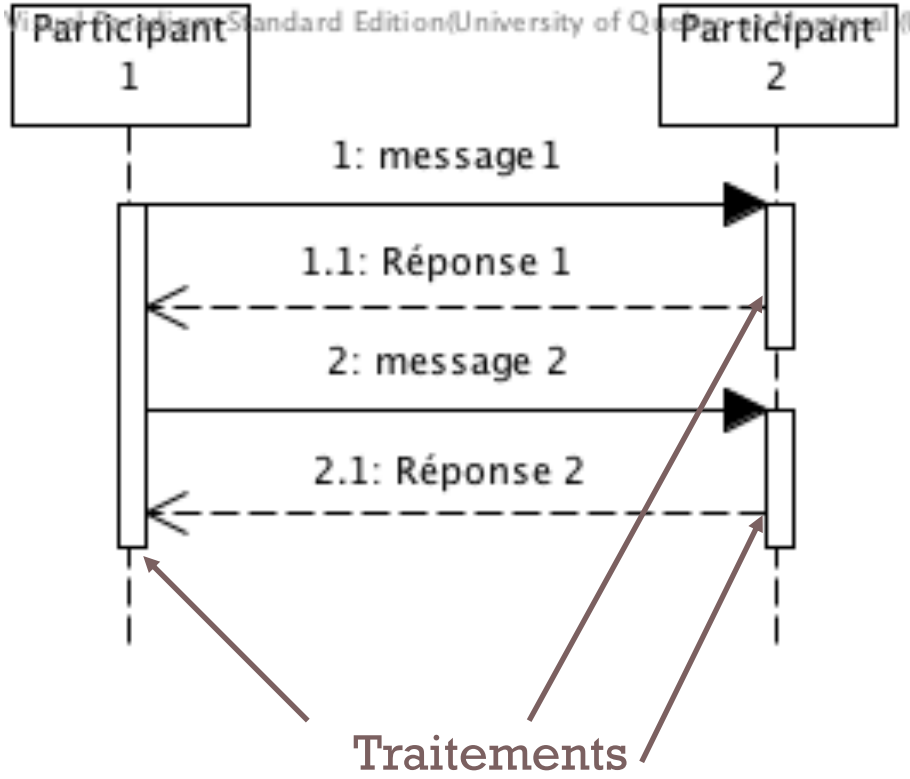
SYNTAXE: MESSAGES

- Une flèche pleine symbolise un message synchrone.
- Un message est une requête envoyée par un participant à un autre.
- La réponse à un message est symbolisée par une flèche pointillée.
- Les messages et leurs réponses sont numérotées selon un ordre croissant.



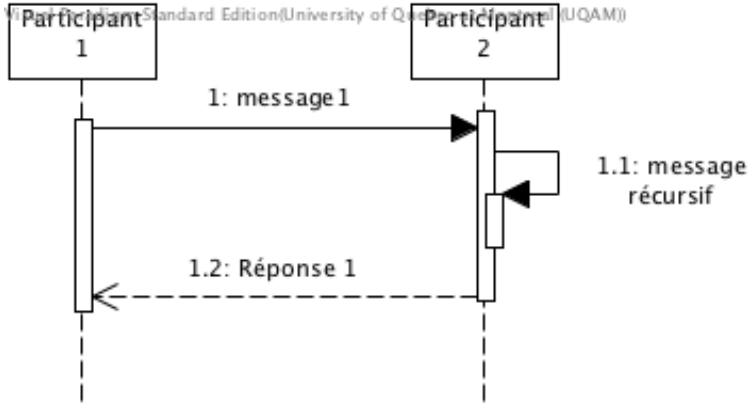
SYNTAXE: TRAITEMENTS

- Un traitement est symbolisé par un rectangle positionné sur la ligne de vie.
- La hauteur du rectangle symbolise le temps (relatif) du traitement.

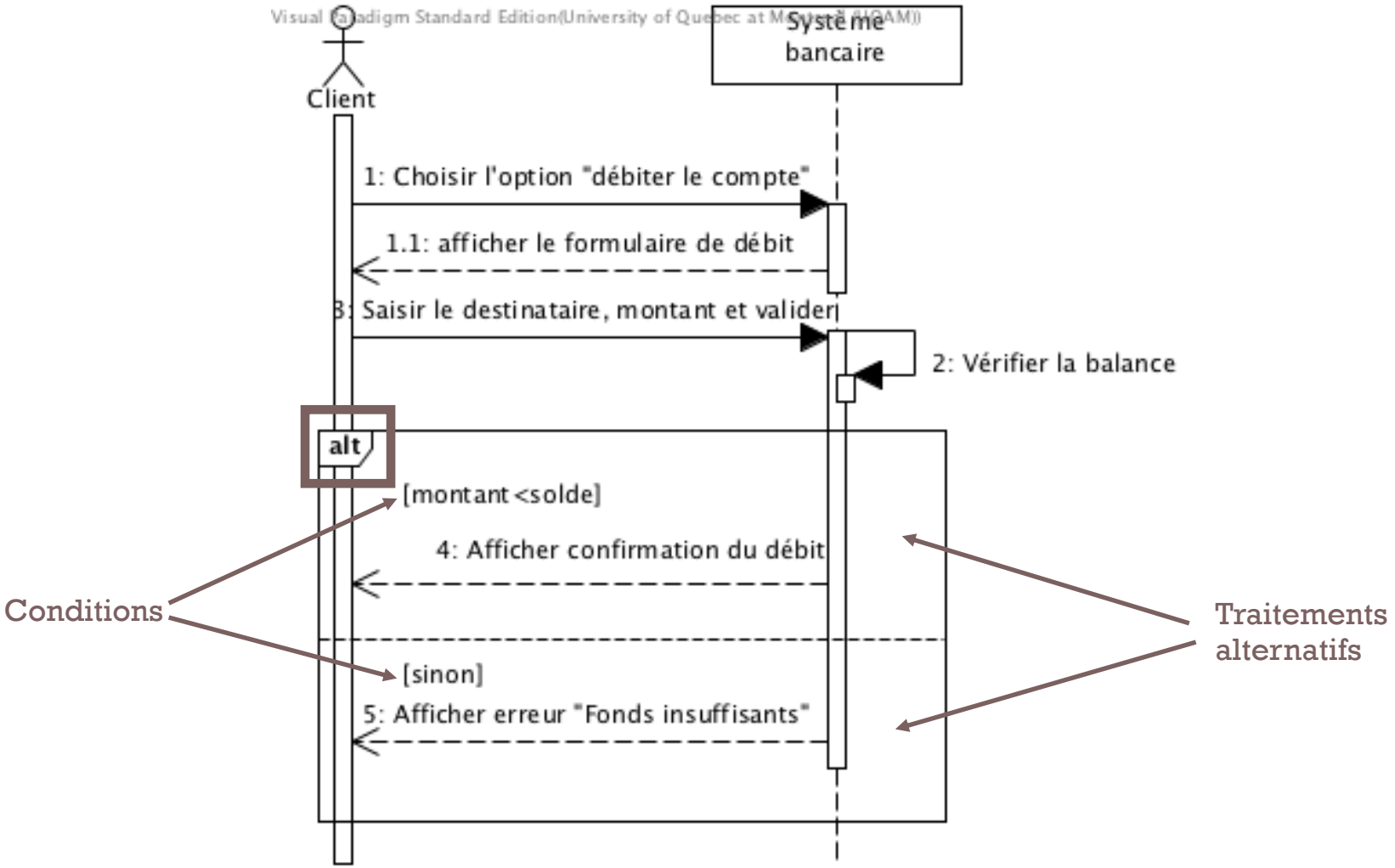


SYNTAXE: MESSAGES RÉCURSIFS

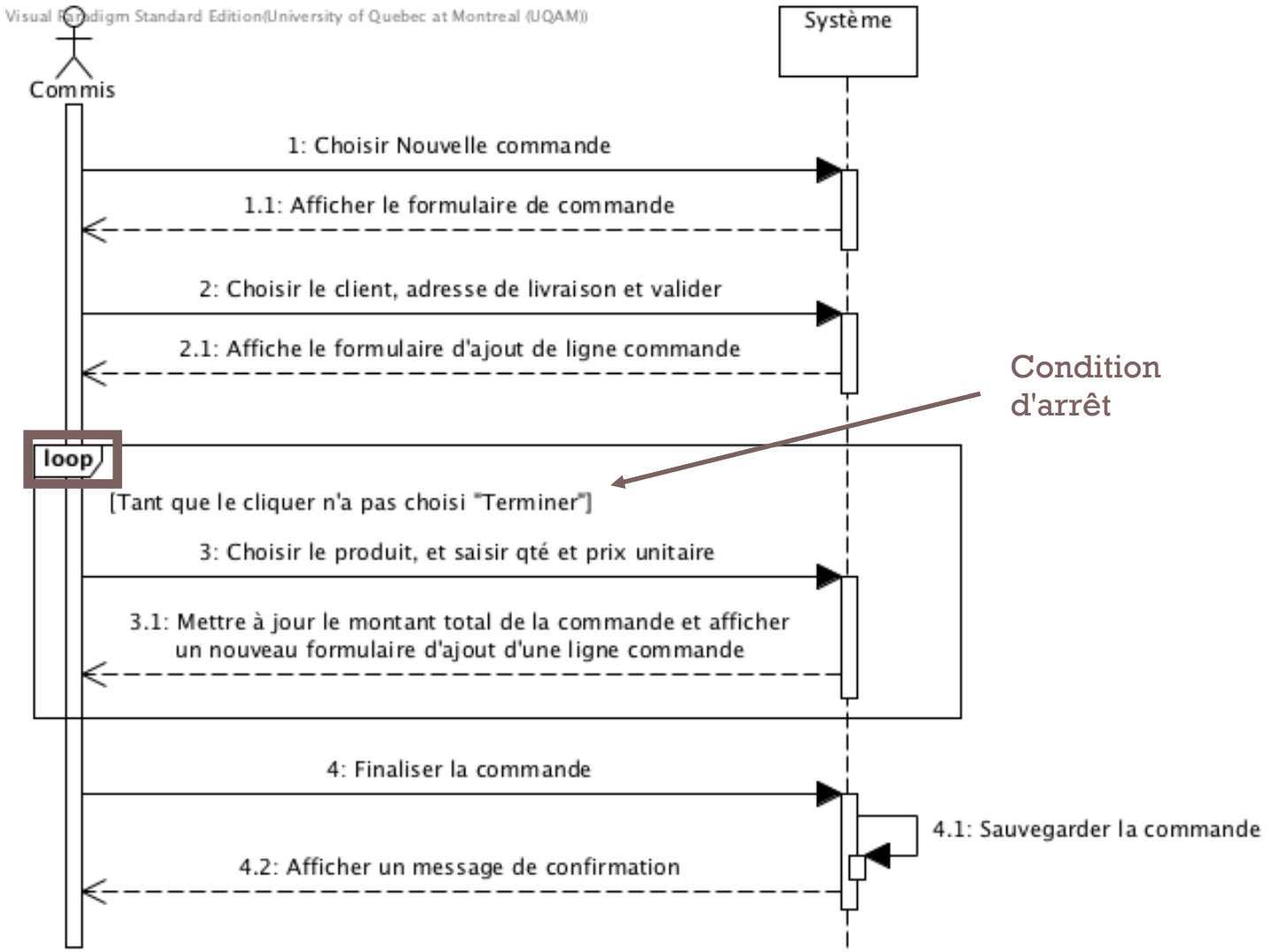
- Un participant peut envoyer un message à lui même durant un traitement: un message récursif.
- Donne lieu à un sous-traitement.



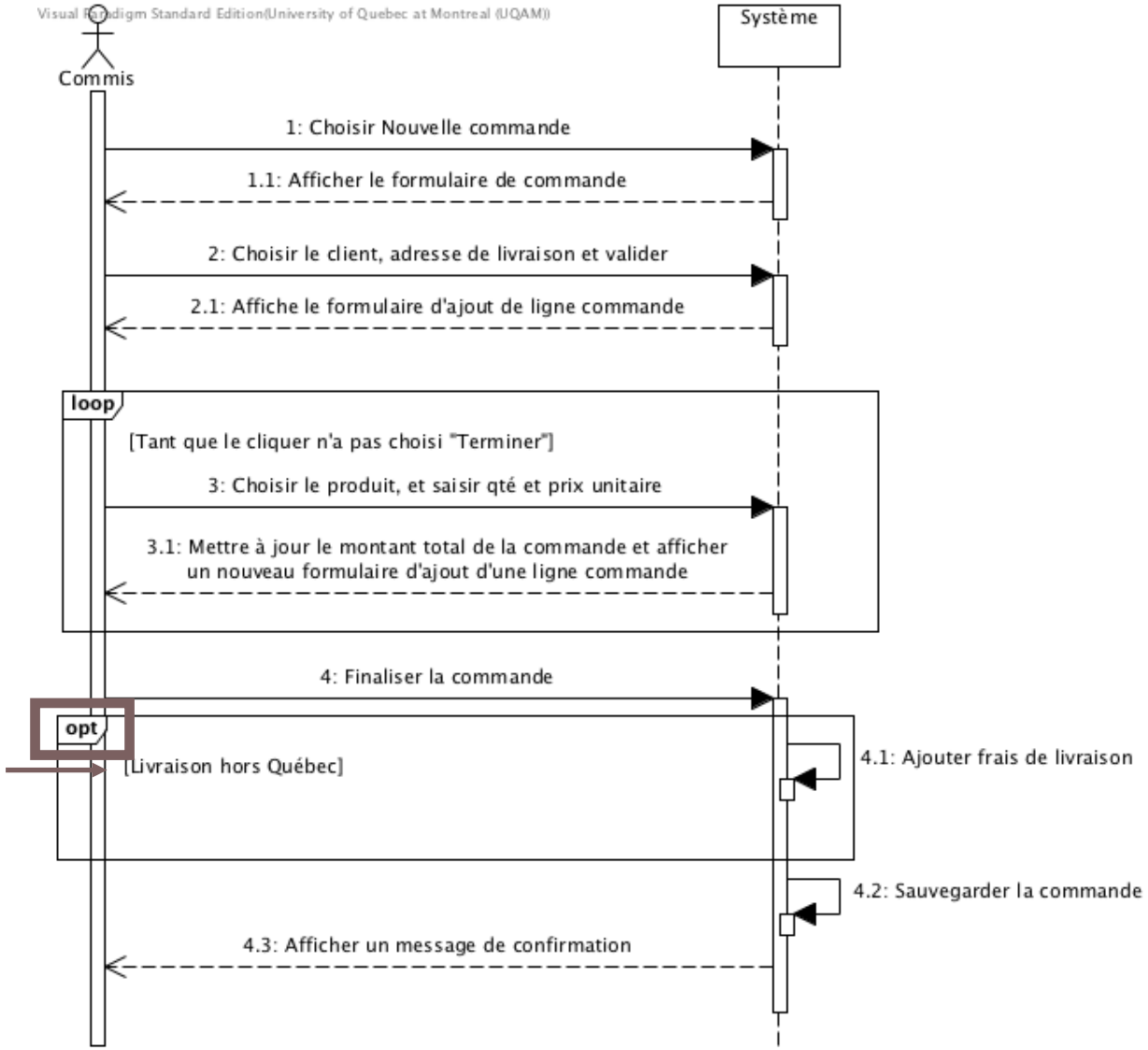
SYNTAXE: CHEMINS ALTERNATIFS



SYNTAXE: BOUCLES

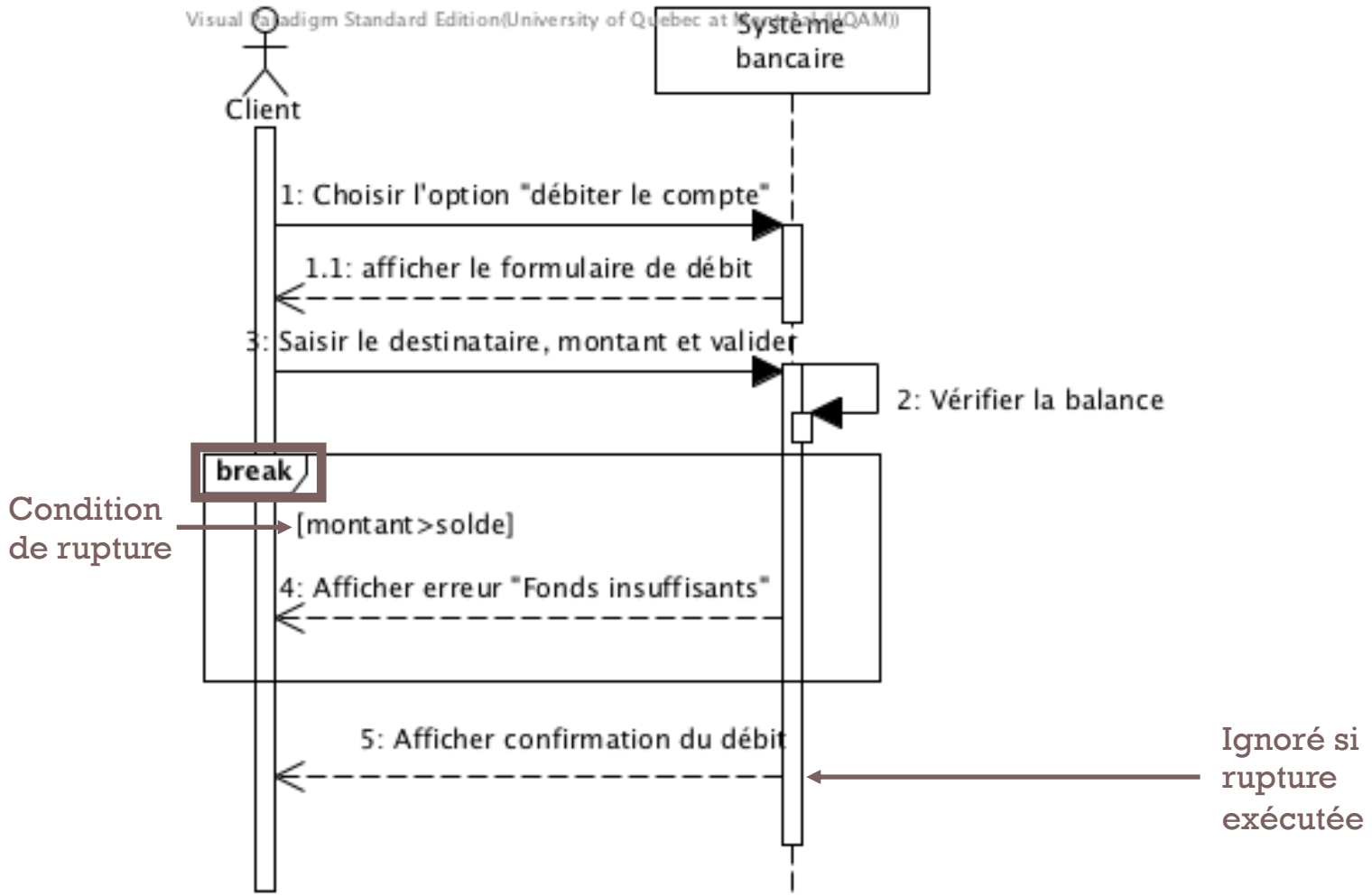


SYNTAXE: CHEMINS OPTIONNEL

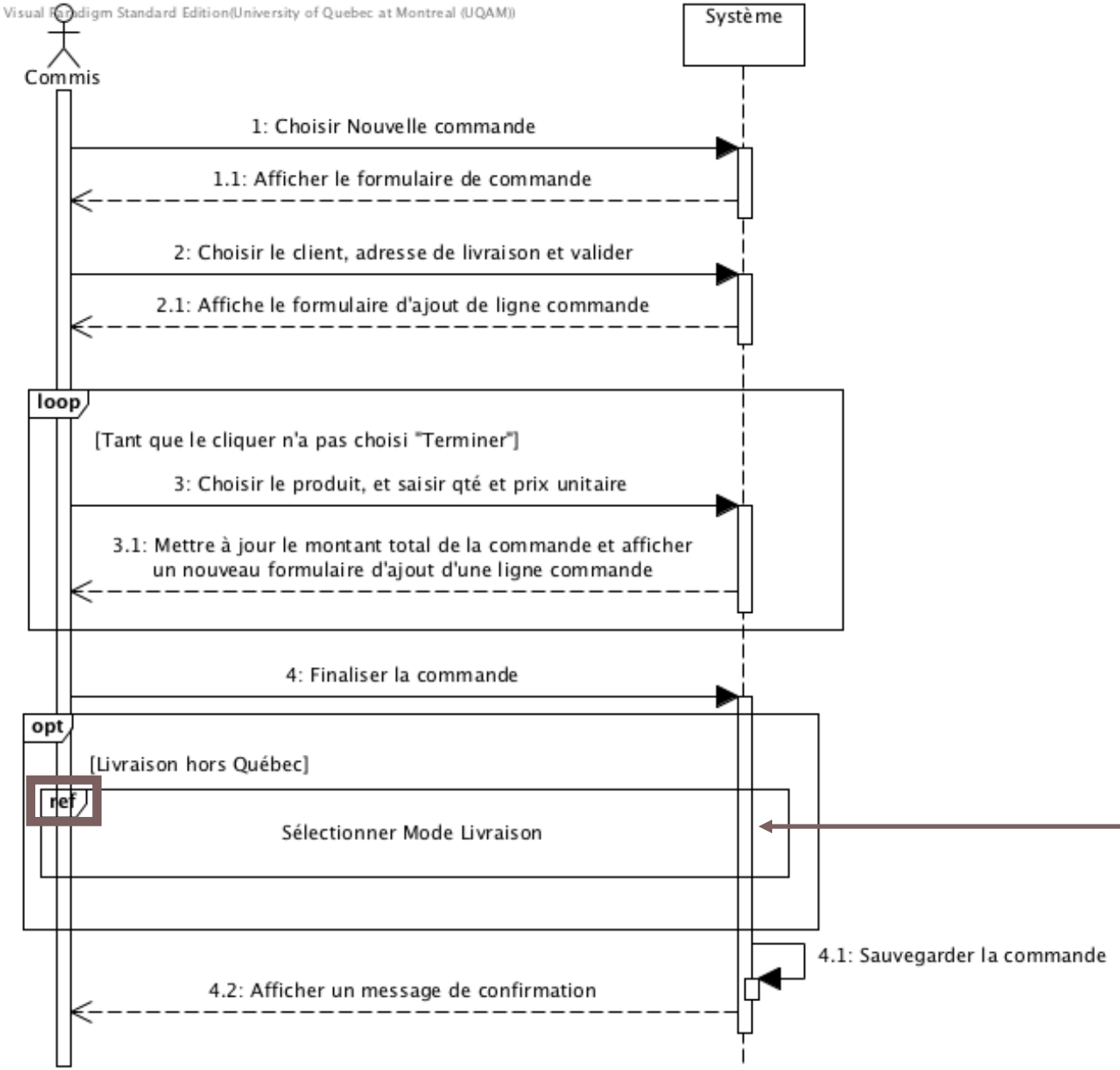


Condition d'accès

SYNTAXE: CHEMINS DE RUPTURE (BREAK)



RÉFÉRENCER UN AUTRE DIAGRAMME



DS Référencé

SYNTAXE: TRAITEMENTS PARALLÈLES

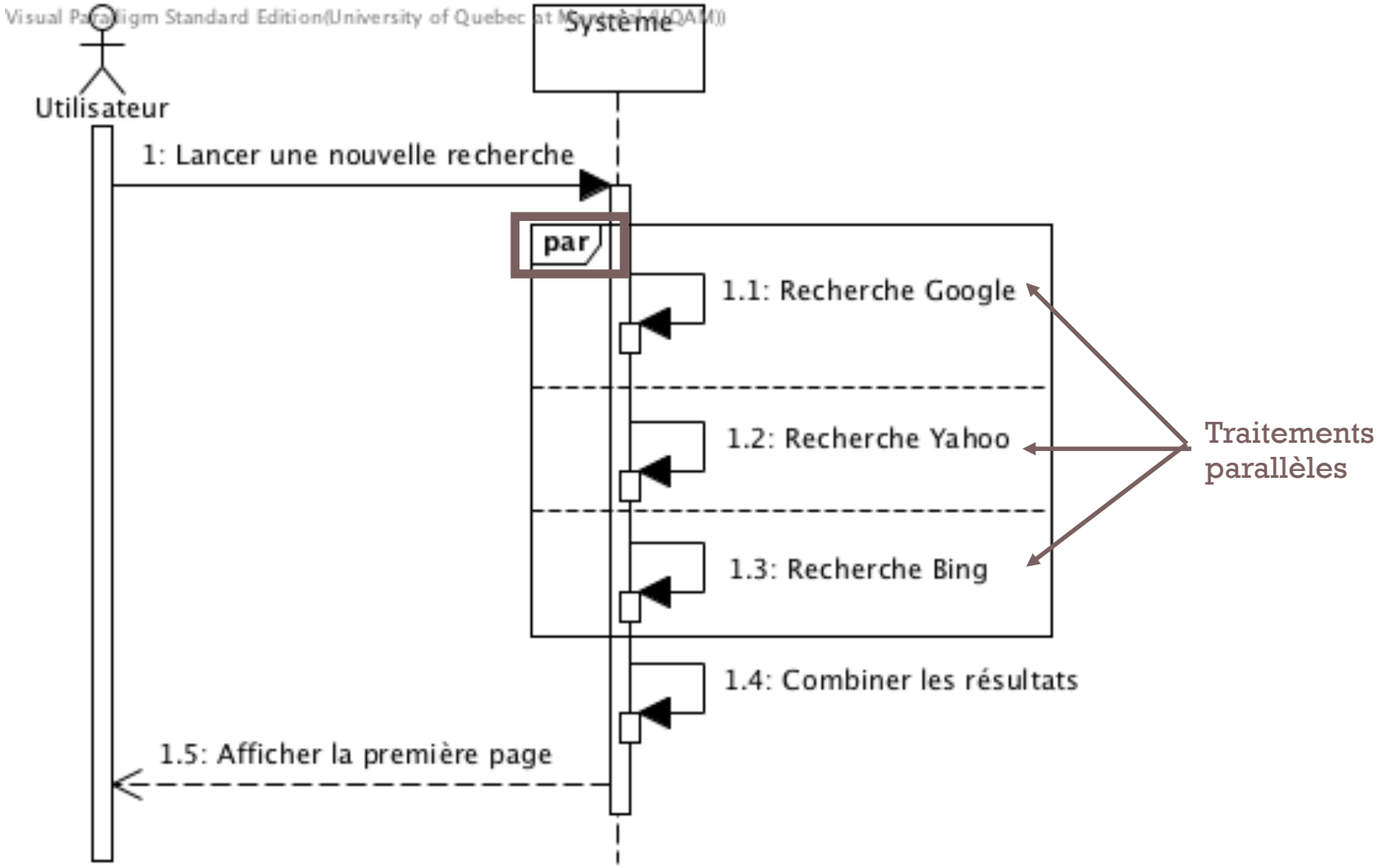
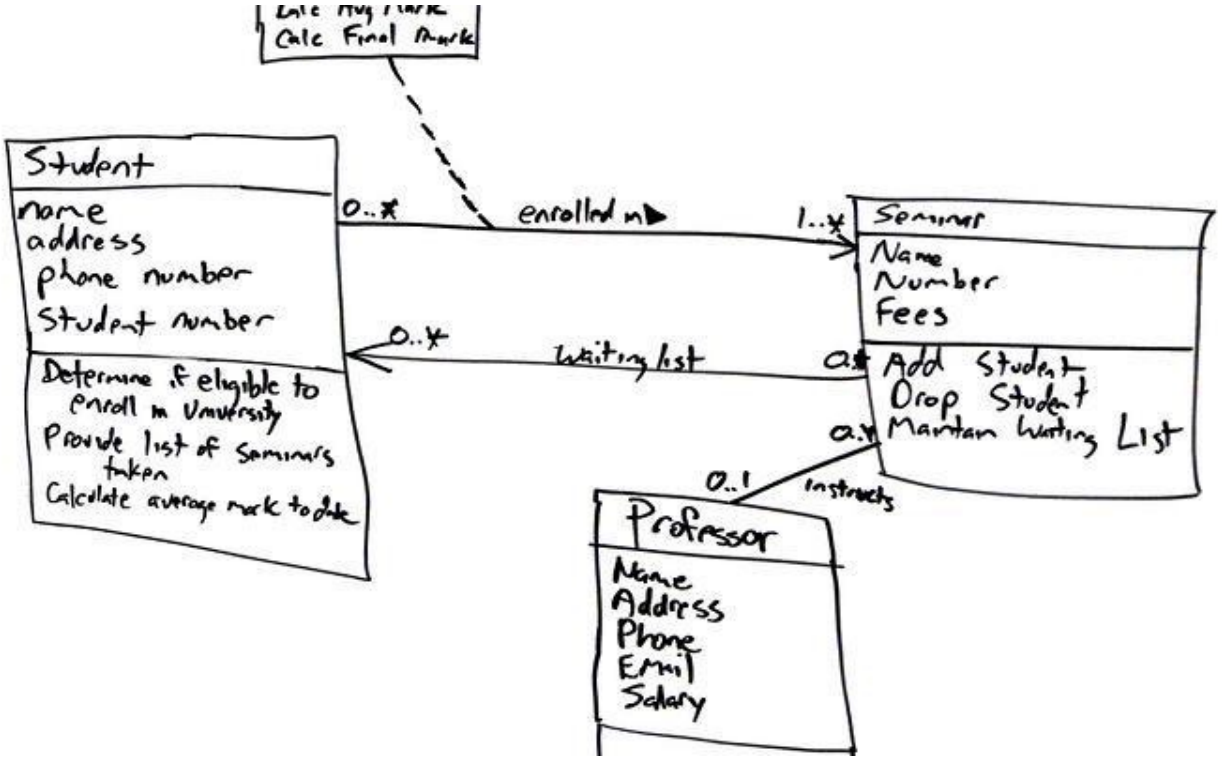
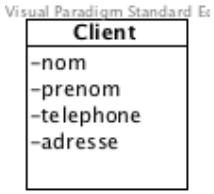


DIAGRAMME DE CLASSES

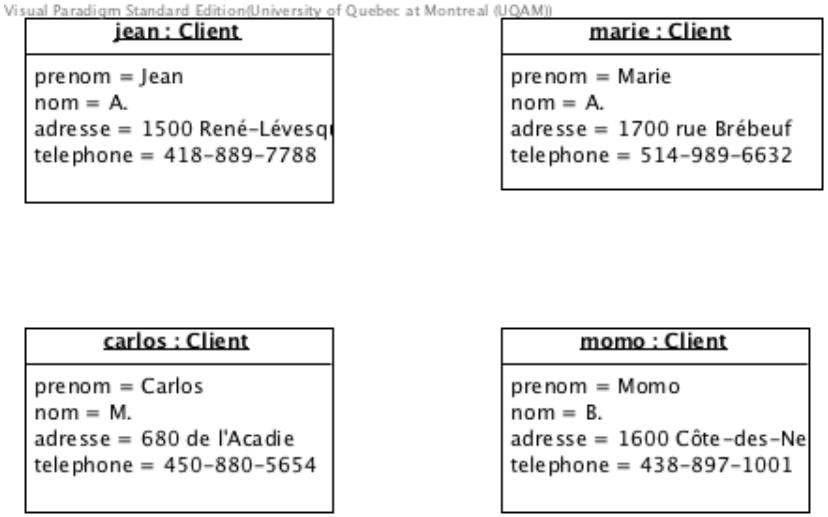


QU'EST-CE QU'UNE CLASSE?



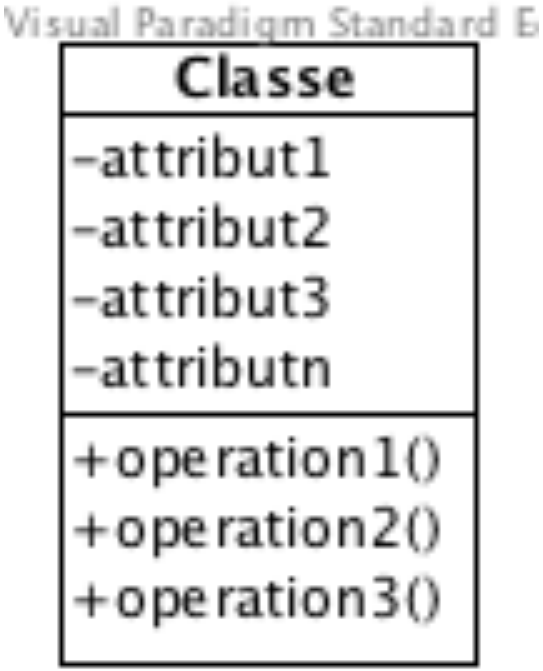
Classes (M0)

Objets (M1)



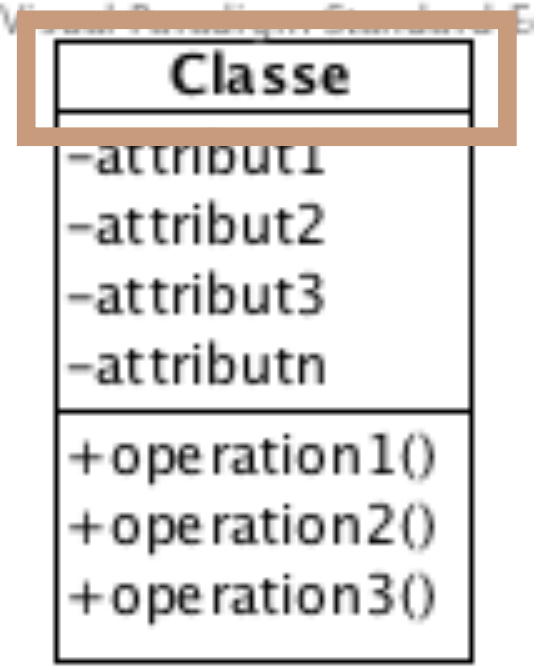
SYNTAXE D'UNE CLASSE

- Représentée par un rectangle divisé en compartiments, typiquement:
  - Le nom de la classe
  - La liste de ses attributs
  - La liste de ses opérations



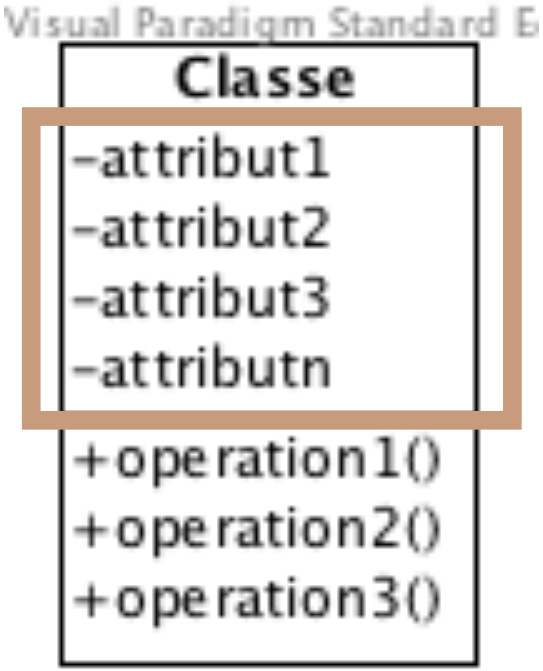
SYNTAXE: NOM DE LA CLASSE

- Nom de la classe
  - Identifie, de façon unique, la classes parmi les autres classes
  - Doit obligatoirement commencer par une majuscule
  - Si le nom comporte plusieurs morts, chaque mot commence par une majuscule (CamelCase). Ex.: ClientPrivilegie



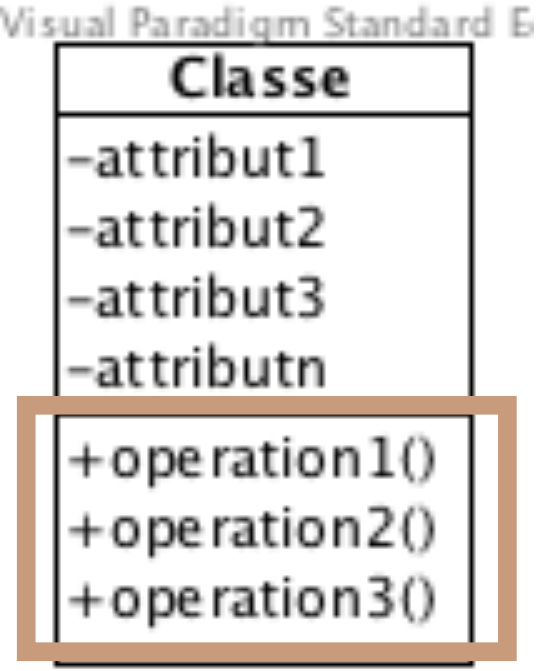
SYNTAXE: ATTRIBUTS

- Les attributs:
  - Spécifie la structure de données des instances de la classe: quelles informations caractérisent des objets de la classe?
  - Chaque attribut est unique au sein d'une classe.
  - Un attribut ne peut pas avoir le nom d'un rôle d'une association (voir associations)
  - Le nom commence toujours par une minuscule et utilise CamelCase. ex.: telDomicile



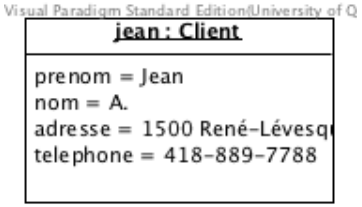
SYNTAXE: OPÉRATIONS

- Les opérations: spécifient actions que nous pouvons effectuer sur les instances de la classe.
- Par exemple, pour la classe Client:
  - modifierNumTelDomicile(nouveauNumero)
  - envoyerCourriel(contenu)
  - definirCommeVIP()
- Chaque operation est unique au sein d'une classe.
- Le nom commence toujours par une minuscule et utilise CamelCase.
  ex.: telDomicile



SYNTAXE: OBJETS

- La norme UML définit aussi les diagramme d'objets.
- Un diagramme d'objets représente un exemple d'instantiation d'un diagramme de classes.
- Syntaxe:
  - Le nom de l'instance suivi du nom de la classe en souligné.
  - La valeur des attributs peut être précisée
  - Les opérations n'apparaissent pas.



CONCEPTS (OU ENTITÉS)

- Lors de la phase d'analyse, nous parlons plutôt de concepts ou d'entités.
- La différence est purement sémantique:
  - un concept décrit les informations du domaine d'affaire
  - une classe décrit les informations manipulées par le logiciel.
- On ne mentionne que les opérations métier principales d'un concept: pas d'accepteurs (getters/setters), ni d'opérations utilitaires.
- Terminologie:
  un diagramme de classes de niveau analyse
  = un diagramme de concepts
  = modèle du domaine

CONCEPTS (OU ENTITÉS)

- Représente typiquement: des personnes (rôles), des objets physiques et informations importantes du domaine d'application;
- Énumérer les concepts pour s'assurer que les fonctionnalités décrites seront supportées par une structure informationnelle adéquate.

DIAGRAMES DE CLASSES VS DIAGRAMME DE CONCEPTS

- Définit l'ensemble des classes *logicielles* qui seront implémentées (et instanciées) par le logiciel ainsi que les relations entre les classes.
- Un diagramme de concepts définit l'ensemble des données du domaine d'affaires, leurs structures et leurs relations.

IDENTIFICATION DES CONCEPTS

- Rechercher les formes nominales dans la description du problème par les parties-prenantes ;
- Typiquement, on commence par plus de concepts que nécessaire puis on raffine la sélection.

CONCEPTS POSSIBLES

- Choses physiques (ex.: produit, salle, équipement) ou intangibles (ex.: cours, facture, rapport, signal) faisant partie du domaine;
- Événements qui se produisent dans le contexte du système et qui doivent être gérés. Ex.: transfert bancaire, livraison, etc.
- Rôles: affectés à des personnes qui interagissent avec le système (ex.: Client, Employé, etc.)
- Unités organisationnelles pertinentes au système. Ex.: groupe, équipe, etc.

CONCEPTS POSSIBLES

- Entités externes: qui interagissent avec le système. Ex.: périphérique, système bancaire.
- Endroits: qui déterminent le contexte des traitements. Ex.: Entrepôt, salle de classe, etc.
- Structures (ou compositions) qui définissent un assemblage d'objets. Ex.: ordinateur, voiture, etc.

SELECTION DES CONCEPTS

- Écarter les concepts qui sont:
  - En dehors du périmètre du logiciel;
  - Réfèrent au système complet;
  - Dupliquent d'autres concepts;
  - Sont trop vagues ou trop spécifiques (trop peu ou beaucoup trop d'instances possibles)

SELECTION DES CONCEPTS

- Persistance: est-ce qu'il est nécessaire que le système retienne les informations portant sur les objets d'une classe?
- Est-ce que les instances d'un concept ont des attributs communs? des opérations communes? Ou les deux?
- Ignorer les entités externes, à moins qu'elles produisent ou consomment des informations essentielles au système.

RELATIONS ENTRE CLASSES

- Une fois les principaux concepts identifiés, nous cherchons à déterminer les liens entre ces concepts.
- Permet :
  - d'établir la structure globale de données;
  - de raffiner les concepts identifiés : ajout ou suppression, regroupement par types, etc.

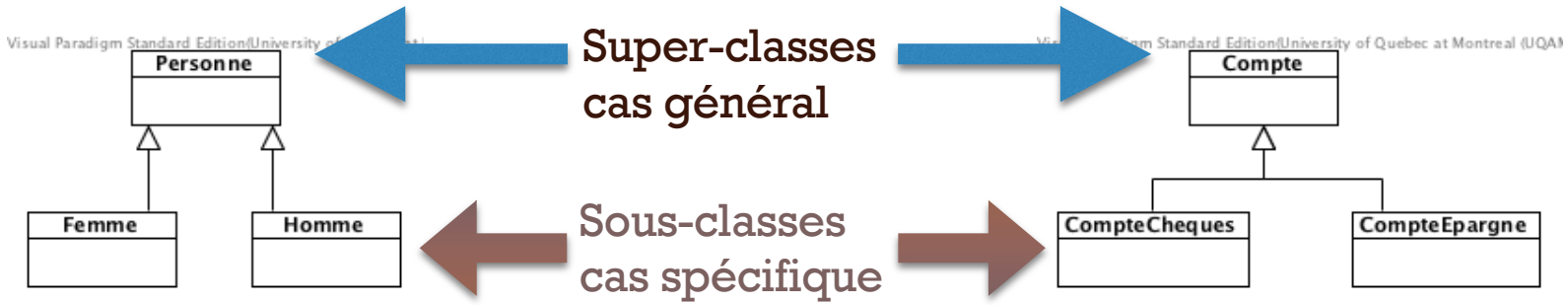
RELATION ENTRE CLASSES

- Deux types de relations:
  - La relation d'héritage (généralisation / spécialisation)
  - L'association entre classes

L'HÉRITAGE

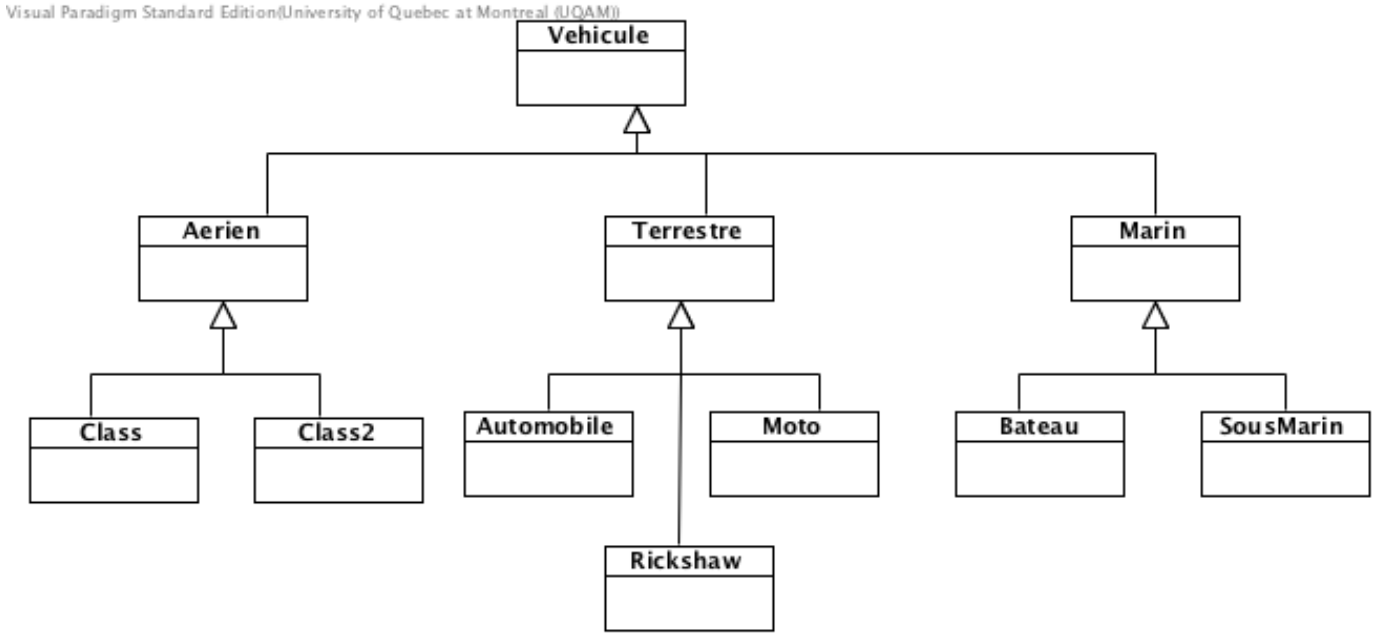
L'HÉRITAGE

- Une relation entre une classe générale et sa décomposition en classes plus spécifiques.



L'HÉRITAGE

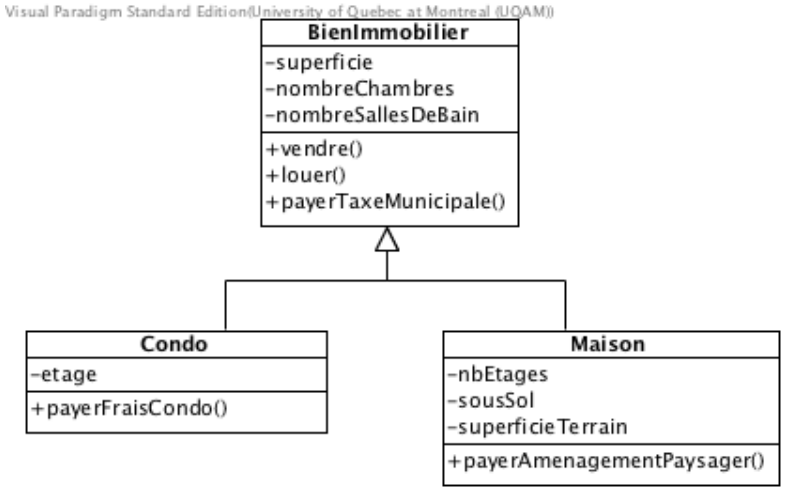
- Permet de définir une structure hiérarchique de spécialisations.



- Une instance de la classe Automobile est aussi une instance de Terrestre et de Vehicule.

L'HÉRITAGE

- Une clase spécialisée “hérite” de la structure (attributs) et du comportement (opérations) de sa super-classe.

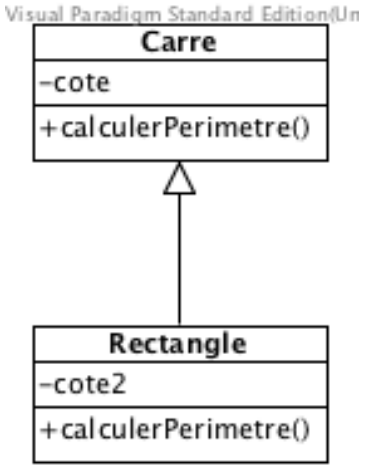


- La classe spécialisée définit les informations (attributs) et les opérations qui lui sont spécifiques.

HÉRITAGE ET RE-DÉFINITION

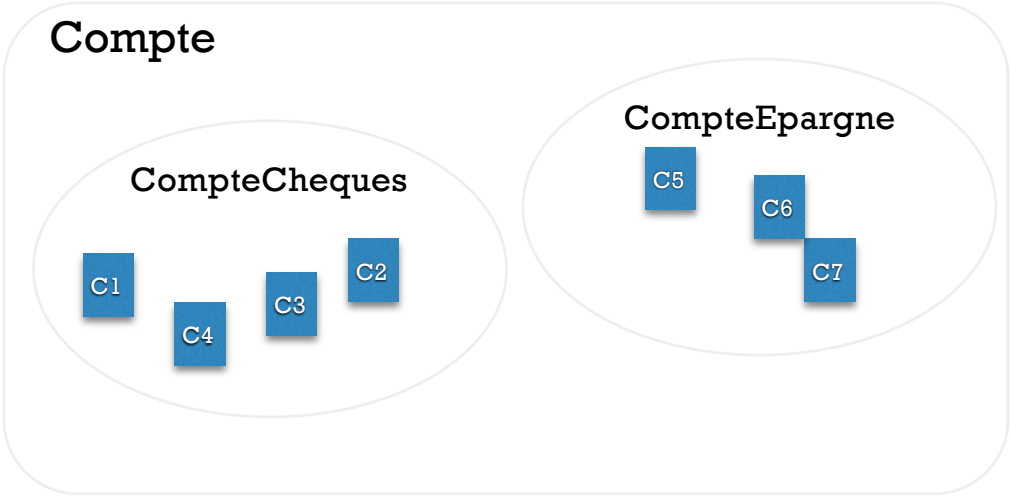
- Une classe spécialisée peut re-définir une opération.
- Une opération redéfinie remplace l'opération héritée de la super-classe.

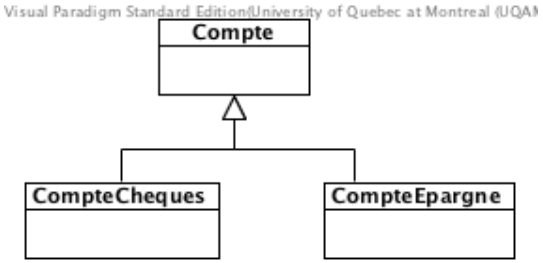
- La classe Rectangle re-définit l'opération *calculerPerimetre*.



HÉRITAGE ET VISION ENSEMBLISTE

- Toute instance d'une sous-classe est aussi une instance de la super-classe.

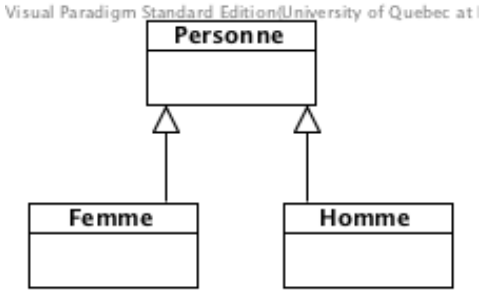


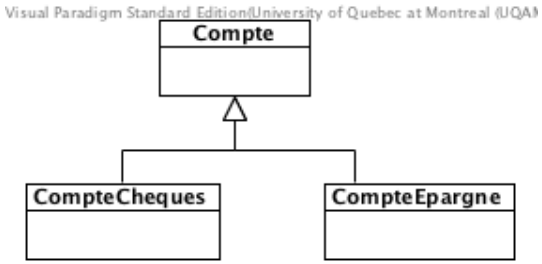


- C1, C2, C3 et C4 sont des CompteCheque et des Compte.s
- C5, C6 et C7 sont des CompteEpargne et des Compte.

HÉRITAGE ET VISION ENSEMBLISTE

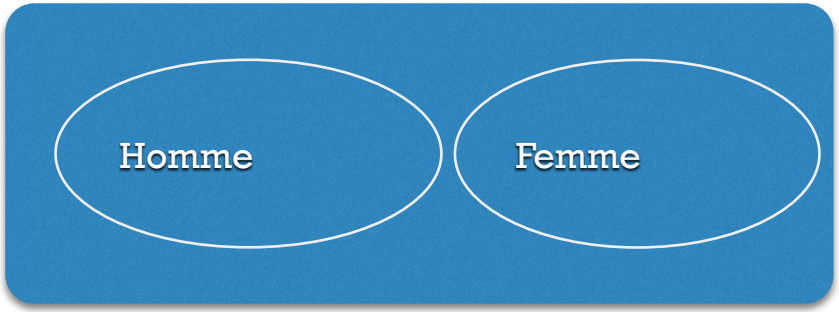
- Une décomposition par héritage peut être:
  - Totale: toute instance de la super-classe est obligatoirement une instance de l'une de ses sous-classe.
  - Non-totale: la super-classe (Compte) admet des instances propres (i.e. ni-chèque, ni-épargne).

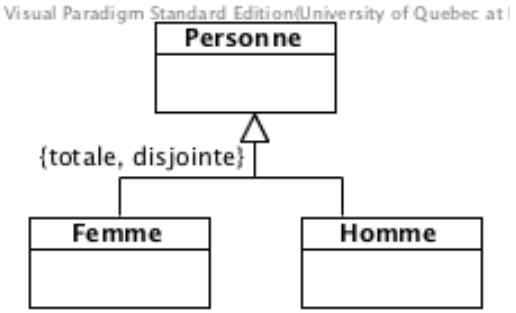




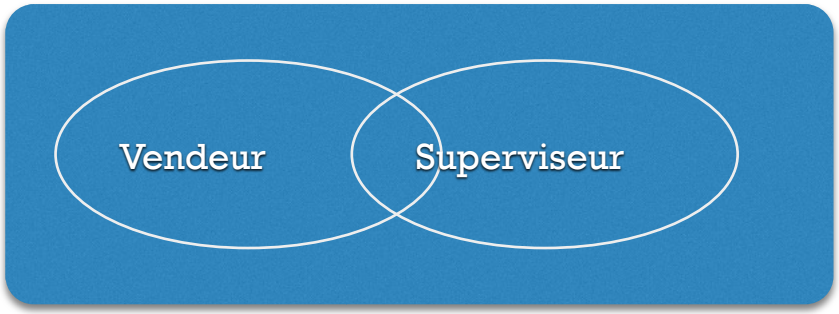
HÉRITAGE ET VISION ENSEMBLISTE

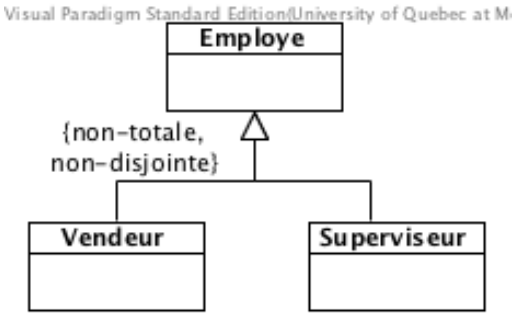
- Une relation d'héritage peut être disjointe





- Ou non-disjointe



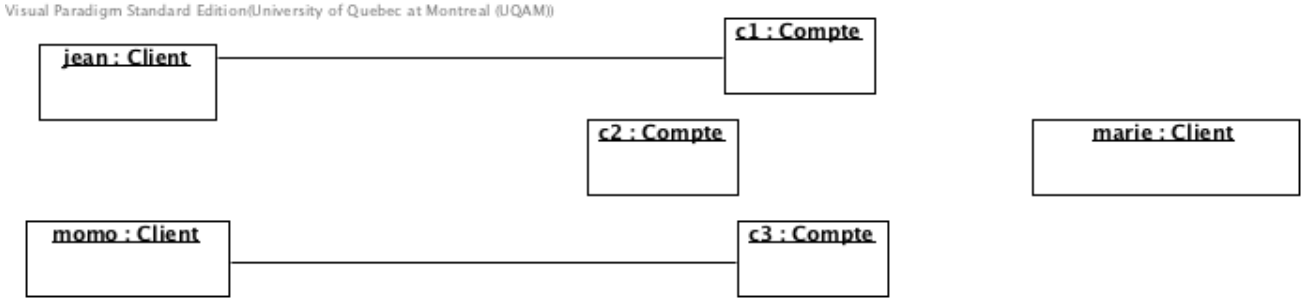


LES ASSOCIATIONS

LES ASSOCIATIONS

- Une association entre deux classes définit une relation entre les instances de chacune des classes.

- Un client donné peut être lié à un compte. Un compte donné peut-être lié à un client.



LES RÔLES

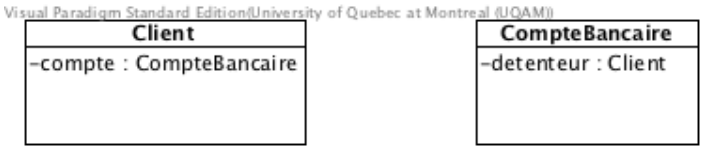
- De part et d'autre d'une association, nous devons mentionner les rôles:

- Le *Client* joue le rôle de *detenteur* pour CompteBancaire.
- Le *CompteBancaire* jour le rôle de *compte* pour Client.

LES RÔLES

- Les rôles d'une association peuvent être assimilés à des attributs de la classe:

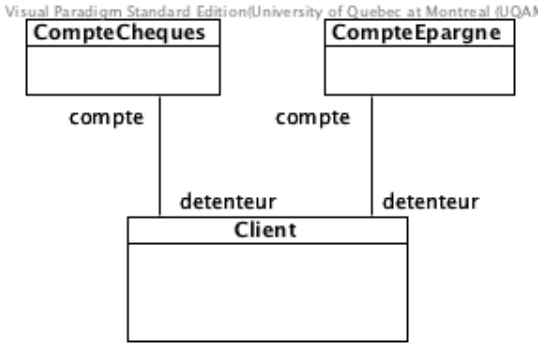
est syntaxiquement équivalent à:



- La seconde notation est moins lisible et déconseillée!

LES RÔLES

- L'identifiant d'un rôle doit respecter les même conditions de nommage qu'un attribut.
- Comme un attribut, on ne peut avoir deux rôles joués, pour la même classe, qui ont des identifiants identiques.
- Trouvez l'erreur:



MULTIPLICITÉ

- La multiplicité permet de contraindre le nombre permis d'instances qui seront liées par une association.

- Un *CompteBancaire* doit avoir un et un seul *Client* qui joue le rôle de *détenteur*.
- Un Client peut avoir un nombre de compte indéfini (incluant 0).

MULTIPLICITÉ

- Notation:
  - min..max : au moins *min* instances, et au maximum *max* instances. *min* et *max* doivent être des entiers positifs.
  - Si une seule valeur est mentionnée, elle est considérée comme la valeur minimale.
  - Le joker * veut dire "indéfini".
    - 1..*: au moins 1 et pas de limite supérieure
    - * : pas de limite inférieure ou supérieure.

NOM DE L'ASSOCIATION

- Pour améliorer la lisibilité, une association peut être nommée.

- Le symbole “>” est utilisé pour préciser le sens de lecture.
- Document contient un nombre indéfini de FormeGeometrique qui jouent le rôle de composantes.
- Le nommage d’association est surtout utilisé dans les modèles d’analyse.

NAVIGABILITÉ

- Par défaut, une association est bidirectionnelle, c.à.d. navigable dans les deux sens:

  - Connaissant un *Client*, nous pouvons obtenir la liste de ses *ComptesBancaire*, ET
  - Connaissant un *CompteBancaire*, nous pouvons identifier son *Client* détenteur.

NAVIGABILITÉ

- Nous pouvons limiter la navigabilité à un seul sens:

  - À partir d'un *Document*, nous pouvons obtenir la liste des *FormeGeometrique* qu'il contient, MAIS
  - À partir d'une *FormeGeometrique* donnée, il n'est pas possible d'obtenir le *Document* qui la contient.

- Note: le rôle du côté non navigable n'est plus nécessaire (mais ce n'est pas faux de le mettre).

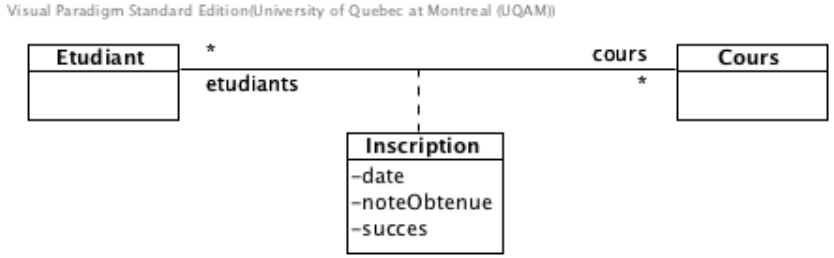
CLASSES ASSOCIATIVES

- Il est parfois nécessaire d'avoir plus d'informations sur le lien qui unit les objets de deux classes.
- Exemple:

Plusieurs informations, relatives à l'inscription de l'étudiant au cours sont nécessaires: date d'inscription, note obtenue, mention de passage ou d'échec, etc.

CLASSES ASSOCIATIVES

- La classe associative permet de répondre à cette problématique;
- Une classe associative est une classe qui porte sur une association entre deux classes



- C'est une classe à part entière!
  Ex.: Elle peut être, elle-même, associée à d'autres classes, hériter d'une autre classe ou être elle même spécialisée.

L'AGRÉGATION ET LA COMPOSITION

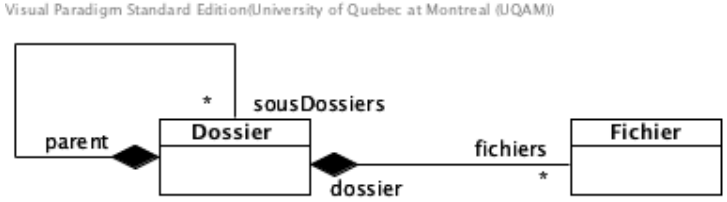
- Parfois, certains objets sont des composantes d'un autre objet ;
- Nécessaire de refléter cette relation spéciale entre objets car elle a des implications sur l'implémentation du logiciel ;
- UML fournit deux construits pour représenter cette relation: la composition et l'agrégation.

LA COMPOSITION

- La composition permet de représenter une relation de composant-composé entre les objets de deux classes reliées par une association.
- Exemple:
  - Un site Web se compose d'un ensemble de pages
  - Un cours se compose des séance de la séquence de seances de cours

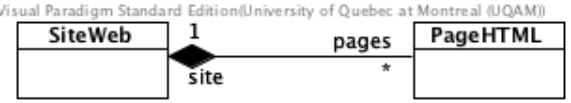
LA COMPOSITION

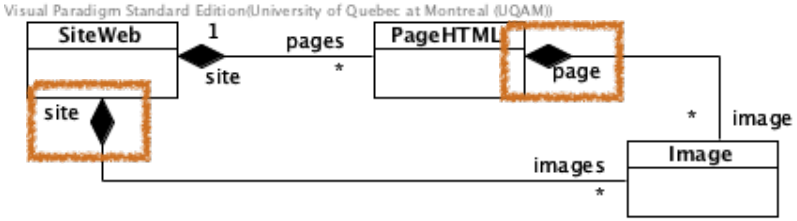
- Une composition est une association représentée par une losange plein du côté du composé



LA COMPOSITION

- Un objet ne peut être lié, par composition, à plus d'un objet!
  - La cardinalité du côté du composé est au maximum 1 (lorsque ce n'est pas mentionné, c'est 0..1).
  - Ceci est interdit:



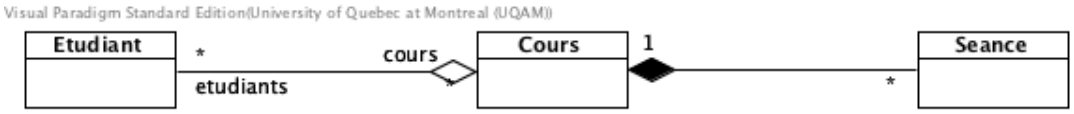


L'AGRÉGATION

- L’agrégation permet également de représenter une relation de composé-composant;
- L’agrégation est aussi appelée une composition partagée (la composition est également appelée une agrégation forte)
- Dans une agrégation, un objet peut-être un composant de plus d’un objet.
- Se représente par une association avec un losange vide du coté du composé

L'AGRÉGATION

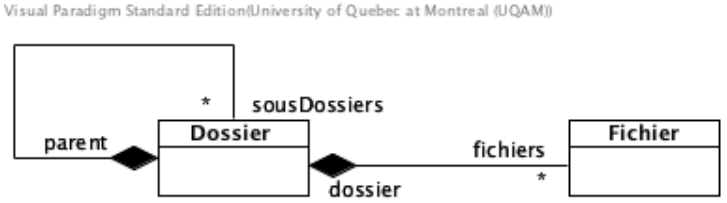
- Exemples:



COMPOSITION OU AGRÉGATION?

- Agrégation: si l'objet composant a une existence propre
  - Si je supprime le composé (le cours), le composant (l'étudiant) continue d'exister dans le système

- Composition sinon
  - La suppression d'un dossier entraine la suppression de ses fichiers et sous-dossiers.



COMPOSITION OU AGRÉGATION?

- Une voiture se compose d'un moteur et de 4 roues. Devons-nous représenter ces relations avec des compositions ou des agrégations?