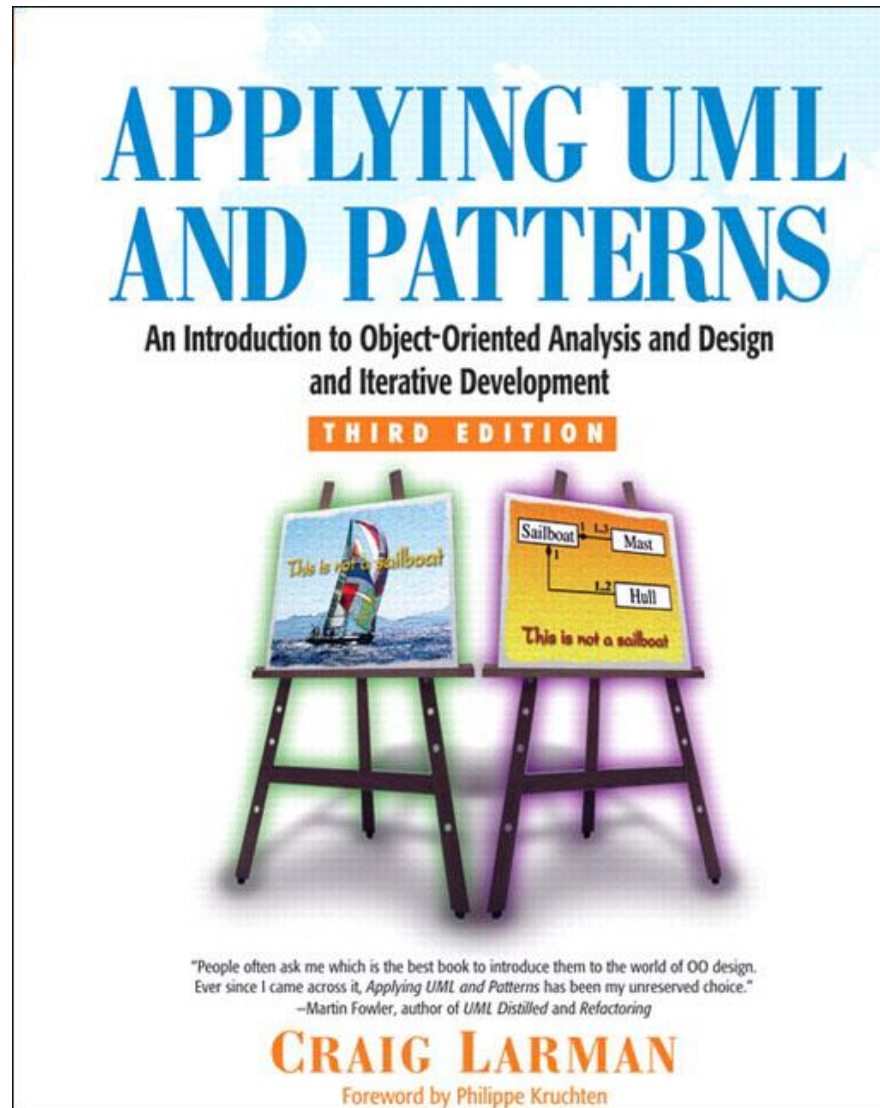


# Modélisation du domaine



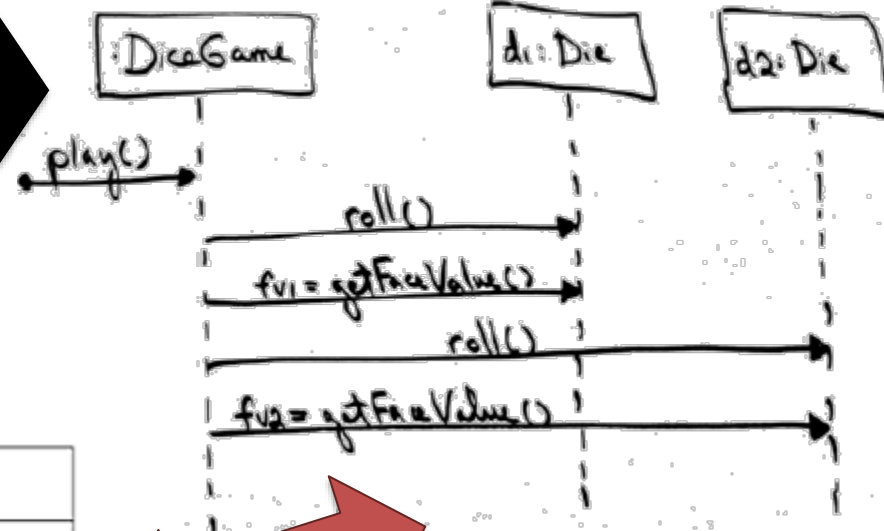
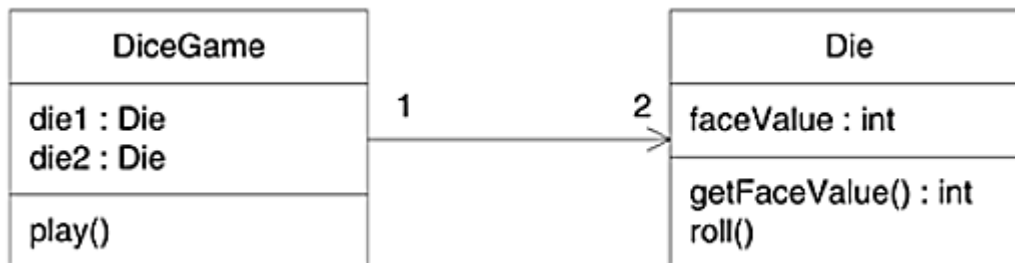
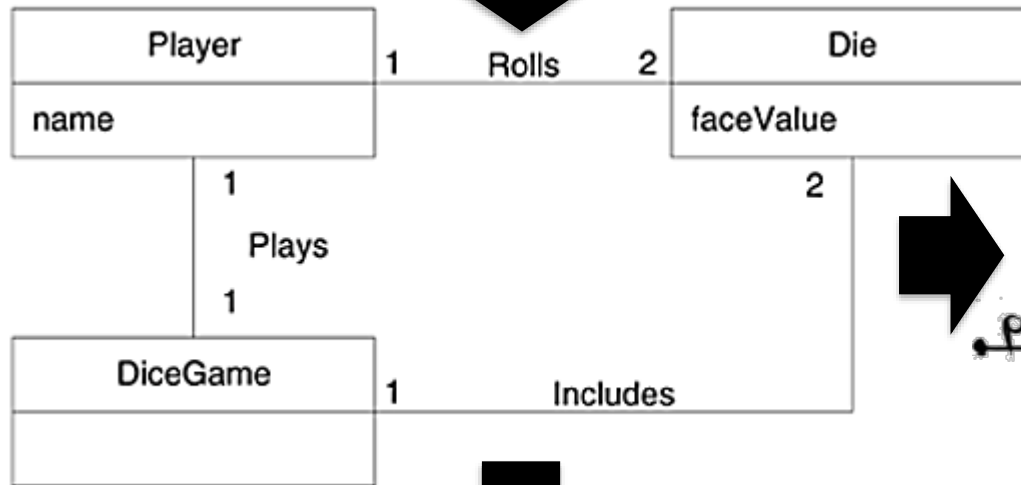
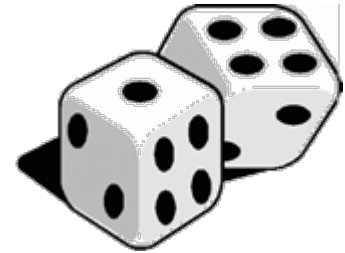
**Prof. Eugene Syriani**

# Livre à suivre

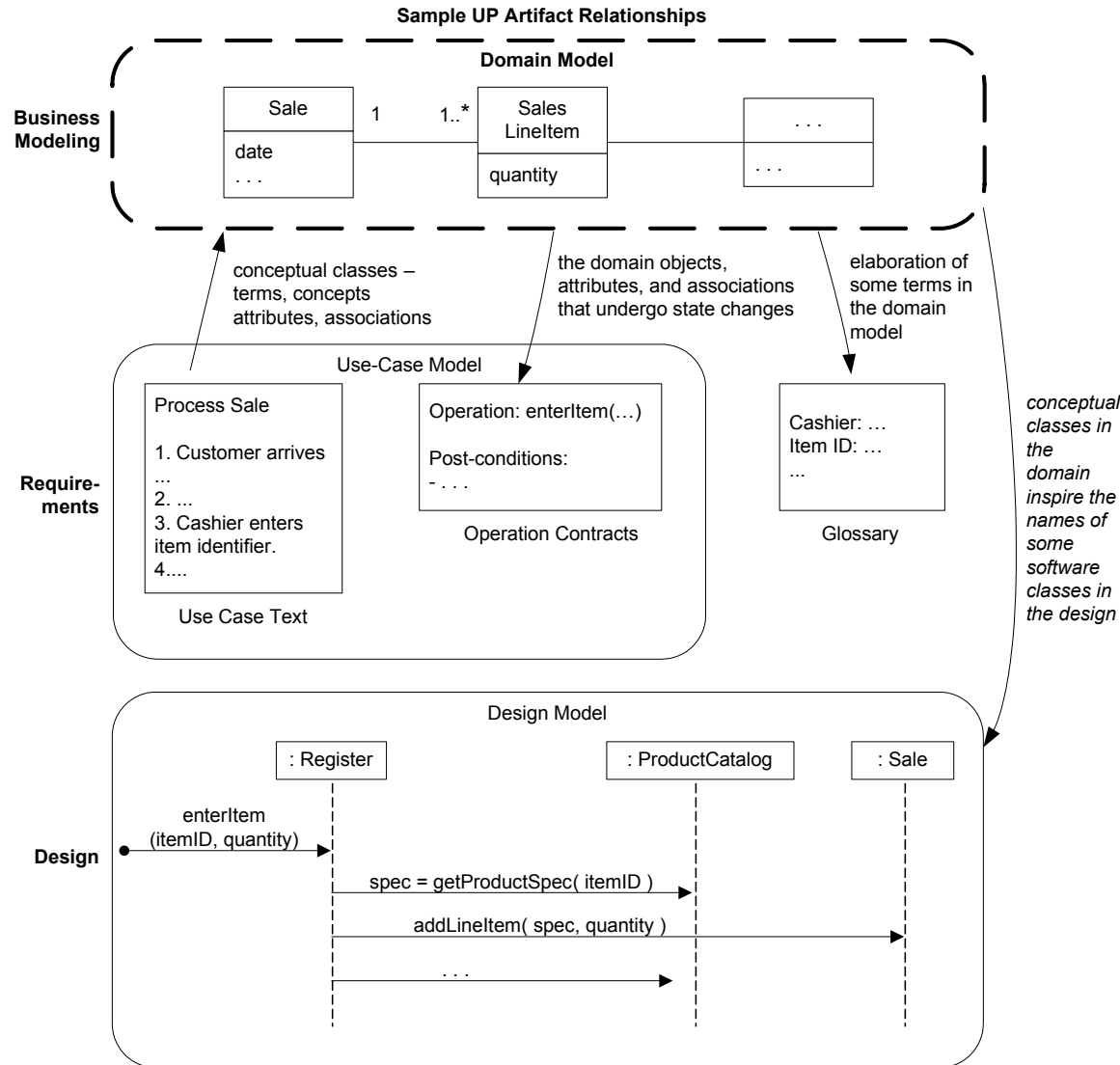


# Vue d'ensemble

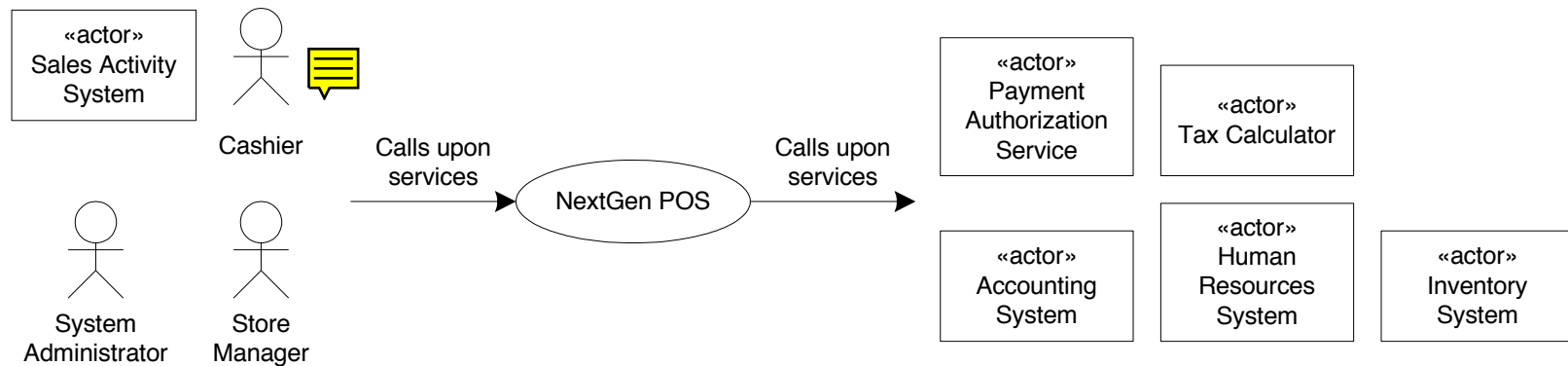
1. Le joueur demande de lancer les dés.
2. Le système présente le résultat.
  - 2.1 Si la face des dés est égale à sept, le joueur gagne.
  - 2.1a Sinon il perd.




# Influence entre les artéfacts



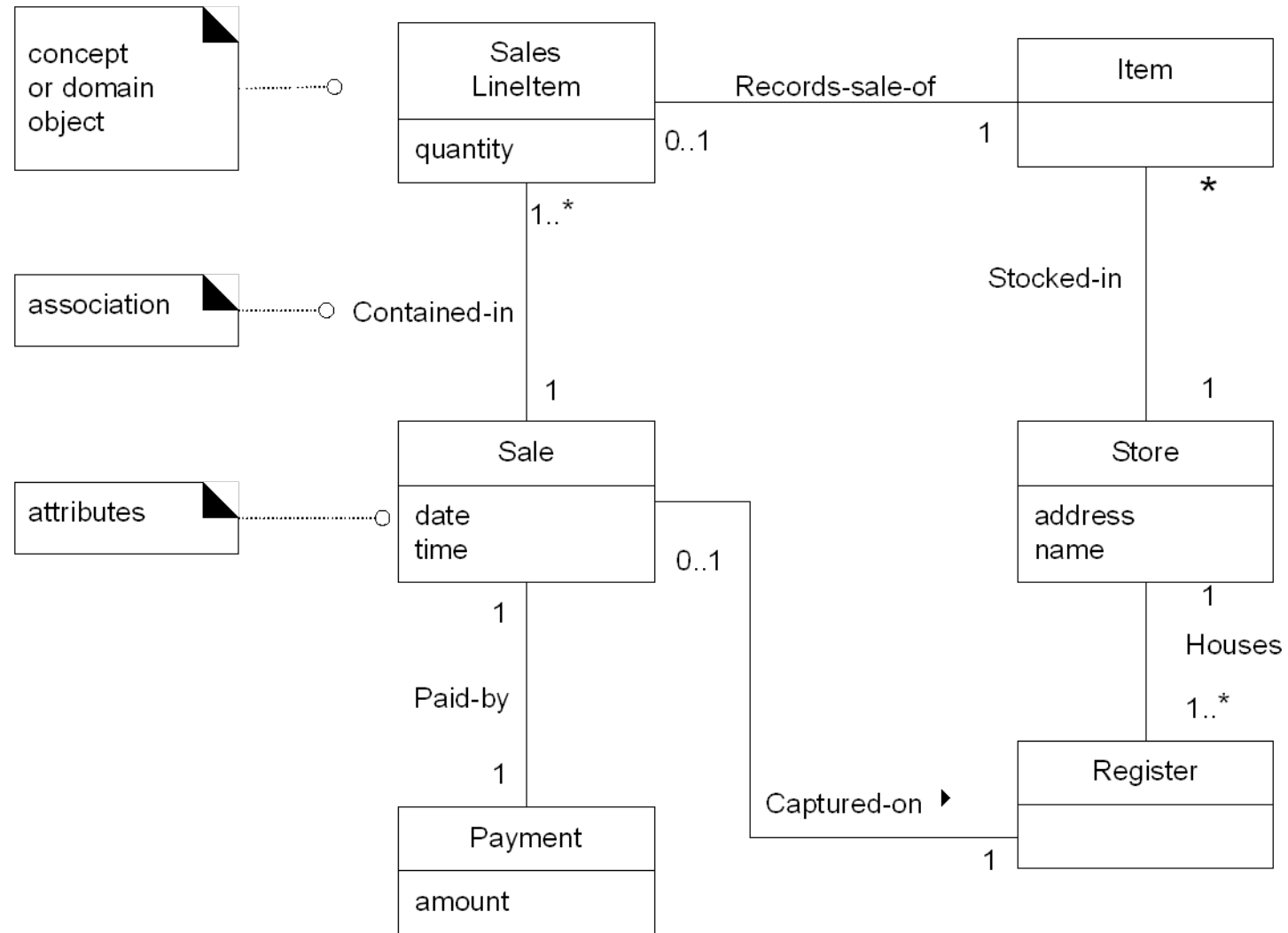
# Cas d'étude: Point de Vente



# Modèle du domaine

- Représentation **visuelle** de classes conceptuelles ou d'objets dans des situations réelles au sein d'un domaine
  - Classe conceptuelle
  - Classe de logiciel
  - Classe d'implémentation 
- Perspective conceptuelle qui montre
  - Classes conceptuelles
  - Associations entre ces classes
  - Attributs de ces classes

# Dictionnaire visuel

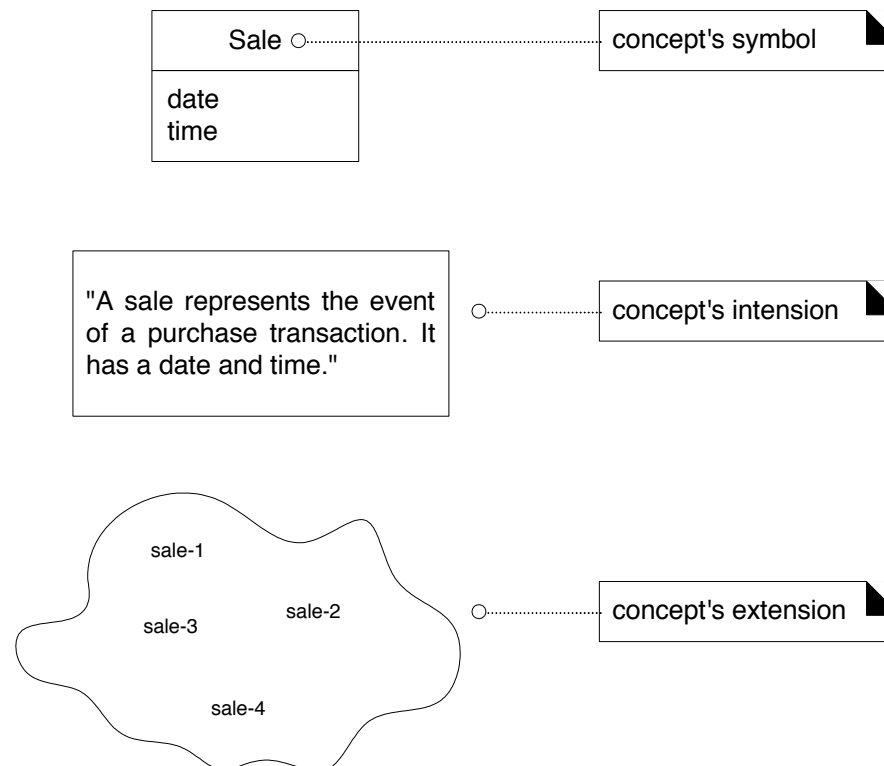


# Classes

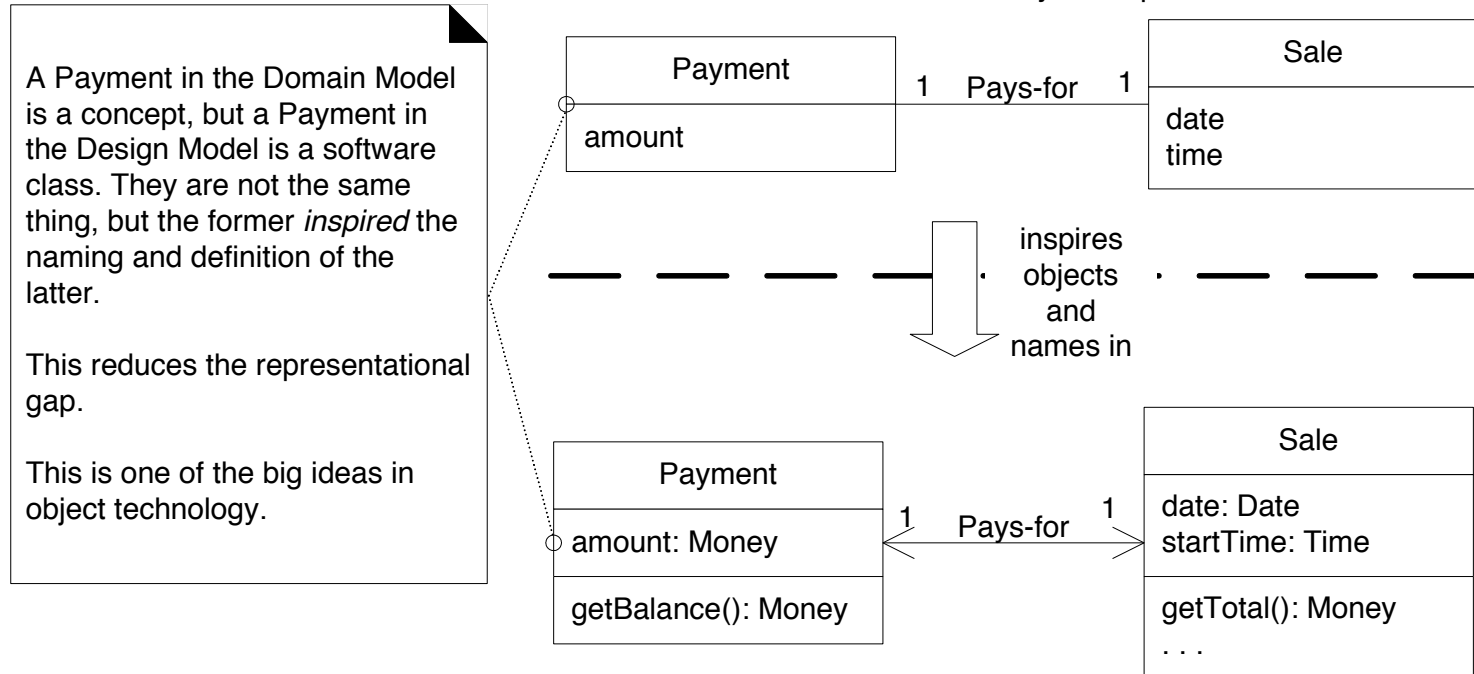


# Classes conceptuelles

- Symbole : nom qui décrit une **abstraction**
- Intension : définition qui spécifie le **contexte**
- Extension : exemples d'application où on retrouve le concept



# Pourquoi créer un modèle du domaine?



## UP Design Model


The object-oriented developer has taken inspiration from the real world domain in creating software classes.

Therefore, the representational gap between how stakeholders conceive the domain, and its representation in software, has been lowered.

# Comment trouver les classes conceptuelles?


- Réutiliser ou modifier des modèles existants
- Établir une liste de catégories
- Méthode d'extraction de noms

# Liste de catégories des classes conceptuelles

Catégorie	Exemples
Transaction <i>Critique (car implique de l'argent), commencer par ça</i>	Sale, Payment Reservation
Ligne d'items de transaction <i>Souvent une transaction vient avec, poursuivre avec</i>	SalesLineItem
Produit ou service relié à une transaction ou ligne <i>Poursuivre avec</i>	Item Flight, Seat, Meal 
Où est enregistrée la transaction? <i>Important</i>	Register FlightManifest
Rôle des personnes reliés à une transaction; acteurs d'un cas d'utilisation	Cashier, Customer, Store Passenger, Airline
Lieu de la transaction ou du service	Store Airport, Plane, Seat
Événements importants, souvent avec un temps et lieu à se rappeler	Sale, Payment Flight
Instruments financiers	Cash, Cheque, LineOfCredit, Voucher

# Extraction de noms

## Scénario de succès principal:

1. Le **client** arrive à un **POS** avec des **biens** et/ou **services** à acheter.
2. Le **caissier** commence la **vente**.
3. Le caissier entre les **items** identifiés. 
4. Le système enregistre une **ligne de vente d'item** et présente une **description de l'item**, son **prix** et le **total** cumulatif. Le prix est calculé selon les règles de prix.
5. Le caissier répète 2-3 jusqu'à ce qu'il saisisse « terminé ».
6. Le système présente le total et les **taxes** calculées.
7. Le caissier informe verbalement le total au client et demande le **mode de paiement**.
8. Le client paye et le système traite le **paiement**.
9. Le système enregistre la vente complétée et envoie l'information de la vente et du paiement aux systèmes externes de **comptabilité** et d'**inventaire**.
10. Le système présente le **reçu**.
11. Le client quitte avec le reçu et les biens.

## Extensions:

### 8a. Payer comptant:

- 8a.1. Le caissier inscrit le **montant d'argent** perçu.
- 8a.2. Le système présente une **balance** due et ouvre le  **tiroir à caisse**.
- 8a.2. Le caissier y dépose le montant et rend la différence en argent au client.
- 8a.3. Le système enregistre le paiement comptant.

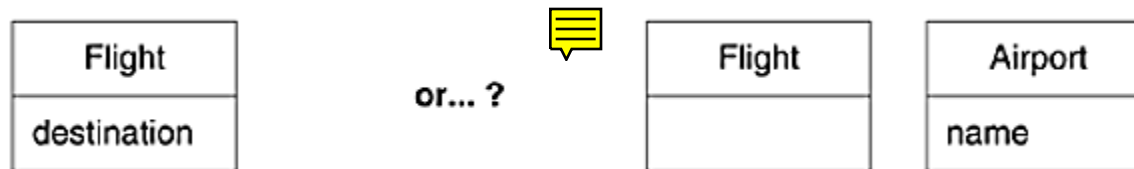
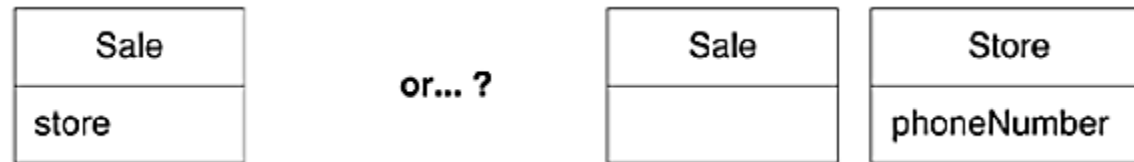


# Première version des concepts modélisés

- Utiliser les mêmes mots que dans le jargon du domaine
- Exclure les fonctionnalités hors de portée ou qui ne sont pas nécessaires
- Ne pas ajouter des choses qui n'apparaissent nulle part

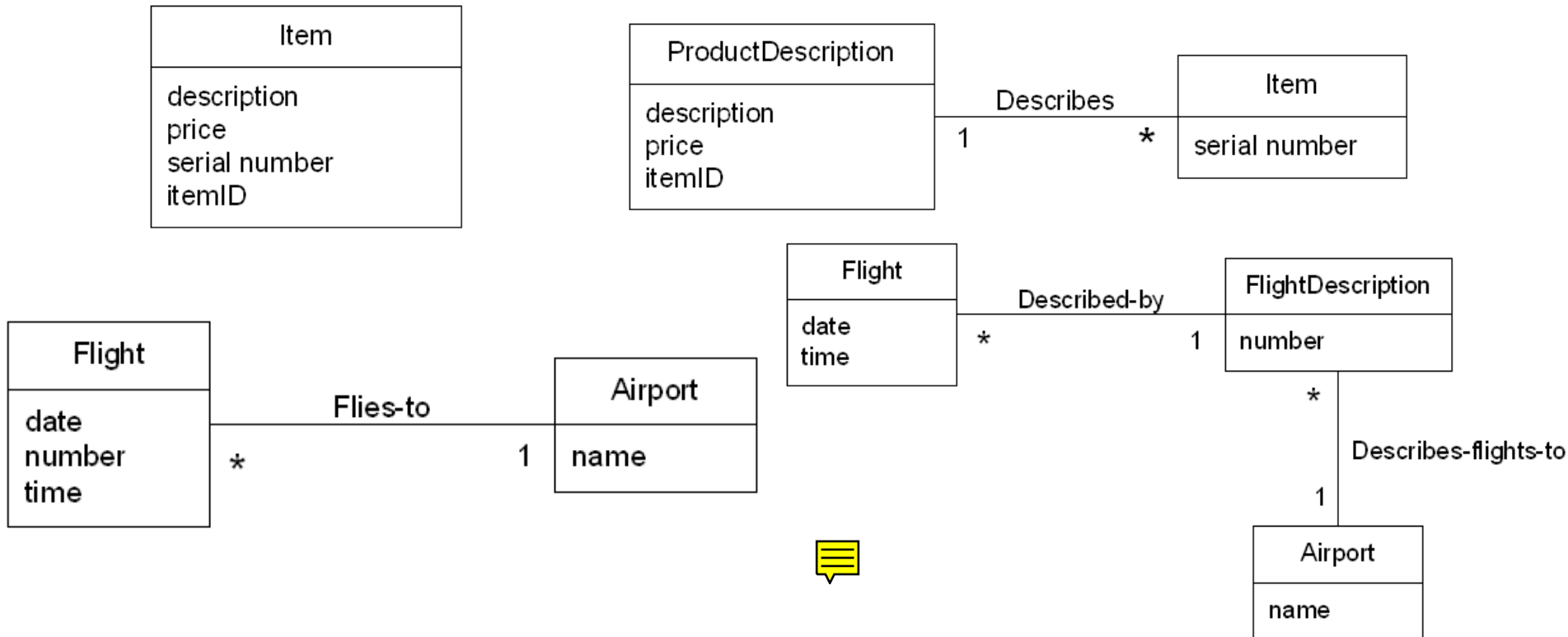


# Comme attribut ou classe?



Si on n'associe pas le concept à un nombre ou du texte, c'est probablement une classe conceptuelle, pas un attribut

# Classes descriptives



La description de l'objet est indépendante de l'existence d'exemples de cet objet et est partagée par plusieurs instance



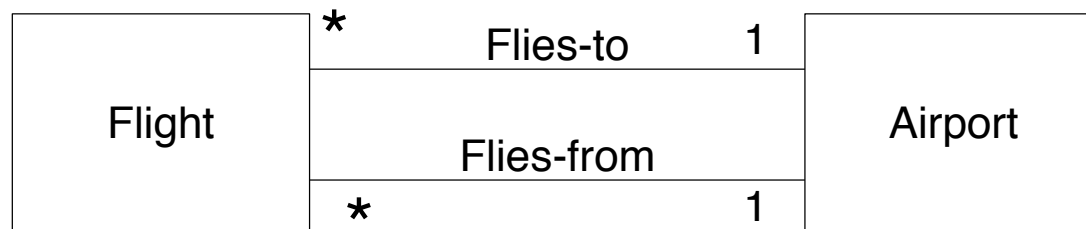
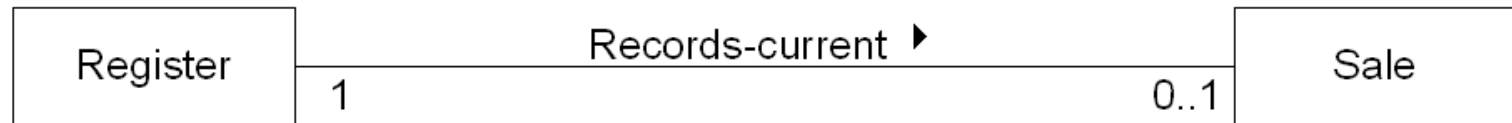
# Associations

# Associations

- Extraire les verbes des cas d'utilisation
  - Vente *Payée-par* PaiementCash
  - Vente *Contient* LigneVenteltem
  - LigneVenteltem *Enregistre-vente-de* Item
- Relation qui doit être préservée sur un intervalle de temps
- Ne mettre que des associations à la première itération!
- Ne pas en créer partout!

# Représentation d'association conceptuelle

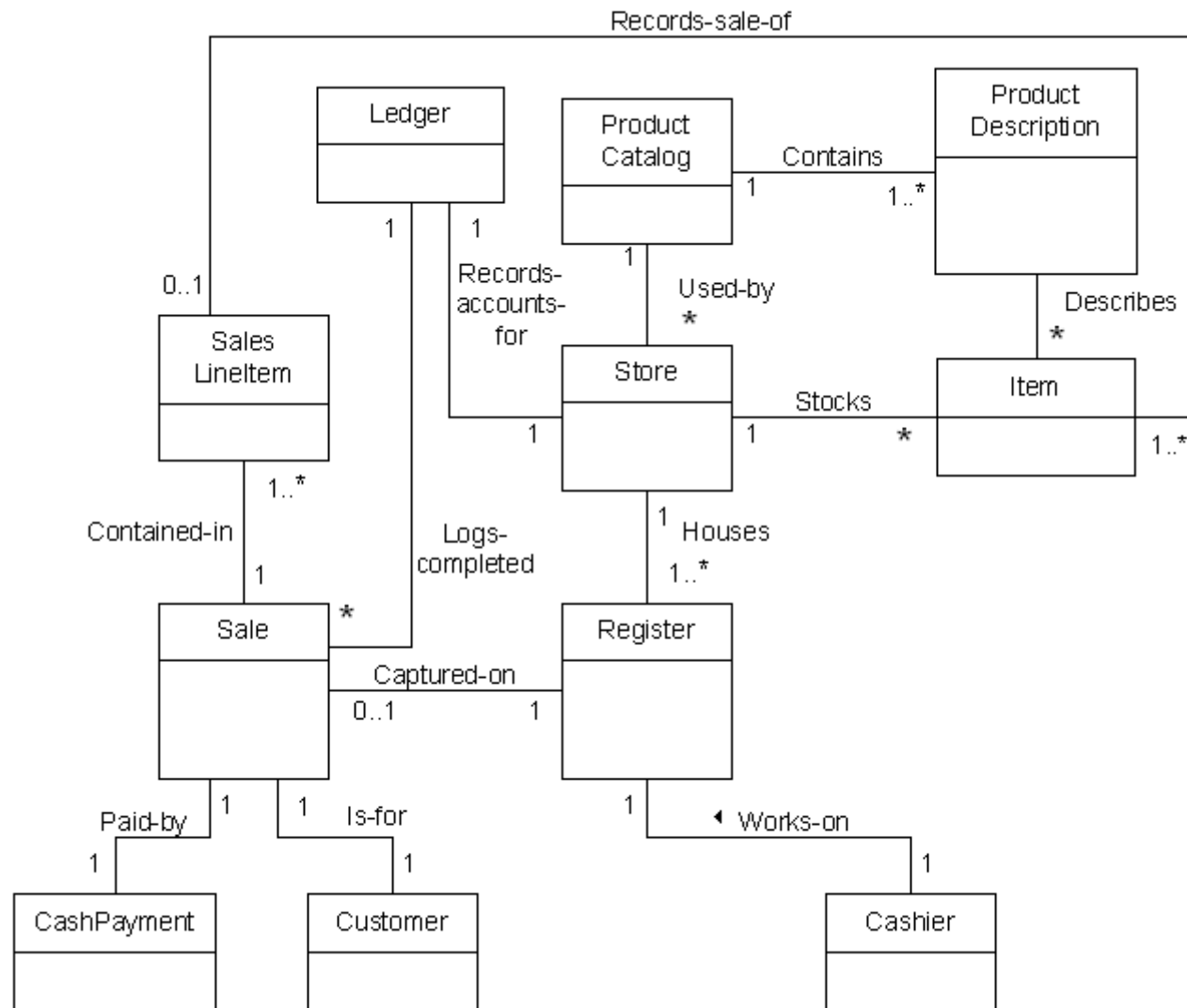
- Nom : verbe ou expression verbale
- Sens de lecture : si pas évident
- Cardinalité : un à un, un à plusieurs, plusieurs à plusieurs



# Cardinalité d'association

Indicateur	Signification
1	Exactement un
0..1	Optionnel
* ou bien 0..*	N'importe quel
1..*	Au moins un
3	Exactement trois
0..5	Optionnel jusqu'à cinq
5..15	Entre cinq et quinze
48,52	Soit 48 soit 52

# Révision 1 du modèle du domaine



# Attributs

# Attribut vs. variable


## Attribut

- Représente une propriété par abstraction
- Indépendant de sa représentation interne
- Défini lors de la conception
- Peut être implémenté par une variable, mais
  - parfois par une combinaison de variables
  - Être dérivé d'autres attributs

## Variable

- Est un mécanisme d'implémentation interne
- Défini lors de l'implémentation

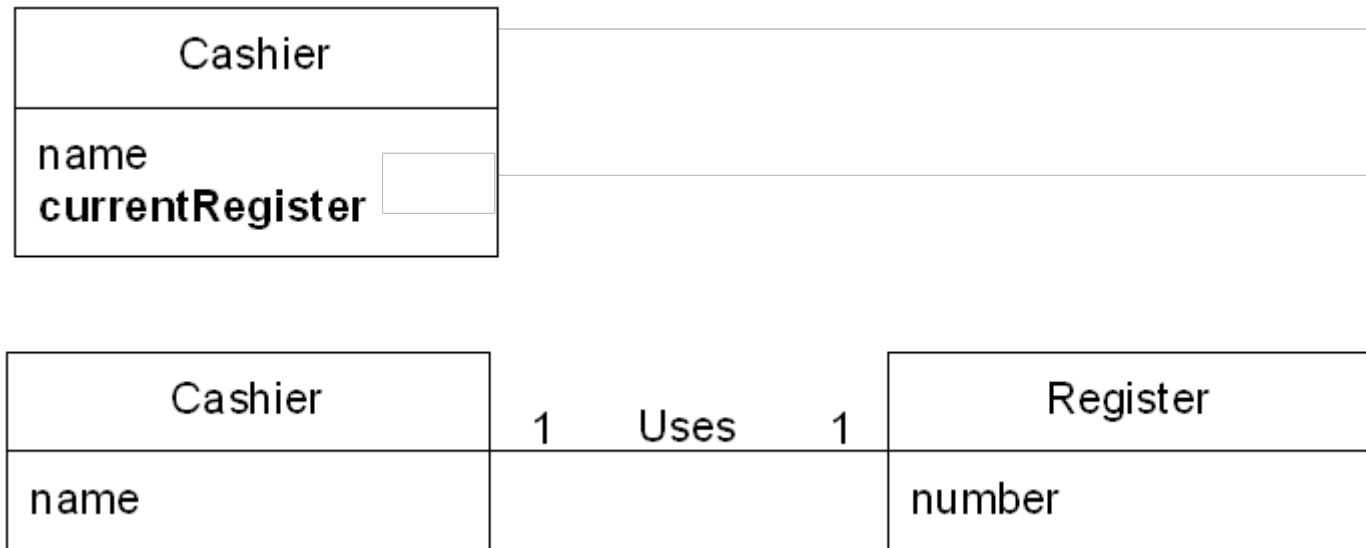
# Attributs

- Valeur de donnée logique d'un objet
- Inclure les attributs si l'information doit être retenue
  - ≠ Variables
- Représente des types de données
  - Boolean, Date (DateTime), Number, Character, String, Time
  - Address, Color, Géométrie (Point, Rectangle), Telephone, PostalCode, énumérations
- Pas de type complexe ou de type classe conceptuelle
  - Association 



# Type d'attribut

- Type complexe ou type classe conceptuelle
- Association

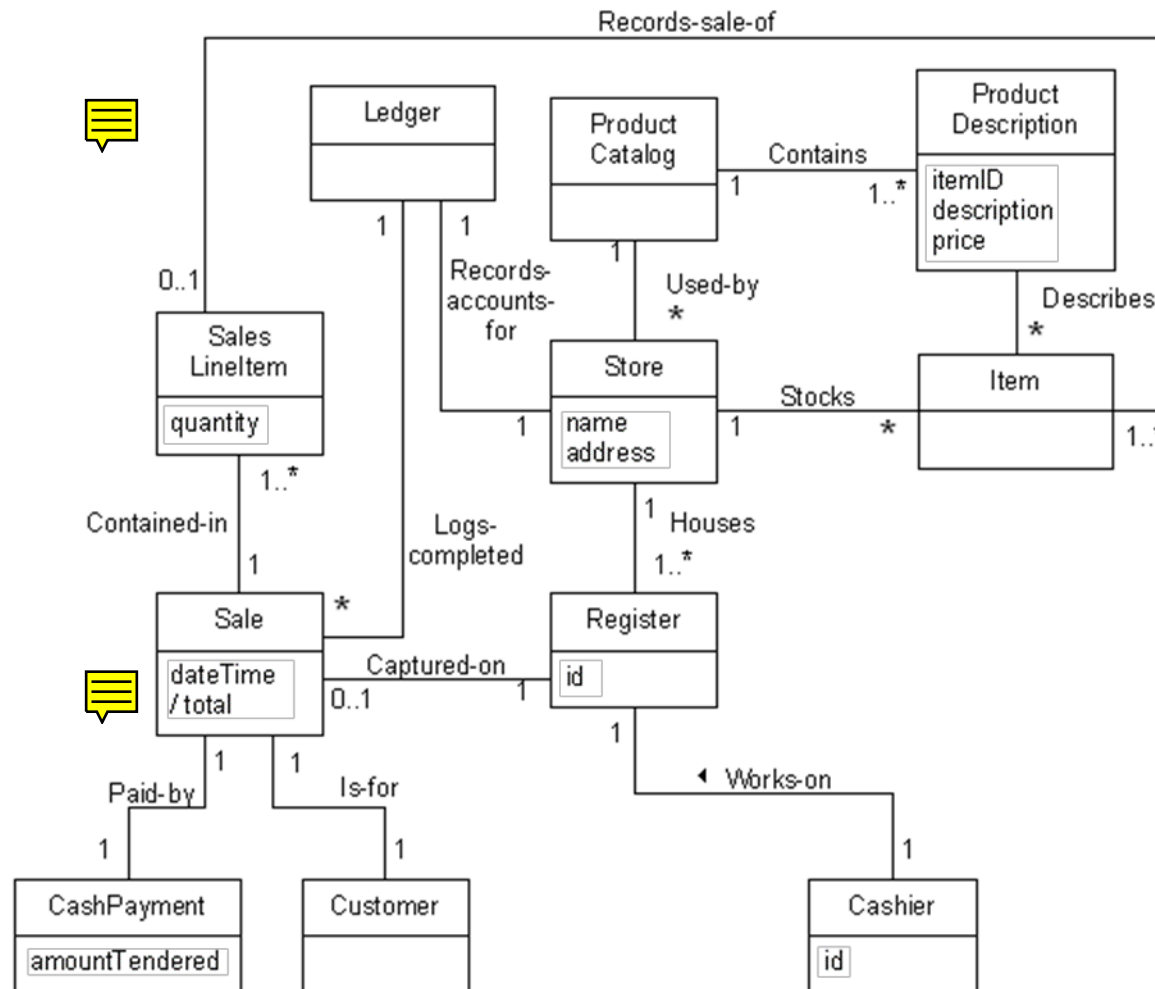


# Nouvelle classe de type de données

Représente un attribut qui était initialement un nombre ou string en une classe de type de donnée si:

- Il est composé de **plusieurs parties**
- Il y a des **opérations** qui lui sont associées, ex. analyse ou validation
- Il a d'autres **attributs**
- C'est une quantité avec une **unité**
- Une **abstraction** d'un ou plusieurs type

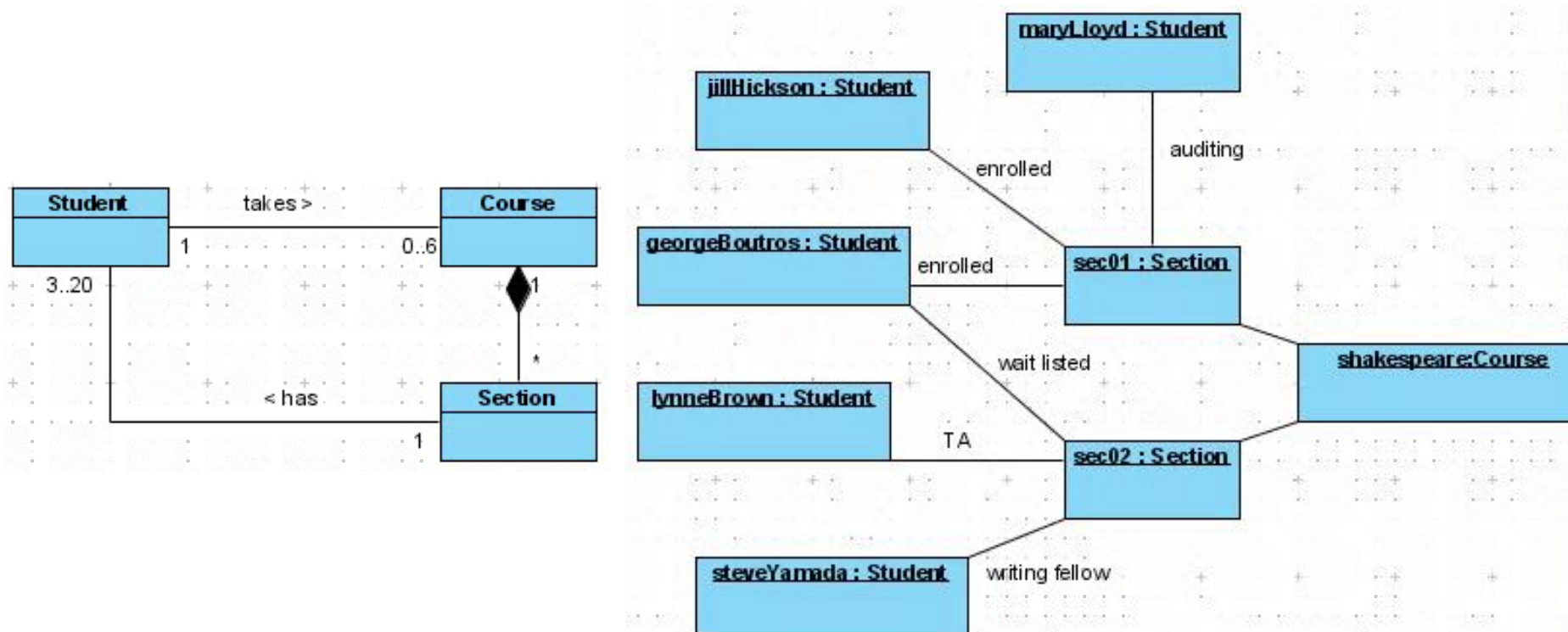
# Révision 2 du modèle du domaine



# Affiner le modèle du domaine

# Toujours penser en terme d'objets

- Diagramme de classe définit toutes les configurations d'objet/liens possibles
- Utiliser le diagramme d'objet pour formellement comprendre les contraintes et cardinalités des associations



# Plus d'extensions

7b. Paying by credit:

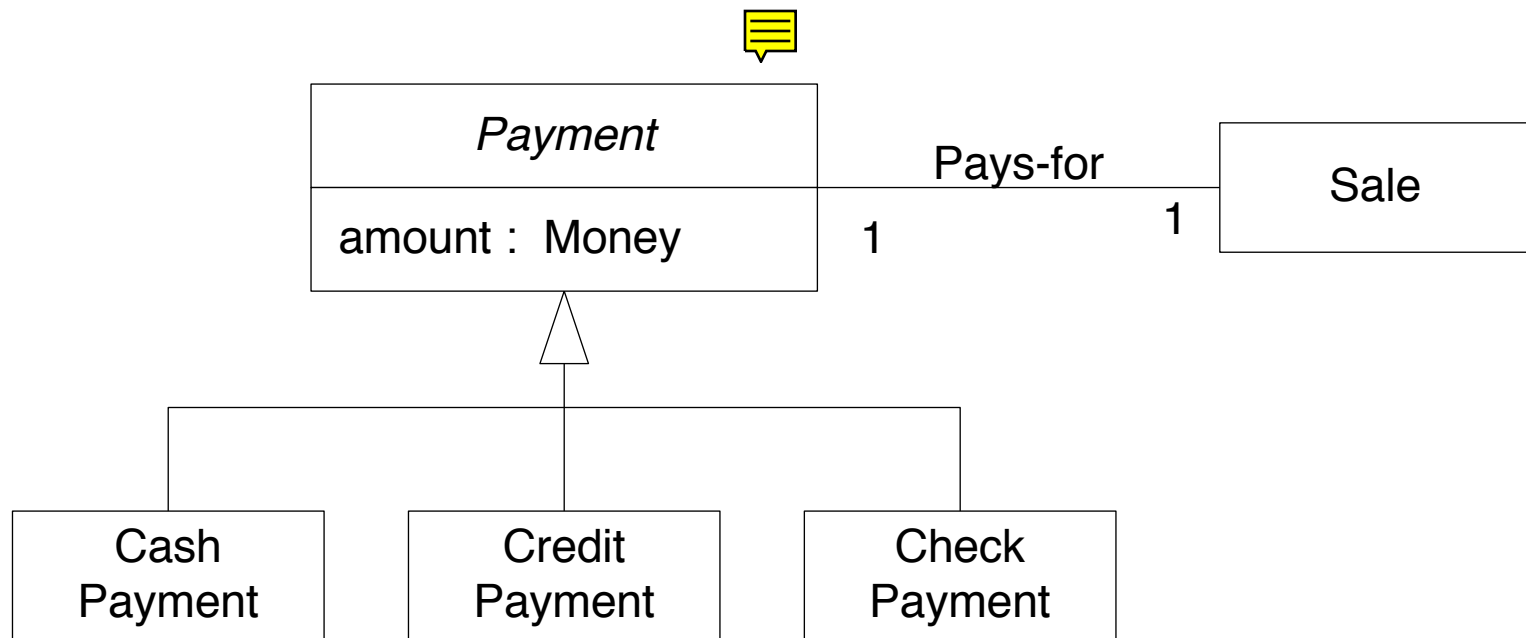
1. Customer enters their **credit account information**.
- System sends **payment authorization request** to an external **Payment Authorization Service** System, and requests **payment approval**.
  - 2a. System detects failure to collaborate with external system:
    1. System signals error to Cashier.
    - Cashier asks Customer for alternate payment.
- System receives **payment approval** and signals approval to Cashier.
  - 3a. System receives **payment denial**:
    1. System signals denial to Cashier.
    - Cashier asks Customer for alternate payment.
- System records the **credit payment**, which includes the payment approval.
- System presents credit payment signature input mechanism.
- Cashier asks Customer for a credit payment signature. Customer enters signature.

7c. Paying by check:

1. The Customer writes a **check**, and gives it and their **driver's license** to the Cashier.
2. Cashier writes the driver's license number on the check, enters it, and requests **check payment authorization**.
3. Generates a **check payment request** and sends it to an external **Check Authorization Service**.
4. Receives a check payment approval and signals approval to Cashier.
5. System records the **check payment**, which includes the payment approval.

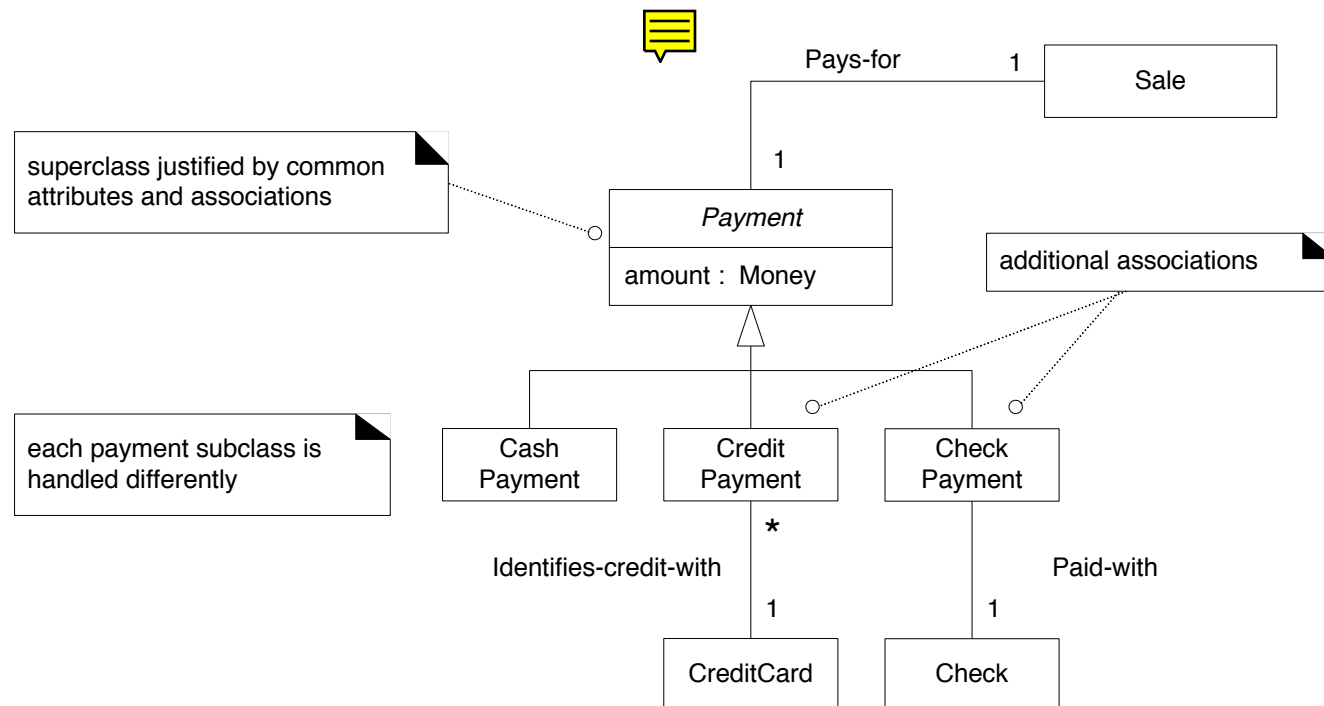
# Généralisation

- Tous les attributs et associations doivent être applicables sur toutes les sous-classes
- La règle du « est un »



# Spécialisation par différence

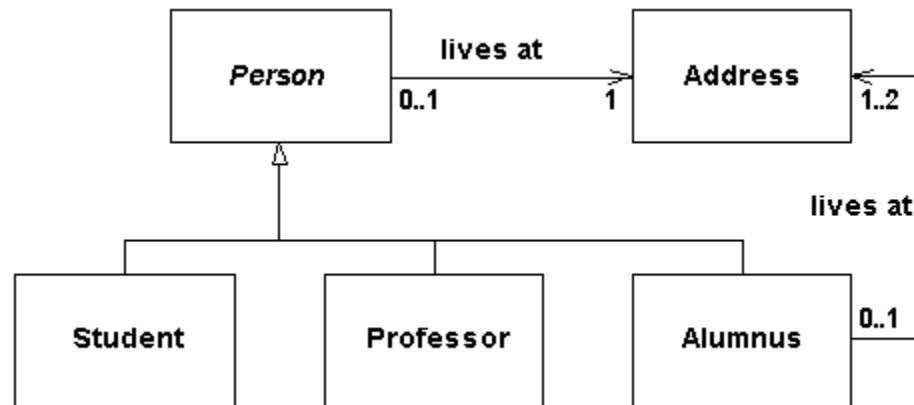
- Sous-classe a des attributs ou associations supplémentaires
- Sous-classe est exécutée, manipulée ou réagit différemment
- Sous-classes représente un concept animé qui se comporte différemment





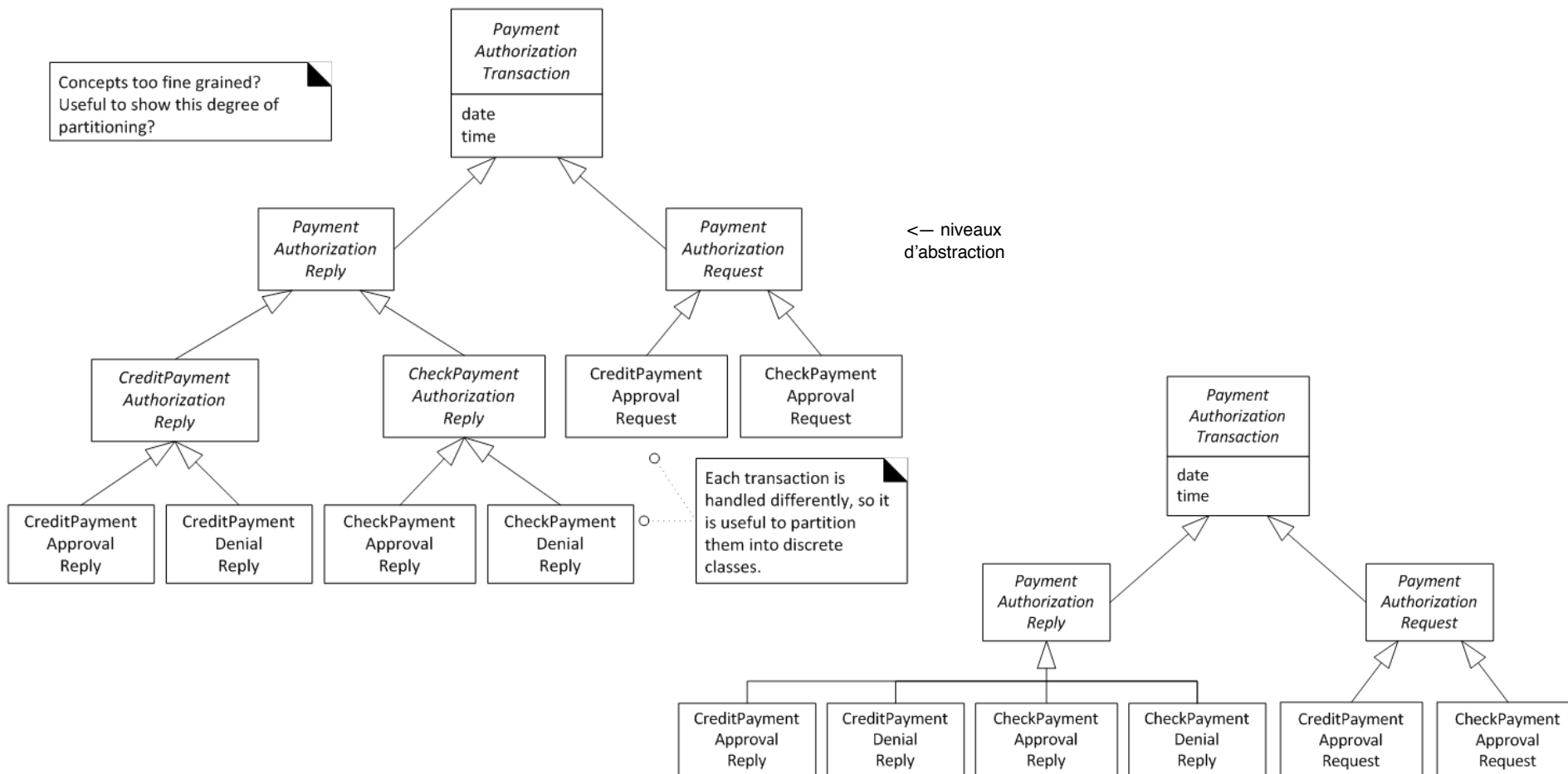
# Redéfinition (*overriding*) d'attributs et d'associations

on override la relation pour la sous classe



# Classe abstraite

- Pour une taxonomie et hiérarchie de concepts
- Réutilisation de l'interface par plusieurs sous-classes

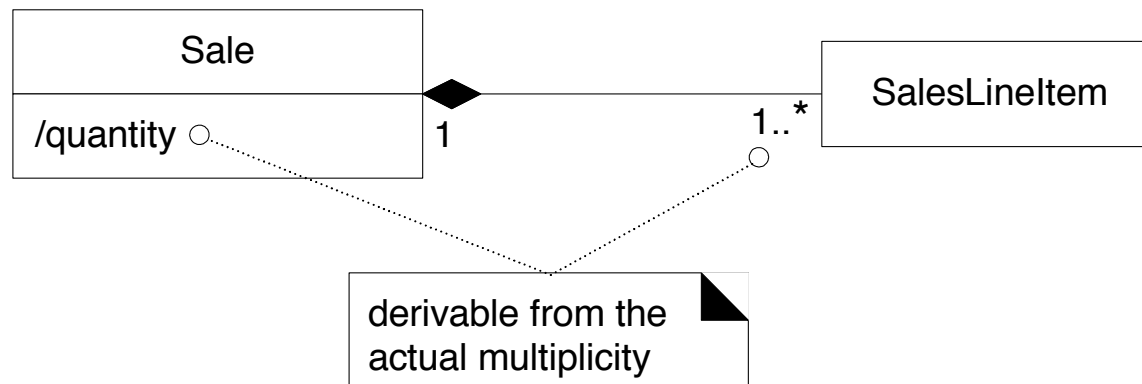
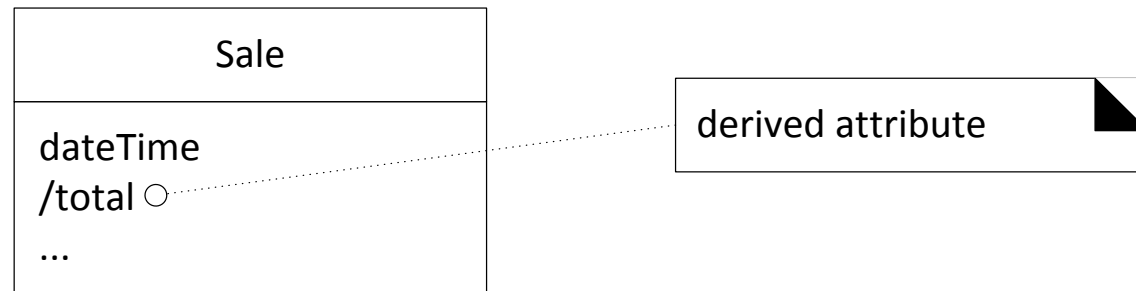


# Implications d'abstraction

- A est une classe concrète et B et C la spécialisent
- Cela implique qu'il peut y avoir une nouvelle classe D qui pourrait la spécialiser dans le futur
- Si A est abstraite, cela implique que la spécialisation est complète
- Aucune nouvelle classe D ne devrait être ajoutée dans le futur

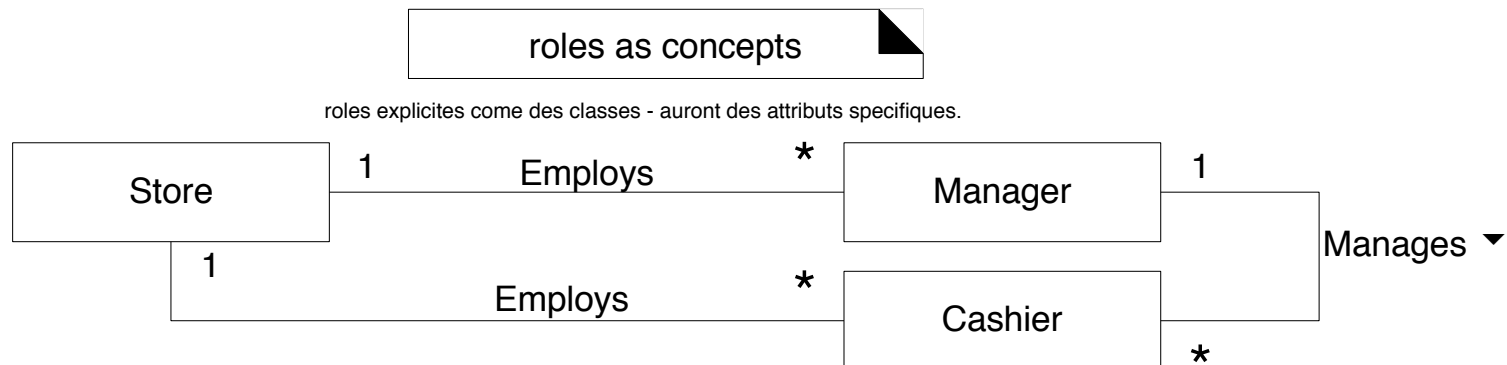
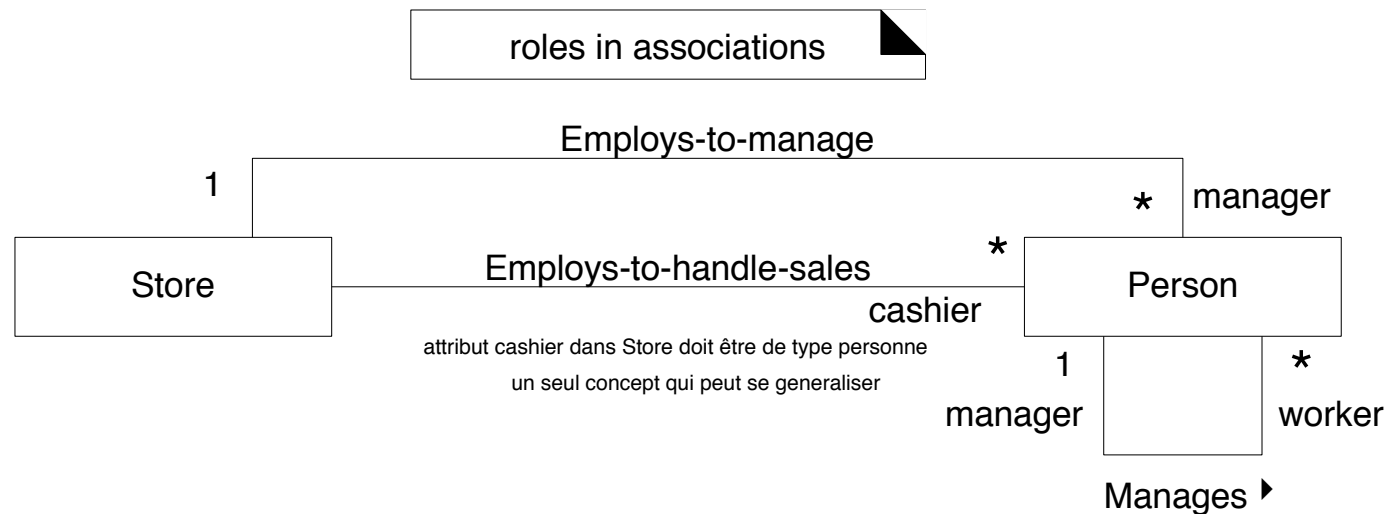
# Attribut dérivé

- Éviter le plus possible
- N'ajouter que si c'est important dans la terminologie et si l'exclure nuit à la compréhension



# Rôles

- Rôles dans les associations vs. rôles modélisés comme concepts



# Attributs communs

- Le service d'autorisation assigne un ID du marchand à chaque magasin pour pouvoir l'identifier. L'ID doit être présent quand le magasin envoie une requête de paiement. Un magasin peut avoir différents ID par service (Visa, MasterCard, ...)

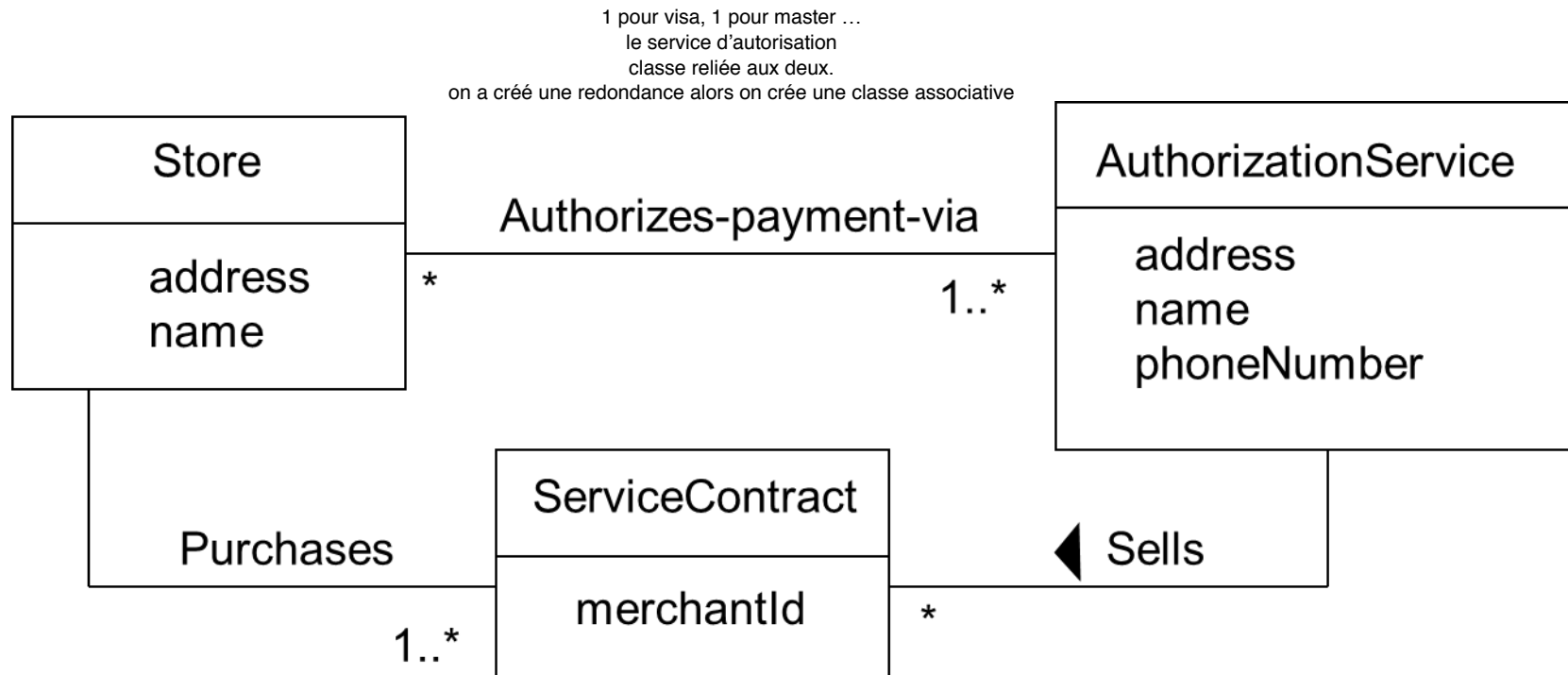
ID utilisé seulement lors de la requete - alors pas dans Store  
AuthorizationService - on ne code pas VISA - un ID gère plusieurs magasins, alors il faudrait avoir plusieurs instances de la classe AuthorizedService.  
Alors — on le met

- Où placer l'ID du marchand?
- Dans Magasin?
- Dans ServiceAutorisation?



# Attributs communs

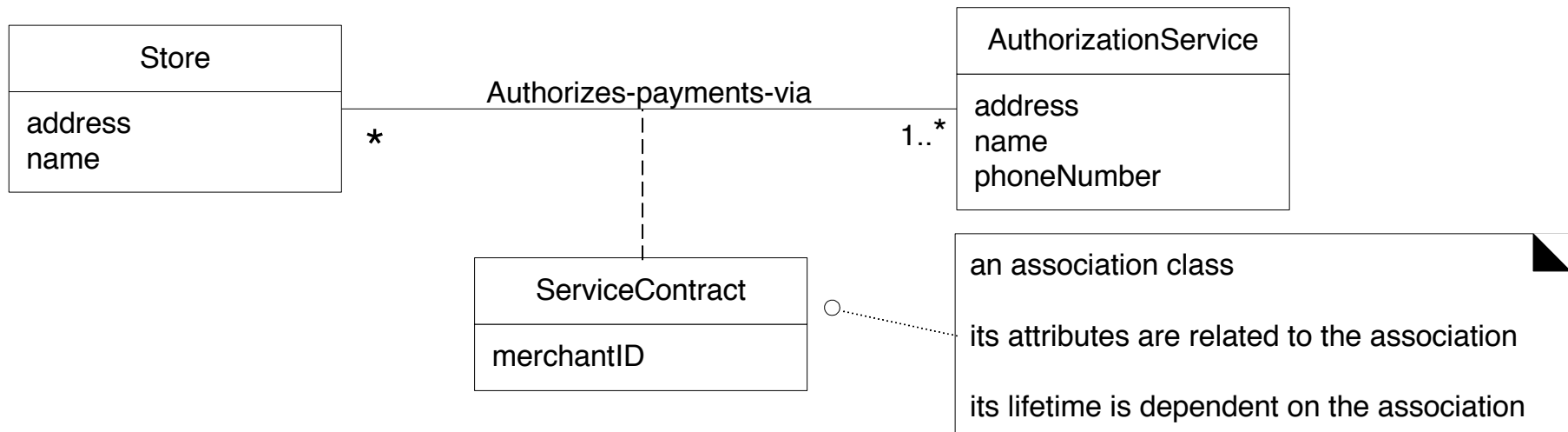
- Si un attribut a peut avoir plusieurs valeurs simultanées pour un objet de la class C, placer a dans une autre classe qui est associée à a



# Classe associative

- Mais l'ID du marchand est reliée à l'association entre Magasin et ServiceAutorisation

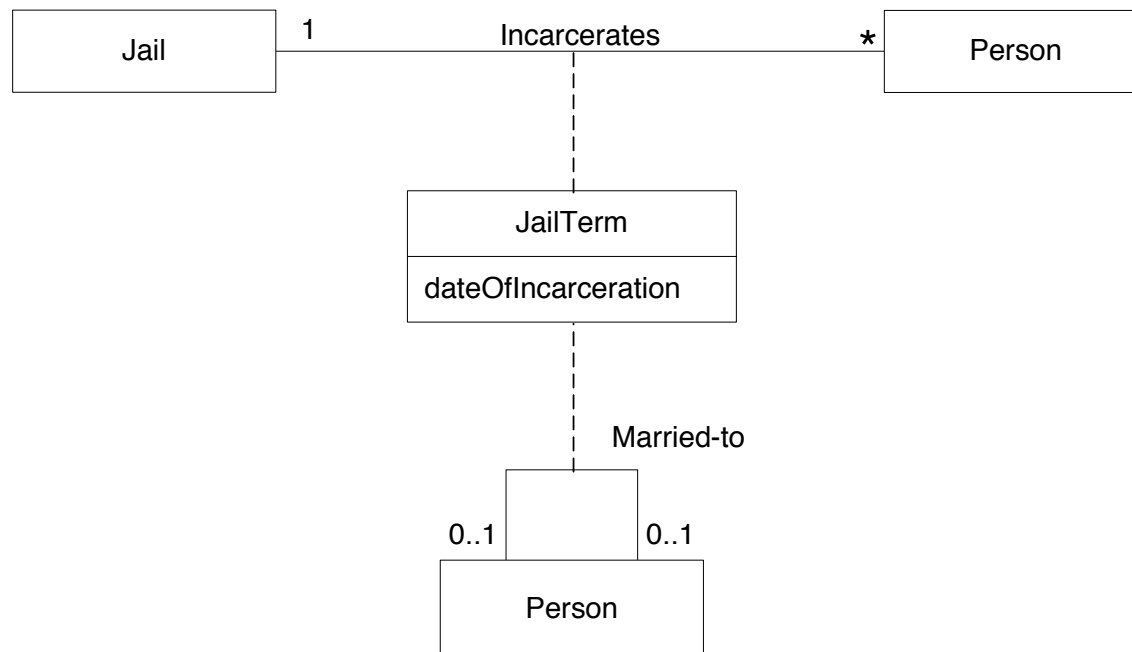
ni la source ni la destination ont l'information, elle l'apporte lors de la communication





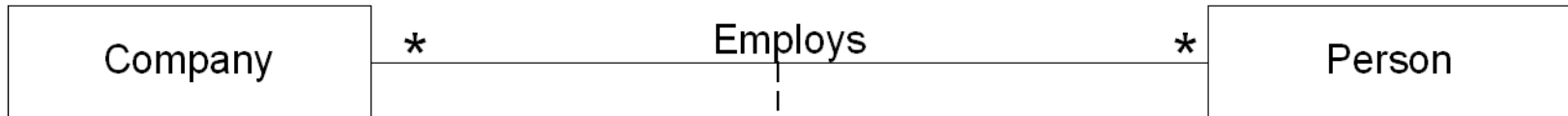
# À la fois classe et association

- C'est à la fois une association entre deux classes
  - Cardinalités, rôles
- Et une classe
  - Attributs, méthodes, héritage, autres associations

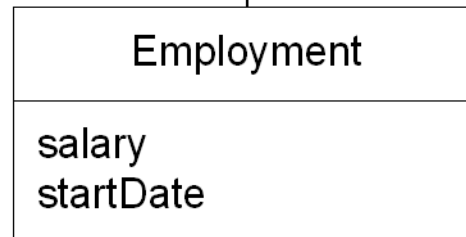


# Preserver de l'information sur la relation entre deux objets

par défaut 1 1  
sinon, mettre les cardinalités !!

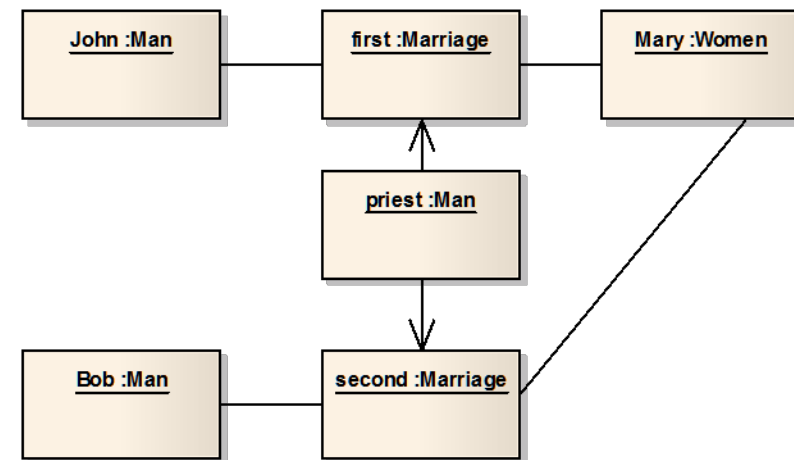
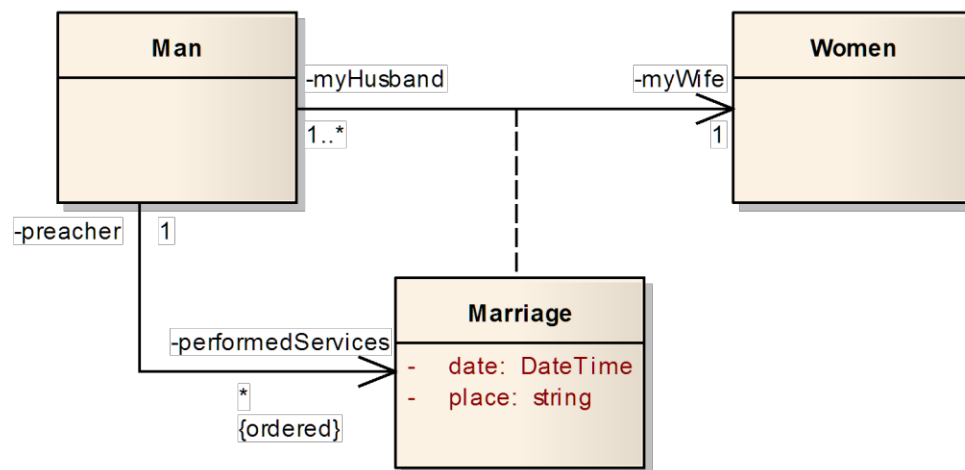


a person may have  
employment with several  
companies



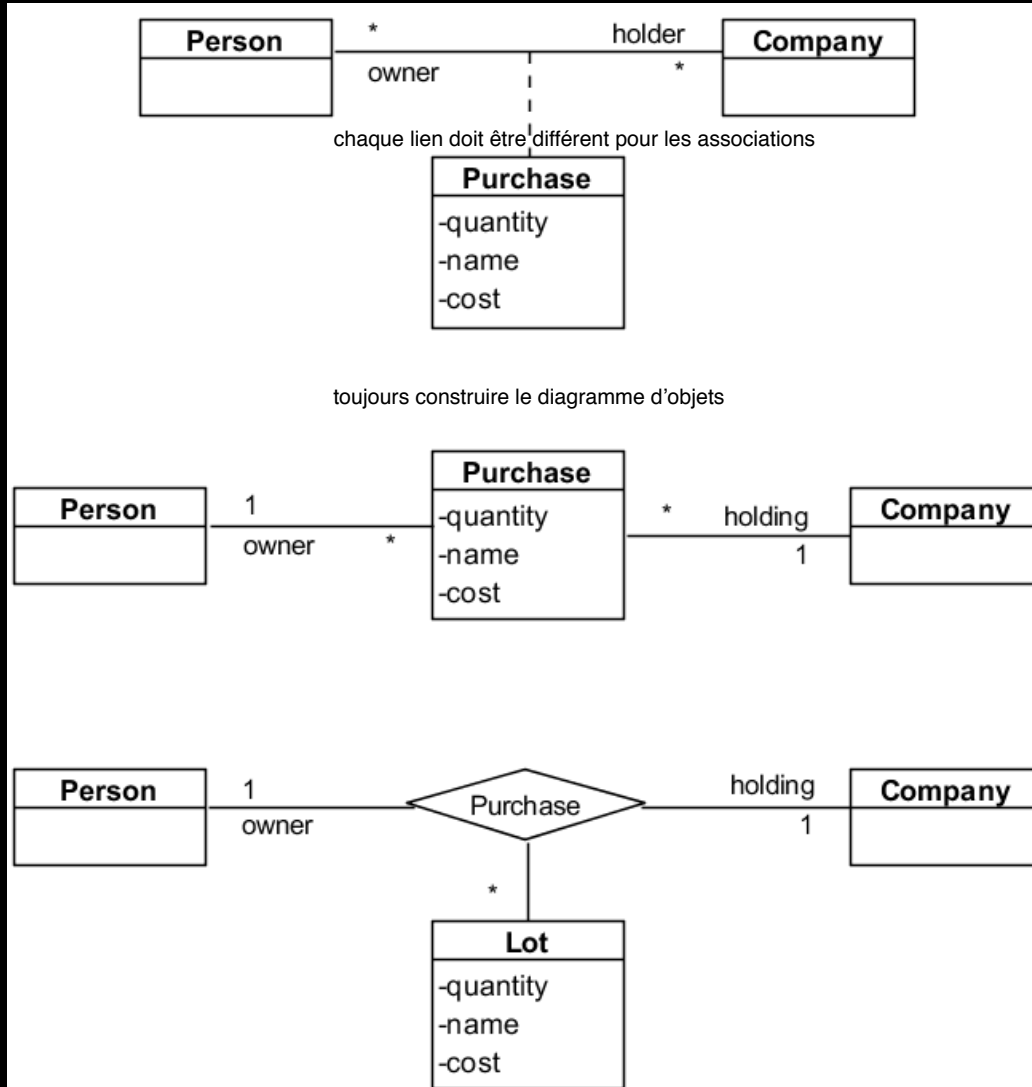
# Classe associative lors de l'exécution

- Les deux classes connaissent la relation et la classe associative
- La relation existe si et seulement si la classe associative existe



# QUESTION

*Quelle est la différence entre ces associations?*



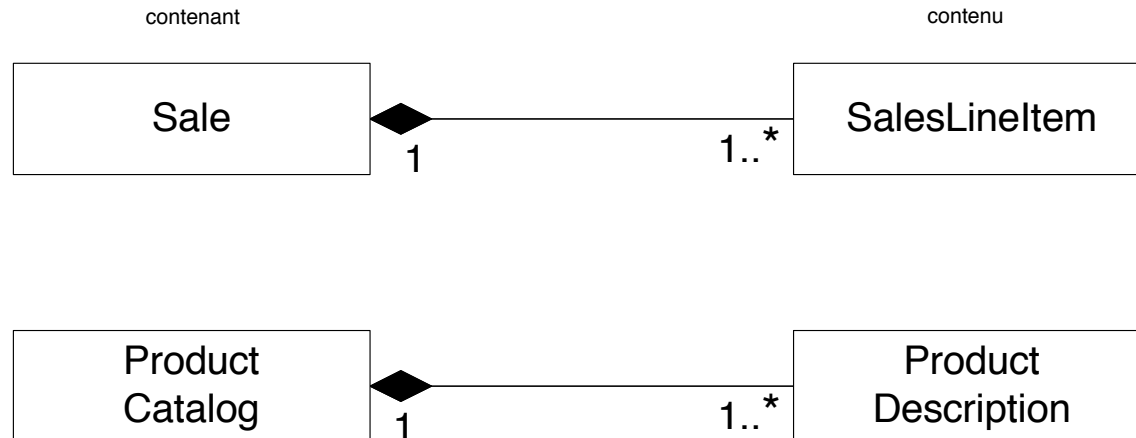
# Composition

- Le cycle de vie de la partie est à l'intérieur de celui du composite
- Dépendance créer-supprimer de la partie dans le composite
- Certaines propriétés du composite se propagent vers la partie
- Certaines opérations du composite se propagent vers la partie
- **Omettre** en cas de doute

conseillé de ne pas utiliser dans le modèle  
si on efface le contenant alors le contenu sera effacé aussi.

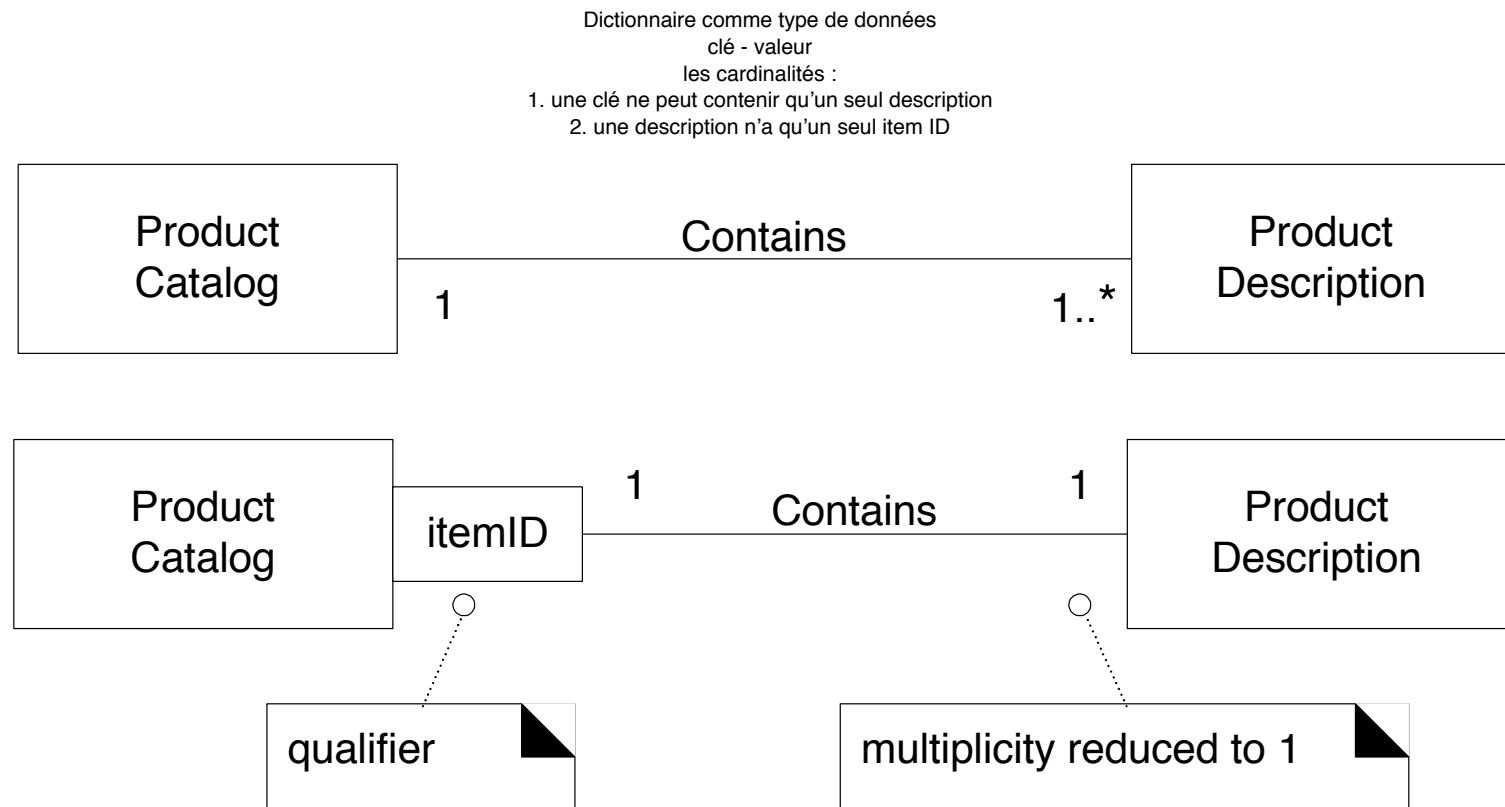
Les compositions par défaut sont 1 1..\*

propagation : sin rabais de 15 % sur le Sale alors elle sera propagée aux items contenus

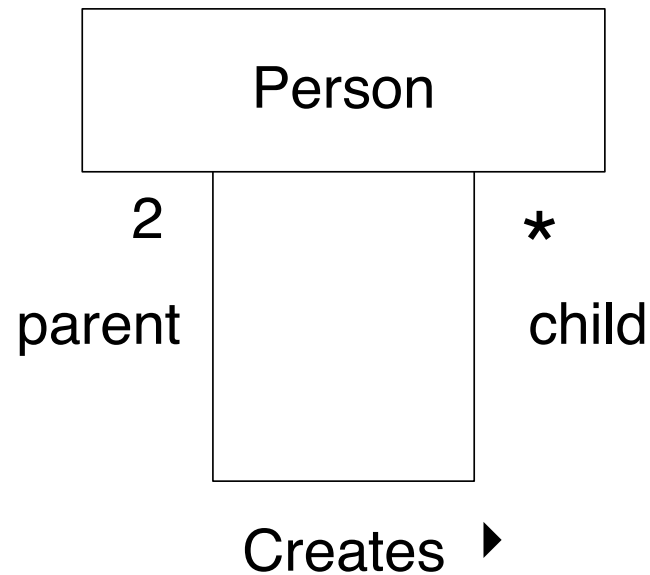


# Association qualifiée

- Qualificateur établit sépare l'ensemble des objets à l'autre bout de l'association par rapport par ses valeurs



# Association réflexive



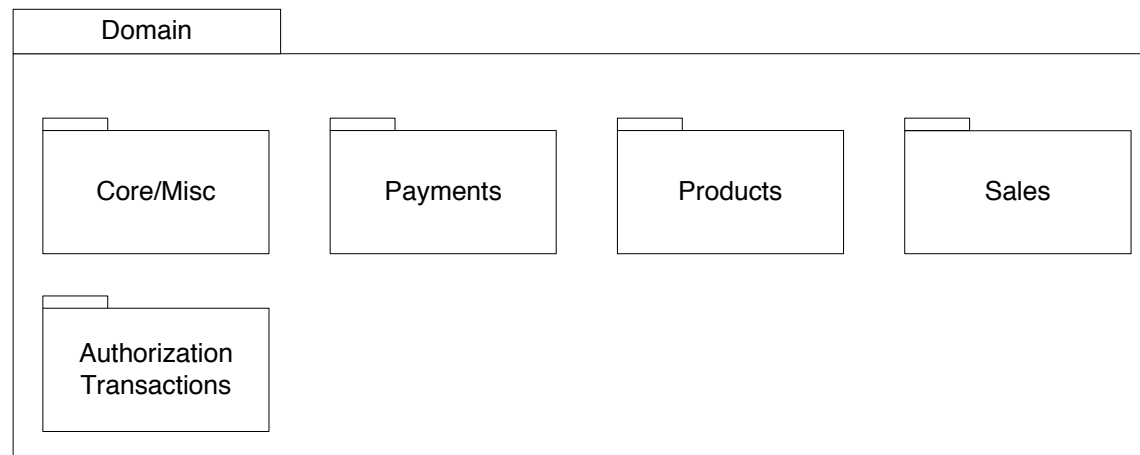
Quels sont les instances possibles de ce modèle ?

# Paquetage

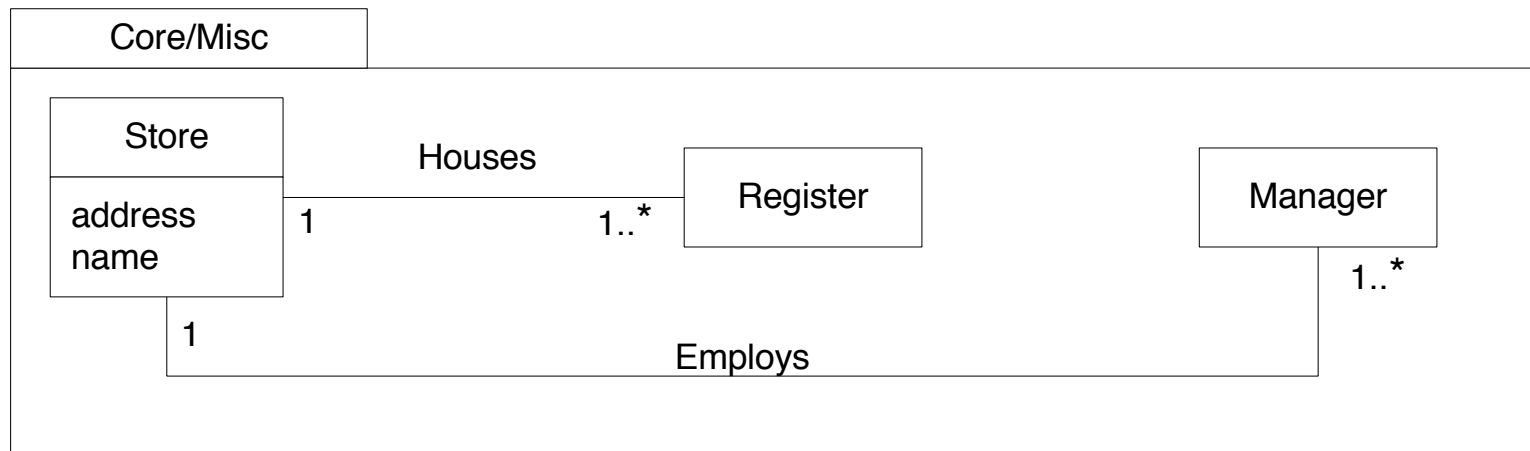


# Paquet (*Package*)

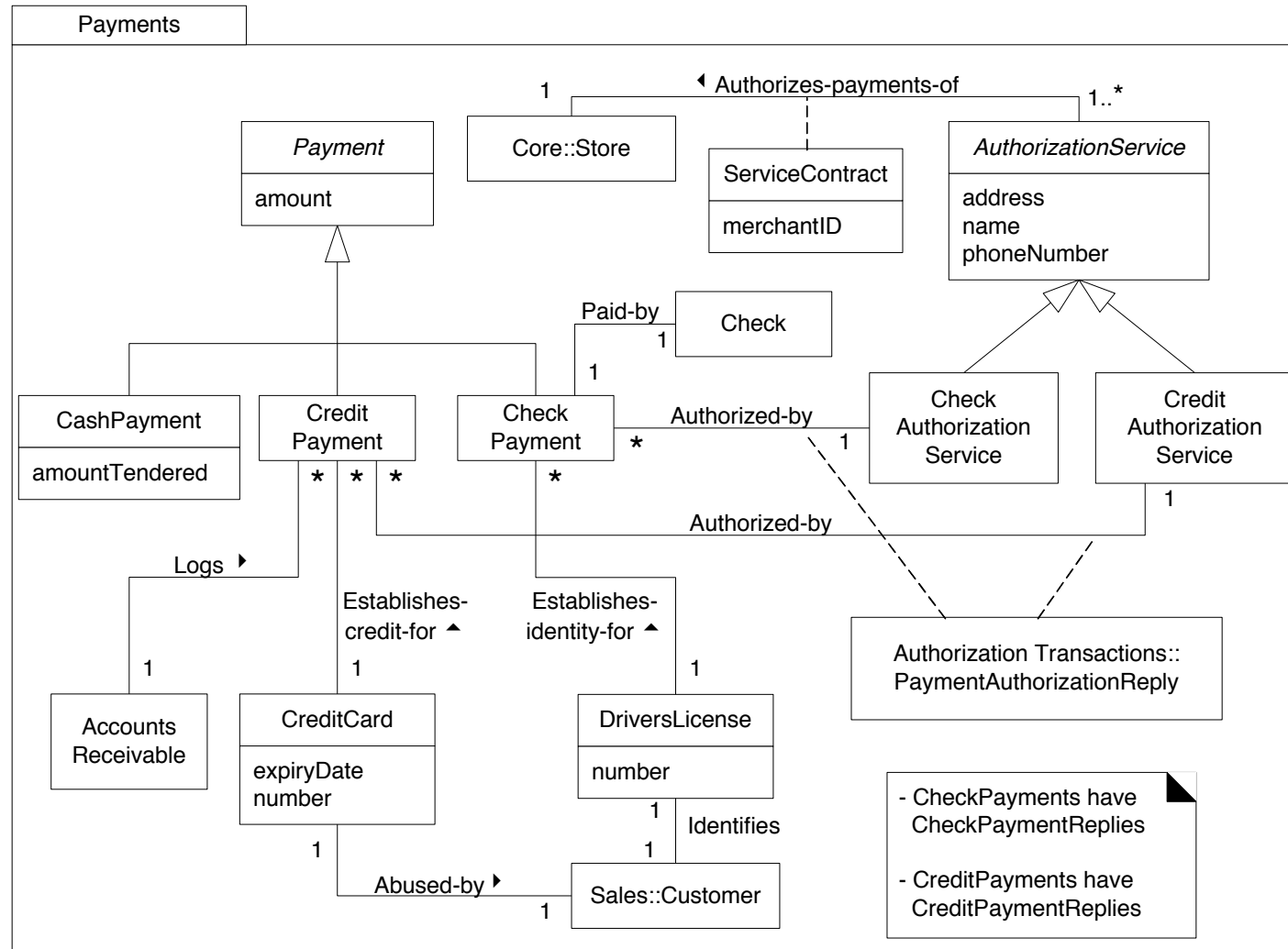
- Éviter l'explosion du modèle du domaine
- Organiser en package les concepts fortement reliés
  - Même sujet est étroitement relié par concept ou but
  - Dans une même hiérarchie de classe
  - Participe dans les mêmes cas d'utilisation
  - Fortement associés



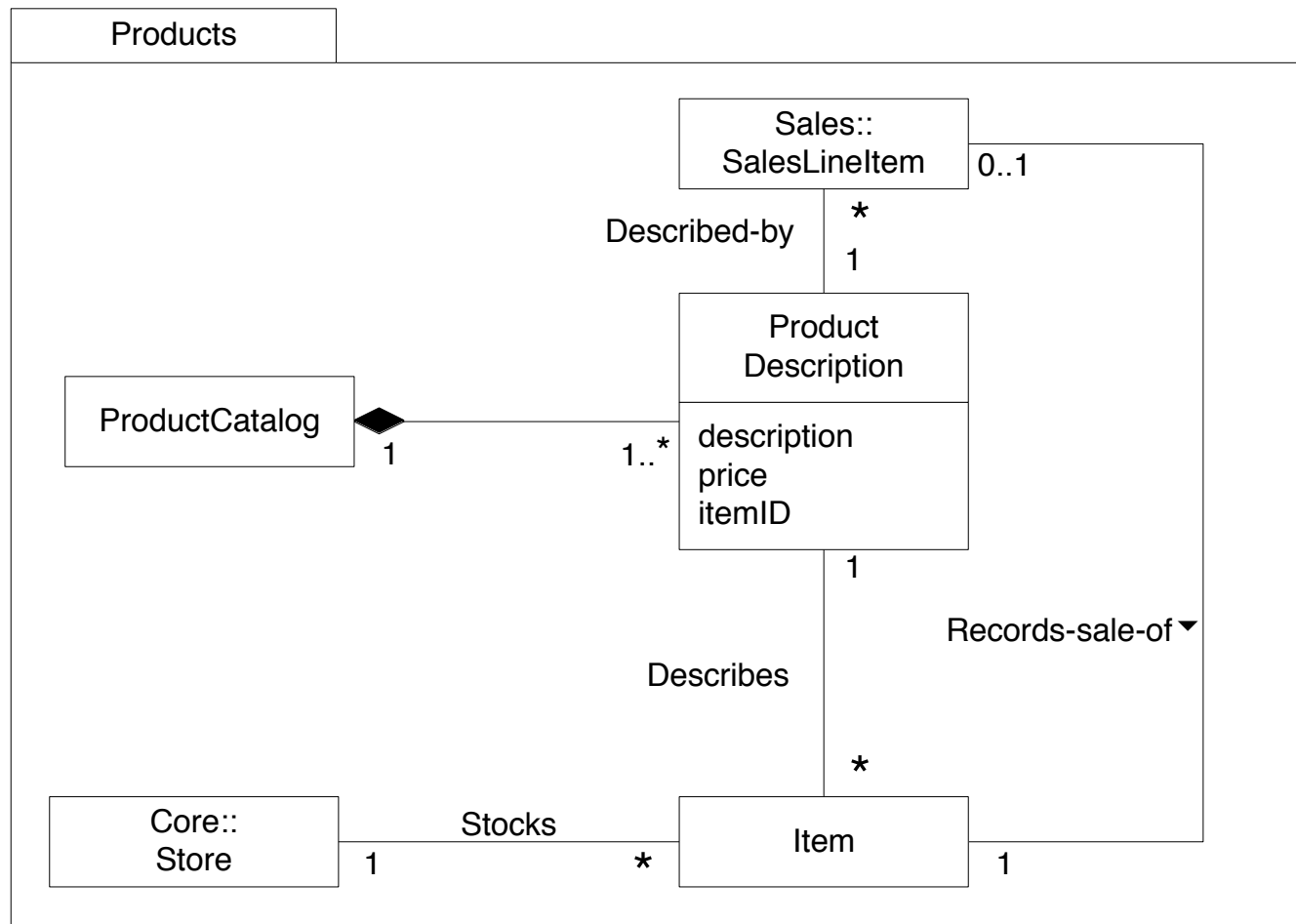
# Paquet : Core



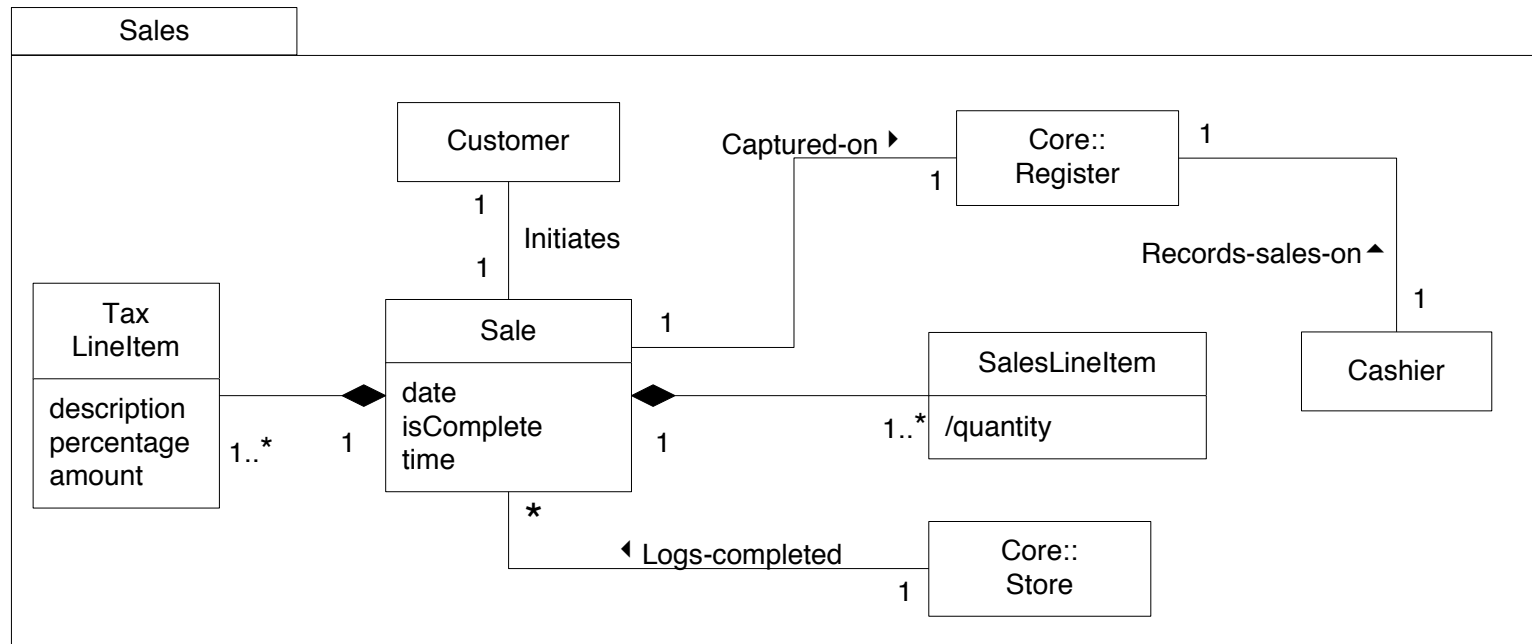
# Paquet : Paiements



# Paquet : Produits



# Paquet : Ventes



# Paquet : Autorisation de Transactions

