GÉNIE LOGICIEL ORIENTÉ OBJET (GLO-2004)
ANALYSE ET CONCEPTION DES SYSTÈMES ORIENTÉS OBJETS (IFT-2007)

Automne 2016

Module 07 - Diagramme de classes de conception

Martin.Savoie@ift.ulaval.ca

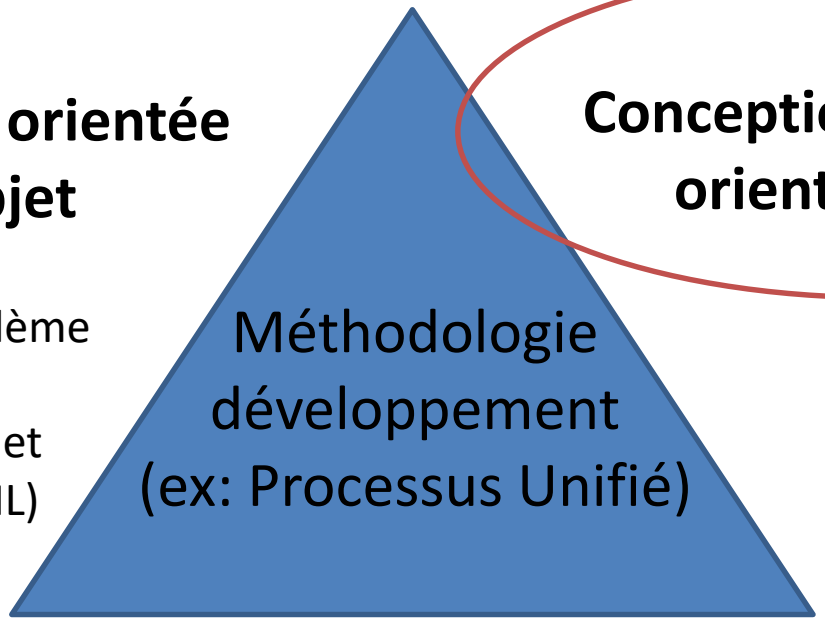
B. ing, Chargé de cours, département d'informatique et de génie logiciel

UNIVERSITÉ LAVAL

Génie logiciel orienté objet

Analyse orientée objet

- Comprendre le problème
- Décrire la situation à l'aide de documents et diagrammes (ex: UML)



Conception (design) orienté objet

- Concevoir une solution informatique
- Tracer des plans (plus ou moins détaillés) sous la forme de documents et diagrammes (ex: UML)

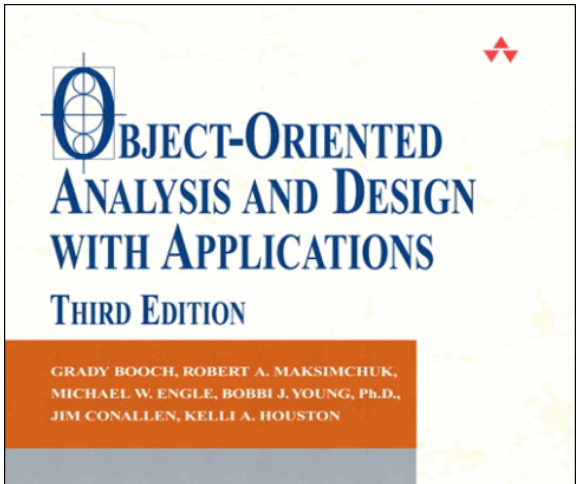
Programmation orientée objet

Mettre en œuvre la solution à l'aide d'un langage (ex: Java)

UNIVERSITÉ LAVAL

Analyse (analysis) vs Conception (design)

- The boundaries between analysis and design are fuzzy.
- In analysis, the focus is to (1) fully analyze the problem at hand, and (2) to model the world by discovering the class/objects that forms the vocabulary of the problem domain.
- In design, (3) we invent the abstractions and mechanisms that provide the design of the solution to be built.



Discipline	Artifact Iteration	Incep. Il	Elab. EL.En	Const. CL.Cn	Trans. T1..T2
Business Modeling	Domain Model		s		
Requirements	Use-Case Model	*s*	r		
	Vision	*s*	r		
	Supplementary Specification	*s*	r		
	Glossary	*s*	r		
Design	Design Model SW Architecture Document Data Model		s s s	r r	
Implementation	Implementation Model (code)		s	r	r

S: start
R: refine

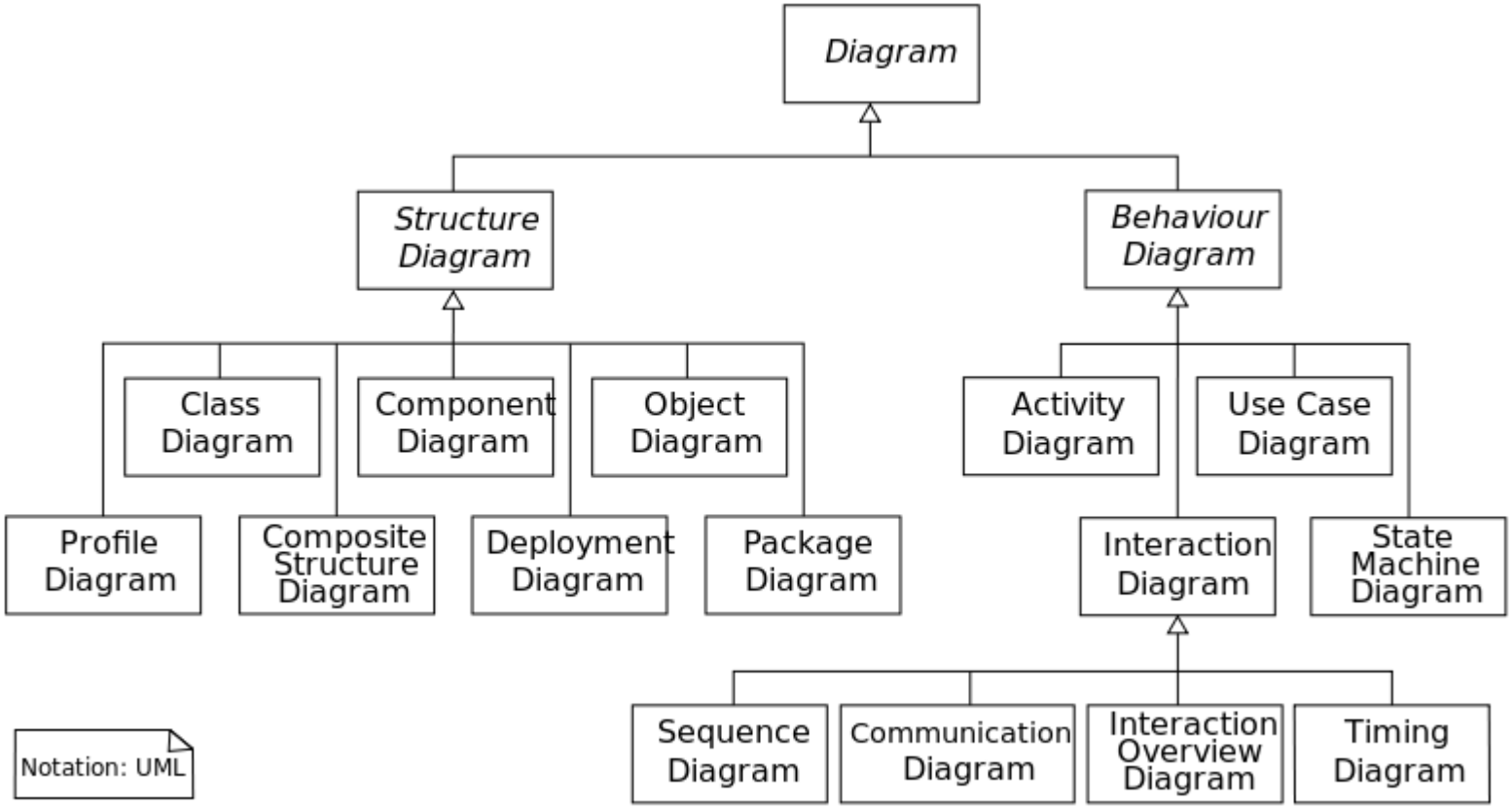
Design orienté-objet

- Identification des classes
- Assignation des responsabilités aux classes
- Organisation et communication entre les classes
- …

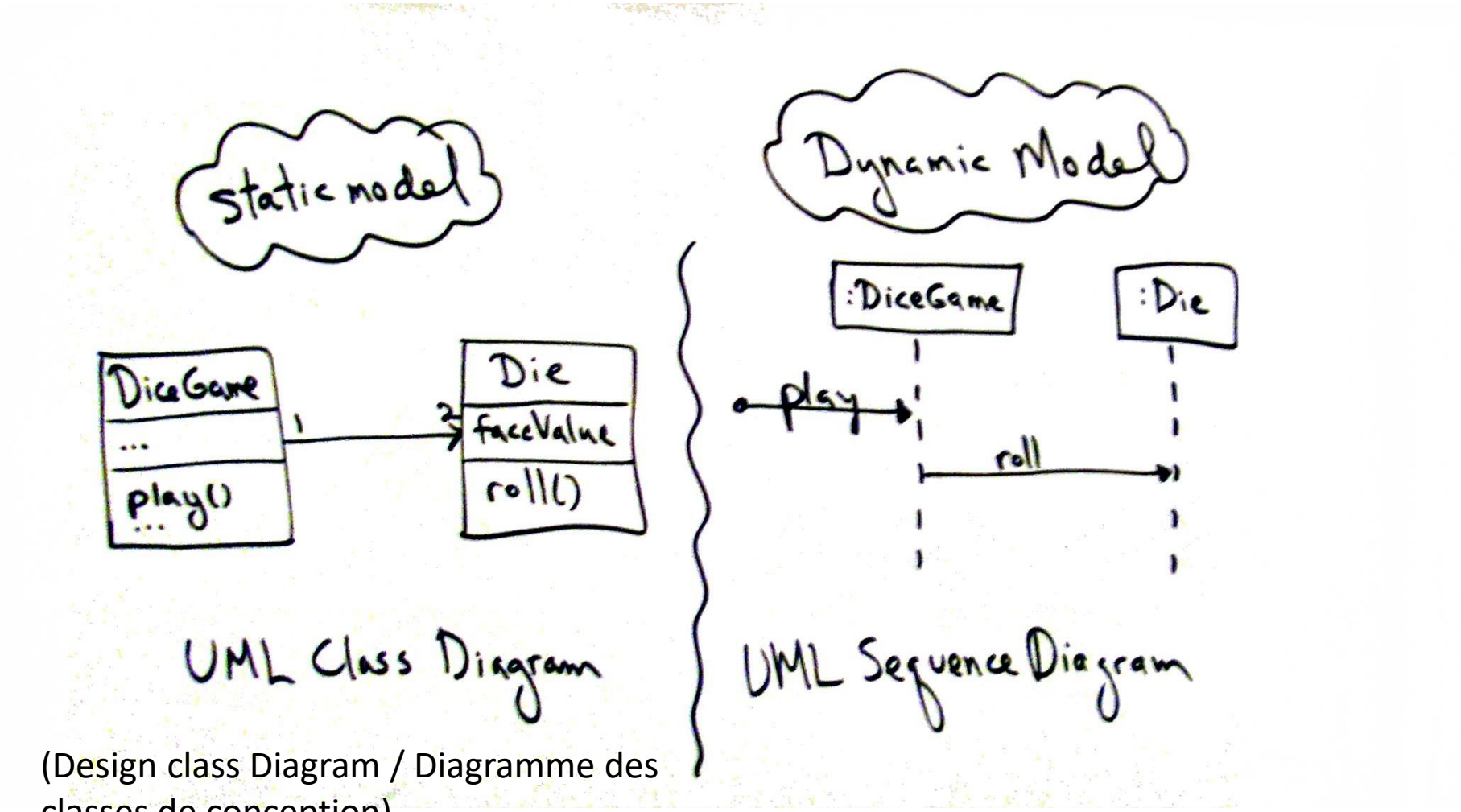
Expertise OO > expertise UML !!

	Activités (appelées *disciplines* dans le Processus Unifié)	**Modèles** et artefacts générés
Analyse	Modélisation domaine d'affaires / Business modeling / Modélisation métier	**Modèle du domaine**: (1) diagramme de classe « conceptuel », (2) parfois un diagramme d'activités
	Analyse des besoins / Exigences / Requirements	(3) Énoncé de vision
		Modèle de cas d'utilisation / Use-case model : (4) diagramme des cas d'utilisation, (5) texte des cas d'utilisation, (6) diagramme de séquence système
		(7) Spécifications supplémentaires
		(8) Glossaire
	Design / Conception	**Modèle de conception / Design model** : (9) diagrammes de classes, (10) diagrammes d'interaction, (11) tout autre diagramme UML pertinent selon le contexte
	Implémentation	(12) Code
	…	

UML : 14 types de diagrammes (ouf!)



Design: vues statiques et dynamique



(Design class Diagram / Diagramme des classes de conception)

Diagramme de classes de conception (DCC) / Design Class Diagram (DCD)

Diagramme de classes de conception

- est un modèle ***statique*** du système
- décrit le système en termes de ses ***classes*** et des ***relations*** entre celles-ci
- sert de ***base*** aux autres diagrammes où d'autres facettes du système sont décrites (tels les diagrammes d'états des objets ou les diagrammes de collaboration qui sont des diagrammes dynamiques)

Diagramme des classes conceptuelles (modèle du domaine) VS diagramme des classes de conception

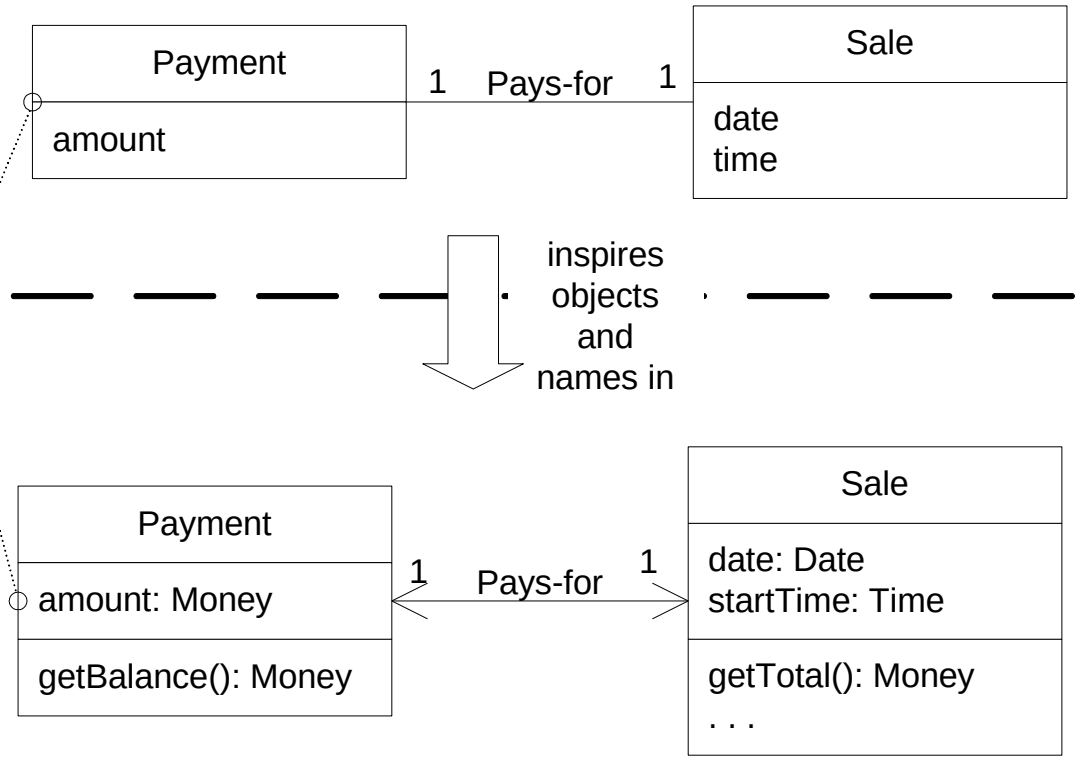
A Payment in the Domain Model is a concept, but a Payment in the Design Model is a software class. They are not the same thing, but the former *inspired* the naming and definition of the latter.

This reduces the representational gap.

This is one of the big ideas in object technology.

UP Domain Model

Stakeholder's view of the noteworthy concepts in the domain.



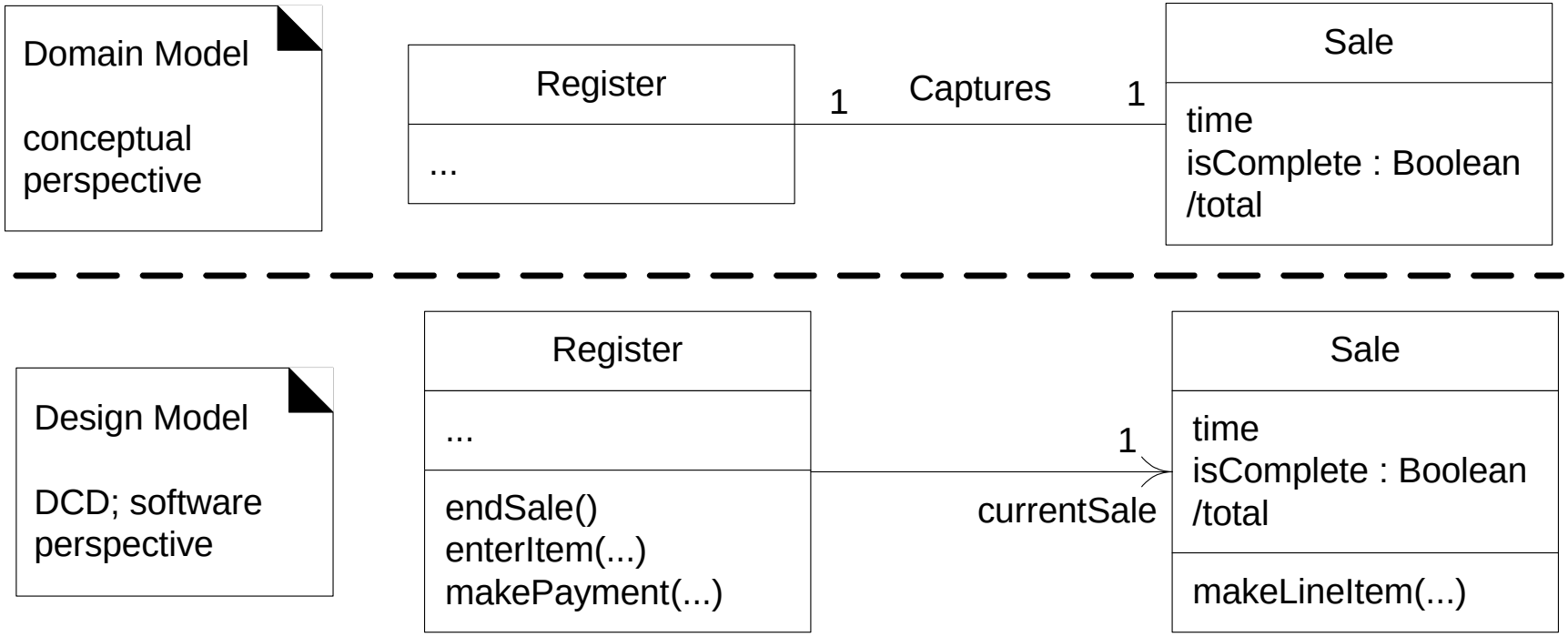
… et plusieurs classes vont disparaître, d'autres vont apparaître

UP Design Model

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

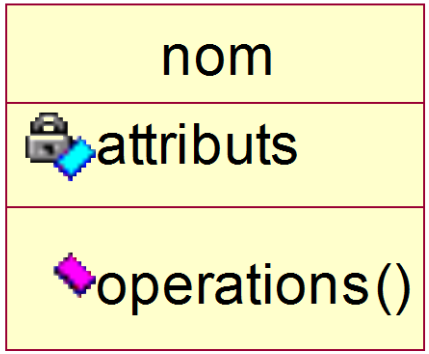
Therefore, the representational gap between how stakeholders conceive the domain, and its representation in software, has been lowered.

NexGen POS – Classes conservées (mais transformées)

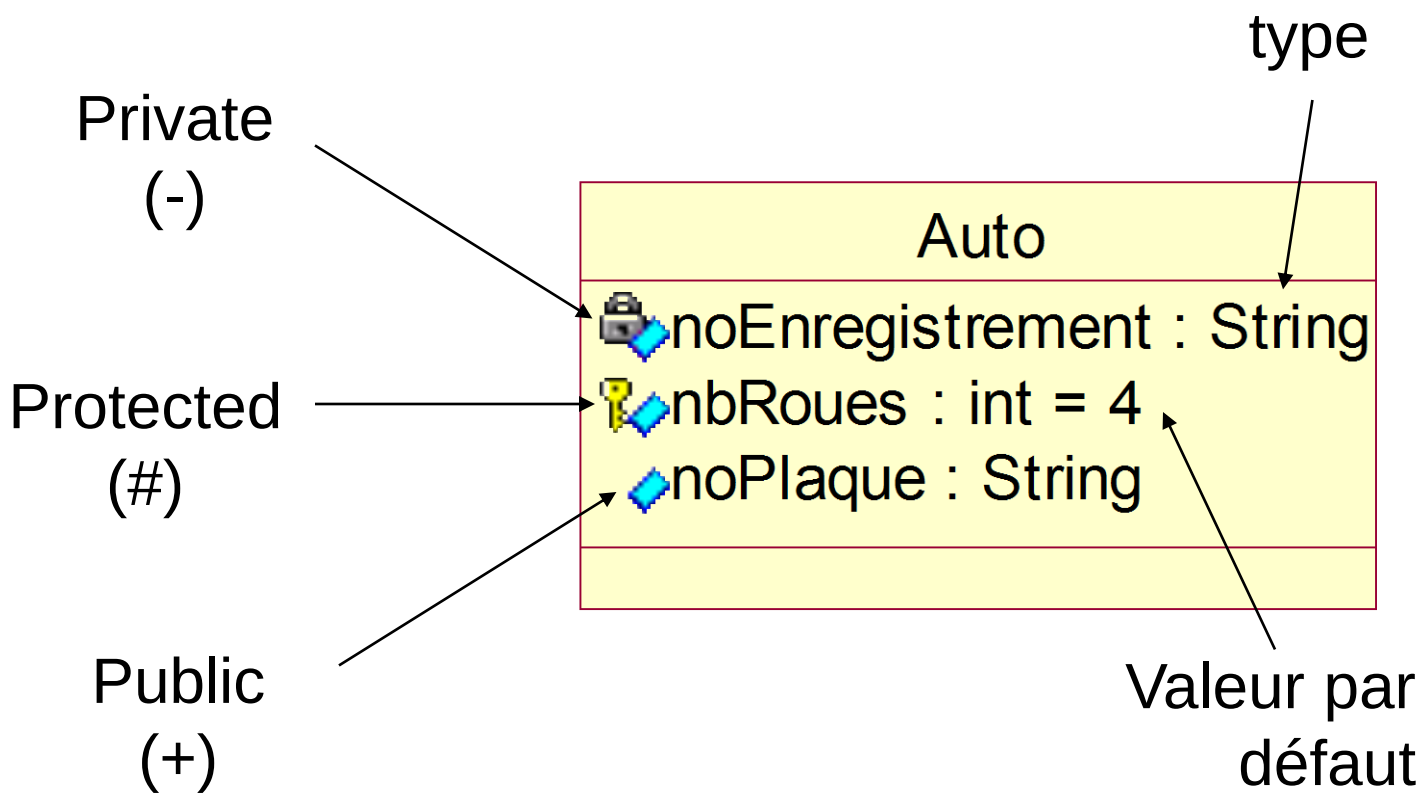


Représentation des classes en UML

- une ***classe*** est représentée par un ***rectangle*** divisé en ***trois compartiments***:
  - le compartiment du ***nom***
  - le compartiment des ***attributs***
  - le compartiment des ***opérations***
- la syntaxe dans les différents compartiments est ***indépendante*** des langages de programmation



Exemples de classe avec attributs

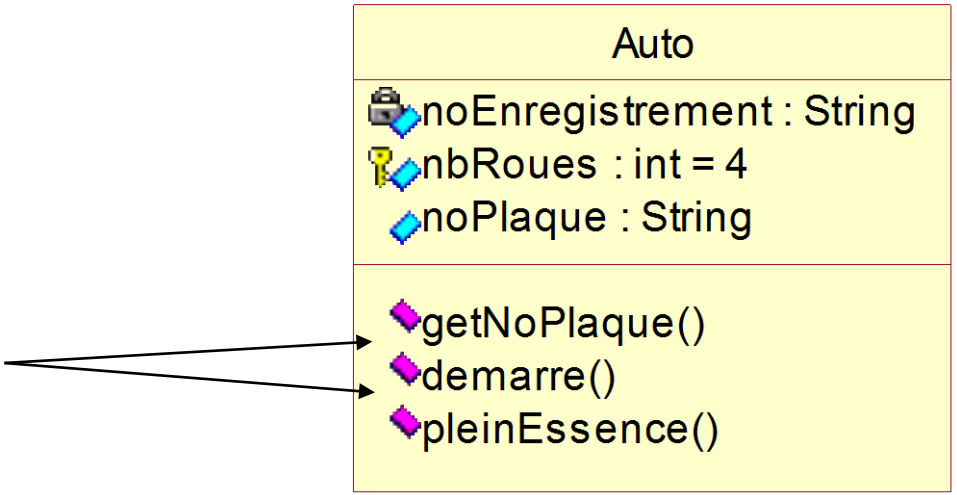


Compartiment des attributs

- Les ***attributs*** servent à décrire les ***caractéristiques*** des objets appartenant à la classe (i.e. décrivent l'***état*** de l'objet)
- chaque attribut possède:
  - un ***type*** (primitif ou défini par l 'usager)
  - un ***niveau de visibilité*** (***public*** (+), ***protected*** (#) ou ***private*** (-))
  - une ***valeur par défaut***
  - …et une ***chaîne de propriétés*** indiquant les valeurs que peut prendre l 'attribut

Exemple de classe avec opérations

Méthodes publiques



Compartiment des opérations/méthodes

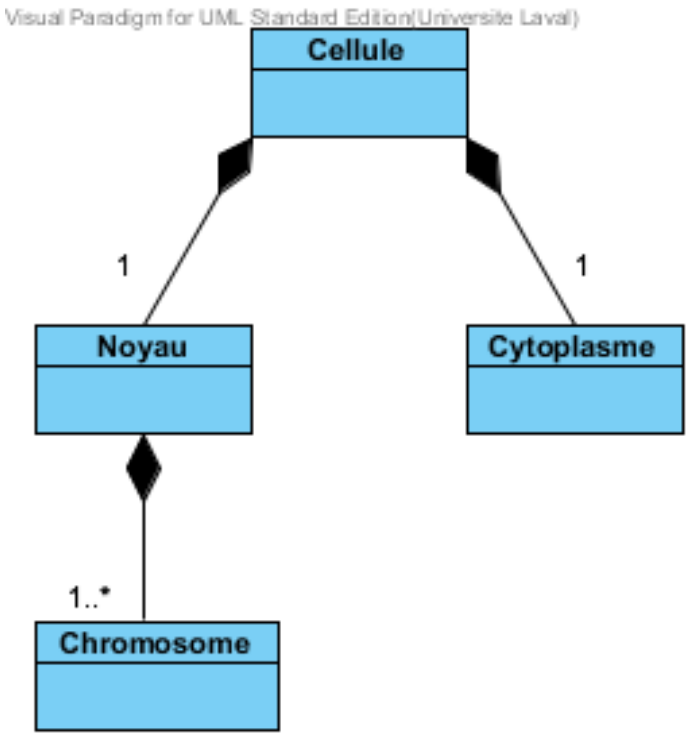
- Les ***opérations*** permettent de ***manipuler*** les attributs et d'effectuer d'autres actions. Elles sont appelées pour des instances (objets) de la classe.
- La ***signature*** des opérations comprend:
  - un ***type*** de retour
  - un ***nom***
  - 0 ou plus ***paramètres*** (ayant chacun un type, un nom et possiblement une valeur par défaut)
  - une ***visibilité*** (private(-), protected(#), public(+))
- Les opérations constituent l'***interface*** de la classe avec le monde extérieur

Relations entre les classes

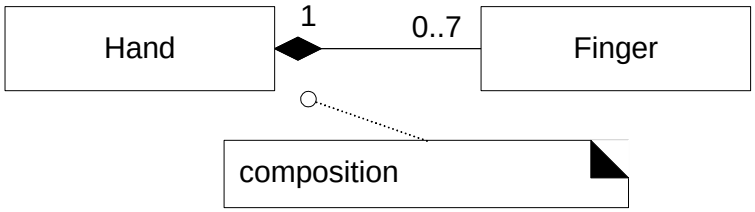
- Un diagramme de classes montre les classes d'un système mais également les ***relations*** entre celles-ci:
- On compte ***quatre*** principales catégories de relations entre classes:
  - les ***associations***
  - les ***généralisations*** (héritage)
  - les ***dépendances***
  - les ***raffinements***

Composition («est constitué de »)

- Indique une relation ***tout-partie avec inclusion « physique »***
- se représente par une ligne avec un ***losange plein***
- Elle possède des ***rôles***, ***multiplicité***, etc, comme une association normale

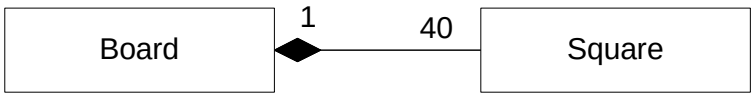


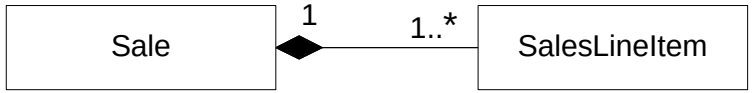
Composition



composition means
-a part instance (*Square*) can only be part of one composite (*Board*) at a time

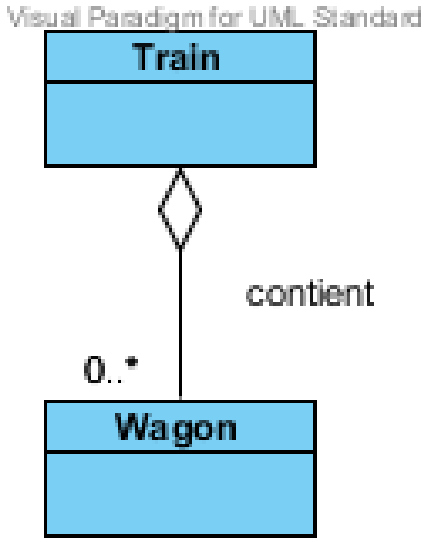
-the composite has sole responsibility for management of its parts, especially creation and deletion



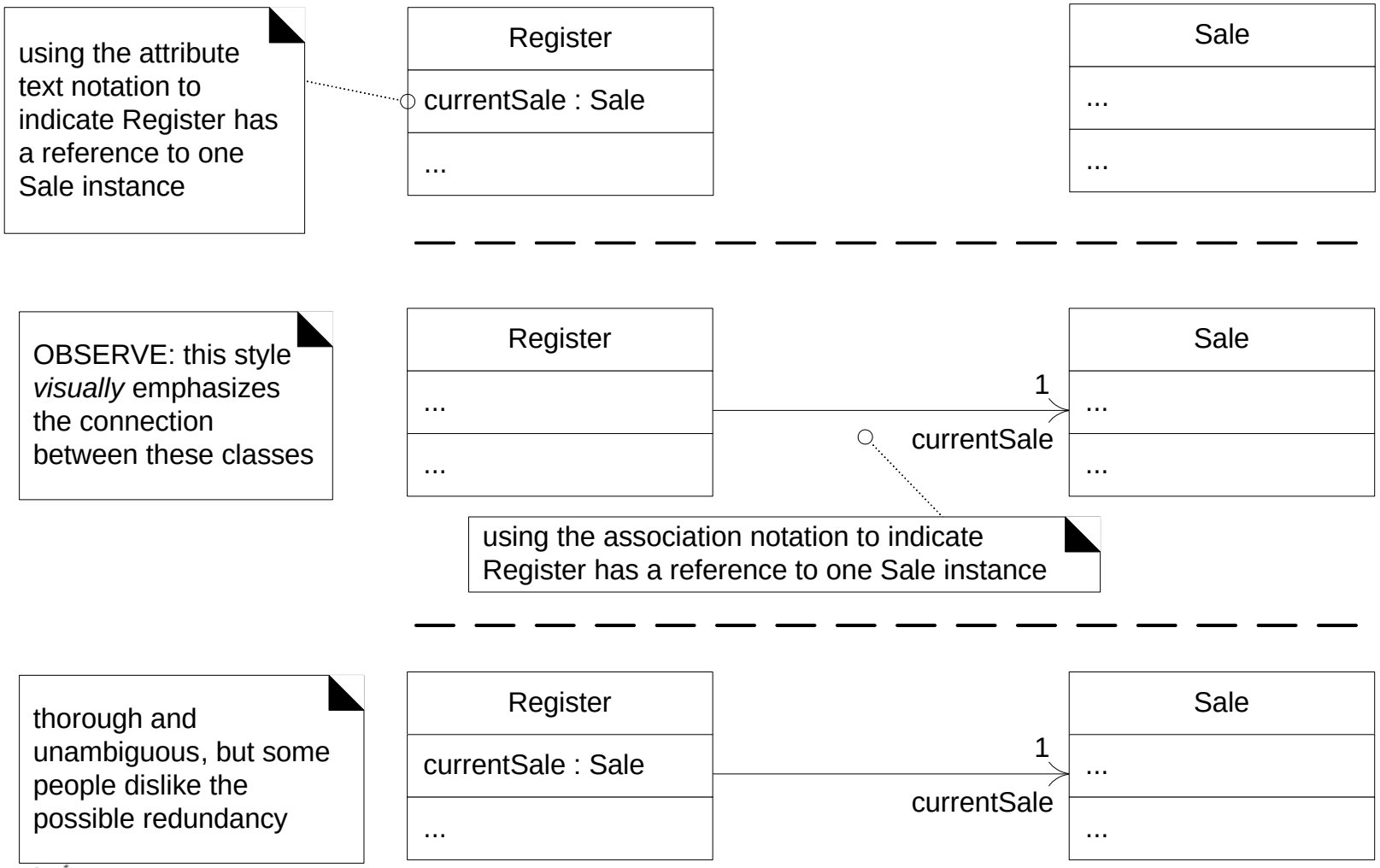


Agrégation («a un»)

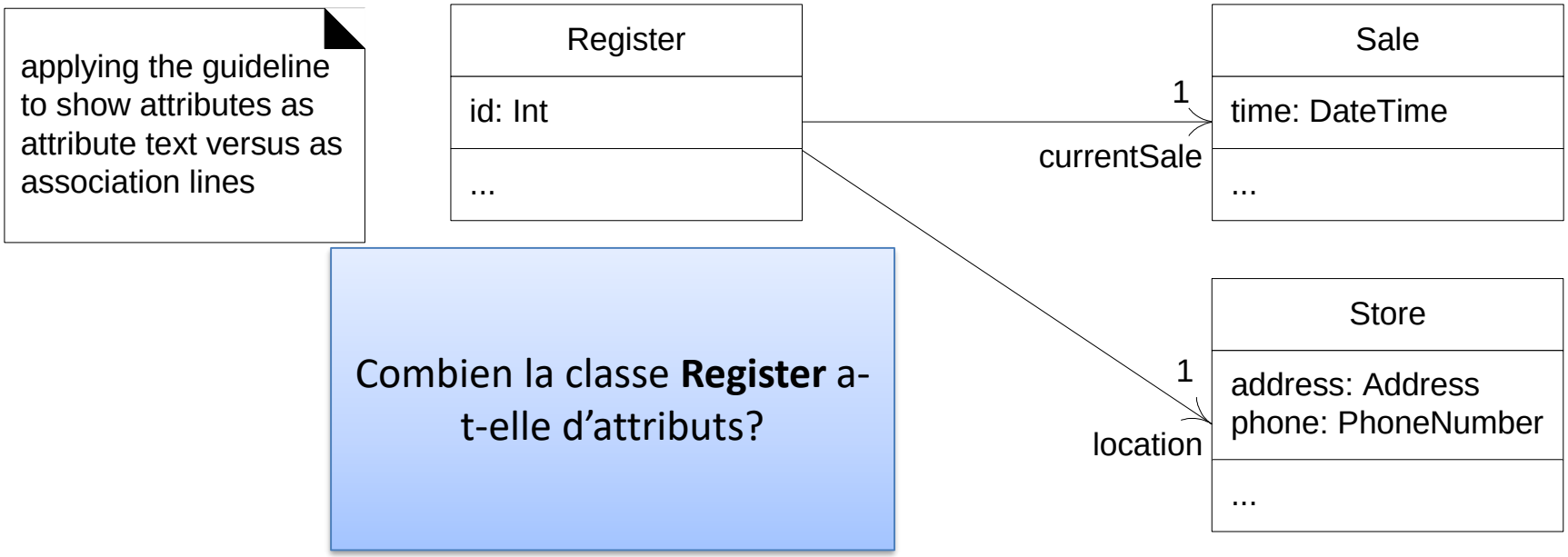
- Indique une relation ***tout-partie*** (moins forte que la composition)
- Elle possède des ***rôles***, ***multiplicité***, etc, comme une association normale
- Se représente par une ligne avec un losange vide
- Techniquement, n'indique rien de plus qu'une relation ordinaire sous UML.
- En pratique, très utile pour la communication.



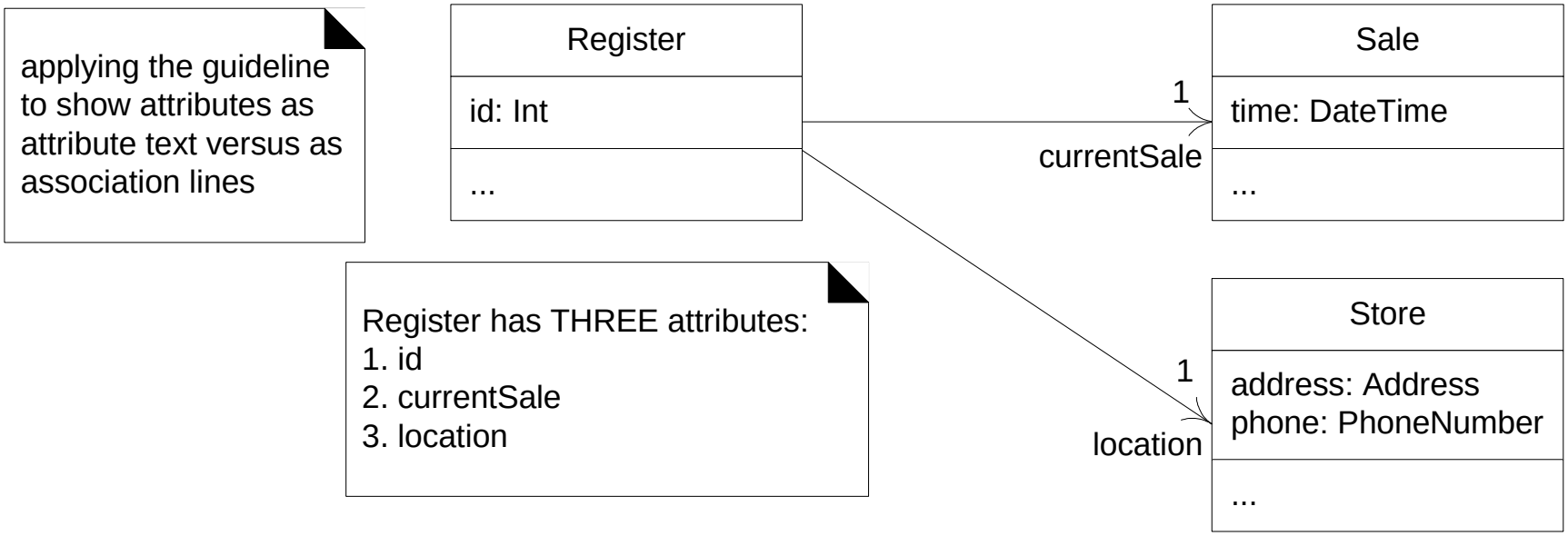
Représentation des attributs



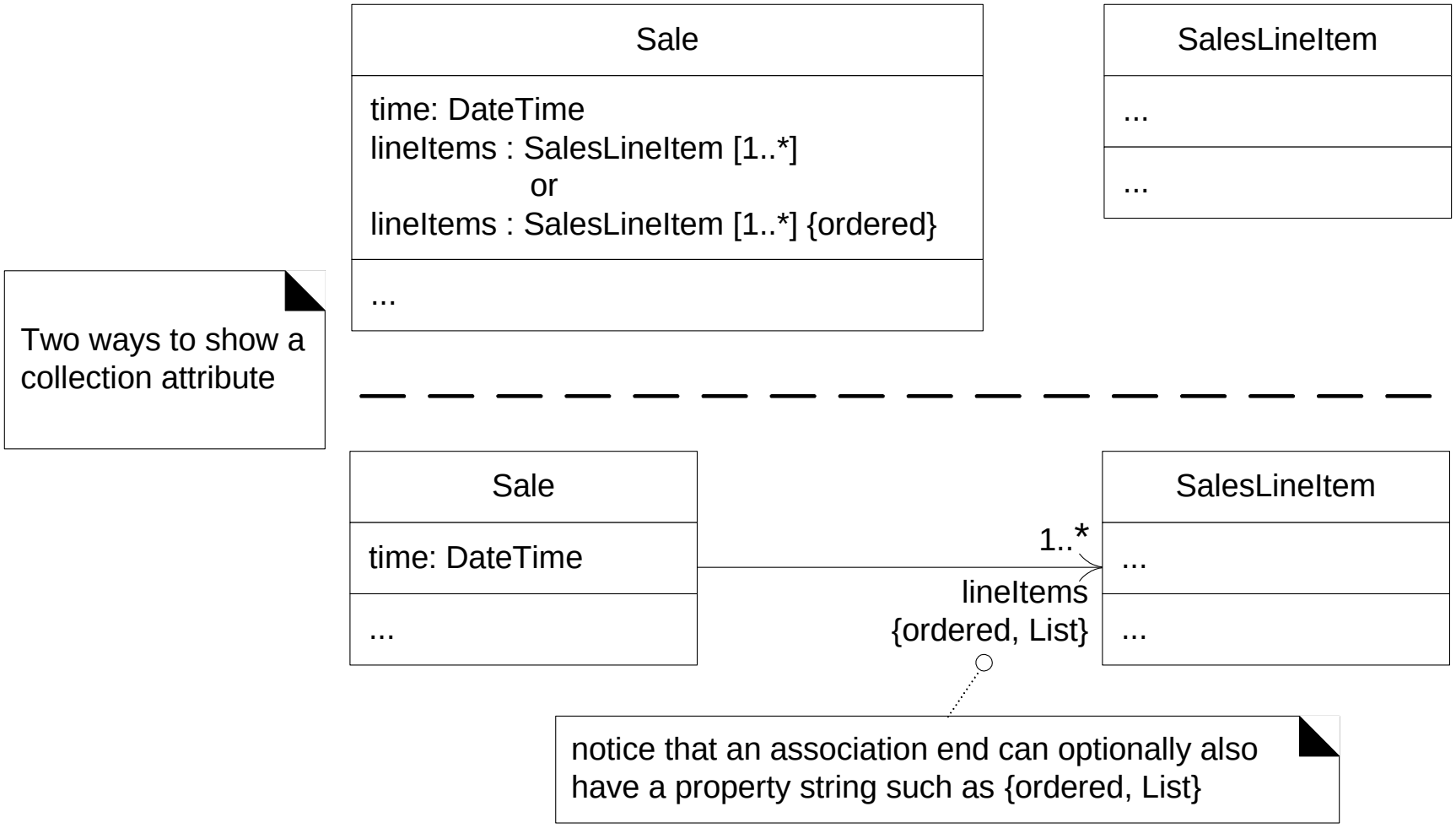
Représentation des attributs



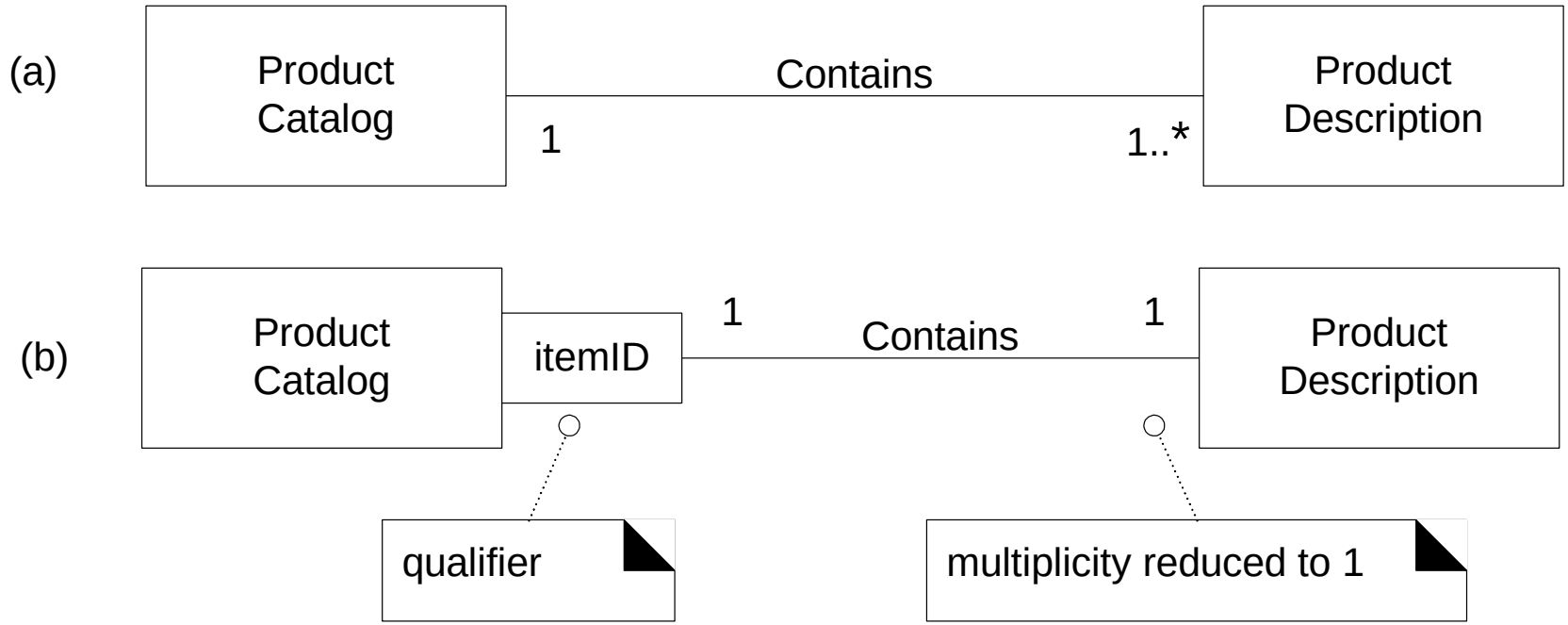
Représentation des attributs



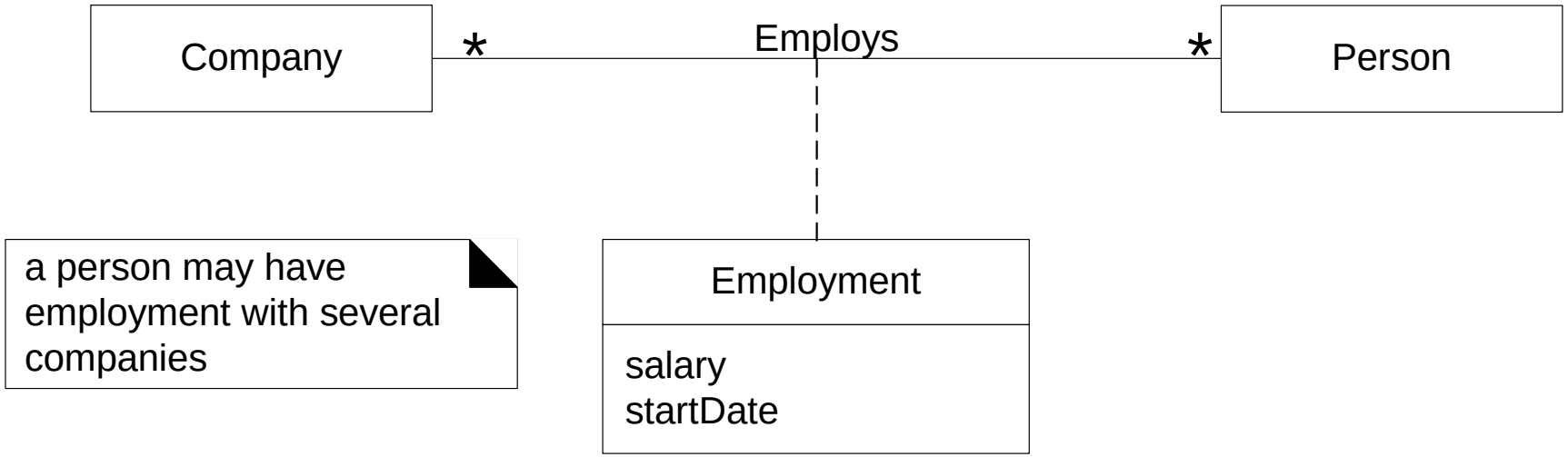
Listes / Vecteurs



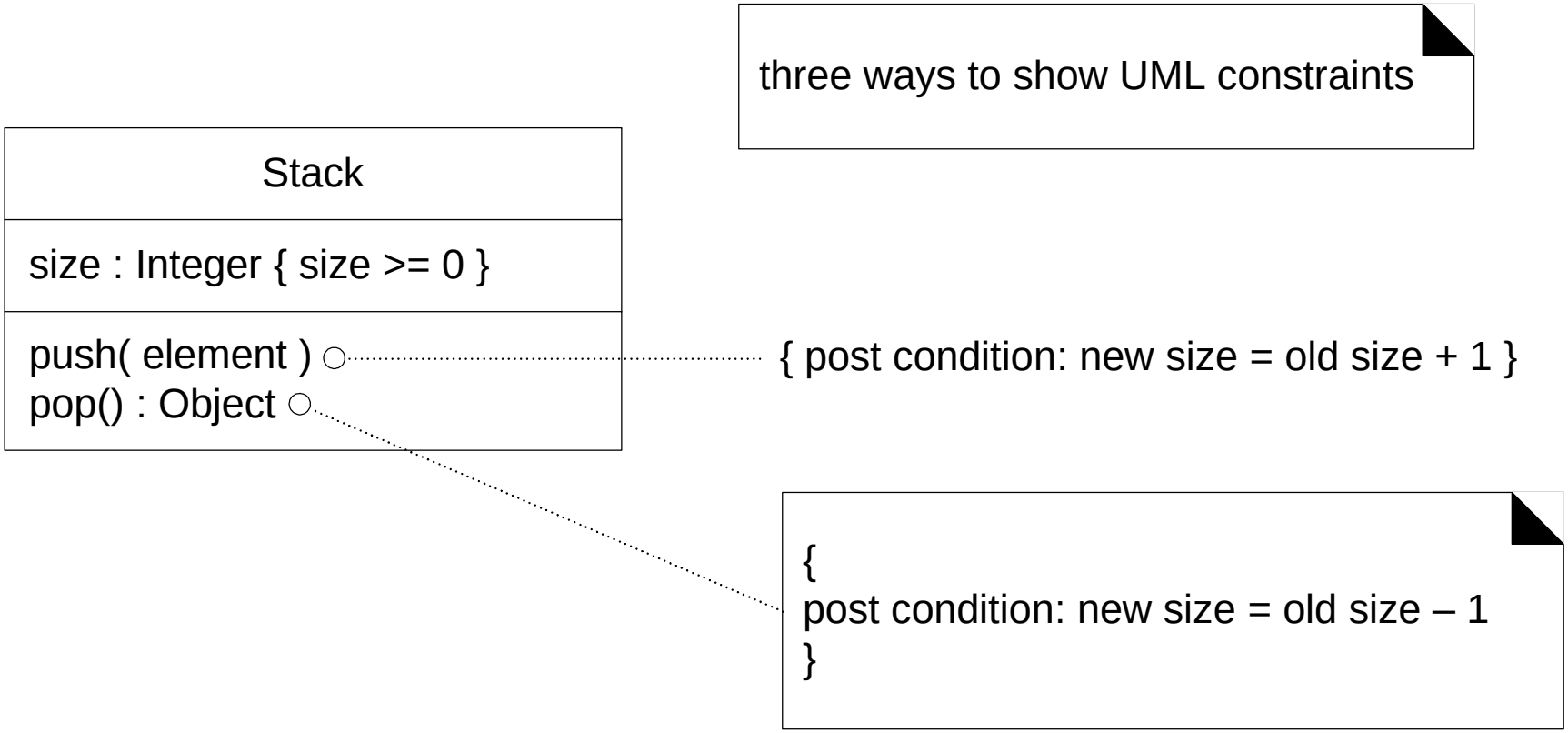
Associations qualifiées (qualificateur)



Classes d'association



Contraintes

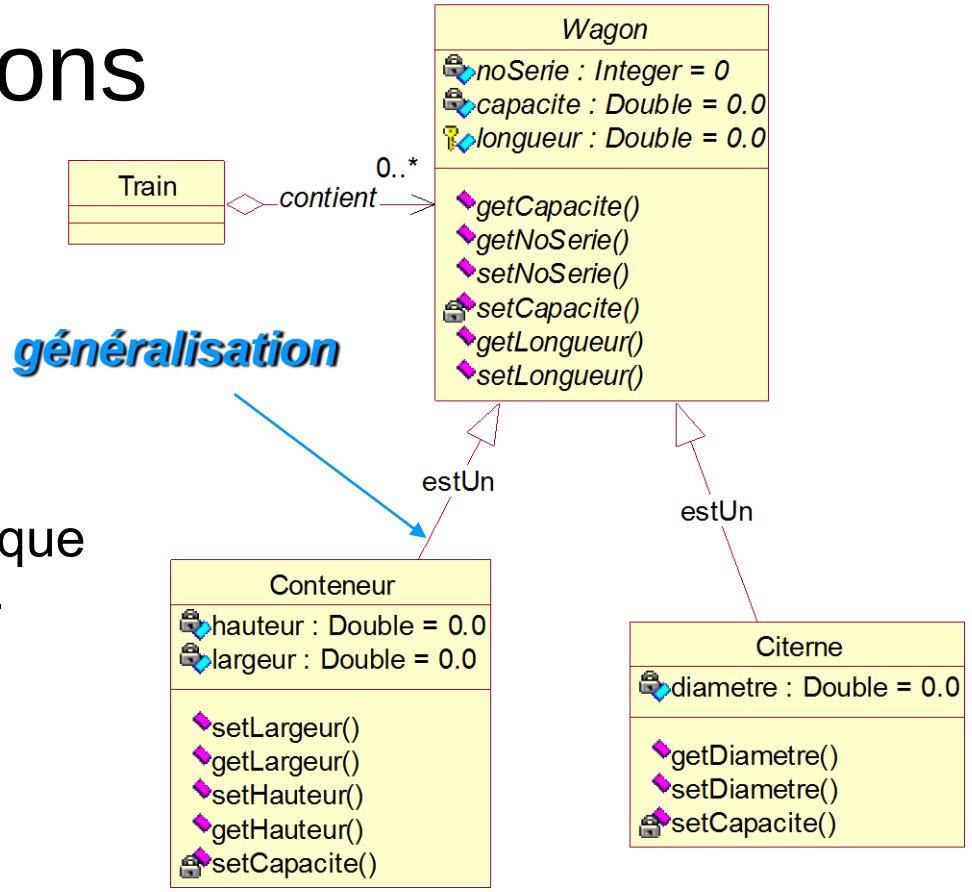


Les généralisations (héritage)

- Une généralisation est une ***relation*** entre une classe ***générale*** et une classe plus ***spécifique***
- la classe ***spécifique*** est appelée ***sous-classe***
- la classe ***générale*** est appelée ***super-classe***
- la sous-classe ***hérite*** des ***attributs*** et ***opérations*** de la super-classe (en fonction de la visibilité de la généralisation)
- une ***sous-classe*** peut être la ***super-classe*** d'une autre classe dans ce que l 'on appelle une ***hiérarchie***

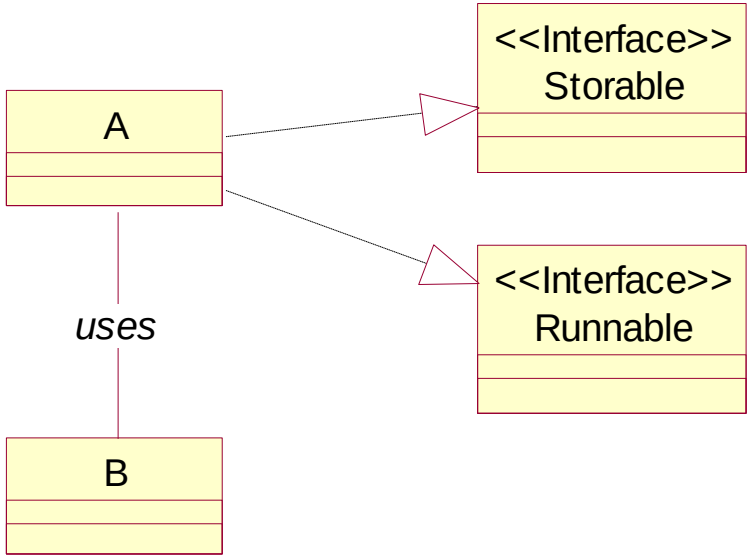
Représentation des généralisations

- On représente les ***généralisations*** par une ***flèche*** allant de la classe spécifique à la classe générale
- Lorsqu'une classe spécifique hérite de plusieurs super-classes, on dit qu'il y a ***héritage multiple***

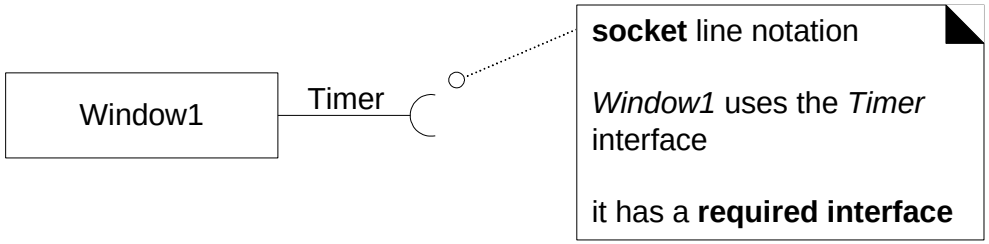


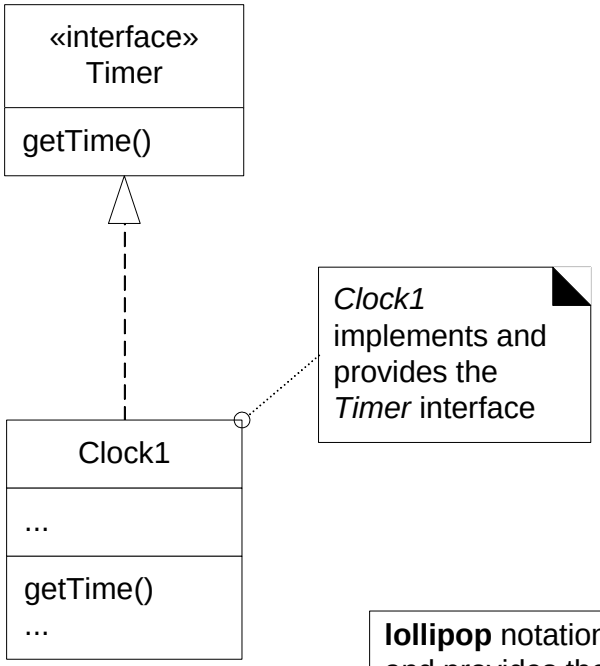
Interfaces

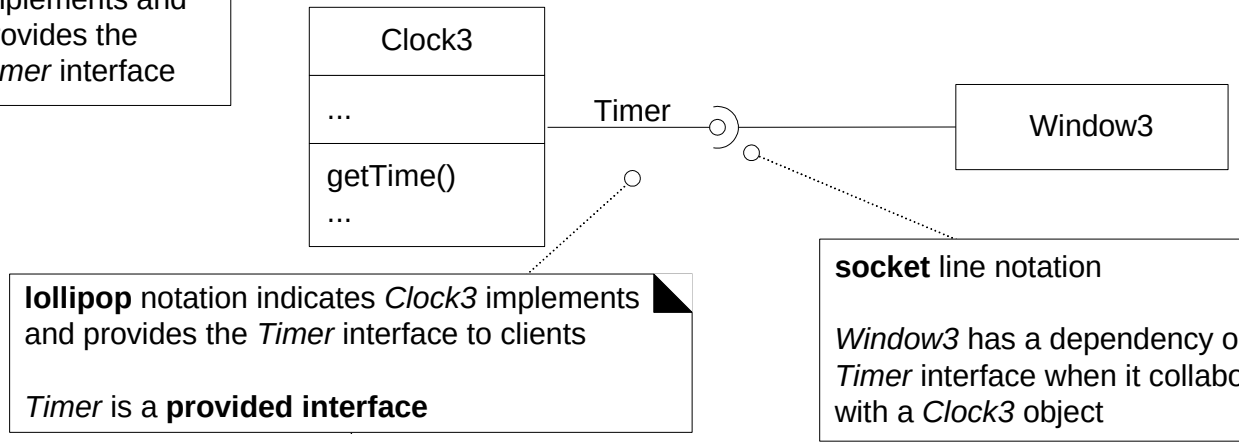
- Les ***interfaces*** sont des classes contenant seulement des ***opérations*** sans ***implémentation***.
- Une classe peut ***implémenter*** une interface
- Le symbole d'implantation est une ***flèche pointillée*** allant de la classe ***vers*** l'interface
- Ici, la classe A implante les interfaces Storable et Runnable et B utilise ces implantations



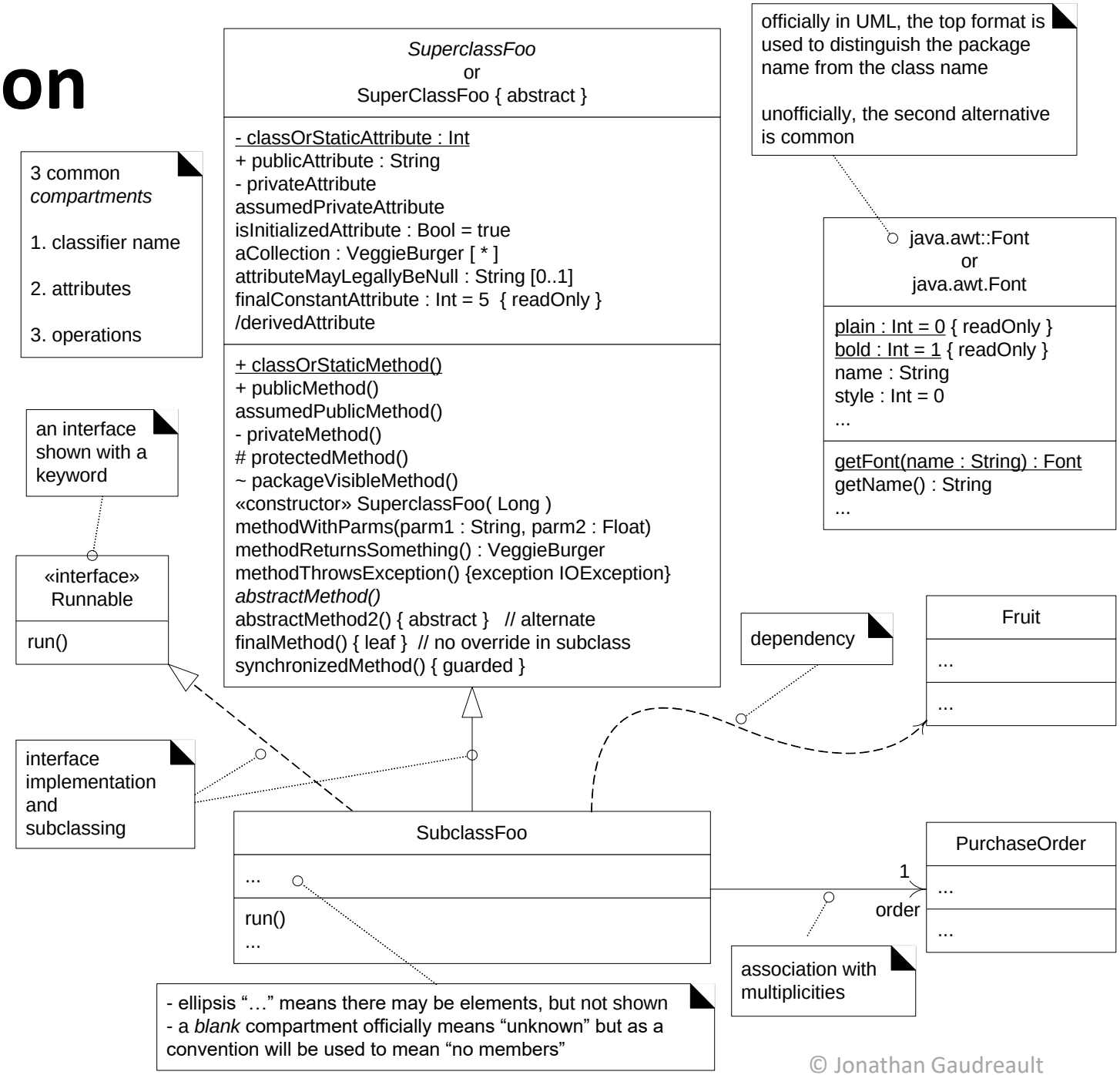
Interfaces







Notation



À faire cette semaine

- Lecture des chapitres
  - Version anglaise: 12,16,13,34,35
  - Version française: 15,12,28,29

- Comprendre la relation entre les concepts suivants:
  - Analyse; Design
  - Diagramme des classes; Diagramme des classes conceptuelles (modèle du domaine); Diagramme des classes de conception
  - Architecture logique / architecture logicielle
  - Architecture en couche;
  - Architecture en couche classique
  - Modèle-Vue
  - Diagramme de package