SWEN30006
Software Modelling and Design

Domain Models

Larman Chapter 9

It's all well in practice, but it will never work in theory.

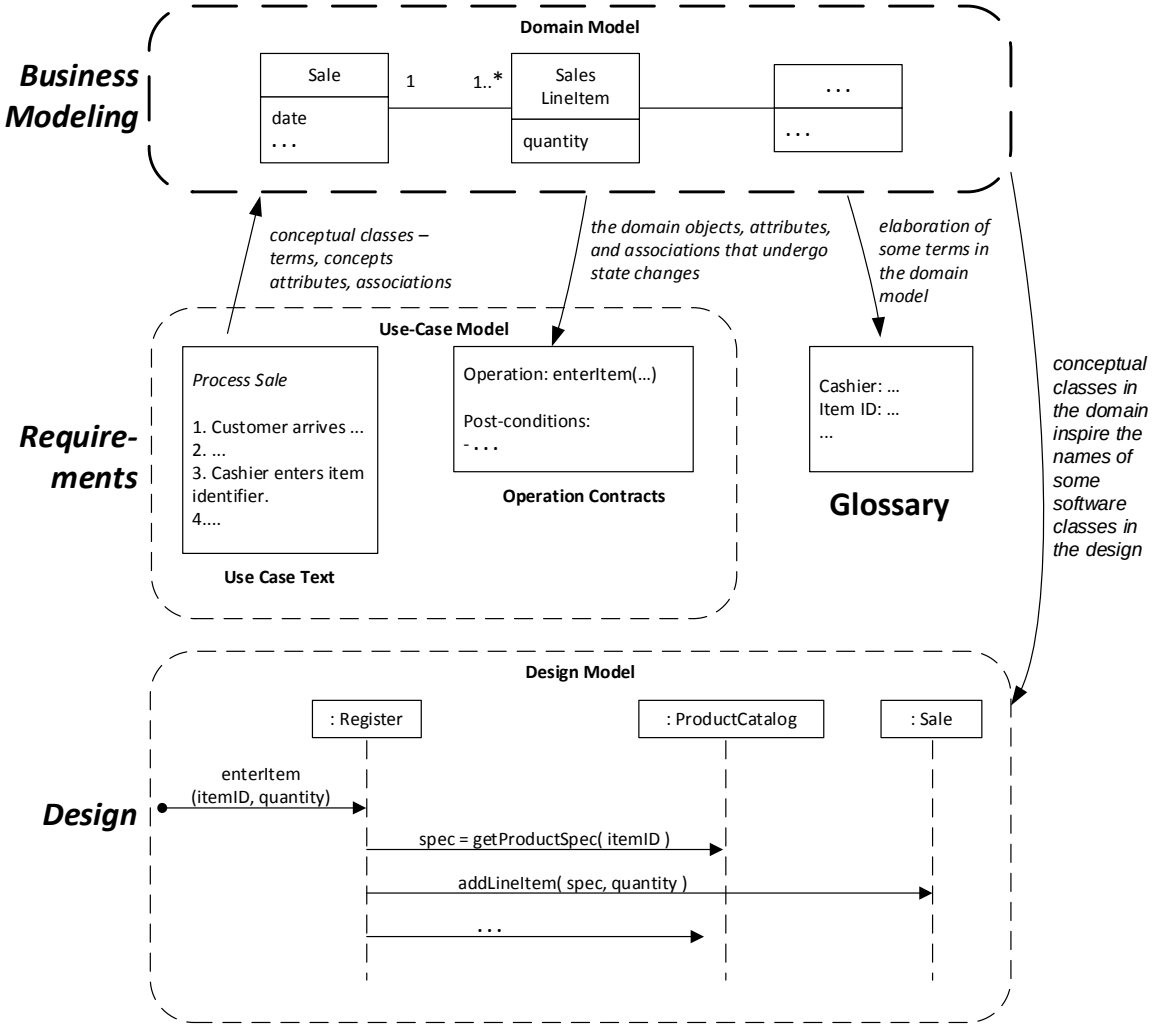
—anonymous management maxim

Objectives

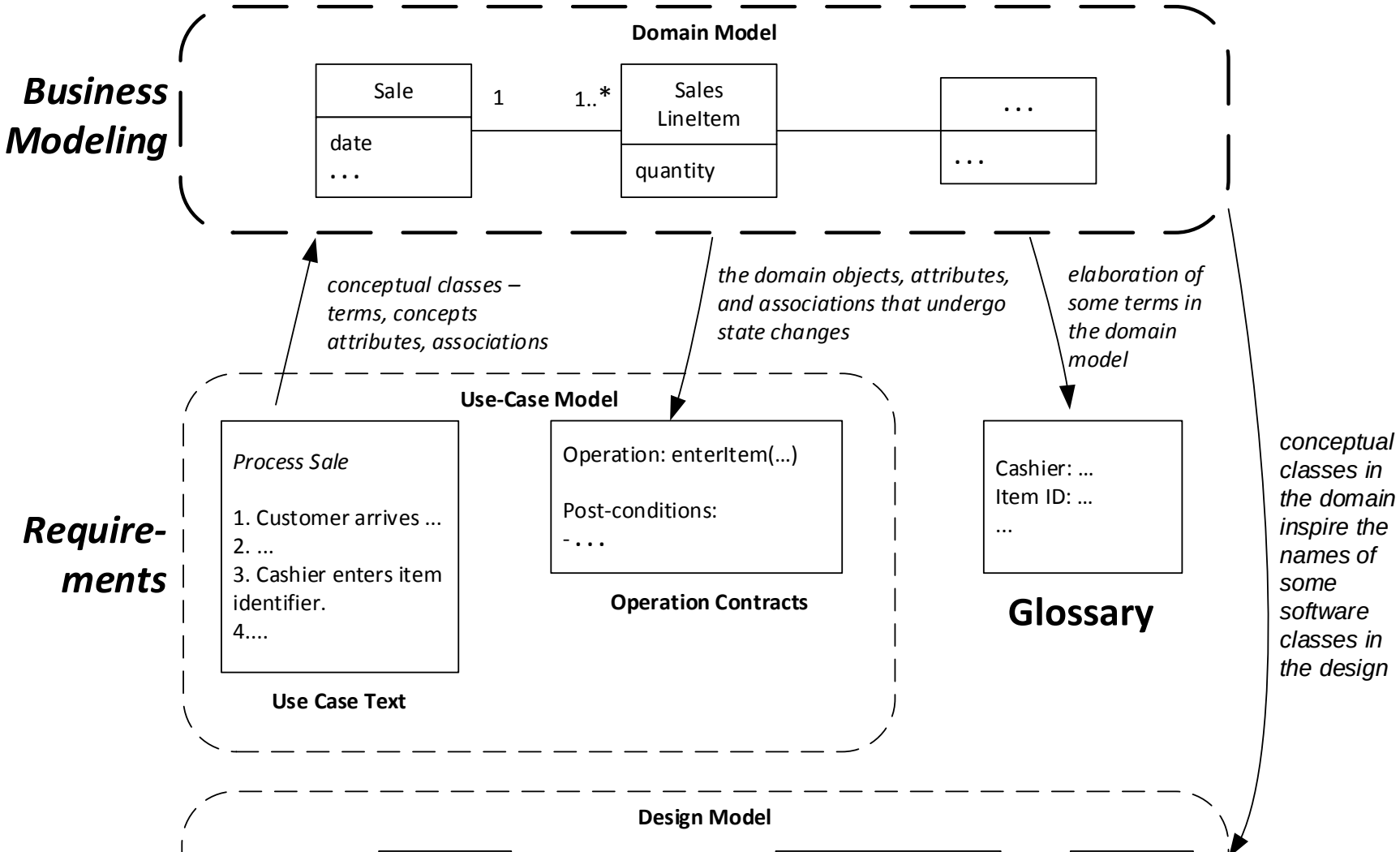
On completion of this topic you should be able to:

- Identify conceptual classes related to the current iteration.
- Create an initial domain model.
- Model appropriate attributes and associations.

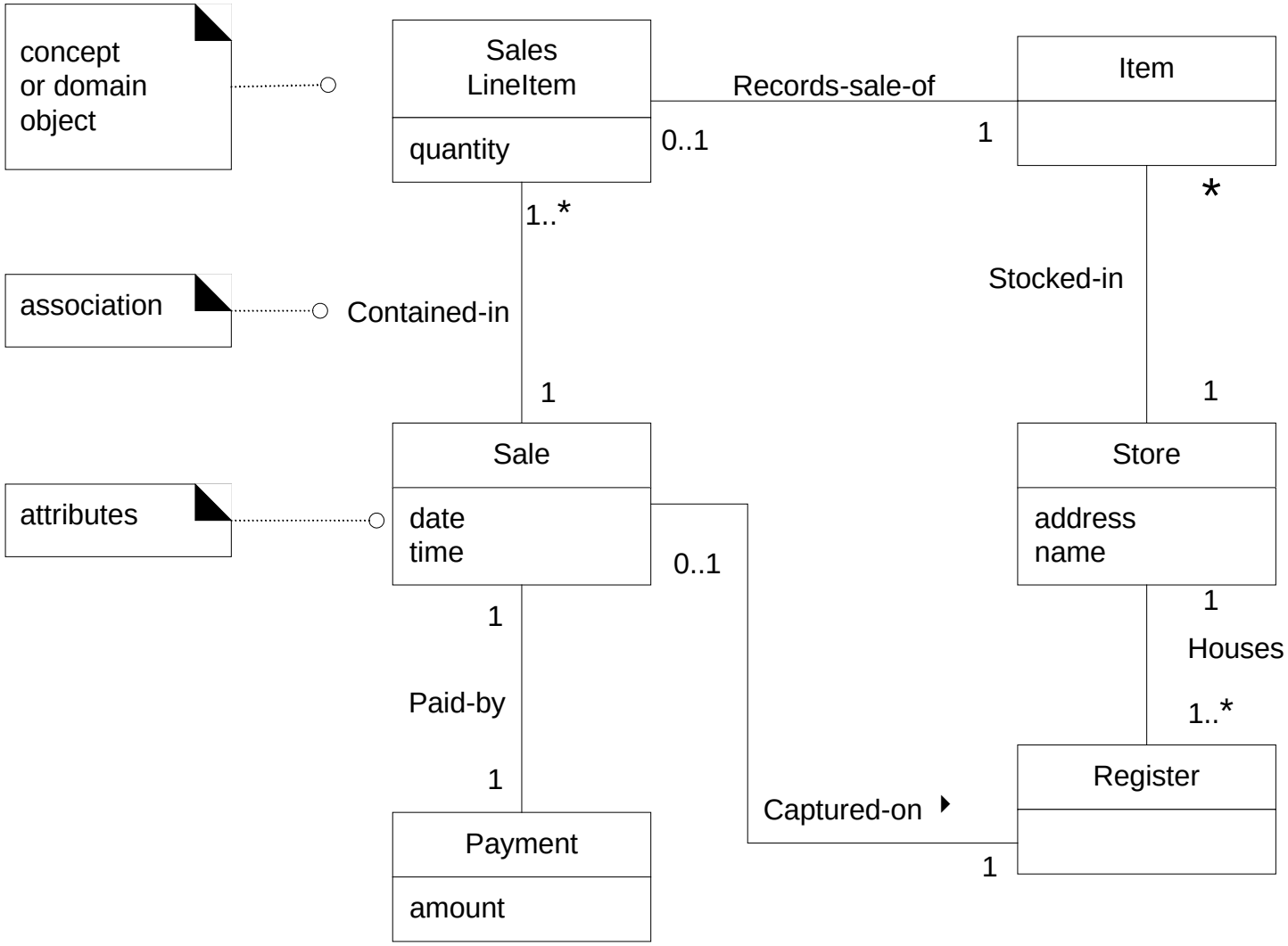
Domain Model: Influences in UP



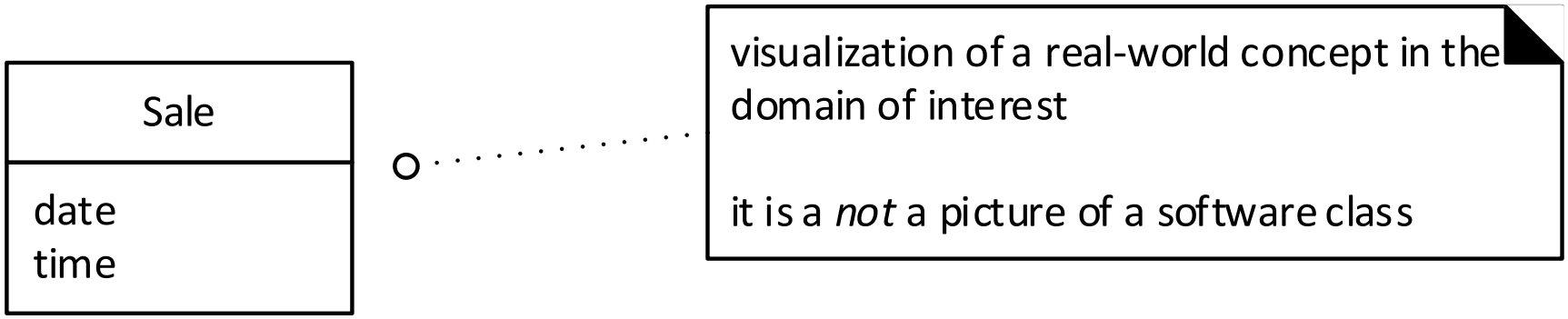
Domain Model: Influences in UP



Domain Model: A Visual Dictionary

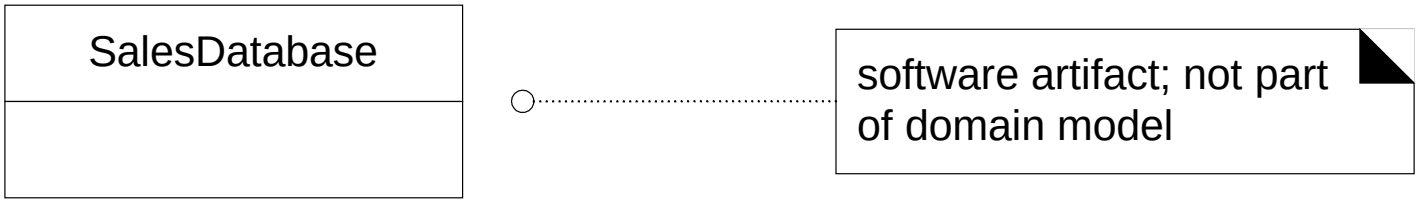


Conceptual classes only

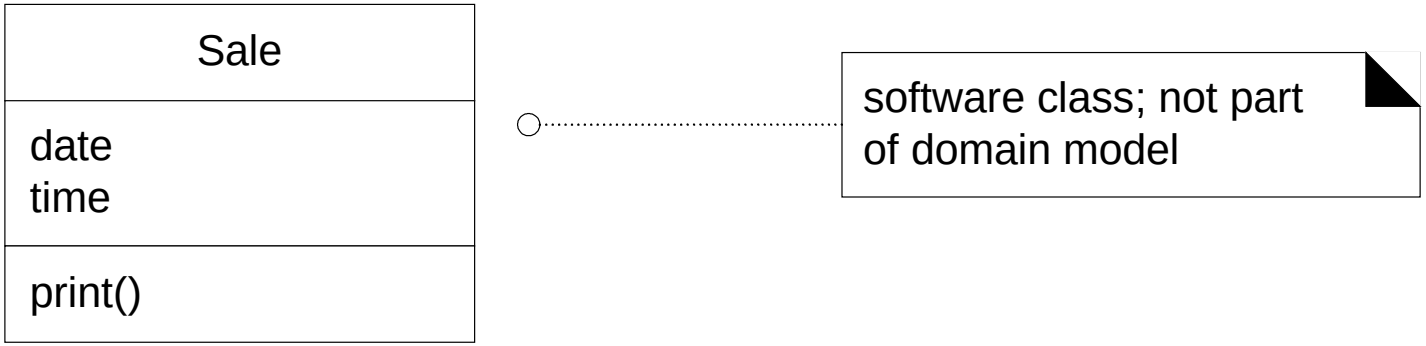


No Software Artifacts/Classes

avoid



avoid



What is a Conceptual Class?

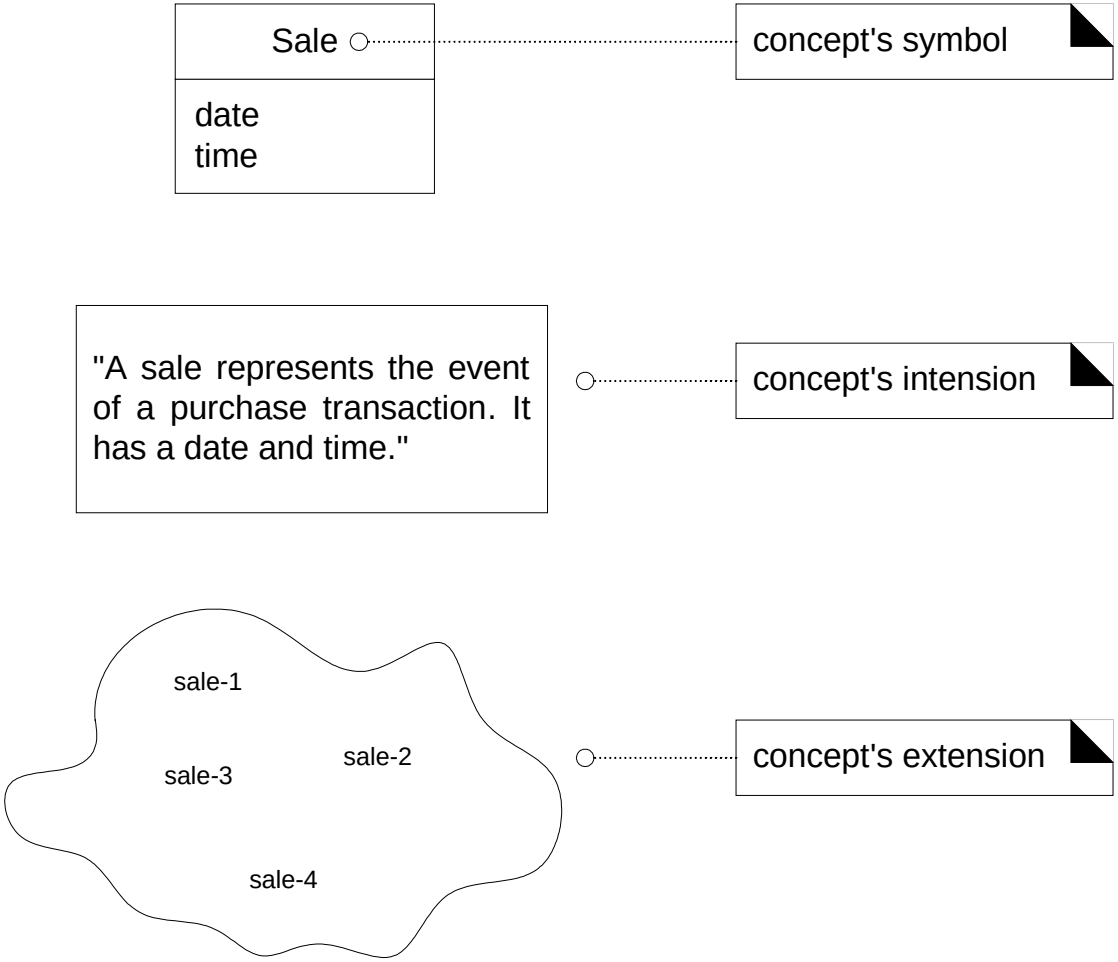
Informally: an idea, thing, or object in the real world.

More *formally*,

a conceptual class *C* is the combination of:

- Symbol—words or images representing *C*
- Intension—the definition of *C*
- Extension—set of instances represented by *C*

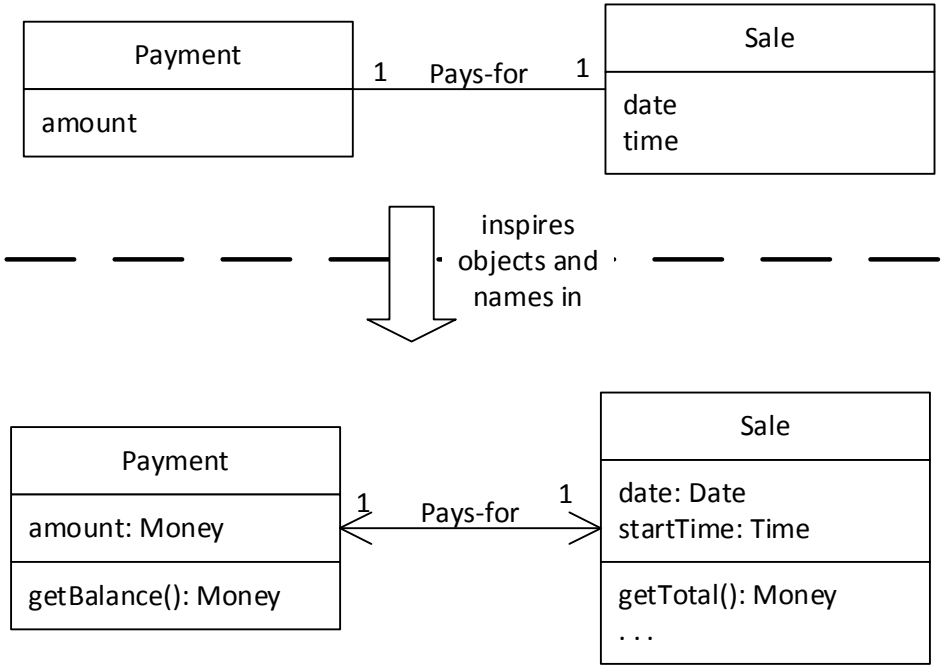
Example: Symbol, Intension, Extension



OO Modelling: Reducing the Rep. Gap

UP Domain Model

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



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.

Guideline: Creating a Domain Model

Bounded by the current iteration requirements under design:

1. Find the conceptual classes
   - Larman p139
2. Draw them as classes in a UML class diagram
3. Add associations and attributes
   - Larman p149,158

Guideline: Finding Conceptual Classes

Three strategies:

1. Reuse or modify existing models, e.g.
   a. Standardised/adopted domain model
   b. Organisational domain model
2. Use a category list
3. Identify noun phrases

Method 1 is primarily reuse; it is not covered further.

Method 2: Use a Category List

Method: Make a list of candidate conceptual classes in the domain, based on commonly occurring categories.

Conceptual Class Category	*Examples*
business transactions *Guideline:* critical (involve money)	*Sale, Payment* *Reservation*
physical objects *Guideline:* esp. for device-controllers or simulations	*Item, Register* *Board, Piece, Die* *Aeroplane*
containers of things (physical or information)	*Store, Bin* *Board* *Aeroplane*
Where are transactions recorded? *Guideline:* important	*Register, Ledger* *FlightManifest, MaintenanceLog*

Examples from Table 9.1 Conceptual Class Category List

Method 3: Use Noun Phrase Identification

Method: Make a list of candidate conceptual classes or attributes by identifying nouns in textual domain descriptions.

POS Use Case: Process Sale (Basic Flow)

1. **Customer** arrives at **POS checkout** with **goods** and/or **services** to purchase.
2. **Cashier** starts a new **sale**.
3. Cashier enters **item identifier**.
4. System records **sale line item** and presents **item description**, **price**, and running **total**. Price calculated from a set of price rules.

Cashier repeats steps 2-3 until indicates done.

5. System presents total with **taxes** calculated.
6. Cashier tells Customer the total, and asks for **payment**.
7. …

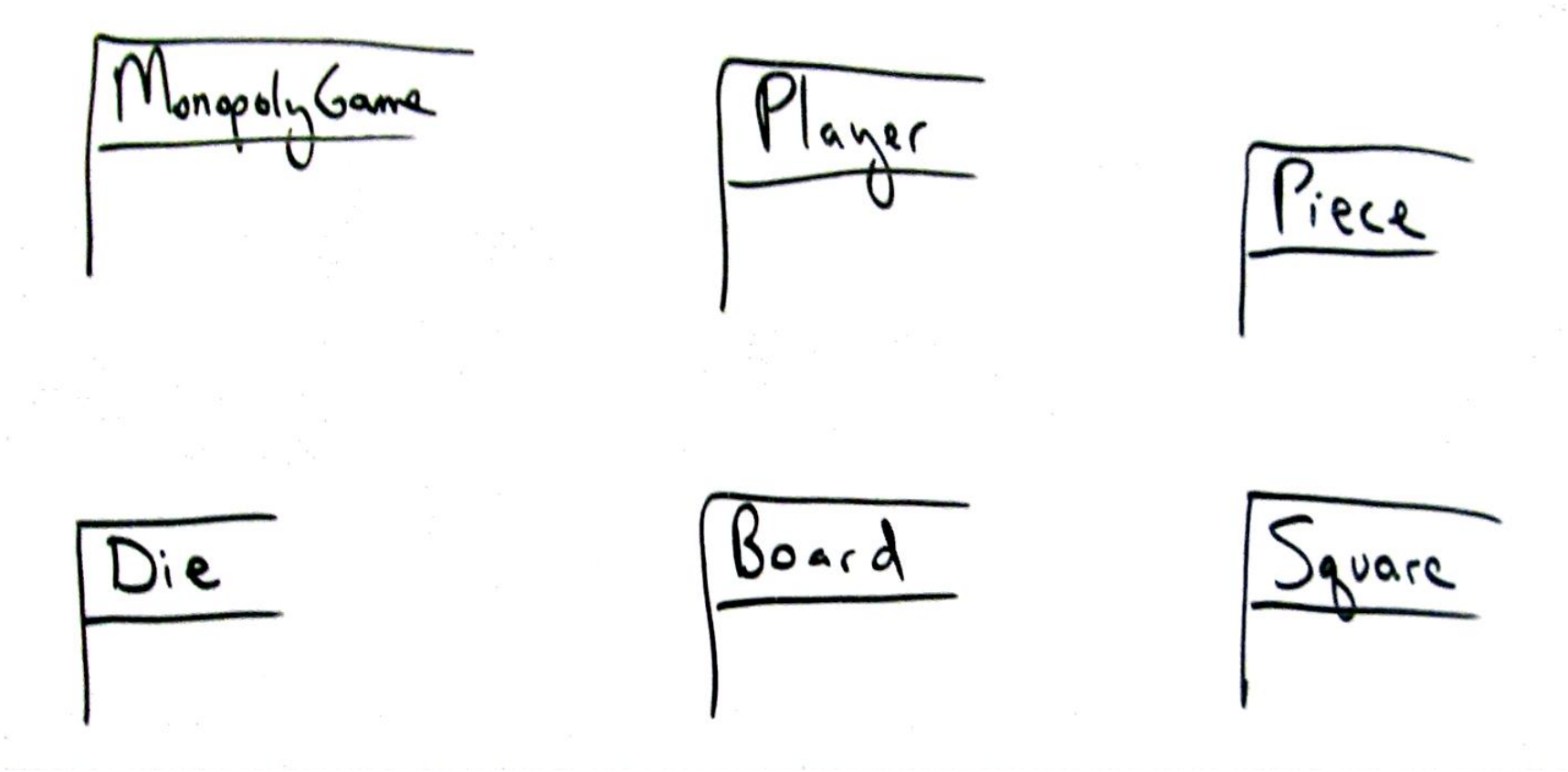
Guideline: Careful evaluation required; not mechanical.

Creating a Domain Model: Step 2

Register	Item	Store	Sale
Sales LineItem	Cashier	Customer	Ledger
Cash Payment	Product Catalog	Product Description	

Initial POS Domain Model

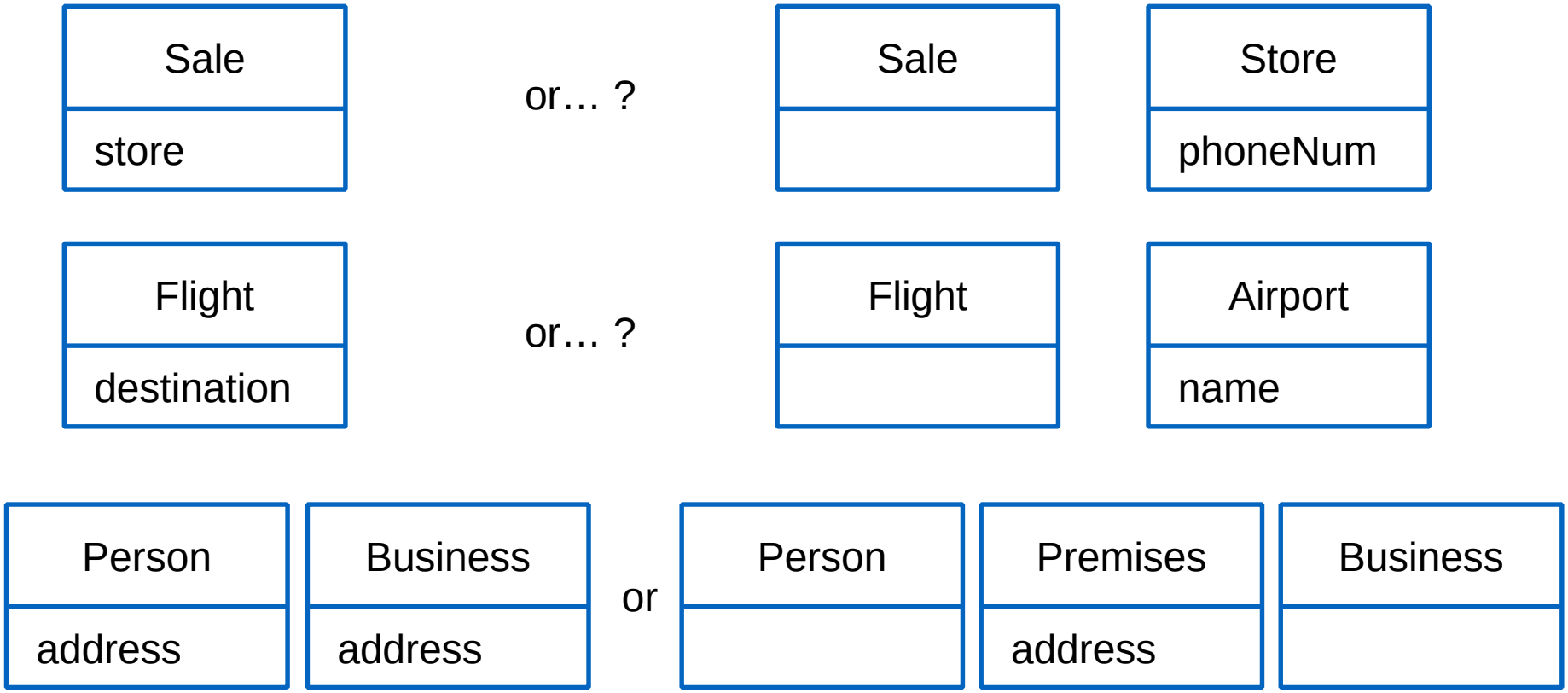
Creating a Domain Model: Step 2



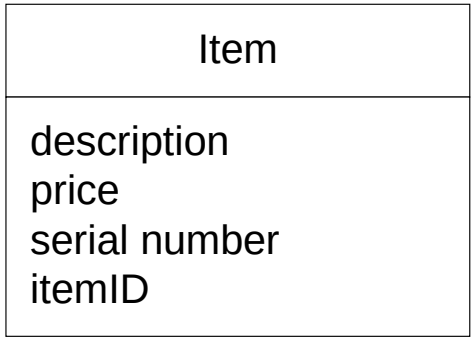
Initial Monopoly Domain Model

Attributes vs. Classes

Guideline: If X not considered a number or text in the real world, X is probably a conceptual class, not an attribute.



Description Classes: Example



Worse

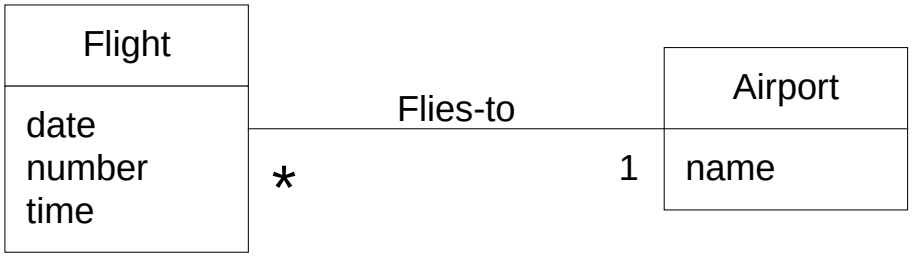
Better

Description Class Guidelines

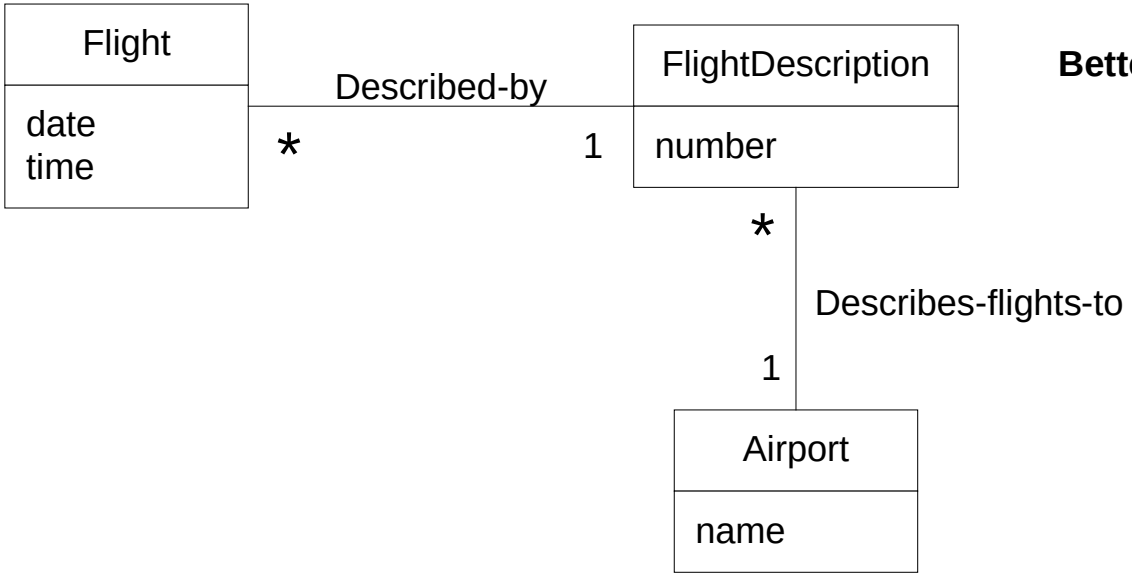
A **description** *class* contains information that describes something else; it should be used when:

- Groups of items share the same description
- Items need to be described even when there are currently no examples.
- It reduces redundant or duplicated information (design)
- Deleting instances results in losing required info (design)

Descriptions involving Other Classes



Worse



Better

Associations

An **association** represents some meaningful and significant relationship between classes.

Guidelines:

- Significant in the domain
- Knowledge of the relationship needs to be preserved
- Derived from the Common Associations List (Table 9.2)

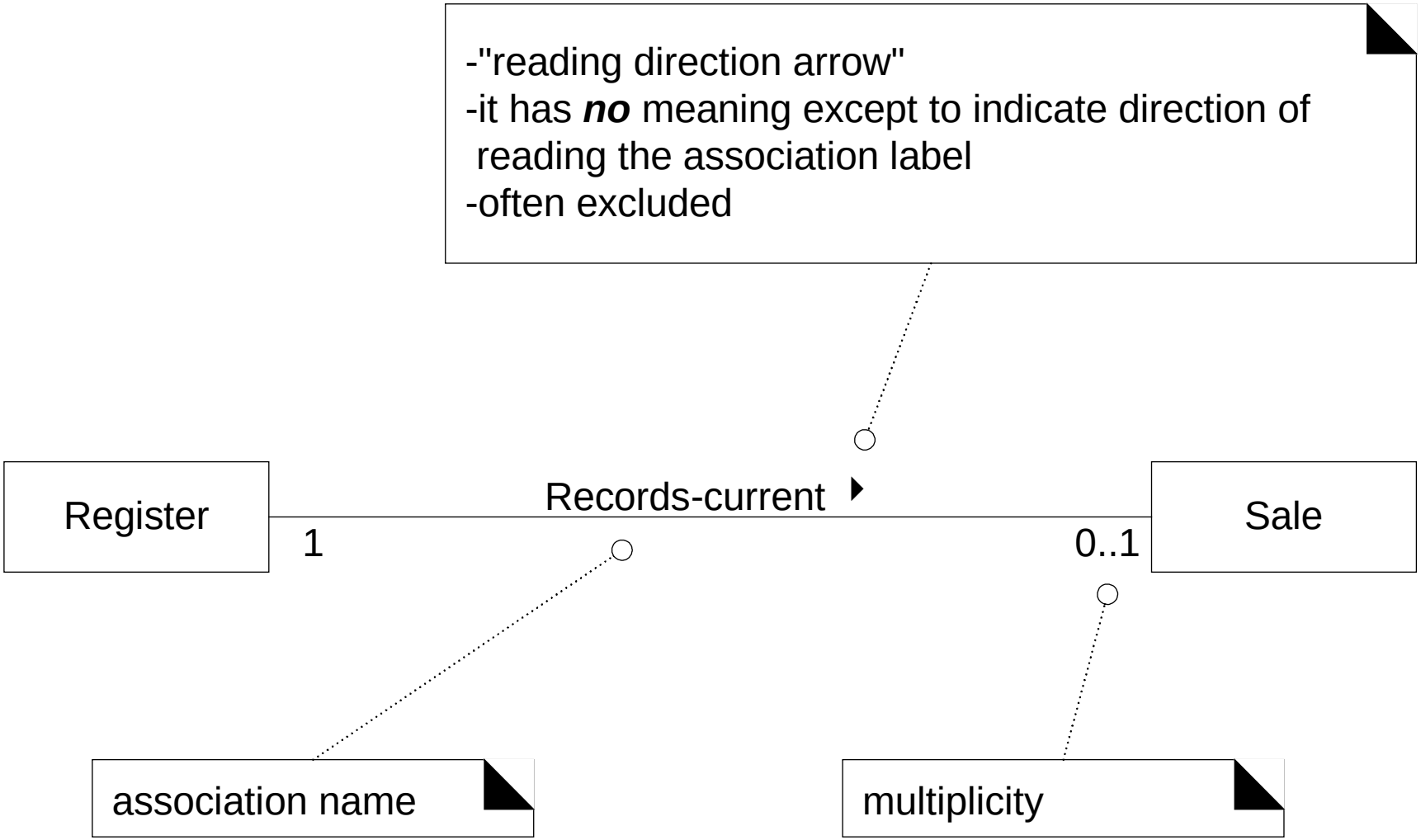
Guideline: Use a Category List

Make a list of candidate associations in the domain, based on commonly occurring categories.

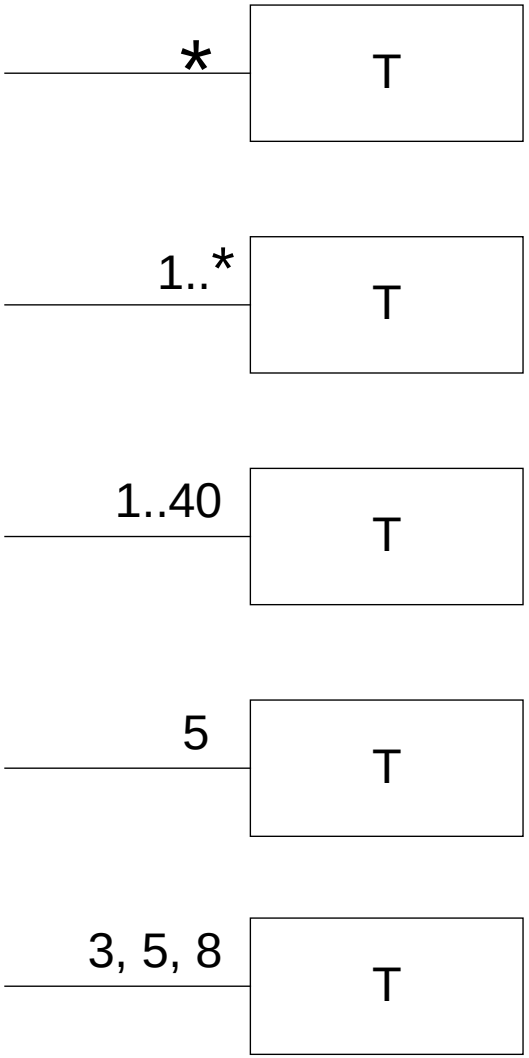
Association Category	*Examples*
A is a member of B	*Cashier—Store* *Player—MonopolyGame* *Pilot—Airline*
A is a role related to a transaction B	*Customer—Payment* *Passenger—Ticket*
A is physically or logically contained in/on B	*Register—Store, Item—Shelf* *Square—Board* *Passenger—Airplane*
A is known/logged/recorded/reported/captured in B	*Sale—Register* *Piece—Square* *Reservation—FlightManifest*

Examples from Table 9.2 Common Associations List

The UML Notation for Associations



Multiplicity Values



zero or more; "many"

one or more

one to 40

exactly 5

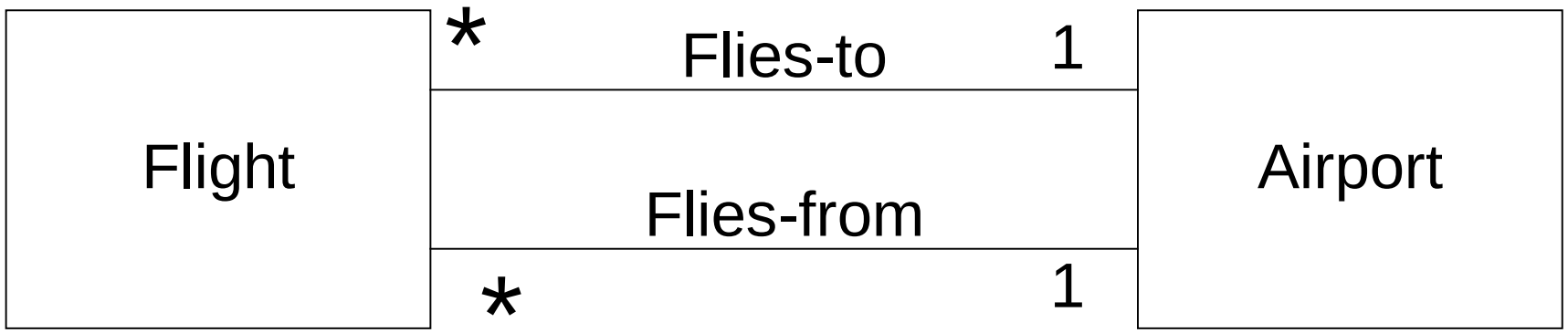
exactly 3, 5, or 8

Multiplicity: Context Dependence

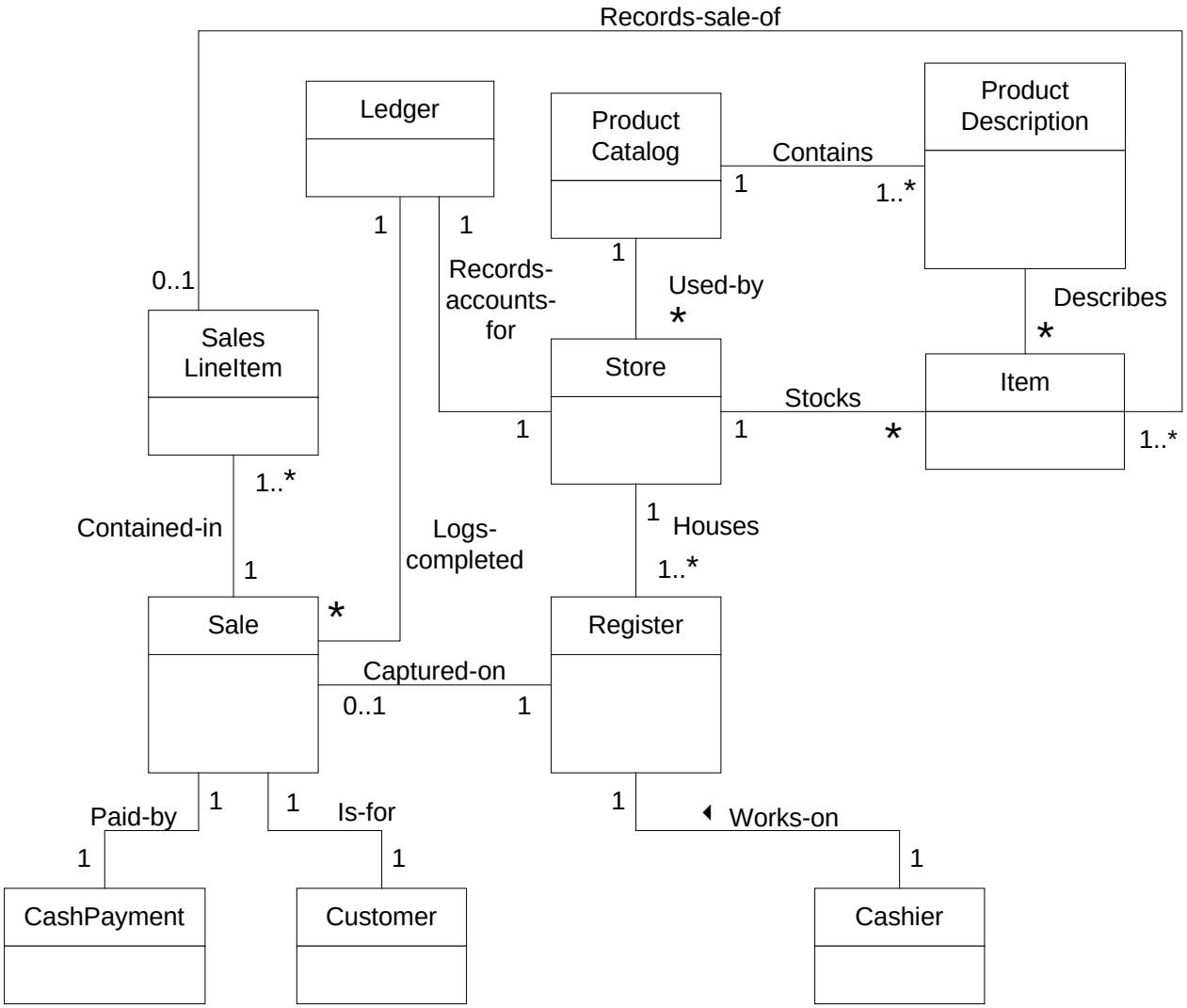
It depends on …

- The scope of our model:
  - Do we care about items before/after the store
- Is this a constraint we want to maintain?
  - Is it a problem if an item is not attached to a store

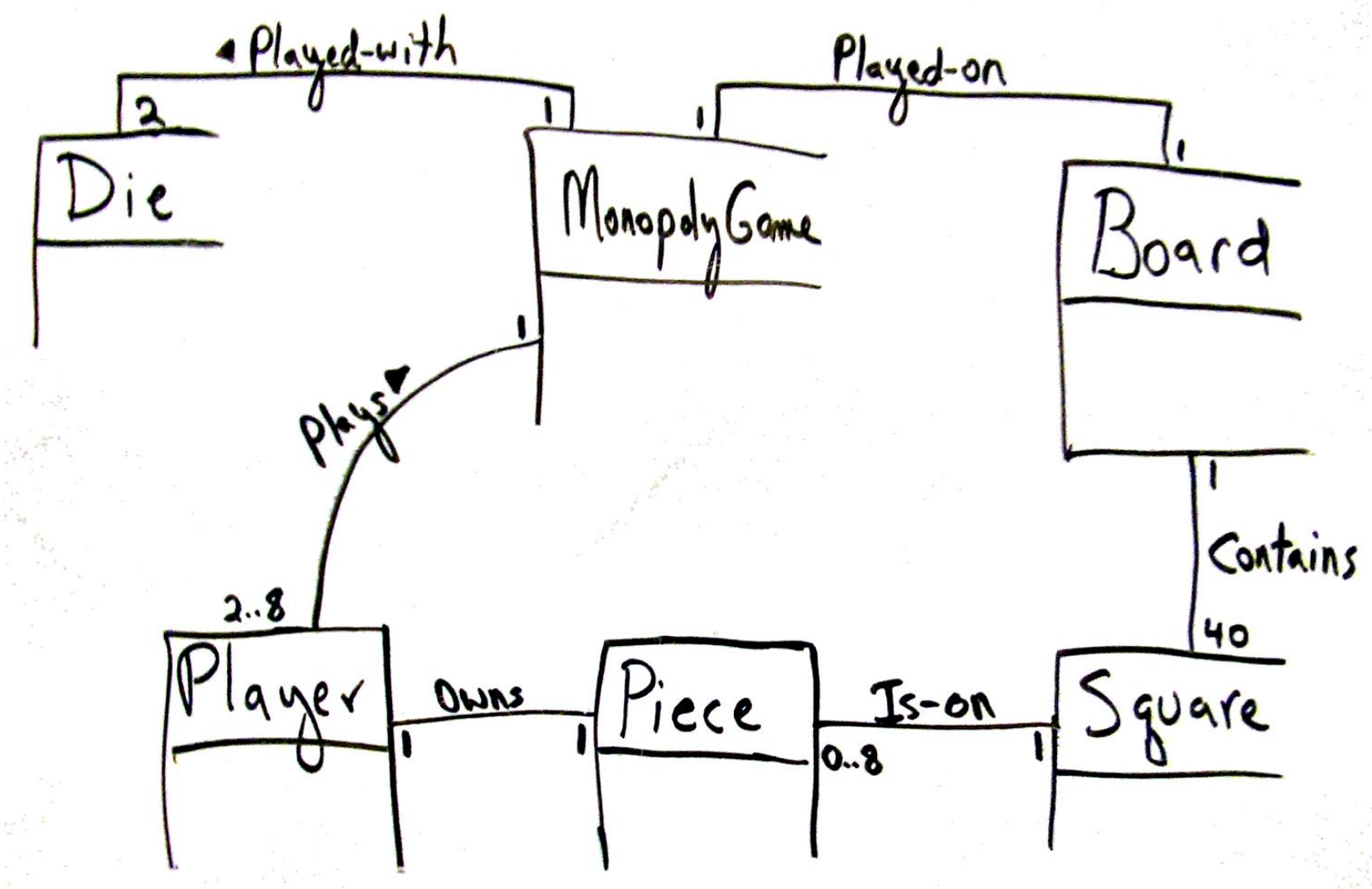
Multiple Associations



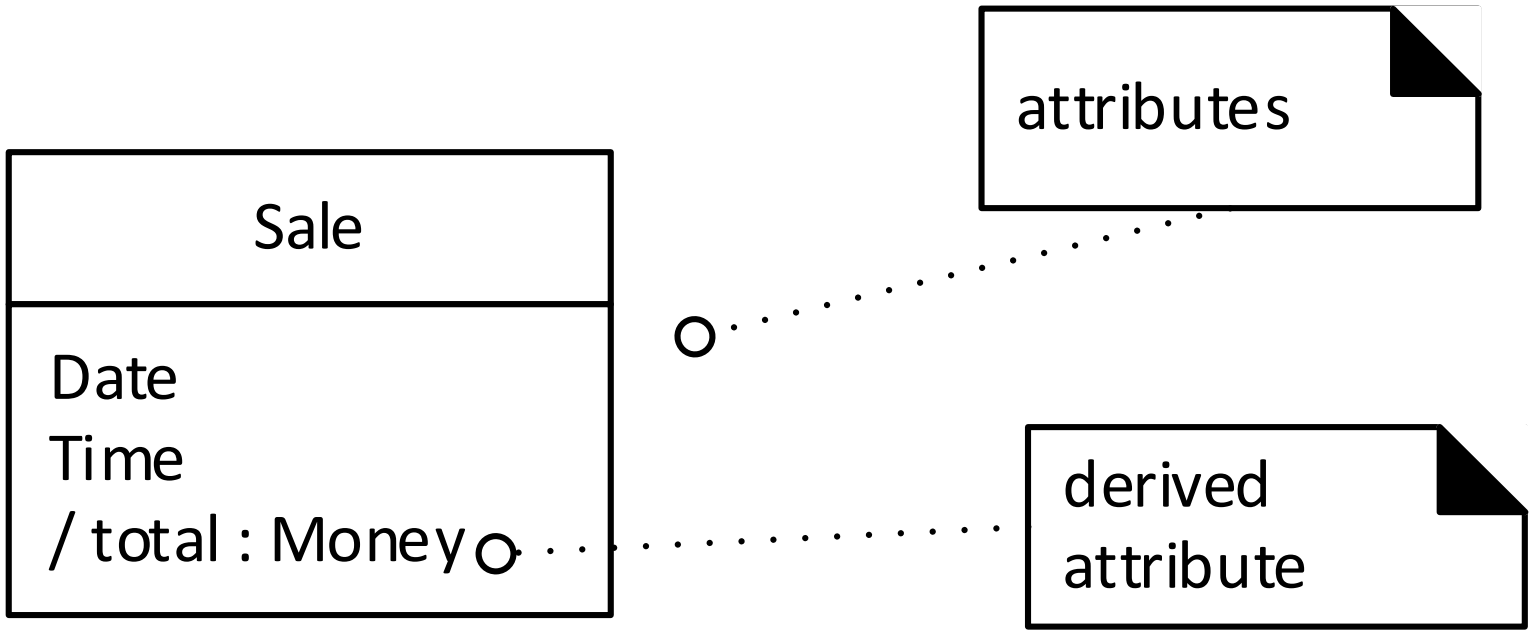
NextGen POS Partial Domain Model



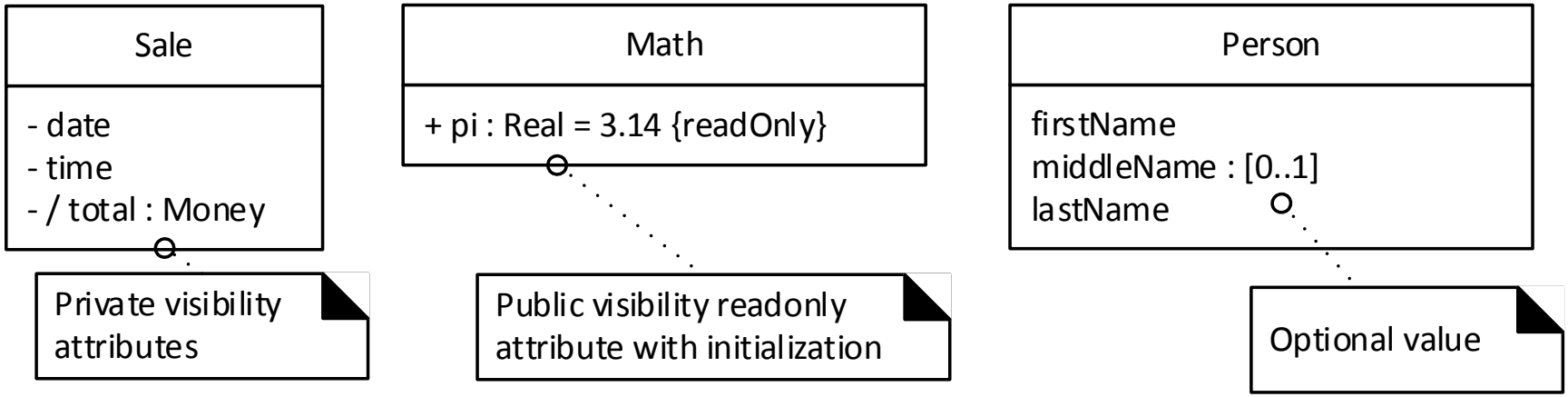
Monopoly Partial Domain Model



Class and Attributes



UML Attribute Notation



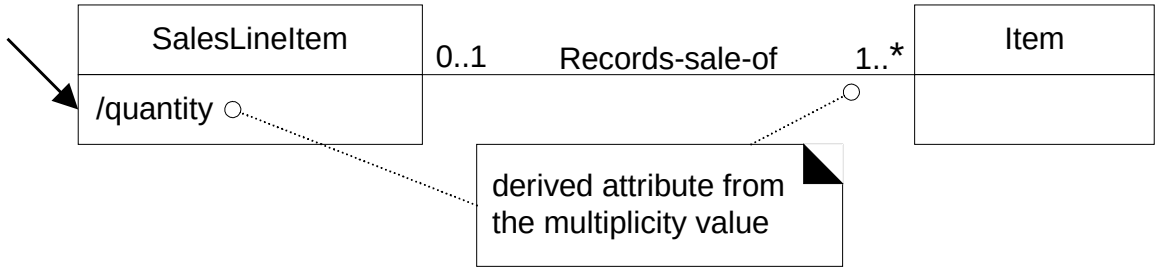
UML Attribute Syntax:

visibility derived name : type multiplicity = default {property-string}

Line Item: Quantity of Items Sold

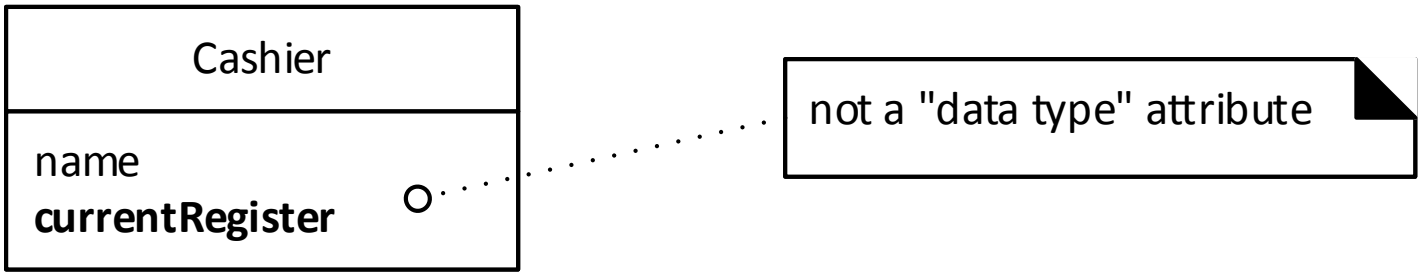
Each line item records a separate item sale.
For example, 1 tofu package.

Each line item can record a group of the same kind of items.
For example, 6 tofu packages.

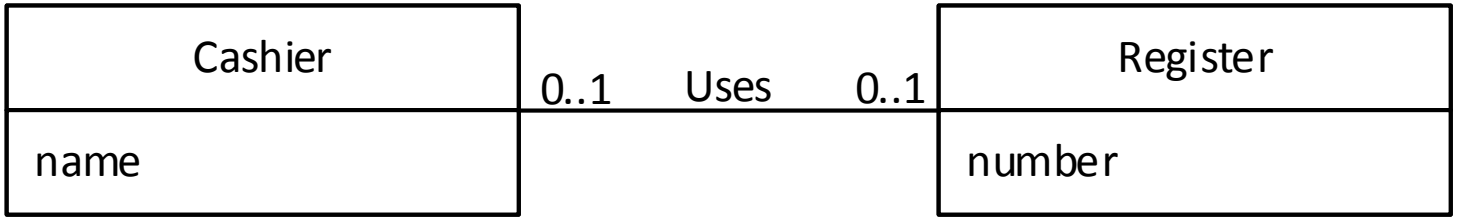


Associations, not Attributes (1)

Worse

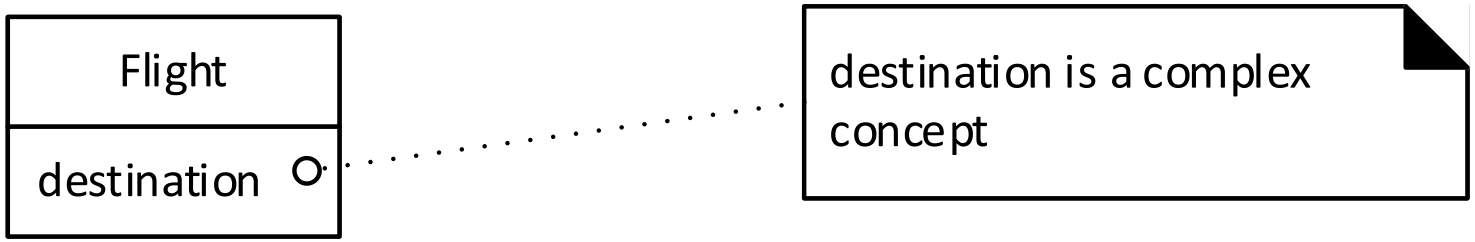


Better



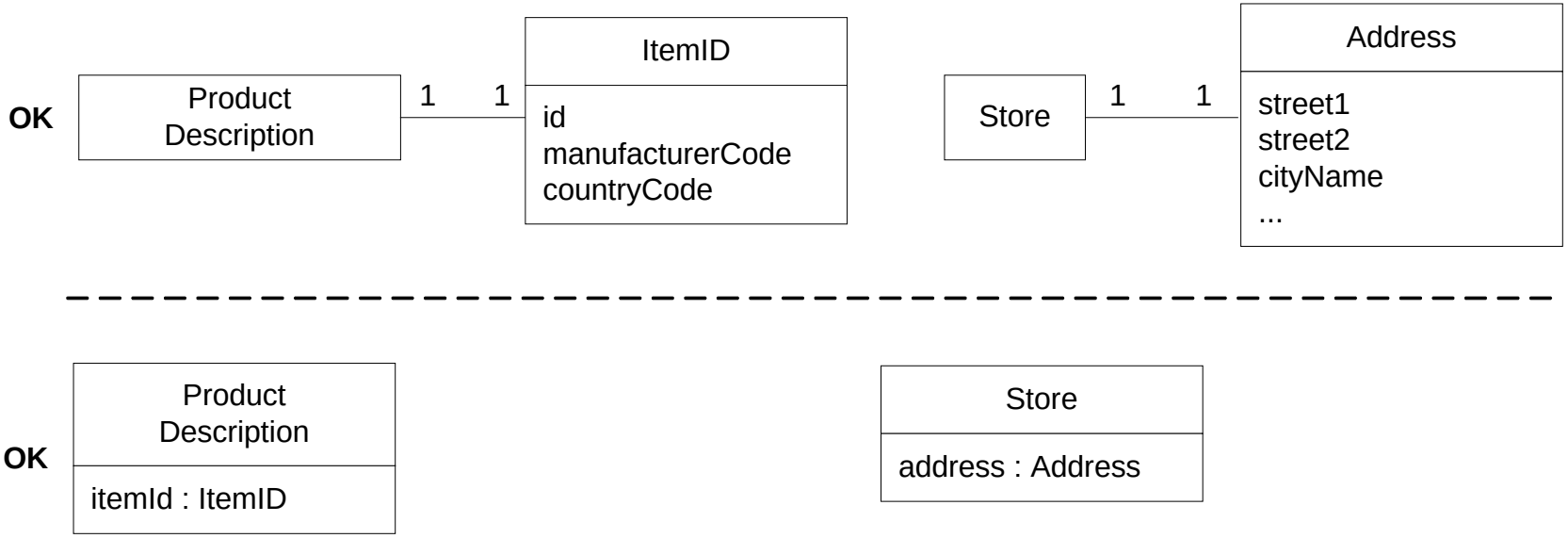
Associations, not Attributes (2)

Worse



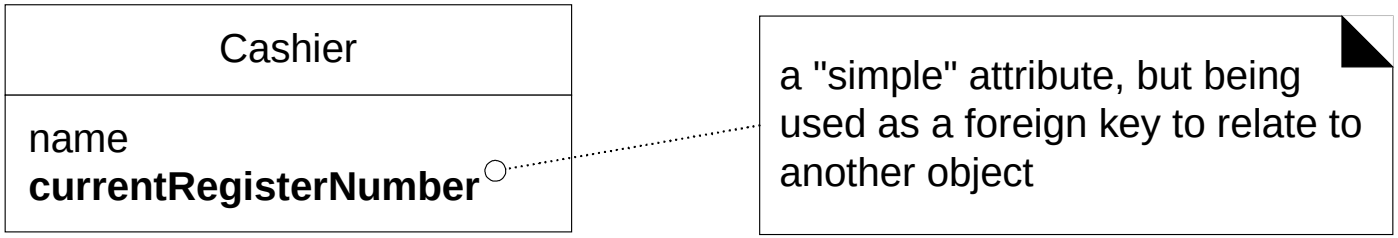
Better

Data Type Classes

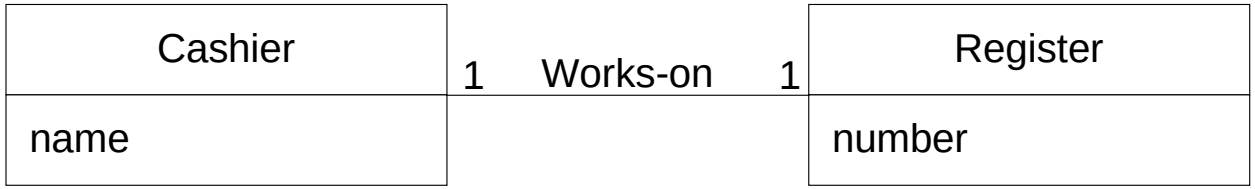


Don’t Use Attributes as Foreign Keys

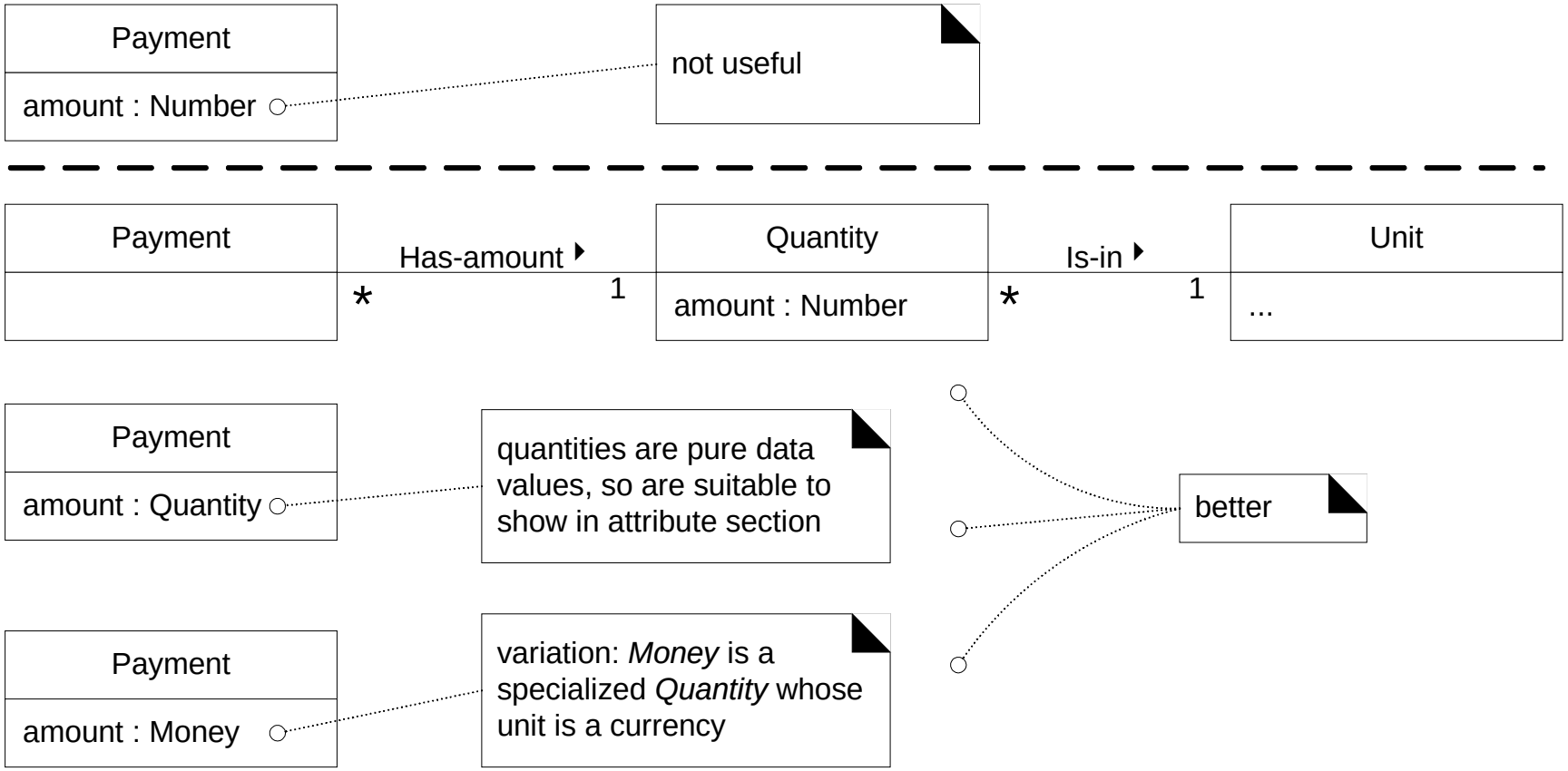
Worse



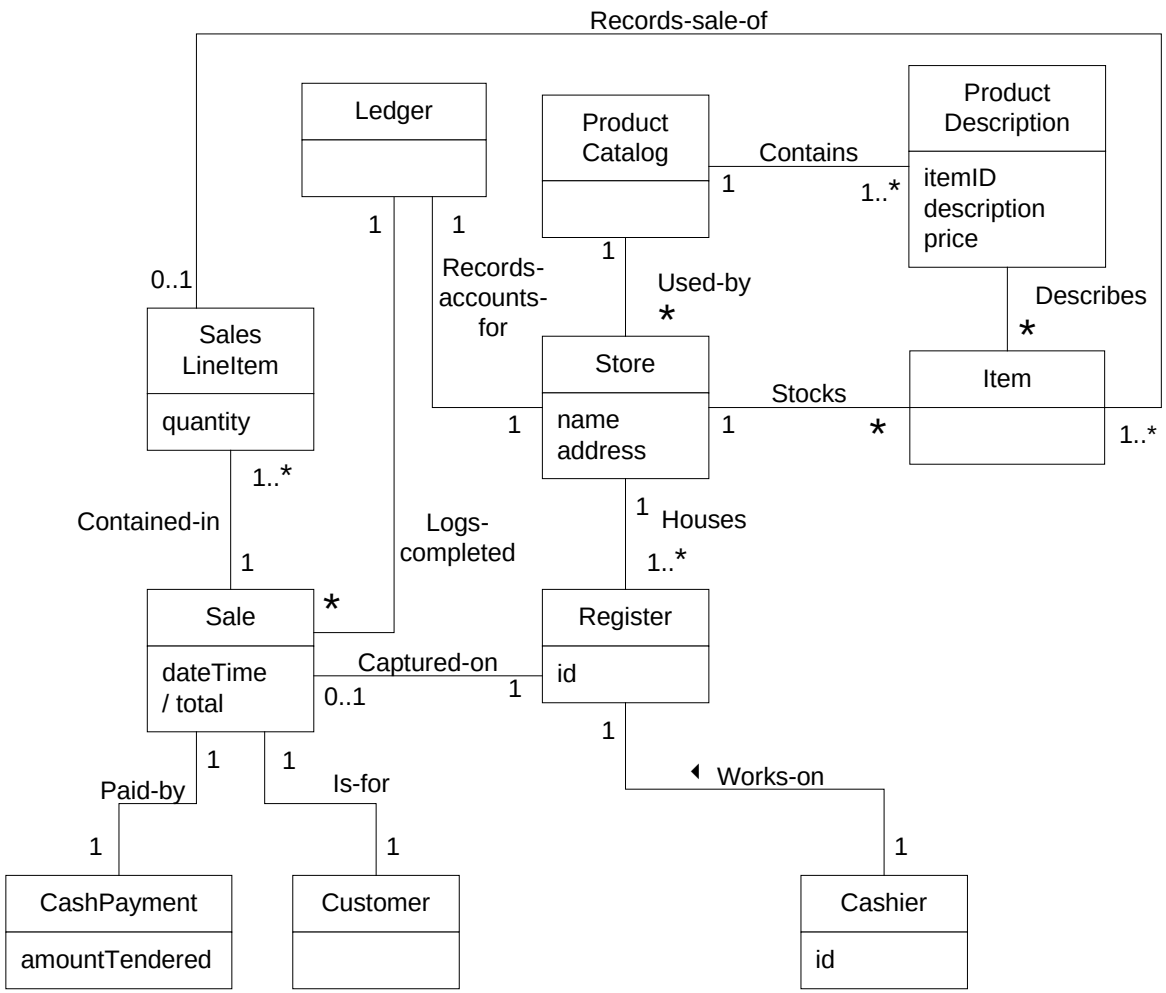
Better



Modelling Quantities



NextGen POS: Attributes in Model



Monopoly: Attributes in Model