The University of Iowa

CS:2820 (22C:22)

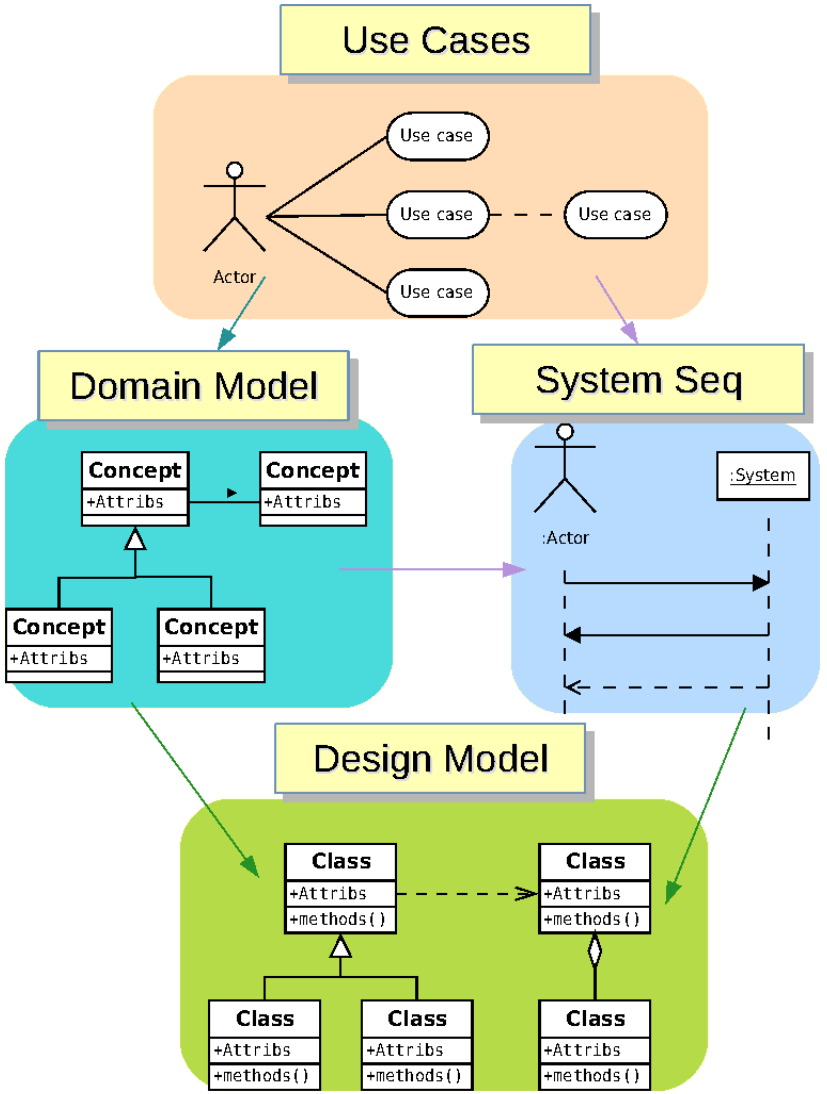
Object-Oriented Software Development

Spring 2015

The Domain Model

by
Mauricio Monsalve

Design road



- Beginning the design
- Domain Model describes the domain (context)
- It is followed by the System System Diagram(s)
- Then follows the OO Design Model(s)

Domain Model

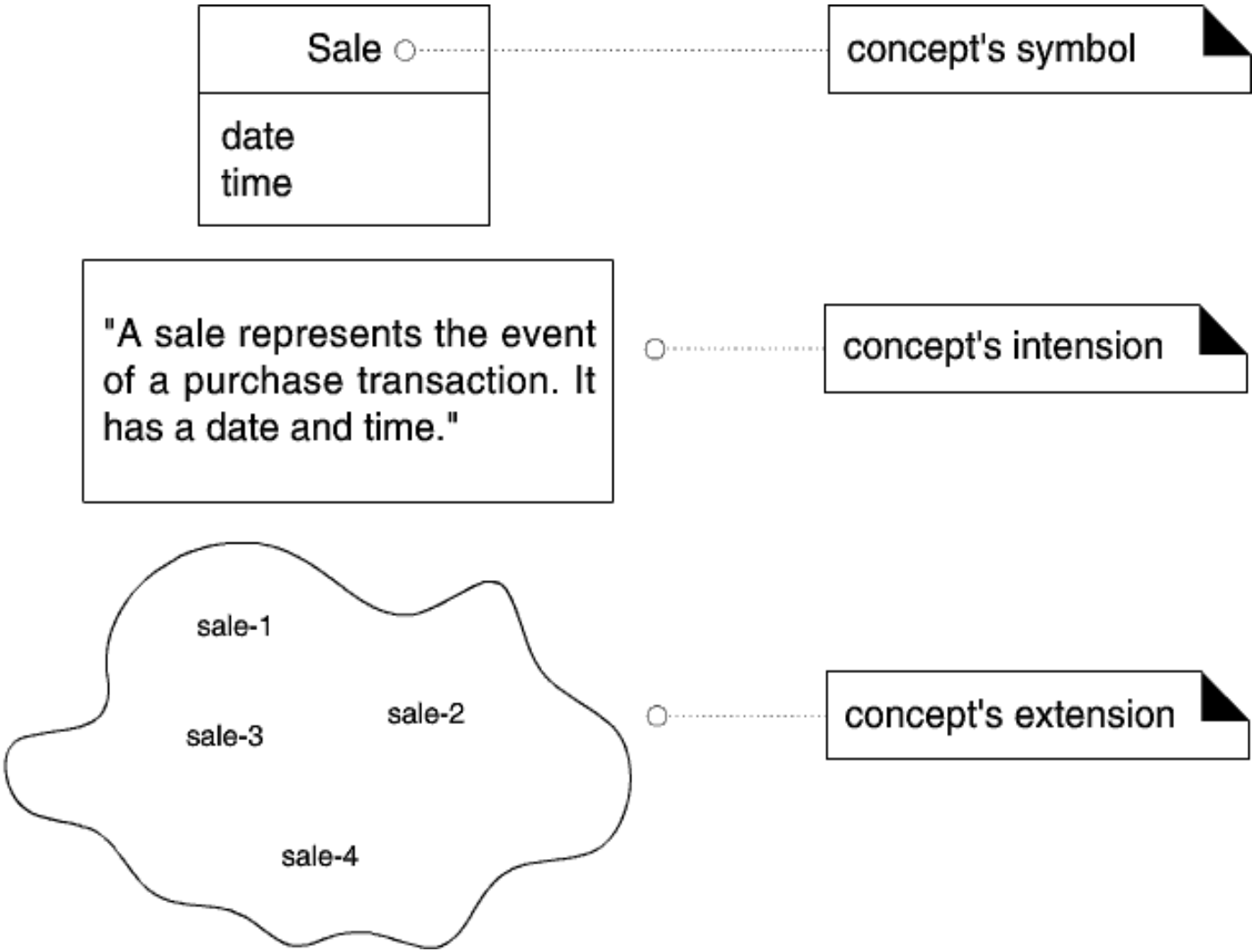
Visual representation of conceptual classes or real-situation objects in a domain

- The objects (conceptual classes) of the domain are **not** software objects (classes)
- In UML, the Domain Model is illustrated with a **set** of **class** diagrams **without methods**
- Also known as *Visual Dictionary*
- Perhaps most essential diagram

Domain Model

- **Conceptual class**: idea, thing, object
- It may be considered in terms of
  - **Symbol**—words or images representing the conceptual class
    - *Visual depiction, diagram*
  - **Intension**—the definition of the conceptual class
    - *Comprehensive definition*
  - **Extension**—the set of examples to which the conceptual class applies
    - *Extensive definition*

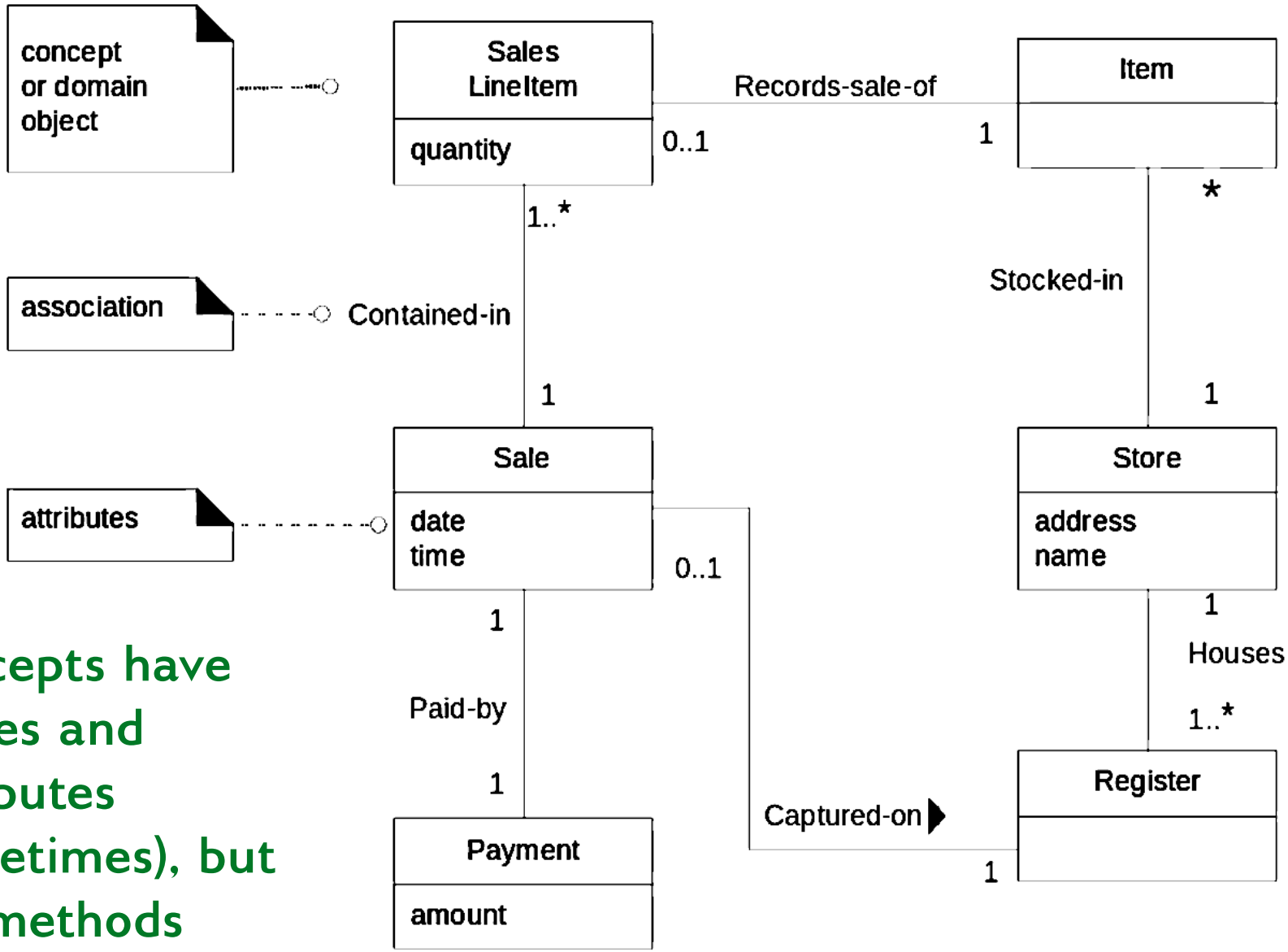
Domain Model



Domain Model

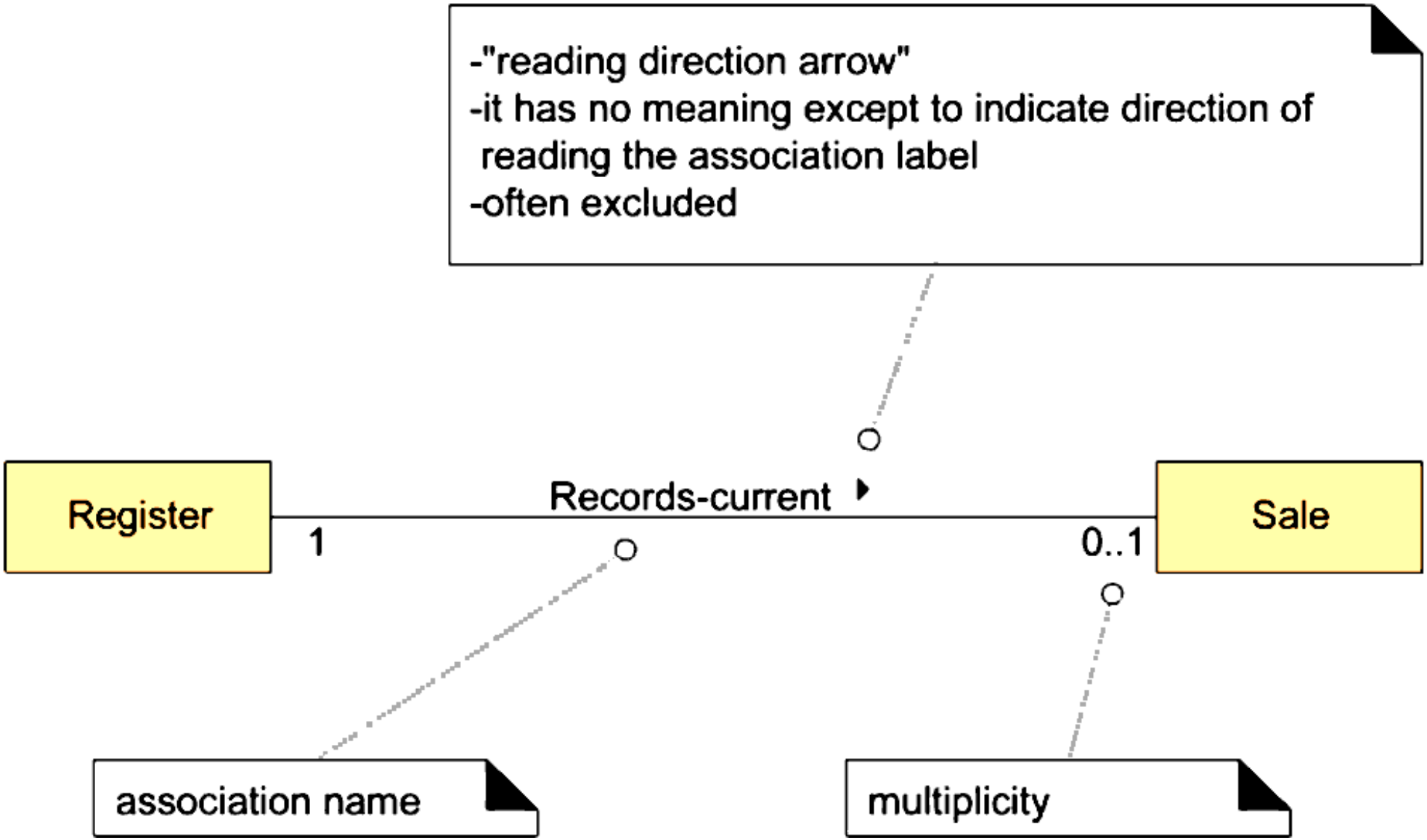
- **Conceptual classes only!**
- Not in domain model
  - Software artifacts
    - *Windows, databases, etc.*
    - *Unless domain is software*
  - Responsibilities or methods
    - *They are part of the design and the software*

Drawing the DM

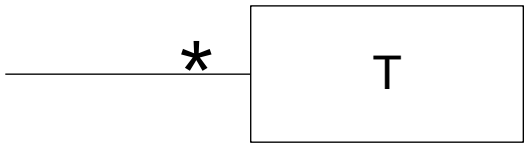


Concepts have names and attributes (sometimes), but not methods

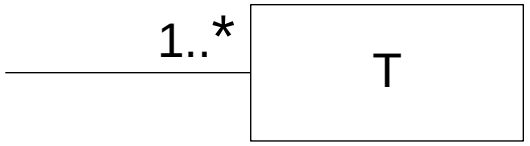
Drawing the DM



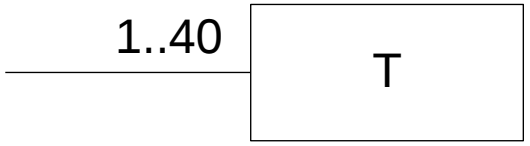
Drawing the DM



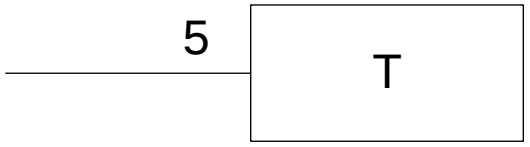
zero or more;
"many"



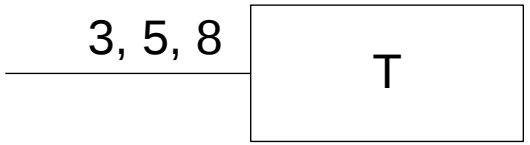
one or more



one to 40



exactly 5



exactly 3, 5, or 8

Associations should have specific names—and so classes

"has" is a bad name!

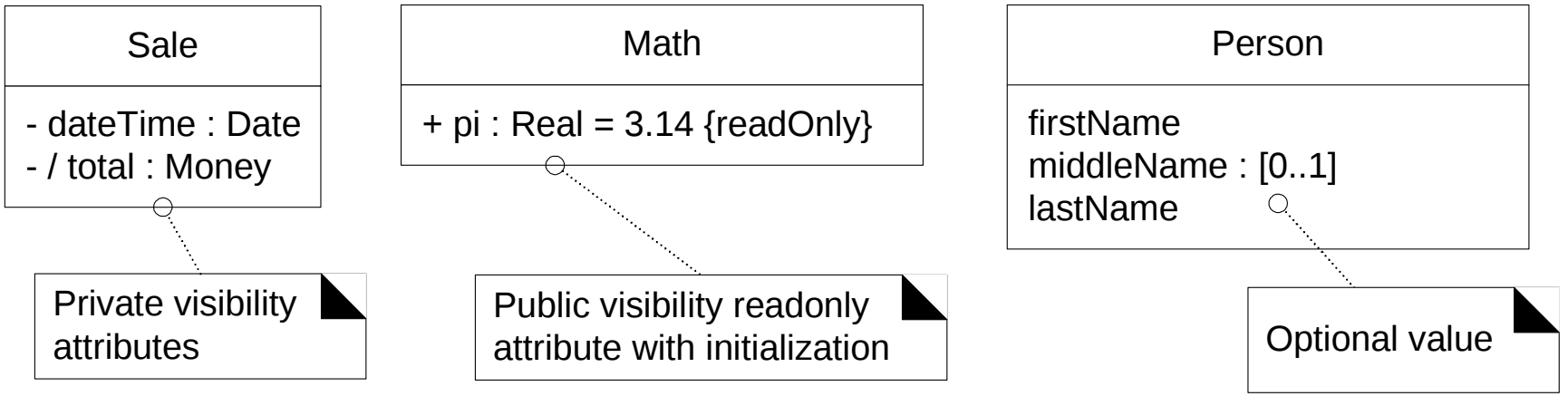
Multiplicity should be kept simple at this stage though; conceptual modeling dœs not need to be as precise

Drawing the DM

Classes also include attributes relevant to the domain

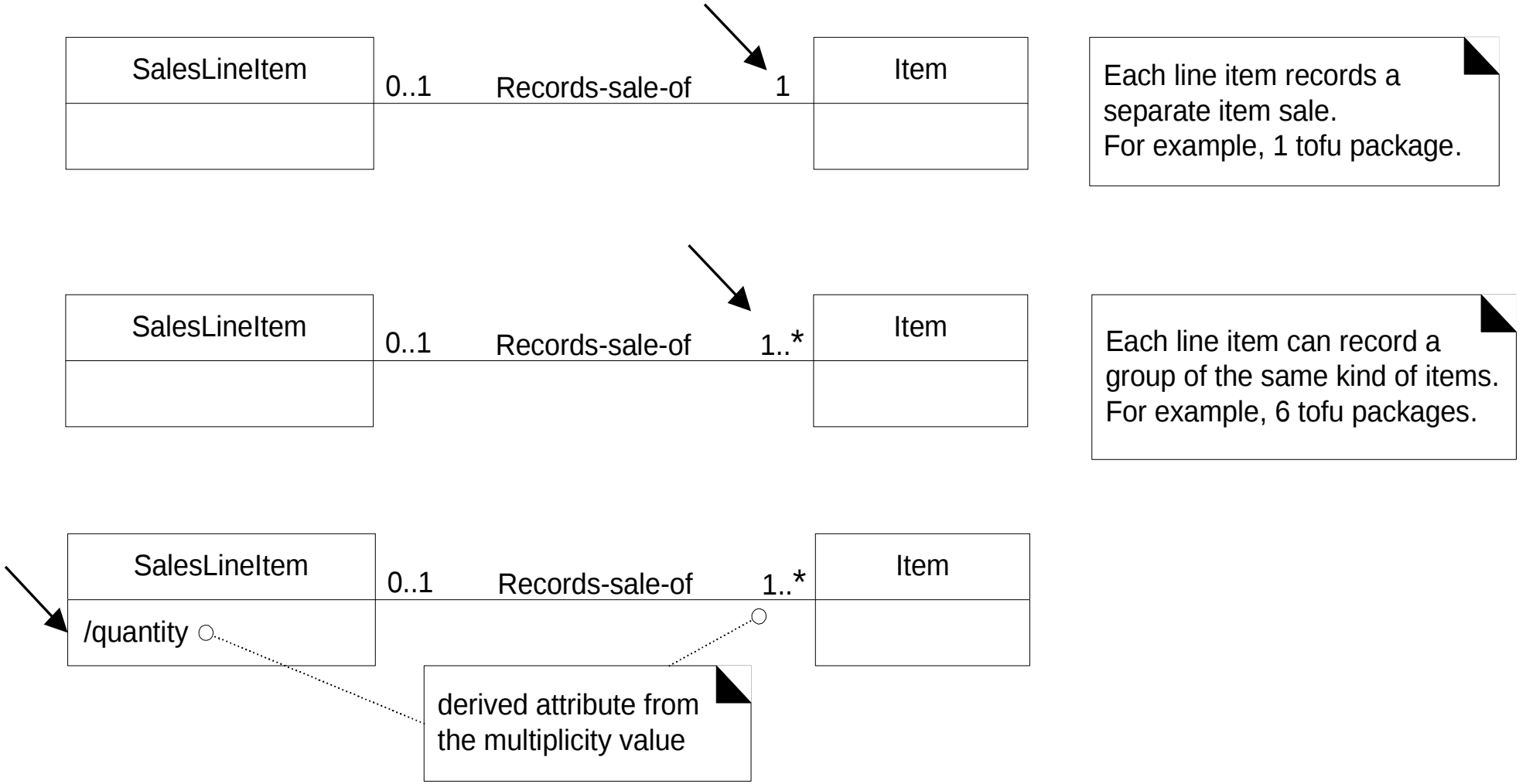
Attributed should have simple types (numbers, text, booleans, etc.)
—this is not so relevant at this stage, though

They sometimes can be "derived", meaning that they can be inferred, by means of a formula, from other attributes or associations



Drawing the DM

Example of association and derived attribute



Drawing the DM

- Pick exact, specific names for objects, associations, etc
- Attributes should have trivial types (numbers, text, etc)
- Keep the most relevant associations in the DM
- **Encapsulation**—don't connect everything to everything!
  - Objects should have strong internal cohesion (attributes)
  - Objects should be associated to few other objects (low coupling)
- Problem: **where do we get our objects from?**

Getting the concepts

- Reuse existing models
  - Copy or adapt existing, well accepted models
  - Use existing DM when modifying a system
- Category lists
  - Table 9.1 in book
- Identify noun phrases (easy!)
  - Pick nouns, events
  - Fully dressed use cases are great source
  - **Beware**: natural language is **imprecise**

Getting the concepts

Class lists consist of common taxonomies

The book has one, but there are other lists—*are they for the DM?*

- Business transactions
- Transaction line items
- Products, services
- Place of transaction
- Noteworthy events
- Physical objects
- Descriptions of things
- Catalogs
- Containers of things
- Things in a container
- Other related systems
- Fiduciary records (work, contracts, finances, etc)
- Financial instruments
- Reference documents (schedules, manuals, etc)

Getting the concepts

Main Success Scenario (or Basic Flow):

1) Customer arrives at a 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) Customer pays and System handles payment.
8) System logs the completed sale and sends sale and payment information to the external Accounting (for accounting and commissions) and Inventory systems (to update inventory).
9) System presents receipt.
10) Customer leaves with receipt and goods (if any).

Getting the concepts

Main Success Scenario (or Basic Flow):

1) Customer arrives at a 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) Customer pays and System handles payment.
8) System logs the completed sale and sends sale and payment information to the external Accounting (for accounting and commissions) and Inventory systems (to update inventory).
9) System presents receipt.
10) Customer leaves with receipt and goods (if any).

Getting the concepts

- Using nouns or noun phrases has [almost] no learning curve (easy!)
- Verbs often imply associations
- This method is as imperfect as natural language
  - Also serves to check use cases and requirements though
- Solutions
  - Refer to class list
  - Talk to domain expert
- **But we can patch our approximations!**

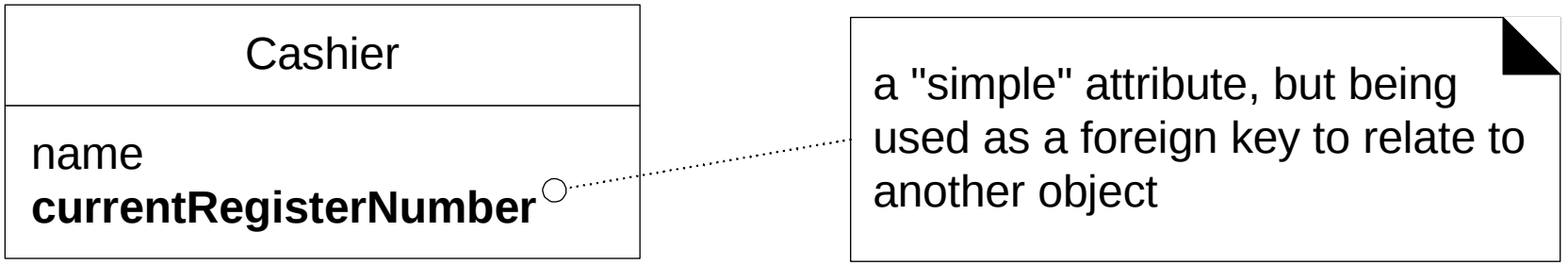
Getting more concepts

We may **infer** more concepts from what we have

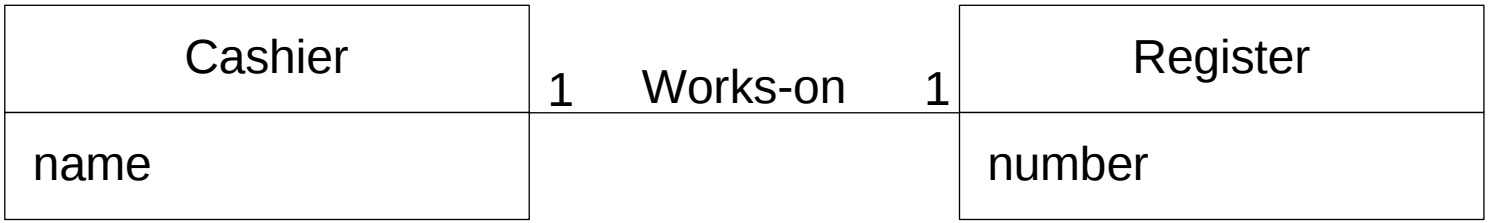
- **Patch the holes**—are there *holes* in the design?
- **Specialize**—are there interesting subclasses?
- **Generalize**—are there interesting superclasses?
- **Decompose**—can an object be described in terms of other, contained objects? (*Many meanings to this*)
- **Compose**—is a collection of given objects relevant?
- **Fix imperfections**—next slides

Getting more concepts

Worse



Better



"Foreign keys" identify other concepts. "Keys" are identifiers; they hold a 1-to-1 relation with an object

Getting more concepts

Item
description price serial number itemID

Worse

Another foreign key example

ProductDescription
description price itemID

Describes

1 — *

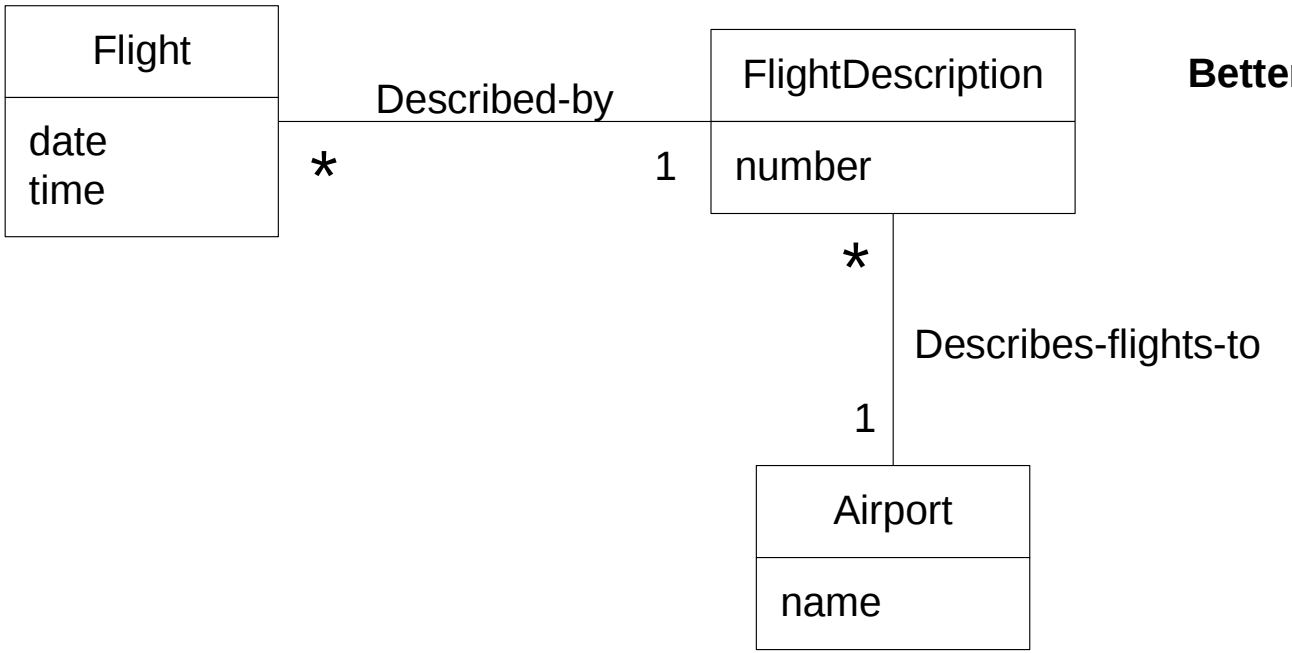
Item
serial number

Better

Getting more concepts

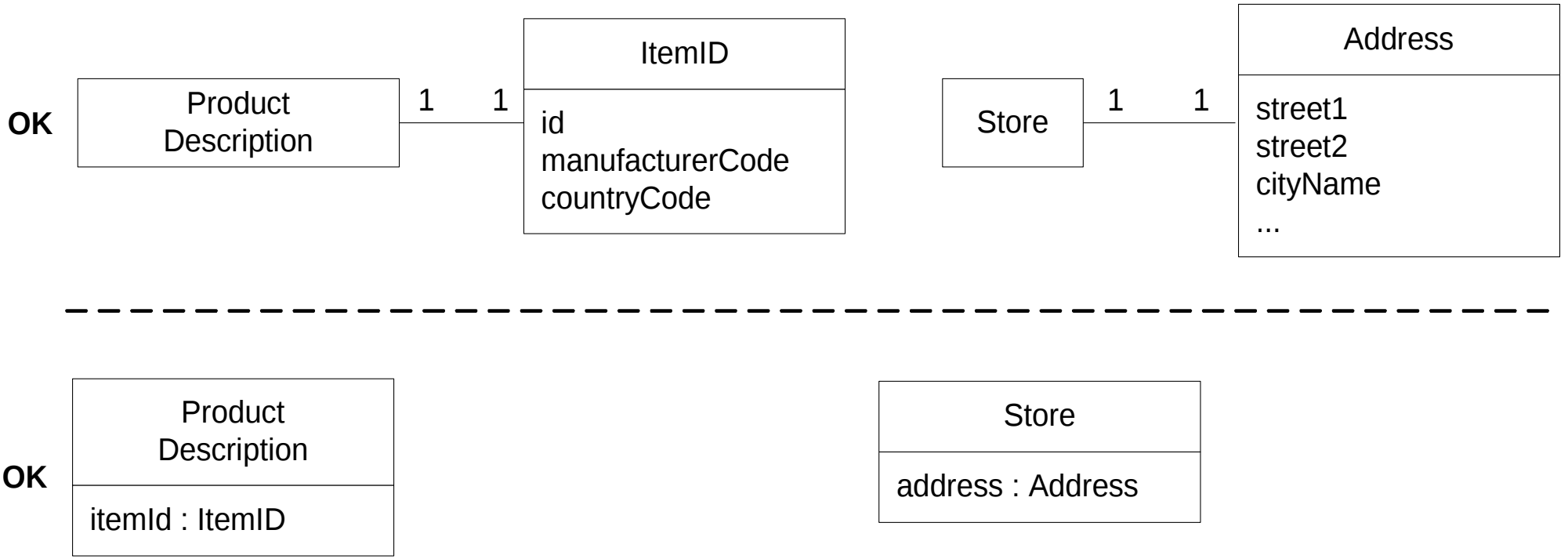
Worse

Associations should not have attributes; if one does have, then chances are it is an object



Better

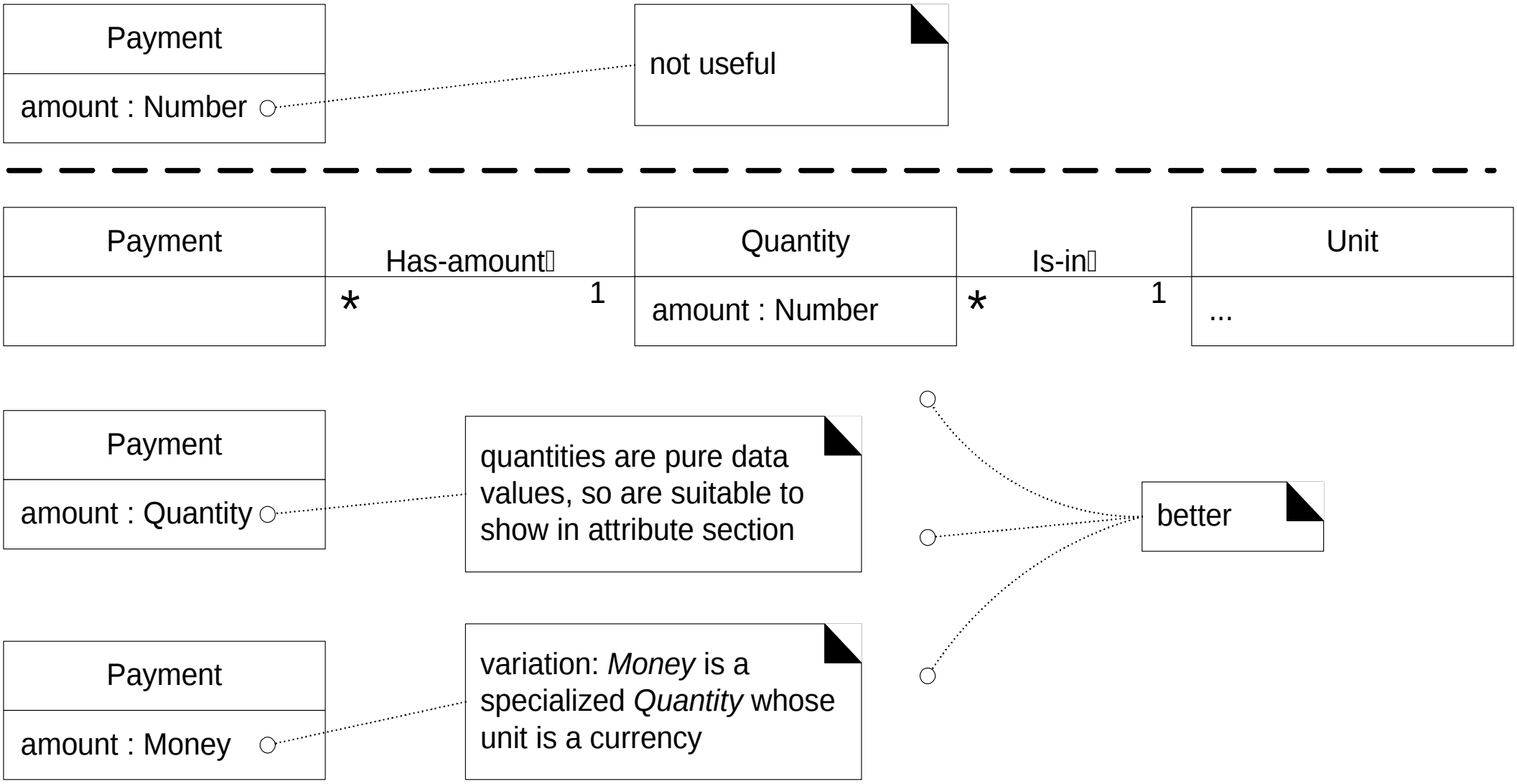
Getting more concepts



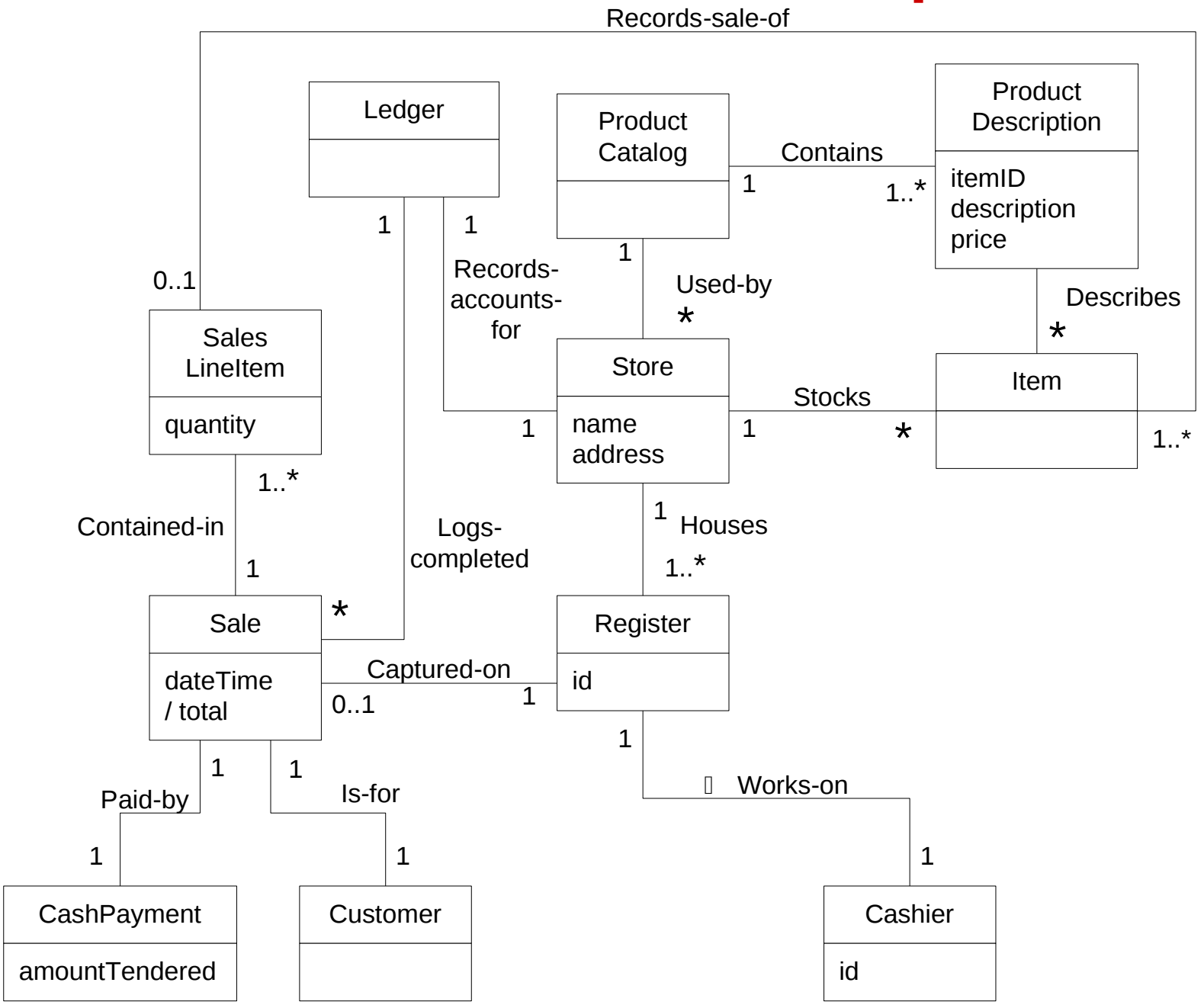
New objects are conditional to the relevance to the model

Also, composite data types may be objects

Getting more concepts



NextGen example



Domain Model v/s Design Model

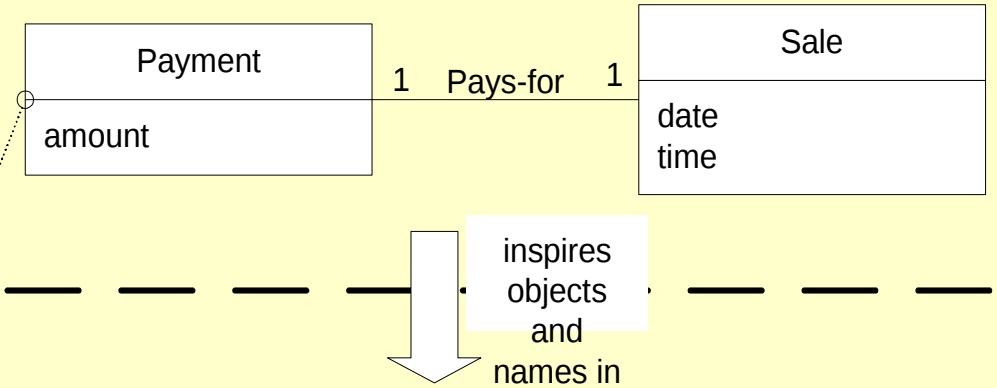
A Payment in the Domain Model is a concept, but a Payment in the Design Model is a software class. They are not the same thing, but the former *inspired* the naming and definition of the latter.

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.



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.

Design Model (later)

- We will use a detailed Class Diagram to describe the Design Model
- We will evolve and infer OO classes from the concepts
- We might drop concepts as well
- Other classes will come from system needs
- The associations will be much more specific
- Code can be partially inferred from Design Model

Thoughts

- No single correct answer—*no exact science either!*
- It might take to become familiar with the ideas
- It may be good to start with a simple domain model
- Then, check what is wrong—fix
- Fixing may add components
- Iterate until good quality
- Don't connect all to all!

Credits

Notes and figures adapted from

Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development by C. Larman. 3rd edition. Prentice Hall/Pearson, 2005.