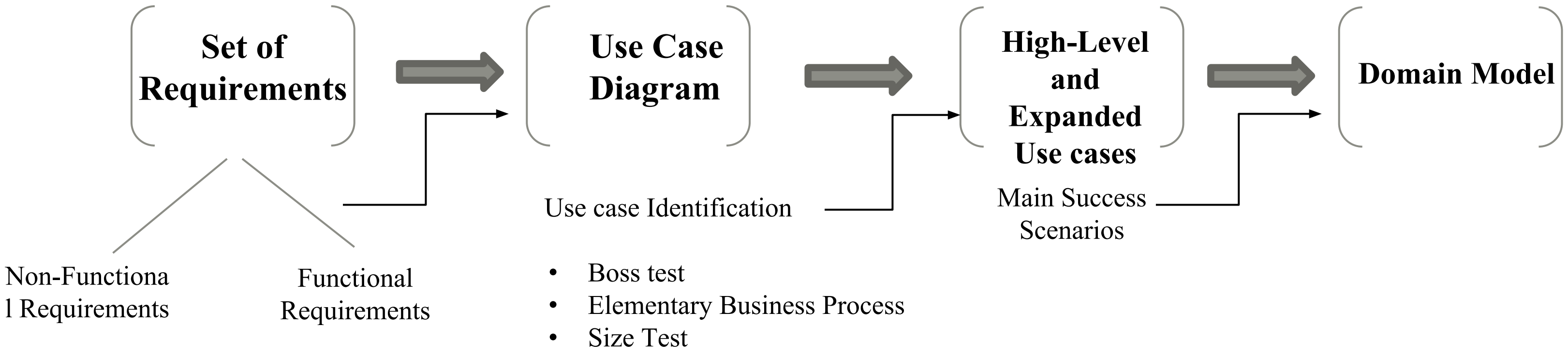
UML MODELLING

Instructor: Mehroze Khan

Outline

- What is Domain Model?
- Why Domain Model?
- Syntax of Domain Model
- Examples on Case Study

Revision up till now



- Boss test
- Elementary Business Process
- Size Test

Domain Model

- A **Domain Model** is a conceptual model that represents the key **concepts**, **entities**, and **relationships** within a specific problem domain.
- The domain model is created during object-oriented **analysis** to decompose the domain into **concepts** or **objects** in the real world
- The model should identify the set of **conceptual classes** (The domain model is iteratively completed)
- It is the basis for the design of the software
- The domain model is also called **conceptual model**, **domain object model** or **analysis object model**

Domain Model

- Illustrates meaningful **conceptual classes** in problem domain (Analysis phase)

- It helps us identify, relate and visualize important information.

- Represents **real-world concepts**, not software components

- Software-oriented class diagrams will be developed later (during design phase)

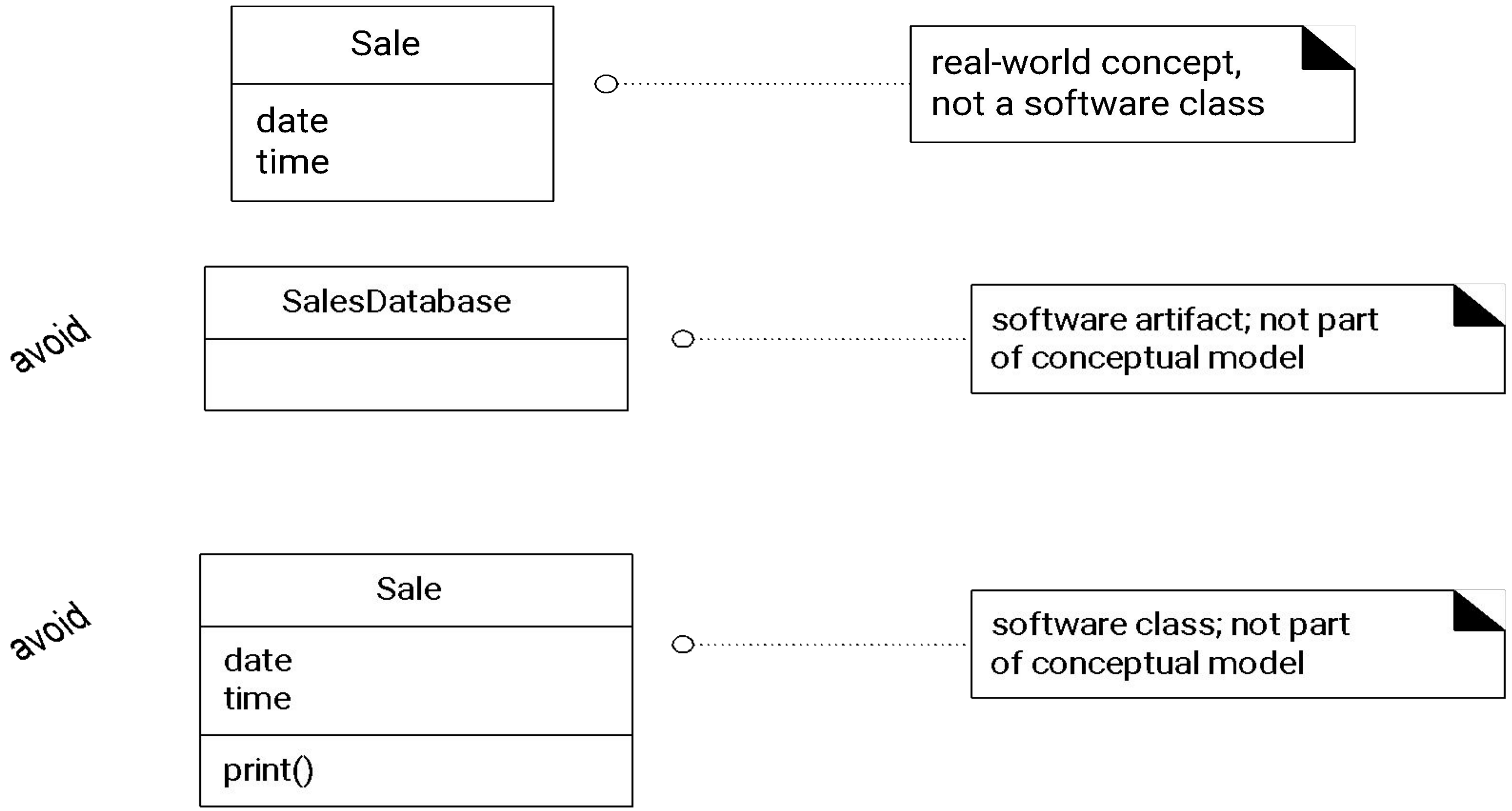
- It provides inspiration for later creation of software design classes, to reduce "representational gap."

To visualize domain models the UML class diagram notation is used. It is also known as **Analysis Class Diagram**

- However, no operations are defined in domain models
- Only …
  - *domain objects and conceptual classes*
  - *associations between them*
  - *attributes of conceptual classes*

Domain models are not models of software design

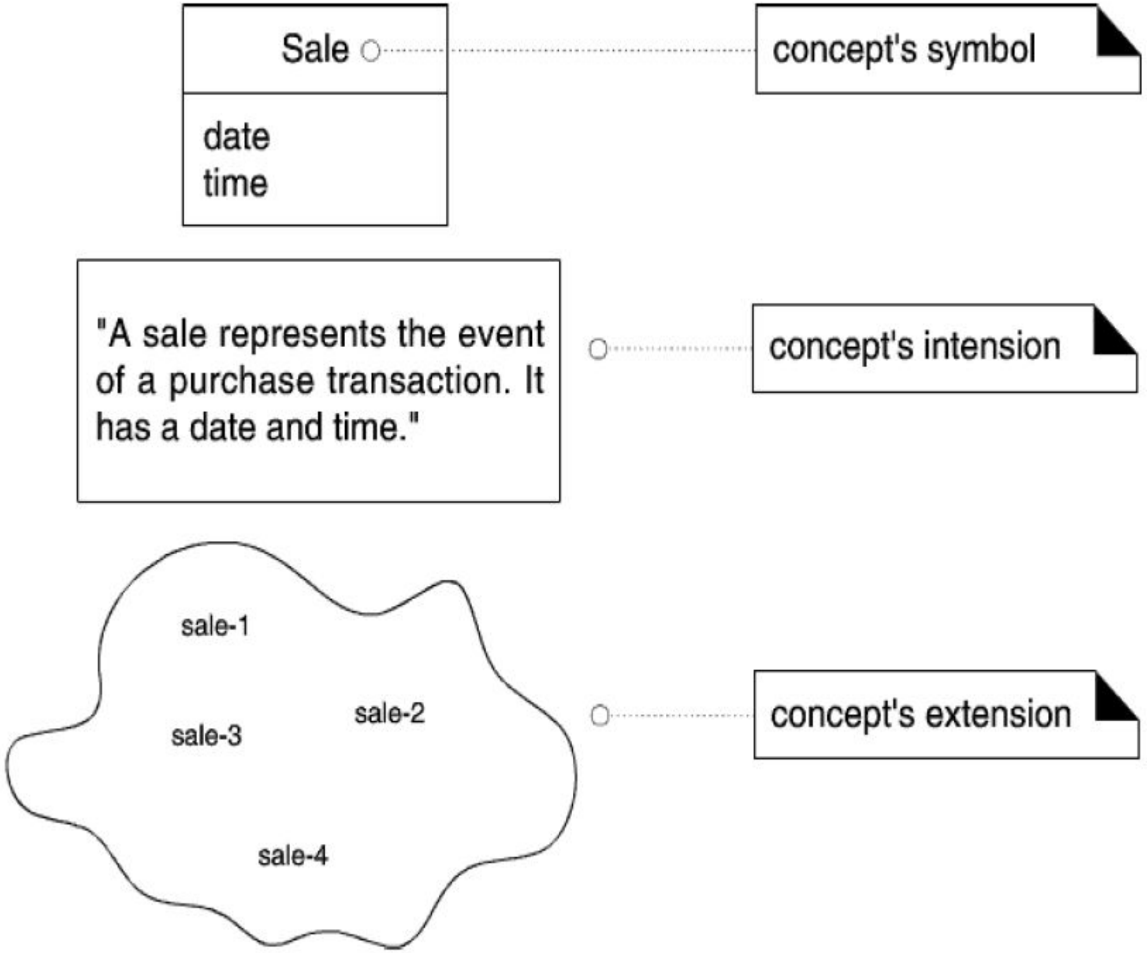
- No responsibilities or methods.



What are Concepts and Conceptual Classes?

Concept: is an idea, thing, or object. It has symbol, intension, and extensions.

- Symbol --words or images representing a conceptual class
- Intention --the definition of a conceptual class
- Extension --the set of examples to which the conceptual class applies.



Sale

date
time

concept's symbol

"A sale represents the event of a purchase transaction. It has a date and time."

concept's intension

concept's extension

WHY DOMAIN MODEL?

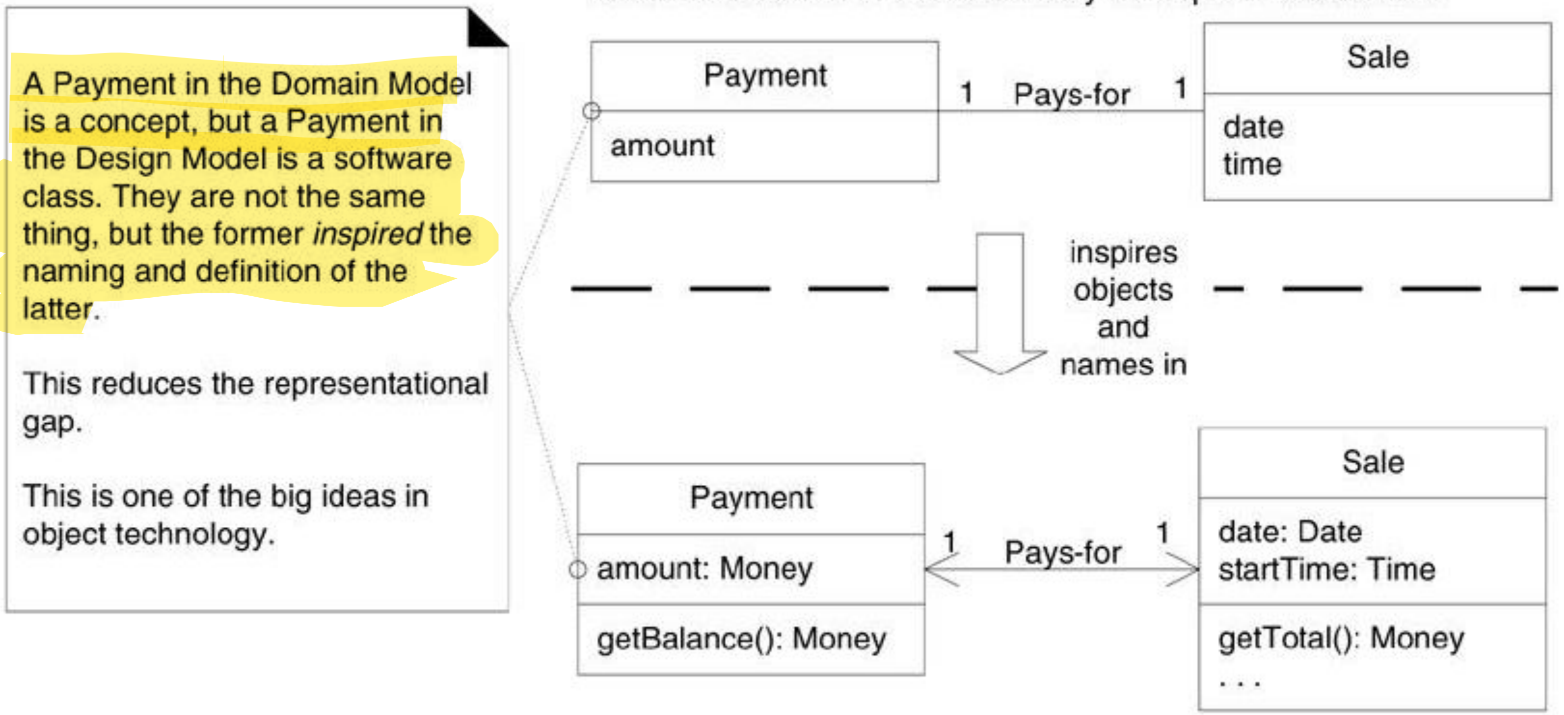
Why do we need a domain model?

- Gives a conceptual framework of the things in the problem space
- Helps you think – focus on semantics
- Provides a glossary of terms – **noun** based
- Based on the defined structure, we can describe the **state** of the problem domain at any time.
- Create a domain model to understand the **key concepts** and **vocabulary**
- **Lower gap** between the software representation and mental model of the domain

Lower representational gap with OO modeling

UP Domain Model

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



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 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.

SYNTAX OF DOMAIN MODEL

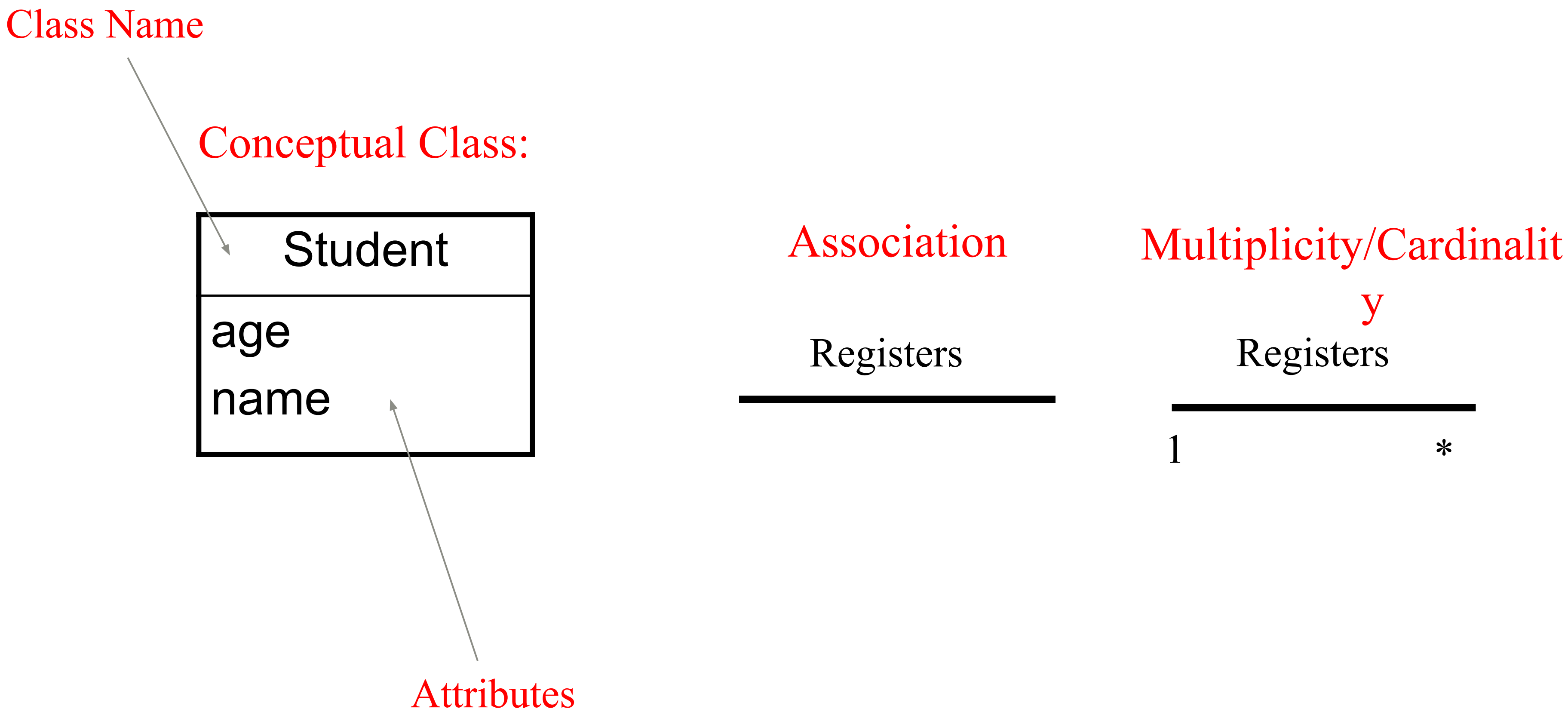
Notation – UML Analysis Class Diagram

- UML analysis class diagram is used to **model a domain**

- Domain Class
  - *Domain class consist of **class name** and **attributes***
  - *Domain class **does not have methods***

- Relation
  - *A line with a **label** represents the relation between two classes*
  - *A relation has a **cardinality constraint***

Syntax of Domain Model



Class Name

Conceptual Class:

Student

age

name

Attributes

Association

Registers

Multiplicity/Cardinality

Registers

1

*

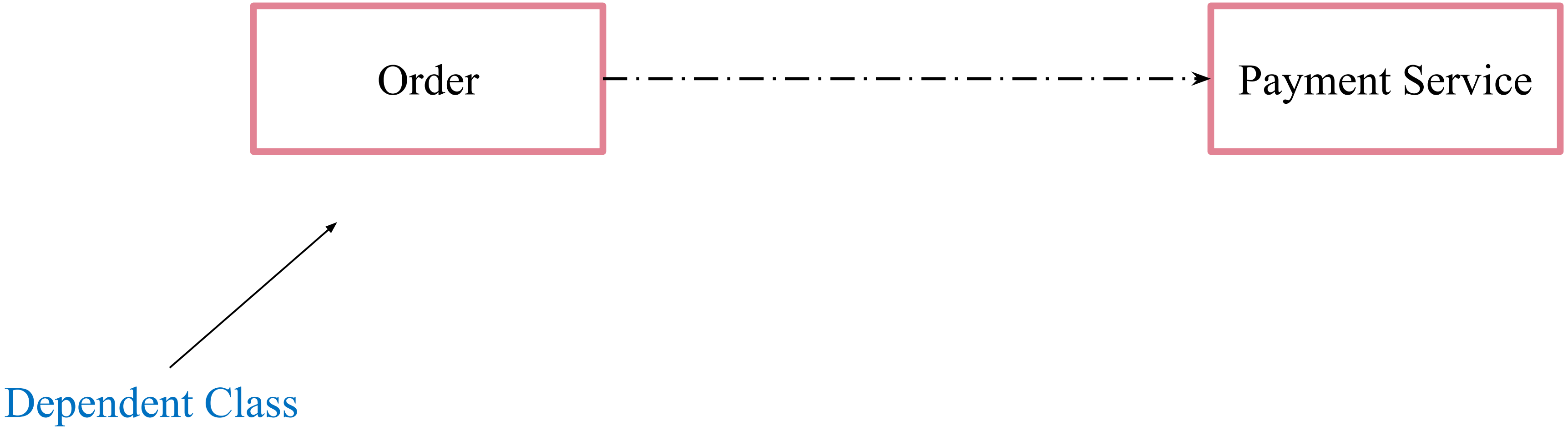
Relationships

- There are mainly four kinds of relationships in UML:

1. Dependency
2. Generalization
3. Association
4. Aggregation/Composition

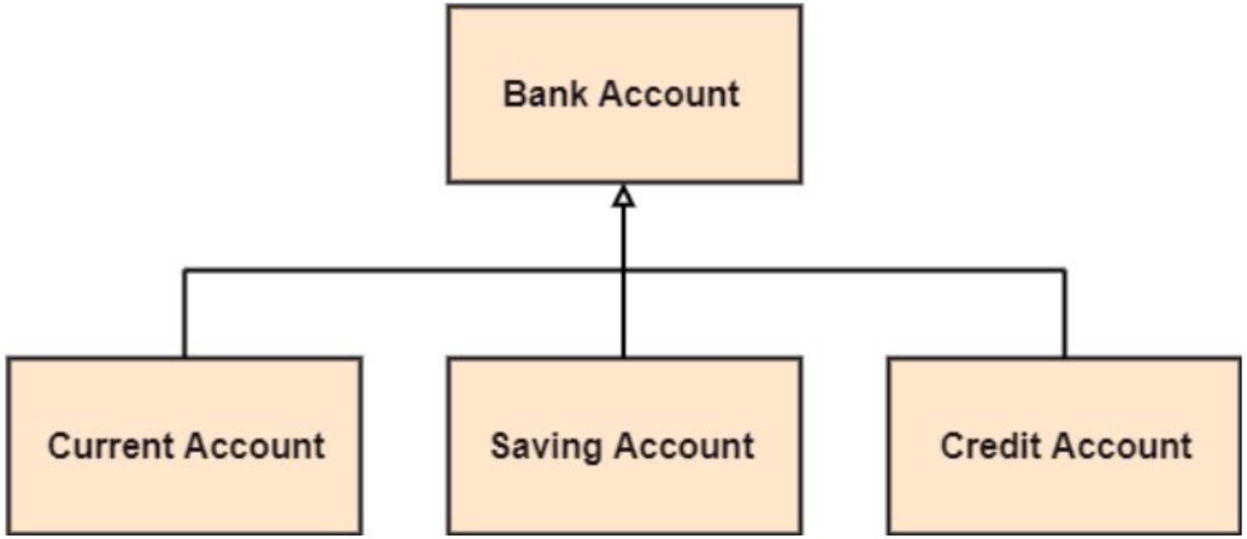
Dependency

- A dependency means the relation between two or more classes in which a change in one may force changes in the other.
- It will always create a **weaker relationship**. Dependency indicates that one class depends on another.
- In the following example, Order has a dependency on Payment Service



Generalization

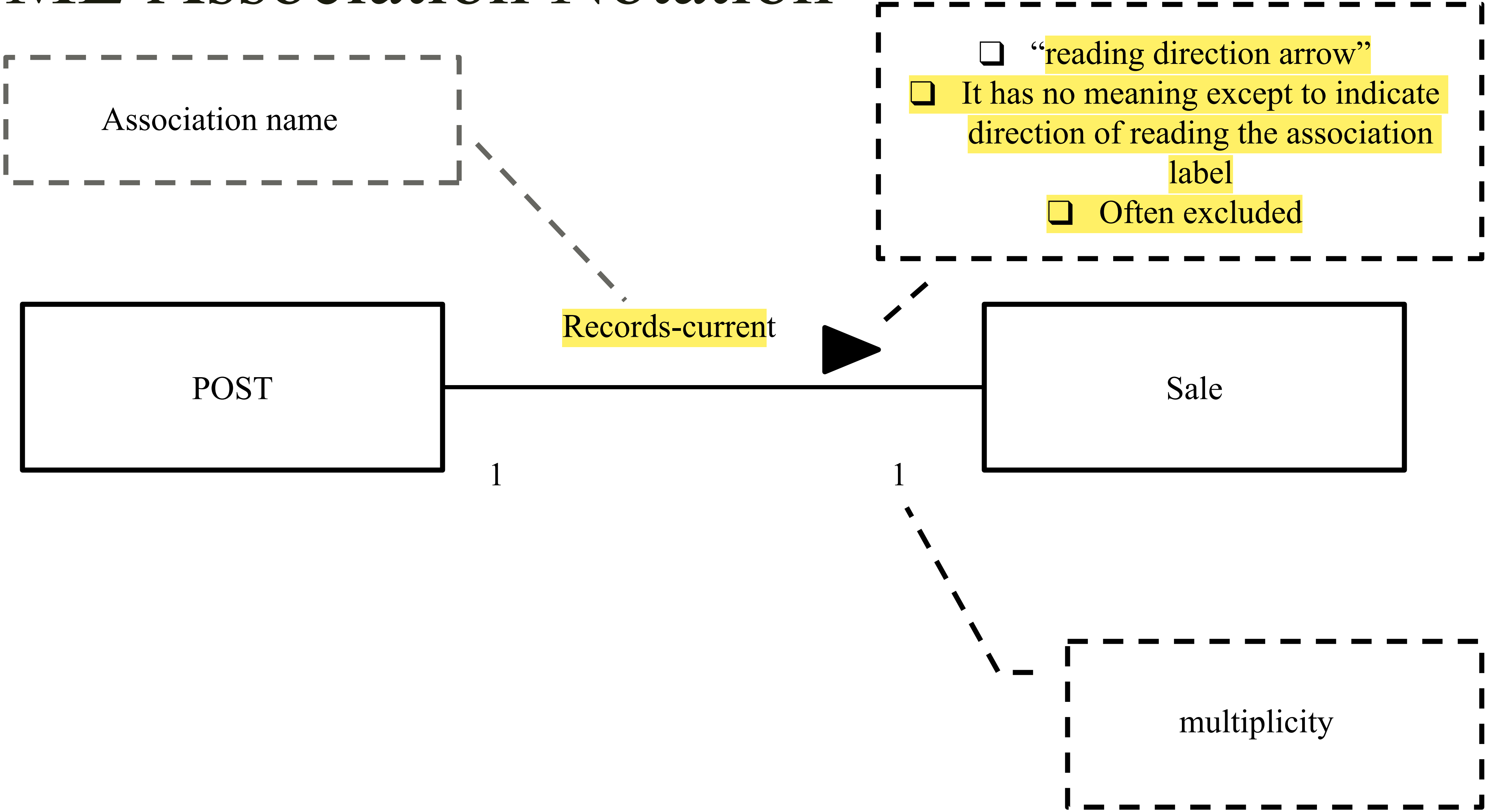
A generalization is a relationship between a parent class (superclass) and a child class (subclass). In this, the child class is inherited from the parent class.
For example, The Current Account, Saving Account, and Credit Account are the specialized form of Bank Account.



Association

- An association is represented as a line between classes with an association name.
- Associations are inherently bidirectional.
- Optional reading direction arrow is only an aid to the reader of the diagram.

UML Association Notation



Association name

Records-current

POST

Sale

1

1

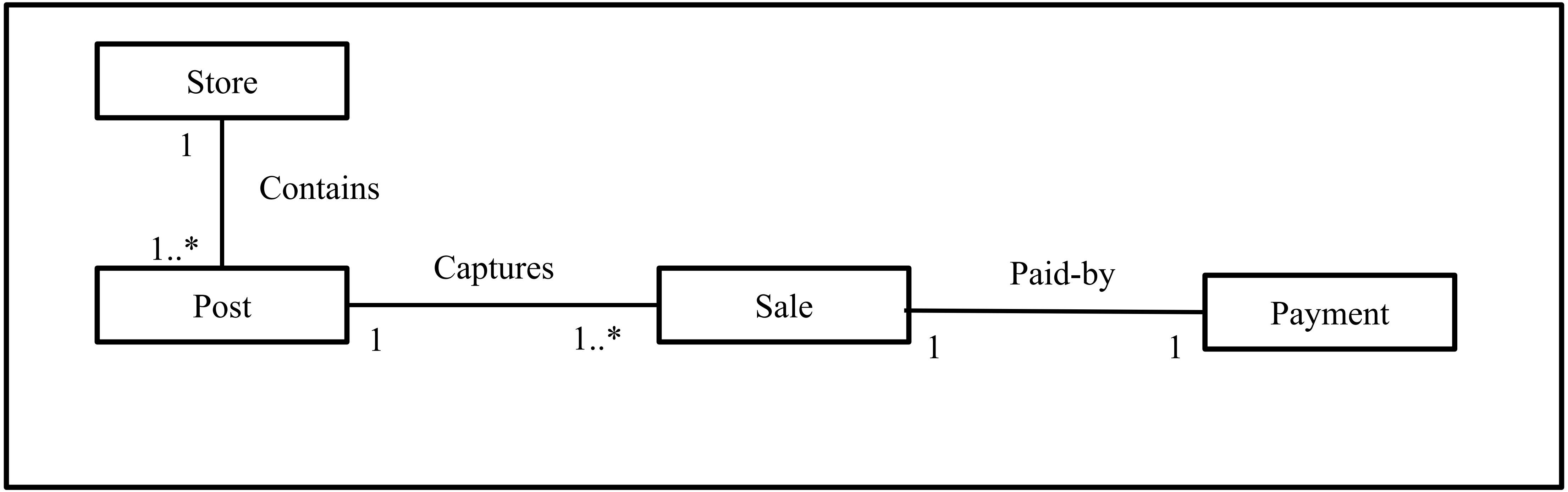
❑ “reading direction arrow”

❑ It has no meaning except to indicate direction of reading the association label

❑ Often excluded

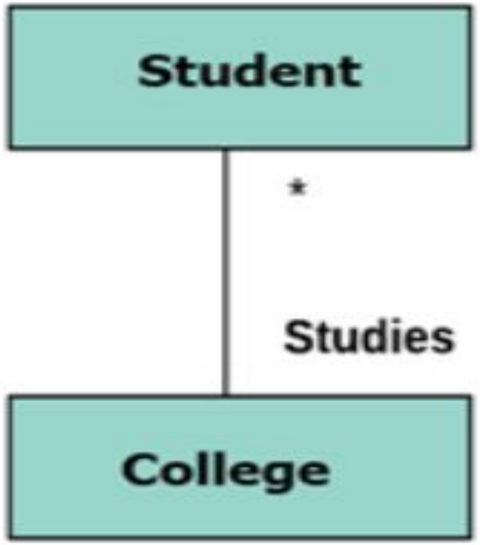
multiplicity

Associations Names

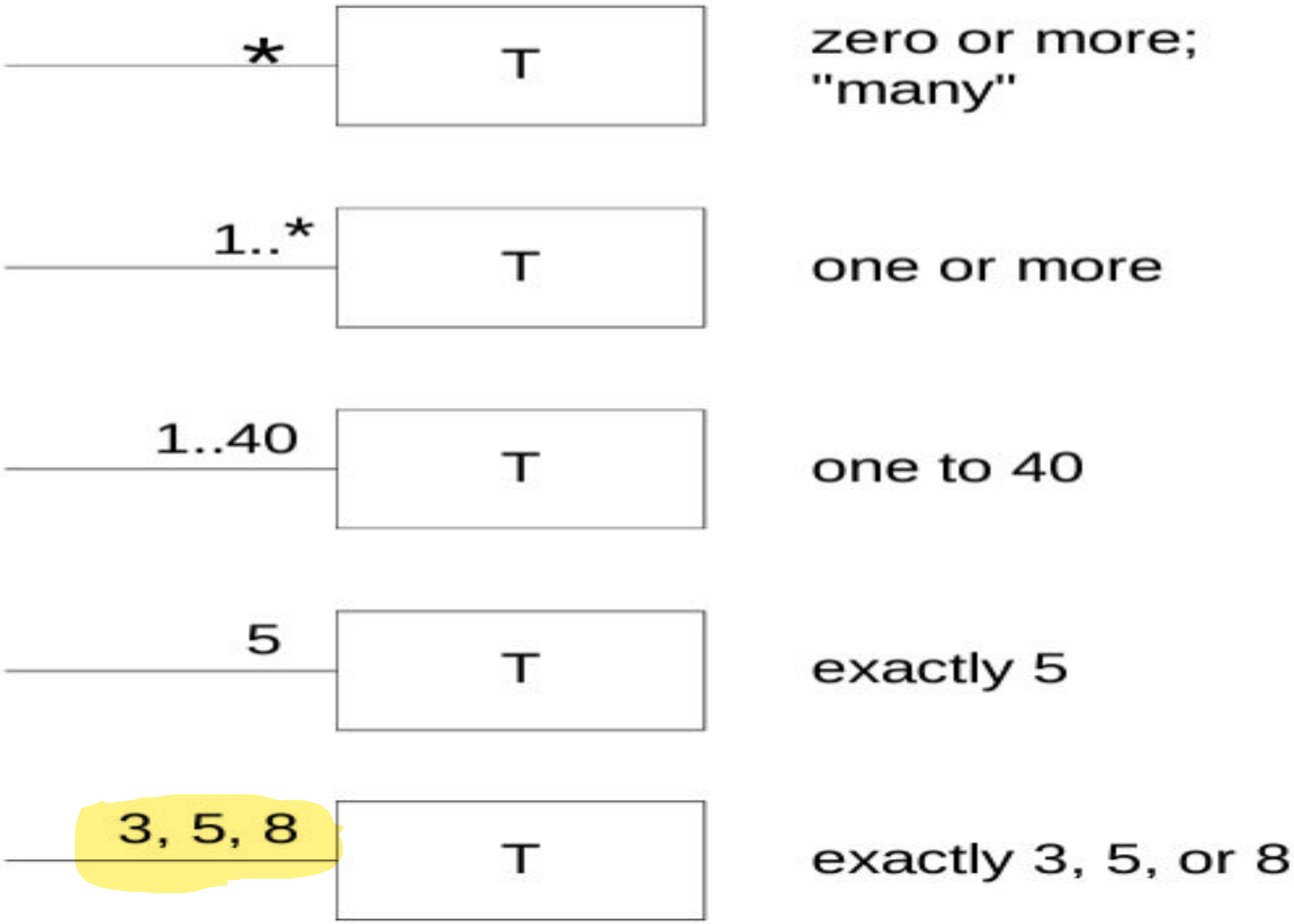


Multiplicity

- A multiplicity is a factor associated with an attribute. It specifies how many instances of attributes are created when a class is initialized.
- If a multiplicity is not specified, by default one is considered as a default multiplicity.
- Let's say that that there are 100 students in one college. The college can have multiple students.



Multiplicity

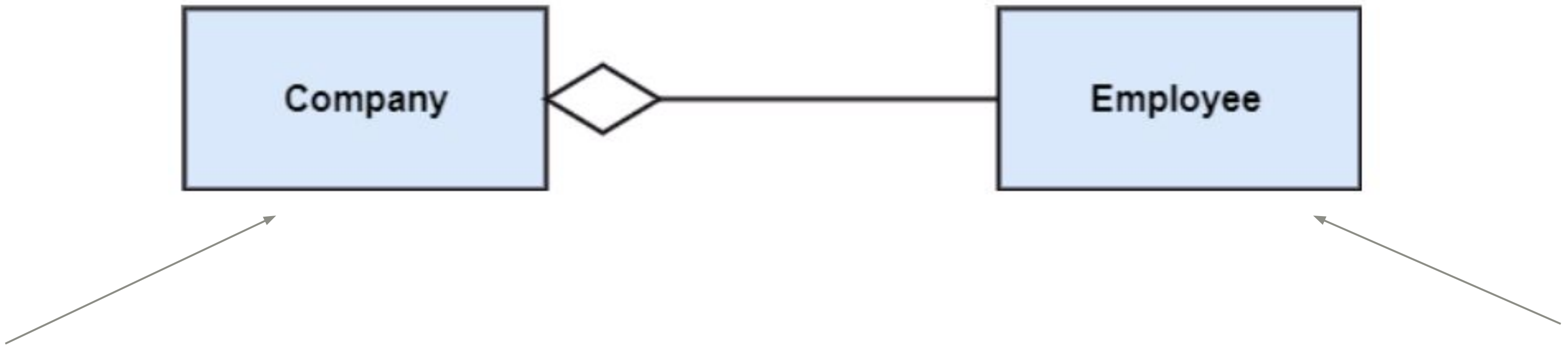


Aggregation/Composition

- They both represent a **"whole-part"** relationship between classes, but they differ in the **degree of ownership** and **lifecycle management**.

Aggregation

- Aggregation represents a "**has-a**" relationship where the part (child) can exist **independently** of the whole (parent).
- The parent does not own the child objects. This relationship is typically implemented using references or pointers.

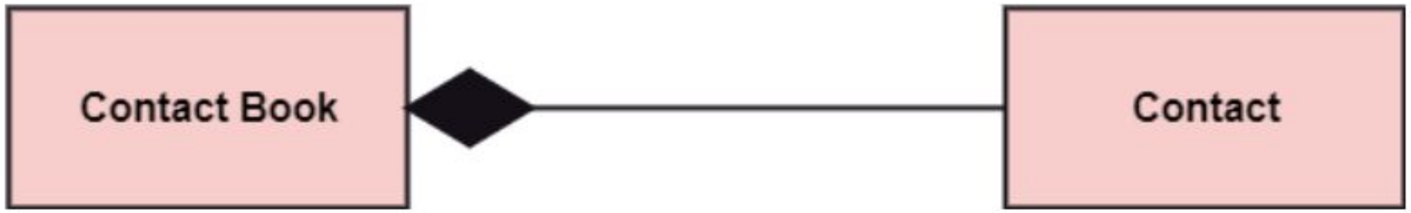


Container Class

Contained Class

Composition

- Composition also represents a "**has-a**" relationship, but with stronger ownership.
- In composition, the parent object owns the child objects, and the child objects **cannot exist independently** of the parent. When the parent is destroyed, the child objects are also destroyed.



HOW TO CREATE DOMAIN MODEL

How to Create Domain Model

- Find conceptual classes
- Draw them as classes in a UML class diagram
- Add associations and attributes

Finding Conceptual Classes

Three Strategies to Find Conceptual Classes

1. Reuse or modify the existing model if one exists
2. Use a Category List
3. Identify noun phrases in your use-cases

Method 1: Reuse or Modify Existing Models

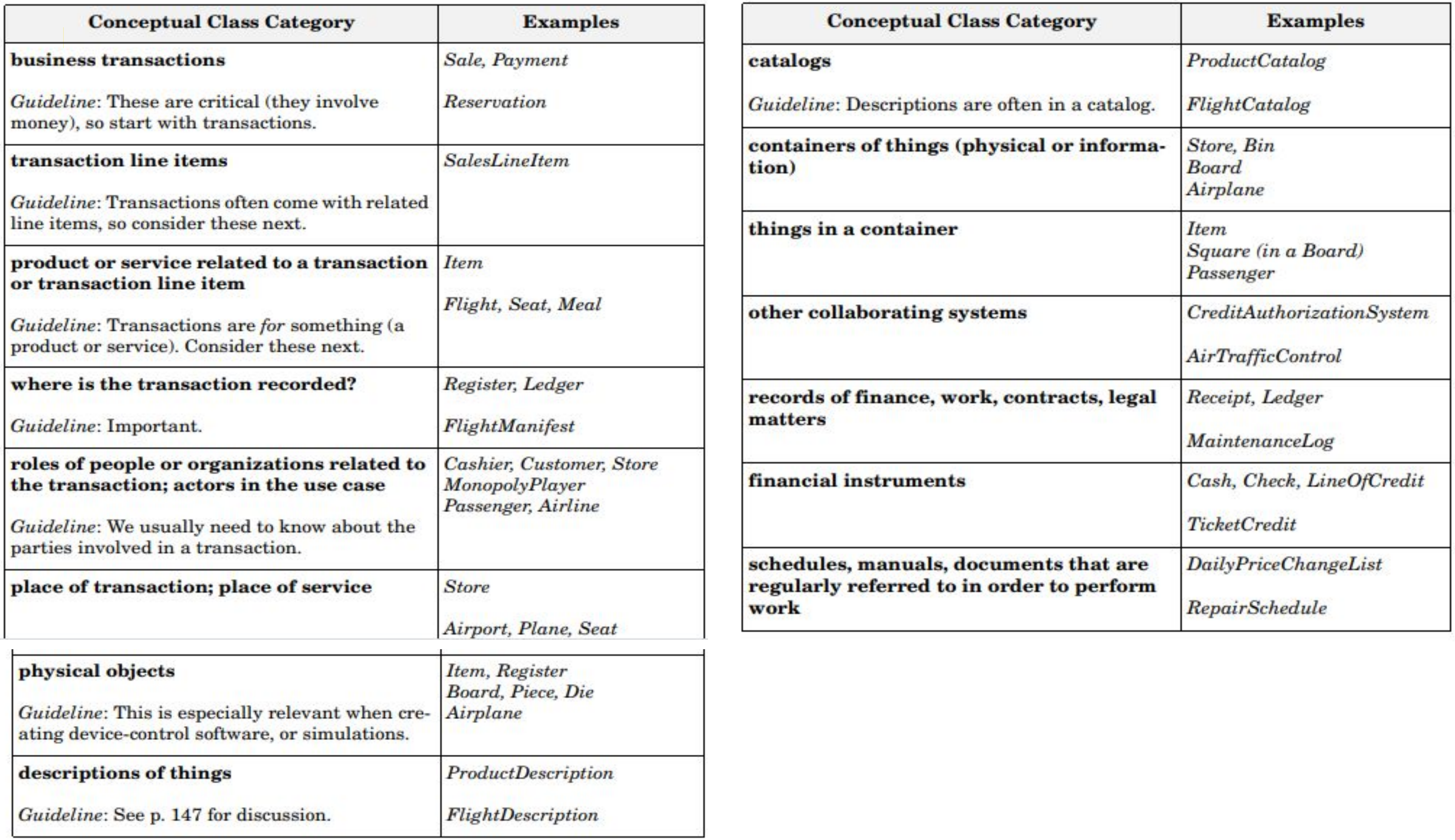
- There are published, well--‑crafted domain models and data models (which can be modified into domain models) for many common domains, such as inventory, finance, health, and so forth..

- Reusing existing models is excellent, but out of the scope of this course

Method 2: Use a Category List

- We can kick-start the creation of a domain model by making a list of candidate conceptual classes.
- Table contains many common categories that are usually worth considering, with an emphasis on business information system needs.
- The guidelines also suggest some priorities in the analysis. Examples are drawn from the
  1) *POS*
  2) *Monopoly*
  3) *Airline reservation domains*

imp



Conceptual Class Category	Examples
business transactions *Guideline*: These are critical (they involve money), so start with transactions.	*Sale, Payment* *Reservation*
transaction line items *Guideline*: Transactions often come with related line items, so consider these next.	*SalesLineItem*
product or service related to a transaction or transaction line item *Guideline*: Transactions are *for* something (a product or service). Consider these next.	*Item* *Flight, Seat, Meal*
where is the transaction recorded? *Guideline*: Important.	*Register, Ledger* *FlightManifest*
roles of people or organizations related to the transaction; actors in the use case *Guideline*: We usually need to know about the parties involved in a transaction.	*Cashier, Customer, Store* *MonopolyPlayer* *Passenger, Airline*
place of transaction; place of service	*Store* *Airport, Plane, Seat*
physical objects *Guideline*: This is especially relevant when creating device-control software, or simulations.	*Item, Register* *Board, Piece, Die* *Airplane*
descriptions of things *Guideline*: See p. 147 for discussion.	*ProductDescription* *FlightDescription*
catalogs *Guideline*: Descriptions are often in a catalog.	*ProductCatalog* *FlightCatalog*
containers of things (physical or information)	*Store, Bin* *Board* *Airplane*
things in a container	*Item* *Square (in a Board)* *Passenger*
other collaborating systems	*CreditAuthorizationSystem* *AirTrafficControl*
records of finance, work, contracts, legal matters	*Receipt, Ledger* *MaintenanceLog*
financial instruments	*Cash, Check, LineOfCredit* *TicketCredit*
schedules, manuals, documents that are regularly referred to in order to perform work	*DailyPriceChangeList* *RepairSchedule*

Method 3: Finding Conceptual Classes with Noun Phrase Identification

- Another useful technique (because of its simplicity) is linguistic analysis: **Identify the nouns and noun phrases** in textual descriptions **(use cases or other documents)** of a domain and consider them as candidate conceptual classes or attributes.

Visualizing Domain Models

Archaeologist Management System

You have been asked to build a management system for a group of archeologists. The group is comprised of multiple teams. Each team consists of researchers. Each team has a letter ID (e.g., team A, team B). Each researcher has an ID number, a first name, and a last name. There are two types of researchers: field staff and lab staff. Each field staff member has a favorite region (string). A lab staff supports up to 2 field staff. Some researchers may not be supported by a lab staff. The archaeologist group also manages an inventory of equipment. There are many pieces of equipment in the inventory. Researchers of any type may check out up to 3 pieces of equipment. Each piece of equipment has a serial number and replacement cost.

Archaeologist Management System

Identify Nouns

You have been asked to build a management system for a group of archeologists. The group is comprised of multiple teams. Each team consists of researchers. Each team has a letter ID (e.g., team A, team B). Each researcher has an ID number, a first name, and a last name. There are two types of researchers: field staff and lab staff. Each field staff member has a favorite region (string). A lab staff supports up to 2 field staff. Some researchers may not be supported by a lab staff. The archaeologist group also manages an inventory of equipment. There are many pieces of equipment in the inventory. Researchers of any type may check out up to 3 pieces of equipment. Each piece of equipment has a serial number and replacement cost.

Archaeologist Management System

Identify Associations and Multiplicity

You have been asked to build a management system for a group of archeologists. The group is comprised of multiple teams. Each team consists of researchers. Each team has a letter ID (e.g., team A, team B). Each researcher has an ID number, a first name, and a last name. There are two types of researchers: field staff and lab staff. Each field staff member has a favorite region (string). A lab staff supports up to 2 field staff. Some researchers may not be supported by a lab staff. The archaeologist group also manages an inventory of equipment. There are many pieces of equipment in the inventory. Researchers of any type may check out up to 3 pieces of equipment. Each piece of equipment has a serial number and replacement cost.