Software Design and Analysis
CS-3004
Lecture#05

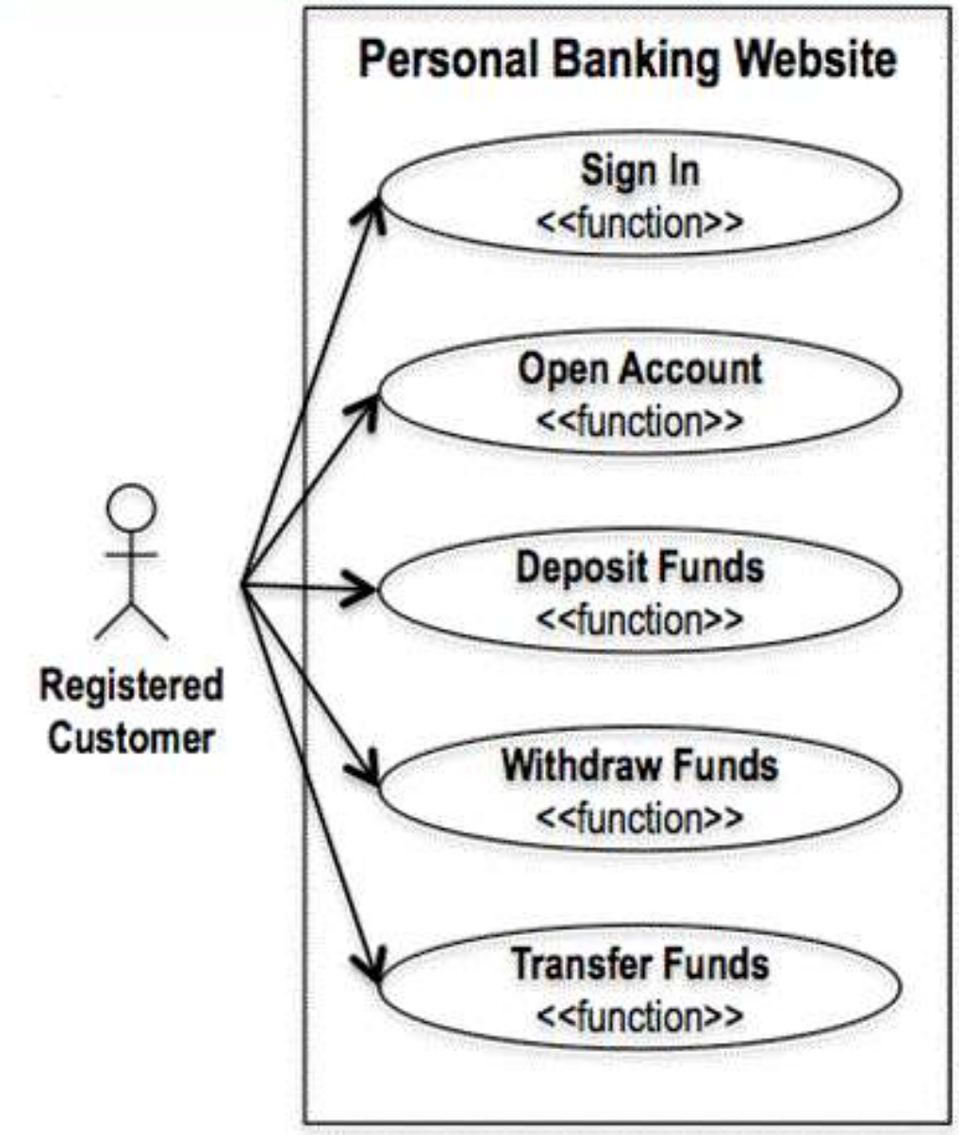
Dr. Javaria Imtiaz,
Mr. Basharat Hussain,
Mr. Majid Hussain

Recap

- What is the source of developing Use case diagram?
- Why we create a Use case diagram?
- What is a use case? Any example from hospital electronic medical record systems domain?
- What is the next step after creating use case diagram?
- How many ways are available in UML for specifying use cases?
- What is the pre and post conditions ? Provide an example for Add an item to Cart
- What is main success scenario?
- What is alternative scenario?
- How many formats are available for writing main success scenario? And which is preferable?

What the system will ensure is true before letting the use case start

Use case diagram



<<function>> (short for "application function") is the stereotype for a use case initiated by a person

Use case preconditions

None

1. User is signed in

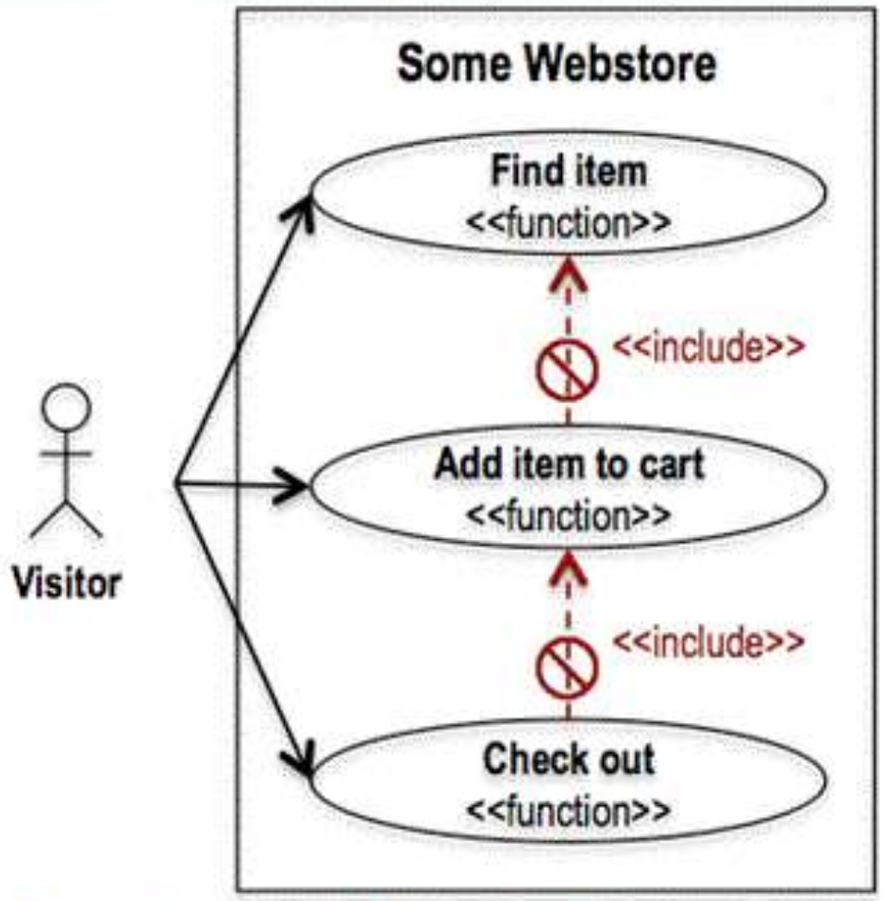
1. User is signed in
2. User has at least **one** active account

1. User is signed in
2. User has at least **one** active account
3. At least one of those accounts has a positive balance

1. User is signed in
2. User has at least **two** active accounts
3. At least one of those accounts has a positive balance

Preconditions are defined in a use case specification, which represents a use case's "method" or "behavior"

Partial use case diagram



Do not represent "sequential" dependencies between use cases through include relationships, which are "functional" dependencies

Use case preconditions

None

1. Item has been identified

Contributes to

1. Item has been identified

Contributes to

1. Cart contains one or more items

Use case postconditions

1. Item has been identified

1. Item has been added to cart

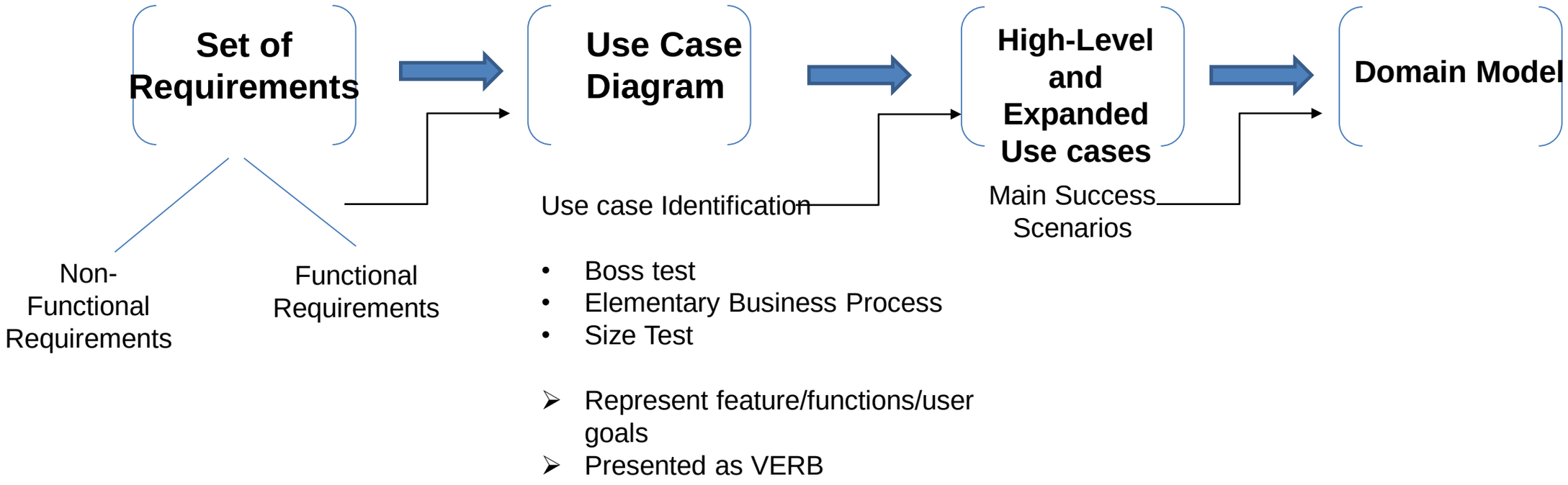
1. Order is paid
2. Cart is empty

Do represent "sequential" dependencies between use cases through postconditions and preconditions

Outline

- What is Domain Model?
- Why Domain Model?
- Features of Domain Model
- Syntax of Domain Model
- Examples on Various Case Studies

Revision up till now



- Boss test
- Elementary Business Process
- Size Test

- ➢ Represent feature/functions/user goals
- ➢ Presented as VERB

Domain Model

- Why:
  Domain modeling helps us to identify the relevant concepts and ideas of a domain

- When:
  Domain modeling is done during object-oriented analysis

- Guideline:
  Only create a domain model for the tasks at hand

“Good designs require deep application knowledge.”

Curtis’law

Domain Model

- The Domain Model illustrates noteworthy concepts in a 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.)
- Also known as Visual Dictionary
- It is the basis for the design of the software.
- The domain model is also called **conceptual model**, **domain object model** or **analysis object mode.**
  - **conceptual is idea, thing, object**

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.”

Domain Model

- **Domain Modeling** is useful to understand the ideas and concepts of the domain and their interrelationships

- A **Domain Model** is usually created at the beginning of a project and is a basis for the design model

- A **Domain Model** is created using a subset of the UML class diagram notation

To visualize domain models the UML class diagram notation is used.

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

A Domain Model is Conceptual, not a Software Artifact

Conceptual Class:

Sale
amt item

vs.

Software Artifacts:

SalesDatabase

Sale
Double amt; Item item;
void print()

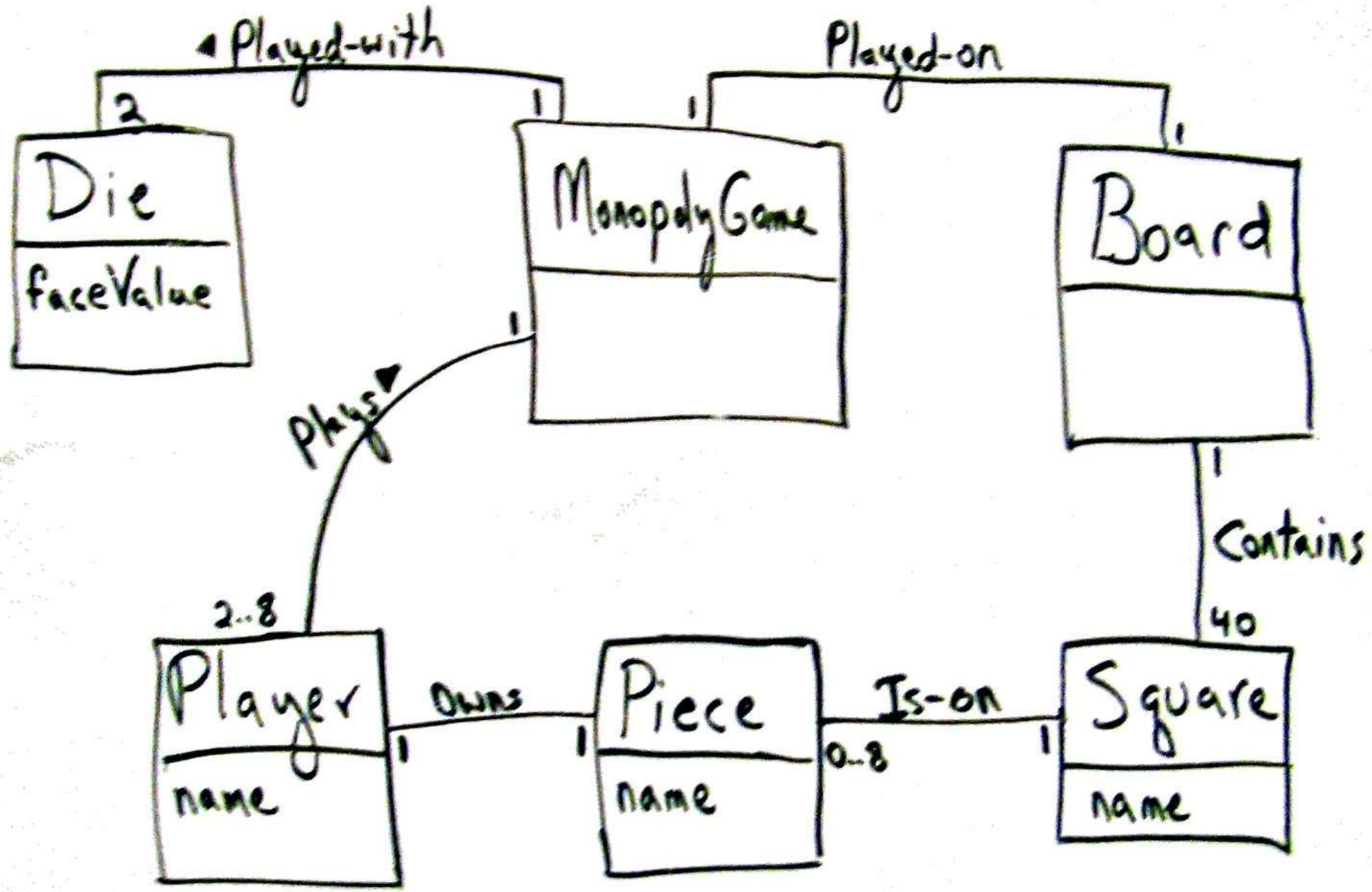
What's the difference?

Monopoly Game domain model (first identify concepts as classes)

Monopoly Game	Dice	Board

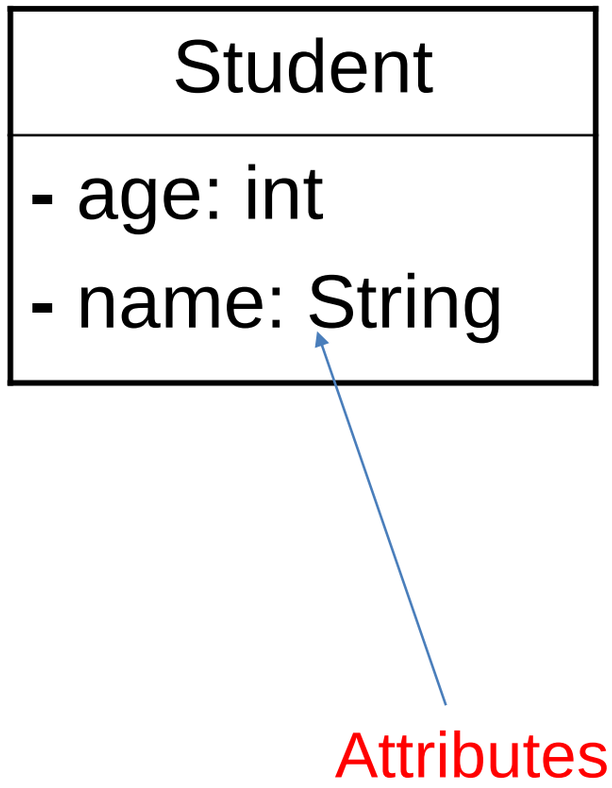
Player	Piece	Square

Monopoly Game domain model
Larman, Figure 9.28



Syntax of Domain Model

Conceptual Class:



Association

Registers

Cardinality

Registers

1 *

How to Create Domain Model

1. Find conceptual classes
2. Draw them as conceptual classes
3. Add associations and attributes

Finding Conceptual Classes

1. Reuse or modify the existing model if one exists
   - There are many published, well-crafted domain models.

2. Use Conceptual Category List

3. Identify noun phrases in your use-cases
   - Identify nouns and phrases in textual descriptions of a domain (use cases, or other documents)

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..

- Knowledge sources are:
  - Interviews
  - Questionnaires
  - Mission Statements
  - Subject Matter Experts (SMEs)

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

How to create the domain model?
Use a category list to find the conceptual classes.

Conceptual Class Category	Classes (for the POS system)
Business transactions...	Sale, Payment
Transaction line items...	SalesLineItem
Product or service related to a transaction or transaction line item.	Item
Where is the transaction recorded?	Register
Roles of people or organizations related to the transaction; actors in use cases.	Cashier, Customer, Store
Place of transactions.	Store
Noteworthy events, often with a time or place that needs to be remembered.	Sale, Payment
...	...

Identify noun phrases to find the conceptual classes

- Identify the nouns and noun phrases in textual descriptions of a domain and consider them as candidate conceptual classes or attributes.

"Use Cases" are also an excellent source for identifying conceptual classes.

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).

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).

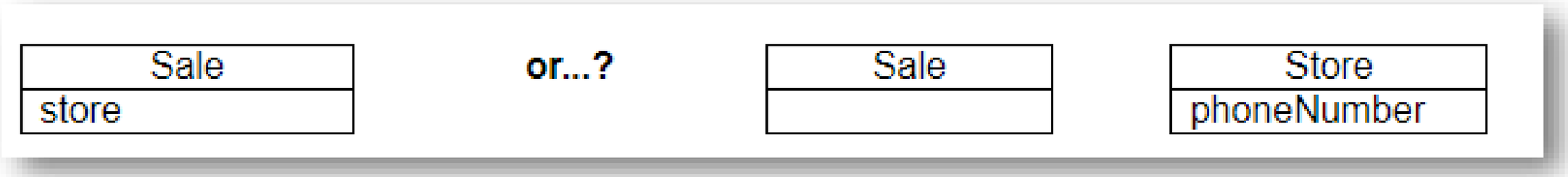
List and Draw UML Diagram

1. Sale
2. Cash Payment
3. SaleLineItem
4. Item
5. Register
6. Amount
7. Cashier

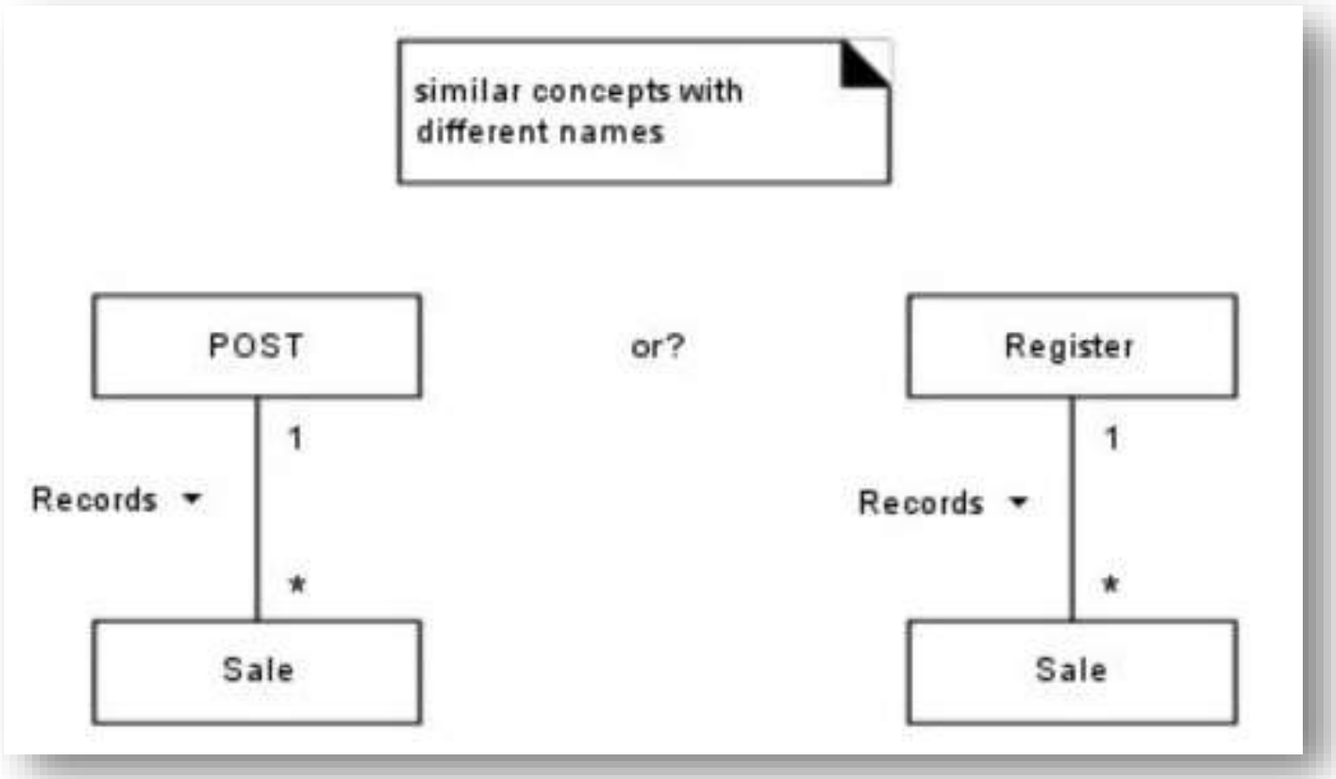
1. Decide which ones are classes and which ones are attributes
2. Add attributes and relation to the identified domain classes

Common Mistake in Identifying Conceptual Classes

1. Most common mistake - to represent something as an attribute when it should have been a concept.



2. Resolving Similar Conceptual Classes



Cont..

3. Modeling the Unreal World
   - Some software systems are for domains that find very little analogy in natural or business domains
     - e.g. telecommunications software
   - Can create a domain model in these domains
     - requires a high degree of abstraction
     - e.g. candidate conceptual classes related to a telecommunication switch: Message, Connection, Port, Dialog, Route, Protocol.

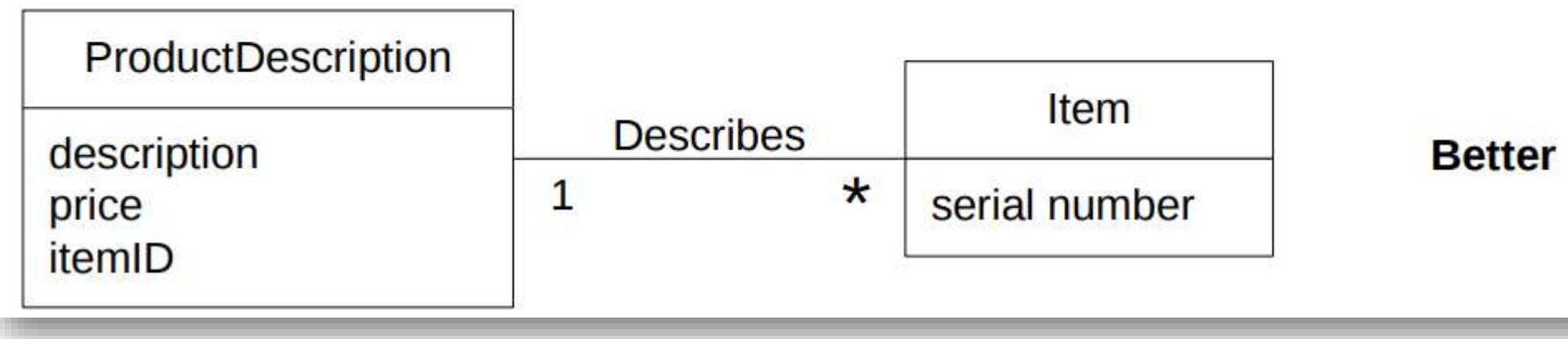
Guidelines when to include a description class into the domain model

- A description class contains information that describes something else
- Examples:
  - a product description records the price, picture and text of an item
  - a flight description contains information about the flight (number) and the source and target destinations

Guidelines when to include a description class into the domain model

- A description class should be added to the domain model when:
  - There needs to be a description about an item or service, independent of the current existence of any examples of those items or services
  - Deleting instances of things they describe results in a loss of information that needs to be maintained, but was incorrectly associated with the deleted thing
  - It reduces redundant or duplicated information

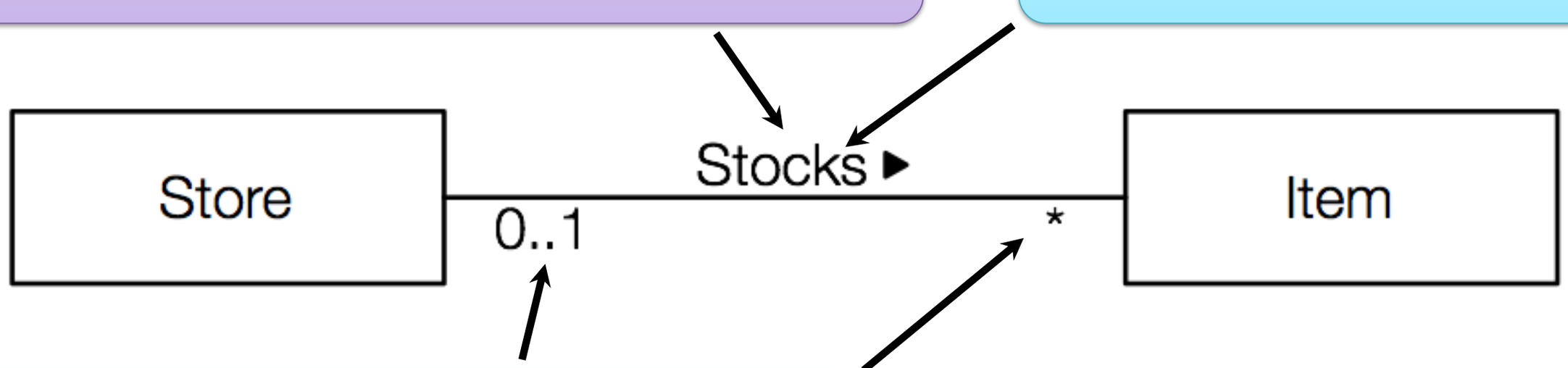
Another foreign key example



Association Notations

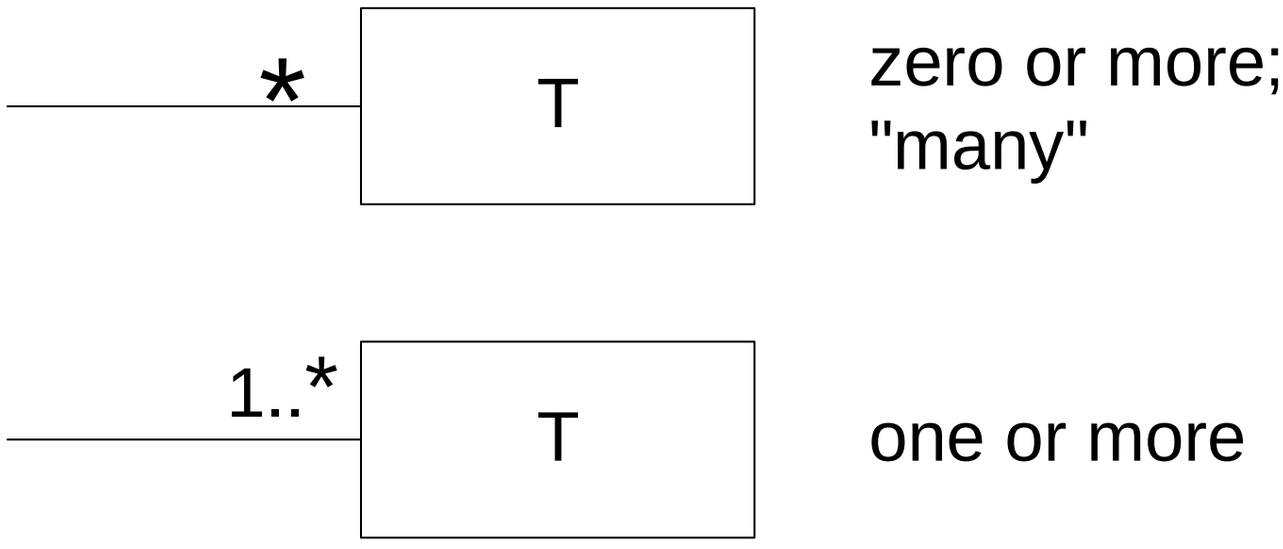
Association name:
- ✓ Use verb phrase
- ✓ Capitalize
- ✓ Typically camel-case or hyphenated
- ✓ Avoid “has”, “use”

Reading direction:
Can exclude if association reads left-to-right or top-to-bottom

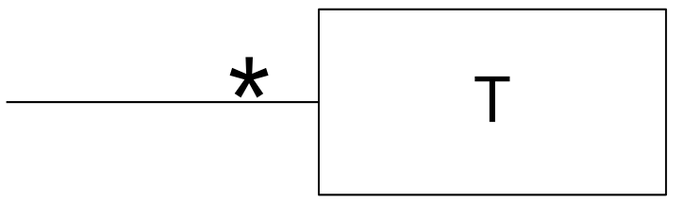


Multiplicity (Cardinality):
- ✓ ‘*’ means “many”
- ✓ x..y means from x to y inclusively

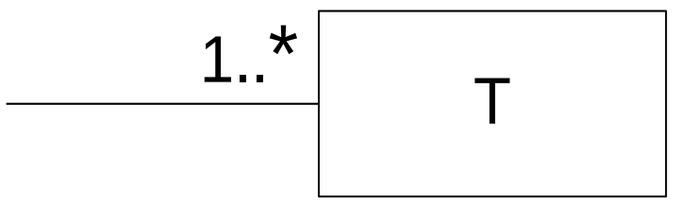
Cardinality (or Multiplicity)



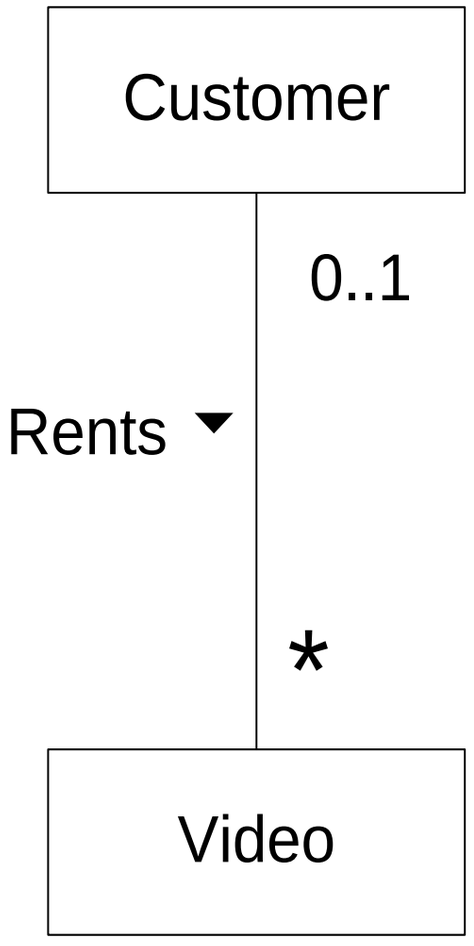
UML: Multiplicity



zero or more;
"many"

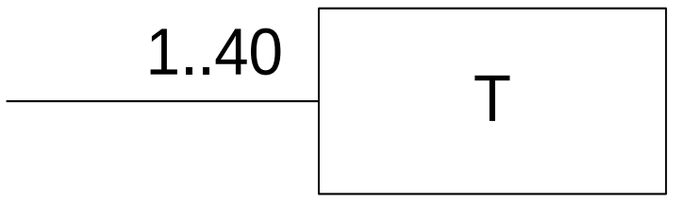


one or more

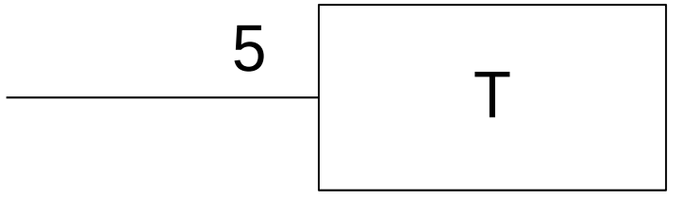


One instance of a Customer may be renting zero or more Videos.

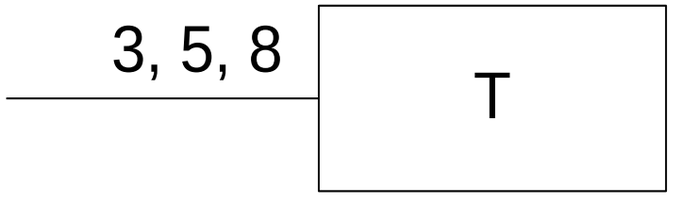
One instance of a Video may be being rented by zero or one Customers.



one to forty

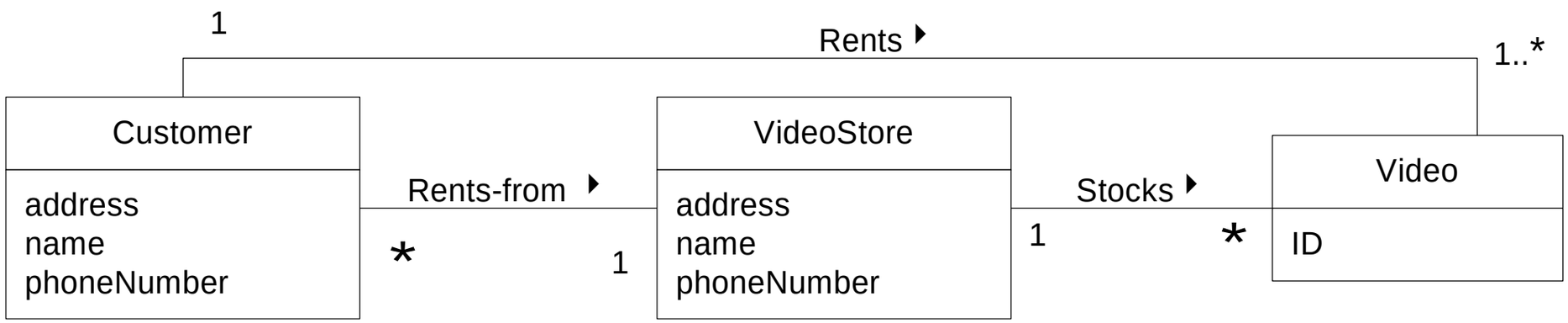


exactly five



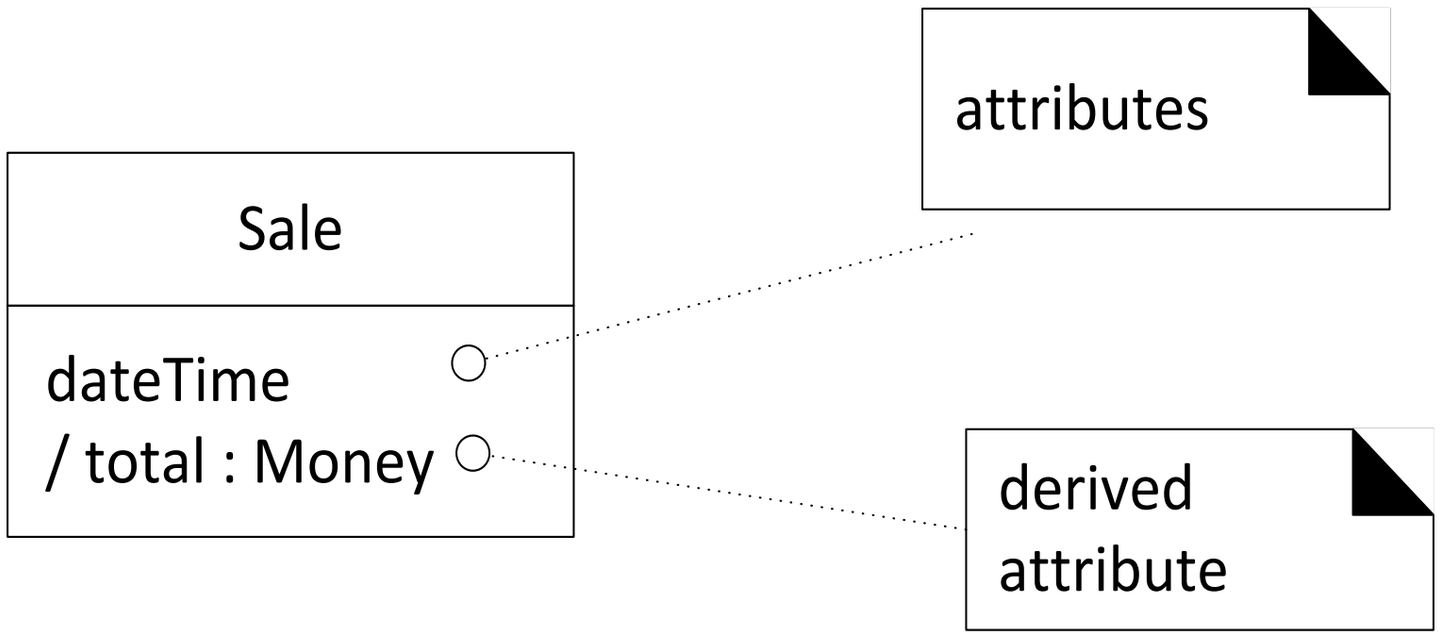
exactly three,
five or eight

EXAMPLE: Partial Domain Model

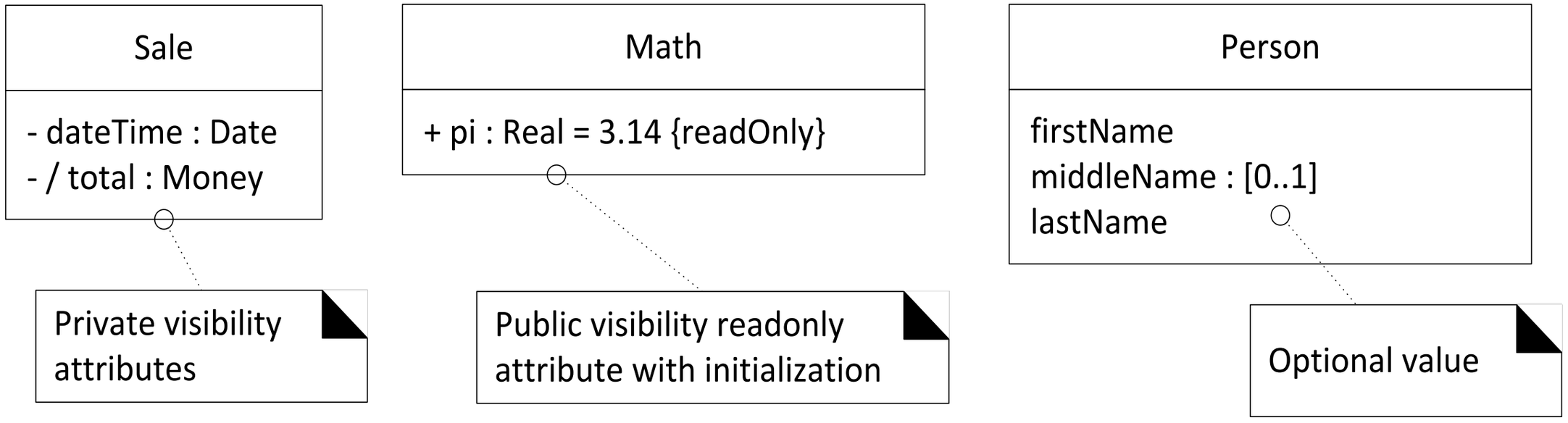


Attributes

- An object's **logical data value** that must be remembered
  - Some attributes are derived from other attributes



Visibility in Domain Models

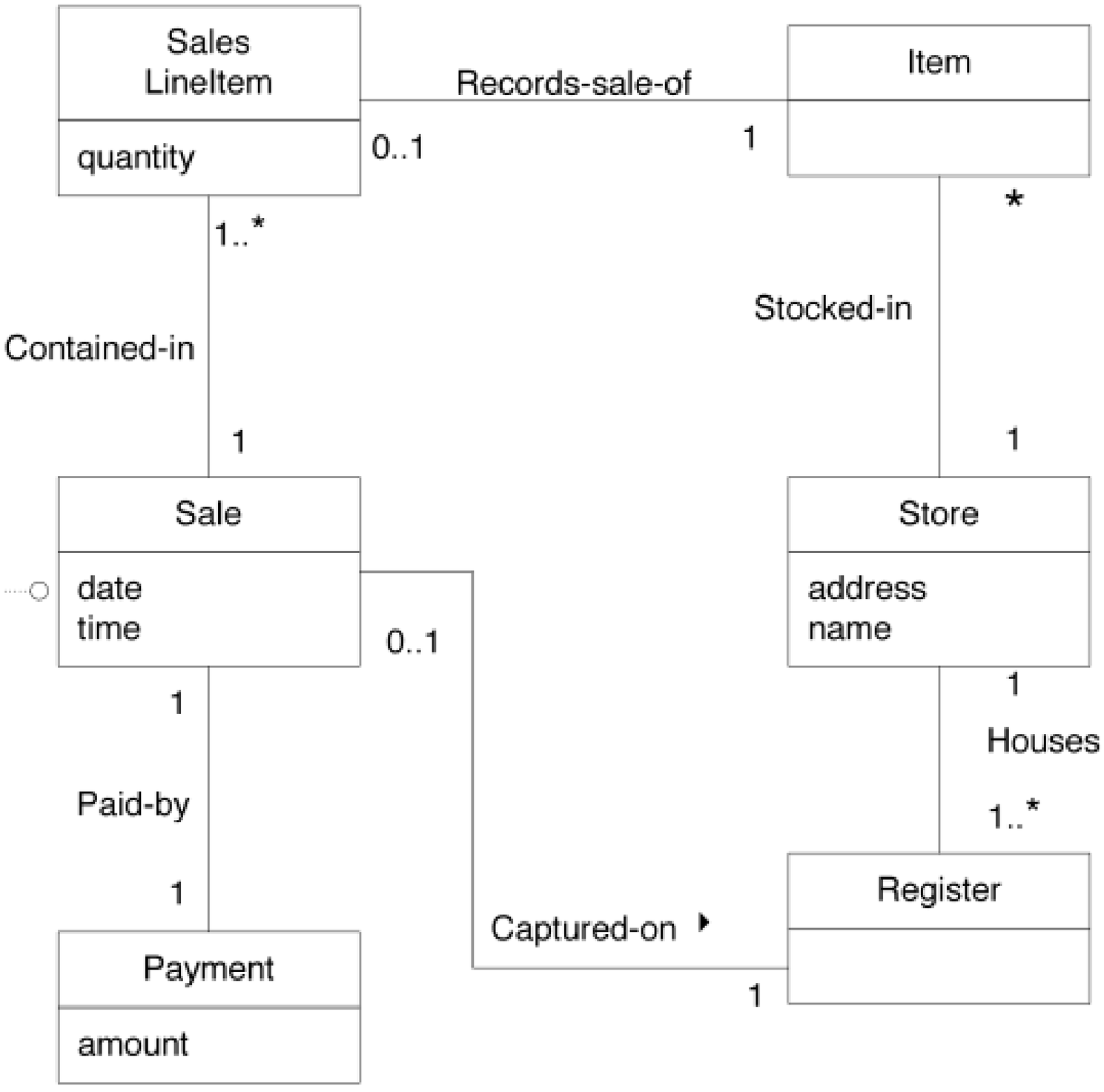


The full UML attribute syntax:

[visibility] name : [type] [multiplicity] [= default] [property-string]

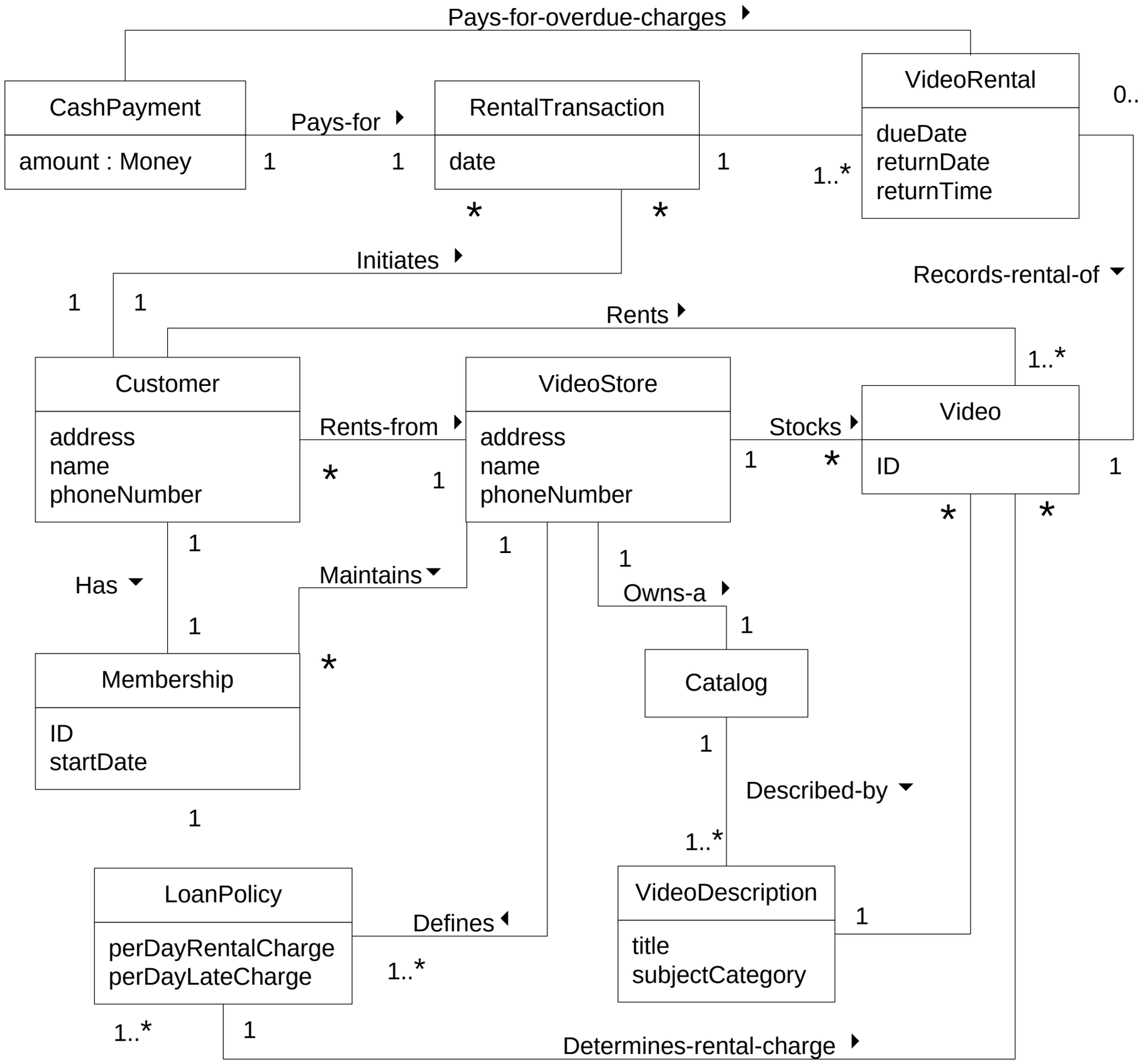
e.g. + store : Walmart [1..*] {readOnly}

Part of POS System



EXAMPLES

EXAMPLE: Domain Model



Notice how this can be viewed as a "visual dictionary." It *illustrates* concepts, words, things in a domain.

Statements about a Course Management System

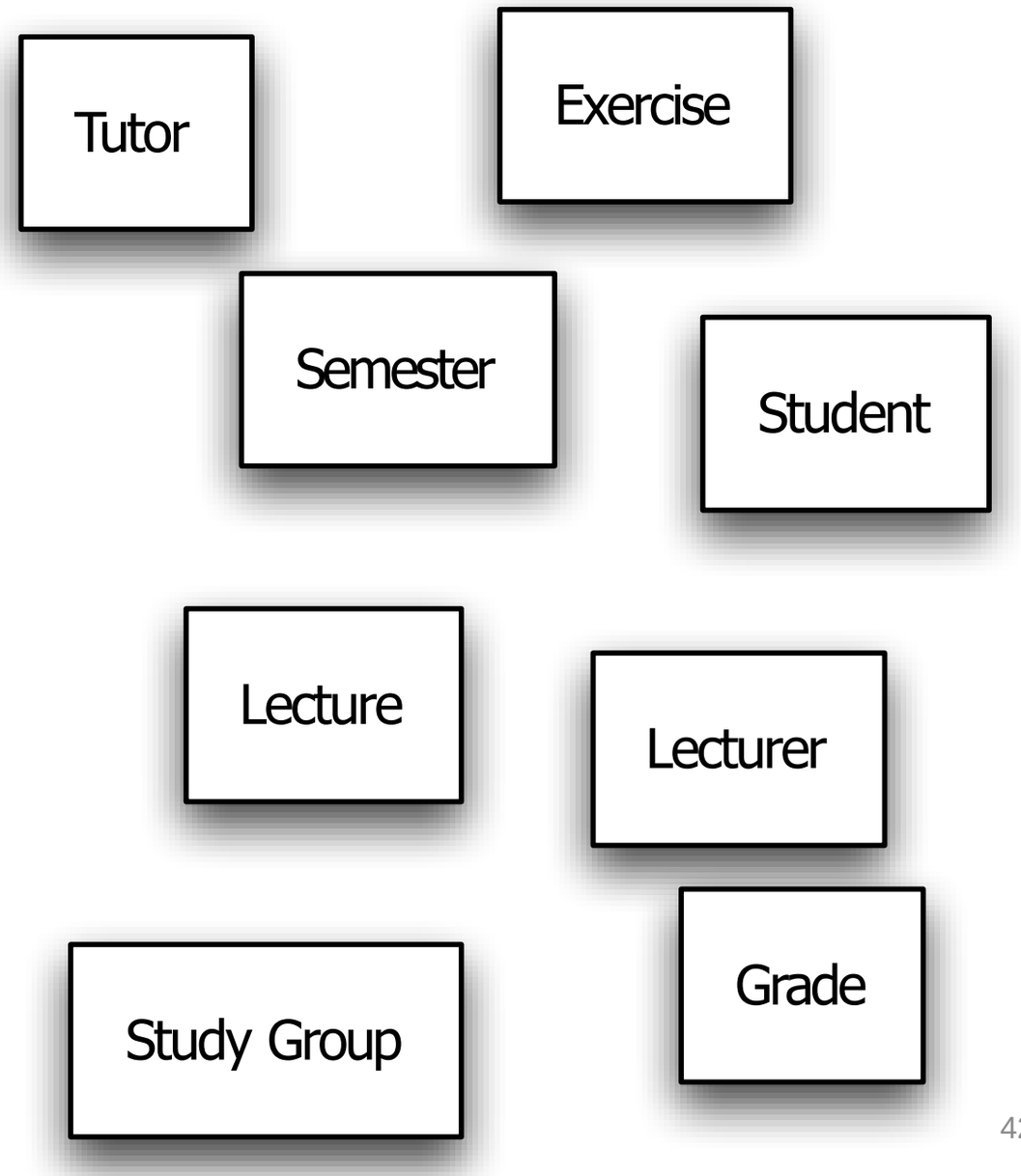
- During a semester a lecturer reads one or more lectures
- Sometimes the lecturer is on leave to focus on doing research, in this case (s)he does not give a lecture
- A student usually attends one or more lectures, unless (s)he has something better to do
- During the semester there will be several exercises which are meant to be solved by small study groups
- Each student is assigned to one particular study group for the whole semester
- A study group consists of two to three students
- After submission of a solution by a study group it is graded by a tutor
- …

Statements about a Course Management System

- During a semester a lecturer reads one or more lectures
- Sometimes the lecturer is on leave to focus on doing research, in this case (s)he does not give a lecture
- A student usually attends one or more lectures, unless (s)he has something better to do
- During the semester there will be several exercises which are meant to be solved by small study groups
- Each student is assigned to one particular study group for the whole semester
- A study group consists of two to three students
- After submission of a solution by a study group it is graded by a tutor
- …

A class describes a set of objects with the same semantics, properties and behavior.

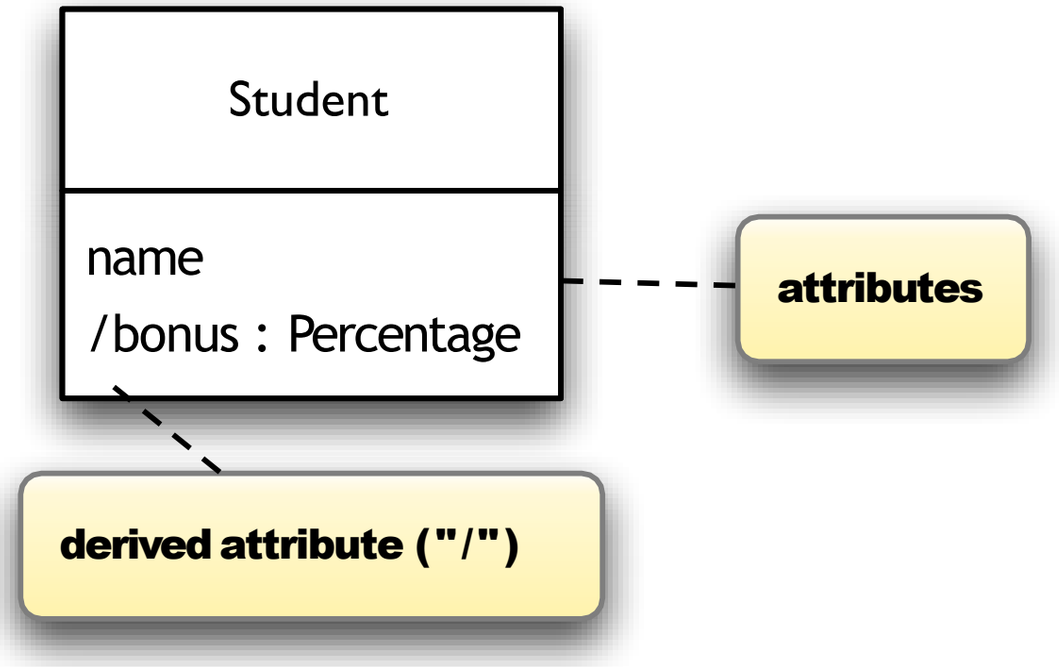
- During a semester a lecturer reads one or more lectures
- A student usually attends one or more lectures, ...
- During the semester there will be several exercises...
- Each student is assigned to one particular study group for the whole Semester
- ... it is graded by a tutor



Attributes are logical data values of an object.

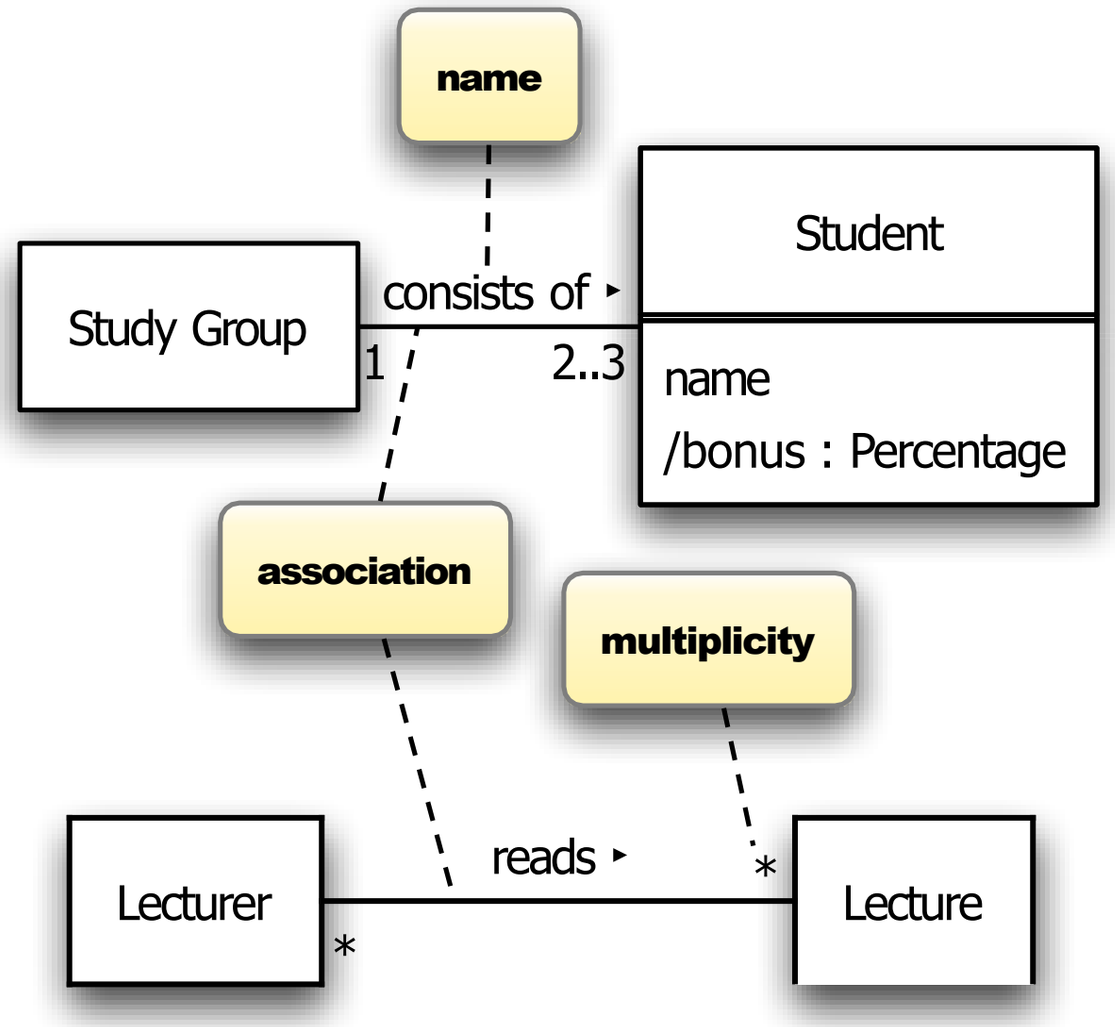
- ... after submitting a solution it is graded by a tutor
- The bonus is a relative bonus that reflects the relative number of exercise points gained during the semester
- ...

The bonus is derived.



The ends of an association are called roles. Roles optionally have a multiplicity, name and navigability.

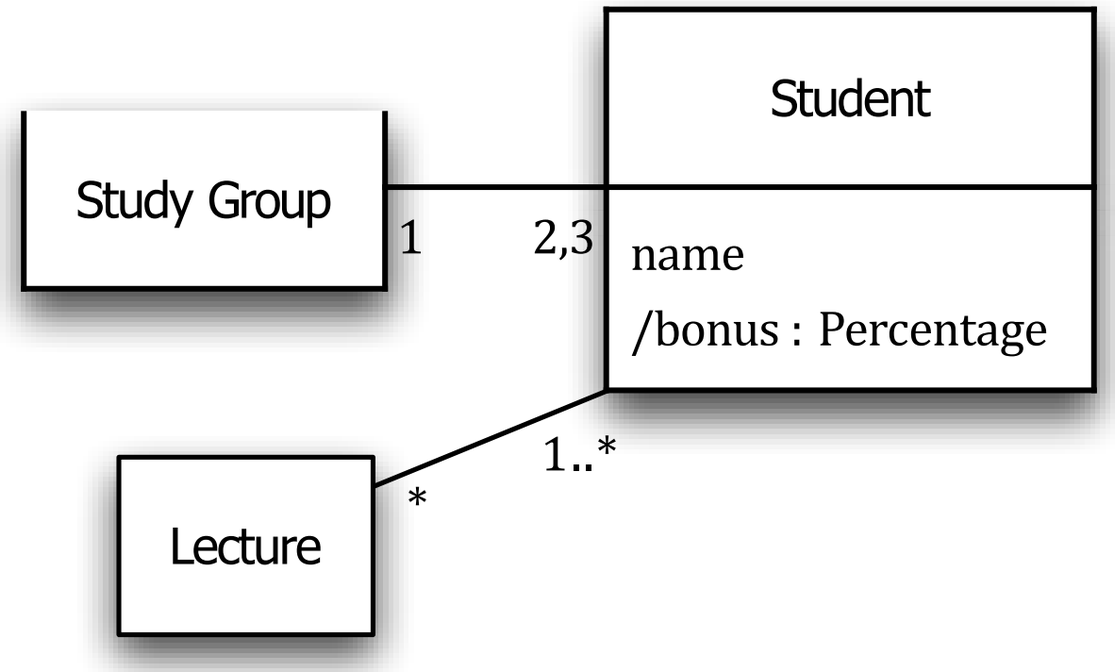
- A lecturer **reads** one or more lectures...
- A student usually attends one or more lectures...
- A study group **consists of** two to three students...
- During the semester there will be several exercises
- ...



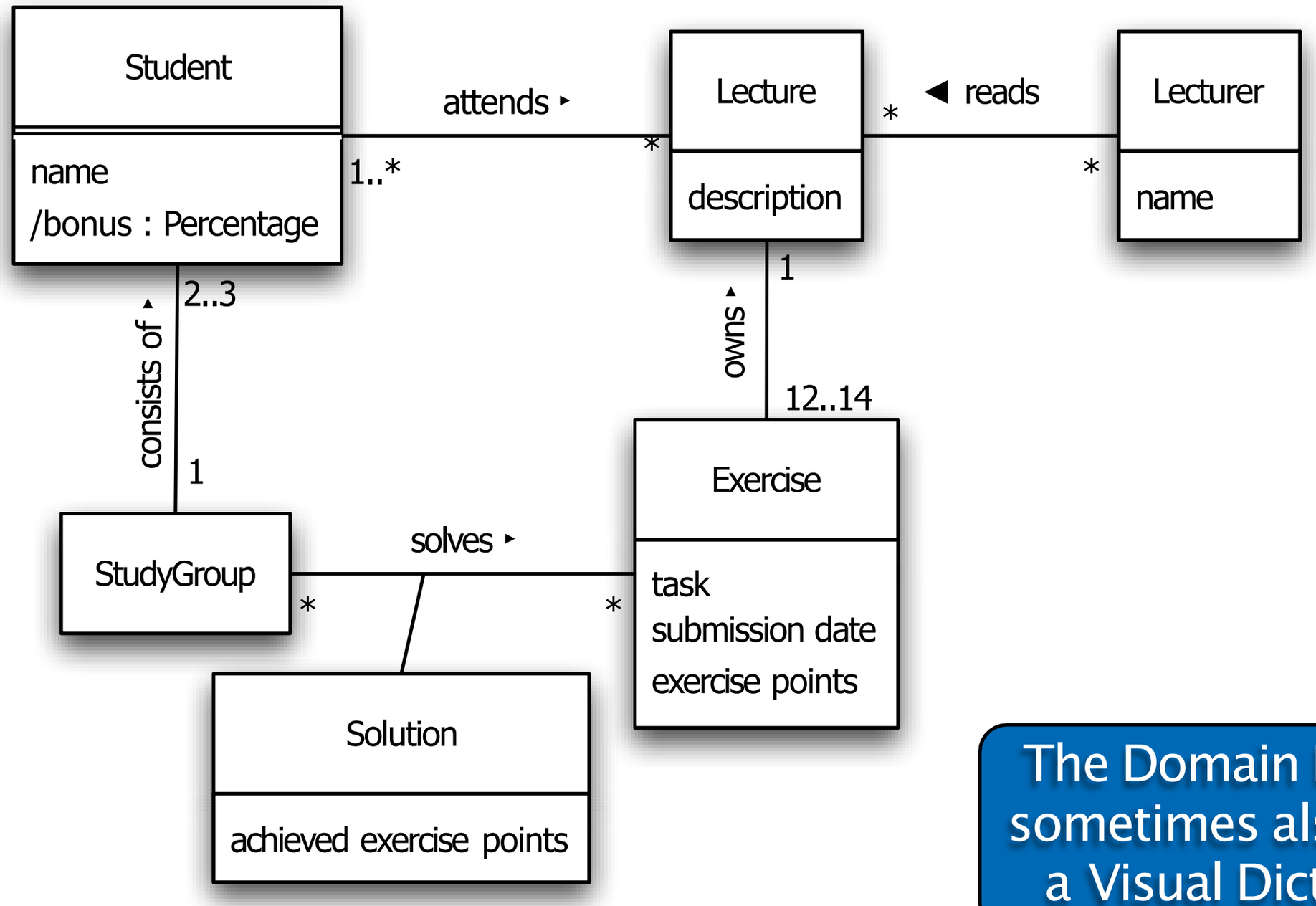
The multiplicity defines how many instances of a class A can be associated with one instance of a class B at any particular moment.

(e.g., * ≙ zero or more; **1..10** between 1 and 10; **1,2** one or two)

- A student usually attends **one or more** lectures, unless (s)he has something better to do
- A study group consists of **two to three** students...
- ...



A preliminary domain model for a course management system.



The Domain Model is sometimes also called a Visual Dictionary.

Example 2

- Overview

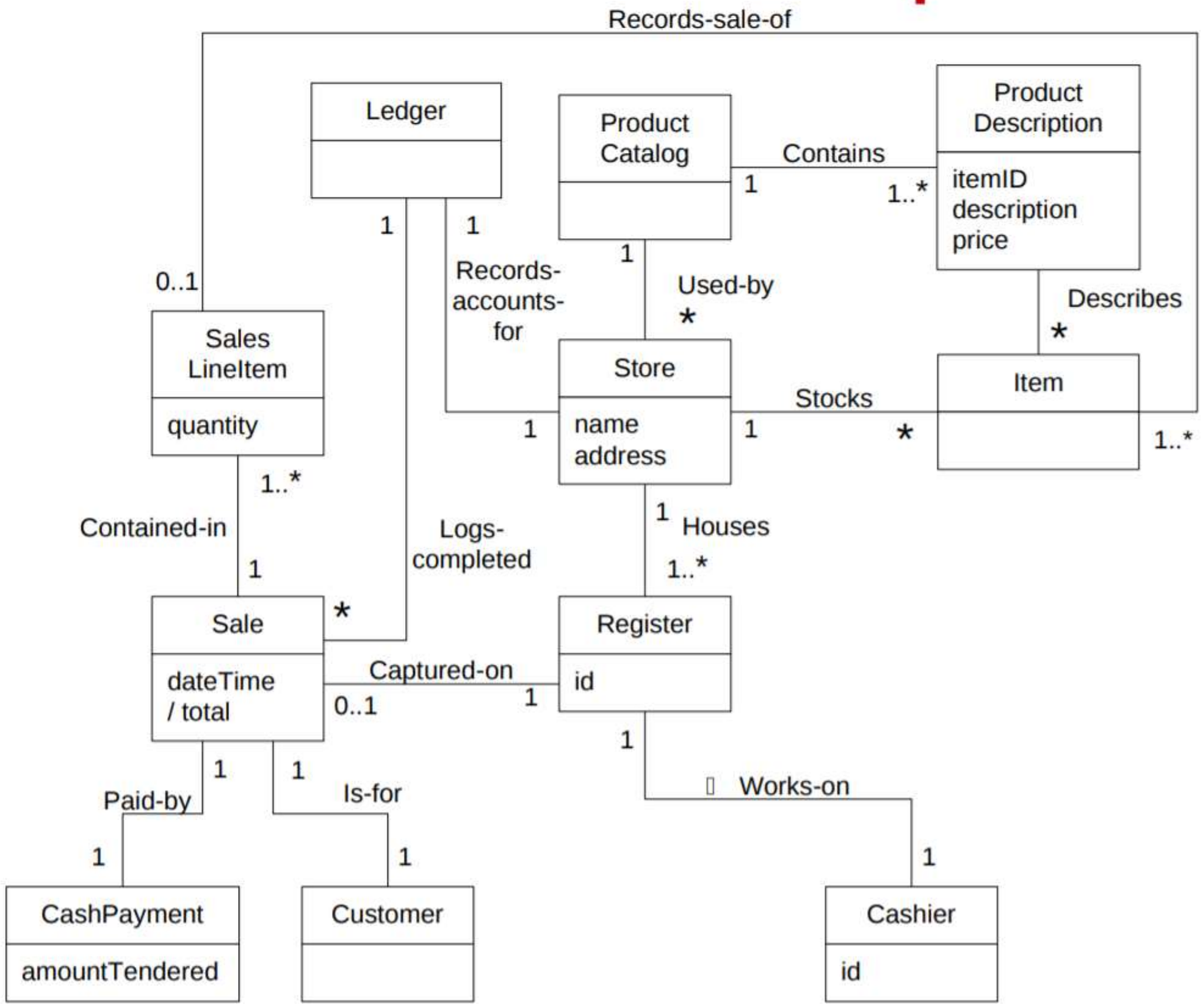
Point of Sale System

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).


Records-sale-of
Ledger
Product Catalog
Contains
Product Description
itemID
description
price
1
1..*
1
1
Records-accounts-for
1
Used-by
*
0..1
Describes
*
Sales LineItem
quantity
Store
name
address
Item
Stocks
1
1
*
1..*
1..*
Contained-in
1
Logs-completed
1
Houses
1..*
Sale
*
dateTime
/ total
Register
id
Captured-on
0..1
1
1
1
Paid-by
Is-for
Works-on
1
1
1
CashPayment
amountTendered
Customer
Cashier
id