

Domain Model

Lecture 3

Introduction

- A **domain model** is the **most important** and **classic-model** in Object Oriented Analysis

YOUR ATTENTION PLEASE



What is a Domain Model?

- ***Problem domain : area (scope) of application that needed to be investigated to solve the problem***
- ***Domain Model : Illustrates meaningful conceptual objects in problem domain.***
- ***So domain model are conceptual objects of the area of application to be investigated***

Domain model representation?

- A domain model is a visual representation of **real world concepts** (real-situation objects), that could be : **idea, thing , event or object.....etc .**
 - **Business objects** - represent things that are manipulated in the business e.g. **Order**.
 - **Real world objects** – things that the business keeps track of e.g. **Contact , book**.
 - **Events** that come to light - e.g. **sale, loan** and **payment**.

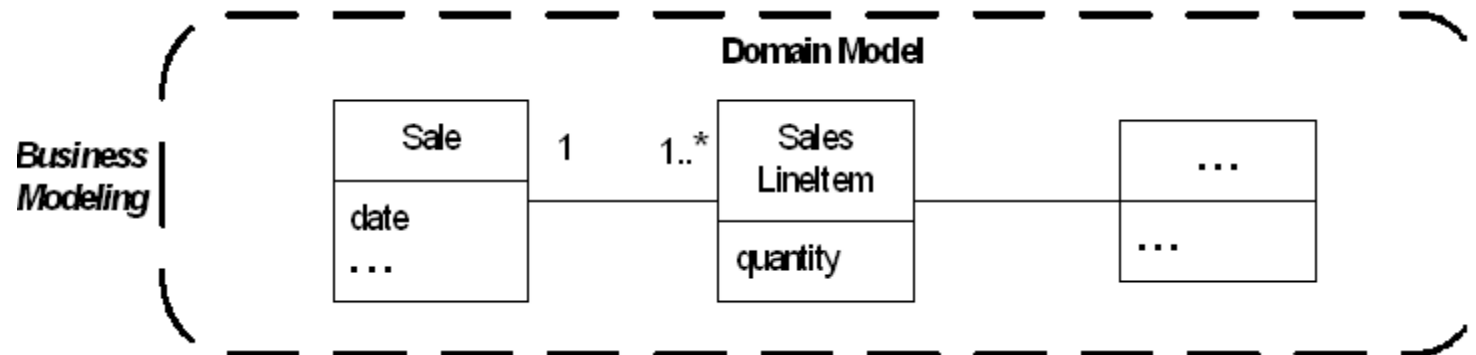
Domain model representation?

Domain Model may contain :

- Domain **objects** (conceptual classes)
 - **Attributes** of domain objects
- **Associations** between domain objects
 - **Multiplicity**

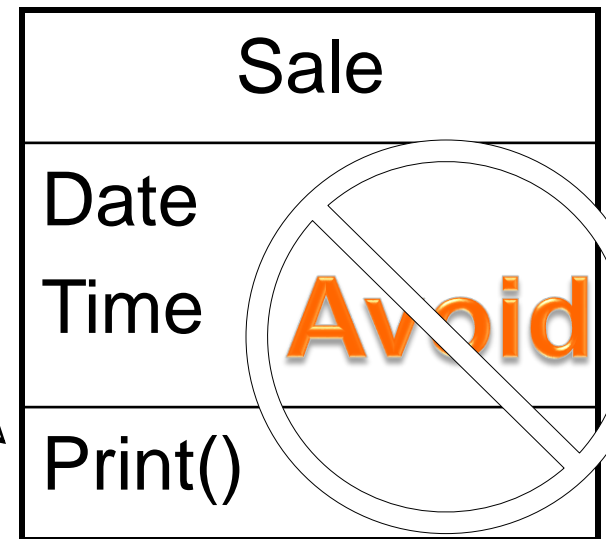
Domain Model - UML Notation

- Illustrated using a set of domain objects (conceptual classes) **with no operations** (*no **responsibility** assigned yet , **this will be assigned during design***).

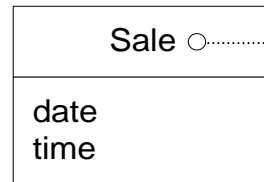


A Domain Model is not a Software document

Object **responsibilities** is not part of the domain model. (*But to consider during Design*)



Symbol, intension and extension.



concept's symbol

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

concept's intension

All the examples of sales called **instances**

sale-1

sale-3

sale-2

sale-4

concept's extension

A Domain Model is Conceptual, not a Software Artifact

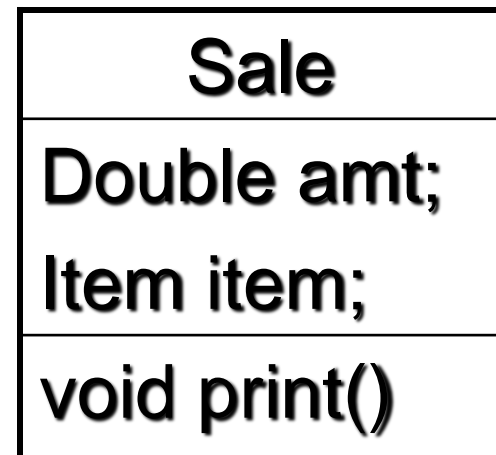
Conceptual Class:



Software Artifacts:



vs.



What's the
difference?

Question : Why create Domain model ?

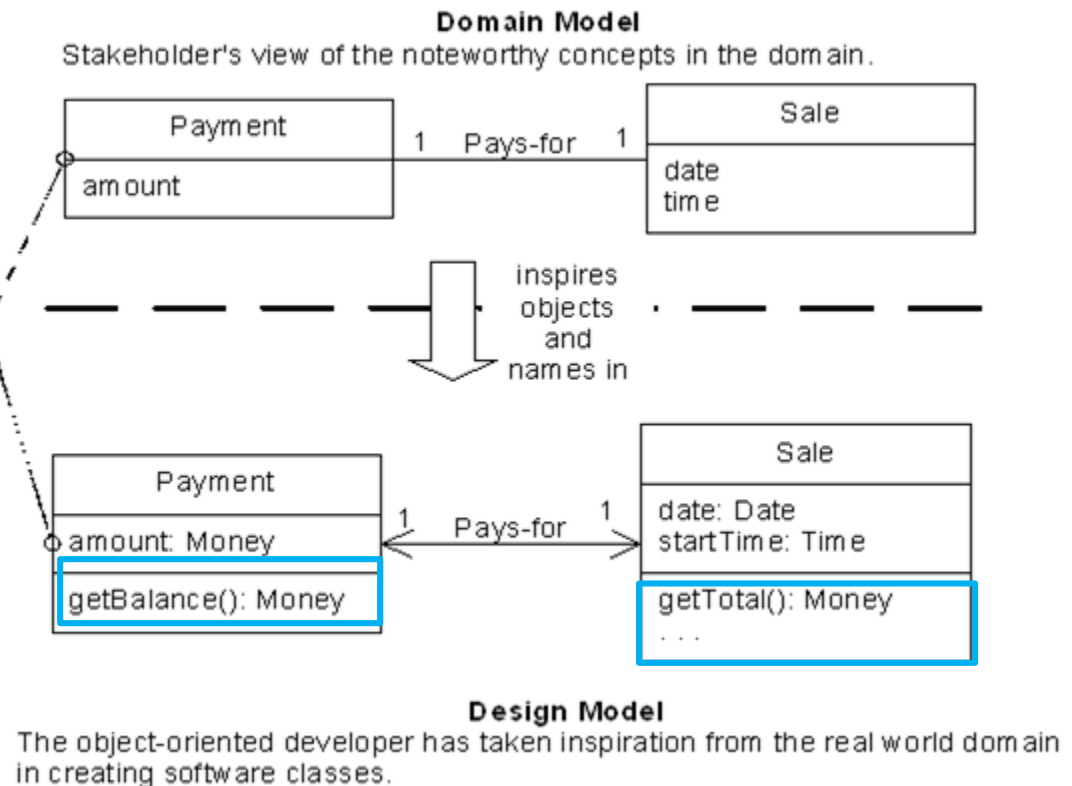
Answer : Get inspiration to create software classes

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.

We assign responsibilities during design



***How to find these
conceptual classes
and attributes ?***

Method1: Noun Phrase Identification

- *Identify Nouns and Noun Phrases in textual descriptions of the domain that could be :*
 - *The problem definition.*
 - *The Scope.*
 - *The vision.*

Method1: Noun Phrase Identification

However,

- *Words may be ambiguous (such as : System)*
- *Different phrases may represent the same concepts.*
- *Noun phrases may also be attributes or parameters rather than classes:*
 - *If it stores state information or it has multiple behaviors, then it's a class*
 - *If it's just a number or a string, then it's probably an attribute*

Noun Phrase Identification

- Consider the following problem description, analyzed for Subjects, Verbs, Objects:

The ATM verifies whether the customer's card number and PIN are correct.

If it is, then the customer can check the account balance, deposit cash, and withdraw cash.

Checking the balance simply displays the account balance.

Depositing asks the customer to enter the amount, then updates the account balance.

Withdraw cash asks the customer for the amount to withdraw; if the account has enough cash, the account balance is updated. The ATM prints the customer's account balance on a receipt.

Noun Phrase Identification

- Consider the following problem description, analyzed for Subjects, Verbs, Objects:

The ATM verifies whether the customer's card number and PIN are correct.

S V O O O

If it is, then the customer can check the account balance, deposit cash, and withdraw cash.

S V O V O V O

Checking the balance simply displays the account balance.

S O V O

Depositing asks the customer to enter the amount, then updates the account balance.

S V O V O V O

Withdraw cash asks the customer for the amount to withdraw; if the account has enough cash,

S O V O O V S V O

the account balance is updated. The ATM prints the customer's account balance on a receipt.

O V S V O V O

Noun Phrase

Analyze each **subject** and **object** as follows:

- Does it represent a person performing an action? Then it's an actor, '**R**'.
- Is it also a verb (such as 'deposit')? Then it may be a method, '**M**'.
- Is it a simple value, such as 'color' (string) or 'money' (number)?

Then it is probably an attribute, '**A**'.

- Which NPs are unmarked? Make it '**C**' for class.

Verbs can also be classes, for example:

- **Deposit** is a class if it retains state information

Noun Phrase Identification

- Consider the following problem description, analyzed for Subjects, Verbs, Objects:

The ATM verifies whether the customer's card number and PIN are correct.

Ⓒ V Ⓓ Ⓐ Ⓐ

If it is, then the customer can check the account balance, deposit cash, and withdraw cash.

Ⓓ V Ⓐ V Ⓐ V Ⓐ

Checking the balance simply displays the account balance.

Ⓜ Ⓐ V Ⓐ

Depositing asks the customer to enter the amount, then updates the account balance.

Ⓜ V Ⓓ V Ⓐ Ⓜ Ⓐ

Withdraw cash asks the customer for the amount to withdraw; if the account has enough cash,

Ⓜ Ⓐ V Ⓓ Ⓐ V Ⓒ V Ⓐ

the account balance is updated. The ATM prints the customer's account balance on a receipt.

Ⓐ V Ⓒ V Ⓐ Ⓐ

Main Success Scenario (or Basic Flow) of POS:

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 3-4 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 completed sale and sends sale and payment information to the external accounting system (for accounting and commissions) and Inventory system (to update inventory).
 9. System presents receipt.
 10. Customer leaves with receipt and goods (if any).

Extensions (or alternative Flows)

7a. Paying by cash:

1. Cashier enters the cash amount tendered.
2. System presents the balance due, and releases the cash drawer.
3. Cashier deposits cash tendered and returns balance in cash to Customer.
4. System records the cash payment.

Class of POS

Register

Item

Store

Sale

Sales
LineItem

Cashier

Customer

Ledger

Cash
Payment

Product
Catalog

Product
Description

Important: There is no such things as a correct list. Brainstorm what you consider noteworthy to be a candidate. But in general, different modelers will find similar lists.

Identification of conceptual classes

- *Identify candidate conceptual classes*
- *Go through them and :*
 - ❑ *Exclude irrelevant features and duplications*
 - ❑ *Do not add things that are outside the scope (outside the application area of investigation)*

Method2 : By Category List

Common Candidates for classes include:

- *Descriptions , Roles , Places , Transactions*
- *Containers , Systems , abstract nouns , Rules*
- *Organizations, Events, Processes, catalogs , Records , services.*

Attributes

- *A logical data value of an object.*
- *Imply a need to remember information.*
 - Sale needs a **dateTime** attribute
 - Store needs a **name** and **address**
 - Cashier needs an **ID**

A Common Mistake when modeling the domain- Classes or Attributes?

Rule

- If we do **not** think of a thing as a **number or text** in the real world, then it is probably a **conceptual class**.
- If it **takes up space**, then it is likely a **conceptual class**.

Examples:

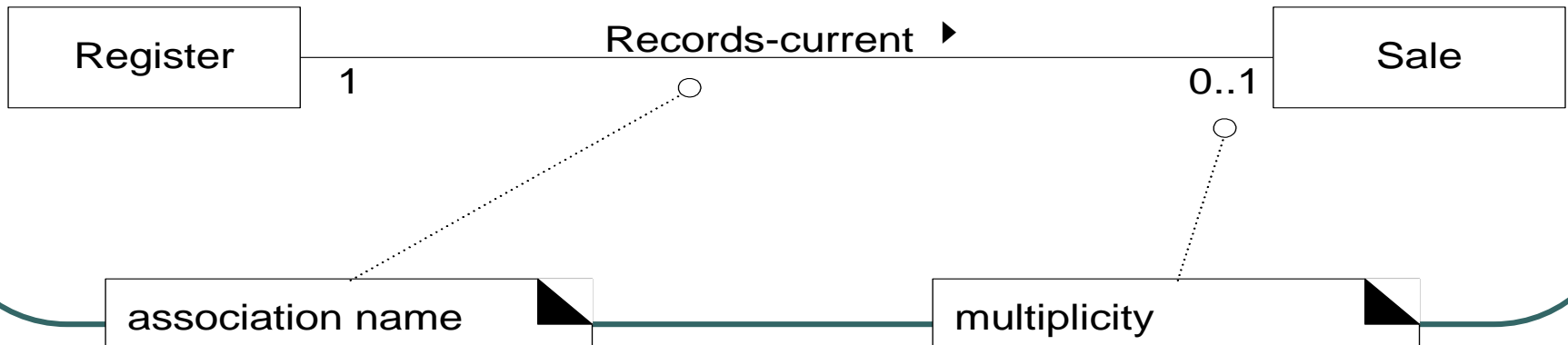
- Is a **store** an attribute of a Sale ?
- Is a **destination** an attribute of a flight ?

***How to find these
associations and
multiplicities?***

Association:

Relationship between classes (more precisely, between instances of those classes) indicating some meaningful and interesting connection)

- "reading direction arrow"
- it has **no** meaning except to indicate direction of reading the association label
- often excluded



Common association list

- A is a physical part of B .
 - Wing - Airplane
- A is a logical part of B
 - SalesLineItem - Sale
- A physical contained in B
 - Register-Sale

Common association list

- A **is a logical contained** in B
 - ItemDescription - Catalog
- A **is a description** of B .
 - ItemDescription - Item
- A **is a member** of B
 - Cashier – Store

Common association list

- A **uses or manage** B
 - Cashier-Register
- A is **recorded in** B
 - Sale-Register
- A is **an organization subunit** of B .
 - Departement - Store

Common association list

- A **communicate with** B
 - Customer - Cashier
- A **is related to a transaction** B
 - Customer - Payment
- A **is a transaction related to another transaction** B .
 - Payment - Sale

Common association list

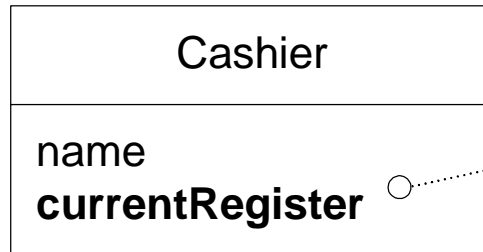
- A **is owned by** B
 - Cashier - Register
- A **is an event related** to B
 - Sale- Customer

High priority association

- A is a physical or logical part of B
- A is physically or logically contained in/on B
- A is recorded in B
- To avoid:
 - *Avoid showing **redundant** associations*
 - *Do not overwhelm the domain model with associations **not strongly required***

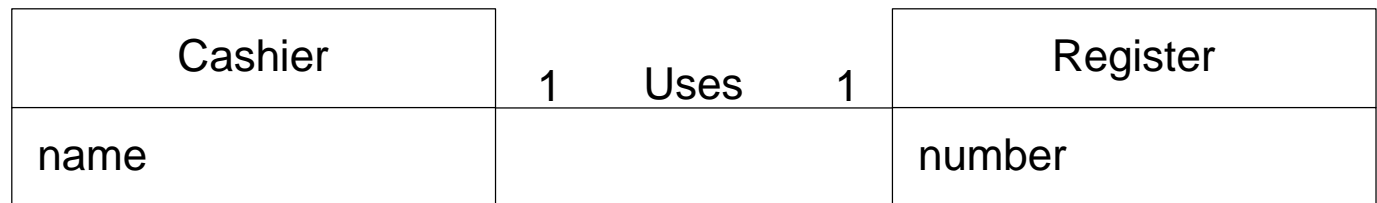
Association or attribute ?

Worse



not a "data type" attribute

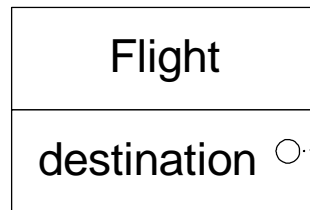
Better



- ☐ Most attribute type should be “**primitive**” data type, such as: *numbers* , *string* or *boolean* (true or false)
- ☐ Attribute should not be a complex domain concept(Sale , Airport)
- ☐ CurrentRegister is of type “Register”, so expressed with an association

Association or attribute ?

Worse



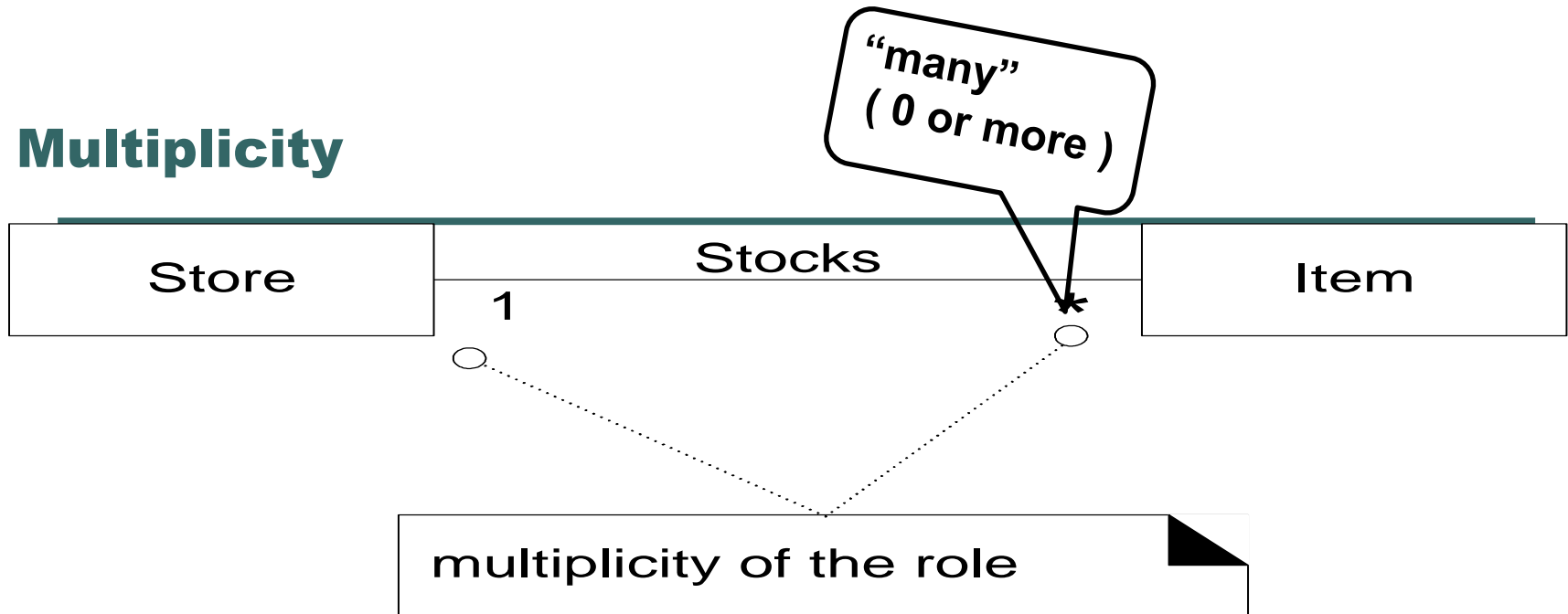
destination is a complex concept

Better



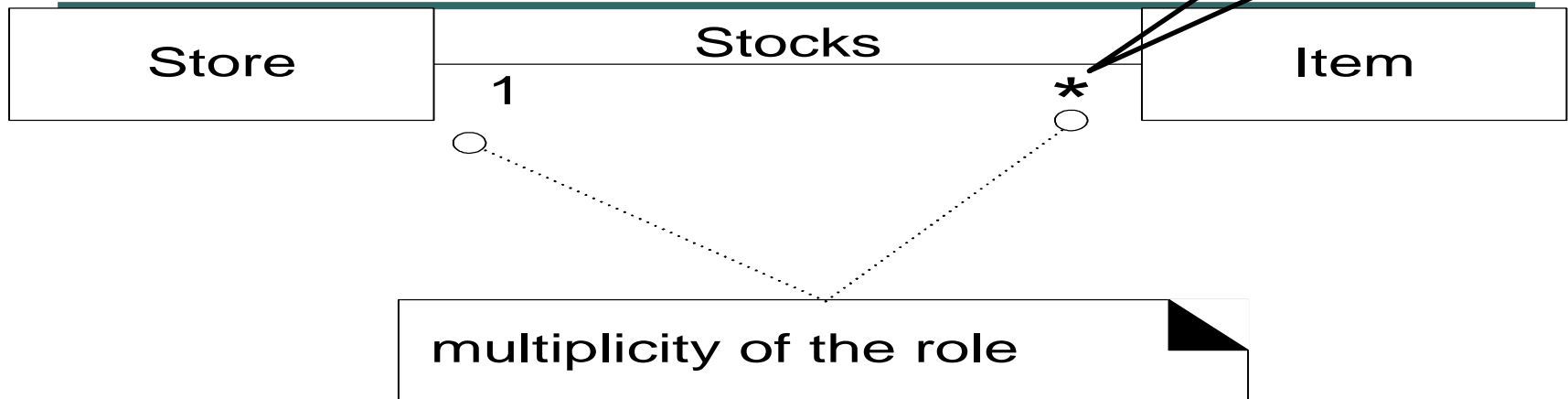
A destination airport is ***not a string***, it is a complex thing that occupies many square kilometers of space. So “Airport” should be related to “Flight” via an association, not with attribute

Multiplicity



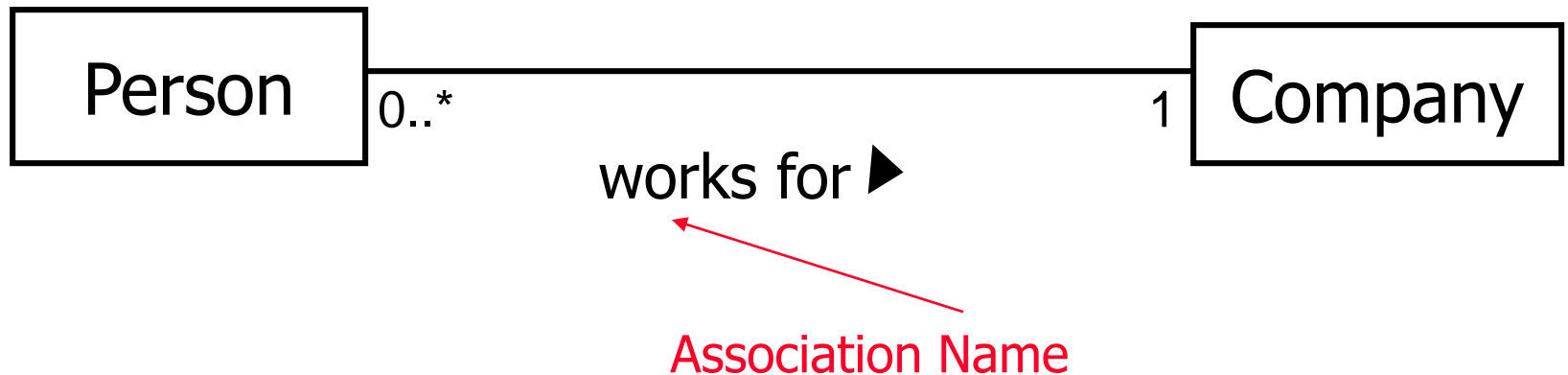
- **Multiplicity** indicates how **many instances** can be validly associated with another instance, at **a particular moment**, rather than over a span of time.

How to determine multiplicity ?



- **Ask these 2 questions :**
 - **1 store may stock how many item ?**
 - **1 item may be stocked in how many stores ?**

How to determine multiplicity ?



Multiplicity

<u> *</u>	T	zero or more; "many"
<u> 1..*</u>	T	one or more
<u> 1..40</u>	T	one to 40
<u> 5</u>	T	exactly 5
<u> 3, 5, 8</u>	T	exactly 3, 5, or 8

How to create a domain model

- *Identify candidate conceptual classes*
- *Go through them*
 - ***Exclude irrelevant features and duplications***
 - ***Do not add things that are outside the scope***
- *Draw them as classes in a UML class diagram*
- *Add **associations** necessary to record the relationship that must be retained*
- *Add **attributes** necessary for information to be preserved*

But remember

- *There is no such thing as a single correct domain model. All models **are approximations of the domain** we are attempting to understand.*
- *We **incrementally evolve a domain model** over several iterations on attempts to capture all possible conceptual classes and relationships.*

Case Study (User Story)

- Customer confirms items in shopping cart. Customer provides payment and address to process sale. System validates payment and responds by confirming order, and provides order number that Customer can use to check on order status. System will send customers copy order details by email.
- ***Perform the steps of Domain model on the mentioned case study.***

Remove Irrelevant Data

Customer	Order
Item	Order Number
Shopping Cart	Order Status
Payment	Order Details
Address	Email
Sale	System

Finding Conceptual Classes

Customer

Shopping Cart

Payment

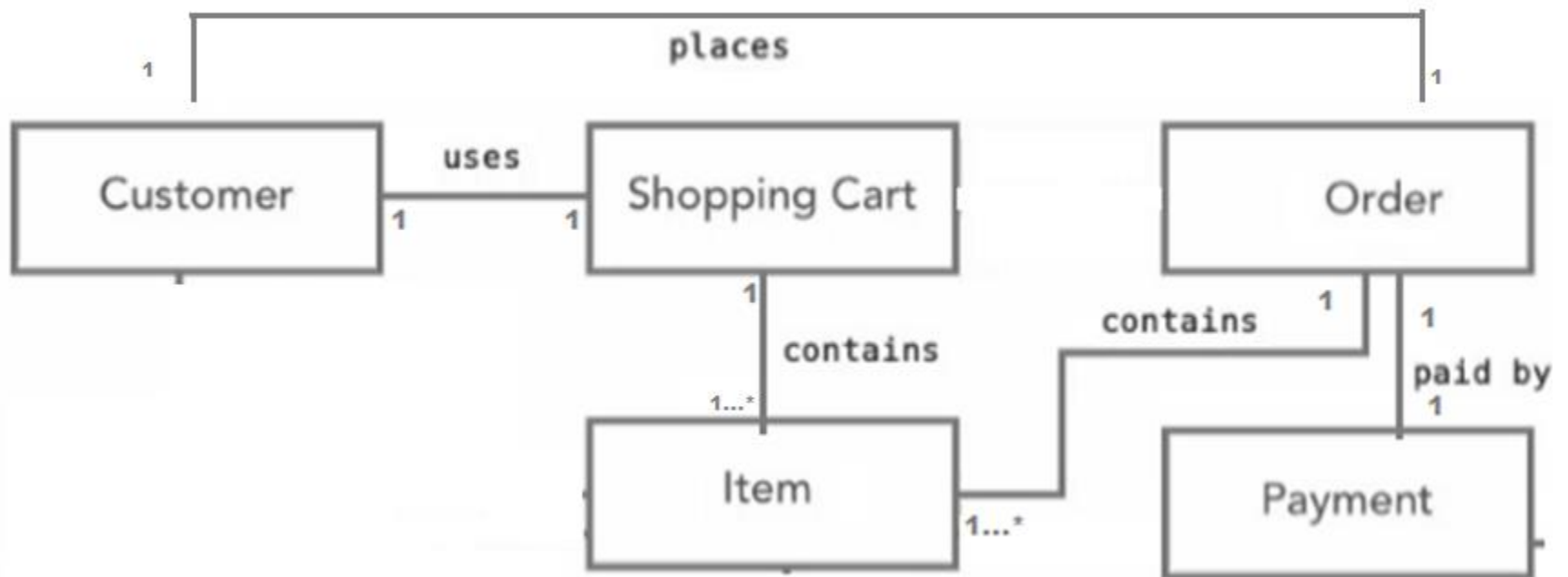
Item

Order

Add Association



Add Multiplicity



Homework: Example Restaurant case

Record Booking

- Receptionist enters date of requested reservation;
- System displays bookings for that date;
- There is a suitable table available: Receptionist enters details (customer's name, phone number, time of booking, number of covers, table number);
- System records and displays new booking.