Domain Model

Lecture # 13, 14, 15
28, 30 Sep 3 Oct

Rubab Jaffar
rubab.jaffar@nu.edu.pk

Software Design and Analysis

CS-324

Today's Outline

- Use case Lab
- Domain Modelling
- How to create domain model?
- Example
- Case Study

Domain Modelling

- The end goal of object-oriented analysis and design is the construction of the classes that will be used to implement the desired software system.
- Domain modeling is a first step in that direction.
- 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

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*
- The Domain Model illustrates noteworthy concepts in a domain

Domain Model Representation

- Captures the most important types of objects in a system.
- *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***.

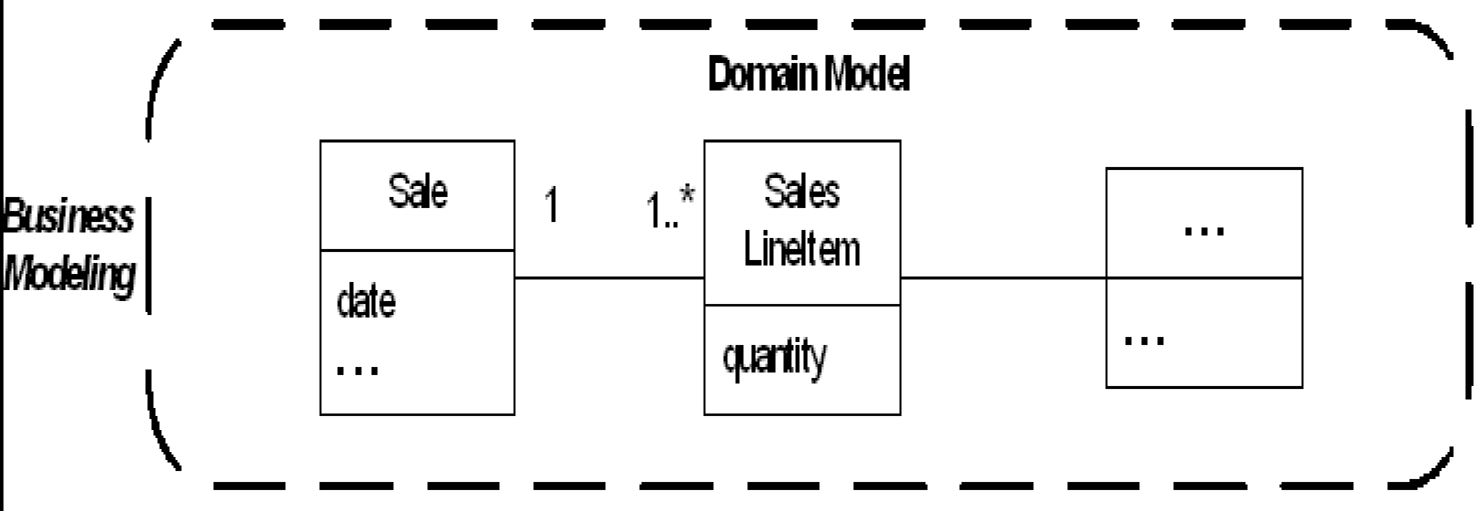
Domain Model Representation

Domain Model may contain :

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

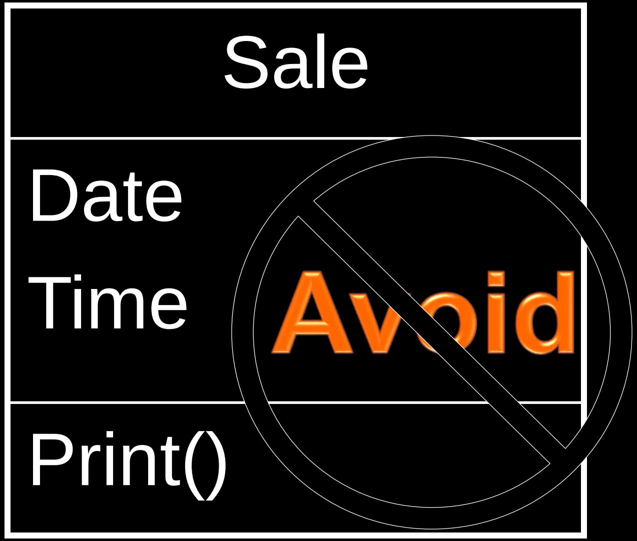
Domain Model - UML Notation

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

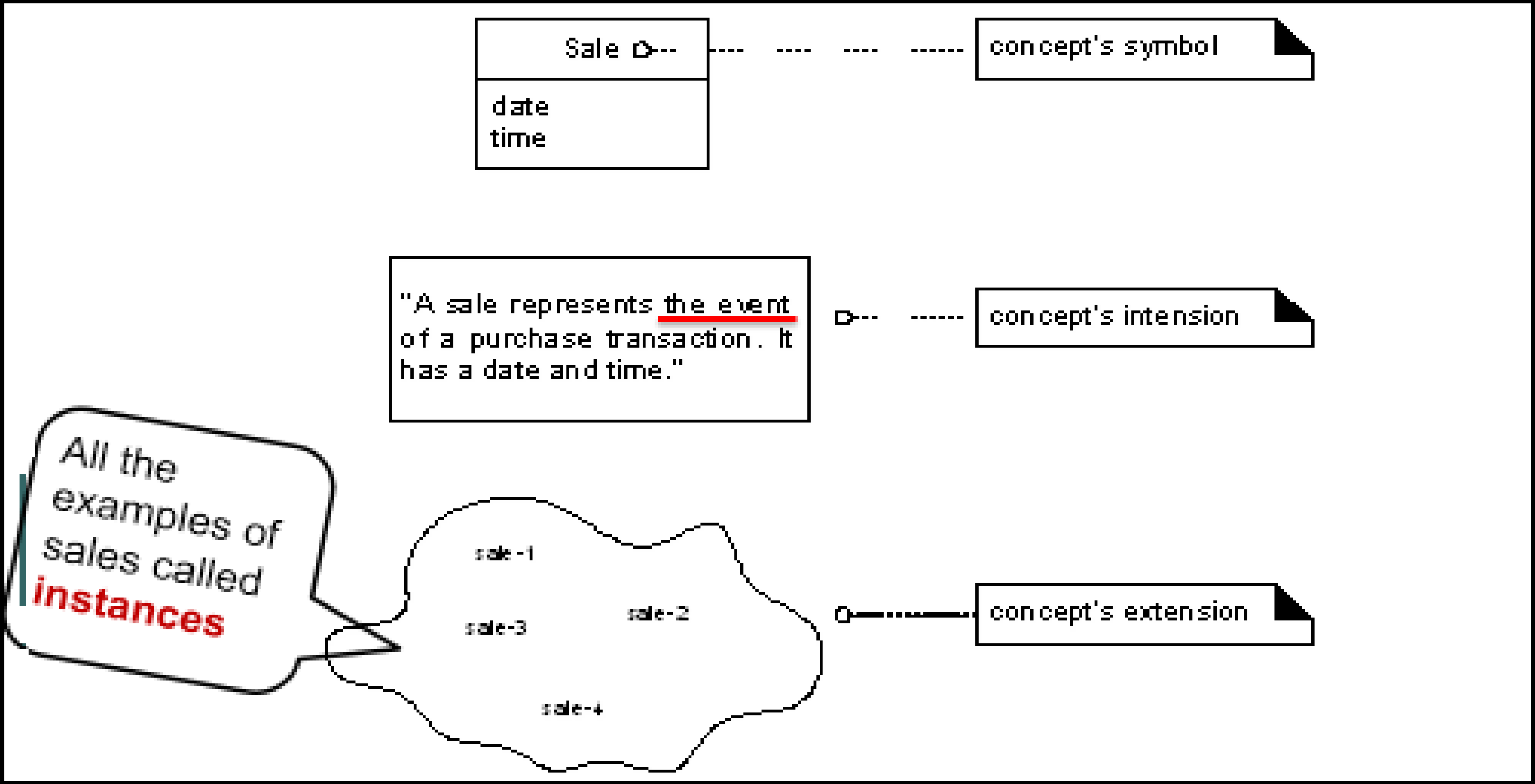


A Domain Model is not a Software document

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



Symbol, Intension and Extension.



Sale

date
time

concept's symbol

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

concept's intension

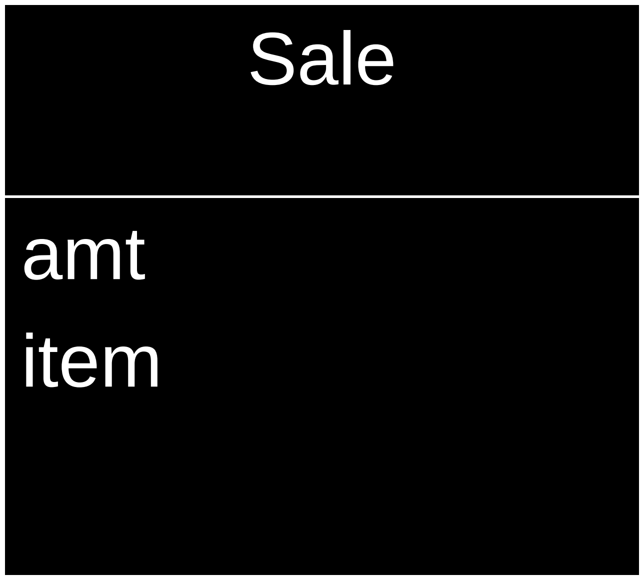
All the examples of sales called **instances**

sale-1
sale-2
sale-3
sale-4

concept's extension

A Domain Model is Conceptual, not a Software Artifact

Conceptual Class:



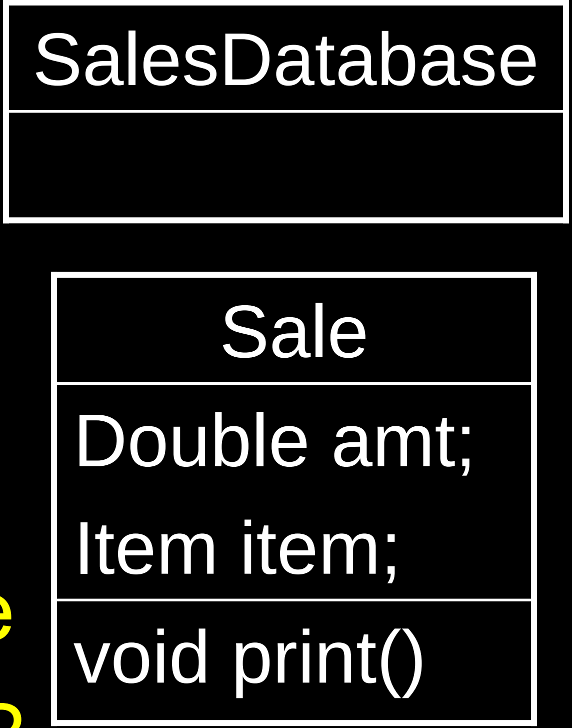
Sale

amt

item

vs.

Software Artifacts:



SalesDatabase

Sale

Double amt;

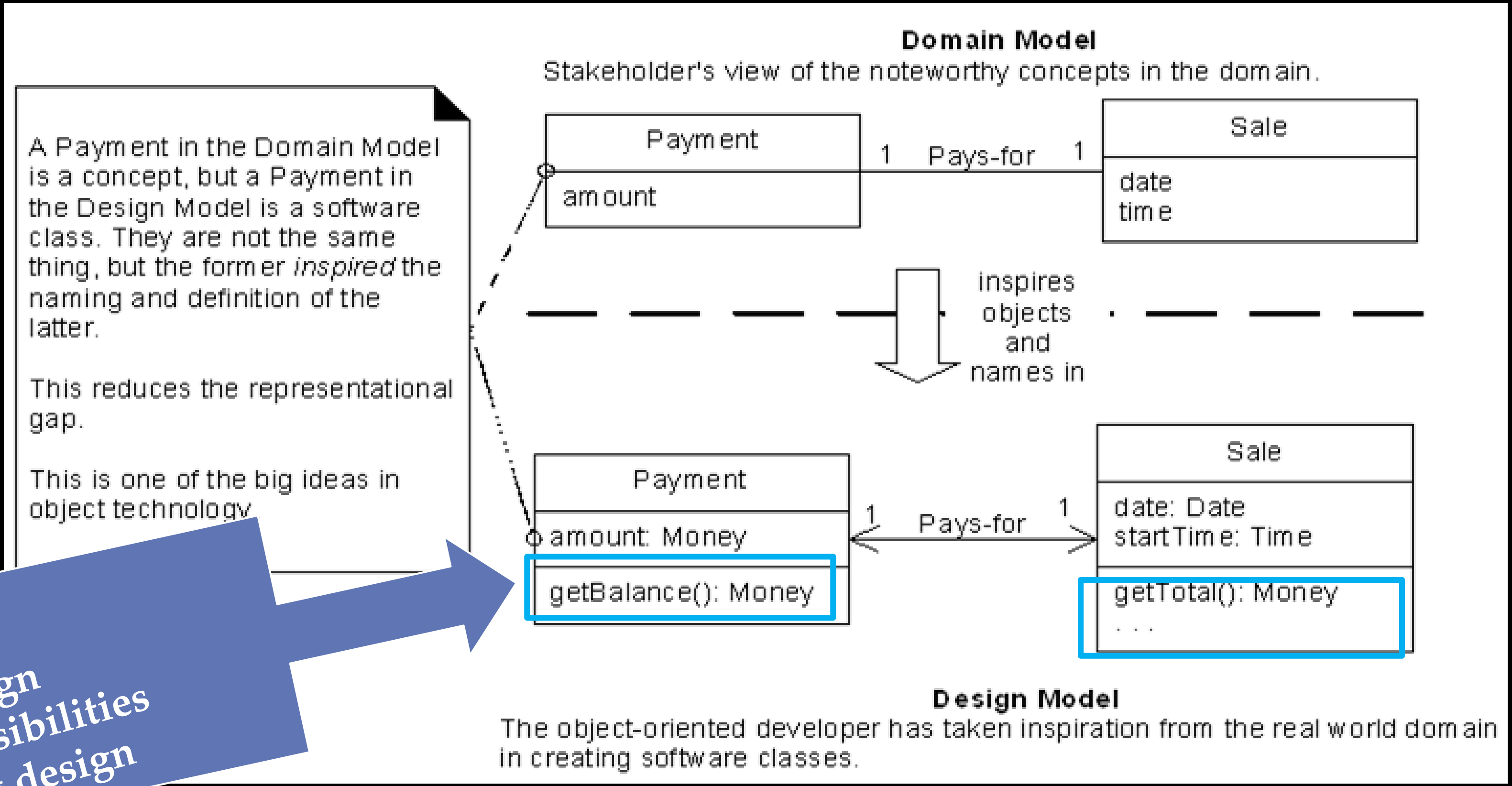
Item item;

void print()

What's the difference?

Why Create Domain model?

Answer : Get inspiration to create software classes



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.

Design Model

The object-oriented developer has taken inspiration from the real world domain in creating software classes.

We assign responsibilities during design

Characteristics of Domain Modeling

- Visual representation of conceptual classes.
- Associations and relationships between concepts (e.g Payment PAYS-FOR Sales).
- Attributes for information content (e.g. Sale records DATE and TIME).
- Does not include operations / functions.
- Does not describe software classes.
- Does not describe software responsibilities.

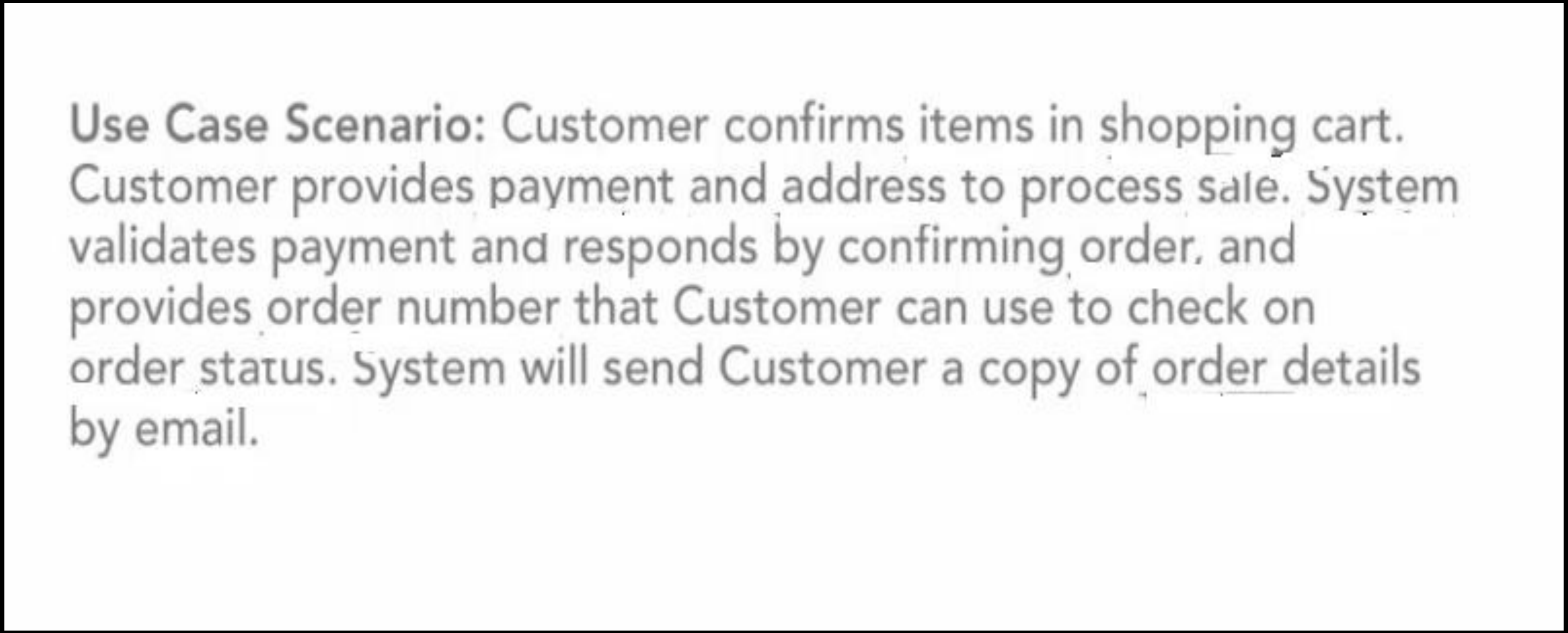
Steps To Create A Domain Model

1. Create User Stories
2. Identify candidate conceptual classes
3. Draw them in a UML domain model
4. Add associations necessary to record the relationships that must be retained
5. Add attributes necessary for information to be preserved
6. Use existing names for things, the vocabulary of the domain

User Stories

- A User Story is one or more sentences in the everyday or business language of the end user or user of a system that captures what a user does or needs to do as part of his or her job function.
- It captures the 'who', 'what' and 'why' of a requirement in a simple, concise way, often limited in detail by what can be hand-written on a small paper note card.
- User stories are the descriptions of the domain that could be :
  - The problem definition.
  - The Scope.
  - The vision.

User Stories



Use Case Scenario: 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 Customer a copy of order details by email.

Identify Objects: Noun Phrase Identification

- *Identify Nouns and Noun Phrases in textual descriptions of the domain.*
- *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,

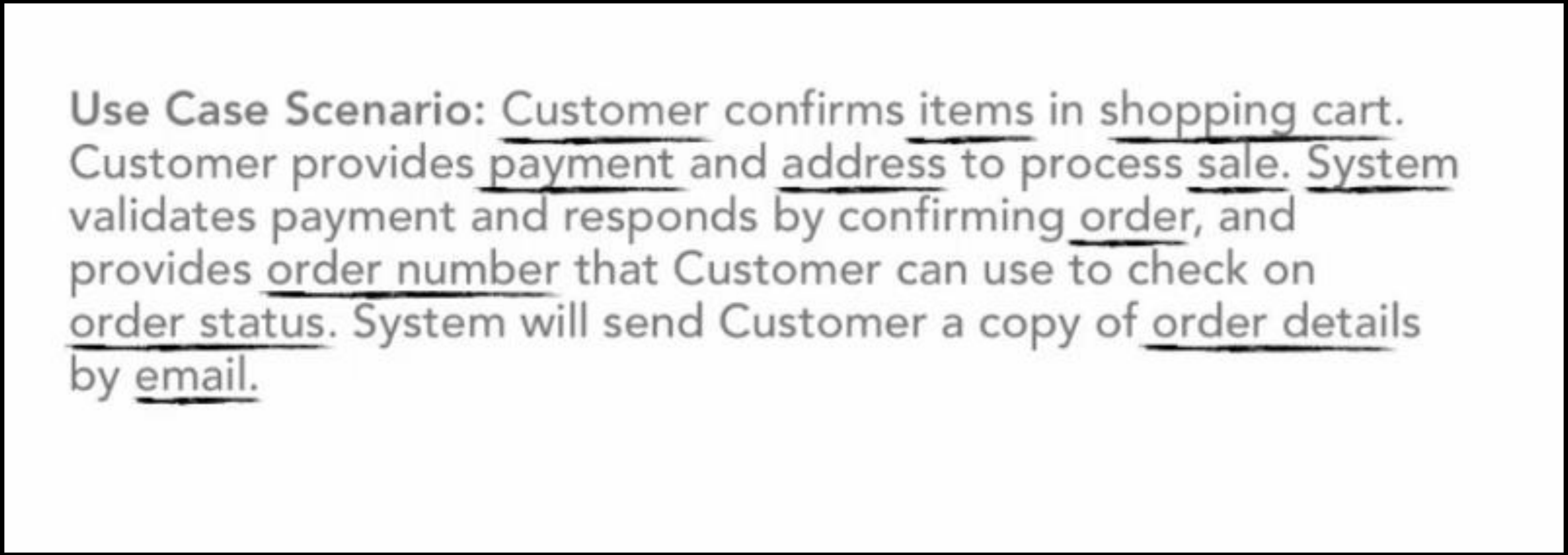
the account balance is updated. The ATM prints the customer's account balance on a receipt.

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                        O
```

Identify Objects

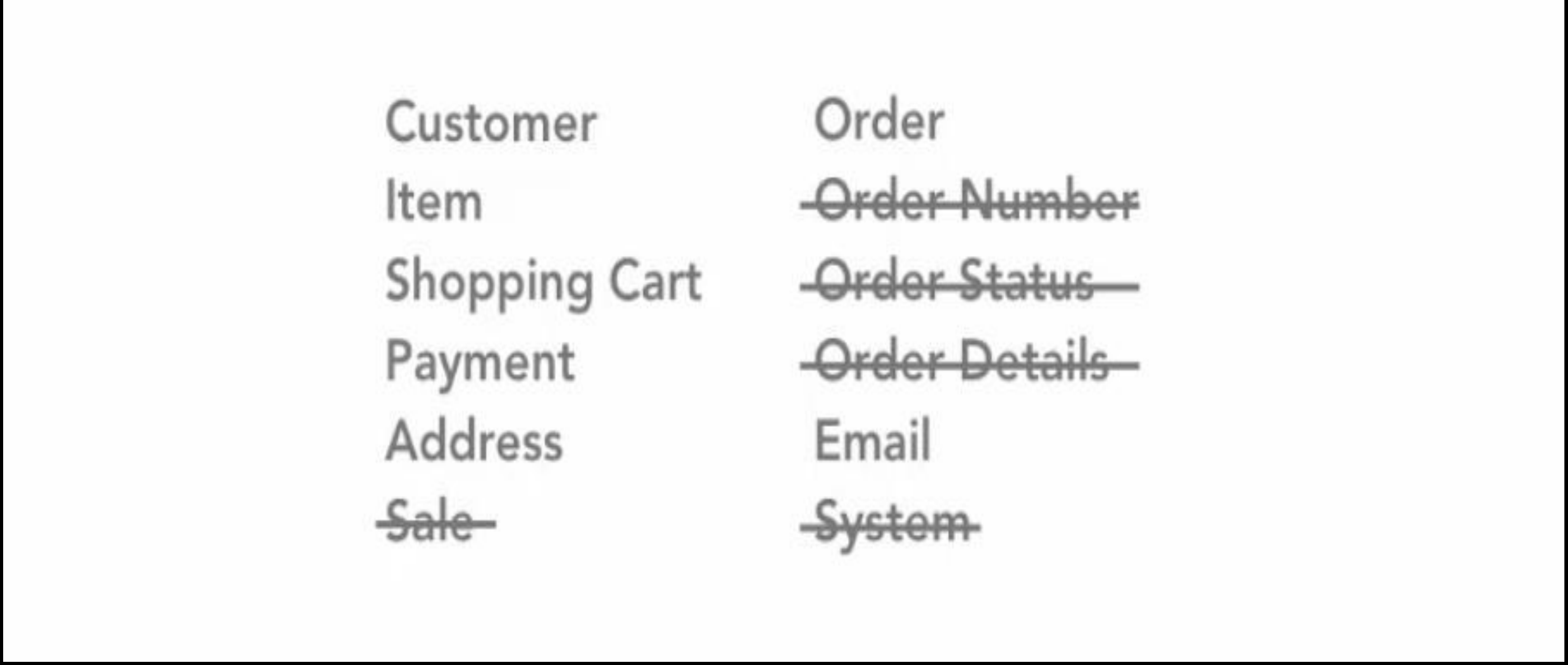


Use Case Scenario: Customer confirms items in shopping cart. Customer provides payment and address to process sale. System validates payment and responds by confirming order, and provides order number that Customer can use to check on order status. System will send Customer a copy of order details by email.

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)*

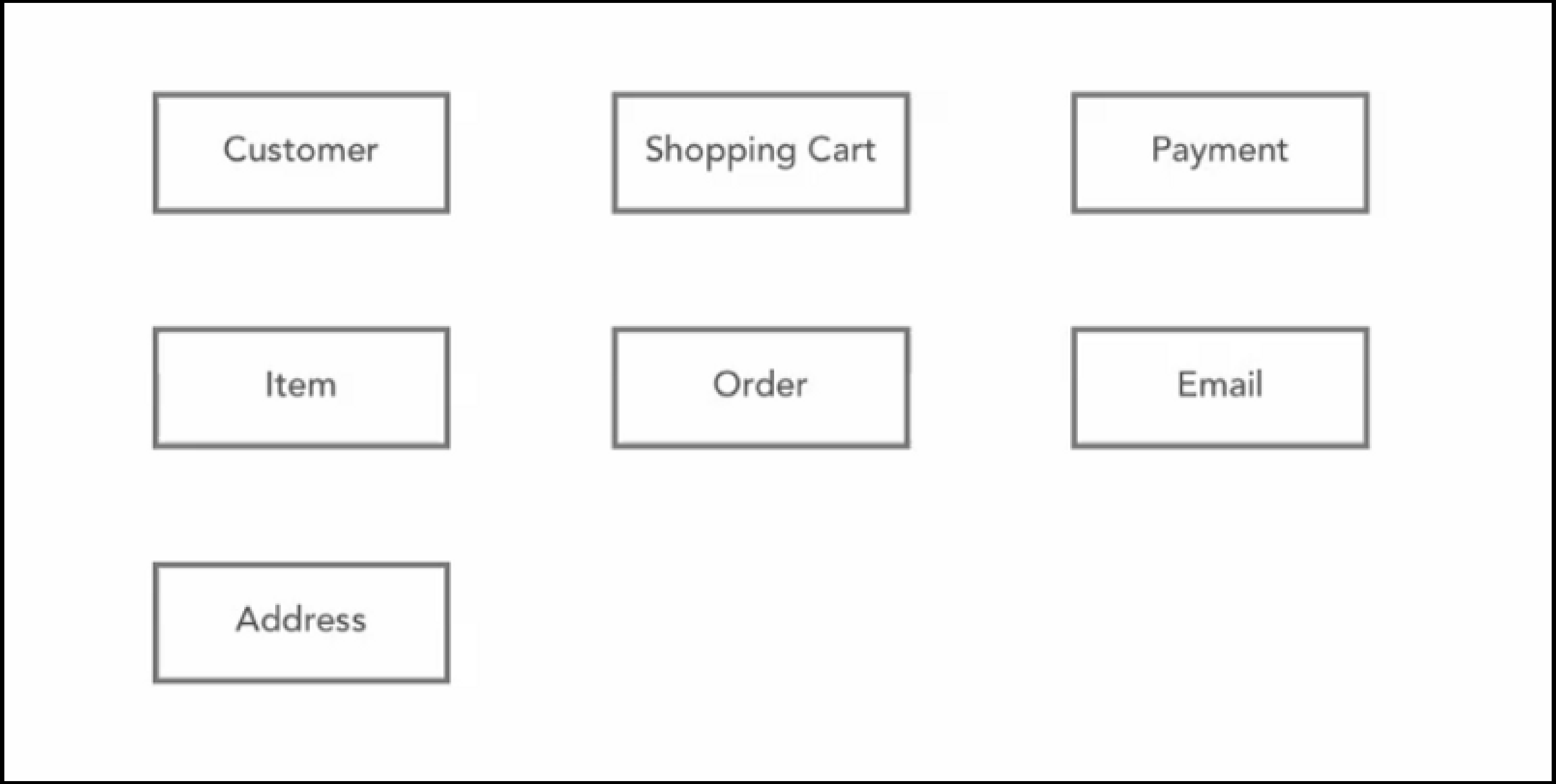
Refine Objects



Customer
Item
Shopping Cart
Payment
Address
~~Sale~~

Order
~~Order Number~~
~~Order Status~~
~~Order Details~~
Email
~~System~~

Drawing Objects



Attributes

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

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

Rule

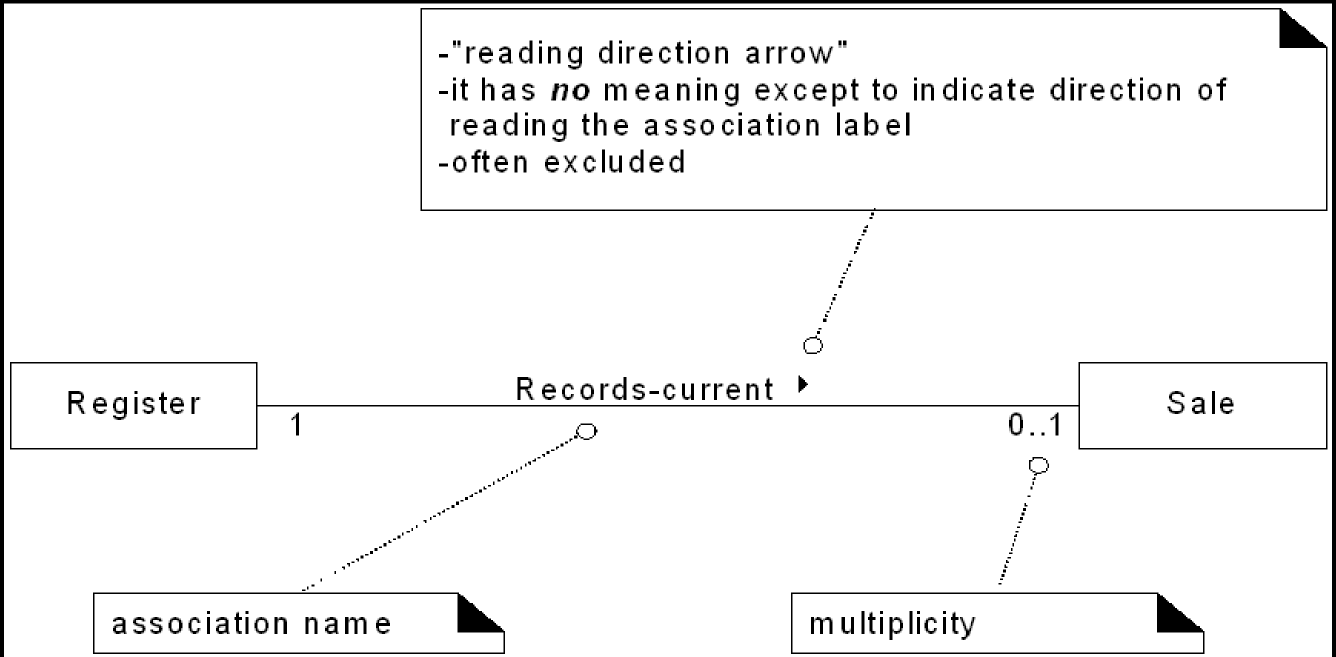
- ***If we do not think of a thing as a number or text in the real world, then it is probably a conceptual class.***
- ***If it takes up space, then it is likely a conceptual class.***

***Examples*:**

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

Identifying Object Relationships-Associations

- 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
Register
Records-current ▸
Sale
1
0..1
association name
multiplicity

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
- 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 an event related to B
  - Sale- Customer
- A is recorded in B
  - Salel-Register
- A is an organization subunit of B .
  - Departement - Store

Common association list

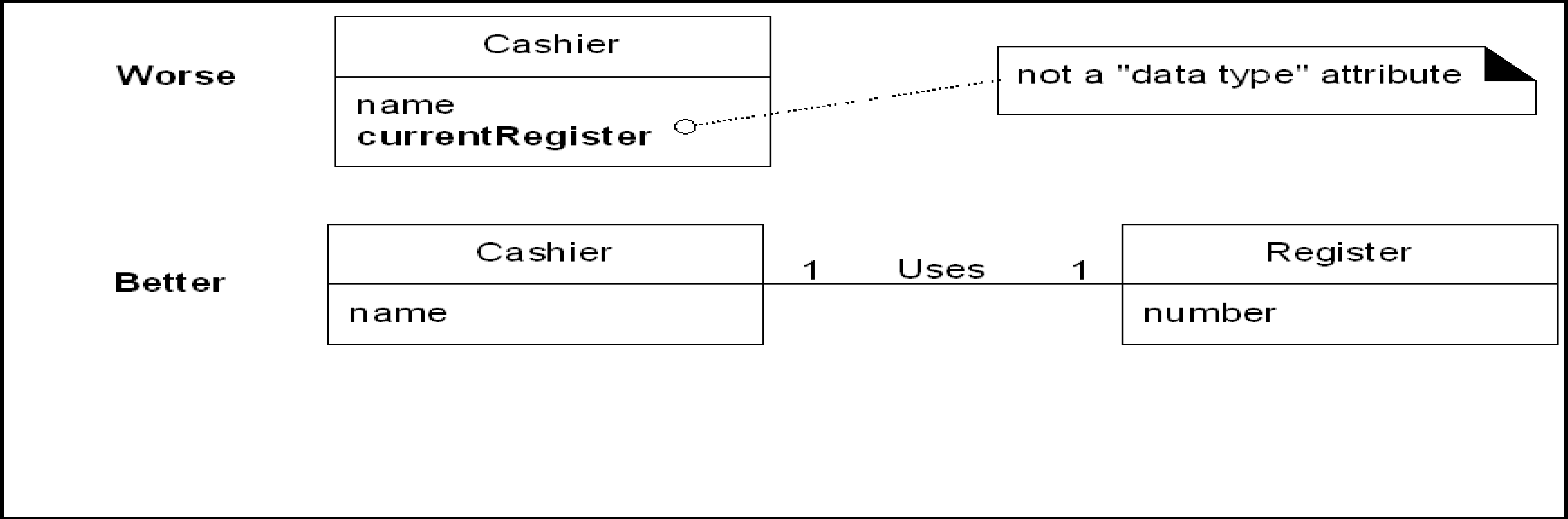
- A communicate with B
  - Customer - Cashier
- A is related to a transaction B
  - Customer - Payment
- A is a transaction related to another transaction B .
  - Payment - Sale
- A is owned by B
  - Register - Store

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

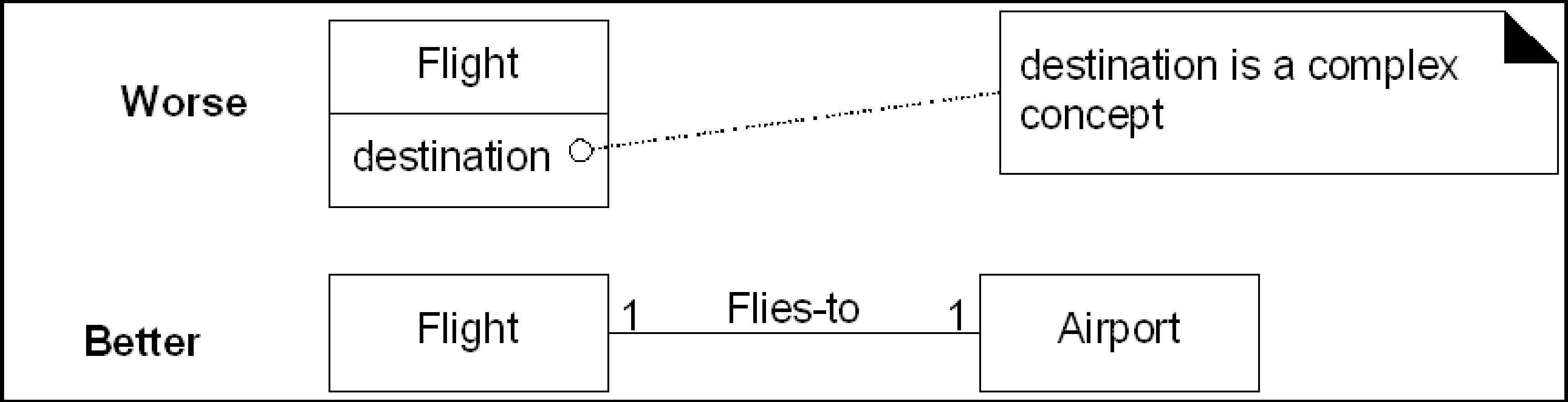
- To avoid:
  - ***Avoid showing redundant or derivable associations***
  - ***Do not overwhelm the domain model with associations not strongly required***

Association or attribute ?



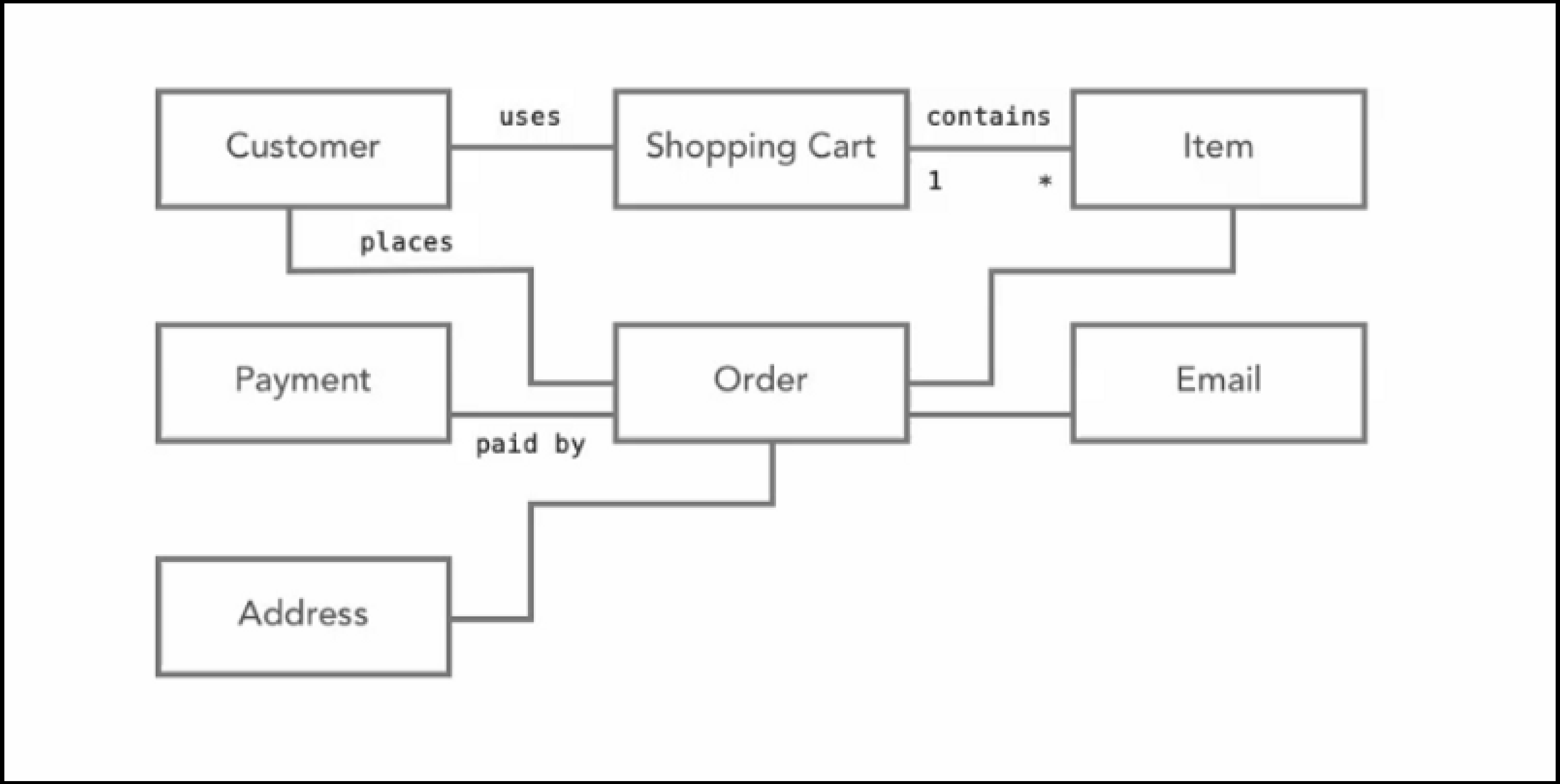
- ❑ *Most attribute type should be "**primitive**" data type, such as: numbers , string or boolean (true or false)*
- ❑ *Attribute should not be a complex domain concept(Sale , Airport)*
- ❑ *CurrentRegister is of type "Register", so expressed with an association*

Association or attribute ?



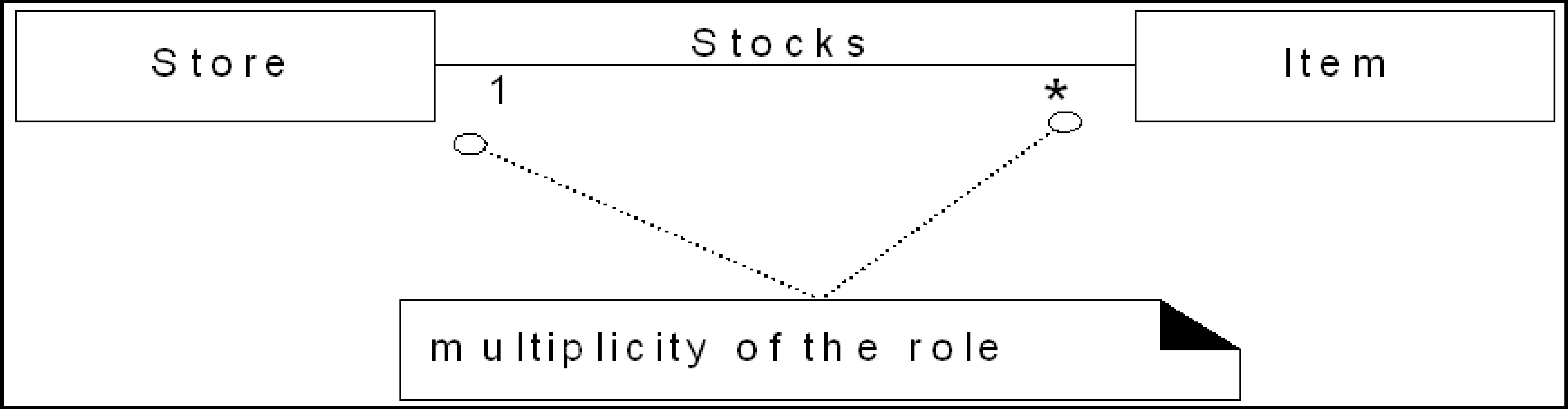
A destination airport is ***not a string***, it is a complex thing that occupies many square kilometers of space. So “Airport” should be related to “Flight” via an association , not with attribute

Identifying Object Relationships



Multiplicity

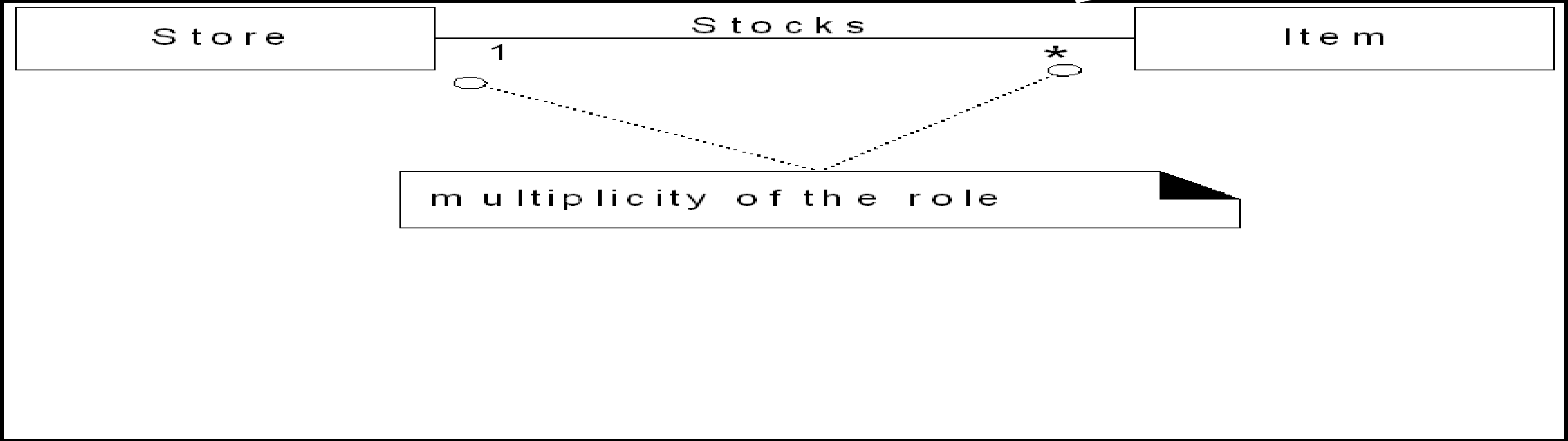
“many”
(0 or more)



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

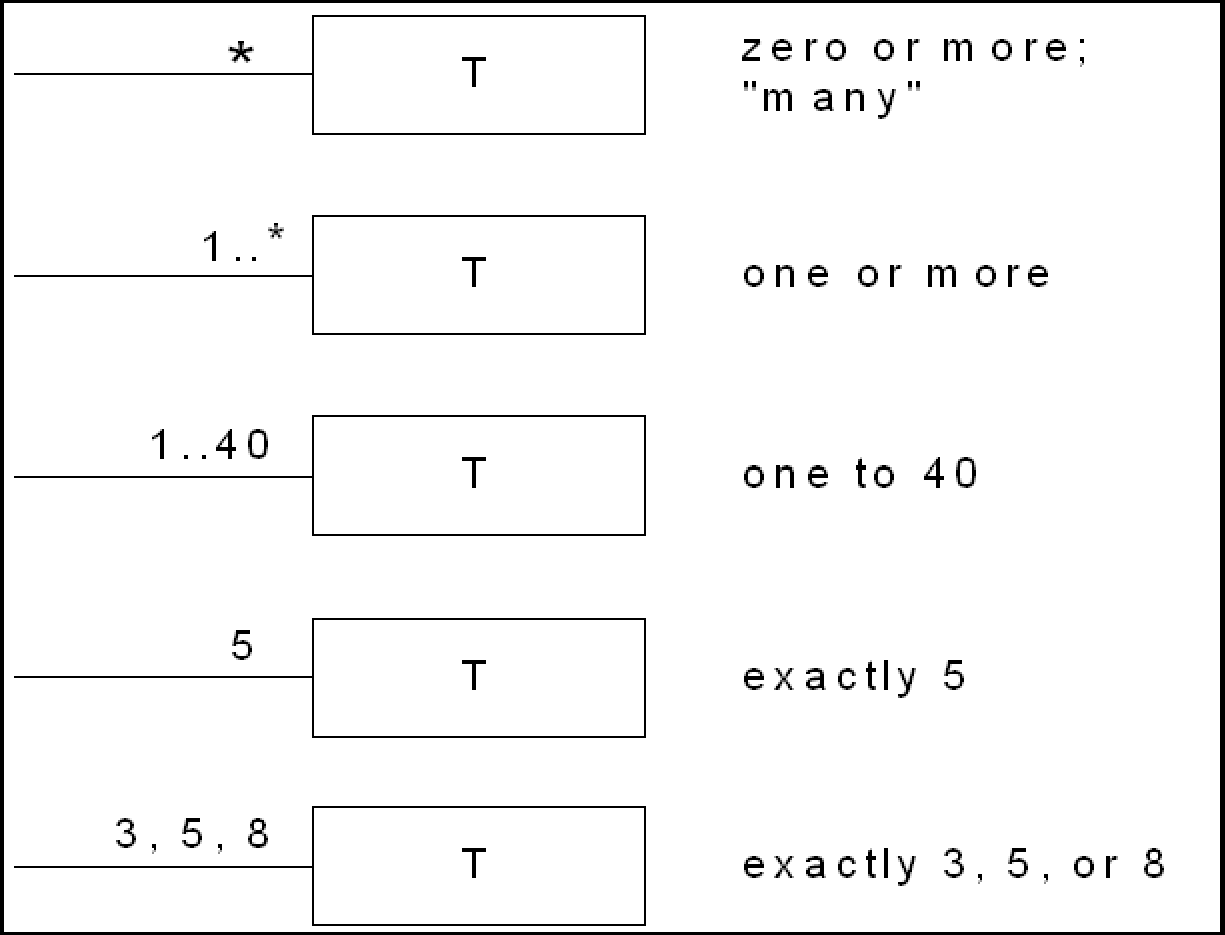
How to determine multiplicity ?

“many”
(0 or more)



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

Multiplicity



zero or more; "many"

one or more

one to 40

exactly 5

exactly 3, 5, or 8

How to create a domain model

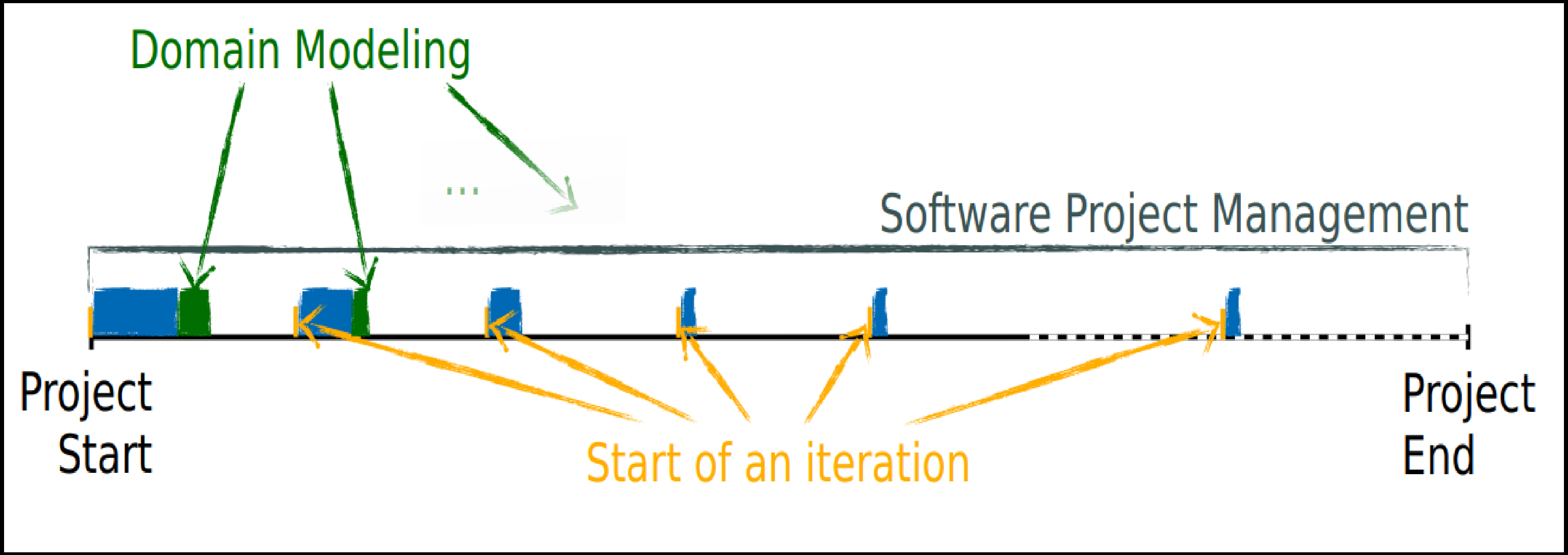
- *Identify candidate conceptual classes*
- *Go through them*
  - ***Exclude irrelevant features and duplications***
  - ***Do not add things that are outside the scope***
- *Draw them as classes in a UML class diagram*
- *Add associations necessary to record the relationship that must be retained*
-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

The goal of this lecture is to enable you to systematically carry out small(er) software projects that produce well-designed software.



That is all