Chapter 4
Domain Modeling

Dr. Supakit Nootyaskool

Faculty of Information Technology

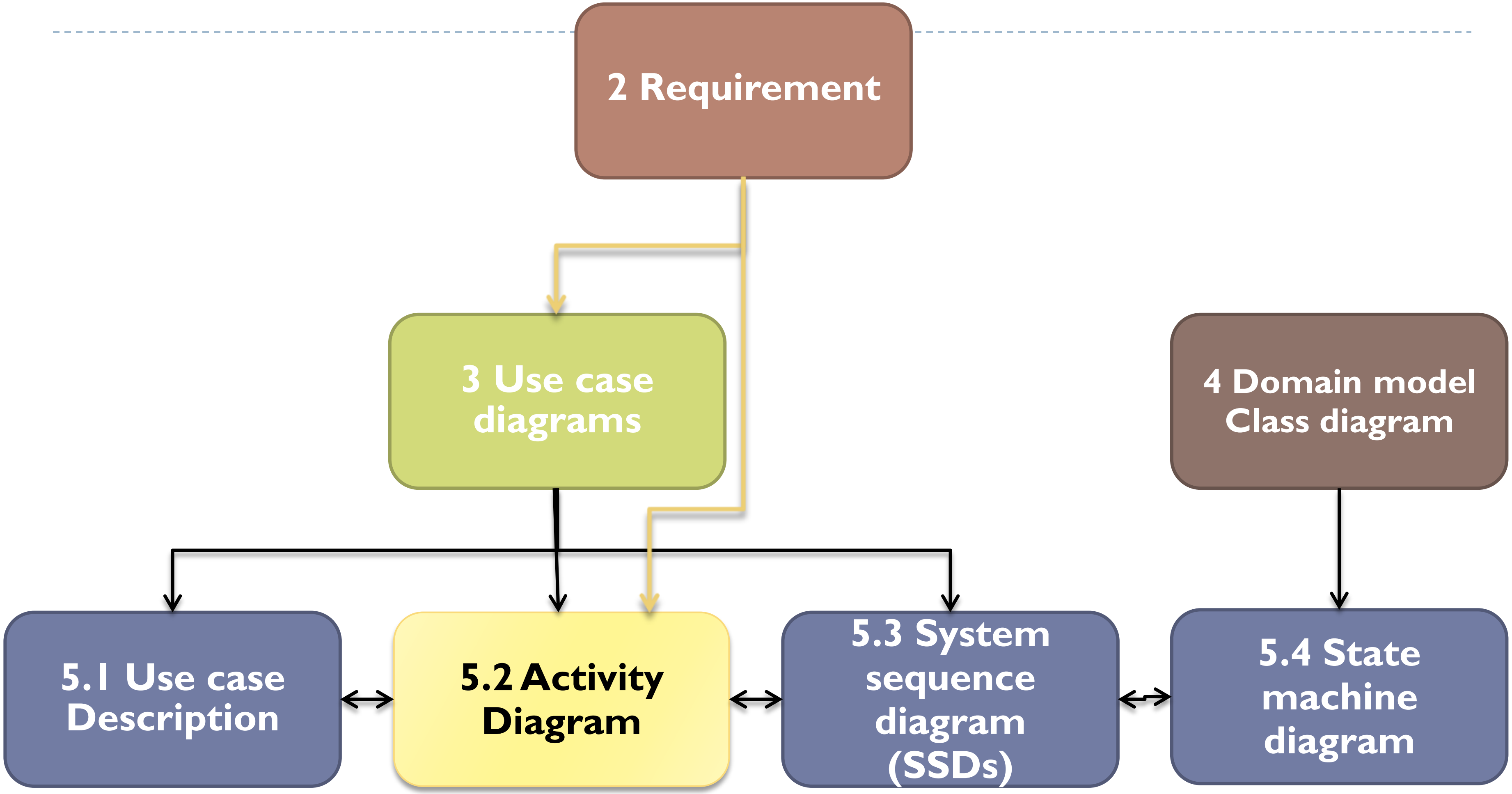
King Mongkut's Institute of Technology Ladkrabang

12-Dec-2013, Version2.1, supakitnootyaskool@gmail.com

Topics

- Problem domain – "Thing"
- Domain model class diagram
- Entities Relationship Diagram

Review2.0


2 Requirement

3 Use case diagrams

4 Domain model Class diagram

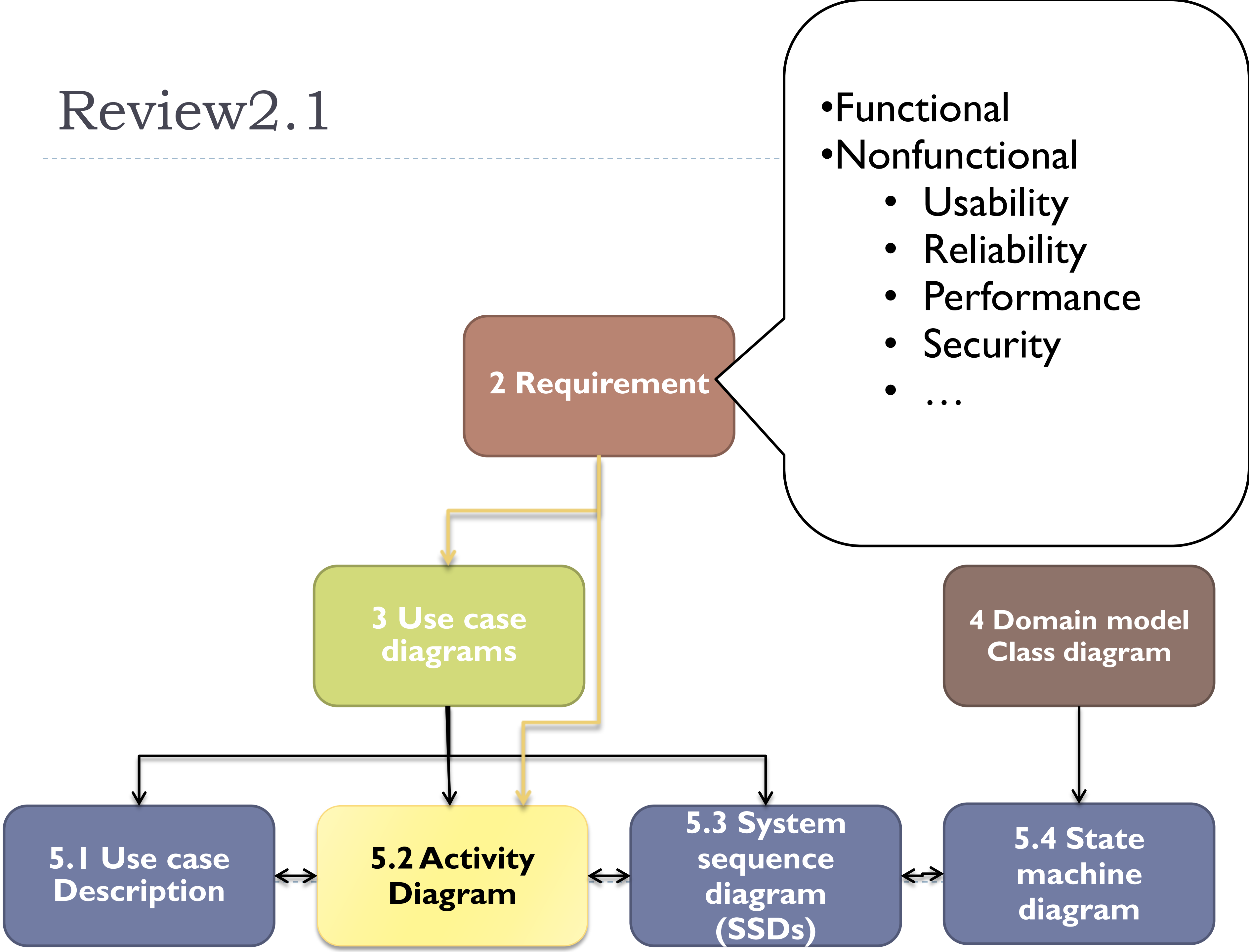
5.1 Use case Description

5.2 Activity Diagram

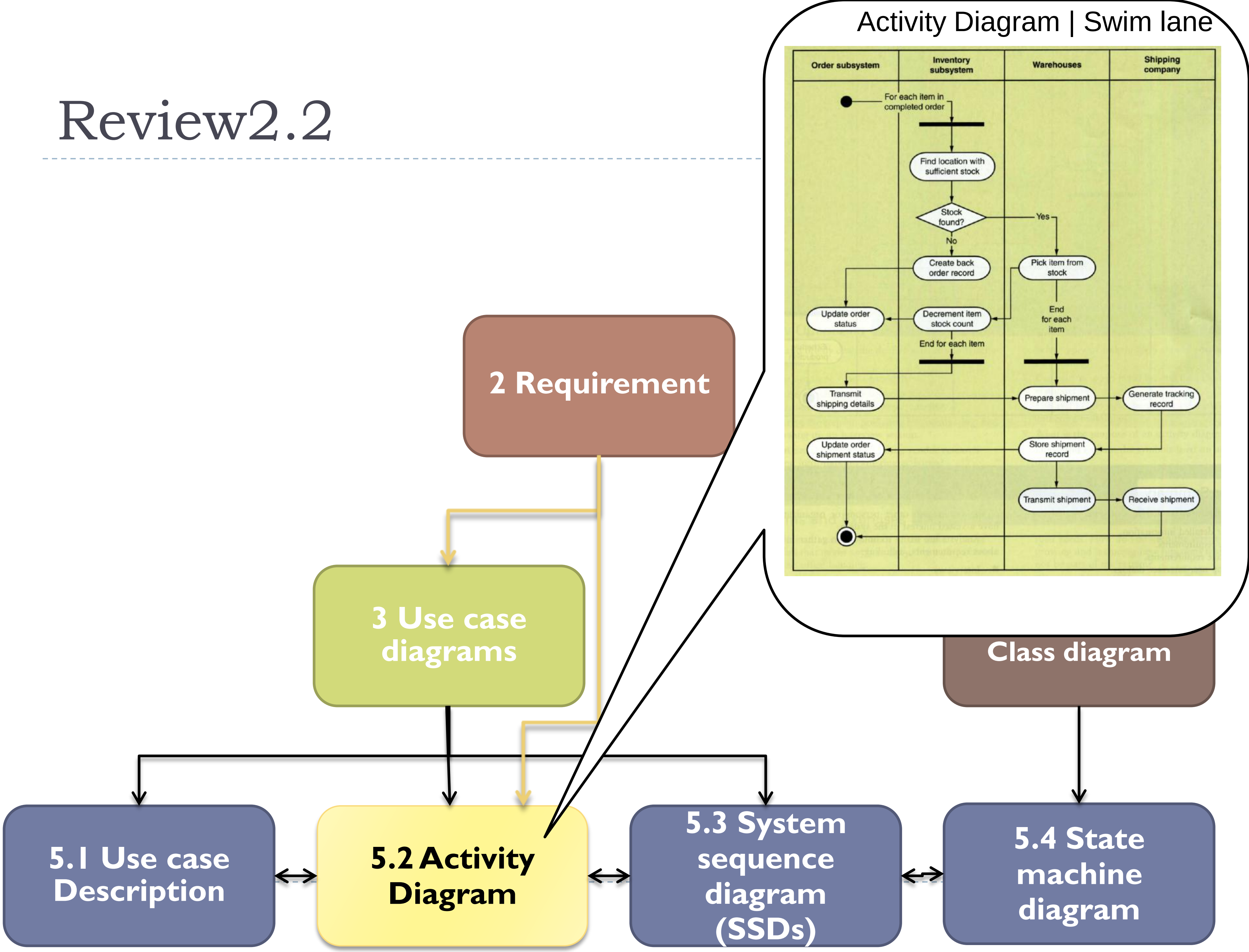
5.3 System sequence diagram (SSDs)

5.4 State machine diagram

Review2.1


•Functional
•Nonfunctional
• Usability
• Reliability
• Performance
• Security
• …
2 Requirement
3 Use case diagrams
4 Domain model Class diagram
5.1 Use case Description
5.2 Activity Diagram
5.3 System sequence diagram (SSDs)
5.4 State machine diagram

Review2.2



Activity Diagram | Swim lane

2 Requirement

3 Use case diagrams

Class diagram

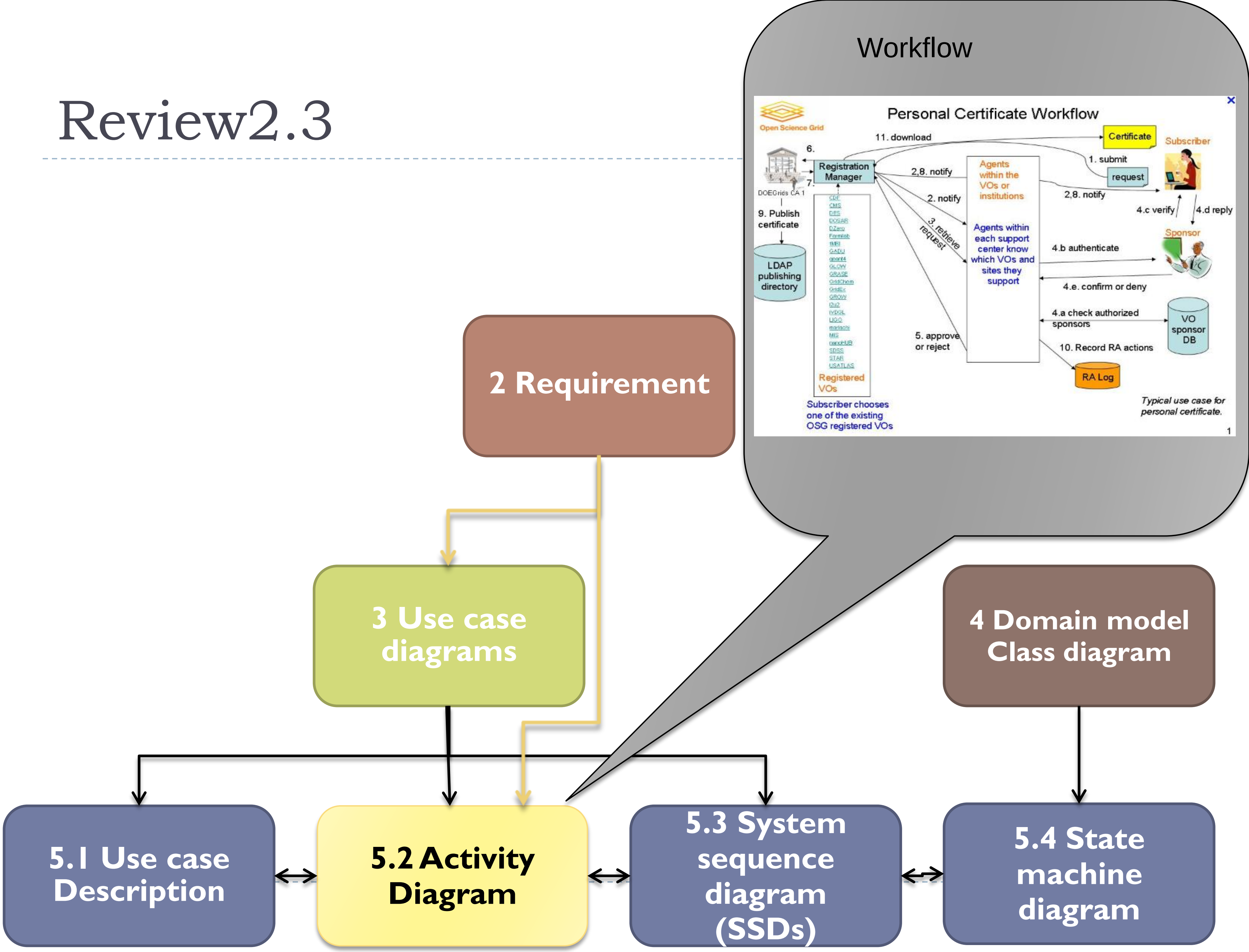
5.1 Use case Description

5.2 Activity Diagram

5.3 System sequence diagram (SSDs)

5.4 State machine diagram

Review2.3



2 Requirement

3 Use case diagrams

4 Domain model Class diagram

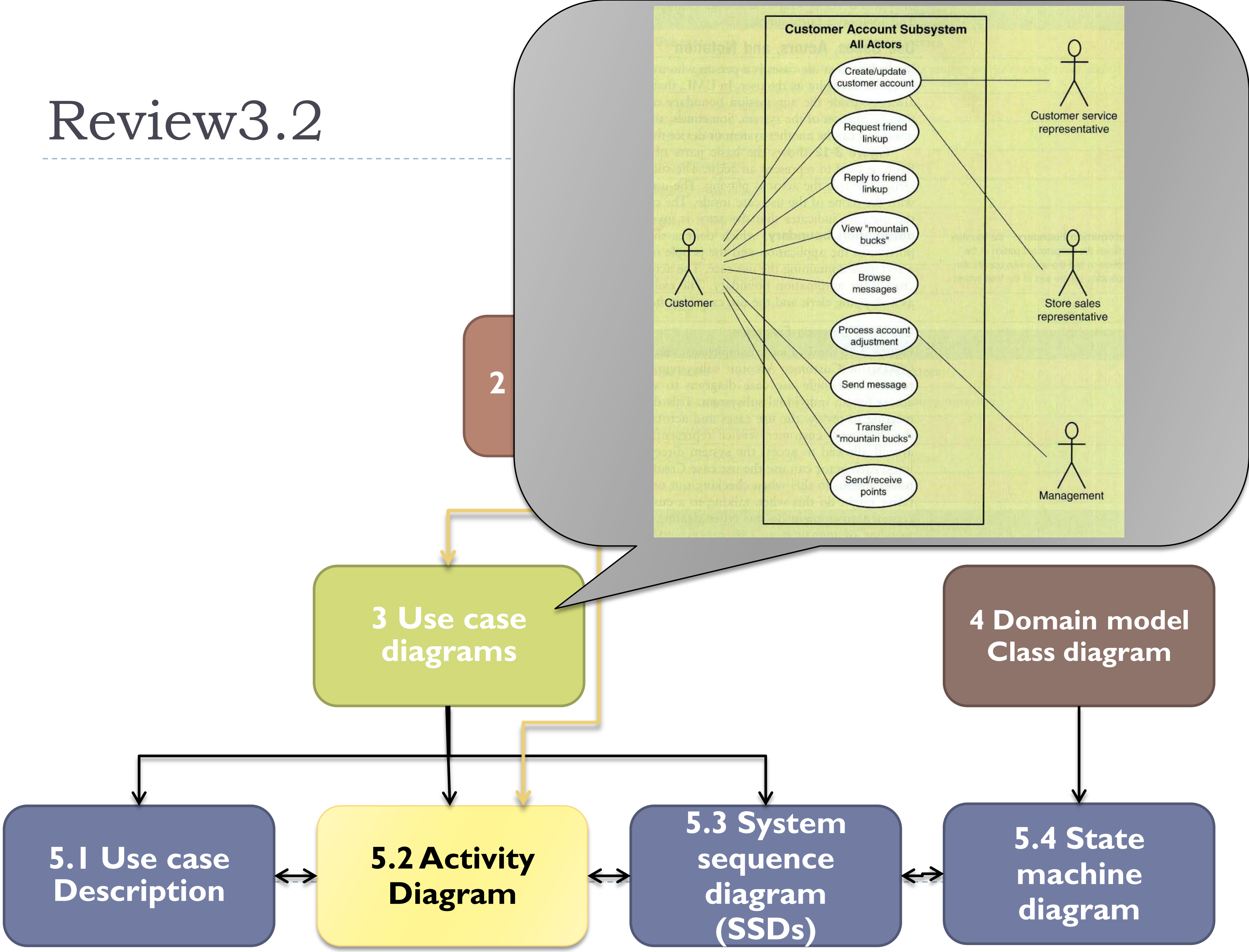
5.1 Use case Description

5.2 Activity Diagram

5.3 System sequence diagram (SSDs)

5.4 State machine diagram

Review3.2



Customer Account Subsystem
All Actors
Create/update customer account
Request friend linkup
Reply to friend linkup
View "mountain bucks"
Browse messages
Process account adjustment
Send message
Transfer "mountain bucks"
Send/receive points
Customer
Customer service representative
Store sales representative
Management

2

3 Use case diagrams

4 Domain model Class diagram

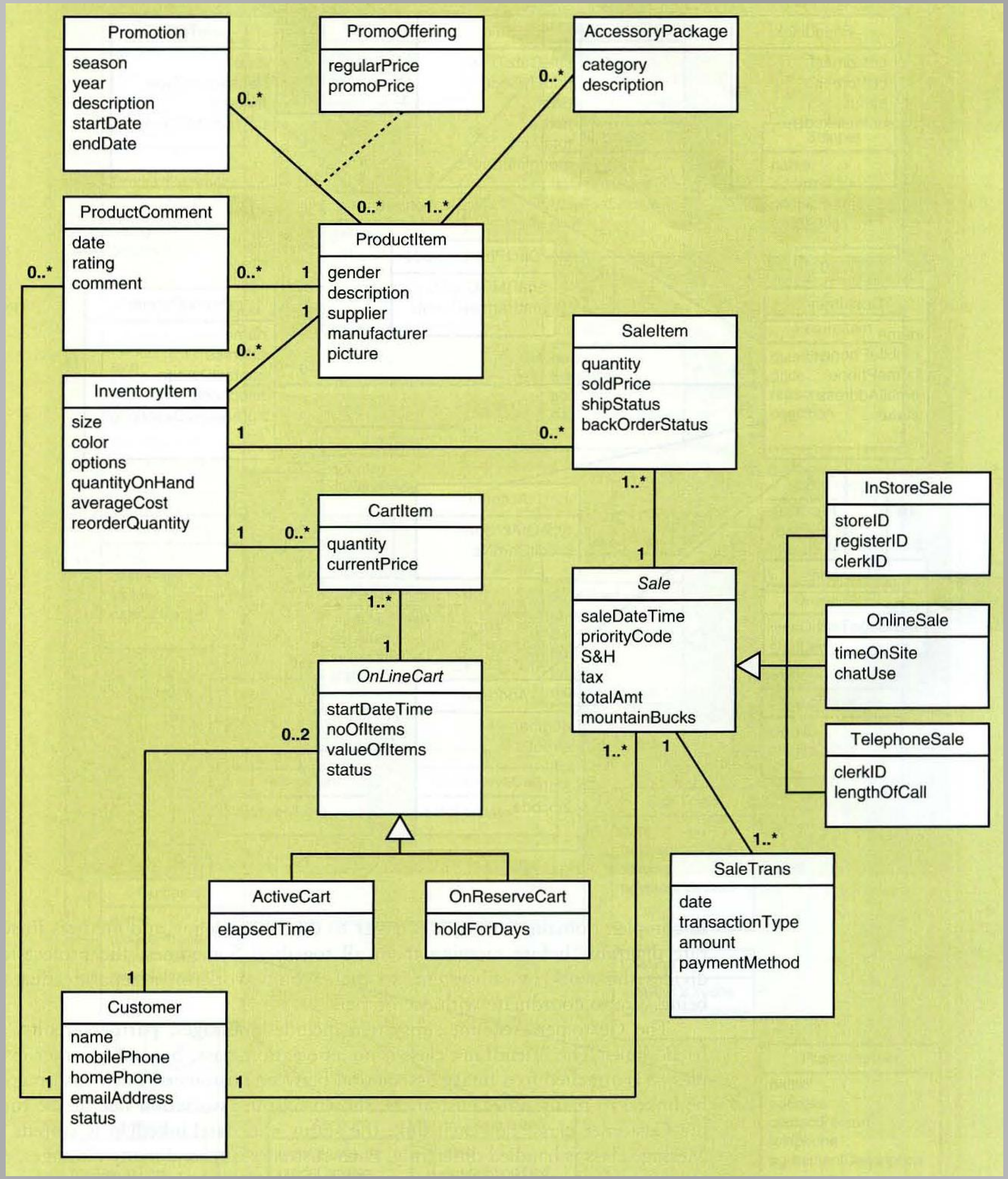
5.1 Use case Description

5.2 Activity Diagram

5.3 System sequence diagram (SSDs)

5.4 State machine diagram

R



4 Domain model Class diagram

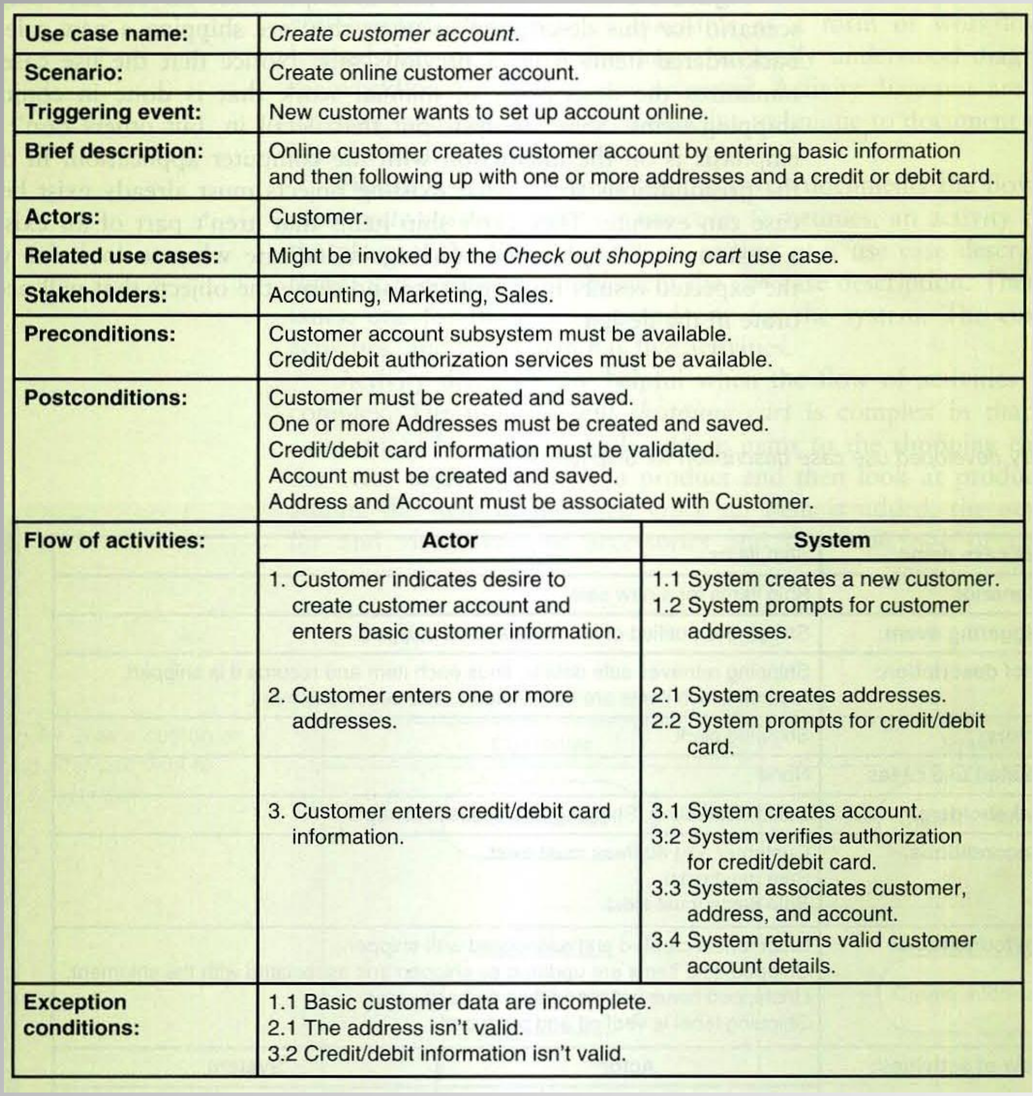
5.1 U

Descrip

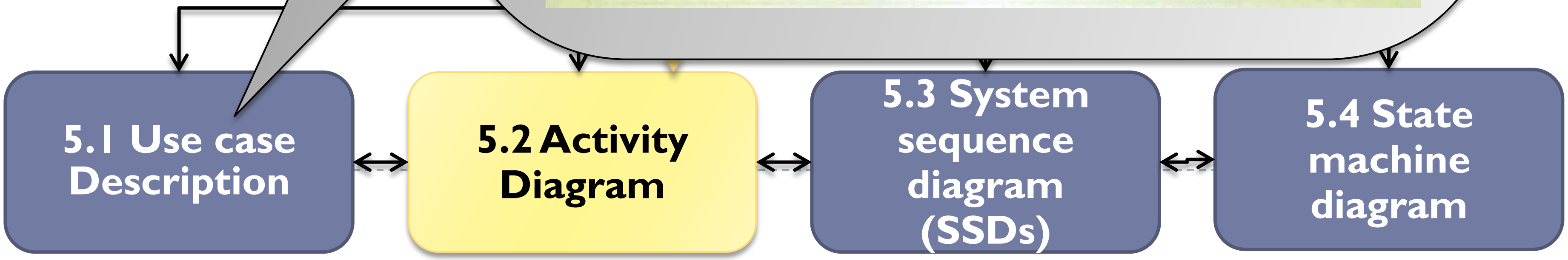
(SSDs)

5.4 State machine diagram

Review



Use case name:	*Create customer account.*	
Scenario:	Create online customer account.	
Triggering event:	New customer wants to set up account online.	
Brief description:	Online customer creates customer account by entering basic information and then following up with one or more addresses and a credit or debit card.	
Actors:	Customer.	
Related use cases:	Might be invoked by the *Check out shopping cart* use case.	
Stakeholders:	Accounting, Marketing, Sales.	
Preconditions:	Customer account subsystem must be available. Credit/debit authorization services must be available.	
Postconditions:	Customer must be created and saved. One or more Addresses must be created and saved. Credit/debit card information must be validated. Account must be created and saved. Address and Account must be associated with Customer.	
Flow of activities:	**Actor**	**System**
	1. Customer indicates desire to create customer account and enters basic customer information.	1.1 System creates a new customer. 1.2 System prompts for customer addresses.
	2. Customer enters one or more addresses.	2.1 System creates addresses. 2.2 System prompts for credit/debit card.
	3. Customer enters credit/debit card information.	3.1 System creates account. 3.2 System verifies authorization for credit/debit card. 3.3 System associates customer, address, and account. 3.4 System returns valid customer account details.
Exception conditions:	1.1 Basic customer data are incomplete. 2.1 The address isn't valid. 3.2 Credit/debit information isn't valid.	



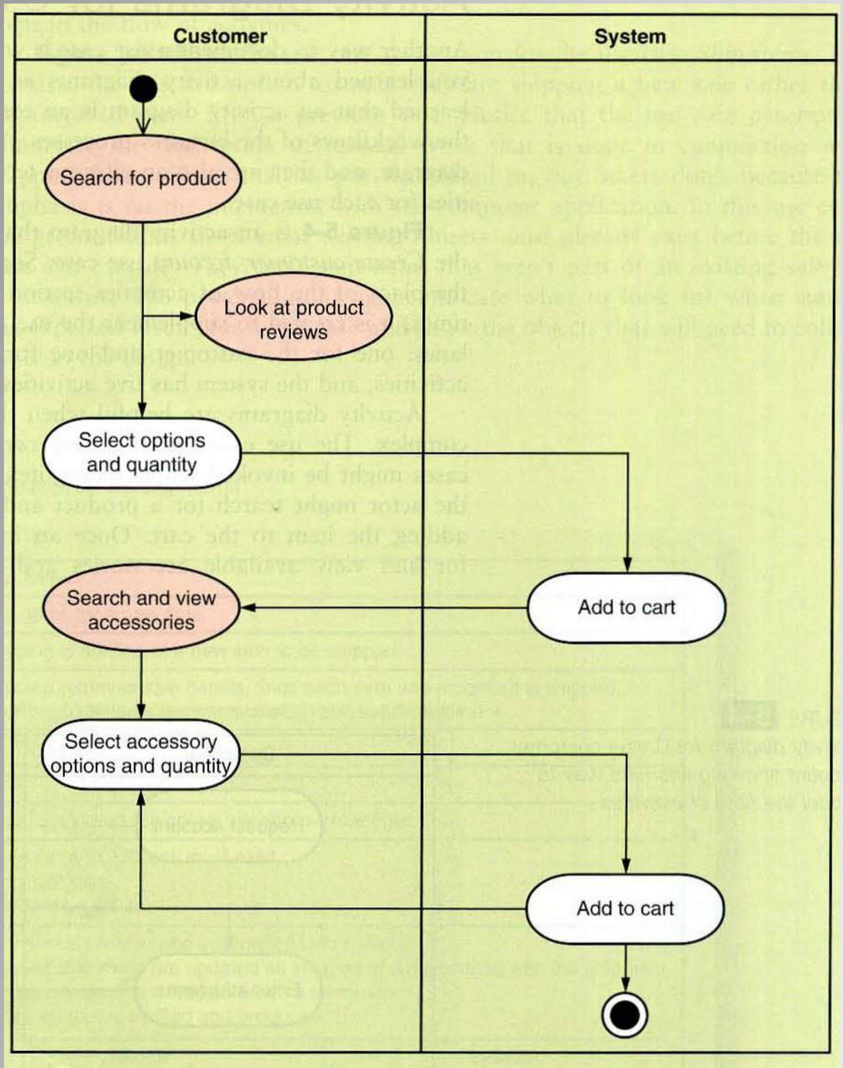
5.1 Use case Description

5.2 Activity Diagram

5.3 System sequence diagram (SSDs)

5.4 State machine diagram

Review


Customer
System
Search for product
Look at product reviews
Select options and quantity
Add to cart
Search and view accessories
Select accessory options and quantity
Add to cart

Domain model Class diagram

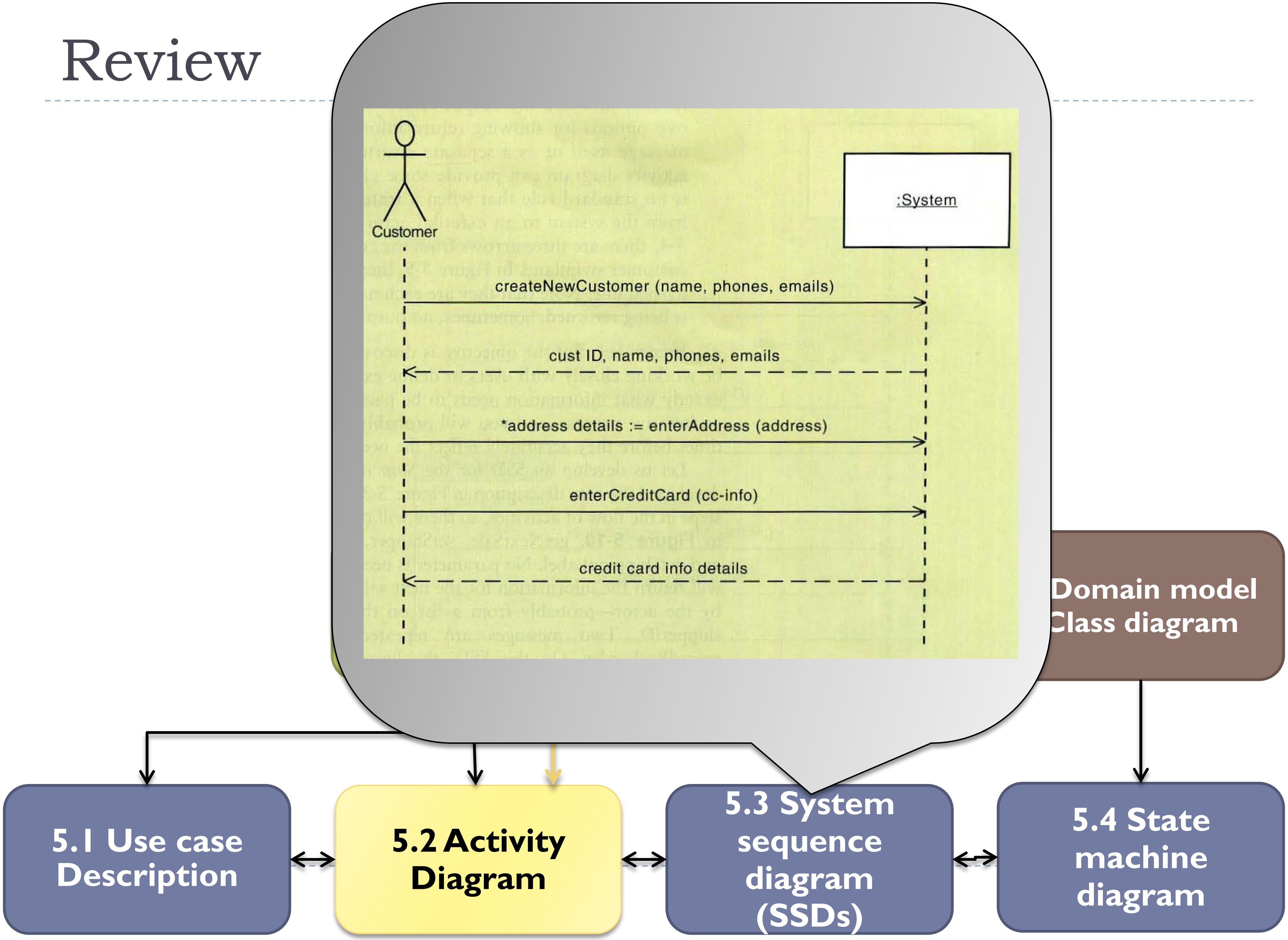
5.1 Use case Description

5.2 Activity Diagram

5.3 System sequence diagram (SSDs)

5.4 State machine diagram

Review



Customer
:System
createNewCustomer (name, phones, emails)
cust ID, name, phones, emails
*address details := enterAddress (address)
enterCreditCard (cc-info)
credit card info details

Domain model Class diagram

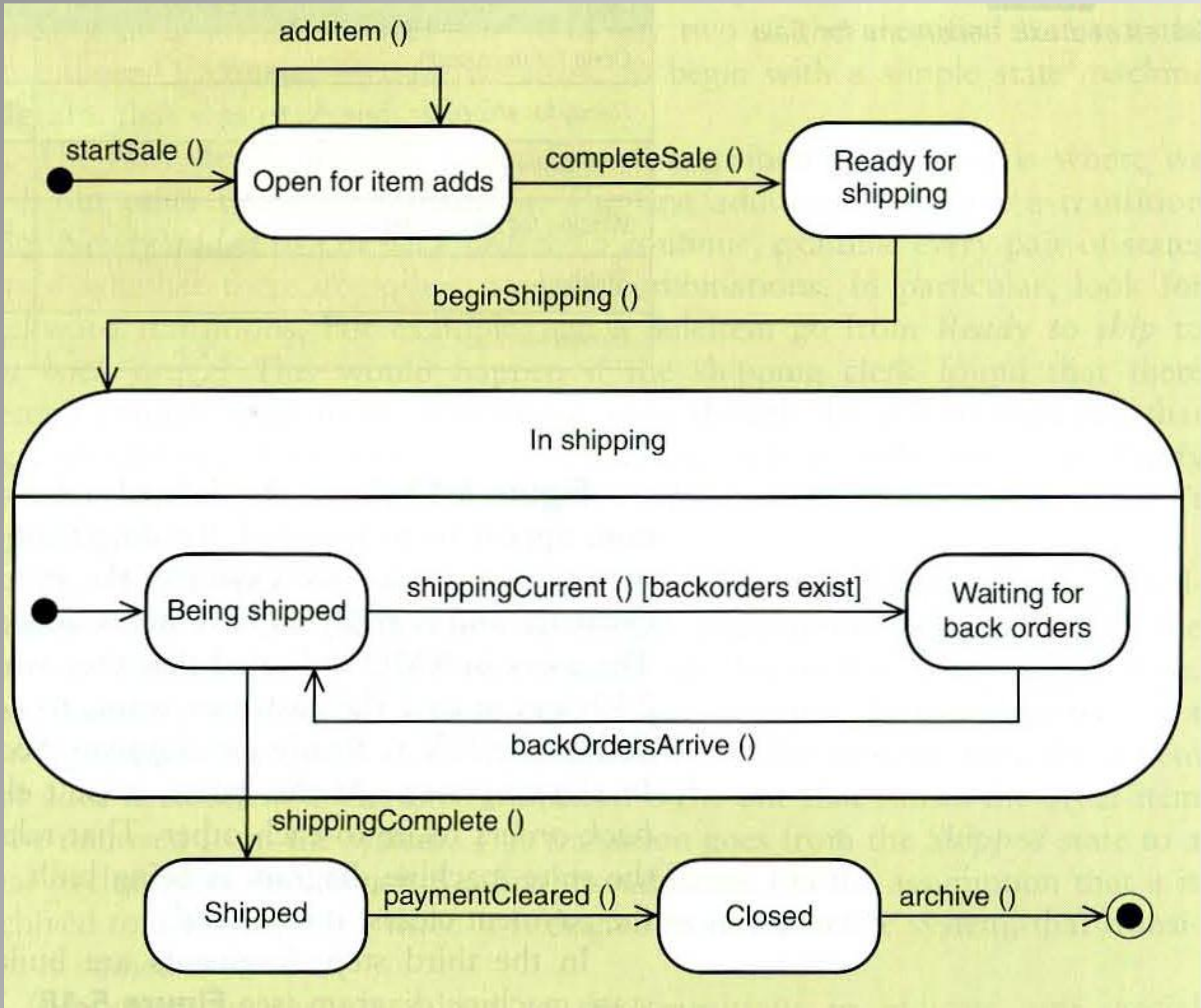
5.1 Use case Description

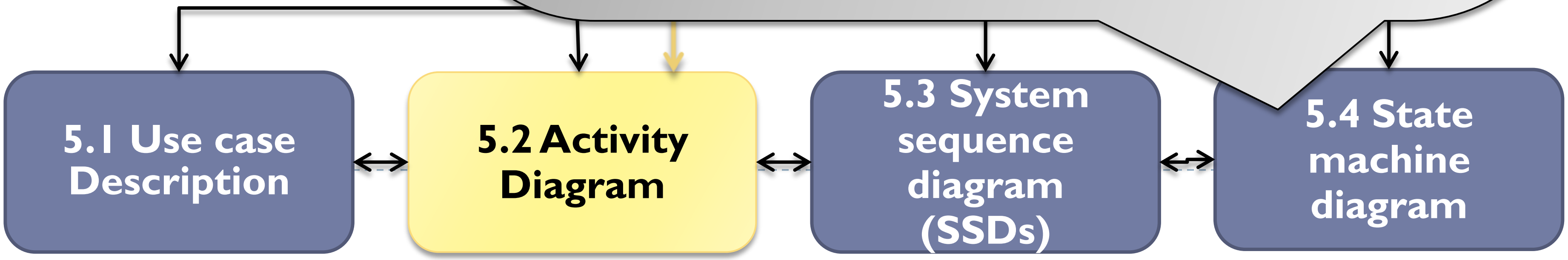
5.2 Activity Diagram

5.3 System sequence diagram (SSDs)

5.4 State machine diagram

Review




5.1 Use case Description

5.2 Activity Diagram

5.3 System sequence diagram (SSDs)

5.4 State machine diagram

Chapter 4

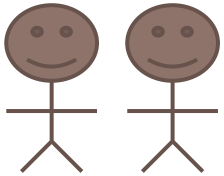
- “Things” in the problem domain
- The entity-relationship diagram
- The domain model class diagram

Learning objective

After learning, you should be able to

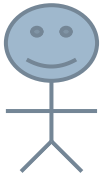
- Explain how the concept of "things" in problem domain also define the requirement
- Identify and analysis data entity and domain class needed in the system
- Read, interpret, and create ER diagram
- Read, interpret, and create domain class diagram

4.0 Case study
Waiters on call meal-delivery system



Sue and Tom want to a new “meal-delivery system” that specific

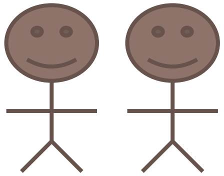
- Automatic
- Improve the business processes



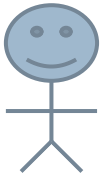
Sam, SA., stats to ask, to get information from them.

- Restaurants, menu items, customers, order
- I think you need to store information, ex: driver, address, route

4.0 Case study
Waiters on call meal-delivery system



Agreed and add their information
They want driver to be signed to a route based on the distance from place to place



Can you tell me if driver pick up order from several restaurants when they go out?

Can you tell me how many items are usually included in one order?

Do you note pickup time and delivery time?

Do you need to plan the route so that hot dishes are delivered first?

4.1 Things

4.1 “Things” in the problem domain

- Problem Domain
  - The specific area of the user’s business that is included within the scope of the new system.

- “Things”
  - Things are related to the people who interact with system or the other stakeholder.
  - Example
    - Store information about customer and product
    - Store information about products, shipments, warehouse

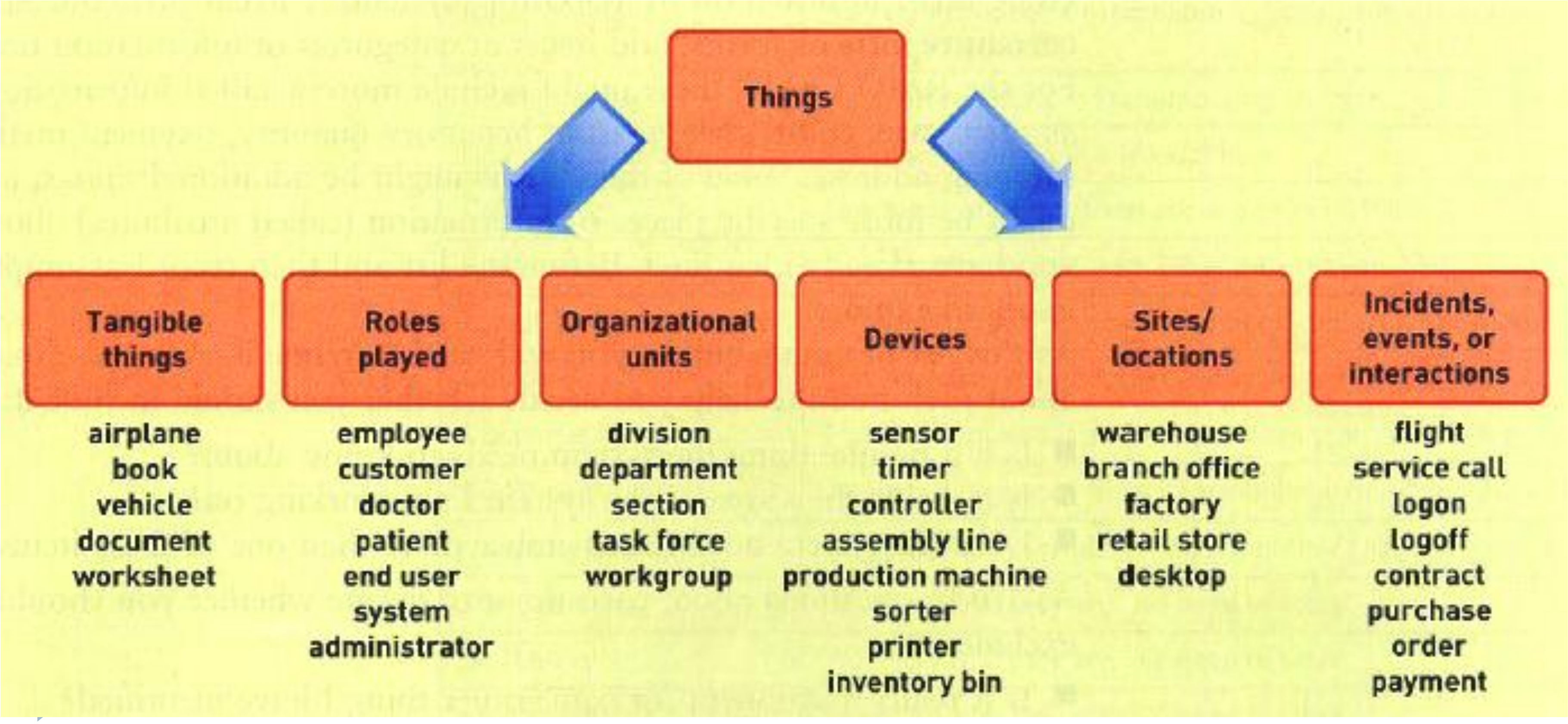
4.1 “Things” in the problem domain (2)

- Many techniques to identify “Thing” in problem domain
  - Brainstorming technique
  - Noun technique

- Things can be divided into
  - Tangible (easy to identify, most obvious)
  - Intangible (difficult identify)

4.1.1 Brainstorming technique

- Use to identify tangible

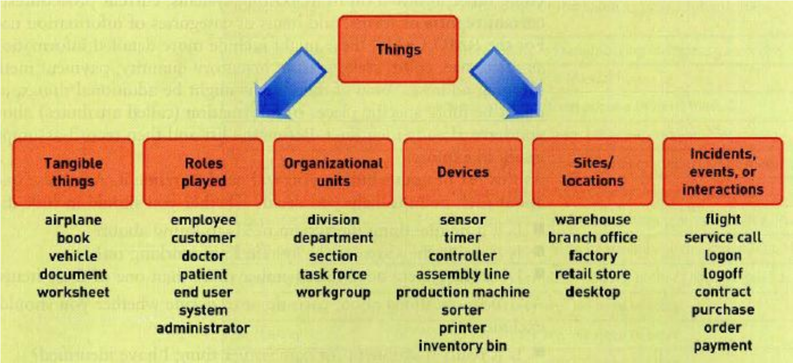


Things

Tangible things	Roles played	Organizational units	Devices	Sites/ locations	Incidents, events, or interactions
airplane	employee	division	sensor	warehouse	flight
book	customer	department	timer	branch office	service call
vehicle	doctor	section	controller	factory	logon
document	patient	task force	assembly line	retail store	logoff
worksheet	end user	workgroup	production machine	desktop	contract
	system		sorter		purchase
	administrator		printer		order
			inventory bin		payment

4.1.1 Brainstorming technique: Steps

1. Identify a user and a set of use cases.
2. Brainstorm with user to carry out the use case
3. Use the types of things to ask question to user,
    1. "Are there any tangible things you store information about?
    2. Are there roles play by people that you need to remember?
4. Continuous to work all types of users and stakeholders
5. Merge the results, cut duplicate, compile an initial list

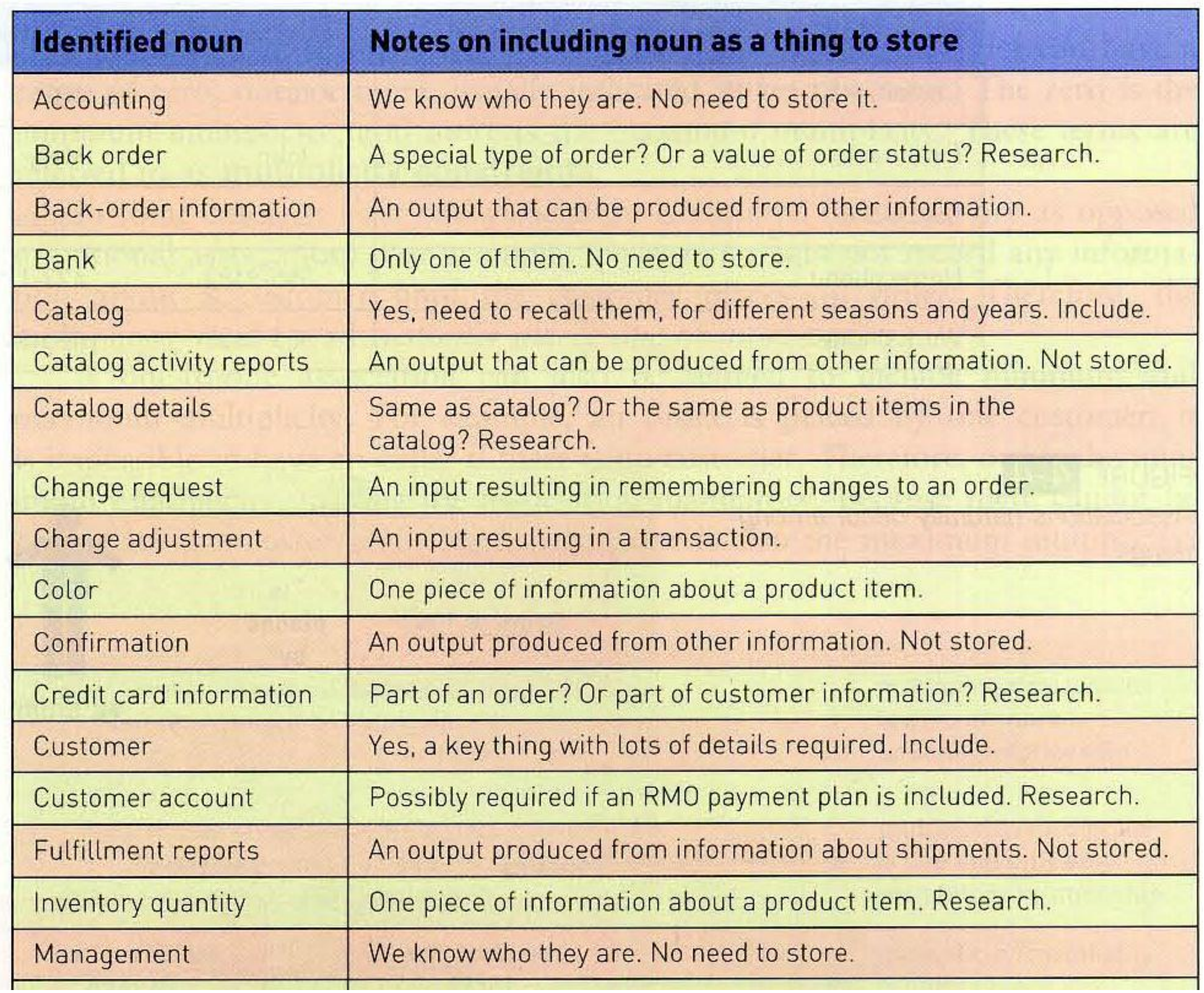


Things

Tangible things	Roles played	Organizational units	Devices	Sites/ locations	Incidents, events, or interactions
airplane book vehicle document worksheet	employee customer doctor patient end user system administrator	division department section task force workgroup	sensor timer controller assembly line production machine sorter printer inventory bin	warehouse branch office factory retail store desktop	flight service call logon logoff contract purchase order payment

4.1.2 Noun technique

- Noun is
  - Person
  - Place
  - Thing
- Identify nouns might help you identify what needs to be stored by the system



Identified noun	Notes on including noun as a thing to store
Accounting	We know who they are. No need to store it.
Back order	A special type of order? Or a value of order status? Research.
Back-order information	An output that can be produced from other information.
Bank	Only one of them. No need to store.
Catalog	Yes, need to recall them, for different seasons and years. Include.
Catalog activity reports	An output that can be produced from other information. Not stored.
Catalog details	Same as catalog? Or the same as product items in the catalog? Research.
Change request	An input resulting in remembering changes to an order.
Charge adjustment	An input resulting in a transaction.
Color	One piece of information about a product item.
Confirmation	An output produced from other information. Not stored.
Credit card information	Part of an order? Or part of customer information? Research.
Customer	Yes, a key thing with lots of details required. Include.
Customer account	Possibly required if an RMO payment plan is included. Research.
Fulfillment reports	An output produced from information about shipments. Not stored.
Inventory quantity	One piece of information about a product item. Research.
Management	We know who they are. No need to store.

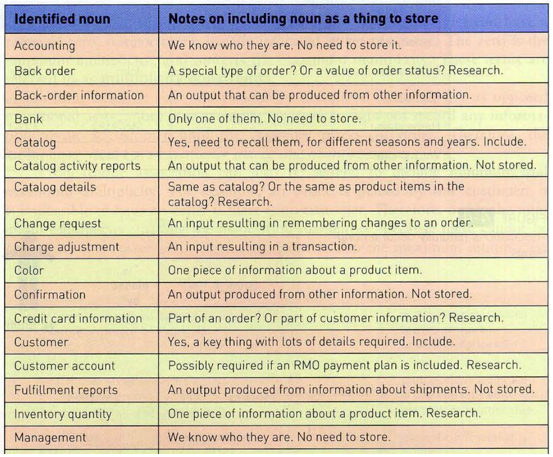
4.1.2 Noun technique: Steps

1. Using the use cases, actors and other information about the system including inputs and outputs – identify all nouns.

   Example RMO: customer, product item, sales, confirmation, transaction, shipping, bank, change request, summary report….

2. Using other information from existing system, current products and current report or form, add items or categories of information needed.

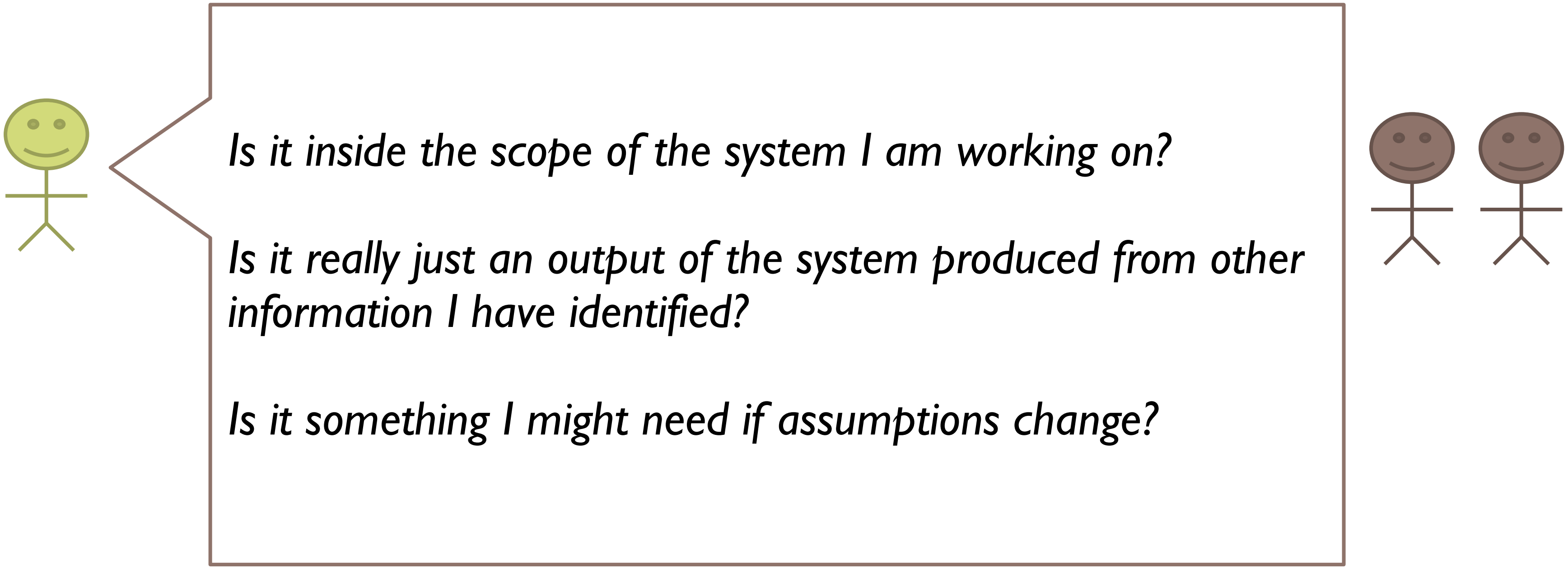
   Example RMO: price, size, color, style, season, inventory quantity, payment method, shipping address,…



Identified noun	Notes on including noun as a thing to store
Accounting	We know who they are. No need to store it.
Back order	A special type of order? Or a value of order status? Research.
Back-order information	An output that can be produced from other information.
Bank	Only one of them. No need to store.
Catalog	Yes, need to recall them, for different seasons and years. Include.
Catalog activity reports	An output that can be produced from other information. Not stored.
Catalog details	Same as catalog? Or the same as product items in the catalog? Research.
Change request	An input resulting in remembering changes to an order.
Charge adjustment	An input resulting in a transaction.
Color	One piece of information about a product item.
Confirmation	An output produced from other information. Not stored.
Credit card information	Part of an order? Or part of customer information? Research.
Customer	Yes, a key thing with lots of details required. Include.
Customer account	Possibly required if an RMO payment plan is included. Research.
Fulfillment reports	An output produced from information about shipments. Not stored.
Inventory quantity	One piece of information about a product item. Research.
Management	We know who they are. No need to store.

4.1.2 Noun technique: Steps (2)

3. Use the list of nouns to ask question about each noun helping you decide whether you should include/exclude/research it.



Is it inside the scope of the system I am working on?

Is it really just an output of the system produced from other information I have identified?

Is it something I might need if assumptions change?

4.1.2 Noun technique: Steps (3)

4. Create a master list of all noun identified and then note each one should be include, excluded, or research further.

5. Review the list with users, stakeholders, and term members and then refine the list of things in the problem domain.

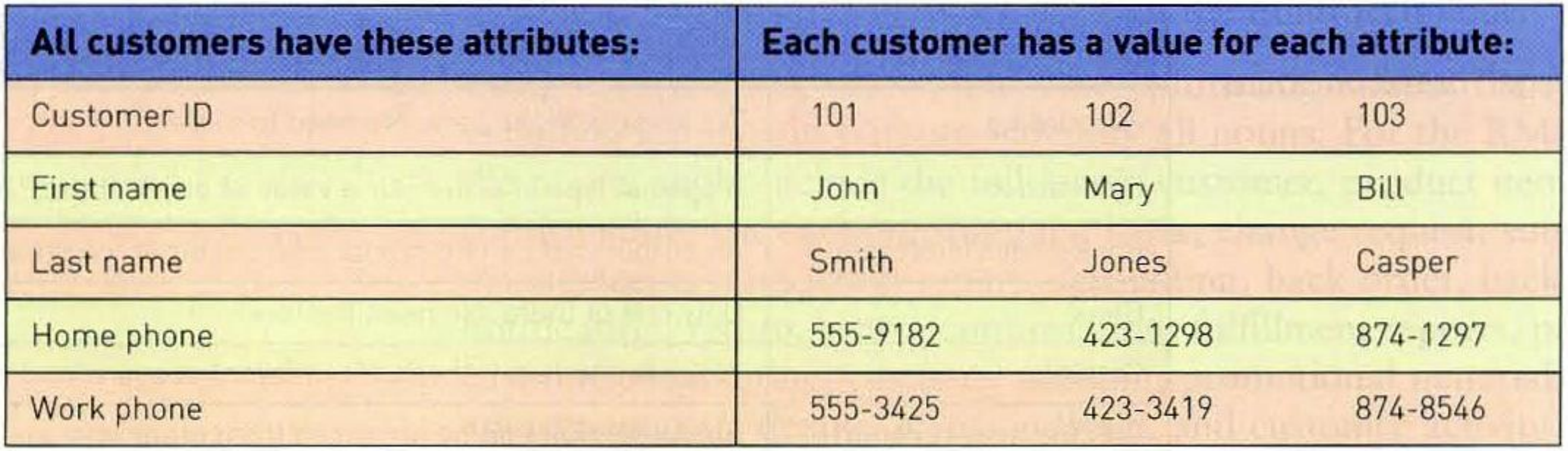
4.1.3 Attributes of things

- Attributes are the specific pieces of information, example
  - Customer has a name
  - Phone number
  - A credit limit

- The analyst needs to identify the attributes of each thing that system need to store.

4.1.3 Attributes of things

- Attribute is classified
  - Identifier OR key
    - Product ID
    - Vehicle ID
    - Etc..
  - Compound attribute
    - Consisting collection of related attribute

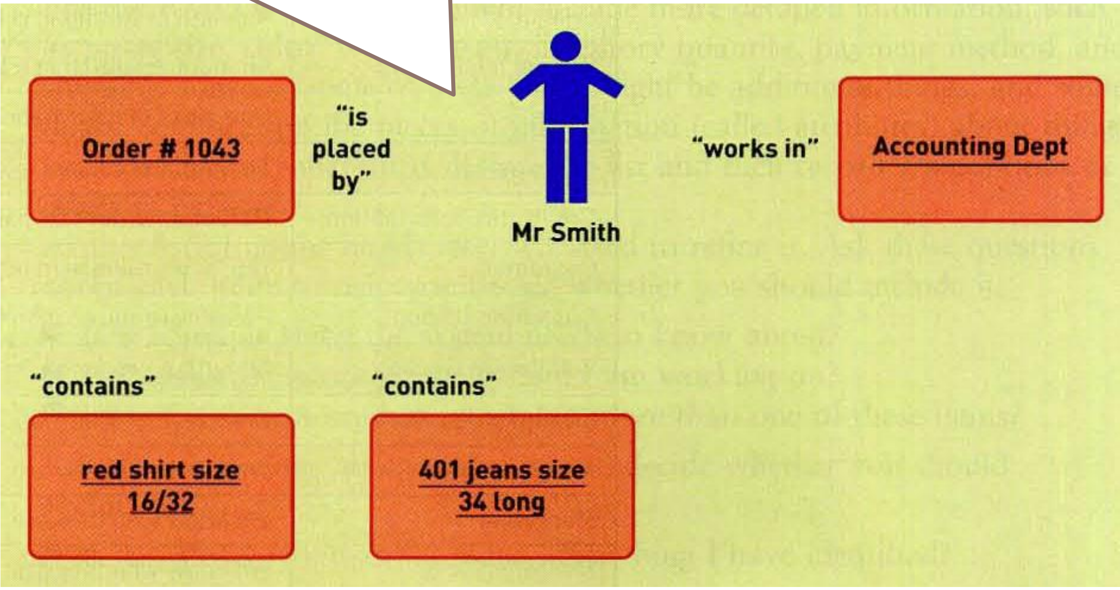


All customers have these attributes:	Each customer has a value for each attribute:		
Customer ID	101	102	103
First name	John	Mary	Bill
Last name	Smith	Jones	Casper
Home phone	555-9182	423-1298	874-1297
Work phone	555-3425	423-3419	874-8546

4.1.4 Association among things

- Attribute is naturally occurring relationship between specific things, such as

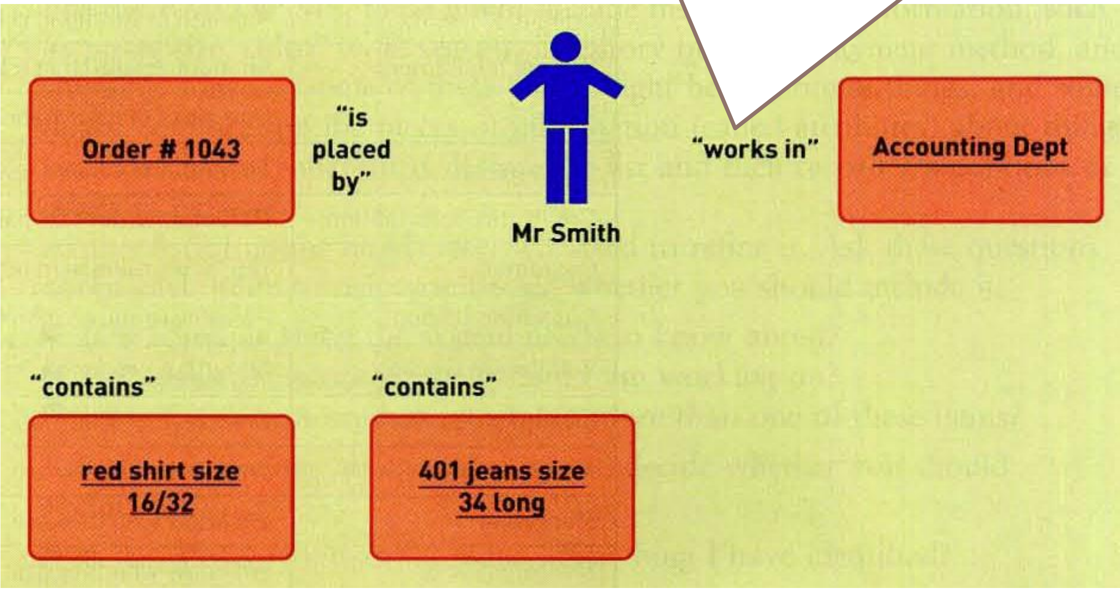
The order 1043 is placed by Smith



4.1.4 Association among things

- Attribute is naturally occurring relationship between specific things, such as

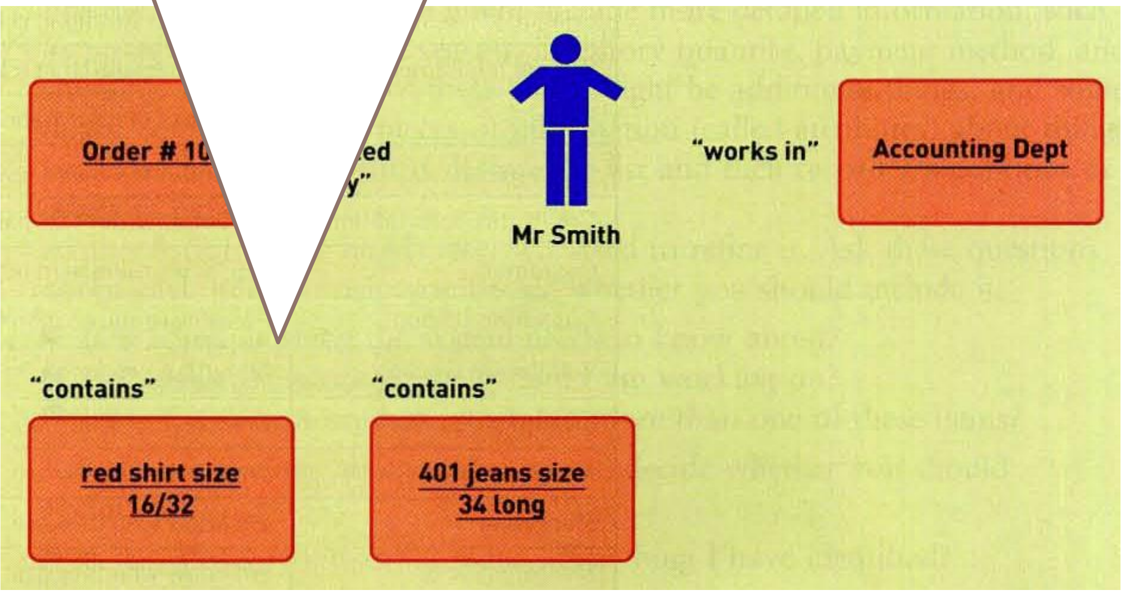
Smith work in accenting department



4.1.4 Association among things

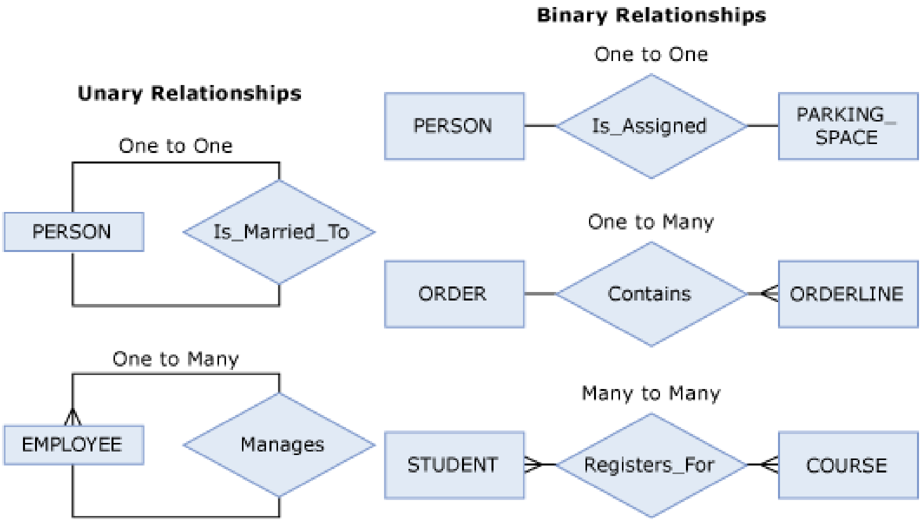
- Attribute is naturally occurring relationship between specific things, such as

An order 1043 contains red shirt size 16/32 and 401 jeans size 34 long.



4.1.4 Association among things

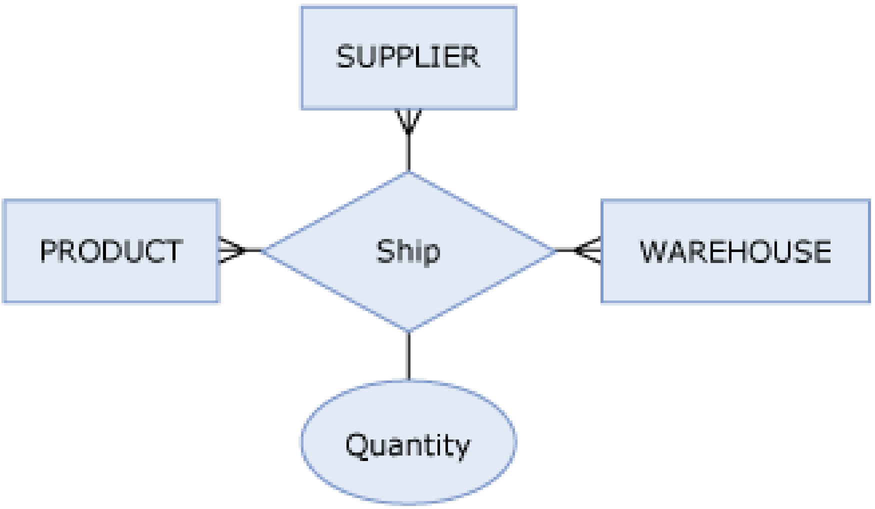
- Binary associations
  - the association is between two things of the same type
- Unary association
  - Called recursive association
  - The association is an organization hierarchy.



4.1.4 Association among things

- Ternary association
  - An association on three different type of thing
- N-ary association
  - Among any number of different type of thing

Ternary Relationships



4.2 ERD

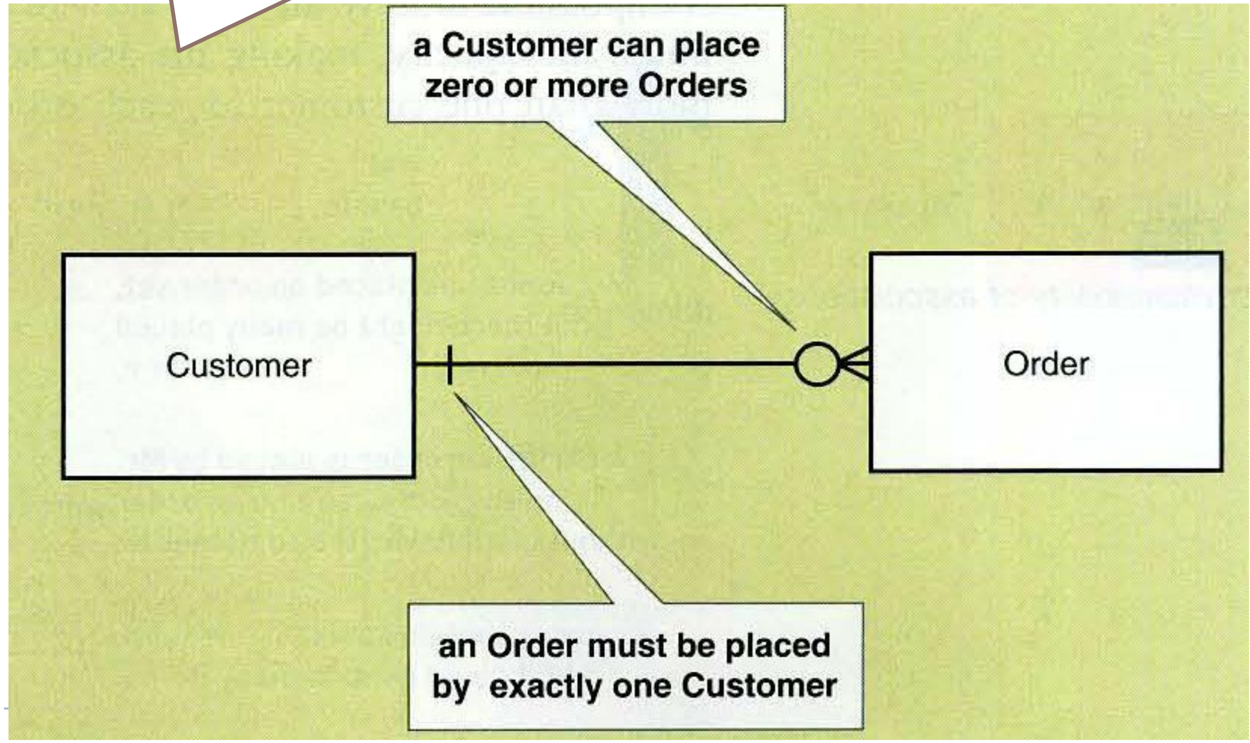
4.2 Entity-Relationship diagram

- ERD is a model commonly that used by traditional analysis and database analysis.
  - Data entities is sets of things or individual things in ER diagram.

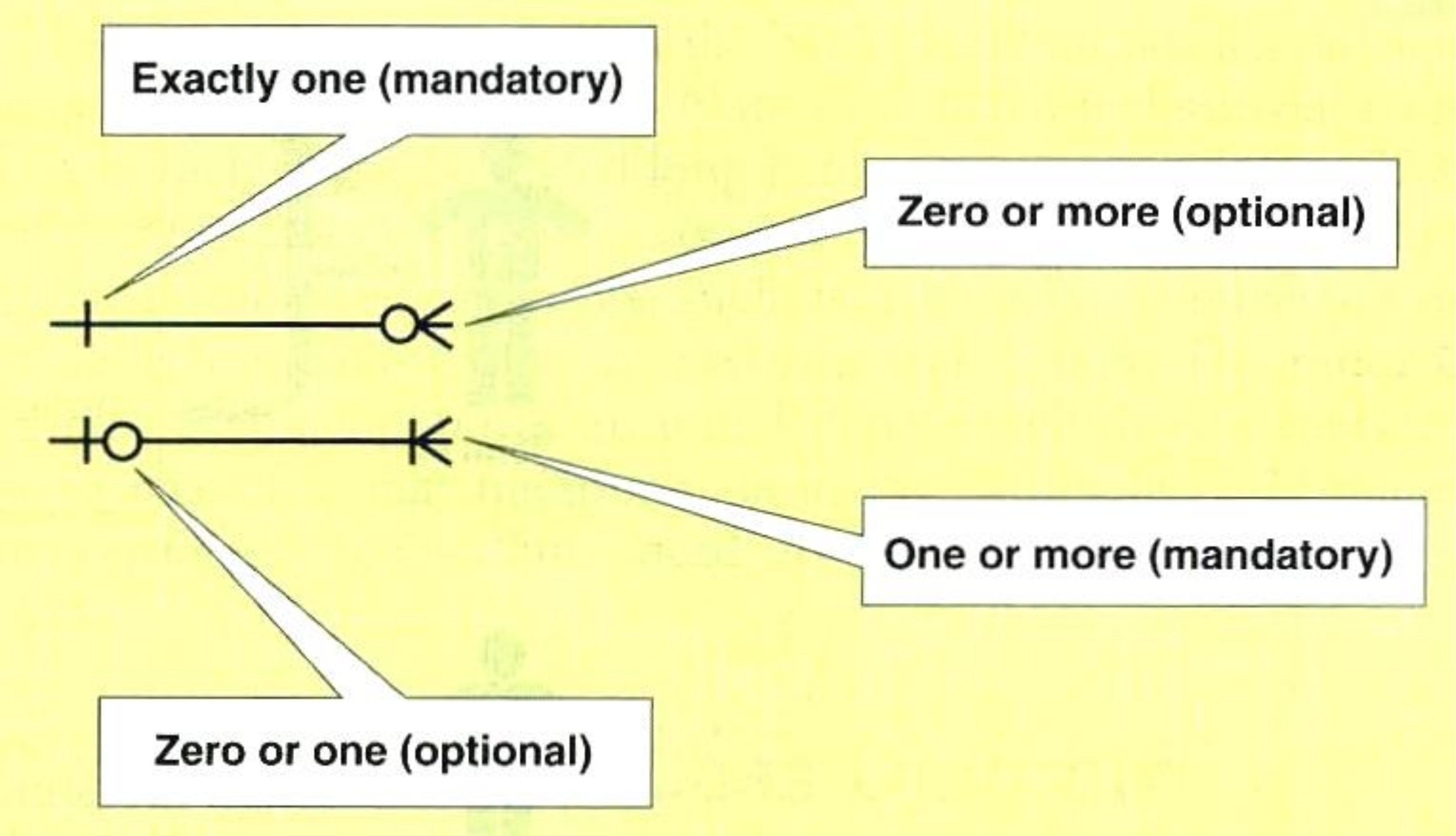
- ERD ≠ UML diagram

4.2.1 ERD Notation

Two data entities (Customer and Order)


a Customer can place zero or more Orders
Customer
Order
an Order must be placed by exactly one Customer

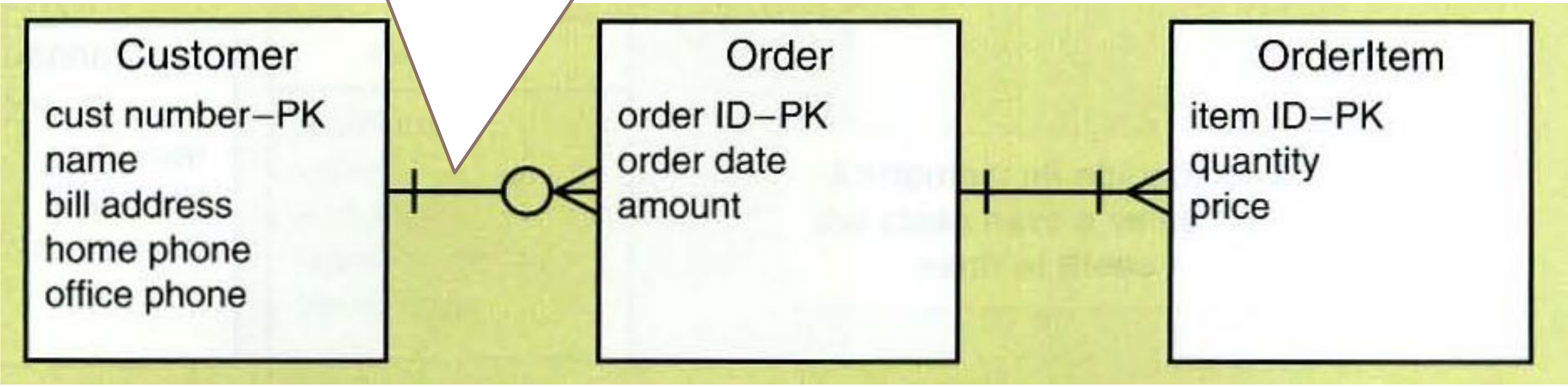
4.2.1 ERD Notation



4.2.1 ERD Notation: Example

- ERD with attributes shown

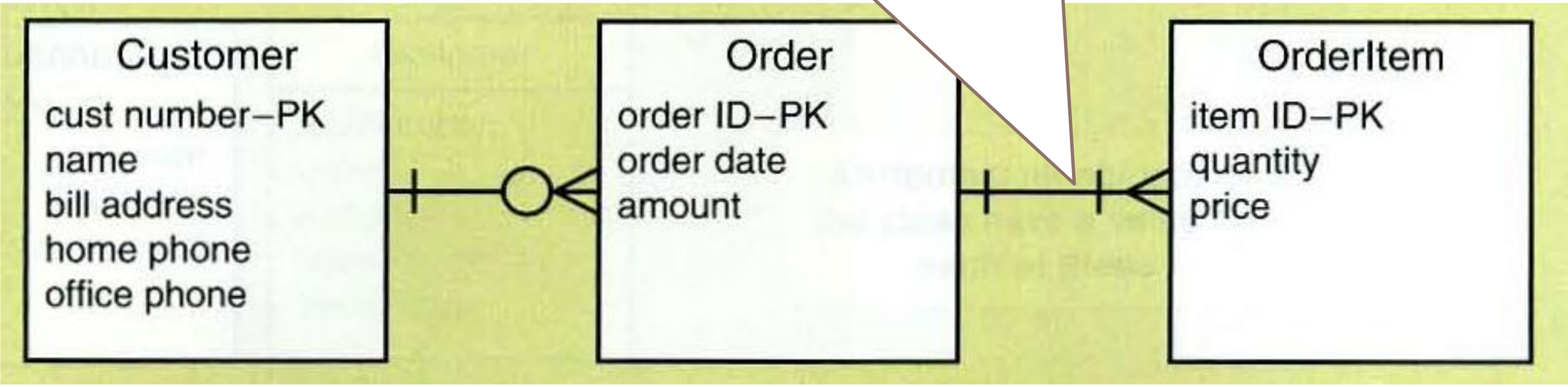
A customer takes many orders.
A customer don't take an order.



4.2.1 ERD Notation: Example

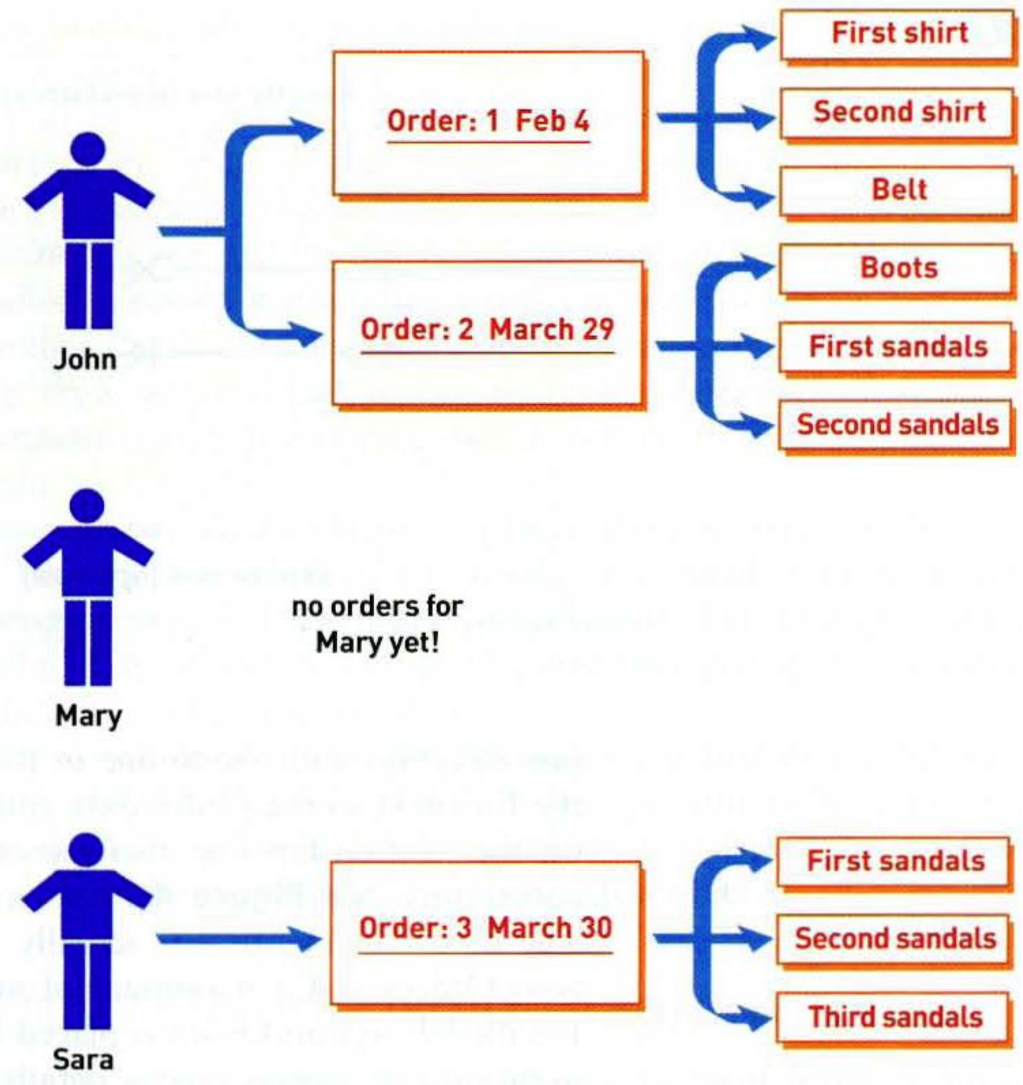
- ERD with attributes shown

An order has minimum one item.
An order has many items.



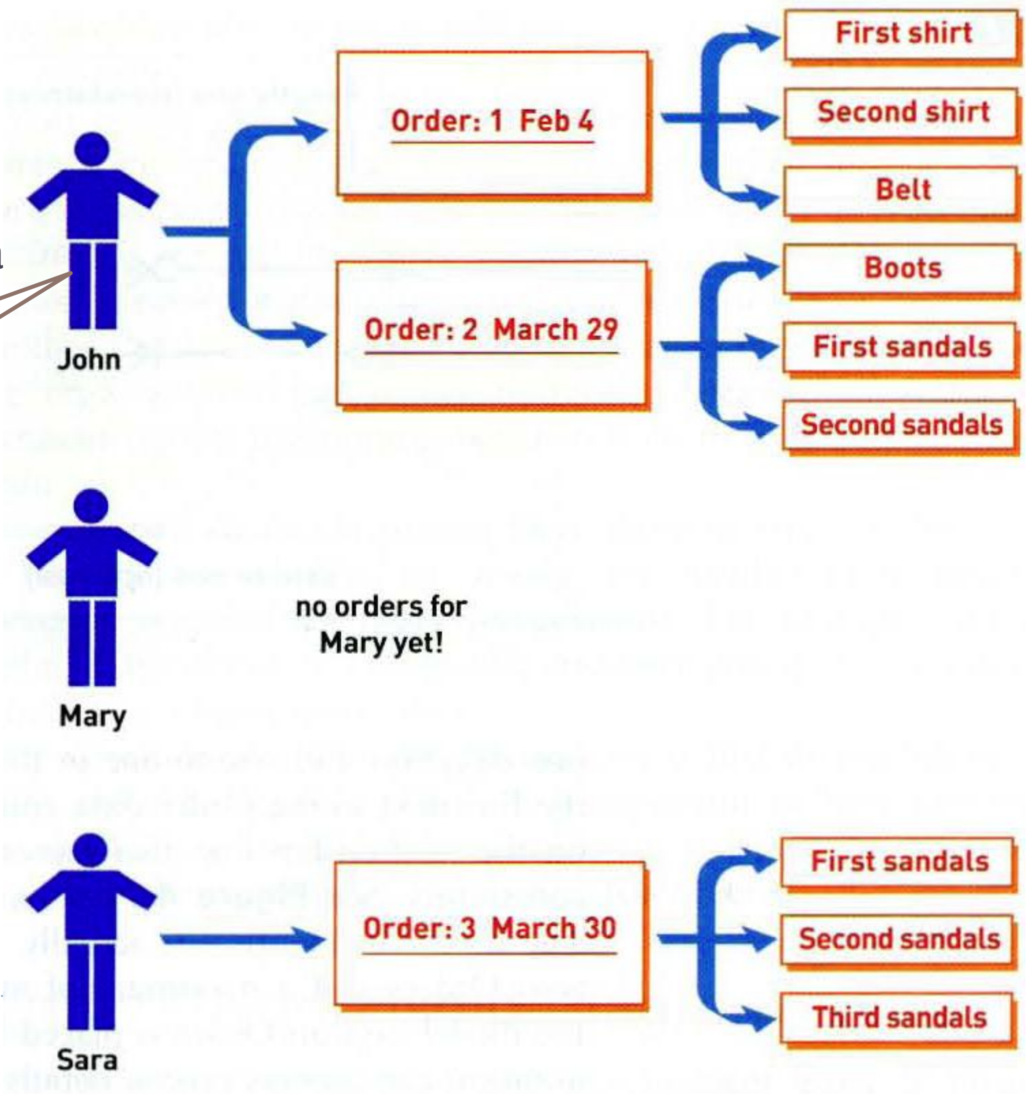
4.2.1 ERD Notation: Semantic Net

- Semantic Net
  - Graphical representation of an individual data entity and it relationship with other individua entity



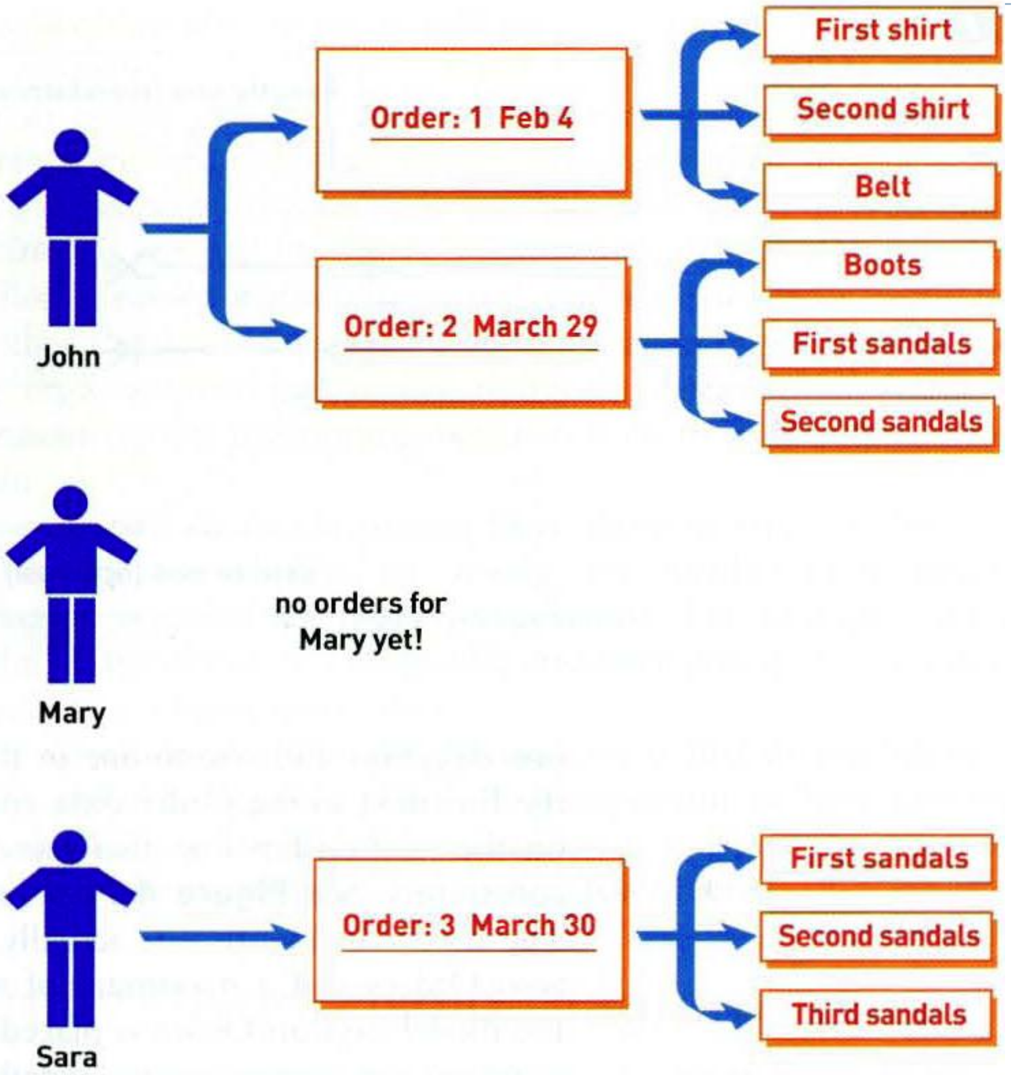
4.2.1 ERD Notation: Semantic Net

- Semantic Net
  - Graphical representation of an individual data entity and it relationship with other individua entity

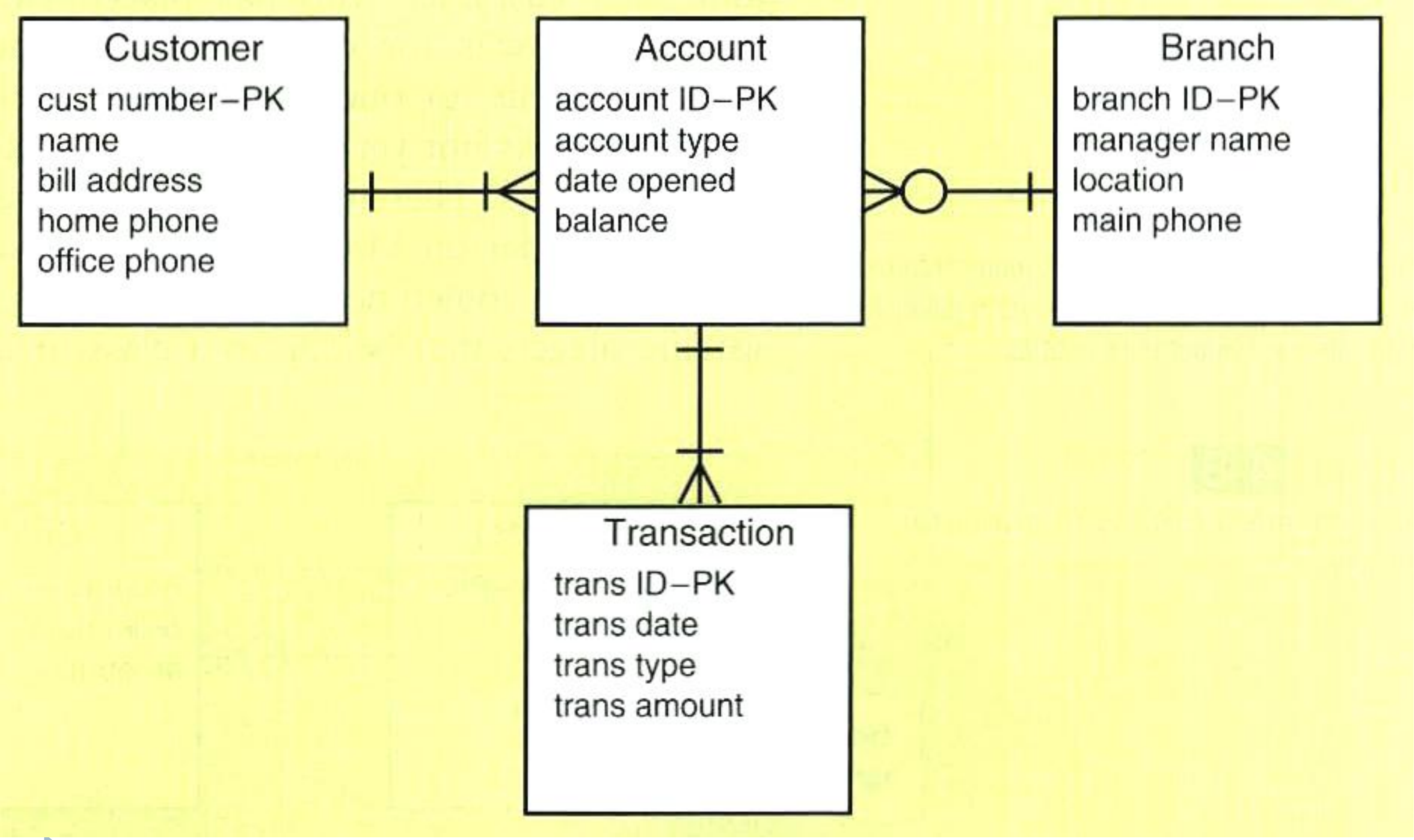


John placed the first order at Feb 4, was 2 shirts and a belt.

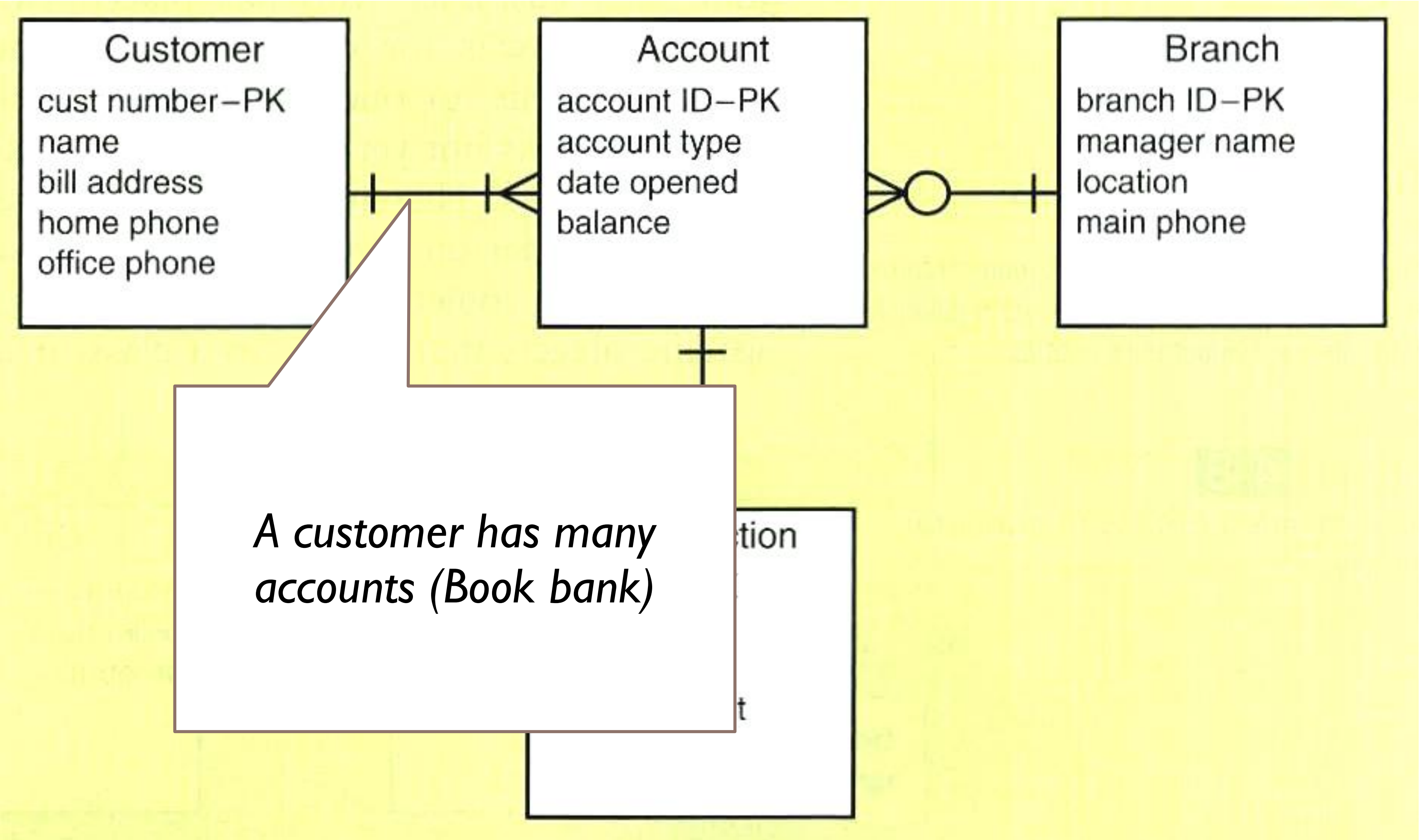
Question: Draw ERD notations (lines) from this problem



4.2.1 ERD Notation: Example many branch of bank

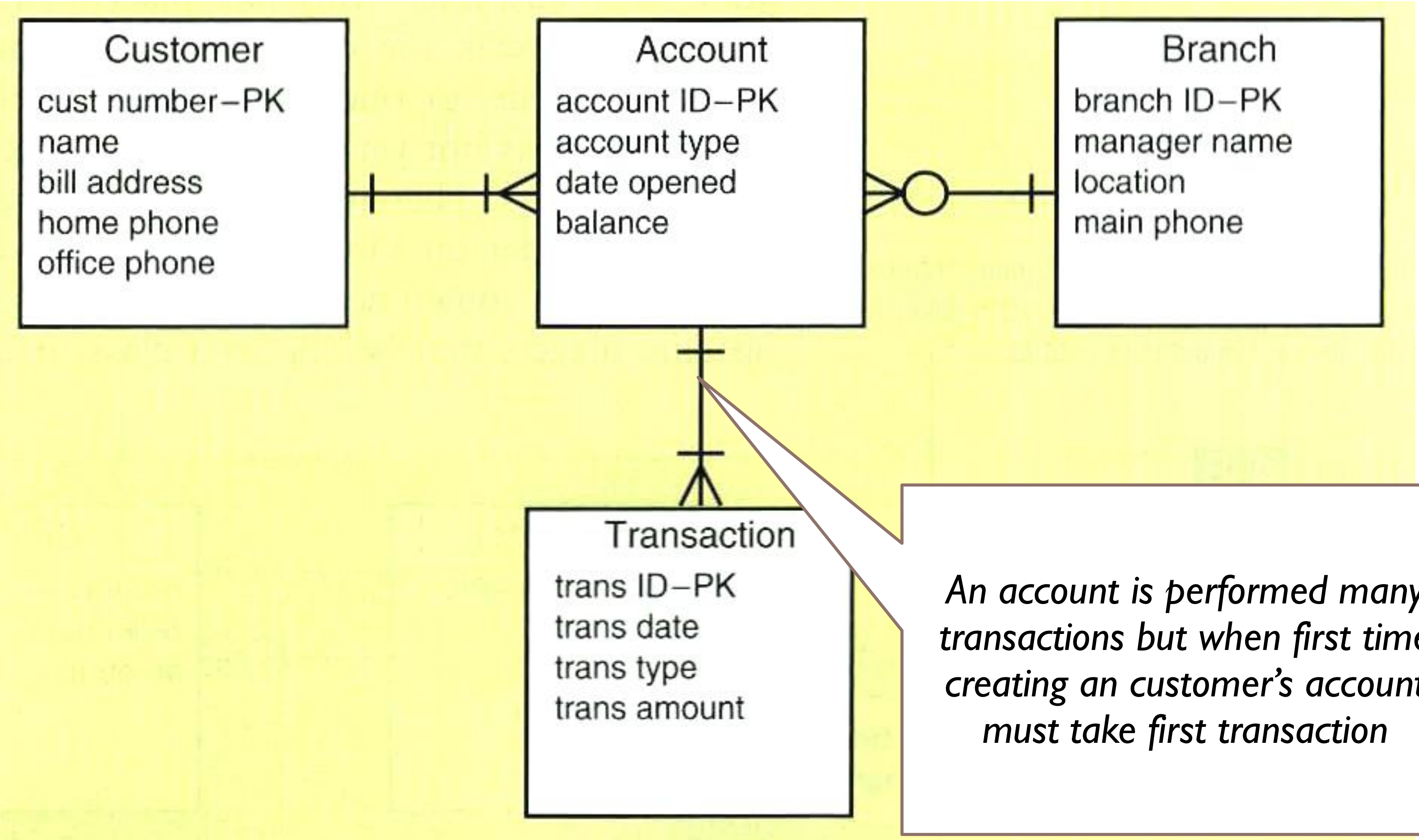


4.2.1 ERD Notation: Example many branch of bank



A customer has many accounts (Book bank)

4.2.1 ERD Notation: Example many branch of bank



Customer
cust number–PK
name
bill address
home phone
office phone

Account
account ID–PK
account type
date opened
balance

Branch
branch ID–PK
manager name
location
main phone

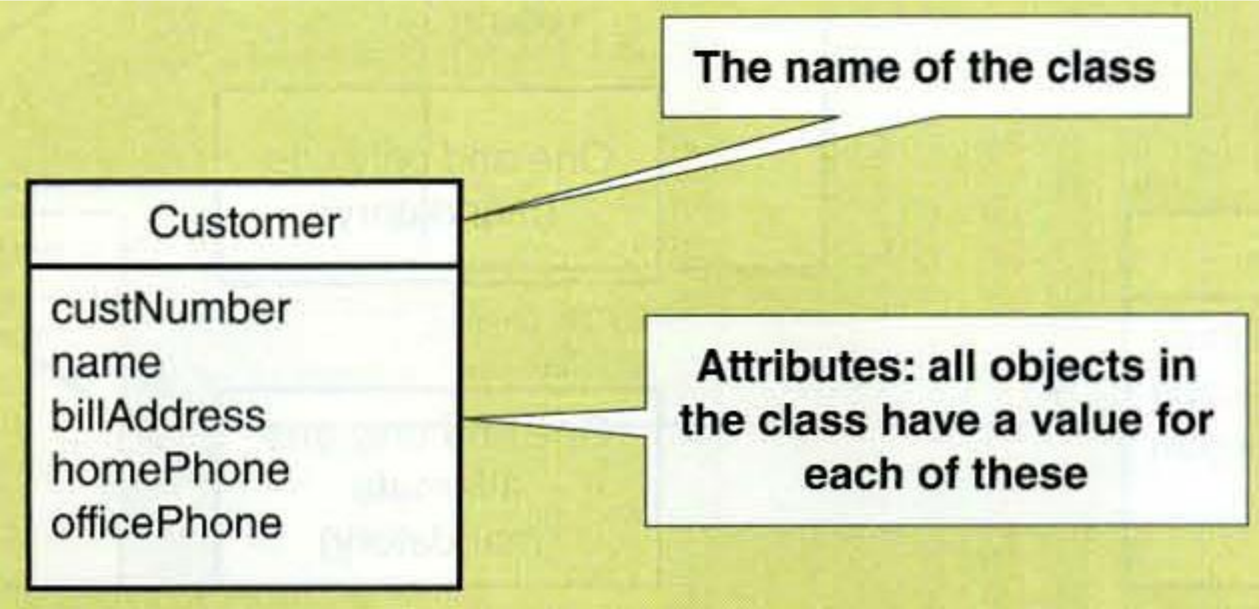
Transaction
trans ID–PK
trans date
trans type
trans amount

An account is performed many transactions but when first time creating an customer's account must take first transaction

4.3 Domain model class diagram

4.3 The Domain Model Class Diagram

- **Class** is category or classification used to describe a collection of object.
  - **Object** is member belongs to a class
- **Domain class** is the classes that describe thing in the problem domain.



The name of the class

Customer

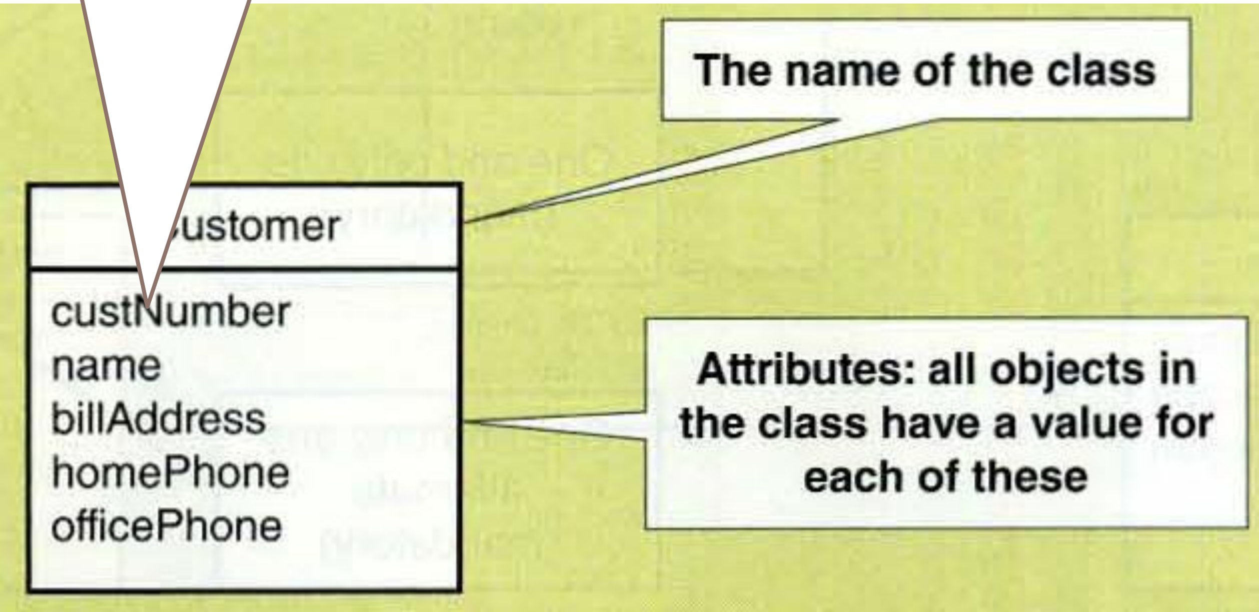
custNumber
name
billAddress
homePhone
officePhone

Attributes: all objects in the class have a value for each of these

4.3 The Domain Model Class Diagram

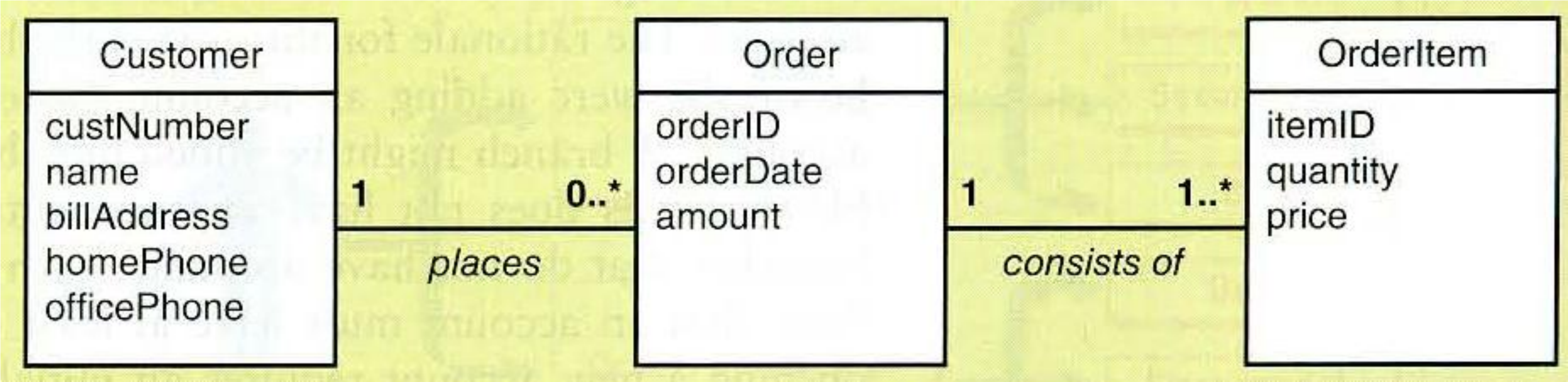
- **Clas** n used to describe a collection of o
  - **O** ass
- **Do** t describe thing in the problem dom

Camelback notation *or* ***camelCase notation*** *are concatenation words to a single word by the first character of each word typing capitalized.*



4.3 The Domain Model Class Diagram

- **Class diagram (UML)** is used to show class object for a system.
- **Domain model class diagram** is one type of UML class diagram that shows the things in the users 'problem domain.

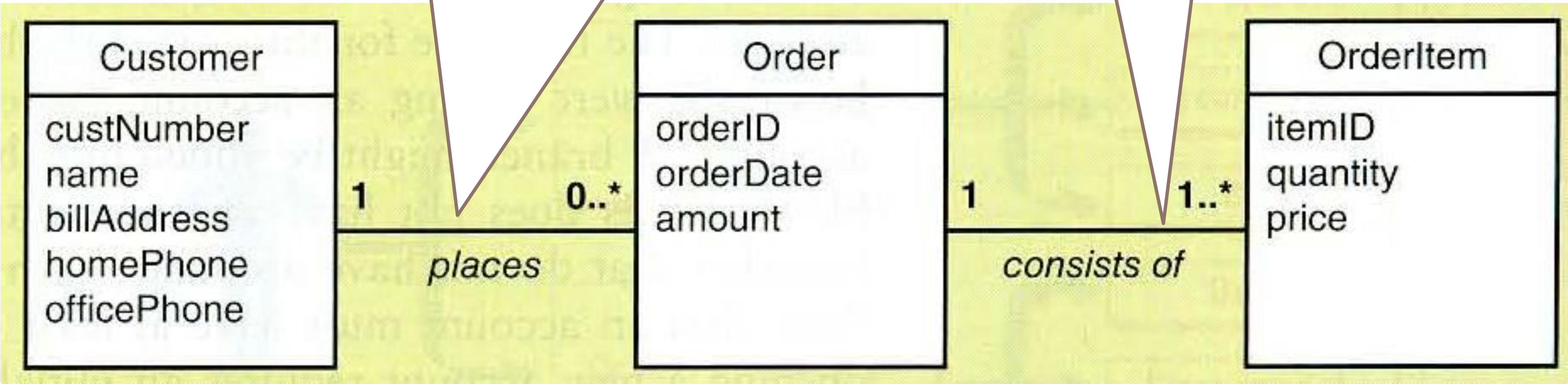


4.3 The Domain Model Class Diagram

- **Class diagram (UML** a system.
- **Domain model class diagram** is one ty ML class diagram that shows the things in the users m domain.

An order must consist 1 to N Items

A customer can place 0 to N orders



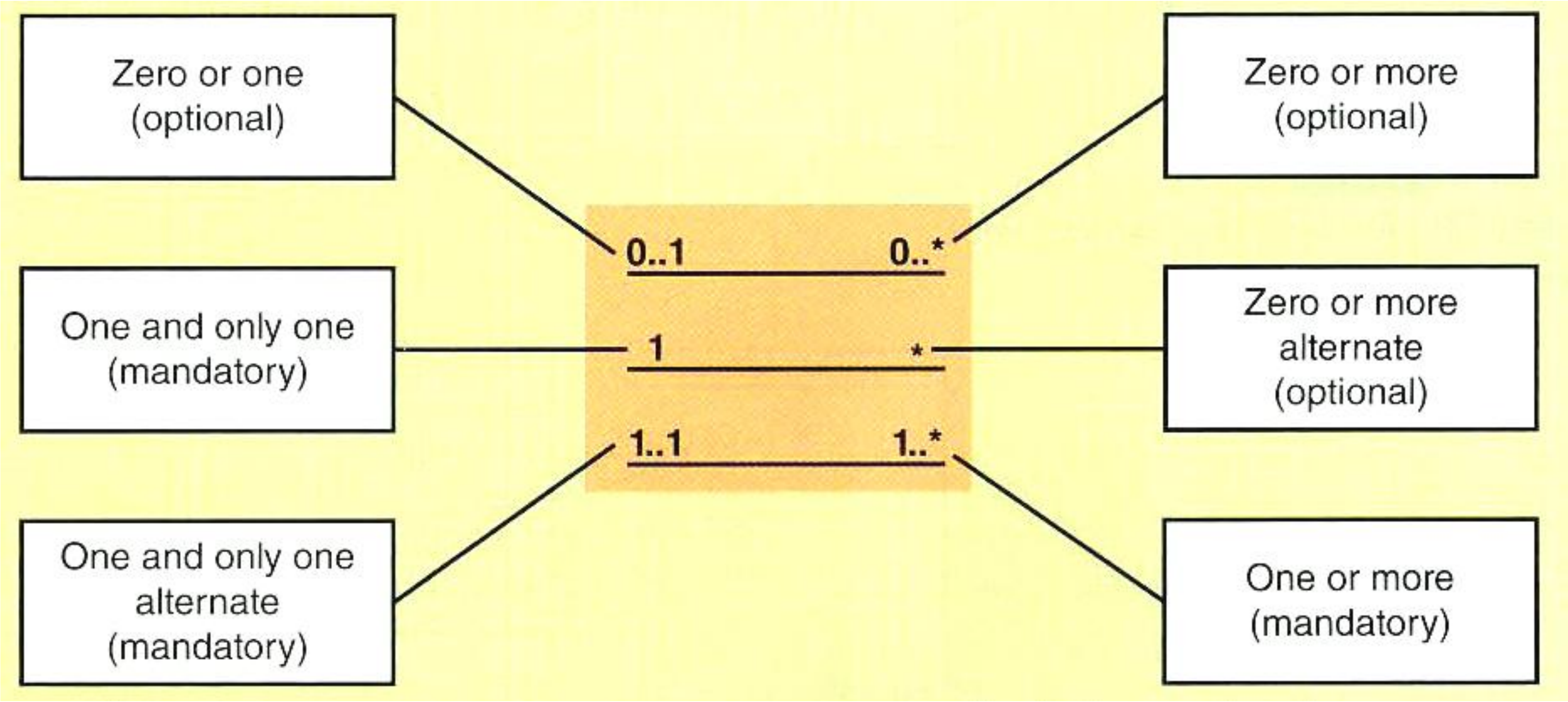
Association among things

- Multiplicity
  - 1 or 1..1 is exactly one customer
  - 0..1 is may or may not have a customer
  - * or 0..* is unlimited number of the customer
  - 2..4 is number of the customer being between 2 to 4

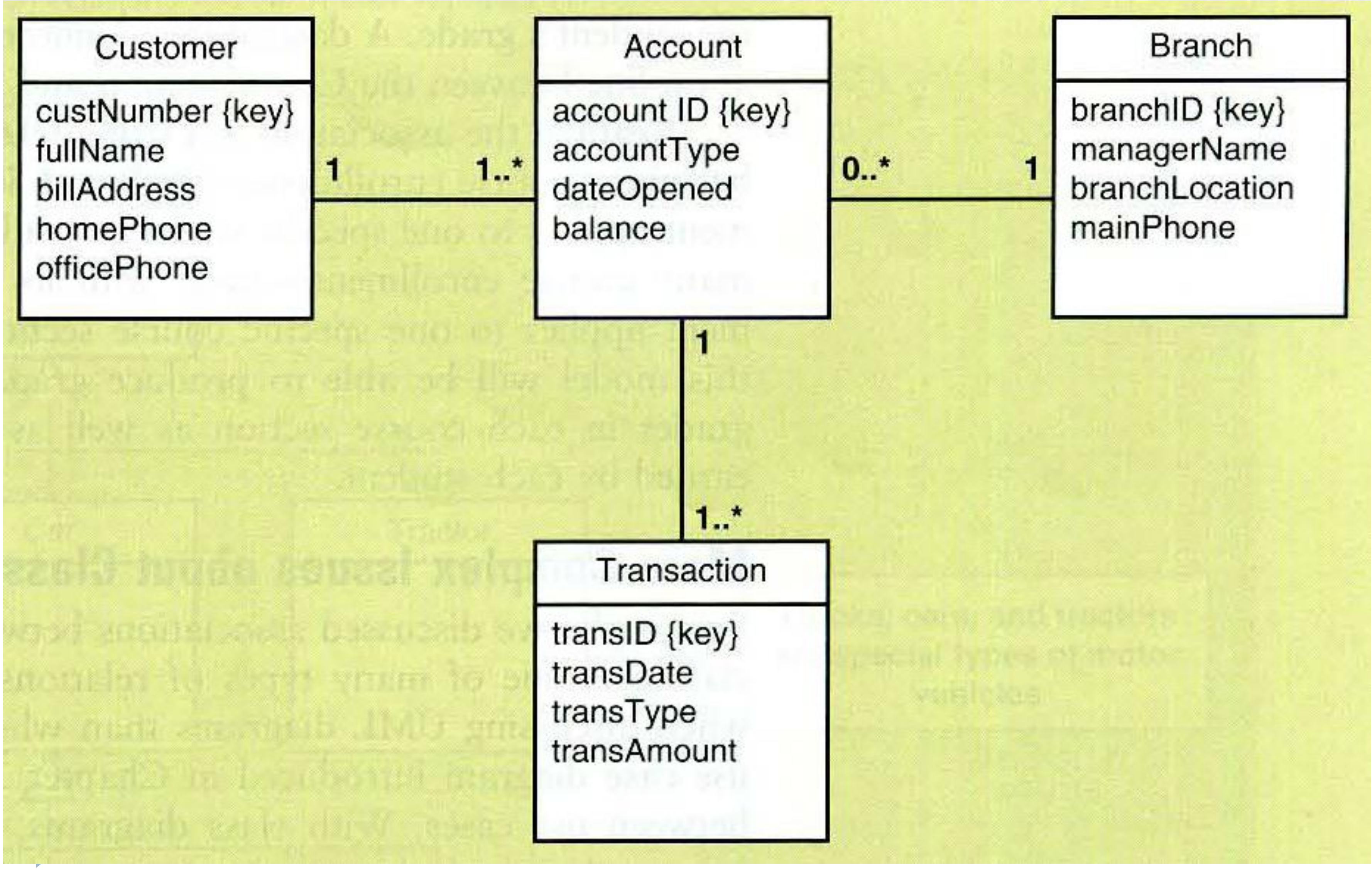
  - 0..1 or 0..2 or 0..* are called *optional*
  - 1 or 2..4 or 1..* or * are called *mandatory*

Multiplicity (n) ความหลากหลาย ความมากมาย
Mandatory (n) ผู้ได้รับมอบอำนาจ

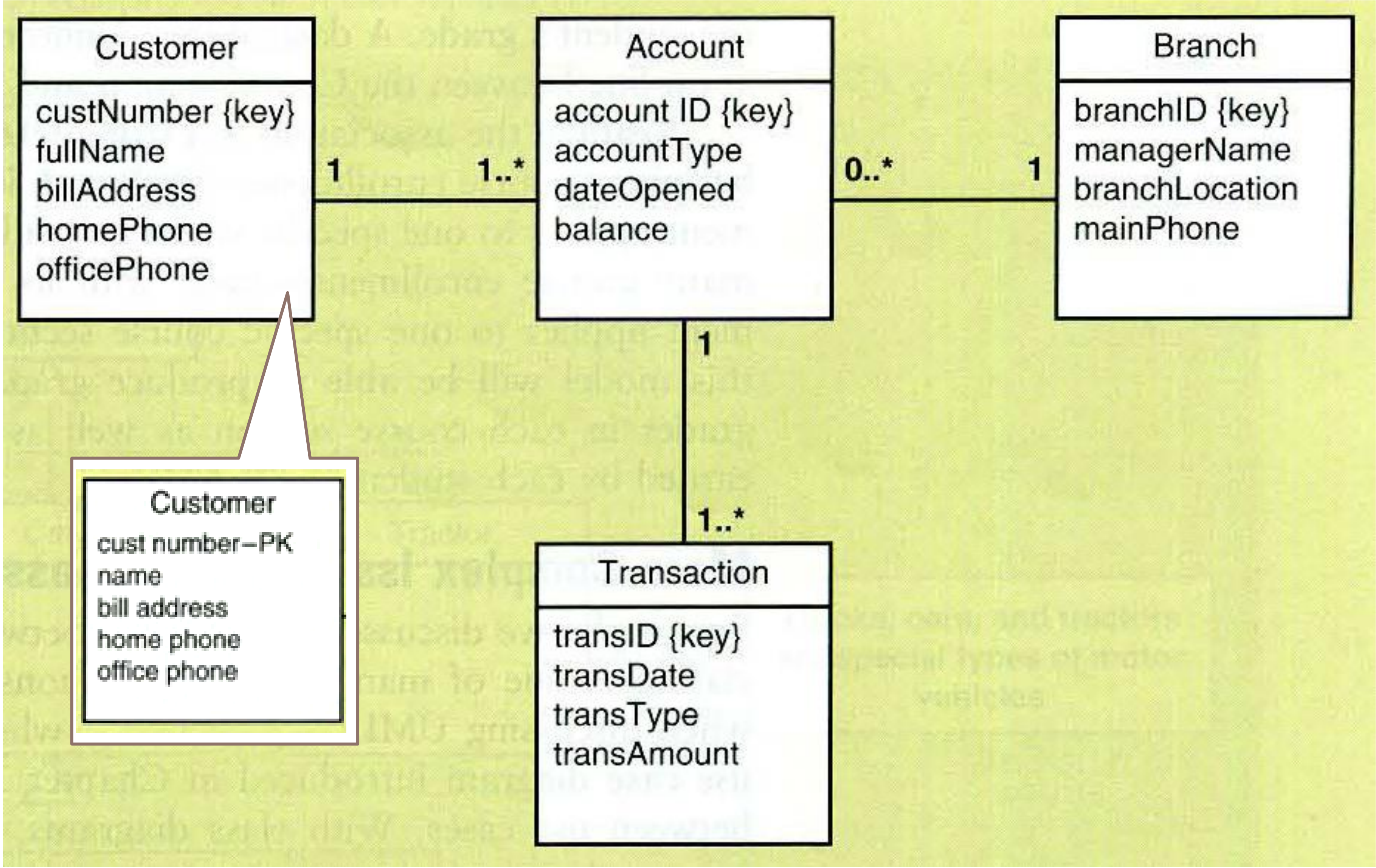
4.3.1 The Domain Model Class Diagram Notation


Zero or one (optional)
One and only one (mandatory)
One and only one alternate (mandatory)
0..1
1
1..1
0..*
*
1..*
Zero or more (optional)
Zero or more alternate (optional)
One or more (mandatory)

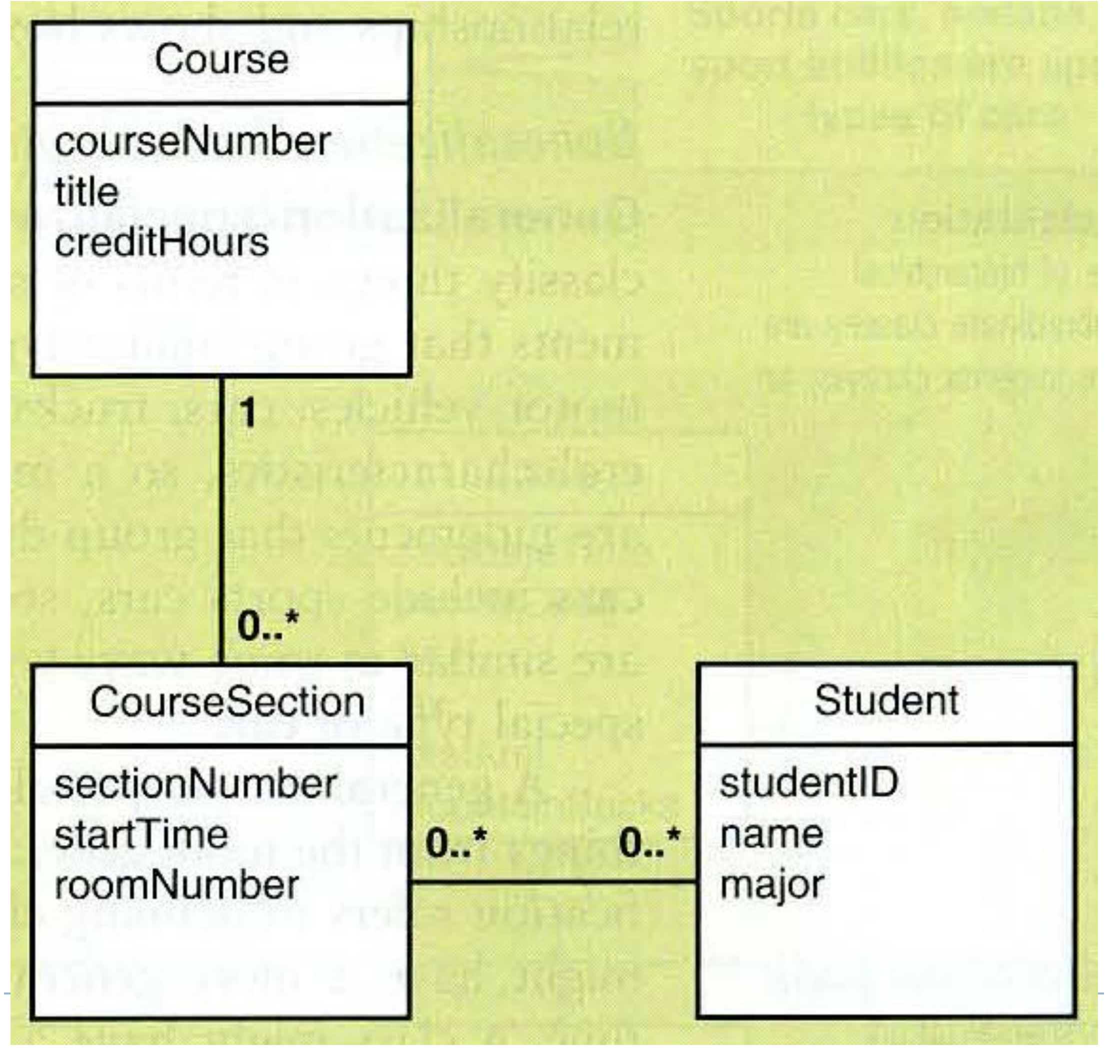
4.3.1 The Domain Model Class Diagram Example: A bank system



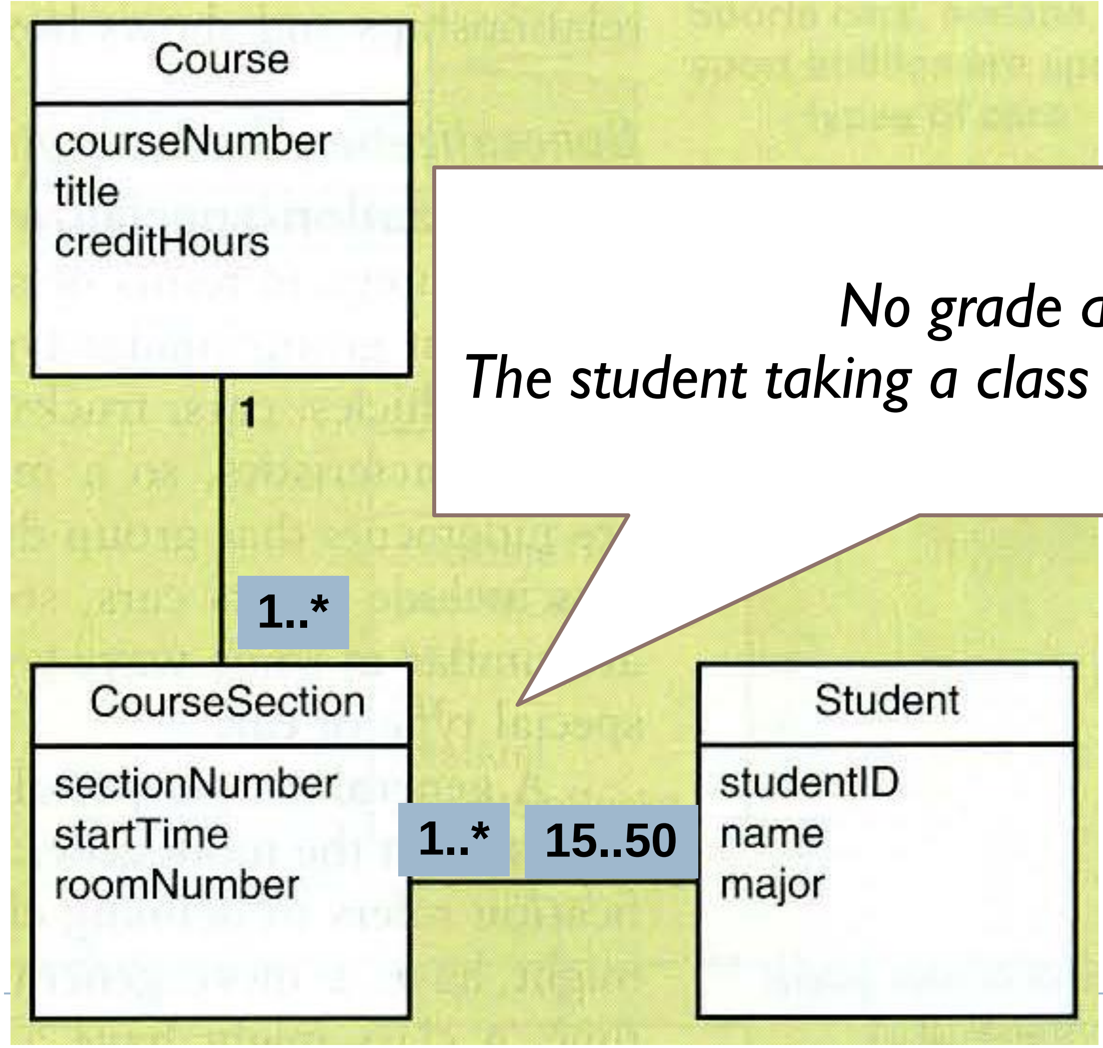
4.3.1 The Domain Model Class Diagram Example: A bank system



4.3.1 The Domain Model Class Diagram Example: A university course enrollment



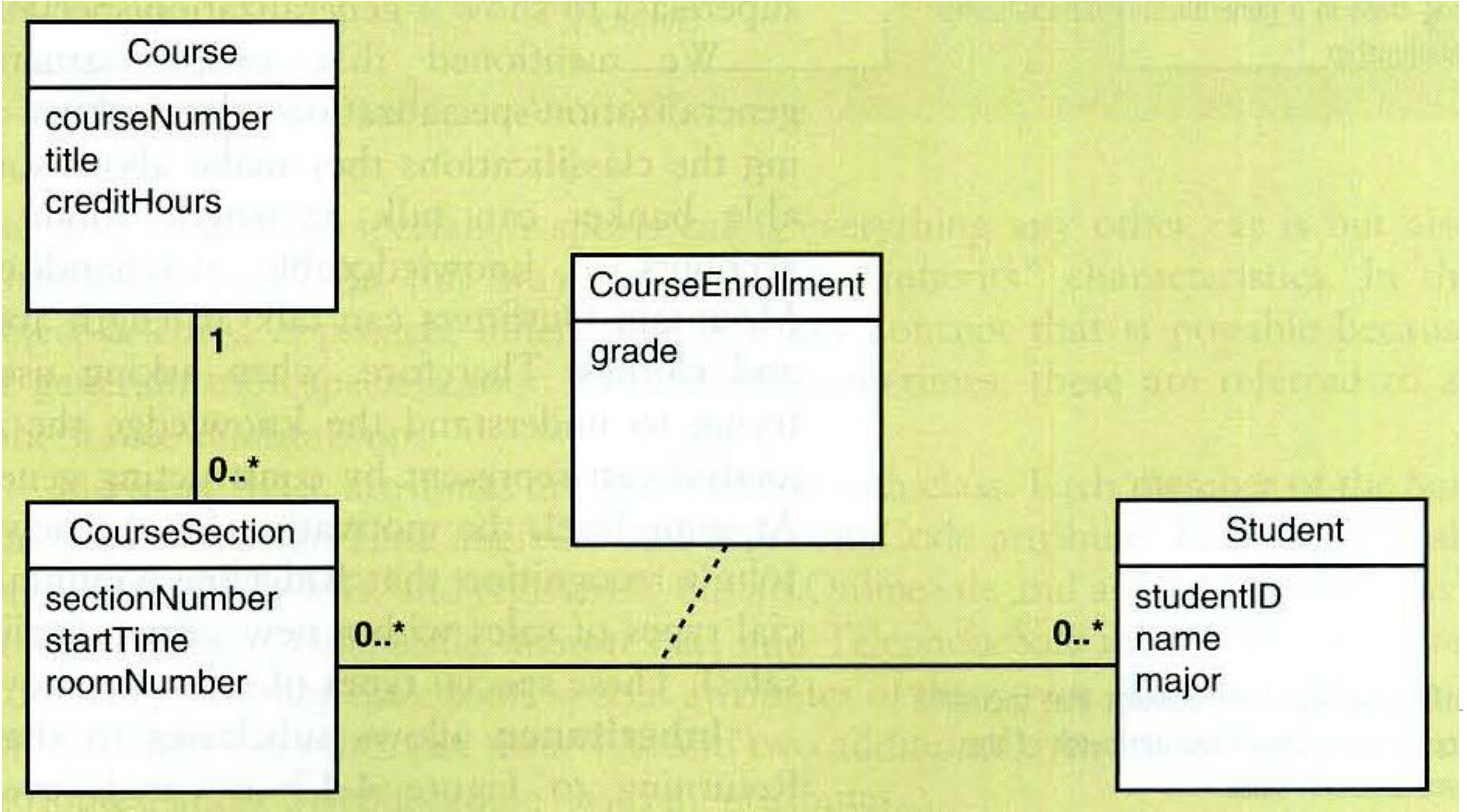
4.3.1 The Domain Model Class Diagram Example: A university course enrollment



No grade data,
The student taking a class don’t have a grade.

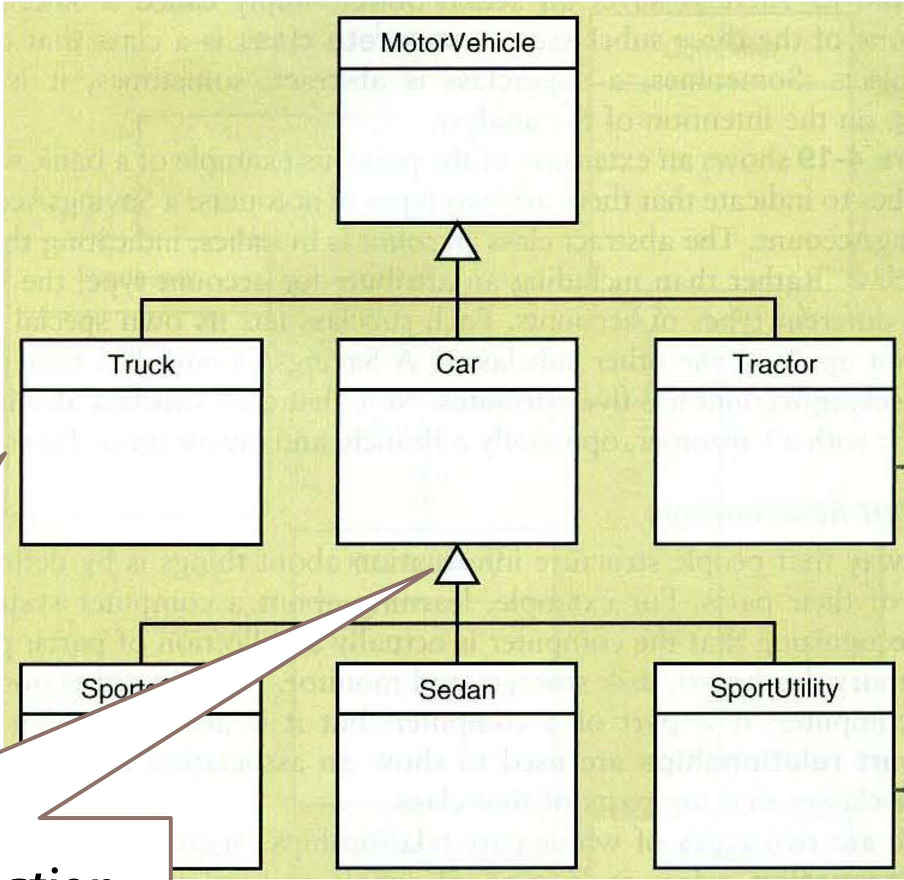
4.3.1 The Domain Model Class Diagram Example: A university course enrollment (2)

- Association class
  - The solution is to add a domain class represent association between two classes.



4.3.2 More complex issue about classes of objects

- Generalization relationship
  - Group similar types of things
- Specialization relationship
  - Group different types of things



Generalize,
Truck, car and tractor are the motor vehicle.

Generalization/ Socialization notation

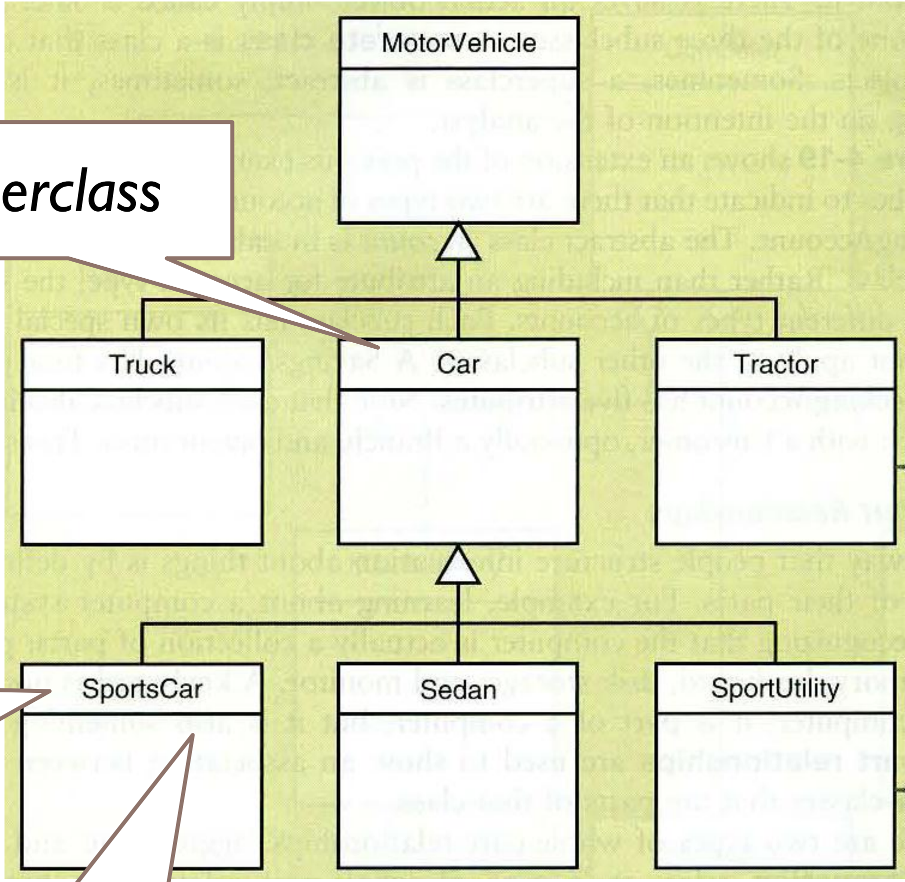
4.3.2 More complex issue about classes of objects

- Generalization relationship
  - Group similar types of things

- Specialization relationship
  - Group different types of things

Specialize,
Sport car is different from tractor and truck, is car.

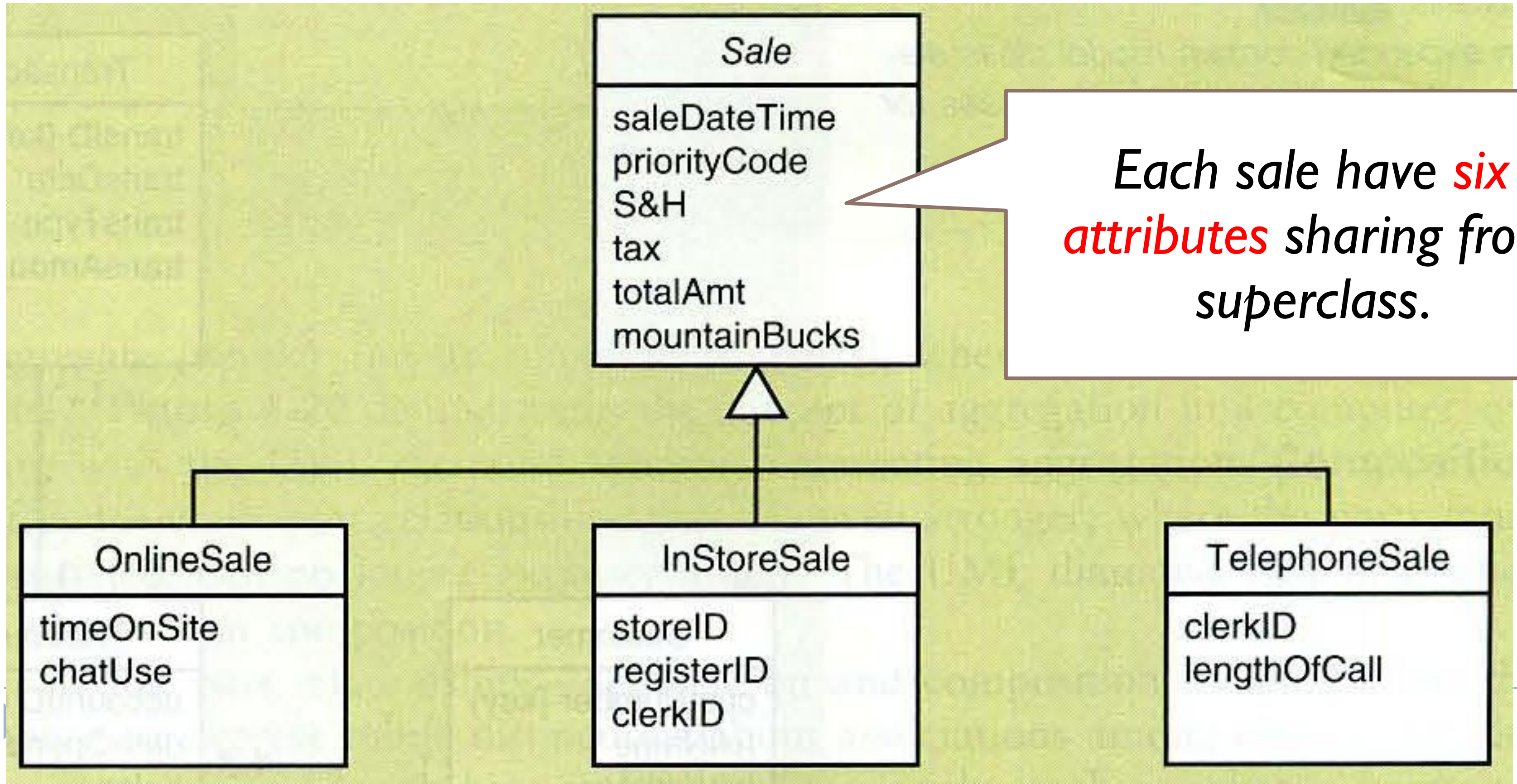
Superclass



Subclass

4.3.2 More complex issue about classes of objects

- Inheritance
  - Shares some characteristic (attribute) from superclass to subclass.

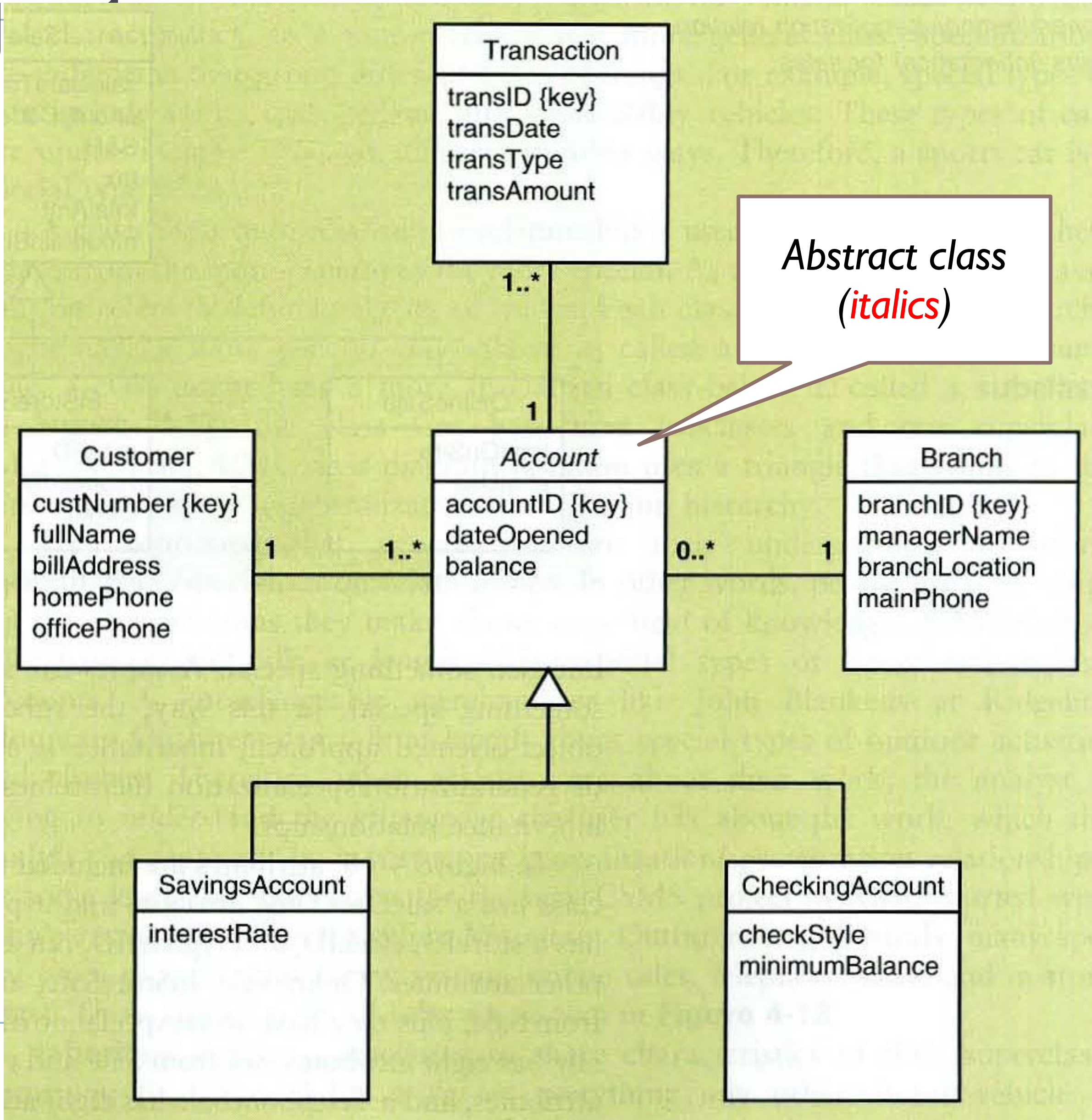


Each sale have six attributes sharing from superclass.

4.3.2 More complex issue about classes of obj

- **Abstract class** is a class that subclass can inherit from it.

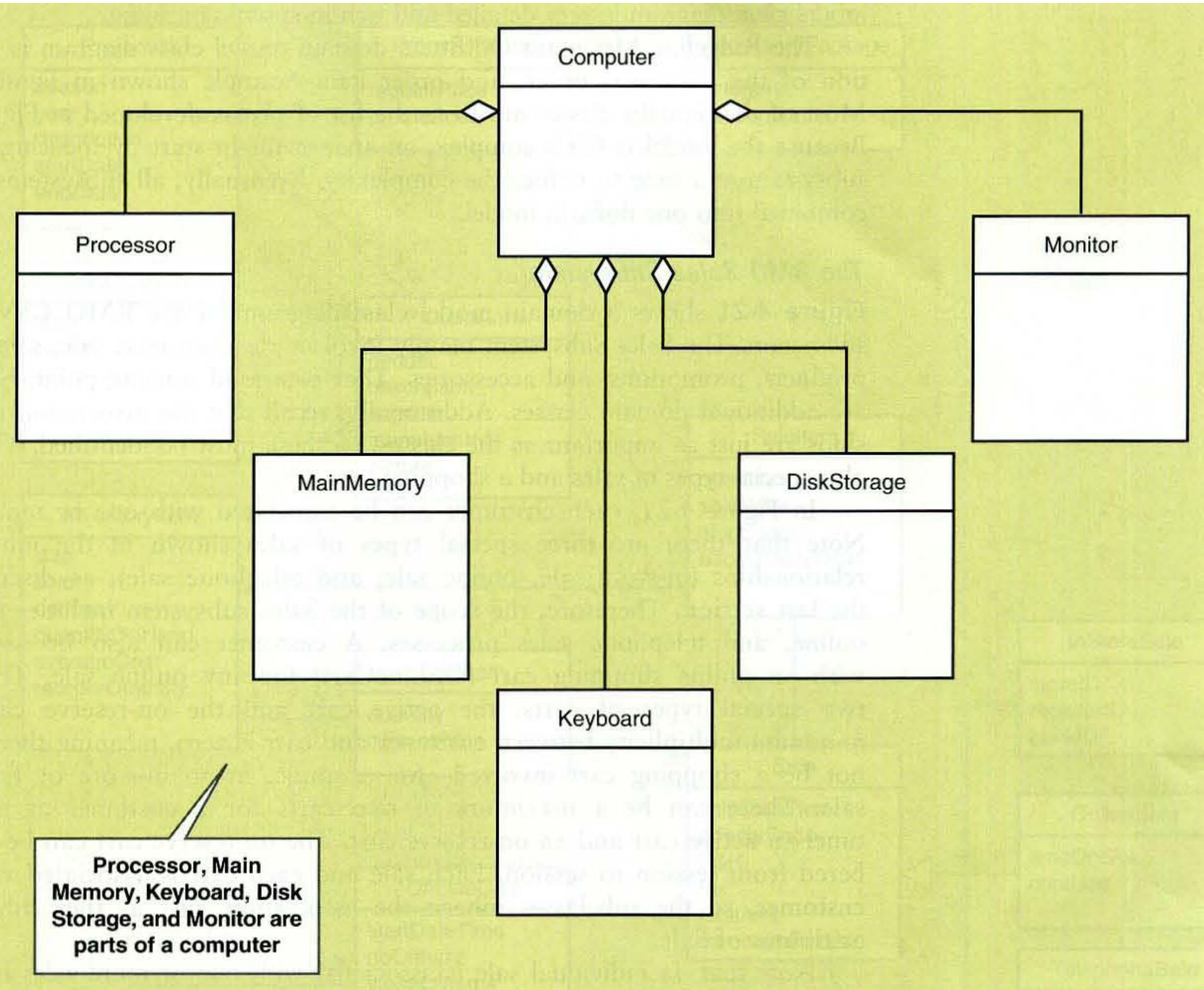
- **Concrete class** is a class that does have actual object.



Abstract class (*italics*)

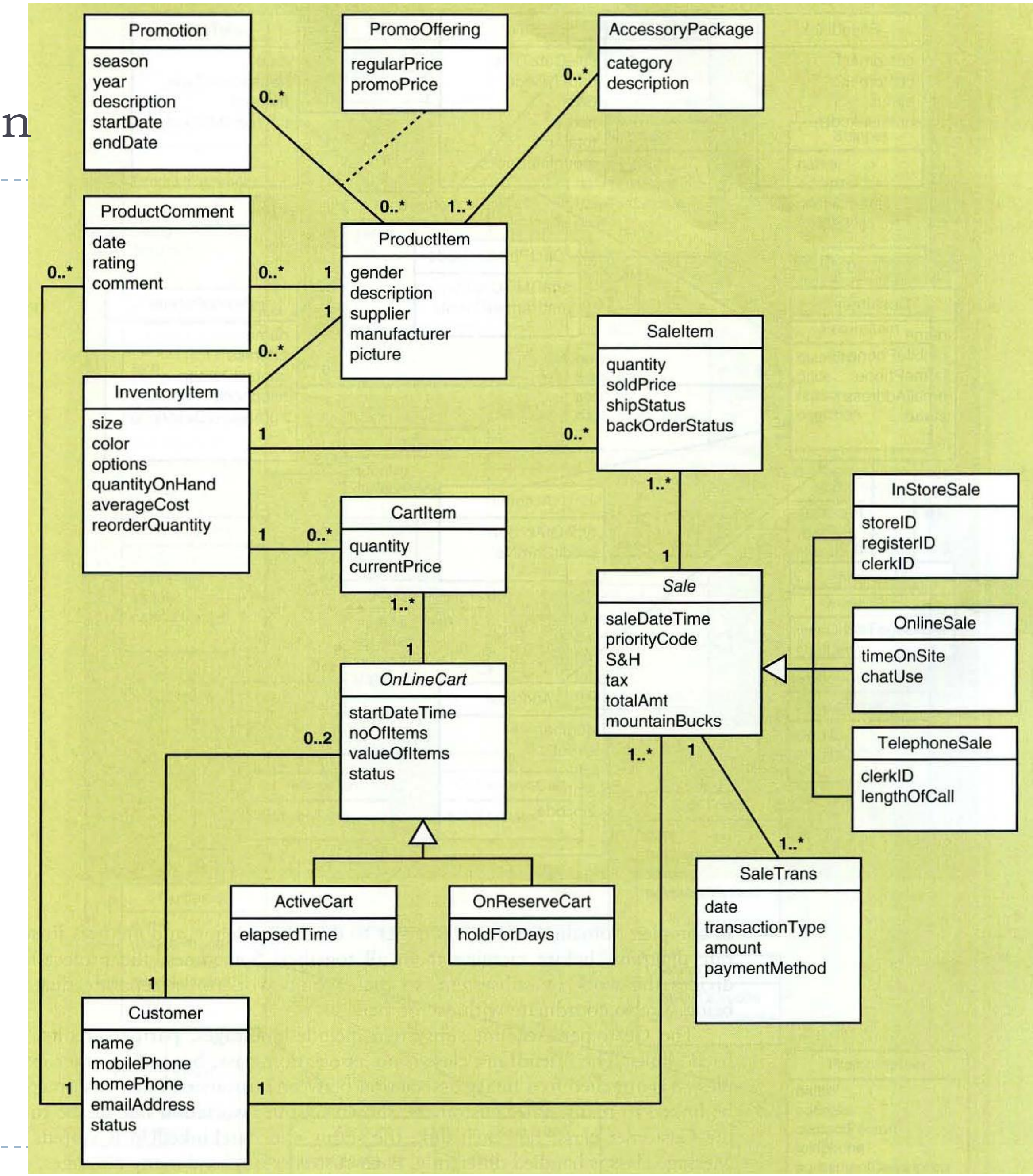
4.3.2 More complex issue about classes of objects

- Whole part relationship
  - Aggregation
  - Composition

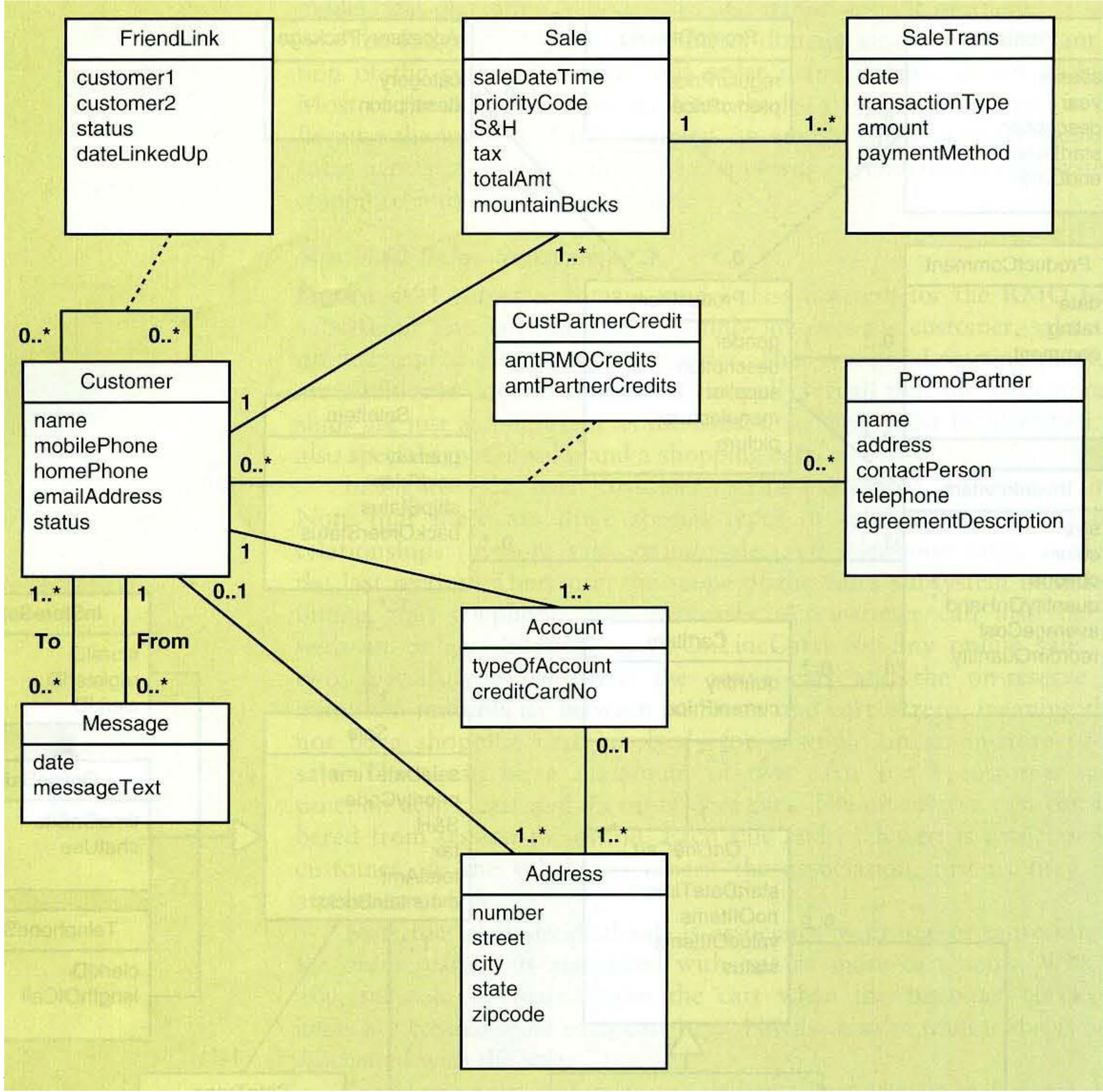


Processor, Main Memory, Keyboard, Disk Storage, and Monitor are parts of a computer

4.3.3 RMO Example: Domain Model Class Diagram (Sales subsystem)



4.3.3 RMO Example: Domain Model Class Diagram (Customer account subsystem)



FriendLink
customer1
customer2
status
dateLinkedUp

Sale
saleDateTime
priorityCode
S&H
tax
totalAmt
mountainBucks

SaleTrans
date
transactionType
amount
paymentMethod

Customer
name
mobilePhone
homePhone
emailAddress
status

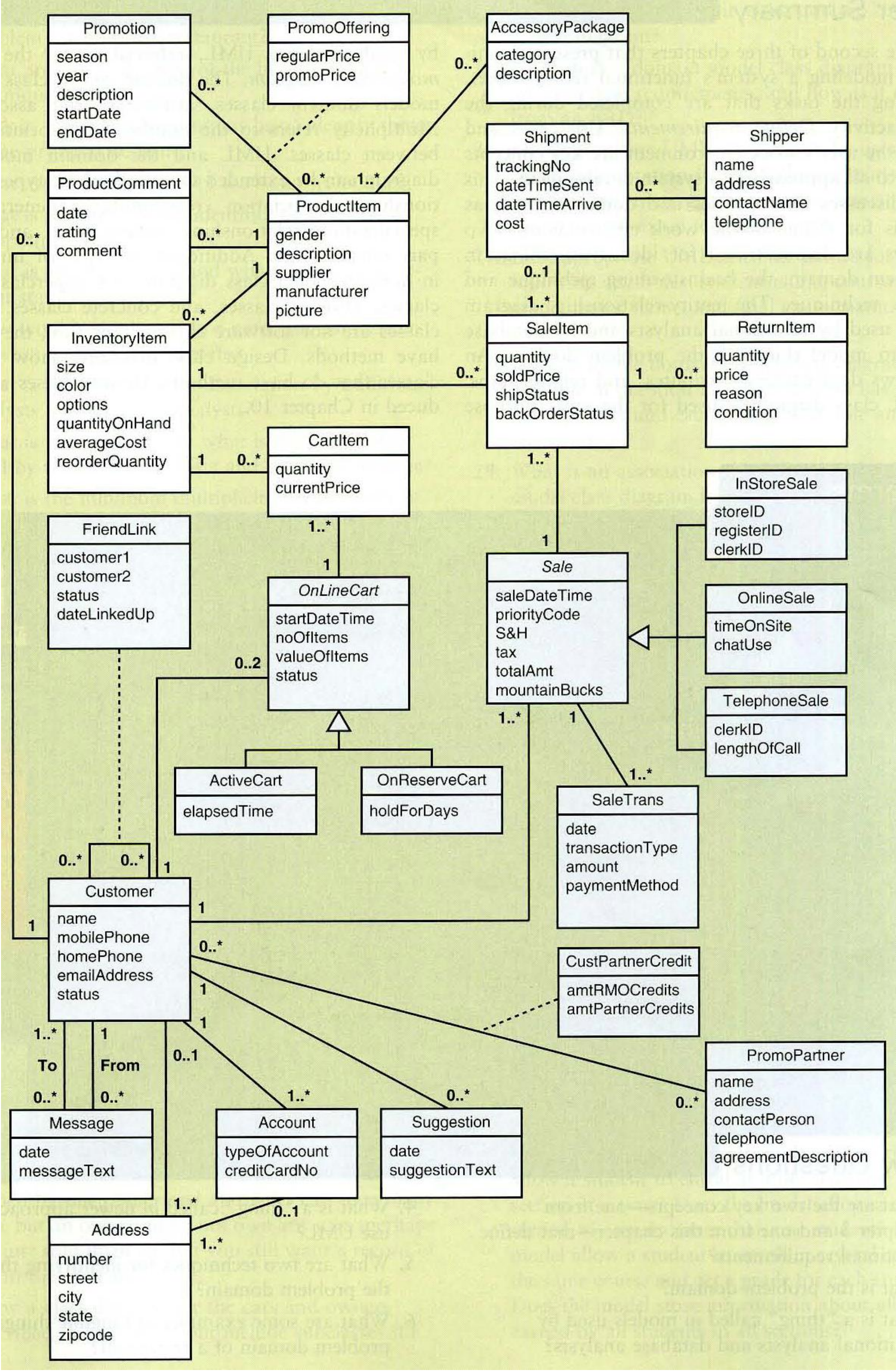
CustPartnerCredit
amtRMOCredits
amtPartnerCredits

PromoPartner
name
address
contactPerson
telephone
agreementDescription

Account
typeOfAccount
creditCardNo

Message
date
messageText

Address
number
street
city
state
zipcode

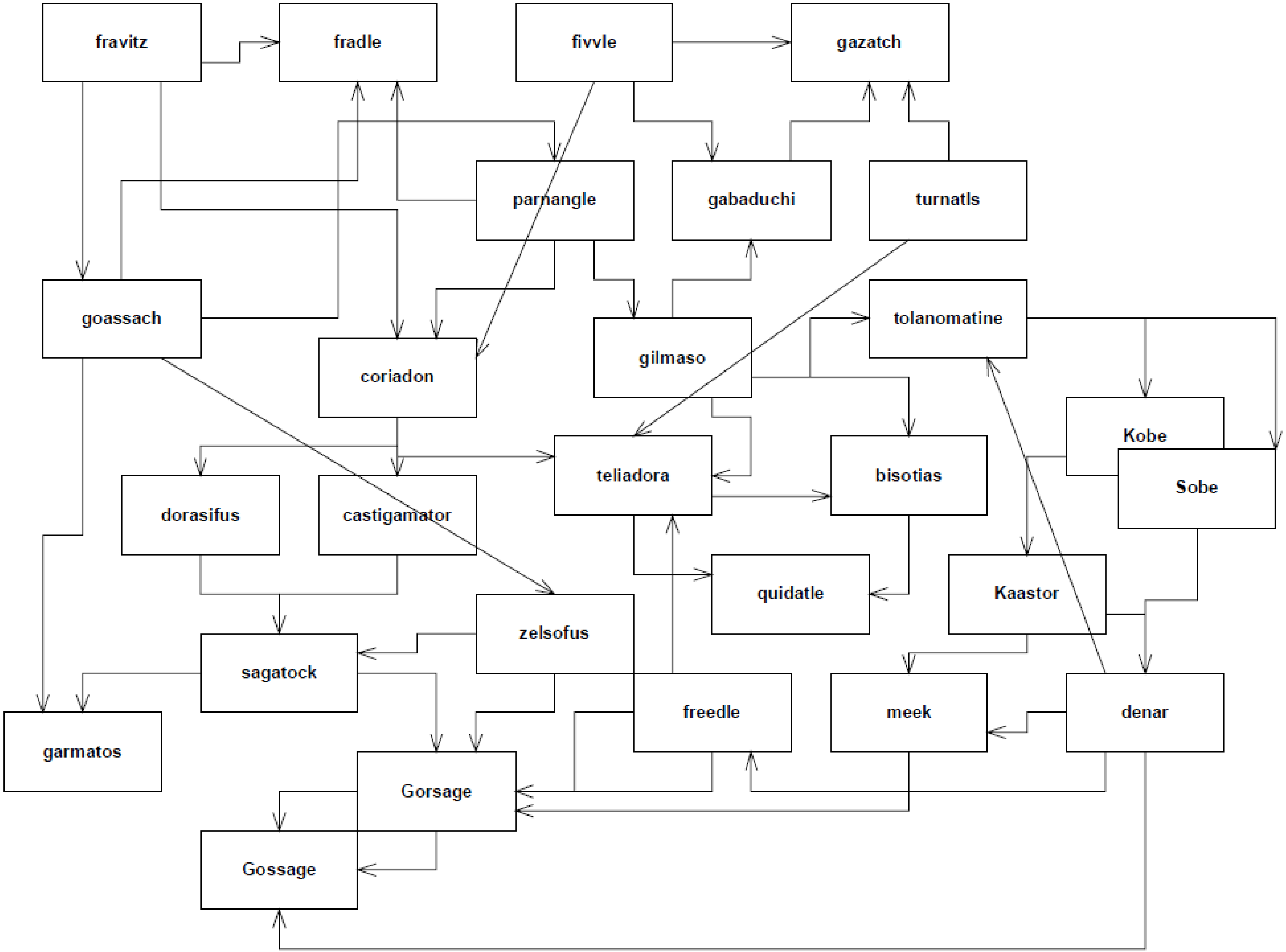

Promotion
season
year
description
startDate
endDate
PromoOffering
regularPrice
promoPrice
AccessoryPackage
category
description
ProductComment
date
rating
comment
ProductItem
gender
description
supplier
manufacturer
picture
Shipment
trackingNo
dateTimeSent
dateTimeArrive
cost
Shipper
name
address
contactName
telephone
InventoryItem
size
color
options
quantityOnHand
averageCost
reorderQuantity
SaleItem
quantity
soldPrice
shipStatus
backOrderStatus
ReturnItem
quantity
price
reason
condition
CartItem
quantity
currentPrice
InStoreSale
storeID
registerID
clerkID
FriendLink
customer1
customer2
status
dateLinkedUp
OnLineCart
startDateTime
noOfItems
valueOfItems
status
Sale
saleDateTime
priorityCode
S&H
tax
totalAmt
mountainBucks
OnlineSale
timeOnSite
chatUse
TelephoneSale
clerkID
lengthOfCall
ActiveCart
elapsedTime
OnReserveCart
holdForDays
SaleTrans
date
transactionType
amount
paymentMethod
Customer
name
mobilePhone
homePhone
emailAddress
status
CustPartnerCredit
amtRMOCredits
amtPartnerCredits
PromoPartner
name
address
contactPerson
telephone
agreementDescription
To
From
Message
date
messageText
Account
typeOfAccount
creditCardNo
Suggestion
date
suggestionText
Address
number
street
city
state
zipcode
0..*
1..*
0..1
0..2
1

Summary

- Two techniques are demonstrated for identifying things in the problem domain
  - Brainstorming technique
  - The noun technique
- Entity-relationship diagram
  - Data entities
  - Attribute
  - Relationship
- UML class diagram
  - Domain model class diagram
  - Attributes
  - Associations

- Generalization/specialization relationship
- Whole part relationship
- Superclass/ Subclass
- Abstract class
- Concrete class

Don’t do this


fravitz
fradle
fivvle
gazatch
parnangle
gabaduchi
turnatls
goassach
coriadon
gilmaso
tolanomatine
Kobe
Sobe
dorasifus
castigamator
teliadora
bisotias
quidatle
Kaastor
zelsofus
sagatock
freedle
meek
denar
garmatos
Gorsage
Gossage