DOMAIN CLASSES

Lecture 6

1.11.2014, Saturday

Outline of This Lecture

- “Things” in the Problem Domain
  - Data entities
  - Domain classes
- The Domain Model Class Diagram
- The Entity-Relationship Diagram

Things in the Problem Domain

- Problem domain—the specific area (or domain) of the users' business need that is within the scope of the new system.
- "Things" are those items users work with when accomplishing tasks that need to be remembered
- Examples of "Things" are products, sales, shippers, customers, invoices, payments, etc.
- These "Things" are modeled as domain classes or data entities
- In this course, we will call them domain classes. In database class you call them data entities

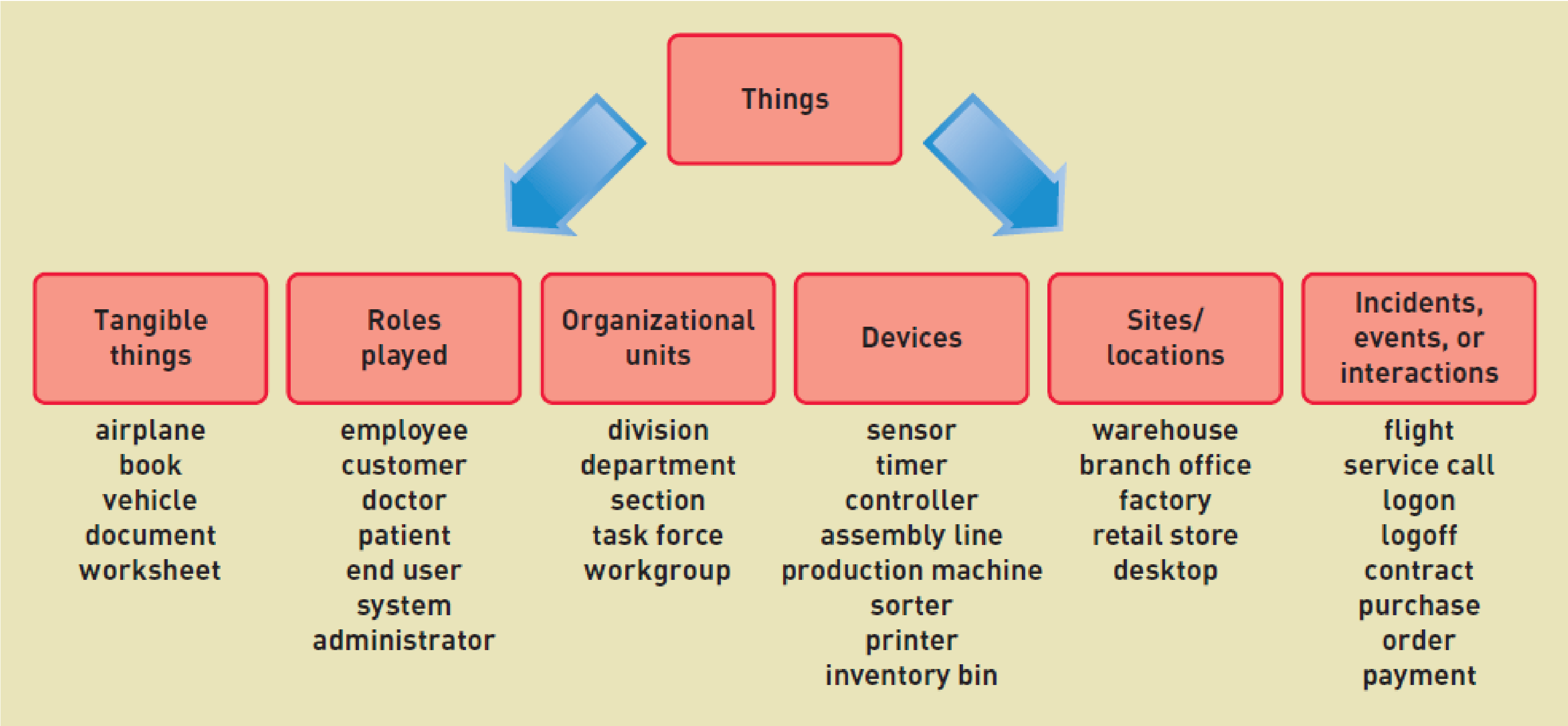
Things in the Problem Domain

Two Techniques for Identifying them

- Brainstorming Technique
  - Use a checklist of all of the usual types of things typically found and brainstorm to identify domain classes of each type
- Noun Technique
  - Identify all of the nouns that come up when the system is described and determine if each is a domain class, an attribute, or not something we need to remember

Brainstorming Technique

- Are there any tangible things? Are there any organizational units? Sites/locations? Are there incidents or events that need to be recorded?



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

Brainstorming Technique: Steps

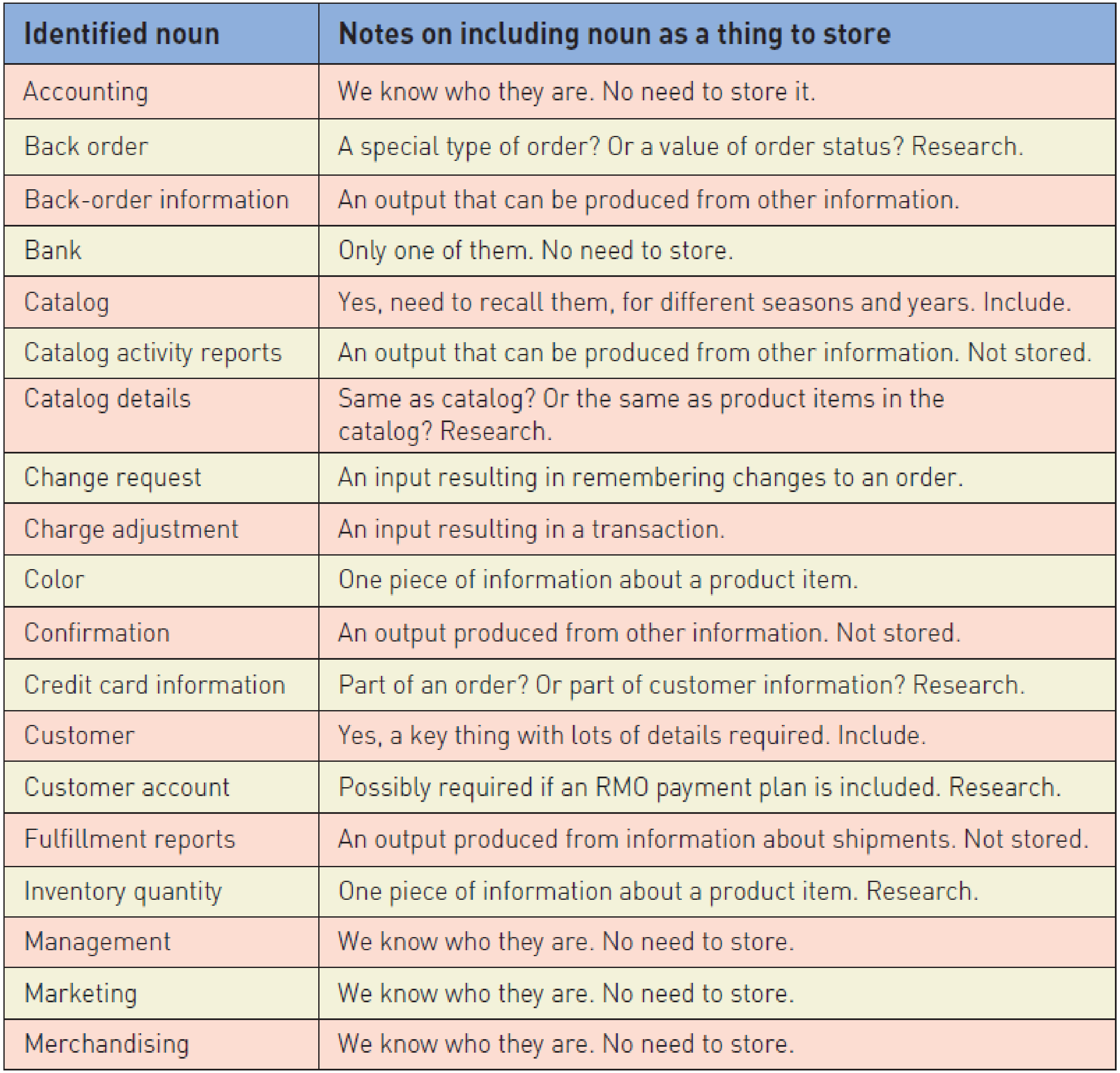
1. Identify a user and a set of use cases
2. Brainstorm with the user to identify things involved when carrying out the use case—that is, things about which information should be captured by the system.
3. Use the types of things (categories) to systematically ask questions about potential things, such as the following: Are there any tangible things you store information about? Are there any locations involved? Are there roles played by people that you need to remember?
4. Continue to work with all types of users and stakeholders to expand the brainstorming list
5. Merge the results, eliminate any duplicates, and compile an initial list

The Noun Technique

- A technique to identify problem domain classes (things) by finding, classifying, and refining a list of nouns that come up in in discussions or documents
- Popular technique. Systematic.
- Does end up with long lists and many nouns that are not things that need to be stored by the system
- Difficulty identifying synonyms and things that are really attributes
- Good place to start when there are no users available to help brainstorm

Partial List of Nouns for RMO

With notes on whether to include as domain class



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.
Marketing	We know who they are. No need to store.
Merchandising	We know who they are. No need to store.

The Noun Technique: Steps

1. Using the use cases, actors, and other information about the system— including inputs and outputs—identify all nouns.
   - For the RMO CSMS, the nouns might include customer, product item, sale, confirmation, transaction, shipping, bank, change request, summary report, management, transaction report, accounting, back order, back order notification, return, return confirmation…
2. Using other information from existing systems, current procedures, and current reports or forms, add items or categories of information needed.
   - For the RMO CSMS, these might include price, size, color, style, season, inventory quantity, payment method, and shipping address.

The Noun Technique: Steps (continued)

3. As this list of nouns builds, refine it. Ask these questions about each noun to help you decide whether you should include it:
   - Is it a unique thing the system needs to know about?
   - Is it inside the scope of the system I am working on?
   - Does the system need to remember more than one of these items?

   Ask these questions to decide to exclude it:
   - Is it really a synonym for some other thing I have identified?
   - Is it really just an output of the system produced from other information I have identified?
   - Is it really just an input that results in recording some other information I have identified?

   Ask these questions to research it:
   - Is it likely to be a specific piece of information (attribute) about some other thing I have identified?
   - Is it something I might need if assumptions change?

The Noun Technique: Steps (continued)

4. Create a master list of all nouns identified and then note whether each one should be included, excluded, or researched further.

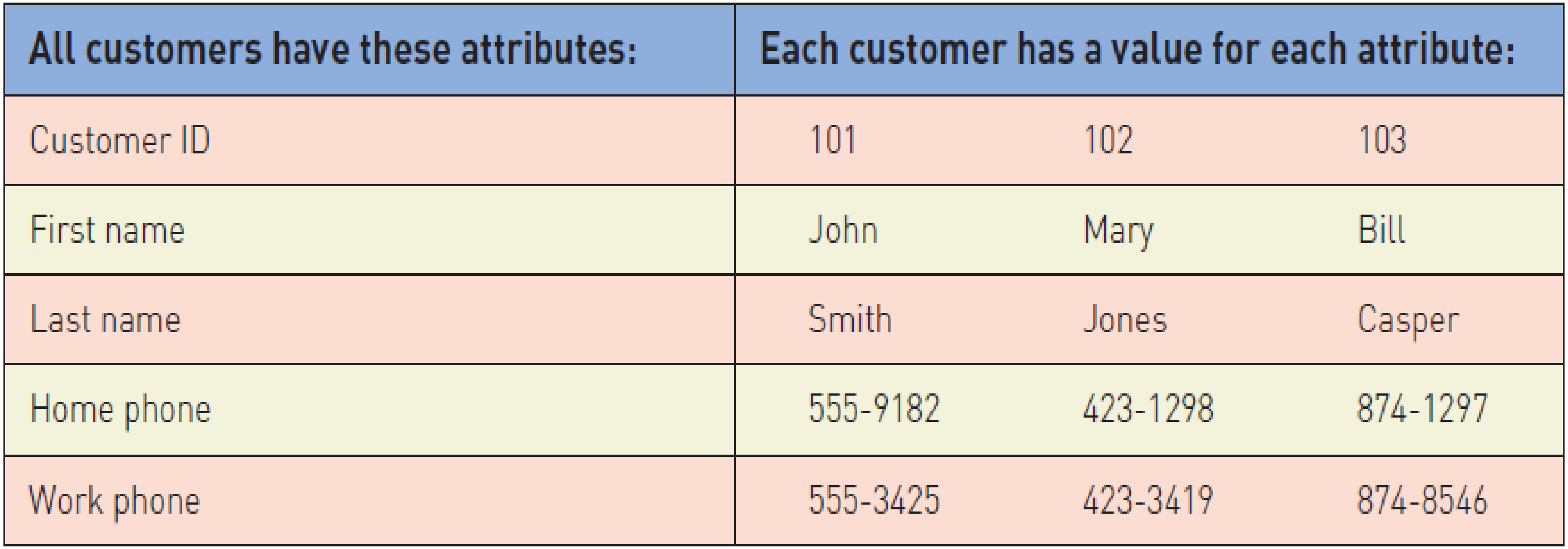
5. Review the list with users, stakeholders, and team members and then define the list of things in the problem domain.

Details about Domain Classes

- Attribute— describes one piece of information about each instance of the class
  - Customer has first name, last name, phone number
- Identifier or key
  - One attribute uniquely identifies an instance of the class. Required for data entities, optional for domain classes. Customer ID identifies a customer
- Compound attribute
  - Two or more attributes combined into one structure to simplify the model. (E.g., address rather than including number, street, city, state, zip separately). Sometimes an identifier or key is a compound attribute.

Attributes and Values

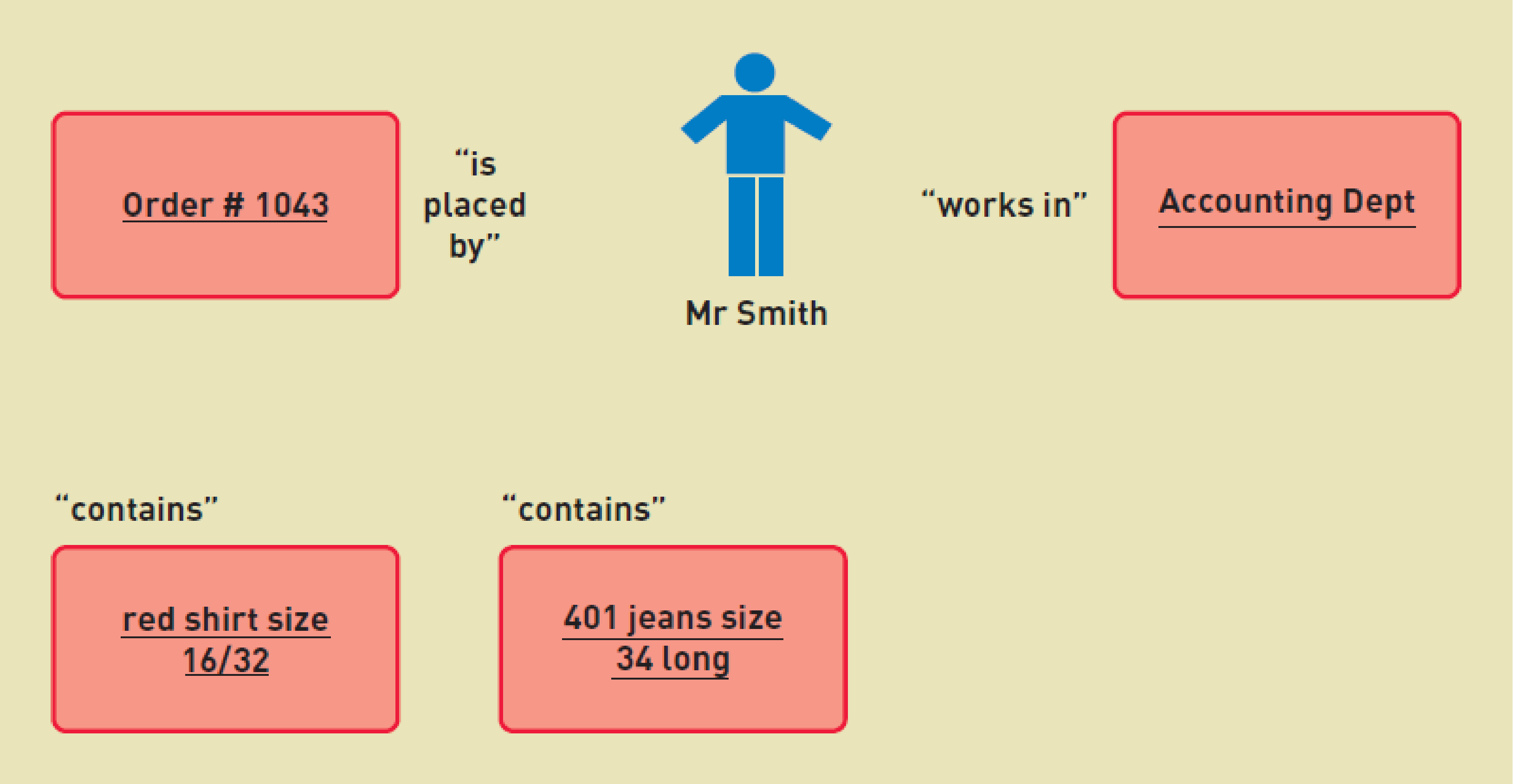
- Class is a type of thing. Object is a specific instance of the class. Each instance has its own values for an attribute



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

Associations Among Things

- Association— a naturally occurring relationship between classes (UML term)

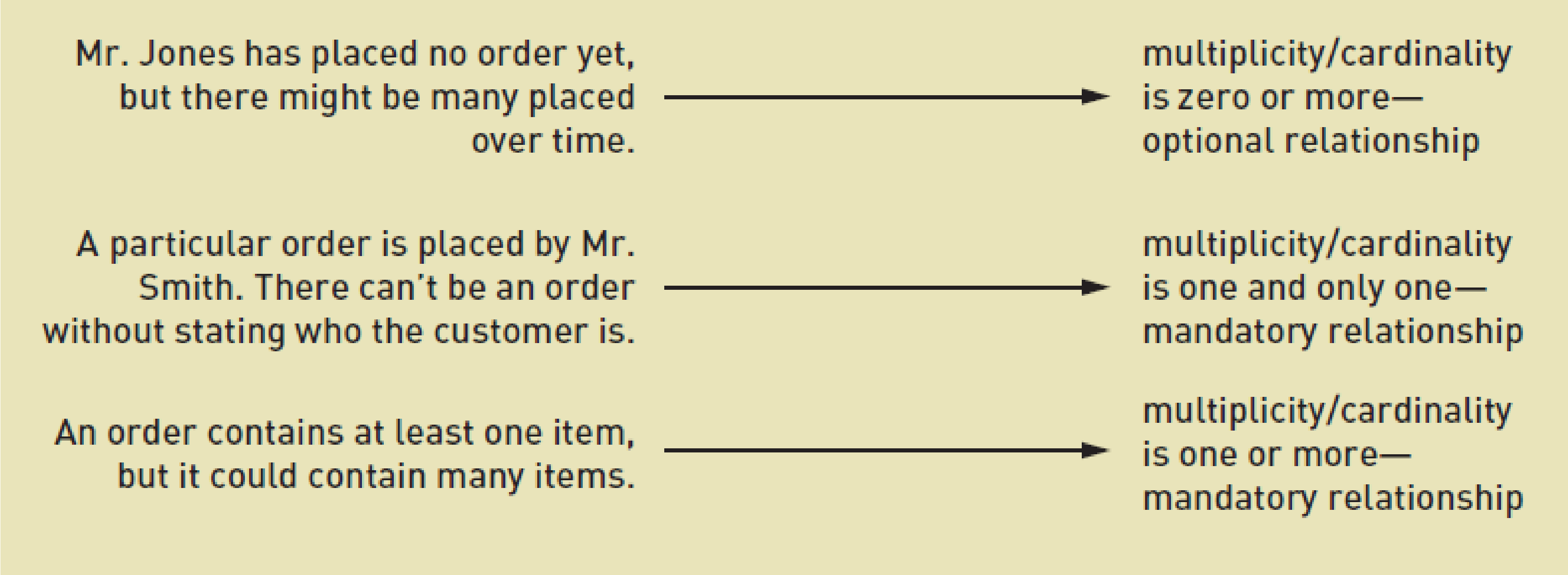


Just to Clarify…

- Called ***association*** on class diagram in UML
  - Multiplicity is term for the number of associations between classes: 1 to 1 or 1 to many
  - We are emphasizing UML in this text
- Called ***relationship*** on ERD in database class
  - Cardinality is term for number of relationships in entity relationship diagrams: 1 to 1 or 1 to many
- Associations and Relationships apply in two directions
  - Read them separately each way
  - A customer places an order
  - An order is placed by a customer

Minimum and Maximum Multiplicity

- Associations have minimum and maximum constraints
  - minimum is zero, the association is optional
  - If minimum is at least one, the association is mandatory



Mr. Jones has placed no order yet, but there might be many placed over time. → multiplicity/cardinality is zero or more—optional relationship

A particular order is placed by Mr. Smith. There can't be an order without stating who the customer is. → multiplicity/cardinality is one and only one—mandatory relationship

An order contains at least one item, but it could contain many items. → multiplicity/cardinality is one or more—mandatory relationship

Types of Associations

- Binary Association
  - Associations between exactly two different classes
    - Course Section includes Students
    - Members join Club
- Unary Association (recursive)
  - Associations between two instances of the same class
    - Person married to person
    - Part is made using parts
- Ternary Association (three)
- N-ary Association (between n)

Semantic Net

Shows instances and how they are linked

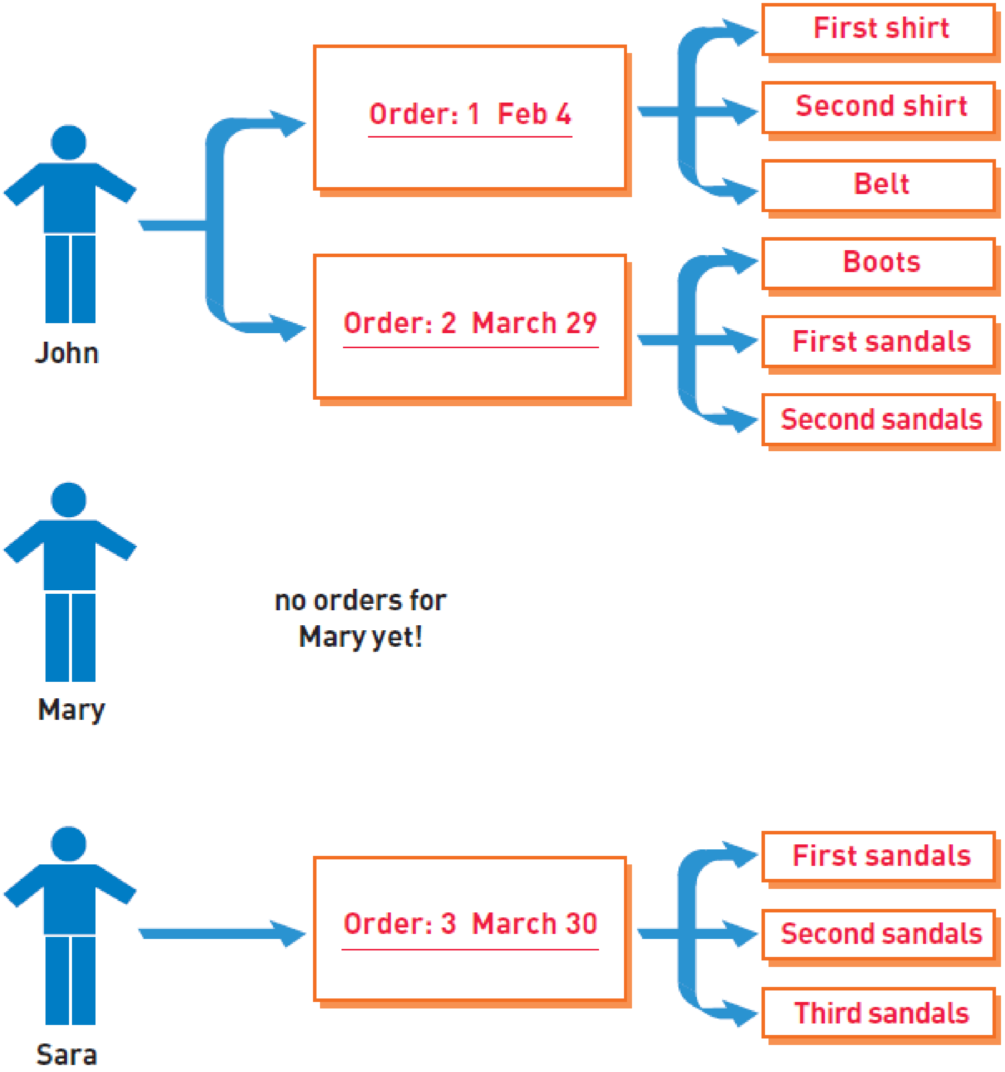
Example shows instances of three classes

Quick quiz:

How many associations are there?

What are the minimum and maximum multiplicities in each direction?

What type of associations are they?



Semantic Net

Shows instances and how they are linked

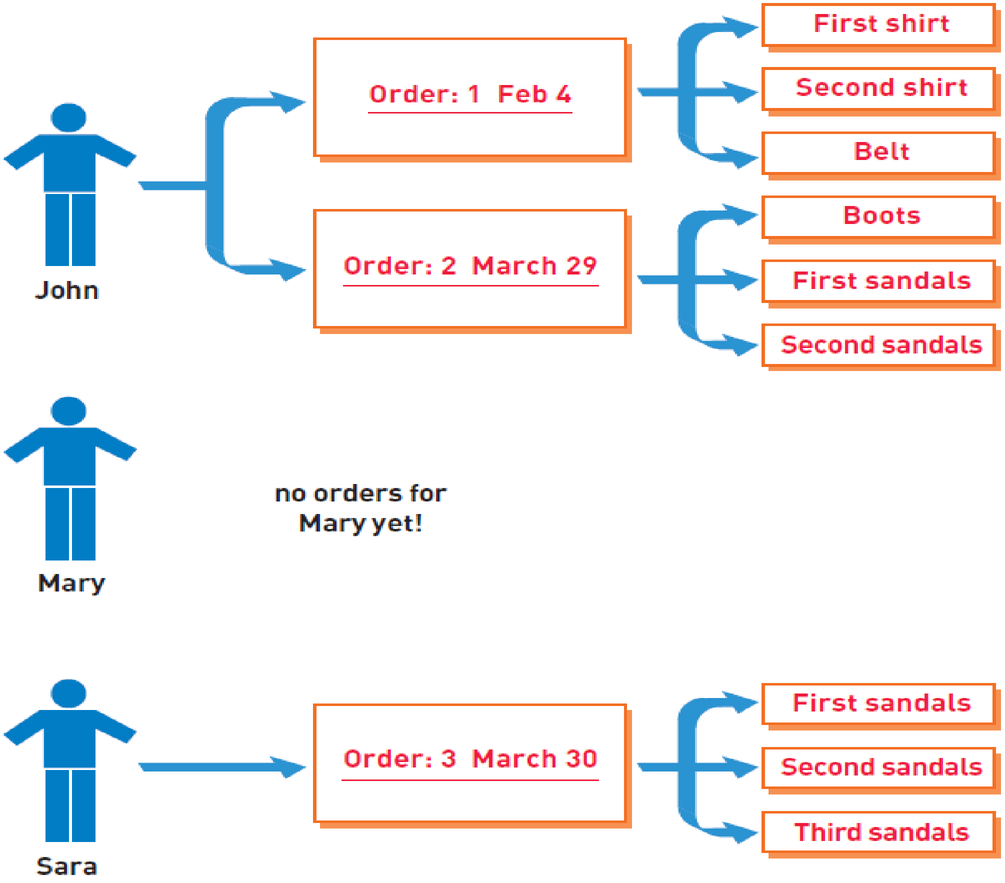
Example shows instances of three classes

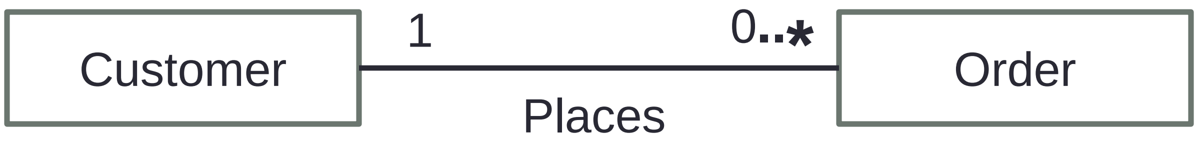
Quick quiz:

How many associations are there?

What are the minimum and maximum multiplicities in each direction?

What type of associations are they?



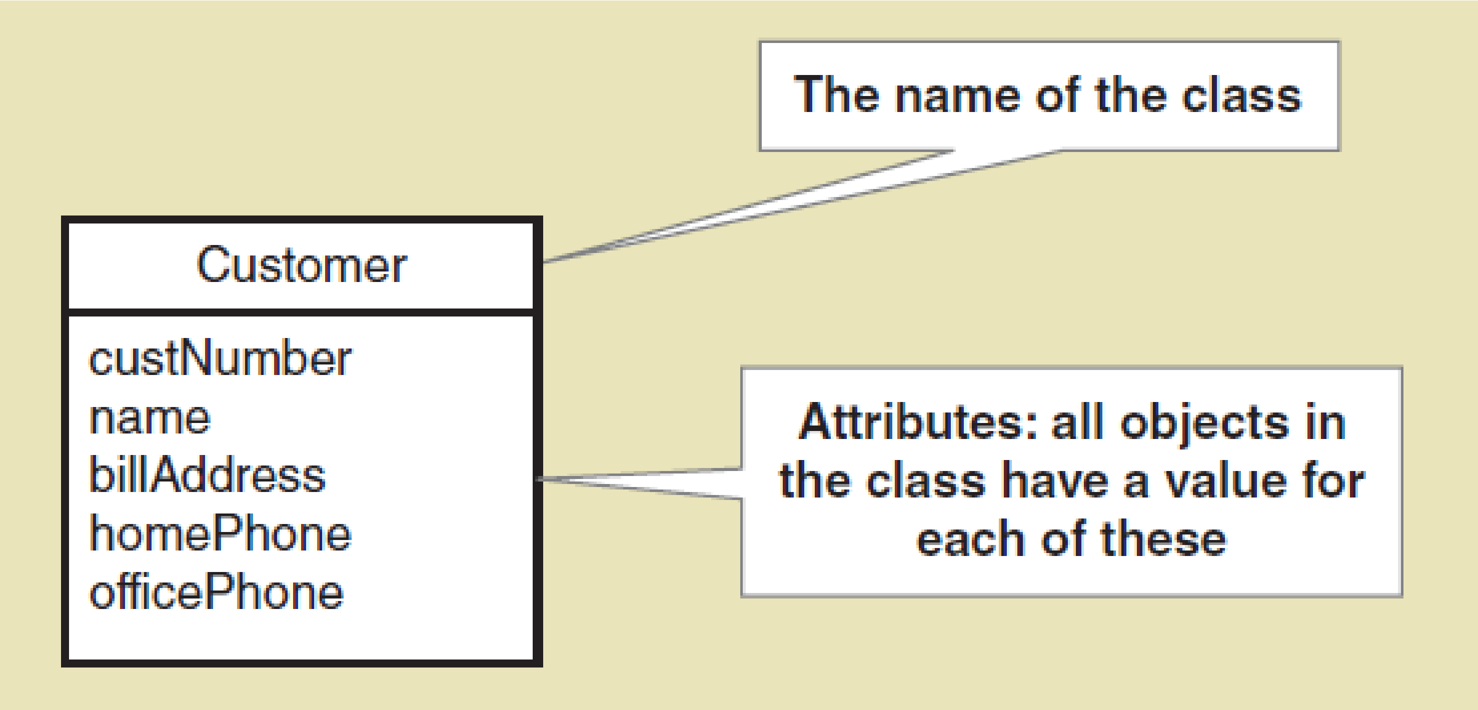


The Domain Model Class Diagram

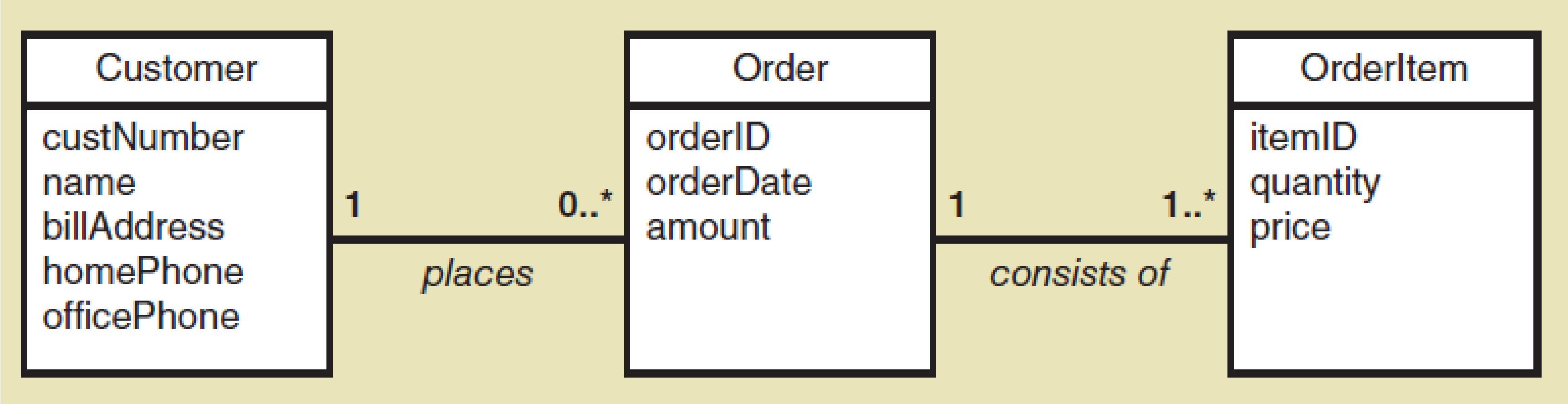
- Class
  - A category of classification used to describe a collection of objects
- Domain Class
  - Classes that describe objects in the problem domain
- Class Diagram
  - A UML diagram that shows classes with attributes and associations (plus methods if it models software classes)
- Domain Model Class Diagram
  - A class diagram that only includes classes from the problem domain, not software classes so no methods

Domain Class Notation

- Domain class has no methods
- Class name is always capitalized
- Attribute names are not capitalized and use camelback notation (words run together and second word is capitalized)

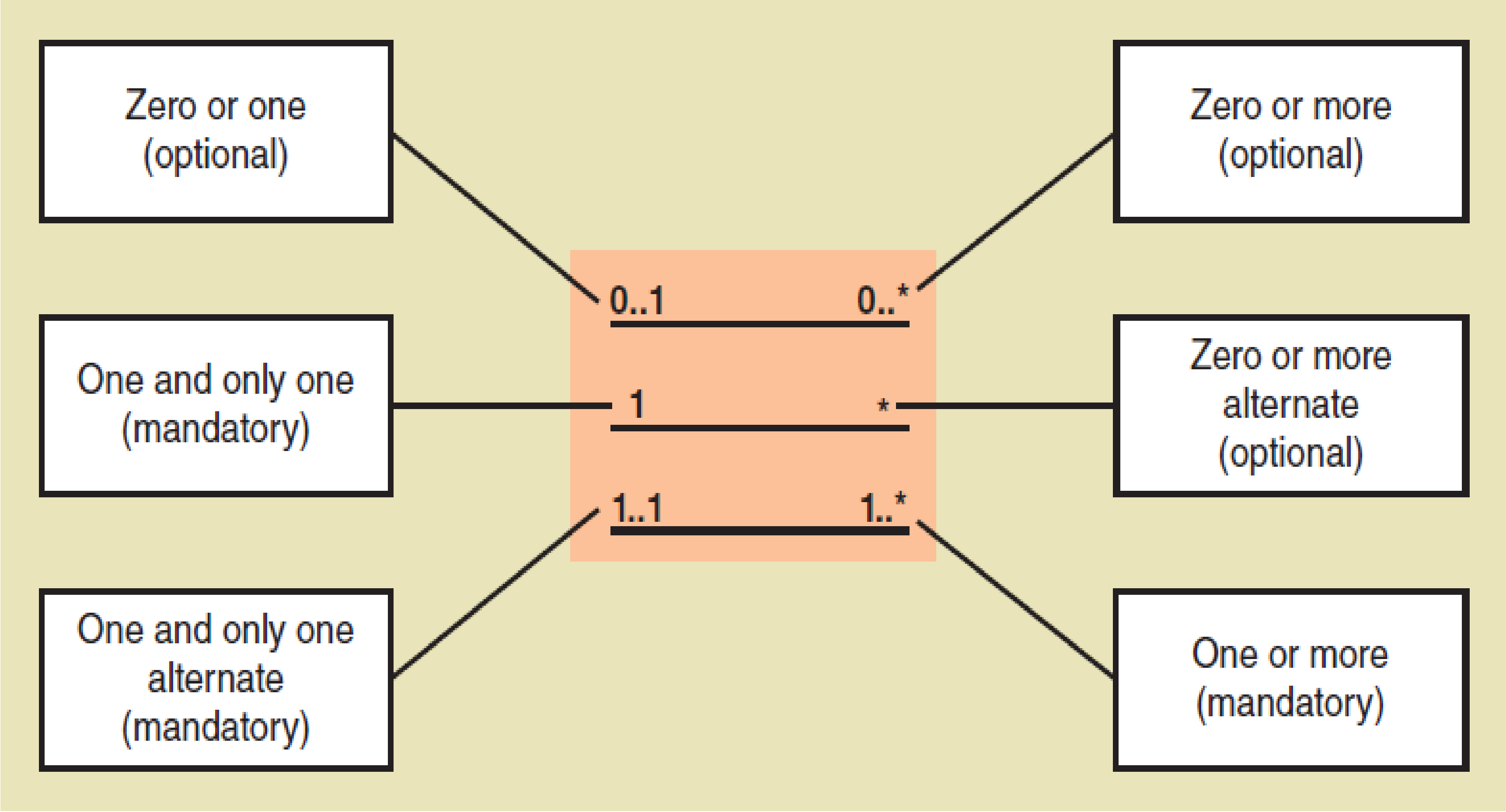


A Simple Domain Model Class Diagram



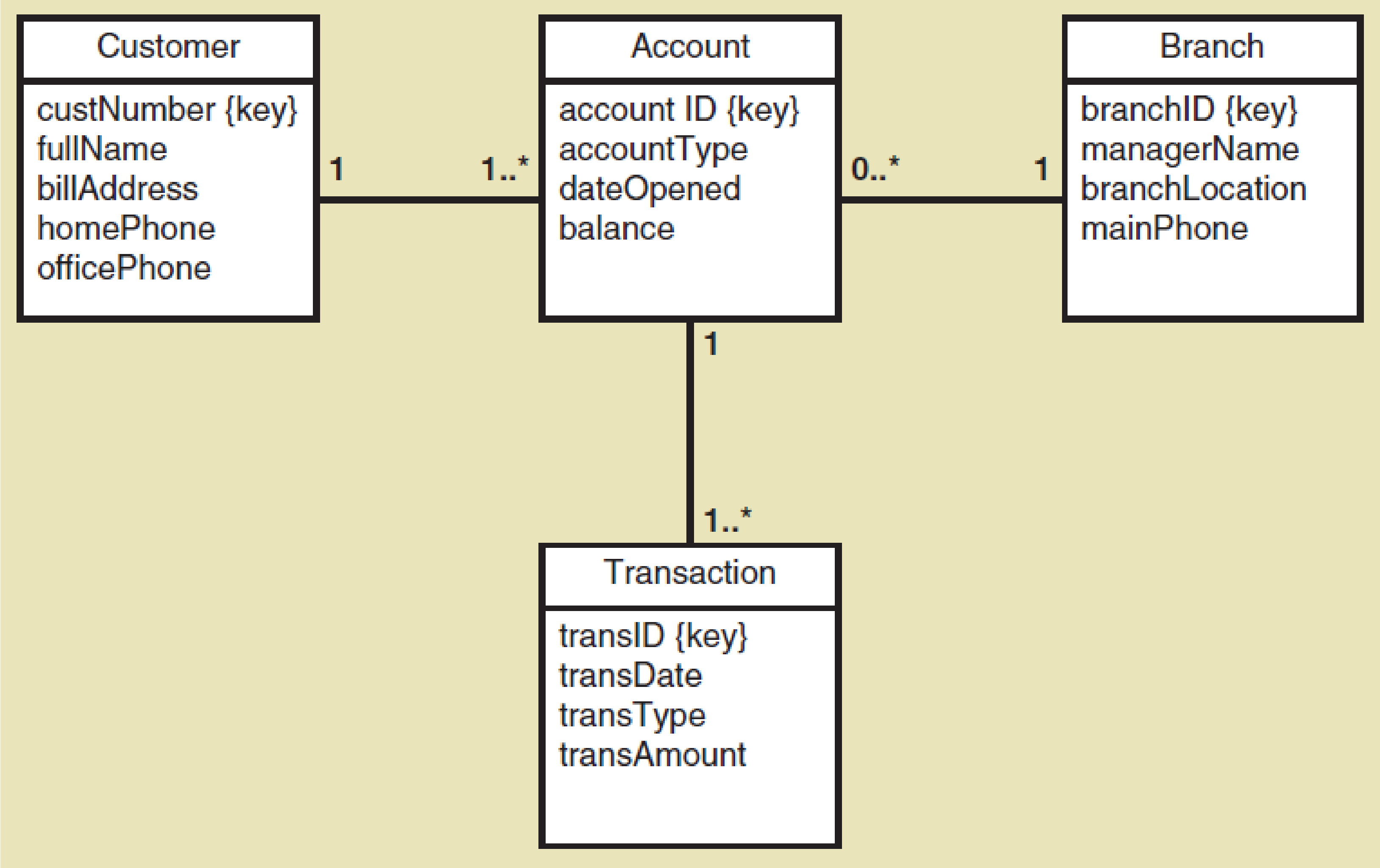
- Note: This diagram matches the semantic net shown previously
  - A customer places zero or more orders
  - An order is placed by exactly one customer
  - An order consists of one or more order items
  - An order item is part of exactly one order

UML Notation for Multiplicity



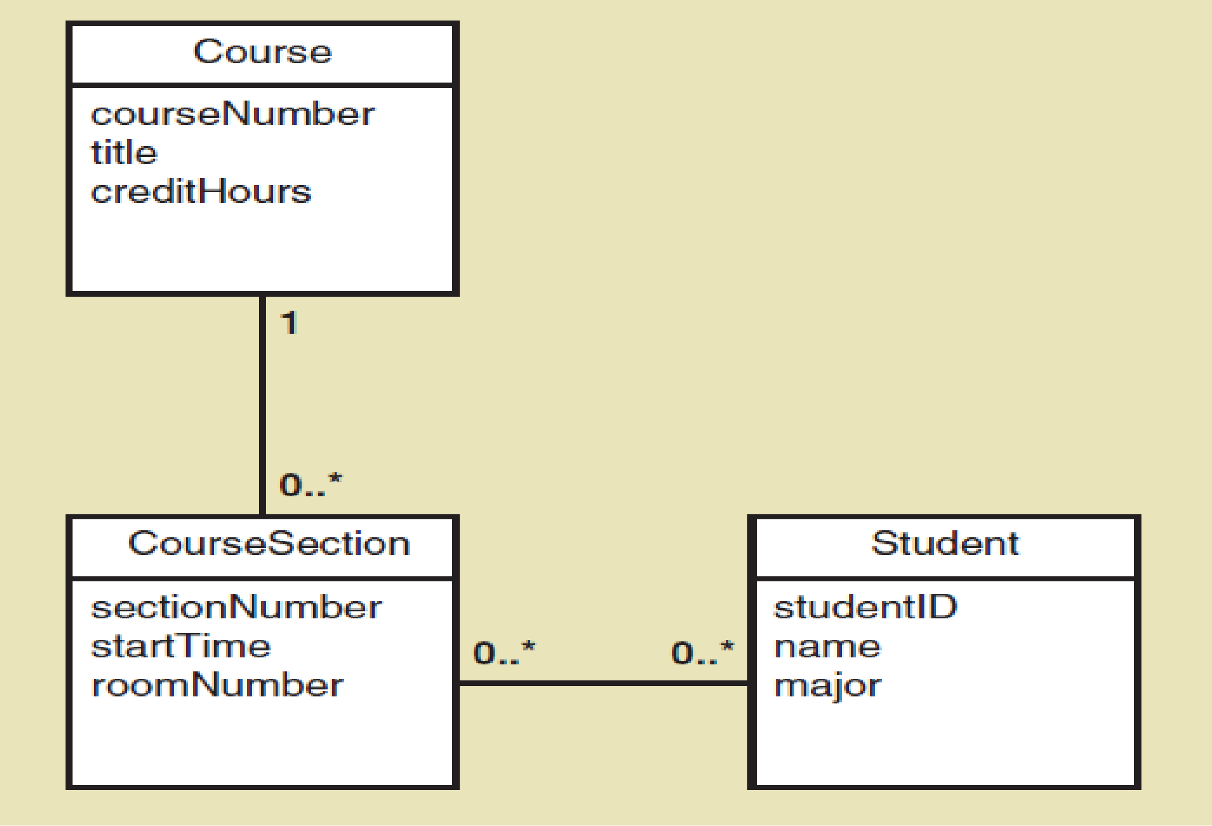
Domain Model Class Diagram

for a bank with many branches



Domain Model Class Diagram

for course enrollment at a university

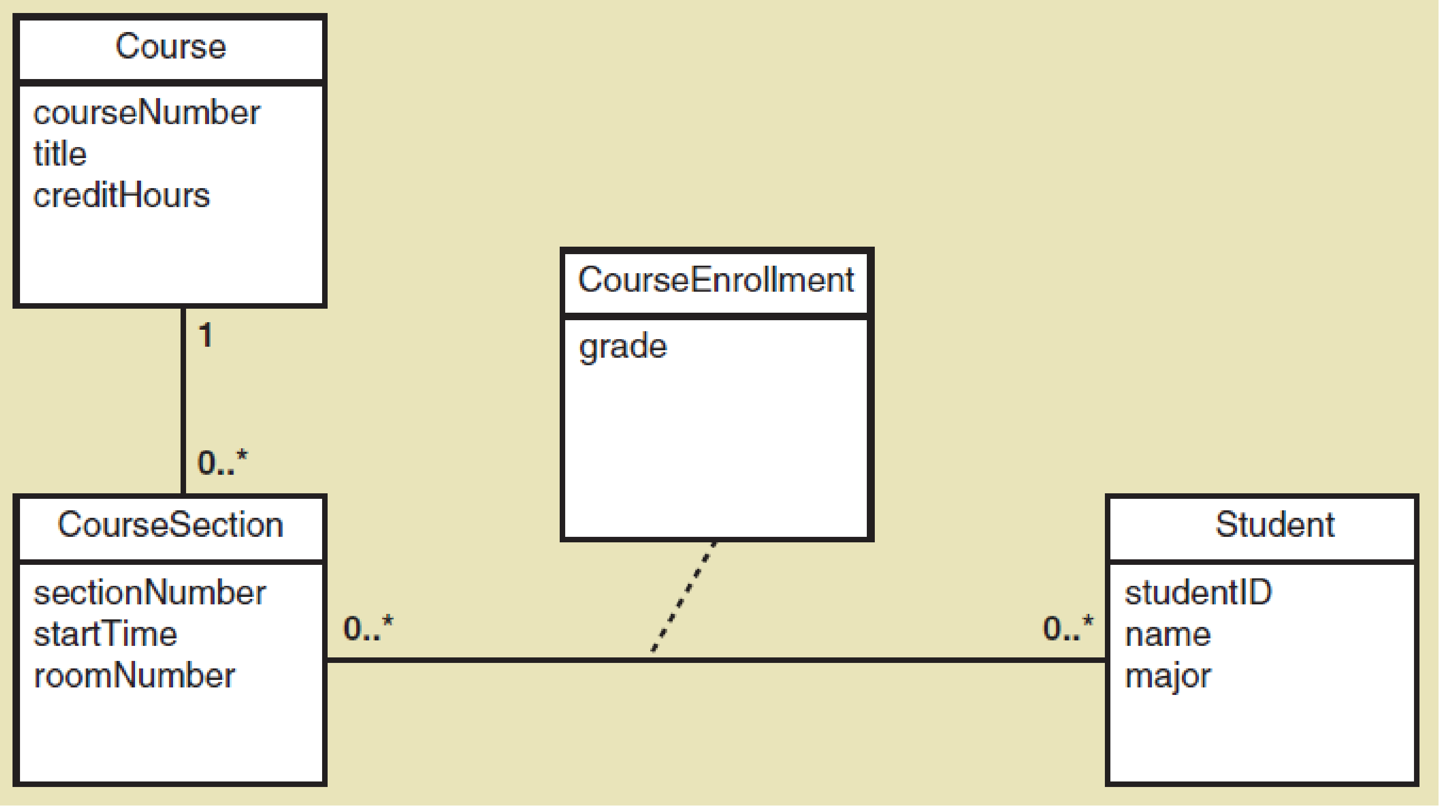


- Where is each student’s grade remembered in this model?
  - Each section has many grades and each grade is association with a student
  - Each student has many grades and each grade is association with a section

Refined Course Enrollment Model

with an Association Class CourseEnrollment

- Association class— an association that is treated as a class in a many to many association because it has attributes that need to be remembered, such as grade



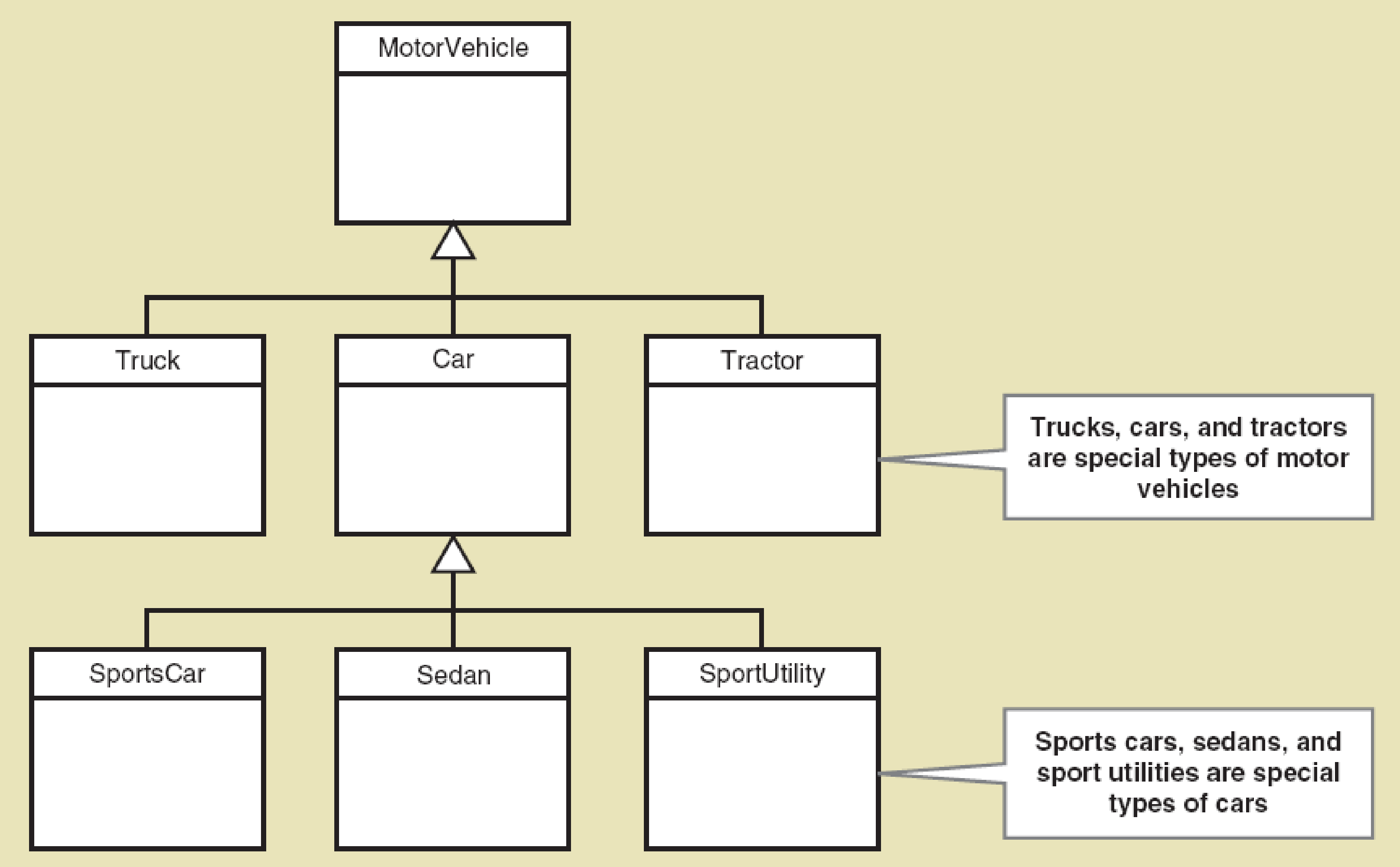
More Complex Issues about Classes:

Generalization/Specialization Relationships

- Generalization/Specialization
  - A hierarchical relationship where subordinate classes are special types of the superior classes. Often called an Inheritance Hierarchy
- Superclass
  - the superior or more general class in a generalization/specialization hierarchy
- Subclass
  - the subordinate or more specialized class in a generalization/specialization hierarchy
- Inheritance
  - the concept that subclasses classes inherit characteristics of the more general superclass

Generalization/Specialization

Inheritance

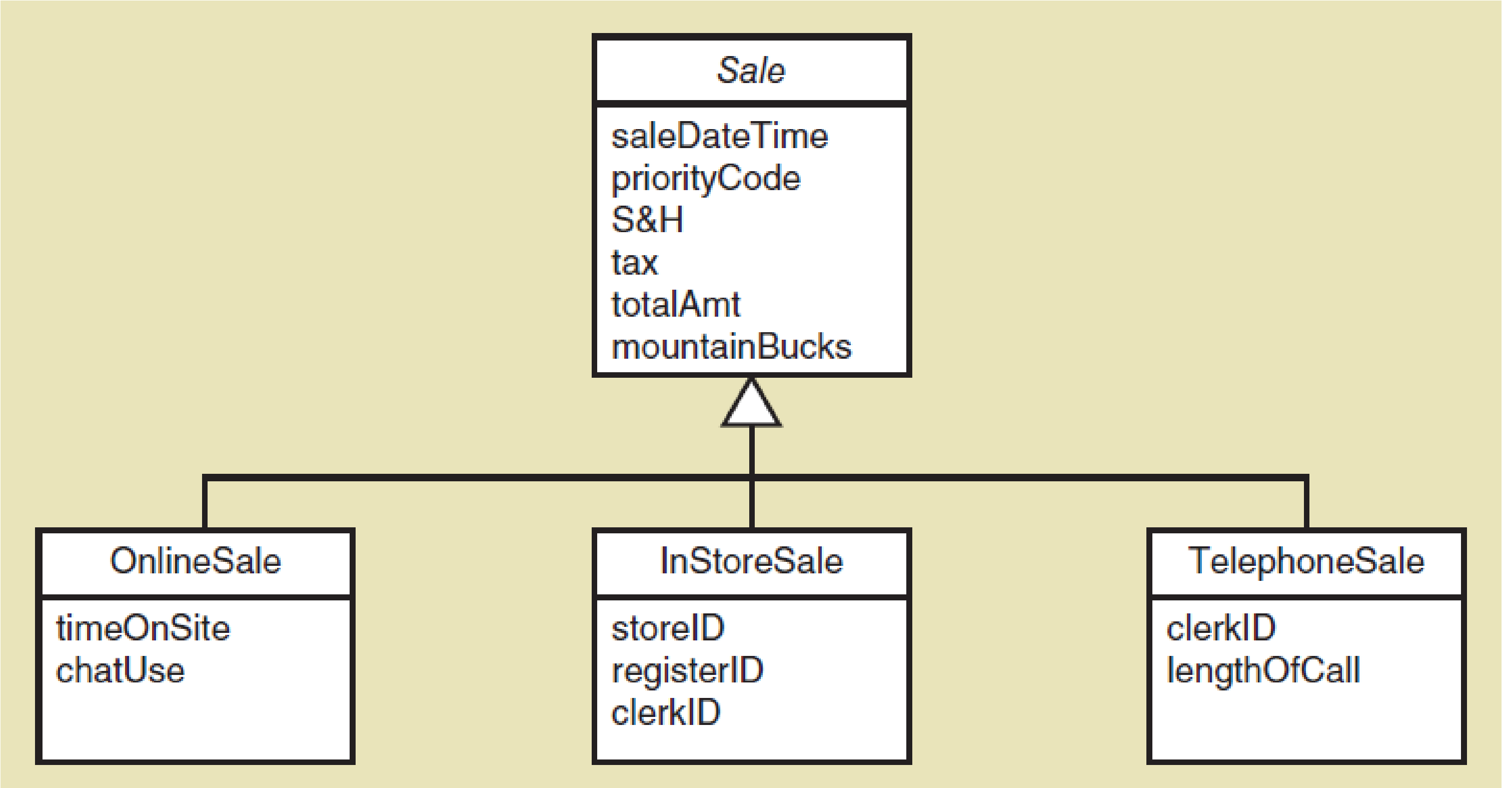


Trucks, cars, and tractors are special types of motor vehicles

Sports cars, sedans, and sport utilities are special types of cars

Generalization/Specialization

Inheritance for RMO Three Types of Sales

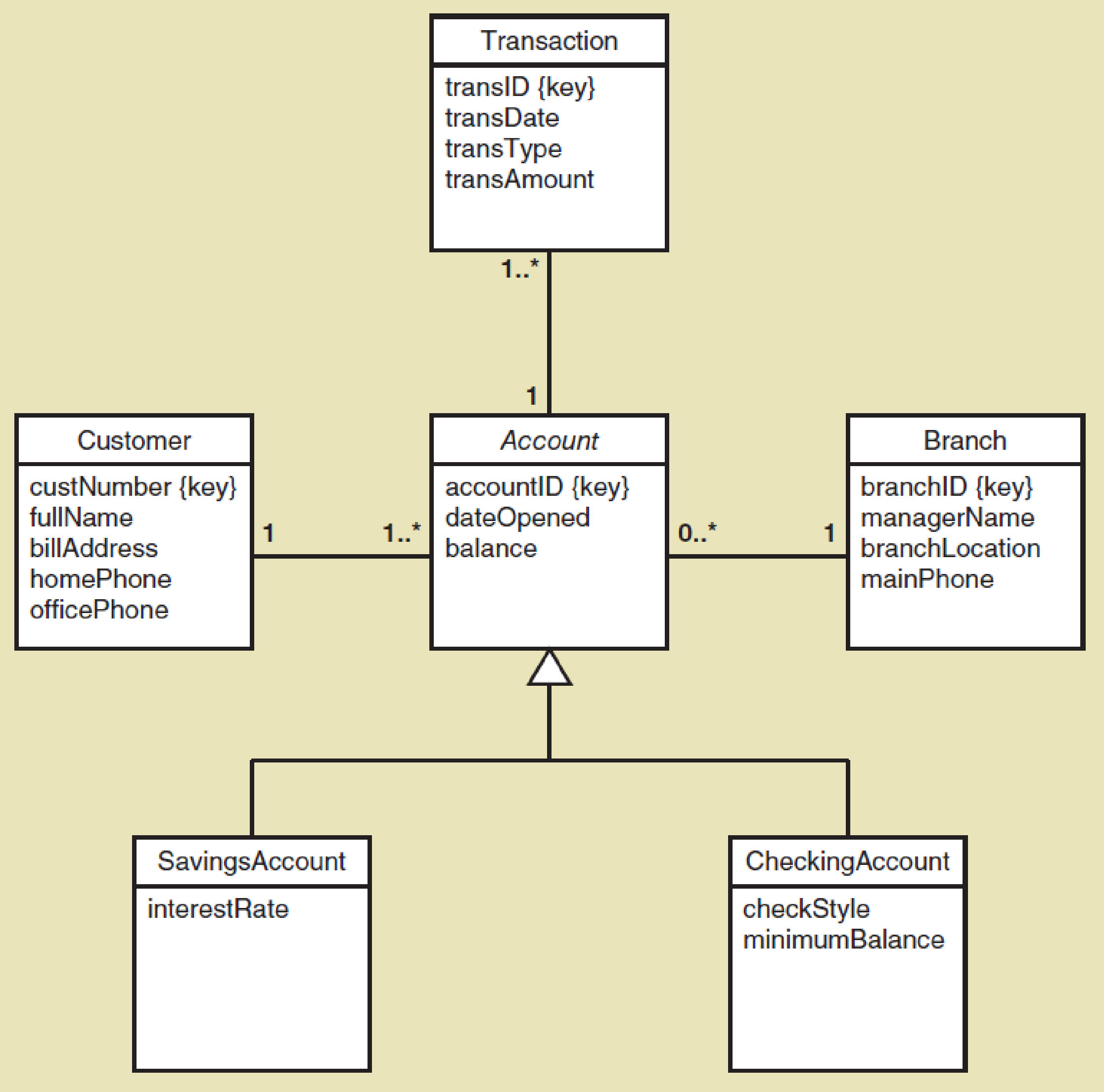


- Abstract class— a class that allow subclasses to inherit characteristics but never gets instantiated. In Italics (*Sale* above)
- Concrete class— a class that can have instances

Generalization/Specialization

Inheritance for the Bank with Special Types of Accounts

- A SavingsAccount has 4 attributes
- A CheckingAccount Has 5 attributes
- Note: the subclasses inherit the associations, too



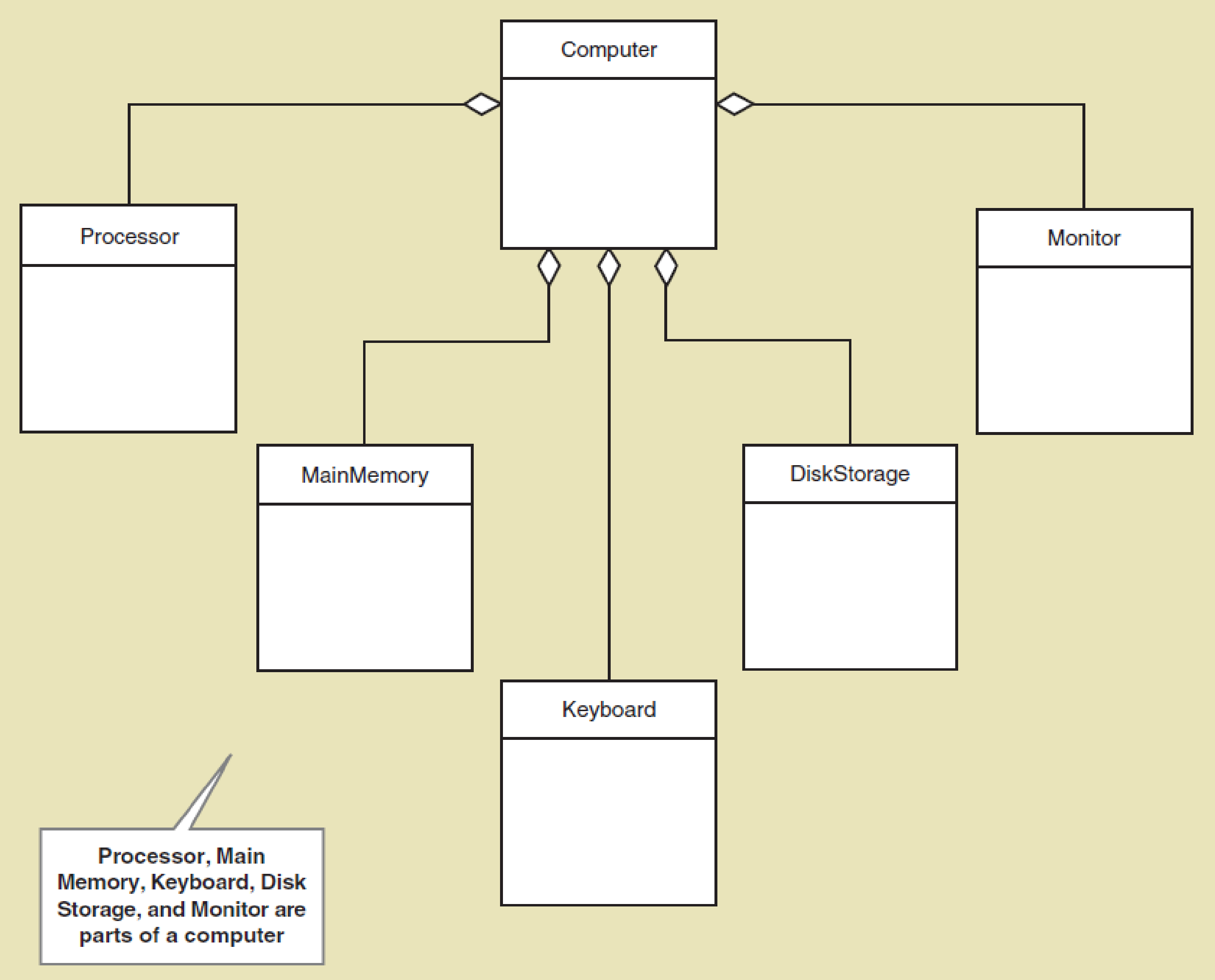
More Complex Issues about Classes:
Whole Part Relationships

- Whole-part relationship— a relationship between classes where one class is part of or a component portion of another class
- Aggregation— a whole part relationship where the component part exists separately and can be removed and replaced (UML diamond symbol, next slide)
  - Computer has disk storage devices
  - Car has wheels
- Composition— a whole part relationship where the parts can no longer be removed (filled in diamond symbol)
  - Hand has fingers
  - Circuit has chips

Whole Part Relationships

Computer and its Parts

- Note: this is composition, with diamond symbol.
- Whole part can have multiplicity symbols, too (not shown)



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

More on UML Relationships

- There are actually three types of ***relationships*** in class diagrams
  - Association Relationships
    - These are associations discussed previously, just like ERD relationships
  - Whole Part Relationships
    - One class is a component or part of another class
  - Generalizations/Specialization Relationships
    - Inheritance
- So, try not to confuse relationship with association

RMO CSMS Project
Domain Model Class Diagrams

- There are several ways to create the domain model class diagram for a project
- RMO CSMS has 27 domain classes overall
- Can create one domain model class diagram per subsystem for those working on a subsystem
- Can create one overall domain model class diagram to provide an overview of the whole system
- Usually in early iterations, an initial draft of the domain model class diagram is completed to guide development and kept up to date

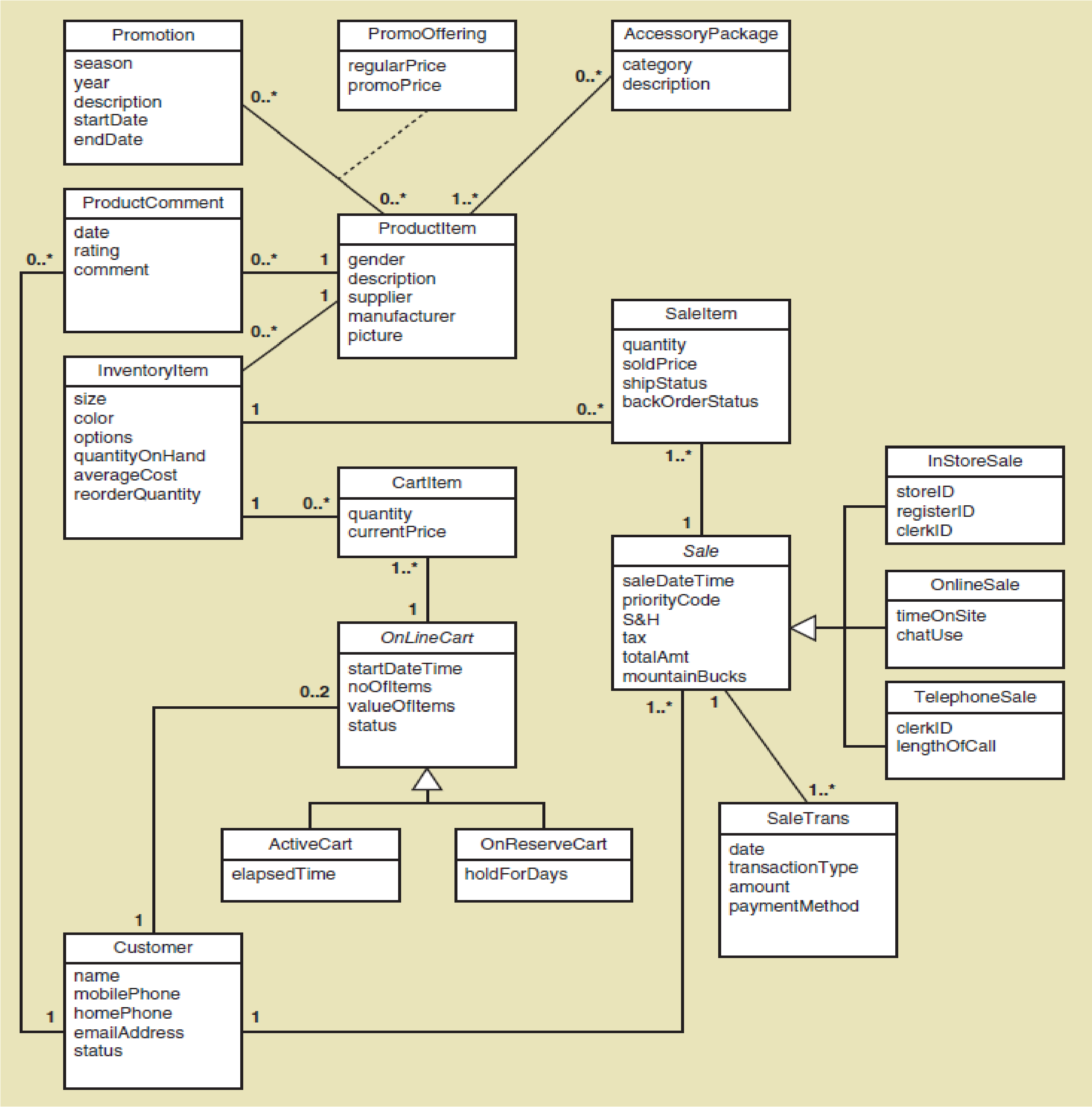
RMO CSMS Project

Domain Model Class Diagrams

- There are several ways to create the domain model class diagram for a project
- RMO CSMS has 27 domain classes overall
- Can create one domain model class diagram per subsystem for those working on a subsystem
- Can create one overall domain model class diagram to provide an overview of the whole system
- Usually in early iterations, an initial draft of the domain model class diagram is completed to guide development and kept up to date

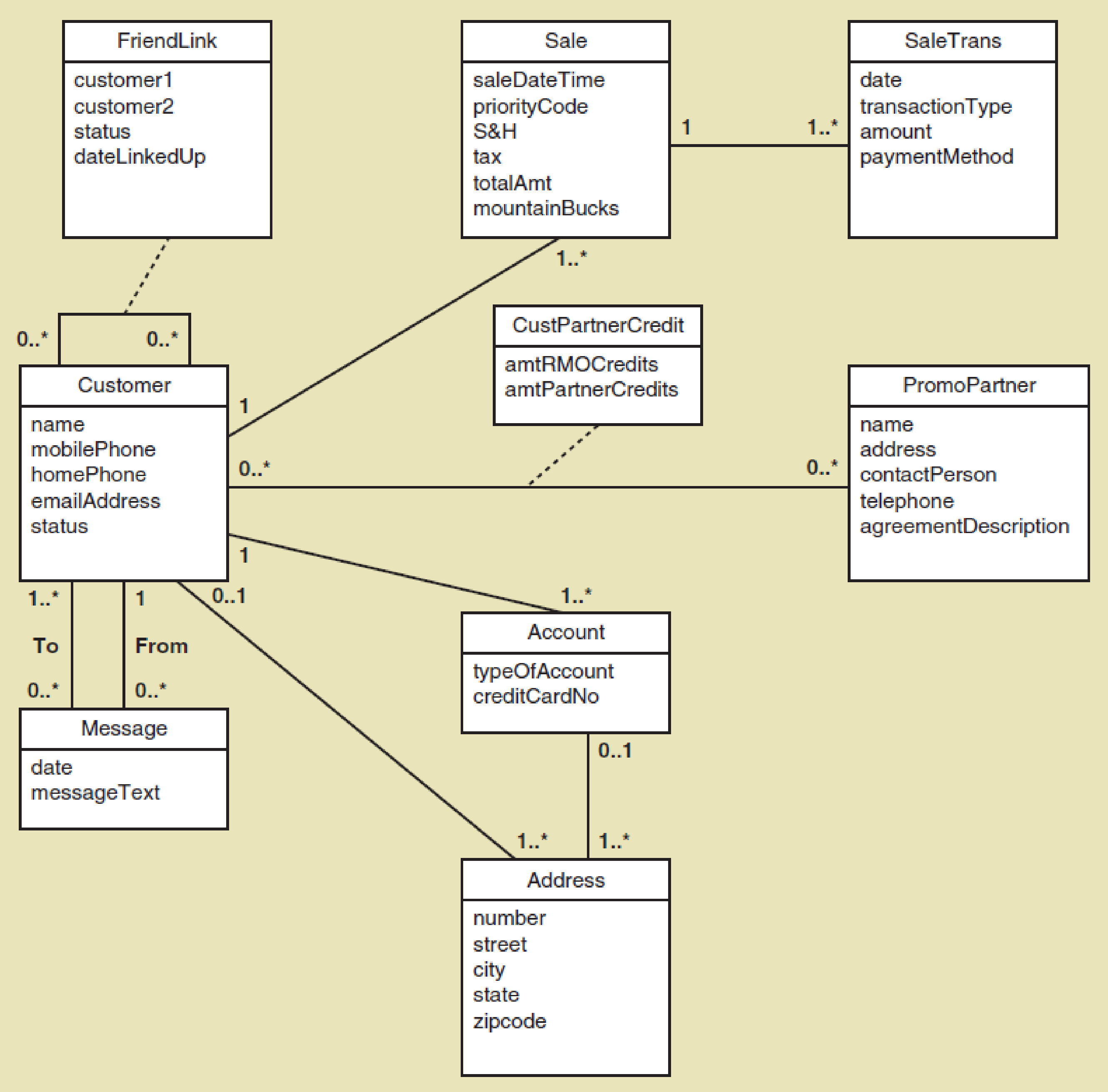
RMO CSMS Project

Sales Subsystem Domain Model Class Diagrams



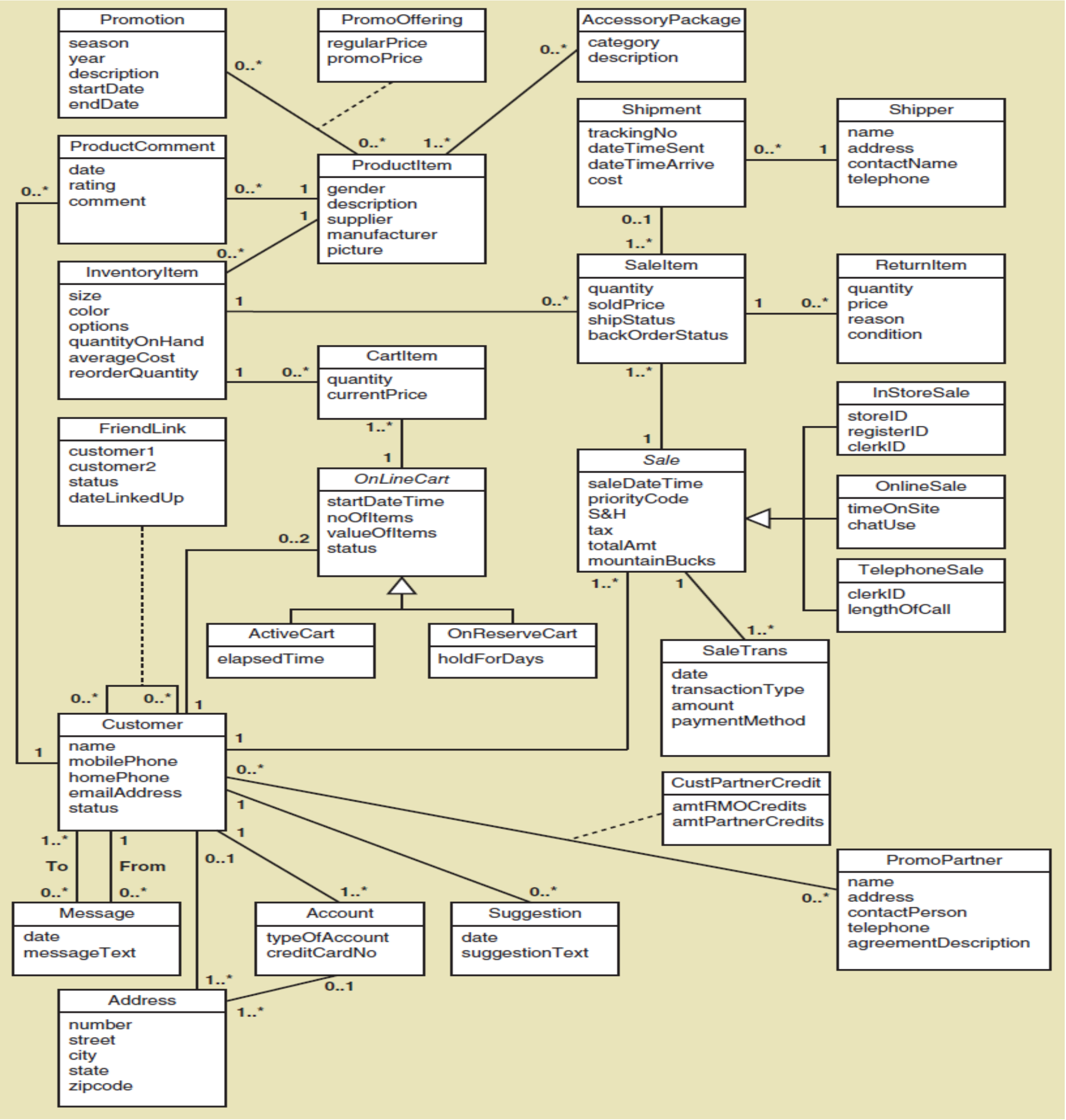
RMO CSMS Project

Customer Account Subsystem Domain Model Class Diagram



RMO CSMS Project

Complete Domain Model Class Diagram



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

Message
date
messageText

Account
typeOfAccount
creditCardNo

Suggestion
date
suggestionText

Address
number
street
city
state
zipcode

RMO CSMS Project
Domain Model Class Diagrams

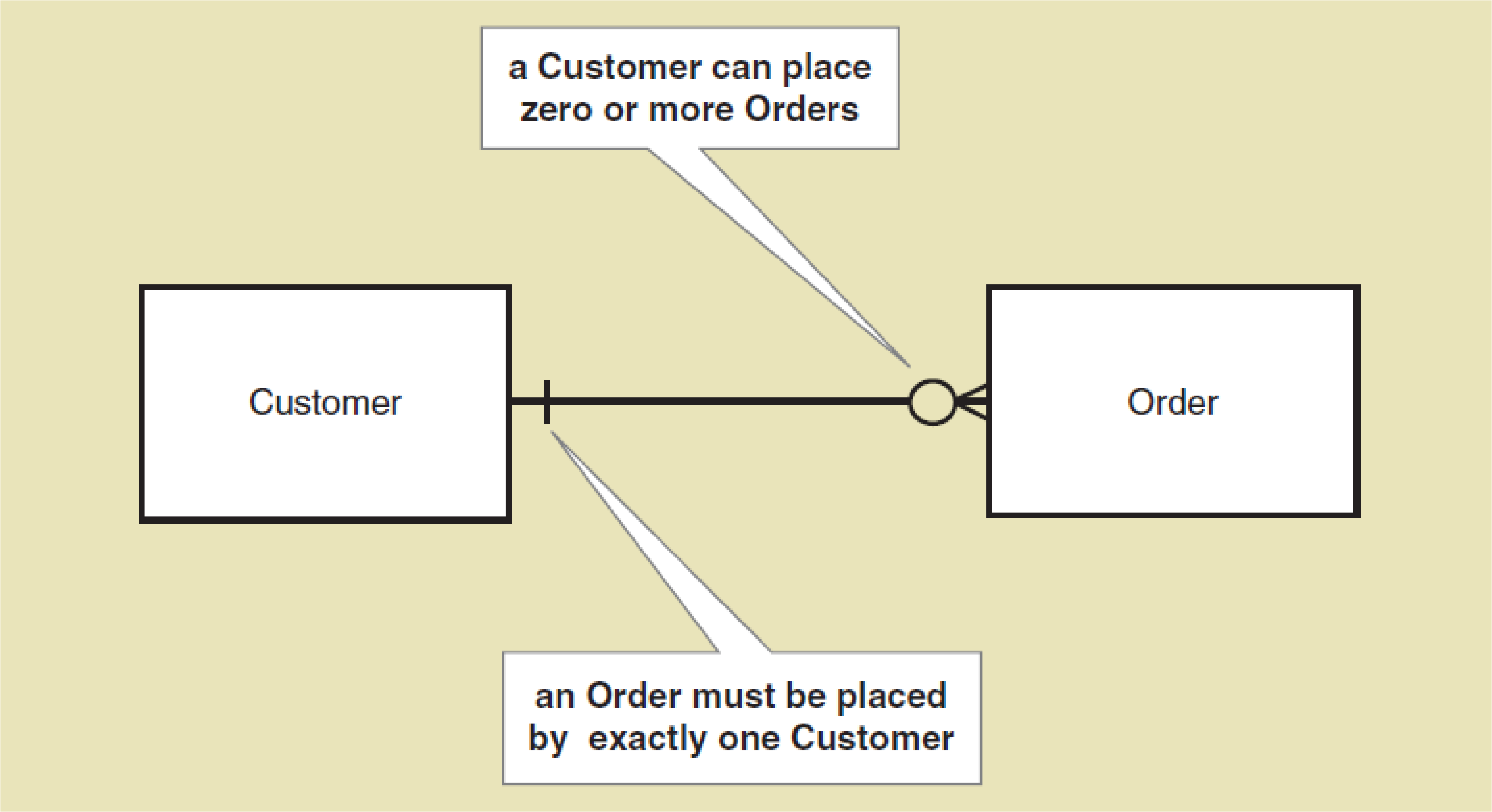
- Given the complete RMO CSMS Domain Model Class Diagram and Sales and Customer Account subsystem examples:
  - Try completing the Order Fulfilment Subsystem Domain Model Class Diagram
  - Try Completing the Marketing Subsystem Domain Model Class Diagram
  - Try Completing the Reporting Subsystem Domain Model Class Diagram
- Review the use cases from the last lecture and decide what classes and associations from the complete model are required for each subsystem
  - Classes and associations might be duplicated in more than one subsystem model

Entity-Relationship Diagrams ERD

- An ERD shows basically the same information as a domain model class diagram
- It is not a UML diagram, but it is widely used by data analysts in database management
- There really is no standard notation, but most developers use the entity and crows feet notation shown in this text
- An ERD is not as effective as Domain Class Diagram for showing generalization/specialization relationships and whole part relationships

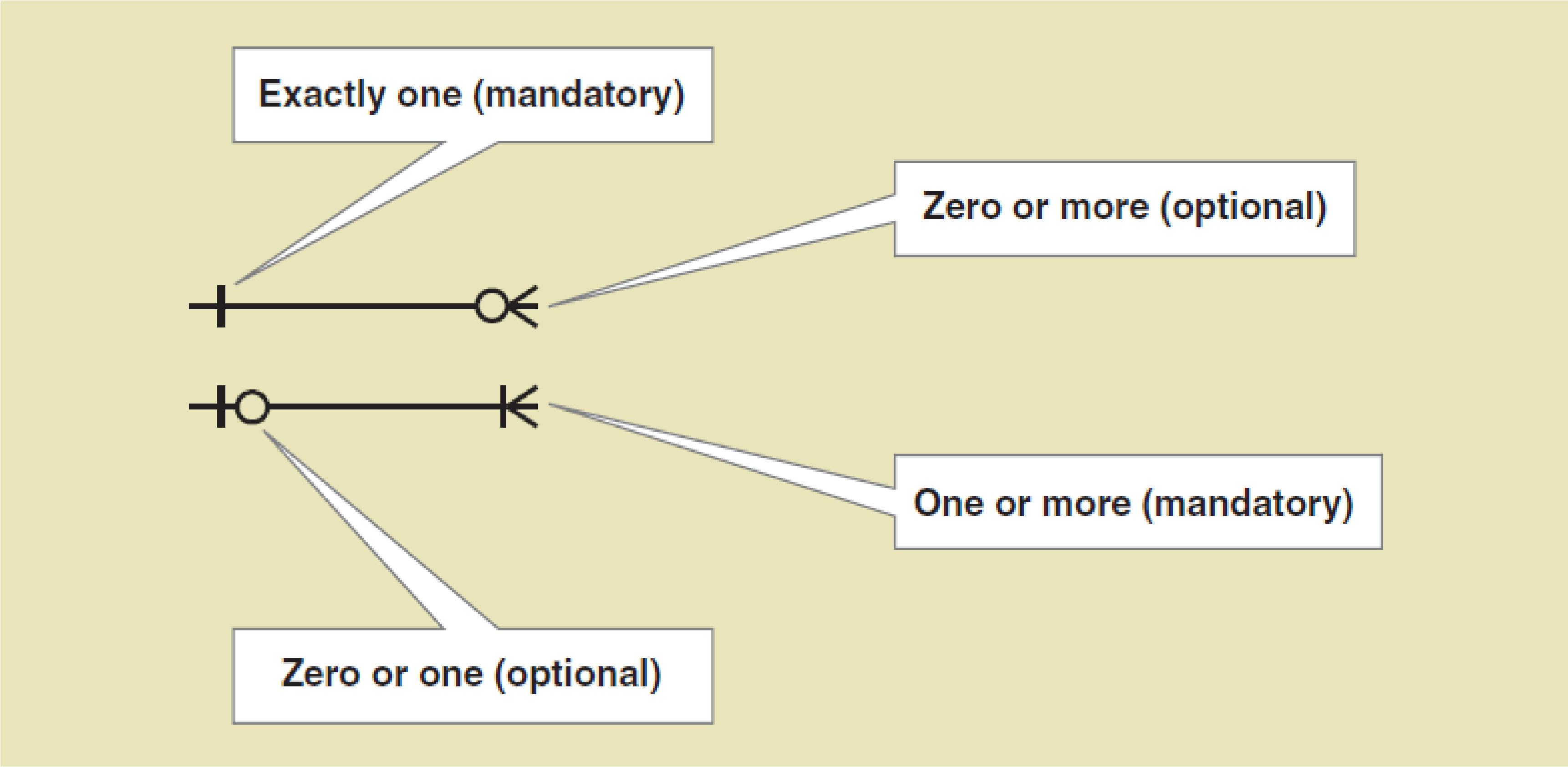
Example of ERD Notation

- A simple ERD without showing attributes

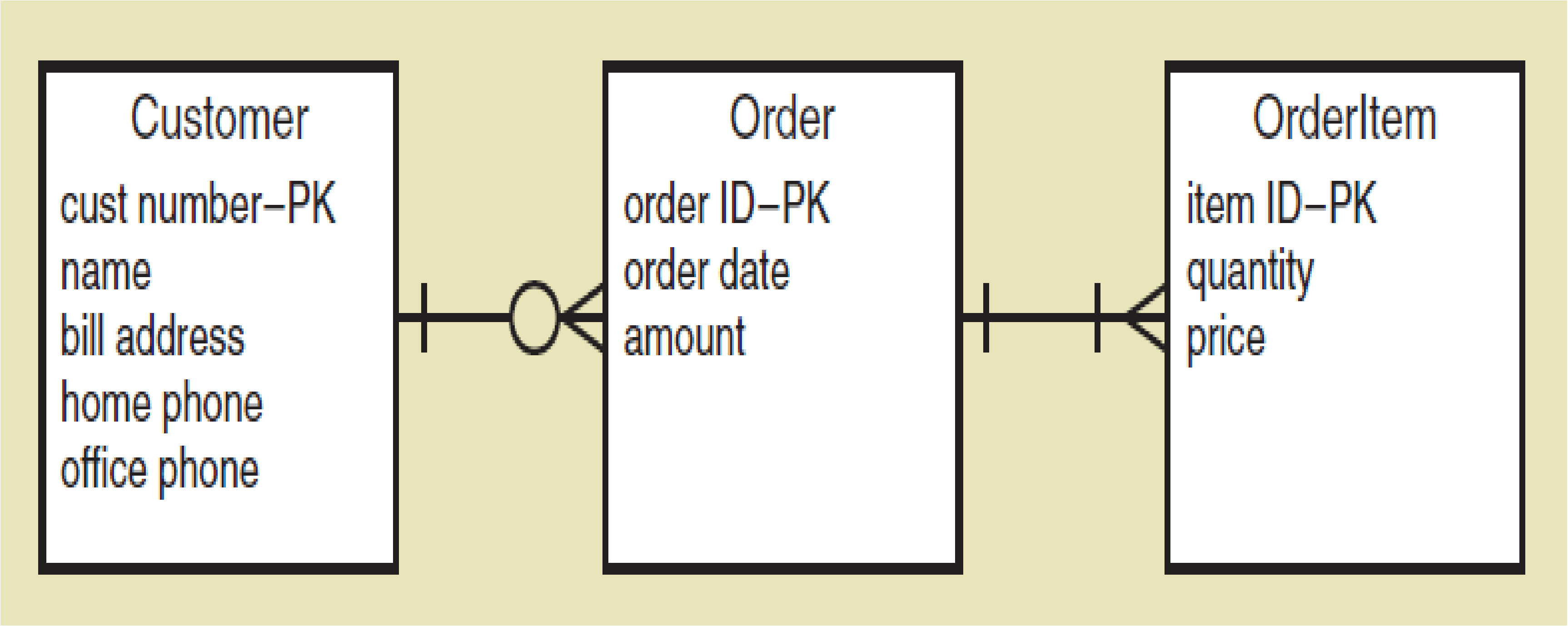

a Customer can place zero or more Orders
Customer
Order
an Order must be placed by exactly one Customer

ERD Cardinality Symbols

often called the crows feet notation

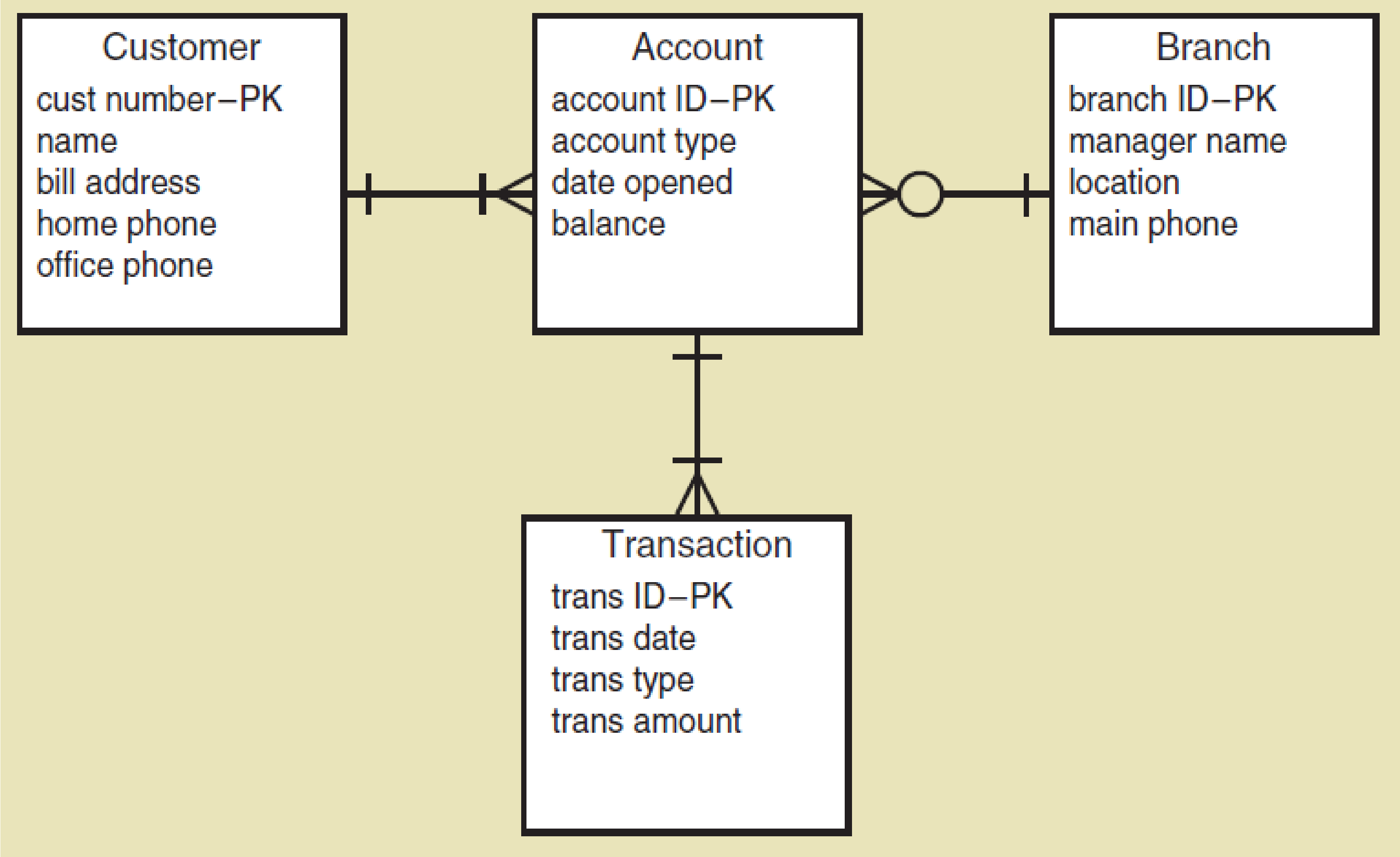


Expanded ERD with Attributes



- Note: This diagram matches the semantic net shown previously
- Also matches a domain model class diagram shown previously

An ERD for a Bank



Summary

- This lecture is the second of three that focuses on modelling functional requirements as a part of systems analysis
- “Things” in the problem domain are identified and modelled, called domain classes or data entities
- Two techniques for identifying domain classes/data entities are the brainstorming technique and the noun technique
- Domain classes have attributes and associations
- Associations are naturally occurring relationships among classes, and associations have minimum and maximum multiplicity

Summary

- The UML class diagram notation is used to create a domain model class diagram for a system. The domain model classes do not have methods because they are not yet software classes.
- There are actually three UML class diagram relationships: association relationships, generalization/specialization (inheritance) relationships, and whole part relationships
- Other class diagram concepts are abstract versus concrete classes, compound attributes, composition and aggregation, association classes, super classes and subclasses

Summary

- Entity-relationship diagrams (ERDs) show the same information as a domain model class diagram
- ERDs are preferred by database analysts and are widely used
- ERDs are not UML diagrams, and an association is called a relationship, multiplicity is called cardinality, and generalization/specialization (inheritance) and whole part relationships are usually not shown