Fox School of Business
TEMPLE UNIVERSITY®

Information Technology Audit & Cyber Security

Structural Modeling

Systems & Infrastructure
Lifecycle Management

OBJECTIVES

demonstrate the differences between object diagrams and class diagrams,

explain the three types of operations possible in class diagrams,

illustrate how associations are represented in class diagrams,

show how associative classes are drawn in class diagrams, and

show how generalization and aggregation are represented in class diagrams

INTRODUCTION

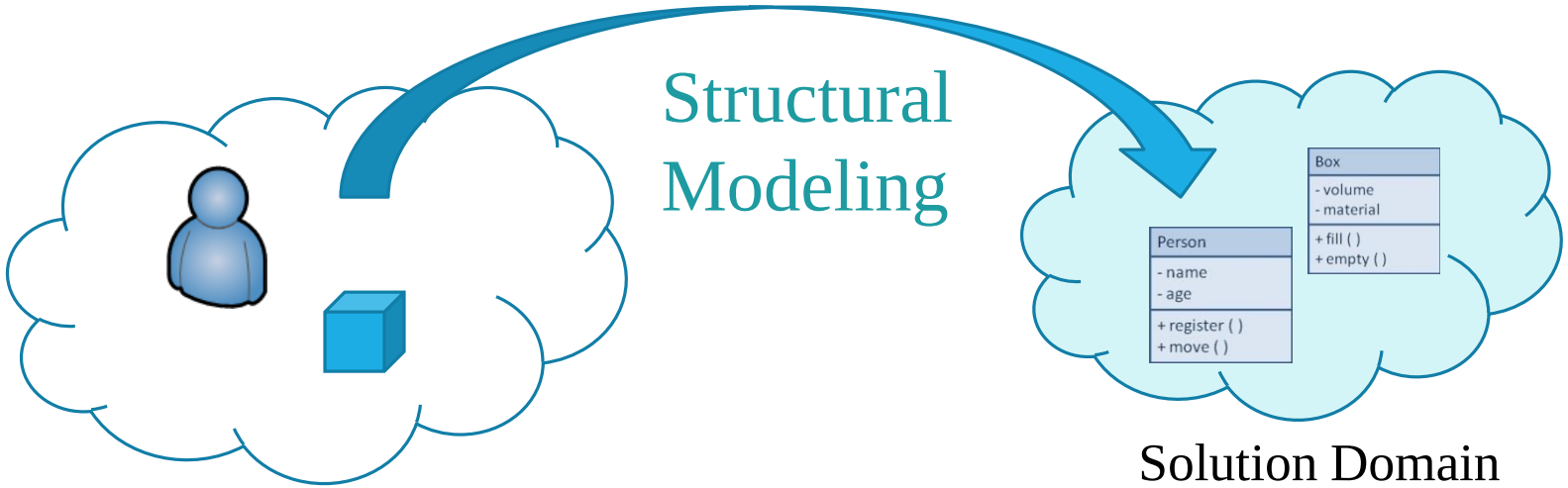
Functional models represent system behavior

Structural models represent system objects and their relationships:

- People
- Places
- Things

STRUCTURAL MODELS

Main goal: to discover the key data contained in the problem domain and to build a structural model of the objects



Problem Domain
(Analysis)

Solution Domain
(Design)

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 a database course you call them data entities

TWO TECHNIQUES TO IDENTIFY "THINGS"

Brainstorming Technique

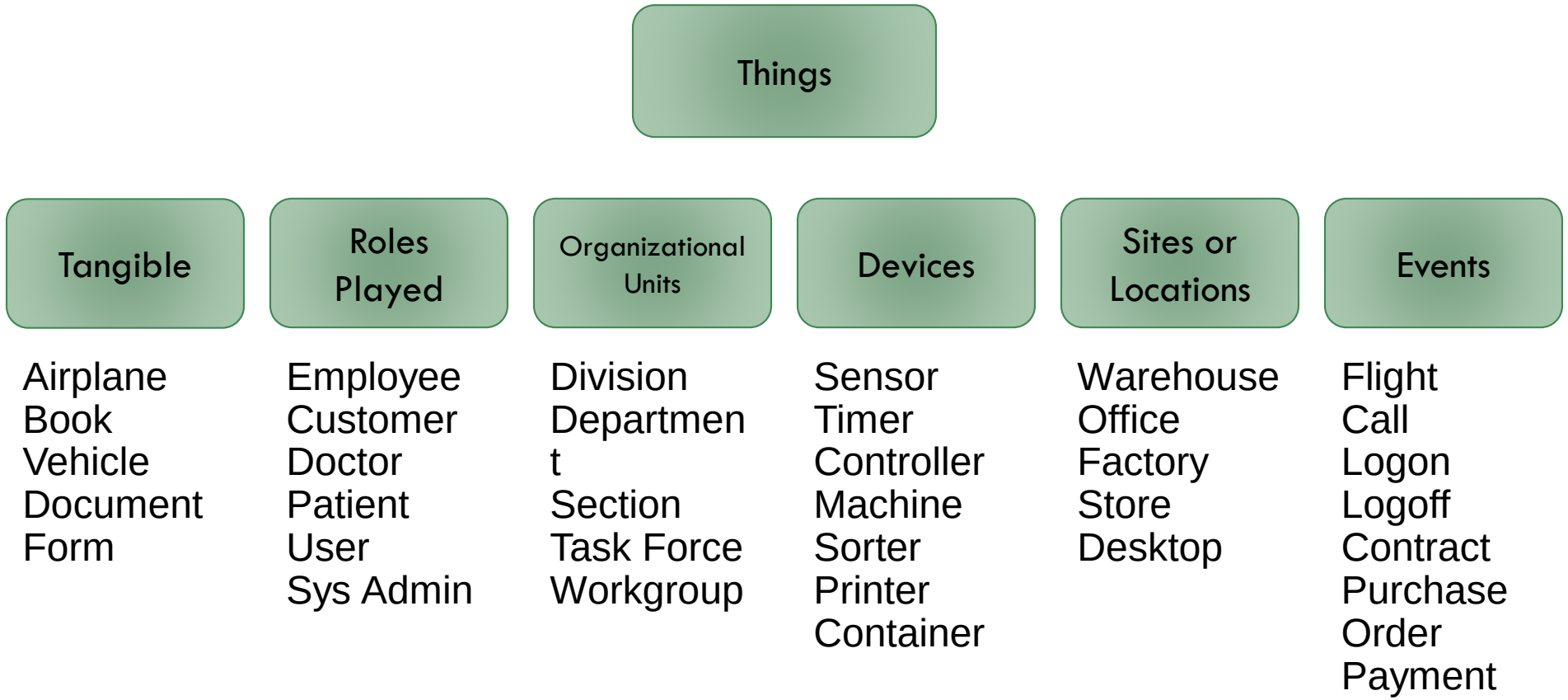
- Use a checklist of all of the usual types of things typically found and brainstorm to identify domain classes of each type

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…



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:
   1. Are there any tangible things you store information about?
   2. Are there any locations involved?
   3. 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

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

NOUN TECHNIQUE EXAMPLE

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.

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.

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?

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.

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.

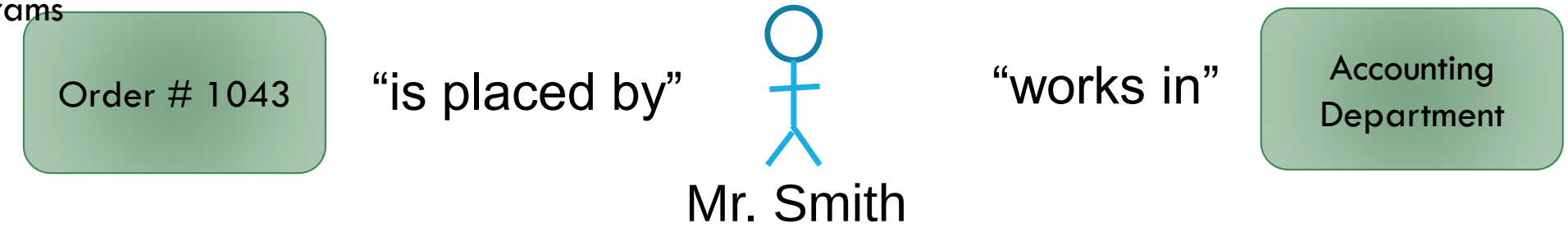
ATTRIBUTES AND VALUES

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 attributes:	Each customer has a value for each attribute:		
Customer ID	101	102	103
First Name	John	Dagny	Henry
Last Name	Galt	Taggart	Reardon
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 unit*

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

MIN AND MAX MULTIPLICITY

Associations have minimum and maximum constraints

- Minimum is zero
  - The association is optional
- 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 is zero or more-optional relationship

A particular order is placed by Mr. Smith. There can't be an order without stating the customer. → Multiplicity is one and only one-mandatory relationship

An order contains at least an item, but could have many items → Multiplicity 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

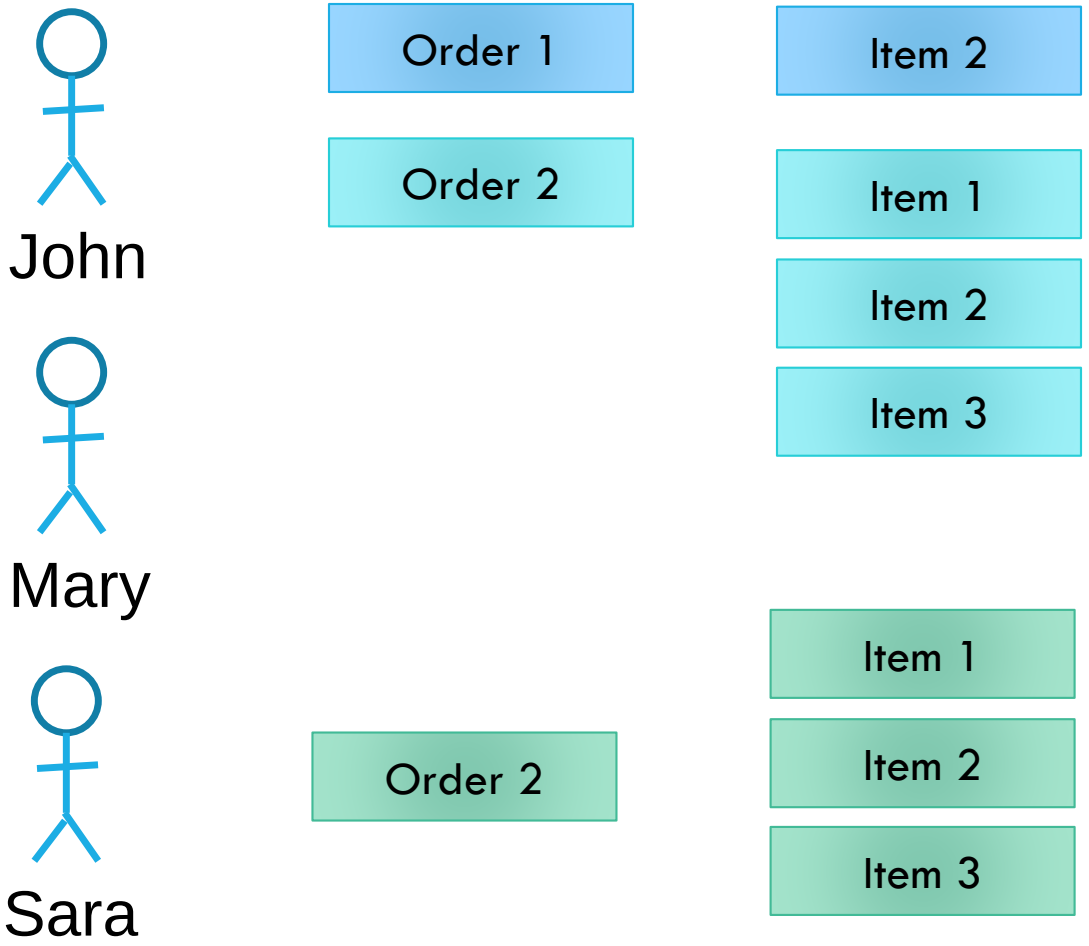
Ternary Association (three)

N-ary Association (between n)

SEMANTIC NET

Shows instances and how they are linked

- Example:



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

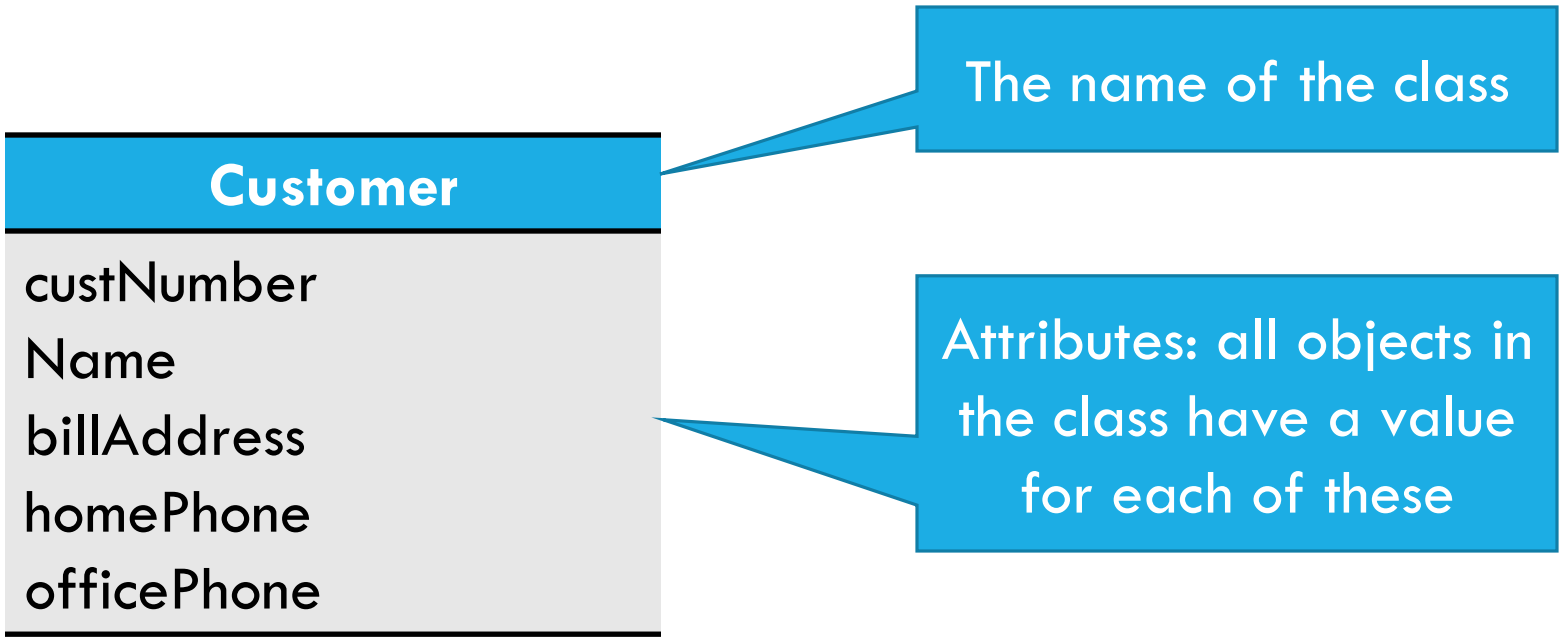
- A class diagram that only includes classes from the problem domain, not software classes so no methods

DOMAIN CLASS NOTATION

Domain class has no methods

Class name is always capitalized

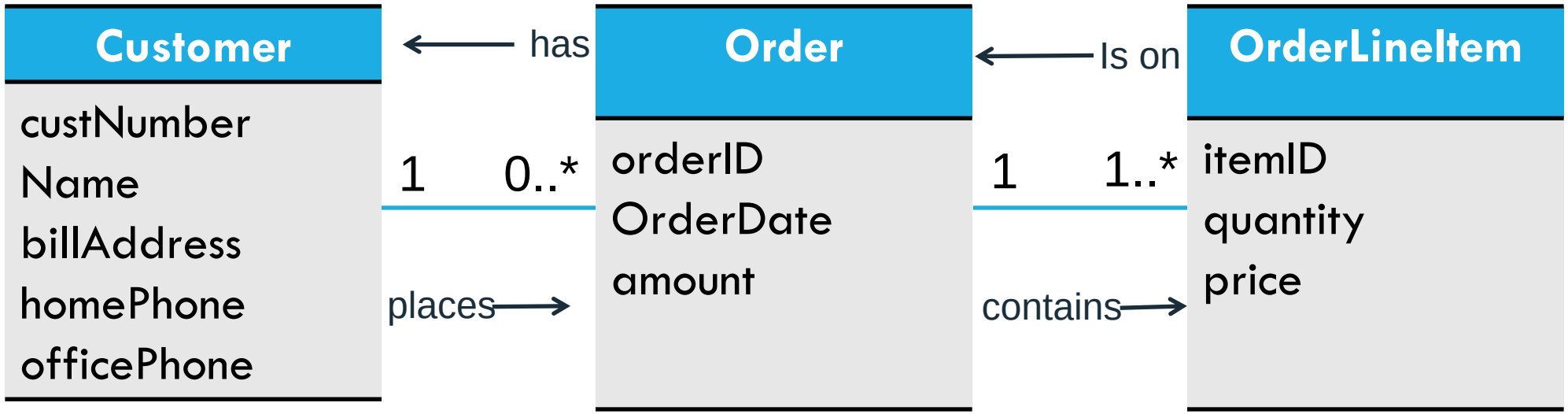
Attribute names are not capitalized and use camelback notation (words run together and second word is capitalized)



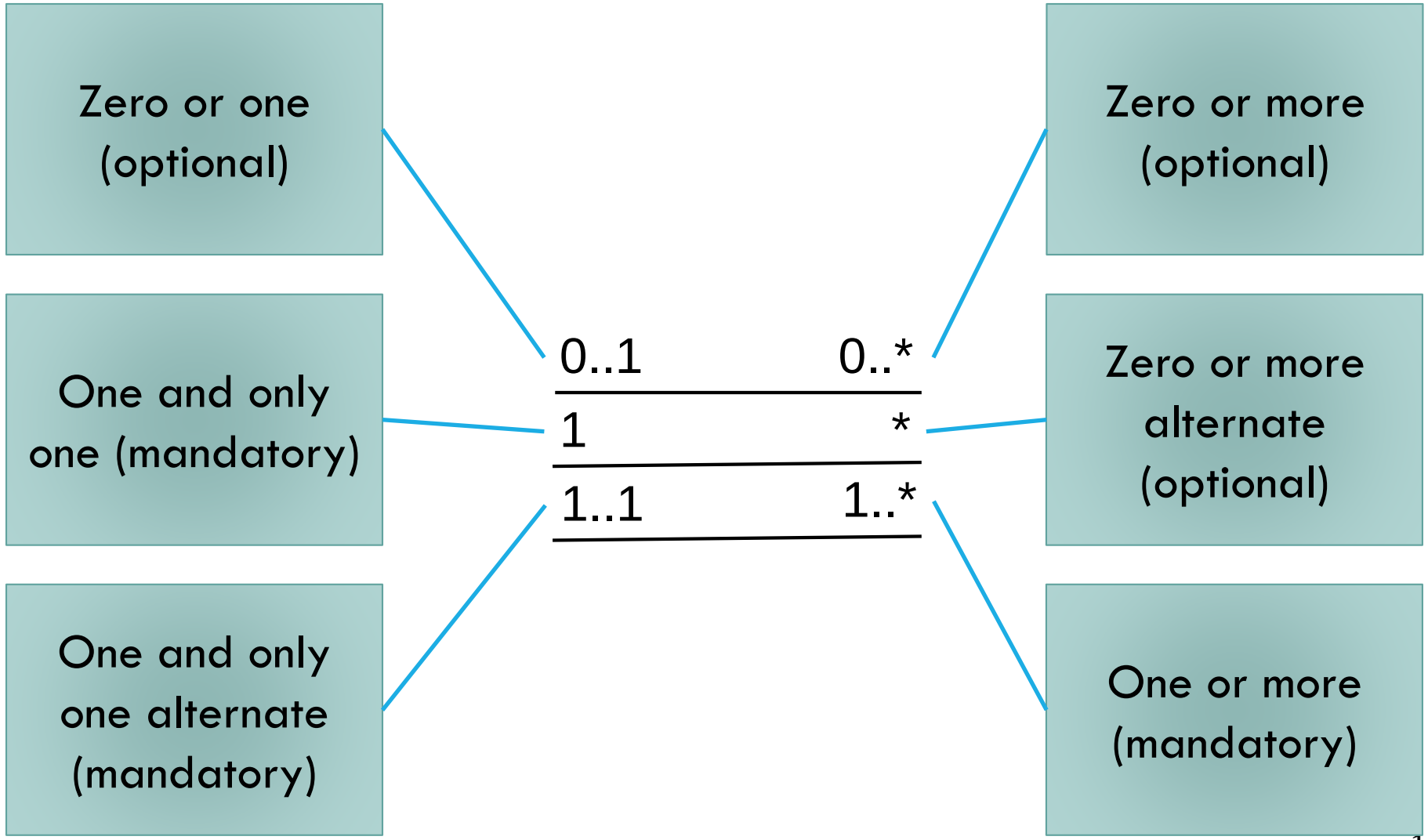
SIMPLE DOMAIN MODEL CLASS DIAGRAM

From the Semantic Net (shown previously)

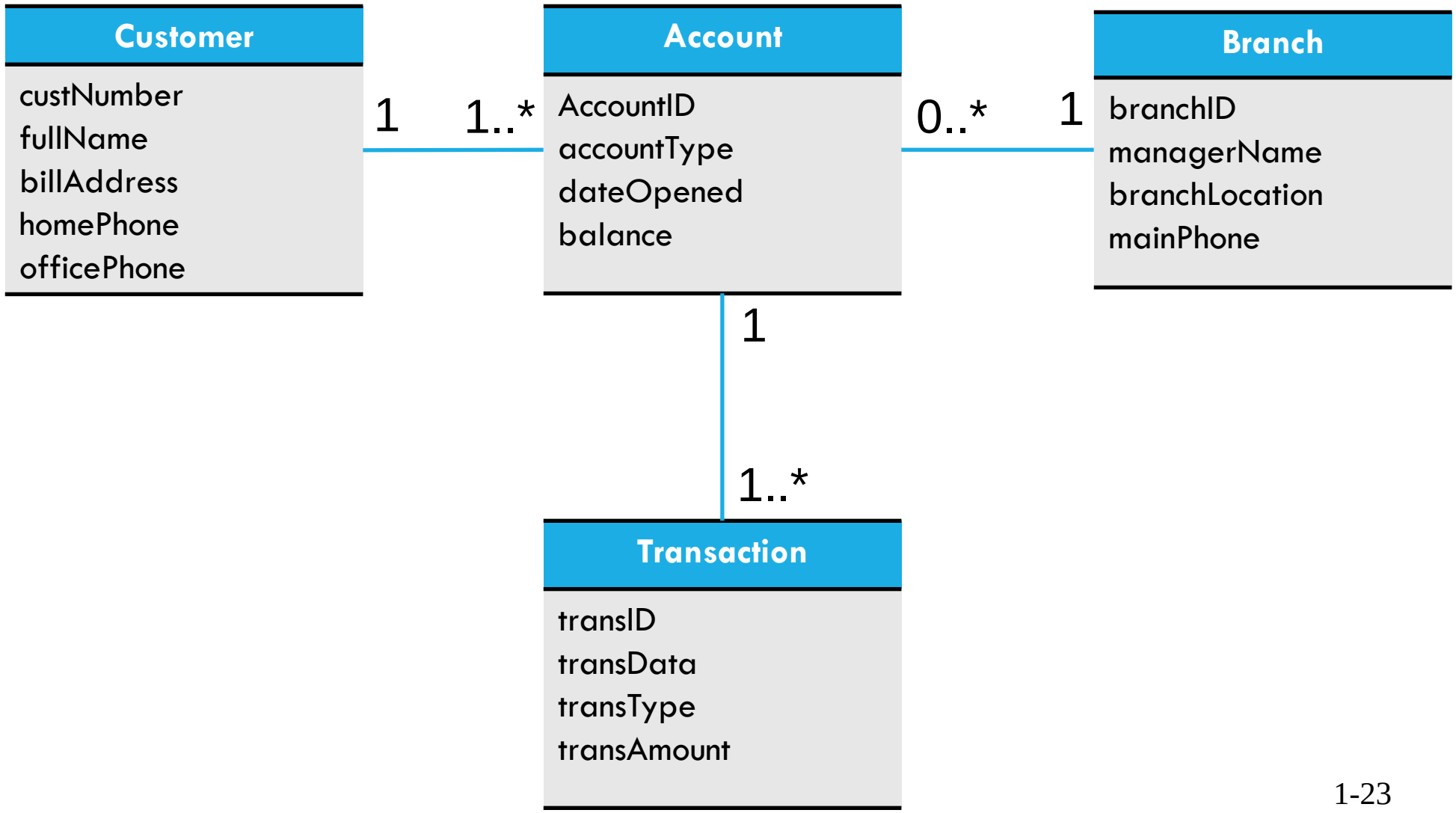
- A customer places zero or more orders
- An order is placed by exactly one customer
- An order consists of one or more order items
- An order item is part of exactly one order



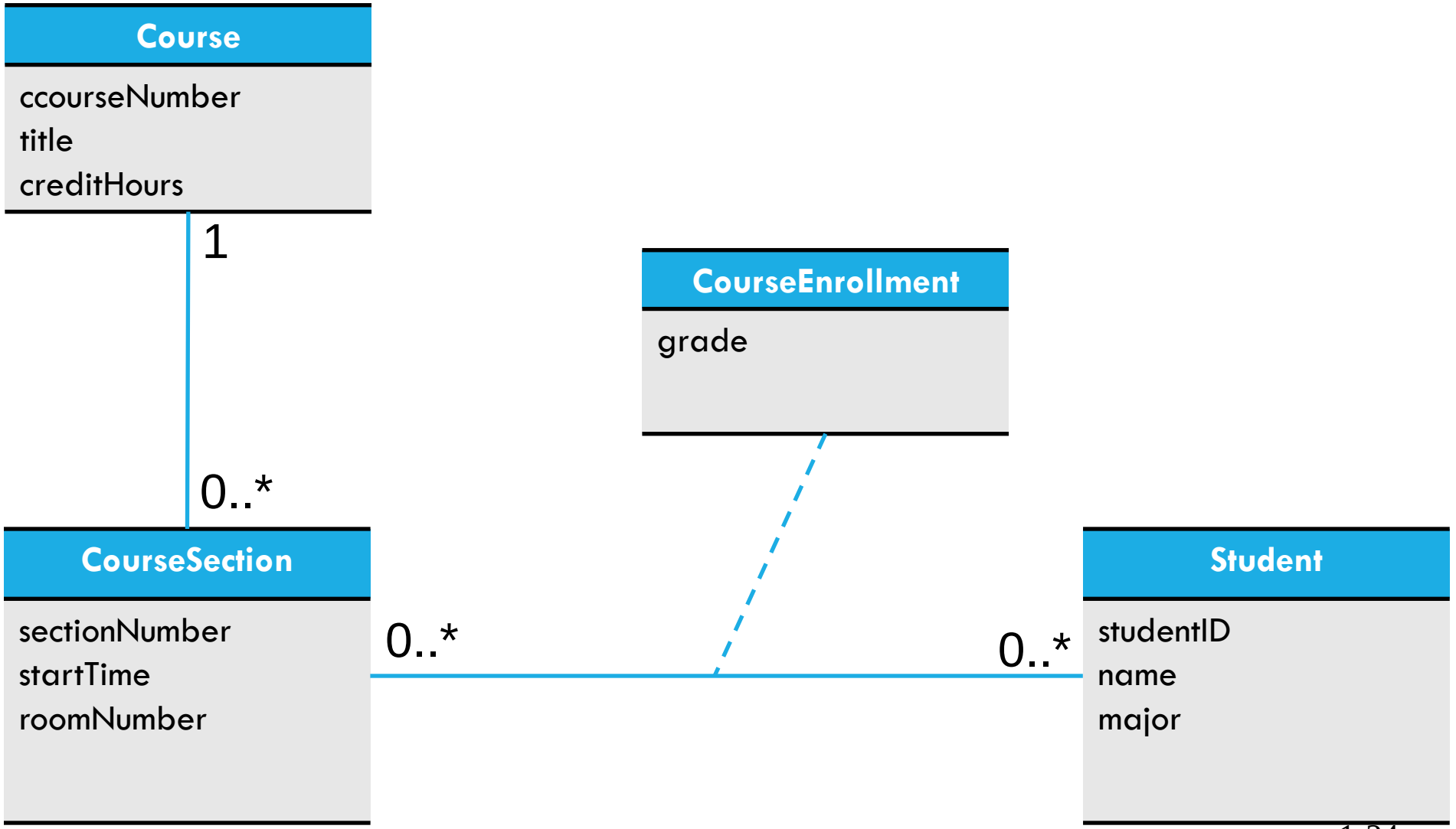
UML NOTATION FOR MULTIPLICITY



DOMAIN MODEL CLASS DIAGRAM



DOMAIN MODEL CLASS DIAGRAM



GENERALIZATION AND 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

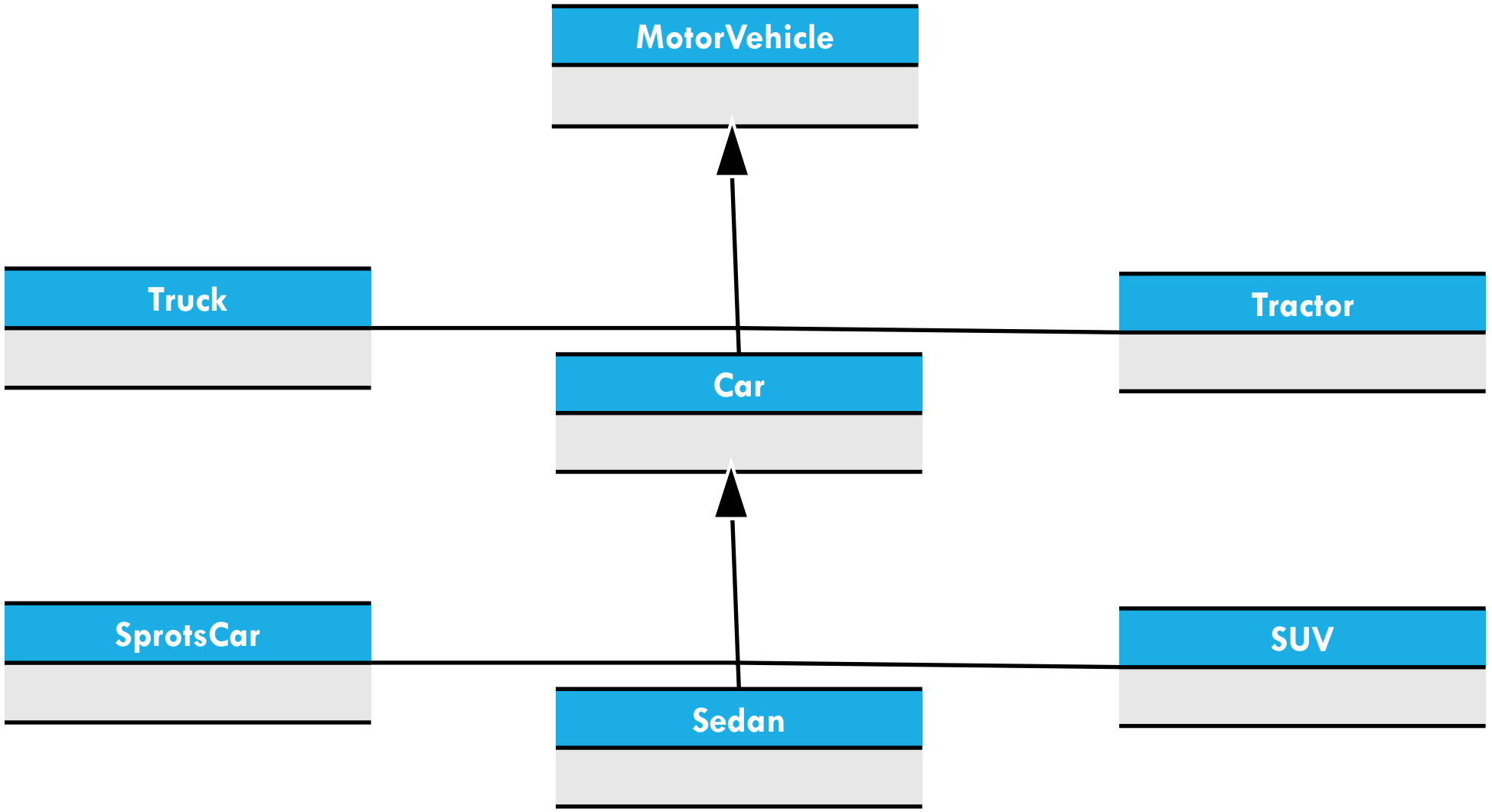
Subclass

- the subordinate or more specialized class in a generalization/specialization hierarchy

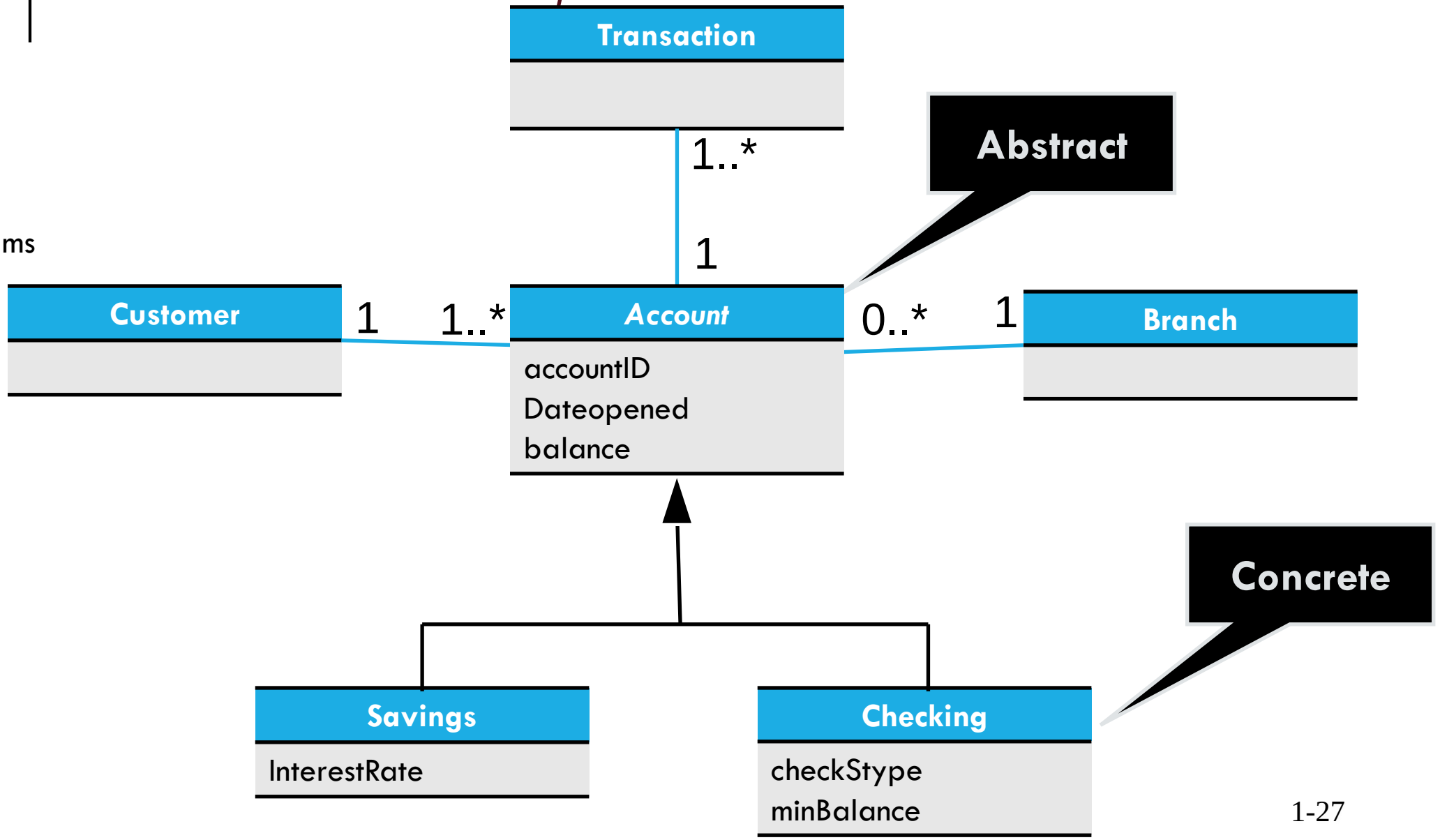
Inheritance

- the concept that subclasses inherit characteristics of the more general superclass

GENERALIZATION/SPECIALIZATION EXAMPLE



GENERALIZATION / SPECIALIZATION EXAMPLE

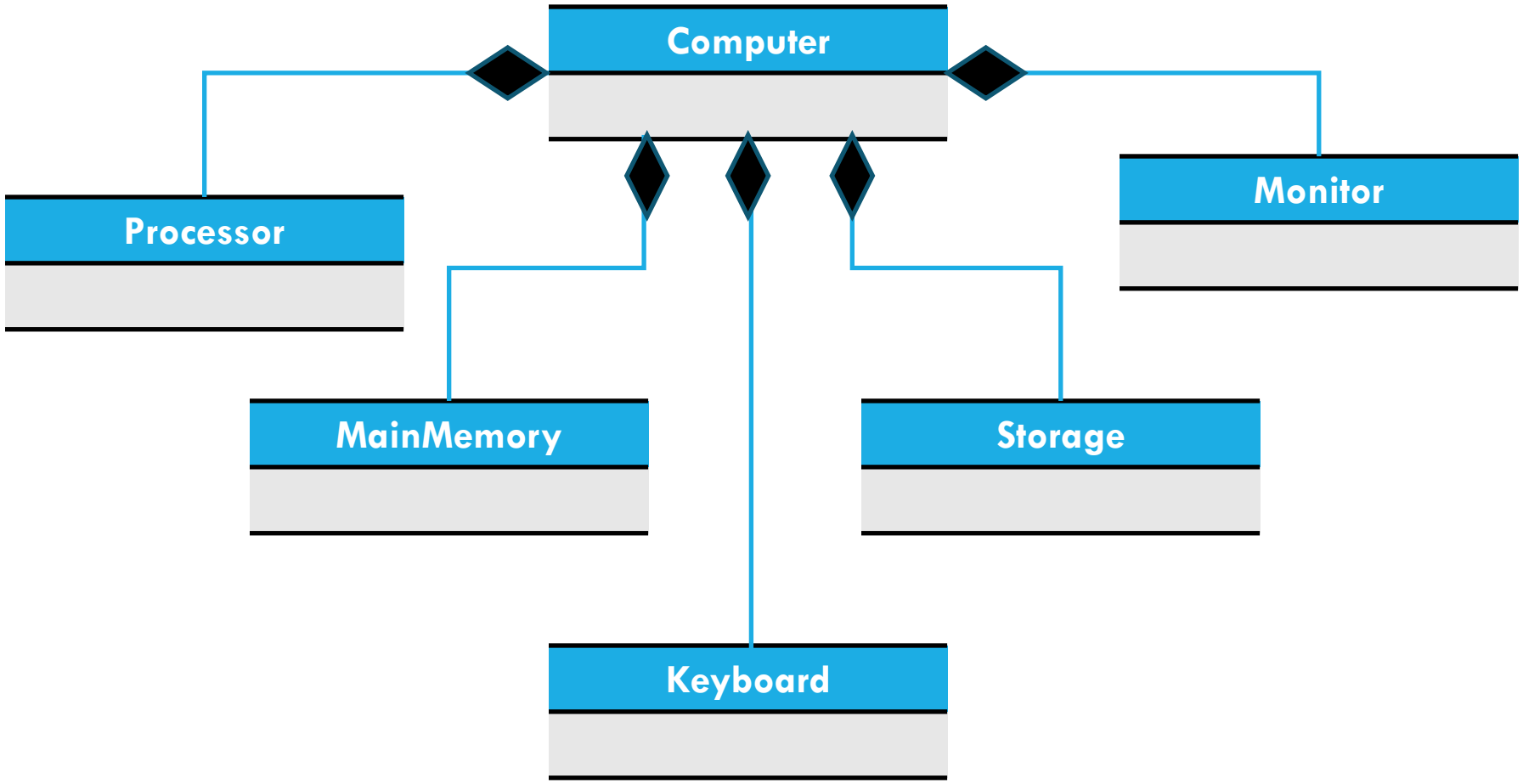


WHOLE PART RELATIONSHIPS

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
  - Chip has circuits

WHOLE PART RELATIONSHIPS EXAMPLE



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

Try not to confuse relationship with association

DESIGN CLASS DIAGRAMS

stereotype a way of categorizing a model element by its characteristics, indicated by guillemets (<< >>)

persistent class an class whose objects exist after a system is shut down (data remembered)

entity class a design identifier for a problem domain class (usually persistent)

boundary class or view class a class that exists on a system's automation boundary, such as an input window form or Web page

control class a class that mediates between boundary classes and entity classes, acting as a switchboard between the view layer and domain layer

data access class a class that is used to retrieve data from and send data to a database

Introduction

Identify Classes

Domain Class Model

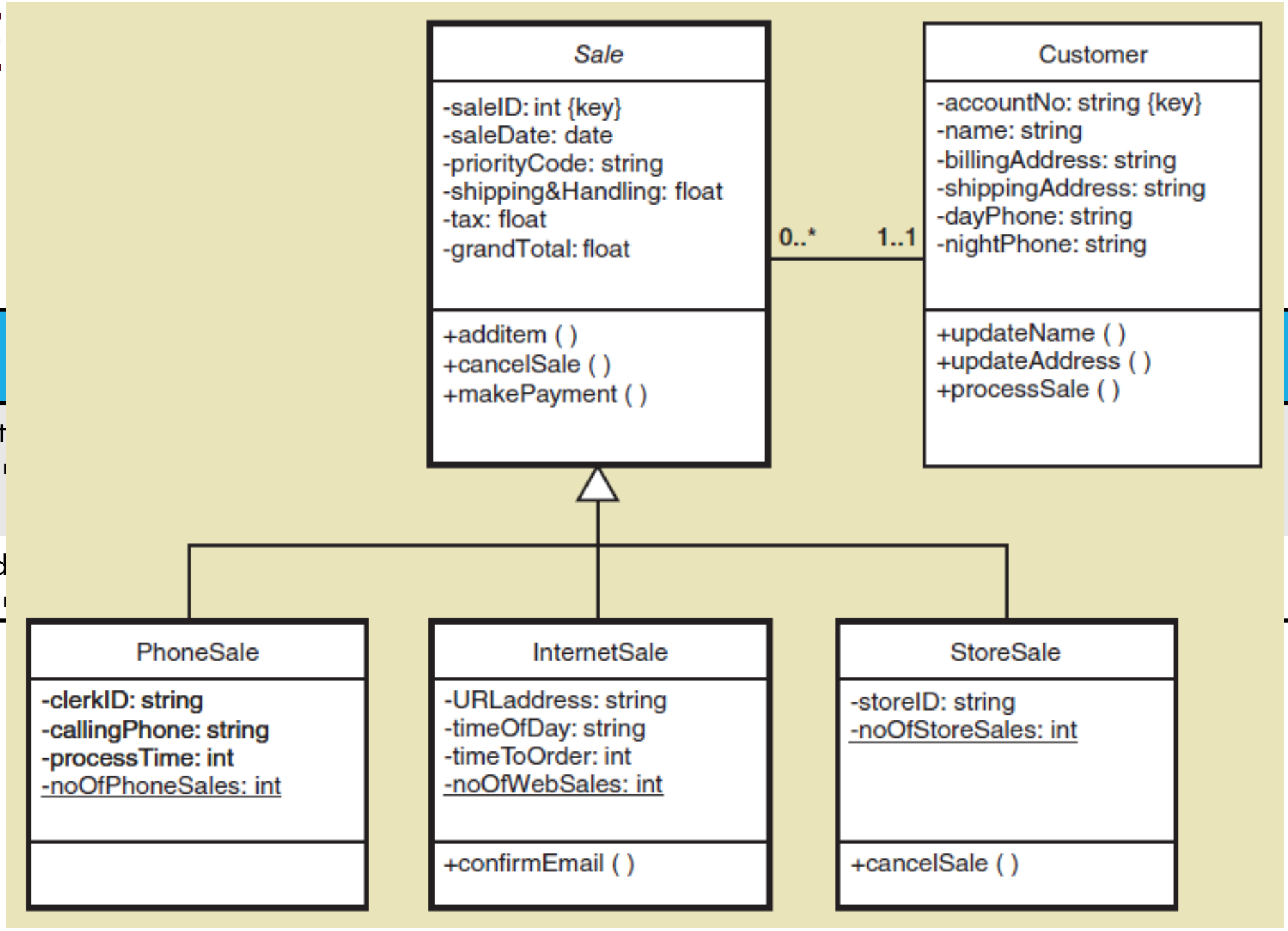
Relationships

Design Class Diagrams

CRC Cards

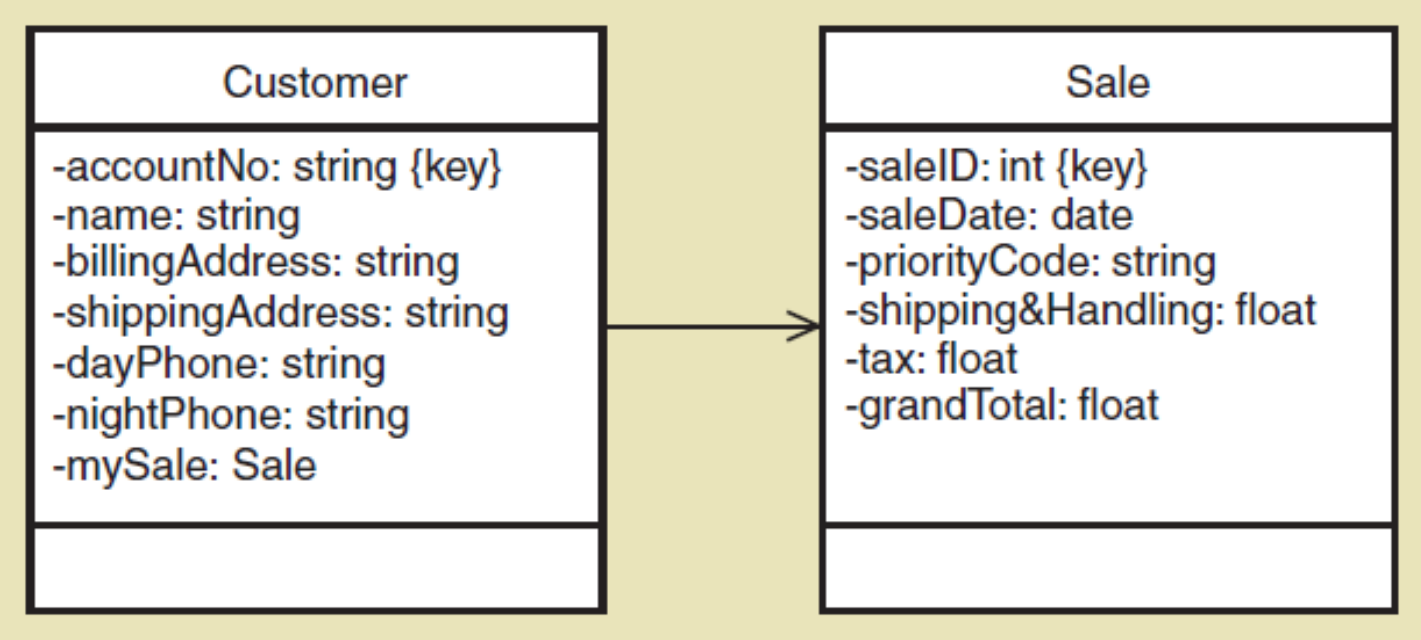
Design Goals

DE



NAVIGATION VISIBILITY

- The ability of one object to view and interact with another object
- Accomplished by adding an object reference variable to a class.
- Shown as an arrow head on the association line—customer can find and interact with sale because it has mySale reference variable



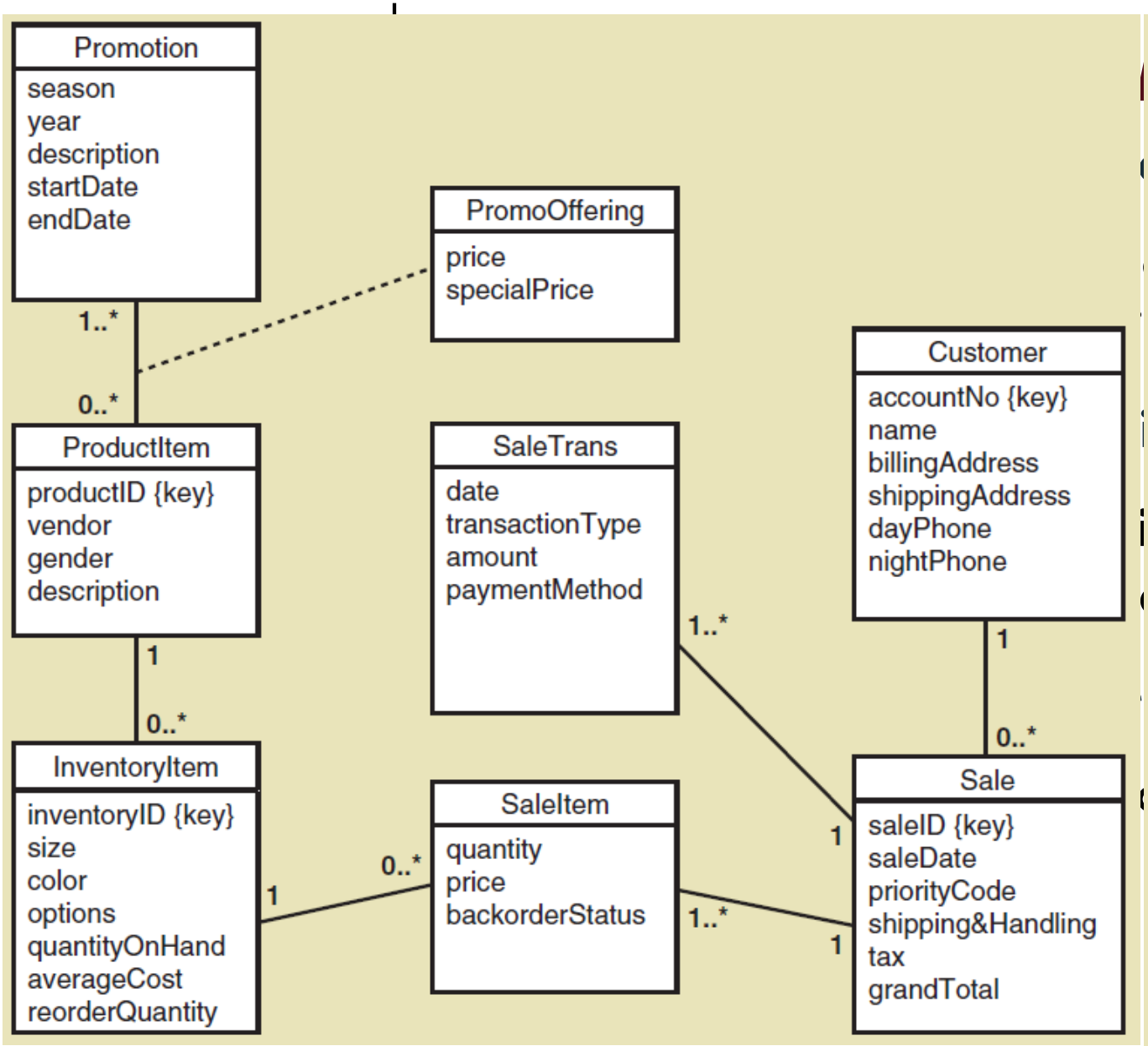
NAVIGATION VISIBILITY GUIDELINES

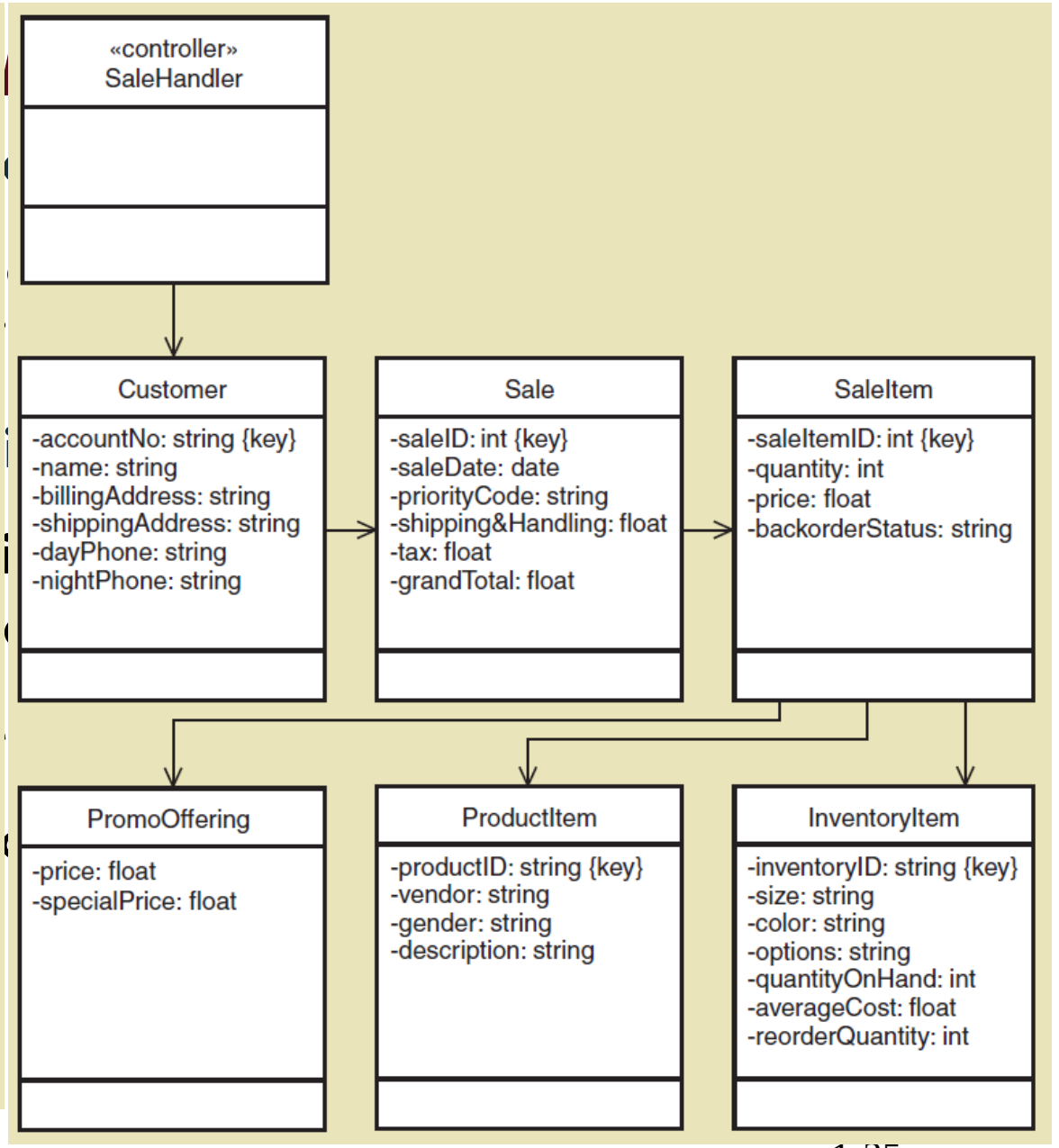
One-to-many associations that indicate a superior/subordinate relationship are usually navigated from the superior to the subordinate

Mandatory associations, in which objects in one class can't exist without objects of another class, are usually navigated from the more independent class to the dependent

When an object needs information from another object, a navigation arrow might be required

Navigation arrows may be bidirectional.


Promotion
season
year
description
startDate
endDate
PromoOffering
price
specialPrice
1..*
0..*
ProductItem
productID {key}
vendor
gender
description
SaleTrans
date
transactionType
amount
paymentMethod
Customer
accountNo {key}
name
billingAddress
shippingAddress
dayPhone
nightPhone
1
0..*
1..*
1
0..*
InventoryItem
inventoryID {key}
size
color
options
quantityOnHand
averageCost
reorderQuantity
SaleItem
quantity
price
backorderStatus
Sale
saleID {key}
saleDate
priorityCode
shipping&Handling
tax
grandTotal


«controller»
SaleHandler
Customer
-accountNo: string {key}
-name: string
-billingAddress: string
-shippingAddress: string
-dayPhone: string
-nightPhone: string
Sale
-saleID: int {key}
-saleDate: date
-priorityCode: string
-shipping&Handling: float
-tax: float
-grandTotal: float
SaleItem
-saleItemID: int {key}
-quantity: int
-price: float
-backorderStatus: string
PromoOffering
-price: float
-specialPrice: float
ProductItem
-productID: string {key}
-vendor: string
-gender: string
-description: string
InventoryItem
-inventoryID: string {key}
-size: string
-color: string
-options: string
-quantityOnHand: int
-averageCost: float
-reorderQuantity: int

DESIGNING WITH CRC CARDS

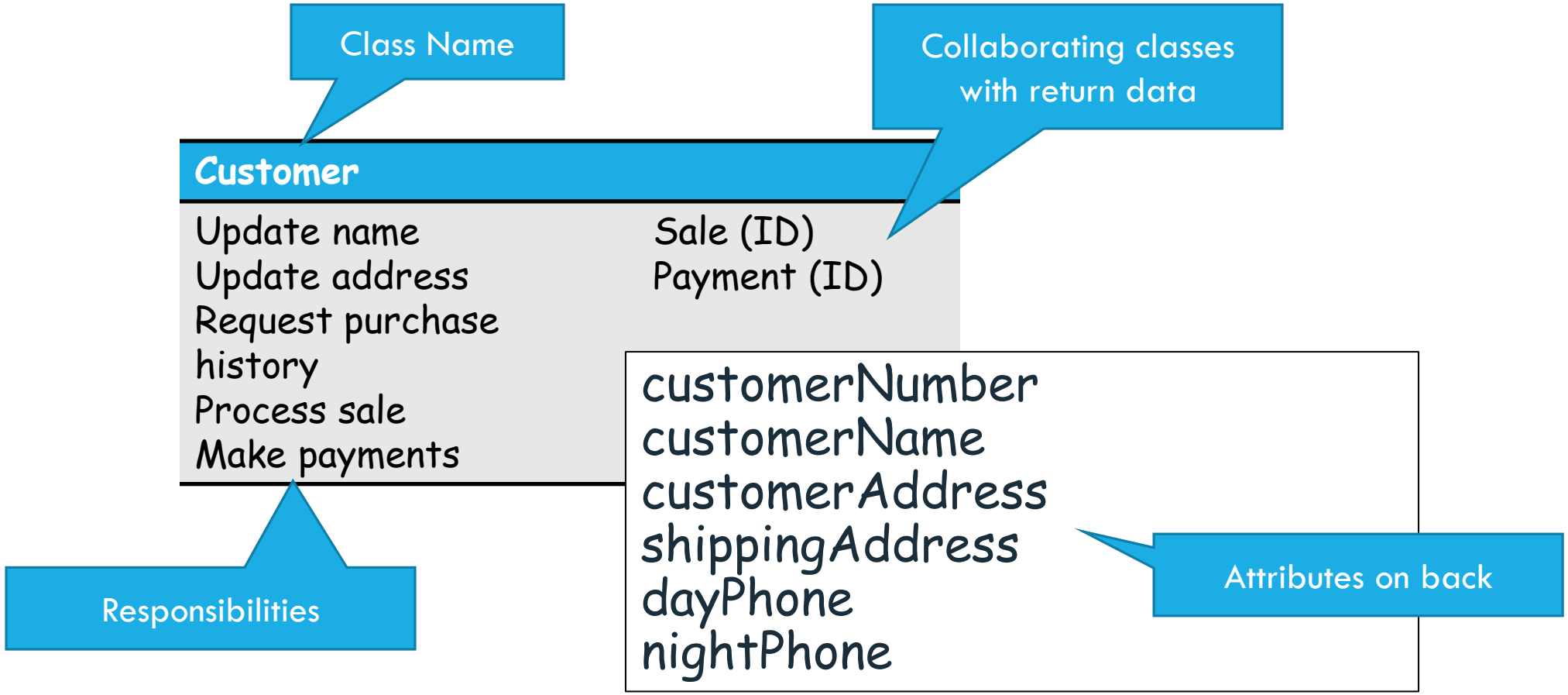
CRC Cards—Classes, Responsibilities, Collaboration Cards

OO design is about assigning Responsibilities to Classes for how they Collaborate to accomplish a use case

Usually a manual process done in a brainstorming session

- 3 X 5 note cards
- One card per class
- Front has responsibilities and collaborations
- Back has attributes needed

CRC CARD EXAMPLE



Introduction

Identify Classes

Do

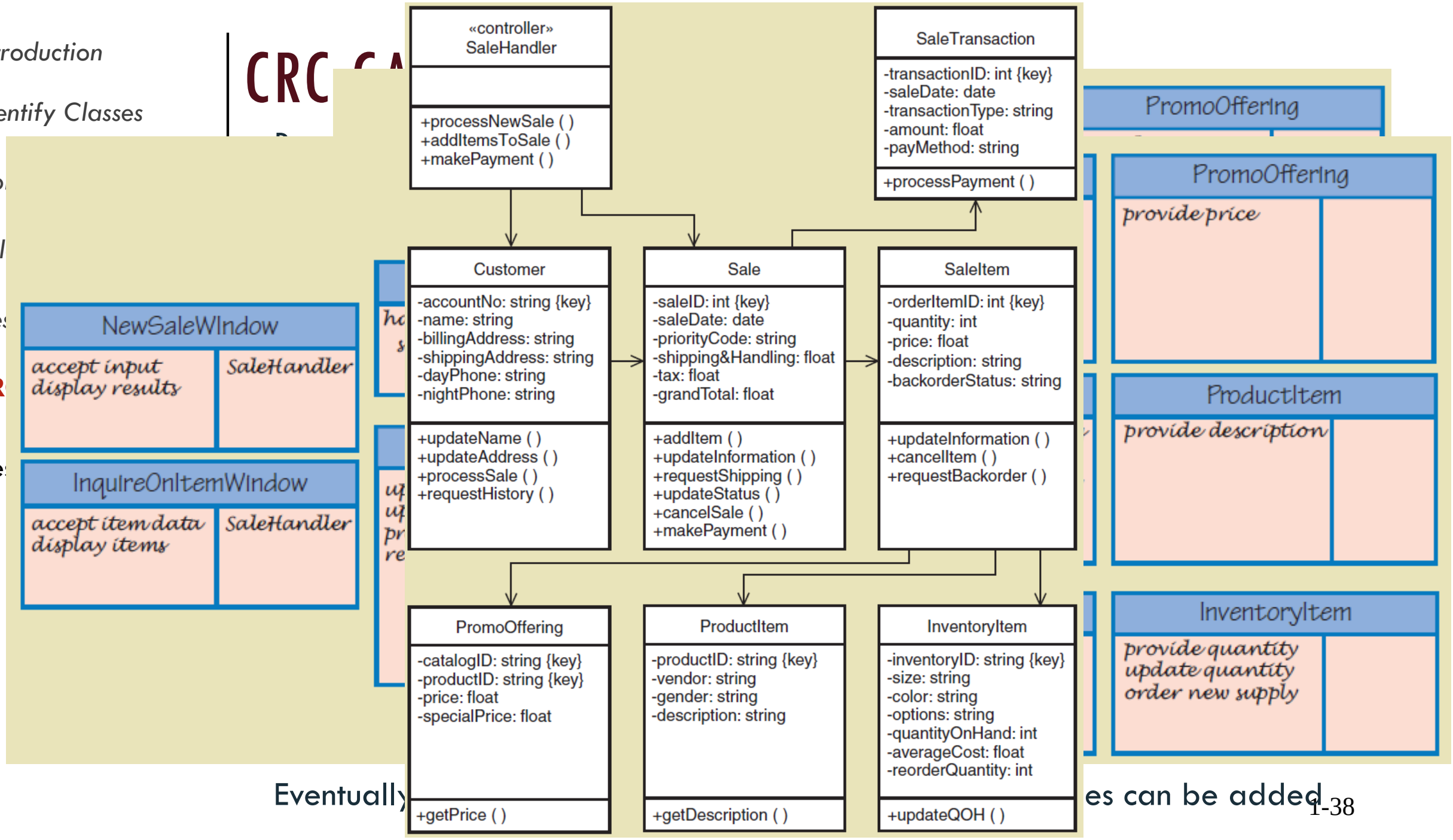
Rel

Des

CR

Des

CRC CA



Eventually

es can be added

COUPLING

- A quantitative measure of how closely related classes are linked (tightly or loosely coupled)
- Two classes are tightly coupled of there are lots of associations with another class
- Two classes are tightly coupled if there are lots of messages to another class
- It is best to have classes that are **loosely coupled**
- If deciding between two alternative designs, choose the one where overall coupling is less

COHESION

- A quantitative measure of the focus or unity of purpose within a single class (high or low cohesiveness
- One class has high cohesiveness if all of its responsibilities are consistent and make sense for purpose of the class (a customer carries out responsibilities that naturally apply to customers)
- One class has low cohesiveness if its responsibilities are broad or makeshift
- It is best to have classes that are **highly cohesive**
- If deciding between two alternative designs, choose the one where overall cohesiveness is high

PROTECTION FROM VARIATIONS

- A design principle that states parts of a system unlikely to change are separated (protected) from those that will surely change
- Separate user interface forms and pages that are likely to change from application logic
- Put database connection and SQL logic that is likely to change in a separate classes from application logic
- Use adaptor classes that are likely to change when interfacing with other systems
- If deciding between two alternative designs, choose the one where there is protection from variations

INDIRECTION

- A design principle that states an intermediate class is placed between two classes to decouple them but still link them
- A controller class between UI classes and problem domain classes is an example
- Supports low coupling
- Indirection is used to support security by directing messages to an intermediate class as in a firewall
- If deciding between two alternative designs, choose the one where indirection reduces coupling or provides greater security

OBJECT RESPONSIBILITY

- A design principle that states objects are responsible for carrying out system processing
- A fundamental assumption of OO design and programming
- Responsibilities include "knowing" and "doing"
- Objects know about other objects (associations) and they know about their attribute values. Objects know how to carry out methods, do what they are asked to do.
- Note that CRC cards and the design in the next chapter involve assigning responsibilities to classes to carry out a use case.
- If deciding between two alternative designs, choose the one where objects are assigned responsibilities to collaborate to complete tasks (don't think procedurally).