CH6. Structural Modeling

School of Computer Science
Prof. Euijong Lee

Before Structural Modeling

- **Realization of Use Case Diagram**
  - to elaborating use cases
  - to connecting use case to objects in the system

> The responsibilities to perform the action described in the use case are allocated to collaborating objects that implement the functionality

- **Realized by Sequence Diagram**
  - a sequence diagram per a use-case
  - helpful to identify covered objects (or classes), omittable classes, and some attributes and/or operations

Objectives

- **Understand the rules and style guidelines for creating**
  - CRC (Class-Responsibility-Collaboration) Cards
  - Class Diagram
  - Object Diagram
- **Understand the process used to CRC cards and class diagram**
- **Be able to create CRC cards, class diagram and object diagram**
- **Understand the relationship between the Structural and Use-case models**

Introduction

- **Functional model**
  - **to represent how the business system will behave as well as what the system should do**
- **Structural model**
  - **representing the objects that are created and used by a business system**
- **Structural model is evolving over time**
  - **In analysis phase**
    - conceptual model
    - shows the logical organization of the objects without indicating how the objects are stored, created, or manipulated
  - **In design phase**
    - design model
    - reflects how the objects will be organized in database and files.
- **The structure of data used in the system is represented through**
  - **CRD cards, class diagrams, and object diagrams.**

Purpose of Structural Models

- Reduce the "semantic gap" between the real world and the world of software
- Create a vocabulary for analysts and users
- Represent things, ideas, and concepts of importance in the application domain

Classes, Attributes, and Operations (1/2)

❖ Classes

- **Templates for creating instances or object in the application domain**
  - **Concrete** : describe the application domain classes
  - **Abstract** : defined by useful abstraction

Abstract Class	**Person**
Concrete Class	**Customer, Employee, Manager**

- **Typical examples :**
  - Application domain class
  - User interface class
  - Data structure class
  - File structure class
  - Operating environment class
  - Document classes
  - and so on.

Classes, Attributes, and Operations (2/2)

❖ Attributes

지속적으로 저장하고 컨트롤

- Units of information relevant to the description of the class
- Only attributes important to the task should be included
- Attributes "hair_color" in class "person" ??

knowing resposibility

❖ Operations

인스턴스화된 어떤 객체가 수행할 수 있는 어떤 작업

- Action that instances/objects can take
- Focus on relevant problem-specific operations (at this point)

doing responsibility

❖ Consideration

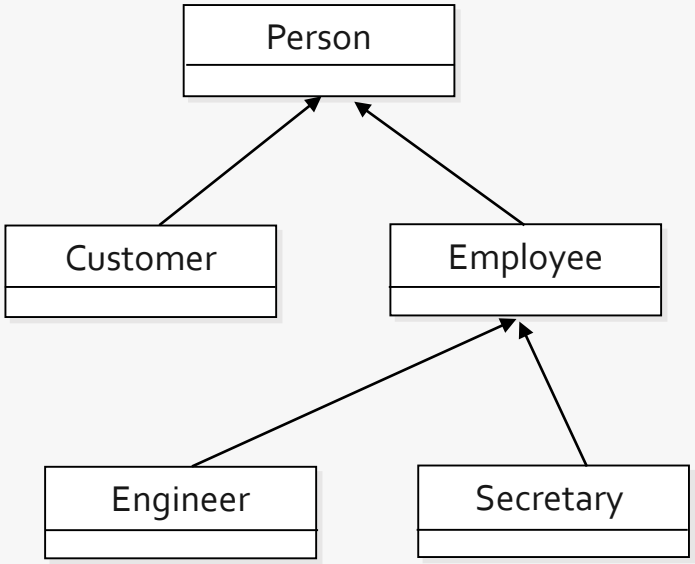
- Avoid creating the implementation-specific classes, attributes, and operations

Relationships (1/2)

❖ **3 basic relationships : generalization, aggregation, and association**

❖ **Generalization**

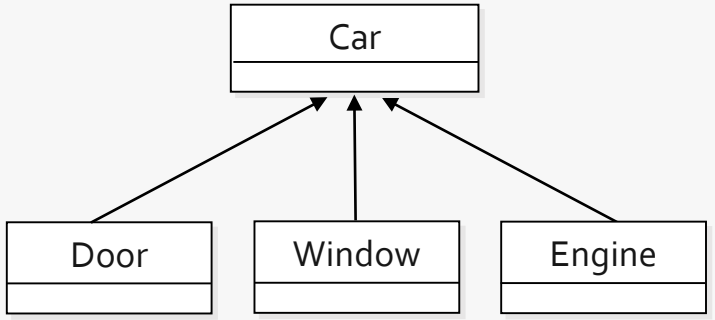
- enables inheritance of attributes and operations from other classes
- **a-kind-of** relationship
- relationship between superclass and subclasses
- Generalization hierarchy
- flip side : specialization
- apply the principle of substitutability



Relationships (2/2)

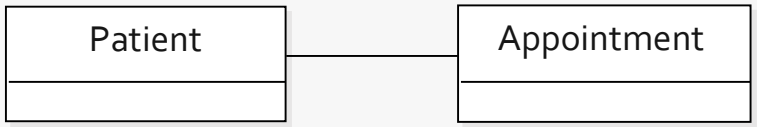
❖ Aggregation

- composition relationship
- **a-part-of or has-parts** relationship
- flip side : decomposition



❖ Association

- Miscellaneous relationships between classes
- not fit neatly into generalization or aggregation



Your Turn – Activity

- What classes, attributes, and operations that would be required to describe the process of borrowing software from PC Lab. ?

CRC Card

- **Class-Responsibility-Collaboration Card**
- **Used to**
  - document the responsibilities and collaborations of a class
  - capture all relevant information associated with a class
- **Responsibilities of a class**
  - Knowing responsibilities
    - an instance of a class knows the values of its attributes and its relationship
  - Doing responsibilities
    - an instance of a class can execute its operations
- **Collaboration of a class**
  - Objects working together to service a request
  - The set of classes involved in a use case
  - in term of Client, Server, and Contract

A CRC Card – Front side

- Captures and describes the essential elements of a class

Front:

Class Name: Old Patient	**ID:** 3	**Type:** Concrete, Domain
Description: An individual who needs to receive or has received medical attention		**Associated Use Cases:** 2

Responsibilities	**Collaborators**
Make appointment	Appointment
Calculate last visit	
Change status	
Provide medical history	Medical history

(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Back of A CRC Card

- Contains the attributes and relationship of the class in back side

Back:

Attributes:

Amount (double)

Insurance carrier (text)

Relationships:

Generalization (a-kind-of): Person

Aggregation (has-parts): Medical History

Other Associations: Appointment

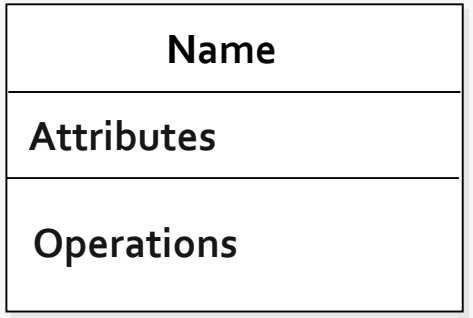
(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Back of A CRC Card

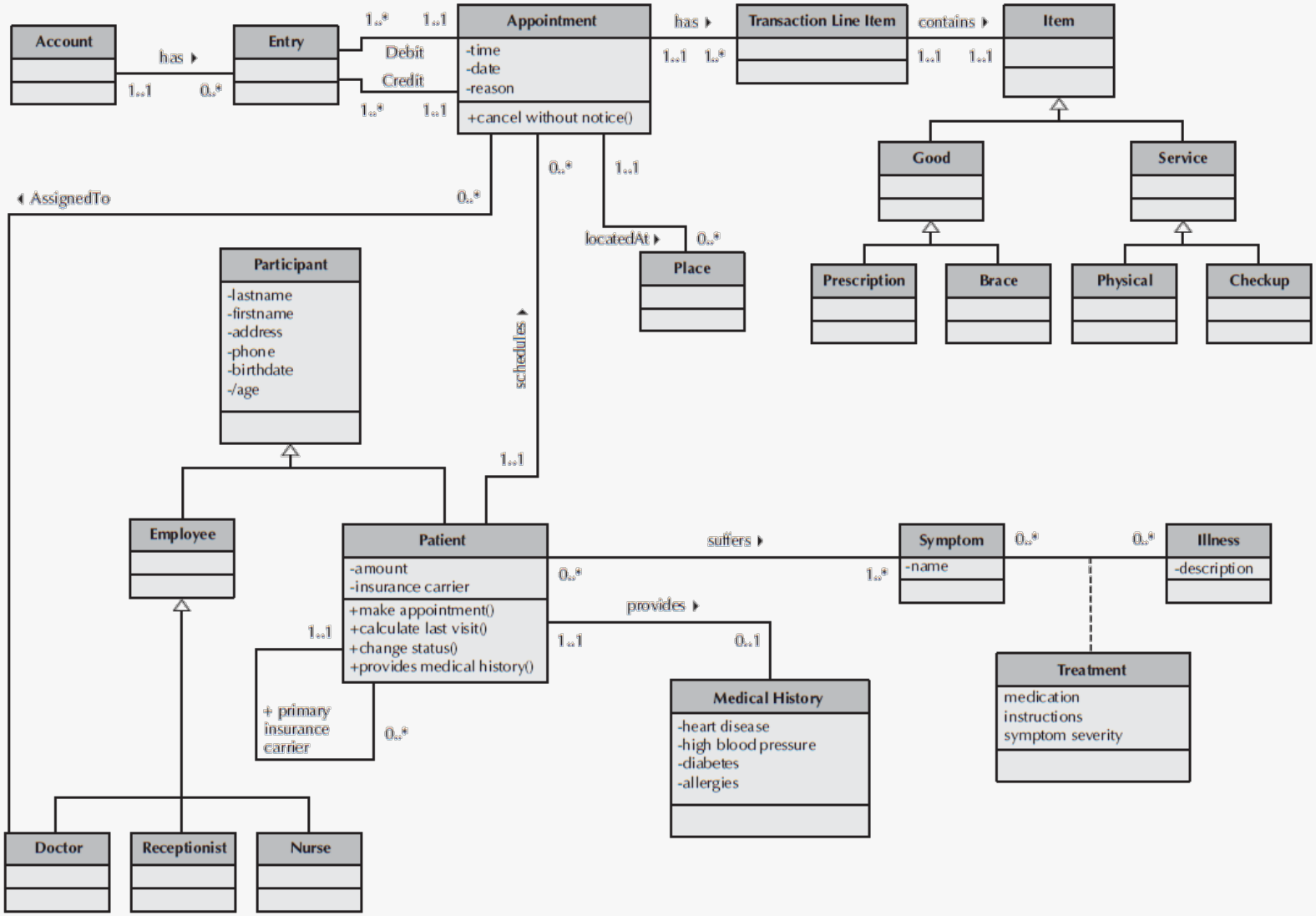
- **Making CRC Card using Software**
  - **CRC Maker in online**
    https://echeung.me/crcmaker/
  - **CRC Maker for Windows, Mac, and Linux**
    https://www.edrawsoft.com/crc-card-software.html

Class Diagram

- **A static model**
- **Shows the classes and the relationships among classes that remain constancy in the system over time**
- **Elements of a class diagram**
  - **Class**
    - class name, attributes, and operations
  - **Relationship**
    - association, generalization, and aggregation



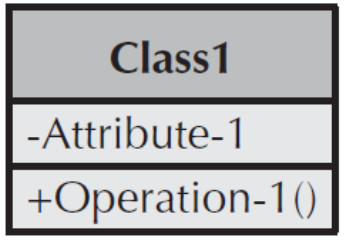
Example Class Diagram



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Class Diagram Syntax (1/3)

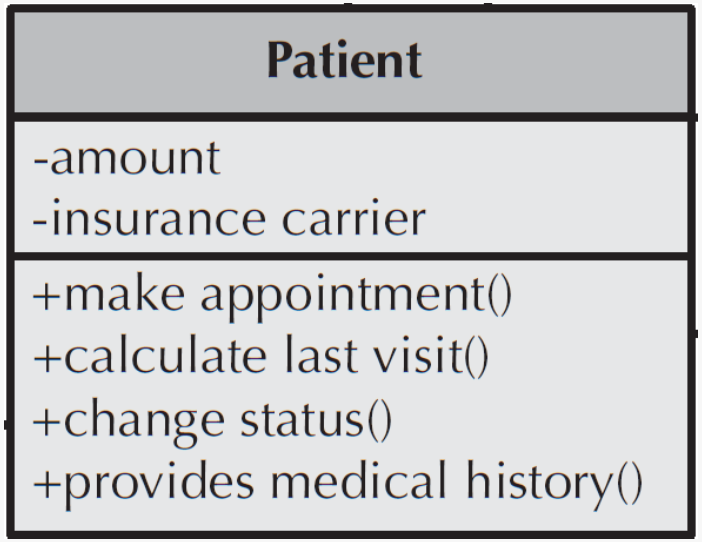
❖ **Class and Attributes**

A class: • Represents a kind of person, place, or thing about which the system will need to capture and store information. • Has a name typed in bold and centered in its top compartment. • Has a list of attributes in its middle compartment. • Has a list of operations in its bottom compartment. • Does not explicitly show operations that are available to all classes.	
An attribute: • Represents properties that describe the state of an object. • Can be derived from other attributes, shown by placing a slash before the attribute's name.	attribute name /derived attribute name

Class Diagram Syntax (2/3)

- **Operation and Association (=Relationship)**

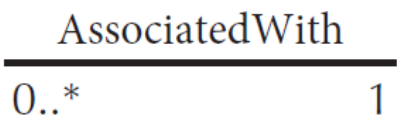
An operation: • Represents the actions or functions that a class can perform. • Can be classified as a constructor, query, or update operation. • Includes parentheses that may contain parameters or information needed to perform the operation.	operation name ()



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

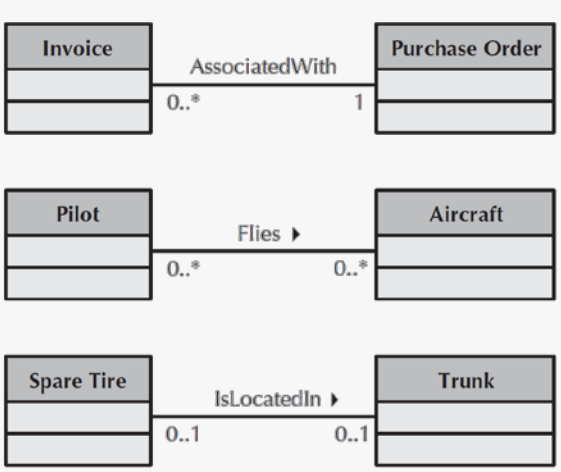
Class Diagram Syntax (2/3)

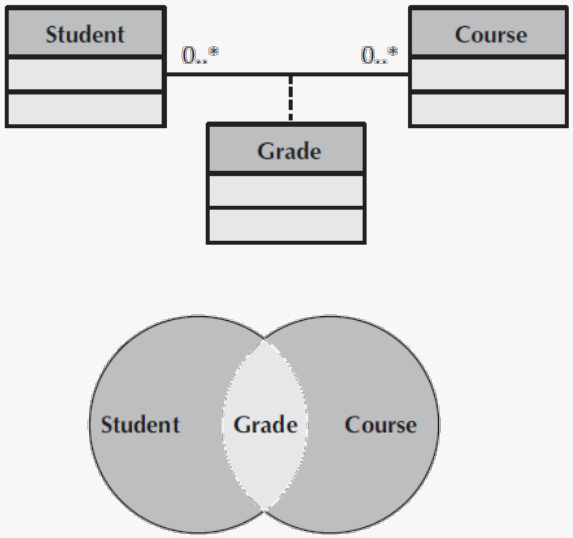
- **Association (=Relationship)**

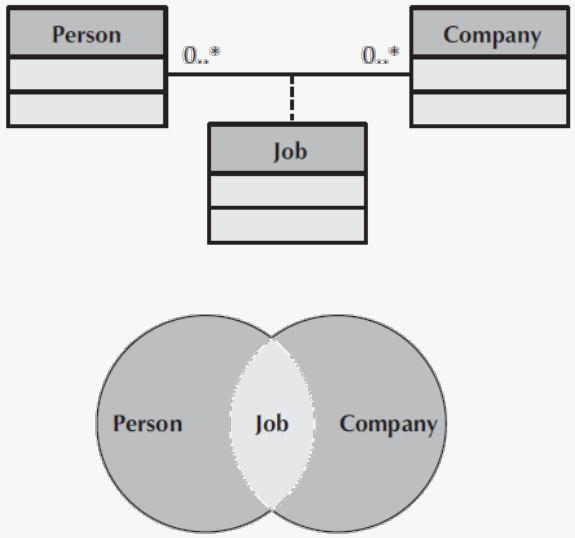
An association:	
• Represents a relationship between multiple classes or a class and itself. • Is labeled using a verb phrase or a role name, whichever better represents the relationship. • Can exist between one or more classes. • Contains multiplicity symbols, which represent the minimum and maximum times a class instance can be associated with the related class instance.	

Class Diagram Syntax (2/3)

- Operation and Association(=Relationship) (cont.)





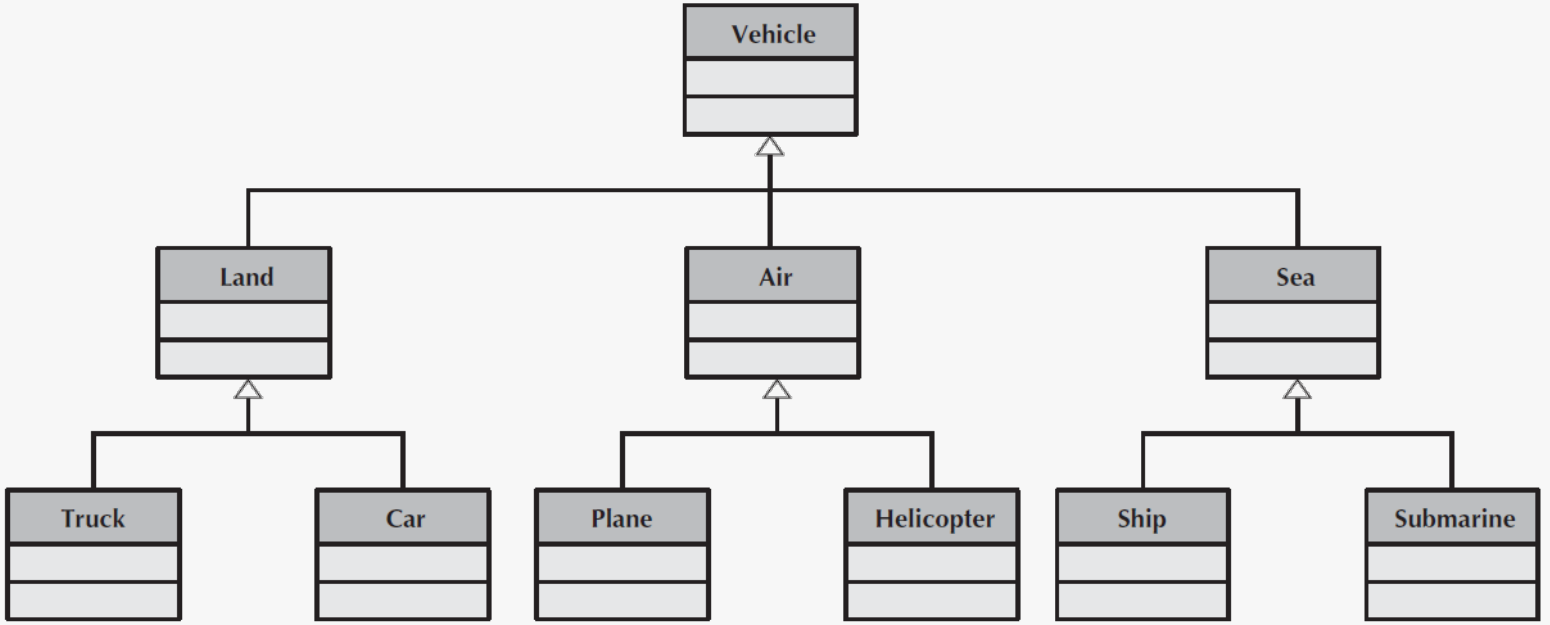


(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Class Diagram Syntax (3/3)

- Generalization, aggregation, and composition

A generalization: • Represents a-kind-of relationship between multiple classes.	

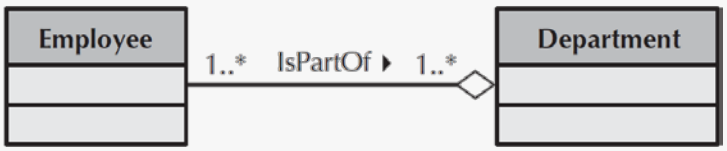


(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. Systems analysis and design. John wiley & sons.)

Class Diagram Syntax (3/3)

- **Generalization, aggregation, and composition**

An aggregation: • Represents a logical a-part-of relationship between multiple classes or a class and itself. • Is a special form of an association.	
A composition: • Represents a physical a-part-of relationship between multiple classes or a class and itself • Is a special form of an association.	



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. Systems analysis and design. John wiley & sons.)

More on Attribute

- **Derived attributes**
  - **attributes that can be calculated or derived**
  - **/age, for example, can be calculated from birth date and current date**

- **Visibility**
  - **the level of information hiding to be enforced for the attributes**
  - **hiding levels**
    - Public (+)
    - Protected (#) : permit to intermediate subclasses
    - Private (-) : default

Invoice
+amount : Real +date: Date = Current date +customer : String -administrator : String = "Unspecified" -number of invoice : Integer +status: Status = unpaid {readOnly}

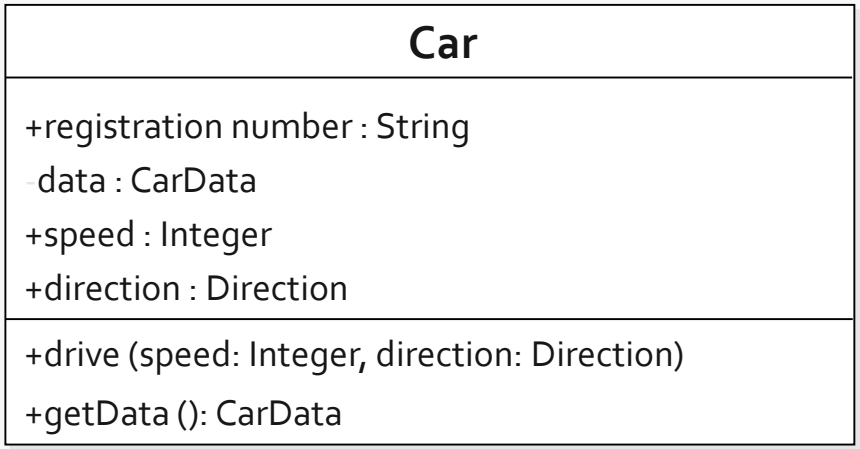
More on Operation

❖ **Some kinds of operations**

- **Constructor operation**
  - Creates object
- **Query operation**
  - Makes information about state available
- **Update operation**
  - Changes values of some or all attributes
- **Etc ..**

❖ **Visibility**

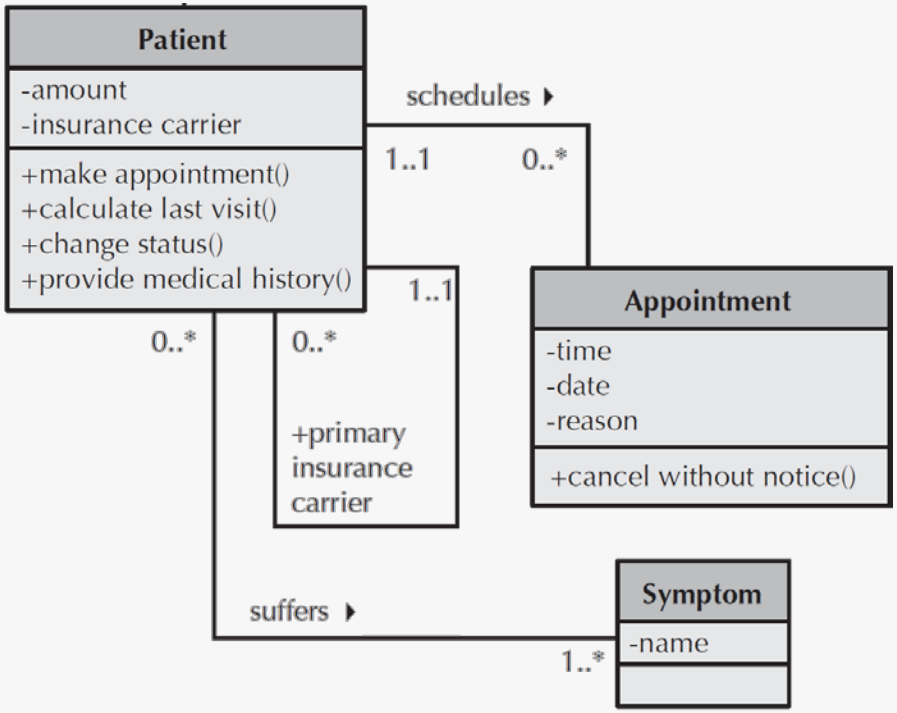
- public : default
- protected
- private



More on Relationship (1/2)

- **Class can be related to itself (role)**
  - Recursive association
  - e.g., the primary insurance carrier (i.e., their spouse, children)
  - "+" sign : a role as opposed to the name of the relationship

- **Triangle on association path**
  - direction to be read
  - increases the readability



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

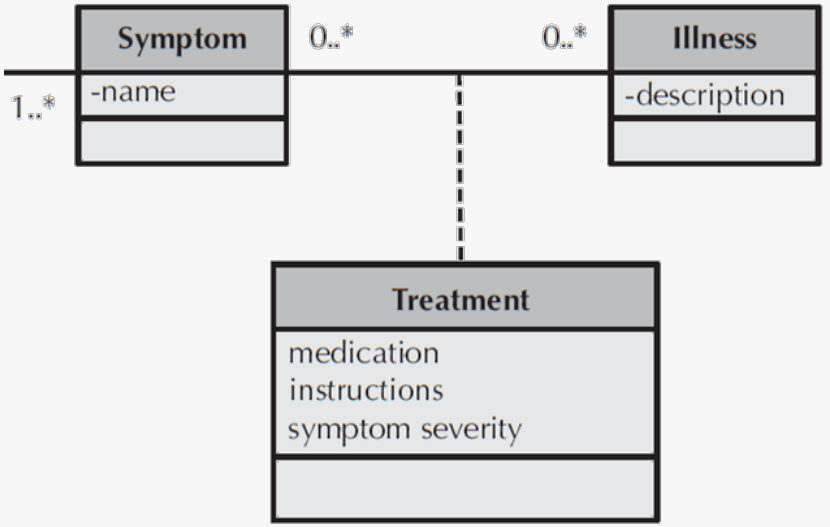
More on Relationship (2/2)

❖ Multiplicity

- how an instance of an object can be associated with other instances
- Exactly one, zero or more, one or more, zero or one, specified range, multiple disjoint ranges

❖ Association class

- when a relationship itself has associated properties
- attached by a dashed line
- the name of this class
  = the label of association



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. Systems analysis and design. John wiley & sons.)

Multiplicity

- **Multiplicities and their examples**

Exactly one	1	Department — 1 — Boss	A department has one and only one boss.
Zero or more	0..*	Employee — 0..* — Child	An employee has zero to many children.
One or more	1..*	Boss — 1..* — Employee	A boss is responsible for one or more employees.
Zero or one	0..1	Employee — 0..1 — Spouse	An employee can be married to zero or one spouse.
Specified range	2..4	Employee — 2..4 — Vacation	An employee can take from two to four vacations each year.
Multiple, disjoint ranges	1..3,5	Employee — 1..3,5 — Committee	An employee is a member of one to three or five committees.

(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth.
Systems analysis and design. John wiley & sons.)

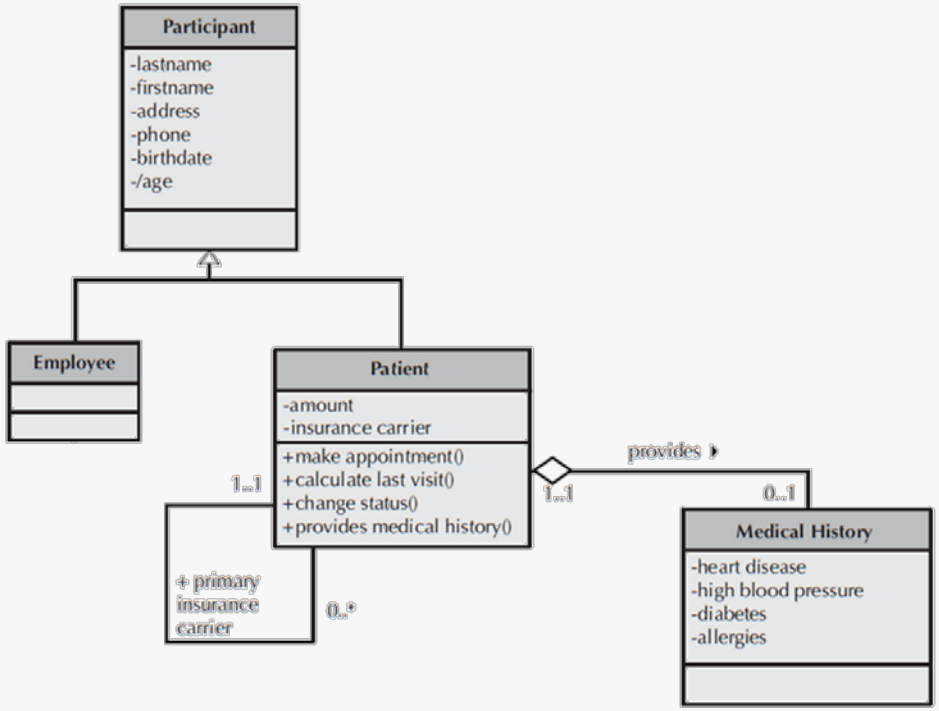
Generalization and Aggregation

❖ Aggregation

- comprise other classes
- a diamond symbol
- Health team class comprised of doctor, nurses, admin classes

❖ Generalization

- shows that a subclass (more specific element) inherits from a superclass
- a hollow arrow symbol
- Doctors, nurses, admin personnel are kinds of employees



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. Systems analysis and design. John wiley & sons.)

Dependency and Abstraction (1/2)

- **Dependency**
  - relationship between elements, one independent and one dependent
  - a change in the independent element will affect the dependent element

- **Abstraction /Realization**
  - relationship between two description of the same thing, but at different levels
  - realization
    - a type of dependency
    - shows a model element that realizes a more general element

Dependency and Abstraction (2/2)

❖ Dependency

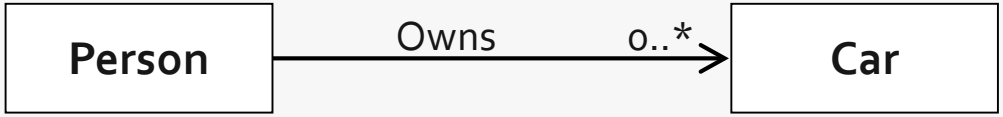
- <<use>> or <<permit>> type

❖ Abstraction /Realization

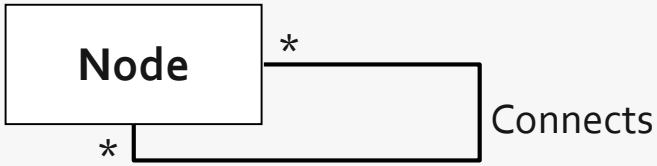
- <<refine>>, <<trace>> or <<derive>> type

More on Association Relationship (1/3)

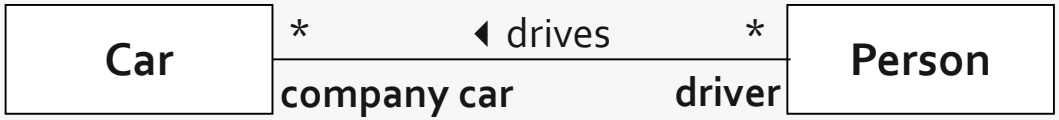
- **Navigable association**



- **Recursive association**

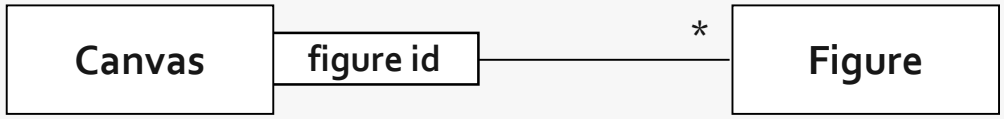


- **Roles in an association**

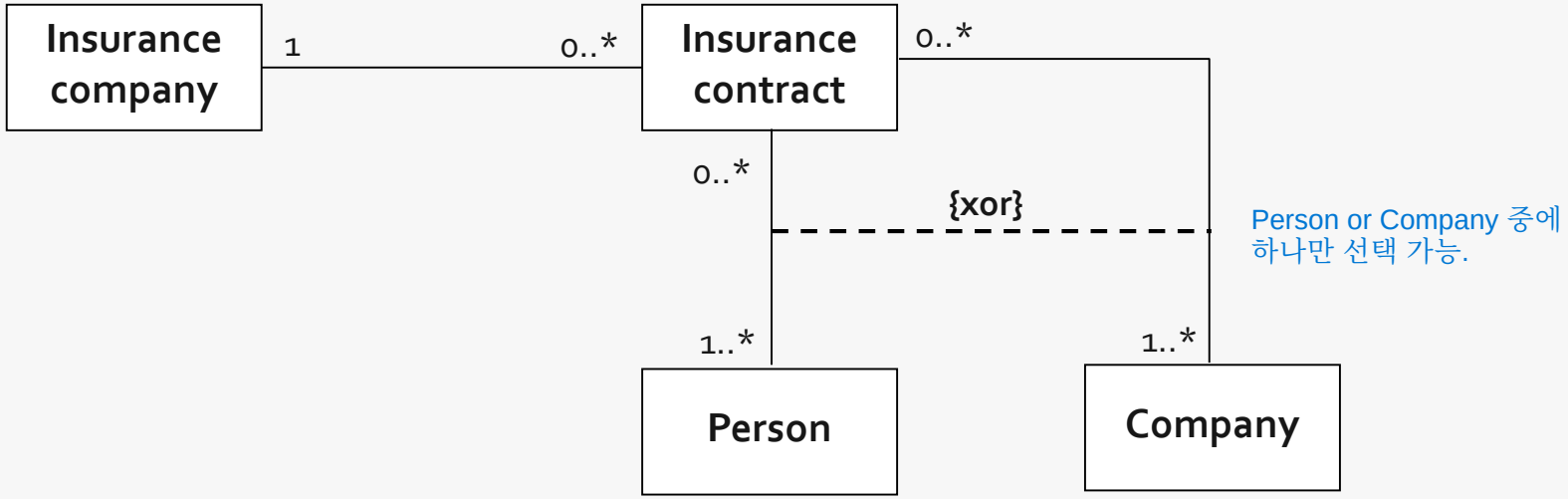


More on Association Relationship (2/3)

❖ Qualified association

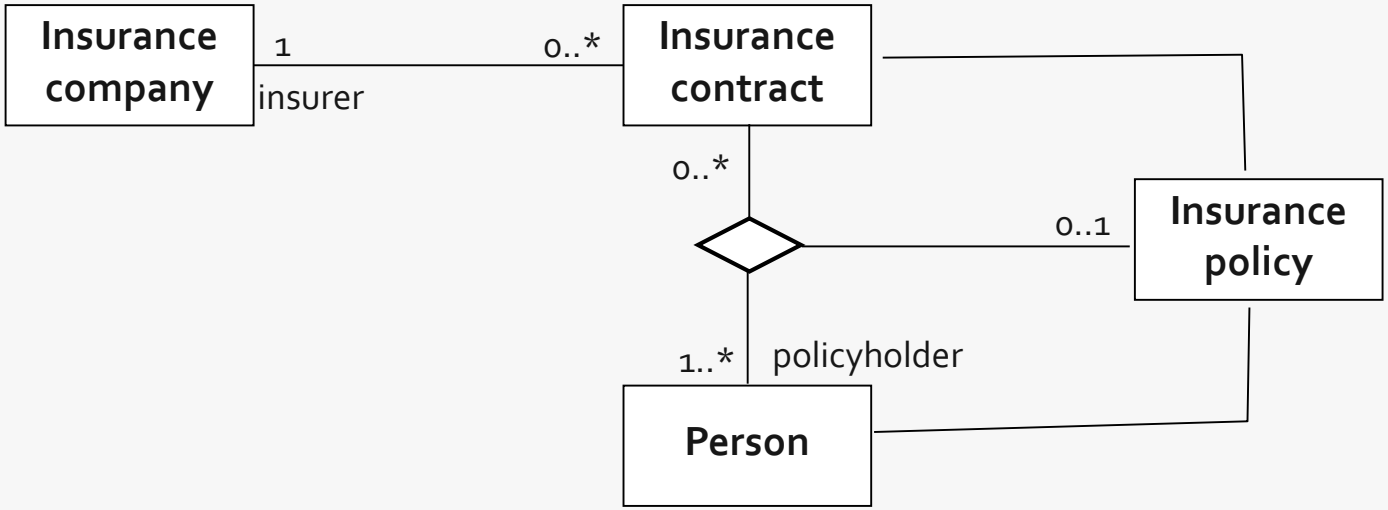


❖ XOR Constraint



More on Association Relationship (3/3)

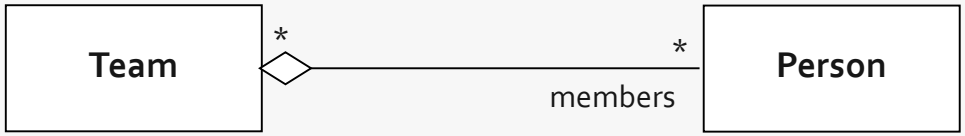
❖ **Ternary association**



→ **Association transformation**

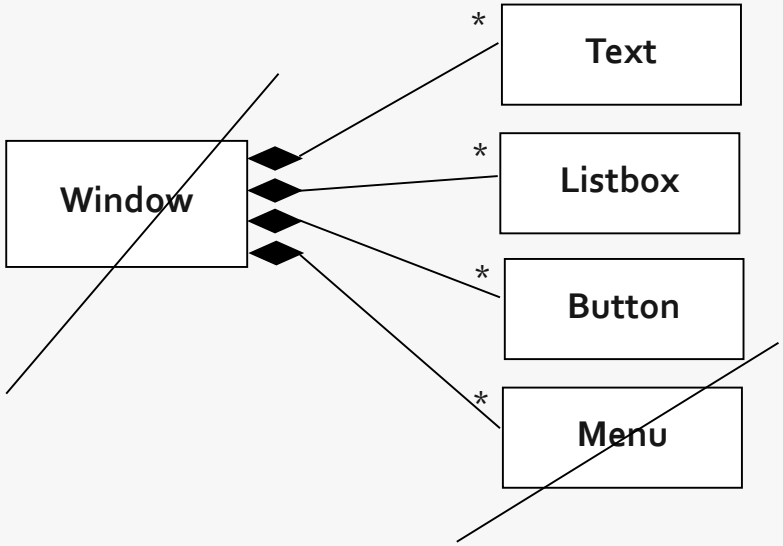
More on Aggregation Relationship (1/2)

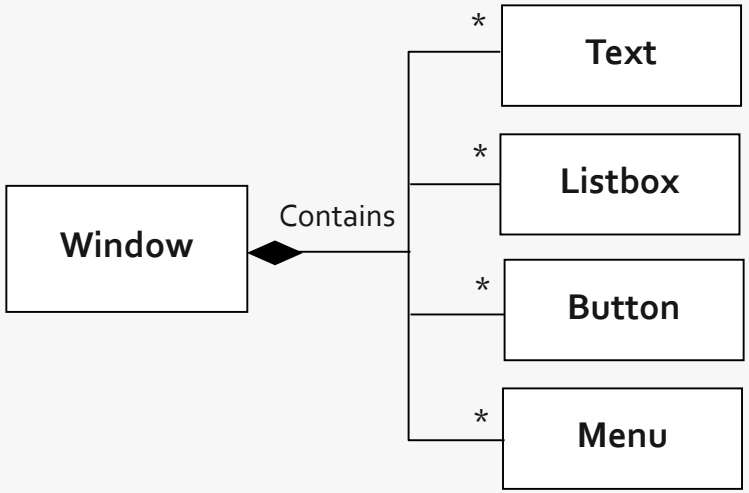
❖ **Shared aggregation**



❖ **Composition Aggregation (1)**

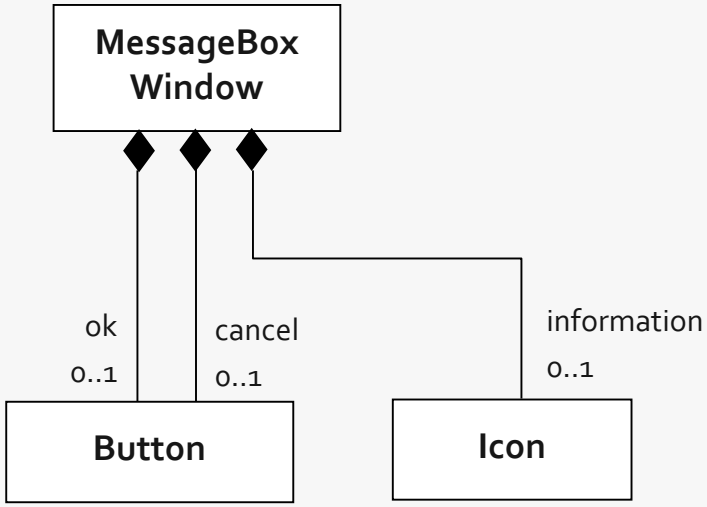
상위 클래스가 없어지면 하위 클래스도 다 사라짐

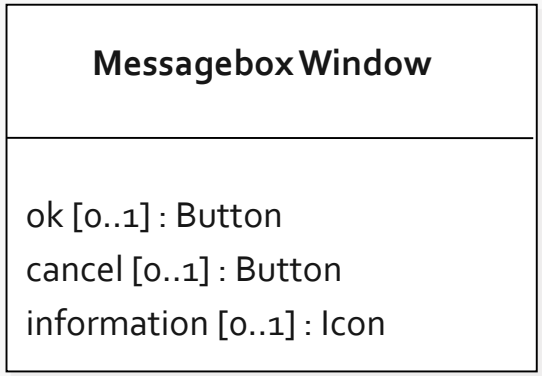




More on Aggregation Relationship (2/2)

❖ **Composition Aggregation (2)**

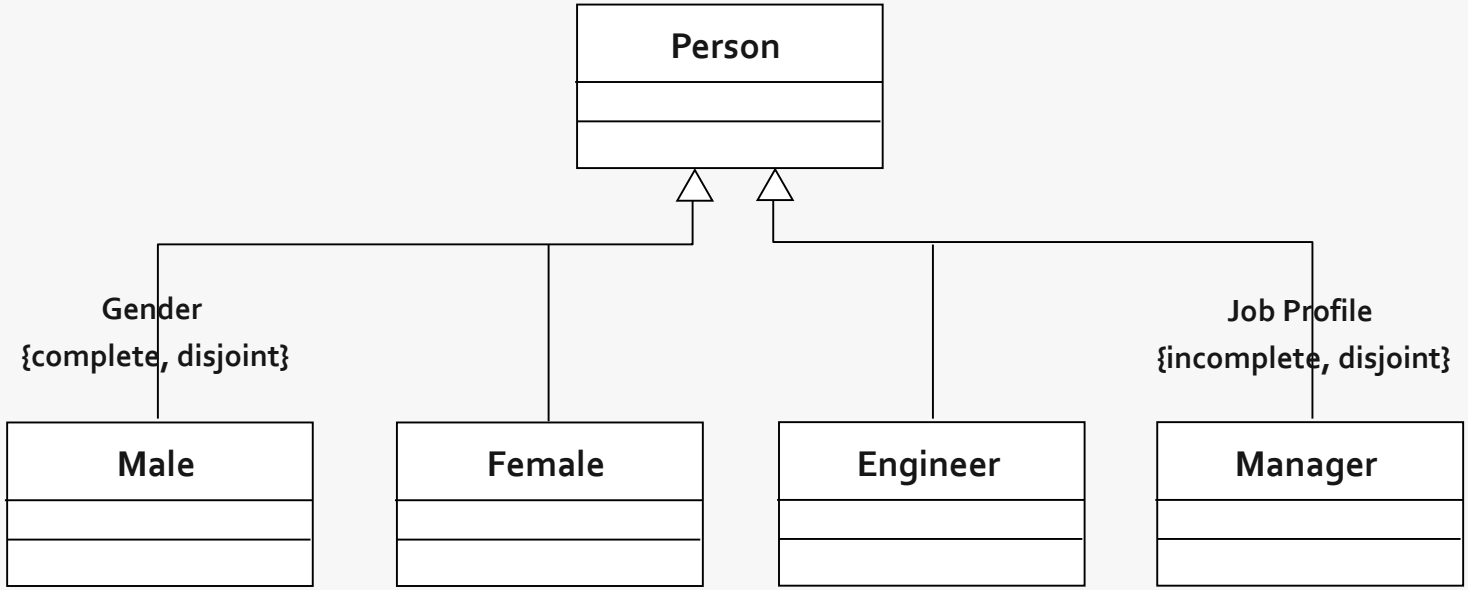




More on Generalization Relationship (1/2)

❖ Generalization Set

- organize different sets of subclasses that inherit from a common superclass

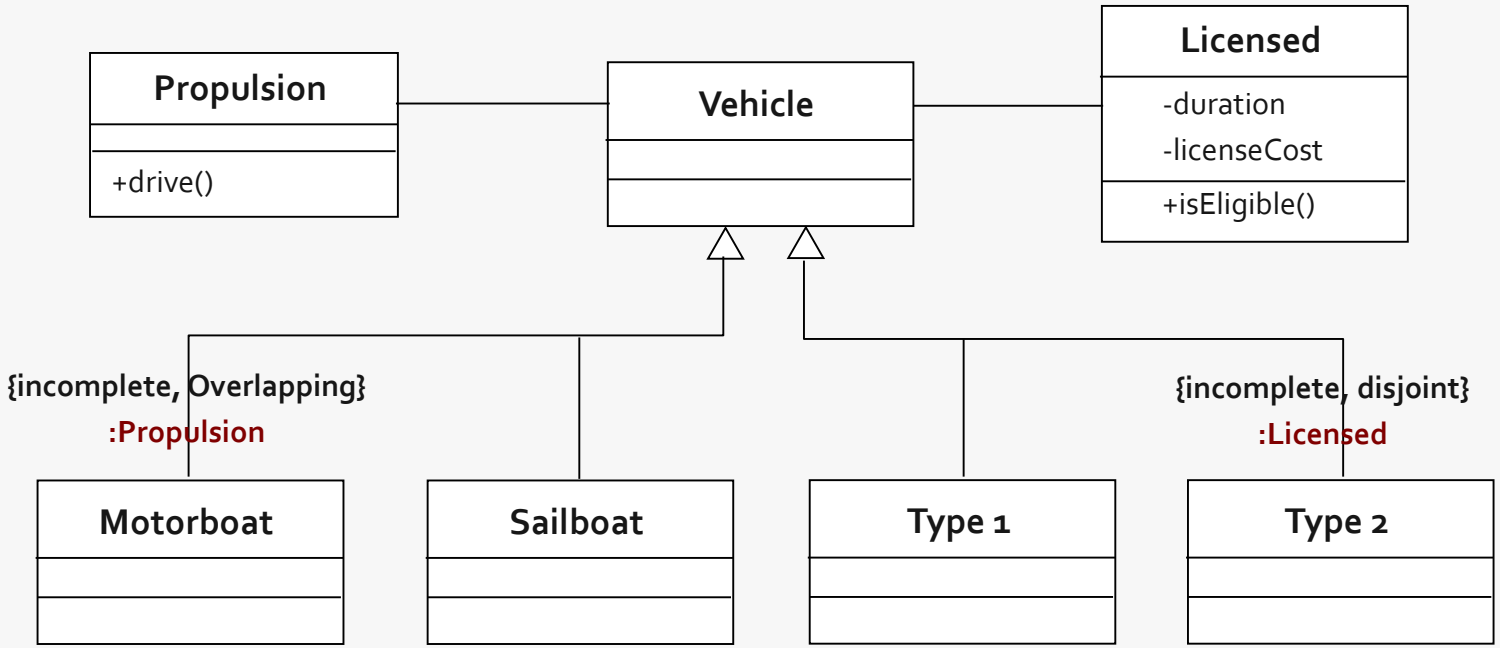


More on Generalization Relationship (2/2)

❖ Powertype

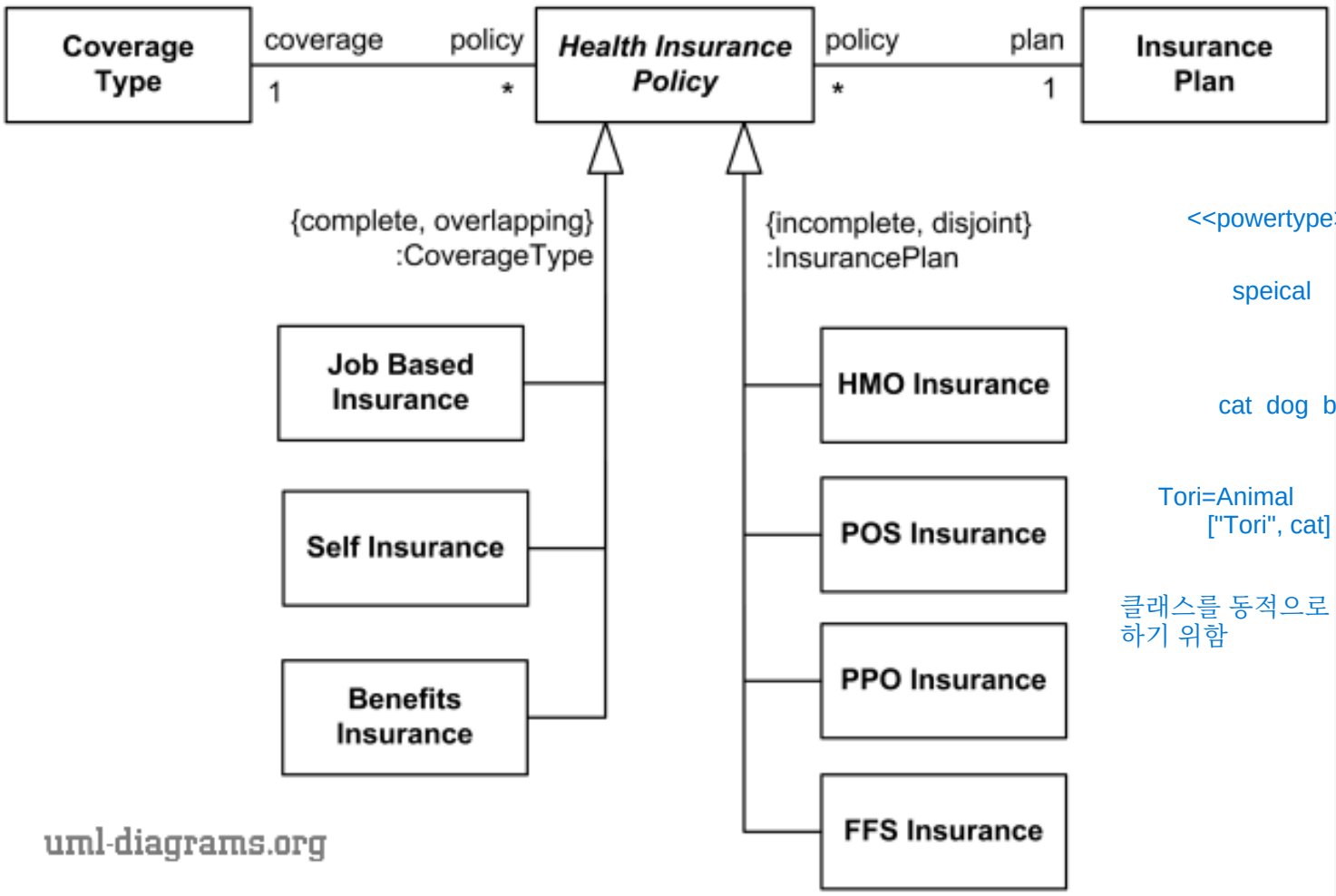
클래스 자체를 분류하는 클래스

- a mechanism to model classes made of subclasses
- all instance of powertype are also subclasses



More on Generalization Relationship (2/2)

- **Powertype (more example)**



<<powertype>>

speical

cat dog bird

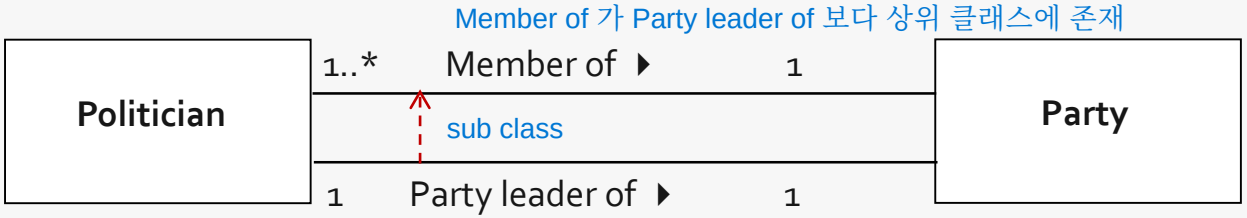
Tori=Animal
["Tori", cat]

클래스를 동적으로 분류해서 사용하기 위함

Other Features (1/3)

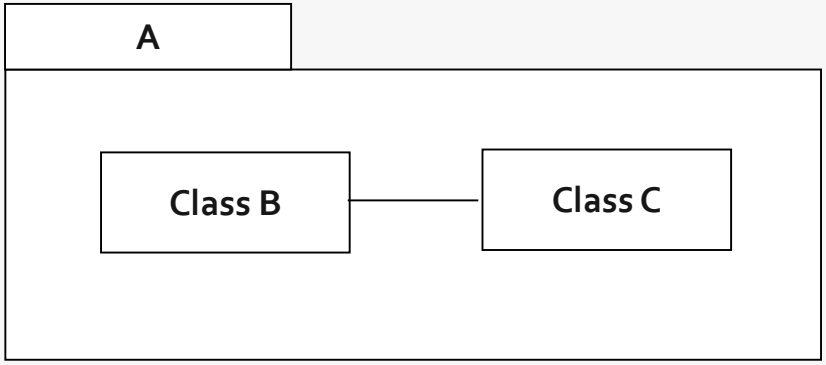
❖ Constraint association

- The party leader is a subset of the member of association



❖ Packages

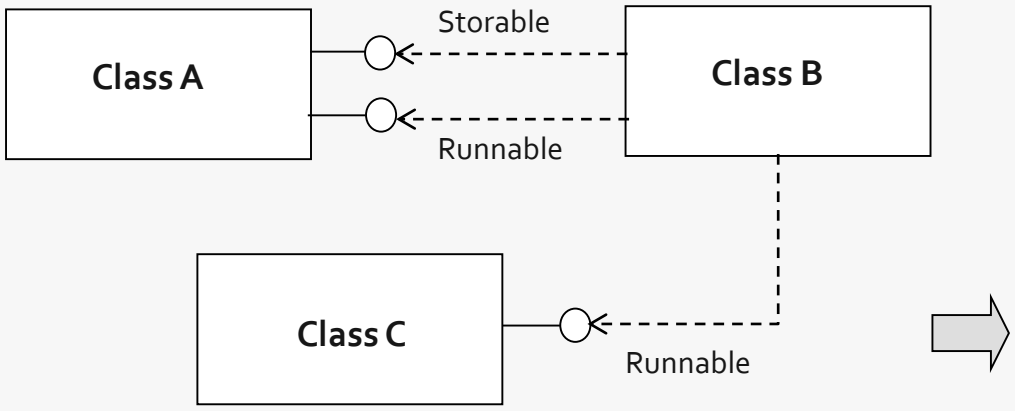
- grouping mechanism for organizing UML elements



Other Features (2/3)

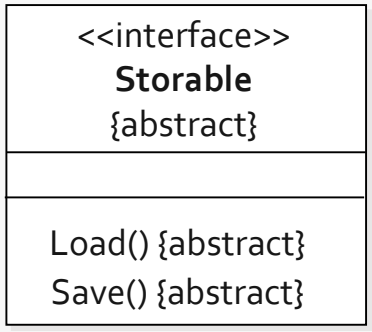
❖ Interface

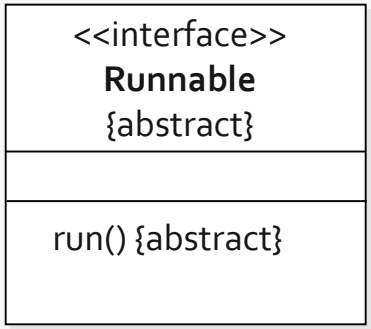
- to implement or use the specified interface




```
interface Storable
{
    public void Save();
    public void Load();
}

public class A implements Storable
{
    public void Save()
    {
        // implementation
    }
    public void Load()
    {
        // implementation
    }
}
```



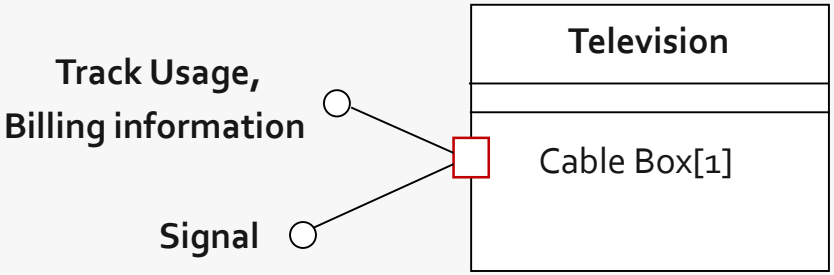


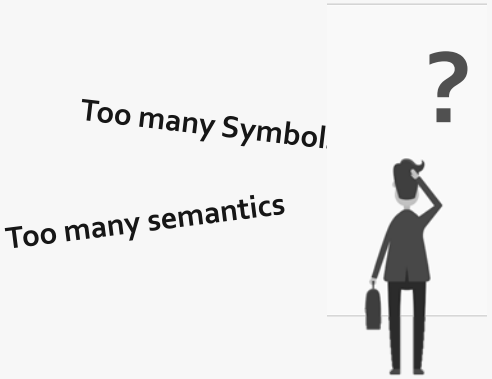
Other Features (3/3)

❖ Port

주변 환경 연결점 묘사

- Not special behaviors, But just environment context

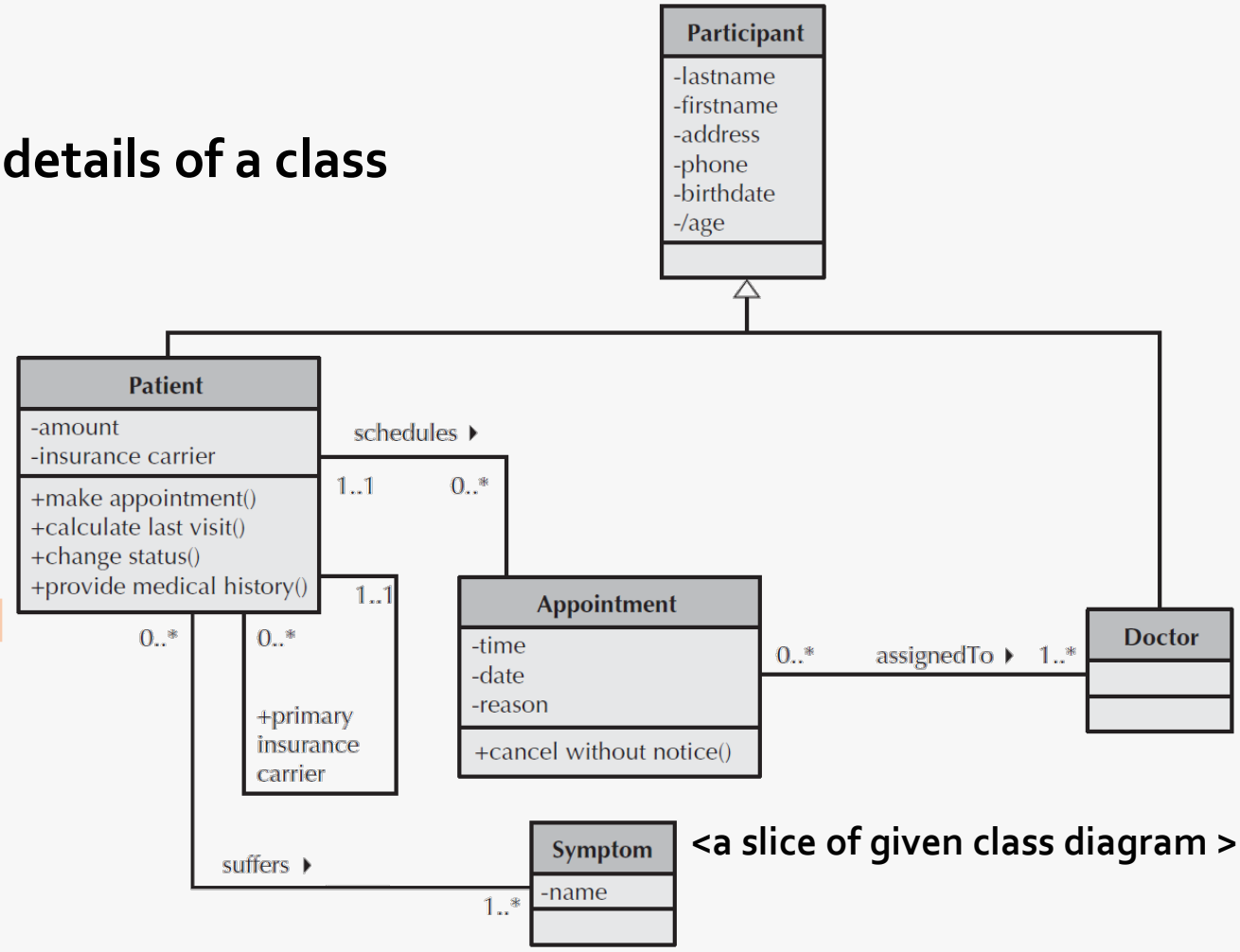




Object Diagrams (1/2)

- An instantiation of all or part of a class diagram
- Useful to uncover details of a class

What is instantiation ?

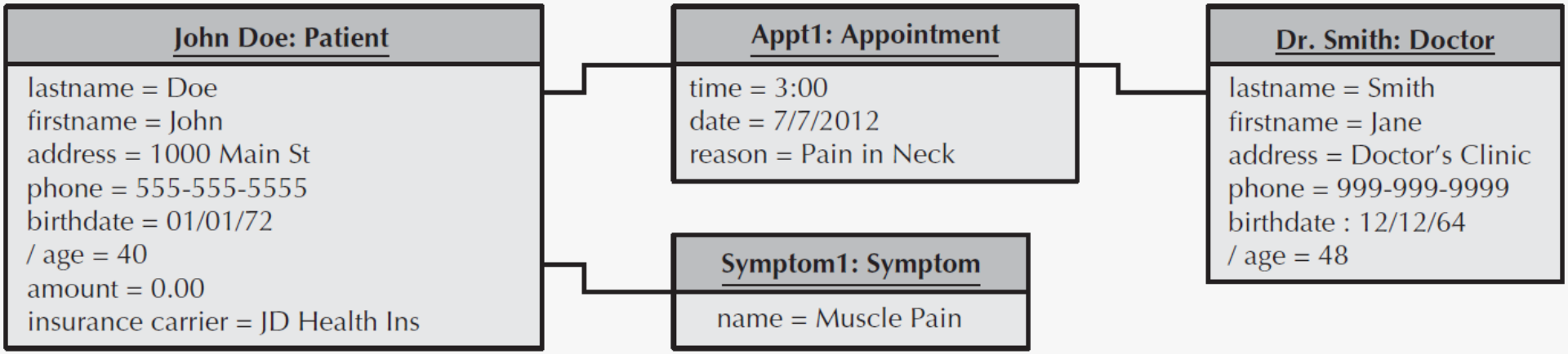


<a slice of given class diagram >

(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Object Diagrams (2/2)

- An object diagram from previous slide



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Creating CRC Card & Class Diagram

❖ 4 approaches for Object Identification

- Textual analysis / Brainstorming / Common object list / Patterns

❖ Textual analysis 문어 분석 = 어떤 클래스를 만들어야 되는가?

- analyze the text of use-case descriptions
- identify potential objects, attributes, operations and relationship

  : Nouns suggest classes

  : Verbs suggest operations

❖ Textual analysis guidelines 분석 순서 및 자세한 기술

- A common or improper noun implies a class of objects.
- A proper noun or direct reference implies an instance of a class.
- A collective noun implies a class of objects made up of groups of instances of another class.
- An adjective implies an attribute of an object.
- A doing verb implies an operation.

Textual Analysis

❖ Textual analysis guidelines (Cont'd)

- A being verb implies a classification relationship between an object and its class.
- A having verb implies an aggregation or association relationship.
- A transitive verb implies an operation.
- An intransitive verb implies an exception.
- A predicate or descriptive verb phrase implies an operation.
- An adverb implies an attribute of a relationship or an operation.

Common Object List

- **A list of objects that are common to the business domain of the system.**
- **Object categories**
  - **physical and tangible things**
    - books, desks, office equipment, etc
  - **incident : event that occur in the business domain**
    - meetings, flights, performance, or accidents
  - **role : people play in the problem**
    - doctor, nurse, patient, etc
  - **interactions : a transaction that take place in the Biz domain**
    - sales transaction
  - **and more such as places, organization, policies, …**

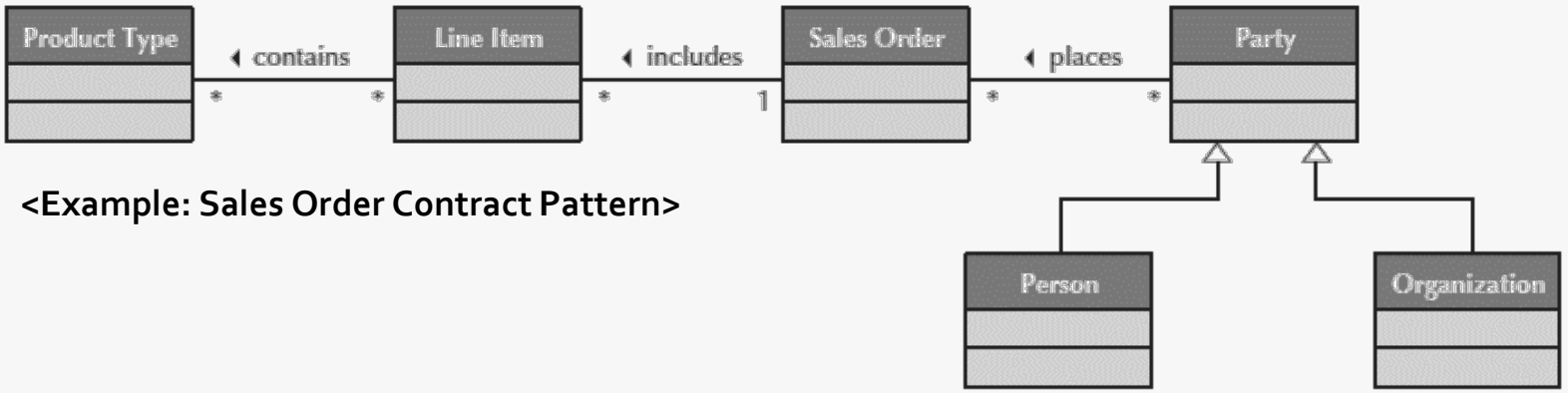
Brainstorming

- **Brainstorming - people offering ideas**
  - Initial list of classes (objects) is developed
  - Attributes, operations and relationships to other classes can be assigned in a second round

- **Considerations**
  - Performed with the coordination of facilitator
  - Does not use the functional models
  - Potential set of objects that come from mind.

Patterns

- **Useful group of collaborating classes that provide a solution to a common occurring problem**
- **Reusable components** 타 객체지향 설계에서 패턴을 가져다 씀. 비슷한 것 끼리.
- **Transactions, for example**
  - Transaction class
  - Transaction line class
  - Item class
  - Location class
  - Participant class



<Example: Sales Order Contract Pattern>

(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Steps for Object Id. & Structural Modeling

1. Create CRC cards by performing textual analysis on the use-cases.
2. Review CRC cards to find additional candidate classes.
3. Role-play each use-case using the CRC cards.
4. Create the class diagram based on the CRC cards.
5. Review the structural model for missing and/or unnecessary classes, attributes, operations, and relationships.
6. Incorporate useful patterns.
7. Review the structural model.

Building CRC Card is OPTIONAL in Small and Well-Known software modeling .

Steps for Structural Modeling

1. **Create CRC Cards** use cases에 나온 걸 기반으로 CRC를 만듦
   - By performing textual analysis on the use-case description which describes in normal flow, sub flows, and alternative flows.
   - Sometimes, review original requirements to look for another information.

2. **Review CRC Cards**
   - Determine if additional candidate classes, attributes, operations, and relationships are missing
   - Using brainstorming and common object list approach

Steps for Structural Modeling

3. Role-playing CRC Cards

- An exercise to help discover additional objects, attributes, relationships & operations
- Team members perform roles associated with the actors and objects previously identified
- Team members perform each step in use-case scenario.
- Discover and fix problems until a successful conclusion is reached

4. Create Class Diagram

- Using the information contained on the CRC Cards
  - Responsibilities :: operations in class
  - Attributes :: attributes in class
  - Relationships :: generalization, aggregation, and association

Steps for Structural Modeling

5. **Review Class Diagram**
   - For missing / unnecessary classes, attributes, operations and relationships
   - By devil's advocate

6. **Incorporate Patterns**
   - To evolve the structural model
   - Compare the classes between the selected pattern and the class model
   - Incorporate the pattern into the model, and modify the CRC Cards

7. **Review the Model**
   - CRC cards and class model

Applying the Steps with an Example

- **Use case description for "Borrow Book"**

자꾸 등장하는 noun은 class로

verb나 기타 등은 action 명시

Normal Flow of Events:

1. The Borrower brings books to the Librarian at the check out desk.
2. The Borrower provides Librarian his or her ID card.
3. The Librarian checks the validity of the ID Card.
   - If the Borrower is a Student Borrower, Validate ID Card against Registrar's Database.
   - If the Borrower is a Faculty/Staff Borrower, Validate ID Card against Personnel Database.
   - If the Borrower is a Guest Borrower, Validate ID Card against Library's Guest Database.
4. The Librarian checks whether the Borrower has any overdue books and/or fines.
5. The Borrower checks out the books.

SubFlows:

Alternate/Exceptional Flows:

- 4a. The ID Card is invalid, the book request is rejected.
- 5a. The Borrower either has overdue books fines, or both, the book request is rejected.

(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.

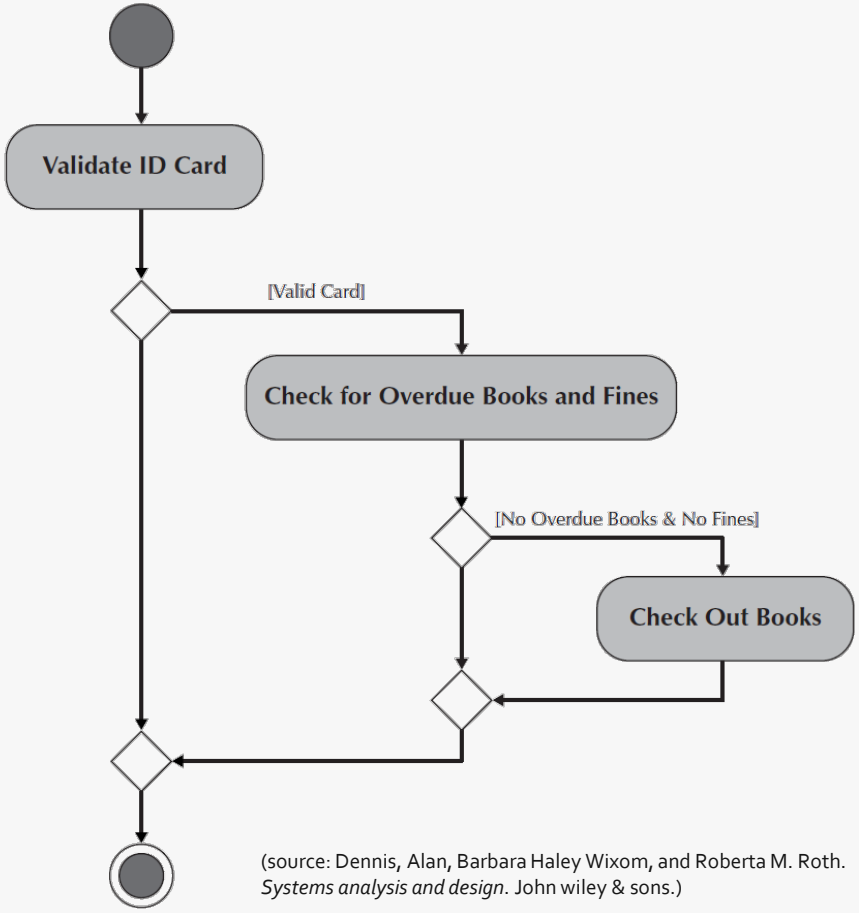
Applying the Steps with an Example

- **Identified candidate classes from ..**
  - **Use Case Description**
    - Borrower, Books, Librarian, Checkout desk, ID card, Student borrower, Faculty/Staff borrower, Guest borrower, Registrar's database, Personnel database, Library's guest database, Overdue books, fines, …
    - Check the validity, Check out, Reject …
  - **CRC Cards**
    - Ability to search the books by title, author, keywords, ISSN number, …
  - **Brainstorming**
    - Retrieve books from storage, Move books to storage, ….
    - Journals, DVDs, and other Media.

Applying the steps with an Example

❖ Role-Playing the CRC cards

- Activity diagram for the "Borrow Book" from Use-Case Normal Flow



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Three types of Borrower

: Student

: Faculty/Staff

: Guest

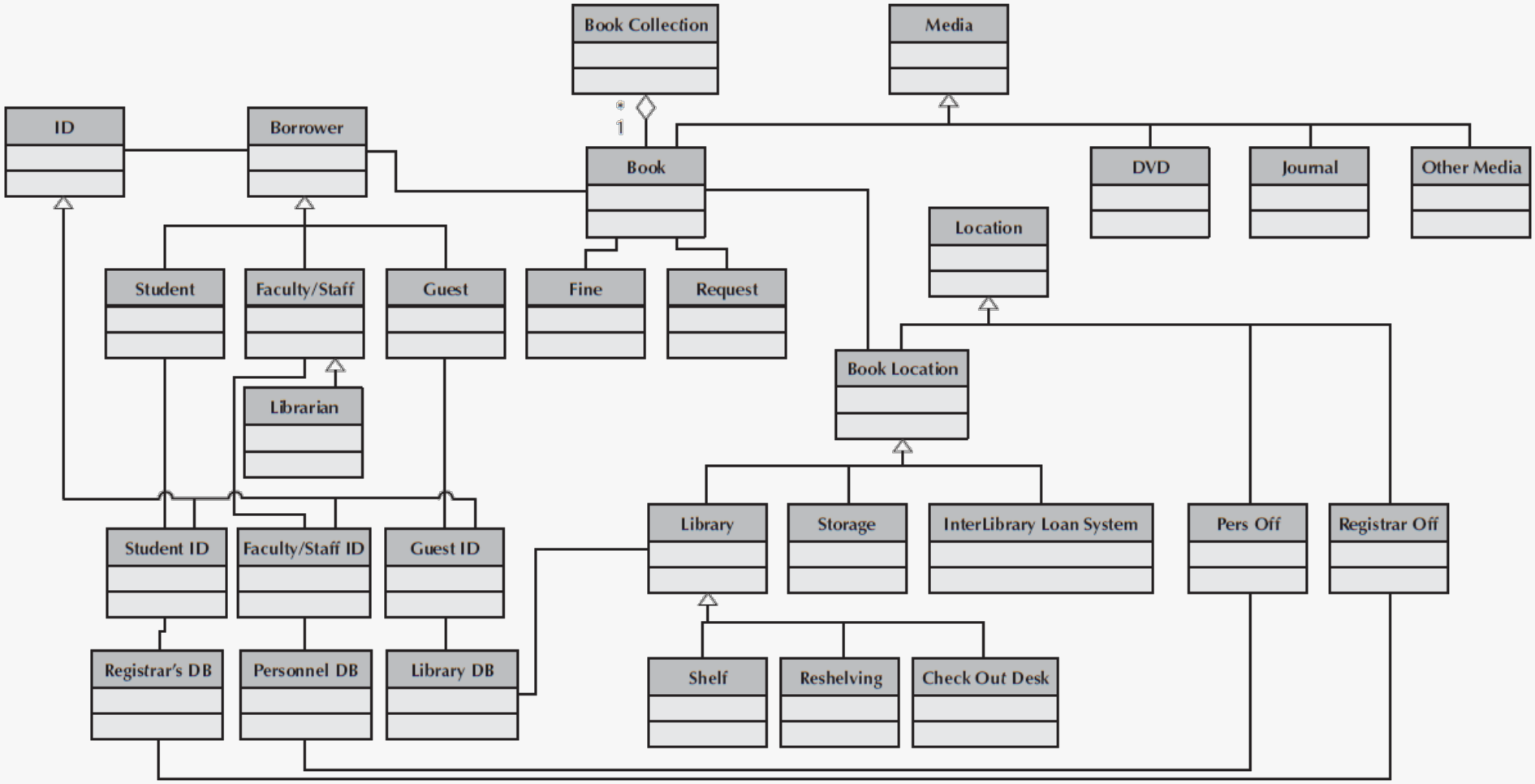
Three Paths from the Activity diagram

Nine Scenarios

3 * 3 = 9

Applying the steps with an Example

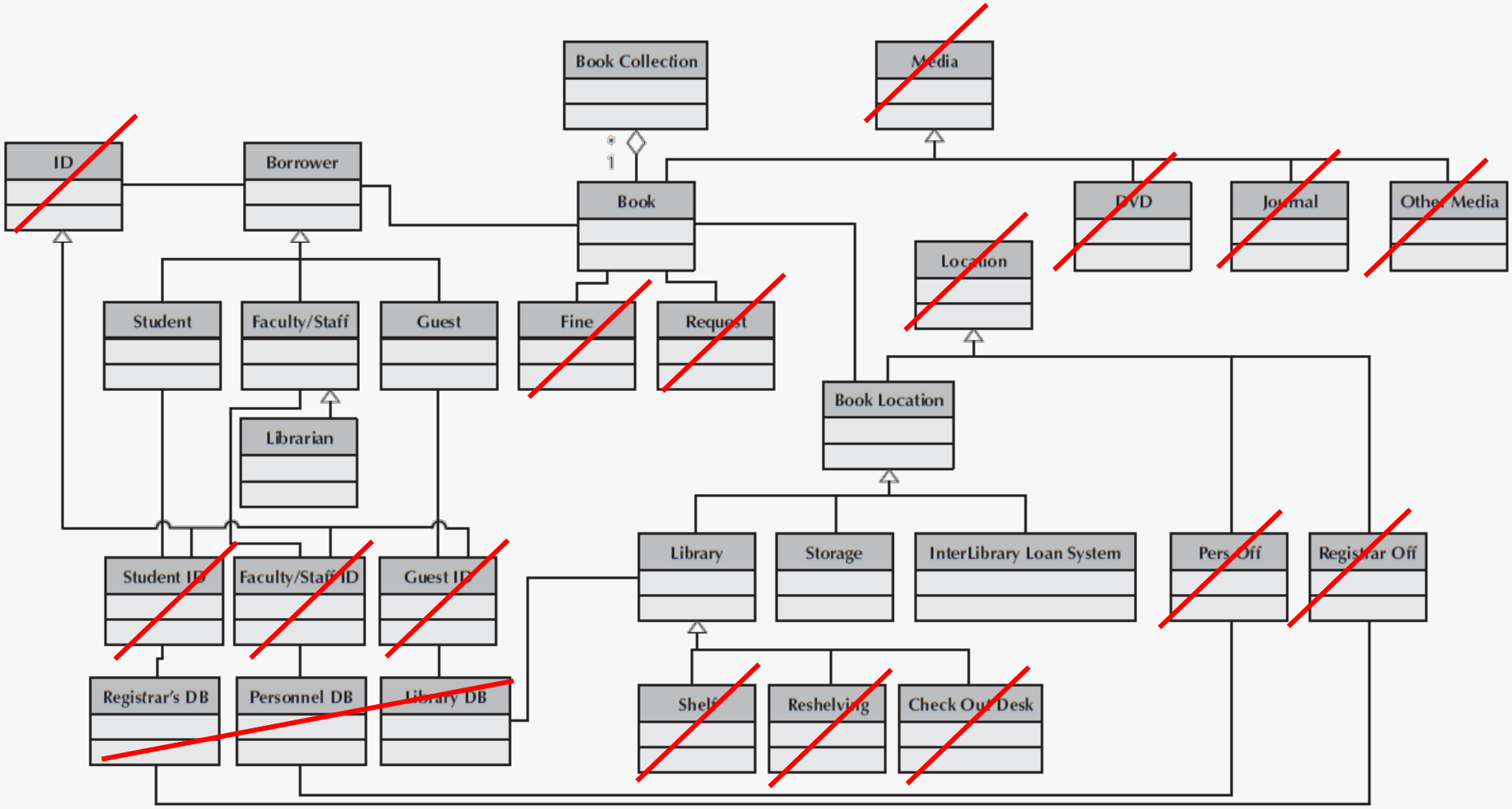
- First-cut class diagram for Library Book Collection System



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Applying the steps with an Example

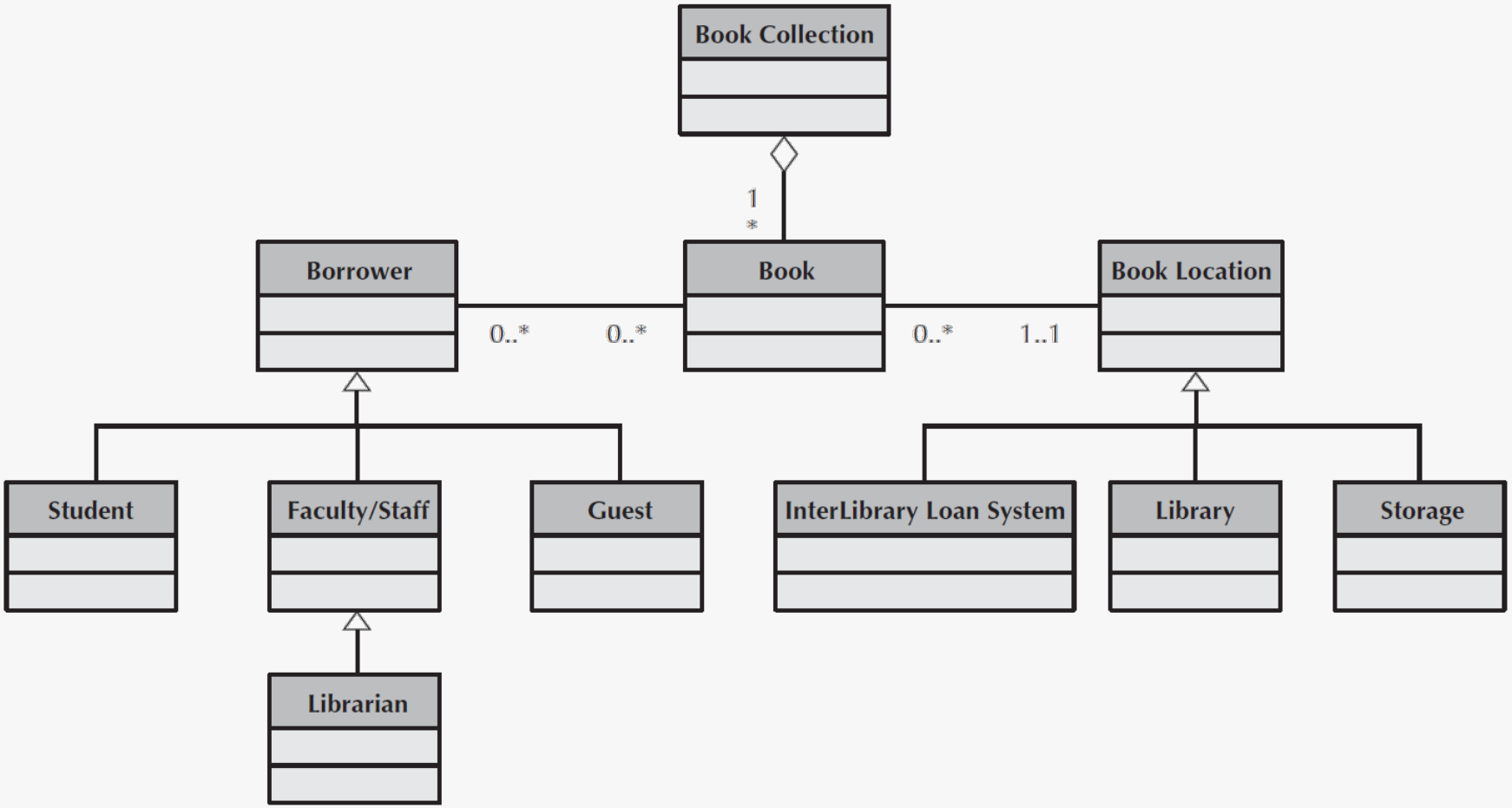
- First-cut class diagram for Library Book Collection System



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Applying the steps with an Example

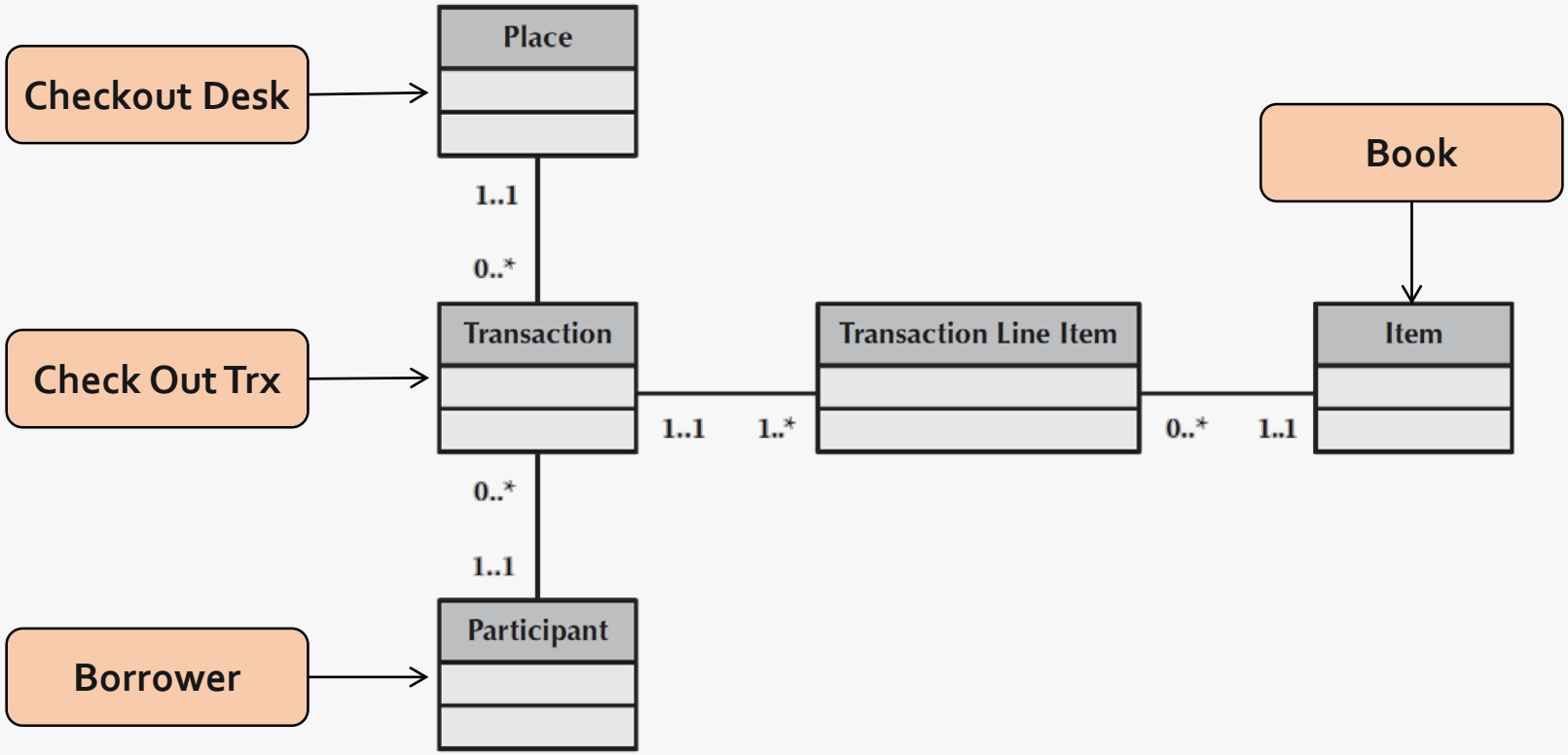
- Second-cut class diagram for Library Book Collection System



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. Systems analysis and design. John wiley & sons.)

Applying the steps with an Example

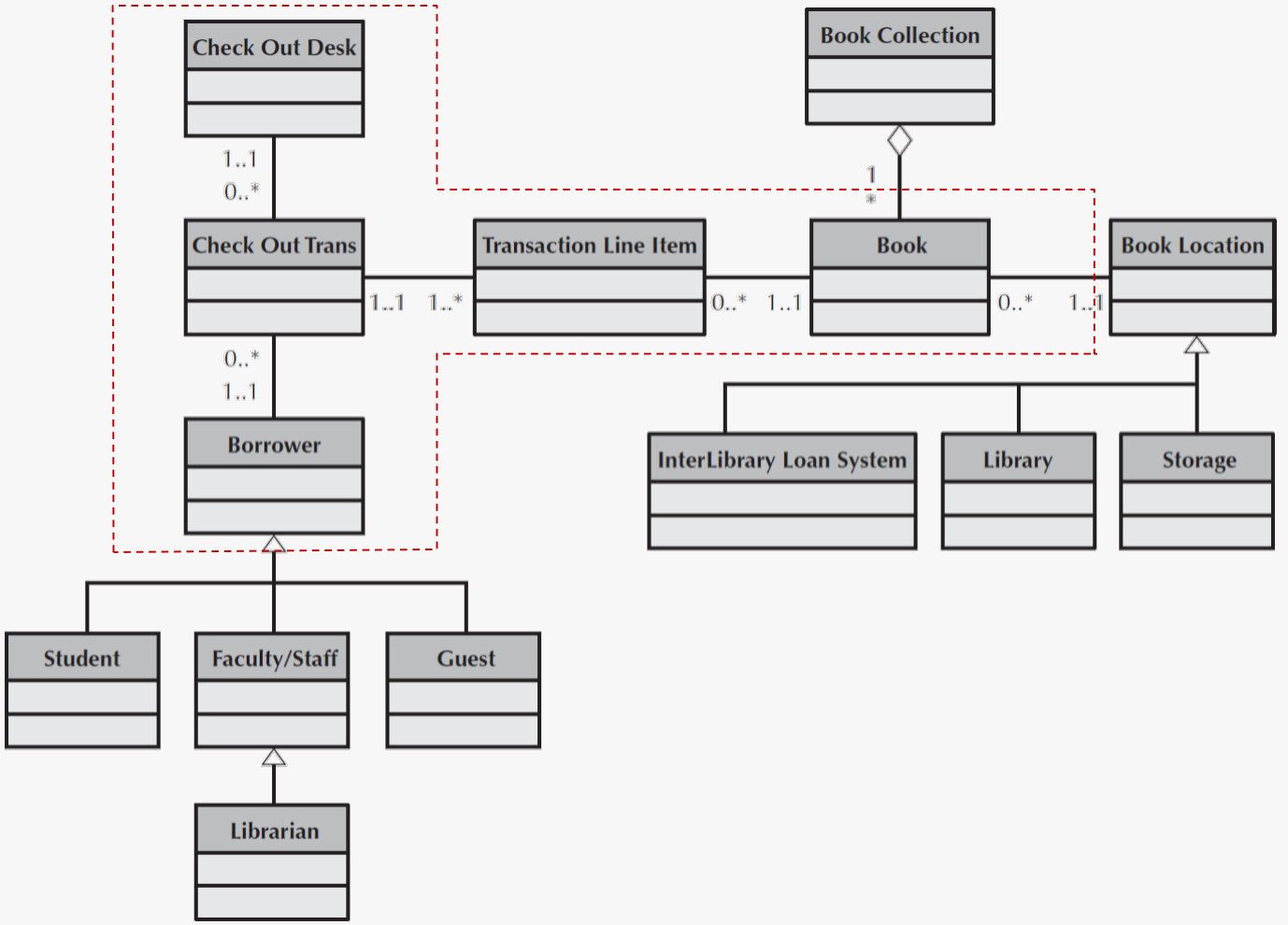
- Pattern used in Library Problem



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. Systems analysis and design. John wiley & sons.)

Applying the steps with an Example

- Class diagram with incorporated pattern



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Case Study : CD Selection Company

- Create CRC cards. (1/2)

Front:

Class Name: Customer	ID: 1	Type: Concrete, Domain
Description: An individual that may or has purchased merchandise from the CD Selections Internet sales system		Associated Use Cases: 3

Responsibilities	Collaborators

Back:

Attributes:

First name	State
Middle initial	Country
Last name	Zip code
Street address	E-mail
City	Credit card

Relationships:

Generalization (a-kind-of):

Aggregation (has-parts):

Other Associations: Order; Search Request

(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Case Study : CD Selection Company

- Create CRC cards. (2/2)

Front:

Class Name: Order	ID: 2	Type: Concrete, Domain
Description: An order that has been placed by a customer which includes the individual items purchased by the customer		Associated Use Cases: 3

Responsibilities	Collaborators
Calculate tax	
Calculate shipping	
Calculate total	

Back:

Attributes:

Tax	
Shipping	
Total	

Relationships:

Generalization (a-kind-of):

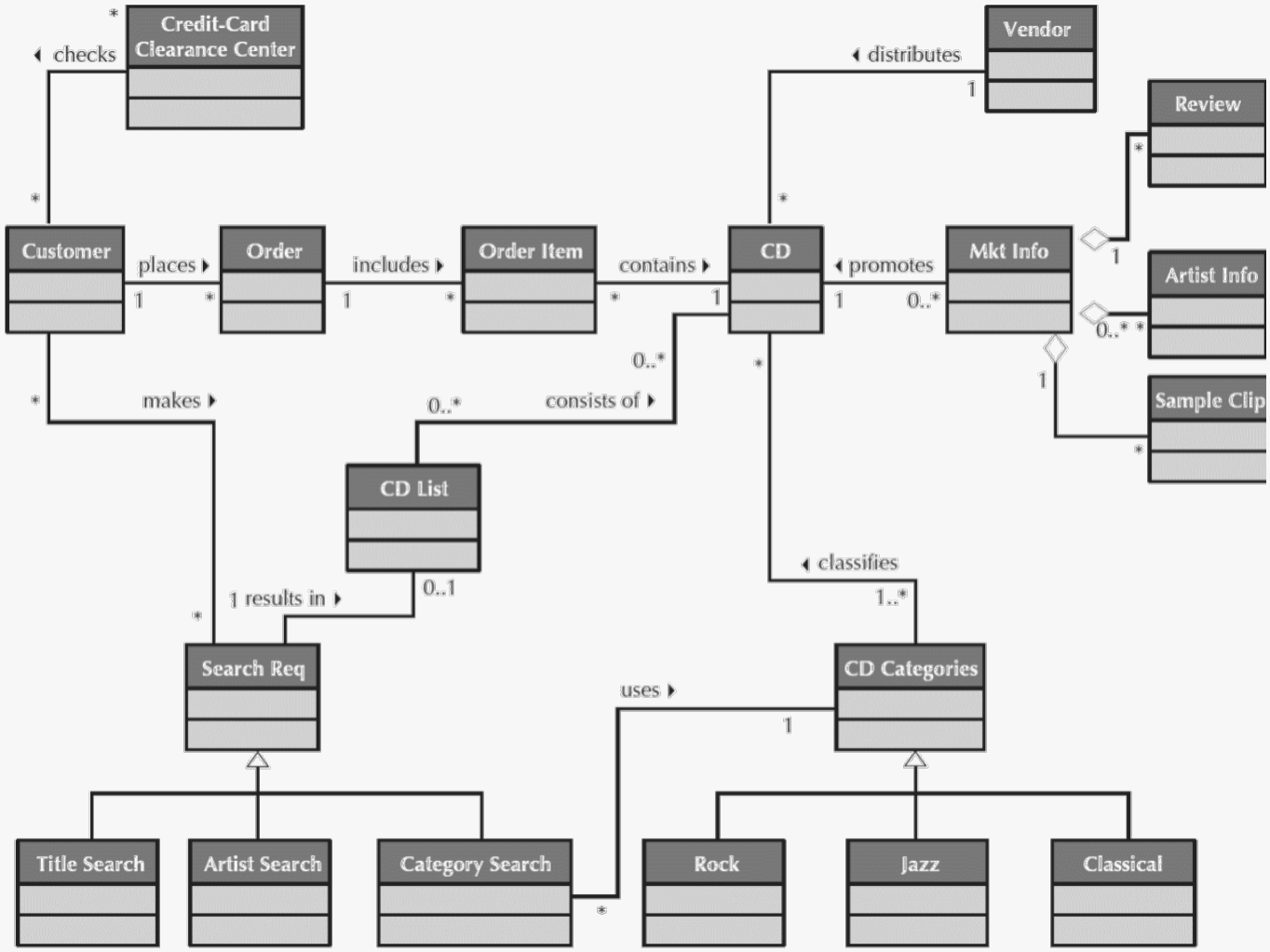
Aggregation (has-parts):

Other Associations: Order Item; Customer

(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Case Study : CD Selection Company

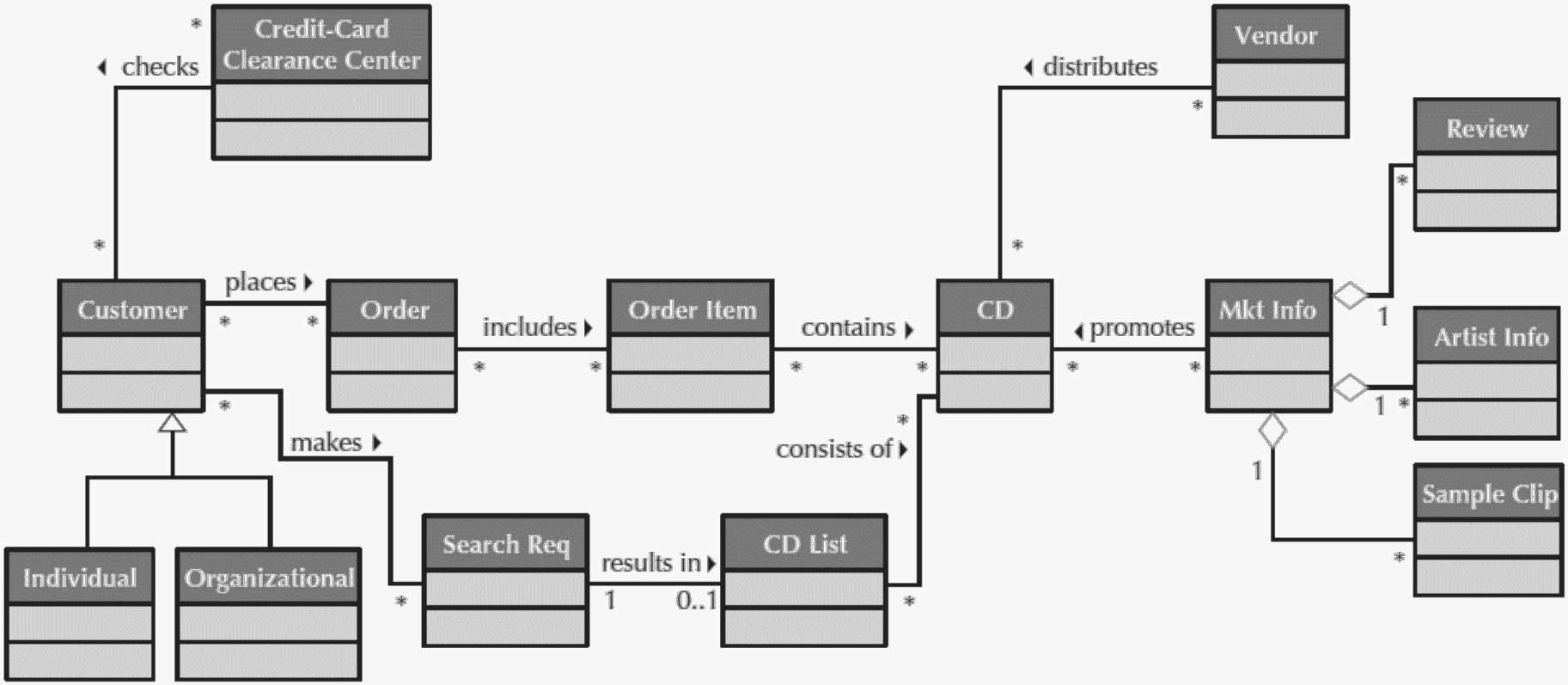
- Create the class diagram. (Place Order Use case View).



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Case Study : CD Selection Company

- Revised class diagram (Place Order Use case View).

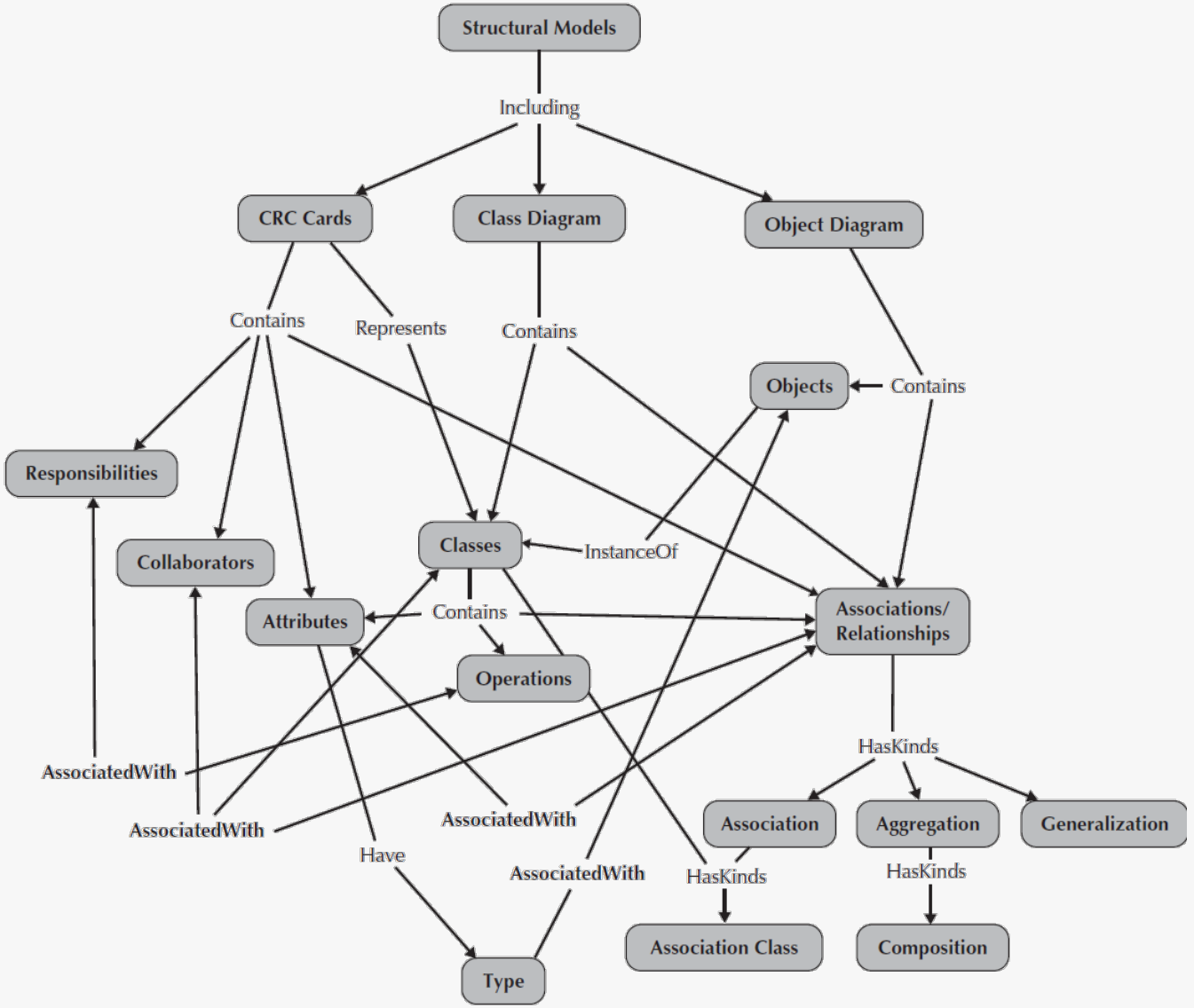


(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Simplifying Class Diagrams

- **To reduce the complexity of a class diagram**
- **View mechanism**
  - a useful set of the class diagram, such as a use-case view
  - show only a particular type of relationship
  - restrict the information shown with each class(show name only)
- **Packages**
  - grouping the classes
  - can be applied to any elements in UML models

Interrelationship among Structural Models



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Summary and Discussion

- CRC cards capture the essential elements of a class.
- Class and object diagrams show the underlying structure of an object-oriented system.
- Class diagram contains : classes and their relationships

- Should be all symbols and semantics represents in a class diagram ?
- First drawn class diagram is not changed, isn't it ?