

5. Modeling

Choi, Kwanghoon

Chonnam National University

Table of Contents

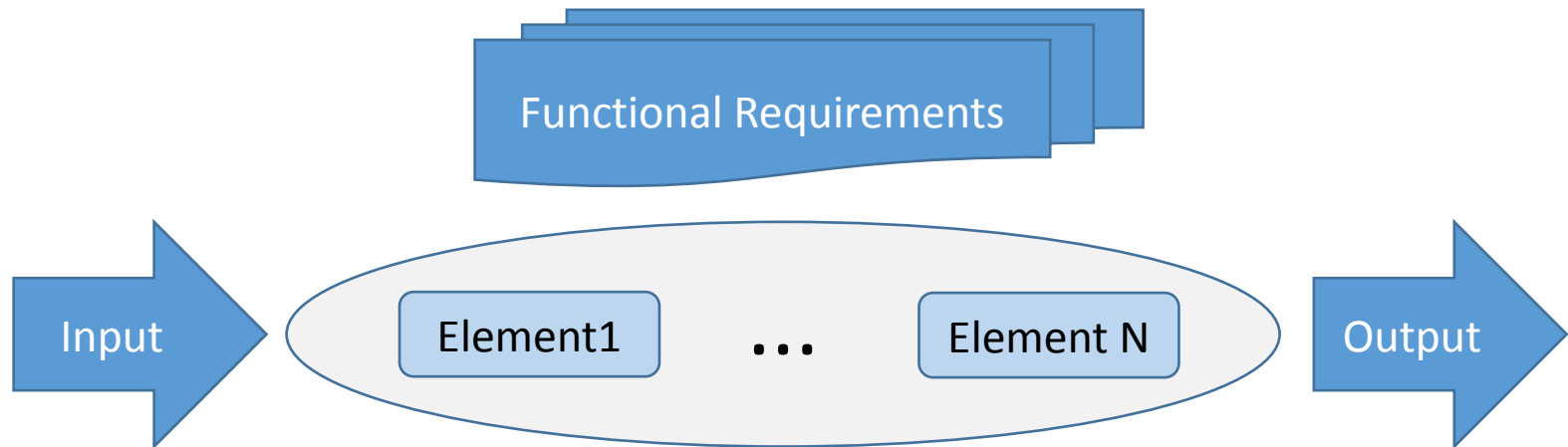
- What is modeling?, what is it for?
- An overview of UML
 - Concepts of Object-orientation
 - UML
 - Static Modeling
 - Dynamic Modeling
 - Tools

What is Modeling?

- Model : a simplification of reality
 - E.g.,
 - Architectural model of house
 - Mathematical model of Newton's law of motion
- Models of software system
 - In analysis stage, a description of a system that customers want (by functional requirements)
 - In design stage, a description of the structure of the software to be implemented

What is Modeling for?

- To identify/define **system elements** that can provide the specified functions of the system



Elements of Software System

- Two software development methodologies

Software Development Methodology	Types of Elements in the Modelling
Structured Methods	Function
Object-Oriented Methods	Class/Object

Software Development Methodology

- Structured Method (Function)

Structured Method

- SP: C, Pascal, Fortran
- SD: Structure Chart
- SA: DFD(Data Flow Diagram)

SP: Structured programming

SD: Structured design

SA: Structure analysis

cf. Modeling with

- 1) **Data flow diagram (DFD)**
- 2) **Data dictionary**

- Object-Oriented Method (Class/Object)

Object-Oriented Method

OOP: C++, Java, C#
OOD: UML
OOA: UML

OOP: OO programming

OOD: OO design

OOA: OO analysis

cf. Modeling with **UML**

Structured Analysis

- What is Structured Programming (구조적 프로그래밍)?

- Edsger W. Dijkstra (엡저 위베 다익스트라) :

- 프로그래밍을 수학적으로 분석하는 방법론을 제안한 최초의 컴퓨터 과학자
=> 컴퓨터분야 노벨상 튜링상(Turing Award) 수상

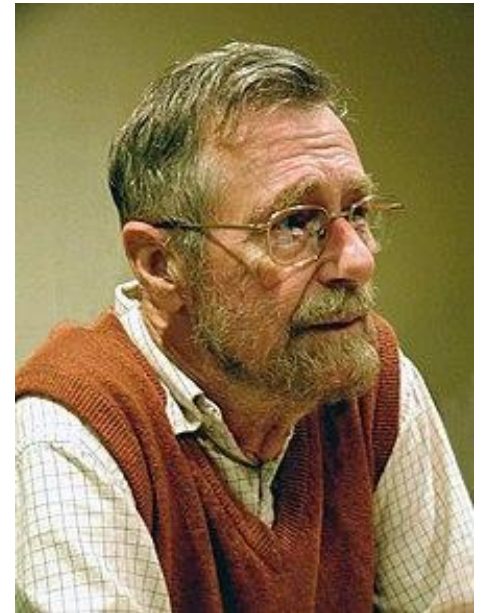
- “Go To Statement Considered Harmful”

- The structured program theorem

- Every program can be written as composed of control structures:

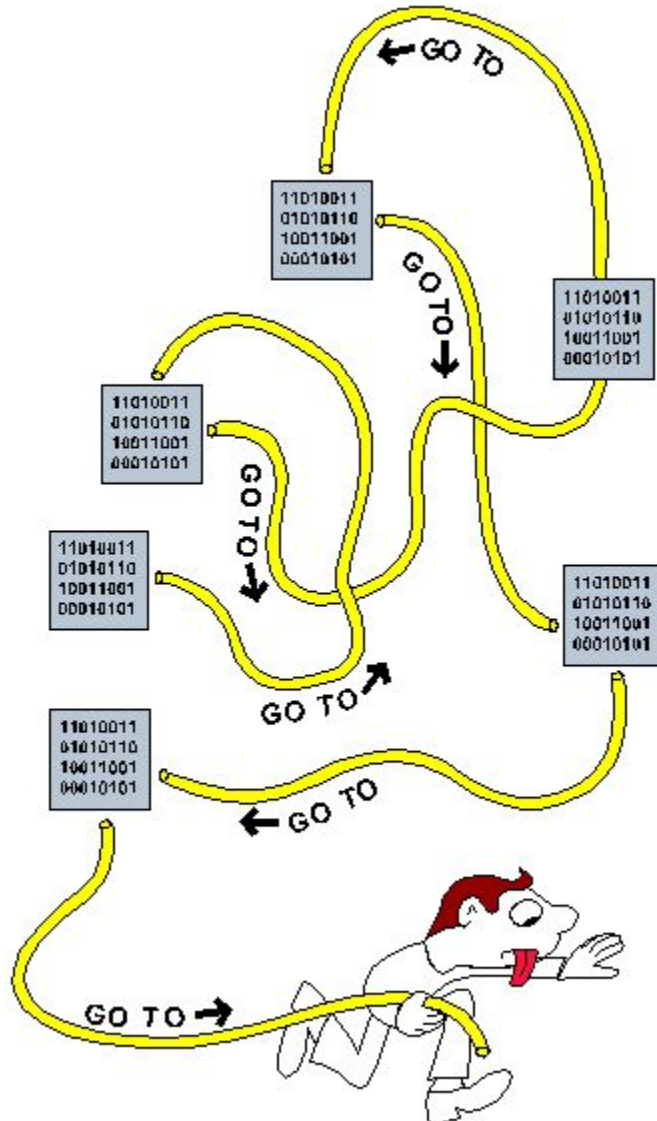
- Sequence (stmt1 ; stmt2 ; ...)
 - Selection (If-then-else)
 - Iteration (for, while)

- (without using Goto statements that could lead to *spaghetti code*)



https://en.wikipedia.org/wiki/Edsger_W._Dijkstra

Spaghetti Code(스파게티 코드)



A spaghetti code for calculating PI (writin in Fortran)
[Book] Computers and the Human Mind, Anchor Books

```

4      PROGRAM PI
5      DIMENSION TERM(100)
6      N=1
7      3  TERM(N)=((-1)**(N+1))*(4./(2.*N-1.))
8      N=N+1
9      IF (N-101) 3,6,6
10     6  N=1
11     7  SUM98 = SUM98+TERM(N)
12     WRITE(*,28) N, TERM(N)
13     N=N+1
14     IF (N-99) 7, 11, 11
15     11 SUM99=SUM98+TERM(N)
16     SUM100=SUM99+TERM(N+1)
17     IF (SUM98-3.141592) 14,23,23
18     14 IF (SUM99-3.141592) 23,23,15
19     15 IF (SUM100-3.141592) 16,23,23
20     16 AV89=(SUM98+SUM99)/2.
21     AV90=(SUM99+SUM100)/2.
22     COMANS=(AV89+AV90)/2.
23     IF (COMANS-3.1415920) 21,19,19
24     19 IF (COMANS-3.1415930) 20,21,21
25     20 WRITE(*,26)
26     GO TO 22
27     21 WRITE(*,27) COMANS
28     STOP
29     23 WRITE(*,25)
30     GO TO 22
31     25 FORMAT('ERROR IN MAGNITUDE OF SUM')
32     26 FORMAT('PROBLEM SOLVED')
33     27 FORMAT('PROBLEM UNSOLVED', F14.6)
34     28 FORMAT(I3, F14.6)
35     END
36

```


Spaghetti Code(스파게티 코드) without Goto

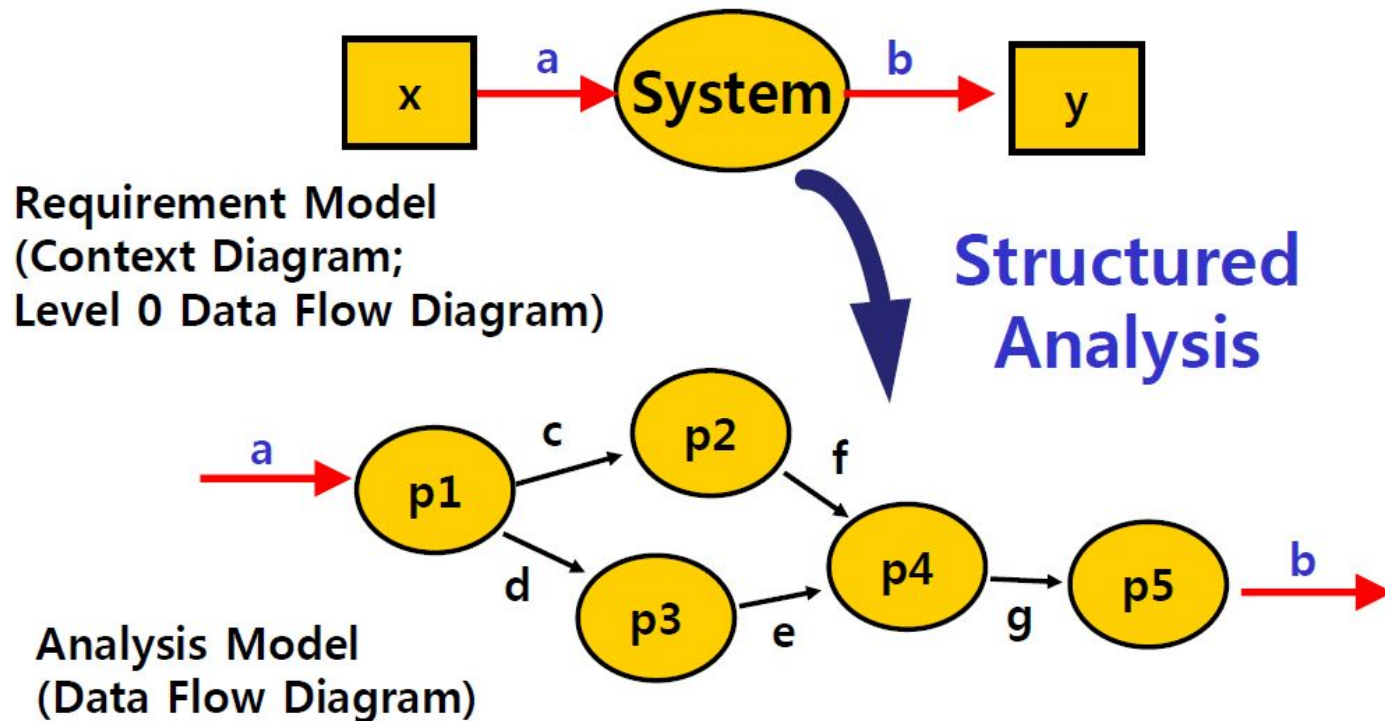
A simple "spaghetti" code example in Java. (We use spaghetti here in the sense of tangling different concerns not in the sense of having many goto statements.)

- [Book] Architectural Transformations: From Legacy to Three-Tier and Services

```
public void Transaction () {
    try {
        //Data access-processing code
        fis = new FileInputStream ("Bank.dat");
        ... //fetch some data from file
    }
    catch (Exception ex) {
        total = rows;
        //Validations and respective UI actions
        if (total == 0) {
            JOptionPane.showMessageDialog (null, "Records File is Empty.\nEnter
            Records First to Display.", "BankSystem - EmptyFile",
            JOptionPane.PLAIN_MESSAGE);
            btnEnable ();
        }
        else {
            try {
                //Data access-processing code
                fis.close();
            }
            catch (Exception exp) {
                ...
            }
        }
    }
}
```

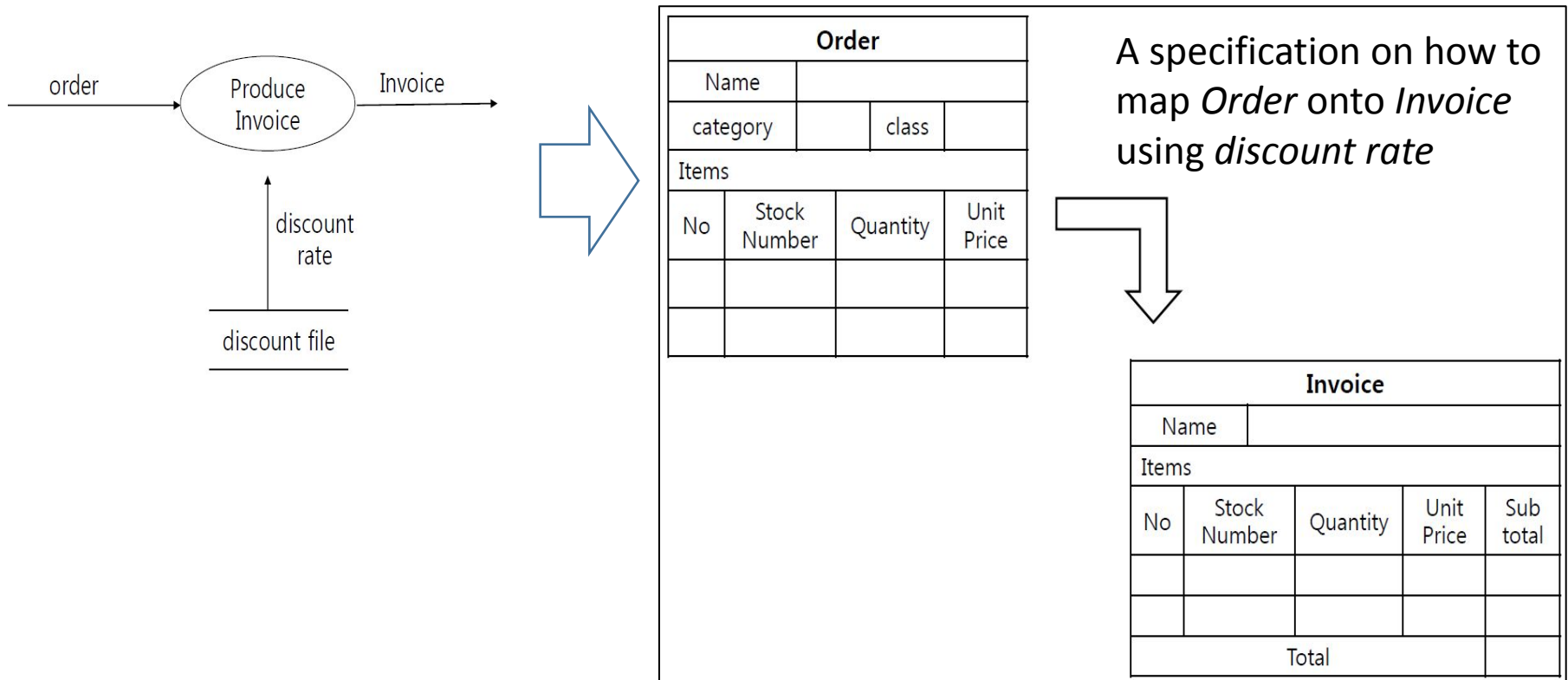
Structured Analysis

- Refine and decompose the system by identifying its basic functions
 - **Data flow diagram (DFD)**



Structured Analysis (cont.)

- Specification for each primitive process
cf. process (== function)



Structured Analysis (cont.)

- **Data dictionary**: definitions of data
 - Sequencing data types: e.g., name + category + class
 - Repeating data types { ... }
 - e.g., Items = { No + Stock Number + Quantity + Unit price }
 - Selecting one from several types
 - e.g., [vacuum cleaner order | jet engine order]
- Order = name + category + class + { no + stock number + quantity + unit price }

Order			
Name			
category		class	
Items			
No	Stock Number	Quantity	Unit Price

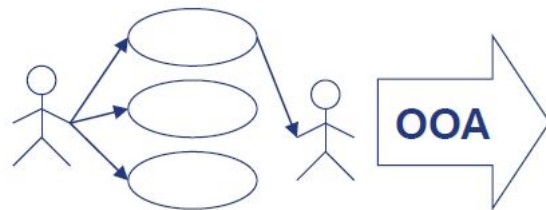
Object-Oriented Analysis

- What is object-orientation?
 - Data over function
 - Information hiding (or encapsulation)
 - Inheritance
- Objects and classes
 - An object is an *instance* of a class
 - A class is the *type* of objects
 - Object : attributes(state) + operation (methods)
 - Class : A set of objects that share the same attributes and operations
- Why OO?
 - Reduce maintenance costs
 - Enforce good design
 - Improve development process

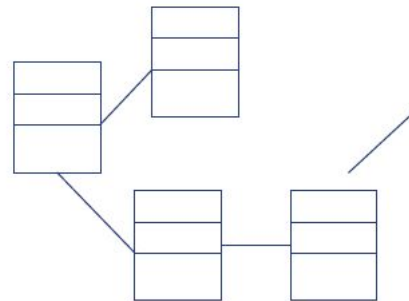
Object-Oriented Analysis

- Transformation of requirement into a precise description of the system in terms of classes and interactions among objects

[UML : Unified Modeling Language]

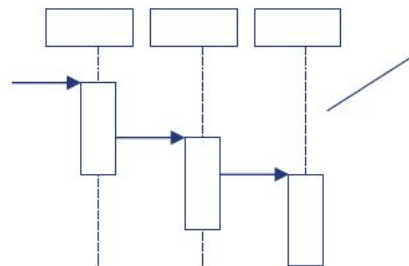


OOA: Object-Oriented Analysis



UML Class Diagram

What classes are required for the requirement ?

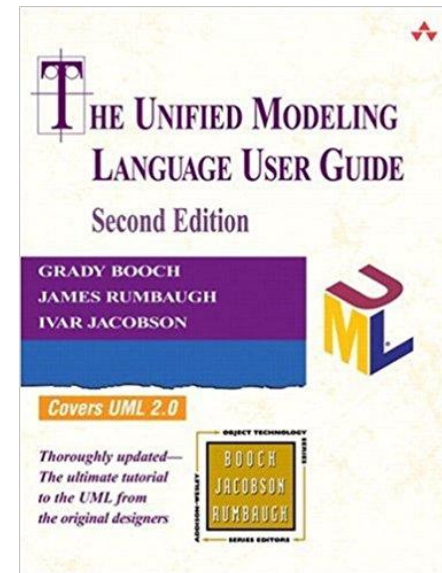


UML Sequence Diagram

How do they interact to realize the requirement ?

UML

- Unified Modeling Language
 - OMT (Object Modeling Technology, J. Rumbaugh)
 - OOSE (OO Software Engineering, I. Jacobson)
 - OOAD (OO Analysis and Design, G. Booch)
- Unification after “Method Wars” of object-oriented paradigm in 1990s
- UML 2.5 (Oct. 2012 ~)



Object-Oriented Analysis: History

Object Modeling Technique (OMT)



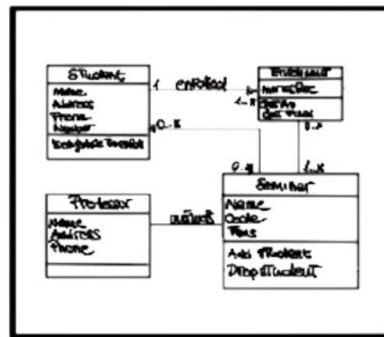
Rumbaugh



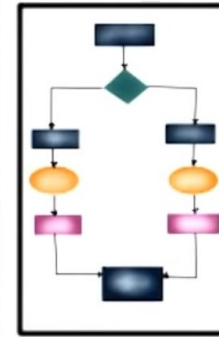
Jacobson



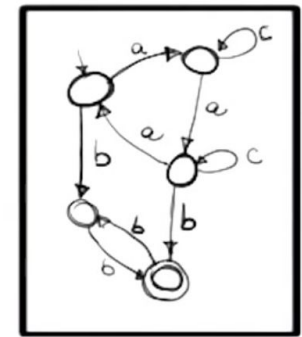
Booch



Data



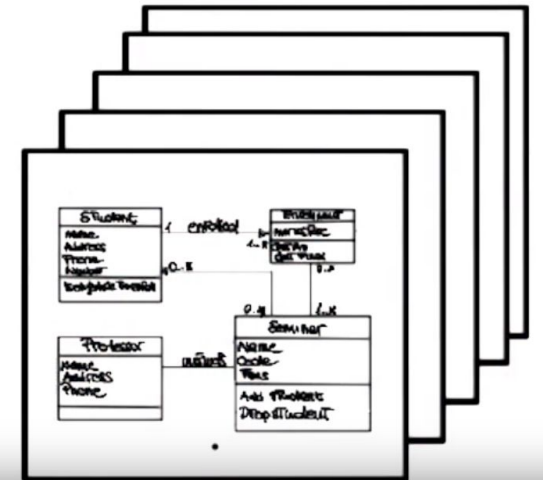
Functions



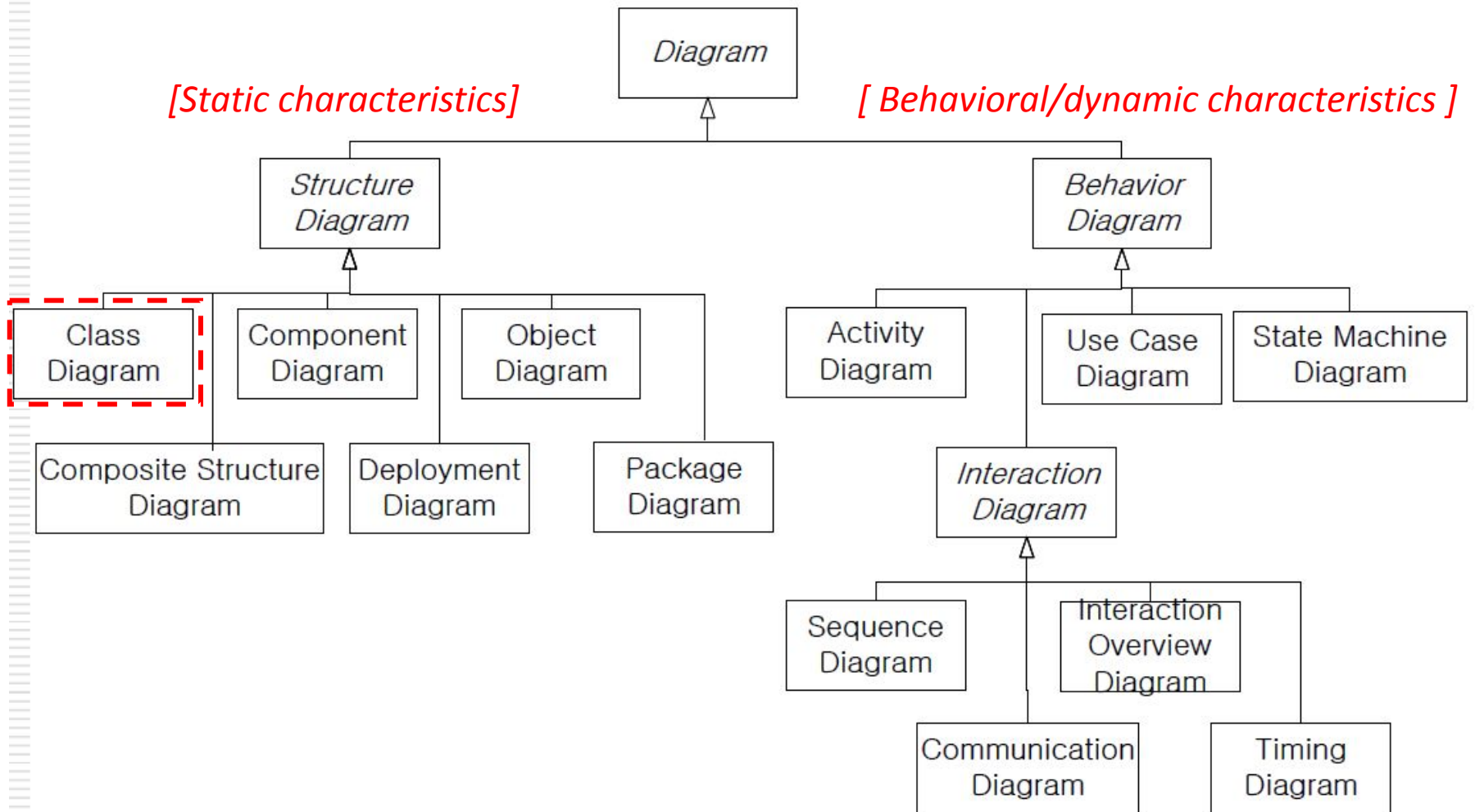
Control



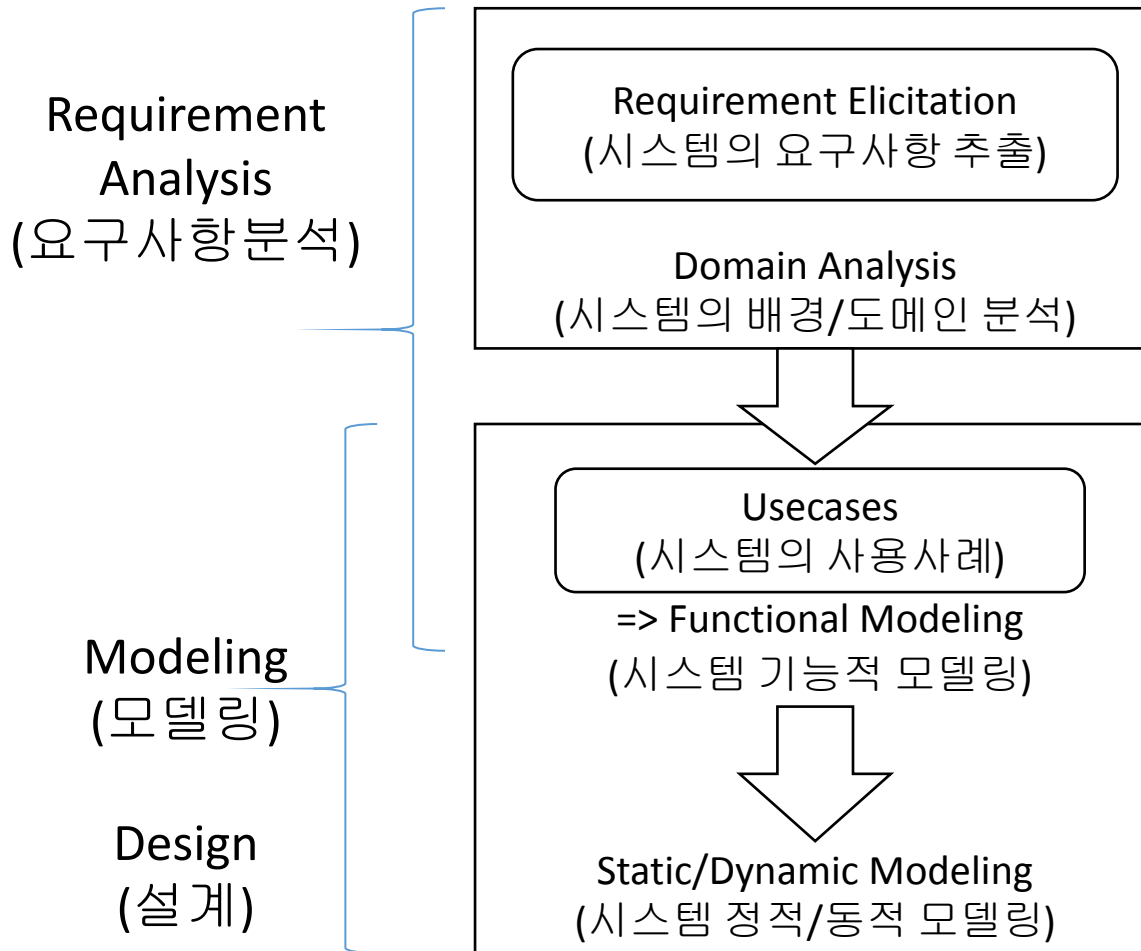
Unified Modeling Language (UML)



UML Diagrams



Roadmap



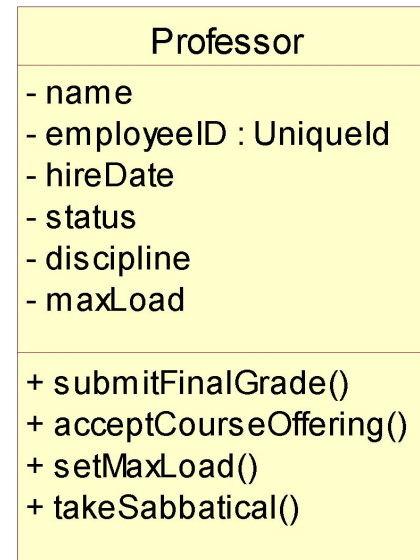
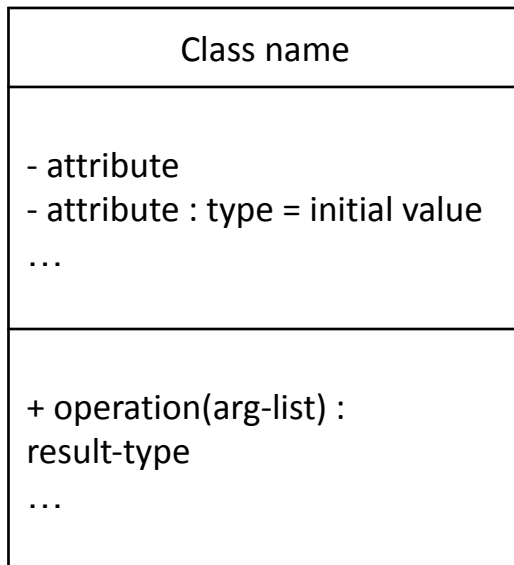
Class Diagram

- Static, structural view of the system with
 - Classes
 - Relationships among classes
 - Association, Aggregation/Composition : *X has a Y*
 - Inheritance (Generalization), Realization : *X is a Y*
 - Dependency : *X uses a Y*

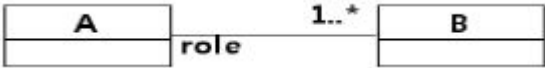


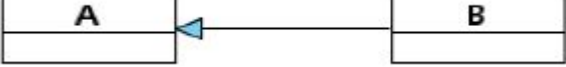
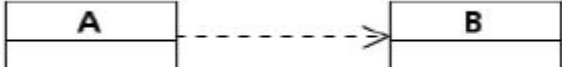
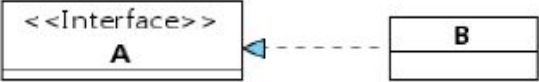
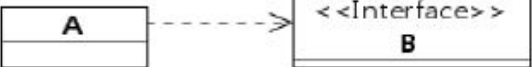
메모 참고 (Java를 예시로 association, inheritance, dependence를 설명)

Class Diagram: Class

- Attributes represent the structure of a class
 - found by examining class definitions, studying requirements, and applying domain knowledge
- Operations represent the behavior of a class
 - found by examining interactions among entities

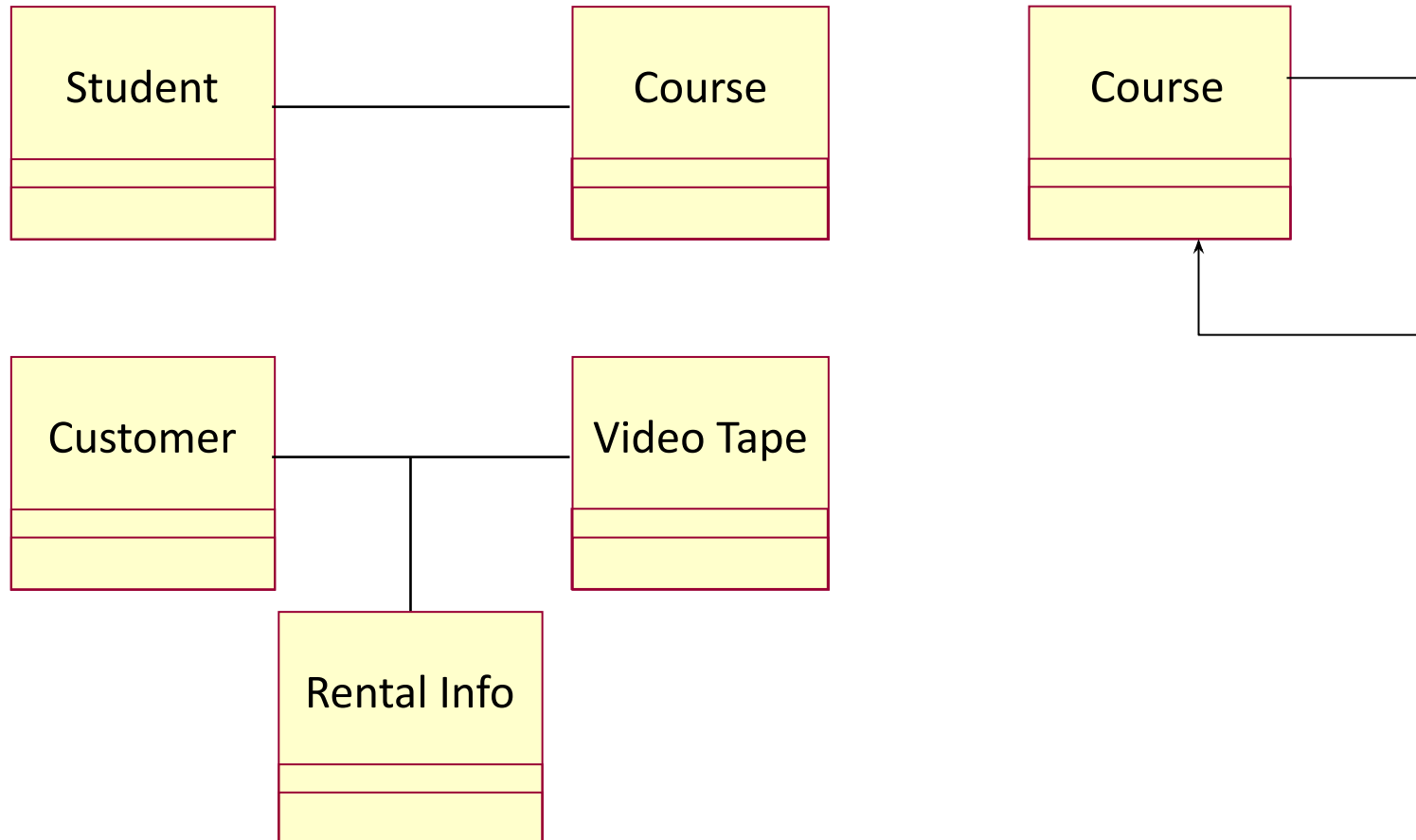


Class Diagram: Relationships among classes

Relationship among Classes	Notation
Association (연관관계)	
Aggregation (포함관계)	
Composition (합성관계)	
Inheritance (상속관계)	
Dependence (의존관계)	
Interface Realization (인터페이스 실현관계)	
Interface Dependence (인터페이스 의존관계)	

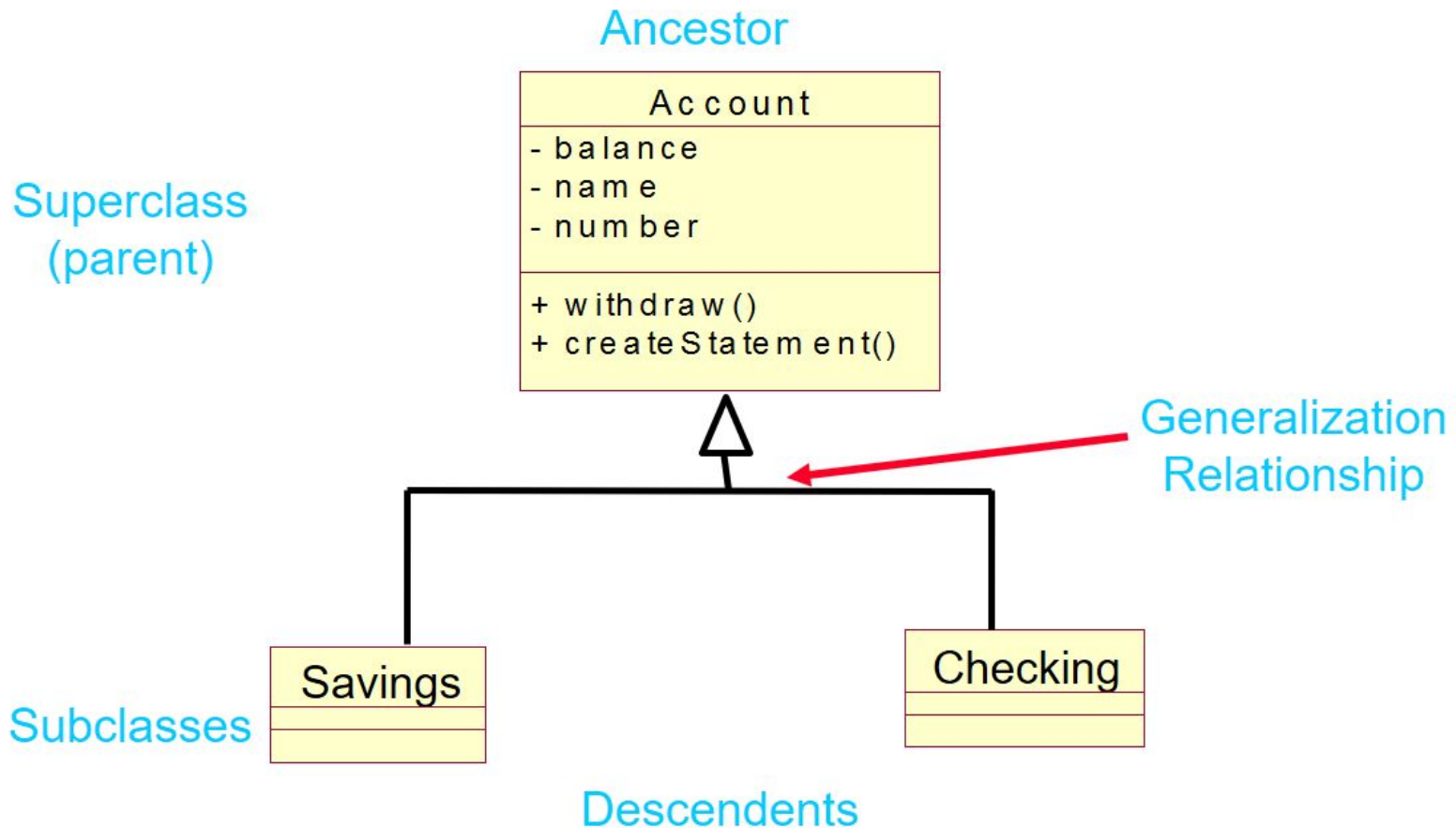
Association

- A relationship between two or more classes that specifies connections among their instances



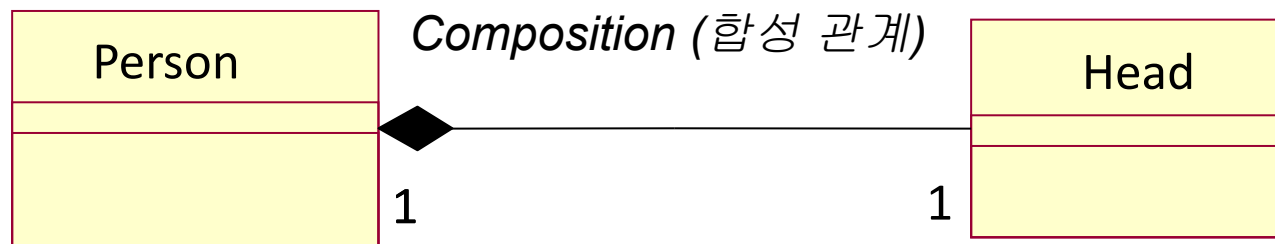
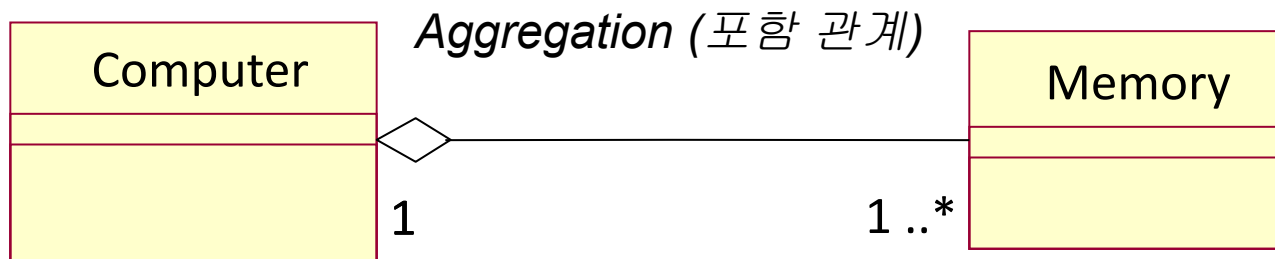
Inheritance

- Single inheritance



Aggregation/Composition

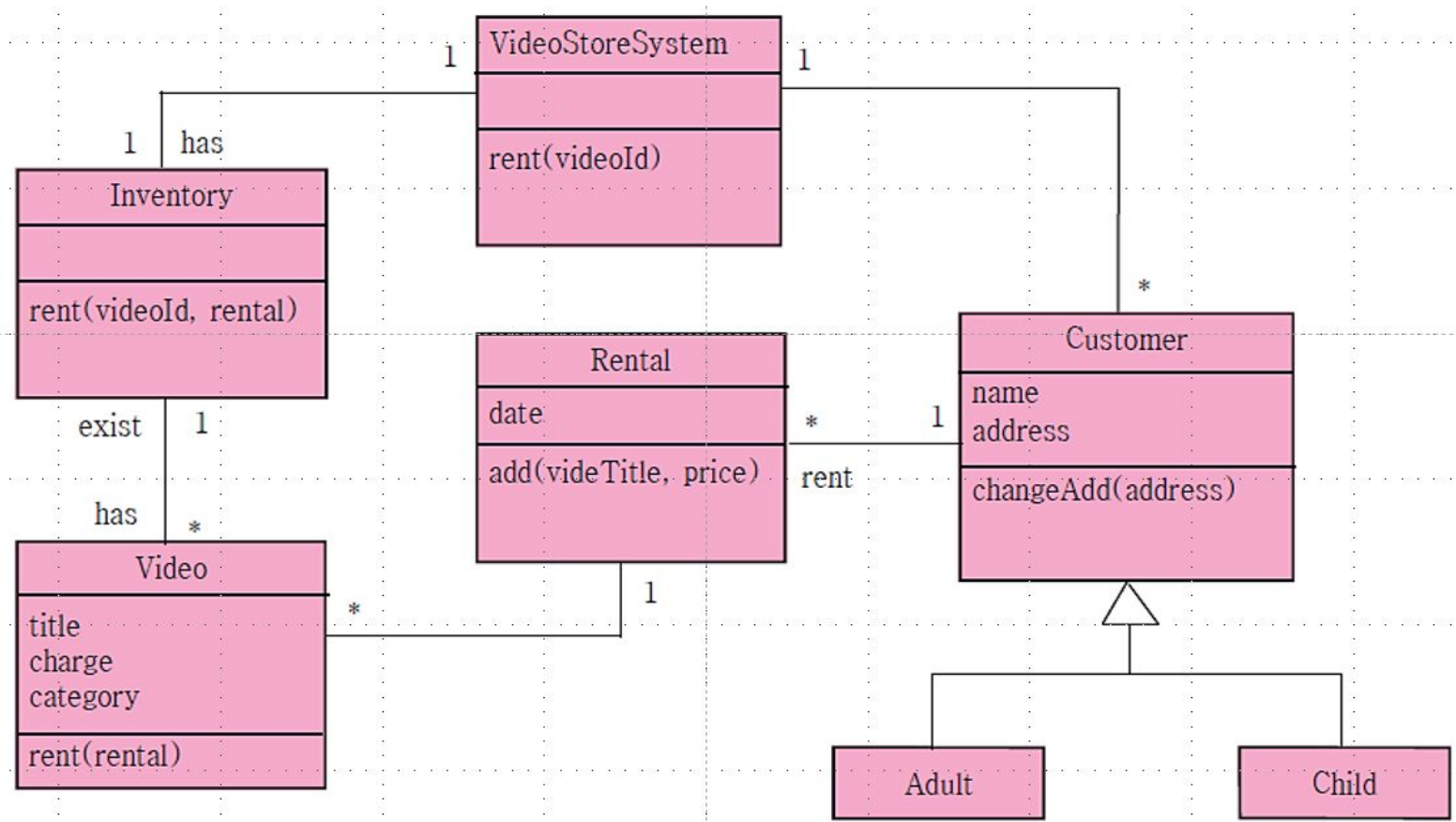
- *A special form of association*
 - a whole-part relationship between an aggregate (the whole) and its parts
 - “Is a part-of” relationship



Class Diagram Example

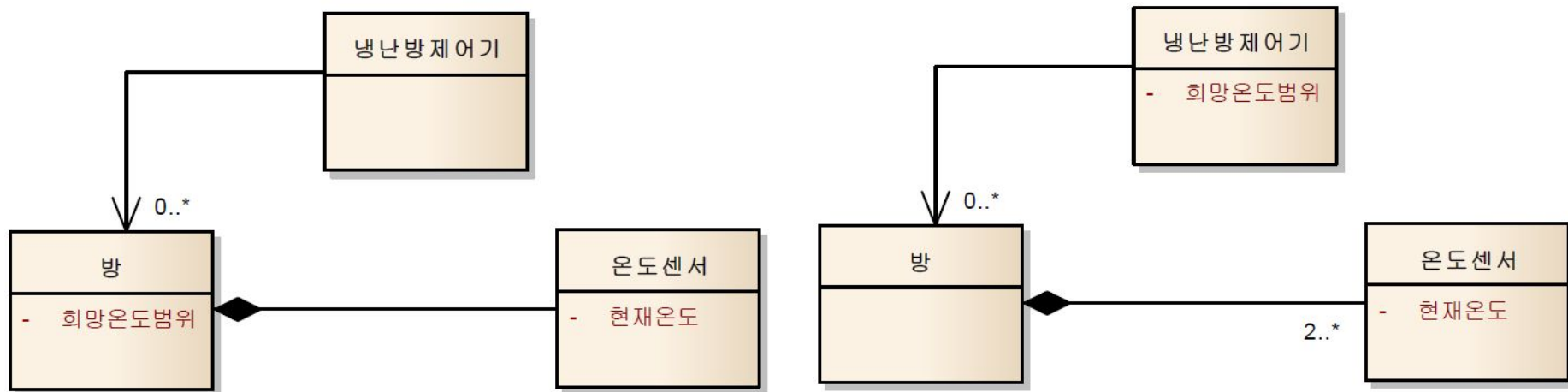
Requirements :

- The video store system can show a list of videos that a specified person rents.
- The video store system can show who rents a particular video.

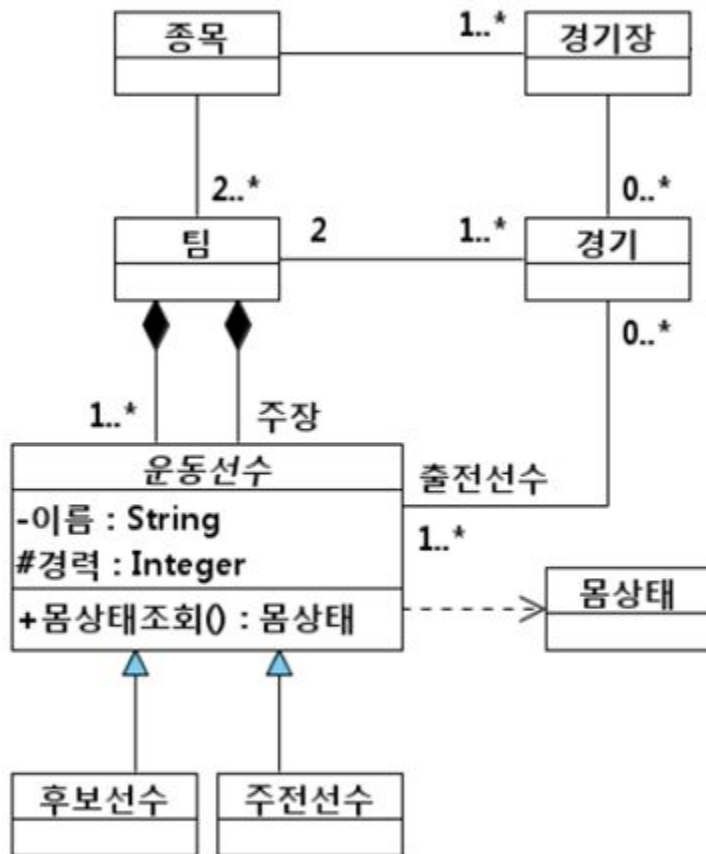


Class Diagram Example

- Explain a difference between two class diagrams



Class Diagram Example



```

class 팀 {
private:
    운동선수 주장 ;
    vector<운동선수> the운동선수 ;

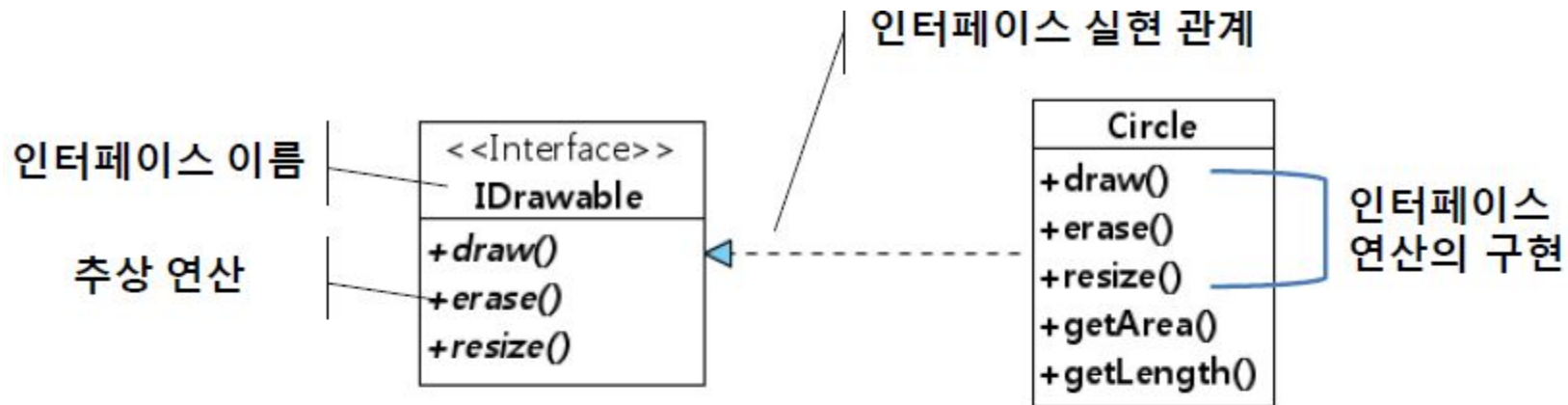
    종목* the종목 ;
    vector<경기*> the경기 ;
};

class 운동선수 {
private: string 이름;
protected: int 경력;
public: 몸상태 몸상태조회() ;

private: vector<경기*> the경기 ;
};

class 후보선수: public 운동선수 { ... };
class 주전선수: public 운동선수 { ... };
    
```

Class Diagram Example



```
class IDrawable {  
public:  
    virtual void draw() = 0 ;  
    virtual void erase() = 0 ;  
    virtual void resize() = 0 ;  
};
```

```
class Circle : public IDrawable {  
public:  
    void draw() { ... }  
    void erase() { ... }  
    void resize() { ... }  
    void getArea() ;  
    void getLength() ;  
};
```

How To Write Class Diagram

- Identifying Classes
 - *[TIP] Abott's technique*
 - Given a (textual) description of problem (e.g. SRS)
 - Nouns(명사) => classes
 - Adjectives(형용사) => attributes
 - Active verbs(능동태 동사) => operations
- Specifying attributes and operations
- Finding associations among classes

Example for Class Diagram

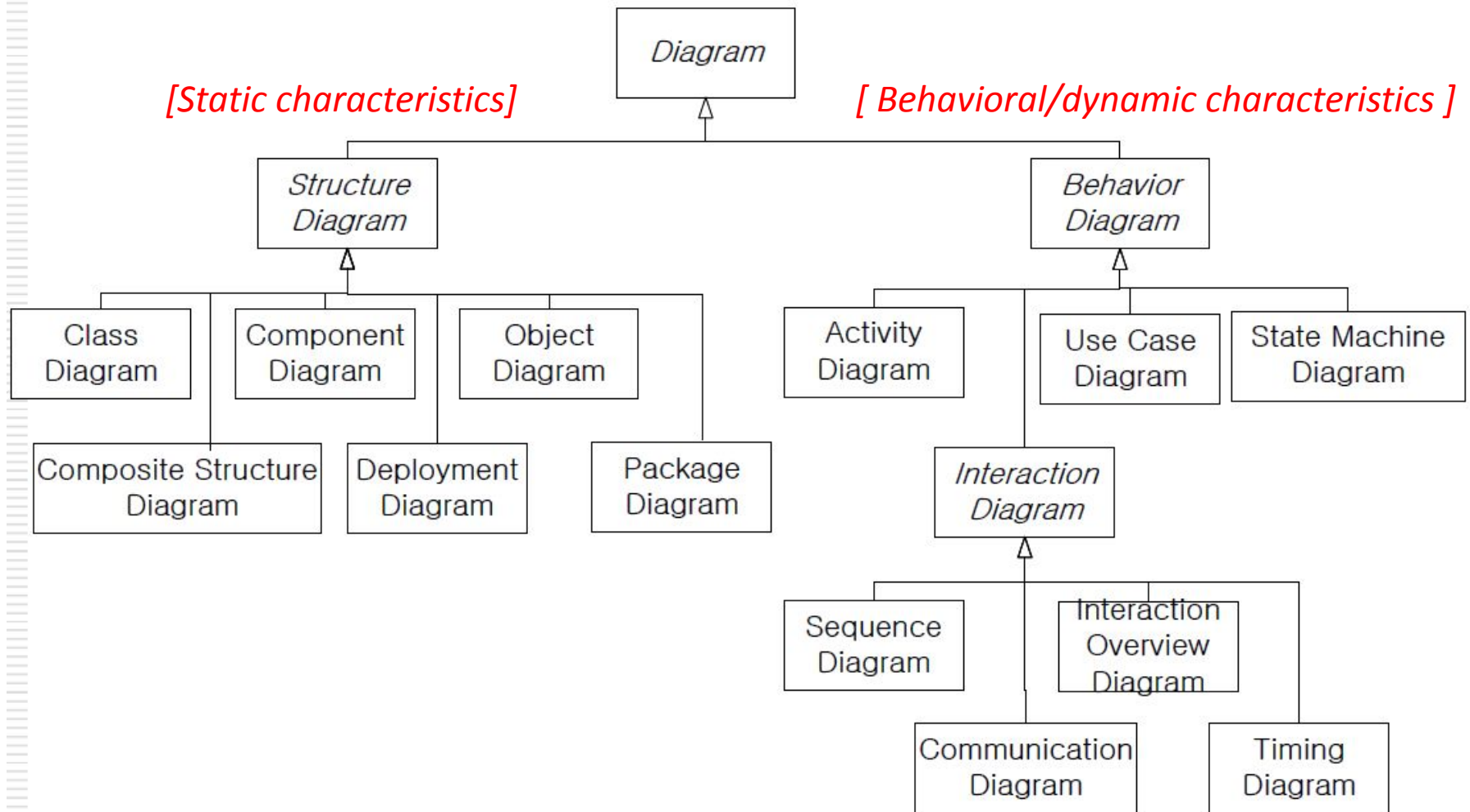
1. In the flight reservation system, customers view and book flights.
(예매 시스템으로 고객은 항공권을 조회/예약한다.)
2. The airline opens new regular flights and operates them.(항공사는 새로운 항공편을 개설하고 운영한다.)
3. The customers search a flight and book seats with the accompanying passengers.(고객은 항공편을 찾고 동반자와 좌석을 예약한다.)
4. Customers can also cancel and confirm the flight bookings.(고객은 항공 예약을 취소하고 예약할 수 있다.)
5. The flight has a schedule, i.e., a departure airport, an arrival airport, the departure time, and the arrival time.(항공편 일정은 출발/도착 공항 및 시간을 포함한다.)
6. Every airport covers several cities around.(공항은 주변 여러 도시들과 연관되어 있다.)
7. Some flights have a stopover.(어떤 항공편은 중간 기착지가 있다.)

Example for Class Diagram

HowTo: Class Diagram

- Types of Classes
 - Entity class : persistent things
 - Boundary class : the interface with users
 - Control class
 - the creation or update of entity objects
 - the instantiation of boundary object

UML Diagrams Again



Structure Diagram

<https://docs.staruml.io/>

Diagram	Description	Note
Class diagram	Describe the structure of a system by classes (시스템을 구성하는 클래스들)	Logical level
Object diagram	Describe the structure of a system by objects (시스템을 구성하는 객체들)	
Package diagram	Organize a system with packages that make up a model (모델을 구성하는 패키지들로 시스템을 구성)	
Component diagram	Describe the structure of a system by logical components (시스템을 구성하는 컴포넌트들)	
Composite diagram (복합구조 다이어그램)	Describe the internal structure of components with parts and connectors(컴포넌트 내부 구성을 파트와 연결자로 표현)	
Deployment diagram (배치 다이어그램)	Describe the deployment of physical components that make up a system (시스템을 구성하는 물리적 컴포넌트의 배치)	Physical level

Behavior Diagram

<https://docs.staruml.io/>

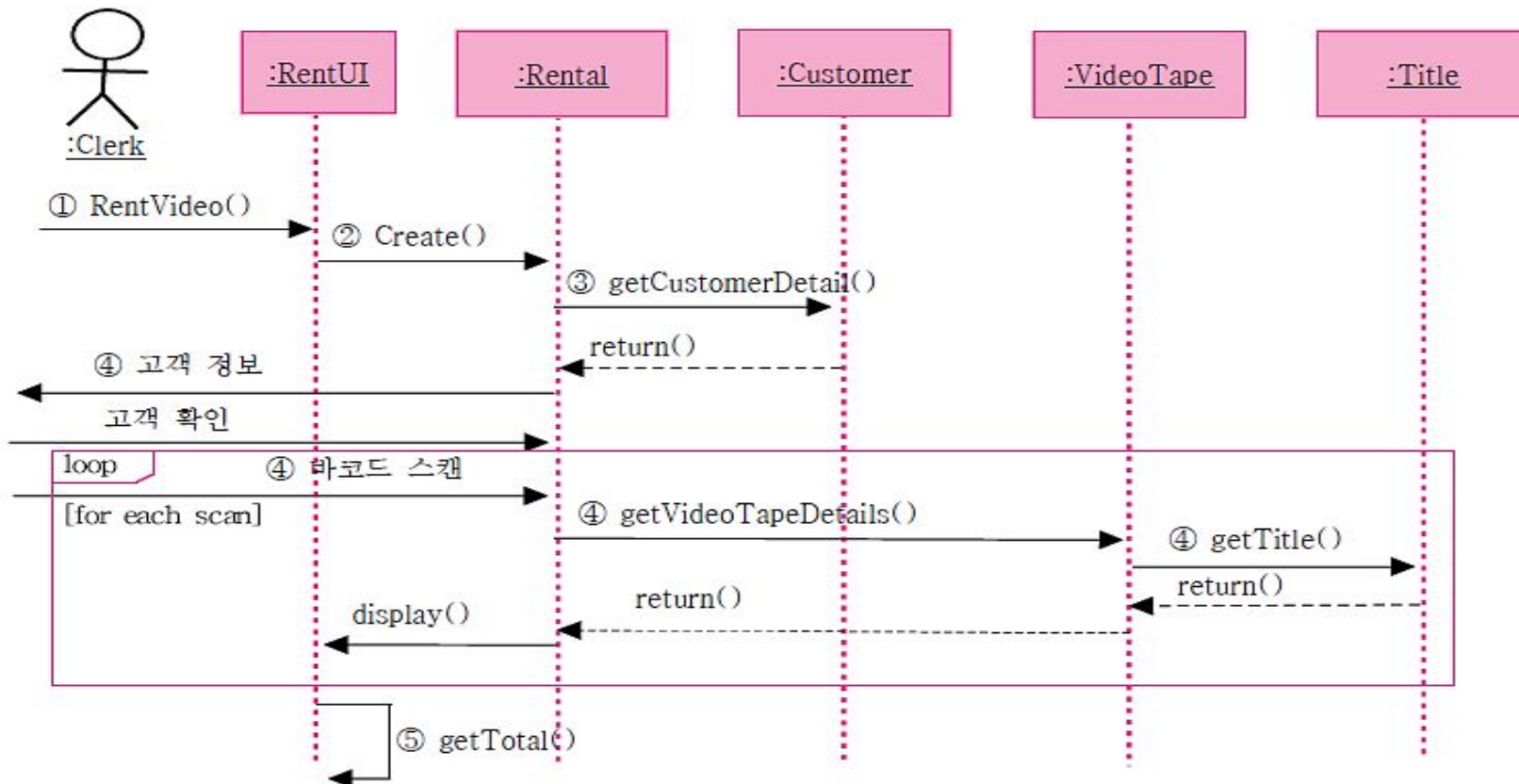
Diagram	Description	Note
Usecase diagram	User's interaction with a system (시스템과 사용자의 상호작용)	Behavior of system
State diagram	The dynamic behavior of a particular element by states and transitions(상태와 전이로 표현한 특정 요소의 동작)	Behavior of individual elements
Activity diagram	Describe the workflow of activities with support for choice, iteration and concurrency (액티비티 흐름을 표현)	
Sequence diagram	Describe interactions among objects in time sequence (객체들 간의 상호작용을 시간 순서로 표현)	Interactions among elements
Communication diagram	Describe interactions among objects, the same as sequence diagram, but focusing on the network of the objects (객체들 간의 네트워크를 강조)	
Timing diagram	The dynamic behavior of elements in terms of time(구성 요소의 상태 변화를 시간에 관하여 표현)	

5.4 Dynamic Modeling

- Sequence Diagram
(Communication Diagram)
- State Diagram
- Activity Diagram

Sequence Diagram

- Modeling the interactions between actors and system objects, and the interactions between system objects

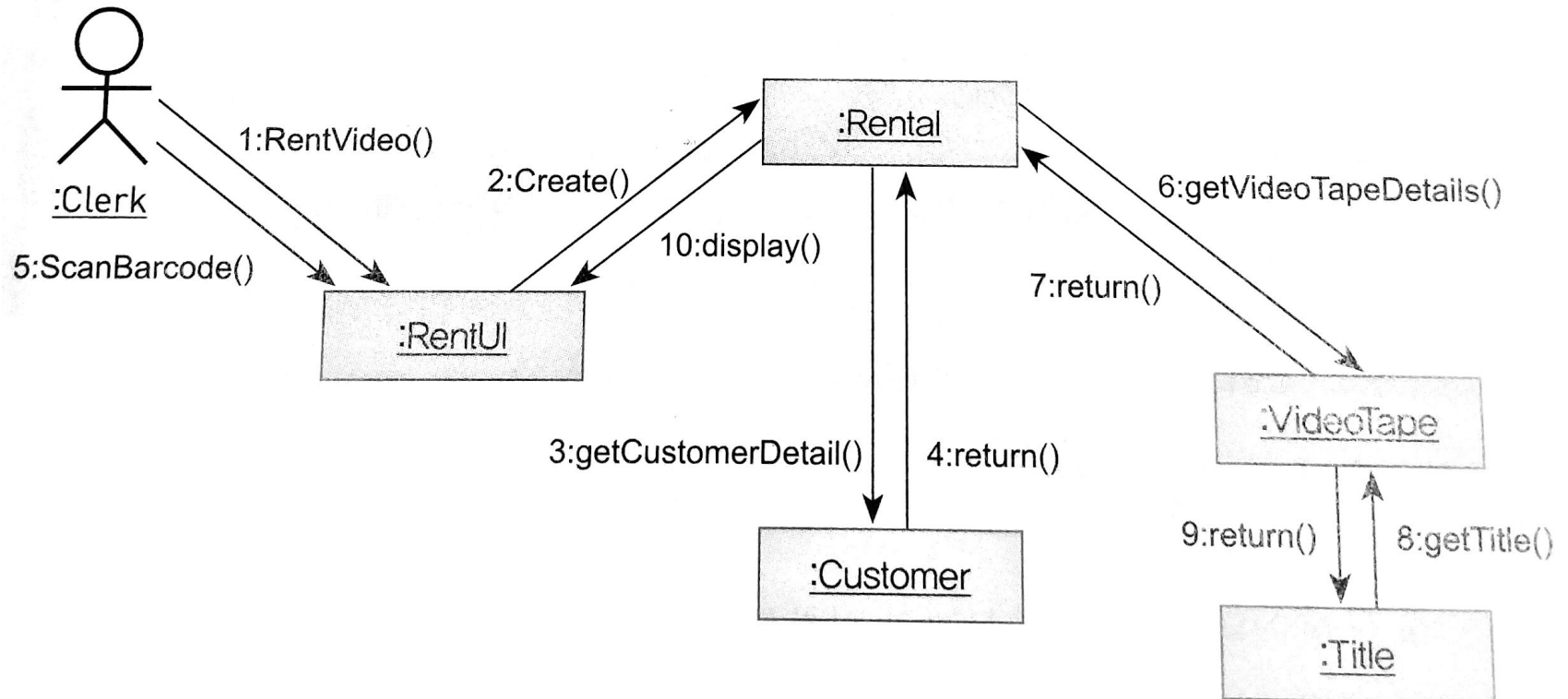


Sequence Diagram

- Notation
 - Participating objects (교재 그림 5.31)
 - Elements in sequence diagrams (교재 표 5.3)
 - Loops and alternatives (그림 5.32)

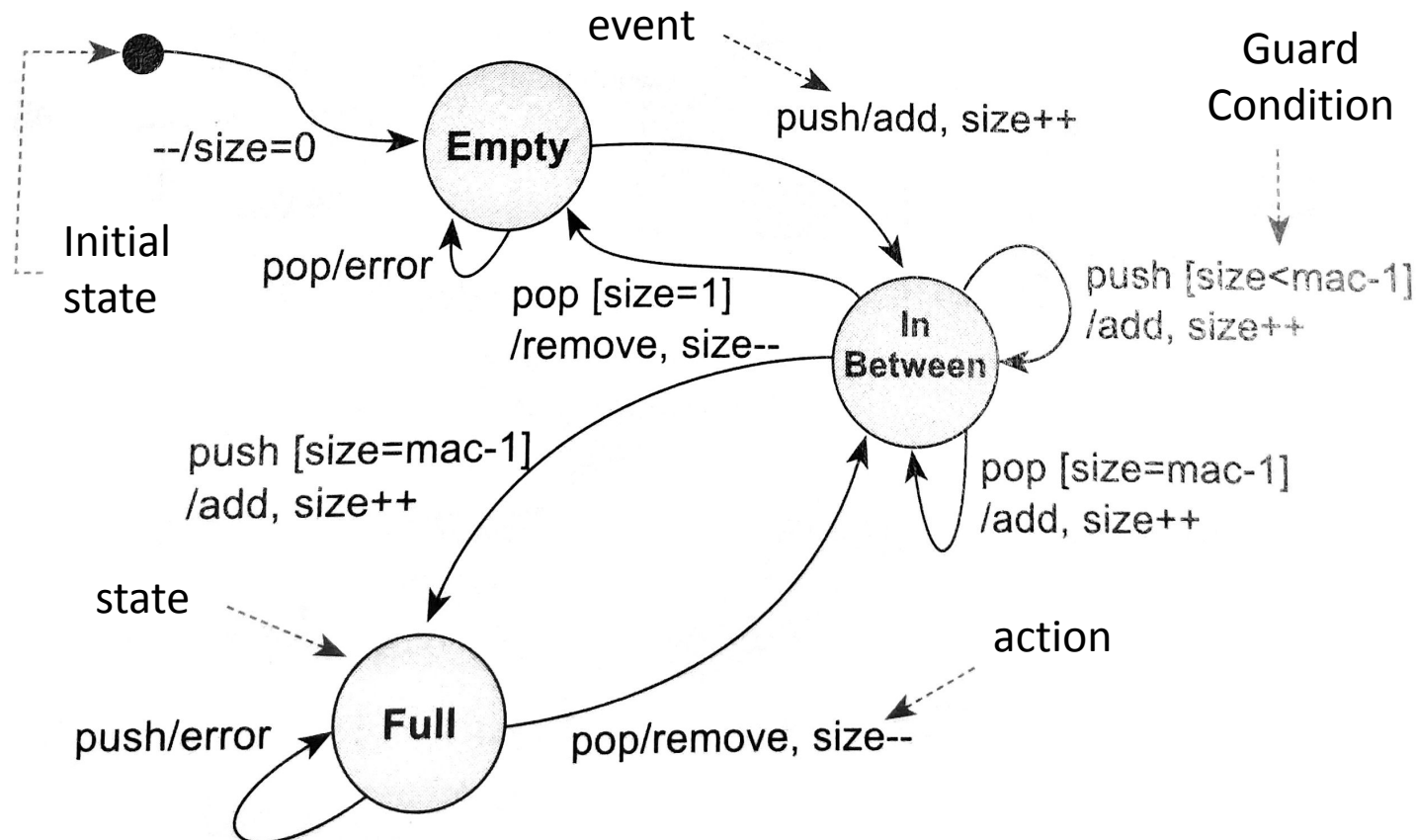
Communication Diagram

- Interactions among actors and objects using sequenced messages



State Diagram

- Modeling system states and events that cause transitions from one state to another
 - E.g., stack (push,pop)



State Diagram

- Example: Word counting

The C Programming Language 2nd Ed. by Kerninghan & Ritchie (Ch.1, Word Counting)

입력 받은 단어의 수를 세는 프로그램을 작성해보자. 단어는 공백(' ', '\t', '\n')으로 구분된 알파벳, 기호, 숫자들이다.

"Yonsei University, Wonju Campus"를 입력하면 단어는 총 4개 "Yonsei", "University,", "Wonju", "Campus"다.

프로그램의 기본 구조는 입력이 끝날 때까지 반복해서 단어를 세는 것이다.

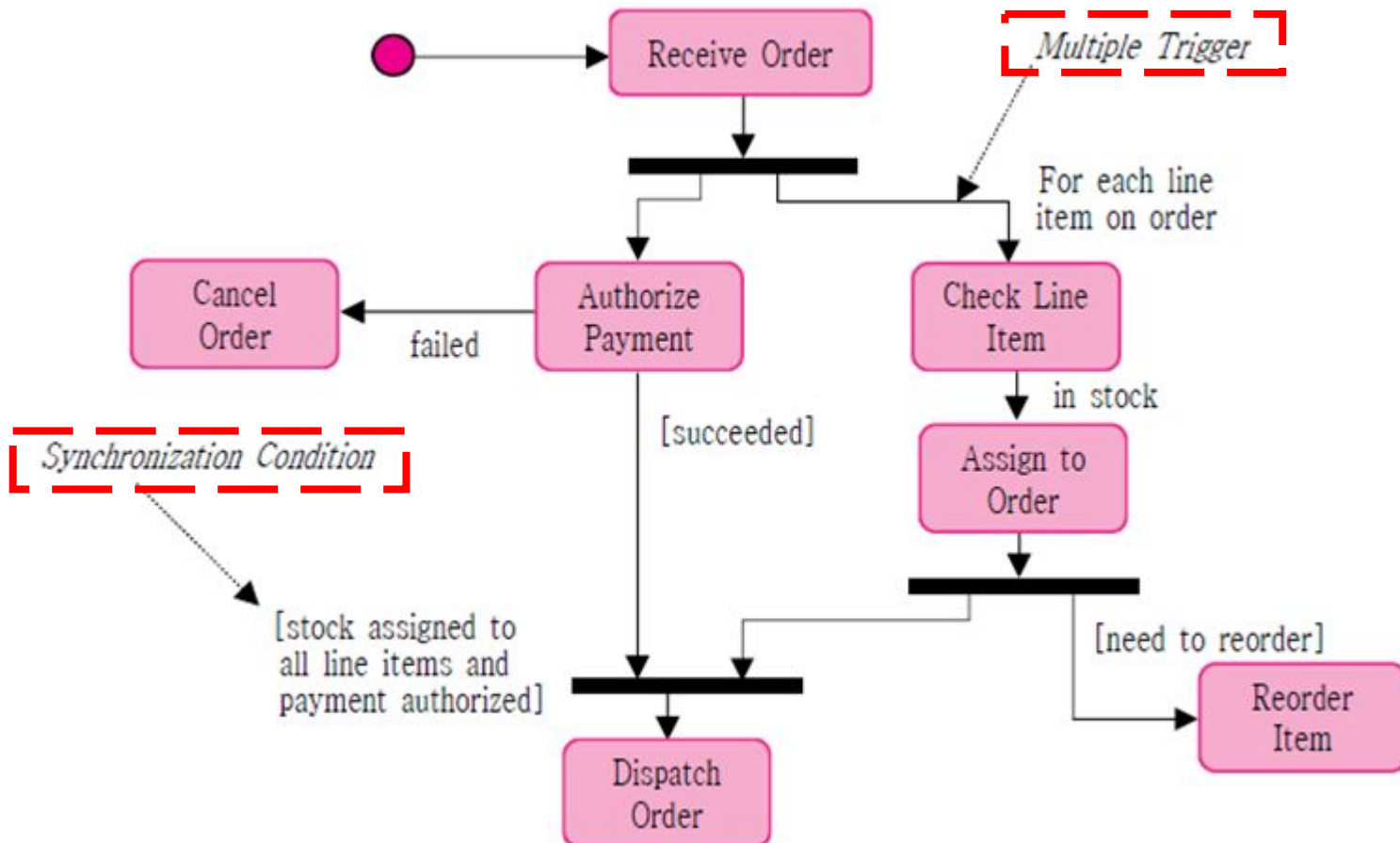
```
int c;  
int nw;  
  
nw = 0;  
c = getchar();  
while ( c != EOF ) {  
    // 단어 세기 ??  
    c = getchar();  
}  
  
printf("# of words : %d \n", nw);
```

State Diagram

- Example: Word counting (계속)

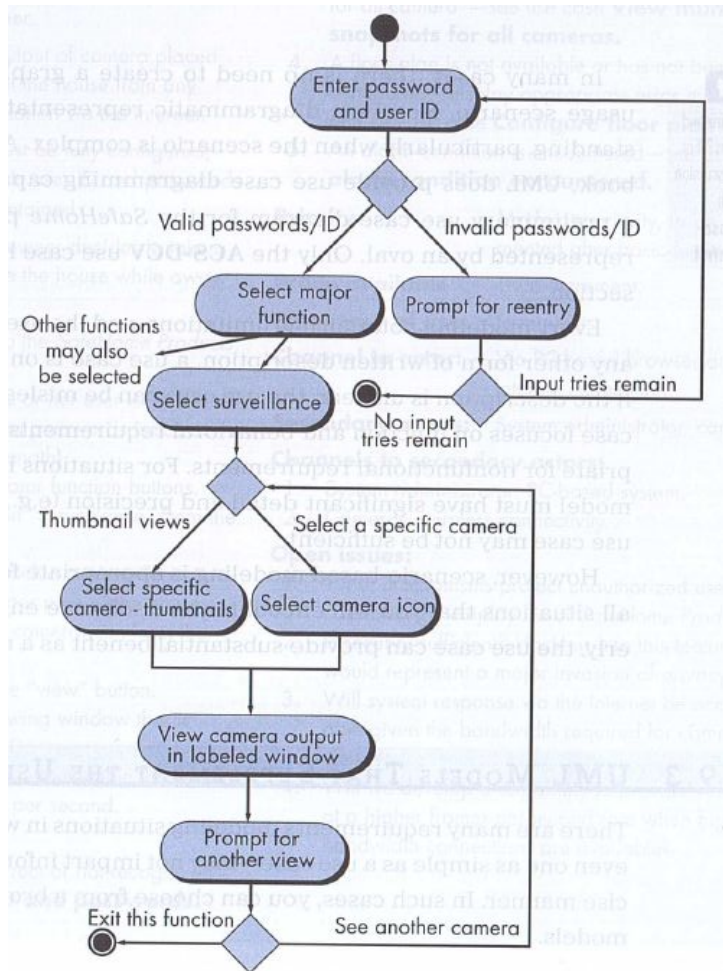
Activity Diagram

- Modeling activities in a process and the flow of control from one activity to another

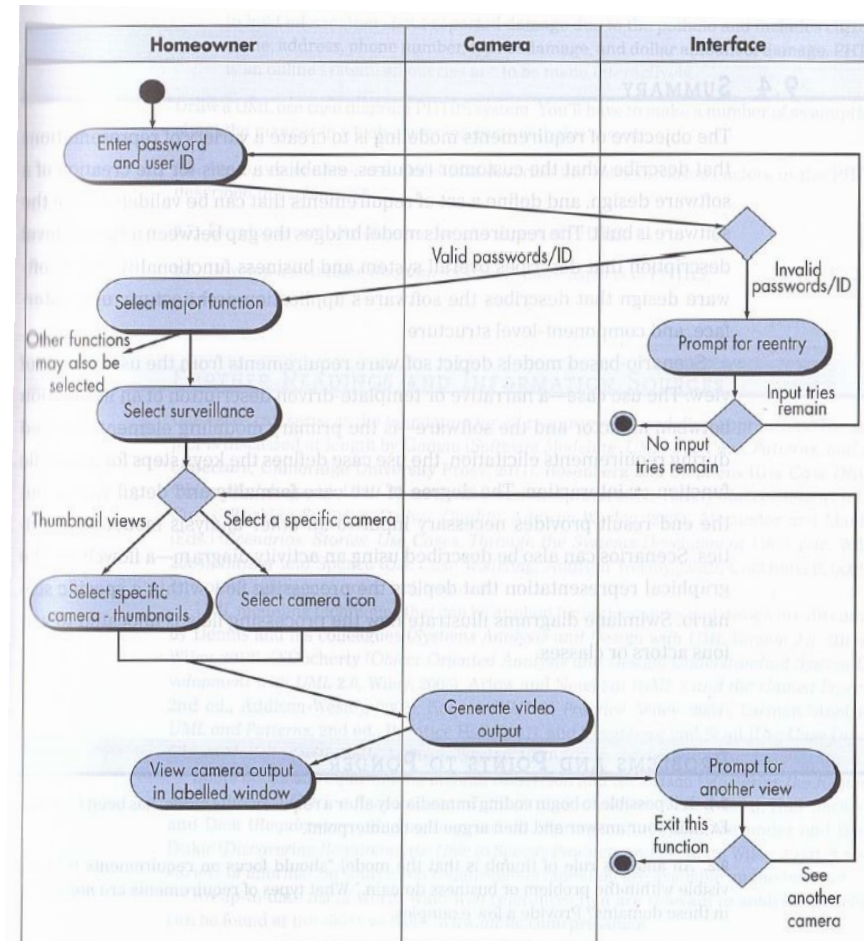


Activity Diagram

- Partition activities by actors or organizations



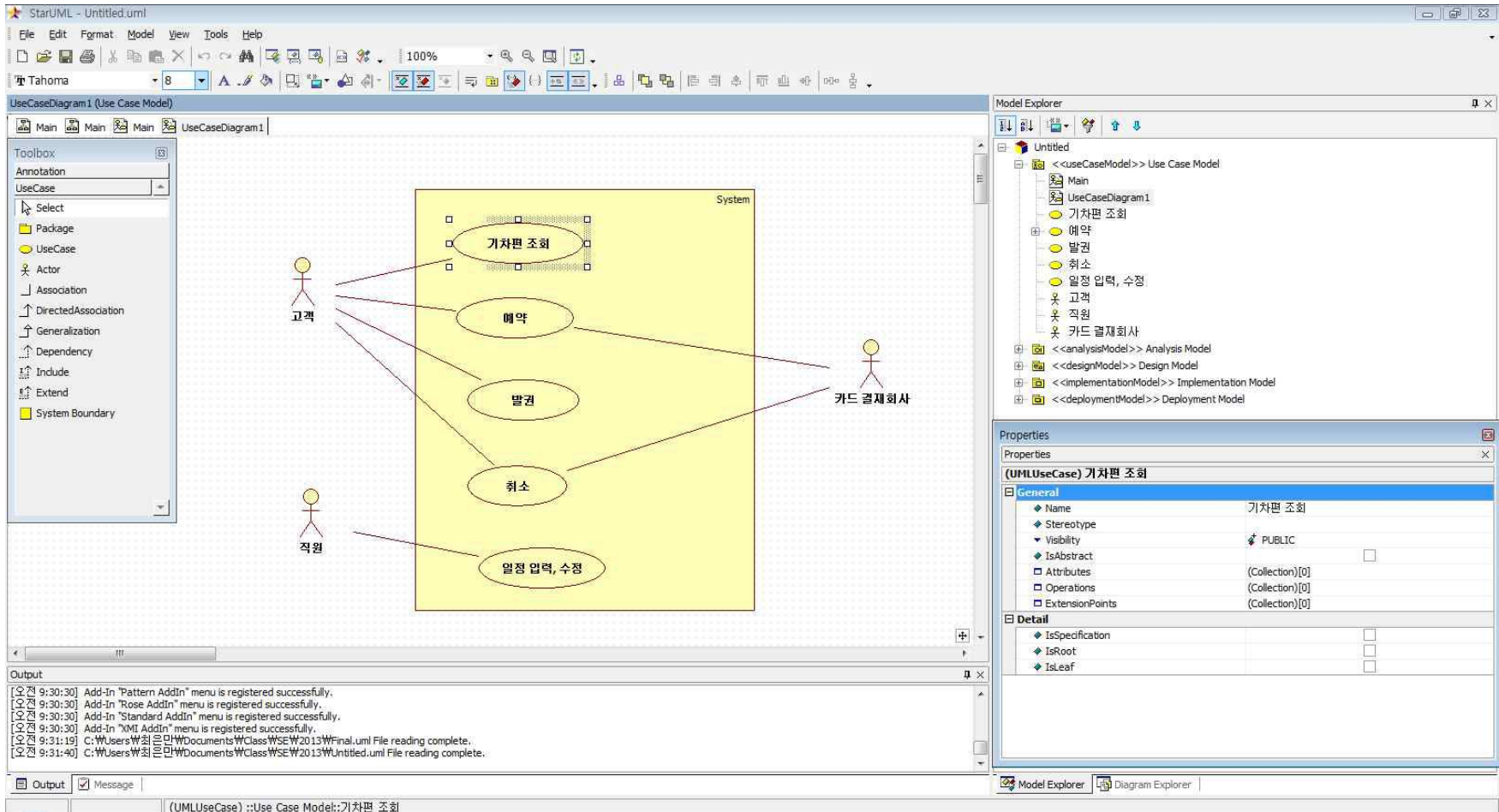
VS.



UML swimlane diagram

5.5 UML Tools

- StarUML



5.5 UML Tools

- StarUML
- ArgoUML
- Enterprise Architect
- IBM Rational Software Architect
- Rhapsody
- Tau
- Visual Paradigm
- Magic Draw

Main Features of UML Tools

- UML Diagramming
- Document Generation
- Forward / Backward Engineering (C, C++, Java)
- Audit
- Traceability
 - From use cases to analysis class, components, source codes

Summary

- UML notation for modeling
- Static modeling with class diagram
- Dynamic modeling with sequence diagram, state diagram, activity diagram