OO Development Process Using UML

OO_based SW Developing Process

- **Inception(준비)**

  목표: 기존의 시스템을 update하거나 새로운 시스템 개발을 위한 business case를 구축

  Step:

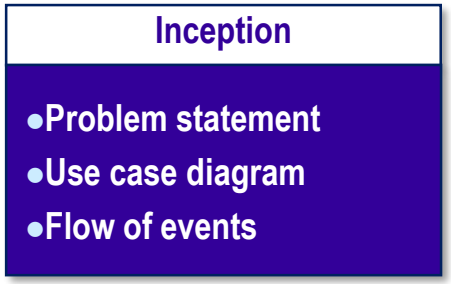
  – Problem statement

  – Use case diagram

    - Actor와 Use case를 추출하여 Use case diagram 작성

  – Flow of events

    - 각 Use case 항목 마다 Flow of events 작성
    - 주요사항
    - 예외사항



- **Elaboration**

목표: Problem domain 분석, **architectural foundation구축**, 프로젝트에서 가장 위험(risk)이 많은 부분을 강조, 프로젝트를 성공적으로 수행할 수 있는 초기 버전 구축.

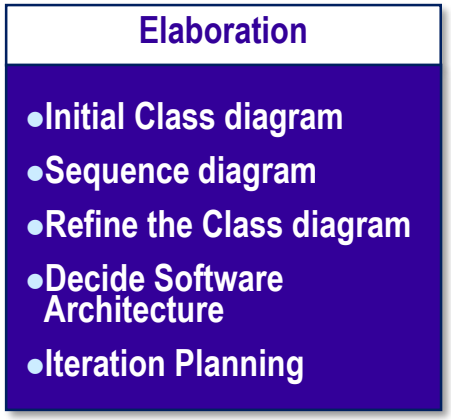
Step:

– Initial Class diagram

  - Class 도출
  - Initial Class diagram 작성
  - Package diagram 작성
    – Class들을 Grouping 하여 Package diagram을 작성

– Sequence diagram

  - Flow 별로 sequence diagram 작성
  - Collaboration diagram 작성→ 객체 사이의 응집력을 알 수 있음.



- Refine the Class diagram
  - Class의 attribute, operation
    - Sequence diagram의 Object에서 Class
    - Sequence diagram의 message에서 Attribute와 Operation
  - Class 사이의 relationship
    - Association, Inheritance, multiplicity, navigation 등
- Decide Software Architecture
  - Collaboration diagram
  - Component diagram
  - Deployment diagram
- **Iteration Planning**

Elaboration

- **Initial Class diagram**
- **Sequence diagram**
- **Refine the Class diagram**
- **Decide Software Architecture**
- **Iteration Planning**

- **Construction**

  목표: 소프트웨어를 **Iteration Planning에 맞추어 단계적으로 개발**

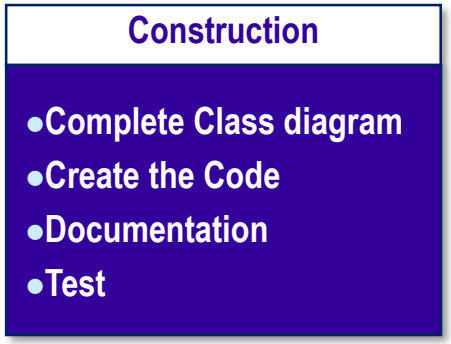
  Step:

  – Identify the classes and relationships to be implemented.

  – Complete the design for the selected classes and relationships

    - Data types for attributes.
    - Operation signatures.
    - Addition of other operations
    - Addition of implementation level classes
    - Specification of association/aggregation design decisions
    - Specification of inheritance design decisions

  – Create the code for the iteration



– Create/update the documentation for the iteration

– Test the iteration

  • Test the functionality specified in the use cases implemented in the iteration

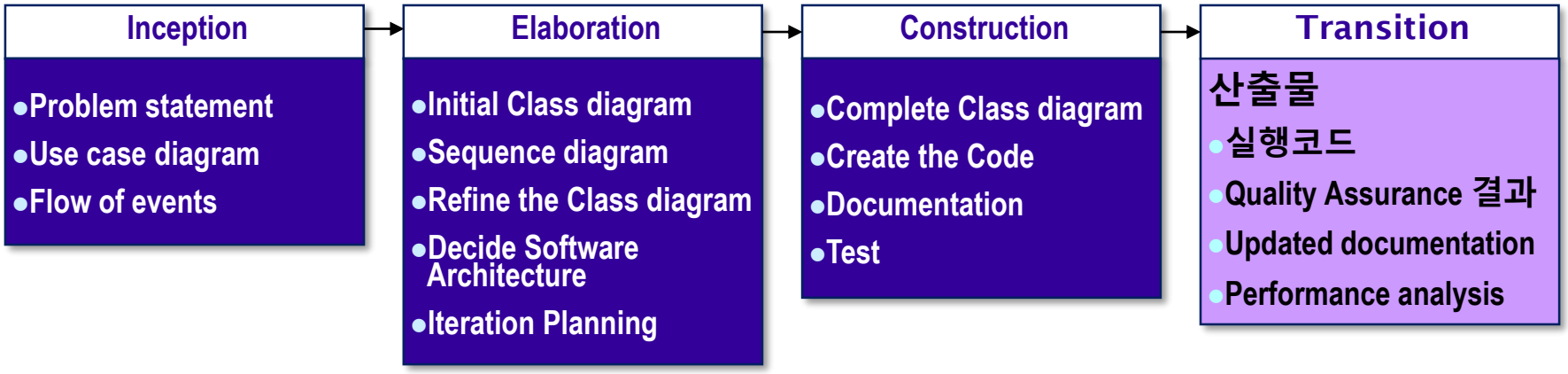
– Integrate and test the iteration with any previous iterations.

  => The process of building iterations continues until the software production is completed

- **Transition(인도)**
  - 목표: 소프트웨어를 user에게 **delivery.**
  - 산출물:
    - 실행코드
    - Quality Assurance 결과
    - Updated documentation
    - Performance analysis

Transition
산출물 ● 실행코드 ● Quality Assurance 결과 ● Updated documentation ● Performance analysis

A CASE Study

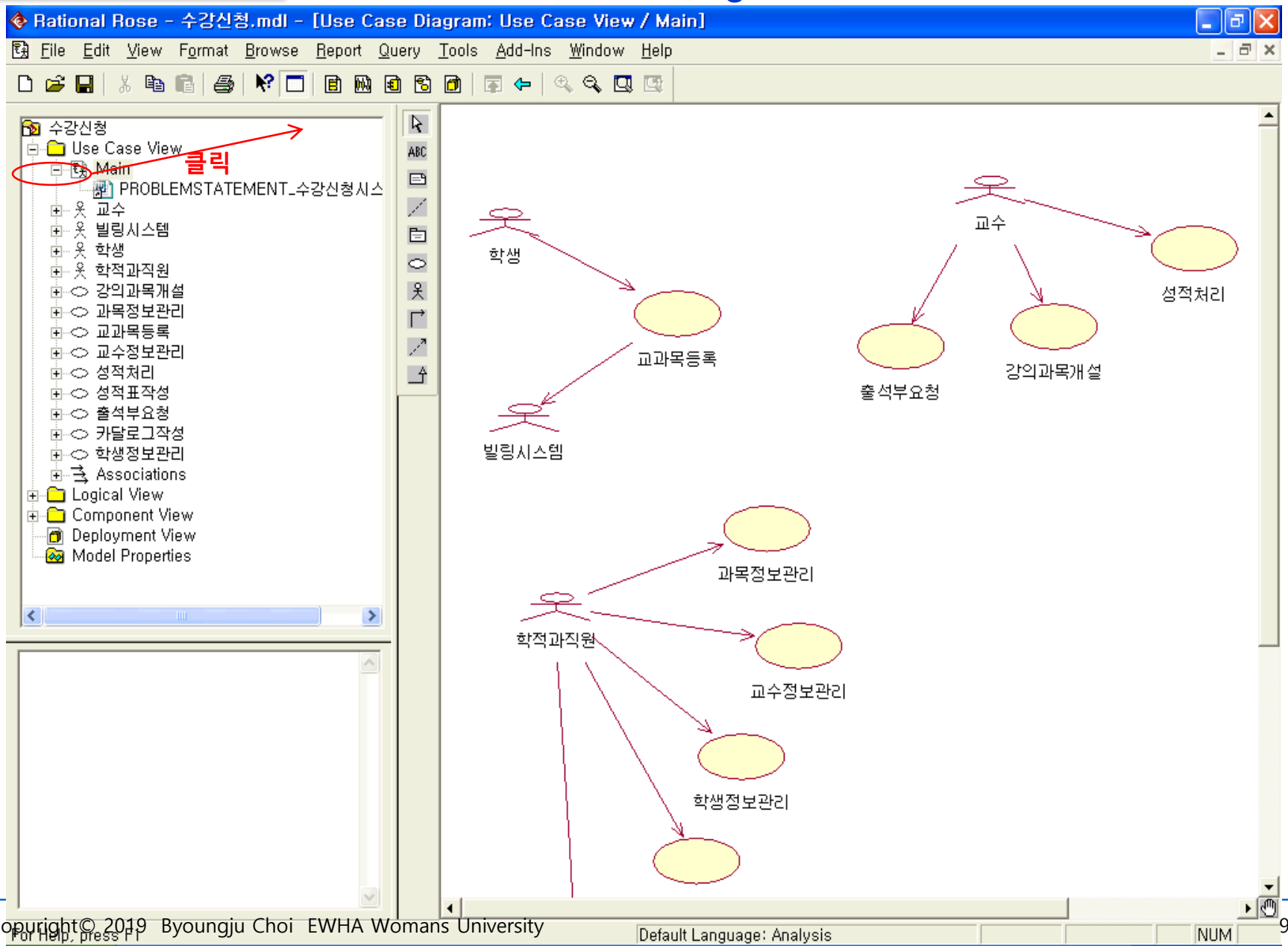

Inception
Problem statement
Use case diagram
Flow of events
Elaboration
Initial Class diagram
Sequence diagram
Refine the Class diagram
Decide Software Architecture
Iteration Planning
Construction
Complete Class diagram
Create the Code
Documentation
Test
Transition
산출물
실행코드
Quality Assurance 결과
Updated documentation
Performance analysis

Problem Statement

<PROBLEMSTATEMENT_수강시스템.doc>

- 이 시스템은 대학에서 학생의 교육과정을 위해 교수들에 의한 교육과정 선정, 학생들에 의한 교육과정 등록부터 학기 이후 성적표 발급까지의 상황을 지원한다.
- 매 학기초에 학생들이 해당 학기에 개설된 과목 목록을 요청하여 받아 볼 수 있고, 이 목록에는 담당 교수, 분야, 강의실, 선수과목 등과 같은 각 과목에 대한 정보가 학생들의 결정을 돕기 위해 포함되어 있다.
- 새 시스템은 학생들이 4개의 과목을 해당 학기에 선택할 수 있도록 한다.
- 각 학생들은 한 개설과목이 마감되거나 취소될 경우를 대비하여 2개의 예비과목을 지정할 수 있게 된다.
- 과목 정원이 10명이고, 최소 인원은 3명이며 3명 미만의 개설 과목은 취소된다.
- 등록 절차가 완료되면, 시스템은 학생들에게 등록금을 청구할 수 있도록 청구 시스템에게 수강 신청 정보를 보낸다.
- 교수는 이 시스템을 사용하여 자신이 강의할 과목을 지정할 수 있고, 자신의 개설과목에 어떤 학생들이 신청했는지 확인할 수 있다.
- 매 학기초 일정 기간 동안은 학생들이 자신의 등록 내용을 변경할 수 있다.
- 학생들은 이 시스템을 이용하여 이 기간 동안 수강 과정을 추가, 취소할 수 있어야 한다.

Use case Diagram



교과목등록 Use case의 Flow of Event

<USECASE_강의과목개설.doc>

<선행조건>

과목 관리 use case의 개설과목 생성 subflow는 이 use case 시작 전에 실행되어야만 한다.

<주요플로우>

이 use case는 교수가 시스템을 login하고 password를 입력했을 때 시작된다. 시스템은 password가 유효한지 확인하고(E-1), 교수가 현재 학기 또는 다음 학기를 선택할 수 있도록 한다 (E-2). 교수는 원하는 학기를 입력한다. 시스템은 교수가 다음 항목 중 하나를 선택할 수 있도록 한다.

- S-1: 개설과목 추가
- S-2: 개설과목 삭제
- S-3: 스케쥴 검토
- S-4: 스케쥴 인쇄

<세부플로우>

S-1: 개설과목 추가

시스템은 과목명과 과목번호 field를 갖고 있는 과목 화면을 출력한다. 교수는 과목명과 번호를 입력한다(E-3). 시스템은 입력된 과목에 대한 개설과목을 출력한다(E-4). 교수는 개설과목을 선택한다. 시스템은 교수와 선택된 개설과목을 연결한다(E-5). 이 use case는 이 때 다시 시작한다.

S-2: 개설과목 삭제

시스템은 과목명과 과목번호 field를 갖고 있는 과목 화면을 출력한다. 교수는 이름과 개설과목 번호를 입력한다(E-6). 시스템은 교수와 개설과목의 연결관계를 삭제한다(E-7). 이 use case는 이 때 다시 시작한다.

S-3: 스케줄 검토

시스템은 교수에게 할당된 모든 개설과목에 대한 다음의 정보를 검색(E-8)하고, 출력한다: 과목명, 과목번호, 개설과목 번호, 요일, 시간, 장소. 교수가 검토를 완료했다고 지시했을 때 이 use case는 다시 시작한다.

S-4:스케줄 인쇄

시스템은 교수의 스케줄을 인쇄한다(E-9). 이 use case는 다시 시작한다.

<예외플로우>

E-1: 잘못된 교수 ID가 입력되었다. 사용자는 교수 ID를 재입력할 수도 있고 또는 use case를 종료한다.

E-2: 잘못된 학기가 입력되었다. 사용자는 학기를 재입력할 수도 있고 또는 use case를 종료한다.

E-3: 잘못된 과목명 또는 과목번호가 입력되었다. 사용자는 재입력할 수도 있고 또는 usecase를 종료한다.

E-4: 개설과목을 출력할 수가 없다. 사용자에게 현재는 이 option을 사용할 수 없음을 알린다. 이 use case를 다시 시작한다.

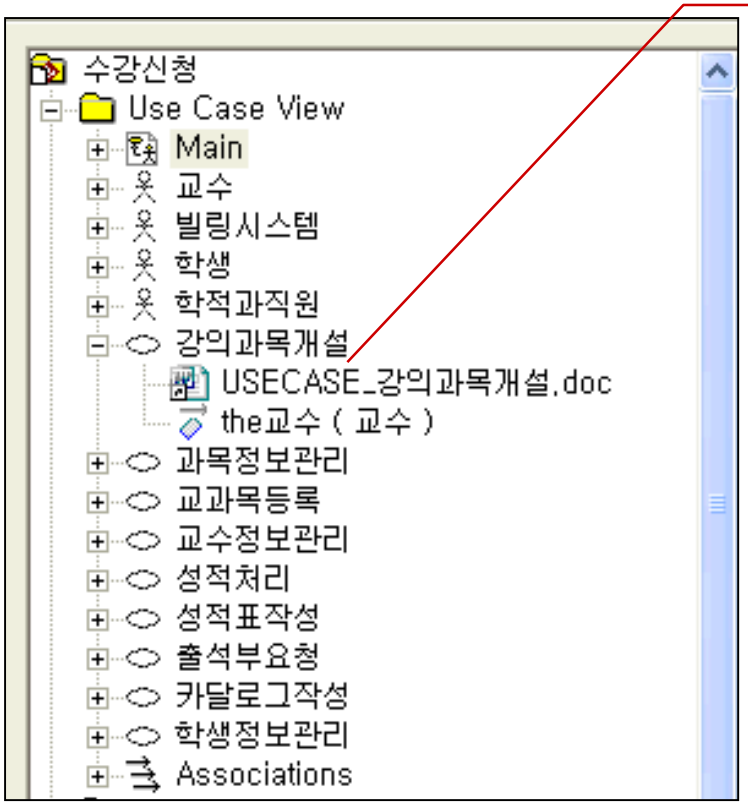
E-5: 교수와 개설과목을 연결할 수 없다. 이 정보는 저장되고 시스템이 나중에 교수와 개설과목을 연결할 것이다. 이 use case는 계속된다.

E-6: 잘못된 개설과목명/번호가 입력되었다. 사용자는 재입력할 수도 있고 또는 use case를 종료한다.

E-7: 교수와 개설과목의 연결관계를 삭제할 수 없다. 이 정보는 저장되고 시스템이 나중에 연결관계를 삭제한다. 이 use case는 계속된다.

E-8: 시스템이 스케줄 정보를 검색할 수 없다. 이 use case를 다시 시작한다.

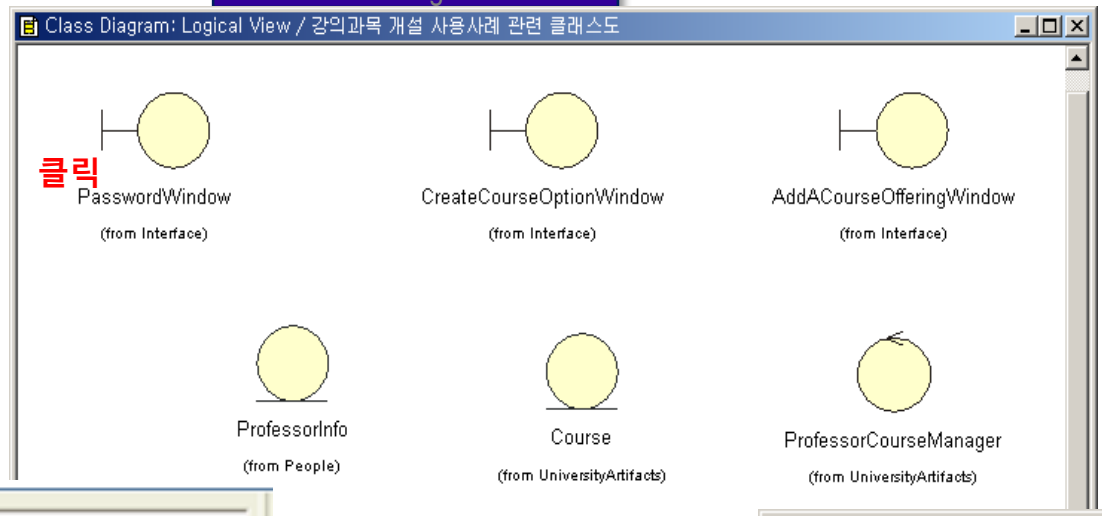
E-9: 스케줄이 인쇄되지 않는다. 사용자에게 현재는 이 option을 사용할 수 없음을 알린다. 이 use case를 다시 시작한다.

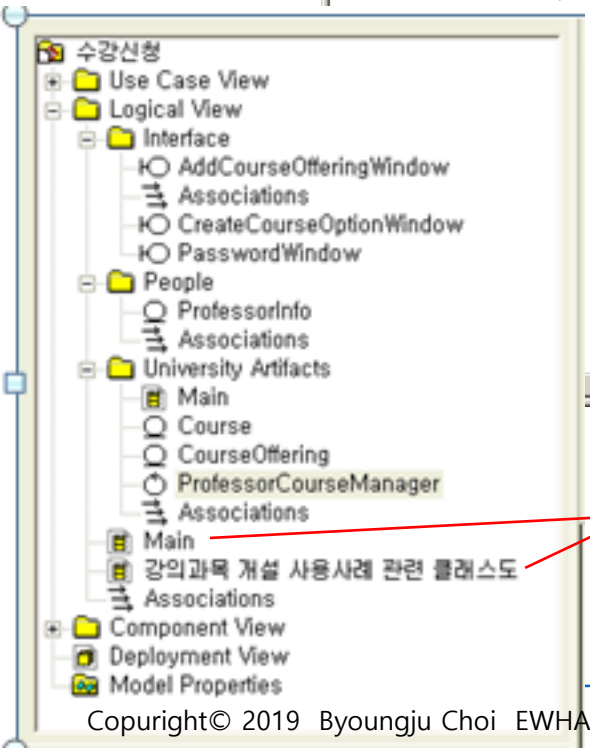


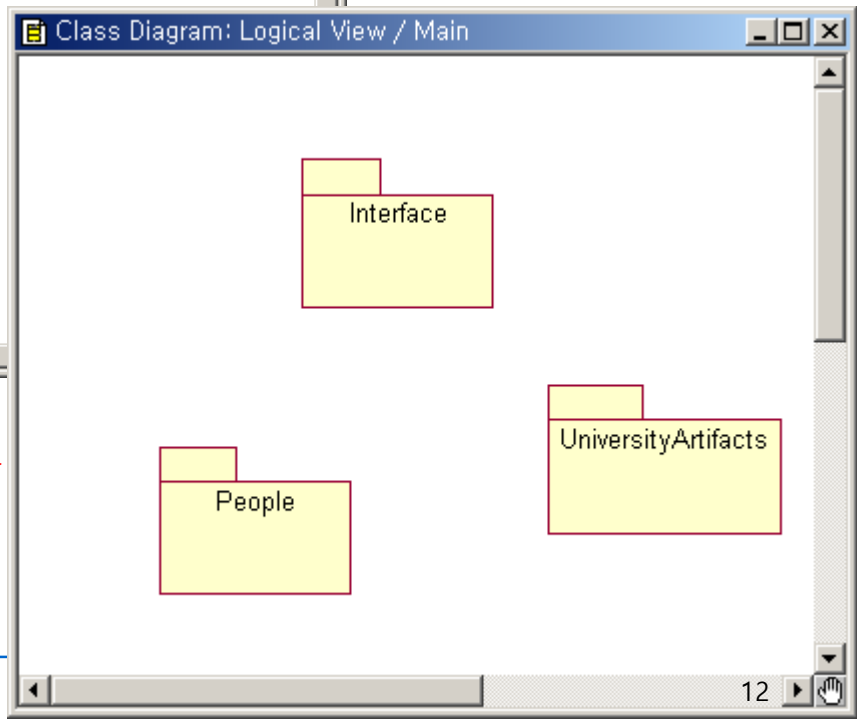
Use case(쓰임새)에 대한 사건의 흐름은 쓰임새에 연결된 외부 문서에 포함된다.

해당 use case 오른click → New → Files → 오른쪽click → Insert file 해당 doc 파일 선택.

Initial Class diagram

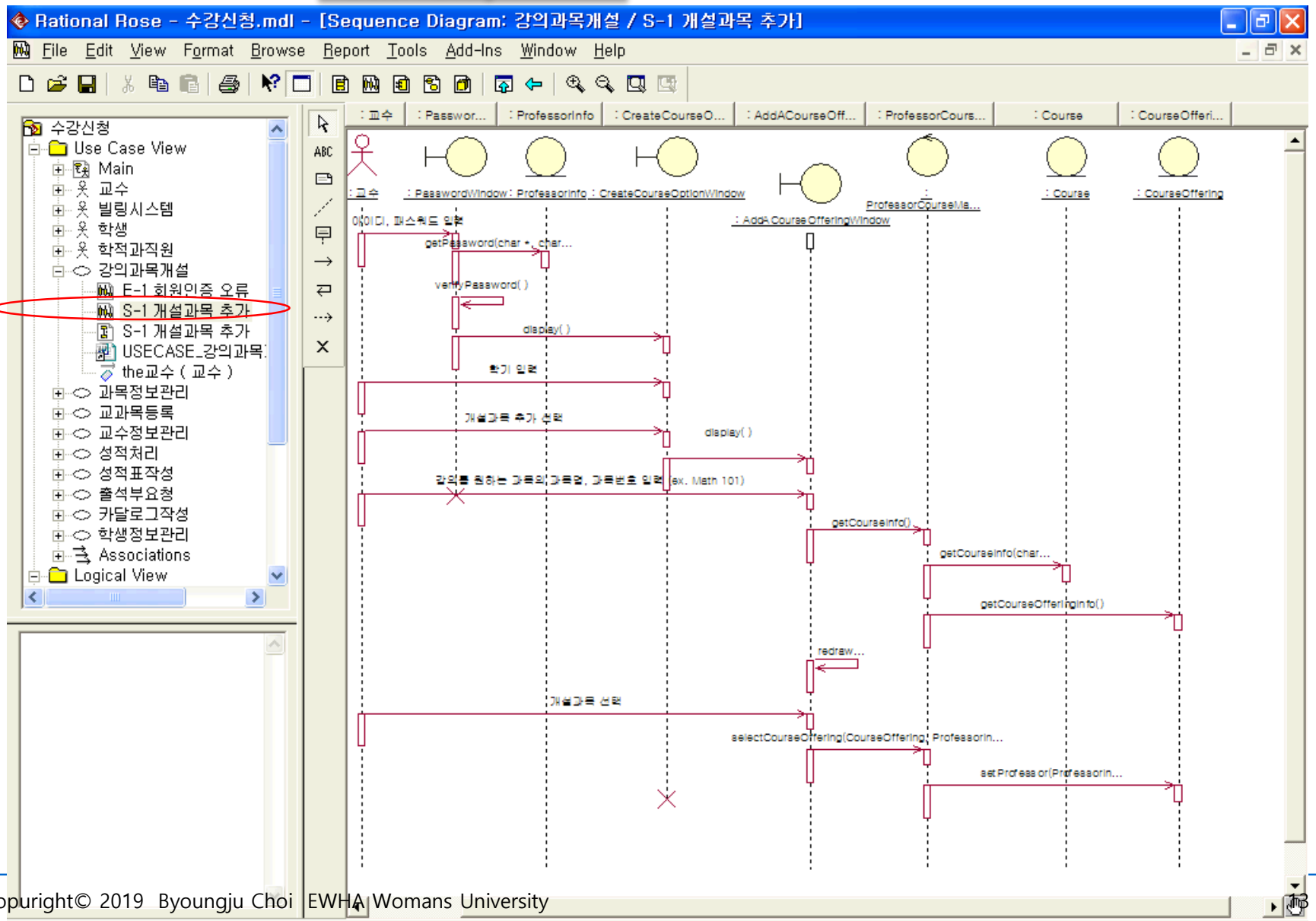






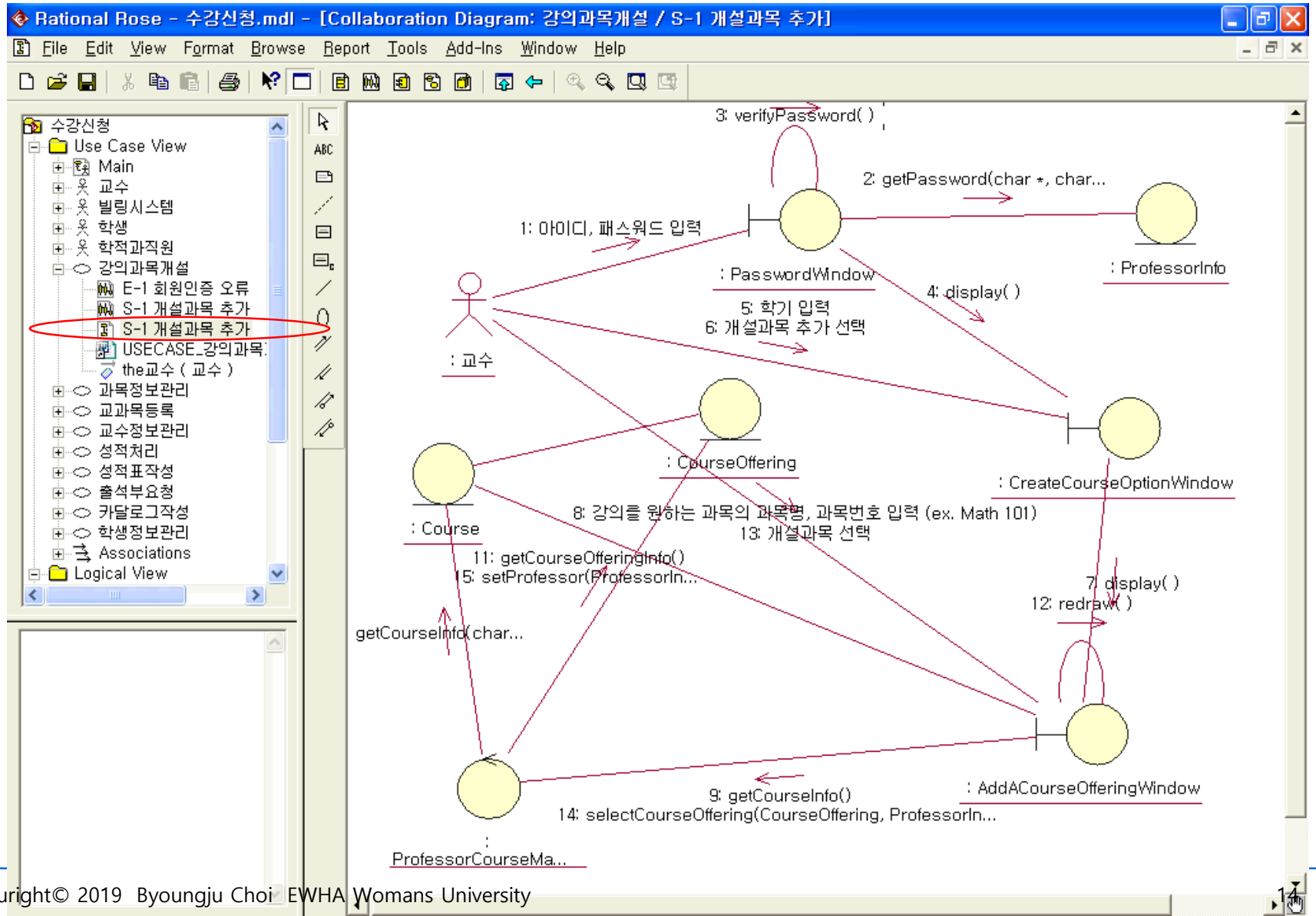
Sequence diagram

강의과목개설 Use case의 S-1 개설과목



Collaboration diagram

강의과목개설 Use case의 S-1 개설과목



Refine Class Diagram

1) **Finding relationships**

2) **Finding multiplicity & navigation**

3) **Finding data and methods**

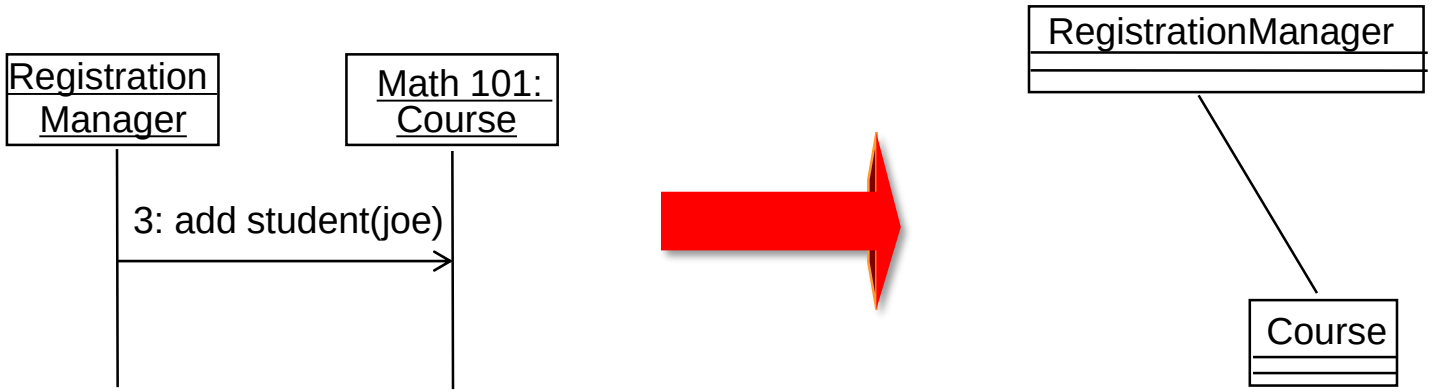
4) **Finding inheritance**

1) Finding relationships

– Interaction diagrams의 메시지 흐름을 통해 relation을 찾을 수 있음.

– Sequence diagram의 Natural language

→ Class의 오퍼레이션(메소드)명



2) Finding multiplicity and navigation

– Multiplicity

- Relation에 참여하는 객체의 수를 multiplicity로 추출.

– Navigation

- 일반적으로 association(연관), aggregation(집합) relation은 양방향 통신이지만, 특수 상황의 경우 단 방향으로 제한 하여야 할 경우 navigation direction으로 표현할 수 있음.
- 재귀적 관계
  - 재귀적 관계는 같은 클래스의 여러 객체들이 서로 협력하는 관계이다.

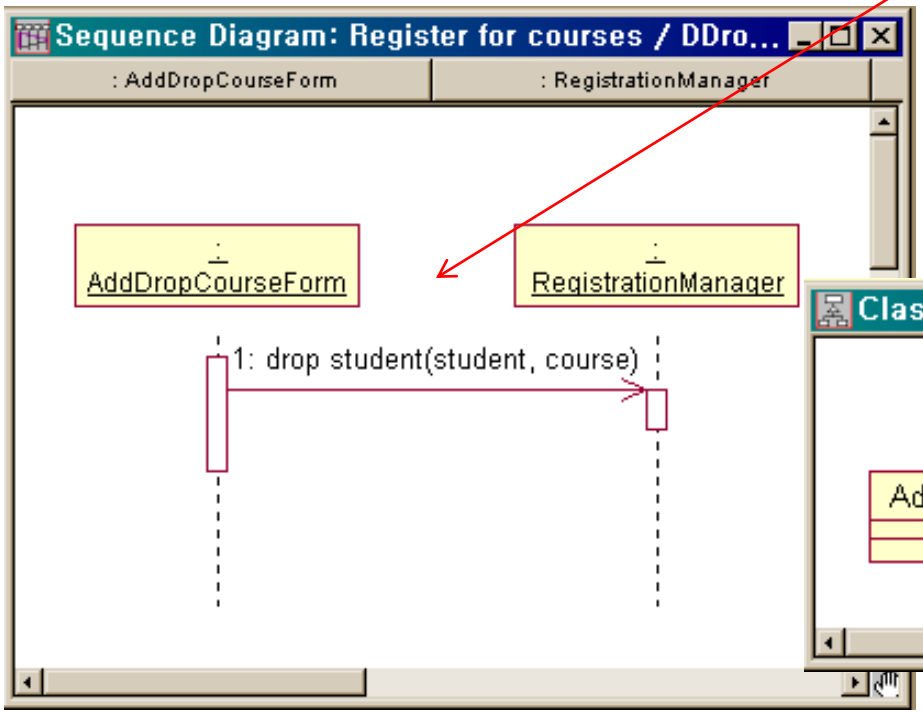
– 제한조건 (constraints)

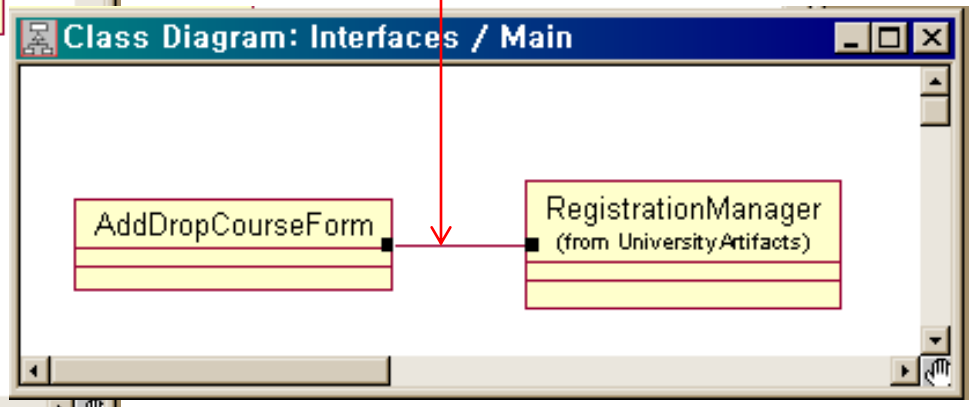
- 집합 기호 안에 표현

예) Association

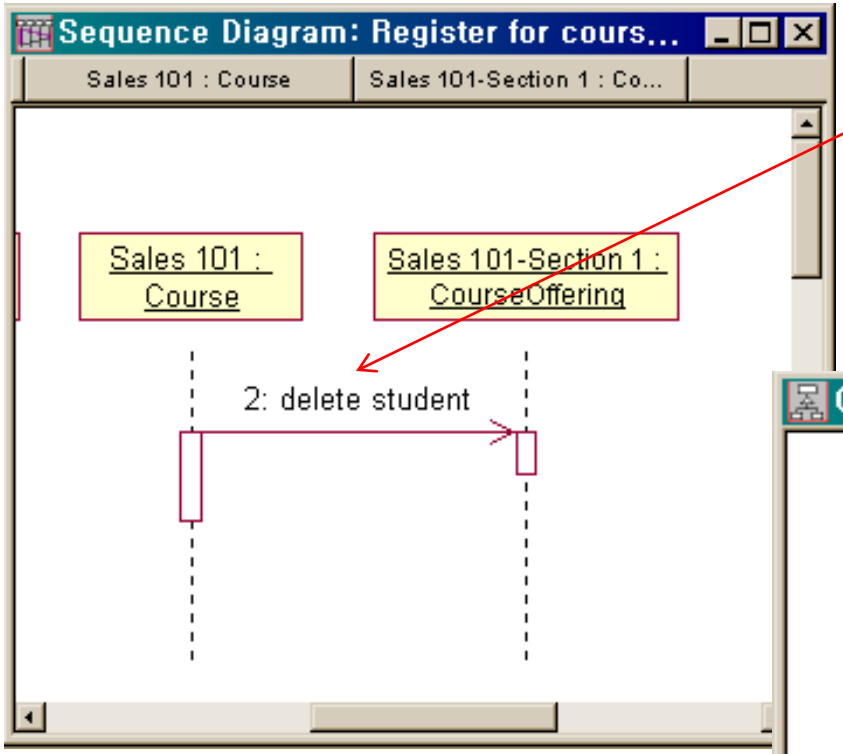
AddDropCourseForm 이 RegistrationManager 에게 메시지를 보낸다.

두 객체들은 독립적이다.



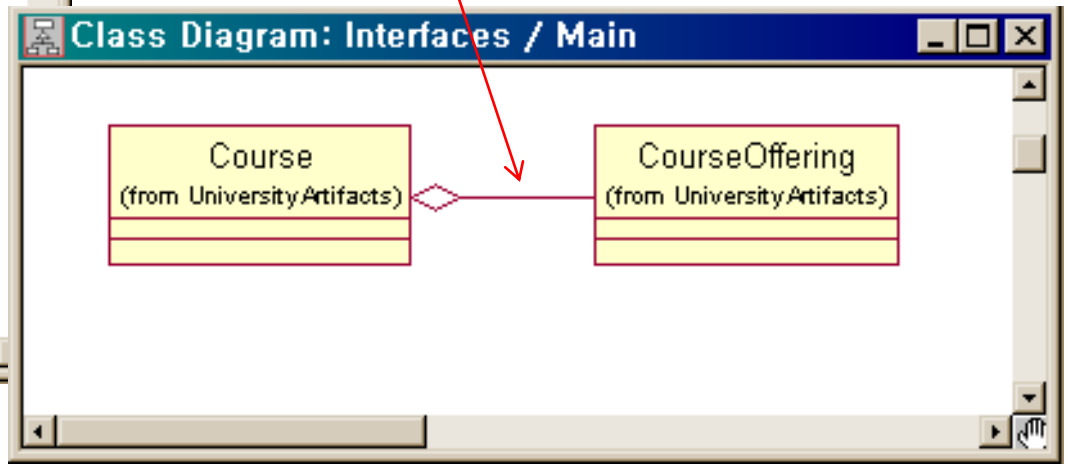


예) Aggregation

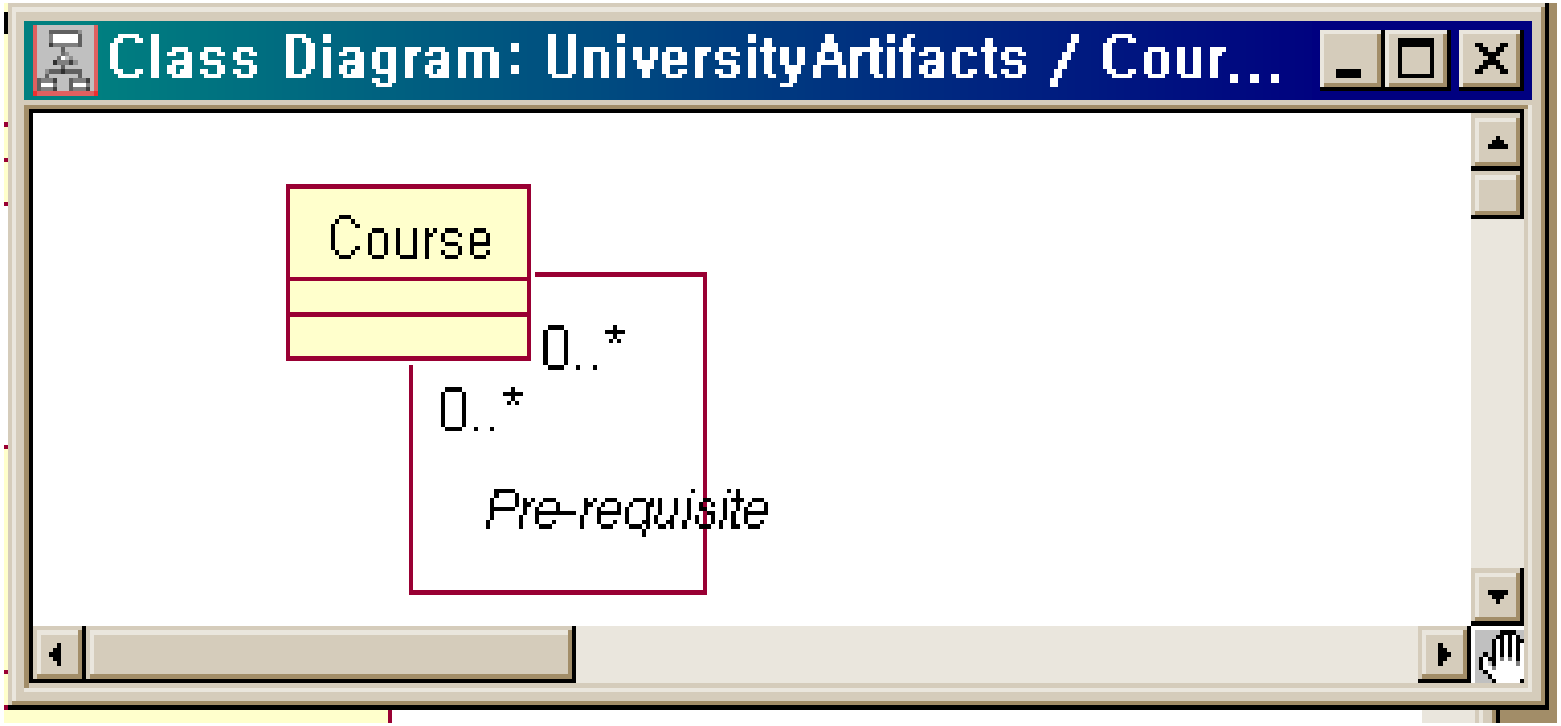


Course는 CourseOffering에게 메시지를 보낸다.

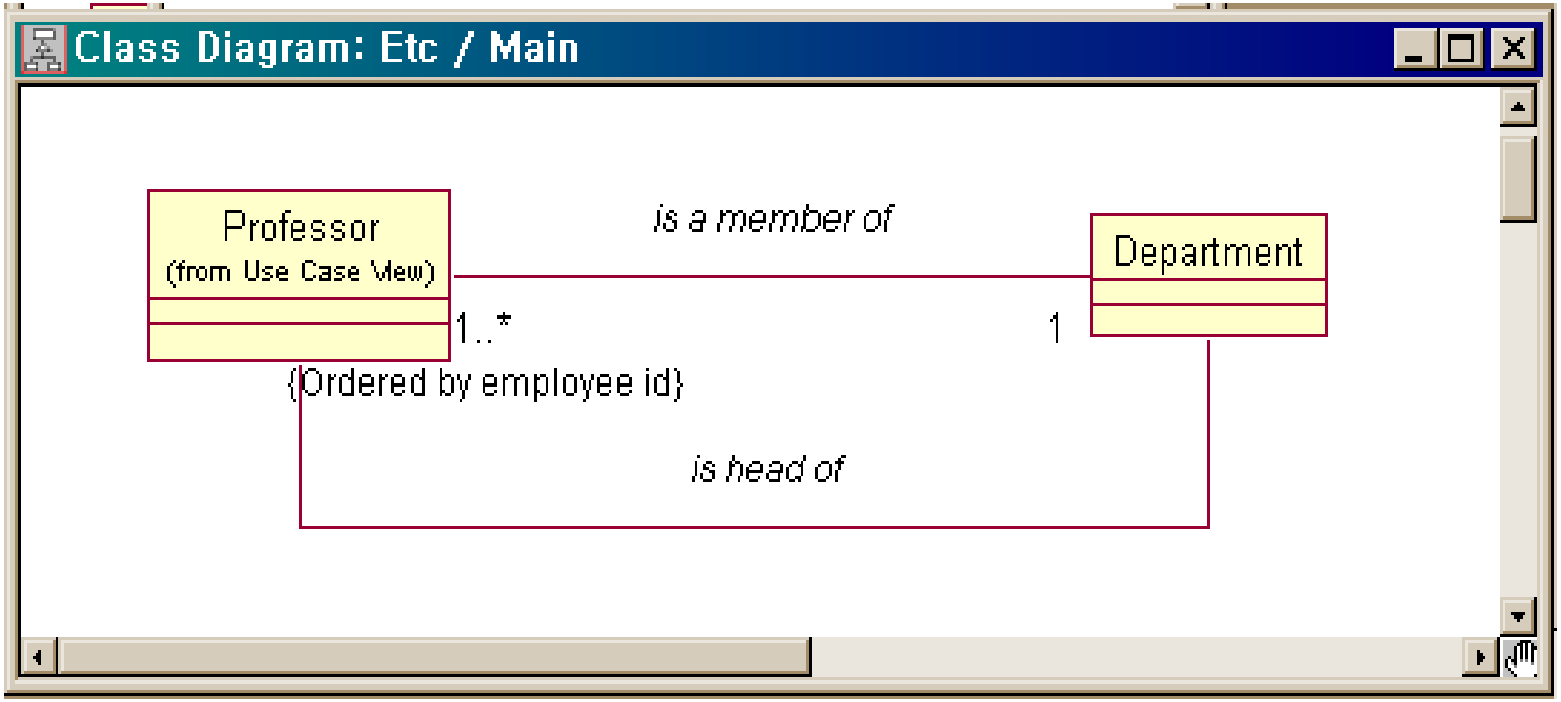
CourseOffering는 Course의 일부분이다.



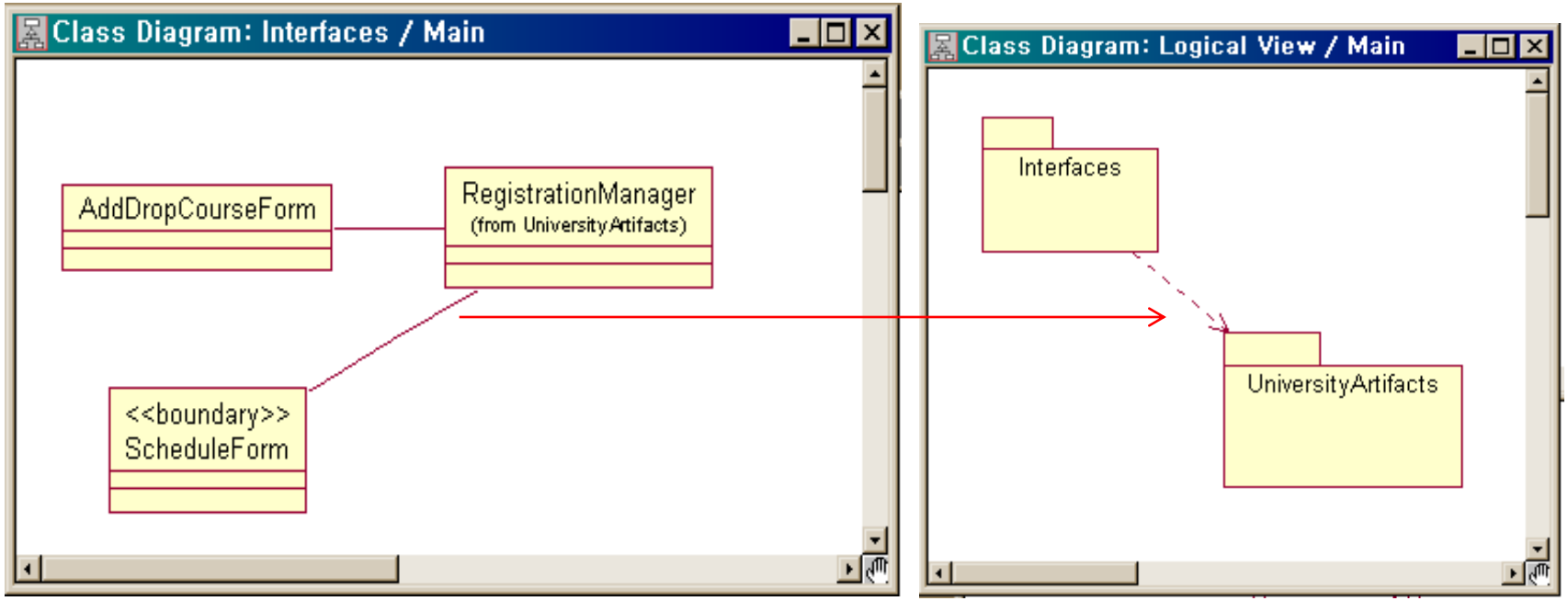
예) 재귀적 관계


Class Diagram: UniversityArtifacts / Cour...
Course
0..*
0..*
Pre-requisite

예) 제한조건(constraints)



예)패키지 관계


Class Diagram: Interfaces / Main
AddDropCourseForm
RegistrationManager
(from UniversityArtifacts)
<<boundary>>
ScheduleForm
Class Diagram: Logical View / Main
Interfaces
UniversityArtifacts

3) Finding data and methods: Operations

– 순서도와 협력도 내의 **메시지 흐름**이 수신 class의 operation의 후보.

– 메시지가 operation이 되지 않는 경우?

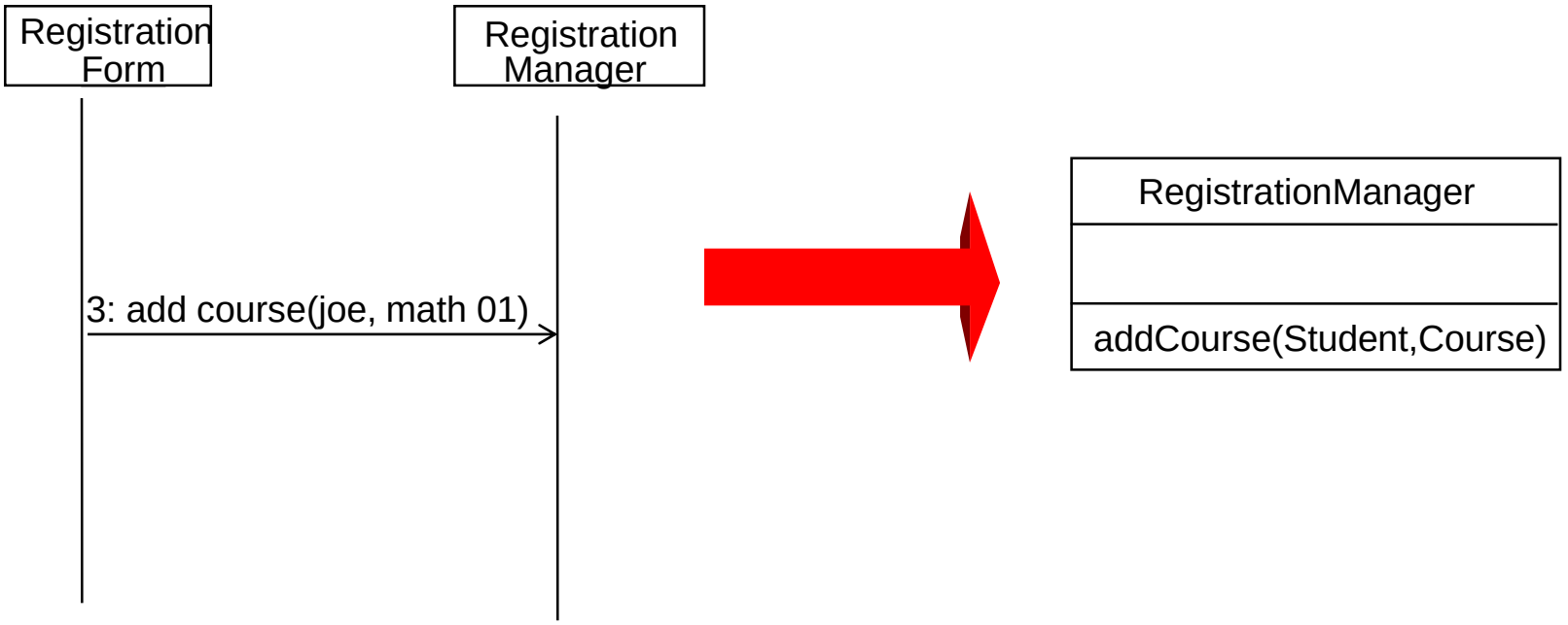
• 수신 Class가 GUI 형태의 Boundary Class

메시지는 GUI의 요구 명세이며, 이런 형태의 메시지들은 일반적으로 GUI control(i.e, a button)로 구현되기 때문에 operation으로 mapping되지 않음.

– Actor가 외부 시스템인 경우

외부 시스템과 통신을 수행하기 위해 사용되는 프로토콜을 가지기 위해 Class가 만들어져야 하고, 이 메시지는 그 Class의 operation으로 mapping.

예)

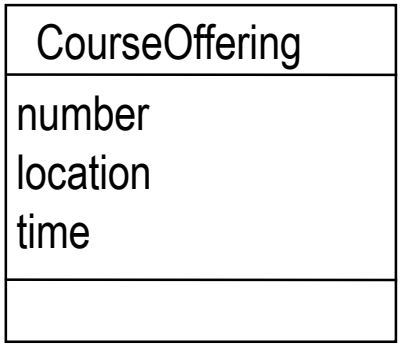

Registration Form
Registration Manager
3: add course(joe, math 01)
RegistrationManager
addCourse(Student,Course)

3) Finding data and methods: Attributes

– Attributes는 class 정의, 문제 정의서, 일반적인 문제 도메인 관련 지식으로부터 추출

예)

Each course offering has a number, location and time



4) Finding Inheritance

– Generalization

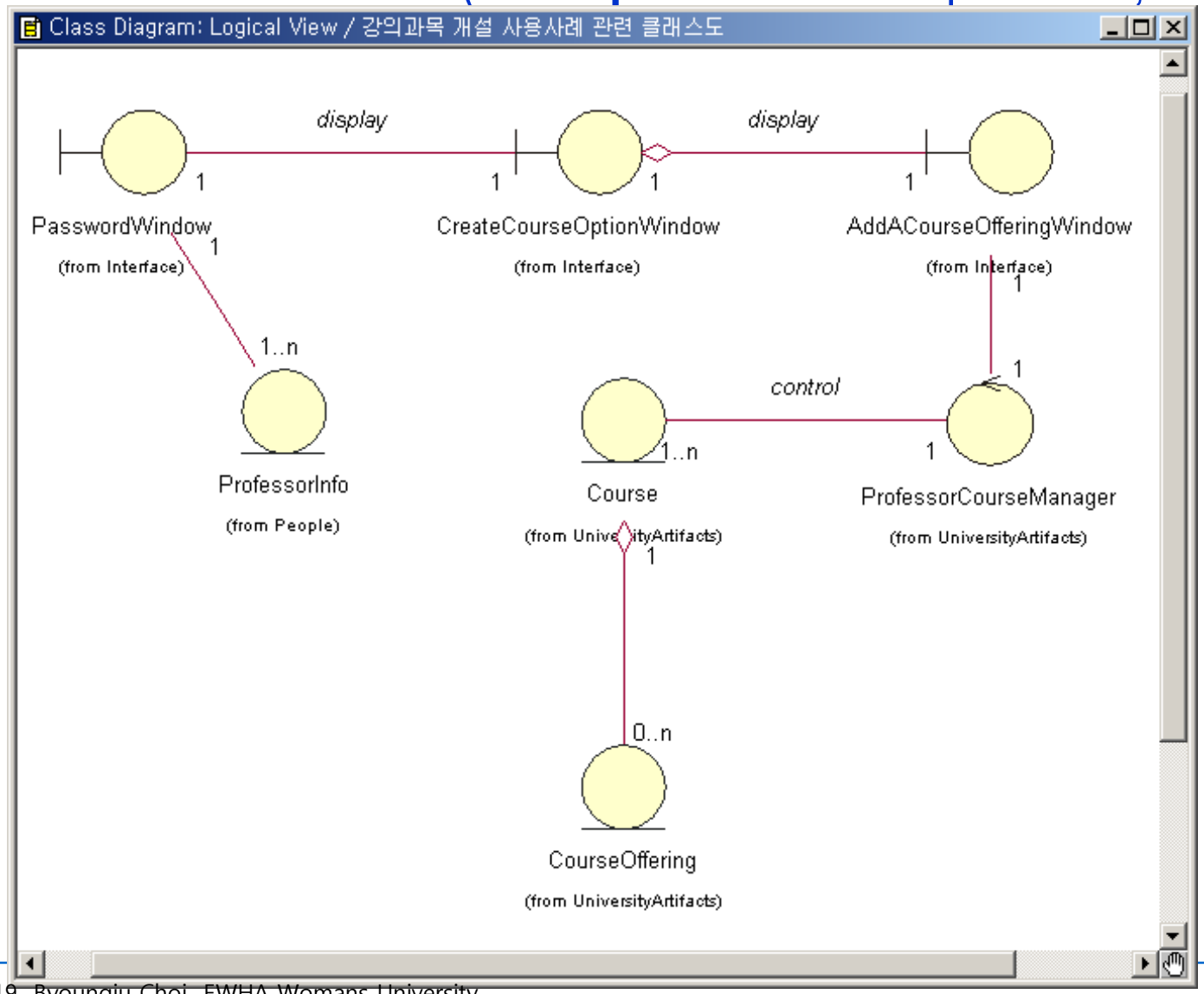
- 여러 Class들의 공통된 attribute와 operation를 캡슐화하여 supperclass를 정의.

– Specialization

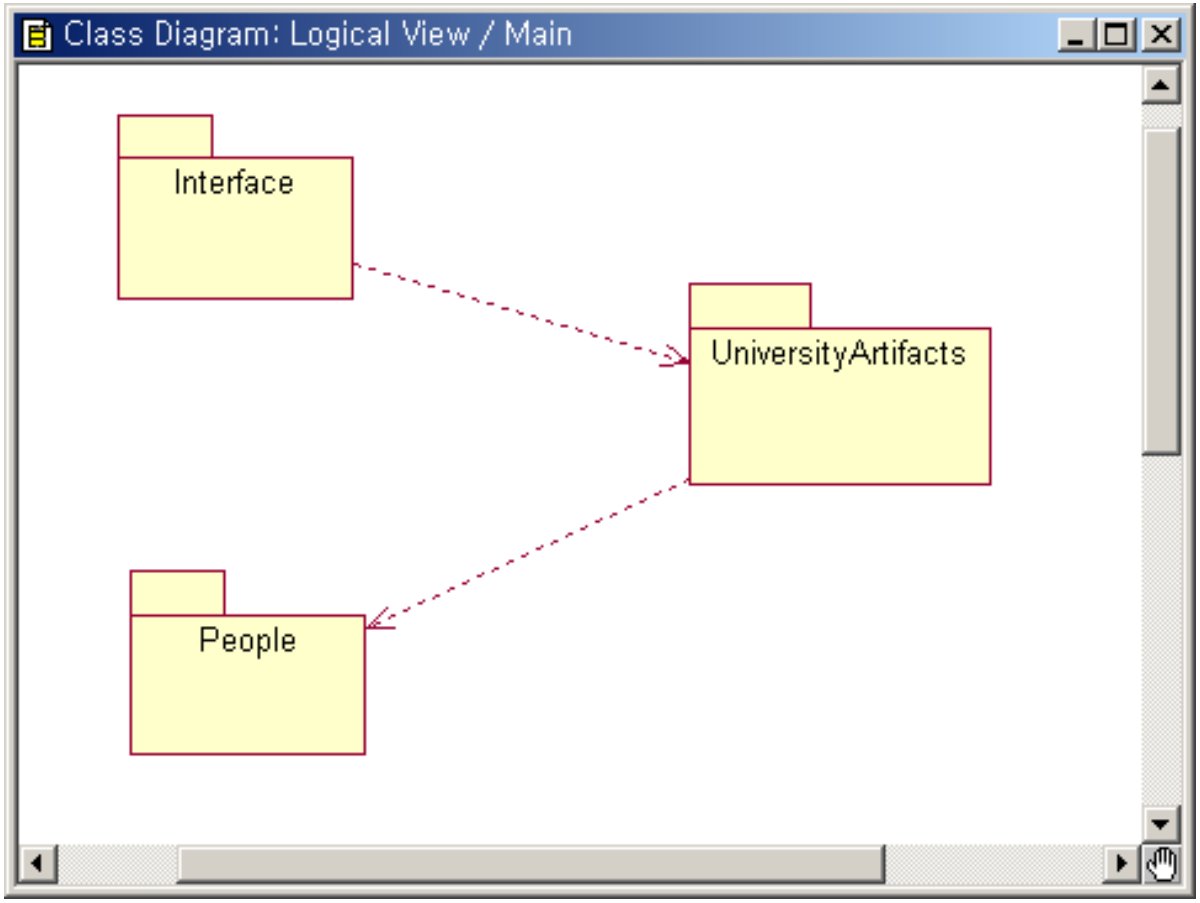
- supperclass를 세련화(상세화)할 수 있는 subclass를 생성.
- supperclass에 구체적인 attribute, operation을 추가하여 상세한 class를 정의.

Refined Class diagram

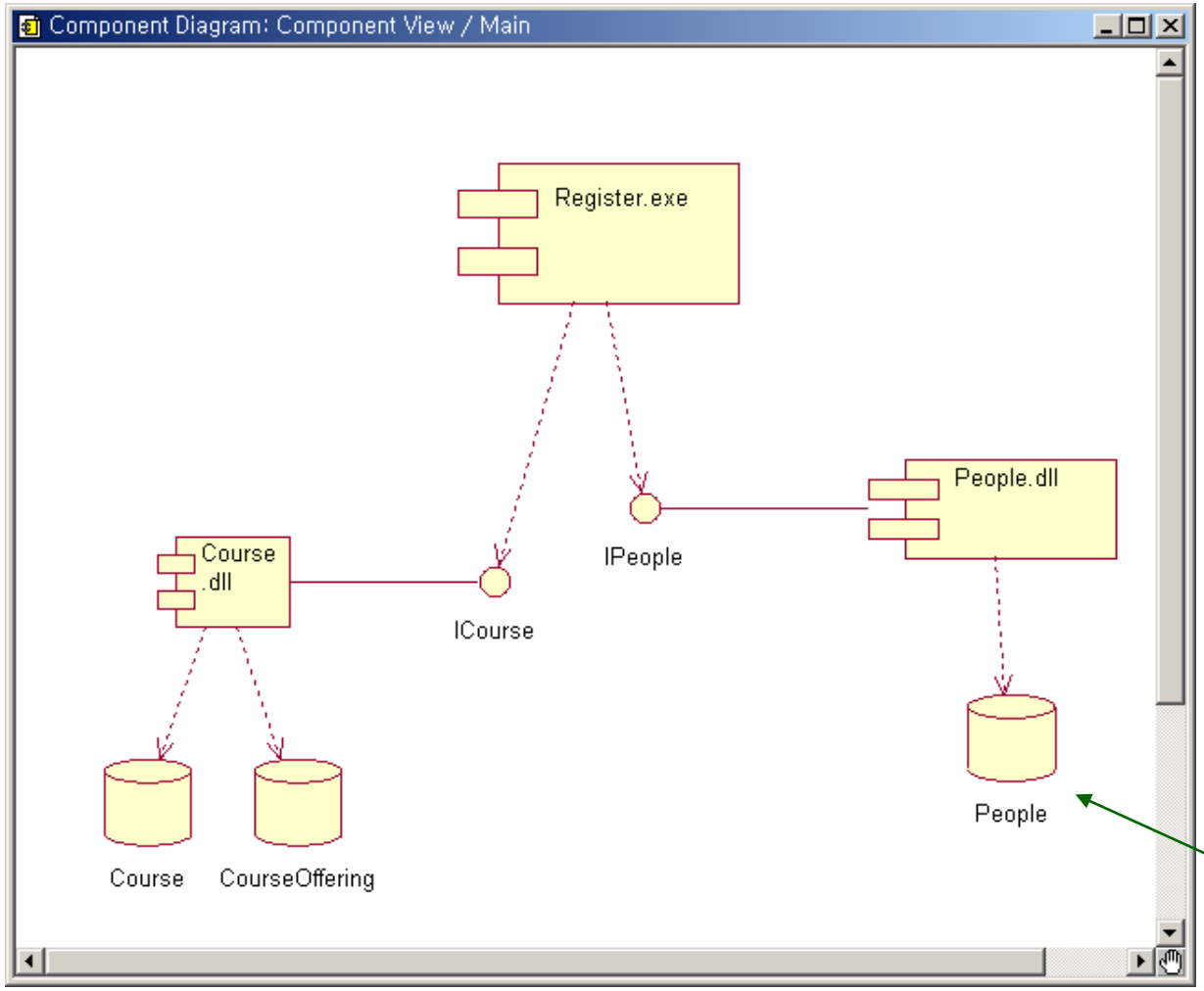
(**Incomplete**.. Without operations, attributes)



- Initial Class diagram
- Sequence diagram
- **Refine the Class diagram**
- Decide Software Architecture
- Iteration Planning



Component diagram



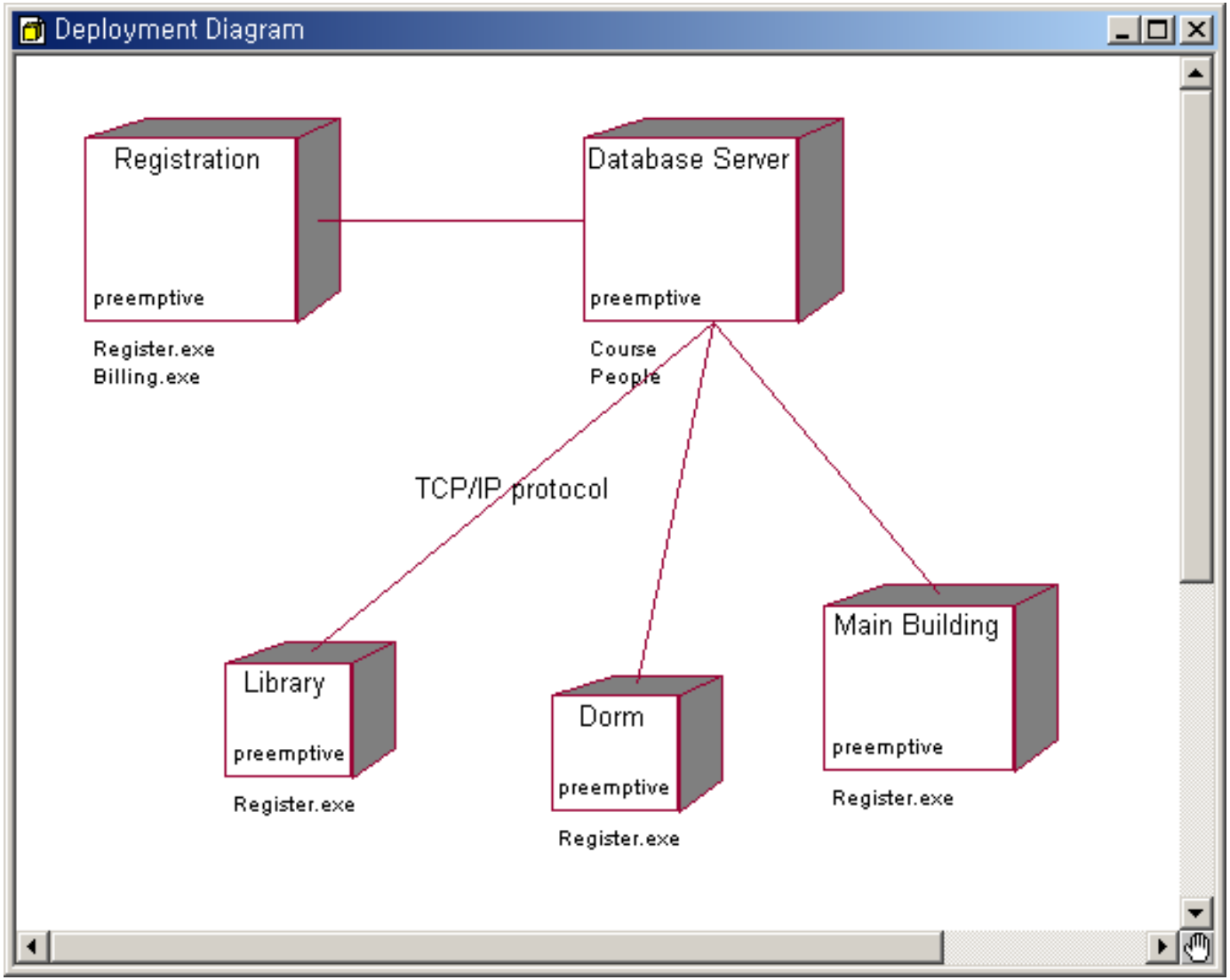
Register.exe
AddACouseOfferingWindow
CreateCourseOptionWindow
PasswordWindow
ProfessorCourseManager

Course.dll
Course
Icourse <<Interface>>
CourseOffering

People.dll
Ipeople <<Interface>>
ProfessorInfo

Stereotype: database
Language Oracle
Stereodisplay: icon

Deployment diagram



실습 및 과제

실습 및 과제2

- **동영상 upload: 사이버캠퍼스 <주차 별 학습활동>**
  - **실습1 (**10월 18일): ROSE Installation, RUP Part1
  - **실습2 (**10월 22일): RUP Part2 → 희망할 경우 10월 18일

- **실습**
  - **실습1, 실습2**: 온라인 실습. 실습은 원칙적으로 개별 수행

- **실습 결과물 제출**
  - 실습1: 10월 18일 오후 11시 사이버캠퍼스 **개인별** 제출 (실습 수행 했다는 "증빙" 내용)
  - 실습2: 과제2 제출하는 것으로 실습 수행한 것으로 인정함

❖**과제 결과물 제출**

- Due: 10월 29일 (화) 오후 6시30분, 과제함 및 사이버캠퍼스 **팀 별** 제출