MAXIMIZE THE
BUSINESS VALUE
OF SOFTWARE

Together 기본 교육

OOC
Open Objected Component

Borland®

목차

- ❖Together Architecture
- ❖Together Overview
- ❖Together 화면설명
- ❖Together Core File
- ❖Together 사전 환경설정
- ❖Together 기능 소개
- ❖Together 알아야 할 사항
- ❖UML 따라하기
- ❖Together Technologies 일반적인 활용

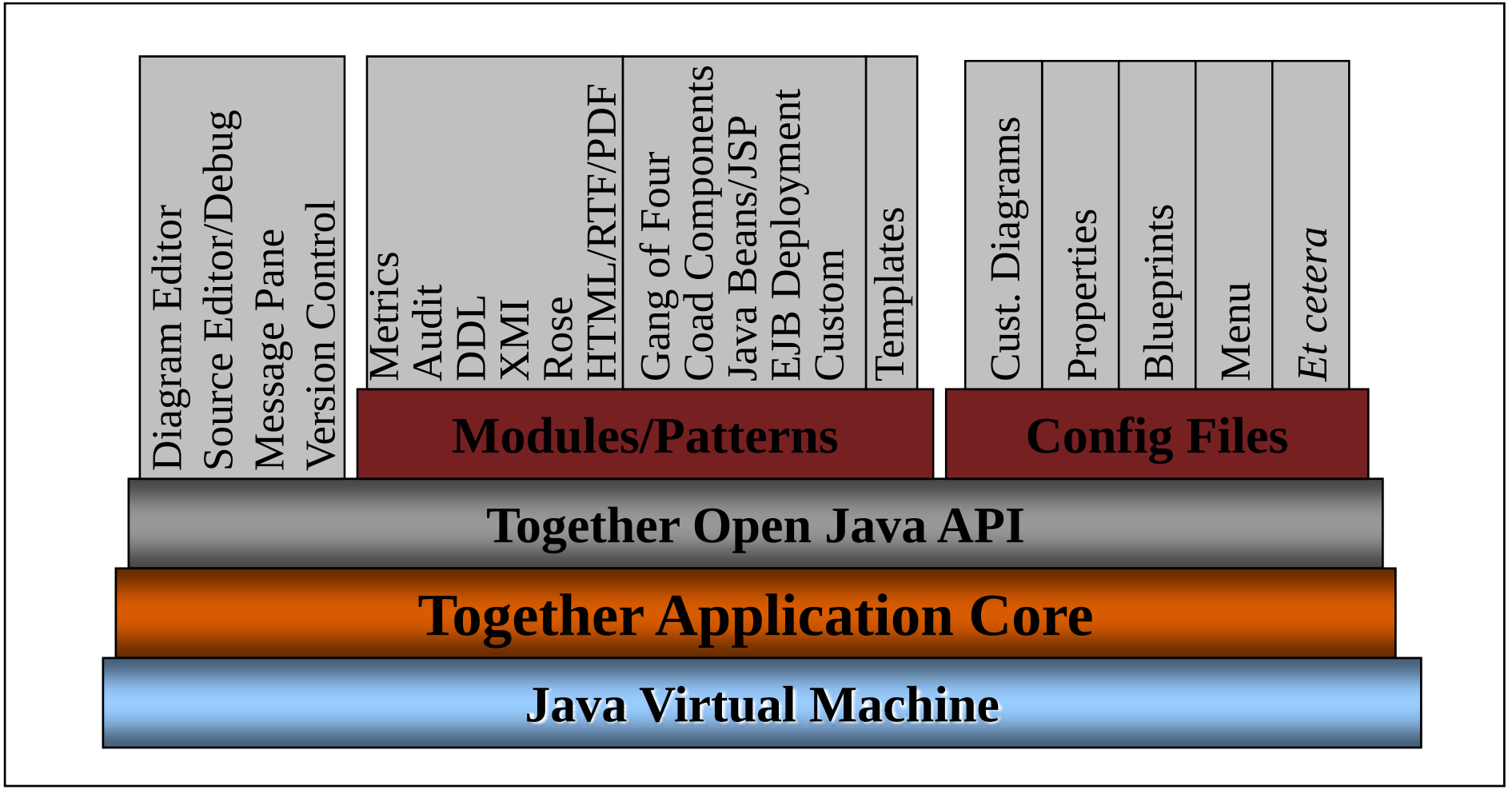
OOC
Open Objected Component

Borland®

Together Architecture

Borland®

Together's Architecture

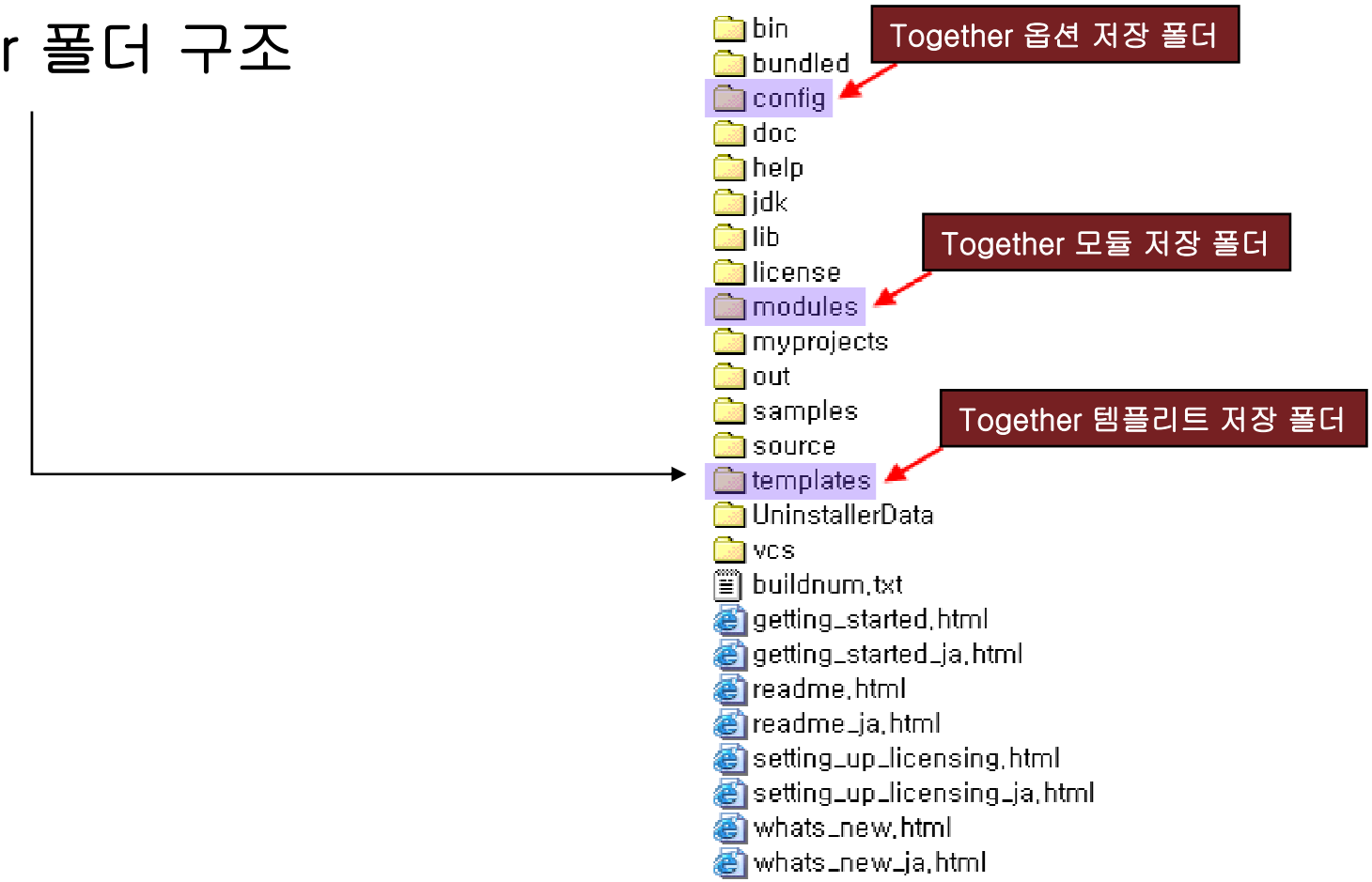


OOC
Open Objected Component

Borland®

Together Overview

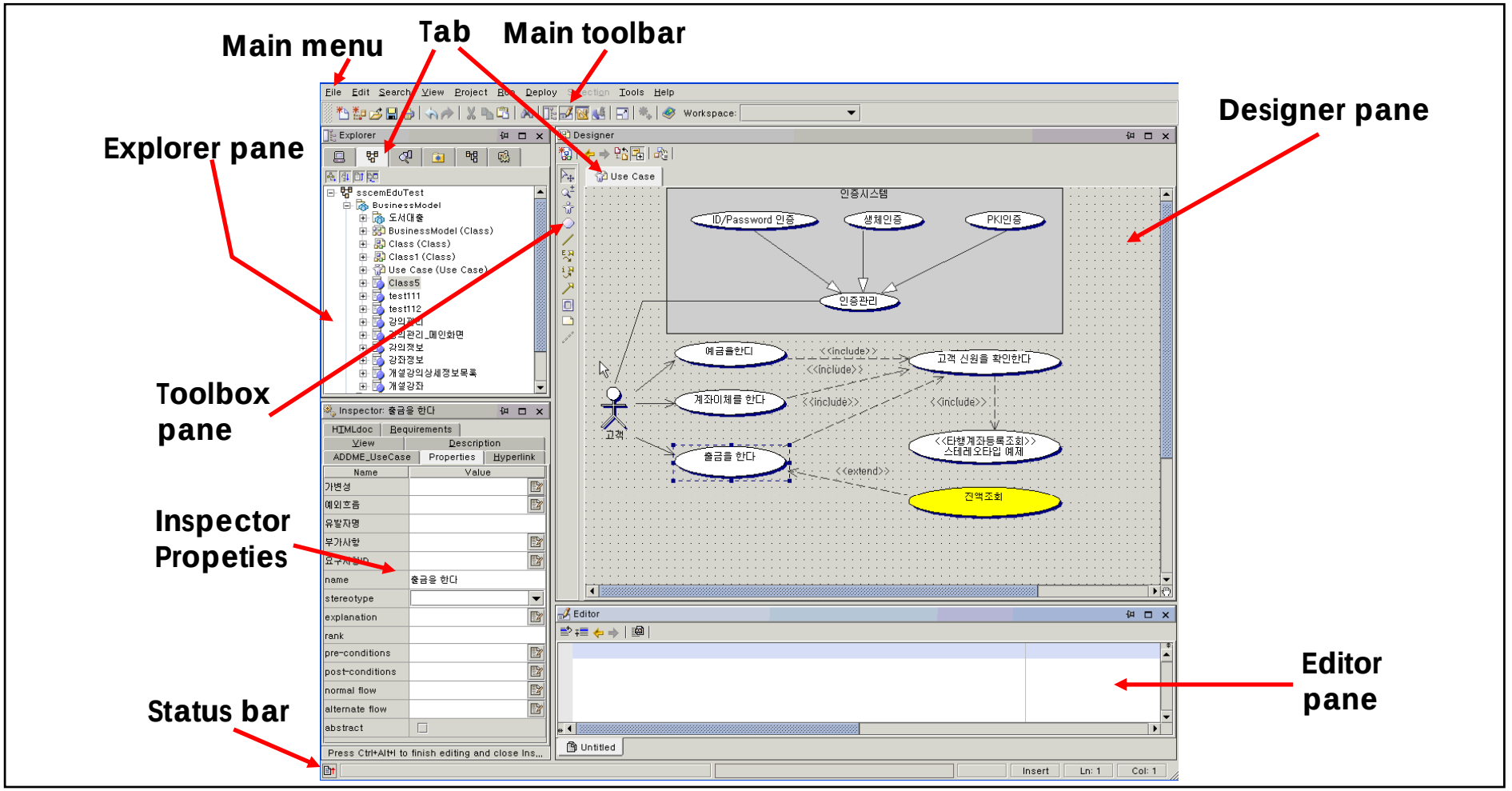
Together 폴더 구조



OOC
Open Objected Component

Borland®

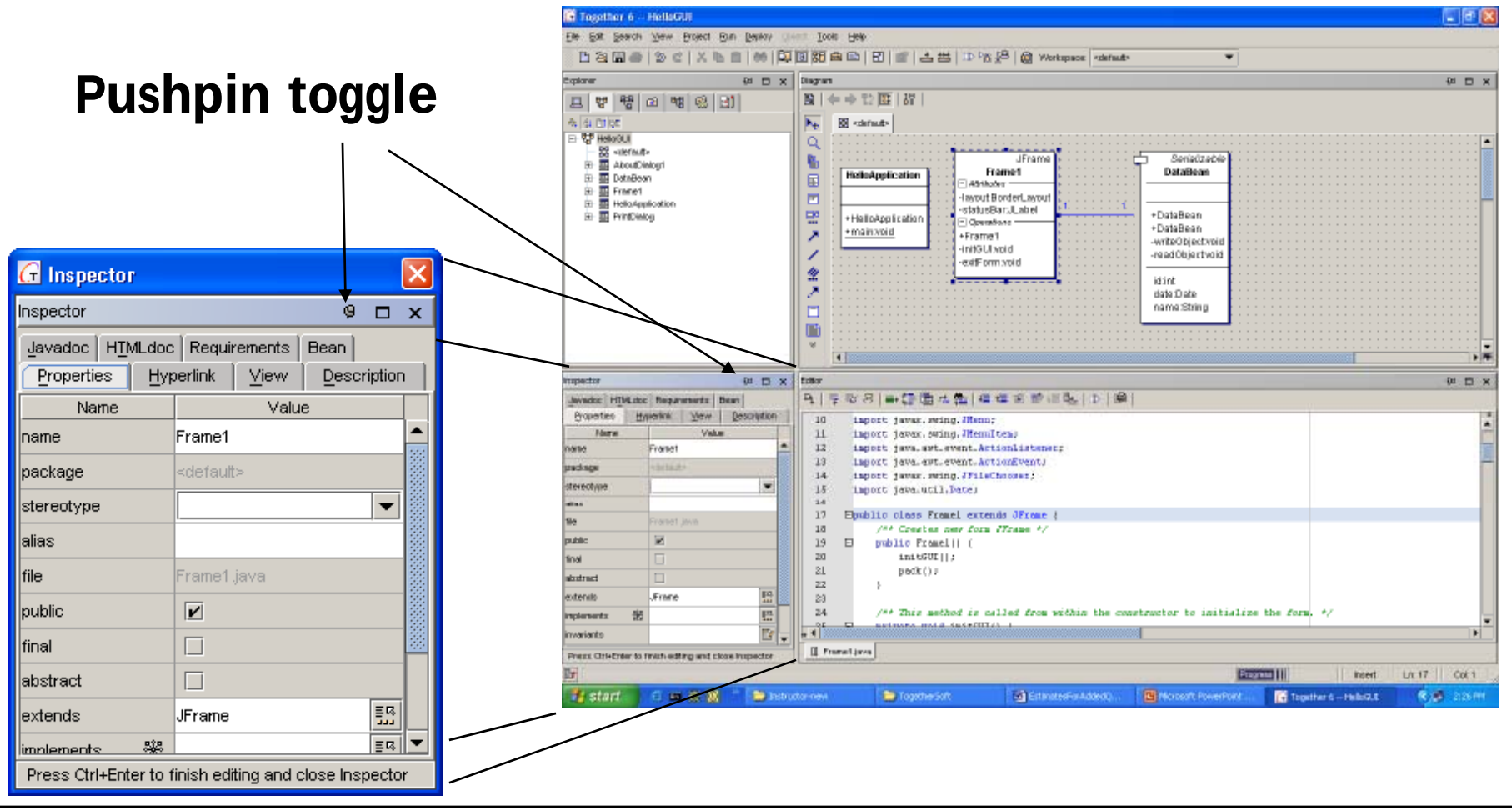
Together User Interface



OOC
Open Objected Component

Borland®

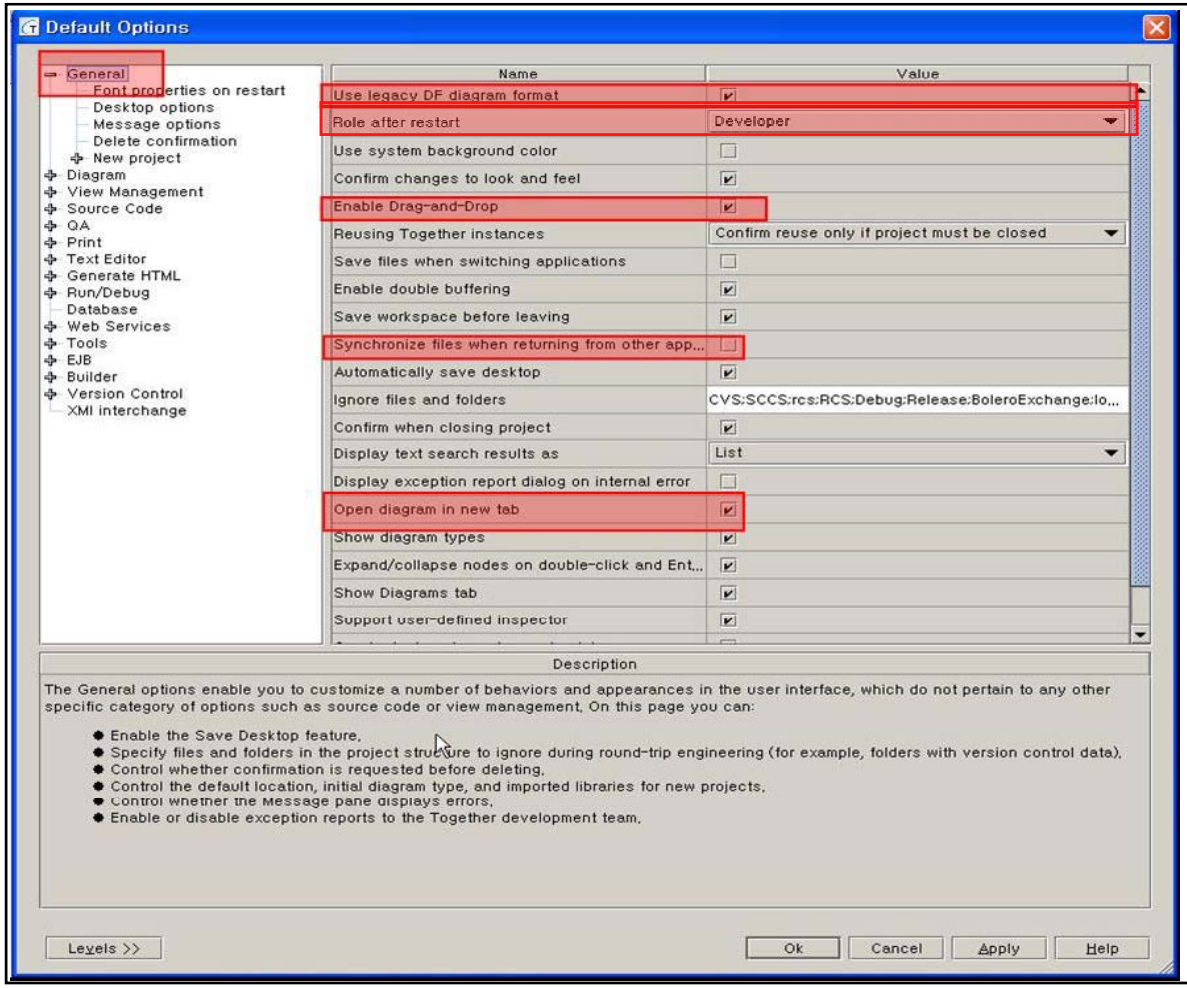
Doctable Panes



OOC
Open Objected Component

Borland®

Together Options 화면



Task 1

이번 설정들은 각각의 다이어그램 마다 새로운 tab이 열리도록, 그리고 외부 tool과의 synch기능을 사용하지 않도록 하기 위해서 사용합니다.

Activity Steps

1)“General” node를 선택

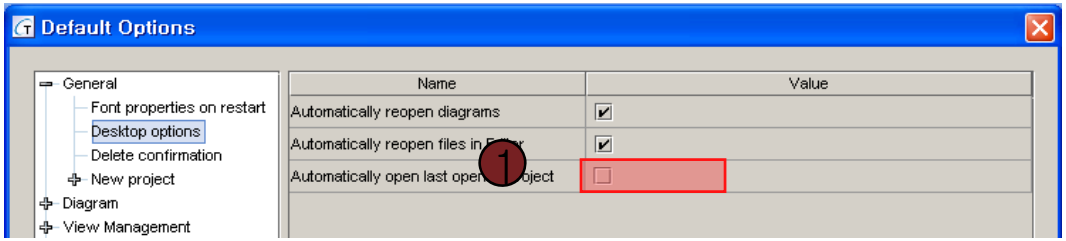
2) Role After Restart - Developer로 세팅

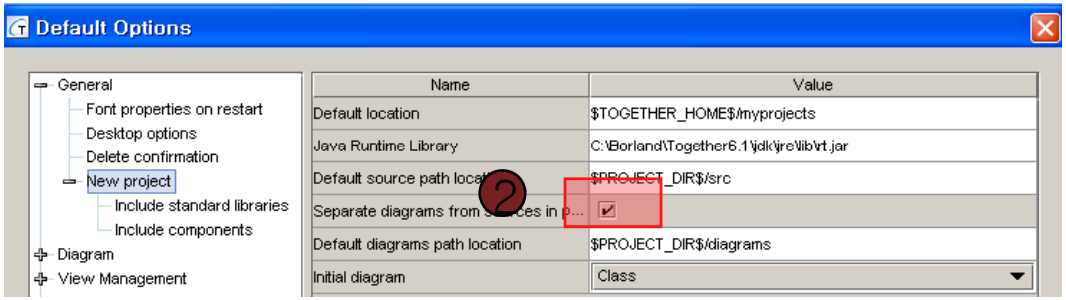
Developer – 다자이너 작업창과 에디터 작업창이 중심이다. Compile, Debug, Assemble, Deploy, Run 과 같은 기능이 UI에서 가능해진다.

3) Name/value list의 스크롤 버튼을 아래로 내려서 “Save files when switching applications”과 “Synchronize files when returning from other applications”를 uncheck로 설정. 위에서 설정한 synch기능을 수동으로 실행하고 싶으면,main menu에서 File | Synchronize with External changes를 선택하면 됩니다.

4) Name/value list에서 아래로 더 내리면 “Open diagram in new tab”이라는 항목을 Check

Together Options 화면



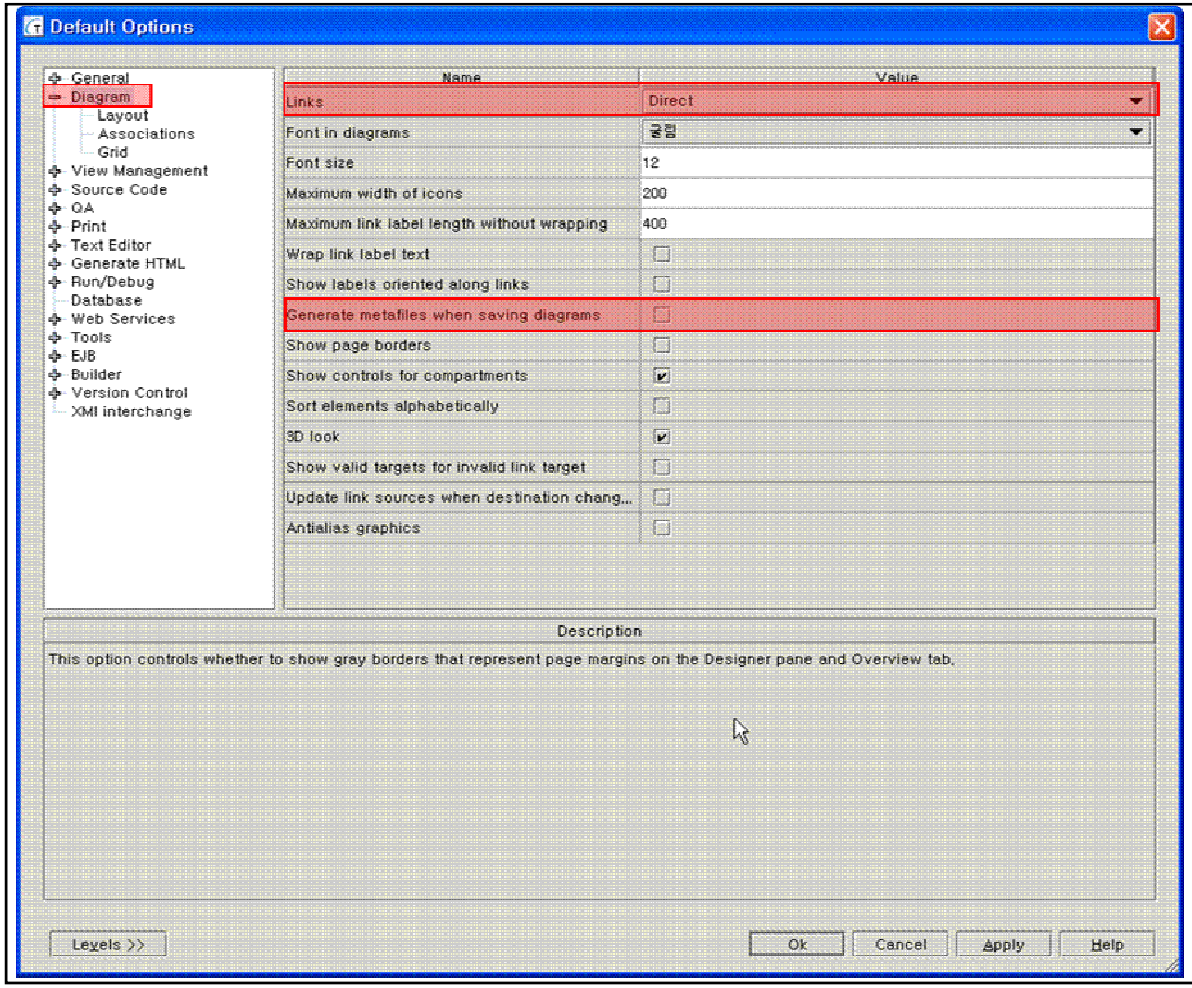


- 1 – 투게더를 실행시 그전 프로젝트를 Open한다. 체크하지 말자
- 2 – 체크를 하면, 모델 데이터와 소스 데이터를 분리해서 관리한다.

OOC
Open Objected Component

Borland®

Together Options 화면



Task 2

다음의 항목들이 Check되어 있는지 확인

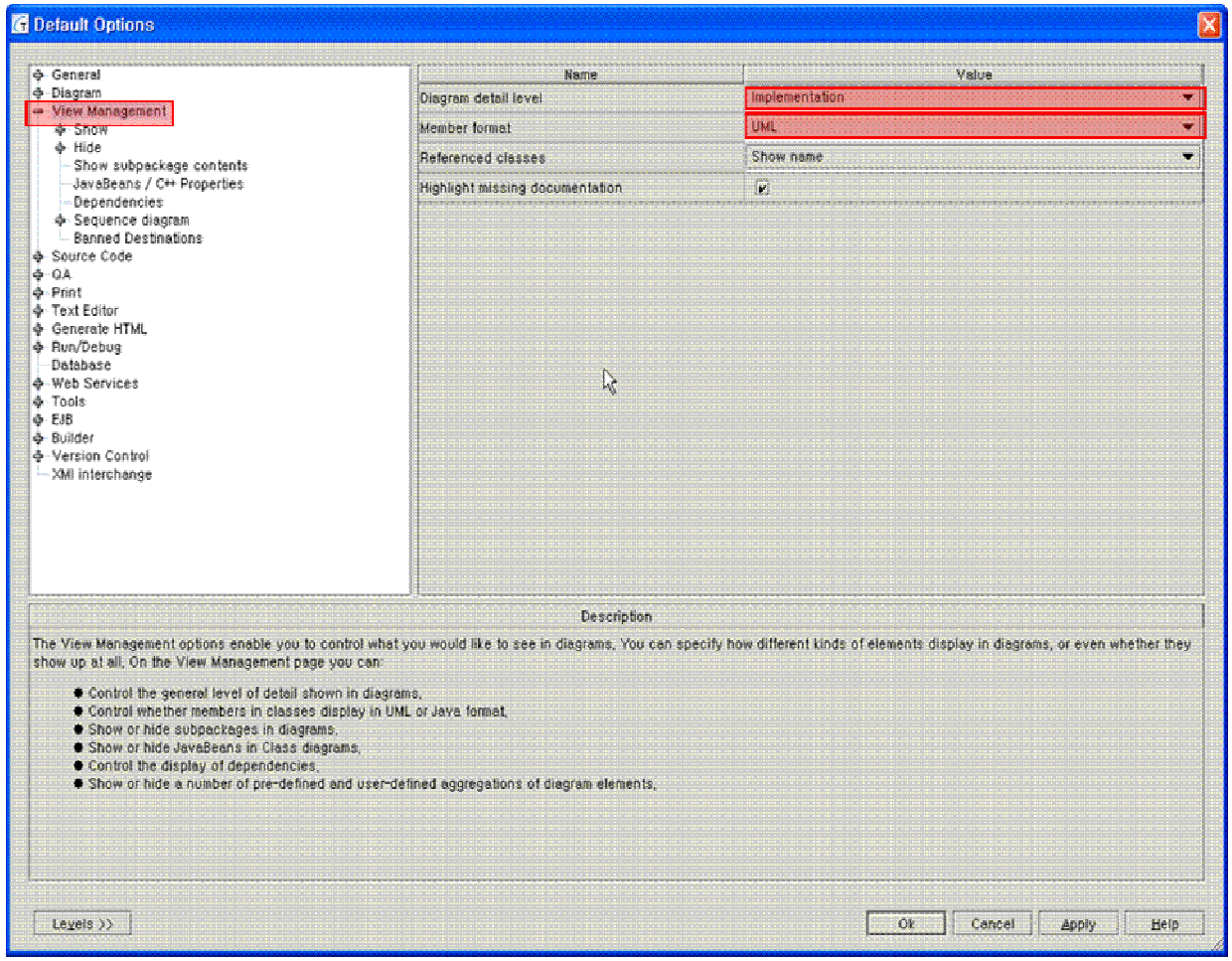
Activity Steps

1) Diagram node를 선택
2) “Links”항목이 “Direct”로 설정되어 있는지 확인
3) “Generate metafiles when Saving diagrams”가 uncheck되어 있는지 확인
4) 옵션 설정 다이어로그 창에서 OK를 선택 하여 옵션사항을 적용합니다.

OOC
Open Objected Component

Borland®

Together Options 화면



Task 3

UML 또는 Language중 여러분 좋아하는 형태로 Diagram에서의 Member Format을 설정토록 하겠습니다.

Activity Steps

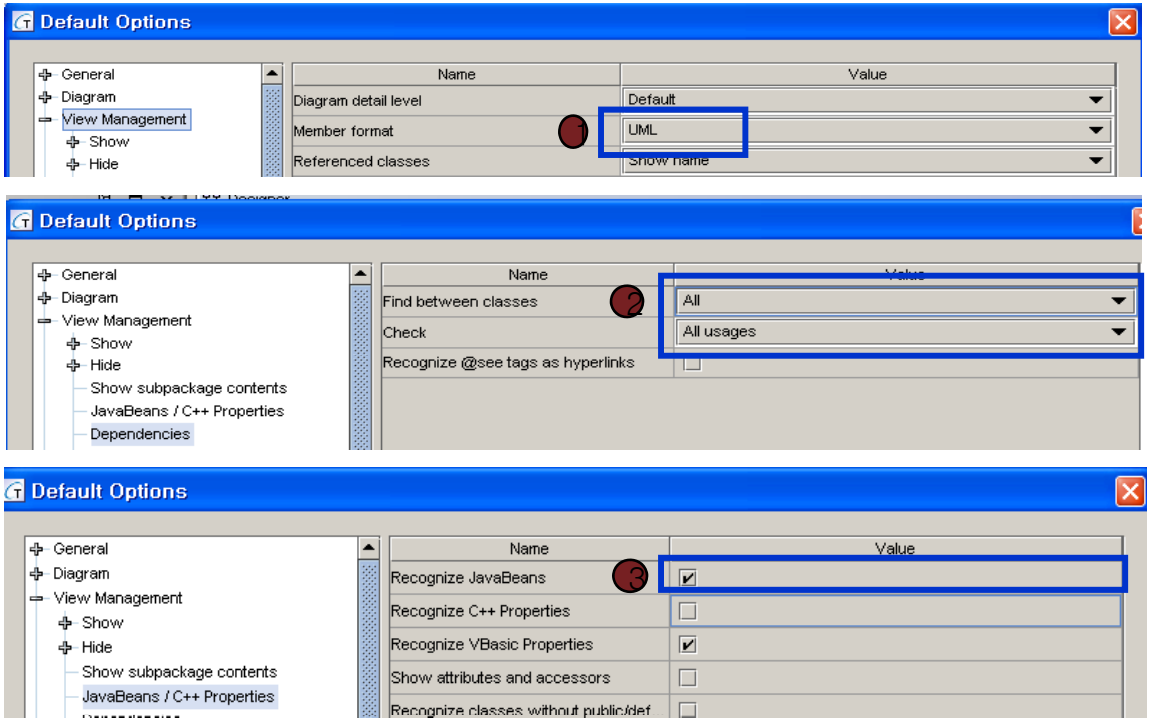
1) "View Management" node를 선택
2) "Diagram detail level"을 "Implementation"으로 설정
3) "Member format"을 "UML"로 설정. UML로 설정하면 name:type 형태로 member들이 다이어그램에 나타나게 됩니다.

OOC
Open Objected Component

Borland®

Together Options 화면

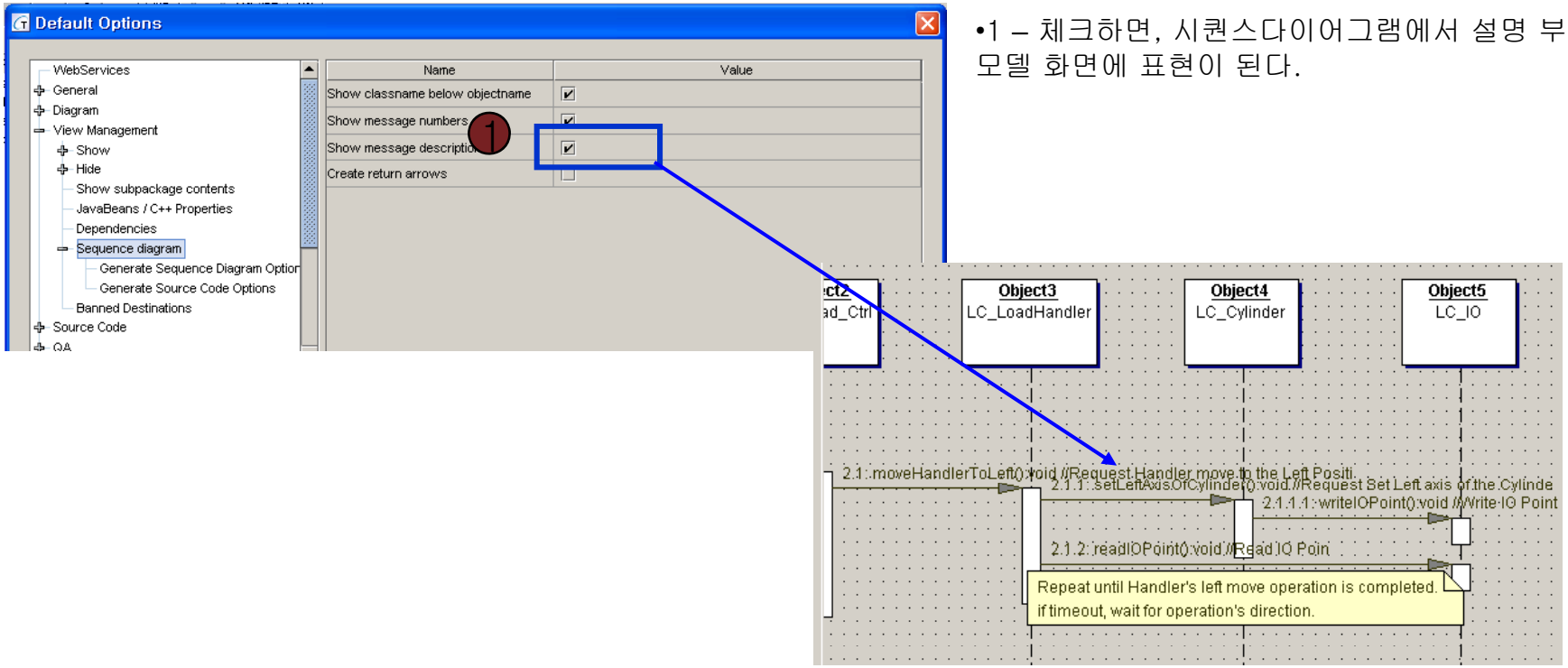
View Management



- 1 – 모델 표기를 UML기반으로 한다.
- 2 – All, All usages로 세팅하면 Dependency가 보인다..
- 3 – unCheck를 하면 메서드가 get, set으로 시작해도 Property로 인식이 안됨. Check를 하면 인식이 된다. 체크하지 말자

Together Options 화면

View Management



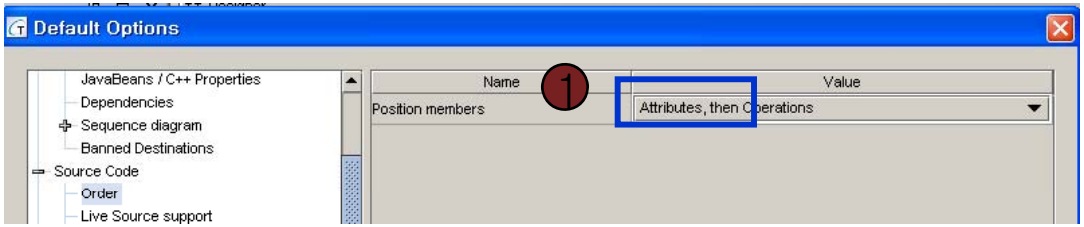
•1 – 체크하면, 시퀀스다이어그램에서 설명 부분이 모델 화면에 표현이 된다.

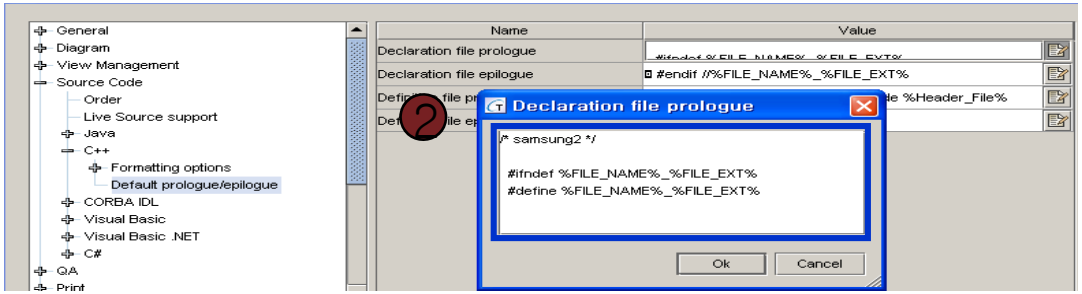
OOC
Open Objected Component

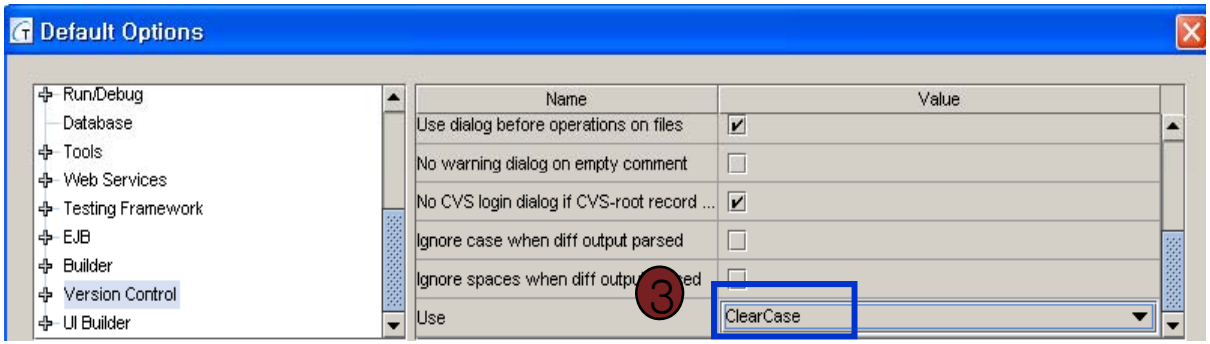
Borland®

Together Options 화면

Source Code-1







- 1 – 모델 표기에서 속성값이 메소드 보다 앞에 나온다.. -> 결정 필
- 2 – 소스 앞에 선언 부분이다. -> 결정 필

■Version Control-1

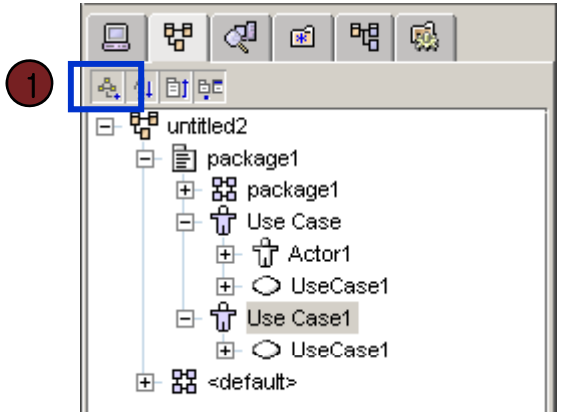
- 3 – ClearCase 연동을 위한 옵션

OOC
Open Objected Component

Borland®

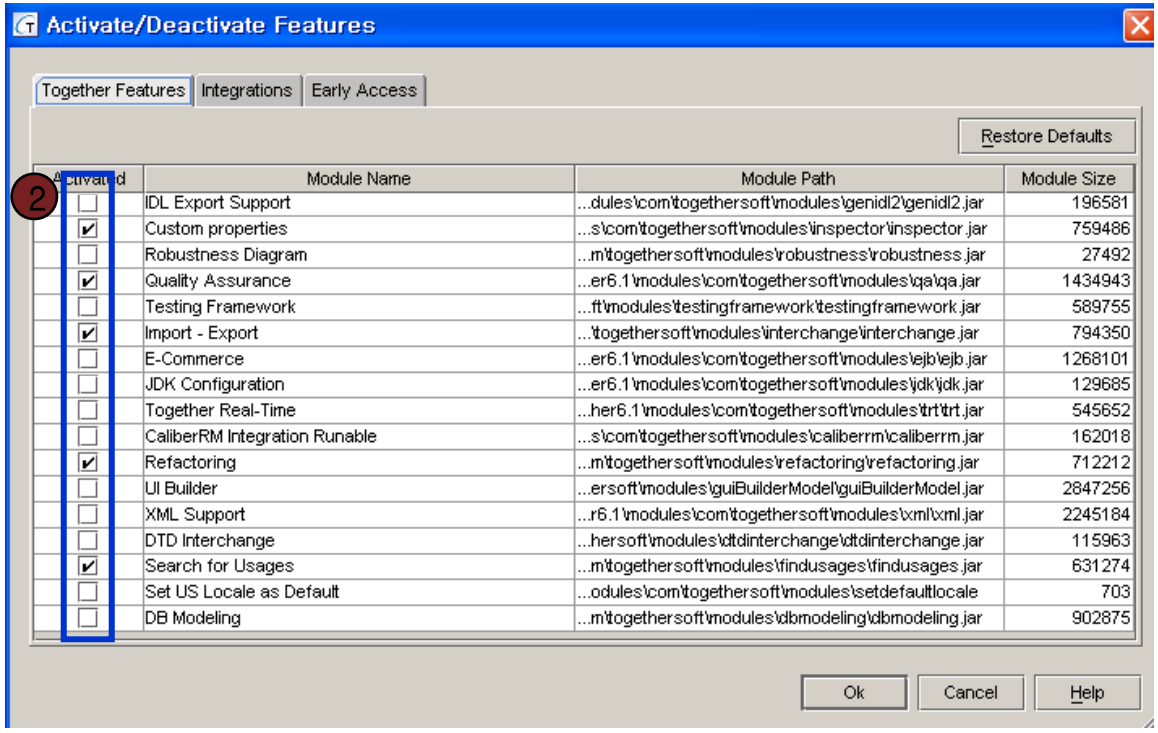
Together Options 화면

기타-> 각자 세팅한다.



•1 – 클릭을 하면, 하위 UML구성요소까지 보인다.

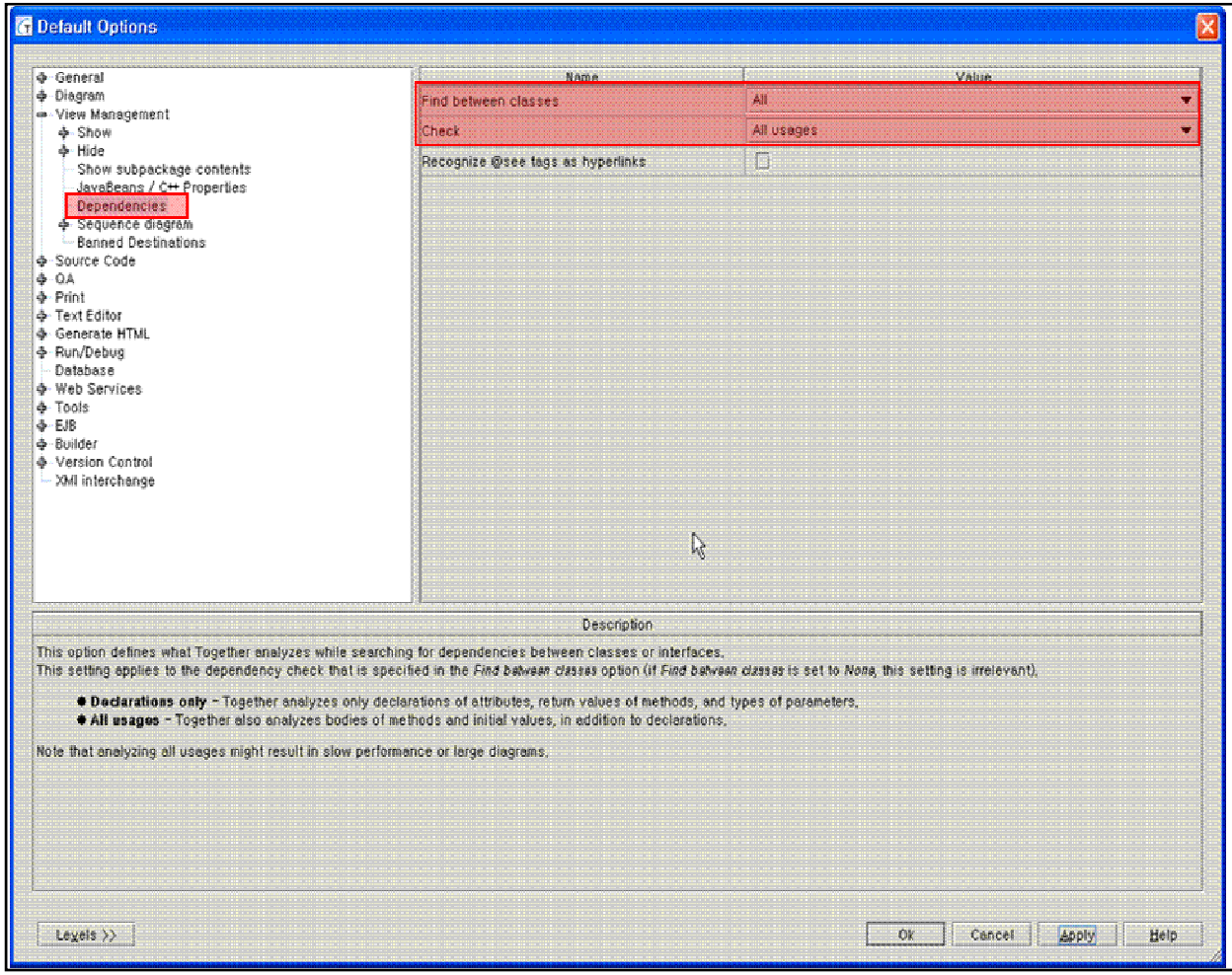
• 2 – 필요한 부분만 체크를 한다.



OOC
Open Objected Component

Borland®

Together Options 화면



Task 3

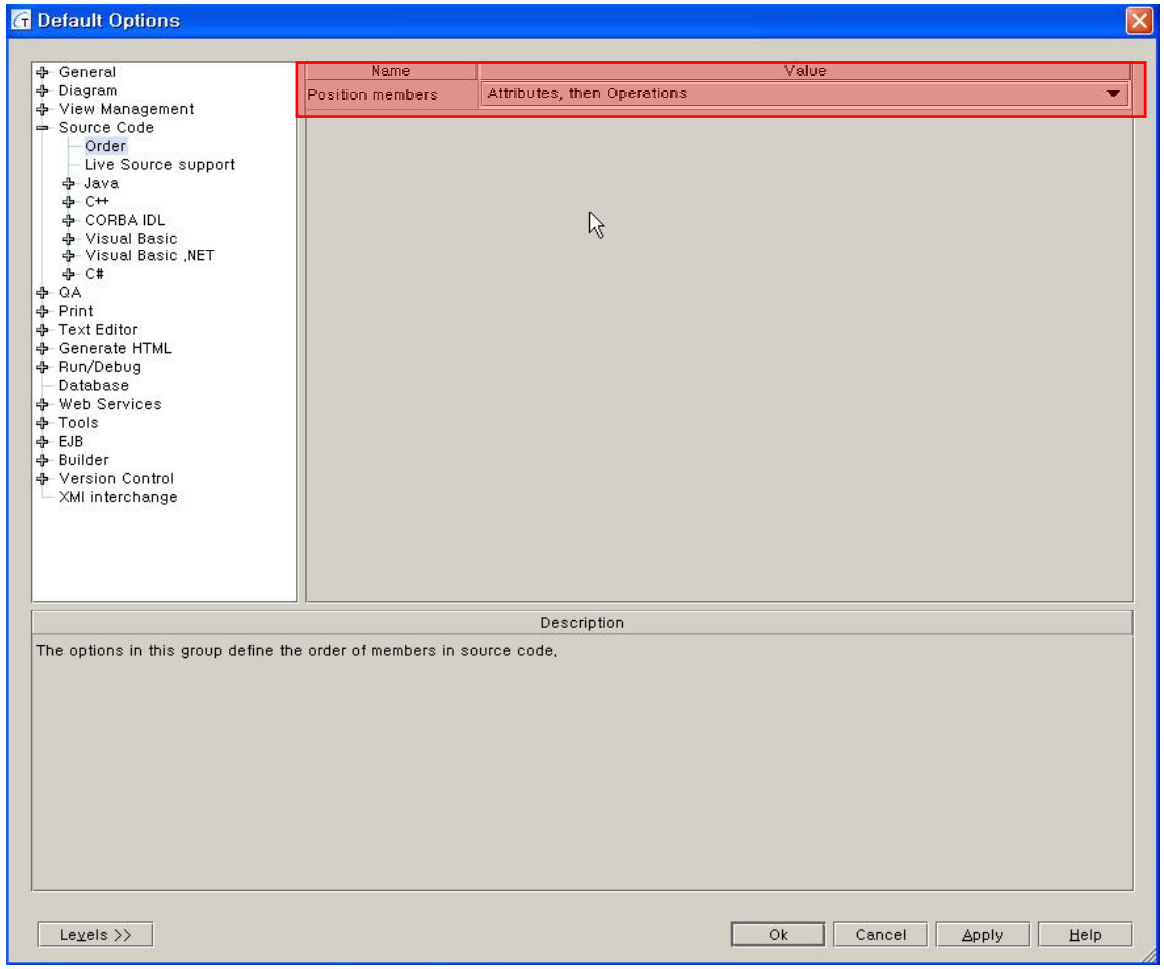
Activity Steps

4) View Management node를 확장하여 Dependencies node를 선택. Class간의 dependency를 모든 Diagram level에서 찾아서 다이어그램에 표시하기 위해서 “Find between classes”를 “All”로 설정. Class와 interface의 정의가 맞게 되었는지를 Check하기 위해서 “Check”를 “All usages”로 설정

OOC
Open Objected Component

Borland®

Together Options 화면



Task 4

소스코드에서 Attributes와 operations의 위치를 초기 설정에서 바꾸기

Activity Steps

1) Default options dialogue 열기. Main menu 로부터 “Tool | Options | Default Level”을 선택

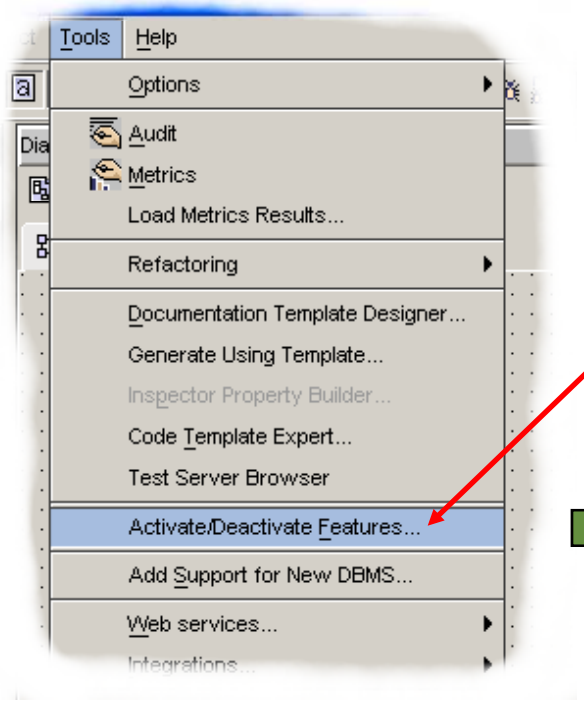
2) “Source Code”를 확장하여 “Source Code | Order”를 선택

3) “Position Members”에서 “Attributes, then Operations”로 설정. 그리고 Task3 가 끝날때까지 Default Options 다이얼로그를 닫지 마십시오.

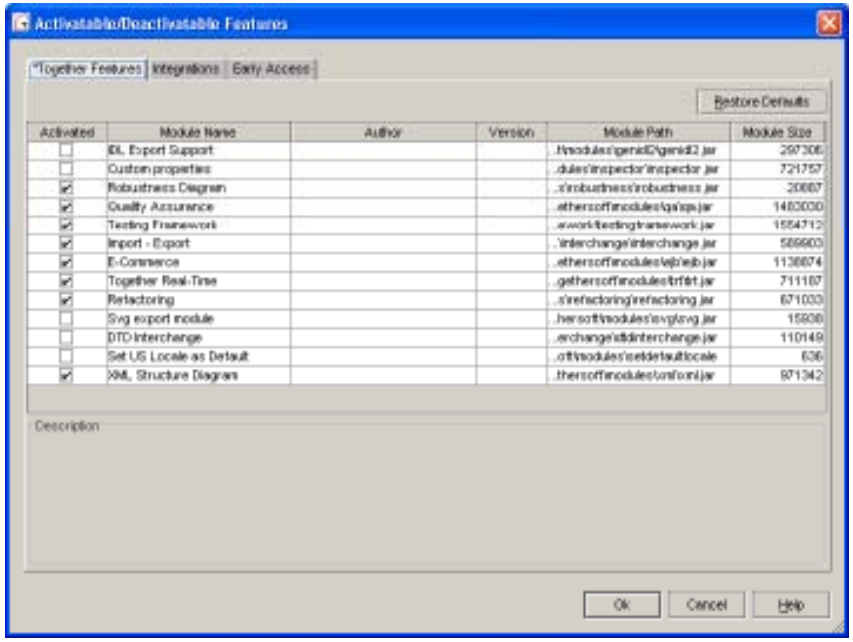
OOC
Open Objected Component

Borland®

Activating Features 화면



Optional Features



Together core 파일

Borland®

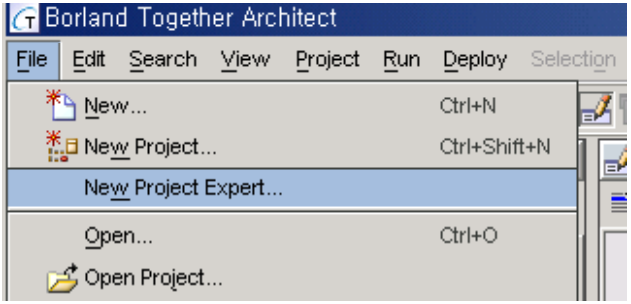
투게더 core 파일

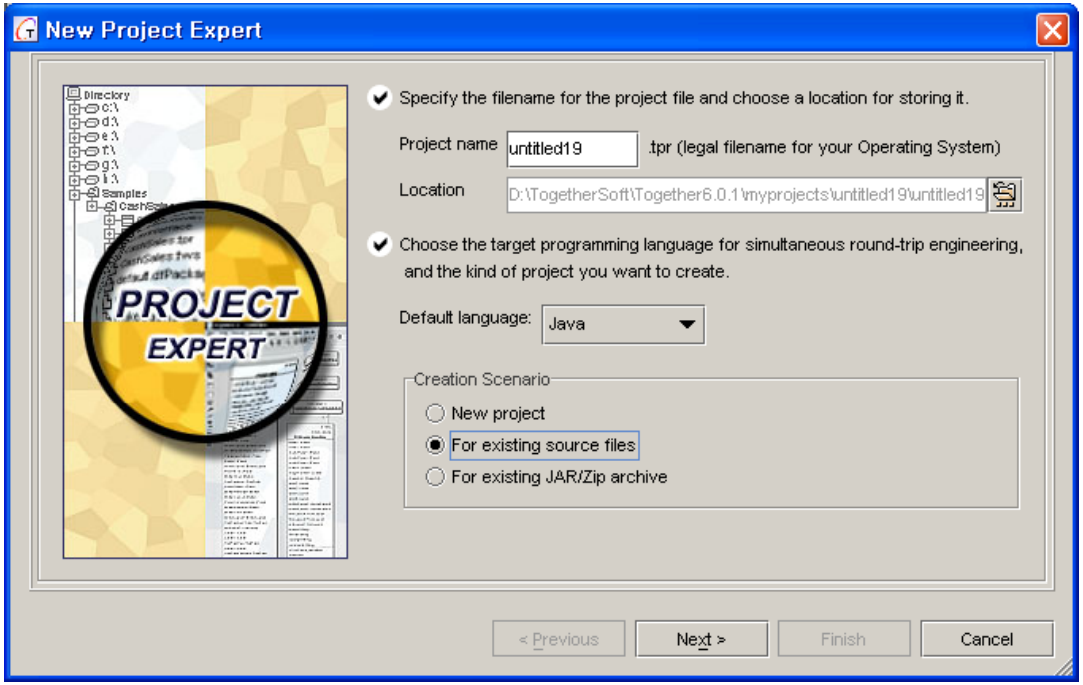
•Together 모델 데이터	•버전관리 여부	•기타
•~~~.tpr	•버전관리 하지 않음	•투게더 전반적인 환경에 대한 정보
•~~~.tws	•버전관리 하지 않음	•워크스페이스 정보가 들어 있다.
•~~~.wmf	•버전관리 하지 않음	•이미지 파일이다. 이 파일은 없어도 다시 생성해준다.
•~~~.dfXXXX	•.dfClass 만 제외하고 버전관리를 한다.	•모델 데이터이다.

Together 기능 소개

Borland®

Reverse 기능



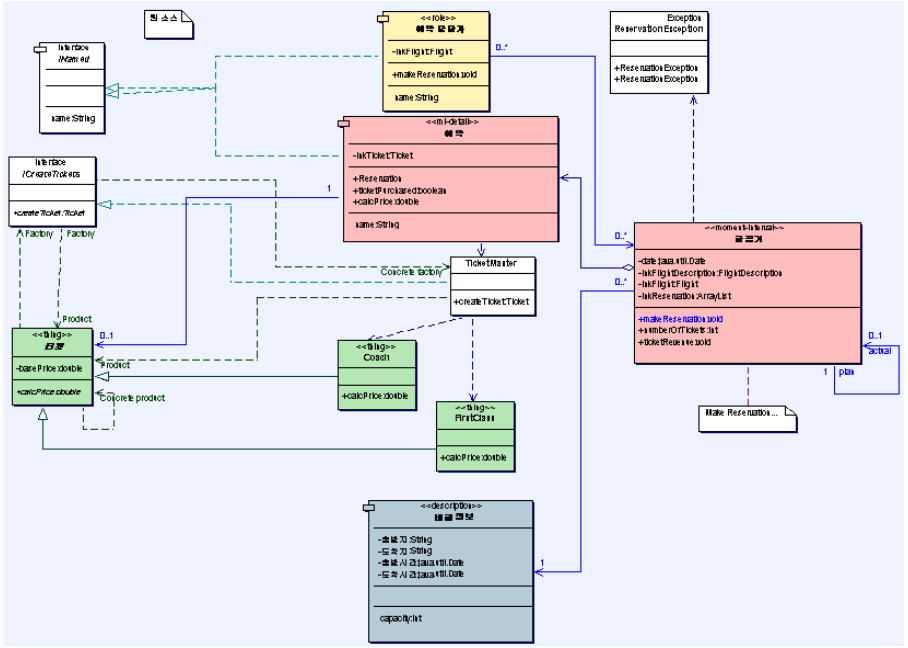


- UML 1.4 기반의 소스 Generation
- 순공학 – Class Diagram, Sequence Diagram, Collaboration Diagram
- 역공학 – Class Diagram, Sequence Diagram, Collaboration Diagram

OOC
Open Objected Component

Borland®

Reverse 기능



Incremental Code generator

Forward Engineering

이름	크기	종류
AirlinePD.dfPackage	9KB	DFPACKAGE 파일
FindRevenue.dfSequence	7KB	DFSEQUENCE ...
Flight.makeReservation(1).dfSe...	19KB	DFSEQUENCE ...
Agent.java	1KB	JAVA 파일
Coach.java	1KB	JAVA 파일
FirstClass.java	1KB	JAVA 파일
Flight.java	2KB	JAVA 파일
FlightDescription.java	1KB	JAVA 파일
ICreateTickets.java	1KB	JAVA 파일
INamed.java	1KB	JAVA 파일
Reservation.java	2KB	JAVA 파일
ReservationException.java	1KB	JAVA 파일
Ticket.java	1KB	JAVA 파일
TicketMaster.java	1KB	JAVA 파일
AirlinePD.dfPackage.wmf	40KB	WMF 파일
FindRevenue.dfSequence.wmf	12KB	WMF 파일
Flight.makeReservation(1).dfSe...	29KB	WMF 파일
ProblemDomain.dfPackage.wmf	4KB	WMF 파일

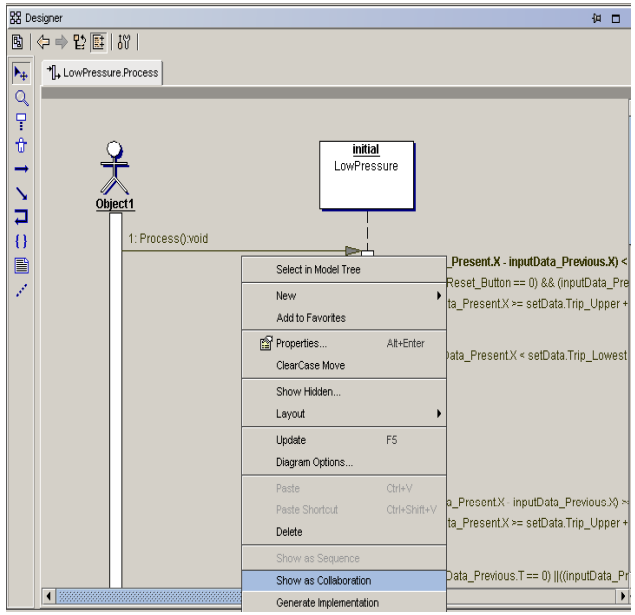
Reverse Engineering

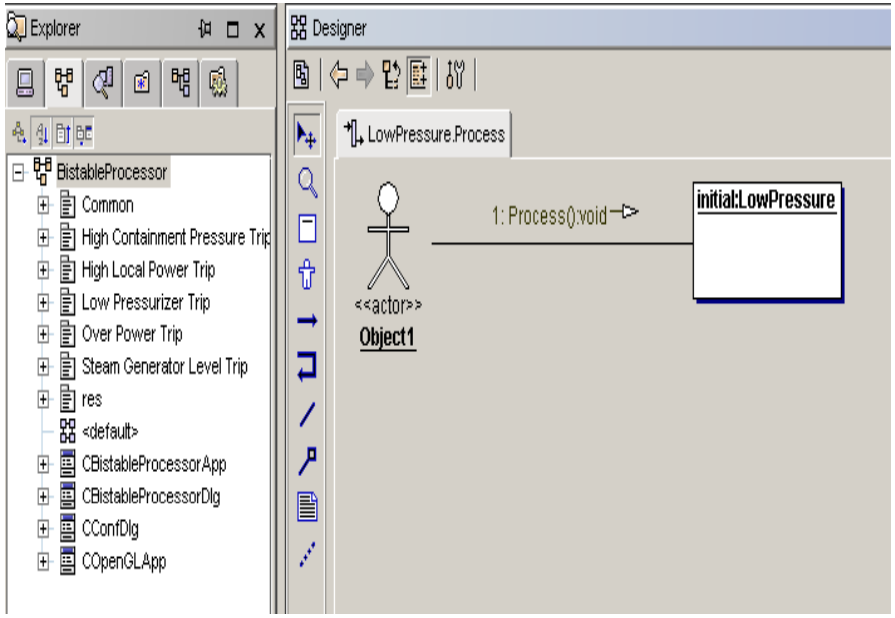
Fuzzy-Parser

OOC
Open Objected Component

Borland®

Reverse





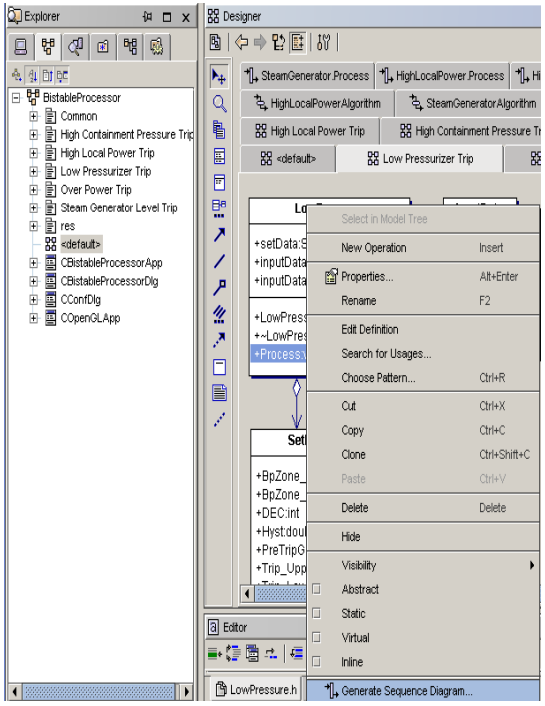
Sequence Diagram

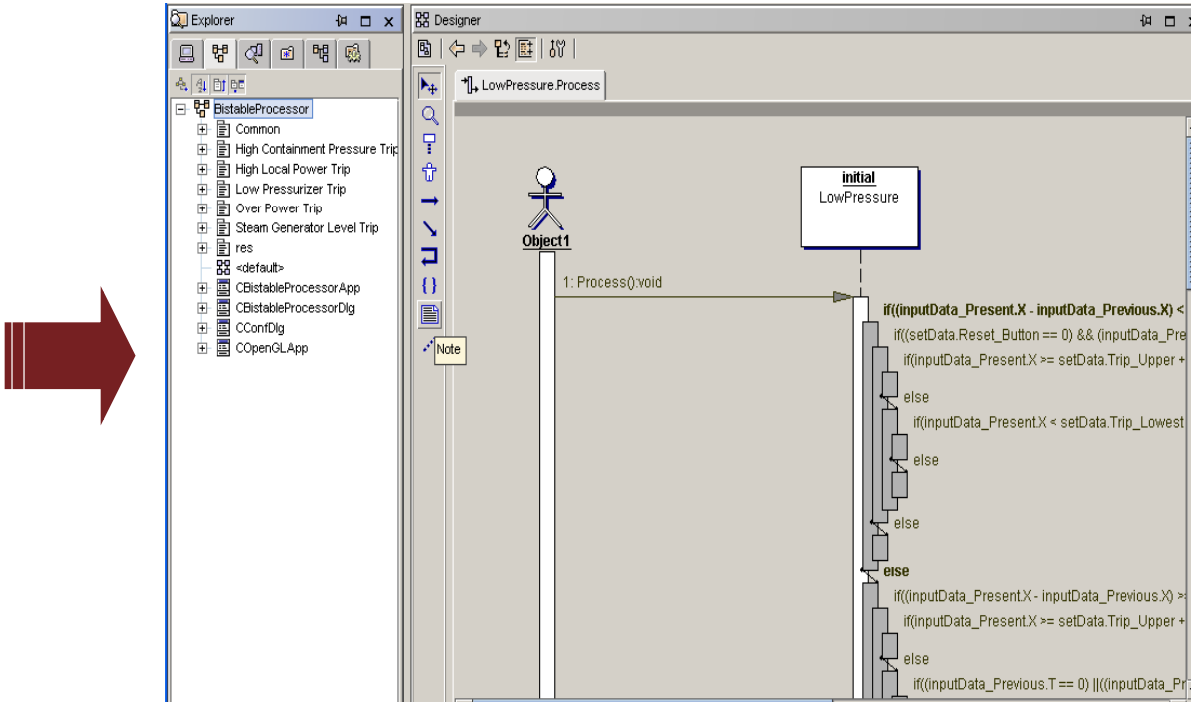
Collaboration Diagram

OOC
Open Objected Component

Borland®

Reverse





Clas Diagram :
Operation

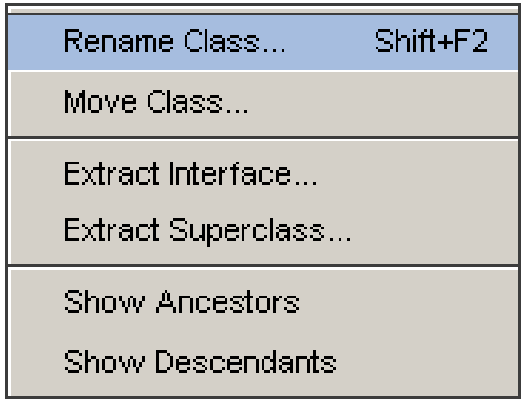
Sequence Diagram
→ Class의 Operation Logic

OOC
Open Objected Component

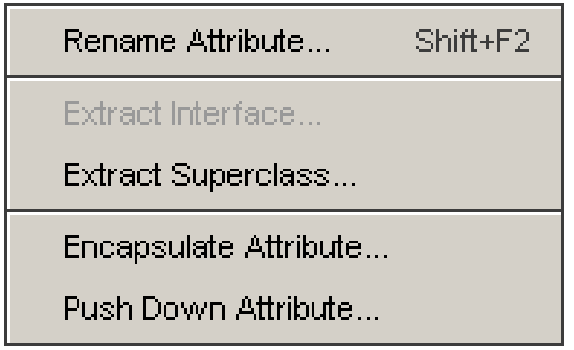
Borland®

Refactoring

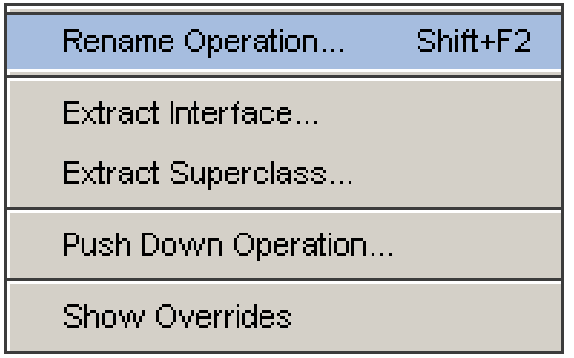
Refactoring [기능 선택하기]



Class Refactoring



Attribute Refactoring



Operation Refactoring

OOC
Open Objected Component

Borland®

Audit & Metrics

Audit

- 소스 코드에 대한 코딩 스타일 및 소스에서 사용되는 프로그램변수의 이름 규칙, 선언 스타일, 주석화 정도, 주요 에러를 검사하고 자동으로 수정

Metrics

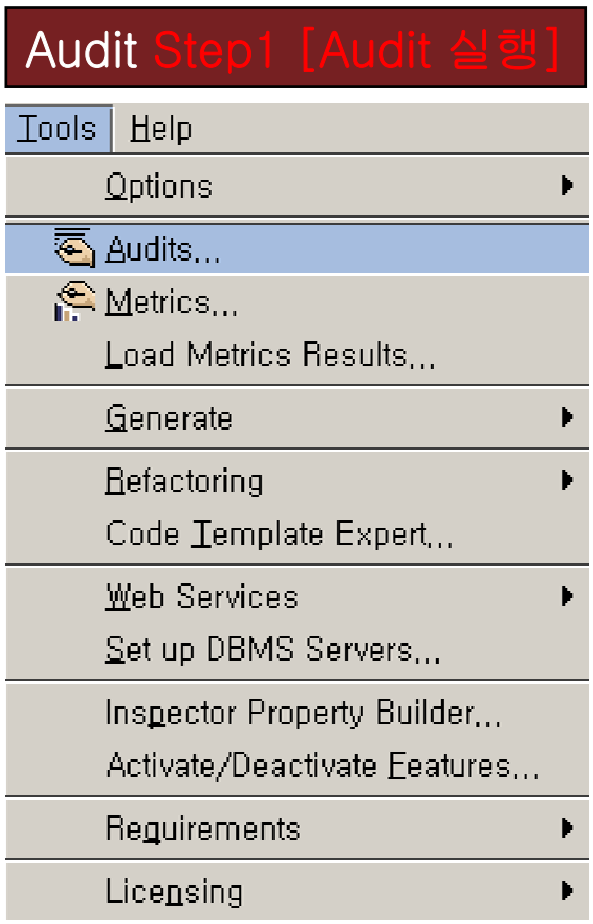
- 소스 코드에 품질을 측정하기 위하여 코드의 물리적인 정보에서부터 논리적인 정보인 소스 코드의 응집도, 복잡도, 결합도, 켑슐화, 다형성 과 같은 정보를 추출하여 일정한 수치로 표현

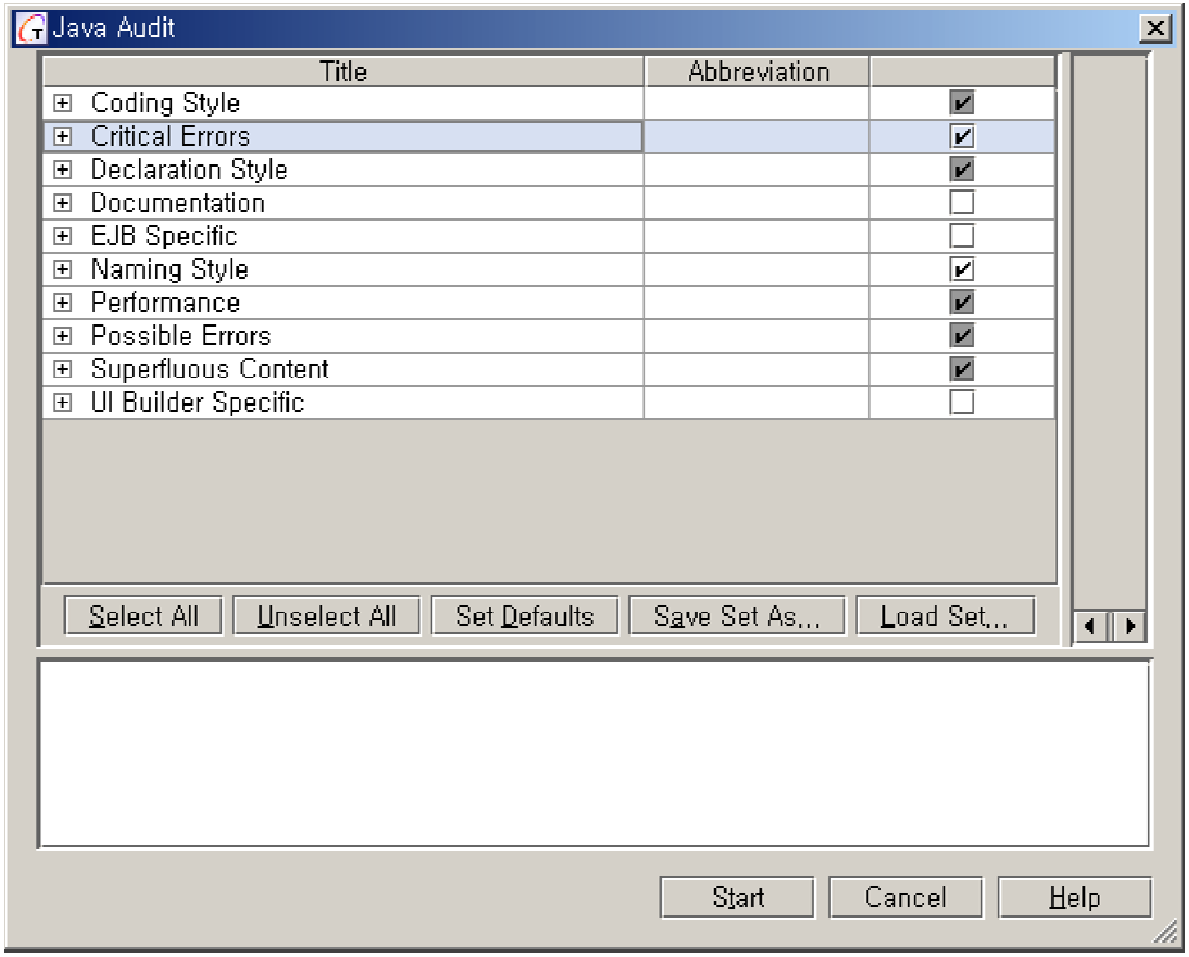
OOC
Open Objected Component

Borland®

Audit & Metrics

Audit Step1 [Audit 실행]



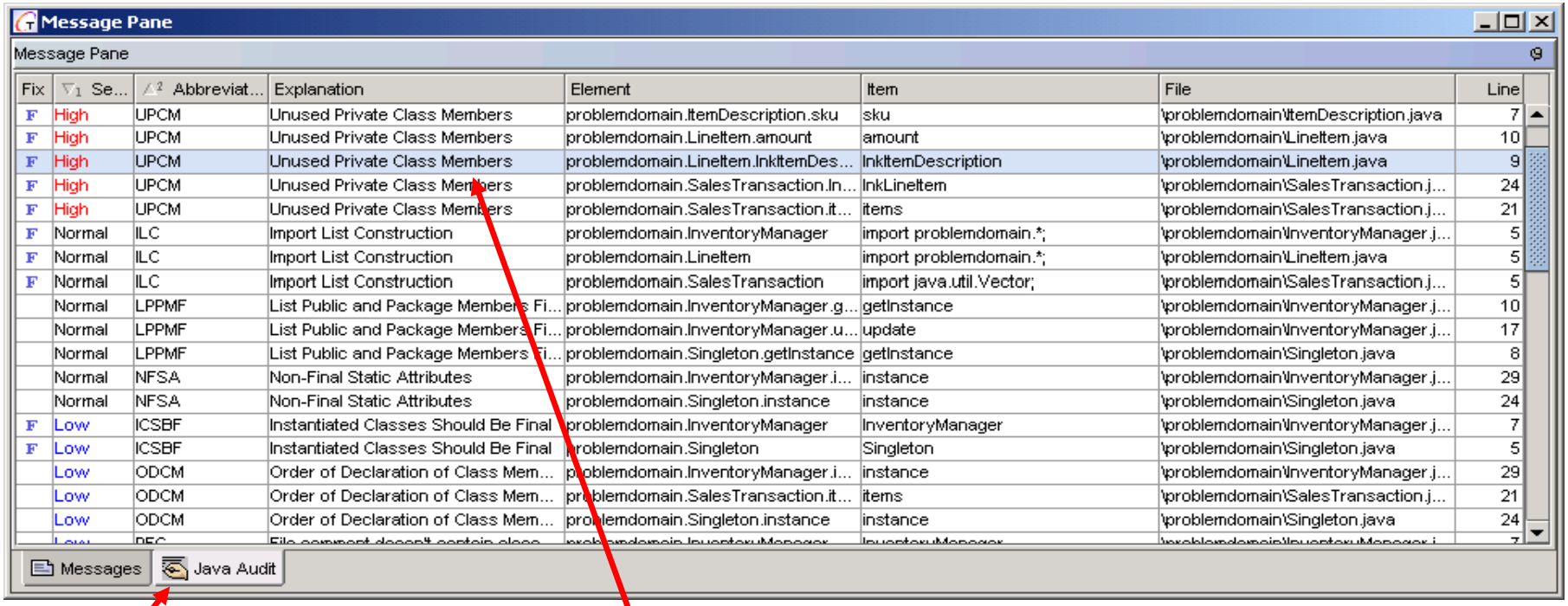


OOC
Open Objected Component

Borland®

Audit & Metrics

Audit Step2 [Audit 결과]



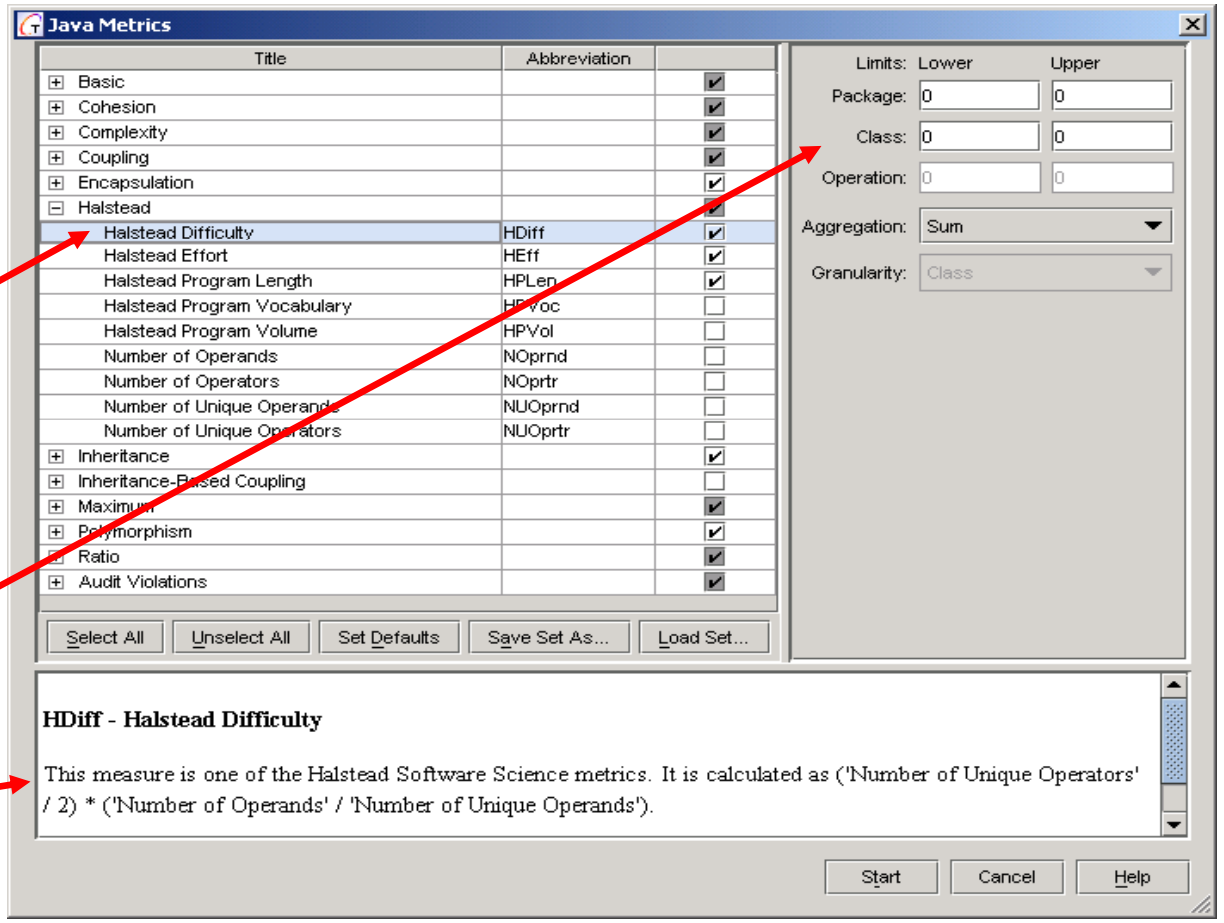
Audit tab

Selecting audit element displays source element in editor

OOC
Open Objected Component

Borland®

Audit & Metrics



Metric

Thresholds, options

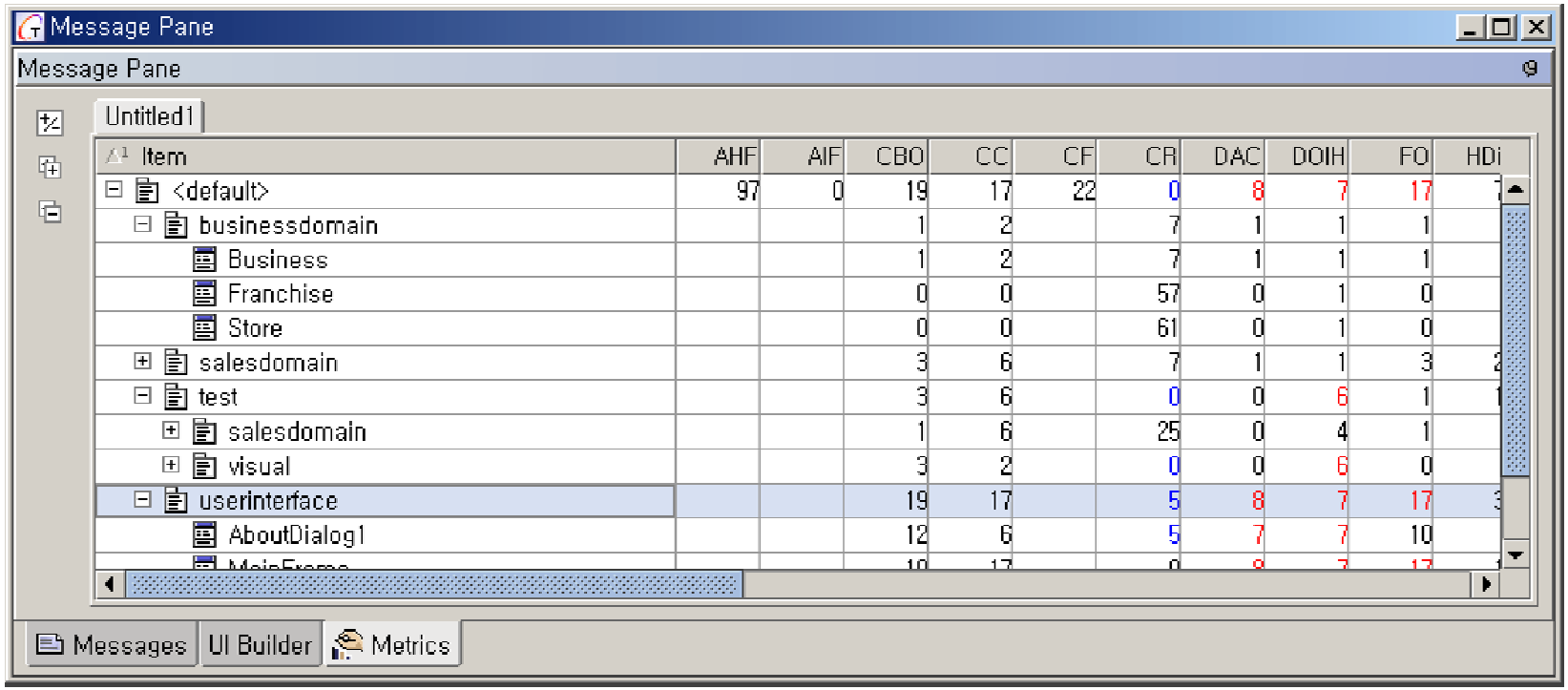
Description

OOC
Open Objected Component

Borland®

Audit & Metrics

Metrics [Metrics 실행 결과]



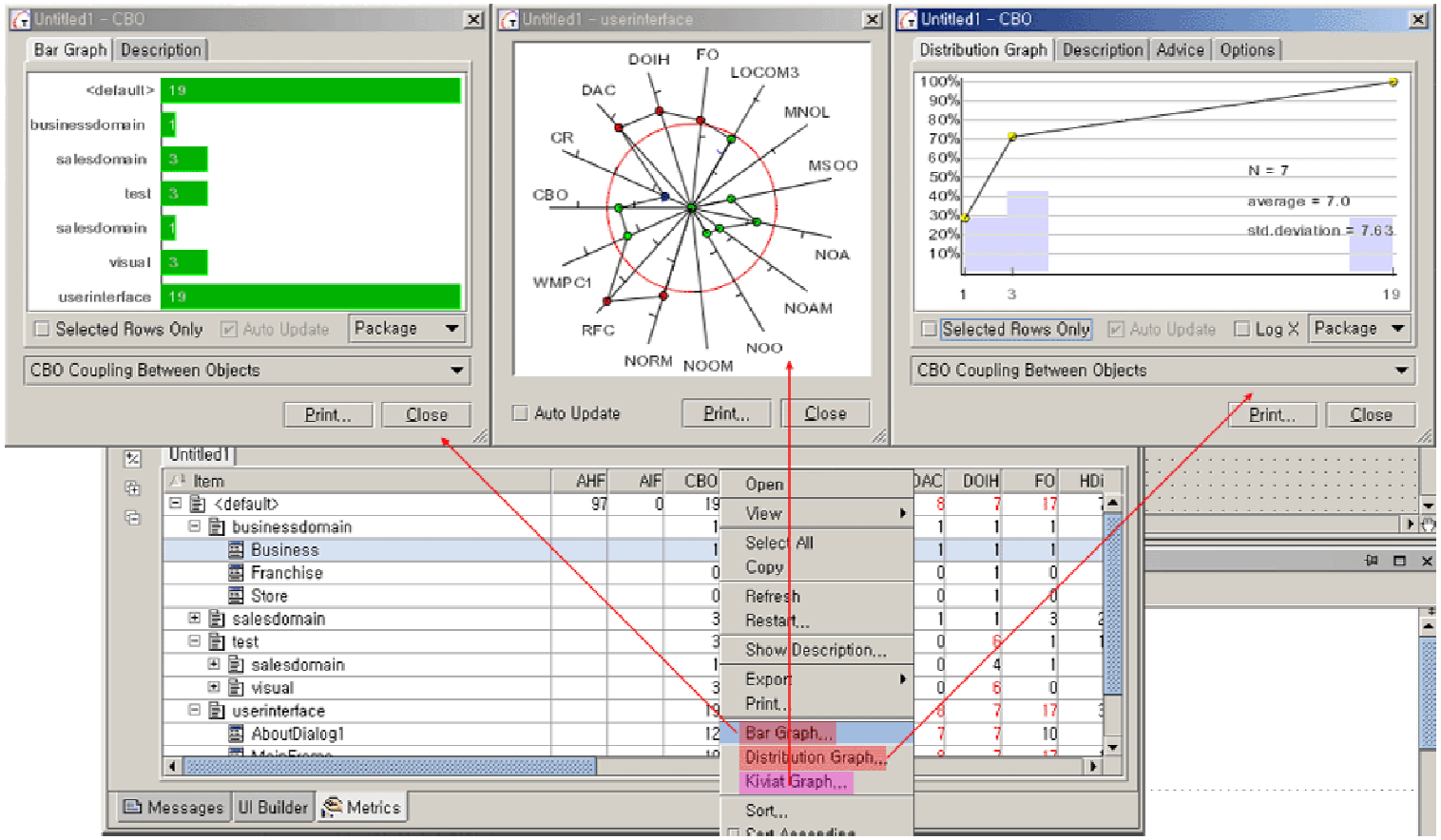
Item	AHF	AIF	CBO	CC	CF	CR	DAC	DOIH	FO	HDi
<default>	97	0	19	17	22	0	8	7	17	
businessdomain			1	2		7	1	1	1	
Business			1	2		7	1	1	1	
Franchise			0	0		57	0	1	0	
Store			0	0		61	0	1	0	
salesdomain			3	6		7	1	1	3	
test			3	6		0	0	6	1	
salesdomain			1	6		25	0	4	1	
visual			3	2		0	0	6	0	
userinterface			19	17		5	8	7	17	
AboutDialog1			12	6		5	7	7	10	

OOC
Open Objected Component

Borland®

Audit & Metrics

Metrics [Metrics 결과를 다양한 다이어그램으로 보기]



OOC
Open Objected Component

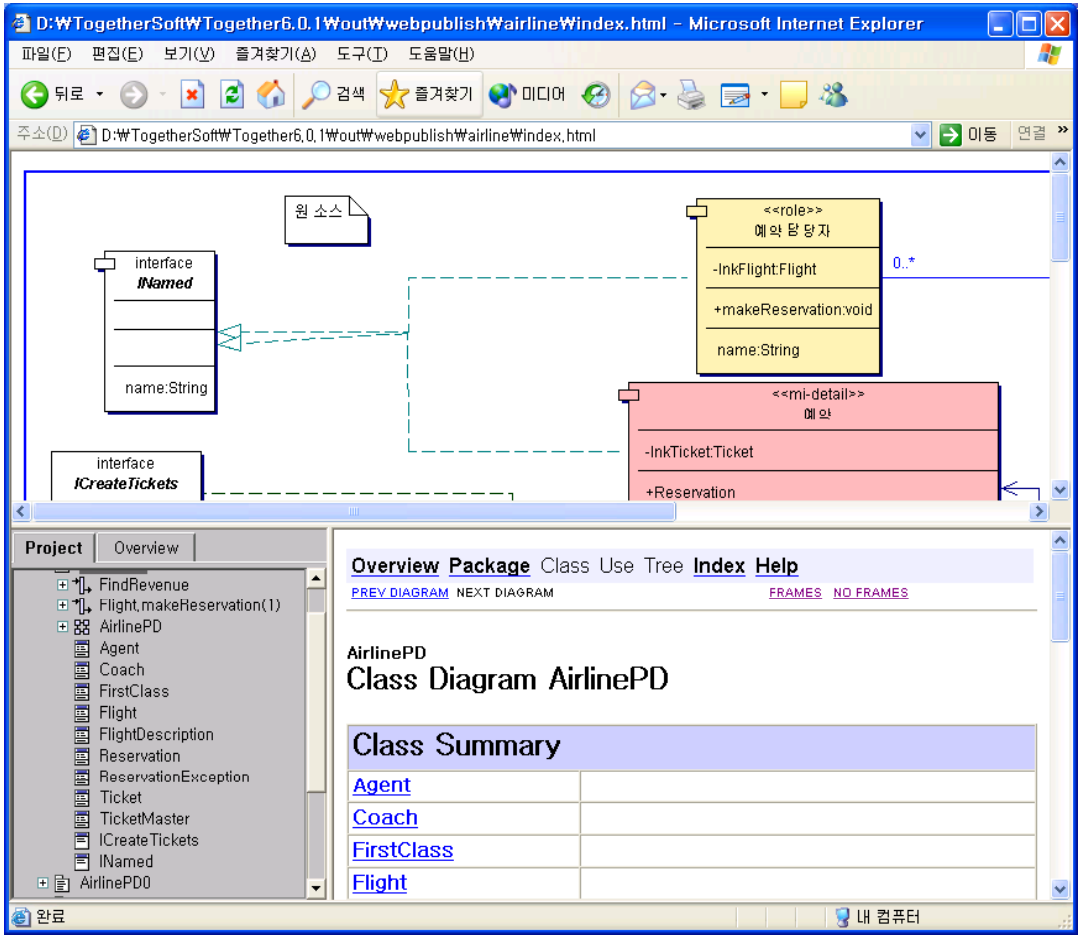
Borland®

Documentation

HTML Page 자동생성

모델링된 데이터를 자동으로 생성한다.

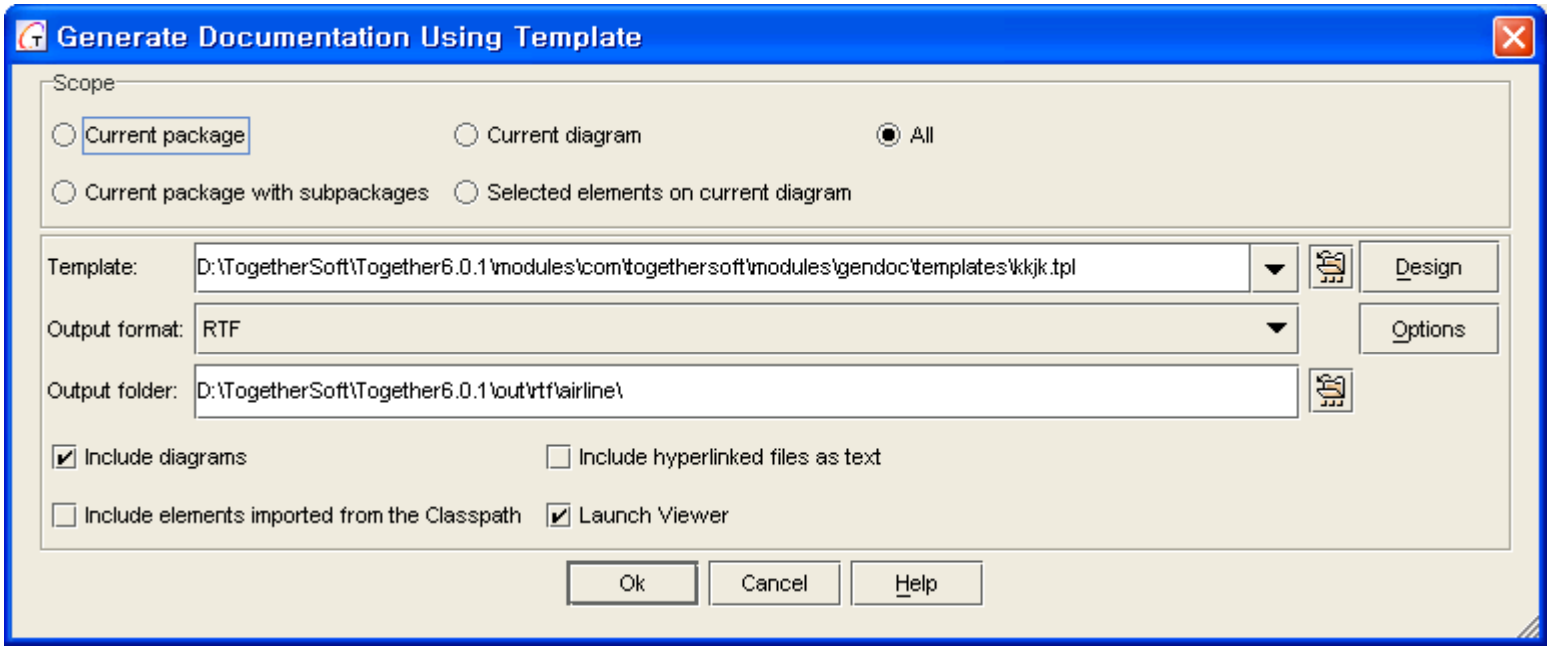
1. HTML Page
2. MS-Word (xxx.rtf) 파일
3. Adobe PDF 파일



OOC
Open Objected Component

Borland®

Documentation

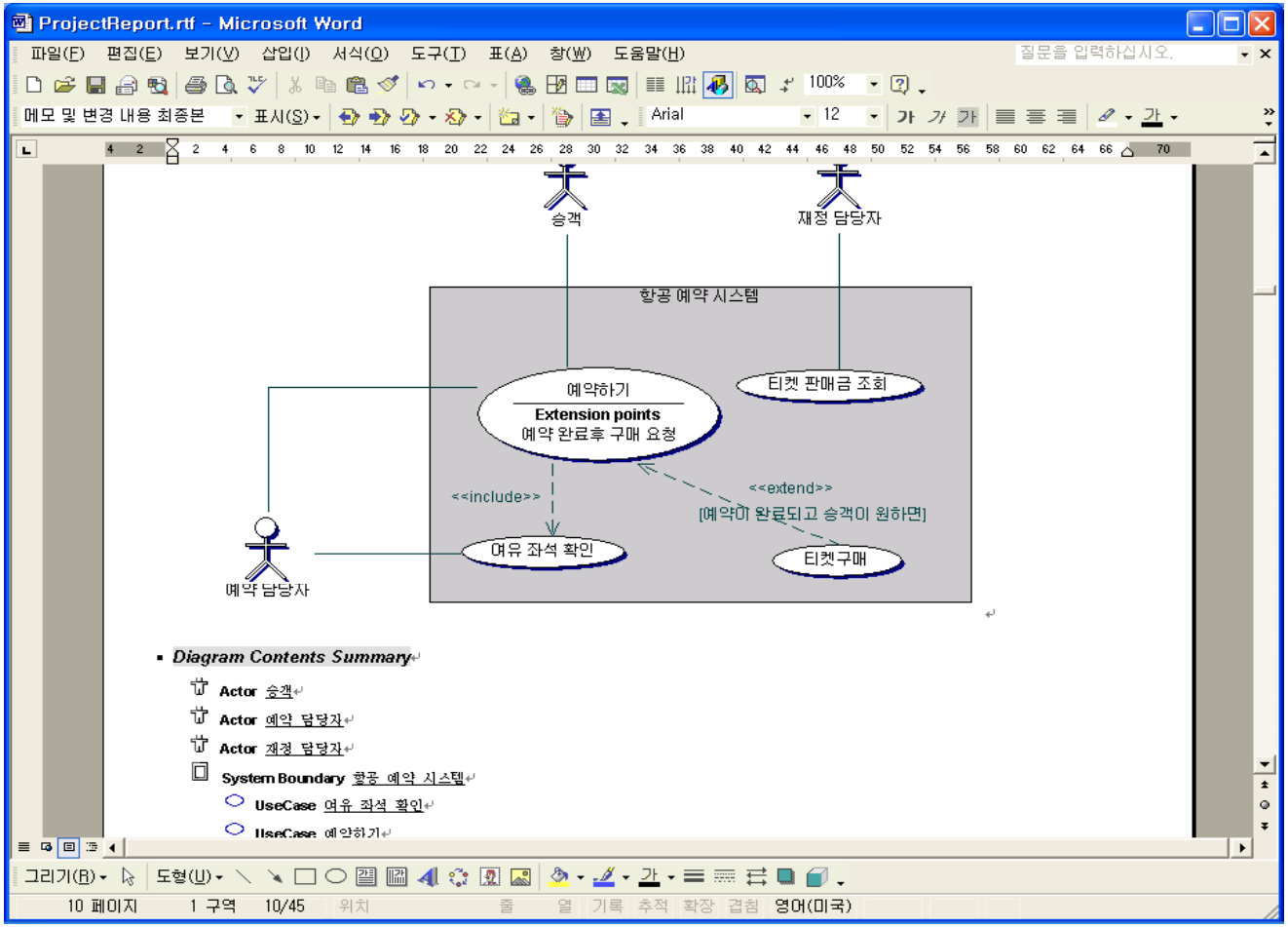


Template 실행

Documentation

MS-Word 문서 자동생성

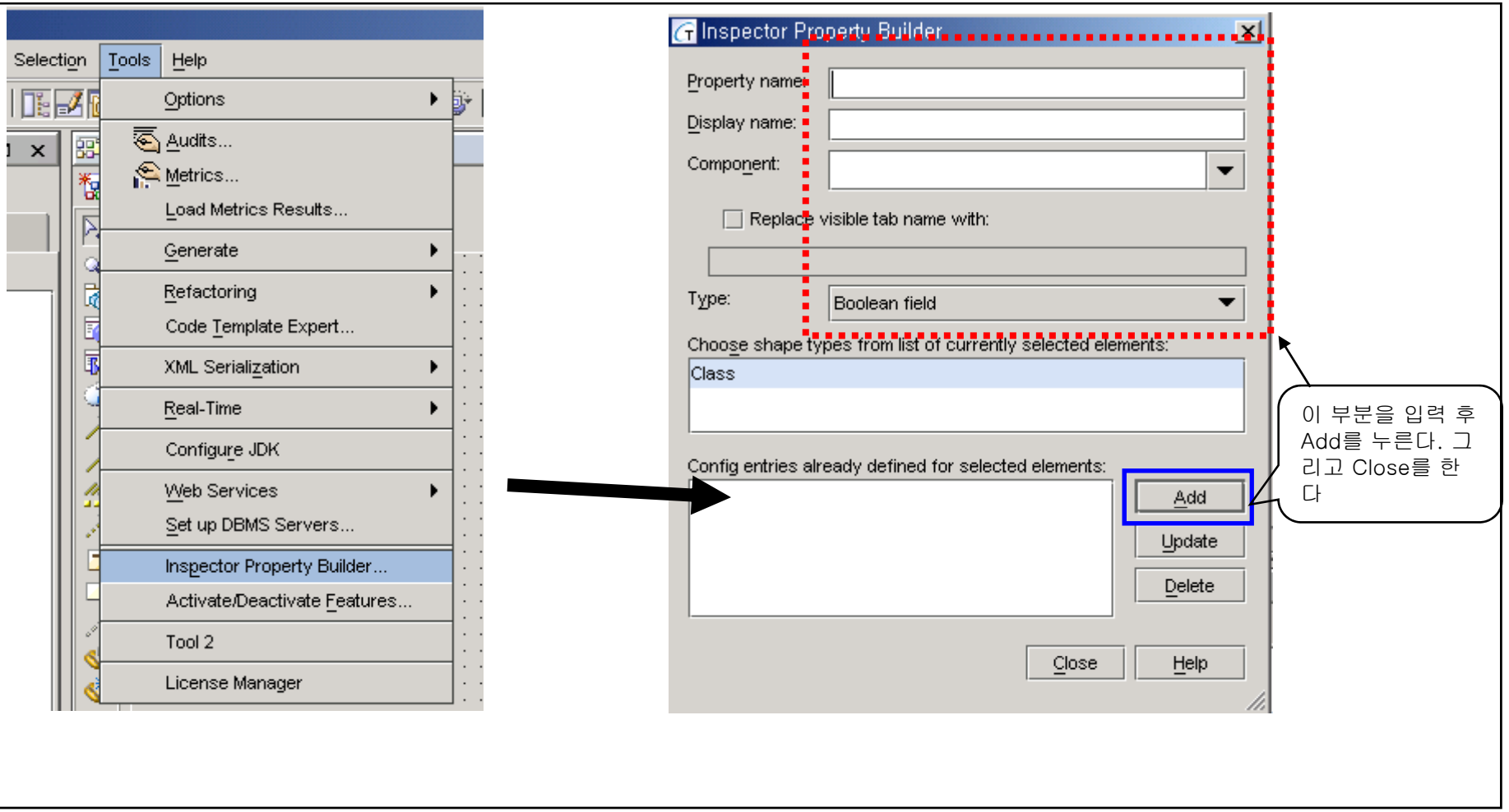
1. ProjectReport.rtf 문서 자동 생성
2. 문서의 사용자 정의 커스터마이징
3. 유지보수의 편리성



OOC
Open Objected Component

Borland®

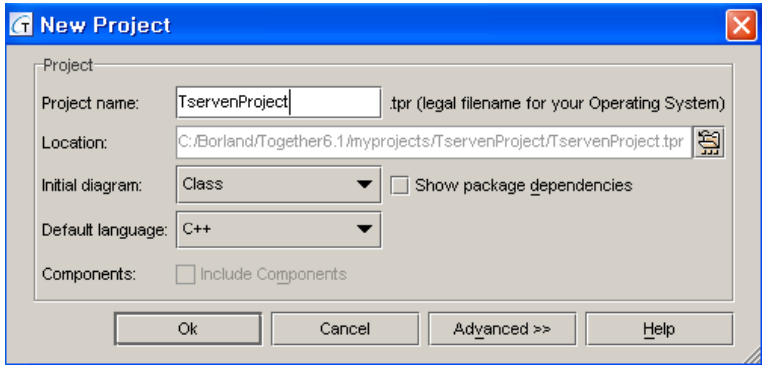
사용자 정의 Property추가



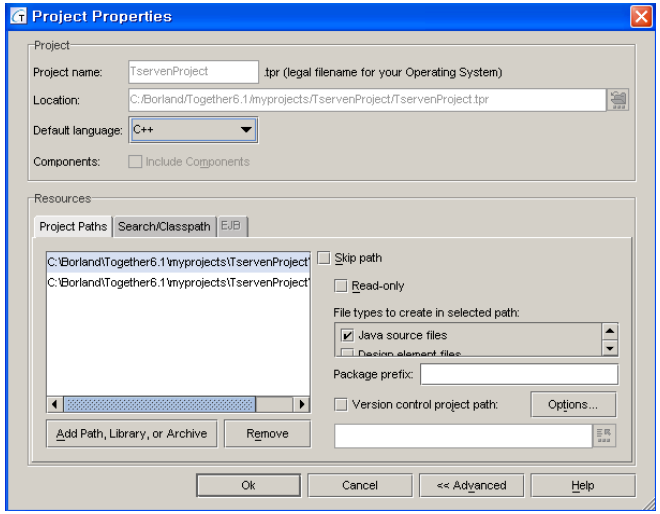
프로젝트 생성

프로젝트 이름생성

- 이름 입력



- 프로젝트 Property 세팅



OOC
Open Objected Component

Borland®

UML 따라하기

Borland®

다이어그램 작성시 주의 사항 및 알아야 할 사항 1

Clone과 shortcut차이

- Clone은 모델을 완전히 새롭게 만드는 것이다.
- Shortcut은 기존 모델을 참조만 하는 것이다.

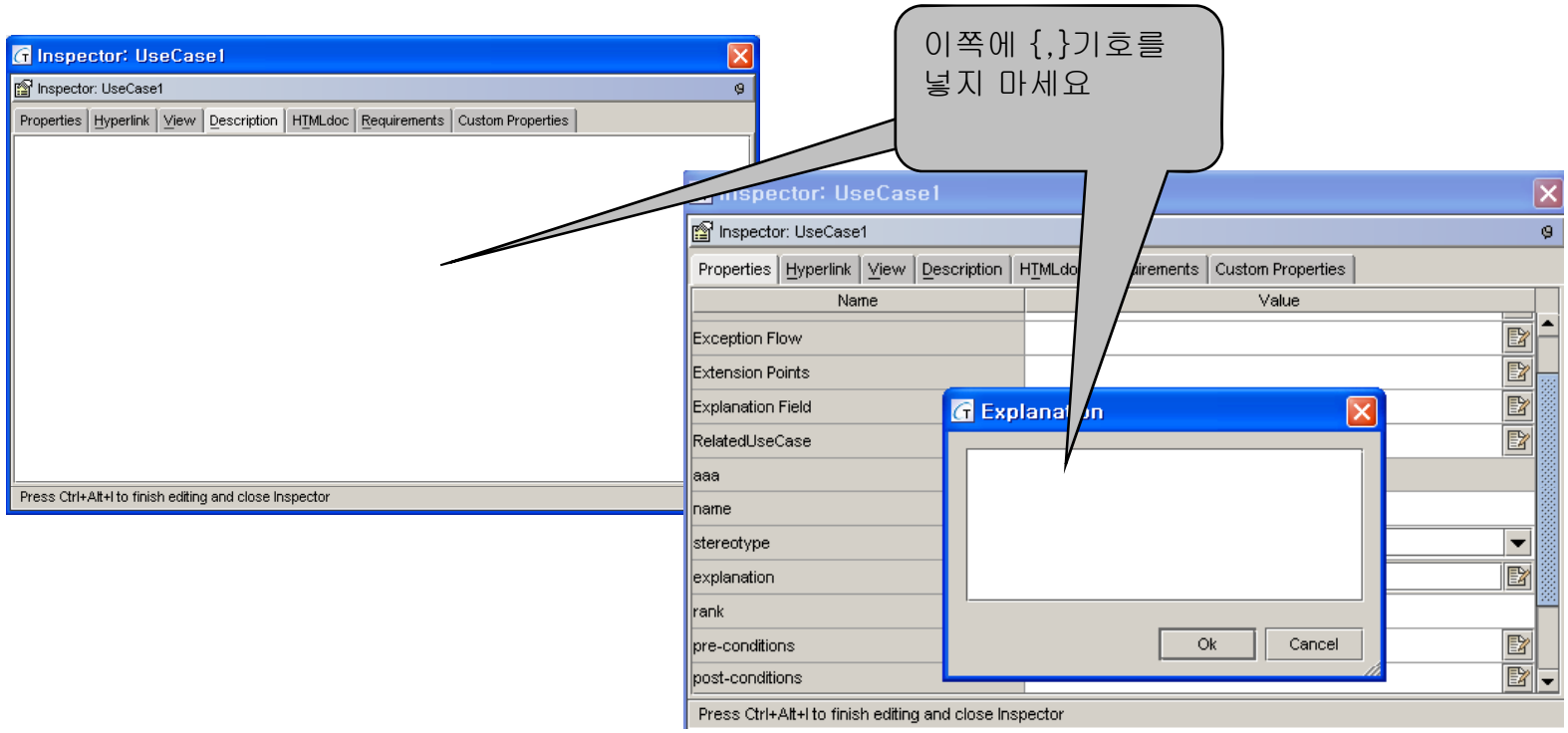
투게더 각 다이어그램은 하나의 아이디로 관리 된다.

- 반드시 모델데이터(df~~~)에 대한 아이디는 한 프로젝트에서 유일해야 한다.
- 투게더에서는 모델 데이터를 하나의 df~~~~~`파일에서 관리 한다. 그러므로 그 파일만 물리적으로 가져와서 투게더에 로딩하면 모델 데이터가 생성이된다. 단 클래스 다이어그램은 예외(클래스까지 가져와야 한다.)
- 주의점 : 모델 데이터를 누군가에게 줄때는 Clone를 이용해서 준다.

Copy & Paste 시 알아야 할 사항 : 모델 정렬 정보는 copy가 되질 않는다.

다이어그램 작성시 주의 사항 및 알아야 할 사항 2

Property관련 데이터 입력시 “{, }”기호는 절대 사용하지 말아야 한다.



OOC
Open Objected Component

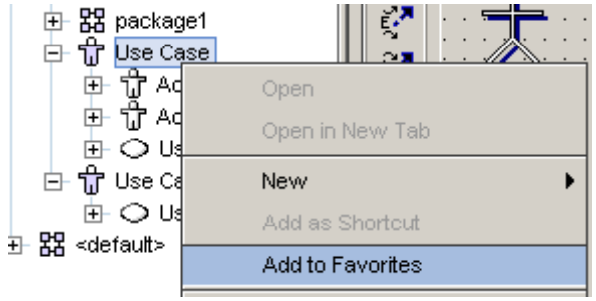
Borland®

다이어그램 작성시 편리한 기능

알면 편리한 기능

- 키워드를 외우자 :
  - Ctrl + A : Attribute(속성) 추가
  - Ctrl + O : Operation 추가
  - Alt + Enter : Inspector – Properties 창 띄우기
  - Ctrl + K : 모델 정렬(단 이쁘게 되질 않는다.)
  - Ctrl + Alt + N : 다이어그램 생성 창 띄우기)
  - Ctrl + Z : undo기능

- 재미있는 기능




```
1  /* samsung2 */
2
3  #ifndef CLASS1_H
4  #define CLASS1_H
5  class Class1 {
6  private:
7                          int attribute1;
8  public:
9
10     void operation1();
11 };
12 #endif //CLASS1_H
```

```
1  /* samsung2 */
2
3  #ifndef CLASS1_H
4  #define CLASS1_H
5  class Class1 {
6  private:
7      int attribute1;
8  public:
9
10     void operation1();
11 };
12 #endif //CLASS1_H
```

소스 정렬(Ctrl + Z)

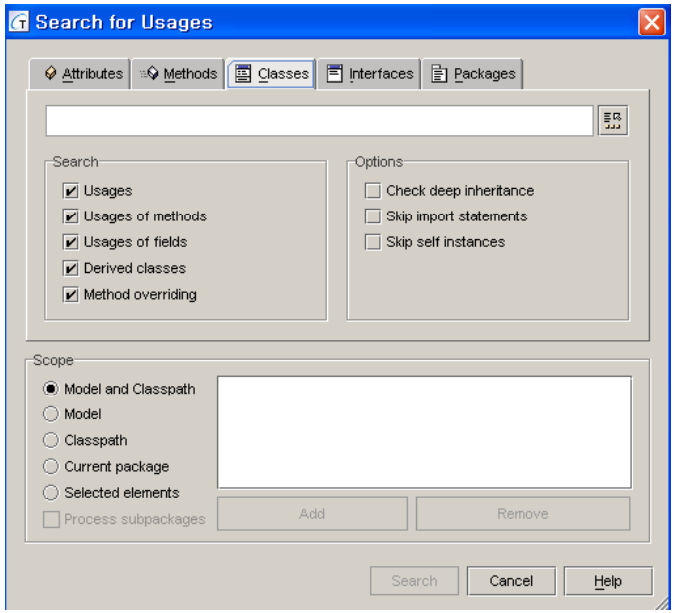
OOC
Open Objected Component

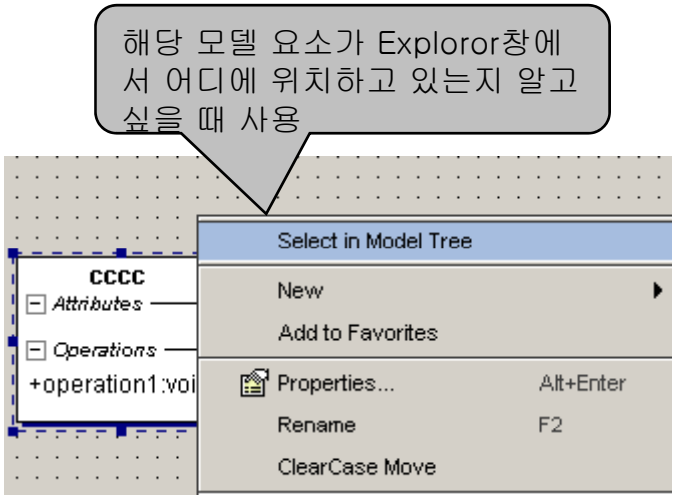
Borland®

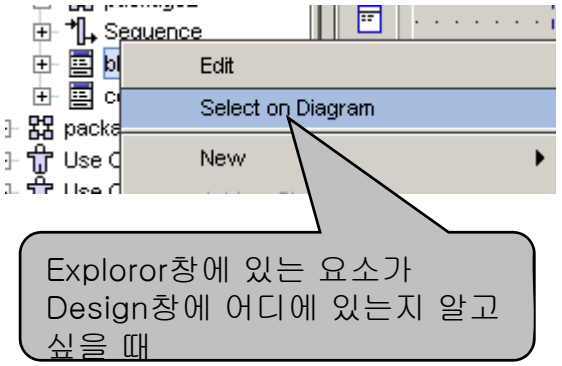
다이어그램 작성시 편리한 기능

알면 편리한 기능

- 재미있는 기능





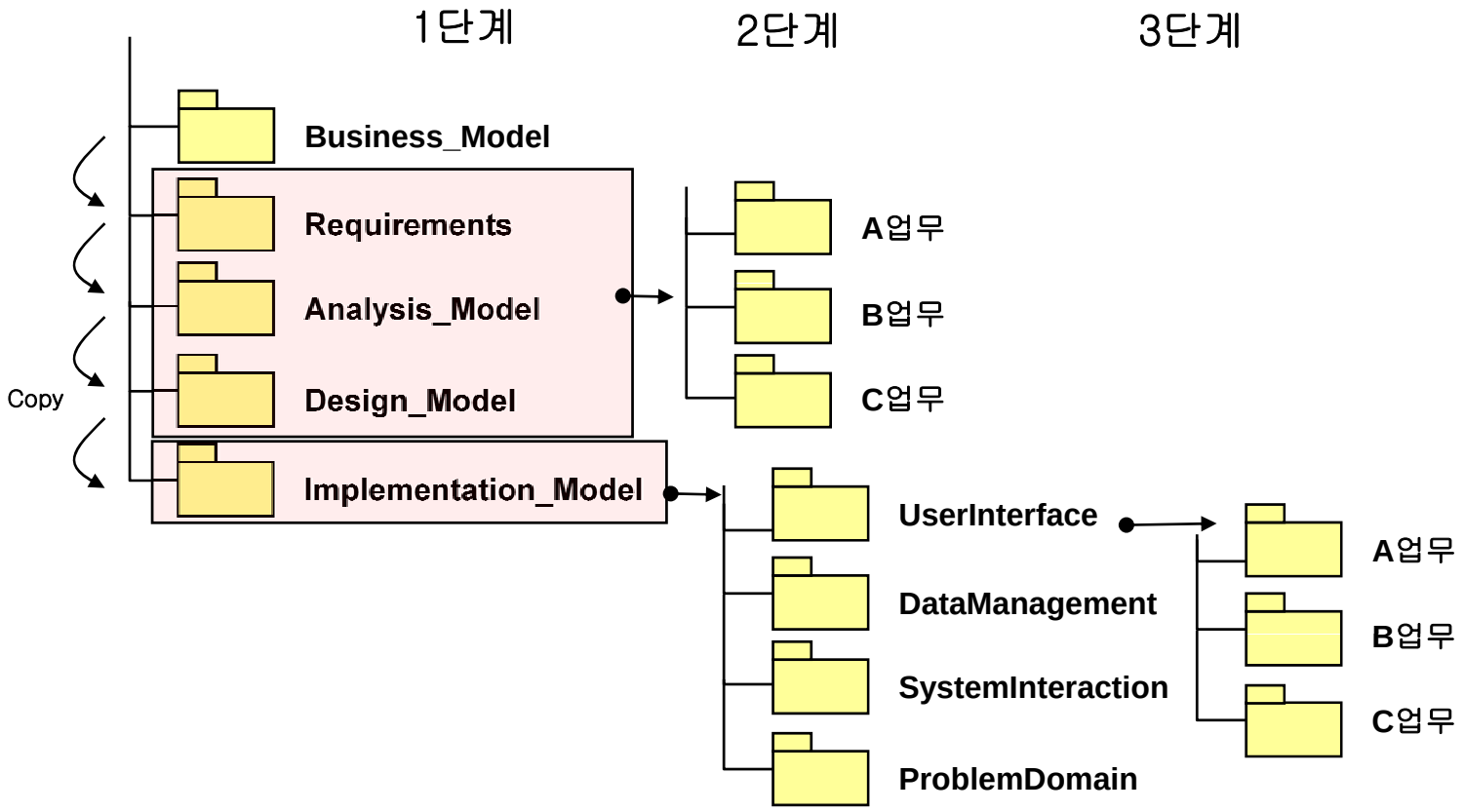


OOC
Open Objected Component

Borland®

Package 구성

Including Workflow(Big Project)



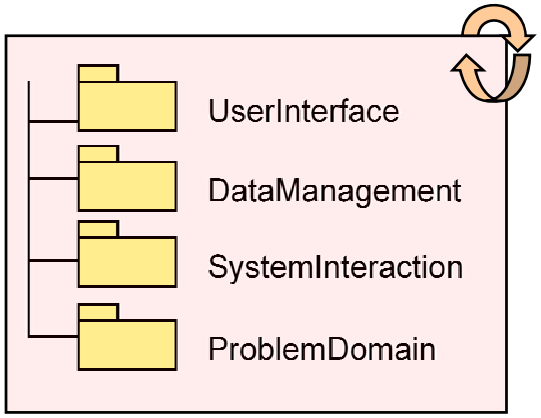
OOC
Open Objected Component

Borland®

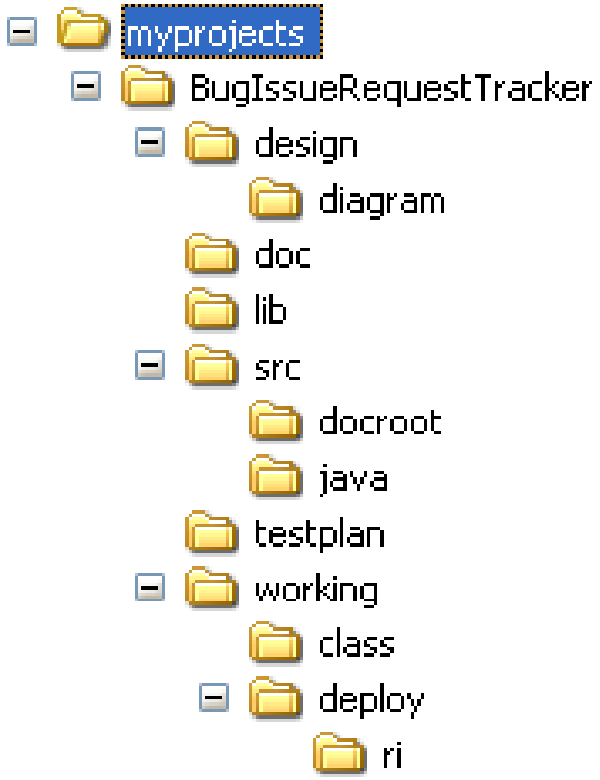
Package 구성

Not Including Workflow(Small Project)

- 모델을 계속 점진적으로 발전시켜나간다



- Sample Project



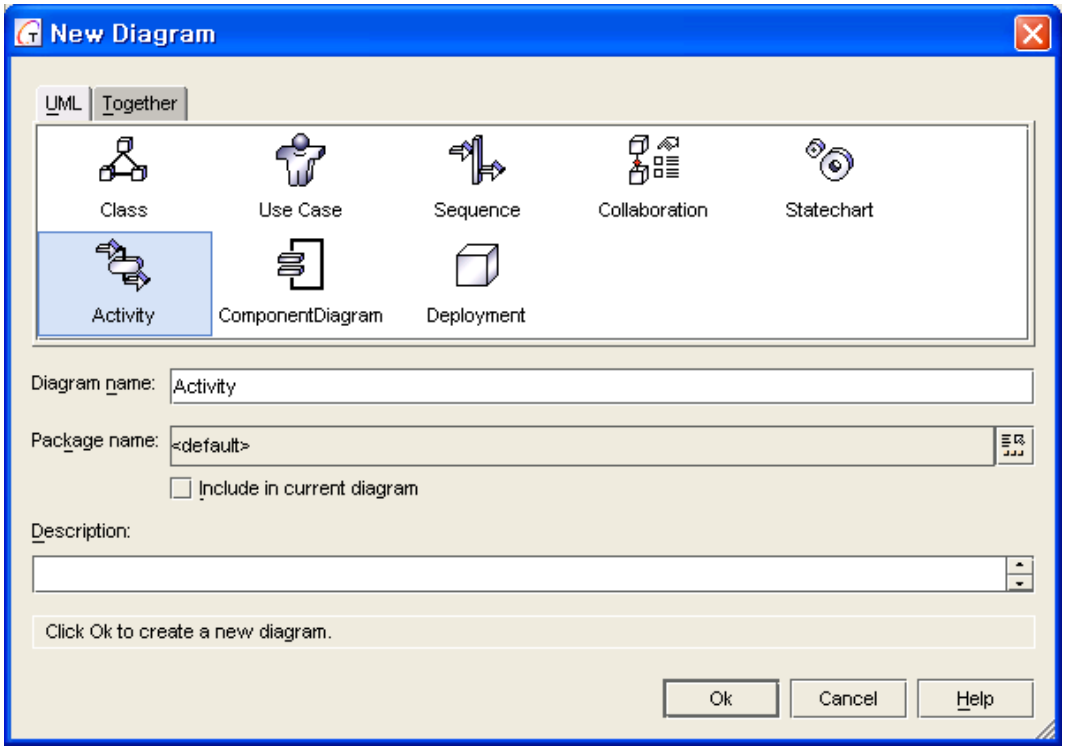
OOC
Open Objected Component

Borland®

Activity Diagram 생성

Business Modeling 산출물

- 비즈니스 분석을 위해 잠재적으로 복잡하거나 은밀한 비즈니스를 상세히 설명하기 위한 업무 흐름도 => Activity Diagram



- 현행 업무(AS-is), 개선된 업무 (To-be)를 분석할 때 사용한다.
- Activity Diagram은 개발자에게는 고객의 업무를 이해하는데 도움을 주고, 다음에 진행될 유즈케이스를 도출할 수 있는 기반이 될 수 있다.
- 고객과의 원활한 커뮤니케이션을 가능하게 한다.

OOC
Open Objected Component

Borland®

Activity Diagram

Activity들의 순차적 표현

- Business Process
- Business rule

사용

- 복잡한 Activity의 분해
- 알고리즘 표현
- Use Case의 시나리오 표현
- High level workflow -> Low level workflow

UML 문법

- State Chart와 유사 Diagram

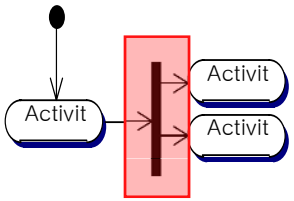
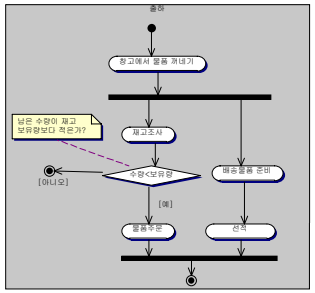
Activity Diagram

Activity Diagram Notation

Together toolbar	Together Default UML Notation	UML Notation 설명	비고
	Start State	프로세스의 시작을 표현한다.	
	End State	프로세스의 끝을 표현한다.	
	Transition	프로세스의 흐름을 표현한다.	
	Activity	하나의 프로세스를 표현한다.	
	Decision	➢ Boolean 표현에 따라 다른 경로를 갖게 되는 경우를 표현한다. ➢ 하나의 입력 전이와 두 개 이상의 출력 전이로 구성된다. ➢ 출력에 해당하는 전이에 Boolean 표현 또는 guard 조건을 입력한다.	

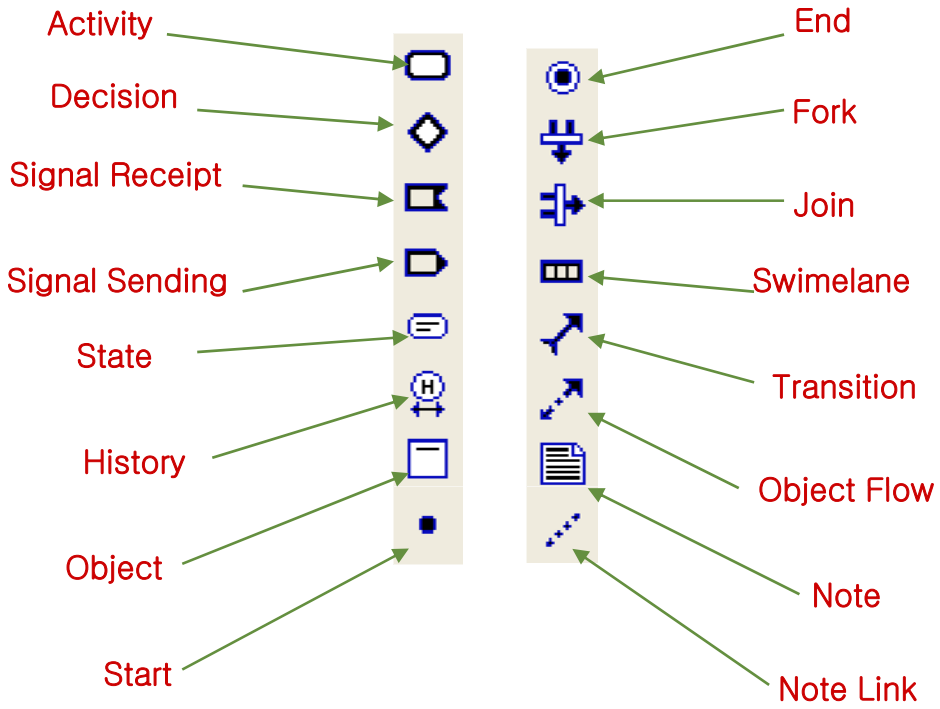
Activity Diagram

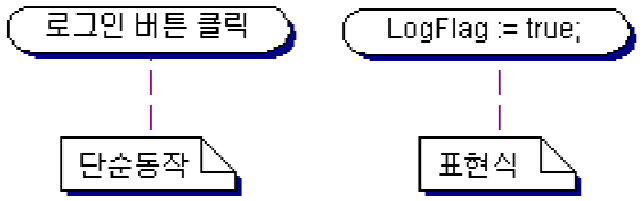
Activity Diagram Notation

Together toolbar	Together Default UML Notation		UML Notation 설명	비고
 	Vertical Fork/Join	 	➢ 하나 이상의 흐름이 둘 이상으로 나누어지거나 둘 이상이 하나로 합쳐짐을 표현 (세로방향) ➢ 하나의 통제를 두 개 이상의 동시 통제로 분리하는 것을 나타낸다. ➢ 하나의 입력 전이와 두 개 이상의 출력 전이를 가지며 각 전이는 독립적인 통제를 갖는다. ➢ Fork 아래에서 이러한 흐름에 참여하는 Activity들은 동시성을 갖는다.	

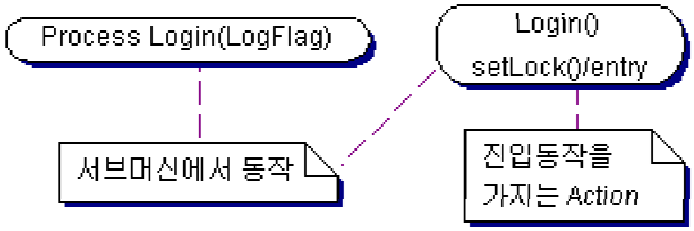
Activity Diagram

Together Architect에서의 구성요소





<동작 상태 표현>



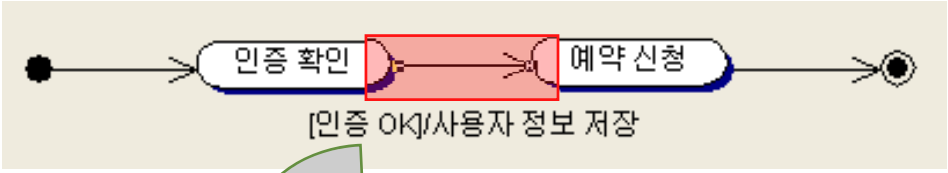
<활동 상태 표현>

OOC
Open Objected Component

Borland®

Activity Diagram

구성요소 및 Property 작성



Properties

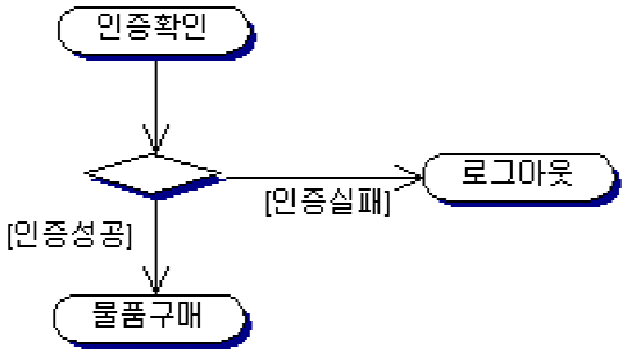
Inspector: (transition link)

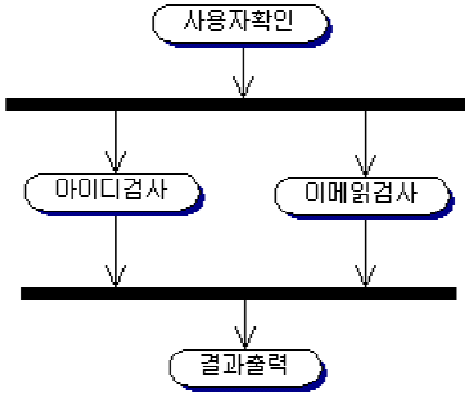
Link | Description | HTMLdoc | Requirements

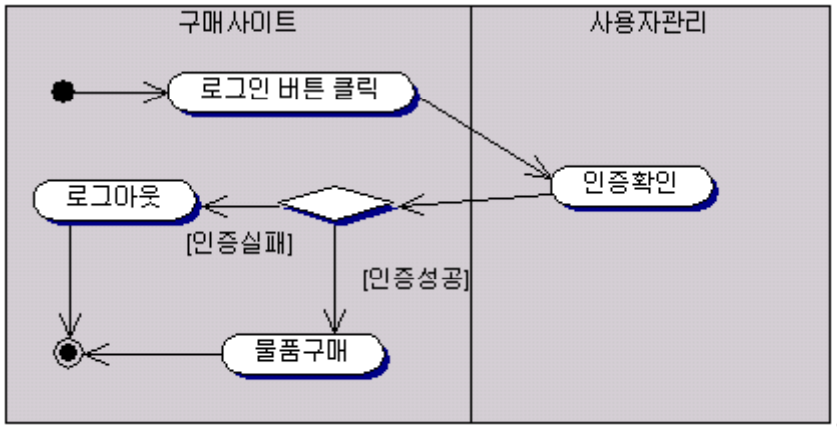
Name	Value
client	인증 확인
supplier	예약 신청
event name	
event arguments	
guard condition	인증 OK
action expression	사용자 정보 저장
send clause	
send time	
receive time	
constraint	

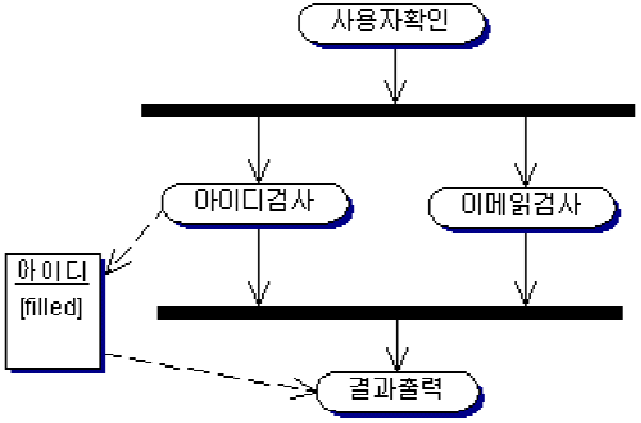
Activity Diagram

구성요소







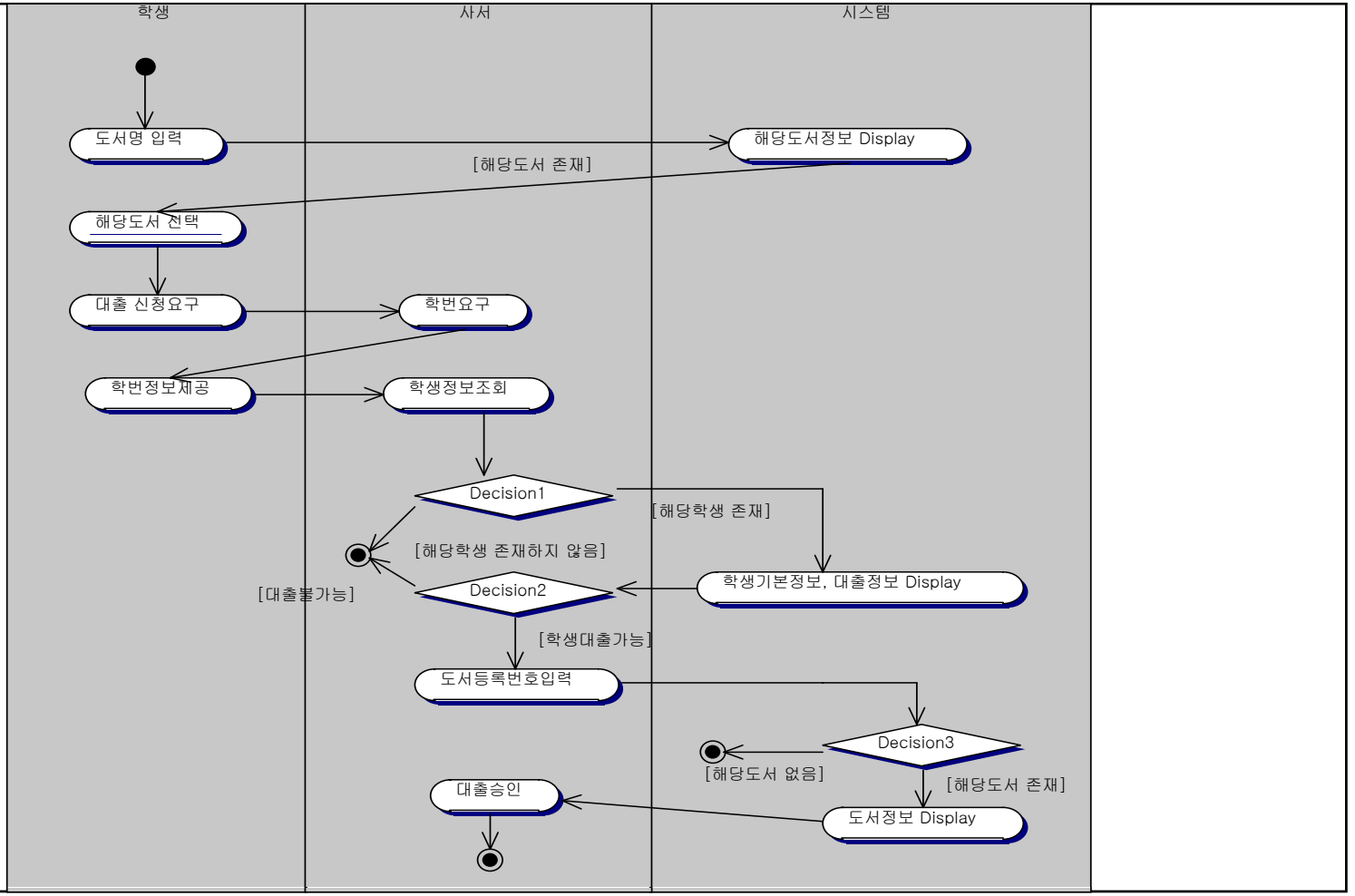


OOC
Open Objected Component

Borland®

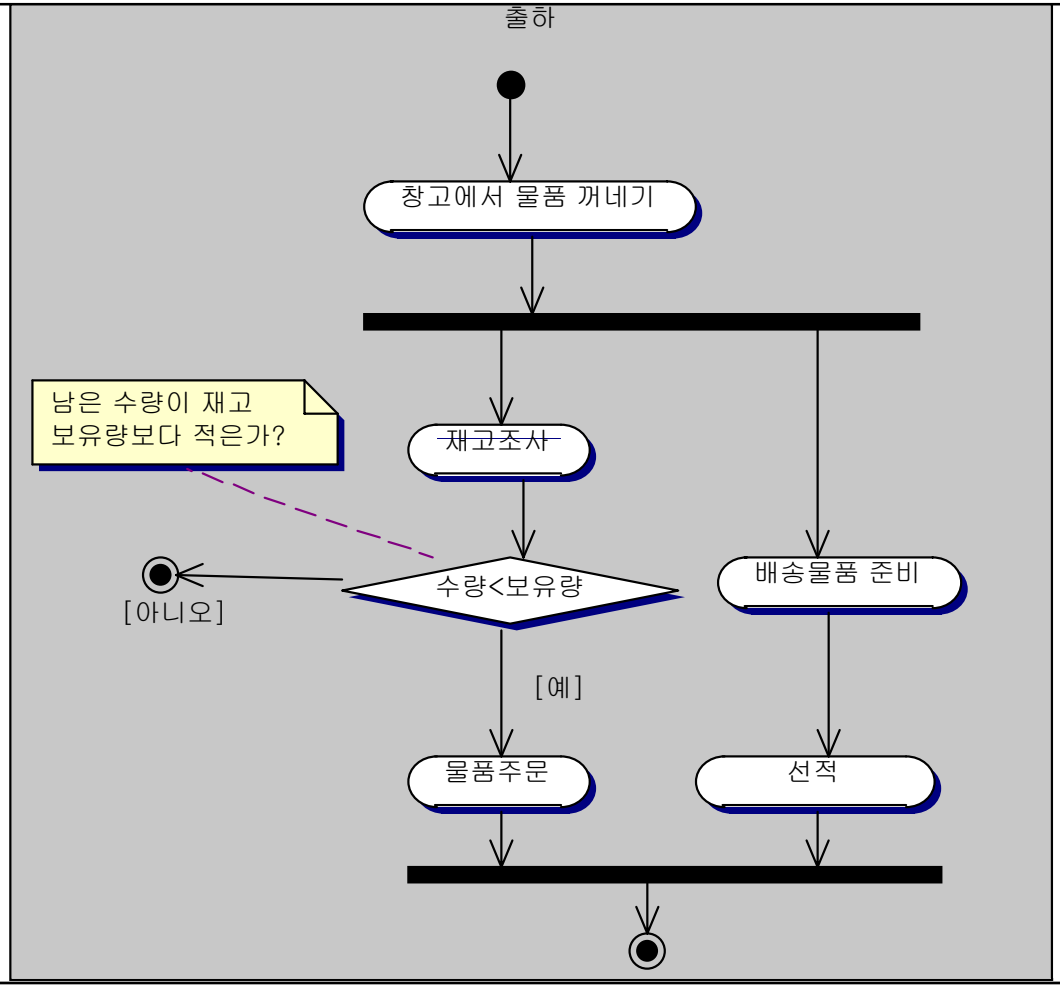
Activity Diagram

실습 예제 1



Activity Diagram

실습 예제 2



OOC
Open Objected Component

Borland®

Use Case Diagram

Use Case Diagram

- 시스템의 필수적인 기능 표현(모든 기능을 표현하지는 않는다)
- 시스템이 사용자를 위해 무엇(What)을 해야 하는지를 표현

- 행동 관점
  - 기능적인 요구사항 수집
  - 점진적인 기능의 상세화
  - 시스템의 이해 (테스트 시나리오의 기준)

- 구성
  - Actor, Use Case, Relationship link

- 목표
  - 요구사항 수집
  - 각 시스템 별로 요구사항 및 Actor 표현

Use Case Diagram

Actor 추출

- 액터는 시스템 외부에 존재? 내부존재 ?
  - 개발될 시스템 외부에서 시스템과 상호작용
- 사람만이 액터가 될 수 있다 ?
  - 사람, 개발 시스템 외의 다른 시스템, 외부 장치 등 시스템을 직접적으로 사용하는 다양한 대상들이 될 수 있다.
- 사용자의 시스템 사용에 있어서의 역할이라 함은 ?
  - 액터를 의미

Use Case 추출

- Actor가 요구하는 시스템의 주요 기능은 무엇인가?
- Actor가 시스템의 어떤 정보를 수정, 조회, 삭제, 저장하는가?
- 시스템이 Actor에게 주는 어떠한 Event가 있는가? Actor가 시스템에 주는 어떠한 Event가 있는가?
- 시스템의 입력과 출력으로 무엇이 필요한가? 그리고 입력과 출력이 어디에서 오고 어디로 가는가?
- 시스템의 구현에서 가장 문제가 되는 점은 무엇인가?

Usecase Diagram

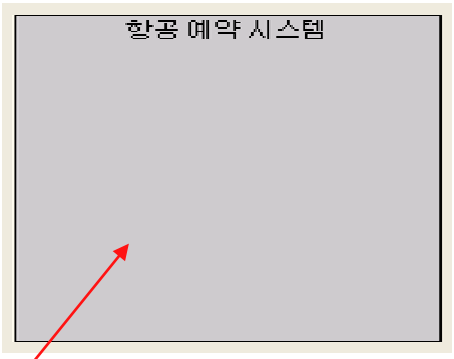
Usecase Diagram Notation

Together toolbar	Together Default UML Notation	UML Notation 설명	비고
	Actor	시스템과 상호작용해야 하는 어떤 사람 또는 어떤 것	
	Usecase	시스템이 액터와 상호 작용 하면서 수행하는 일련의 기능적인 트랜잭션	
	System Boundary	물리적인 시스템과 상호 작용하는 액터 사이의 영역을 표현	
	Associations	엑터, 유즈케이스 사이의 상호 작용에 대한 표현	연관
	Generalizations	추상화 유즈케이스와 상세 유즈케이스 간의 관계 표현	일반화
	Includes	기본 유즈케이스에 포함되는 유즈케이스 사이의 관계 표현	Mandatory
	Extends	기본 유즈케이스에 확장되는 유즈케이스 사이의 관계 표현	Optional

Use Case Diagram

구성요소

toolbar



System Boundary

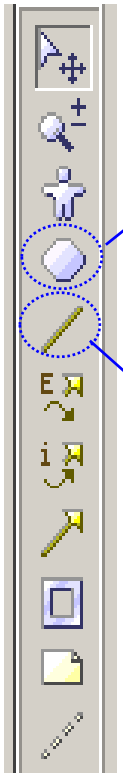
Actor

- 물리적인 시스템과 상호 작용하는 액터 사이의 영역을 표현
- 액터들은 System Boundary 밖에 표현

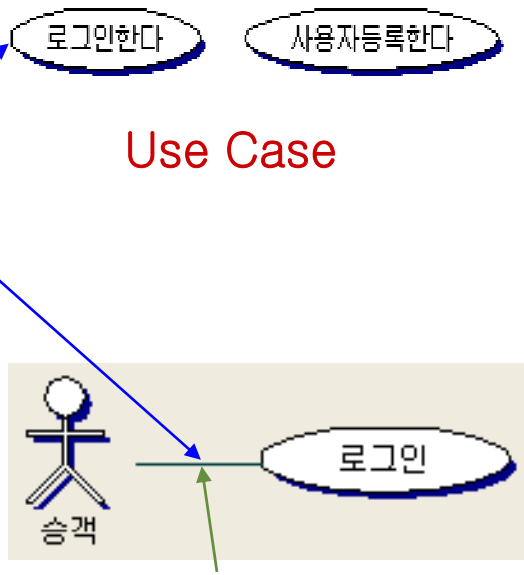
- 시스템과 상호작용해야 하는 어떤 사람 또는 어떤 것
- 시스템 사용자의 역할(Roles)을 의미
- 액터들은 개체들이 아니라 유즈케이스와 상호 작용한 시스템의 밖에 역할들을 표현
- 유즈케이스를 동작시키는 요소
- 사람, 기계, 다른 시스템이 될 수 있음

Use Case Diagram

구성요소



toolbar

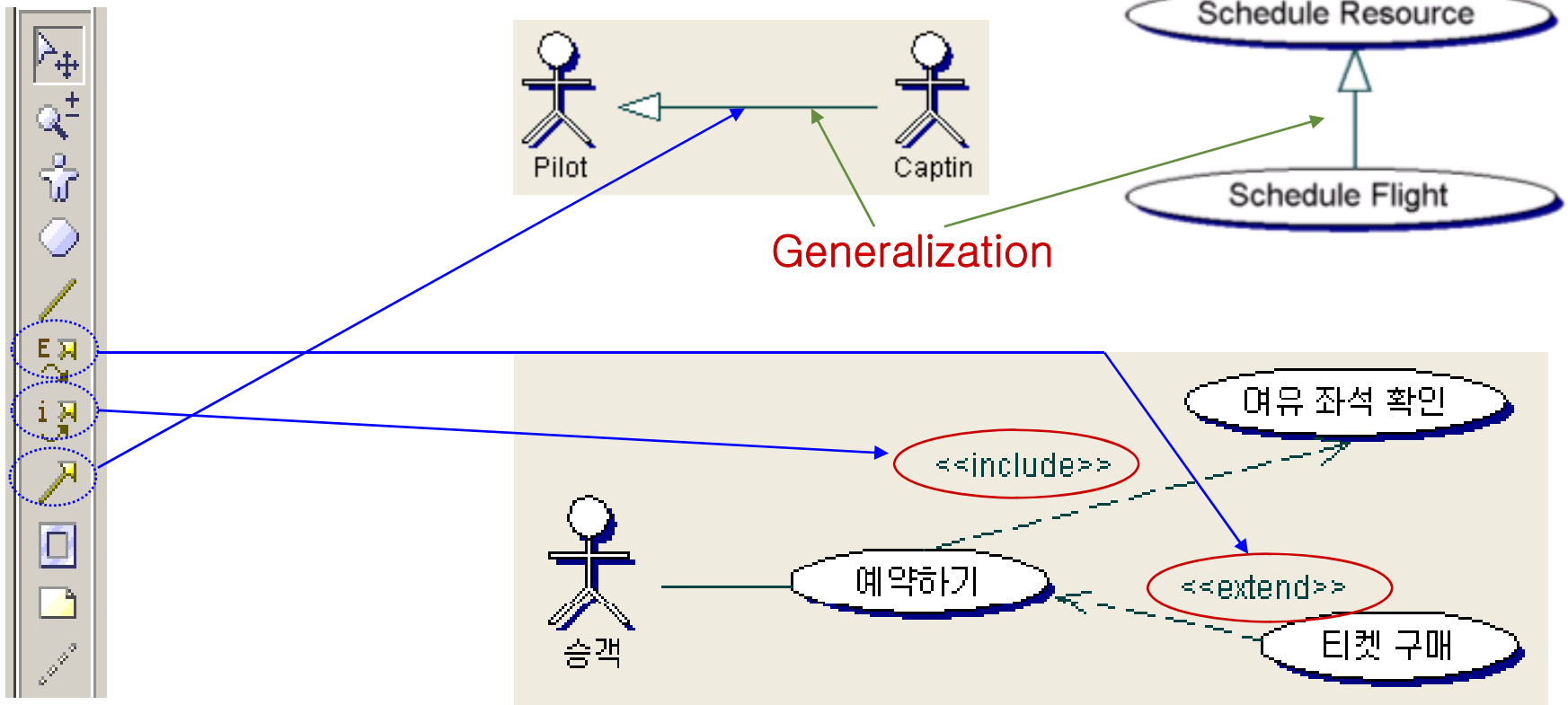


- 개발될 시스템의 개개 액터가 시스템 사용 목적을 달성할 수 있도록 개발될 시스템이 제공해야 하는 서비스 (이러한 서비스는 일련의 트랜잭션으로 수행된다.)
- 시스템의 핵심적인 기능
- Actor가 이루어야 하는 목표
- 시스템 외부 인터페이스로 표시

- Actor와 Use Case간의 정보 교환
- System에서 제공되는 기능과 연결

Use Case Diagram

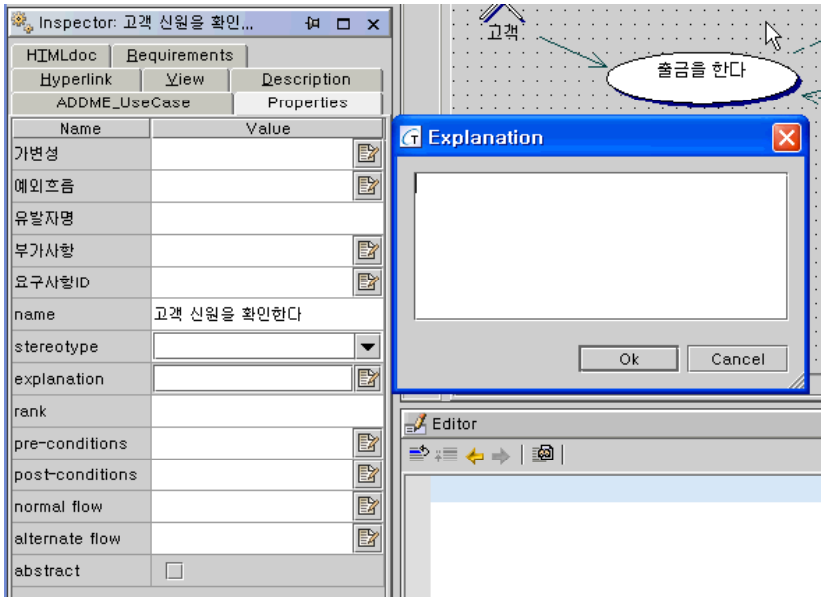
구성요소



Use Case Diagram

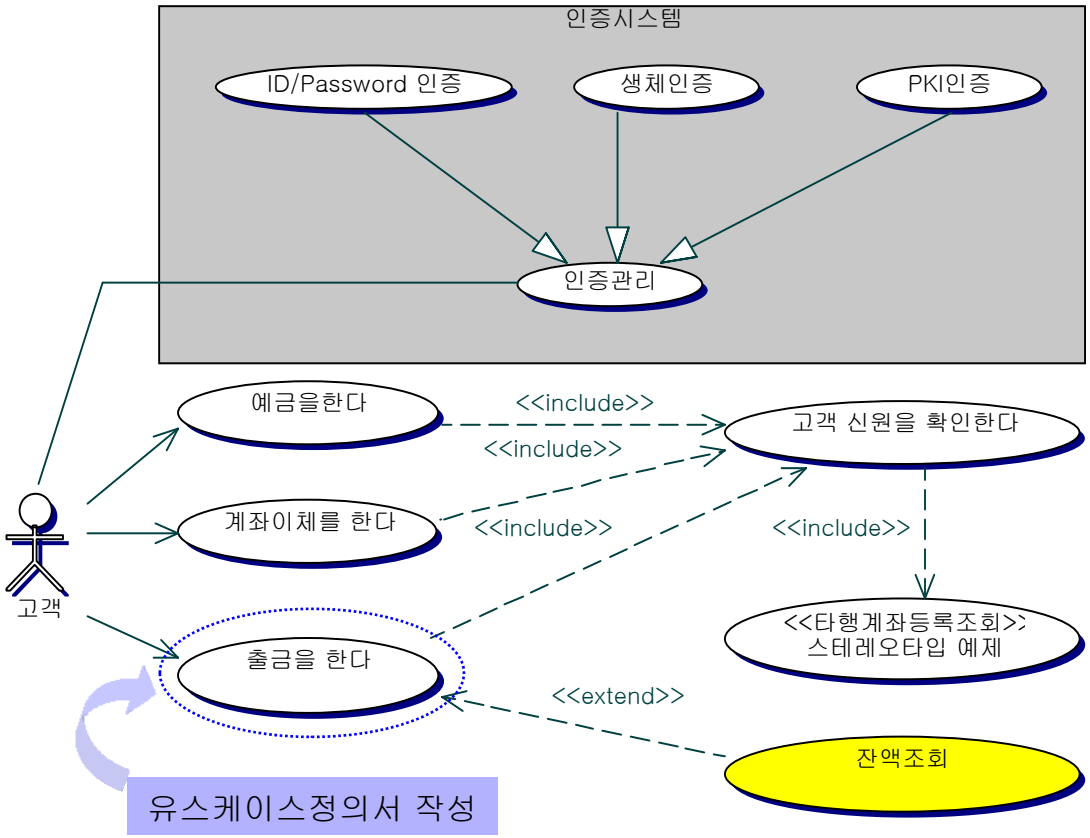
UseCase Specification (유즈케이스 기술서)

- 유즈케이스 한 개당 하나의 유즈케이스 기술서(시나리오)
- 유즈케이스 속성
  - 이름
  - 개요
  - 시작조건
  - 선행조건(precondition), 후행조건(postcondition),
  - 이벤트 플로어, 특수 요구사항(special requirements)
- 작성방법
  - Properties 사용



Use Case Diagram

UseCase Diagram 실습 1



유스케이스 정의서 작성

유스케이스 명 : 출금을 한다.

1. 개요(Description)

이 유즈케이스는 사용자가 어떻게 출금을 하는지를 기술한다. 이미 다른 서비스를 사용하면서 사용자 승인 된 경우에는 사용자 승인은 생략된다.

2. 시작조건(StartCondition)

사용자가 Banking 시스템에 접속하면서 시작한다.

3. 선행 조건(Pre-condition)

시스템은 서비스 운영상태이어야 하고 시스템에 접속하면서 서비스를 시작했다면 회원에게 사용자 인증 폼이 제공되어야 한다.

4. 후행 조건(Post-conditions)

4.1 출금 신청이 접수 되었다.

5. 기본흐름(Normal Flow)

5.1 사용자는 출금 정보를 입력하고 출금 신청을 요청하였다.
5.2 시스템은 사용자 입력 정보를 검증한다.
5.3 시스템은 사용자에게 검증된 출금정보를 보여준다.

6. 대안흐름(Alternative Flow)

해당사항 없음.

7. 예외흐름(Exception Flow)

7.1 사용자 입력 출금 정보 오류 처리
{기본흐름}에서 만약 사용자 입력 출금 정보가 유효하지 않을 경우
7.1.1 시스템은 사용자 입력 출금 정보가 유효하지 않음을 알리고 재 입력을 요청한다.
기본흐름에서 다시 시작한다.

8. 에러 상황(Error Situation)

사용자 출금계좌가 미 등록

OOC
Open Objected Component

Borland®

Class Diagram

Class Diagram

- Class Diagram은 시스템의 구조적인 모습을 그리는 다이어그램으로 주로 설계 단계에서 만들어지는 다이어그램
- Class diagram의 경우 여러 가지 객체들의 타입, 즉 클래스들을 표현하고 그 클래스들의 정적인 관계(associated, dependent, specialized, packaged)를 표현한다.
- 이러한 정적인 요소는 시스템의 life cycle과 수명을 같이하며 하나의 시스템은 여러 개의 class diagram으로 표현이 가능하다.

- 구성
  - Class (Attribute, Operation, Visibility)
  - Interface
  - Relationship (Association, Aggregation, Composition, Implementation, Generalization, Dependency)

- 목표
  - 시스템을 구현할 때 어떤 클래스가 필요한지
  - 클래스 사이의 관계를 나타냄
  - 유스케이스 정적 모델링

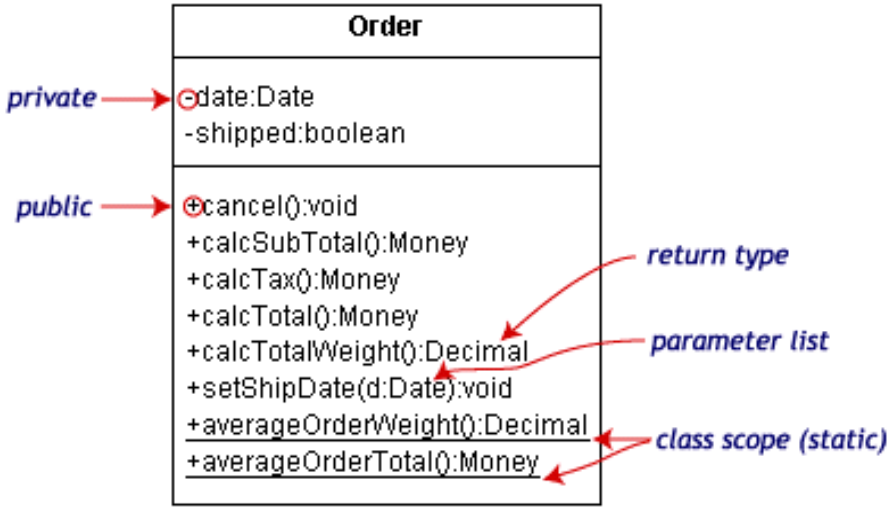
OOC
Open Objected Component

Borland®

Class Diagram

구성 요소

- Class



- Relationship

Association
Aggregation
Composition
Generalization
Dependency
Implementation

Class Diagram

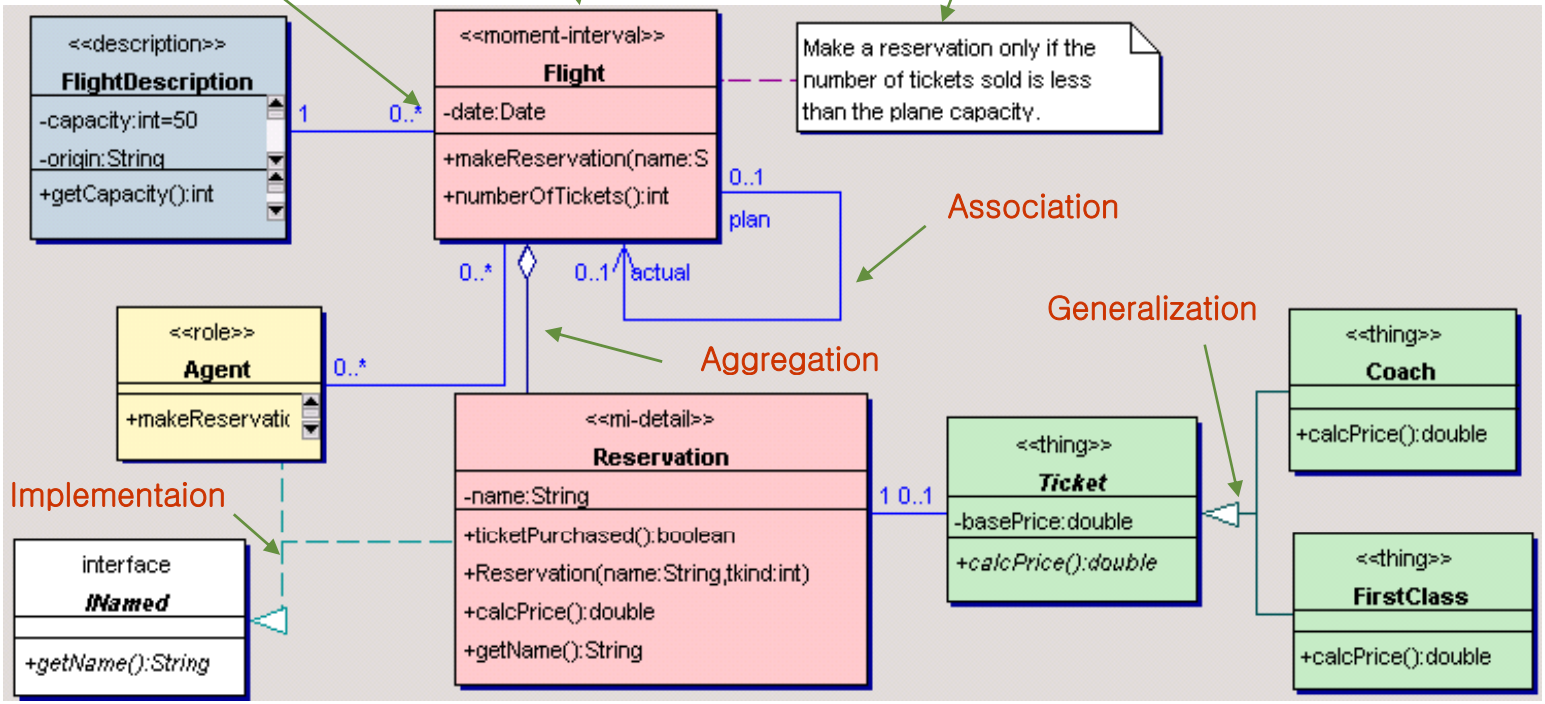
Class Diagram Notation

Together toolbar	Together Default UML Notation	UML Notation 설명	비고
	Package	시스템의 패키지 표현	
	Class	시스템의 클래스 표현	
	Association	클래스간의 연관관계를 표시하는 Link	연관
	Generalization	클래스간의 일반화 관계를 표시하는 Link	상속
	Dependency	클래스간의 참조관계를 표시하는 Link	
	Note	다이어그램 요소에 대한 설명	
	Note Link	노트와 다이어그램 요소간의 연결	

Class Diagram

구성 요소

- Cardinality
- Stereotype
- Note
- Package



OOC
Open Objected Component

Borland®

Class Diagram

Stereotype

- UML의 확장 메커니즘
- 사용자 확장 가능
- Color modeling을 함께 적용
  - Peter Coad
  - Together에서는 기본 적용
- 표기
  - <<, >> or «,»

Class Diagram

Live Source

- 소스와 모델 정보의 일치
  - 소스 수정 -> 모델 반영
  - 모델 수정 -> 소스 수정

Class에 Attribute 추가

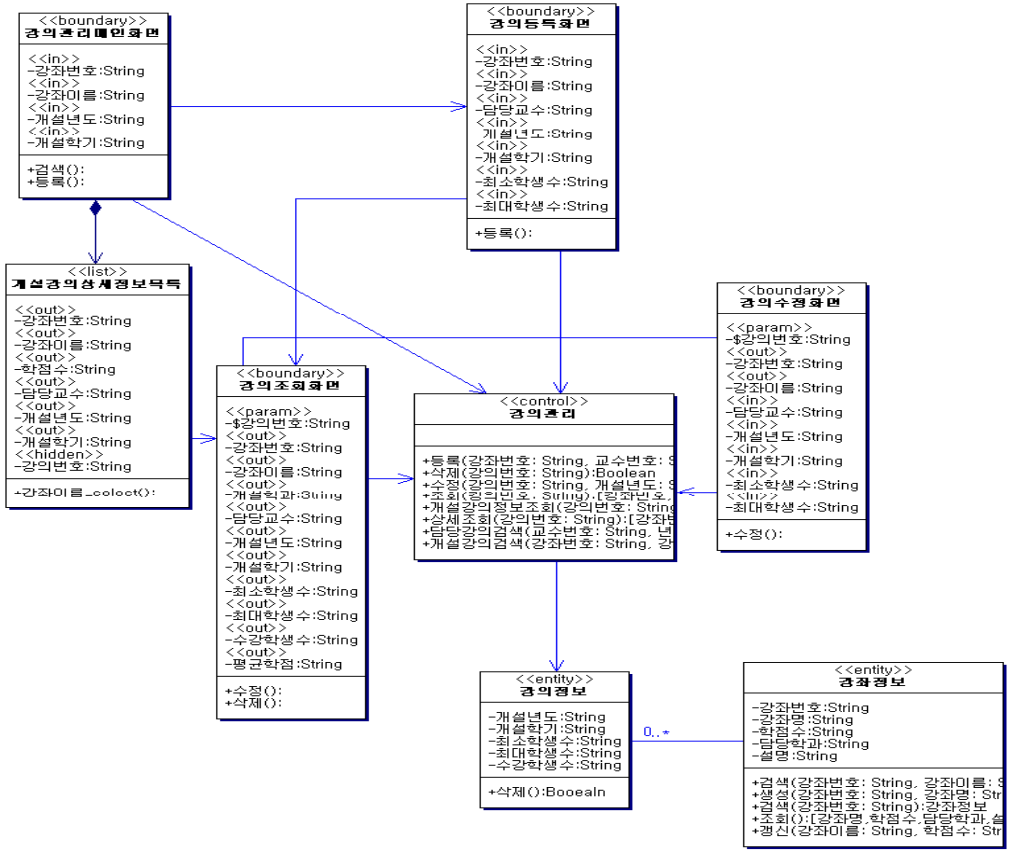
- Ctrl + A

Class에 Operation 추가

- Ctrl + O

Class Diagram – 개념모델링(분석)

Class Diagram은 시스템의 구조적인 모습을 그리는 다이어그램으로 주로 설계 단계에서 만들어지는 다이어그램입니다. 구성요소로는 **Class , Interface,** 그리고 이들간의 **Relationship** 으로 모델링 합니다.

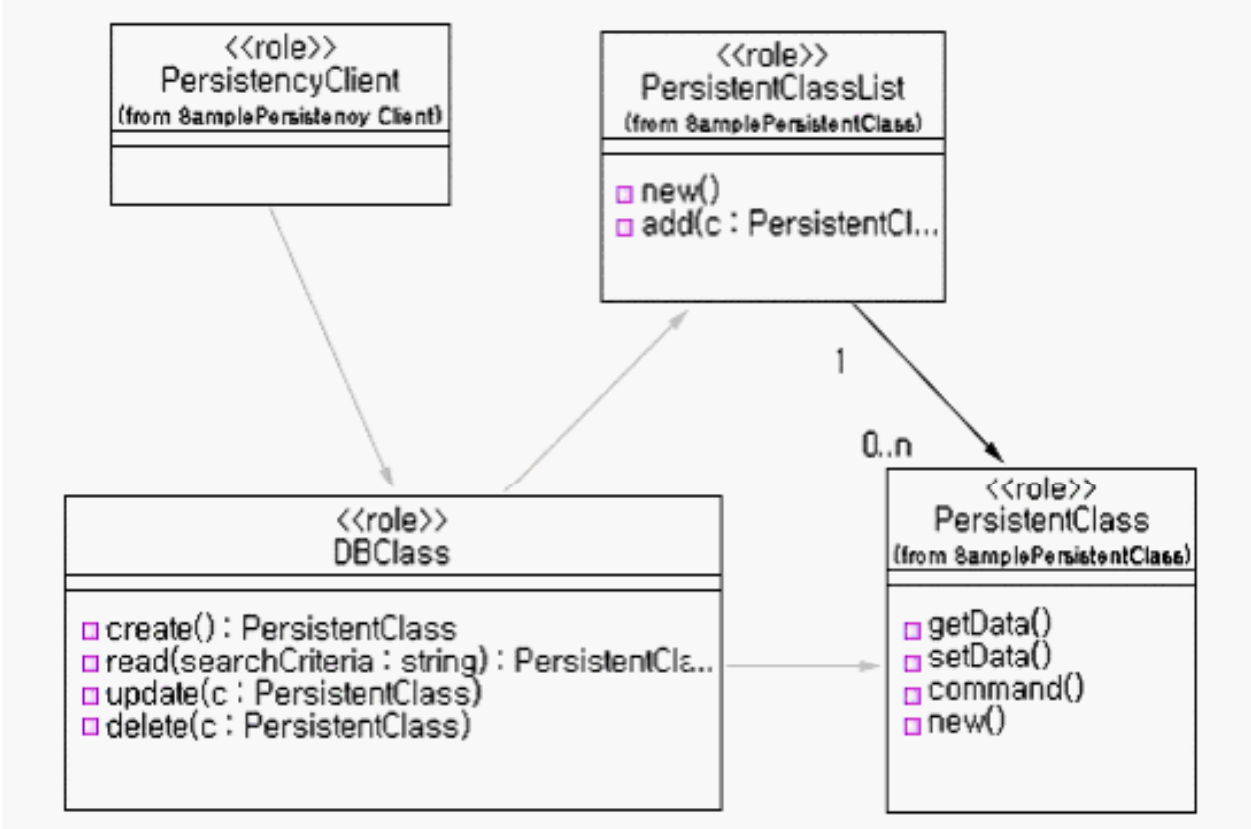


OOC
Open Objected Component

Borland®

Class Diagram – 상세모델링(설계)

설계 단계에서는 보다 구체적인 클래스 들의 내용, 즉 관련 **Attribute**와 **Operation, Relationship**등을 모델링 하게 됩니다.



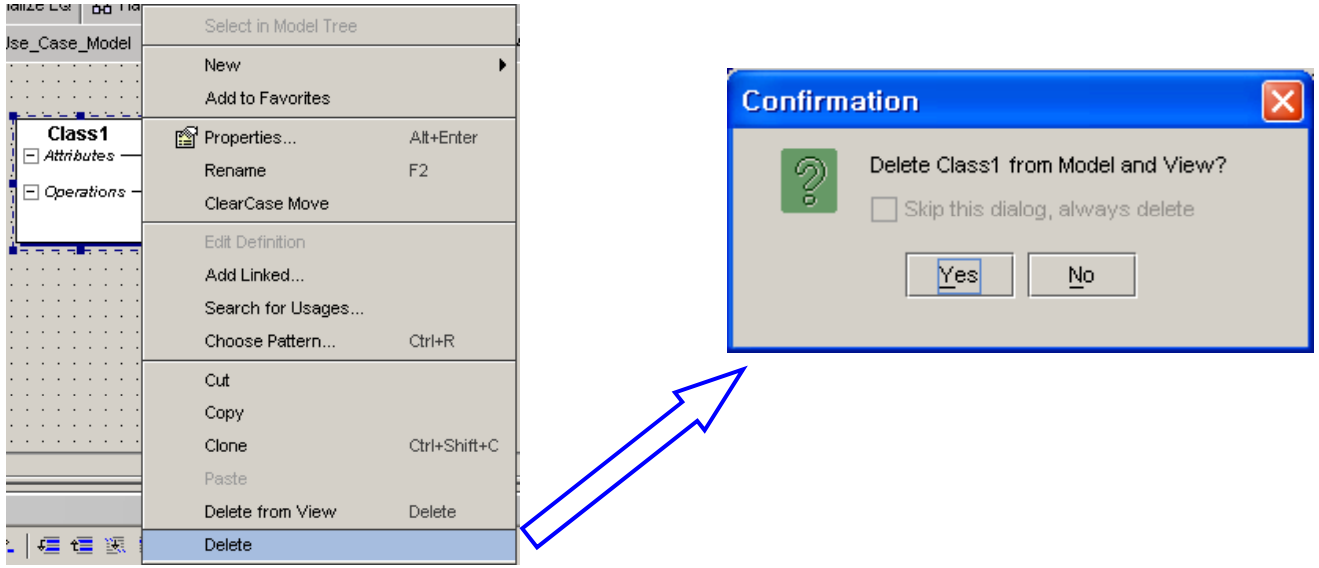
OOC
Open Objected Component

Borland®

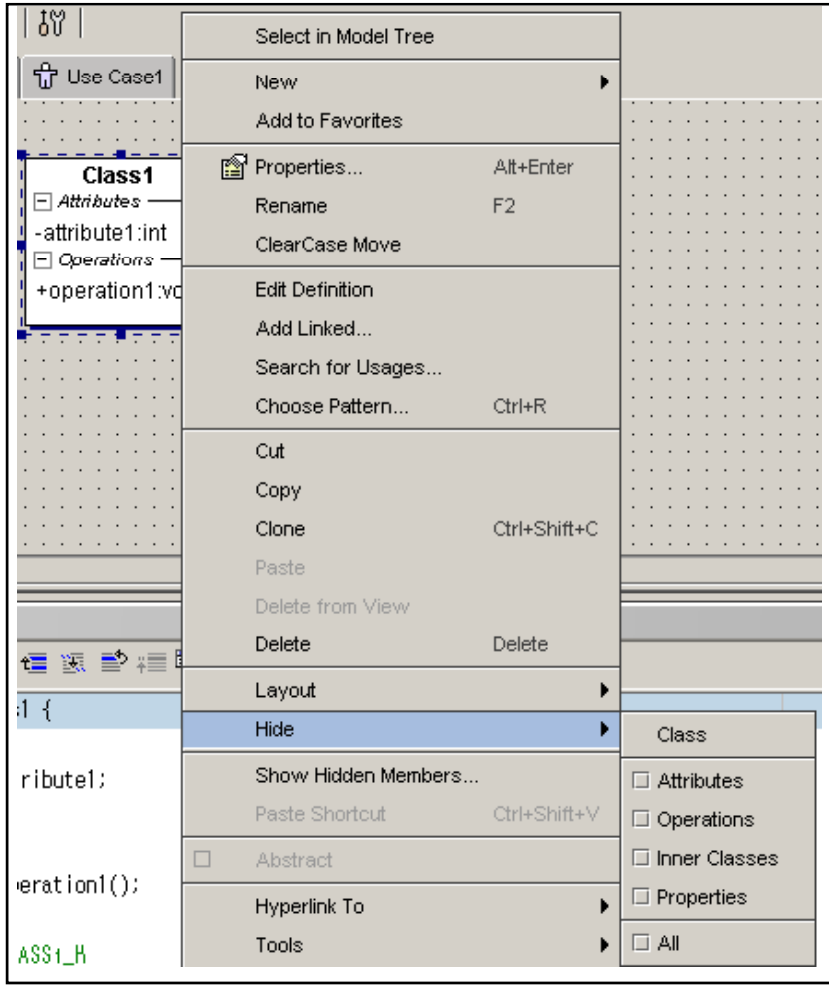
Class Diagram 작성시 주의 사항 및 알아야 할 사항 1

Package Diagram과 Class Diagram의 차이

- package Diagram는 물리적 개념, Class Diagram는 논리적 개념이다.
- delete키를 누르면 package Diagram에서는 실제적으로 해당 클래스가 삭제되나 Class Diagram에서는 논리적으로만 삭제된다. Class Diagram에서 실제로 삭제를 하려면 다음화면 처럼 오른 쪽 마우스를 누른 후 delete부분을 선택한다.



Class Diagram 작성시 주의 사항 및 알아야 할 사항 2



Hide기능

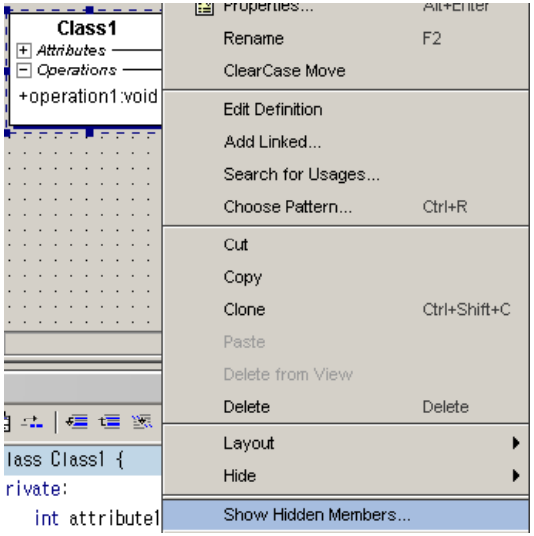
- 클래스 다이어그램에서 생성된 클래스는 클래스 다이어그램에서도 생성이 되고 또한 패키지 다이어그램에서도 생성이 된다. 이럴 경우 모델관리상 패키지 다이어그램에서 hide기능을 이용해서 생성된 클래스를 숨긴다. 이 hide 기능을 모델 관리 시 중요한 기능이다.
- Hide기능

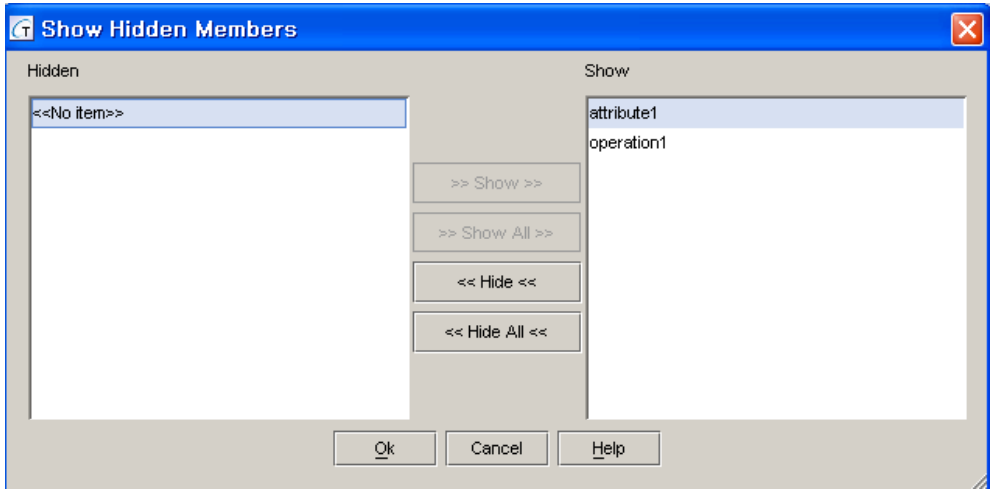
  Class 전체 숨김, 속성만 숨김, 오퍼레이션만 숨김 등.

OOC
Open Objected Component

Borland®

Class Diagram 작성시 주의 사항 및 알아야 할 사항 3

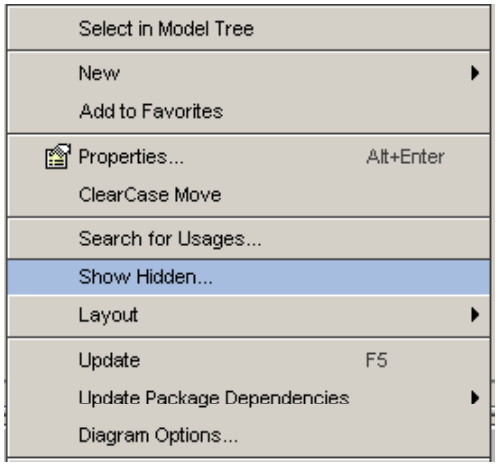


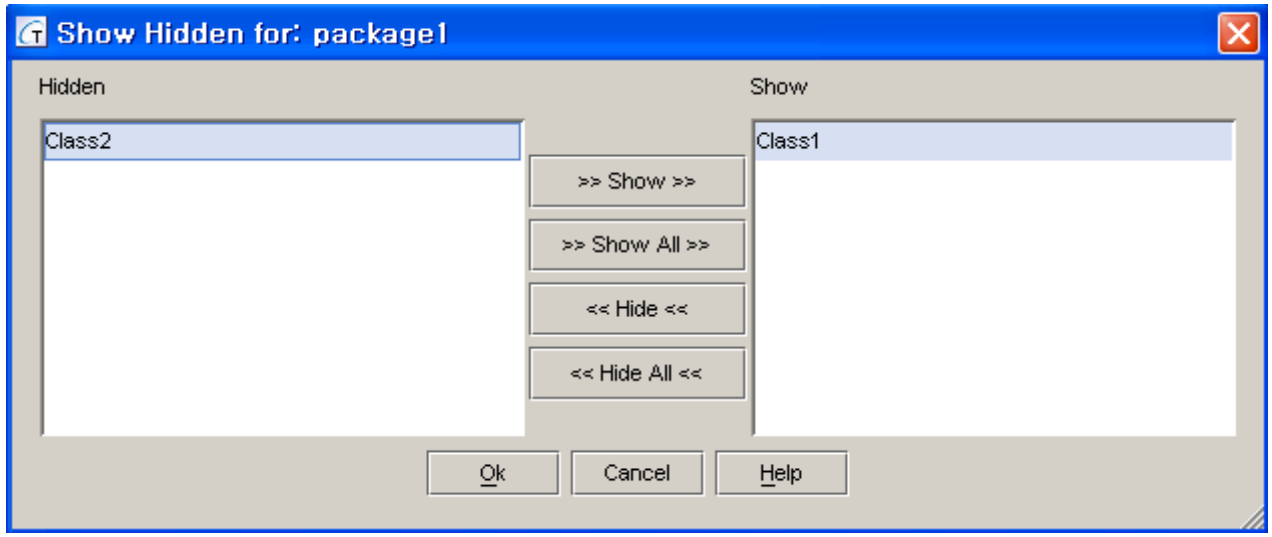


Hide기능

- Show Hidden Member 기능을 통해서 hide기능을 수행할 수있다.

Class Diagram 작성시 주의 사항 및 알아야 할 사항 4



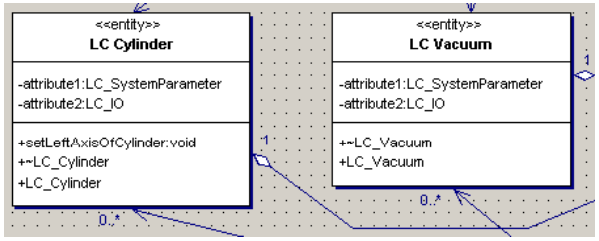


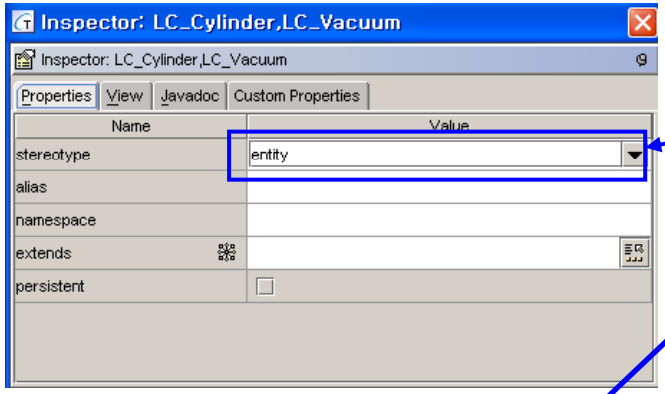
Hide기능

- 다이어그램내에서 숨겨진 부분을 찾기 위해서는 옆 화면처럼 show hidden를 누르면 됩니다

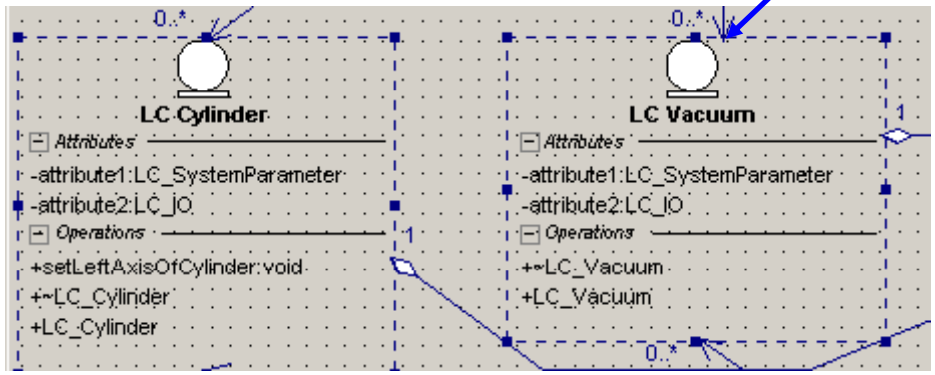
Class Diagram 작성시 주의 사항 및 알아야 할 사항 5

Entity, Boundary, Control 노테이션 변경





entity를 Entity로 바꾸면 된다.

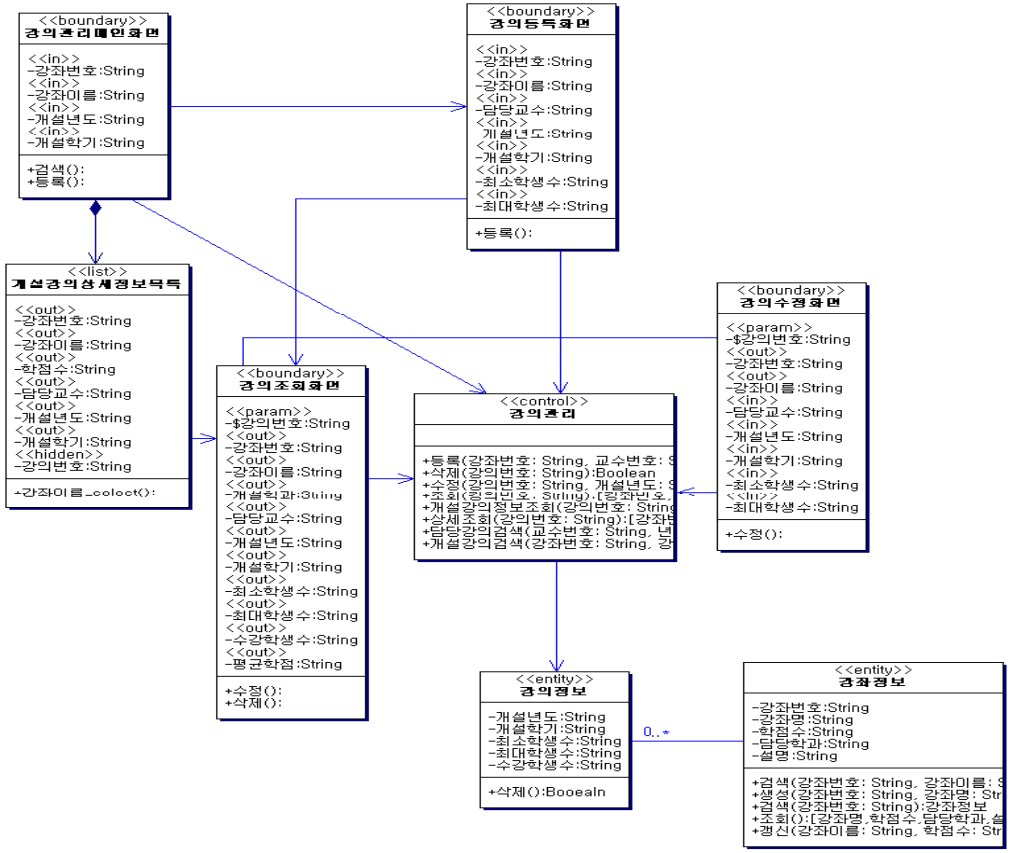


OOC
Open Objected Component

Borland®

Class Diagram – 개념모델링(분석)

Class Diagram은 시스템의 구조적인 모습을 그리는 다이어그램으로 주로 설계 단계에서 만들어지는 다이어그램입니다. 구성요소로는 Class , Interface, 그리고 이들간의 Relationship 으로 모델링 합니다.

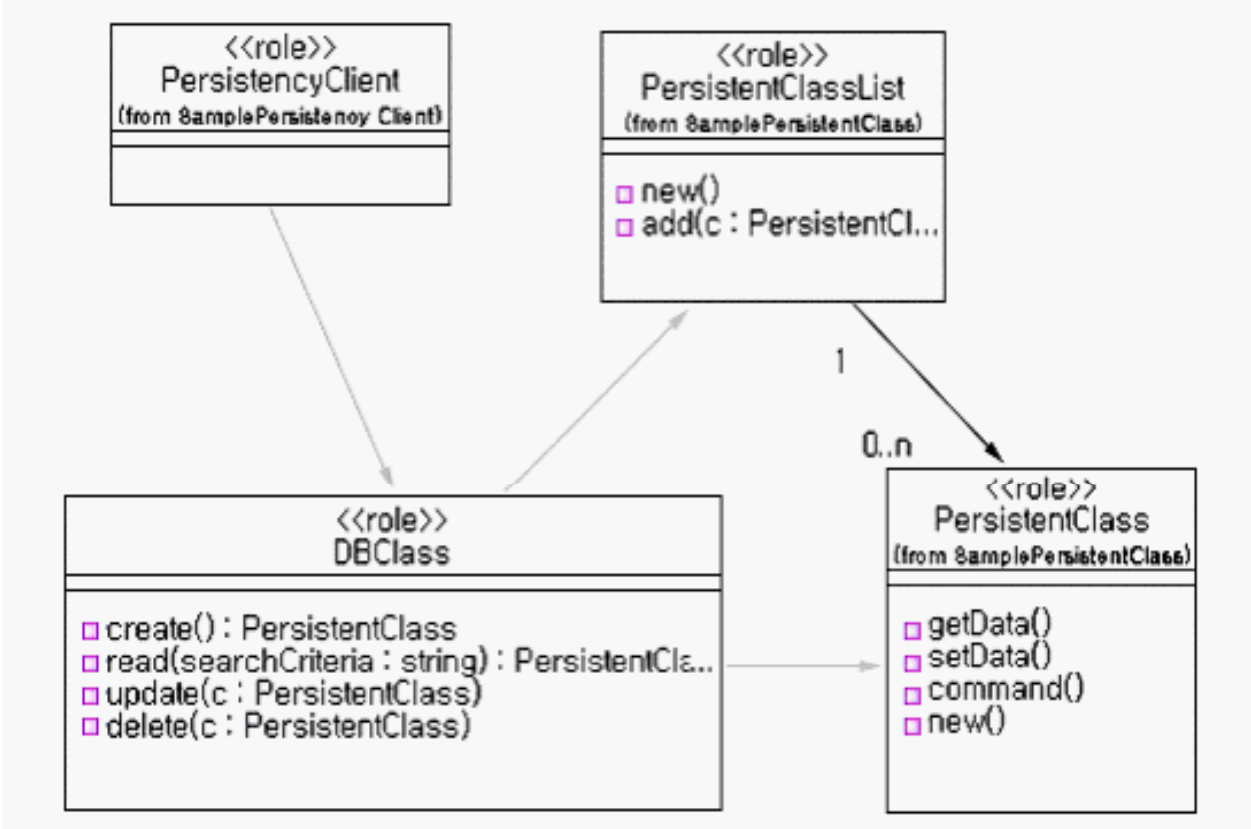


OOC
Open Objected Component

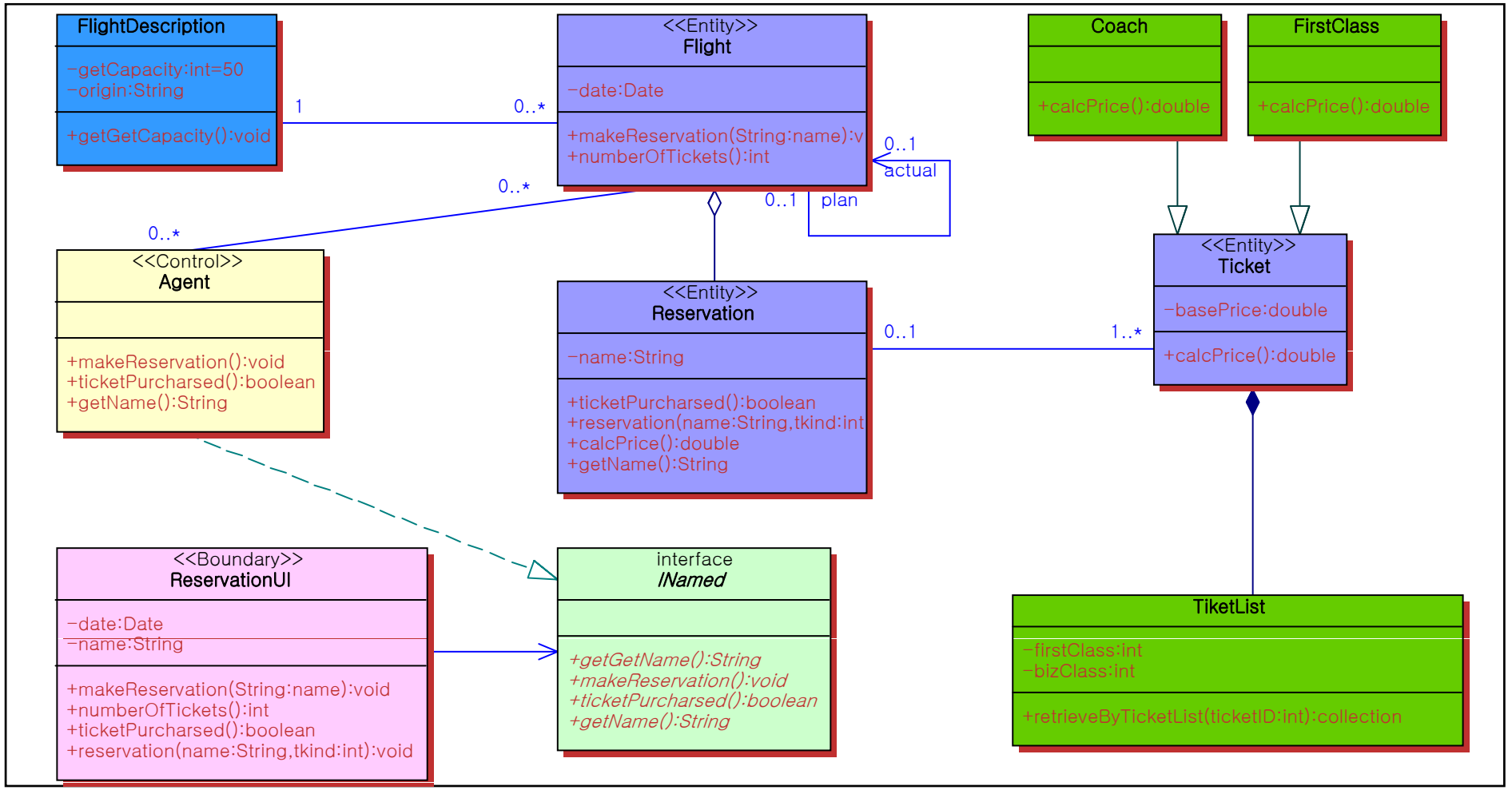
Borland®

Class Diagram – 상세모델링(설계)

설계 단계에서는 보다 구체적인 클래스 들의 내용, 즉 관련 Attribute와 Operation, Relationship등을 모델링 하게 됩니다.



Class Diagram 실습 예제



OOC
Open Objected Component

Borland®

Statechart Diagram

Statechart Diagram

- 상태 차트 다이어그램은 객체의 상태와 상태 변경을 일으키는 이벤트를 설명
- 액티비티가 자연스럽게 결론에 도달하는 프로세스인 반면에 상태는 이벤트로 변경될 때까지 같은 상태를 유지
- 이벤트로 발생하는 단계 상의 변경이 상태차트에서 모델링 하려는 대상
- 상태차트의 매우 발전된 형태가 UML에서 만든 액티비티 다이어그램

- 용도
  - 상태차트는 정확히 언제 그리고 어떻게 거래(Transaction)나 계약(agreement)이 상태를 바꾸는 지와 같은 비즈니스 규칙을 설명하는데 효과적
  - 하드웨어 메커니즘을 컨트롤하는 임베디드 시스템(embed system)의 동작을 기술하는 경우

- 구성
  - State, Start, End, Horizontal Fork/Join, Vertical Fork/Join, Transition, Note, Note Link

- 목표
  - 오브젝트가 가질 수 있는 모든 상태와 어떠한 event를 받았을 때 결과로 어떠한 상태로 변화하는지를 나타낸다.

Statechart Diagram

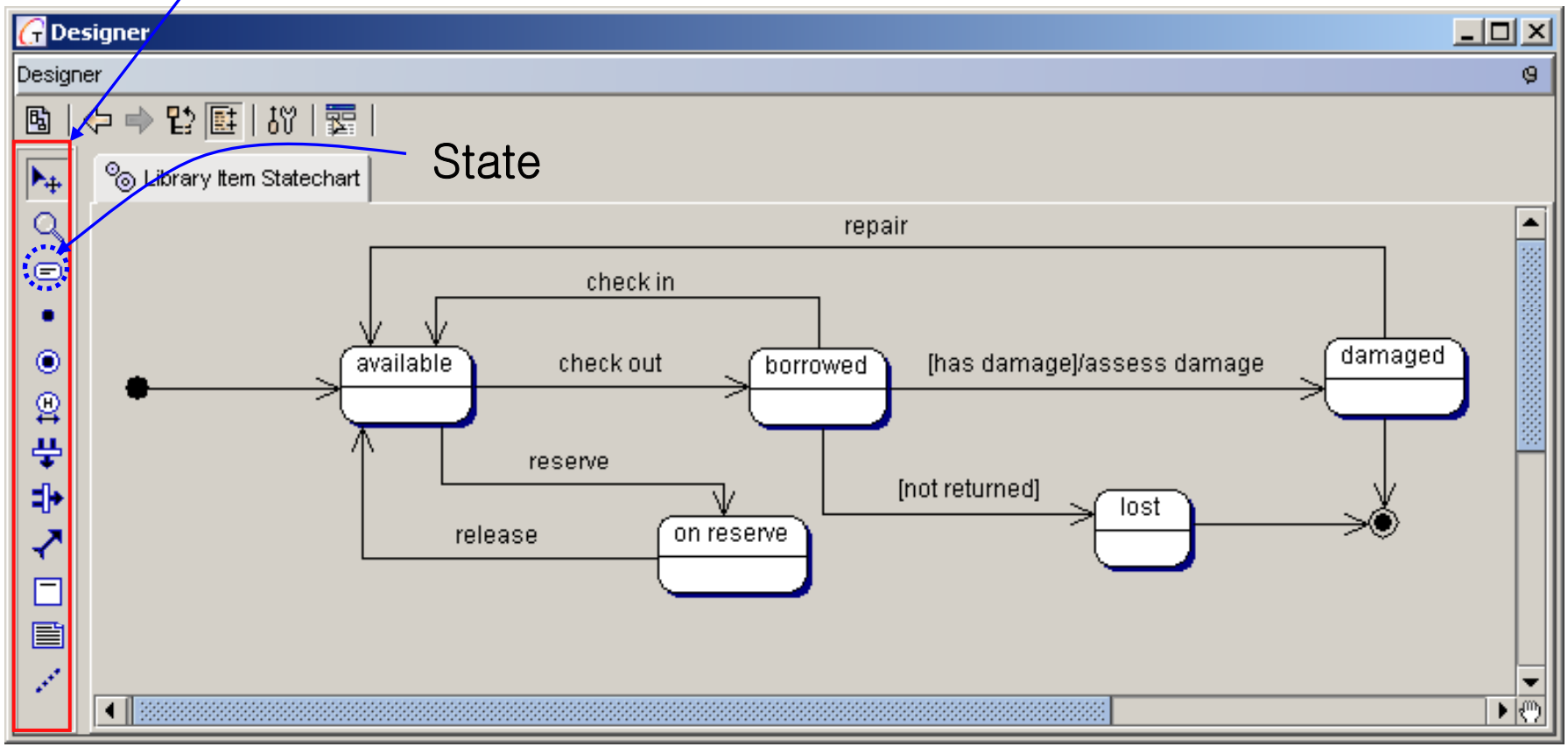
Statechart Diagram Notation

Together toolbar	Together Default UML Notation	UML Notation 설명	비고
	Start State	프로세스의 시작을 표현한다.	
	End State	프로세스의 끝을 표현한다.	
	State	하나의 상태를 표현한다.	
	Vertical Fork/Join	하나 이상의 흐름이 둘 이상으로 나누어지거나 둘 이상이 하나로 합쳐짐을 표현 (가로 방향)	
	Horizontal Fork/Join	하나 이상의 흐름이 둘 이상으로 나누어지거나 둘 이상이 하나로 합쳐짐을 표현 (세로 방향)	
	Transition	프로세스의 흐름을 표현한다.	
	Note	다이어그램 요소에 대한 설명	
	Note Link	노트와 다이어그램 요소간의 연결	

OOC
Open Objected Component

Borland®

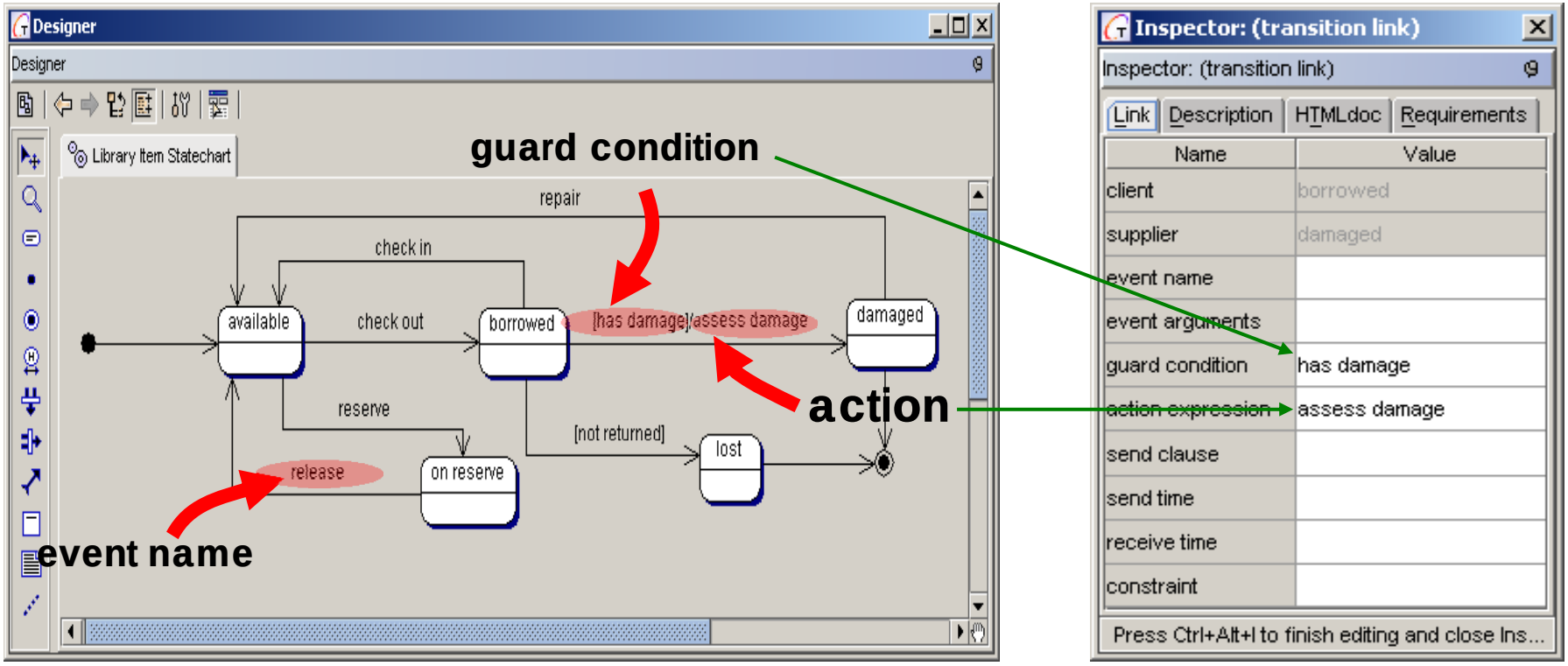
Statechart Diagram Notation



OOC
Open Objected Component

Borland®

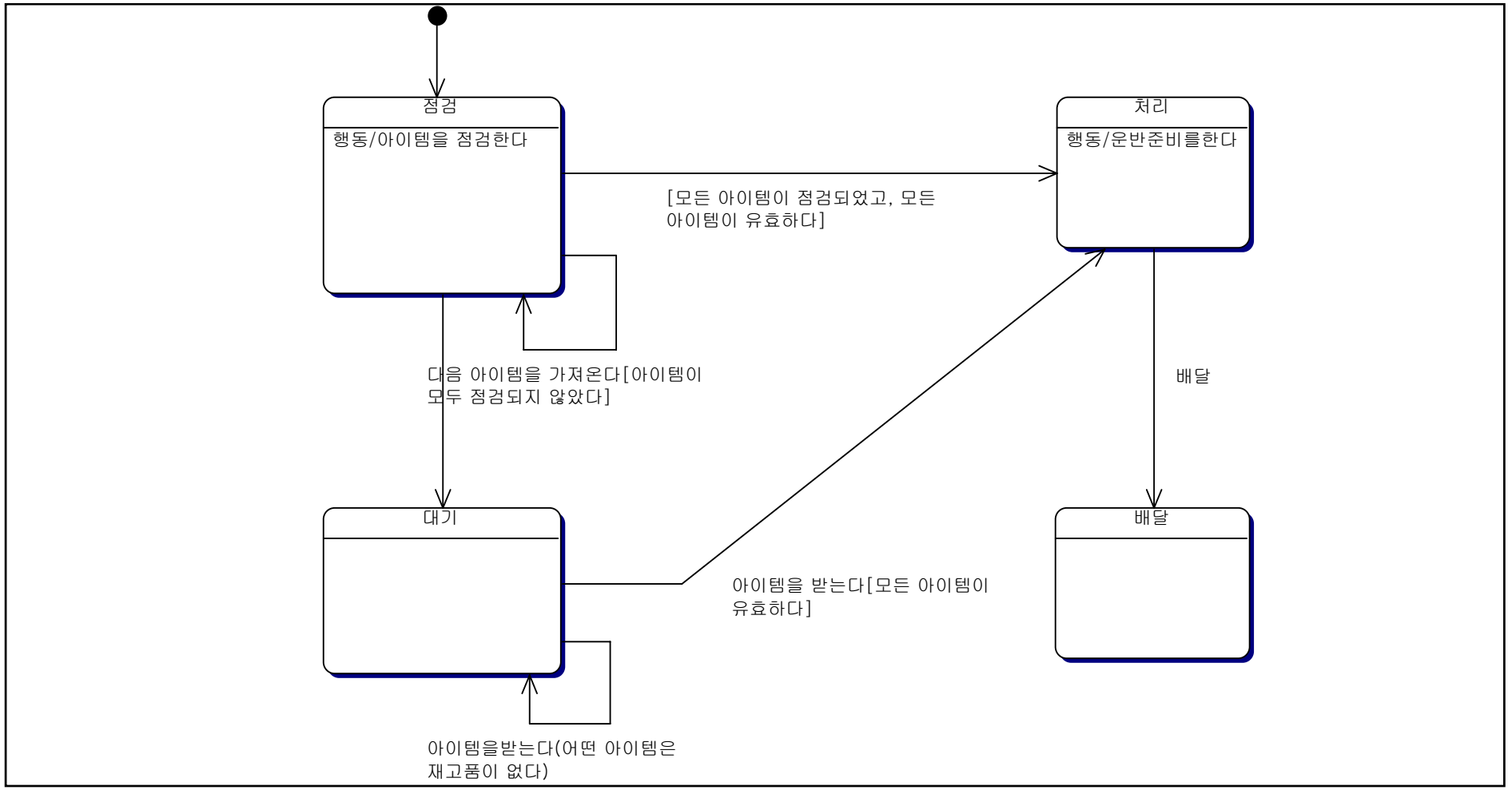
Statechart Diagram Notation 계속



OOC
Open Objected Component

Borland®

Statechart Diagram 실습 예제



OOC
Open Objected Component

Borland®

Interaction Diagram

System의 동적 모델링

- 객체와 그들간의 관계 표현
- 객체간의 메시지 전달 표현

종류

- Sequence Diagram
- Collaboration Diagram

Interaction Diagram

Sequence Diagram 특징

- 메시지의 순서에 초점
- Collaboration과 다른 점
  - 객체 생명선(lifeline)
  - 제어 초점(focus of control)

Collaboration Diagram 특징

- 객체의 구조에 초점
- Collaboration과 다른 점
  - 객체 생명선(lifeline)
  - 제어 초점(focus of control)

Sequence Diagram

Sequence Diagram Notation

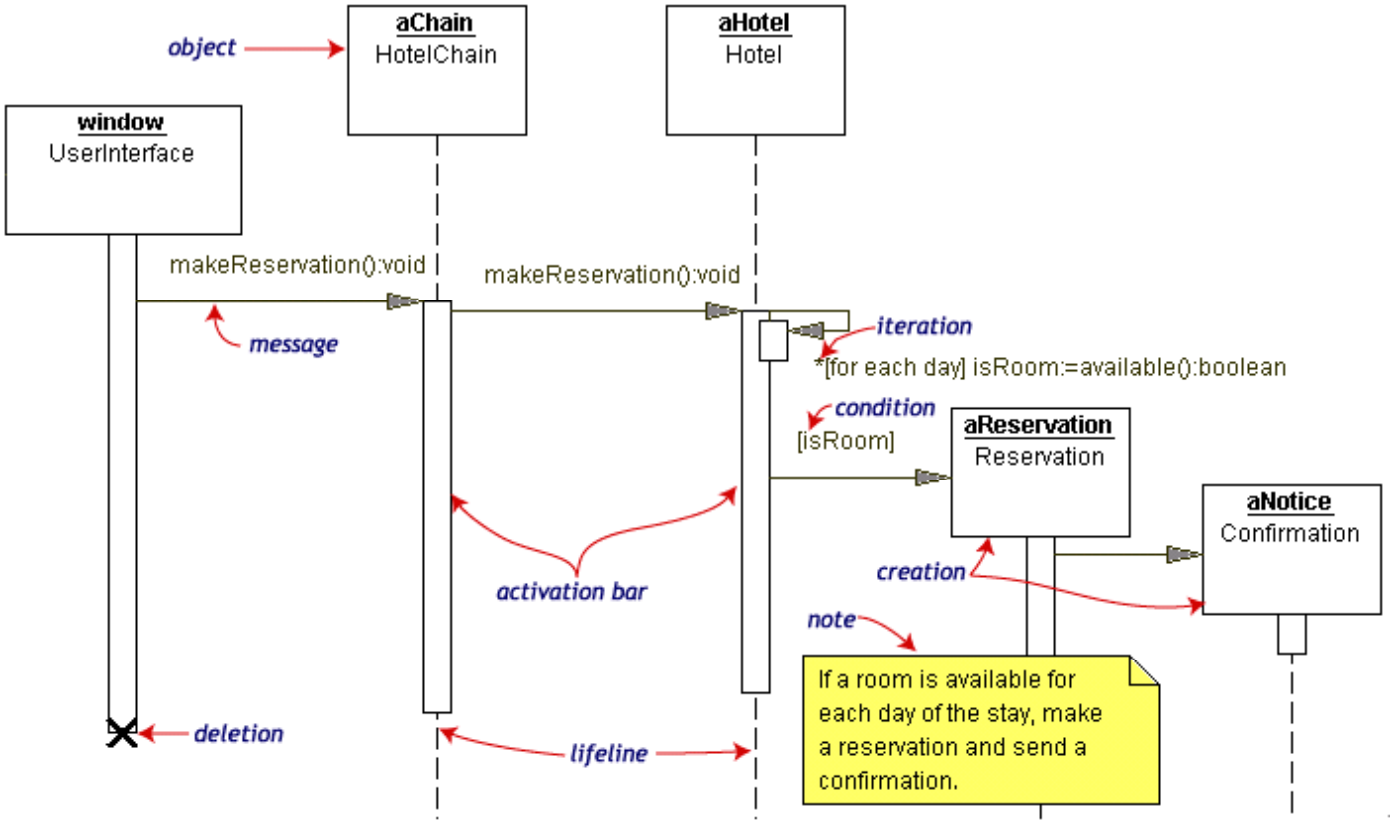
Together toolbar	Together Default UML Notation	UML Notation 설명	비고
	Actor	참여되는 행위자를 표시한다.	
	Object	참여되는 객체에 대한 표현으로 액터와 도출된 객체가 대상이 된다.	
	Message	객체간에 교환되는 정보를 표현한다. (Operation, Parameter, return Value..)	
	Self Message	객체가 객체 자신에게 보내는 정보를 표현한다. (Validation Check)	
	Life line	각 객체의 생명의 주기를 표현한다.	
	Focus of Control	메시지들의 트랜잭션 처리단위를 표현한다.	
	Note	다이어그램 요소에 대한 설명	
	Note Link	노트와 다이어그램 요소간의 연결	

OOC
Open Objected Component

Borland®

Sequence Diagram

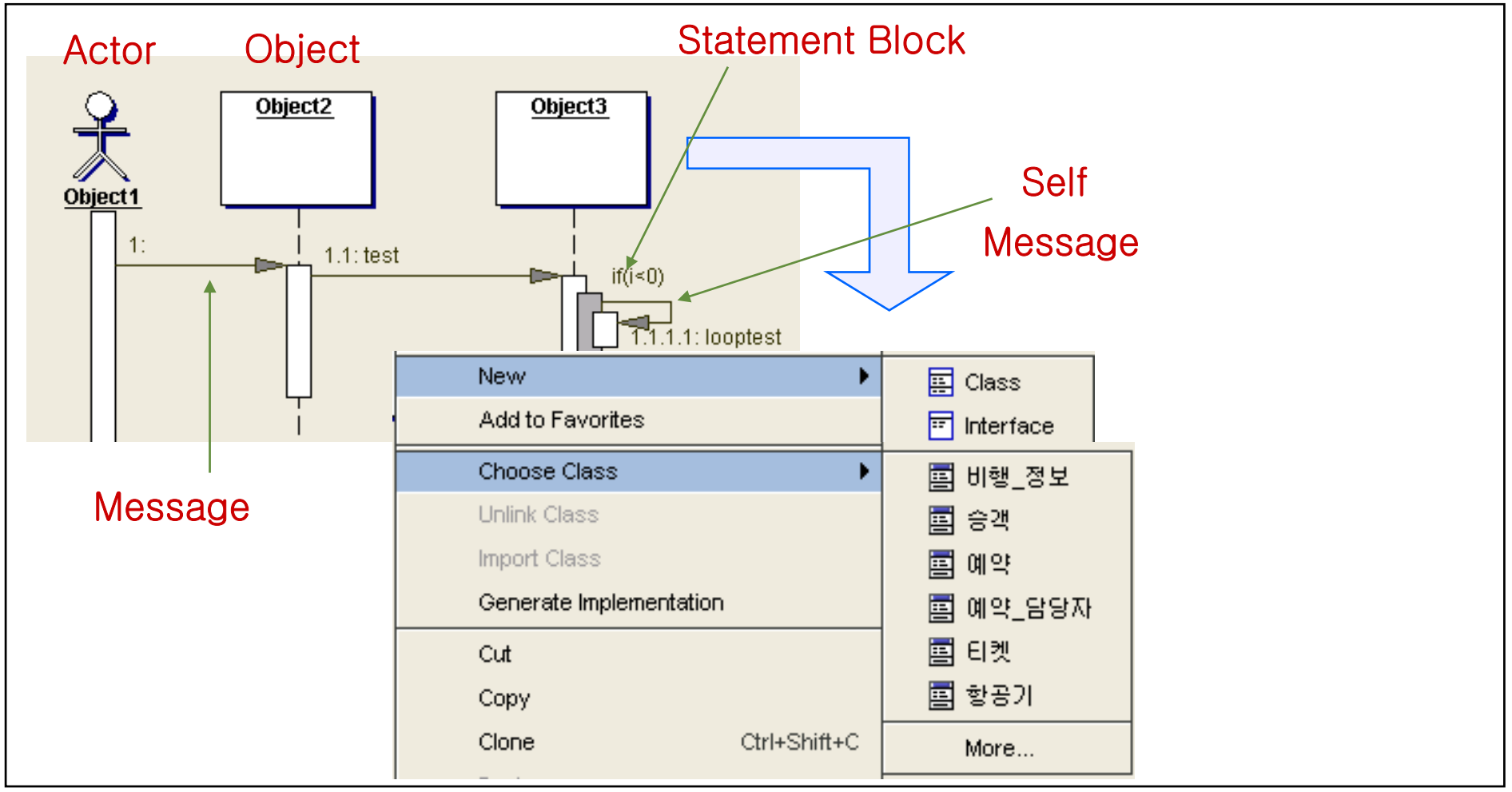
Sequence 다이어그램은 주로 분석과 설계과정에서 객체 간의 메시지 전달을 시간적 흐름에 따라 분석 할 때 주로 사용되는 다이어그램으로 객체 분석과 설계과정에서 이용된다.



OOC
Open Objected Component

Borland®

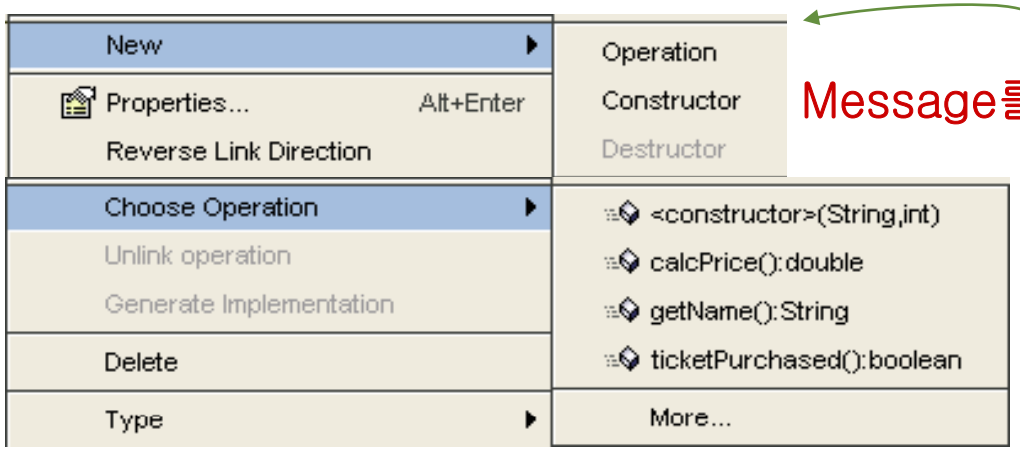
Sequence Diagram



OOC
Open Objected Component

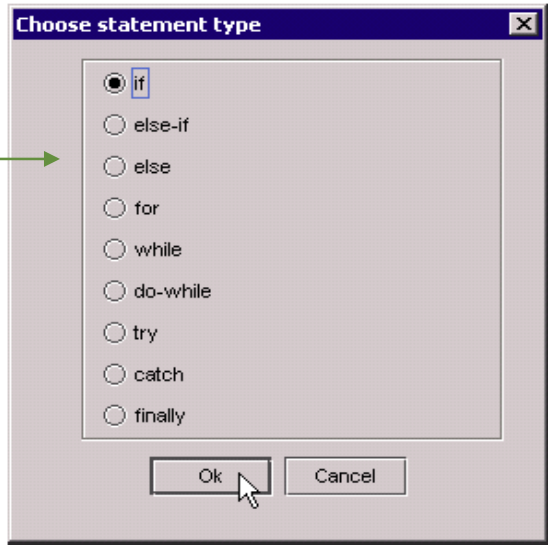
Borland®

Sequence Diagram



Message를 Operation으로 변경 및 설정

Statement block 리스트



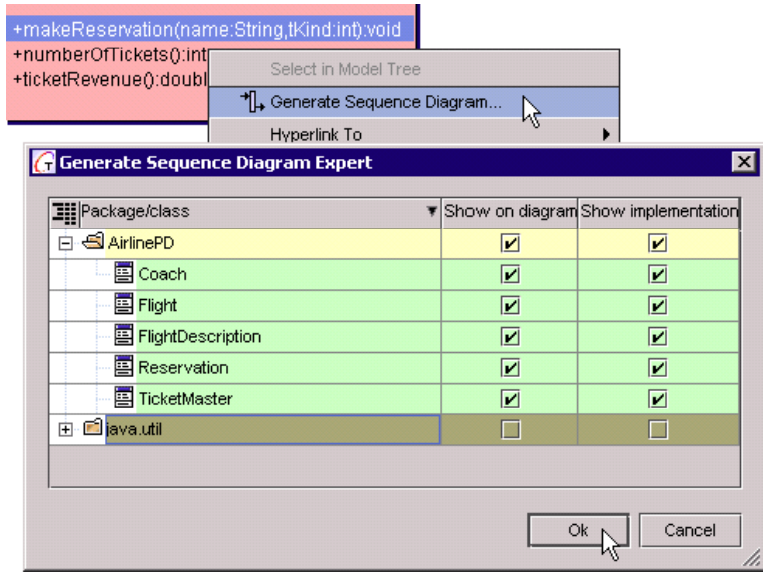
소스 코드 생성

OOC
Open Objected Component

Borland®

Sequence Diagram

- 소스 코드에서 Diagram 추출
  - Generate Sequence Diagram

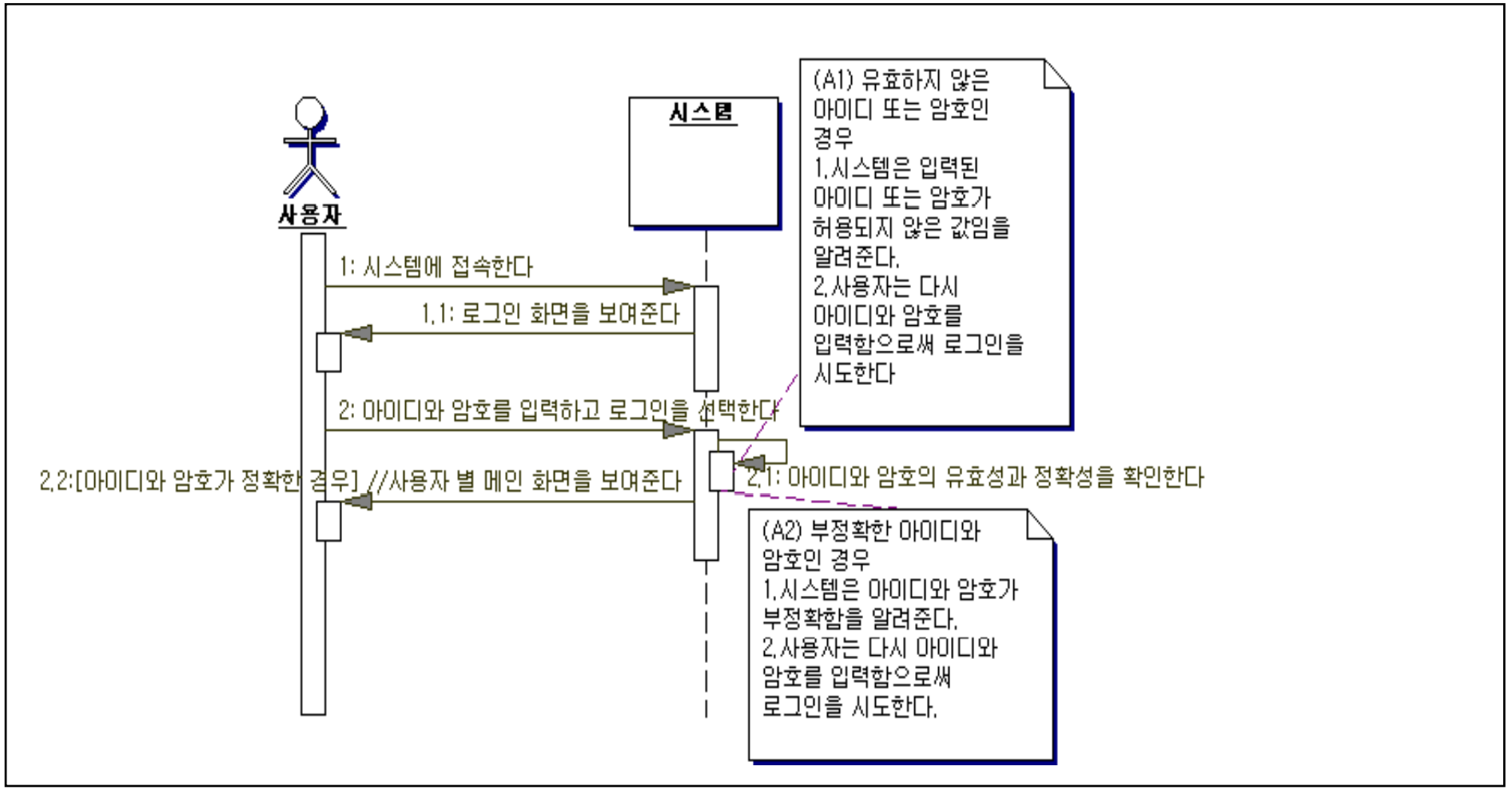


- Collaboration Diagram으로 변경하기
  - Show as Collaboration

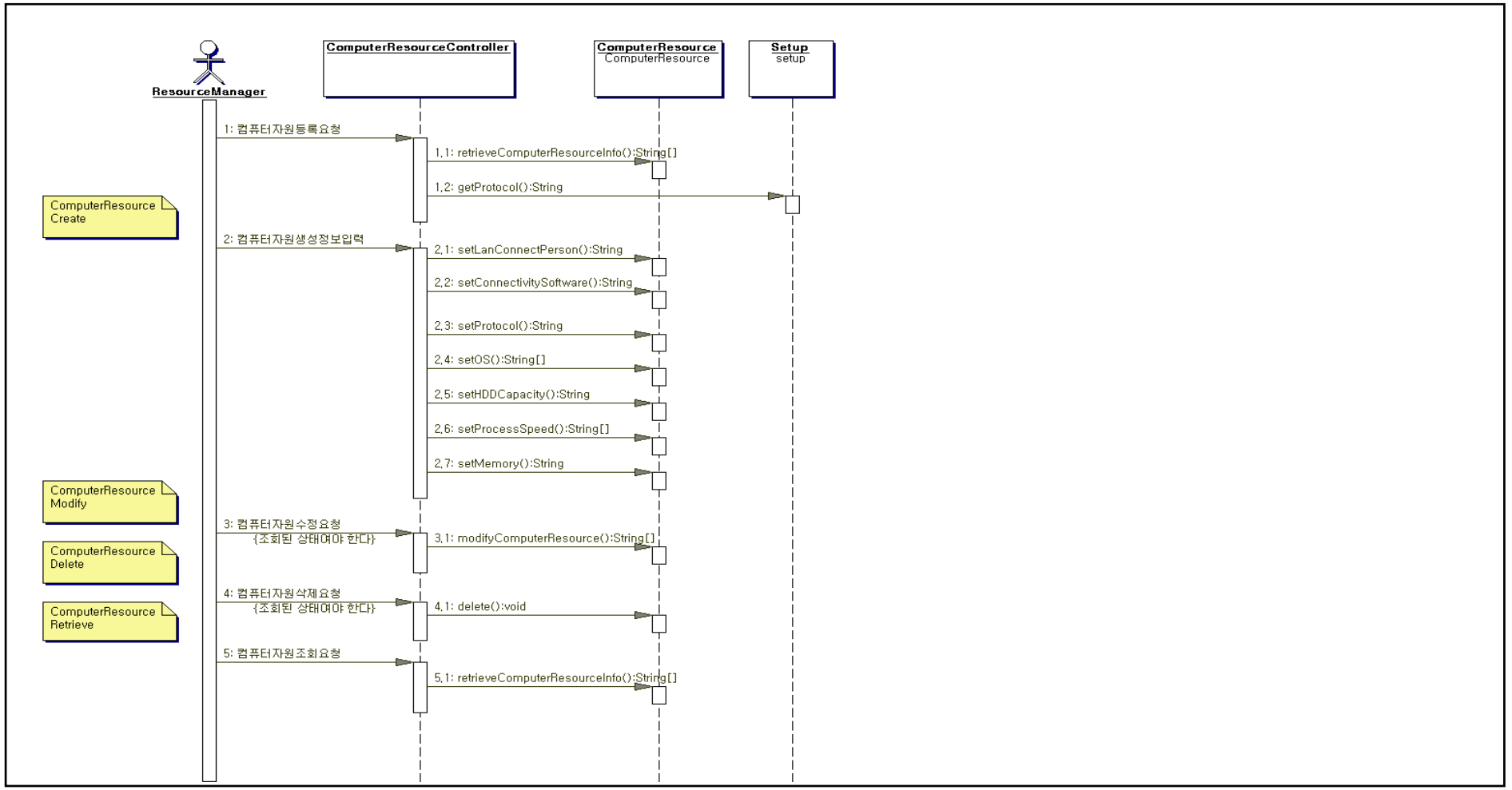
OOC
Open Objected Component

Borland®

Sequence Diagram - 분석



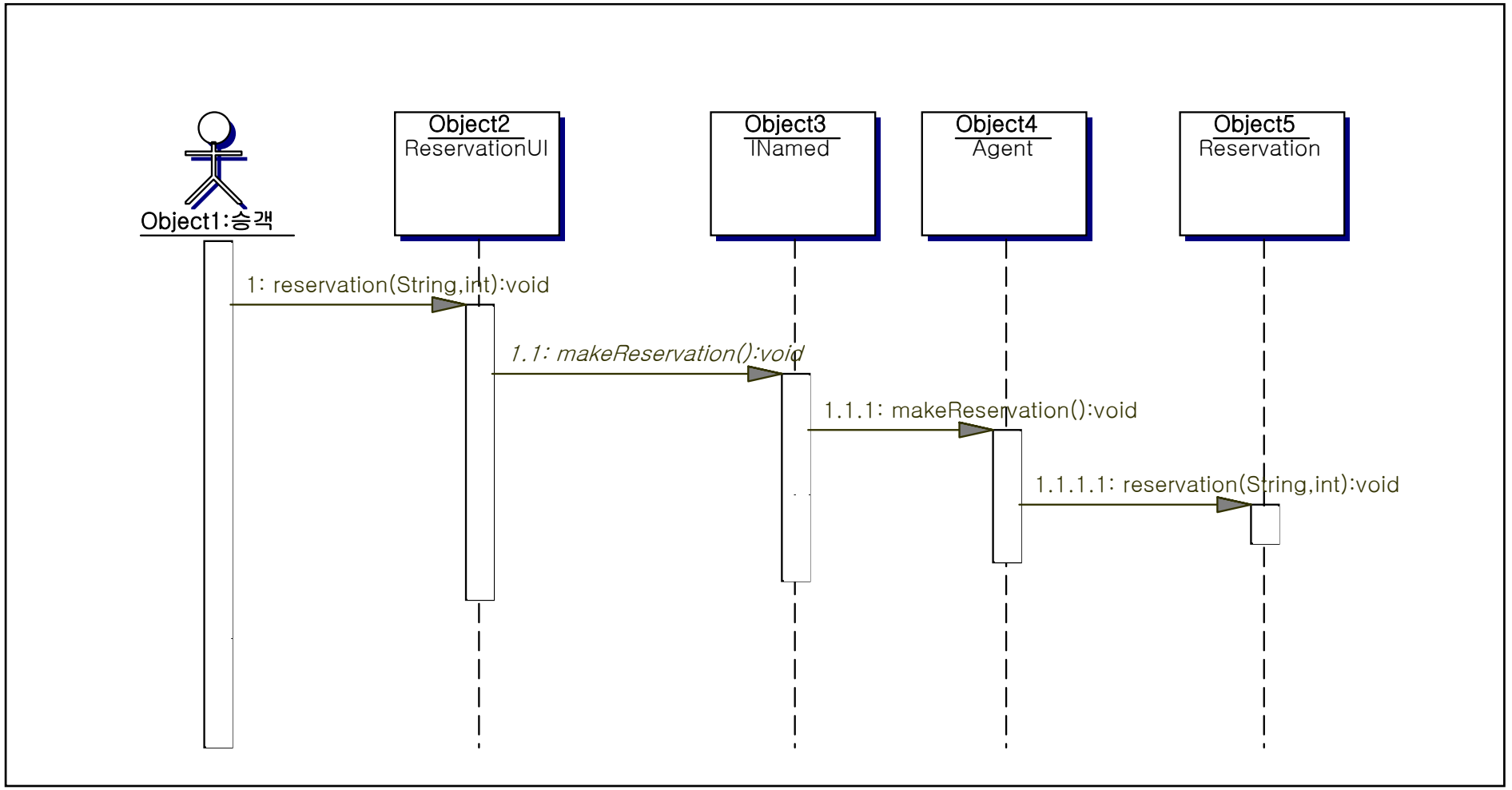
Sequence Diagram - 설계



OOC
Open Objected Component

Borland®

Sequence Diagram - 실습예제



OOC
Open Objected Component

Borland®

Component Diagram

Component Diagram

- 컴포넌트는 다른 컴포넌트에서 호출할 수 있는 잘 정의된 인터페이스를 공개
- 용도
  - 컴포넌트는 외부 라이브러리, COM 컴포넌트, jar 파일, Enterprise Java Bean, 가상 머신, 운영 체제 등의 요소를 나타내는 물리적인 포장 단위
- 구성
  - Component, Interface, Support, Dependency, SubSystem
- 목표
  - 컴포넌트 라고 불리는 요소 사이의 대략적인 상호작용과 의존성을 표시

OOC
Open Objected Component

Borland®

Component Diagram

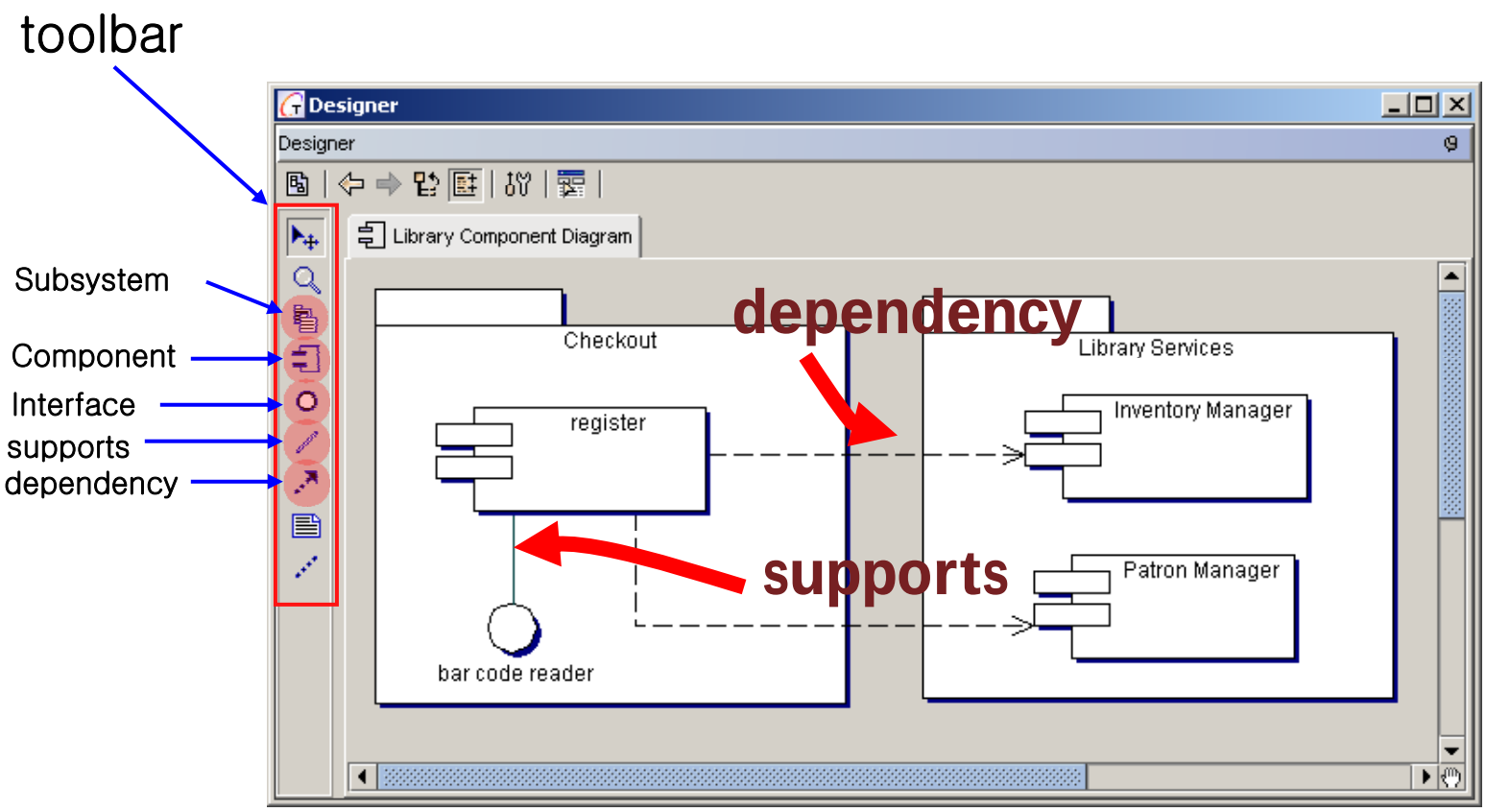
Component Diagram Notation

Together toolbar	Together Default UML Notation	UML Notation 설명	비고
	Package	시스템의 패키지를 표현한다.	
	Component	컴포넌트를 표현한다.	
	Interface	인터페이스를 표현한다.	
	Support	컴포넌트와 이와 관련된 인터페이스와의 연결을 표현한다.	
	Dependency	컴포넌트간(인터페이스간)의 연결을 표현한다.	
	Note	다이어그램 요소에 대한 설명을 표현한다.	
	Note Link	노트와 다이어그램 요소간의 연결을 표현한다.	

OOC
Open Objected Component

Borland®

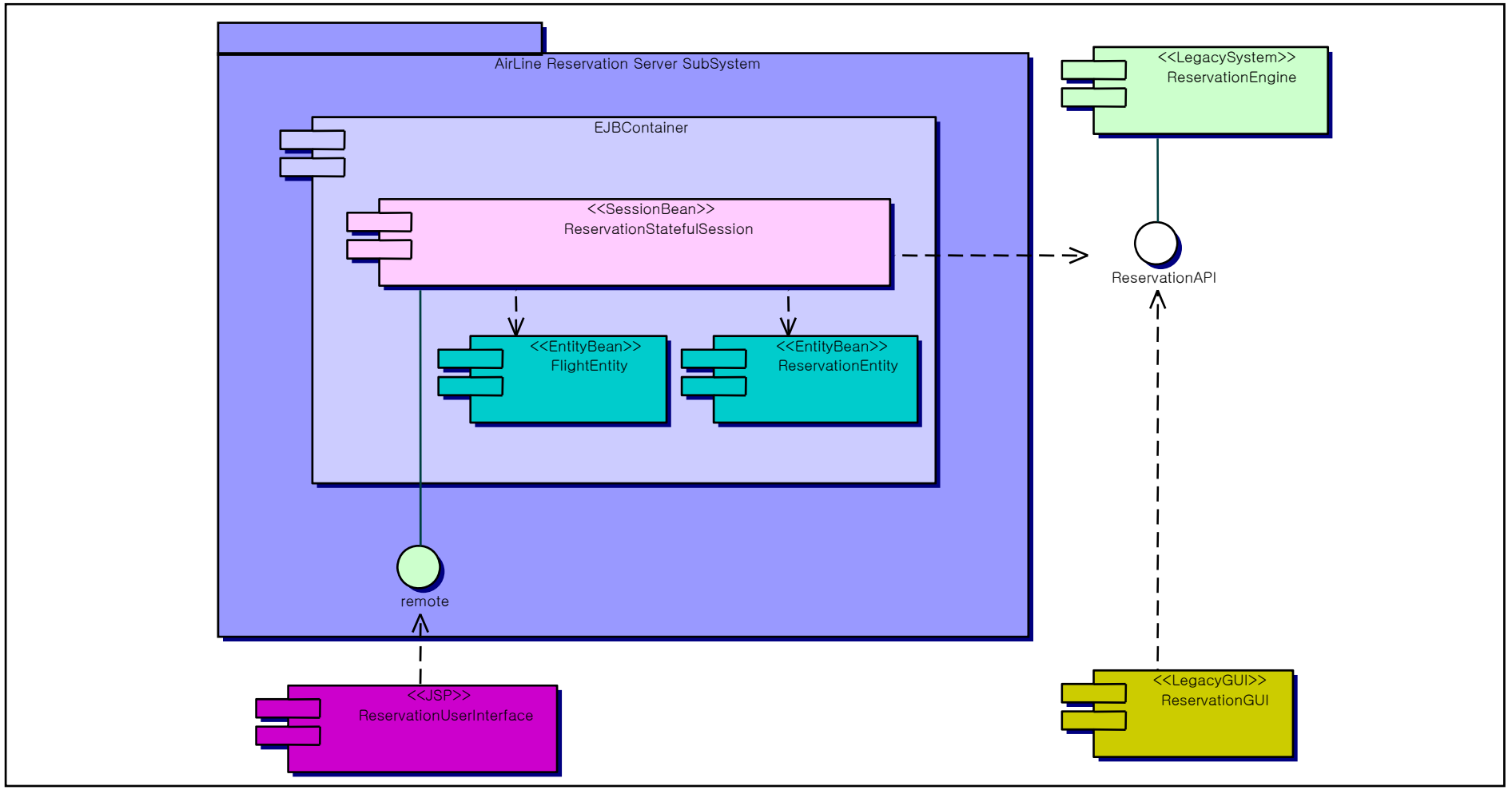
Component Diagram



OOC
Open Objected Component

Borland®

Component Diagram 실습예제



OOC
Open Objected Component

Borland®

Deployment Diagram

Deployment Diagram

- 물리적인 하드웨어는 노드로 구성
- 노드(Node) : 컴포넌트가 배포되는 물리적인 머신
- 노드 사이의 링크 : 논리적 또는 물리적인 연결
- 이런 링크를 연관(Association)이라 하고 연관은 물리적 장치 사이의 연결을 나타낸다. 여기에도 이름을 기술하거나 그 외의 방법으로 관련 연결 타입을 표시할 수 있다

- 용도
  - 컴포넌트가 배포되는 물리적인 장치사의 연결을 표시, 통신방법 등.

- 구성
  - Node, Component, Interface, Supports, Associates, Aggregates, Object, Dependency

- 목표
  - 소프트웨어와 하드웨어의 물리적인 형상 표시

Deployment Diagram

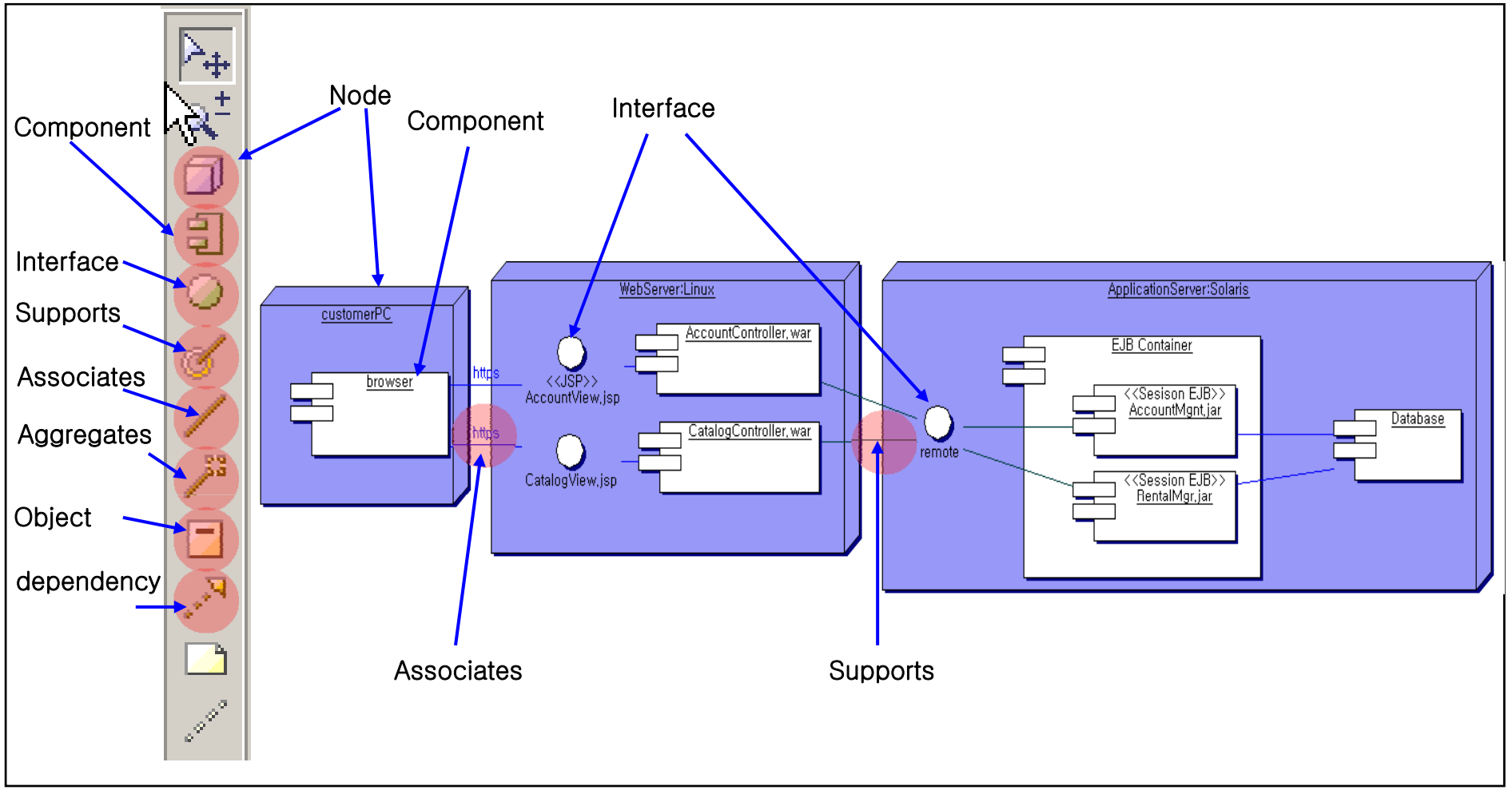
Deployment Diagram Notation

Together toolbar	Together Default UML Notation	UML Notation 설명	비고
	Node	시스템의 개별 노드를 표현한다.	
	Component	시스템의 핵심적인 컴포넌트를 표현한다.	
	Interface	컴포넌트가 제공하는 인터페이스를 표현한다.	
	Support	컴포넌트와 인터페이스를 제공하는 컴포넌트의 관계를 표현한다.	
	Associates	각 노드 간의 연관관계를 표현한다.	
	Dependency	각 노드나 컴포넌트의 의존관계를 표현한다.	
	Note	다이어그램 요소에 대한 설명을 표현한다.	
	Note Link	노트와 다이어그램 요소간의 연결을 표현한다.	

OOC
Open Objected Component

Borland®

Deployment Diagram



OOC
Open Objected Component

Borland®

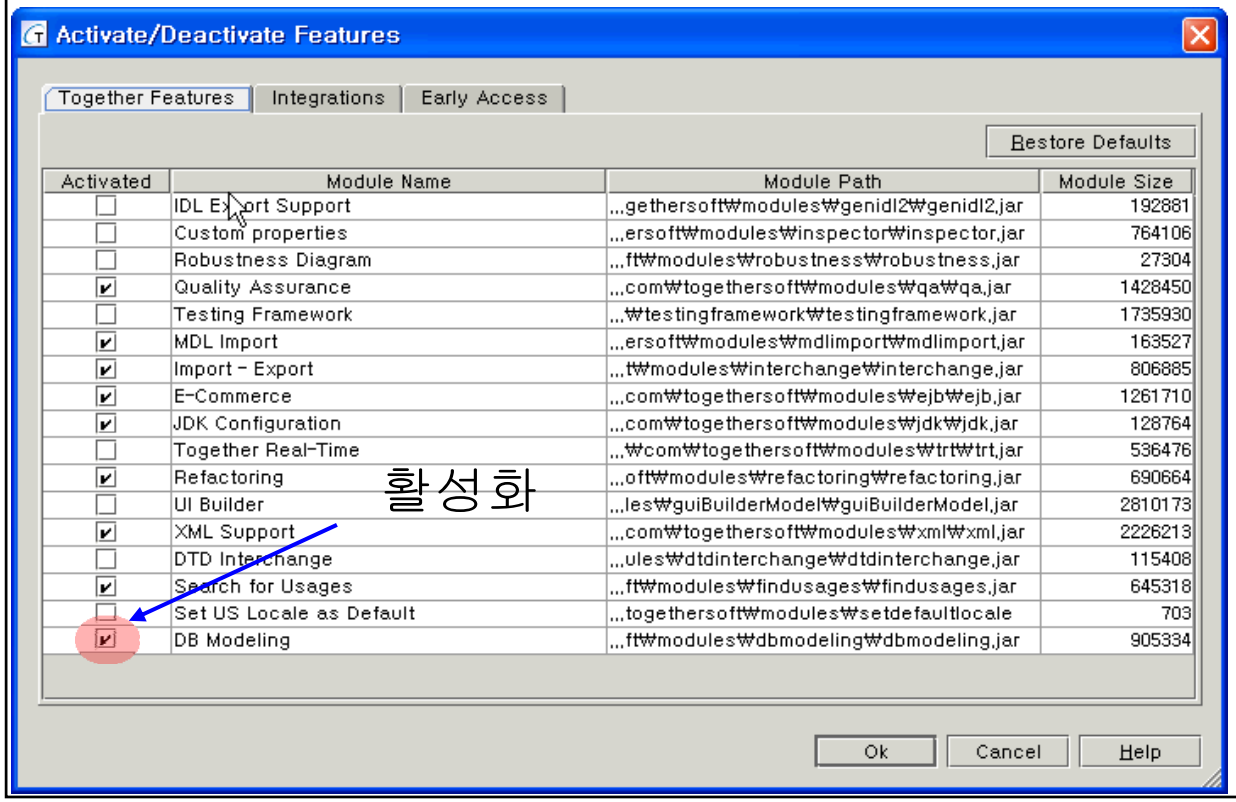
Together를 활용한 DBMS 실무

Borland®

DBMS 환경설정

Activate / Deactivate DBMS 환경설정

DB 설계과정 중 논리적/물리적 데이터모델을 통해 주어진 저장 아키텍쳐내에서 가장 효율적인 방법으로 업무를 표현



Activity Step 1

❖ Activate/Deactivate DB Modeling

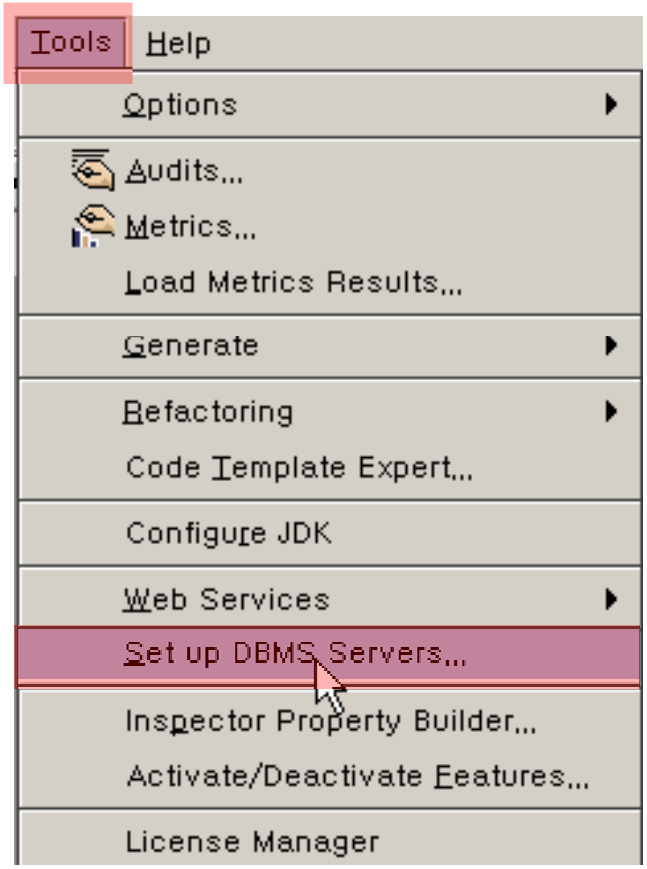
메인 메뉴 → Tools | Activate/Deactivate Features 를 선택 → Together Features 탭에서 DB Modeling을 체크

OOC
Open Objected Component

Borland®

DBMS 환경설정

DBMS 환경설정



Activity Step 2

❖ Set up DBMS Server

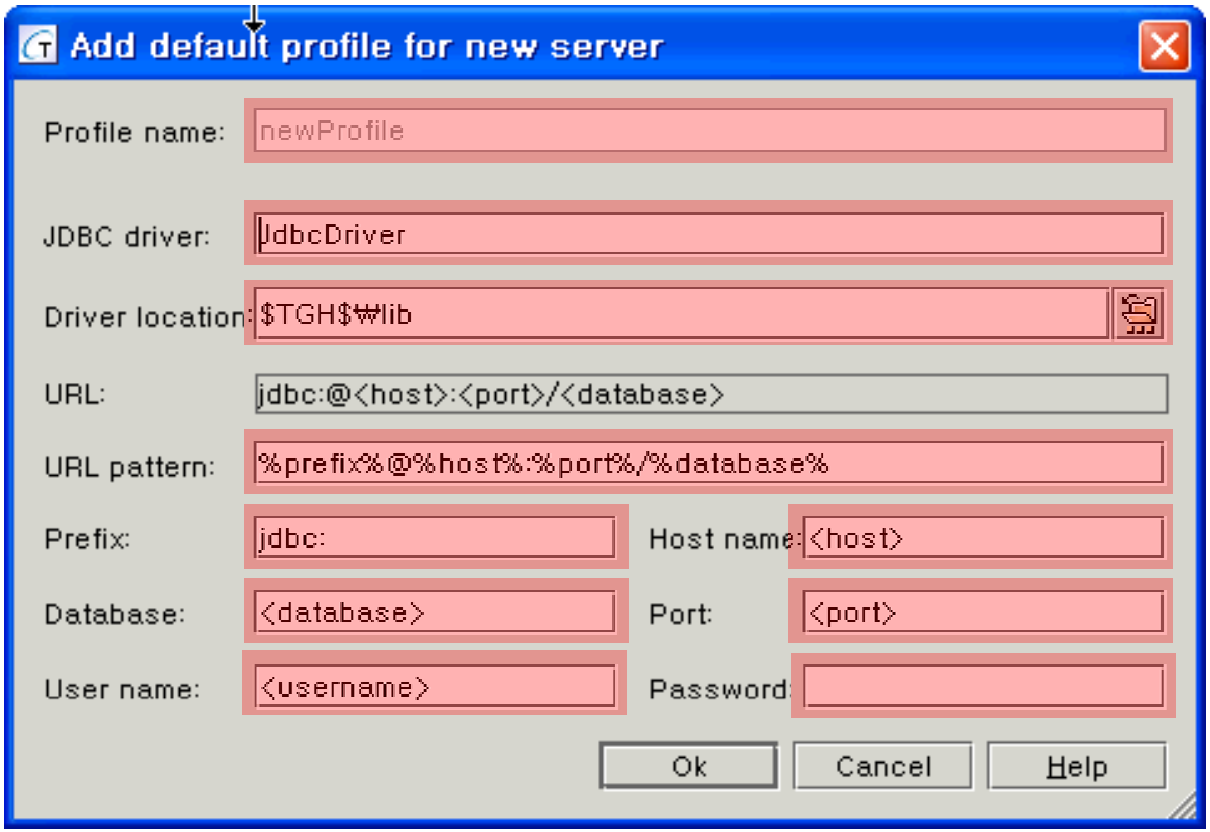
메인 메뉴 → Tools | Set up DBMS Servers 선택

OOC
Open Objected Component

Borland®

DBMS 환경설정

DBMS 환경설정 (계속)



Activity Step 3

❖ Add default profile for new server

메인 메뉴 → Tools | Set up DBMS Servers 선택 → DBMS 정보 입력

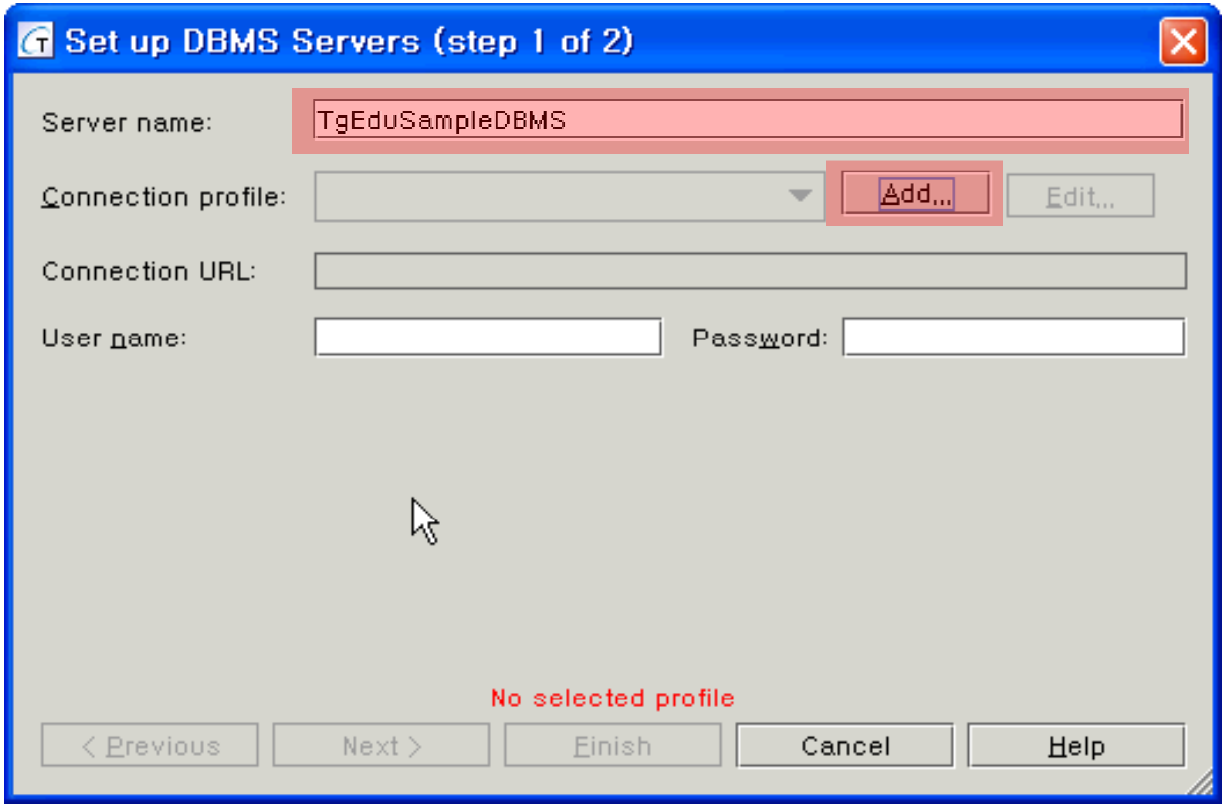
1. Profile name : EduSample
2. JDBC Driver : oracle.jdbc.driver.OracleDriver
3. Driver location : C:₩Borland₩TogetherArchitect ₩lib₩classes12.zip
4. URL pattern : %prefix%@%host%:%port%:%database%
5. Prefix : jdbc:oracle:thin:
6. Host Name : localhost(IP Address)
7. Database : orcl
8. Port : 1521
9. UserName : scott
10. Password : tiger

OOC
Open Objected Component

Borland®

DBMS 환경설정

DBMS 환경설정 (계속)



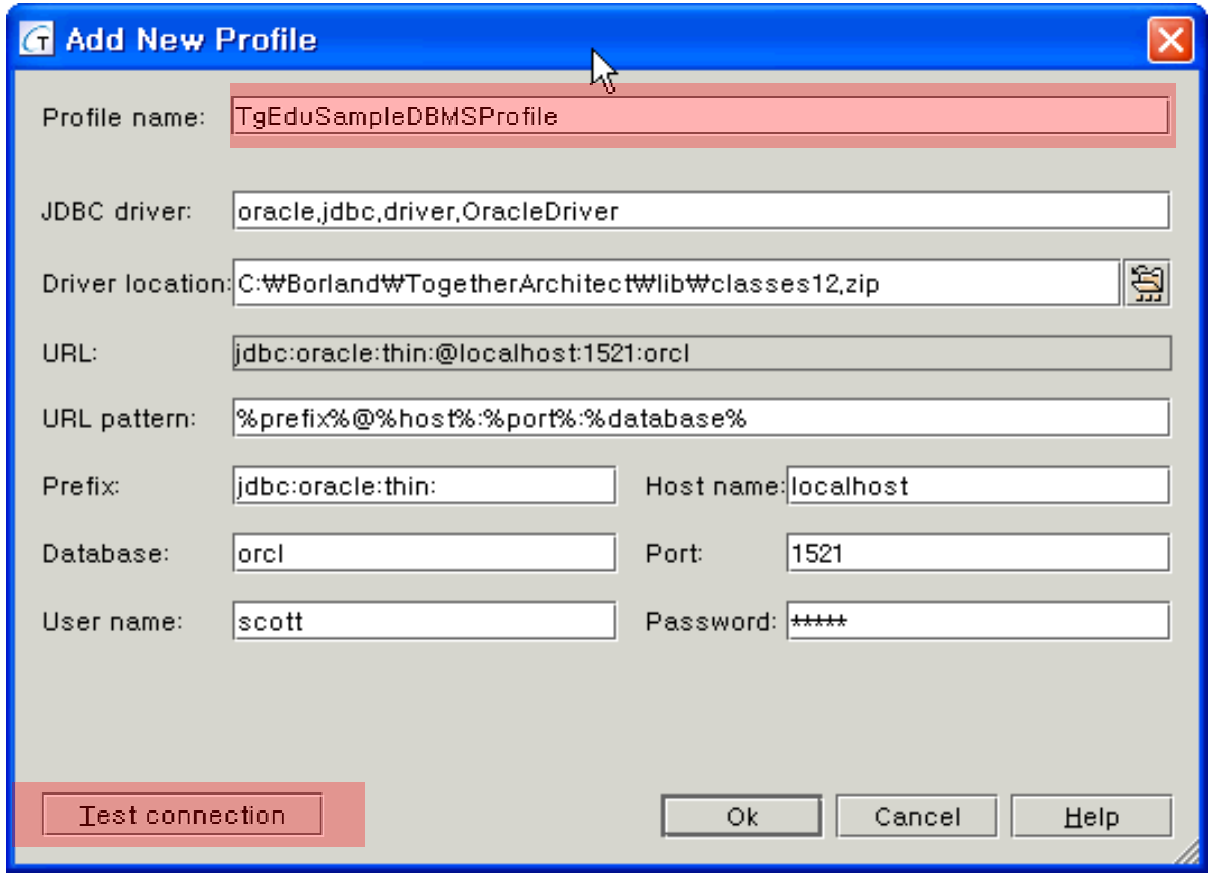
Activity Step 4

❖ Add default profile for new server

1. Server Name : EduSampleDBMS
2. Connection profile : Add

DBMS 환경설정

DBMS 환경설정 (계속)



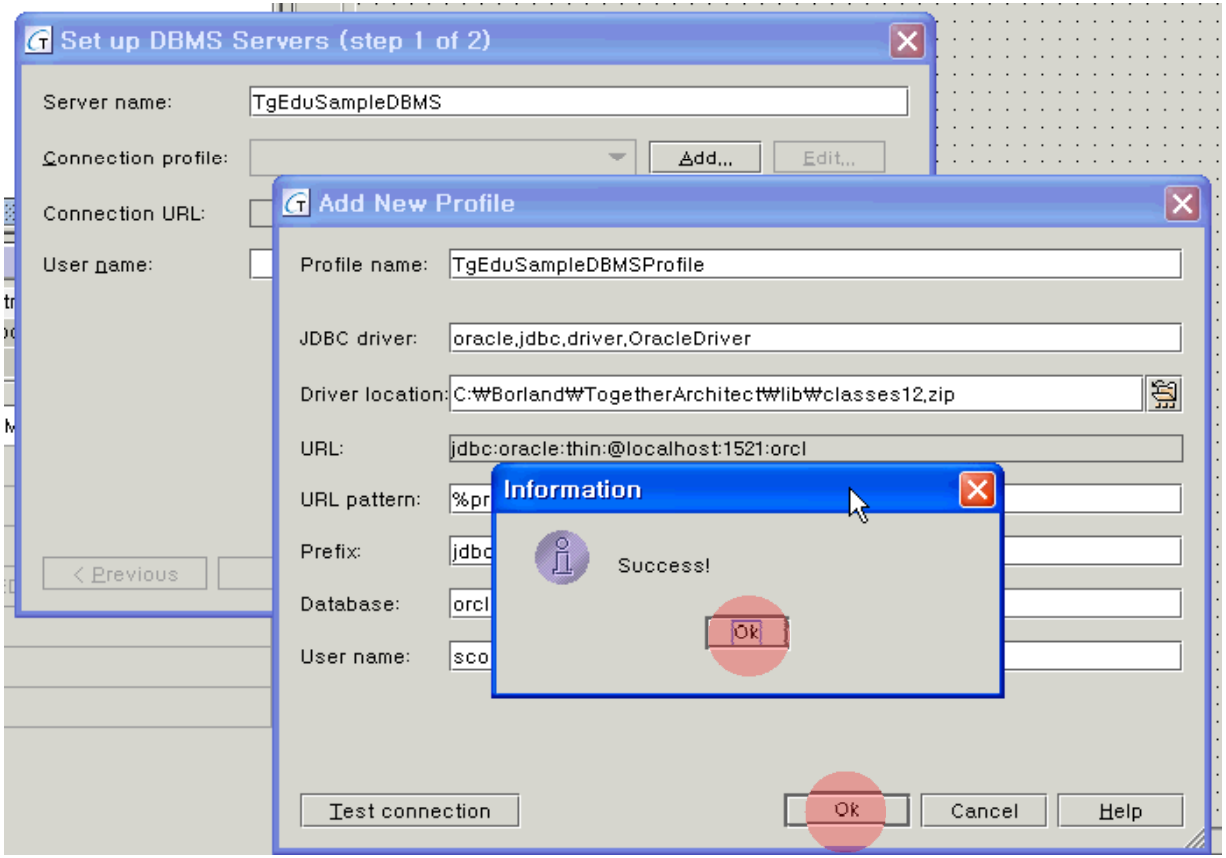
Activity Step 5

- ❖ Add default profile for new server

1. Profile name : TgEduSampleDBMSProfile
2. Test connection 버튼 클릭

DBMS 환경설정

DBMS 환경설정 (계속)



Activity Step 6

❖ Add default profile for new server

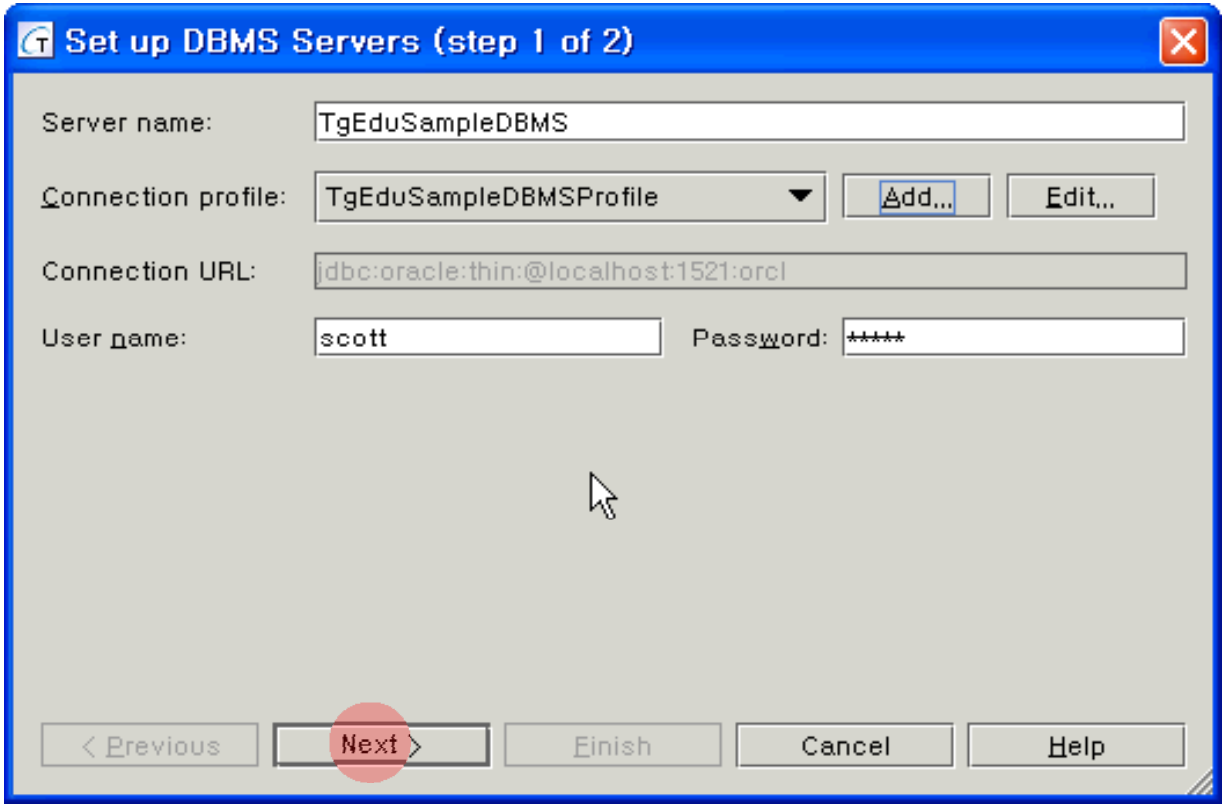
1. Sussess ! → Ok 버튼 클릭
2. Add New Profile 창의 Ok 버튼 선택

OOC
Open Objected Component

Borland®

DBMS 환경설정

DBMS 환경설정 (계속)



Activity Step 7

❖ Set up DBMS Servers (Step 1 of 2)

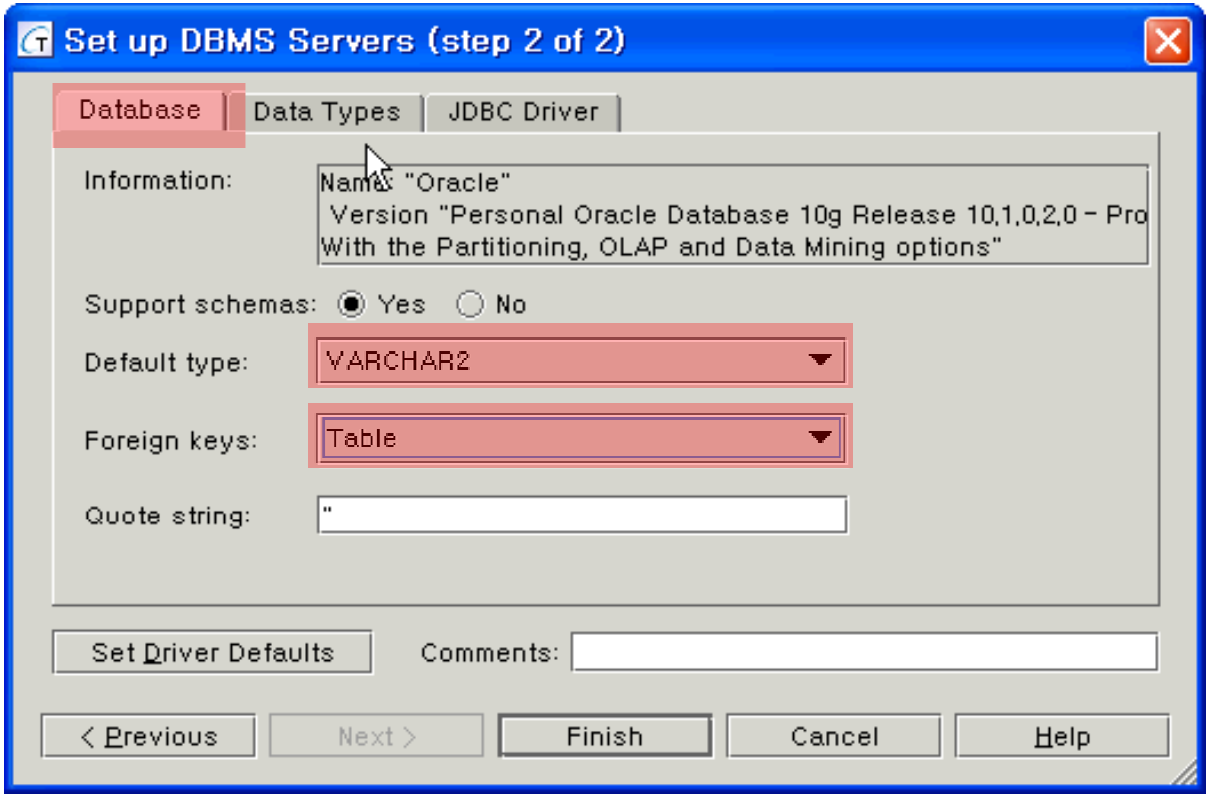
1. Next 버튼 클릭

OOC
Open Objected Component

Borland®

DBMS 환경설정

DBMS 환경설정 (계속)



Activity Step 8

❖ Set up DBMS Servers (Database Tab)

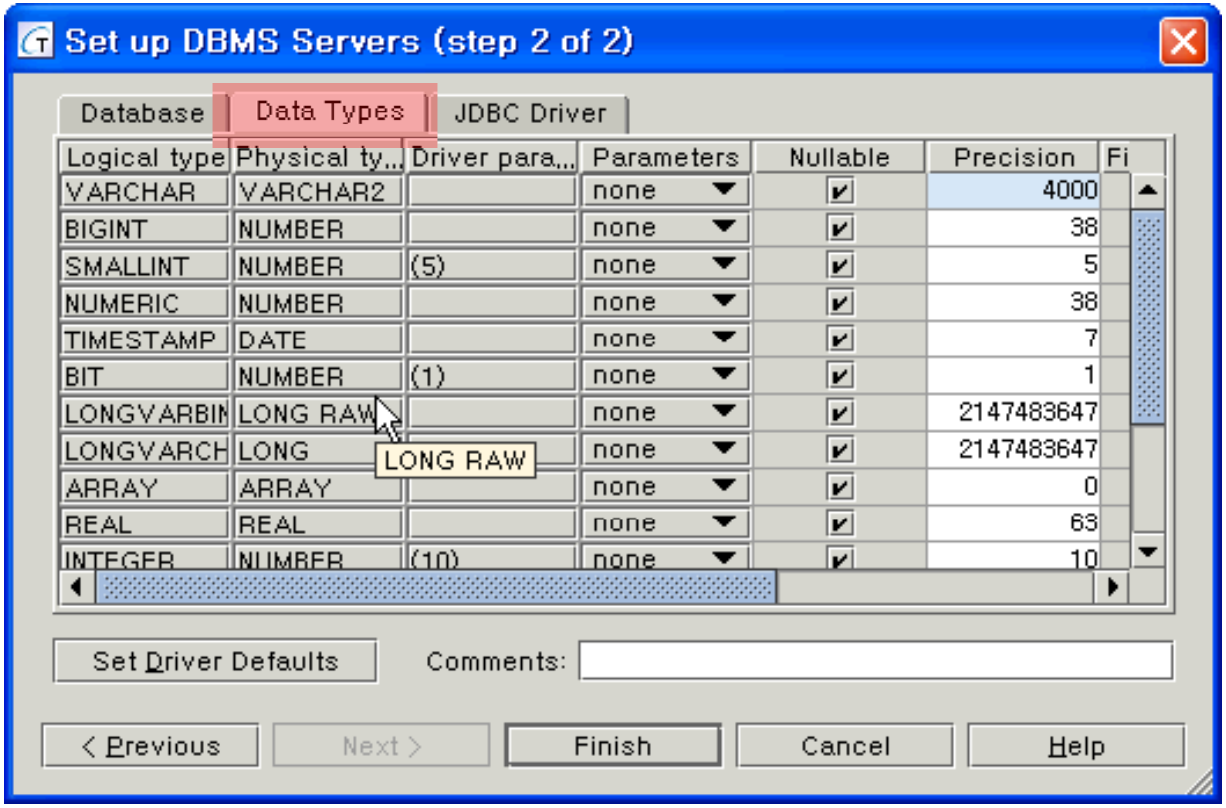
1. Default type : VARCHAR2
2. Foreign keys : Table
3. Data Types 탭 선택

OOC
Open Objected Component

Borland®

DBMS 환경설정

DBMS 환경설정 (계속)



Activity Step 9

❖ Set up DBMS Servers (Data Type Tab)

- JDBC Driver에서 사용가능한 데이터 종류에 대한 조건을 설정하는 단계
- 데이터는 JDBC Driver 마다 종류와 크기가 다름

1. **Logical type** : JDBC에서 인식하는 데이터 타입들을 목록으로 표시
2. **Physical type** : DBMS에서 인식하는 데이터 타입으로 Logical type과 대응되는 것을 표기

(다음 장 계속)

OOC Open Objected Component

Borland®

DBMS 환경설정

DBMS 환경설정 (계속)

Activity Step 9

❖ Set up DBMS Servers (Data Type Tab) 계속

3.Driver parameter value : JDBC Driver를 통해 반환되는 데이터 타입에 대한 파라메터 값을 설정한다. (예) varchar(30) – 30 이 파라메터에 해당

- Parameters : 위에서 선택한 파라미터에 제약사항을 두고자 할 때 사용. 제약사항은 다음과 같이 4가지가 존재한다. Default로 None 이 설정된다.
- None : 파라미터를 사용하지 않는다
- %size% : 데이터 타입의 크기를 설정한다 (예) Char(30)
- %precision%, %scale%, template : 데이터 타입의 크기를 소수로 표현할 때 사용, precision은 데이터의 총 길이를 나타내고, scale은 소수점 이하 자릿수를 나타낸다. (예) Decimal(10,2)
- Default value : 데이터 타입이 아닌 실제 값을 반환하는 경우에 사용한다.

Nullable : 데이터 타입이 NULL 값을 가지는지 여부를 결정한다.

Precision : 파라메터로 %size%가 설정되면 precision 값이 %size%의 최대크기가 된다.

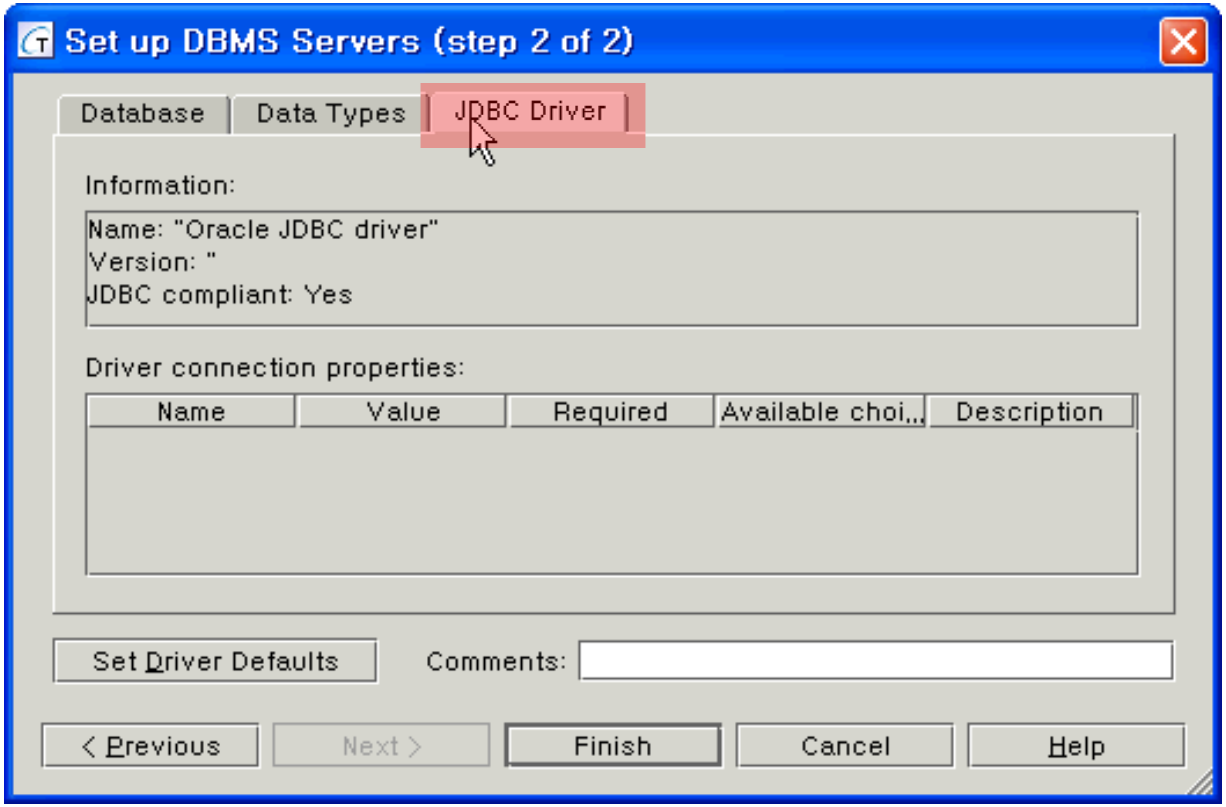
Fixed precision/scale : 고정된 precision과 scale값을 사용한다는 설정으로 체크를 하게 되면 ER Diagram을 통해 size를 수정할 수 없게 된다.

Minimum scale : %precision%, %scale%에서 %scale%의 최소 값을 설정한다.

Maximum scale : %precision%, %scale%에서 %scale%의 최대 값을 설정한다.

DBMS 환경설정

DBMS 환경설정 (계속)



Activity Step 9

❖ Set up DBMS Servers (JDBC Driver Tab)

➢ JDBC Driver 설정을 보여주는 화면으로 읽기 전용 property로 등록되어 있다.

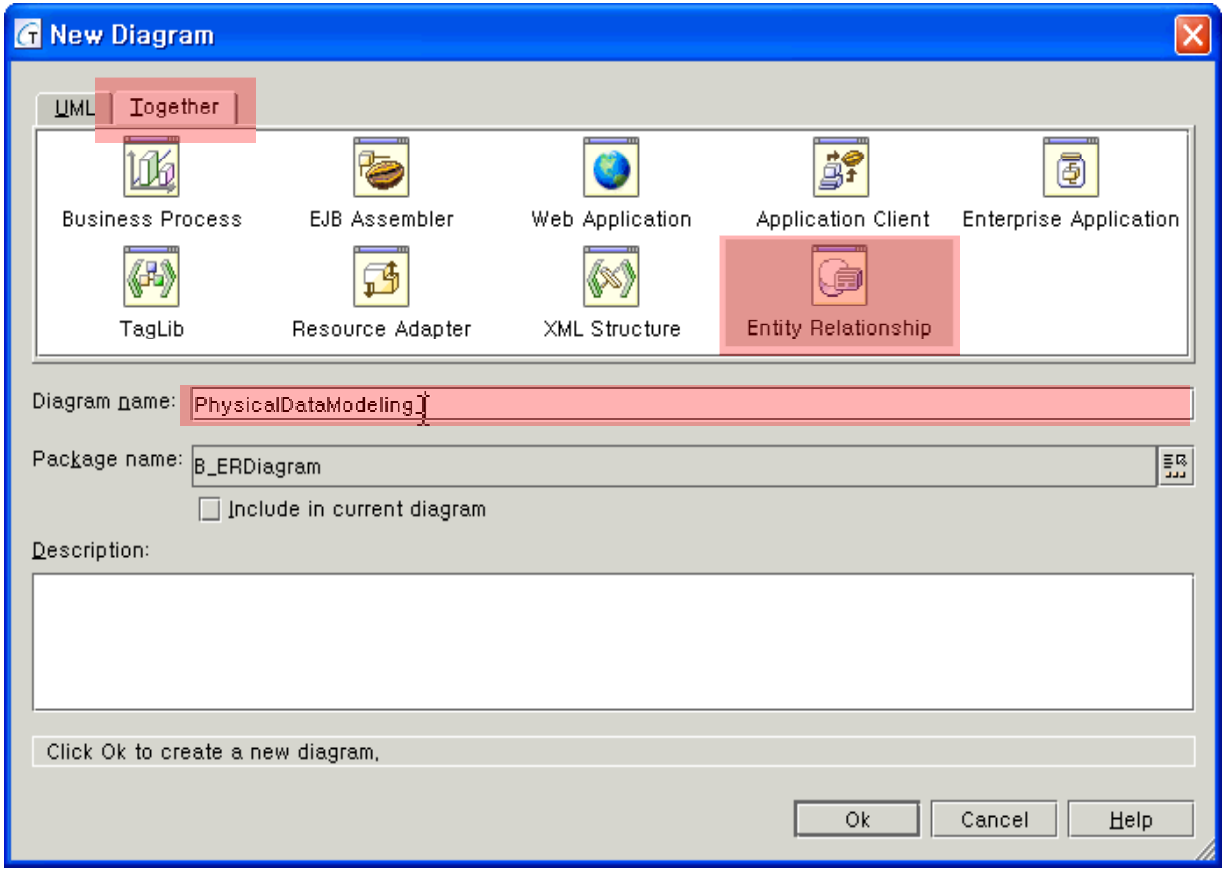
1. Information : driver의 이름과 버전 JDBC 내용과 호환이 가능한지를 표시한다. JDBC와 호환을 위해서는 SQL 92 Entry Level과 JDBC API를 완벽히 지원해야 한다.

2. Driver connection properties : 해당 driver를 통해 DBMS에 연결될 때 추가적으로 사용되는 파라미터들이다.

3. Finish를 클릭

Entity Relationship Diagram

DBMS 환경설정 (계속)



Activity Step 10

❖ ER Diagram 생성

1. Together Explorer tab → C_Design_Modeling → B_ERDiagram
2. New Diagram 생성
3. ER Diagram Name : Physical Data Modeling

OOC
Open Objected Component

Borland®

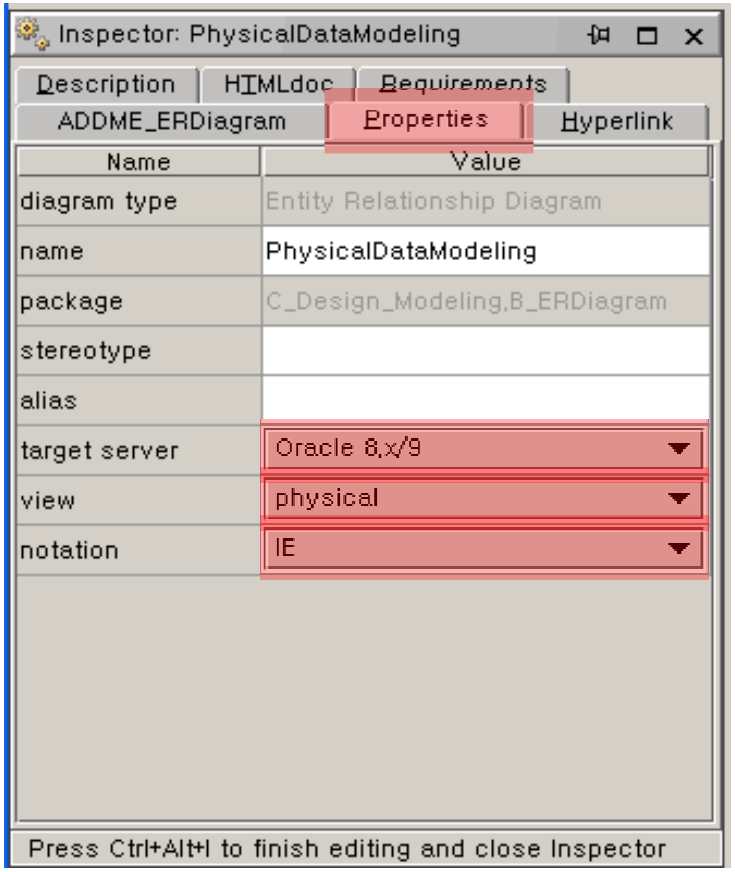
Entity Relationship Diagram

Entity Relationship Diagram

- DB 설계과정 중 논리적/물리적 데이터모델을 통해 주어진 저장 아키텍처 내에서 가장 효율적인 방법으로 업무를 표현
- 용도
  - 데이터 베이스의 Logical View와 Physical View의 분석, 설계에 사용 되며, 산출로 활용할 수 있다.
- 구성
  - Information Engineering(IE), Barker, IDEF1X
- 목표
  - RDB (Relational Database)를 기준으로 함
  - 관리하고자 하는 유용한 정보와 그 관계성에 대한 논리적/물리적 표현을 통해 업무에 활용
  - 데이터 관점에서 설계
  - 논리적/물리적 데이터 모델작성

Entity Relationship Diagram

DBMS 환경설정 (계속)



Activity Step 11

❖ ER Diagram의 Properties 환경설정

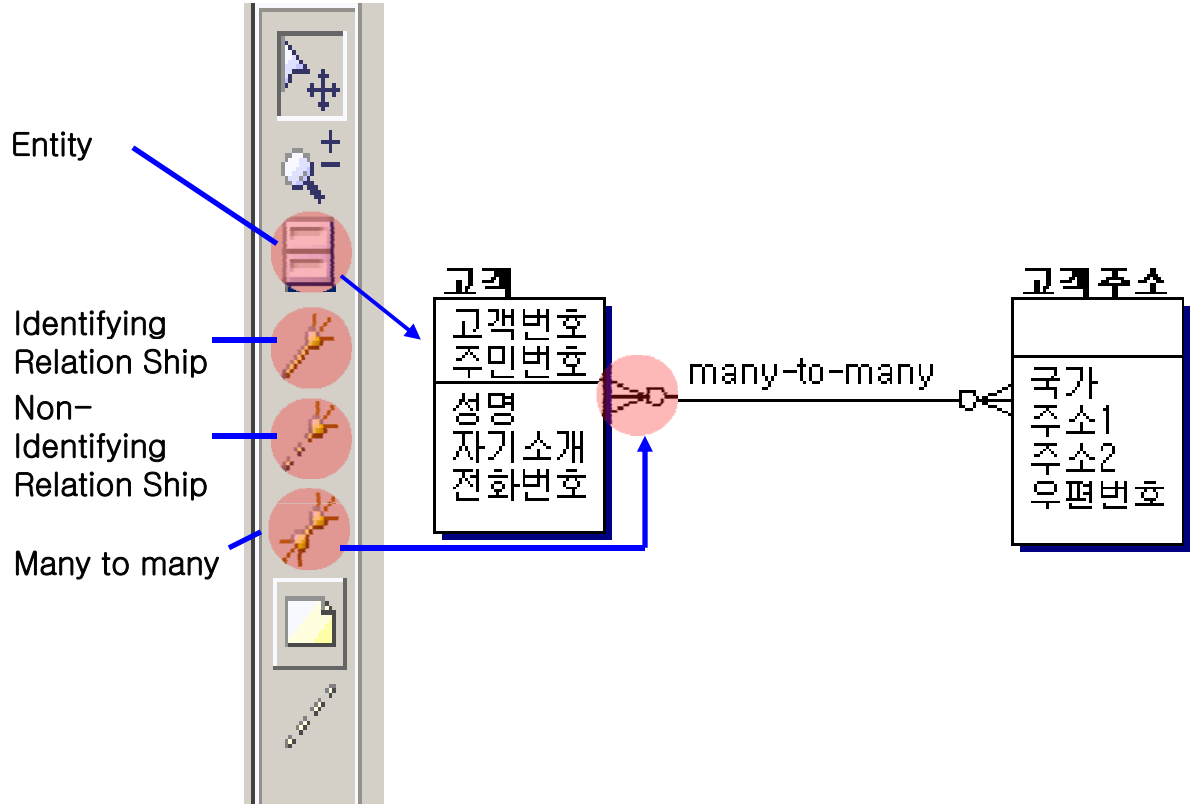
1. Target Server : Oracle 8.x/9
2. View : physical
3. Notation : IE (사용자 편의 선택)

OOC
Open Objected Component

Borland®

Entity Relationship Diagram

Entity Relationship Diagram



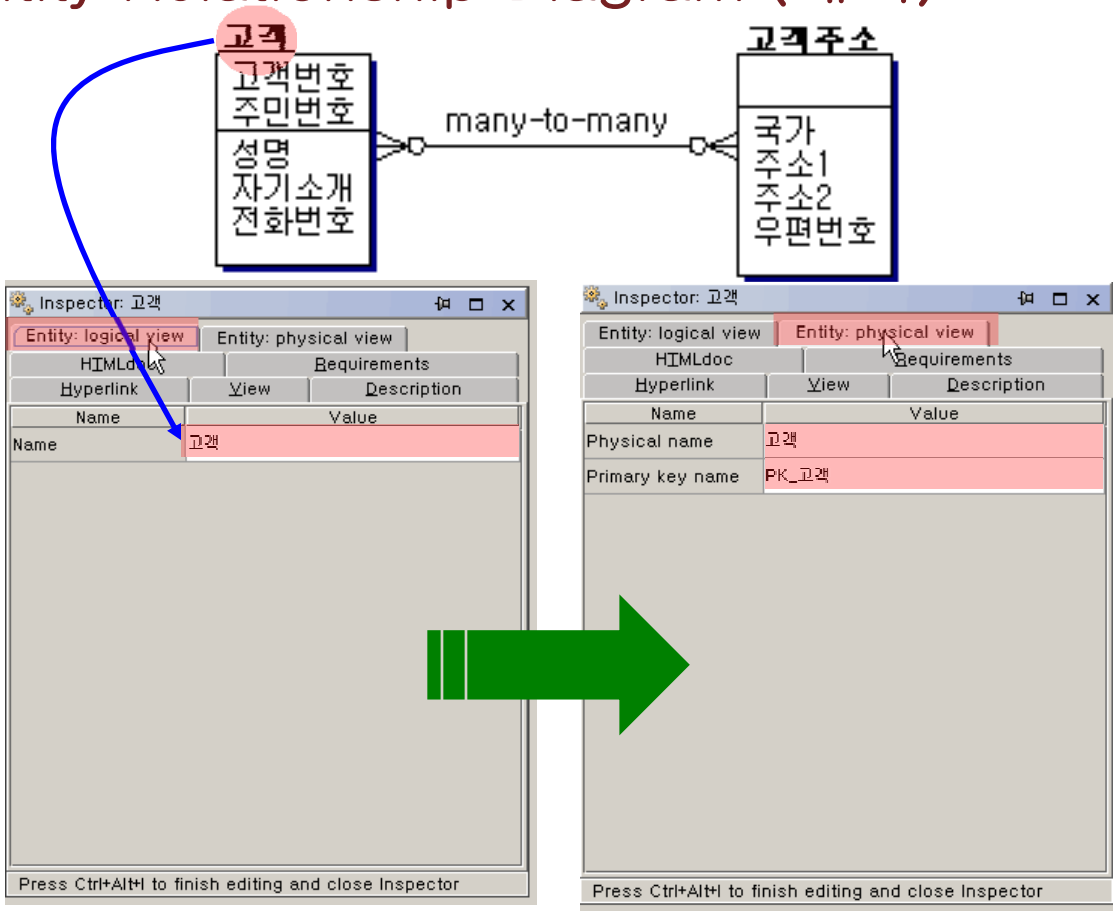
Activity Step 11

- ❖ Logical Data Modeling

1. Designer Pane을 이용하여 개념적인 데이터 모데링을 작성한다.

Entity Relationship Diagram

Entity Relationship Diagram (계속)



Activity Step 11

- ❖ Logical Data Modeling

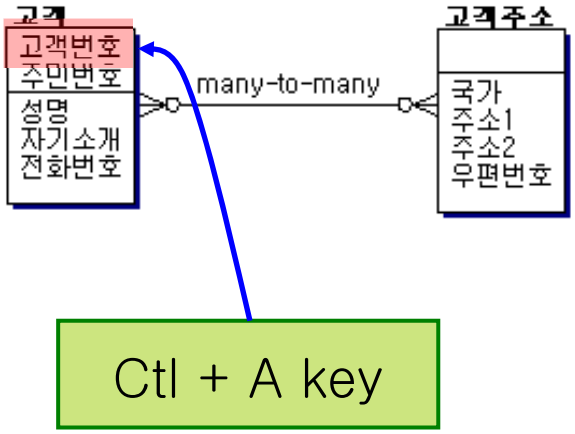
1. Inspector pane의 Entity : Logical view tab의 Entity 명 입력
   - ➢ Entity Name : 고객
   - ➢ Entity Name : 고객주소
2. Inspector pane의 Entity : Physical view tab의 Entity 명 확인

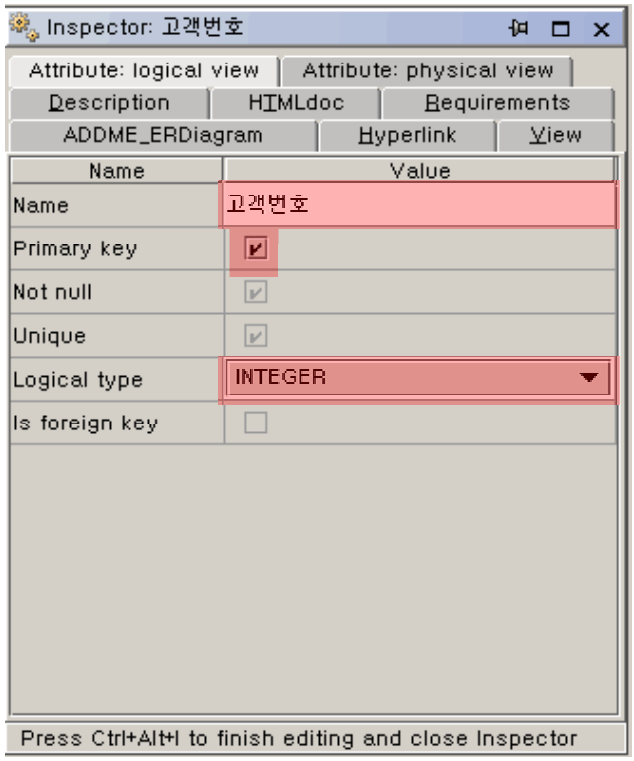
OOC
Open Objected Component

Borland®

Entity Relationship Diagram

Logical Data Modeling





Activity Step 1

❖ Entity의 attribute 추가

1. ER Diagram Open
2. Table 생성

➢ Table name : 고객

➢ Attribute Name : 고객번호

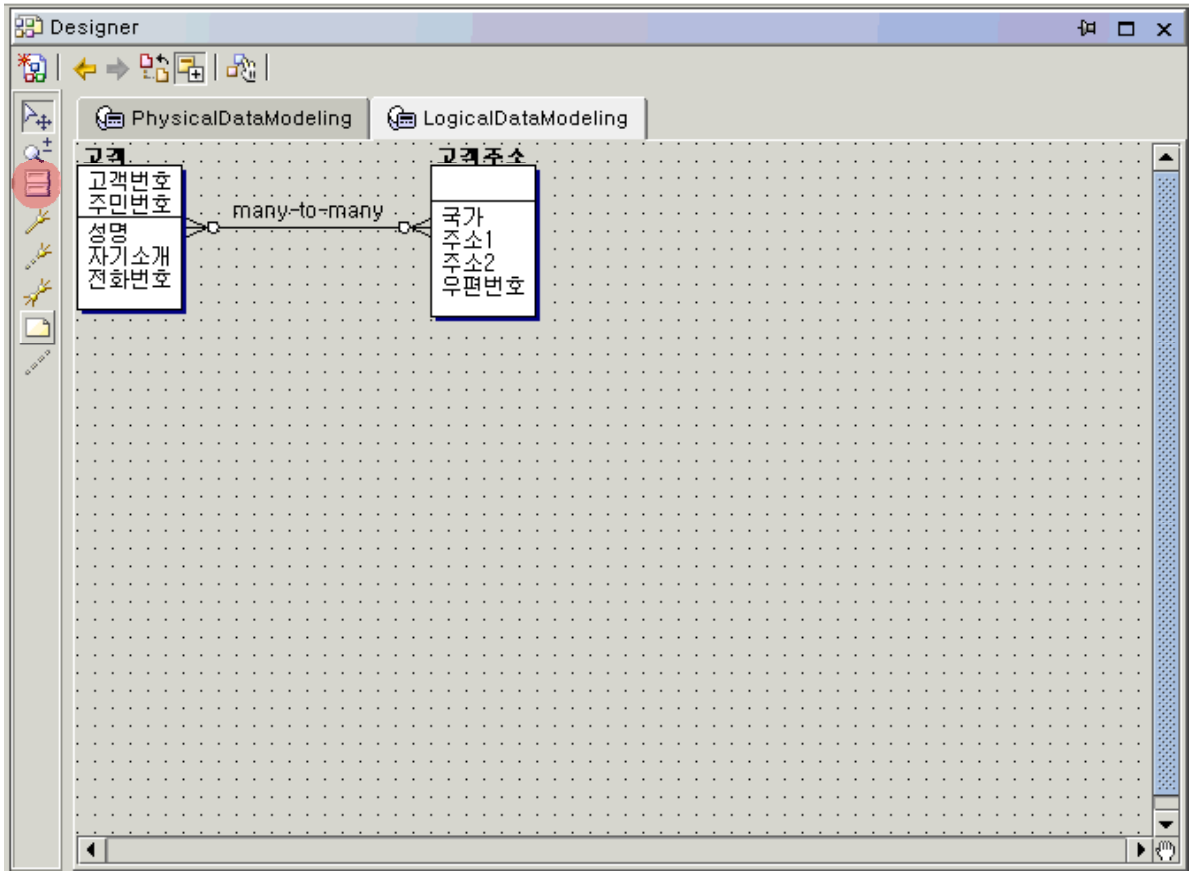
1. Notation : IE (사용자 편의 선택)

OOC
Open Objected Component

Borland®

Entity Relationship Diagram

Logical Data Modeling



Activity Step 1

- ❖ Logical Data Modeling

1. ER Diagram Open
2. Table 생성
   - ➢ Table name : 고객
   - ➢ Attribute Name : 고객번호
1. Notation : IE (사용자 편의 선택)

OOC
Open Objected Component

Borland®

Together DBMS Schema Import

Set up DBMS Servers

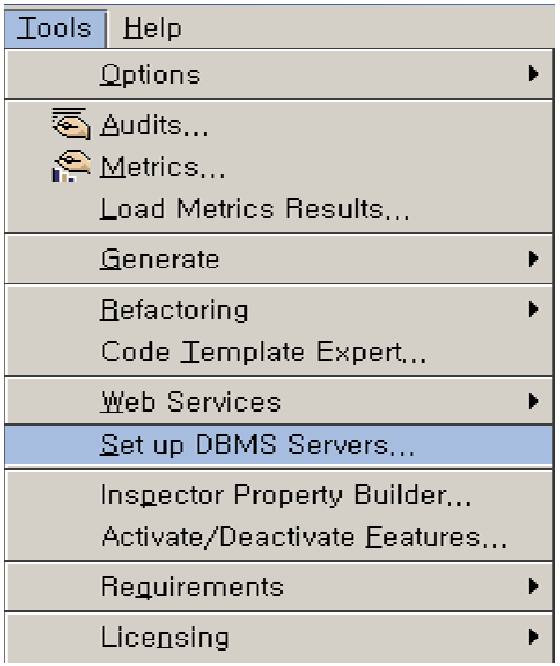
- JDBC를 이용하여 DBMS와 연결
- “Tools | Set up DBMS Server” 프로파일 작성
- “Files | Import | Database Schema” ← DB 스키마 Import

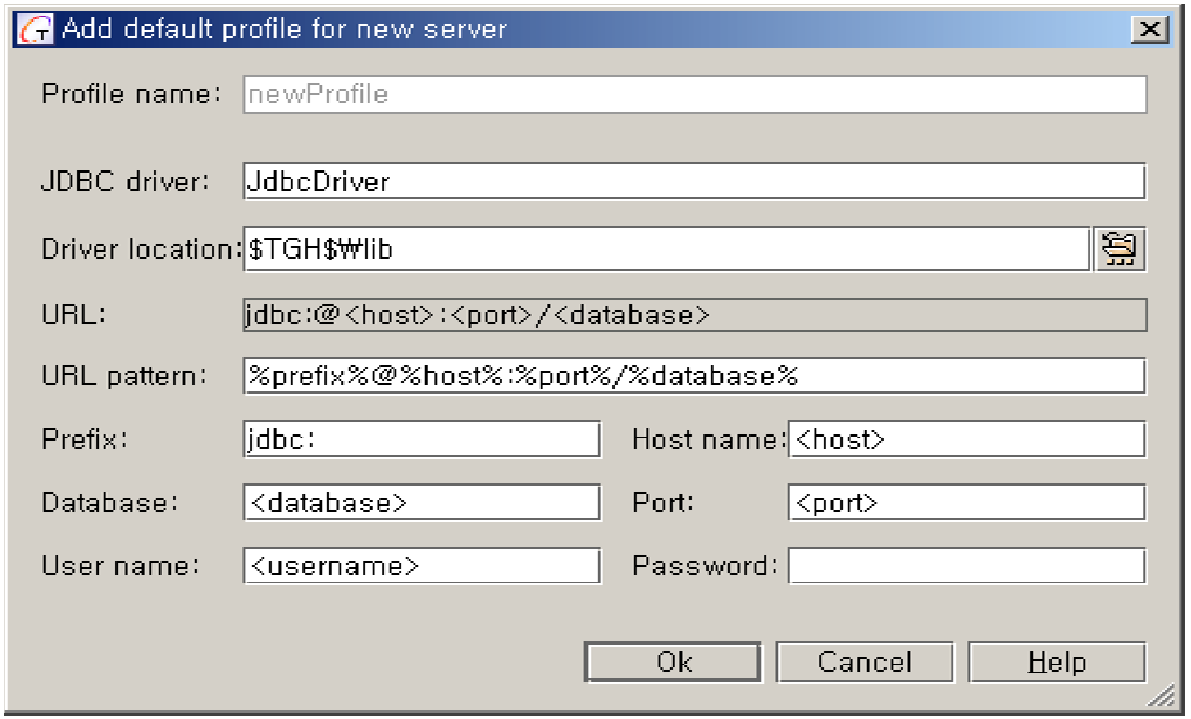
OOC
Open Objected Component

Borland®

Together DBMS 작업

Database Schema Import Step1



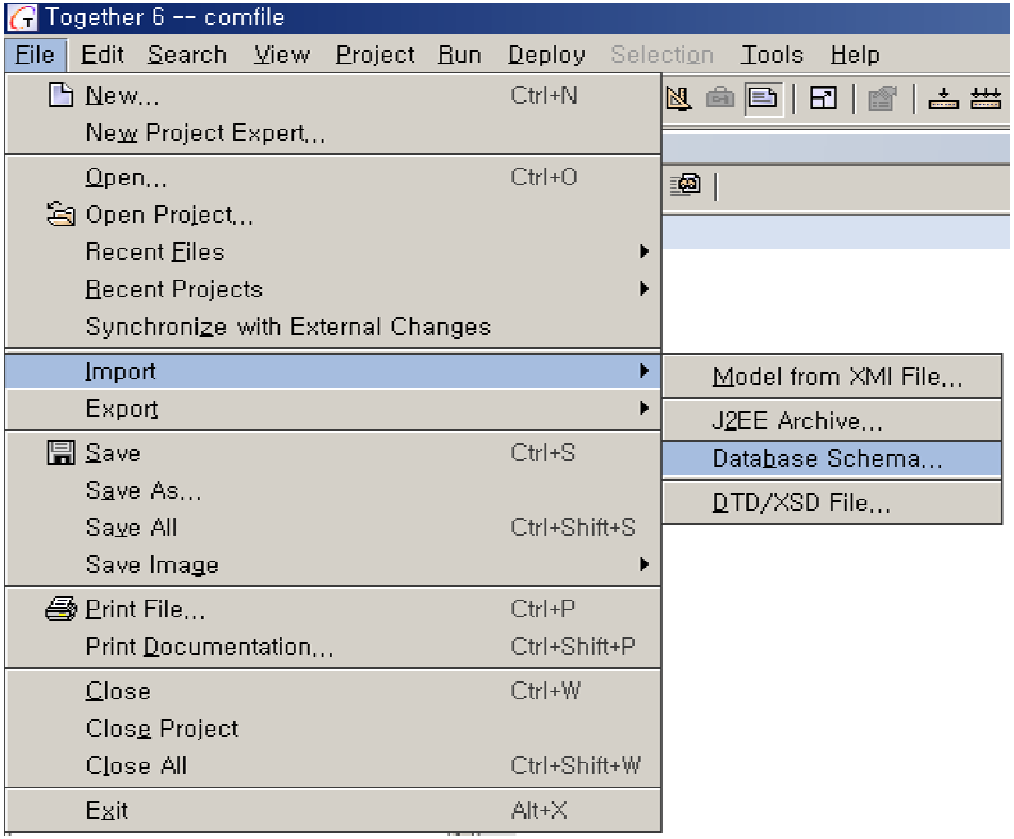


OOC
Open Objected Component

Borland®

Together DBMS 작업

Database Schema Import Step2

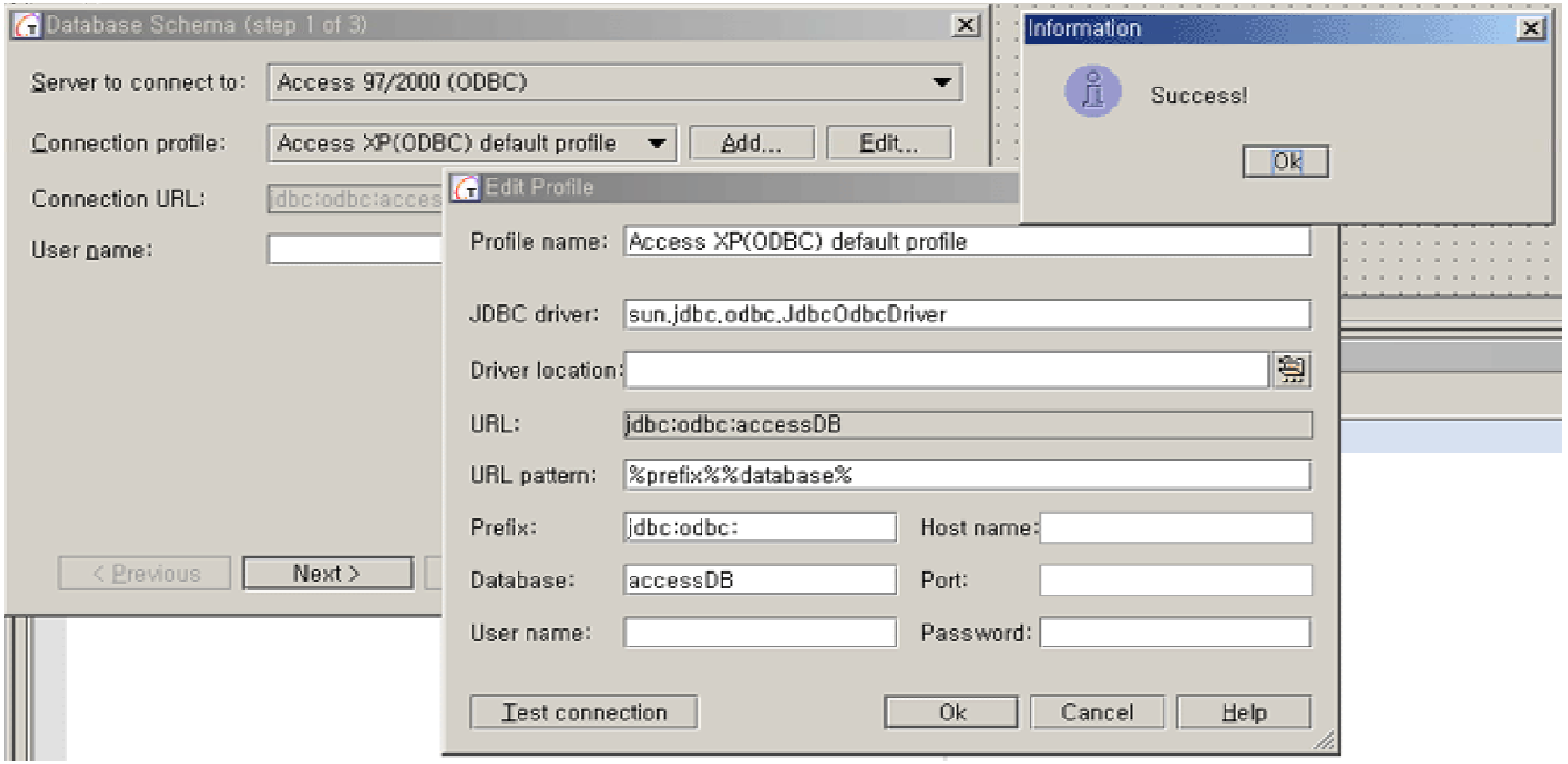


OOC
Open Objected Component

Borland®

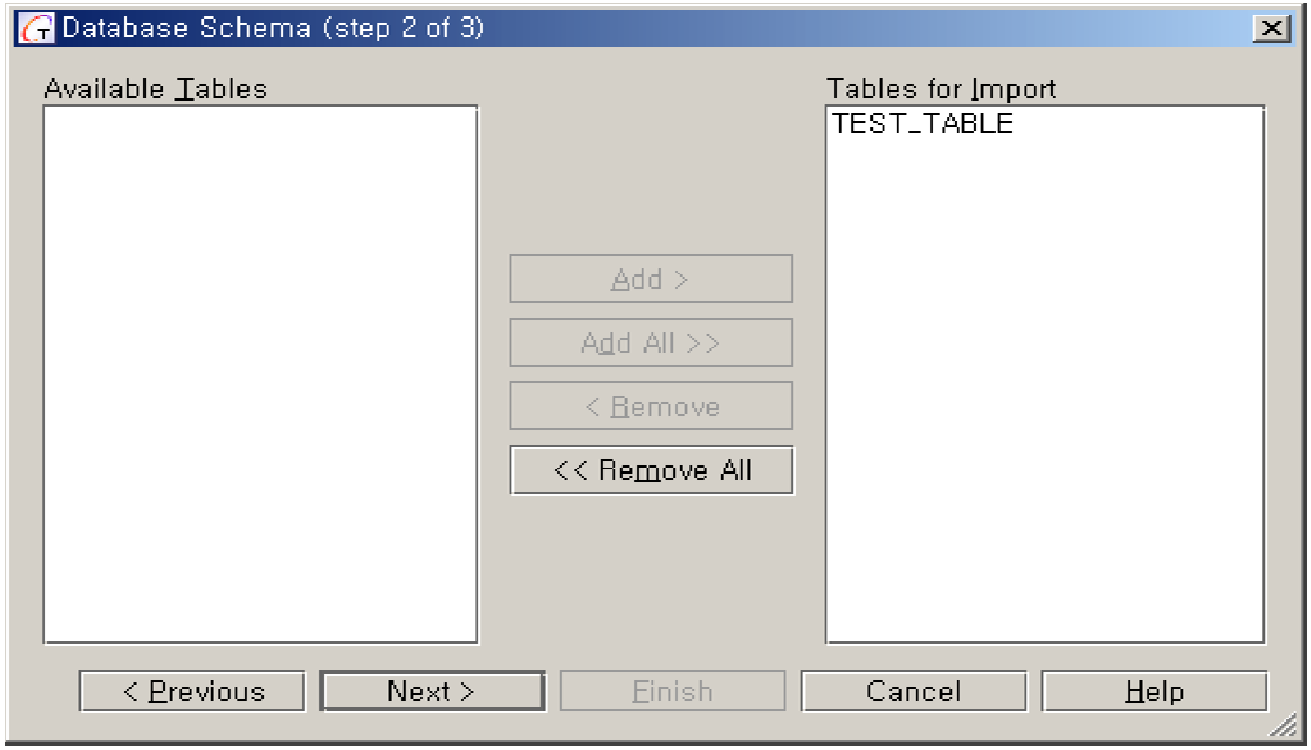
Together DBMS 작업

Database Schema Import Step3



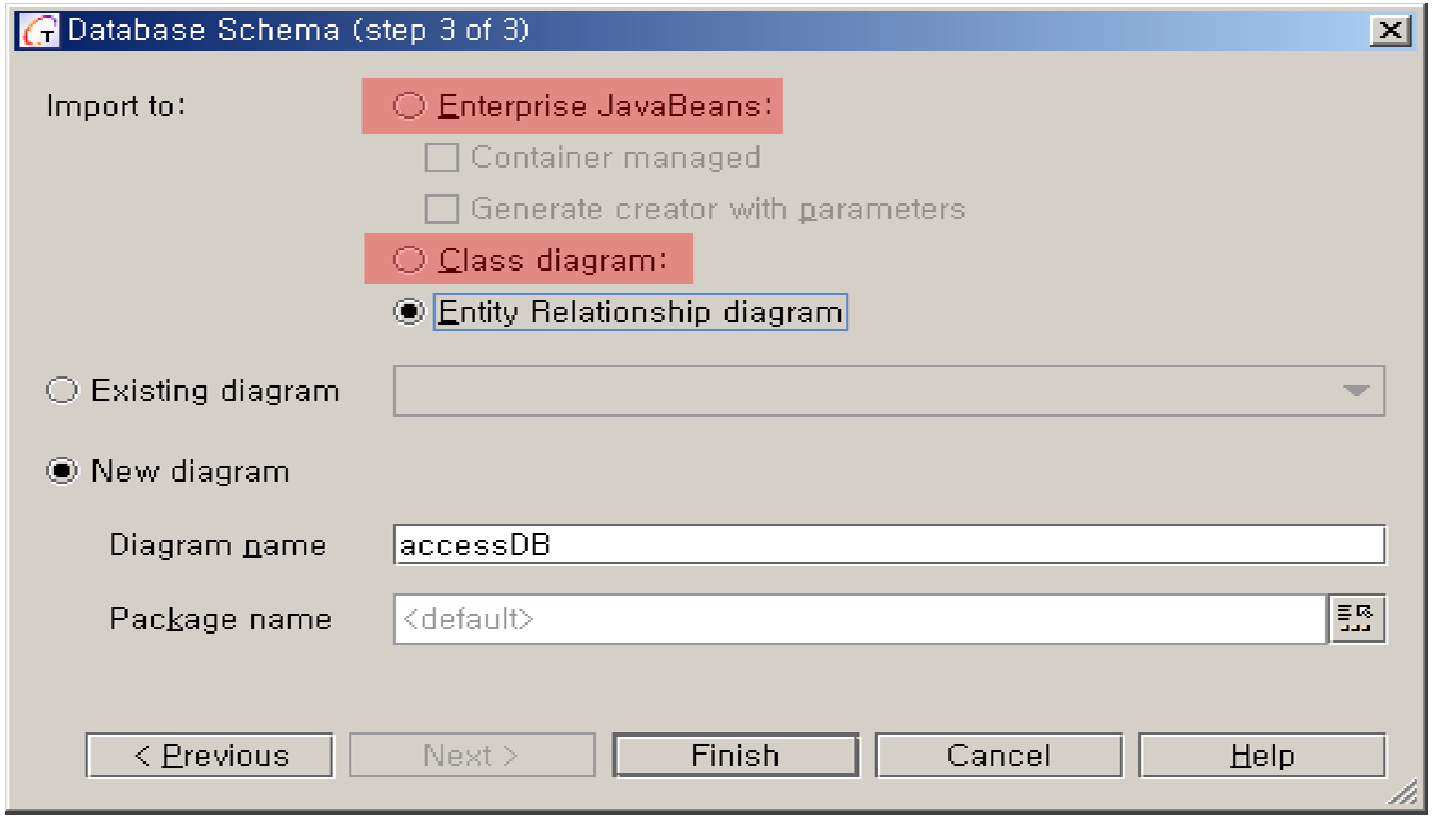
Together DBMS 작업

Database Schema Import Step4



Together DBMS 작업

Database Schema Import Step5



Together Technologies 일반적인 활용

Borland®

Together Technologies 활용

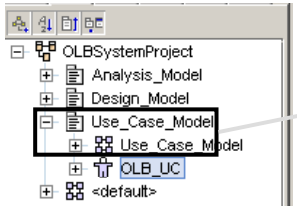
- 모델링 도구로써
- 산출물 생성 도구로써
- 품질 관리 도구로써
- 유지 보수 도구로써
- 투게더 활용을 위한 투게더 커스터마이징

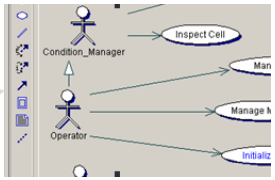
모델링 도구로써

활용 1

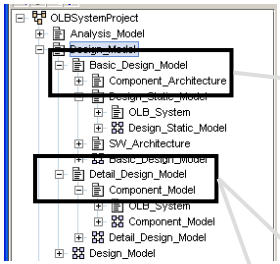
모델링 도구

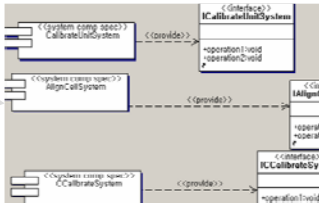
요구 분석/분석/설계 모델링 도구(UML 도구)로써 활용
컴포넌트 모델링 도구로써 활용



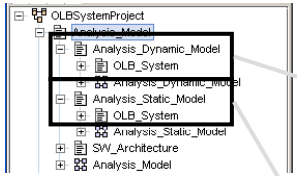


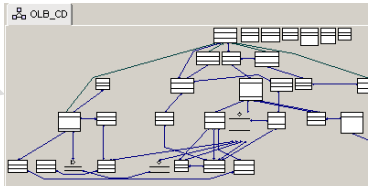
유즈케이스 모델



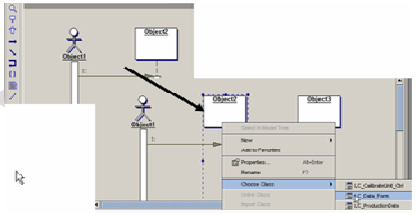


컴포넌트 모델링

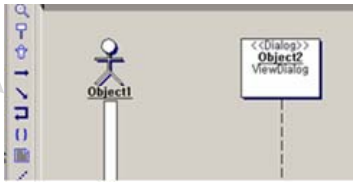




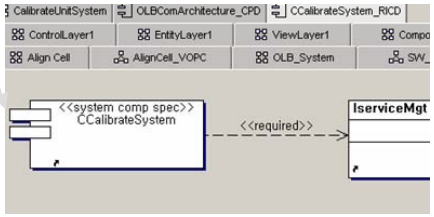
정적모델



동적모델



상세 설계 모델



OOC
Open Objected Component

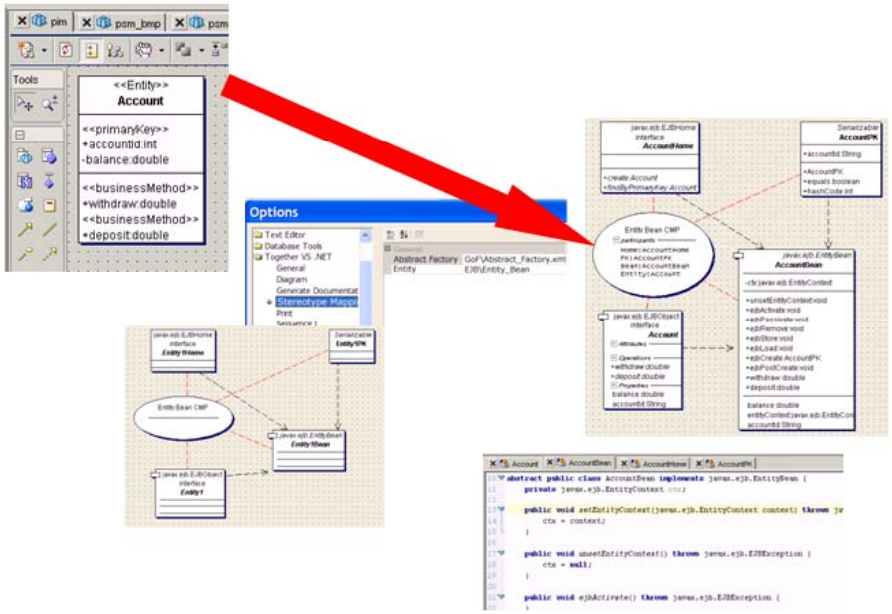
Borland®

MDA 도구로써

활용 2

MDA 도구

MDA 도구로써 활용을 할 수있다.

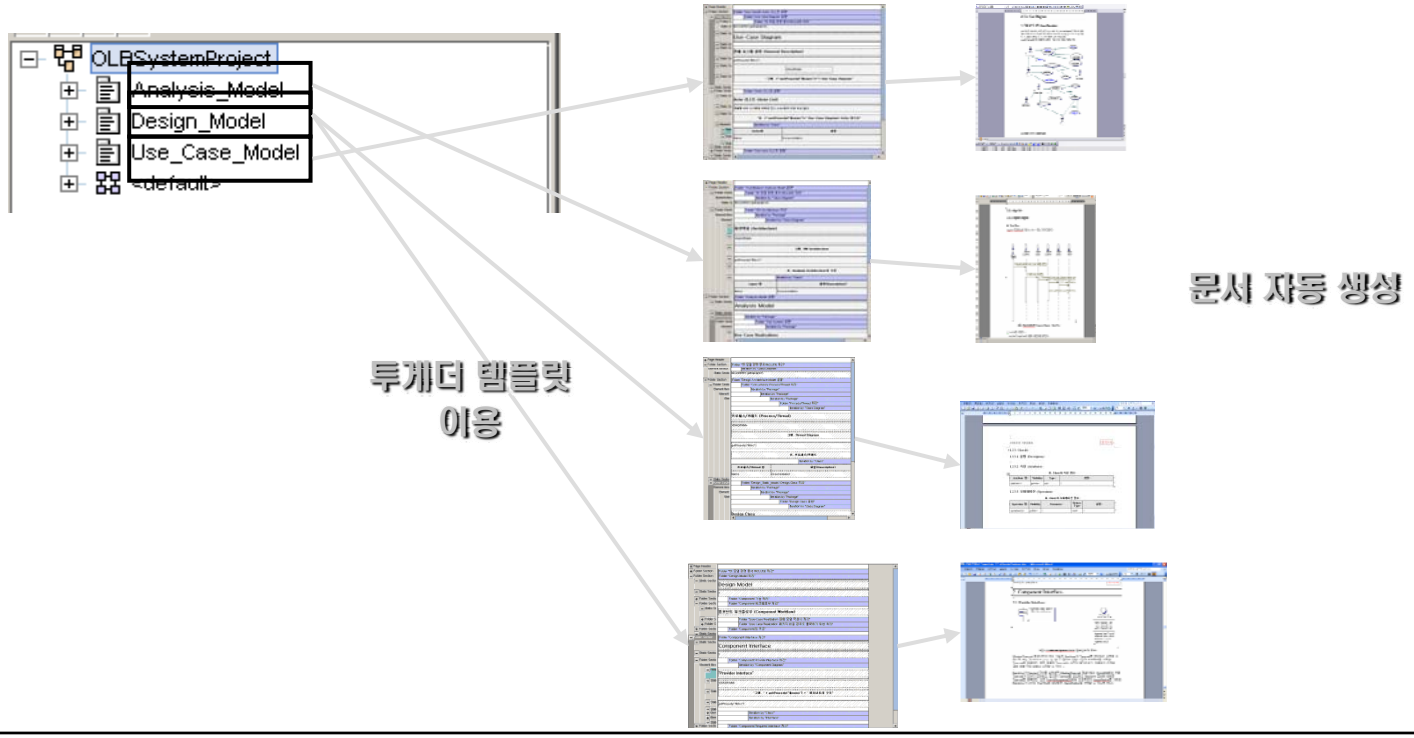


산출물 생성 도구로써

활용 3

산출물 생성 도구

개발프로세스에 맞게 각 단계의 산출물들을 자동으로 생성해 주는데 활용.



OOC
Open Objected Component

Borland®

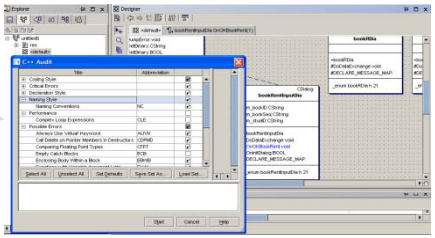
품질 관리 도구로써

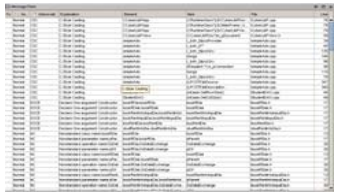
활용 4

품질 도구

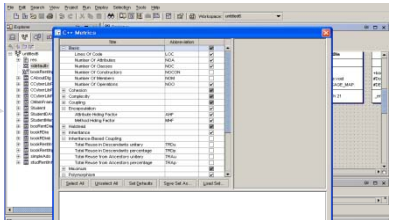
Audit, Metrics를 가지고 소스 및 모델 검증하는데 사용

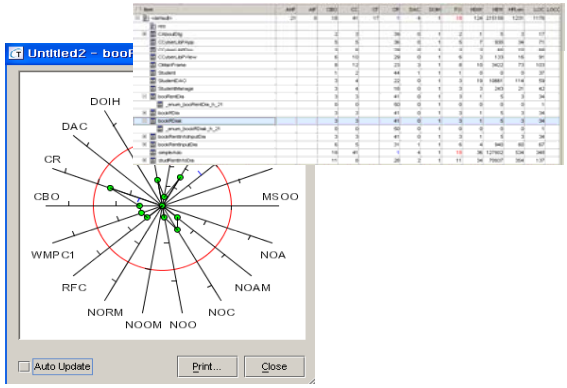
소스





모델





OOC
Open Objected Component

Borland®

유지보수 도구로써

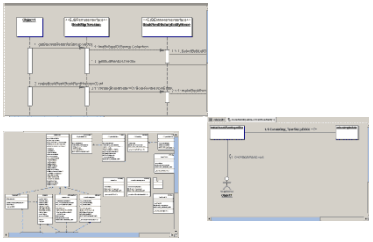
활용 5

유지보수 도구

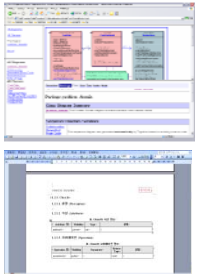
Together의 강력한 reverse engineering기능으로 유지 보수 시 변경된 소스를 모델링에 반영을 하여 새롭게 산출물을 만드는데 사용한다

소스 변경

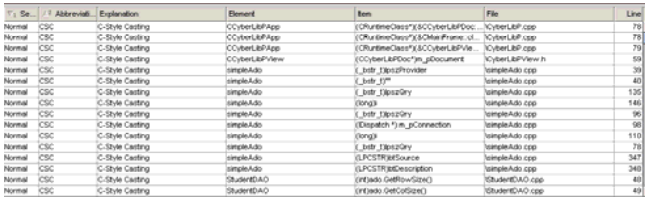
Revrse

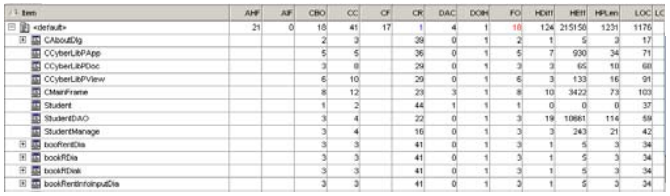


산출물 재 생성



Audit, Metrics로 유지 보수에 필요한 데이터 축출





OOC
Open Objected Component

Borland®

유지보수 중요성

- 소프트웨어 예산 중 유지보수의 비중이 증가
  - 1970년대 : 35 ~ 40%, 1980년대 : 40 ~ 60%, 1990년대 : 70% 이상
- 소프트웨어 종사자의 대부분이 신규 프로젝트 보다 기존의 소프트웨어 개선에 많이 투입

연 대	프로그래머의 수 (단위: 1,000명)		
	신규 프로젝트	유지보수 작업	합계
1950	0.09	0.01	0.1
1960	8.5	1.5	10
1970	65	35	100
1980	1,200	800	2,000
1990	3,000	4,000	7,000
2000	4,000	6,000	10,000

유지보수 문제점

- 소프트웨어에 대한 변경이 수시로 일어나며 이를 문서에 반영하지 않는 경우 이를 추적하는 것은 거의 불가능하다.
- 다른 사람이 작성한 프로그램을 이해하는 일은 쉽지않다. 문서화되어 있지 않거나 주석도 달지 않았다면 문제는 매우 심각하다.
- 소프트웨어가 변경을 가정하여 설계되는 경우가 드물다. 그러므로 변경(소프트웨어 변경, 개발 모듈 변경 등) 시 바로 반영될 수 있는 프로세스가 상당히 중요하다.
- 관리적인 측면에서 유지보수를 담당하는 프로그래머에게 동기부여를 하지 못한다.
- 유지보수를 위한 적극적인 도구를 사용하지 못한다.
- 프로그램 이해를 위하여 테스트나 디버깅 도구를 사용하는데 그친다.