

# **소프트웨어 공학 (Software Engineering)**

## **상세 설계 (Detailed Design)**



## In this chapter ... (1/2)

상세 설계(Detailed Design)

- 디자인 패턴(설계 패턴)이란 무엇이며 어떤 것이 있는가?
- 클래스 인터페이스와 내부를 설계하는 방법은?
- 사용자 인터페이스를 설계하는 원리와 방법은?
- 영구적인 데이터를 저장하는 방법은?





# In this chapter ... (2/2)

상세 설계(Detailed Design)



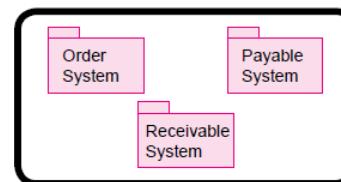
## 상세 설계 목적

- 설계 모델을 작동하는 소프트웨어로 변환
- 아키텍처-모듈 사이의 추상 수준의 갭을 줄이기 위함



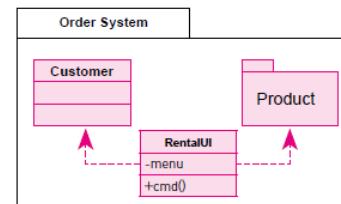
## 패러다임에 따른 작업

- 구조적 방법:  
프로시저 안의 로직  
(알고리즘) 설계
- 객체지향 방법:  
클래스 안의 메소드 설계



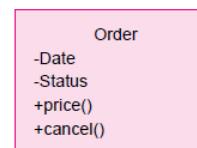
아키텍처 설계

서브시스템이나 패키지 수준  
– 아키텍처 스타일이 적용되는 수준



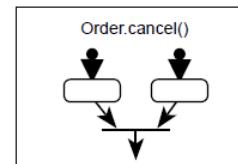
서브시스템 설계

패키지 내부의 클래스  
– 설계 패턴이 적용되는 수준



클래스 설계

클래스 내부의 데이터와 메소드 수준



메소드 설계

메소드 내부 알고리즘 설계



# In this chapter ...

상세 설계(Detailed Design)



7.1 디자인 패턴



7.2 클래스 설계



7.3 사용자 인터페이스 설계



7.4 데이터 설계



## 디자인 패턴이란?

- 소프트웨어 설계에서 자주 발생하는 문제에 대한 일반적이고 반복적인 해결책
- 여러가지 상황에 적용될 수 있는 일종의 템플릿

## 디자인 패턴 구성 요소

- 패턴의 이름과 구분
- 문제 및 배경 – 패턴이 사용되는 분야 또는 배경
- 솔루션 – 패턴을 이루는 요소들, 관계, 협동과정
- 사례 – 적용 사례
- 결과 – 패턴의 이점, 영향
- 샘플 코드 – 예제 코드

 패턴의 분류 (Gamma가 제안한 23개 패턴)

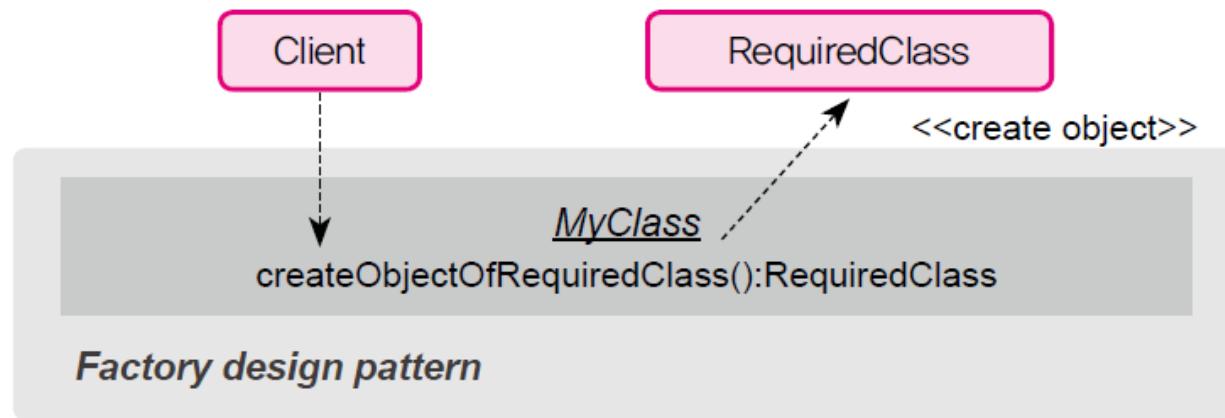
		목적에 의한 분류		
		생성유형	구조적	행위적
범위	클래스	팩토리 메소드	어댑터(클래스)	인터프리터 템플릿 메소드
	객체	추상 팩토리 싱글톤 프로토타입 빌더	어댑터(객체) 브리지 컴포지트 데코레이터 퍼싸드 플라이웨이트 프록시	책임 체인 커맨드 반복자 중재자 메멘토 옵서버 상태 전략 비지터

→ 자주 사용되는 9가지 패턴에 대해서 살펴 봄



- 객체 생성을 위한 인터페이스(← 팩토리 메소드)의 정의
- 위임(delegation) 형태

- RequiredClass의 생성을 팩토리 메소드를 가진 MyClass에게 위임
- 팩토리 메소드는 createObjectOfrequiredClass()

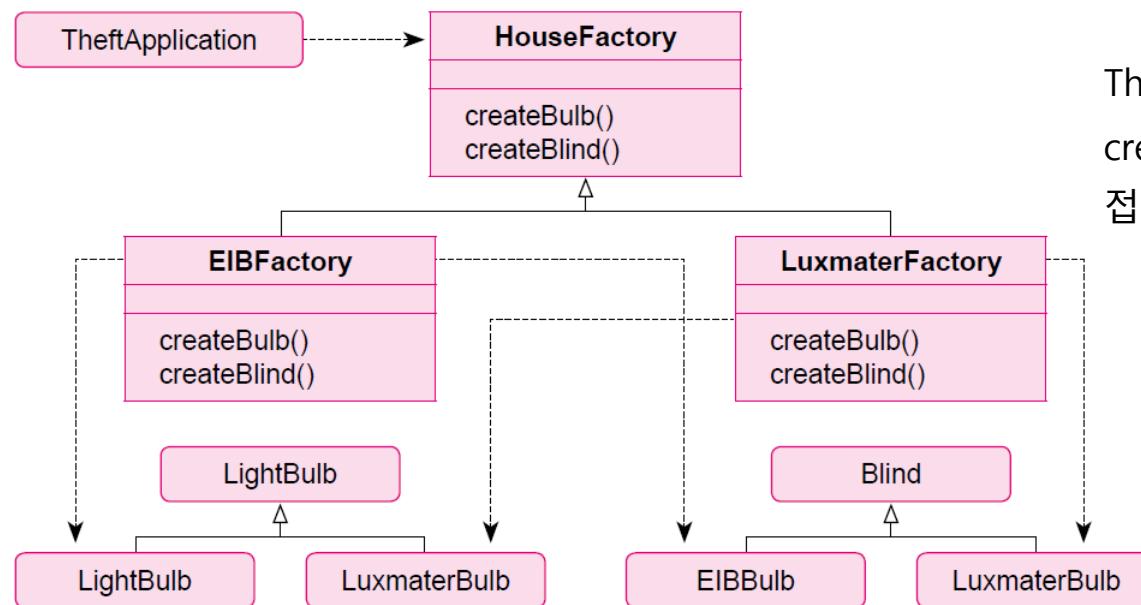


- 사용 이유: 베이스 클래스에 속하는 객체 중 하나가 필요한데, 그 아래에 있는 자식 객체 중 어떤 것이 필요한지 실행 시간까지 알 수 없을 때



## 클래스의 집합의 종류를 지정하여 관련된 객체의 집합을 생성할 수 있게 하는 패턴

- 관련된 객체의 패밀리를 생성하여 추상적인 클래스 패턴을 정의
- 추상 팩토리 패턴이 필요한 경우: 가전제품을 생산할 때, 여러 업체가 관련되면 호환성이 취약하다. 여러 생산자가 만든 장치들을 혼합하여 사용할 때 취합이 어려우므로, 추상 팩토리 패턴을 사용하여 해결한다.



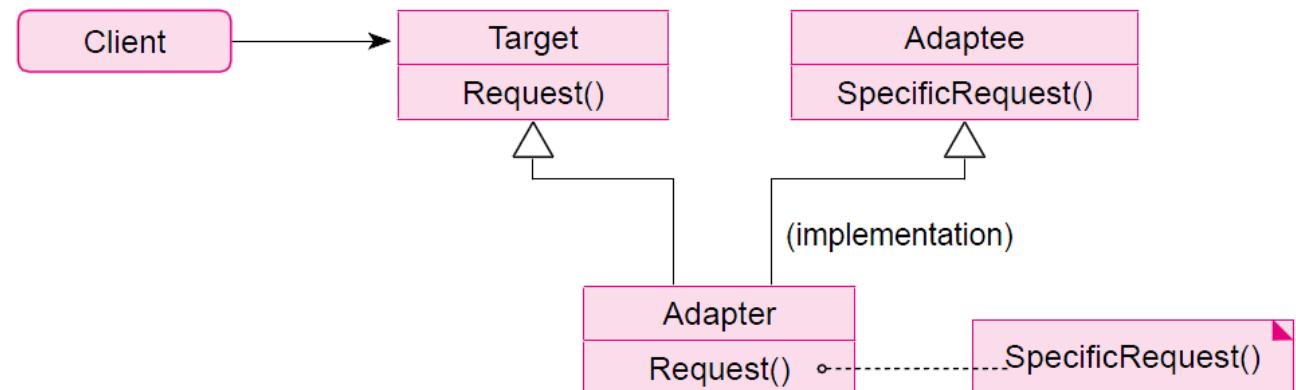
TheftApplication은 HouseFactory의 createBulb()와 createBlind()만을 접근한다.



# 어댑터 패턴(Adapter Pattern)

상세 설계(Detailed Design)

- 이미 개발된 클래스, 즉 레거시 시스템의 인터페이스를 다른 클래스의 요구에 맞게 인터페이스를 변환해주는 것
- 이미 만들어져 있는 클래스를 사용하고 싶지만 인터페이스가 원하는 방식과 일치하지 않을 때 사용
- 어댑터 패턴 적용은 **상속 이용** 방법과 **위임 이용** 방법이 있음
- 상속을 이용한 어댑터 패턴의 모형 사례
  - Client가 Target의 Request()를 호출하면 Adapter를 통해 Adaptee의 SpecificRequest()가 호출되도록 한다.



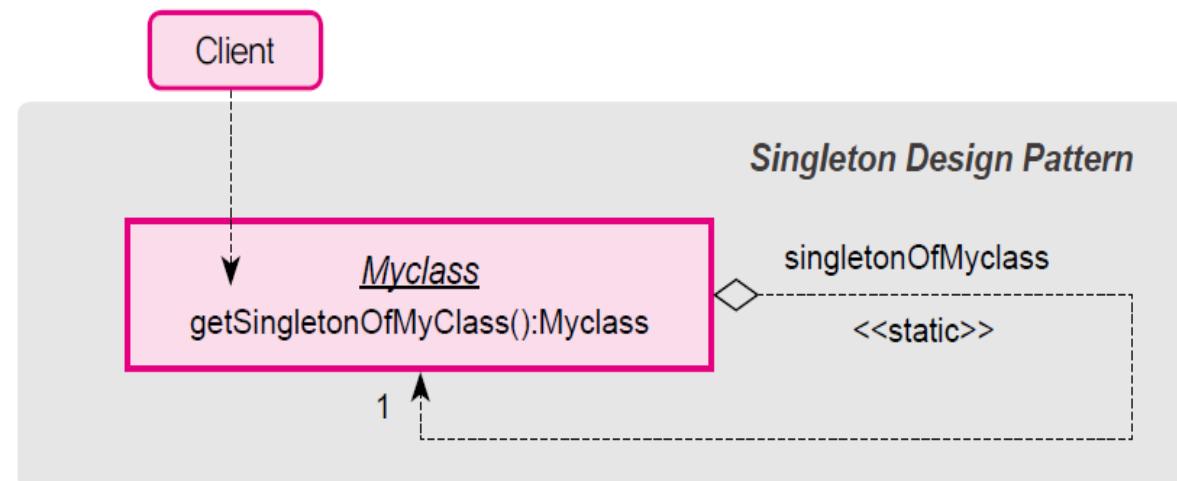


## ① 시스템에서 단 하나의 인스턴스만 갖도록 할 필요가 있는 경우 사용

- 연결된 프린터는 많아도 프린터 스플러는 하나임
- 디스크는 많아도 파일 시스템 관리자는 하나임

## ② 패턴 사용 요령

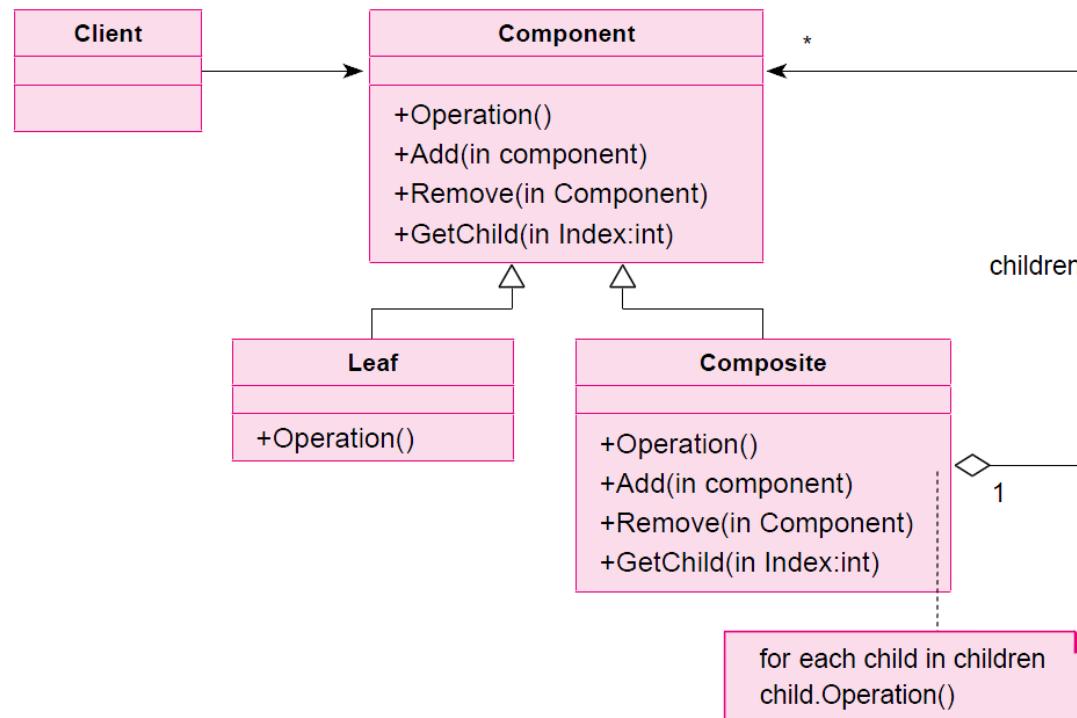
- 클래스의 유일한 인스턴스를 넣을 위치를 지정
- 생성자 작성 시, static, final 등을 사용하여 지정  
(Java specific한 자세한 내용은 생략)





## 객체 집합 속에 또 다시 객체 집합을 갖는 경우 패턴 사용

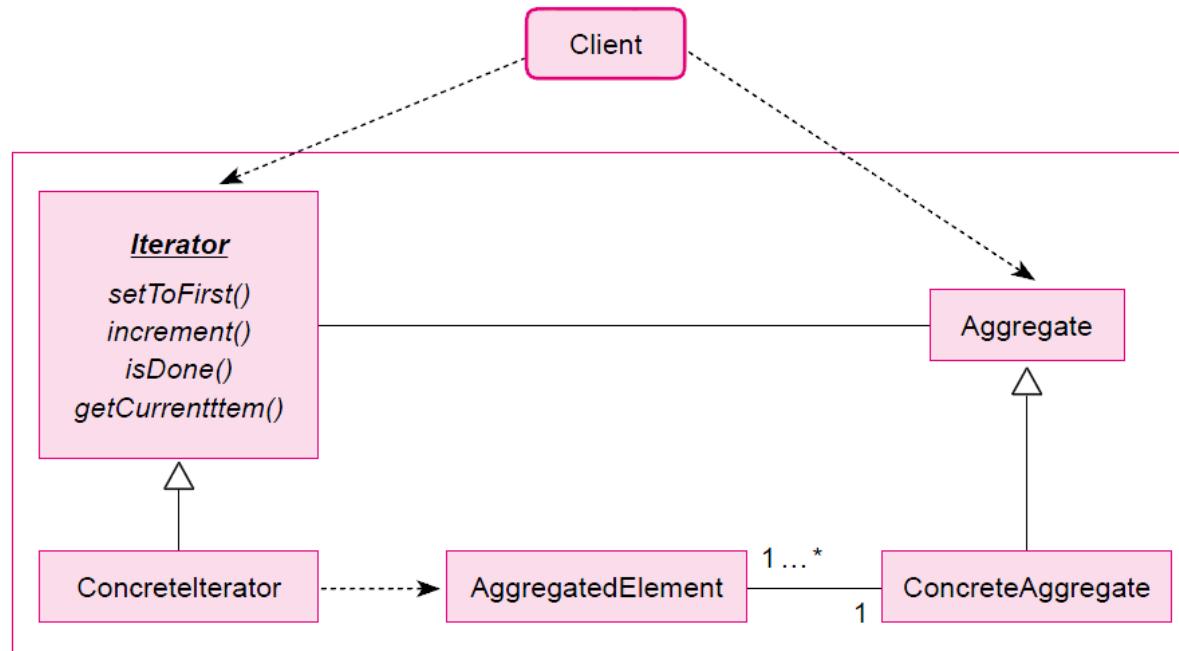
- 집합 속에 포함될 객체와 집합을 가지고 있는 객체, 이들 모두가 자기 자신과 동일한 타입(메소드와 데이터)의 객체 리스트를 가질 수 있도록 도와준다.
- 기본 클래스와 이를 포함하는 컨테이너 클래스를 구분하지 않고 처리하는 재귀적 합성을 이용할 수 있다.





▣ 시스템의 유사한 객체를 다룰 때, 동일한 인터페이스를 이용하여 반복적으로 접근할 수 있도록 만드는 패턴

- 처리하려는 자료구조가 다른 형태로 바뀌더라도 클라이언트가 영향을 받지 않음
- 자료의 자세한 구조는 모르더라도 반복자가 제공하는 인터페이스를 이용하여 검색하거나 하나씩 처리할 수 있음





## 옵서버 패턴

- 1대다 객체 의존관계를 정의함
- 한 객체의 상태가 바뀌면 의존관계의 다른 객체들에게 자동 통지하고 변경

## 상태 패턴

- 이벤트 의존 애플리케이션에 적합
- 시스템이 동작하면서 발생되는 이벤트에 따라 변경이 이루어질 수 있도록 설계

## 파싸드 패턴

- 서브시스템 내부가 복잡하여 클라이언트 코드가 사용하기 힘들 때 사용
- 간단한 인터페이스만 알아도 서브시스템의 주요 기능을 사용
- 복잡한 것을 단순하게 보여주며, 내부 시스템을 몰라도 사용 가능



## In this chapter ...

상세 설계(Detailed Design)

 7.1 디자인 패턴

 7.2 클래스 설계

 7.3 사용자 인터페이스 설계

 7.4 데이터 설계



- 분석 단계에서 아직 확정되지 않은 클래스 내부 부분 중 구현에 필요한 중요한 사항을 결정하는 작업
  - 클래스의 서비스 인터페이스에 대한 정확한 정의, 메소드 내부의 로직 등
- 객체의 상태 변화와 오퍼레이션의 관계를 상세히 설계해야 함
- 클래스가 가지는 속성 값에 따라 오퍼레이션 구현이 달라짐
  - 객체의 상태 변화 모델링 필수



## 관점이 다른 개발자들이 클래스 명세의 어떤 부분이 관심이 있는가?

- 클래스 구현: 실제 설계로부터 클래스를 구현하려는 개발자
  - 클래스 사용: 구현된 클래스를 이용하여 다른 클래스를 개발하려는 개발자
  - 클래스 확장: 구현된 클래스를 확장하여 다른 클래스로 만들고자 하는 개발자
- 관점에 따라 관심이 다르므로, 클래스 인터페이스가 중요함

## 서브시스템의 서비스 인터페이스를 통한 **API(Application Program Interface)**라 부름



## ④ 클래스에 대한 여러 가정을 공유하도록 명세한 것을 **협약에 의한 설계** (**Design by Contract**)라 함

- 소프트웨어 컴포넌트에 대한 정확한 인터페이스 명세를 위하여 선행조건, 결과조건, 불변조건을 나타내는 설계 방법

## ⑤ 협약에 의한 설계의 세 가지 타입

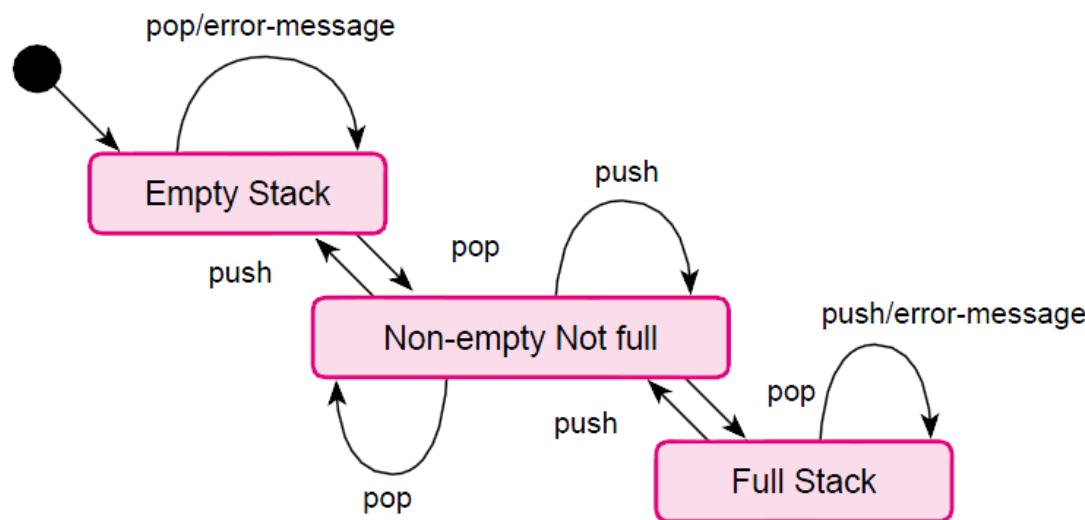
- 선행조건(precondition): 오퍼레이션이 호출되기 전에 참이 되어야 할 조건
- 결과조건(postcondition): 오퍼레이션이 수행된 후 만족하여야 하는 조건
- 불변조건(invariant): 클래스 내부가 실행되는 동안 항상 만족하여야 하는 조건  
(예: 리스트에 있는 노드가 항상 오름차순으로 되어야 함)



▣ 객체가 가지는 상태와 여러 오퍼레이션 관계를 오토마타(상태변환 다이어그램)으로 나타냄

- 클래스에 정의하는 여러 오퍼레이션의 동작 효과를 이해하는데 큰 도움을 줌
- 객체가 논리적으로 가질 수 있는 상태를 나타냄

▣ 예제: 스택의 상태 다이어그램





## 프레임워크

- 재사용 가능한 부분적인 응용 프로그램
- 클래스 라이브러리와 달리, 자료처리, 이동통신, 실시간 항공 등 특수한 기술이나 응용 도메인을 목표로 함

## 재구성

- 프레임워크 클래스를 솔루션에 맞게 적용한 후, 목표 시스템에 맞도록 재구성
- 연관관계 구현, 상속 재검토 등을 통해 재사용도를 높임

## 최적화

- 재사용에 의한 비효율 제거를 위해 응용을 고려한 최적화 수행
- 설계 목표(반응시간, 실행시간, 기억공간 최소화 등) 달성을 위한 최적화



## In this chapter ...

상세 설계(Detailed Design)

 7.1 디자인 패턴

 7.2 클래스 설계

 7.3 사용자 인터페이스 설계

 7.4 데이터 설계



## 사용자의 입장을 더욱 중시함

- 모든 소프트웨어는 사용자의 만족이 큰 목적이 됨
- 소프트웨어 시스템에 있어서 사용자의 만족

## 사용자 인터페이스의 평가 기준

- 배우기 쉬워야
- 반응 속도가 빨라야
- 사용 중 오류가 발생하지 않아야
- 다수의 사용자가 만족할 수 있어야
- 사용법이 유지되어야 (한번 써보면, 다음에 더 쉽고, 그 다음에 더 쉽고...)

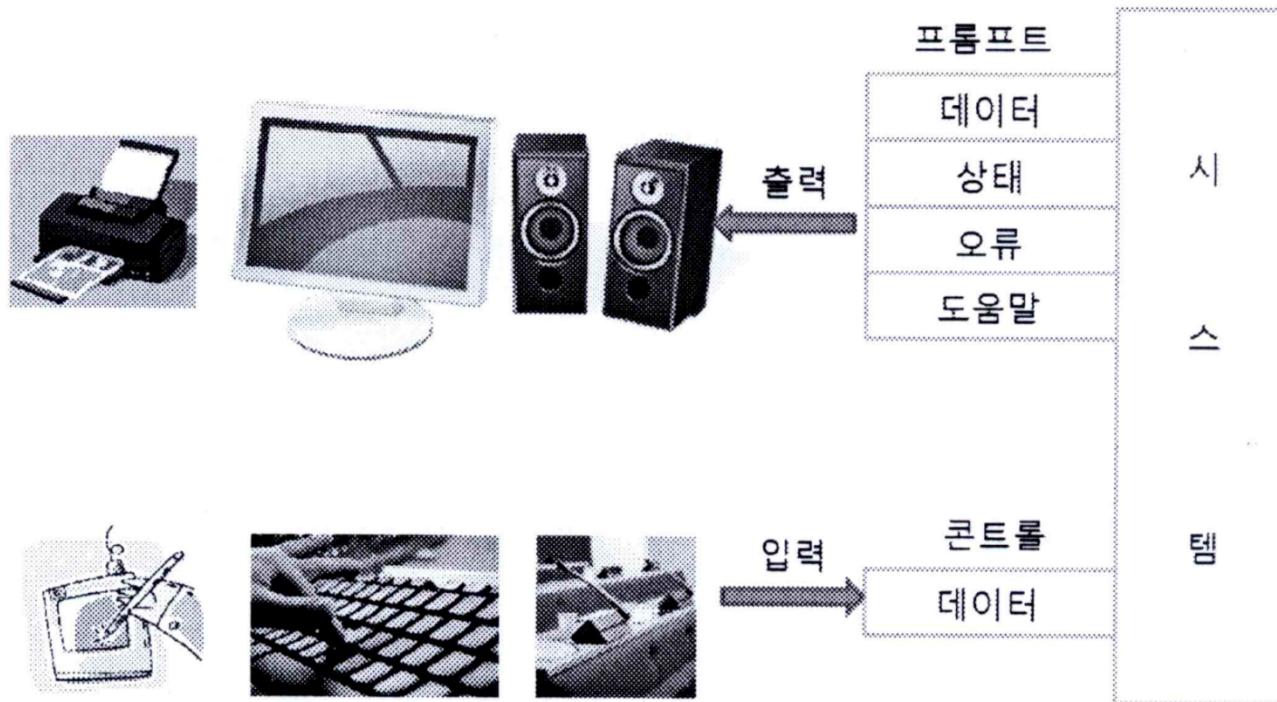


# 사용자 인터페이스 유형 (1/2)

상세 설계(Detailed Design)

## ④ 사람과 컴퓨터 사이의 대화를 체계적으로 표현

- 사람: 컴퓨터 시스템에 명령을 내리고 자료를 전달
- 컴퓨터: 사람에게 처리된 정보나 처리 상태, 도움말 등을 전달





- ④ **그래픽 사용자 인터페이스**(Graphic User Interface):  
직접 조작형 인터페이스
- ⑤ **웹 인터페이스**: 인터넷과 웹 브라우저를 통하여 페이지를 열람하고 조작하는 인터페이스
- ⑥ **명령어 인터페이스**: 정해진 명령 문자열을 입력하여 시스템을 조작하는 인터페이스
- ⑦ **텍스트 사용자 인터페이스**: 자연어에 가까운 문장을 입력하여 시스템을 조작하는 인터페이스



## 좋은 사용자 인터페이스

- 직관적으로 사용
- 사용하는 도중에 오류가 적음

## 사용자와 시스템 간의 대화설계에서 따라야 할 큰 원칙

- 일관성
- 적절한 사용자 지원
- 적당한 피드백
- 최소의 사용자 입력

## 사용자를 배려하는 사용자 중심의 설계



## ④ 사용자 인터페이스 작성 시의 기본 가이드라인

- 배우기 쉽고 쓰기 쉬운 인터페이스를 만들라.
- 효율성을 높이는 기능을 제공하라.
- 사용자가 도움을 받거나 오류를 수정하기 쉽게 하라.
- 입력 자료의 오류를 최소화하라.
- 사용자에게 피드백을 제공하라.
- 끌리는 레이아웃과 설계를 창조하라.
- 익숙한 용어와 이미지를 사용하라.





## 화면 설계를 위한 원리

- 화면을 설계하는 동안 사용자의 특성을 기억한다.
- 논리적으로 관련 있는 항목들은 빈 줄 등으로 구별하기 쉽게 한다.
- 정보를 조직적으로 표현하기 위하여 다양한 정렬 방식을 사용한다.
- 다중 화면의 경우 화면 사이의 일관성이 중요하다.
- 여러 가지 다른 배치로 시험한다.



## UI 컨트롤의 종류

- 메뉴 바
- 다이얼로그 박스, 텍스트 박스
- 명령 버튼, 토글 버튼, 리스트 버튼, 라디오 버튼
- 드롭 다운 리스트 박스, 체크 박스



# 화면 설계 (2/3)

상세 설계(Detailed Design)

The screenshot shows a portion of the E\*TRADE website with several UI elements highlighted by pink arrows and Korean labels:

- 메뉴바** (Menu Bar): Points to the top navigation bar with links like QUOTES, GO, SITE MAP, HELP, and LOG ON.
- 다이얼로그 박스** (Dialog Box): Points to a modal window titled "An Inane Question" asking "Would you like green eggs and ham?" with Yes and No buttons.
- 텍스트 박스** (Text Box): Points to a text input field labeled "Years: 30".
- 체크 박스** (Check Box): Points to a checkbox labeled "Check 1" which is checked.
- 라디오 버튼** (Radio Button): Points to a radio button labeled "Radio 2" which is selected.
- 명령 버튼** (Command Button): Points to the "OK" button at the bottom right of the dialog box.
- 리스트 박스** (List Box): Points to a scrollable list box containing months: January, February, March, April.
- 드롭다운 리스트 박스** (Drop-down List Box): Points to a dropdown menu listing animals: Pig, Bird, Cat, Dog, Rabbit, Pig, with "Cat" currently selected.
- Blah blah**: Points to the placeholder text "Blah blah" in a form field.

Below the main screenshot, two smaller windows illustrate specific UI components:

- Radio Buttons**: A window showing three radio buttons: Red (selected), Blue, and Green.
- Check Boxes**: A window showing three checkboxes: Red (checked), Yellow (unchecked), and Blue (checked). Below it is a label "This is the label text".



## 쉽게 배울 수 있고 사용할 수 있는 자료 입력 화면 가이드라인

- 항목의 입력이 끝났음을 알리기 위한 키를 반드시 정한다.
- 콤보 박스를 사용하여 입력하게 하면 입력 오류를 줄일 수 있다.

## 입력 양식 (양식 채움)

- 처리할 자료를 요청하고 모으는데 사용하는 양식

이름	고혜원		
주민번호	610720-*****		
닉네임	<input type="text"/>	<input checked="" type="checkbox"/> 닉네임 적용	<a href="#">닉네임 중복확인 &gt;</a>
아이디	<input type="text"/>	<a href="#">아이디 중복확인 &gt;</a>	4~12로 해야하며 한글/특수문자 입력불가입니다.
비밀번호	<input type="text"/>	영문 및 영문 숫자 조합 6~16자리로 입력하실 수 있습니다.	
비밀번호확인	<input type="text"/>	패스워드 확인을 위해 다시한번 입력해주세요.	
핸드폰번호	010	<input type="button" value="▼"/>	<input type="checkbox"/> SMS 수신동의
주소	<input type="text"/> - <input type="text"/>	<a href="#">▶ 우편번호찾기</a>	
이메일	<input type="text"/> @ <input type="text"/>	<input type="button" value="선택하세요"/>	<input checked="" type="checkbox"/> 기획전, 세일, 이벤트, 쿠폰관련 메일 수신동의



▣ 출력물은 다양한 종류가 있고 다양한 기술이 사용

▣ 출력물의 대부분은 인쇄된 리포트

- 리포트는 매력적이고 전문적이어야 하며 무엇보다 읽기 쉬워야 한다.
- 모든 리포트는 머리말과 꼬리말이 있어야 한다.
- 항목은 논리적인 순서로 표현되고 그룹핑 되어야 한다

초과 근무 보고서 마감 날짜: 2009. 6. 22						보고서 머리말
점포명	직원 성명	직위	정규근무시간	초과근무시간	총근무시간	페이지 머리말
목동점	김현영	부팀장	40.0	1.5	41.5	
			-----	-----	-----	
양재점	최지현	부팀장	중간 합 40.0	1.5	41.5	
양재점	홍길동	직원	40.0	5.5	45.5	그룹 머리말
			32.7	0.0	32.7	
			-----	-----	-----	
일산점	박찬종	직원	중간 합 72.7	5.5	77.7	
일산점	변창일	팀장	40.0	12.0	52.0	
			40.0	10.0	50.0	
			-----	-----	-----	
			중간 합 80.0	22.0	102.0	보고서 꼬리말
			총합 192.7	28.0	220.7	페이지 꼬리말
						1 쪽



## In this chapter ...

상세 설계(Detailed Design)

7.1 디자인 패턴

7.2 클래스 설계

7.3 사용자 인터페이스 설계

7.4 데이터 설계

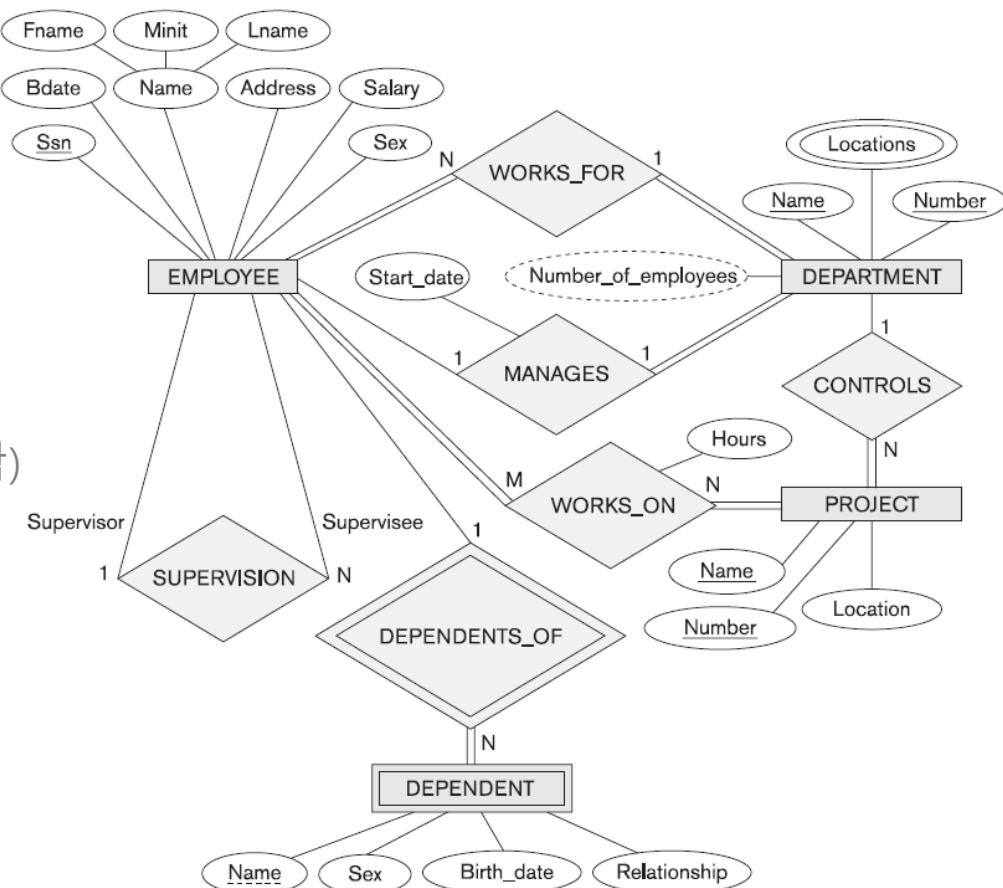


## ④ 엔티티 관계 다이어그램(Entity-Relationship Diagram)

- 관계형 데이터베이스를 위한 주요 모델링 도구
- 실세계 데이터를 엔티티와 관계로 모델링

## ER 다이어그램

- 테이블보다는 엔티티를 강조
- 관계형 데이터베이스 모델링에  
가장 널리 사용되는 도구  
(→ 데이터베이스 시간에 자세히 공부함)





## 비디오 대여 시스템

### ERD를 작성

- 시스템의 엔티티를 파악하기 위하여 클래스 다이어그램을 리뷰
- 1:1, 1:M, M:N 관계를 결정



MEMBER(MEMBER-NUMBER, NAME, ADDRESS, CITY, ZIP, HOME-PHONE,  
WORK-PHONE, CREDIT-CARD-CODE, CREDIT-CARD-NUMBER, (VIDEO-ID,  
TITLE, DATE-RENTED, DATE-RETURNED))  
VIDEO(VIDEO-ID, TITLE)

### 최종 ERD로 부터 테이블(릴레이션)들을 도출하고 정규화 등 작업 진행



# 객체와 테이블 (1/2)

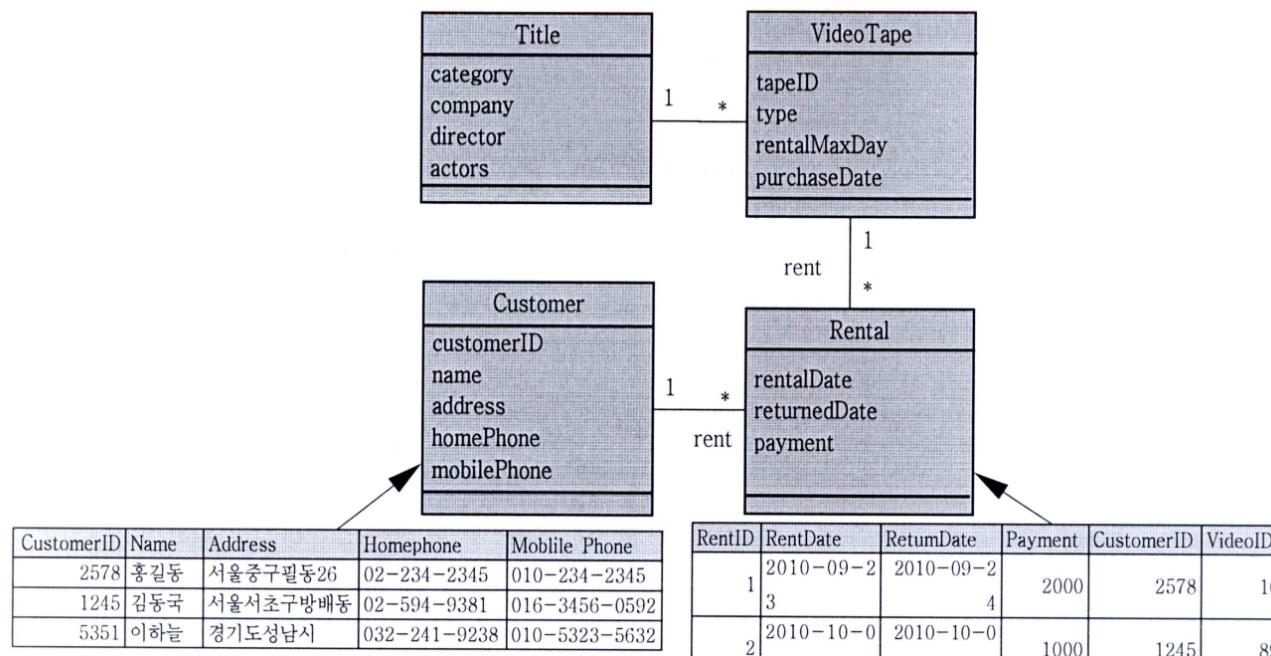
상세 설계(Detailed Design)

## 관계형 데이터베이스를 객체지향 프로그램에서 사용하는 경우

- 관계형 데이터베이스 자체를 설계
- 애플리케이션에서 DBMS와 어떻게 통신하는지 설정
- 매팅 되는 과정에 테이블 개념 및 정규화 규칙은 클래스와 관계에도 적용

## 객체의 속성은 테이블의 열(column)에 해당되는 테이블 속성으로 매팅

- 속성: 객체지향과 관계형 데이터베이스에서 공유하는 단어





## 데이터베이스의 테이블은 항상 기본키를 가져야 함

- 기본키가 될 수 있는 속성: null이 될 수 없는 값, 자주 변경되지 않아야 함

객체지향 요소	관계형 데이터베이스
클래스	테이블
속성	테이블 속성(열)
기본키	기본키
반복되는 속성	별도의 테이블
연관	기본키-외부키의 쌍
다중도	외부키
집합 관계	연관으로 변경 후 외부키
상속	서브클래스를 별도의 테이블로 하여 미상속 속성만 맵핑

→ 자세한 내용은 데이터베이스 교과목에서 다룸



# In this chapter ...

상세 설계(Detailed Design)

 7.1 디자인 패턴

 7.2 클래스 설계

 7.3 사용자 인터페이스 설계

 7.4 데이터 설계