8장. 객체지향설계

숙명여자대학교
SOOKMYUNG WOMEN'S UNIVERSITY

Contents

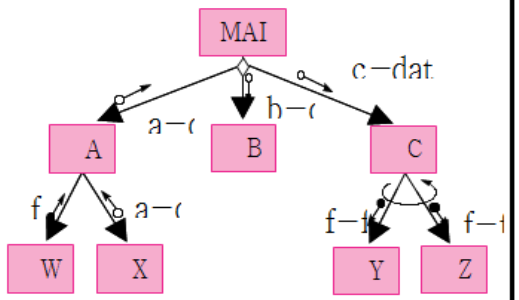
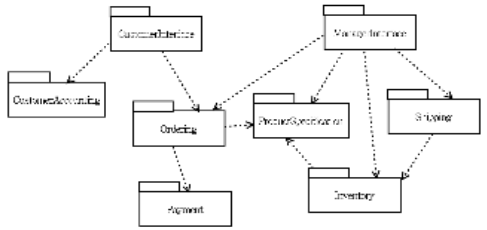
1. 객체지향설계 원리
2. 패키지다이어그램
3. 자료 구조 설계
4. 인터페이스 설계
5. 기술환경명세

상세설계 (1/2)

- 상세 설계 목적
  - 설계 모델을 작동하는 소프트웨어로 변환
  - 아키텍처와 모듈 사이의 추상 수준의 갭을 줄이기 위함
- 아키텍처 설계 vs. 모듈설계
  - 아키텍처 설계는 시스템 전체의 구조를 설계하는 개념적 모델링
  - 모듈 설계는 구체적인 컴포넌트 내부 설계

상세설계 (2/2)

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

	아키텍처 설계	모듈 설계
구조적		● 프로시저 내부의 로직 설계
객체지향		● 클래스 내부의 메소드 로직 ● 클래스의 상태 변화

1. 객체지향 설계 (1/3)

- 객체지향 분석과 설계는 순차적인 과정이 아님
  - 설계와 구현 사이의 공통 개념을 사용: 분석과 설계에서 도출된 부분이 바로 프로그램으로..
  - 반복적인 사이클로 완성
- 분석과 설계 작업의 명확한 경계가 없음

- 객체지향 분석(OOA)과 설계(OOD) 작업의 근본적인 목표는 시스템에 있어야 할 클래스와 그들의 관계를 찾는 것

1. 객체지향 설계 (2/3)

- 구조 모델링의 결과물인 클래스 다이어그램을 설계 모델로 진화시킴
  - 클래스를 구성하는 요소들에 대한 상세 정보와 메서드의 내부 로직을 설계
  - 패키지 다이어그램을 작성
  - 클래스 다이어그램을 이용하여 대응하는 UI클래스와 데이터베이스 테이블 생성
  - 각 패키지를 하드웨어 플랫폼에 할당하는 배치 다이어그램을 작성
  - SW를 구현하는 데 필요한 적용 기술 스펙 정의

1. 객체지향 설계 (3/3)

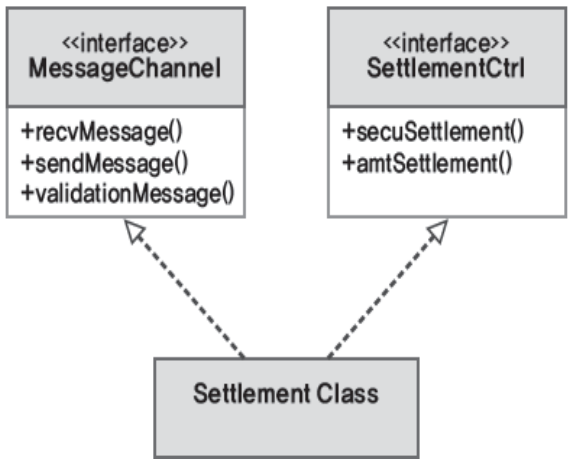
- 클래스 설계를 위한 5대 원칙 SOLID
  - SRP(Single Responsibility Principle)
  - OCP(The Open-Closed Principle)
  - LSP(Liskov Substitution Principle)
  - ISP(Interface Segregation Principle)
  - DIP(Dependency Inversion Principle)

1.1 SRP

- 단일 책임의 원칙(Single-Responsibility Principle)
  - 클래스에는 한 가지 종류의 책임만을 두어야 한다는 원칙
- SRP의 목적은 변화의 유연성 확보에 있음
  - SRP 원칙에 따라서 설계를 하게 되면 객체의 약한 결합(Loose coupling)과 강한 응집력(Tight Cohesion)을 실현할 수 있음

단일 책임의 원칙 위반 사례

```
Interface Settlement
{
    public void secuSettlement(Document doc); //증권에 대한 결제
    public void amtSettlement(Document doc); //대금에 대한 결제
    public void validationMessage(Document doc); //전문 검증
    public void recvMessage(Document doc); //전문 수신
    public void sendMessage(Document doc); //전문 송신
}
```



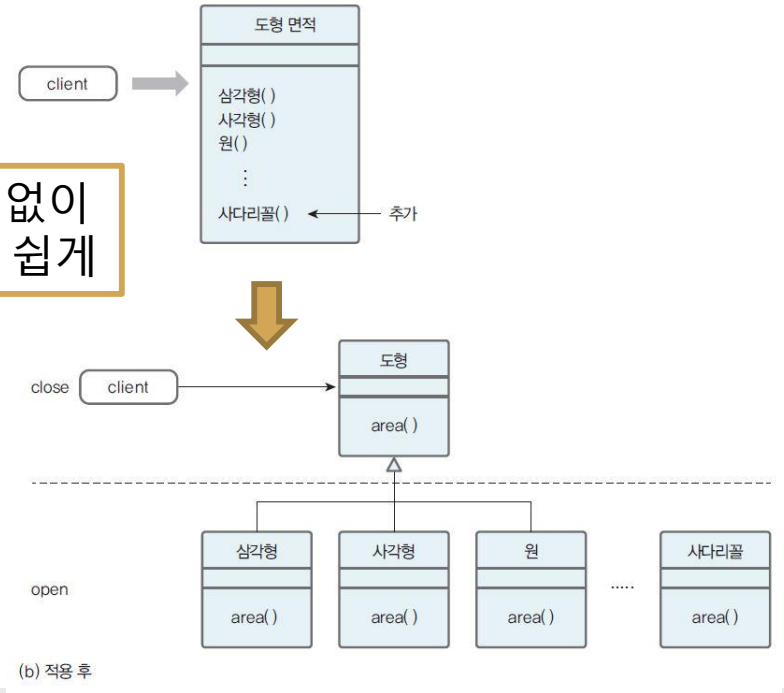
1.2 OCP

- 개방-폐쇄의 원칙(Open-Closed Principle)
  - 소프트웨어의 클래스, 모듈, 함수 등의 구성 요소는 확장(Extension)에 대해서는 열려 있어야 하지만, 변경 (Change)에 대해서는 닫혀 있어야 함

개방-폐쇄의 원칙 준수 사례

```
// 모형 클래스 정의
class Shape
{
    public:
        virtual void Draw() const = 0;
};
// 모형 클래스를 상속받은 사각형 클래스 정의
class Square : public Shape
{
    public:
        virtual void Draw() const;
};
// 모형 클래스를 상속받은 원 클래스 정의
class Circle : public Shape
{
    public:
        virtual void Draw() const;
};
// 부모 클래스인 모형 클래스를 활용하여 확장성을 보장함
void DrawAllShapes(Set<Shape*>& list)
{
    for (Iterator<Shape*>i(list); i; i++)
        (*i)->Draw();
}
```

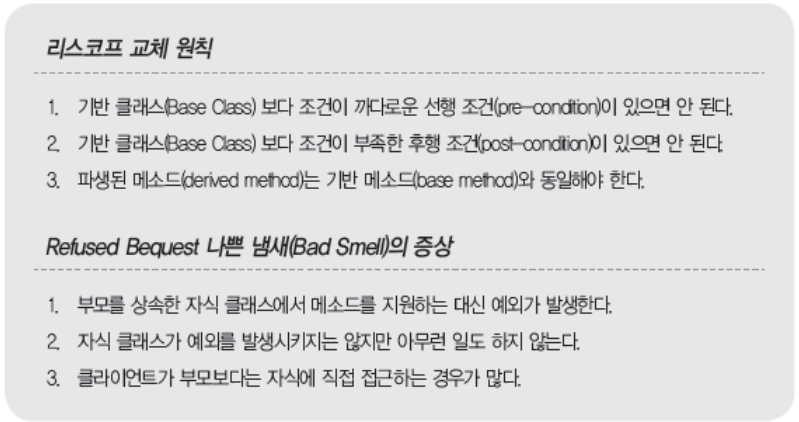
기존코드 변경없이
클래스 확장은 쉽게



(b) 적용 후

1.3 LSP

- 리스코프 교체 원칙(Liskov Substitution Principle)
  - 자식 타입은 언제나 부모 타입을 대체할 수 있어야 한다.
  - 상속 관계에서 부모와 자식 간에는 is-a 관계가 성립해야 되고, 이는 자식이 부모의 메소드 중 일부를 거부하면 안 된다는 것을 의미



리스코프 교체 원칙

1. 기반 클래스(Base Class) 보다 조건이 까다로운 선행 조건(pre-condition)이 있으면 안 된다.
2. 기반 클래스(Base Class) 보다 조건이 부족한 후행 조건(post-condition)이 있으면 안 된다.
3. 파생된 메소드(derived method)는 기반 메소드(base method)와 동일해야 한다.

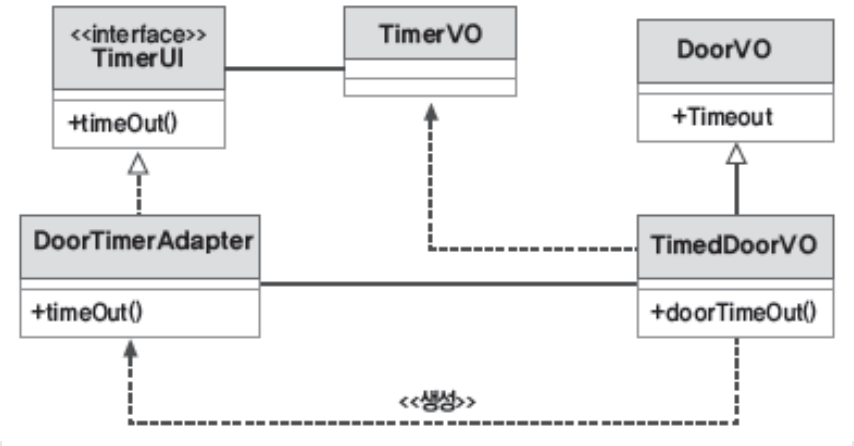
Refused Bequest 나쁜 냄새(Bad Smell)의 증상

1. 부모를 상속한 자식 클래스에서 메소드를 지원하는 대신 예외가 발생한다.
2. 자식 클래스가 예외를 발생시키지는 않지만 아무런 일도 하지 않는다.
3. 클라이언트가 부모보다는 자식에 직접 접근하는 경우가 많다.

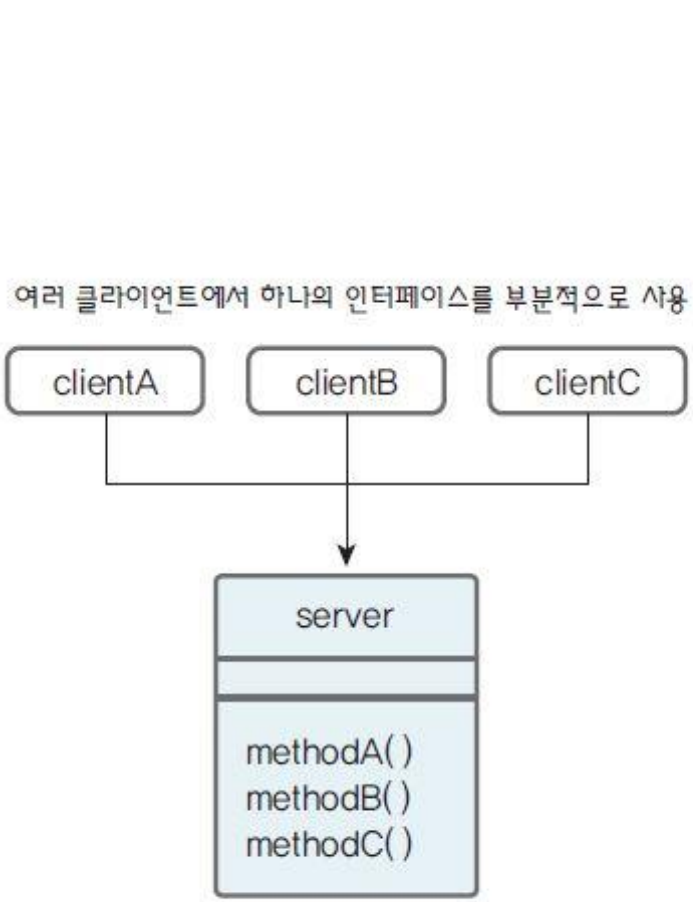
자식객체의 확장이 부모 객체의 방향을 온전히 따르도록 권고하는 원칙

1.4 ISP (1/2)

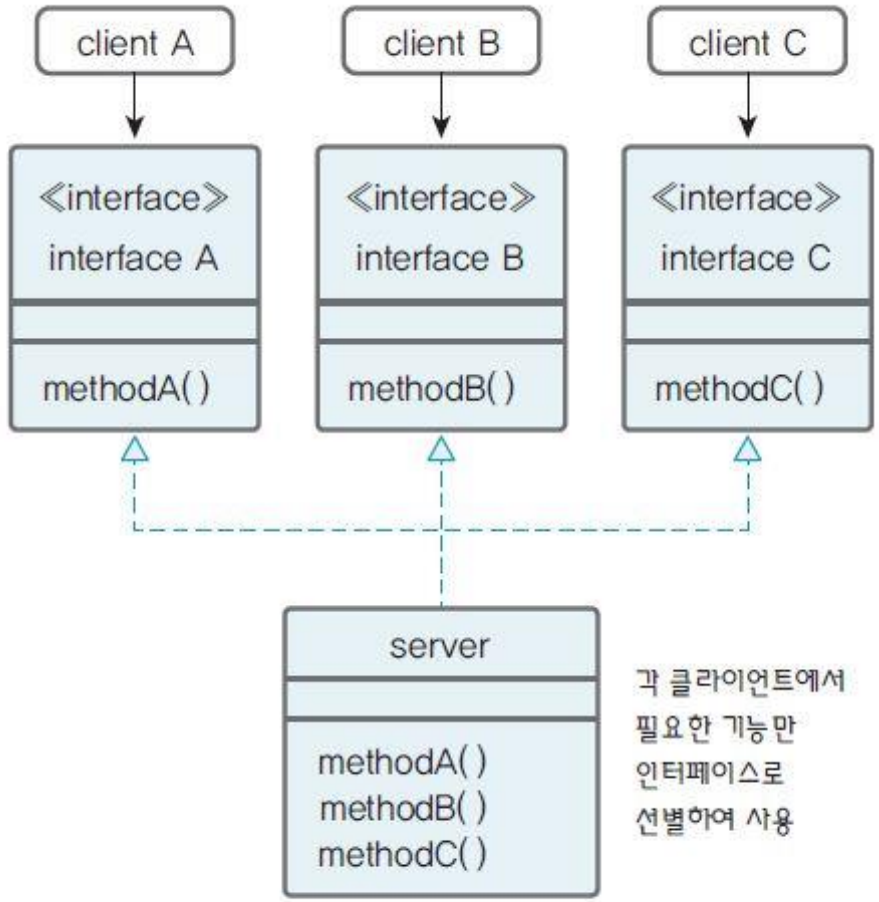
- 인터페이스 분리의 원칙(Interface Segregation Principle)
  - ISP를 따를 경우, 클라이언트는 자신이 사용하지 않는 메소드와 의존관계를 갖지 않도록 해야 함
  - 하나의 일반적인 인터페이스보다는 구체적인 여러 개의 인터페이스가 낫다



1.4 ISP (2/2)



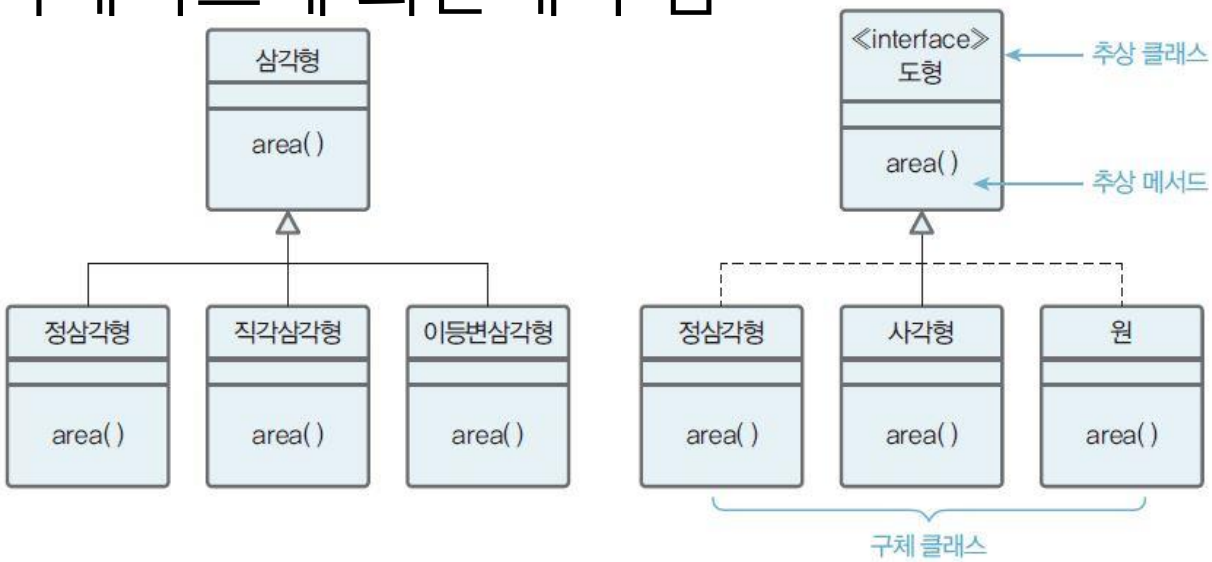
(a) 적용 전



(b) 적용 후

1.5 DIP

- 의존관계 역전의 원칙(Dependency Inversion Principle)
  - 클라이언트는 자주 변경되는 구체적(concrete) 클래스에 의존하지 말고, 추상클래스나 인터페이스에 의존해야 함



(a) 적용 전 : 구체 클래스에 의존하는 상속 구조

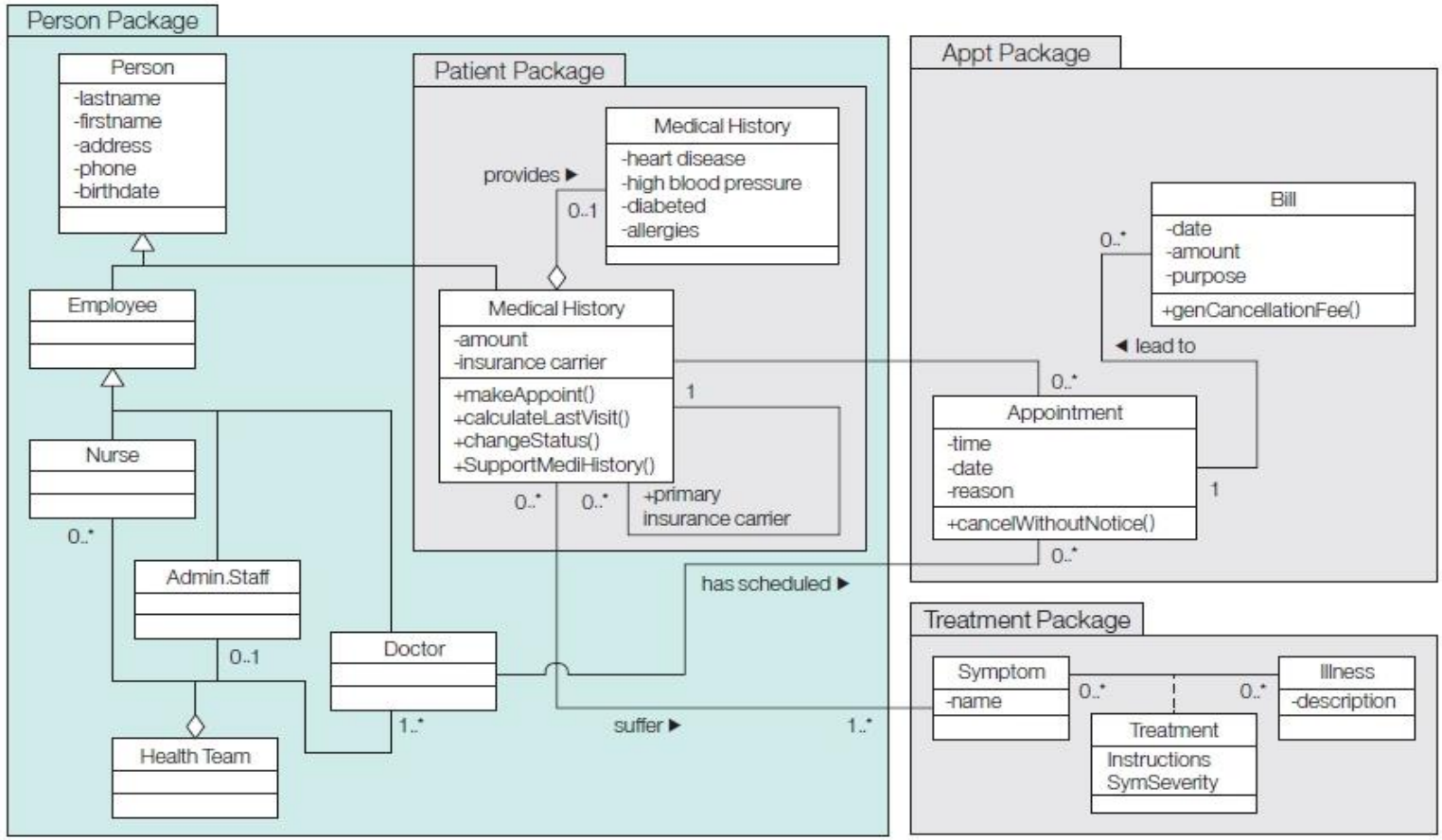
(b) 적용 후 : 추상 클래스에 의존하는 상속 구조

2. 패키지 다이어그램 (1/5)

- 패키지 다이어그램
  - 클래스 다이어그램의 클래스 간 관계를 고려하여 서로 관련성이 높은 클래스들, 즉 클래스 간 응집력이 높은 것들을 하나의 패키지로 묶어 표현한 것
  - 전체 시스템을 상위 수준(혹은 서브시스템 수준)에서 바라볼 수 있도록 지원하기 때문에 직관적으로 소프트웨어의 구조를 이해하기 쉬움
  - 추후 하드웨어 장치에 소프트웨어를 할당(혹은 배치)하기 위한 기본 단위로 패키지를 활용

2. 패키지 다이어그램 (2/5)

- 패키지 다이어그램 예시



환자 진료 예약 시스템의 패키지 다이어그램 작성 예

2. 패키지 다이어그램 (3/5)

- 패키지다이어그램 작성 절차
  1. 클래스들의 관계를 고려하여 관련성이 높은 것들을 그룹핑하여 패키지로 정의
     - 일반화(Generalization) 관계나 집합(Aggregation) 관계를 형성하는 모든 클래스는 각각 하나의 패키지로 묶는다
     - 클래스의 의미적 속성, 즉 클래스의 멤버 변수나 멤버 함수의 유사성을 고려하여 클래스를 패키지화 할 것인지 결정

2. 패키지 다이어그램 (4/5)

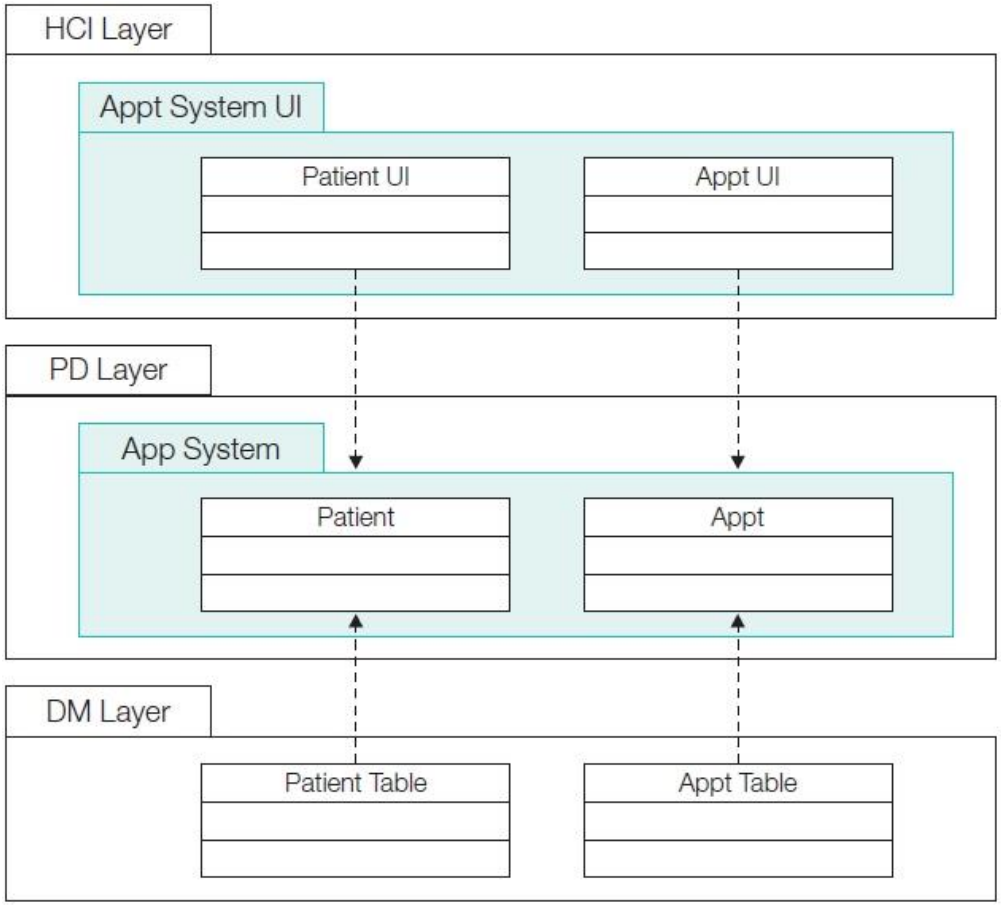
2. 패키지 간의 관계 부여 : 서로 다른 패키지를 가로지르는 기존 클래스 간의 관계 타입을 고려하여 패키지 관계를 결정
3. 패키지 관계를 정의하면 다이어그램의 프레임을 사용하여 패키지 다이어그램 완성

- 3계층 아키텍처
  - HCI(Human Computer Interaction) 계층
  - PD(Problem Domain) 계층
  - DM(Data Manipulation) 계층

2. 패키지 다이어그램 (5/5)

- 환자 진료 예약 시스템의 3계층 아키텍처 예

예제 사진 참고 11120



3. 자료구조 설계 (1/5)

- 자료구조 설계
  - 지속적으로 보관해야 하는 데이터를 유지·관리하기 위하여 파일 혹은 데이터베이스 테이블을 설계함
  - PD 계층에 포함된 모든 클래스를 살펴보고 클래스 정보로부터 자료 구조를 정의해야 함
- RDB를 객체지향 프로그램에서 사용하는 경우
  - 관계형 데이터베이스 자체를 설계
  - 애플리케이션에서 DBMS와 어떻게 통신하는지 설정
  - 매핑 되는 과정에 테이블 개념 및 정규화 규칙은 클래스와 관계에도 적용

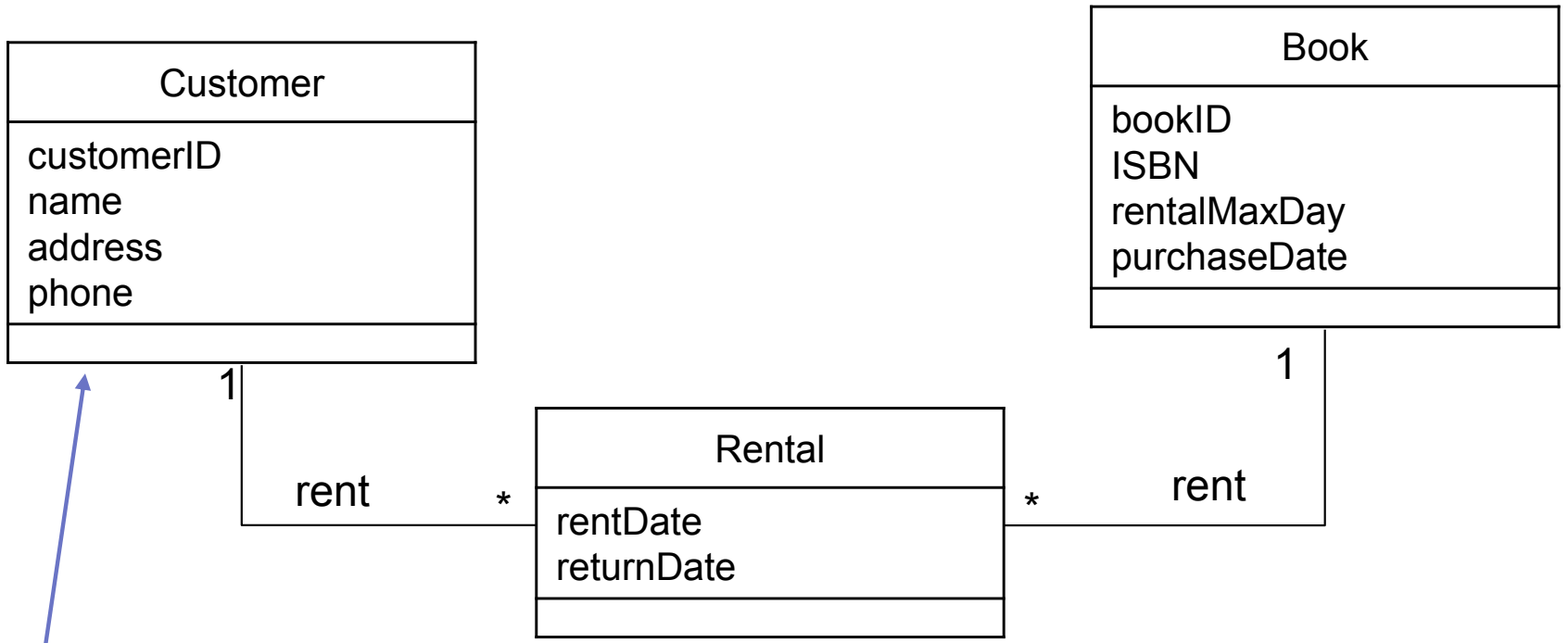
3. 자료구조 설계 (2/5)

- 클래스 다이어그램으로부터 임시로 값을 저장하는 휘발성 변수를 제거
- 모든 클래스를 하나의 데이터베이스 테이블로 매핑
- 클래스의 속성(멤버)은 DB 테이블의 열(column)로 매핑
  - 속성 : 객체지향과 관계형 데이터베이스에서 공유하는 단어
- 클래스의 메소드(멤버 함수)는 프로그램의 함수 또는 DB의 저장형 프로시저로 매핑

3. 자료구조 설계 (3/5)

- 데이터베이스 테이블 생성 규칙
  - 일대일 관계에 있는 연관관계 혹은 집합관계를 테이블의 키(Key)로 정의하는 컬럼을 추가
  - 다대다 관계에 있는 연관관계 혹은 집합관계에 대해 새로운 테이블을 생성하고 관련된 클래스를 연결할 키를 저장하는 컬럼을 생성
  - 일반화 관계에 대해서는 하위 클래스의 대응 테이블에 키를 저장할 컬럼을 추가

3. 자료구조 설계 (4/5)

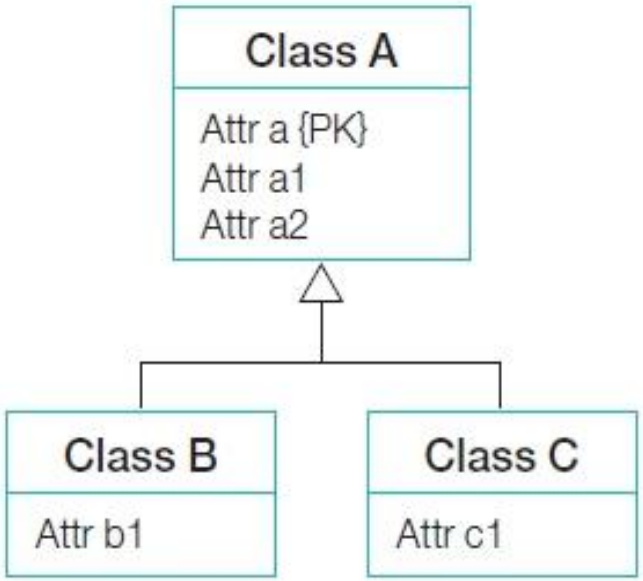


customerID	name	address	phone
2578	Park	Seoul	010-123-4567
1245	Kim	Busan	010-789-1234
5351	Lee	Incheon	010-567-0123

rentID	rentDate	returnDate	customerID	bookID
1	2019-10-29	2019-11-29	2578	10
2	2019-11-02	2019-12-02	1245	89

3. 자료구조 설계 (5/5)

- 일반화 관계가 있는 클래스에 대한 테이블 구성



Table_A

Attr a {PK}	Attr a1	Attr a1	Flag

Table_B

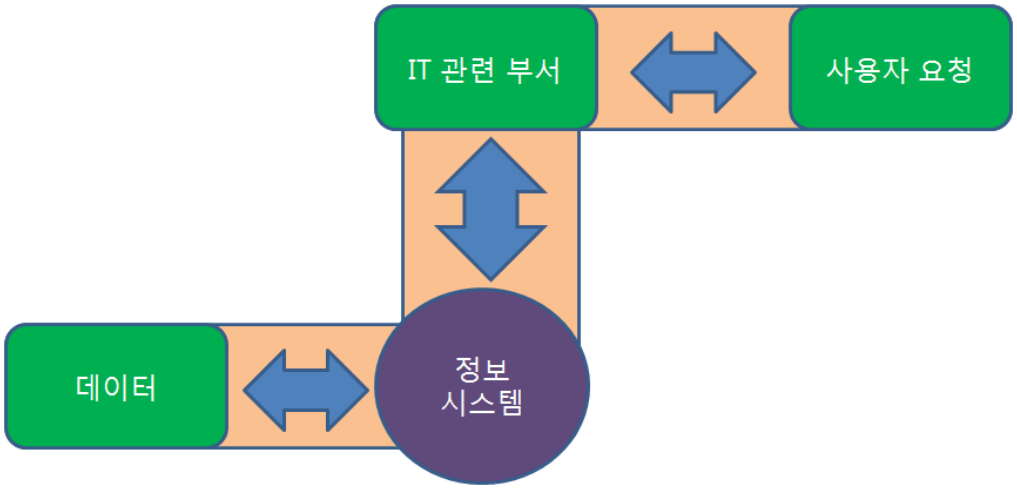
Attr a {PK, SK}	Attr b1	...

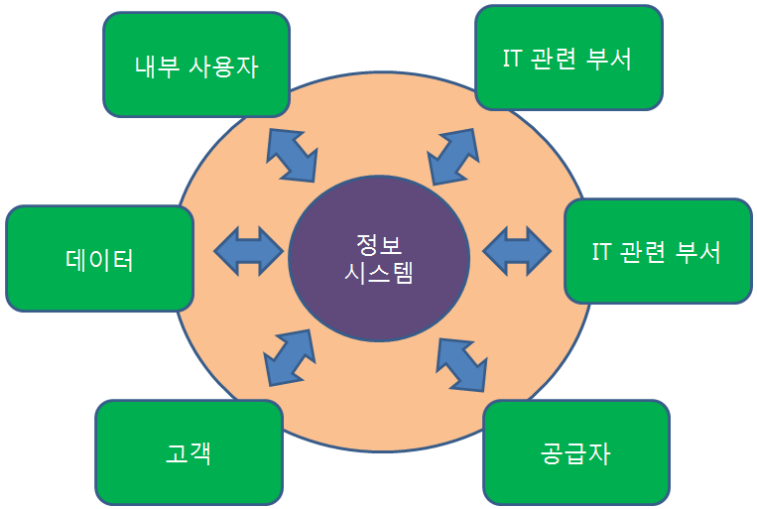
Table_C

Attr a {PK, SK}	Attr c1	...

4. 인터페이스 설계 (1/3)

- 패러다임의 변화
  - 하드웨어 가격이 높을 때는 UI 설계가 큰 이슈가 아니었음





- 처리 중심에서 사용자 중심의 시스템으로 진화

4. 인터페이스 설계 (2/3)

- UI(User Interface)
  - 정보기기나 소프트웨어의 화면 등 사람과 접하는 면
  - 보편적 인간을 모델로 '분석'
- UX(User Experience)
  - 사용자가 어떤 시스템, 제품, 서비스를 직간접적으로 이용하면서 느끼고 생각하게 되는 총체적 경험
  - 단순히 기능이나 절차상의 만족 뿐 아니라 사용자가 참여, 사용, 관찰하고 상호 교감을 통해서 알 수 있는 가치 있는 경험
  - UX에서는 특정 사용자를 모델로 '공감'

4. 인터페이스 설계 (3/3)

- 모든 소프트웨어는 사용자의 만족이 큰 목적이 됨
  - 시스템의 기능이 정확하고 완벽해야 하며 사용하기 편리해야 함
- 사용성과 좋은 UI 설계는 매우 밀접한 관계
- UI 설계 원리
  - 사용자 제어권 확보, 사용자 기억부담 감소, 인터페이스의 일관성 유지

4.1 인터페이스 구조 설계 (1/2)

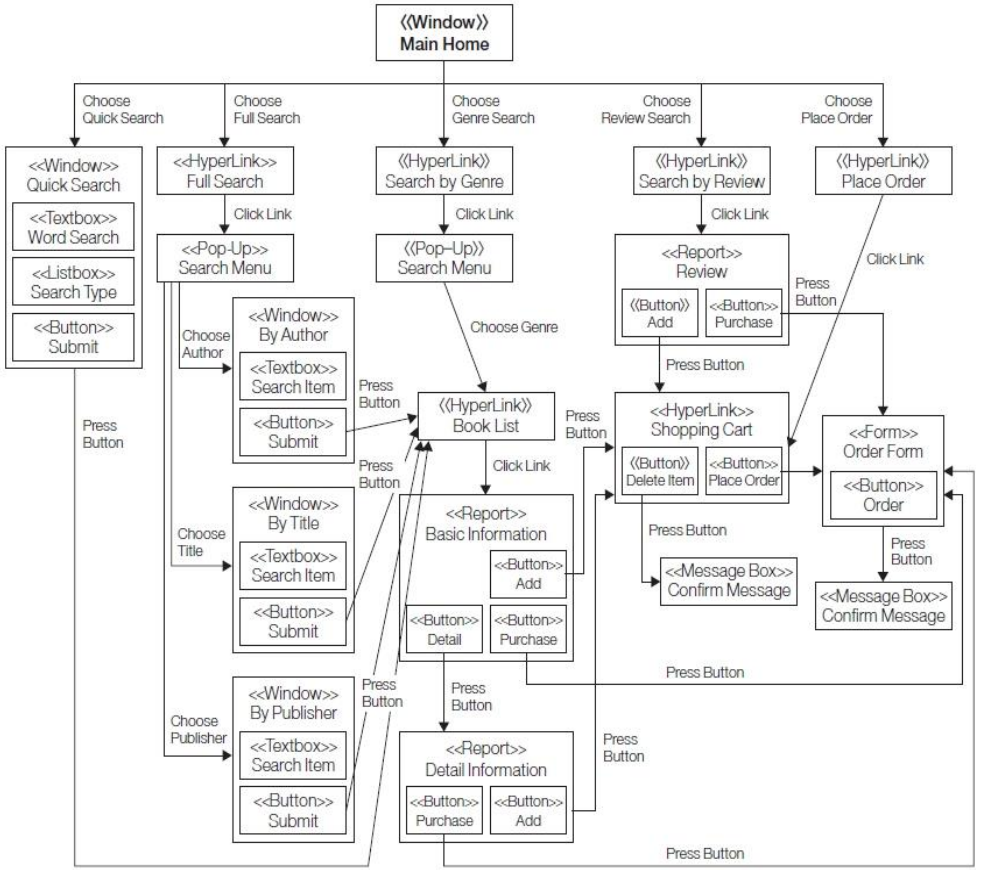
- 사용 시나리오(Use Scenario) 개발
  - 시스템 사용자 입장에서 '어떤 시나리오에 따라 시스템이 사용되는가'를 기술한 것
  - 예) 인터넷 쇼핑몰에서 특정한 상품을 주문 시

1. 사용자는 검색을 수행
2. 검색 결과에서 특정 상품의 상세 정보를 얻음
3. 상품을 장바구니에 담기
4. 장바구니에 담긴 상품을 결제하여 구매하기

  - 이때 각 시나리오 스텝은 대응되는 유스케이스 시나리오 스텝과 연결

4.1 인터페이스 구조 설계 (2/2)

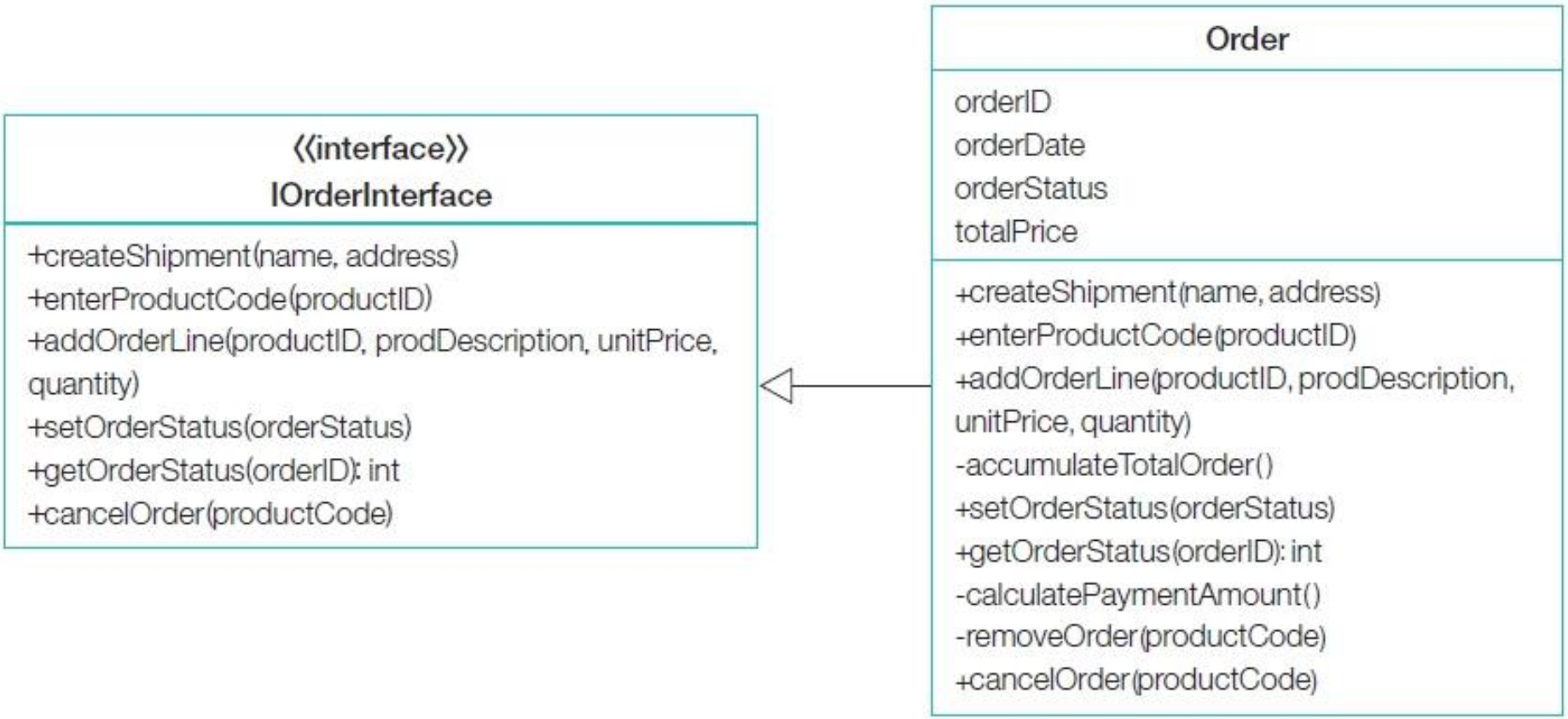
- 인터페이스 구조 설계
  - 도출된 모든 사용 시나리오의 각 스텝을 블록 다이어그램 형태로 도식화하는 것
  - 윈도우 내비게이션 다이어그램 작성



4.2 인터페이스 클래스 설계 (1/2)

- 클래스에 대해 인터페이스 클래스를 정의
  - 한 클래스의 멤버 함수가 모두 Private 메서드로만 구성된다면 별도의 인터페이스 클래스를 구성하지 않음
  - 스테레오타입 <<interface>>로 정의하고, 대응되는 도메인 클래스의 Public 메서드를 순수 가상 함수로 선언하여 포함
  - 인터페이스 클래스에 정의된 메서드는 UI의 메뉴와 연결

4.2 인터페이스 클래스 설계 (2/2)



5. 기술 환경 명세

- 설계 단계의 마지막 활동 : 하드웨어 및 지원 소프트웨어에 대한 세부 사항을 명세하는 것

- 예) 인터넷 쇼핑몰의 HW 및 SW 기술환경 명세

하드웨어 및 소프트웨어 기술 명세(예)

구분	클라이언트	웹 서버	DB 서버
운영체제	윈도우, 크롬/익스플로러	리눅스	리눅스
지원 SW	아크로뱃 리더, 리얼 오디오	아파치	오라클
하드웨어	40G HD, 17인치 모니터	200G HD	1TB HD, RAID
네트워크	Dial-up 56Kbps, Wi-Fi	Dual 100Mbps 이더넷	Dual 100Mbps 이더넷