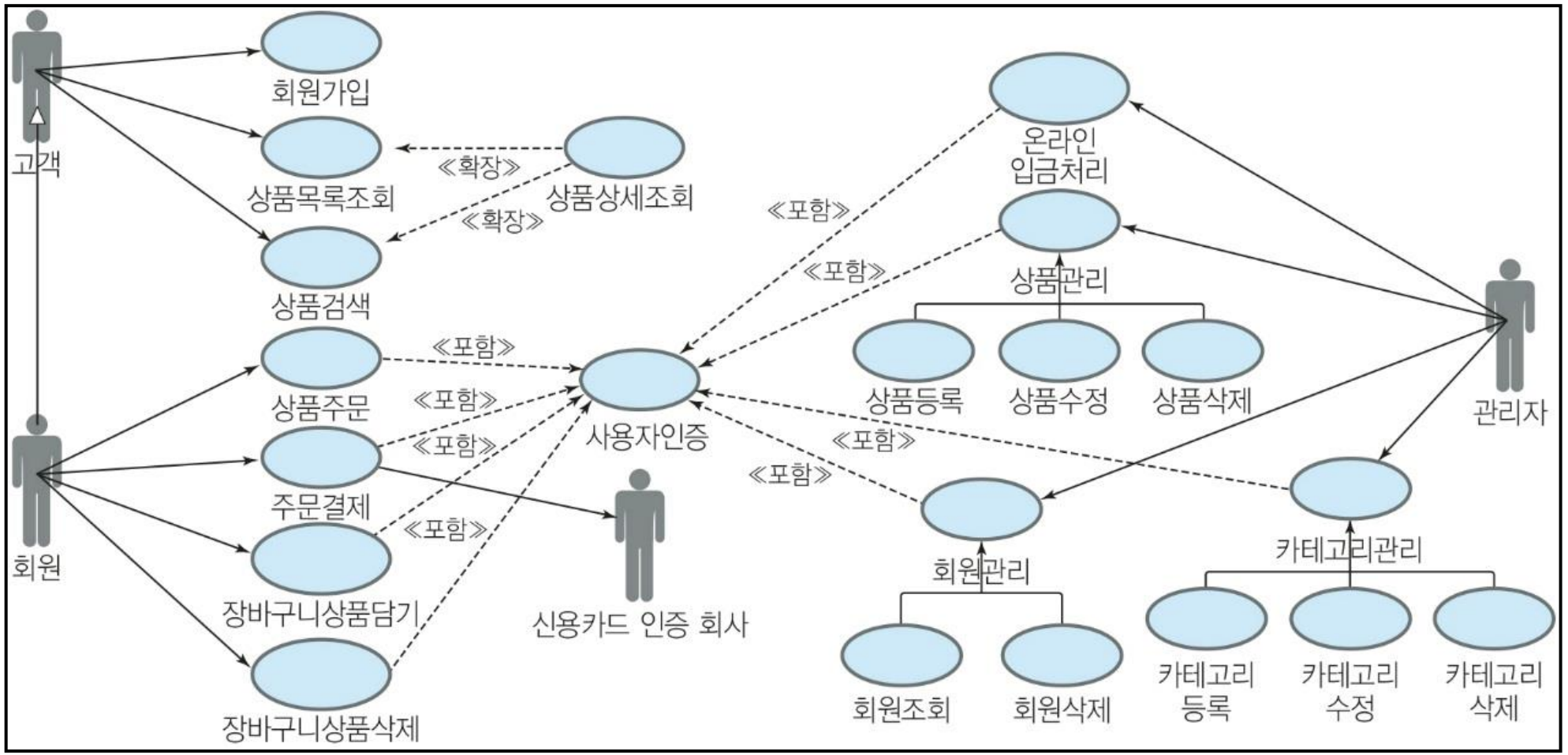
소프트웨어공학

한성대학교 컴퓨터공학부

신 성

인터넷 쇼핑몰 예

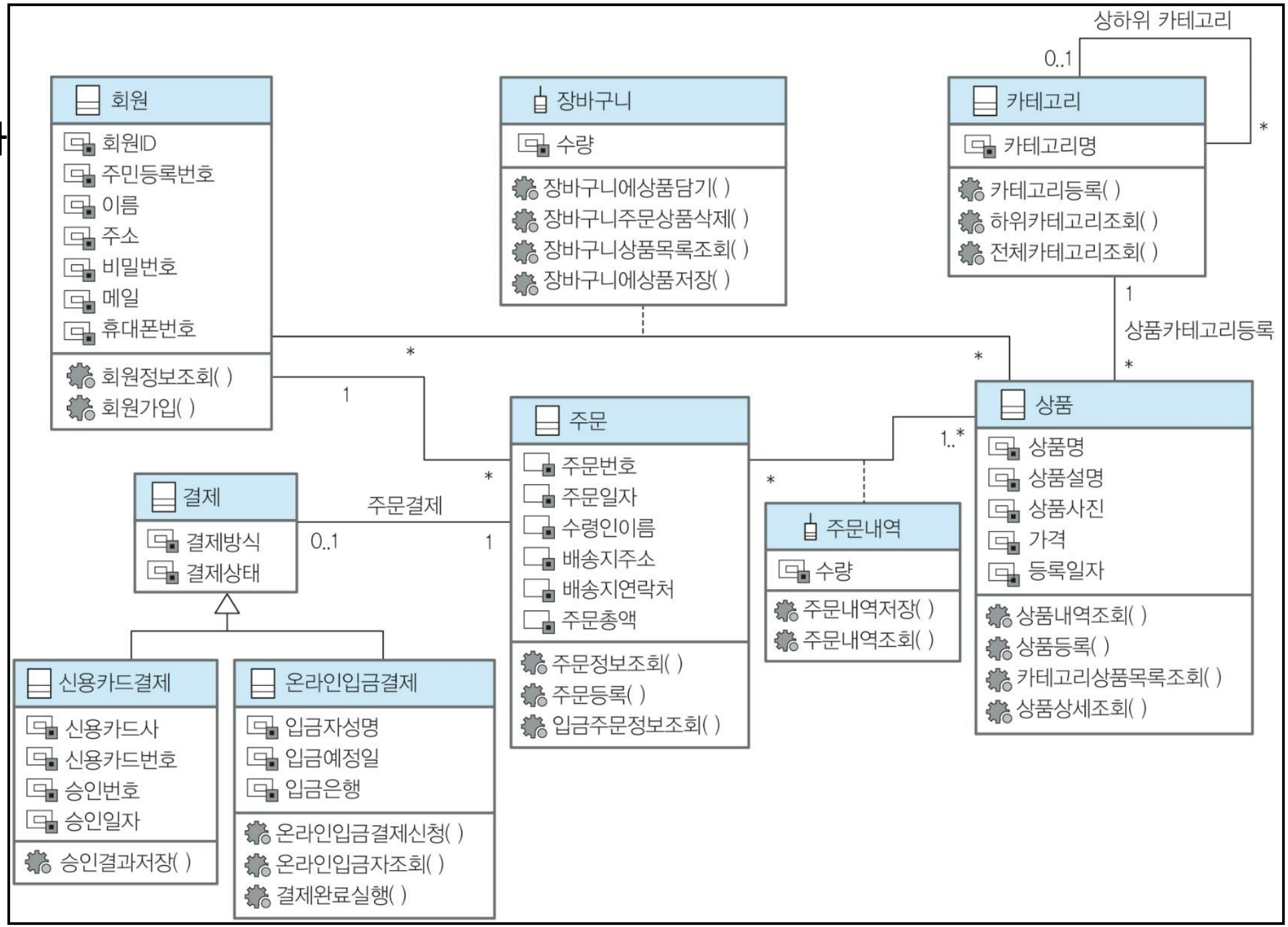
- 유스케이스 다이어그램



- 클래스 다이어그램

- 유스케이스 다이어그램과

시나리오를 바탕으로 작성



클래스 식별(스테레오 타입)

- Jacobson이 제안한 '유스케이스 기반 소프트웨어 개발 방법'에서는 클래스들을 다음과 같이 세 가지로 분류하여 식별한다.

- 바운더리(boundary) 클래스 : 주로 외부의 사용자와 상호작용하는 클래스로 사용자 인터페이스를 제어하는 역할을 수행(보통 화면) <<boundary>>

- 컨트롤(control) 클래스 : 바운더리 클래스와 엔티티 클래스 사이에서 중간 역할을 수행, 일련의 비지니스 로직 (작업 흐름을 중재, 실제 기능이 구현되는 자바 코드) <<control>>

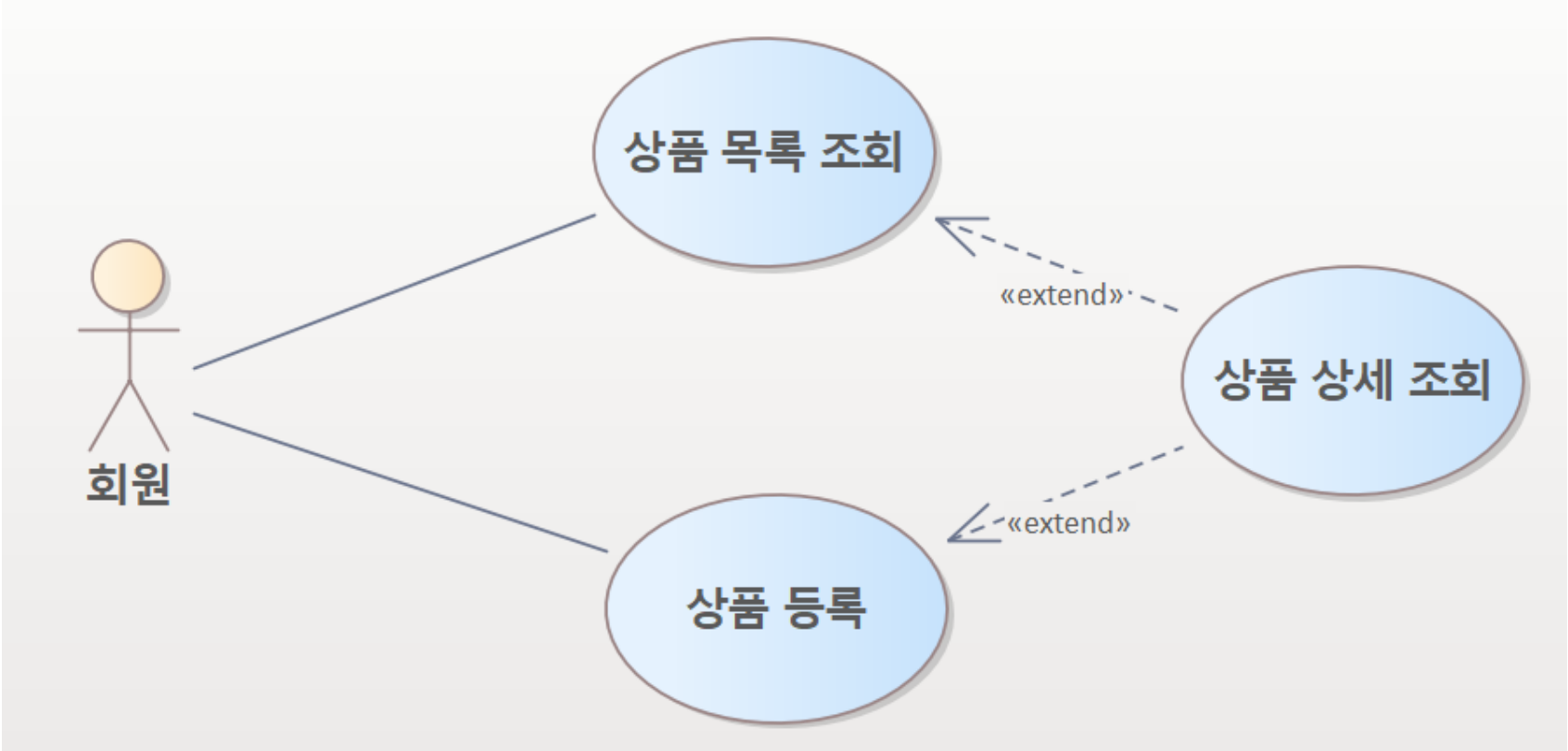
- 엔티티(entity) 클래스 : 지속적으로 존재할 필요할 있는 데이터를 모델링(데이터)

<<entity>>

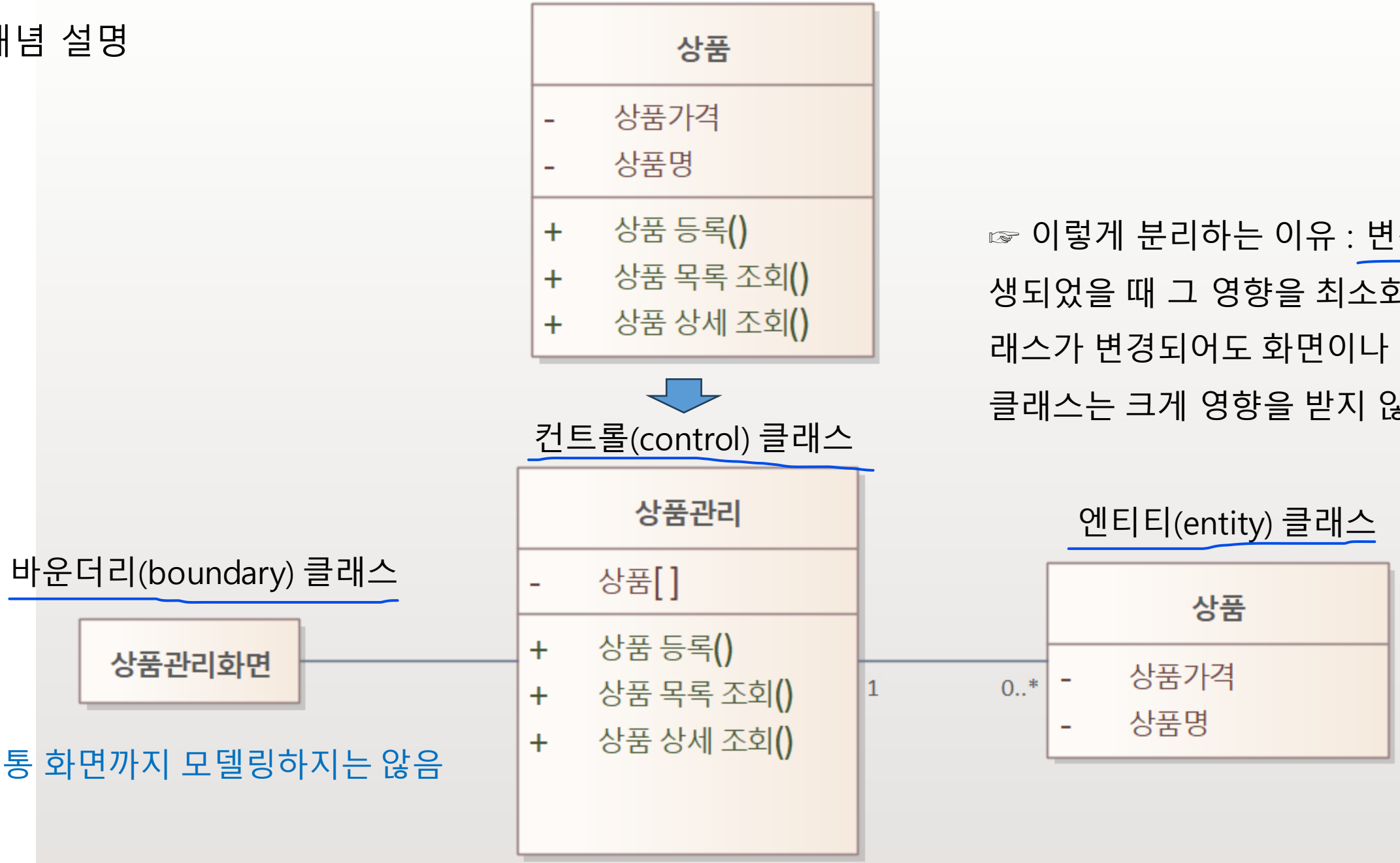
☞ 이렇게 분리하는 이유 : 변경 사유가 발생되었을 때 그 영향을 최소화, 컨트롤 클래스가 변경되어도 화면이나 데이터 저장 클래스는 크게 영향을 받지 않는다. (MVC)

클래스 식별(스테레오 타입)

- 예제)

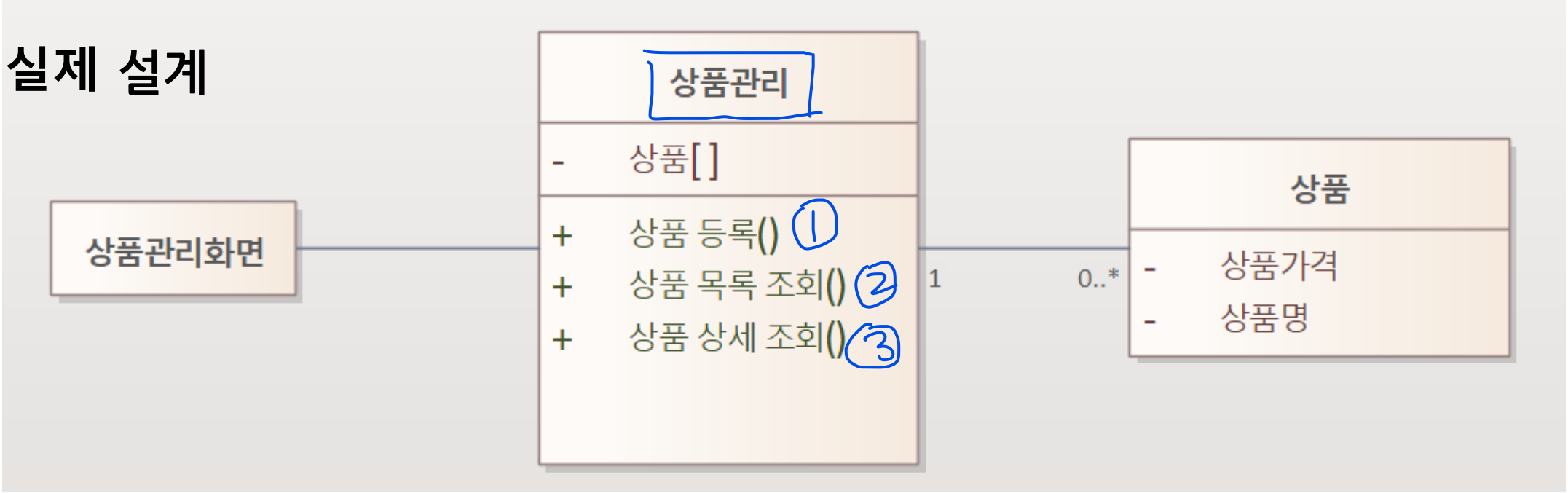


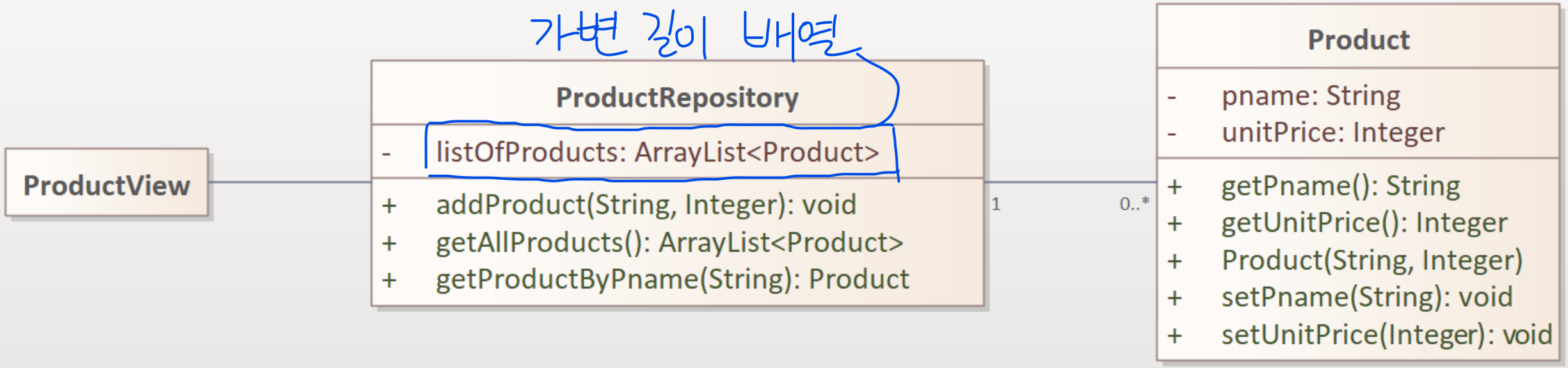
- 개념 설명



☞ 이렇게 분리하는 이유 : 변경 사유가 발생되었을 때 그 영향을 최소화, 컨트롤 클래스가 변경되어도 화면이나 데이터 저장 클래스는 크게 영향을 받지 않는다.

☞ 실제 설계


상품관리
- 상품[ ]
+ 상품 등록() ①
+ 상품 목록 조회() ②
+ 상품 상세 조회() ③
상품관리화면
1
0..*
상품
- 상품가격
- 상품명


가변 길이 배열
ProductRepository
- listOfProducts: ArrayList<Product>
+ addProduct(String, Integer): void
+ getAllProducts(): ArrayList<Product>
+ getProductByPname(String): Product
ProductView
1
0..*
Product
- pname: String
- unitPrice: Integer
+ getPname(): String
+ getUnitPrice(): Integer
+ Product(String, Integer)
+ setPname(String): void
+ setUnitPrice(Integer): void

☞ 실제 구현

❖ ProductRepository 클래스

```java
import java.util.ArrayList;

public class ProductRepository {

    private ArrayList<Product> listOfProducts = new ArrayList<Product>();

    // 전체 상품 목록 반환
    public ArrayList<Product> getAllProducts() {
        return listOfProducts;
    }

    //상품 상세 정보 반환
    public Product getProductByPname(String Pname) {
        Product productByPname=null;

        for (int i = 0; i < listOfProducts.size(); i++) {
            Product product = listOfProducts.get(i);
            if (product.getPname().equals(Pname)) {
                productByPname = product;
            }
        }
        return productByPname;
    }

    //상품 등록
    public void addProduct(String pname, Integer unitPrice) {
        Product newProduct = new Product(pname, unitPrice);
        listOfProducts.add(newProduct);
    }
}
```

☞ 구현

❖ Product 클래스

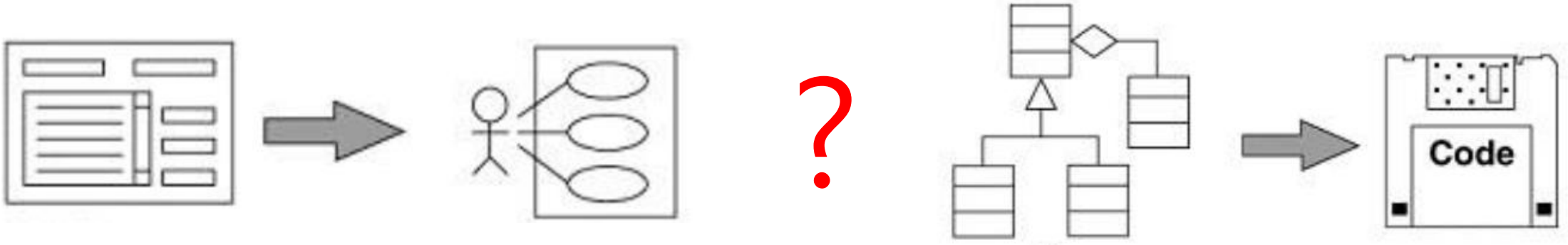
```java
public class Product {

    private String pname;         //상품명
    private Integer unitPrice; //상품 가격


    public Product(String pname, Integer unitPrice) {
        this.pname = pname;
        this.unitPrice = unitPrice;
    }

    public String getPname() {
        return pname;
    }

    public void setPname(String pname) {
        this.pname = pname;
    }

    public Integer getUnitPrice() {
        return unitPrice;
    }

    public void setUnitPrice(Integer unitPrice) {
        this.unitPrice = unitPrice;
    }
}
```

[참고] 시퀀스 다이어그램

- 지난 시간까지 클래스 다이어그램(관계)까지 수업 => 정적 모델링
- 가장 어려운 것은 유스케이스 다이어그램과 클래스 다이어그램 및 코드 사이의 차이를 줄이는 것

  - 이때 사용할 수 있는 것이 유스케이스 시나리오, 액티비티 다이어그램, 시퀀스 다이어그램 등

시스템의 행위를 나타내는 동적 모델링



소프트웨어 개발 | 유스케이스 다이어그램 (요구사항 분석) | 클래스 다이어그램 (설계) | 코드 (구현)

☞ 모든 것을 취합하여 설계 및 구현 진행

[참고] 설계 및 구현

- 유스케이스 설계 및 구현 : 하나의 유스케이스가 구현되기 위해 어떤 클래스가 필요하며, 그 클래스 간의 상호작용이 어떻게 이루어지는지를 표현
- 하나의 유스케이스는 한 개 이상의 클래스 다이어그램 및 시퀀스 다이어그램으로 구성
- 유스케이스와 단순한 1:1 매핑이 아니라, 유스케이스 모델링의 결과로 파악된 유스케이스의 규모나 상호 연관의 정도 및 복잡도를 고려하여 분할하거나 통합하여 작성할 수 있다.

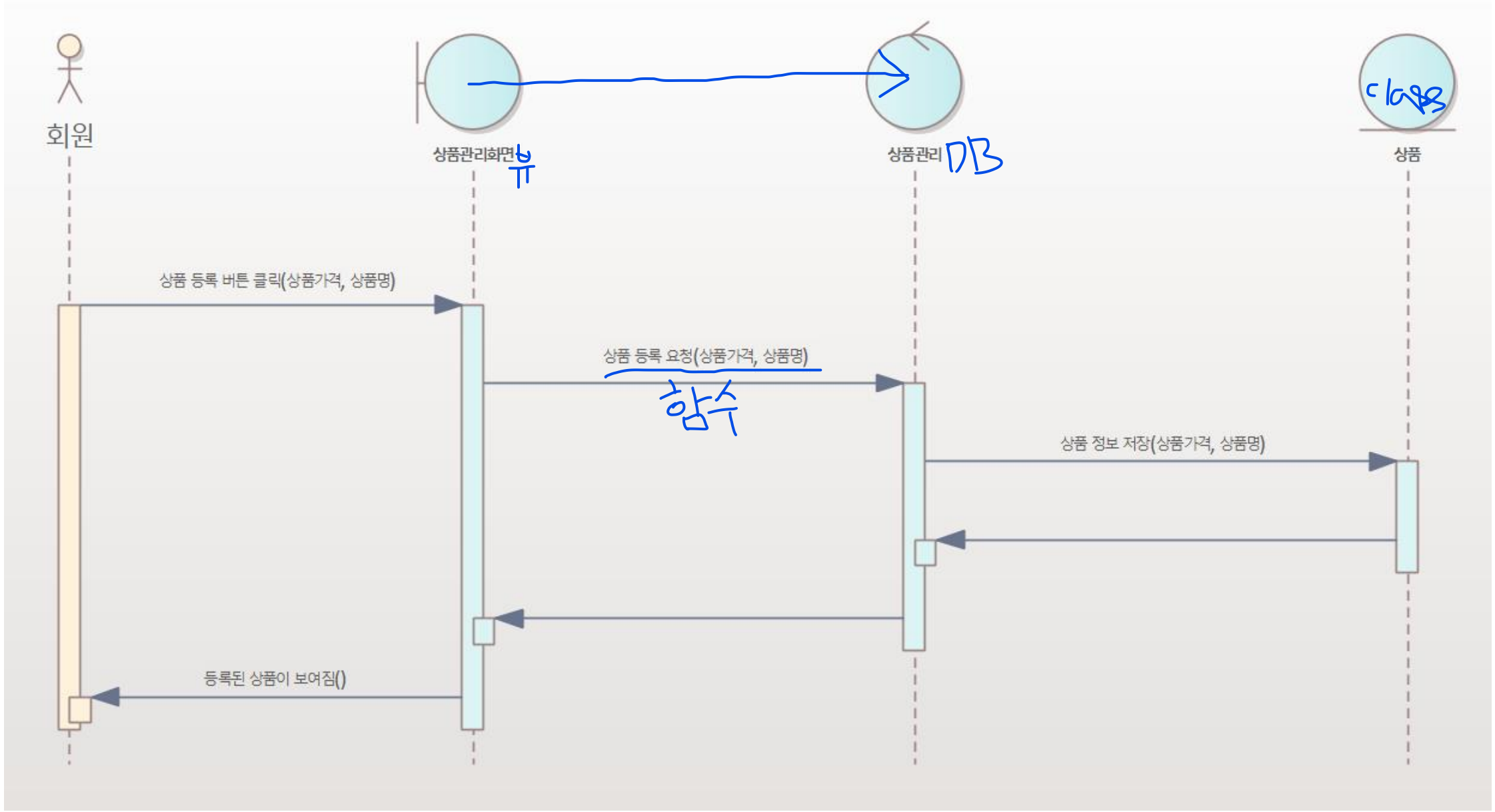
[참고] 시퀀스 다이어그램

- 정적인 클래스 다이어그램은 상당히 중요, 이 다이어그램은 소프트웨어를 이루는 클래스를 보여주지만 어떻게 동작하는지는 알려주지 않음
- 이때 사용할 수 있는 다이어그램 중의 하나가 시퀀스 다이어그램
- 유스케이스 시나리오나 액티비티 다이어그램과 같이 이벤트의 순서를 시각적으로 표현

  (다른 클래스들끼리 메소드 호출 등을 표현)

  - 보통 유스케이스당 1개씩 작성

[참고] 예시


회원
상품관리화면
상품관리
상품
상품 등록 버튼 클릭(상품가격, 상품명)
상품 등록 요청(상품가격, 상품명)
상품 정보 저장(상품가격, 상품명)
등록된 상품이 보여짐()

Q/A

☞ 궁금한 내용 등은

카톡, 문자, 메일, 전화, 방문, 이클래스 Q/A 등으로

언제든 편하게 물어보시기 바랍니다.

- 휴대폰 : 010-8873-8353
- 카카오톡 : sihns929 (또는 전화번호로 친구 등록)
- 메일 : sihns@hansung.ac.kr, 연구실: 우촌관 702호

T h a n k y o u

TECHNOLOGY

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Velit ex
ullicabo ipsum, labore sed tempora ratione asperiores des
cipserat bore sed tempora rati jgert one bore sed tem!