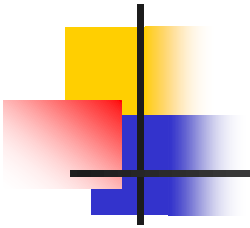


UML다이어그램

1.클래스 다이어그램(CLASS DIAGRAM)

2.쓰임새 다이어그램(USE CASE DIAGRAM)

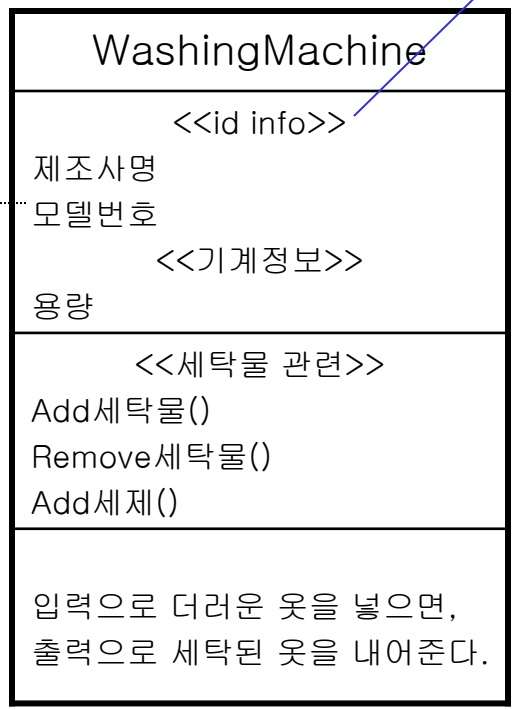
3.시퀀스 다이어그램(SQUENCE DIAGRAM)

Class Diagram

- **클래스**는 지식 도메인에 기반한 **어휘**와 **용어**로부터 만들어진다. 시스템 분석가는 의뢰인과 상담하여 그들이 가지고 있는 지식 도메인을 파악하여 정리하고, 그 도메인에서 발생되는 문제를 해결할 컴퓨터 시스템을 설계해 나가면서, UML에 사용할 용어를 선정하고 이것을 **클래스**로 모델링 하는 것이다.
- **고객**과의 대화 속에서 **명사**와 **동사**에 귀를 기울여야 한다. 명사는 클래스의 이름이 될 확률이 높다. 동사는 모델링한 클래스의 Operation이 될 가능성이 있다.
- **클래스의 핵심**이 되는 리스트**(클래스 이름, 속성, 오퍼레이션)**를 모두 정리한 다음에는, 각각의 클래스가 의뢰인의 업무에서 어떤 역할을 할 지에 대하여 묻도록 하자. → 이에 대한 대답은 클래스의 책임 설명으로 적을 수 있다.

Class Diagram

- 예제) **세탁기 클래스**



스테레오타입을 사용하여 속성 또는 오퍼레이션 리스트를 구분 지을 수 있다.

• 스테레오타입

UML의 구성요소를 확장하여 새로운 UML 요소를 만들 수 있게 한다.

표기 :<<스테레오타입이름>>

모델번호 생성 방법은 회사문서 [DT-100] 문서를 참고하라.

• 노트(Note)

추가적인 정보를 덧붙을 때 사용. 대개 속성이나 오퍼레이션에 붙인다.

{용량=5 or 8 or 10 Kg}

• 제약(Constraints)

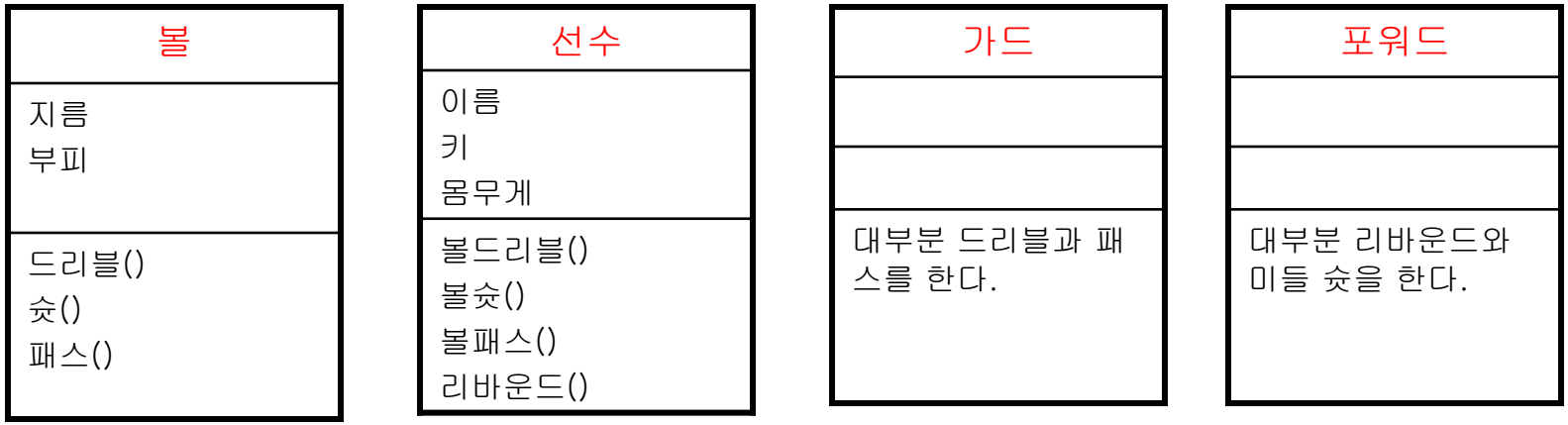
클래스가 따라야 할 규칙을 붙여줄 때 사용.

표기 : { } 안에 자유 형식의 텍스트로 표현한다.

예제는 세탁기에 수용할 수 있는 세탁물의 용량을 5, 8, 10 Kg으로만 제한함을 나타낸다.

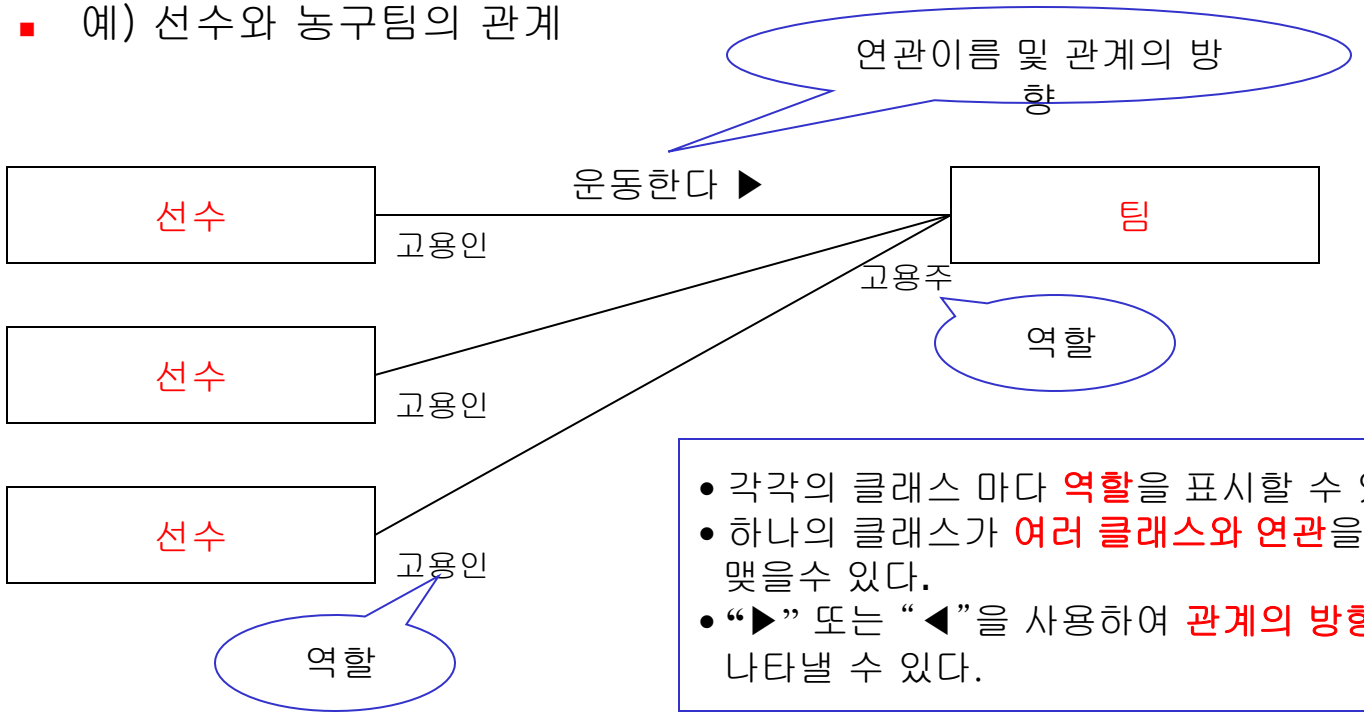
Class Diagram

- 예제) 농구 게임을 클래스 모델링 하기 :
  - *명사* : 볼, 바스켓, 팀, 선수, 가드, 포워드, 센터, 슛, 슛 시간, 3점라인, 자유투, 파울, 자유투 라인, 코트, 게임시간
  - *동사* : 슛하다, 드리블하다, 패스하다, 파울하다, 리바운드하다
  - 이 외에도 개인의 상식을 동원해서 클래스 모델링에 사용할 수 있다.
  - 다음은 이런 정보를 바탕으로 만든 Class Diagram이다. 이렇게 만든 Class Diagram은 나중에 다시 농구팀 감독과 이야기할 때 충분한 기본 자료로 사용하여 더 많은 정보를 얻어내는데 도움을 줄 것이다.



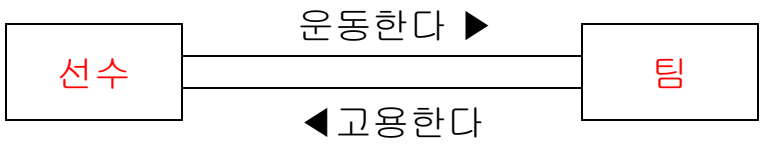
Class Diagram – 연관

- 연관(Association)
  - 클래스가 개념적으로 서로 연결되어 있음을 말한다.
  - 예) 선수와 농구팀의 관계

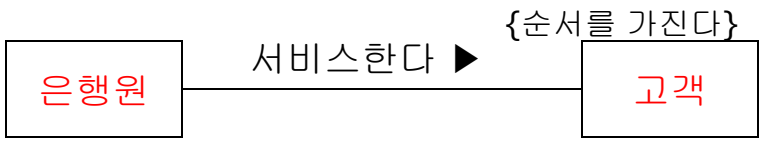


- 각각의 클래스 마다 **역할**을 표시할 수 있다.
- 하나의 클래스가 **여러 클래스와 연관**을 맺을수 있다.
- "▶" 또는 "◀"을 사용하여 **관계의 방향**을 나타낼 수 있다.

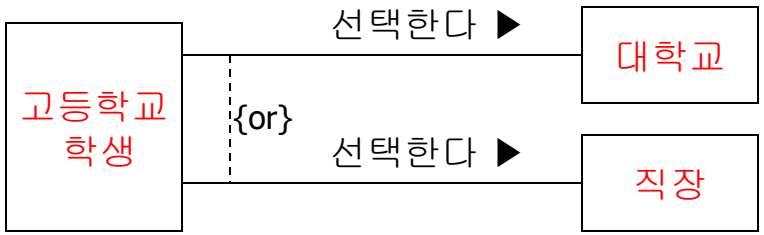
Class Diagram – 연관



- 클래스 사이에 **두개의 연관**이 나타날 수 있다.



- 연관에 **제약(Constraints)** 을 둘 수 있다. 그림에서는 "서비스한다"에 {순서를 가진다} 라는 제약이 가해져서 고객의 순서대로 은행원이 서비스 한다.

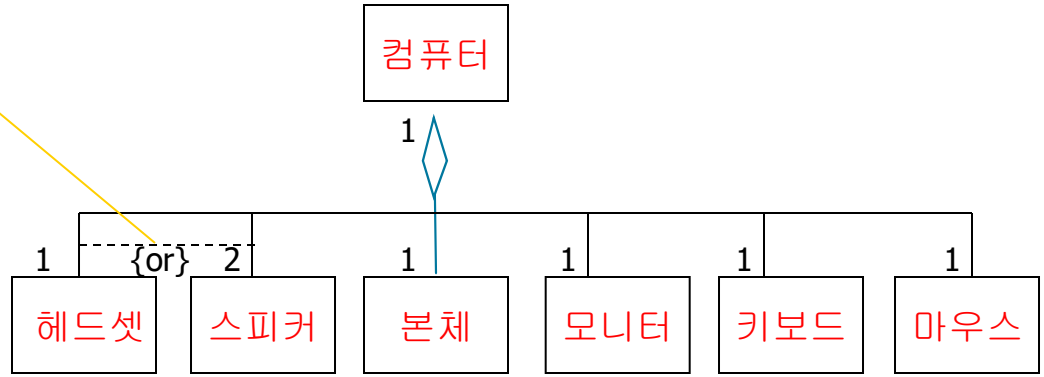


- 또 한가지 **제약** 은 두개의 연관선 사이를 점선으로 잇고 이 위에 {or} 로 표기하는 **{or} 관계**이다. 그림은 고등 학생이 진로를 정하는 상황을 모델링 한 것이다.

집합연관 (Aggregation)

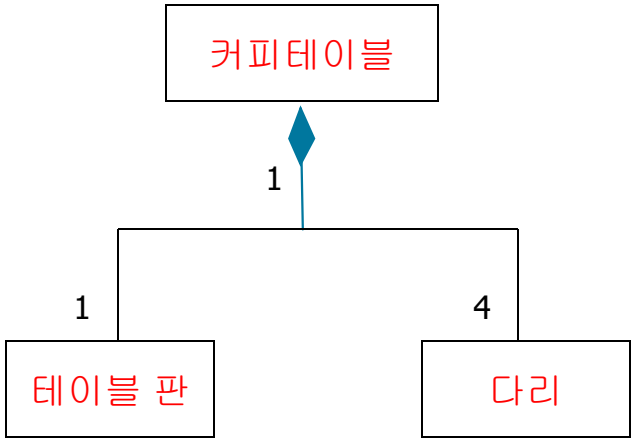
- **집합연관 (Aggregation)**
  - 하나의 클래스가 여러 개의 컴포넌트 클래스로 구성되어 있는 경우가 있다. 즉, 컴포넌트 클래스와 전체 클래스가 "부분-전체" 연관 관계를 가질 때 **집합연관**이 된다.
  - *표기* : 컴포넌트 클래스와 전체 클래스를 선으로 잇고, 빈 마름모꼴을 전체 클래스 쪽에 붙여서 나타낸다.

• 집합연관에 대한 제약
OR 관계를 모델링 하려면, 두개의 집합 연관선 사이를 점선으로 이은 다음에 **{or}**을 써주면 된다.



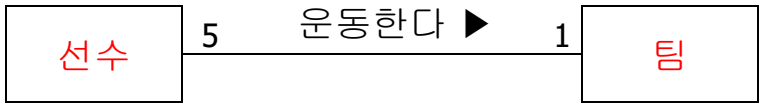
복합연관 (Composition)

- **복합연관 (Composition)**
  - 강한 집합 연관으로써 각 컴포넌트 클래스가 오직 하나의 전체 클래스에서만 의미를 가질 때, 복합연관으로 표현한다.
  - *표기* : 각각의 컴포넌트는 전체 클래스 쪽으로 향하여 안이 채워진 마름모 꼴의 화살표를 연결한다.



Class Diagram – 다중성

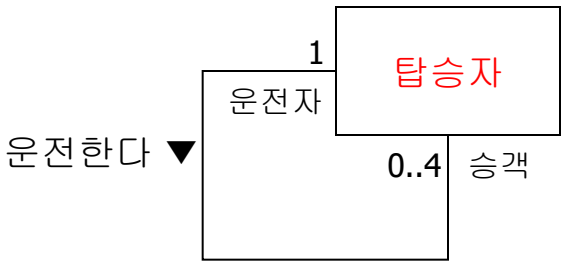
- 다중성
  - 연관되어 있는 두 클래스 사이에서 한 클래스의 객체와 관계를 가질수 있는 다른 클래스의 객체 개수. 이것을 다이어그램에서 나타내면 해당 클래스 가까이(그리고 연관선 위)에 객체의 수를 써준다.
  - 예를 들면, 팀의 입장에서 보면 다섯명의 선수와 연관되어 있지만 선수의 입장에서 보면 한 팀과 연관되어 있다는 것이다.



  - 표기
    - UML은 “more” 와 “many”를 표현하는 기호로서 ‘*’ 를 사용한다.
  - 예)
    - 1..* → 1또는 그이상 (one or more)
    - 2..7 → 2이상 7까지 (2 through 7)
    - 5,7 → 5 또는 7 (5 or 7)

Class Diagram – Reflexive

- 반사연관(Reflexive)
  - 클래스는 자기 자신과 연관을 가질 수도 있다. 하나의 클래스가 여러가지의 역할을 가질 때 **"재귀연관"**을 갖도록 한다.
  - 예를 들면, 탑승자(CarOccupant)는 운전자(Driver)도 될 수 있고 승객(Passenger)도 될 수 있다.
  - 예) 운전자의 역할을 맡은 탑승자는 승객의 역할을 맡은 탑승자를 0명이상 4명까지 실어 나를 수 있다.

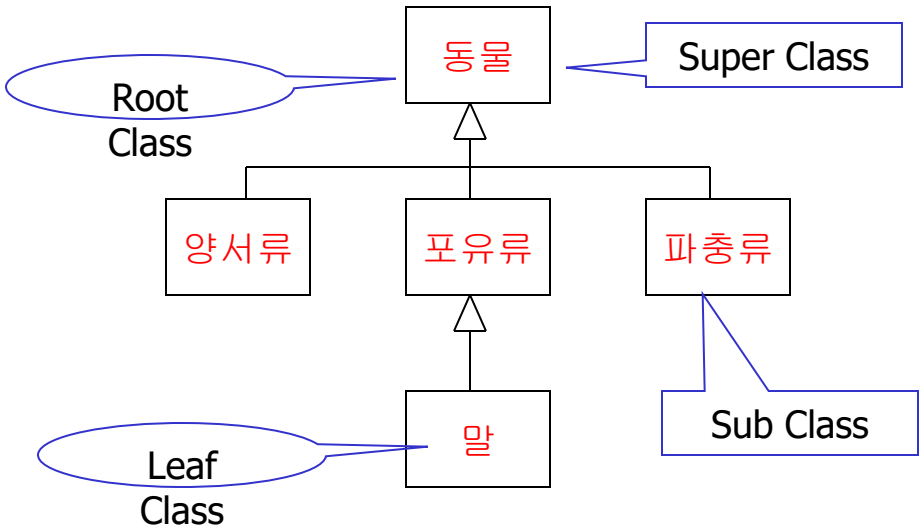


• reflexive :

1 《문법》 재귀(용법)의.
a ~ verb 재귀 동사.
2 반응하는, 반동적인.
3 반사하는.
4 반성적인 (reflective).

Class Diagram – 상속,일반화

- 한 클래스는 다른 클래스로부터 **속성**과 **오퍼레이션**을 물려 받을 수 있다. 이것을 객체지향 개념에서는 "**상속**" 이라 하고, UML에서는 "**일반화**" 라 한다.
- 상속 관계에서 상속을 받는 쪽을 Child Class 또는 Sub Class 라고 하고, 상속을 해주는 쪽을 Parent Class 또는 Super Class 라고 한다.
- Sub Class 에서 Super Class 쪽으로 속이 빈 화살표(——▷)를 연결한다. 이러한 타입의 연결 관계를 "...의 일종(is a kind of)" 라고 부른다.



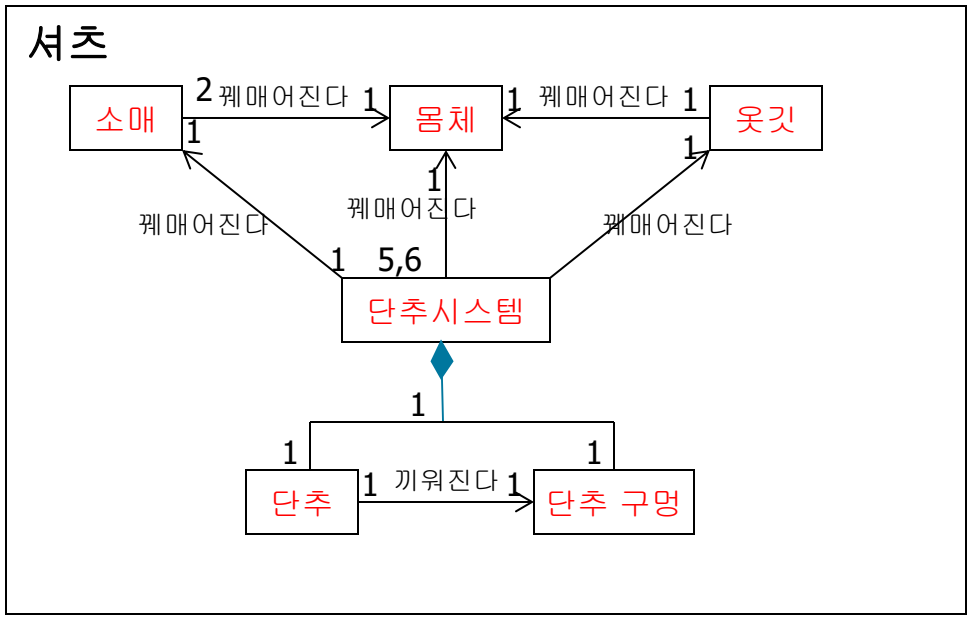
• 상속 관계를 모델링 할 때에는, 반드시 서브 클래스가 수퍼 클래스에 대해 **"is a kind of"** 관계를 가지도록 만들자.

• 만약 두 클래스가 이 관계로 맺어지지 않는다면, 차라리 다른 타입의 관계를 맺어주는 것이 더 낫다.

• Root Class: Super Class를 가지지 않는 클래스
• Leaf Class : Sub Class를 가지지 않는 클래스

문맥 (Context) – 복합체

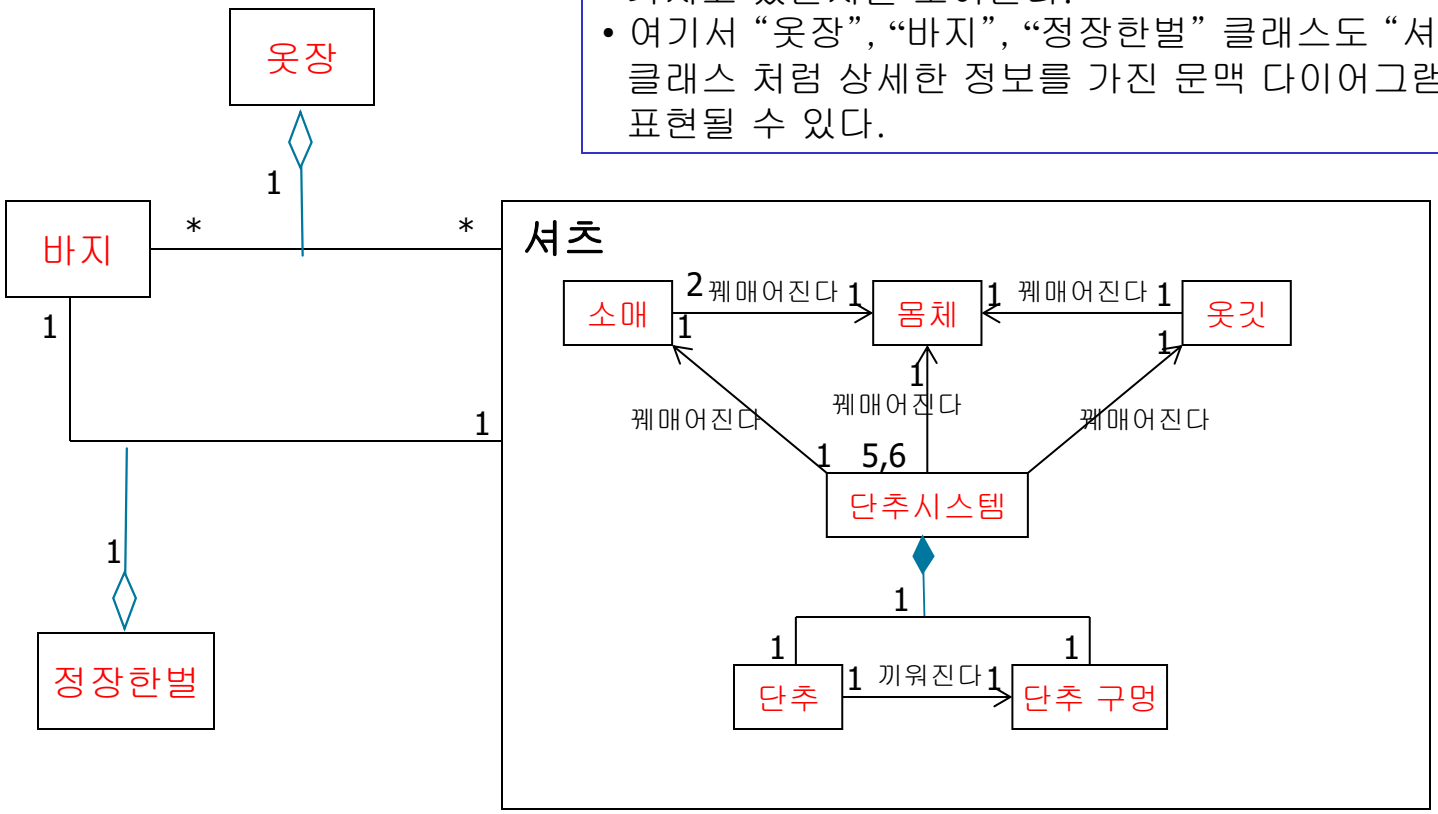
- 문맥 다이어그램은 큰 지도의 일부를 **확대**해서 그린 것이다. 상세한 정보를 더 써주어야 할 필요도 있기 때문이다.
- 예를들면, "**셔츠**" 클래스가 있다면 이 클래스를 구성하는 컴포넌트 클래스는 무엇이 있고, 이들 사이의 연관 관계는 어떻게 이루어져 있는지를 표현한다.



- **복합체 문맥 다이어그램**에서는 클래스의 컴포넌트를 나타낼 때 전체 클래스 사각형 안에 다이어그램을 그려준다.

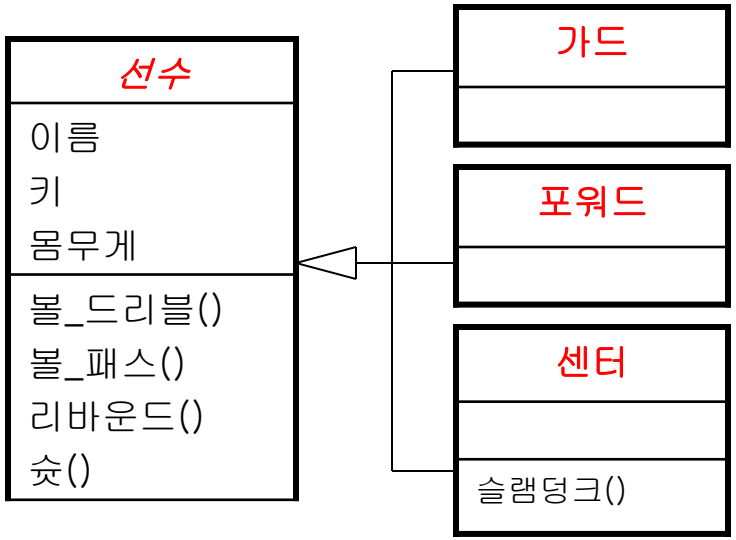
문맥(Context) – 시스템

- **시스템 문맥 다이어그램**은 한 클래스의 컴포넌트와 이 클래스가 시스템내의 다른 클래스와 어떤 관련을 가지고 있는지를 보여준다.
- 여기서 “옷장”, “바지”, “정장한벌” 클래스도 “셔츠” 클래스 처럼 상세한 정보를 가진 문맥 다이어그램으로 표현될 수 있다.



Class Diagram – 추상클래스

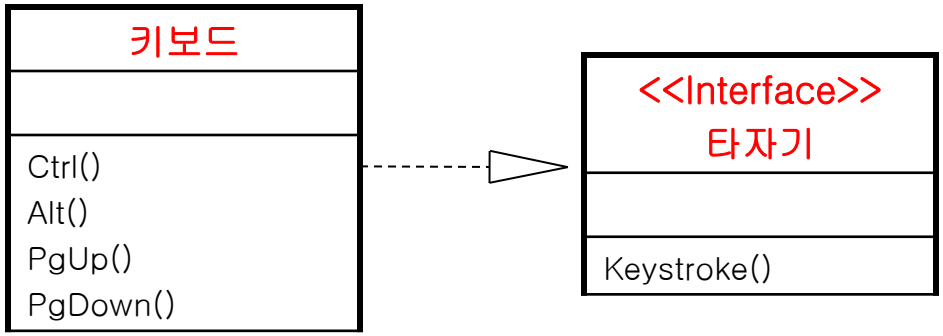
- 어떤 Sub Class의 Super Class가 있을 때, 만약 이 Super Class의 구체적인 인스턴스(Instance)를 만들 필요가 없을때에는 "**추상 클래스**"로 만들자. 즉, 클래스의 객체를 생성하지 않는 클래스를 "추상 클래스" 로 만든다.
- *표기* : 클래스명을 이탤릭으로 쓴다.



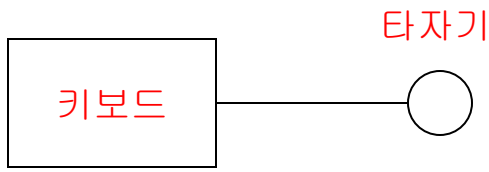
인터페이스와 실체화

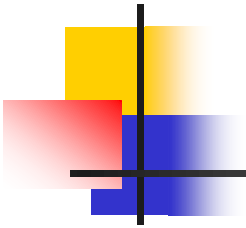
- 어떤 클래스들이 동일한 Super Class와 관계를 가지지 않았는데, 같은 signature를 가진 Operation이 존재한다면 이것은 Operation의 재사용으로 간주된다.
- 이런 재사용을 위한 Operation의 집합을 인터페이스(Interface)라고 한다. 인터페이스는 클래스의 일정한 행동(behavior)을 나타내는 Operation의 집합으로, 다른 클래스에서 사용될 수 있다.
- 자바에서 인터페이스는 method의 prototype만 선언되어 있고, 인터페이스를 구현 (Implementation)한 클래스에서 method를 정의한다. 이것을 UML에서는 실체화(realization)이라 한다.

• 타자기 의 행동을 실체화한 키보드 클래스



• 인터페이스를 실체화한 클래스를 나타내는 간단한 방법





힌트와 조언

- 구조가 좋은 Class Diagram
  - System의 정적 설계 View의 한 관점을 전달하는데 초점
  - 해당 관점을 이해하는데 필수적인 요소들만 표현
  - 추상화 계층에 알맞은 정도의 상세 내용을 제공하고 이해하는데 필수적인 장식만을 사용
  - 중요한 의미를 이해할 수 있을 정도로 복잡하게
- Class Diagram 작성 규칙
  - 목적에 맞는 명칭 사용
  - 서로 교차하는 선(관계 표현)을 최소화 하도록 배치
  - 의미적으로 관련 있는 Class들을 가까운 위치에 배치
  - 시각적 암시를 활용 (Note 또는 색깔)
  - 너무 많은 관계를 나타내지 않도록 도해

Use Case Diagram

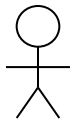
- 쓰임새(Use Case) 란? 시스템의 사용에 대한 시나리오의 집합.
- 목적
  - **사용자의 관점에서 시스템을 모델링 하기 위함.**
  - 즉, 사용자가 시스템에 대하여 바라는 바를 표현.
  - 사용자의 시점을 빨리 이해함으로써 쓸모있고(useful), 쓸수있는(usable) 시스템을 만들 수 있도록 함.
- 사용
  - 대개 의뢰인과 개발팀이 참조하는 **설계 문서의 한 부분**으로 사용.
  - 새로 만들어진 시스템을 테스트하는데 사용 : 사용자의 시스템 사용 시나리오를 표현한 것이기 때문에 테스트도 그 시나리오에 따라서 하면 됨.
  - 시스템 분석가와 사용자와 힘을 합쳐 시스템의 사용 방법을 결정하는데 도움
  - 시스템이 가진 User Interface 설계 & 프로그래밍 방식에 대해 도움.
  - [요구사항 수집]의 "시스템 요구사항 파악"의 과정과 [분석] 과정에서의 출력물 →사용자와의 대화를 통하여 얻어낸 정보를 표현.
- 예) 음료수 자동 판매기 시스템을 설계
  - 시나리오 : "음료수 사기" ← Use Case Name
    - 1. 동전을 넣는다
    - 2. 자판기가 종이컵에다가 음료를 따른다.

Use Case Diagram – 2

- **쓰임새의 분석 과정 및 방법** : *사용자를 만나 의견을 듣는 것으로 시작.*
  - 시스템 사용자를 시스템 분석과 설계의 초기 단계에 참여 시킴.
  - [요구 사항 수집]
    - 의뢰인과 대화 → 초기 클래스 다이어그램 → 사용하고 있는 용어를 통해서 개념 정리 → 사용자 와의 대화가 유연해짐.
  - **사용자와 대화 ➔ 설계하고자 하는 시스템을 가지고 어떤 일을 하는지 질문 ➔ 얻어낸 대답은 미래에 쓰임새를 만드는 토대로 사용.**
  - 쓰임새에 간단한 설명을 붙임 ➔ 설명이 많을 수록 사용자와 할 수 있는 대화의 밀도는 진해진다.
- **쓰임새의 재사용을 위한 방법**
  - **포함 (Inclusion)** : 다른 쓰임새를 구성하는 하나의 진행 과정으로 쓰임새를 사용.
  - **확장 (extension)** : 기존의 쓰임새에 몇 개의 단계를 추가하여 새로운 쓰임새를 만듬.
- **쓰임새 만들 때 주목할 사항**
  - 해당 시나리오를 시작할 때 만족 되어야 하는 선행 조건
  - 시나리오가 완료될 때 만족되어야 하는 종료조건

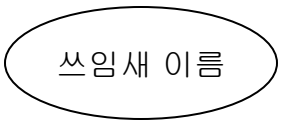
Use Case Diagram – 3

- 쓰임새 그리기
  - 행위자 (Actor)

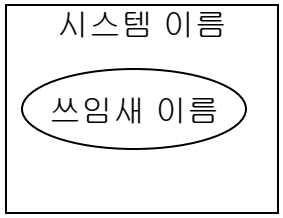


행위자 이름

    - 시스템과 교류하는 사람이나 시스템 또는 장치.
    - 쓰임새를 시작시키고, 쓰임새를 구성하는 진행 단계가 끝나면 그 결과를 받음.
    - 막대기로 사람 모양을 표현.
    - 행위자의 이름은 막대인간 아래에 쓴다.
  - 쓰임새 (Use Case)

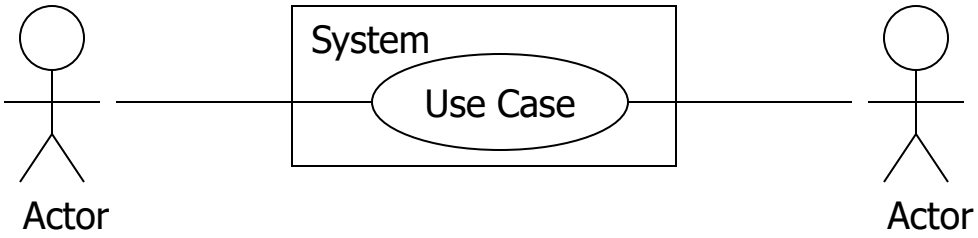


    - 타원으로 표현.
    - 쓰임새의 이름은 타원의 안쪽 또는 아래에 쓴다.
    - 쓰임새를 시작시킨 행위자는 왼쪽, 결과를 받는 행위자는 오른쪽에 표현.
    - 행위자와 쓰임새 간의 연결은 실 선으로 그림.
  - 시스템



    - 쓰임새를 둘러싸는 사각형으로 표현.
    - 행위자는 대개 시스템 외부에 있는 반면, 쓰임새는 시스템의 내부에 있다.
    - 즉, 쓰임새 분석을 통하여 시스템과 외부세계와의 경계를 효과적으로 보여줄 수 있다.

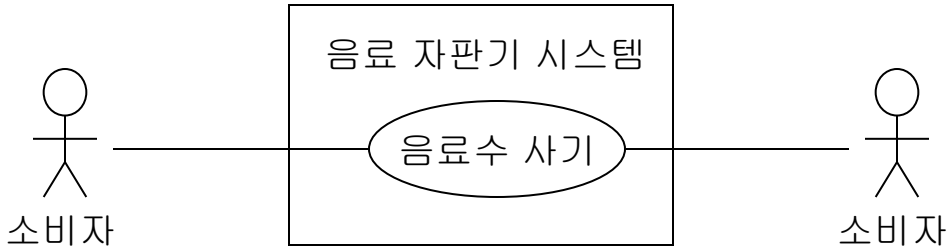
Use Case Diagram – 4



- 각 다이어그램은 한 페이지당 하나씩 그려진다.

Use Case Diagram Sample

[음료수 사기]



쓰임새 설명 : 자판기에서 음료수를 사는 경우이다.

가정 : 한번에 한명의 소비자만 사용한다.

시작 행위자 : 소비자

선행조건 : 목이 마르다

진행 단계

1. 동전 또는 지폐를 넣는다.
2. 컵이 내려오고, 음료가 채워진다.

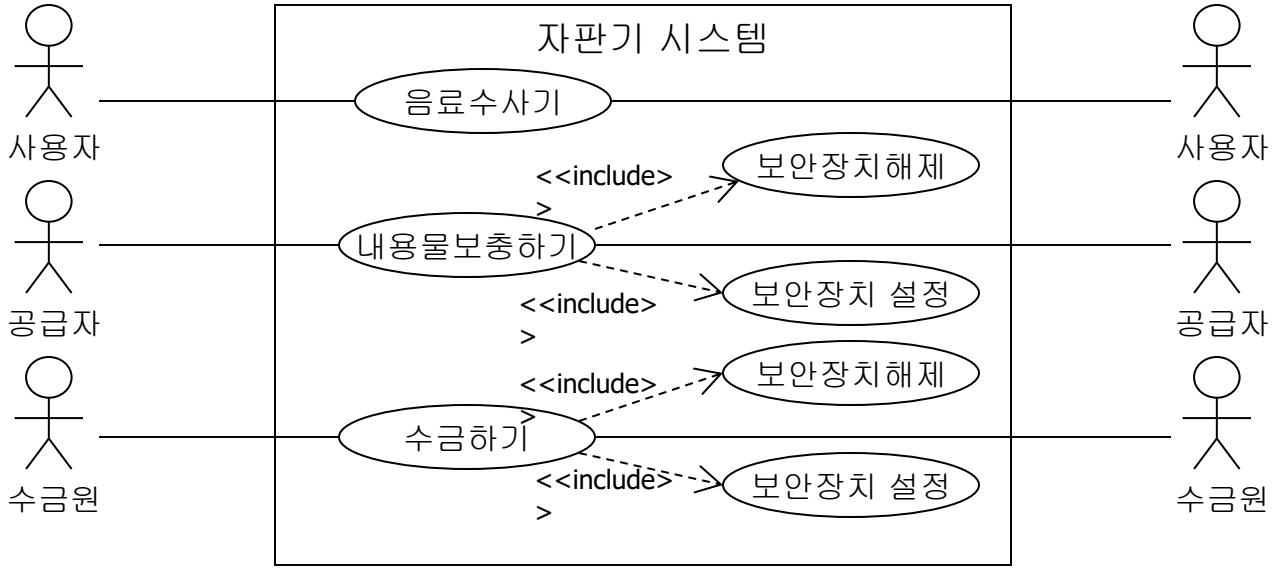
종료조건 : 음료를 가졌다.

결과 받는 행위자 : 소비자

Use Case Diagram Sample

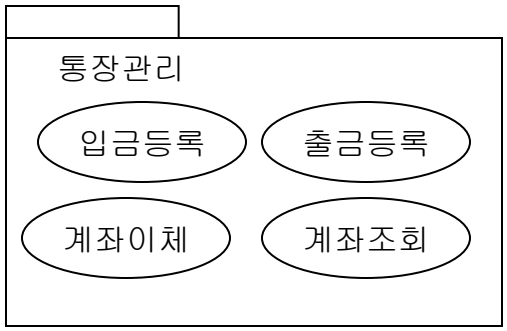
예) 자판기 내용물 공급자와 수금원이 포함된 자판기 시스템의 쓰임새 다이어그램

"내용물 보충하기" 쓰임새와 "수금하기" 쓰임새의 시나리오에서 **공통으로 중복된** *보안장치 해제나 보안장치 설정과 관련된 진행 단계들*을 따로 떼어내서 새로운 쓰임새로 만들고 포함 시켰다.



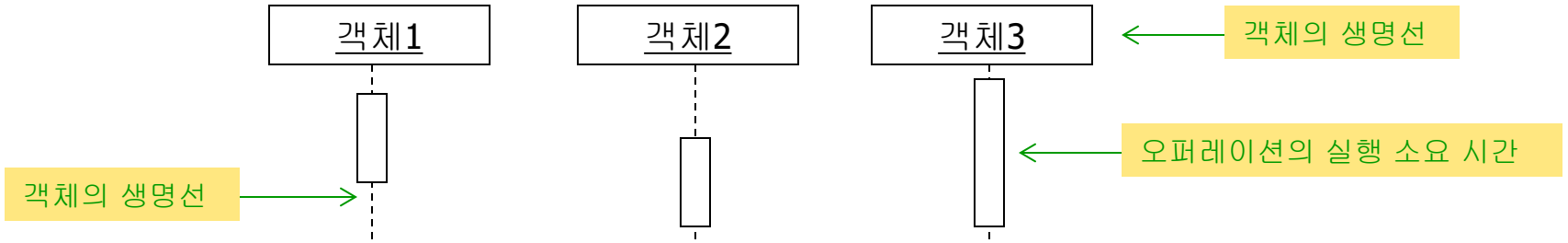
Use Case Diagram – 5

- 그룹화
  - 관련된 쓰임새를 하나의 패키지로 그룹화하는 것.
  - 시스템이 여러 개의 서브 시스템으로 구성되어 있거나, 시스템에 대한 다양한 요구를 수집하기 위하여 사용자의 의견을 조사할 때(즉, 요구사항이 각각의 쓰임새로 표현 되기 때문에) 그룹화 한다.
  - 패키지 다이어그램에서 패키지 안에 관련된 쓰임새를 넣어서 표현.
  - 예) 가계부 시스템에서 통장관리에 관련된 쓰임새를 패키지로 그룹화 한 것.



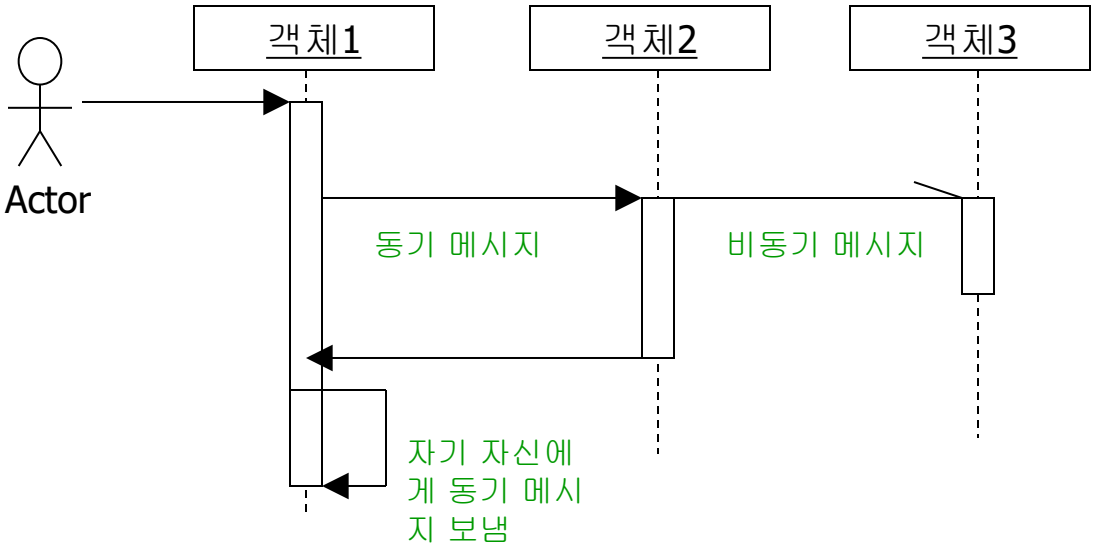
Sequence Diagram

- **시퀀스 다이어그램이란?**
  - 상태 다이어그램이 촉발 사건에 따른 단일 객체의 상태 변화를 표현한 것이라면, **시퀀스 다이어그램은 여러 객체들이 시간 경과에 따라 객체 상호간 교류 과정을 표현한 것.**
  - 구성
    - 객체(Object) : 사각형으로 나타내며 이름에 밑줄이 들어가 있다.
    - 메시지(Message) : 실선 화살표로 그려진다.
    - 시간 : 진행 상황을 나타내기 위하여 수직선으로 그린다.
  - 객체
    - 시퀀스 다이어그램의 가장 위 부분에 위치, 왼쪽에서 오른쪽으로 배열.
    - 배열순서 – 다이어그램을 간략하게 하는 방향으로 기준을 삼는다
    - 각 객체로부터 아래로 뻗어 가는 점선은 객체의 생명선(lifeline)이라 불린다.
    - 생명선을 따라 좁다란 사각형이 나타나는데, 이 부분은 실행(activation)이라 한다.즉, 객체가 수행되고 있음을 나타낸다. 사각형의 길이는 오퍼레이션의 실행 소요 시간을 나타낸다.



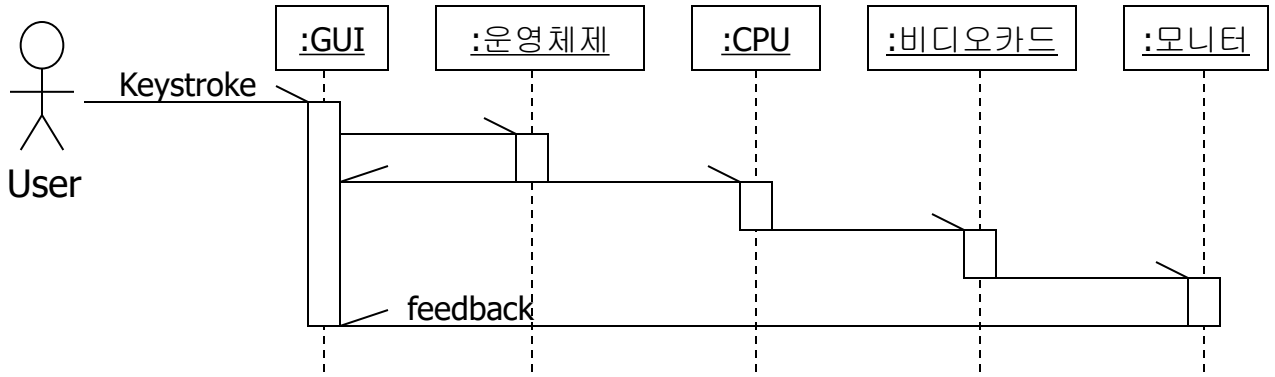
Sequence Diagram – 2

- 시간
  - 시간을 수직 방향으로 표현
  - 시간의 흐름은 위에서 아래로 흐른다.



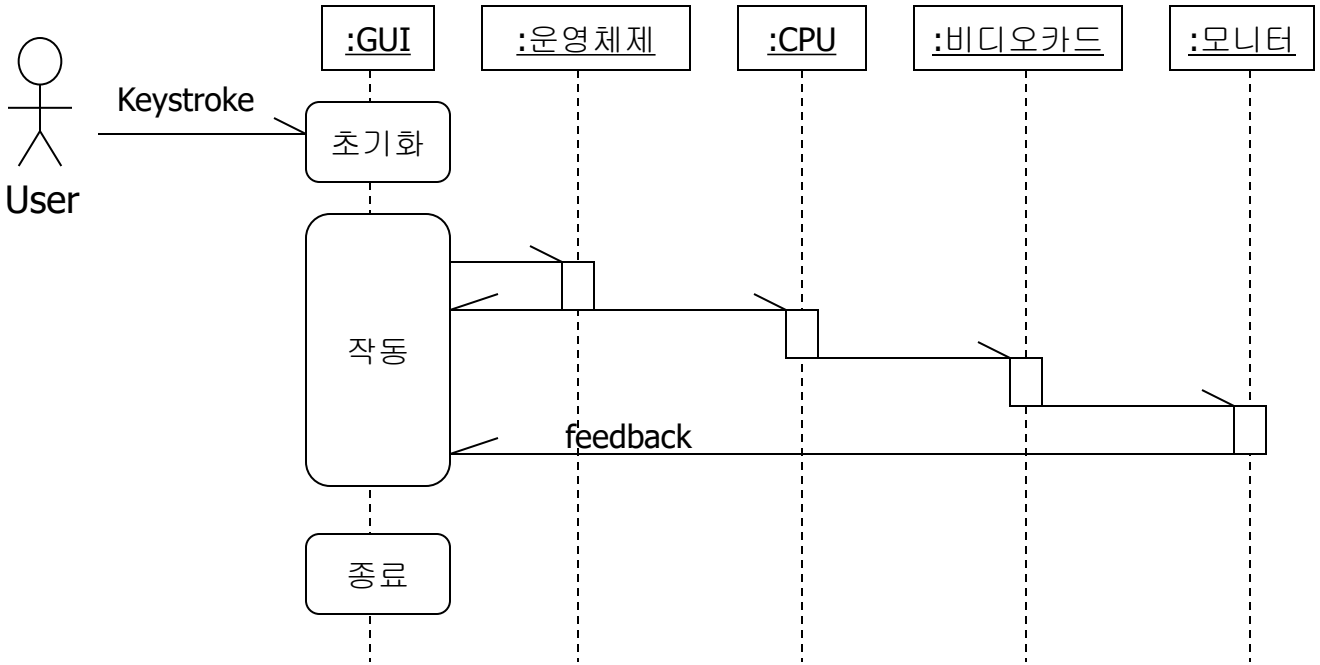
Sequence Diagram – 3

- 예제) State Diagram의 GUI 시스템 예에서 GUI 가 다른 객체들과 어떻게 교류를 하고 시간에 따라 어떻게 작동하는지를 알아보자.
  - 1. GUI는 키 입력을 운영체제에게 알린다.
  - 2. 운영체제는 CPU에게 그 사실을 알린다.
  - 3. 운영체제는 GUI를 갱신한다.
  - 4. CPU는 비디오 카드에게 GUI 갱신에 필요한 명령을 내린다.
  - 5. 비디오 카드는 모니터로 메시지를 전송한다.
  - 6. 모니터는 화면에 Alpha Numeric 문자를 표시하고, 사용자에게 피드백을 제공한다.
  - 위 과정을 Sequence Diagram으로 옮겨보자.



Sequence Diagram – 4

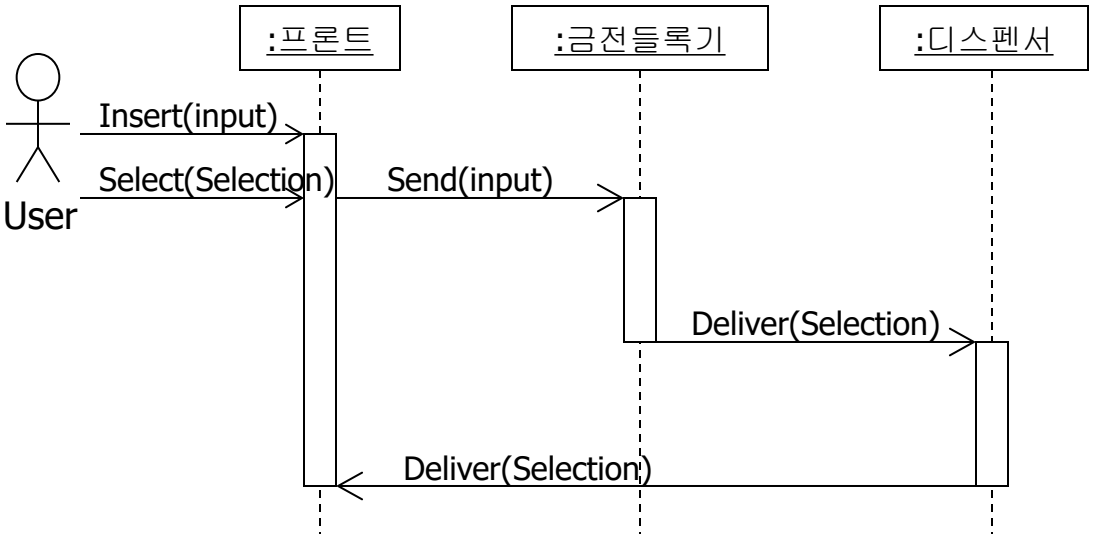
- 예제) State Diagram에서 본 GUI 객체 State Diagram을 포함한 Sequence Diagram을 그려보자.



Sequence Diagram – 5

- 예제) Use Case Diagram에서 Sequence Diagram 접근한 예이다. 쓰임새 편에서 나온 "음료수 사기" 의 예를 Sequence Diagram에 적요하여 보겠다.
  - 객체를 골라내면
    - 프론트 : 음료수 자판기가 고객과 대화하는 유일한 인터페이스, 판매기 앞판에 있음.
    - 금전 등록기 : 돈을 체크하고 등록함.
    - 디스펜서 : 음료수를 따르고 내어줌.
  - 처리과정을 써보면 다음과 같다.
    - 1. 소비자가 자판기의 프론트 앞에 서서 투입구에 돈을 넣는다.(Insert)
    - 2. 소비자가 마실 음료수를 고른다.(Select)
    - 3. 돈이 금전등록기에 들어간다.(Send)
    - 4. 등록기는 선택된 음료가 디스펜서에 들어 있는지를 체크한다.
    - 5. 가장 간단한 시나리오이기 때문에, 선택된 소다가 준비되어 있고, 등록기는 현금 잔고를 갱신한다.
    - 6. 등록기는 디스펜서를 사용하여 소다를 자동 판매기의 프론트로 보낸다.
  - 이 예는 "음료수 사기" 쓰임새에서 단지 한 개의 시나리오(즉, 하나의 인스턴스)에 대하여 그려지기 때문에, 이 다이어그램을 ***인스턴스 시퀀스 다이어그램***이라고 부른다.

Sequence Diagram – 6

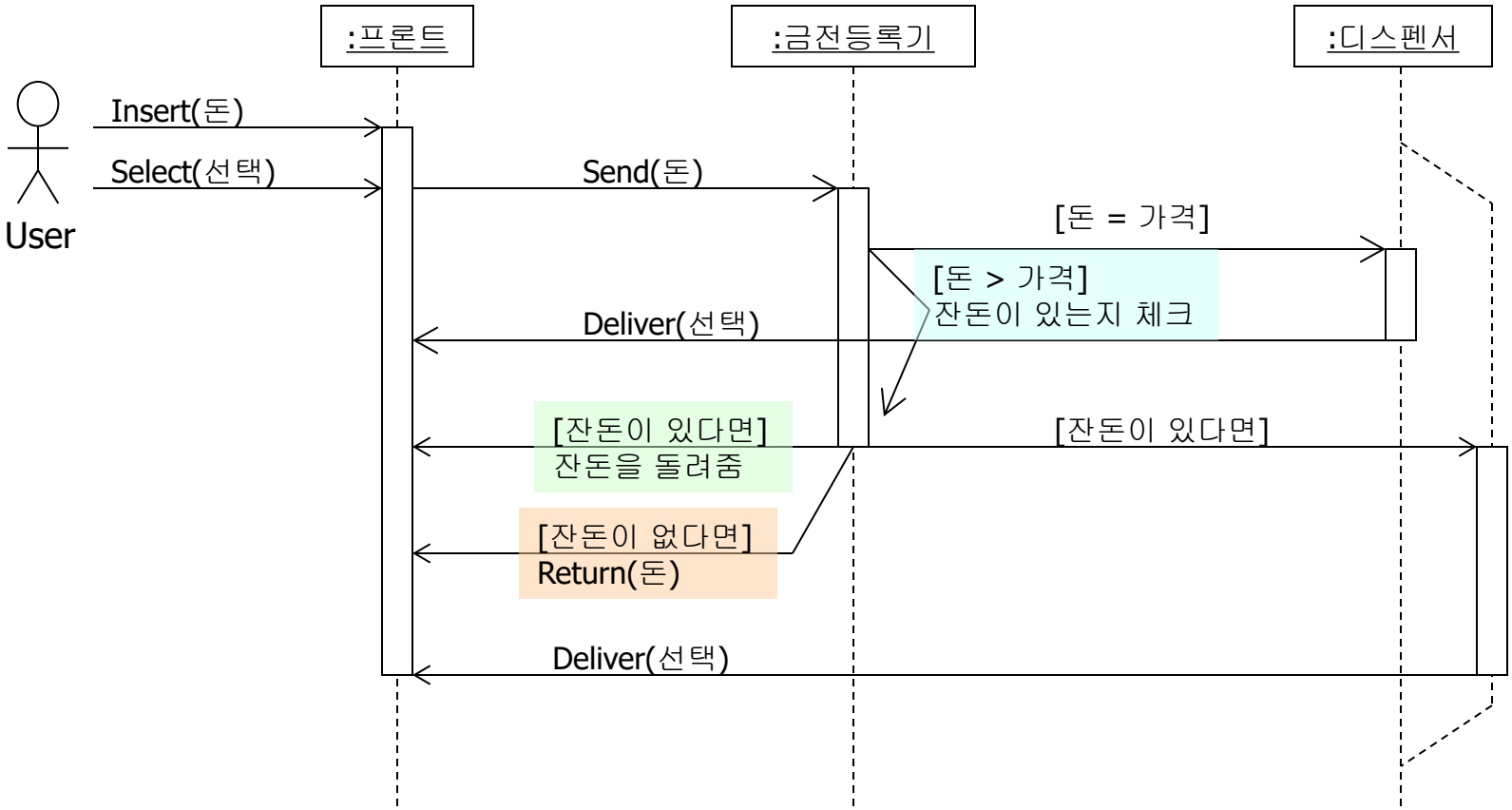


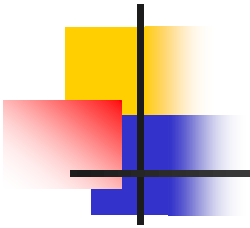
Sequence Diagram – 7

- 예제) “음료수 사기” 쓰임새에서 우리는 또 다른 시나리오를 가정할 수 있다. 선택된 음료수가 다 떨어진 경우 또는 소비자가 넣은 돈이 음료수 값과 맞지 않을 때. 이런 모든 경우를 하나의 시퀀스 다이어그램에 표현하고자 할 때 ***일반 시퀀스 다이어그램***을 그리면 된다. 다음은 **‘액수에 맞지 않는 경우’** 의 시나리오 이다.
  - 1. 등록기는 소비자가 투입한 돈의 액수(input)가 음료수의 값(price)과 맞는지 체크한다.
  - 2. 만일 액수가 음료수 값보다 많으면, 등록기는 차액을 계산하고 그 만큼의 현금 잔고가 있는지 체크한다.
  - 3. 만일 차액 만큼의 현금이 잔고(cash reserve)에 남아 있다면, 등록기는 거스름돈(change)을 내어주고 나머지 동작은 그전과 똑같이 진행한다.
  - 4. 만일 차액 만큼의 현금이 잔고에 남아 있지 않으면, 등록기는 소비자가 투입한 돈을 그대로 돌려주고 “맞는 액수의 돈을 넣어 주세요”란 메시지를 표시한다.
  - 5. 만일 소비자가 투입한 돈의 액수가 음료수 값보다 적으면, 등록기는 아무것도 하지 않고 돈이 더 들어올 때까지 대기한다.
  - 조건문을 표현하기 위해서는 대괄호 [ ] 안에 조건을 써주고, 이것을 메시지 화살표 위에 놓으면 된다.

Sequence Diagram – 8

- "액수가 맞지 않는 경우" 의 시나리오





마무리

- 요구사항수집(requrirment gathering)→분석(Analysis)→설계(Design)
- →개발(Development)→배치(Development)의 과정을 진행하면서, 각각의 다이어그램은 시스템분석과 설계의 산출물이라고 할수 있습니다.
- 설계, 분석이 잘 이루어지고, 각각의 다이어그램이 유기적인 관계를 맺었을 때, 고객이 원하는 시스템을 구축할 수 있습니다. 따라서 다이어그램을 유출해내는 과정은 5개의 과정에 있어서 중요한 비중을 차지합니다.