

Unified Modeling Language

Part 1. UML이란?

(1)왜 UML이 필요할까?

UML은 오늘날의 객체지향 시스템 개발 분야에서 가장 각광받는 도구 중 하나이다. Why? UML은 시스템 개발자가 자신의 비전(Vision)을 구축하고 반영하는데 있어서 표준적이고 이해하기 쉬운 방법으로 할 수 있도록 도와주며, 자신의 설계 결과물을 다른 사람의 효과적으로 주고 받으며 공유할 수 있는 매커니즘을 제공하기 때문이다.

UML이 있기전 과거 시스템 개발

UML이 있기 전 시스템 개발은 “때려 맞추어서 운 좋으면 성공하고, 틀리면 망한다”는 명제였다. 시스템분석가는 의뢰인의 요구사항을 받아들이고 적당히 조정하고, 분석가들이 이해할 수 있는 일정한 표기법을 사용하여 요구사항을 명세화하며(하지만 고객은 알 길이 없다.) 분석 결과를 프로그래머 혹은 개발팀에게 넘긴 다음 최종 제품이 제발 의뢰인이 원하는 것으로 만들어지길 기도하기 일쑤였다.

-> 시스템 개발은 인간 생활과 똑같기 때문에, 어느 단계에서든지 오류가 발생할 가능성은 다분히 존재한다. 분석가는 의뢰인의 요구를 잘못 이해할 수 있으며, 의뢰인의 이해할 수 없는 문서를 만들어낼 수도 있다. 프로그래머가 분석결과를 명쾌하게 이해하지 못하고 사용하기 어려운 프로그램을 만들어 냈으므로, 의뢰인의 문제를 제대로 해결해 주지 못할 수도 있다.

(2)UML의 중요성


- 오늘날에는 치밀한 사고와 기획만이 살아 남는다. 요즘 시스템 개발 추세는 전체 개발 기간을 짧은 시간 간격(TimeFrame)으로 나누는데 있다. 각각의 시간 간격 끝에 작업기간이 놓여지면 이제 완전한 설계가 뒤를 따를 수밖에 없다. 시스템 개발에 참여하는 분석가, 의뢰인, 프로그래머, 그 외의 모든 이들이 이해하고 동의할 수 있는 방법으로 설계과정을 조직화 하는 것이다. UML은 바로 이 조직화 수단을 제공하기 위해 준비된 것이다.

- 한 회사가 다른 회사를 받아들였을 때 새 조직이 진행중인 개발 프로젝트 중요한 부분을 바꿀지도 모르는 일이다. (구현 도구, 코딩언어 등), 이 때 집행부의 집중사격을 막을 수 있는 완전한 설계 계획이 문서로 구비되어 있으면 업무변경이 부드럽게 이루어 진다. 탄탄한 설계에 대한 필요성은 바로 디자인 표기(Design Notation)에 대한 필요성을 낳았다. 전자 회로를 체계적으로 그려 둔 도면을 엔지니어가 읽을 수 있듯이 분석가, 개발자, 의뢰인이 표준으로 받아들일 수 있는 디자인 표기이다. UML은 바로 그 표기이다.

(3)UML의 구성요소

-UML의 여러 가지 그래픽 요소는 하나의 큰그림, 즉 다이어그램을 그리는데 사용된다. UML은 언어이기 때문에, 이들 그래픽 요소들을 맞추는 데에는 규칙이 필요하다.

-다이어그램의 목적은 시스템을 여러 가지 시각에서 볼 수 있는(View)를 제공하는 것이며, 이러한 뷰의 집합을 모델(Model)이라고 한다. 시스템의 UML모델은 건물을 짓는 건축가의 스케일 모델과도 비슷하다.

 UML모델은 시스템 자체의 “목적행동”을 설명하는 언어라는 점과 UML모델은 시스템의 “구현 방법을 설명하는 수단”이 아니라는 점을 꼭 명심해야한다.

3-1) 클래스 다이어그램

- 객체지향 기술은 여러분의 주변 상황과 가끔 흡사하다. 대부분의 사물은 자기만의 속성(Attribute)과 일정한 행동(Behavior) 수단을 가지고 있다. 이러한 행동을 오퍼레이션(Operation)의 집합으로 생각할 수 있을 것이다.

택시는 탈 것, 의자는 가구 범주에 넣을 수 있듯이, 사실 우리 주변에서 발견할 수 있는 대부분의 것들은 어떠한 범주(Category)에 넣을 수 있다. 이러한 범주를 클래스라고 한다.

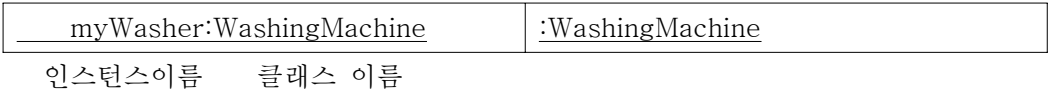
☞ 클래스란 ? 비슷한 속성과 공통적인 행동 수단을 지닌 것들의 범주 혹은 그룹을 일컫는다. 세탁기의 클래스가 있다고 가정하고 예를 들어보자. 이클래스의 속성은 브랜드의 이름, 모델, 일련번호, 용량 등이고, 이클래스의 행동은 “ 옷을 넣는다, 세제를 뿌린다, 켜다, 끄다” 등일것이다

WashingMachine	<<클래스 이름	☞ UML에서는 두 단어이상으로 이루어진 클래스 이름은 단어 사이의 공백을 없애고, 각 단어의 처음 문자를 모두 대문자로 한다. 속성과 행동의 이름 또한 마찬가지이지만, 가장 앞단어의 처음문자는 소문자로 한다. 행동을 나타내는 이름 뒤에 소괄호()가 붙는 이유는 매개변수 리스트를 넣어줄 수 있는 공간이다.
brandName modelName serialNumber capacity	<<속성을 넣는 공간	
acceptClothes() acceptDetergent() turnOn() turnOff()	<<오퍼레이션공간(행동)	

<UML 클래스 아이콘>

3-2)객체 다이어그램

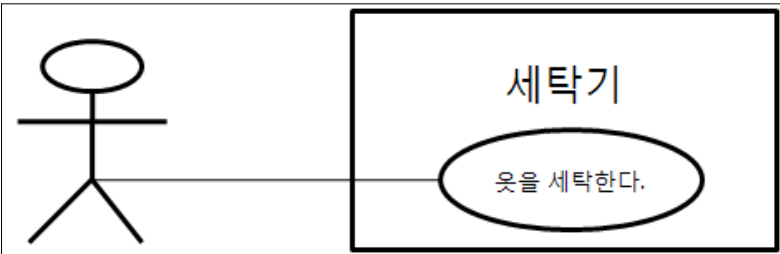
- 객체(Object)란, 클래스의 인스턴스 즉, 값이 매겨진 속성과 행동을 가지고 있는 개별적인 개체를 일컫는다. ex)세탁기로 예를 들면 “정보전자”라는 브랜드명과 “빨빨래세탁기”라는 모델명 그리고 “GL57774”라는 일련번호와 ”16파운드“라는 용량을 가진 하나의 세탁기가 객체가 될 수 있다.



☞ 객체를 UML로 나타낸 그림이다. 아이콘자체는 클래스와 똑같이 사각형이지만, 이름에 밑줄이 그어져 있다. 인스턴스의 이름은 콜론(:)의 왼편에 쓰며, 클래스의 이름은 콜론의 오른편에 쓴다. 인스턴스의 이름은 소문자로 시작하고 그림 오른쪽처럼 **이름의** 객체도 가능하다. 객체가 속해 있는 클래스를 보여주는 것에만 중점을 둬으로써, 특정 이름을 지정해 주지 않는 것이다.

3-3)유스 케이스 다이어그램

- 유스 케이스(use case)는 사용자의 입장에서 본 시스템의 행동을 일컫는다.

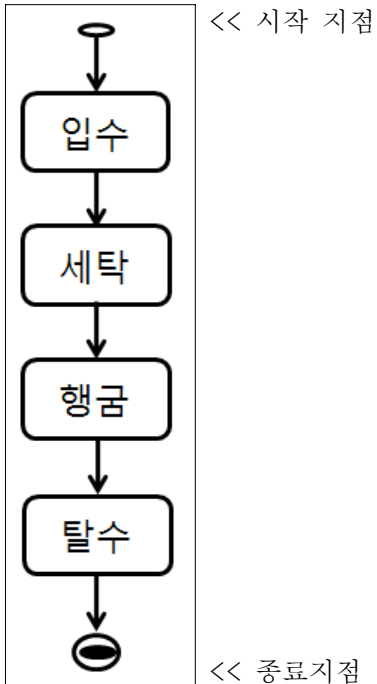


☞ 현재 우리가 세탁기를 사용하는 이유는 옷을빨다(wash cloths)위해서 인데 이것을 UML로 유스케이스 다이어그램을 그려보았다. 세탁기가 사용자를 나타내는 막대 인간 그림을 행위자(actor)라고 한다. 타원은 유스케이스(usecase)를 나타낸다. 행위자(유스 케이스와 대화를 시작하는 개체) - 사람 혹은 다른 시스템이 될 수 있다. 또한, 유스케이스가 시스템을 의미하는 사각형 내에 있고, 행위자는 사각형 바깥에 있음을 유의하여 보

기 바란다.

3-4)상태 다이어그램

- 객체는 시간에 따라 각기 다른 상태에 있을수 있다. 인간도 마찬가지이다. 갓난아기로 태어나 영/유아가 되고, 어린이가 되고, 10대가 되고, 성인으로 자란다. 엘리베이터는 올라갔다, 쏘다가, 내려갈 수 있다. 세탁기는 물을담고(soak), 세탁하고(wash), 행구고(rinse), 돌리고(spin), 정지(stop) 될수 있다.



이 그림의 상단에 있는 기호는 시작상태(start state), 하단에 있는 기호는 종료 상태(end state)를 나타낸다.

3-5)시퀀스 다이어그램

- 클래스 다이어그램과 객체 다이어그램을 정적인 정보를 나타낸다. 하지만 특정행동을 행하는 시스템에서는 여러 개의 객체들이 서로 메시지를 주고 받으며 작업을 진행하는 것이 보통이다. UML시퀀스(Sequence)다이어그램은 객체들끼리 주고 받는 메시지의 순서를 시간의 흐름에 따라 보여주는 그림이다.

- 세탁기를 가지고 예를 들어보자 세탁기는 타이머, 입수관(물을 넣기 위한 관), 드럼(빨래 담는 통) 등으로 구성되어 있다. 물론 이것들은 모두 객체이다. (객체는 다른 여러 개의 객체들로 구성될 수 있다.)

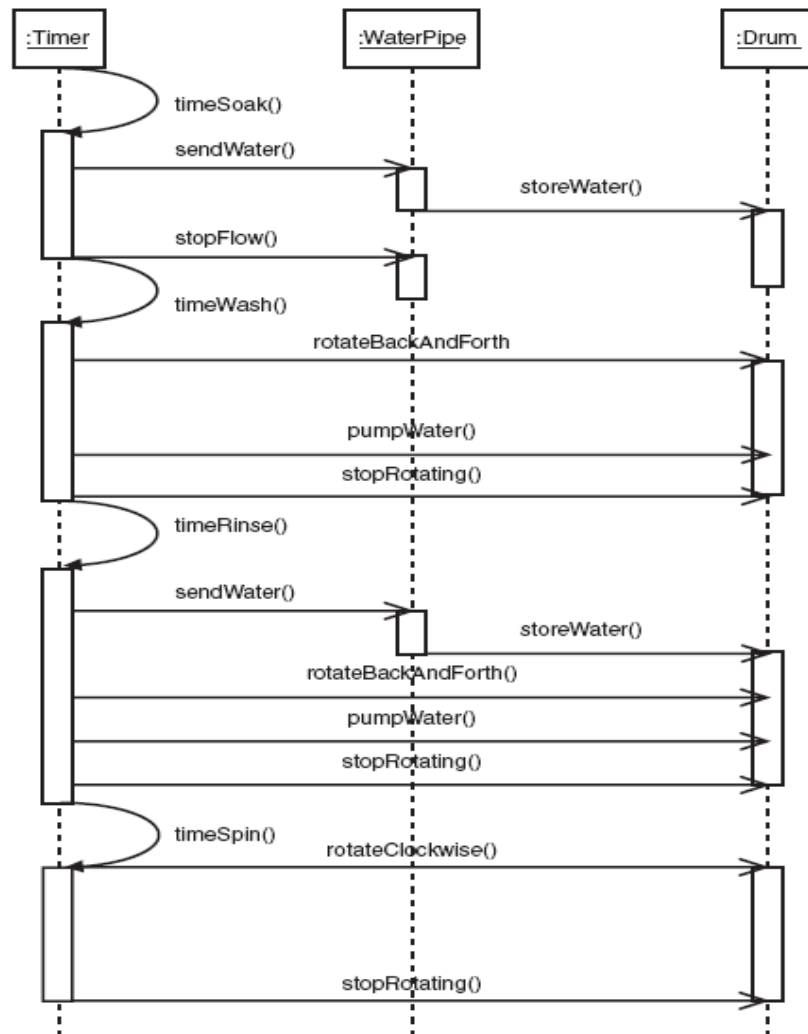
- 이제 세탁기 유스 케이스를 작동 시켰을 때 어떤 일이 일어날까?“ 옷을 넣는다, 세제를 뿌린다, 그리고 켜다”의 오퍼레이션은 이미 마쳤다고 가정하면, 이 유스 케이스에서는 다음의 행동이 단계별로 이루어질 것이다.

1. 입수단계의 시작으로써 물이 입수관을 통해 드럼으로 들어간다.
2. 드럼은 5분 동안 정지된 상태를 유지한다.
3. 입수단계의 끝으로써 물이 들어가다 멈춘다.
4. 세탁단계의 시작으로써 드럼이 앞, 뒤로 15분간 회전한다.
5. 세탁단계의 끝으로써 세제와 때가 섞인 물이 배수관을 통해 나온다.
6. 드럼의 회전이 멈춘다.
7. 행굼 단계의 시작으로써 물이 다시 들어간다.
8. 드럼은 다시 앞뒤로 회전한다.
9. 15분 후에 물이 들어가다 멈춘다.
10. 행굼단계의 끝으로써 물이 배수관을 통해 나온다.
11. 드럼의 회전이 멈춘다.
12. 탈수단계의 시작으로 드럼이 시계 방향으로 회전하기 시작하면서 5분동안 가속한다.
13. 탈수 단계의 끝으로써 드럼의 회전이 멈춘다.
14. 세탁이 종료된다.

- 타이머, 입수관, 드럼을 각각 객체라고 해보자. 각 객체는 하나 이상의 행동을 하면서 서로에게 메시지를 보내며 상호적으로 작동하게 된다. 메시지는 “보내는 객체”로부터 “받는 객체”로 전달되는 하나의 요청이라 볼 수 있다. “받는 객체”가 할 수 있는 행동 중에서 어느 하나를 수행해 달라는 의미로 부탁(?) 되는 것이다.

타이머	입수관	드럼
입수의 시간을 쟀다.	물을 흘려 보낸다.	물을 저장한다.
세탁의 시간을 쟀다.	물을 막는다.	앞, 뒤로 회전한다.
행굼의 시간을 쟀다.		시계 방향으로 회전한다.
탈수의 시간을 쟀다.		

시간의 흐름에 따른 입수관, 드럼, 배수관 사이의 상호 대화를 나타낸 시퀀스 다이어그램이다.

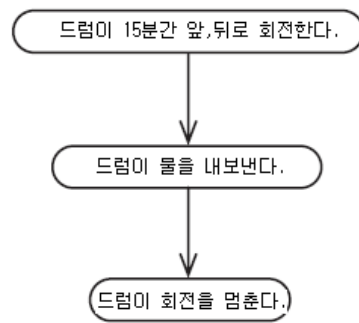


3-6)활동 다이어그램

- 유스 케이스 내부 혹은 객체의 동작중에 발생하는 활동(activity)은 대개 시퀀스 내에서 발견할 수 있다.

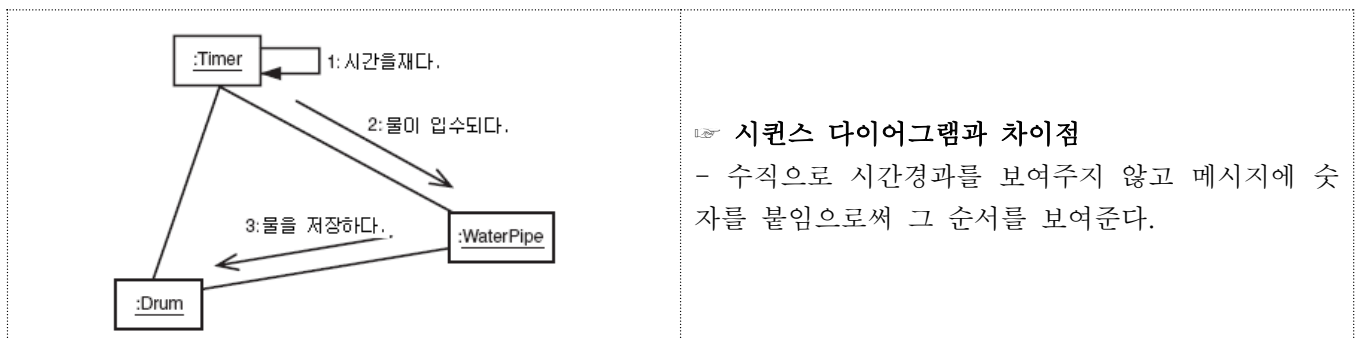
4. 세탁단계의 시작으로써 드럼이 앞, 뒤로 15분간 회전한다.
5. 세탁단계의 끝으로써 세제와 때가 섞인 물이 배수관을 통해 나온다.
6. 드럼의 회전이 멈춘다.

앞에서 설명한 부분중 4단계부터 6단계까지를 활동 다이어그램으로 그려보았다.



3-7)통신 다이어그램

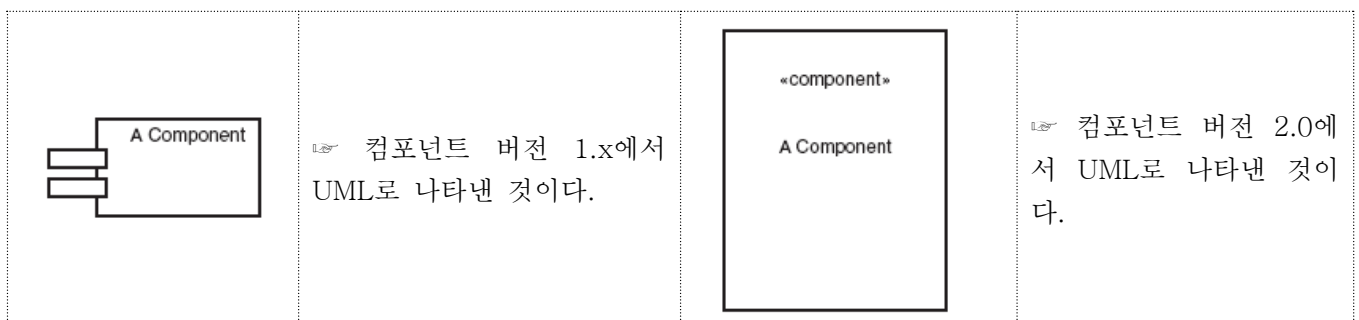
- 하나의 시스템을 구성하는 요소들은 다른 요소들과 손발을 맞추면서 시스템 전체의 목적을 이루어 나가는 것이 보통이기 때문에, 모델링 언어는 이것을 표현할 수 있어야한다. 앞서 말한 시퀀스 다이어그램이 그것이고, UML통신(commuication) 다이어그램 또한 이러한 목적을 위하여 디자인 된 것으로 아래 그림에서 그 예를 보여준다. 하지만, 시퀀스 다이어그램과는 어느정도 다른점이 있다. 시퀀스 다이어그램에서 타이머와 입수관, 드럼 사이에 오가는 메시중에 초반의 것들만을 뽑아내어 보여주고 있는데, 보다시피 위에서 아래, 즉 수직으로 시간경과를 보여주는 것이 아니고, 메시지에 숫자를 붙임으로써 그 순서를 보여 주고 있다.
- 시퀀스 다이어그램과 협력 다이어그램은 객체 사이의 상호관계를 나타내는 다이어그램이기 때문에 UML에서는 이 둘을 통합적으로 “**교류(interraction)다이어그램**” 이라고 부른다.



Tip) 통신다이어그램의 이름은 버전 2.0에서 처음 생겨난 것이다. 버전1.x에서는 **협력(collaboration)다이어그램**이라고 불렀다.

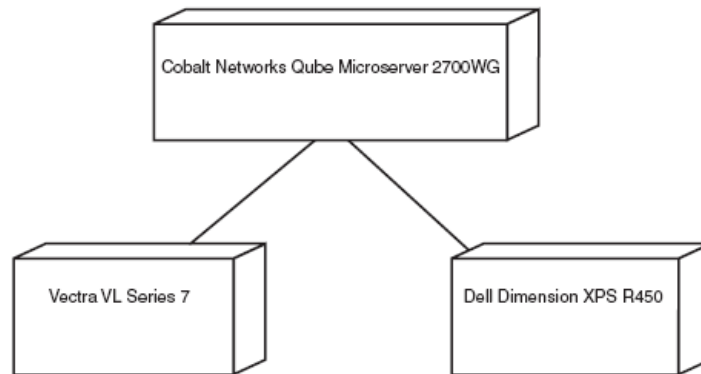
3-8)컴포넌트 다이어그램

- 컴포넌트 다이어그램과 배포 다이어그램은 컴퓨터 시스템을 명확하게 나타낼 수 있도록 준비된 것이다.
- 현대의 소프트웨어 개발 추세는 컴포넌트 중심으로 되어가고 있기 때문에 팀 단위 수행하는 프로젝트라면 특히 중요한 컴포넌트이다.



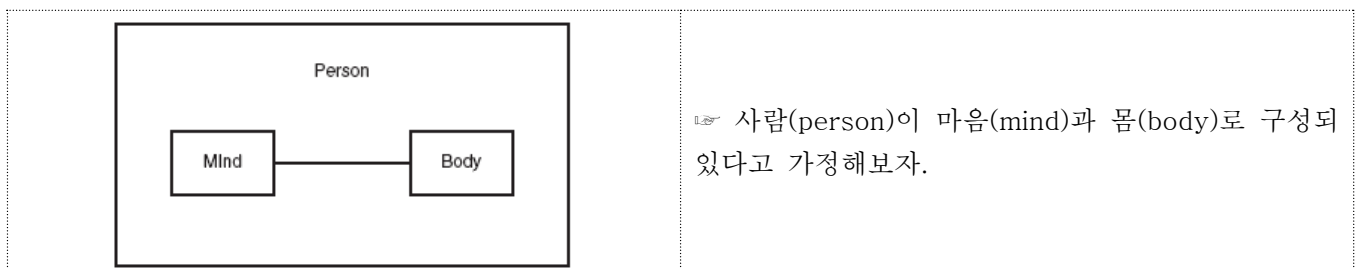
3-9)배포 다이어그램

- UML 배포(deployment) 다이어그램은 컴퓨터를 기반으로 하는 시스템의 물리적 구조를 나타낸 그림이다. 이 다이어그램은 컴퓨터와 부가장치, 그리고 각각의 연결 관계뿐만 아니라, 각각의 기계에 설치된 소프트웨어 까지 표시 해준다. 컴퓨터는 큐브로 나타내며, 컴퓨터 사이의 연결관계는 큐브와 큐브 사이를 선으로 이어줌으로써 나타낸다.



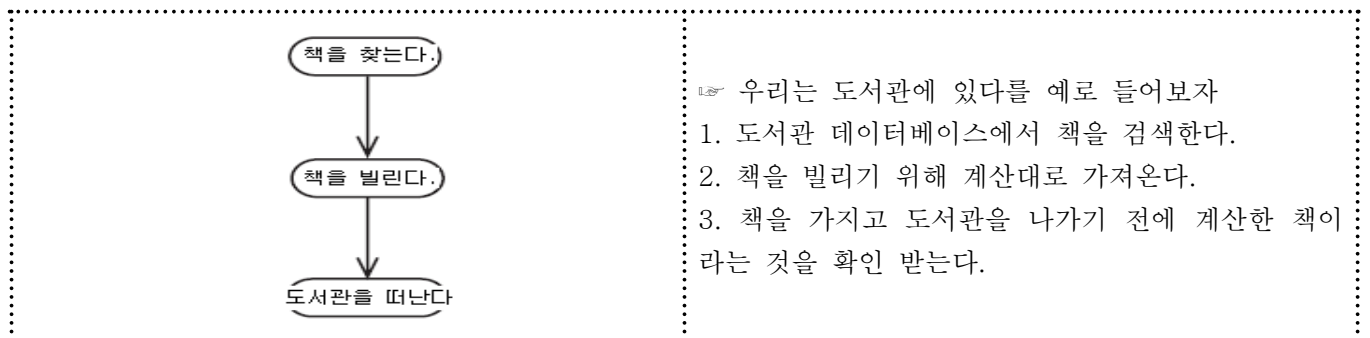
3-10)복합체 구조 다이어그램

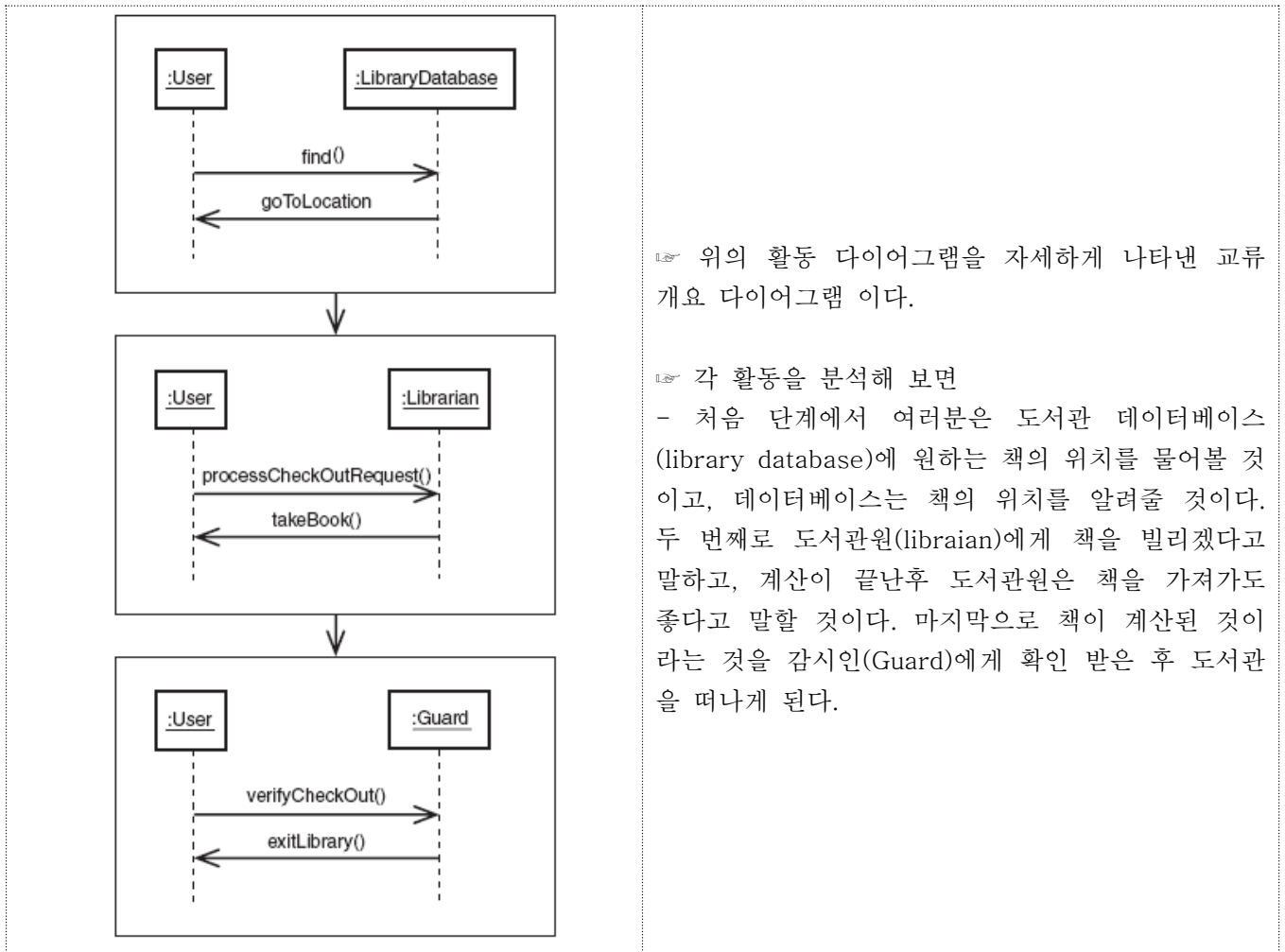
- UML2.0의 복합체 구조 다이어그램(composite structure diagram)에는 각 컴포넌트 클래스를 전체 클래스 안에 위치시킴으로써 넓이의 개념을 포함한다. 클래스로만 국한되던 시야를 전체 구조로 넓혀보도록 하자.



3-11)교류 개요 다이어그램

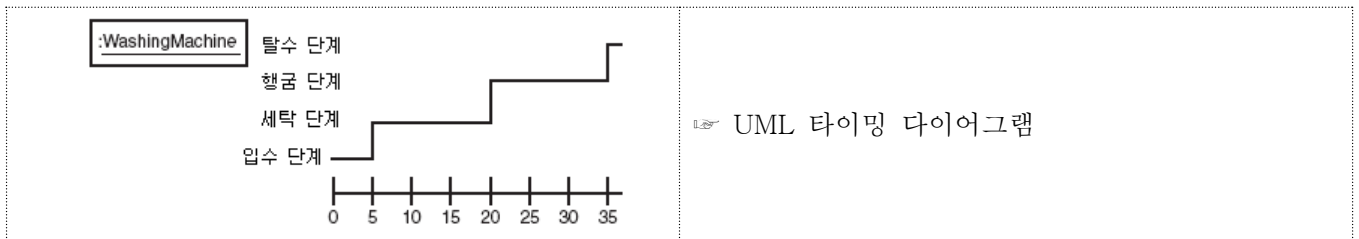
- 활동다이어그램의 경우 하나의 단계가 하나의 “활동”이 된다. 각 활동마다 객체 사이에 시간의 흐름을 갖는 메시지가 존재한다고 가정한다고 보겠다. 그렇다면 몇몇 활동 부분은 시퀀스 다이어그램이나 통신 다이어그램 (혹은 두 다이어그램의 조합)으로 바뀌어야 할 것이다. 이것이 UML 2.0에서 새로 생긴 교류 개요 다이어그램(interaction overview diagram)이다.





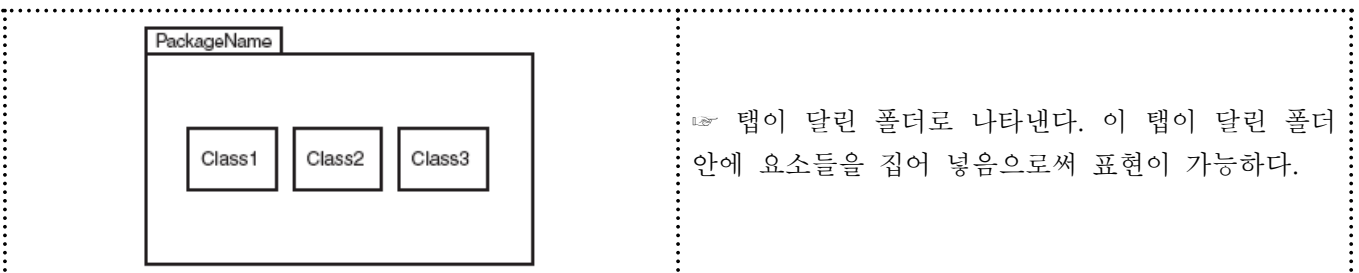
3-12)타이밍 다이어그램

- UML2.0의 타이밍 다이어그램에서는 시간을 다룰 수가 있다. 한 상태에서 객체가 얼마나 오랜 시간을 지체하는지를 명시하는데 타이밍 다이어그램이 사용된다.



3-13)패키지 다이어그램

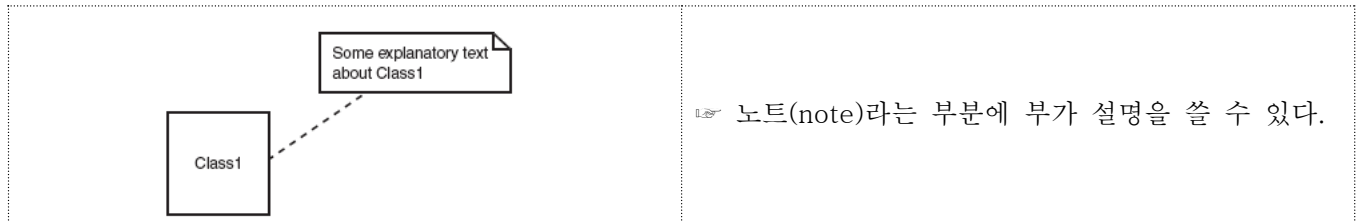
- 버전 1.x버전에서도 다이어그램의 요소를 조직화하는 기능이 존재한다. 패키지(package)라고 부르는 것이다



3-14)UML 다이어그램을 조직화하고 확장할 수 있는 장치

1) 노트

- 다이어그램을 그리다 보면 “왜 이것이 필요한지” 혹은 “어떻게 작동하는지” 에 대한 명확한 설명을 하고 있지 않는 부분이 생길수 있다. 이 경우 **UML노트(note)**를 사용하면 된다.



2)키워드와 스테레오타입

- **스테레오타입(stereotypes)**은 기존의 UML 요소를 기본으로 하여 다른 요소를 새로 만들 수 있게 하는 장치이다.(일종의 변형이라고 생각하면 쉽다.)
- 스테레오타입은 원하는 이름을 **거듭인용표(<<>>)**로 감싸는 것으로 간단히 마무리 되며, 이렇게 된 이름을 원하는 곳에 사용하면 된다. 그리고 거듭인용표에 감싸여진 이름을 **키워드(keyword)**라고 부른다.

(4)왜이렇게 복잡하고 다양한 다이어그램이 존재 하는가 ?

- UML은 시스템을 여러 가지 시점(view)에서 점검하고 관찰할 수 있게 하는 도구이다. 모든 다이어그램이 모든 UML모델에 등장하는 것은 아니라는 점을 주목해주시기 바란다. 왜 여러 가지 시점에서 시스템을 설계하고 점검해야 할까? 대개 다른 많은 참여자(stake-holder)들이 하나의 시스템을 지켜본다. **많은 사람들이 각자의 관심에 맞추어 시스템을 바라본다는 뜻이다.**

여러분이 모터 설계자라면 모터설계자 시점에서 세탁기 시스템을 바로보고 세탁기를 조작하는 명령어를 프로그래밍하는 하는 엔지니어는 또 다른 시점에서 세탁기 시스템을 바라보고 있을 것이다. 세탁기의 외관을 디자인하는 사람은 그 세탁기가 옷을 어떻게 빨든 간에 세탁기의 성능과 전혀 관계없는 관점을 가지고 있을 것이다.

- UML모델은 시스템이 “**무엇을**” 의도하고 있는 지를 말해줄 뿐 “**어떻게**” 동작하는지를 말해 주지 않는다.

🔗 퀴즈

1. 시스템 모델링에 있어서 다이어그램을 다양하게 그려야 하는 이유는 무엇인가?
2. 시스템의 정적인 뷰를 제공하는 다이어그램에는 무엇이 있을까?
3. 시스템의 동적인 뷰를 제공하는(즉, 시간에 따른 변경 상황을 보이는) 다이어그램에는 어떤 것이 있을까?
4. 그림에 있는 객체들의 종류는 무엇인가?

Part 2. 객체지향이란?

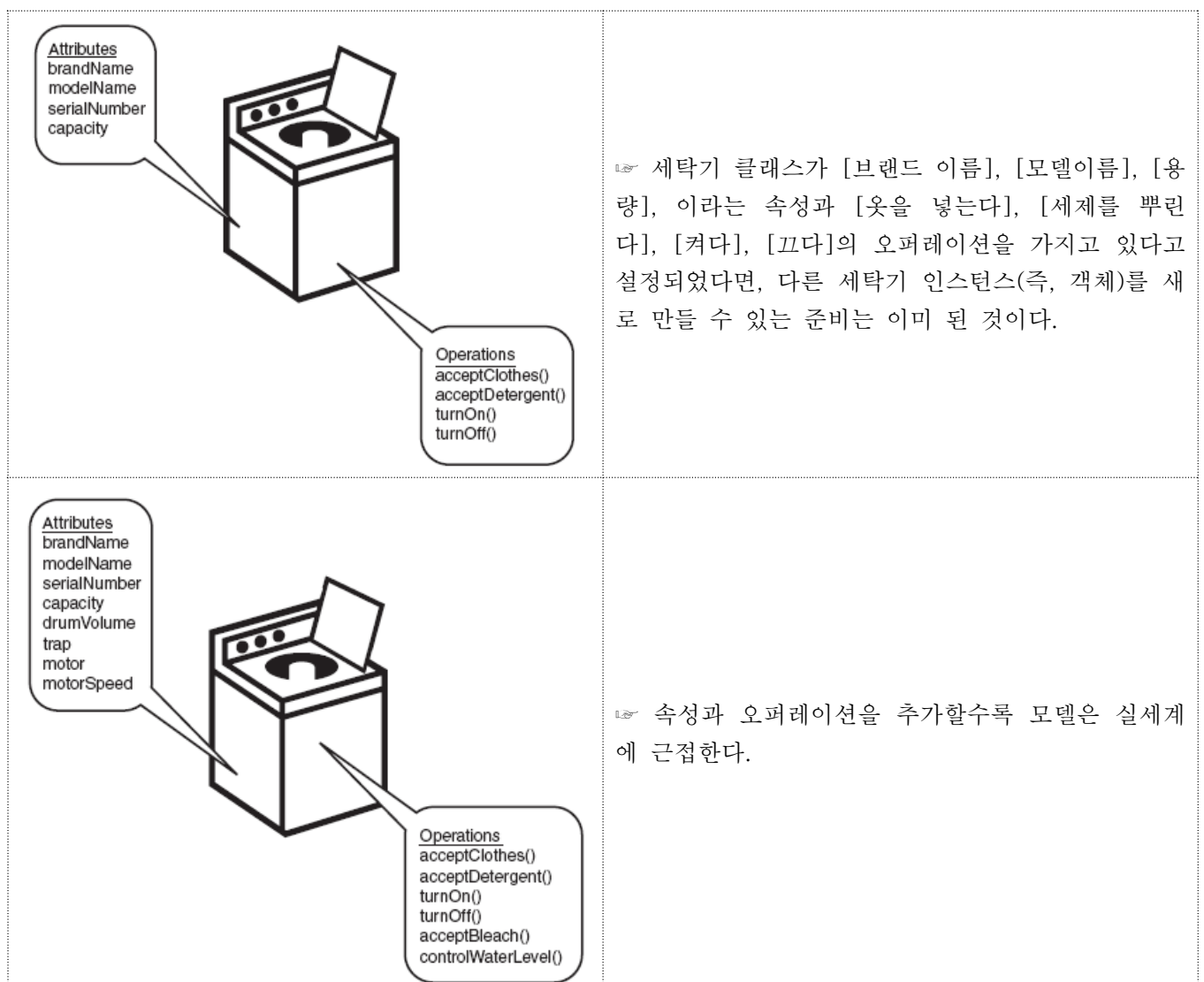
(1) 객체 지향에 대해서

- 객체는 클래스(범주)의 인스턴스이다. 예를들어, 여러분과 나는 인간(person)클래스의 인스턴스 이다. 하나의 객체 구조(structure)를 가지고 있다. 쉽게말해서, 속성과 행동을 가지고 있다는 뜻입니다. 객체의 행동은 자신이 수행하는 **오퍼레이션(operation)**으로 구성되어 있다. 속성과 오퍼레이션을 모두 합쳐서 **특성(feature)**이라고 한다.

- 여러분과 나는 인간 클래스의 객체로서 어떠한 속성을 가지고 있을까? 키, 몸무게, 나이 등이 있을 것이다. 이번에는 어떤 오퍼레이션을 수행할 수 있을지 생각해보자. 아마 먹다, 자다, 읽다, 쓰다, 말하다, 등이 있지 않을까? 만일 인간에 대한 정보를 처리하는 시스템을 만들고자 한다면 이러한 속성과 오퍼레이션 모두 이용할 수 있어야 할 것이다.

- 객체지향 세계에서 클래스는 어떤 범주를 나누는 역할 외에 하나의 목적이 더 있다. 클래스는 객체를 생성하는 틀, 혹은 템플릿(template)이기도 하다. 새로운 객체를 찍어내는 붐어빵 기계정도로 생각하면 아주 간단할 것이다.

- 객체지향의 목적은 실세계의 일부 혹은 전체를 본뜬 것(이것을 모델링이라고 한다.) 이라는 점이다. 속성과 오퍼레이션들이 좀 더 많이 반영된 클래스일수록 실세계 더 가까운 모델이 만들어진다.



(2)객체 지향을 이루는 몇 가지 개념들

2-1)추상화

- 추상화(Abstraction)란, 객체를 모델링할 때 “여러분이 필요로 하는 만큼의” 속성과 오퍼레이션을 추출해내는 것이다.

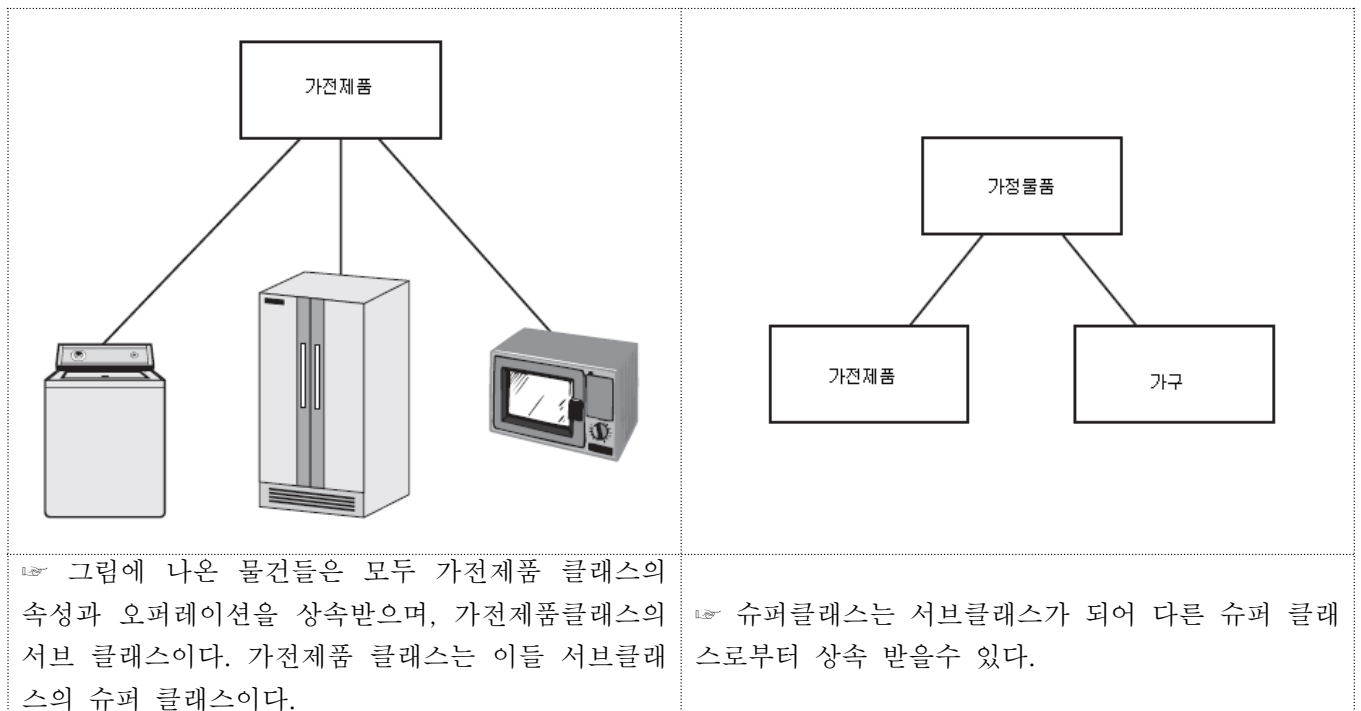
- 처음 만들었던 클래스보다 더 많은 속성과 오퍼레이션이 필요할 수 있다. 바람직한 일일까? 컴퓨터 프로그램을 개발하는 개발팀이라면 기술자들에게는 매우 유용하다. 세탁기가 제조되고, 기능을 발휘하고, 옷을 세탁할 때 어떤 과정이 필요하고, 어떤일이 진행되는 지에 대해 정확히 예측하는 용도로 사용되어도 충분하다. 이러한 용도라면 일련번호 속성을 없애도 상관이 없다. 그 이유는 전혀 소용이 없기 때문이다. 한편 세탁기를 여러 대 들여 놓고 사용하고 있는 세탁소에서 세탁물의 출입을 관리하는 소프트웨어를 개발하고 있다면 별로 바람직한 일은 아니다. 이러한 프로그램은 앞에서처럼 모든 속성과 오퍼레이션을 꼬치꼬치 캐내어 클래스에 넣지 않아도 된다. 거의 필요가 없기 때문이다. 그러나 위에서 필요 없었던 일련번호 속성이 이번에는 꼭 필요하다.

- 어떤 경우이든 설계중의 클래스에 무엇을 포함시키고, 무엇을 제외시킬 지를 결정하고 나면 세탁기의 추상화는 다 된 것이다.

2-2)상속

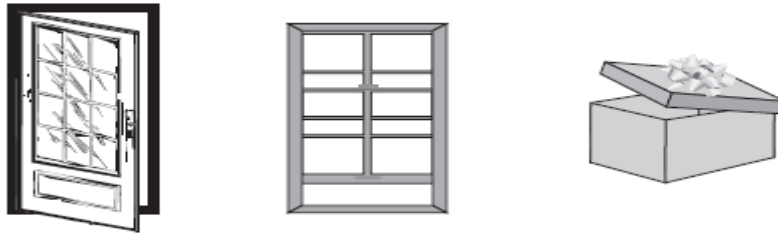
- 세탁기, 냉장고, 전자레인지, 토스터, 라디오, 다리미 등은 모두 가전제품이다. 이들은 각각 “가전제품”이라는 클래스의 **서브 클래스(subclass)**라고 말할 수 있겠다. 또한 가전제품 클래스는 각각의 제품의 **슈퍼 클래스(superclass)**가 된다.

- 가전제품 클래스는 전원 스위치, 전기배선의 속성과 켜다, 끄다의 오퍼레이션을 가지고 있다. 따라서 “어떤 것이 가전제품이다”라는 사실만 알면, “가전제품 클래스가 가진 속성과 오퍼레이션을 가지고 있구나”라고 바로 알아챌 수 있는 것이다. 객체지향에서 이러한 관계를 **상속**이라 한다.



2-3)다형성

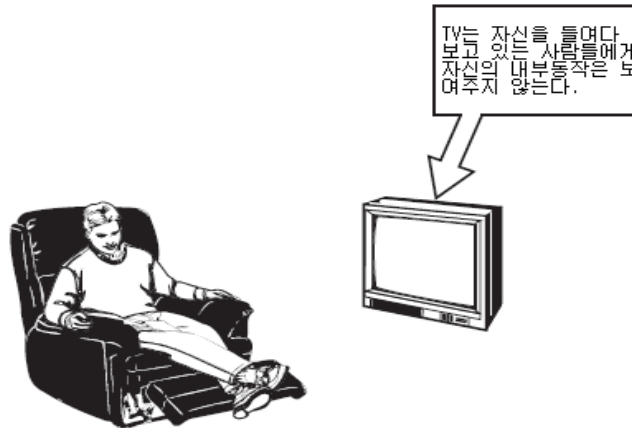
- 다른 클래스인데 같은 이름의 오퍼레이션을 가지게 되는 경우가 있다.



☞ 같은 오퍼레이션인데도 문을 열고(open door), 창문을 열고(open widow), 선물상자를 열고(open conversation)이 동사를 사용할 수 있다는 것이다.

- 행위의 대상이 모두 다르기 때문에 다른 오퍼레이션이 이루어지는 것이다. 객체지향 세계에서는 각각의 클래스마다 자신의 오퍼레이션이 어떻게 행동하는 지를 알고 있다. 이것을 **다형성(polymorphism)**이라고 한다.

2-4)캡슐화



☞ 객체는 자신의 동작 원리를 클래스라는 껍데기로 캡슐화 한다. 즉, 그 객체만이 자신의 오퍼레이션이 어떻게 작동하는 지를 알고 있으며, 외부에서는 알 수 없다.

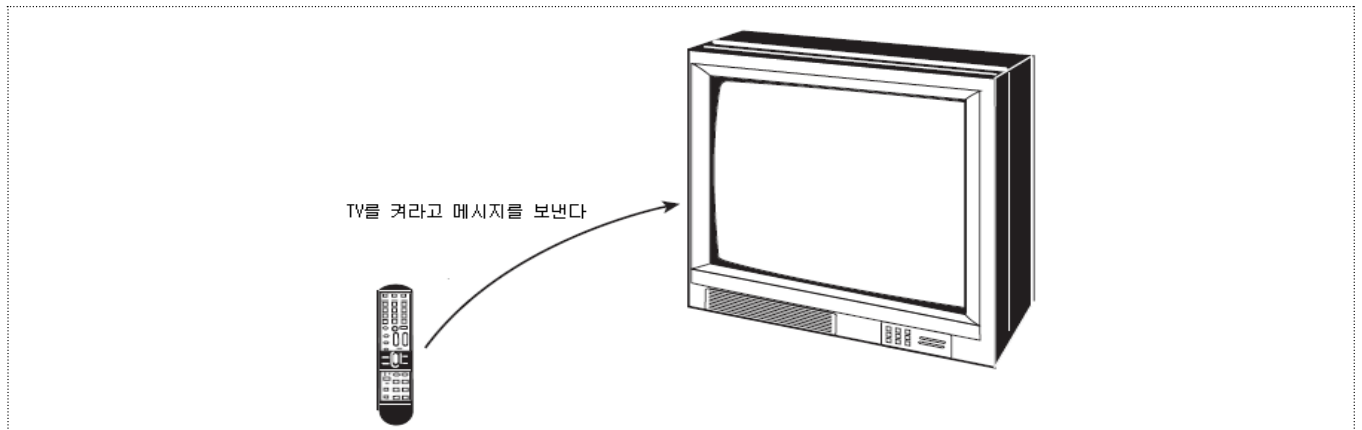
☞ 이게 바로 **캡슐화(encapsulation)**이다.

☞ 우리가 사용하는 대부분의 가전제품들이 이런 식이다.

- 왜 그럼 이 개념이 중요할까? 예로 여러분이 사용하고 있는 컴퓨터 모니터는 컴퓨터 CPU에서 일어나고 있는 모든 오퍼레이션을 숨기고 있다. 모니터에 무엇인가 잘못된 일이 생기면 모니터만 바꾸거나 고치면 그만이다. CPU까지 고칠 필요는 없는 것이다. “ 다른 객체에 자신의 오퍼레이션 동작 원리를 알리지 않는다는 사실”은 결국 “ 다른 객체를 고칠 필요가 없다” 라는 뜻이다.

2-5)메시지 전송

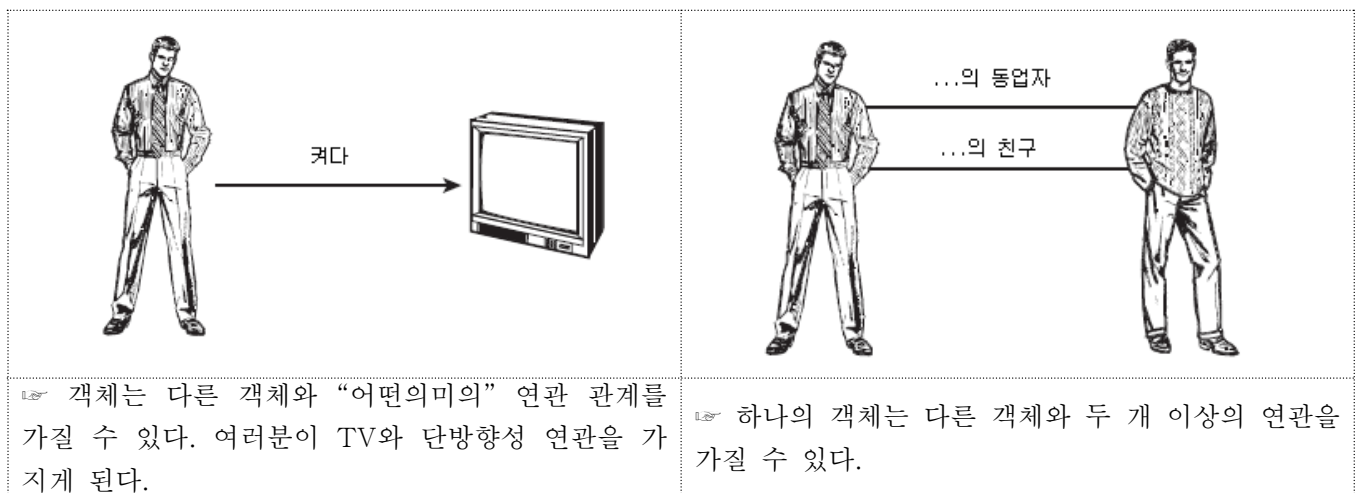
- 시스템 안에서 객체들이 서로 연결되어 행동한다고 이야기 했었다. 객체들은 어떻게 서로 발을 맞추어 행동하는 것일까? 바로 메시지(message)를 사용하는 것이다. 한 객체가 다른 객체에게 메시지를 보내어 어떤 오퍼레이션을 수행하도록 하면, 메시지를 받는 객체는 지시 받은 대로 해당 오퍼레이션을 수행하는 것이다.



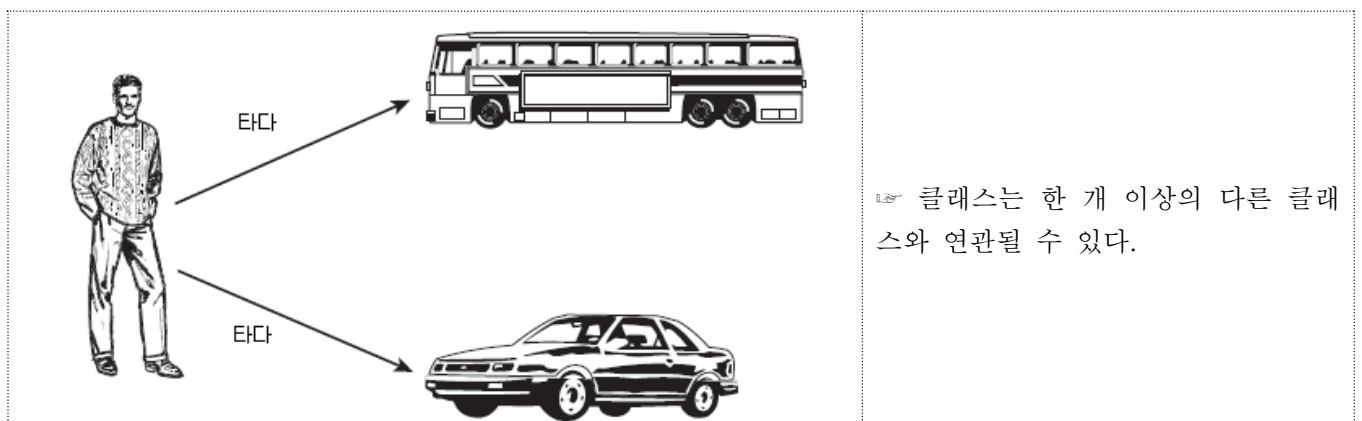
- ☞ 한 객체에서 다른 객체로 메시지를 전송하는 예, 리모콘 객체를 TV객체로 메시지를 보내어 TV를 켜도록 한다. TV객체는 자신의 인터페이스인 적외선 수신기를 통해 이 메시지를 받는다.
- ☞ 시퀀스 다이어그램에서 화살표가 한 객체에서 다른 객체로 전송되는 메시지를 의미한다.

2-6)연관

- 여러분이 TV를 켜는 일을 객체지향적으로 해석하면 여러분은 TV와 **연관(association)**관계에 있게 되는 것이다.

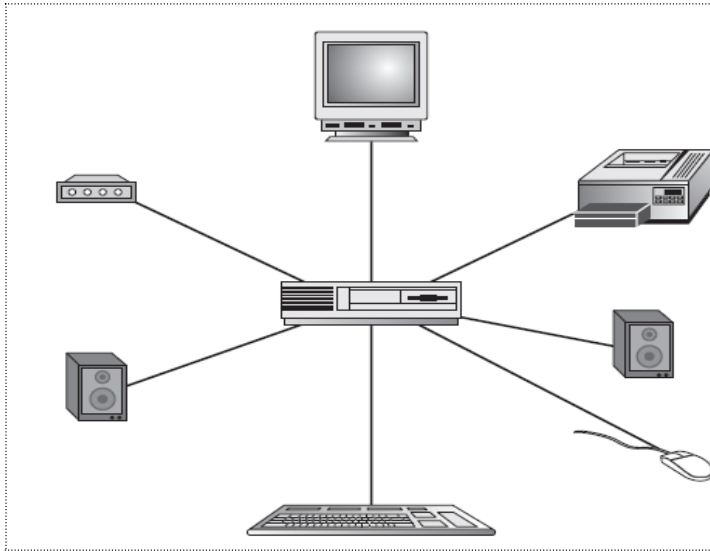


- 하나의 클래스는 한 개 이상의 다른 클래스와 연관될 수 있다. 사람은 자동차에 탈 수 있을 뿐만 아니라, 버스에 탈 수 있다.



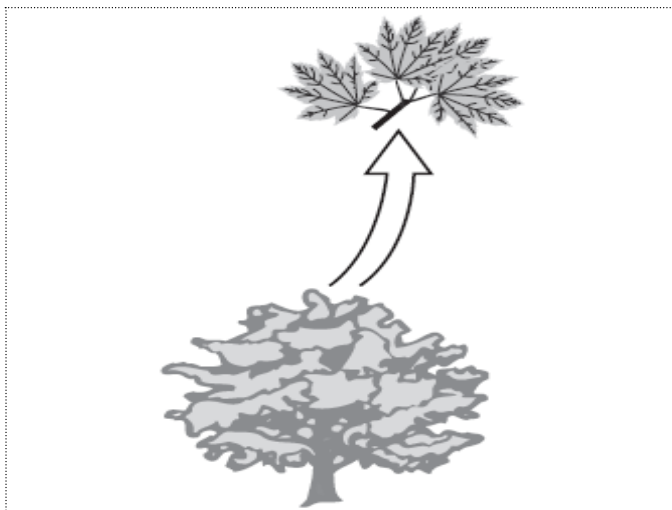
- **다중성(multiplicity)**은 객체 사이의 연관 관계를 설명할 때 매우 중요한 특징이다. 다중성은 특정한 집합내에서 하나의 객체가 몇 개의 객체와 연관될 수 있는지를 알려준다.

2-7)집합연관



☞ 컴퓨터는 또 다른 타입의 연관 관계인 집합연관(aggregation)관계에 있다. 컴퓨터는 수많은 구성요소로 이루어져 있기 때문이다.

- 집합체(aggregate)전체와 그 객체를 구성하는 컴포넌트 부분끼리 밀접한 관계를 가지는 집합 연관의 한 형태가 있는데, 이것을 **복합연관(composition)**이라고 한다. 복합연관의 열쇠는 바로 “컴포넌트가 복합체(composite)에서만 컴포넌트로써 존재”한다는 점이다. 예를 들어, 셔츠는 몸통, 칼라, 소매, 단추, 단추구멍, 소매 끝동의 복합체이다. 셔츠와 칼라를 떼어 놓으면 아무 의미도 없다.



☞ 복합연관에서는 복합체보다 컴포넌트가 먼저 죽는 경우가 있다. 하지만 복합체가 없어지면 컴포넌트도 따라 없어진다.

(3)결론은?

- 모든 시스템의 중심은 객체와 객체들간의 연관 관계라고 할 수 있다. 이들 시스템을 모델링 하기 위해서는 객체 사이의 연관 관계를 꼭 이해해야 한다. 연관 관계가 어떻다는 것을 잘알고 있다면 의뢰인과 이야기하여 요구사항을 정리할 때 한층 세심하고 편하게 진행할 수 있을뿐 아니라, 의뢰인이 필요로 하는 시스템을 모델링할 수 있을 것이다.

☞ 퀴즈


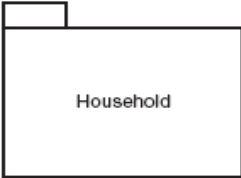
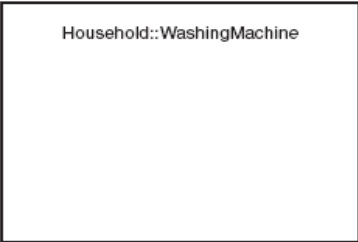
1. 객체란 무엇일까?
2. 객체들은 어떤 방법을 사용하여 함께 수행될까?
3. 다중성(multiplicity)은 어떤 것을 명시하고 있을까?
4. 두 개의 객체는 한 개 이상의 연관 관계를 가질 수 있을까?
5. 상속이란 무엇인가?
6. 캡슐화란 무엇인가?

Part 3. 객체지향 개념을 적용해보자

(1)클래스를 그림으로 나타내기

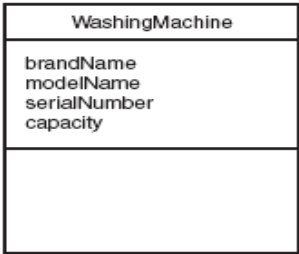
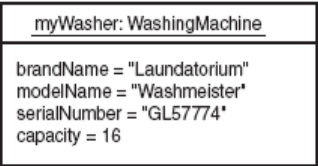
☞ 객체 지향 표기법

- 클래스 이름은 대문자로 시작한다.
- 두 단어 이상으로 된 클래스 이름은 각 단어의 처음을 대문자로 시작하며 함께 붙여쓴다.
- 속성이나 오퍼레이션은 소문자로 시작한다.
- 두 단어 이상으로 이루어진 속성이나 오퍼레이션은 제일 앞 단어만 제외하고 각 단어의 처음을 대문자로 시작하며 함께 붙여쓴다.
- 오퍼레이션의 이름 뒤에는 괄호를 붙인다.

	☞ 클래스를 UML로 나타낼 때는 사각형 안에다가 클래스 이름을 적는다. 이클래스 이름은 첫 자를 대문자로 적고, 사각형의 상단에 가깝게 두는 것이 상례이다.
	☞ UML구조를 나타내는 또 하나의 아이콘으로 패키지도 클래스 대신 사용될 수 있다.
	☞ WashingMachine클래스가 Household라는 이름의 패키지에 속해 있다면 Household::WashingMaching라는 이름을 부여할 수 있다. 여기에 쓰인 더블콜론(::)은 패키지 이름(왼쪽)과 클래스 이름(오른쪽)을 구분해 주는 기호이고, 이렇게 쓴 클래스 이름을 경로이름(pathname)이라고 부른다.

(2)속성

- 속성(attribute)이란 ‘클래스’에 속한 특성에 이름을 붙인 것으로 이것이 가질 수 있는 값의 범위를 설정한다. 클래스는 0개 이상의 속성을 가질 수 있다. 한 단어로 된 속성의 이름은 소문자로 쓰는 것이 보통이며, 속성 이름이 두 단어 이상으로 되어 있을 경우에는 두 단어를 붙여 쓰되 둘째 단어부터 첫 문자를 대문자로 쓴다.

	☞ 클래스 이름과 속성
	☞ 객체는 그 객체마다 다른 값을 가진 속성을 지닌다.

<div> <div>WashingMachine</div> <div> brandName: String = "Laundatorium" modelName: String serialNumber: String capacity: Integer </div> </div>	<p>☞ 클래스 아이콘에 들어갈 수 있는 타입으로는 문자열, 부동 소수점, 실수, 정수, 블라인드 등이다. 타입을 써주려면 속성 이름의 뒤에 콜론을 하나 찍고 나서 타입을 쓴다. 물론 속성의 기본값을 그대로 써줄 수 있다.</p>
--	--

(3)오퍼레이션

- 오퍼레이션(operation)이란, 객체에게 요청할 수 있는 행동을 말한다.

<div> <div>WashingMachine</div> <div> brandName modelName serialNumber capacity </div> <div> acceptClothes() acceptDetergent() turnOn() turnOff() </div> </div>	<p>☞ 클래스의 오퍼레이션은 속성 리스트 영역의 아랫부분에 두고, 선을 그어 구분한다.</p>
<div> <div>WashingMachine</div> <div> brandName modelName serialNumber capacity </div> <div> acceptClothes(c:String) acceptDetergent(d:Integer) turnOn():Boolean turnOff():Boolean </div> </div>	<p>☞ 오퍼레이션 이름 뒤에 따라오는 괄호 안에 매개 변수(parameter)리스트를 넣어줄 수 있는데, 하나의 매개 변수는 “이름:타입”의 형태를 가진다.</p> <p>☞ 옆에서 나열한 이러한 오퍼레이션의 정보를 오퍼레이션의 시그니처(signature)라고 한다.</p>

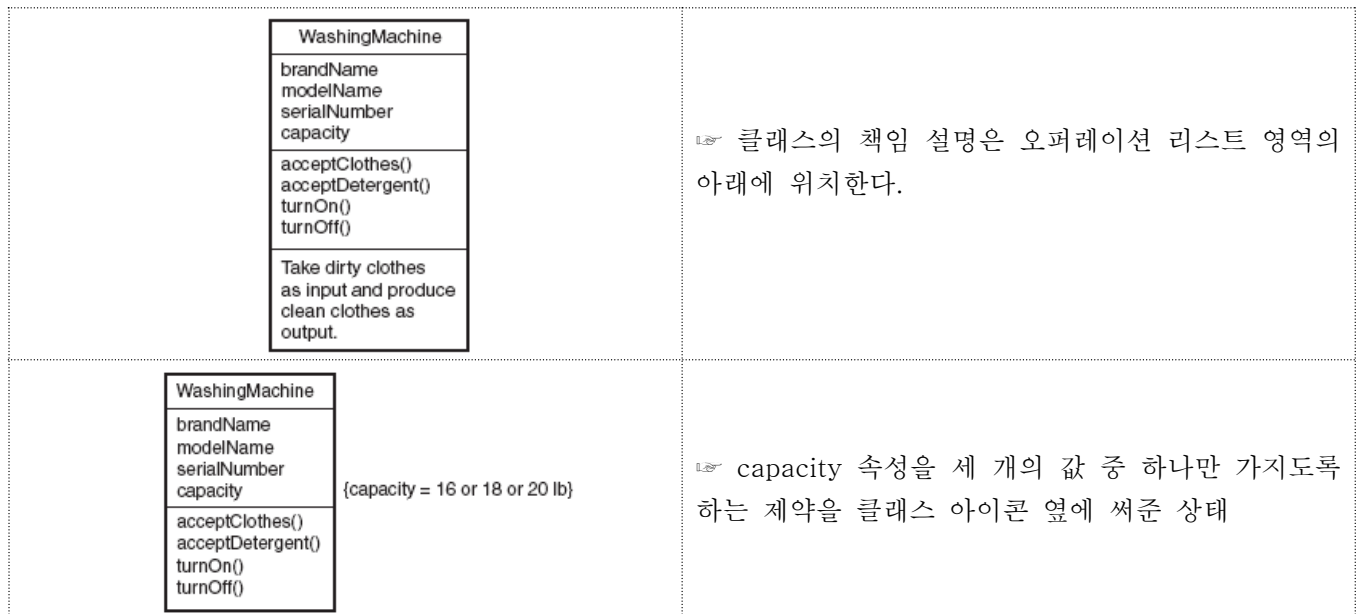
(4)클래스 표기

<div> <div>WashingMachine</div> <div></div> <div></div> </div>	<p>☞ 실제로는 속성과 오퍼레이션을 모두 적지 않아도 상관 없다.</p>
<div> <div>WashingMachine</div> <div> brandName ... </div> <div> acceptClothes() ... </div> </div>	<p>☞ 생략기호는 속성과 리스트가 전부 표시되지 않았음을 나타낸다.</p>
<div> <div>WashingMachine</div> <div> «id info» brandName modelName serialNumber «machine info» capacity </div> <div> «clothes-related» acceptClothes() acceptDetergent() «machine-related» turnOn() turnOff() </div> </div>	<p>☞ 키워드를 사용하여 속성 또는 오퍼레이션 리스트를 구분 지을 수 있다.</p>

(5) 책임과 제약

- 책임(responsibility)이란 특정한 타입 혹은 클래스가 “해야 하는” 일을 설명해 둔 것이다. 좀더 자세히 말하자면 속성과 오퍼레이션이 무엇을 이루고자 하는 지에 대한 설명이라 할 수 있다.

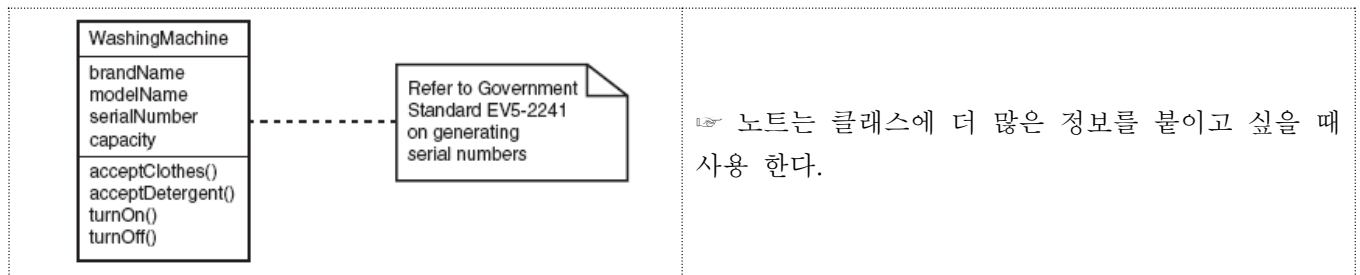
-다소 공식적으로 사용되는 표기법으로 **제약(constraints)**이 있다. 제약은 중괄호({})안에 자유 형식의 텍스트가 들어 있는 형태로서, 클래스가 따라야하는 규칙을 붙여줄 때 사용한다.



(6) 노트 붙이기

- 속성, 오퍼레이션, 책임, 제약 이외에도 클래스에 추가적인 정보를 덧붙일 수 있는 수단이 하나 더 있는데, 노트(note)라고 불리는 것이다.

- 대개 노트는 속성이나 오퍼레이션에 붙인다.



※ 노트는 텍스트와 그림을 동시에 사용할 수 있다.

(7) 클래스 모델링 시작하기

- 클래스는 지식 도메인에 기반한 어휘와 용어로부터 만들어진다. 시스템 분석가는 의뢰인과 상담하여 그들이 가지고 있는 지식 도메인을 파악하여 정리하고, 그 도메인에서 발생하는 문제를 해결할 컴퓨터 시스템을 설계해 나가면서, UML에 사용할 용어를 선정하고 이것을 클래스로 모델링하는 것이다. 의뢰인과 이야기할때는 명사(noun)와 동사(verb)는 놓치지 말아야한다. 명사는 모델링할 클래스의 이름이 될 가능성이 매우높고, 동사는 모델링한 클래스의 오퍼레이션이 될 후보들이 되기 때문이다. 클래스의 핵심이 되는 리스트(클래스이름, 속성, 오퍼레이션)를 모두 정리한 다음에는, 각각의 클래스가 의뢰인의 업무에서 어떤 역할을 할 지에 대하여 묻도록 하자.

실습) 농구게임을 모델링하는 시스템 분석가라고 가정하자. 여러분은 이 농구 게임을 이해하기 위하여 자료 수집차 모 대학 농구팀 감독을 만난다. 코치와 나눈 대화는 다음과 같다.

분석가 : 감독님, 농구게임이 뭔지 한마디로 정리해 주세요.

감 독 : 농구의 목적은 상대방의 골대에 걸린 바스켓(basket)에다가 공(ball)을 던져 넣어서 스코어가 많은 편이 이기는 것이죠, 각 팀은 두명의 가드(guard) 두명의 포워드(forward), 센터(center), 이렇게 다섯 선수(players)로 구성 됩니다. 이 다섯 명을 공을 어떻게든 상대방 바구니 근처로 몰고(advance) 가서 슛(shoot)하죠.

분석가: 공을 어떻게 몰고 가지요?

감 독 : 드리블(dribble)와 패스(pass)를 하면 되죠. 그리고 공격 시간(shot clock)이 다되기 전에 슛을 해야 합니다.

분석가 : 공격 시간(shot time)은 어떻게 되는지요?

감 독 : 한 팀이 공격권을 가진 후에 슛하는데 까지 걸리는 제한 시간입니다. 프로농구는 24초이고, 국제경기는 30초, 대학농구에서는 35초입니다.

분석가 : 스코어는 어떻게 계산합니까?

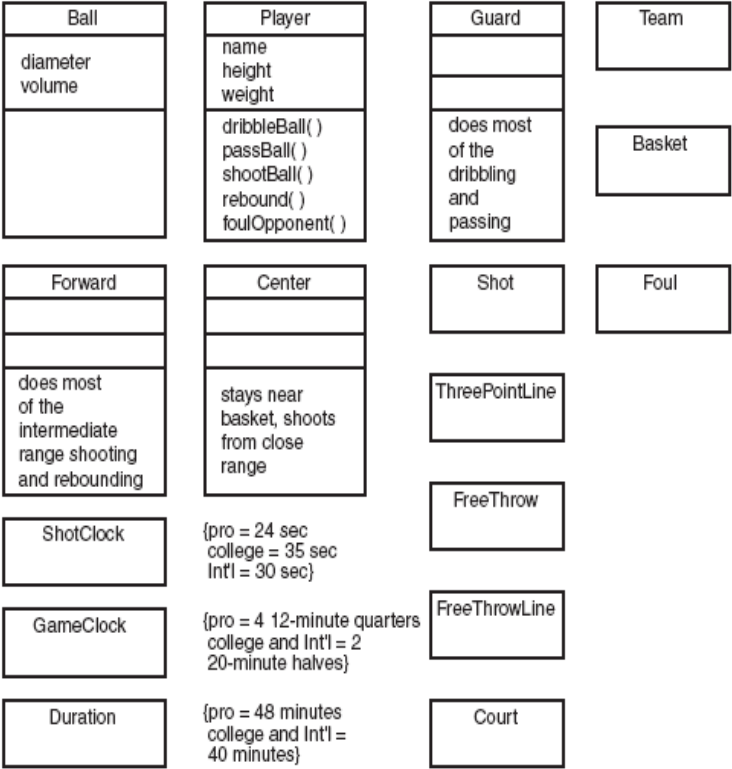
감 독 : 3점슛 라인(three point line) 뒤에서 슛을 하지 않으면 한 골당 2점씩입니다. 3점슛라인 뒤에서 슛을 해서 넣었다면 3점입니다. 프리 드로우(free throw)는 1점입니다. 프리 드로우는 파울을 했을 때 던지는 것입니다. 상대방(opponent)에게 파울(foul)을 하면 경기는 중단되고, 상대방 프리 드로우 라인(free throw line)에서 바스켓에 공을 던집니다.

분석가 : 각 선수들이 어떤 역할을 하는지 조금 자세 설명해 주세요.

감 독 : 가드는 드리블과 패스를 주로 하여 공의 수급을 맡습니다. 대개 포워드보다 키가 작죠. 또 포워드는 센터보다 키가 작은 게 보통입니다. 모든 선수들은 드리블, 패스, 슛, 리바운드(rebound)를 할 수 있습니다. 포워드는 대부분의 리바운드와 미들 슛(intermediate range shooting)을 맡고, 센터는 바스켓 근처에서 있다가 골밑 슛(shoots from clos range)을 주로 하죠.

분석가 : 코트(court)는 어떻게 되어 있나요? 그리고 한 게임의 제한 시간은 얼마나 됩니까?

감 독 : 국제 경기에서는 길이 28미터, 폭 15미터의 코트를 사용합니다. 바스켓은 지면에서 10피트 높이에 있지요. 이제 경기 시간을 말해 드릴까요? 프로농구에서는 전체가 48분이고, 12분 단위로 네 개의 쿼터(quarter)로 나누어져 있습니다. 대학 농구와 국제 경기에서는 전체가 40분이고 20분 단위로 전후반전(halves)으로 나뉩니다. 게임 시간(game clock)은 시작후 남은 시간을 계속 표시 하도록 되어 있지요.



농구 게임을 모델링한 초기 다이어그램

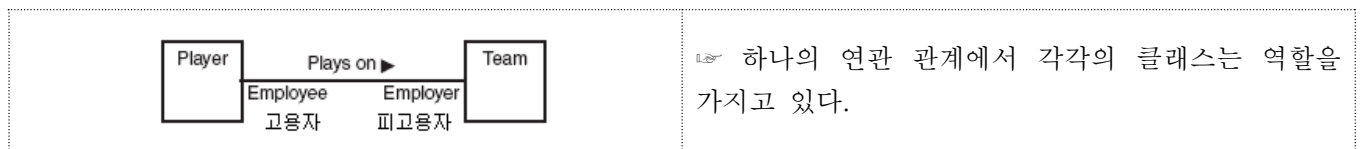
Part 4. 관계를 지어 봅시다.

(1)연관

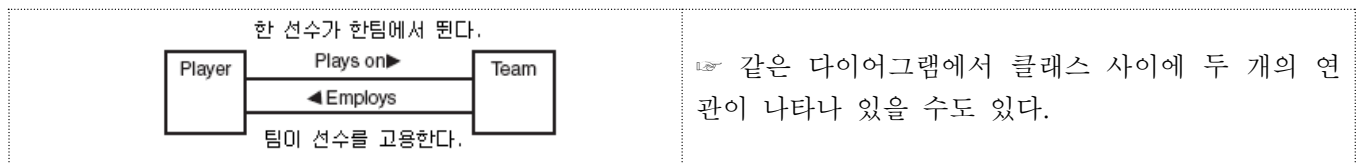
- 클래스가 개념적으로 서로 연결되어 있을 때, 이 관계를 연관(association)이라고 부른다.



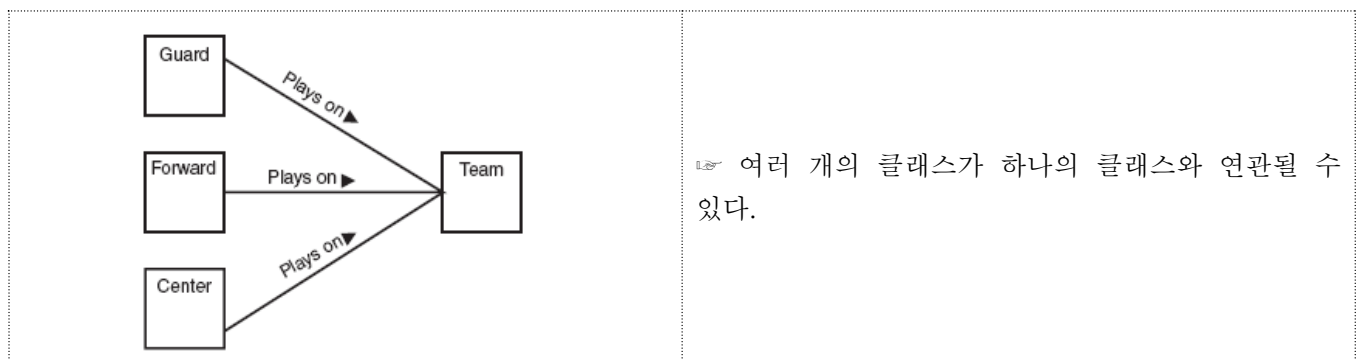
- 한 클래스가 다른 클래스와 연관 되면, 각각은 해당 연관 관계 내에서 역할을 가진다. 클래스 옆(연결선 가까이) 에다가 원하는 역할을 써줌으로써 연관 관계 내에서 역할을 표시할 수 있다.



- 선수와 팀의 연관 관계는 반대 방향으로 생각해 볼 수 있다. 반대 방향이라면 팀이 선수를 고용하는 ("Employs") 관계일 것이다.

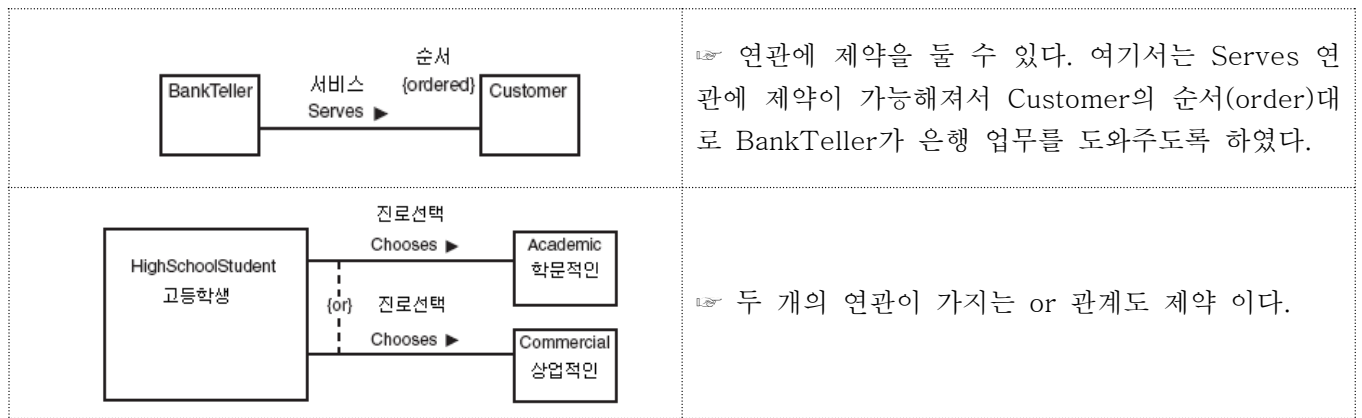


- 연관은 하나의 클래스가 다른 클래스에 연결된 것 이상으로 복잡해질 수 있다. 여러 개의 클래스가 하나의 클래스에 연결되는 것도 가능하다.



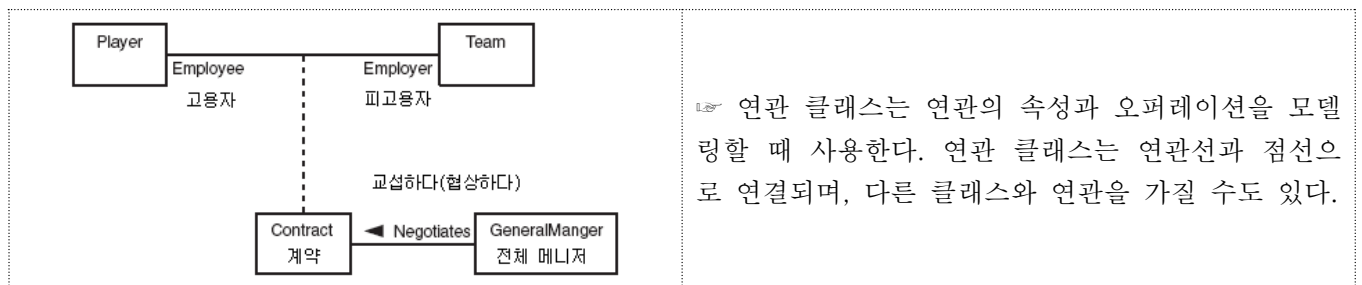
(2) 연관에 대한 제약

- 두 클래스 사이의 연관 관계가 어떠한 규칙을 따라야 할 경우가 있다. 이 규칙을 덧붙일 수 있는데, 앞장에서 알아본 바로 그 제약(constraint)을 연결선 부근에다가 쓰면 된다.



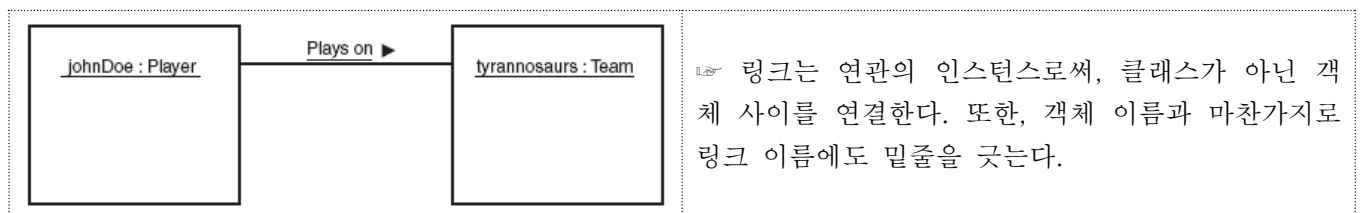
(3)연관 클래스

- 속성과 오퍼레이션을 가진 연관은 **연관클래스(association class)**라는 이름으로 따로 구분한다.



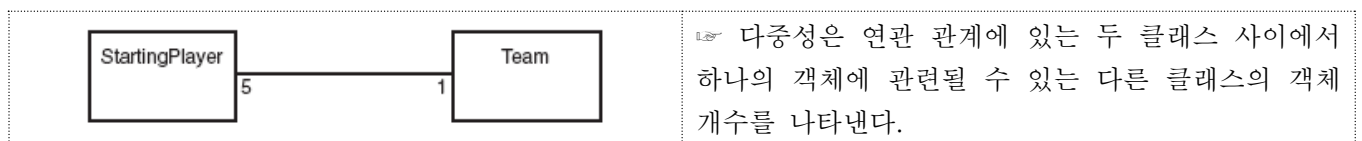
(4)링크

- 객체가 클래스의 인스턴스인 것처럼, 연관도 자신의 인스턴스를 가질 수 있다. 어떤 특정한 선수가 특정한 팀에 소속되어 있는 관계를 생각하면, 이 때의 plays on 연관 관계를 **링크(link)**라고 부른다.

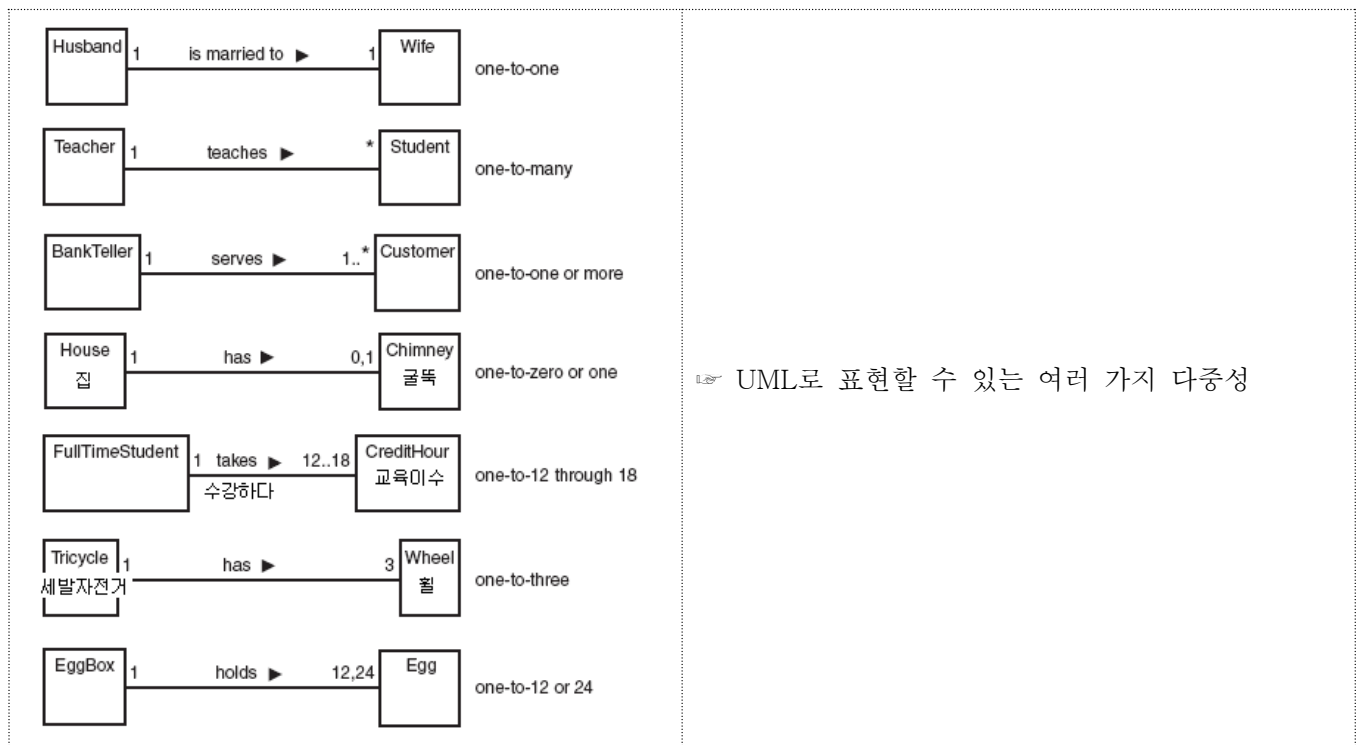


(5)다중성

- “연관되어 있는 두 클래스 사이에서 한 클래스의 객체와 관계를 가질 수 있는 다른 클래스의 객체 개수의 “의 예인게 바로 **다중성(Multiplicity)**이다.



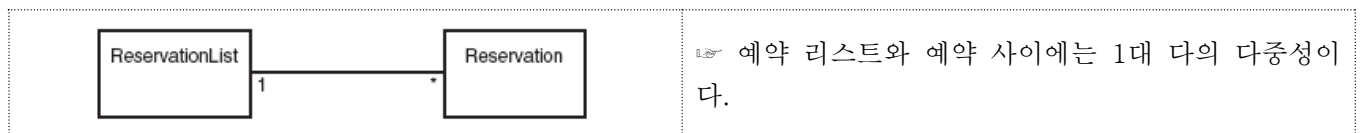
-위에서 본 것만이 다중성의 전부라고 생각하면 안된다. 하나의 클래스가 다른 클래스에 연관될 때 존재할 수 있는 다중성은 일 대일(one-to-one), 일 대 다(one-to-many), 일 대 일 또는 그이상 (one-to-one or more), 일 대 0 또는 (one-to-zero or one), 일 대 일정범위(one-to-a bounded interval)



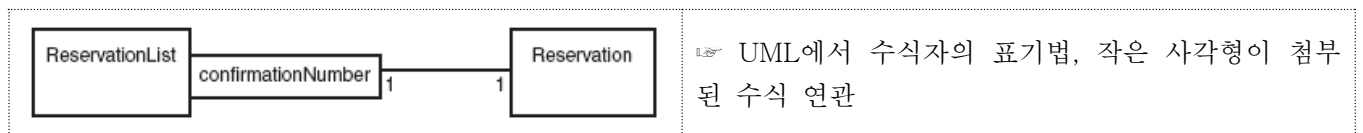
UML로 표현할 수 있는 여러 가지 다중성

Tip) 클래스 A가 클래스 B에 대하여 일 대 0 혹은 1의 다중성을 가질 때, “클래스B는 클래스 A에 대하여 선택적(optional)이다”라고 한다.

(6)수식연관

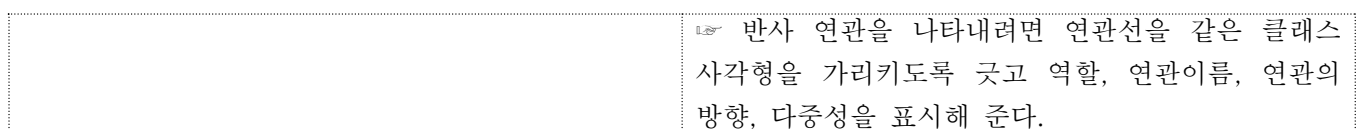


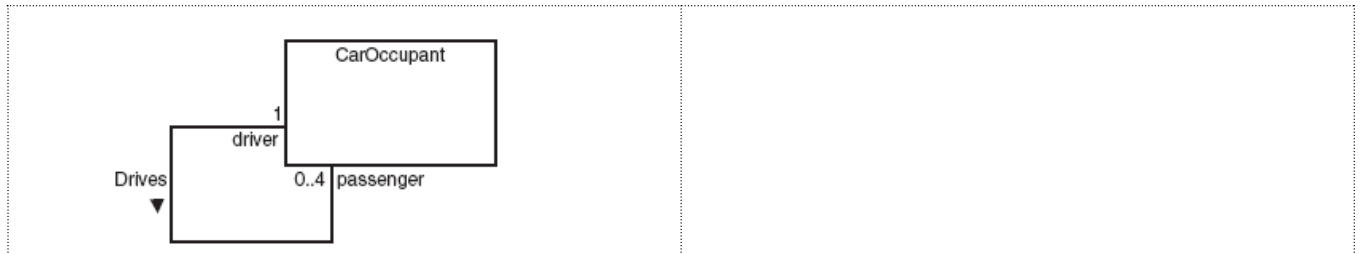
- 여러분이 호텔에 방을 예약(Reservation)할 때 호텔로부터 확인 번호(Confirmation number)를 받게 된다. 예약과 관련하여 문의를 하고자 한다면, 예약 리스트에서 해당 예약 건을 찾을 수 있도록 확인 번호를 불러주어야만 한다. UML에서 이러한 식별 정보는 수식자(qualifier)라고 하여 작은 사각형으로 나타내고, 1 대 다 다중성에서는 “1”을 의미하는 클래스 옆에 붙게 된다.



(7)반사 연관

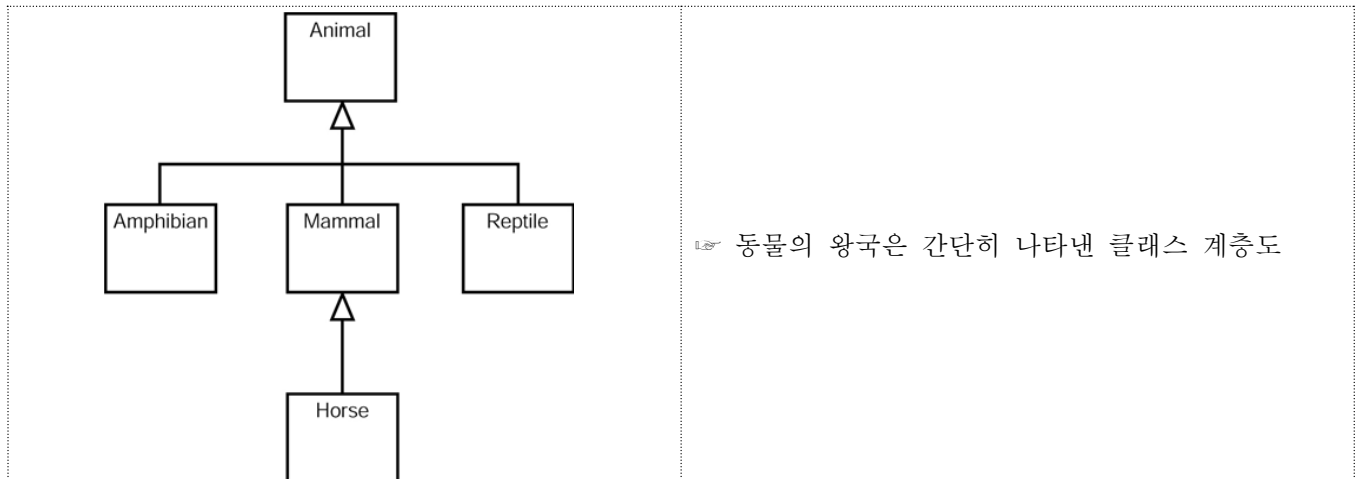
- 클래스는 자신과 연관 관계를 가질 수도 있으며, 이것을 반사 연관(reflexive association)이라고 한다. 여러 가지 역할을 맡을 수 있는 객체를 가지고 있을 때 이런 경우가 발생한다.





(8)상속과 일반화

- 객체지향 개념에서는 **상속(inheritance)**이라고 하면 UML에서는 이것을 **일반화(generalization)**라고 한다.



동물의 왕국은 간단히 나타낸 클래스 계층도

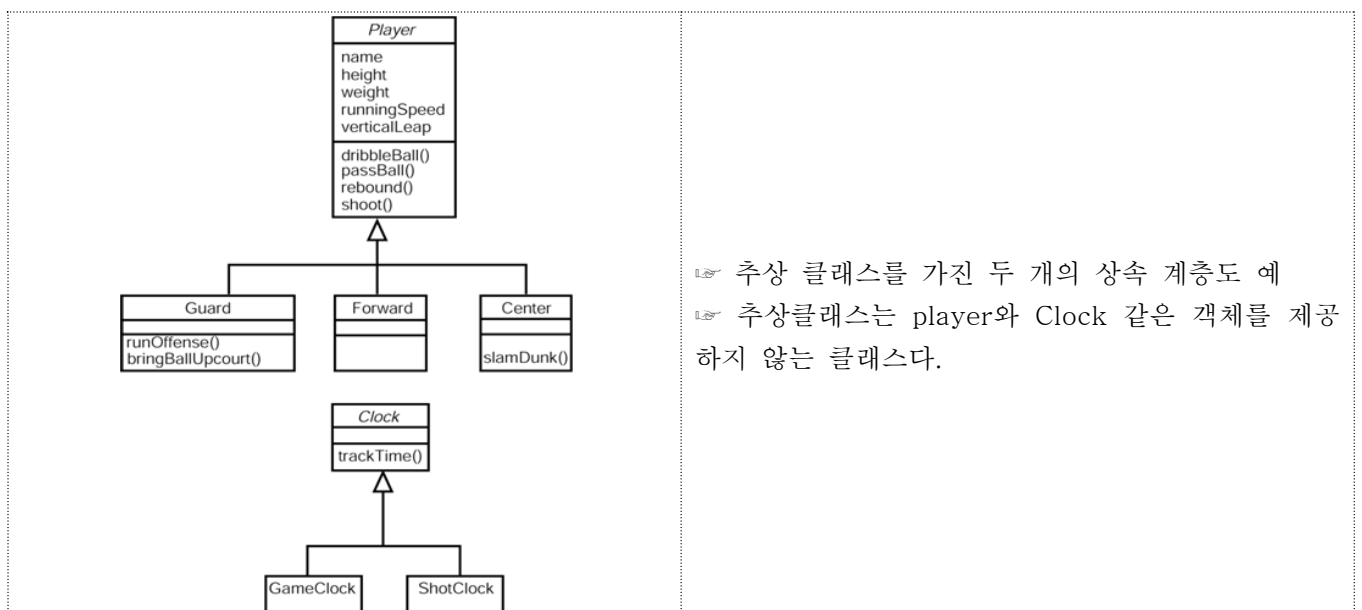
Tip) 상속받은 속성과 오퍼레이션은 서브클래스의 사각형에 써주지 않는 다는 것이다.

(9)상속 관계는 이렇게 정할 수 있다.

- 시스템 분석가는 고객과 상담하는 과정에서 여러 가지의 상속 관계를 발견해낸다.
- 슈퍼클래스와 서브클래스를 동시에 가지는 어떤 클래스가 충분히 존재할 수 있다.
- 시스템 분석가는 한 클래스에 속해 있는 속성과 오퍼레이션의 일반성을 찾아내어 다른 클래스에 적용할 수 있는지를 재빨리 간파해야 한다.

(10)추상 클래스

- 추상클래스(abstract class)라고 한다. 추상 클래스는 이탤릭체로 쓴다.

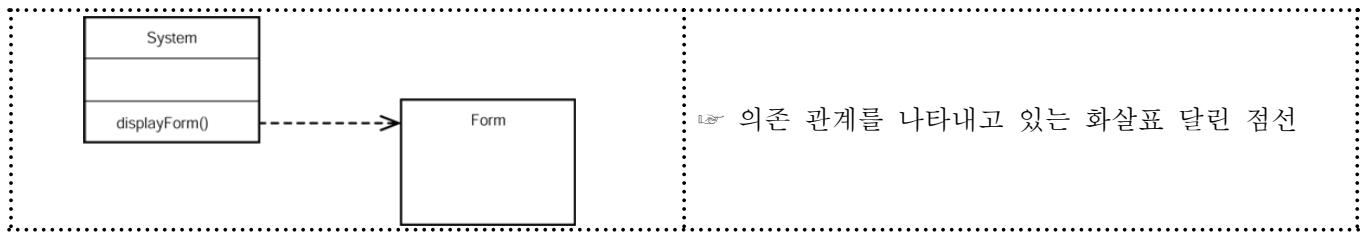


추상 클래스를 가진 두 개의 상속 계층도 예

추상클래스는 player와 Clock 같은 객체를 제공하지 않는 클래스다.

(11)의존관계

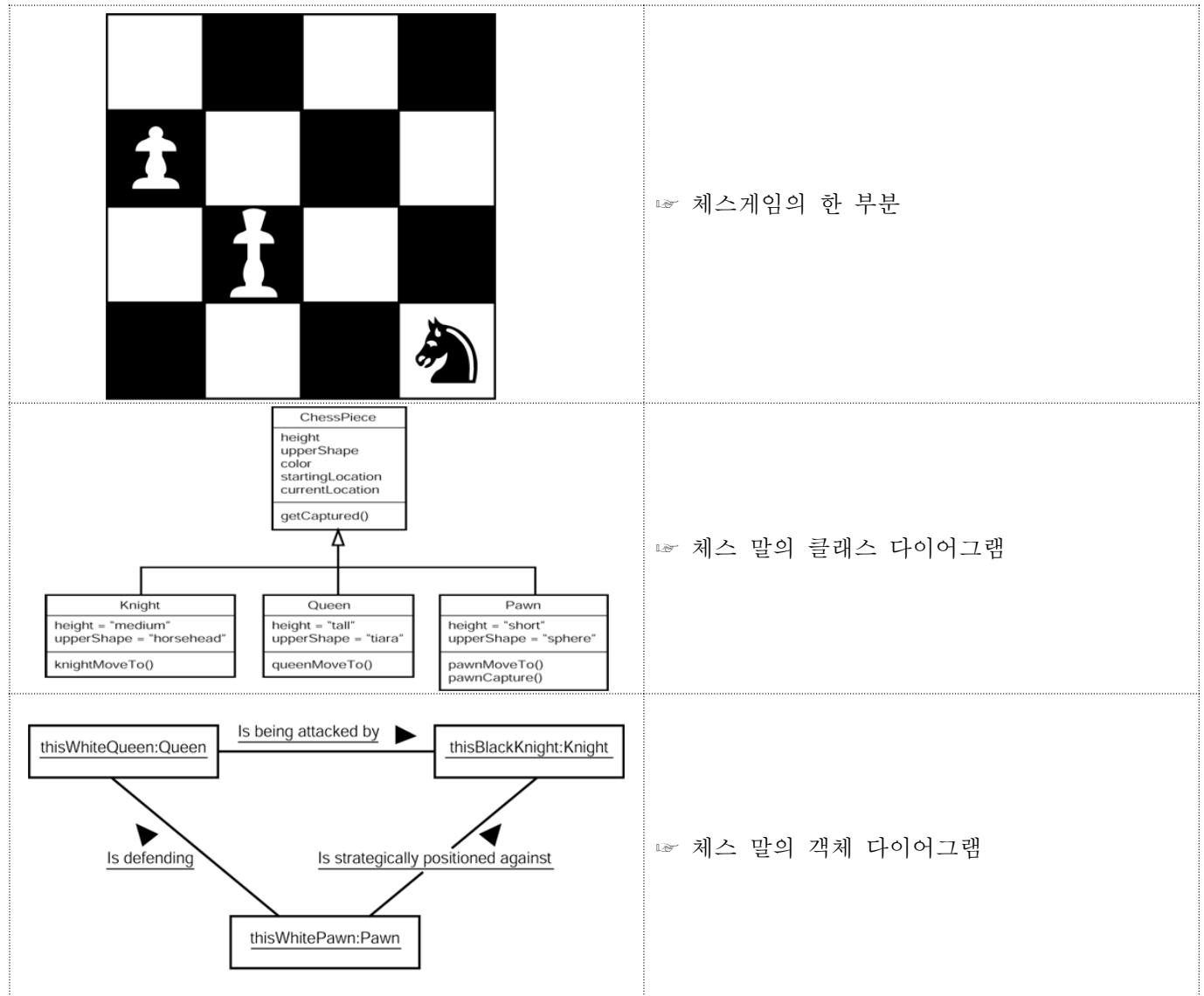
- “한 클래스가 다른 클래스를 사용하는” 관계가 하나 더 있다. 이것을 의존관계(dependency)라고 한다. 의존 관계가 가장 흔하게 드러난 예는 다른 클래스를 사용하는 오퍼레이션의 시그니처를 보일 때이다.



의존 관계를 나타내고 있는 화살표 달린 점선

(12)클래스 다이어그램과 객체 다이어그램

- 클래스 다이어그램이란 클래스의 특성과 속성, 다른 클래스와의 연관 같은 일반적인 정보를 알려준다. 반면에 객체 다이어그램은 클래스의 특정한 인스턴스 정보와 그 객체가 현재 어떠한 상황에 처해 있는지를 보여준다.



체스게임의 한 부분

체스 말의 클래스 다이어그램

체스 말의 객체 다이어그램

퀴즈

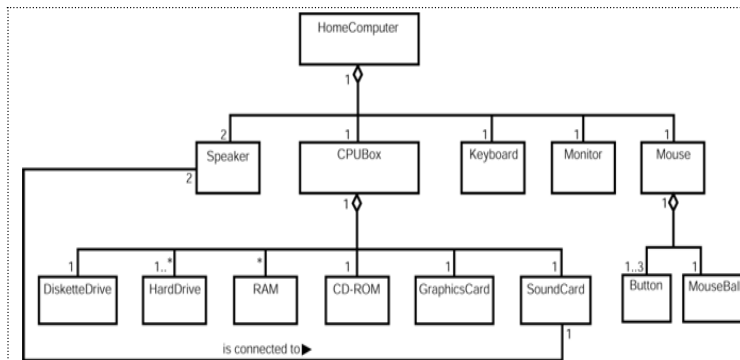
1. 다중성은 어떻게 나타낼까?
2. 상속 관계는 어떻게 정할까?
3. 추상 클래스란 무엇일까?
4. 수식자의 효과는 어떤 것일까?

Part 5. 집합연관, 복합연관, 인터페이스 그리고 실체화

(1) 집합연관

- 간혹 하나의 클래스가 여러 개의 컴포넌트 클래스로 구성되어 있는 경우가 있다. 이러한 상황은 집합연관 (aggregation)이라 불리는 특수한 관계이다.

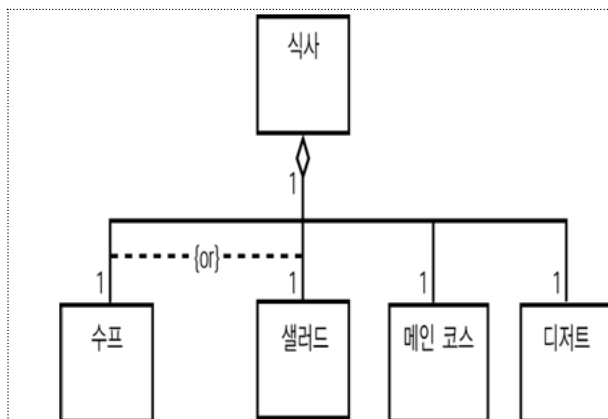
- 집합연관은 계층도로 나타낸다 “전체(whole) 클래스는 윗부분에 위치하며, 컴포넌트 클래스는 아랫 부분에 위치한다. 전체 클래스와 컴포넌트 클래스는 선으로 연결되며, 전체 클래스 쪽에 빈 마름모꼴이 붙는다.



이 그림에서는 각각의 컴포넌트 클래스가 하나의 전체 클래스 속에 있는 것으로 나타나지만 이러한 상황만이 집합연관은 아니자 통합 엔터테이먼트 시스템 예로 들면, 리모콘은 TV의 컴포넌트이고, VCR의 컴포넌트도 된다.

1-1) 집합연관에 대한 제약

- 집합연관에 속해 있는 컴포넌트 들이 or 관계에 놓일 때 도 있다.



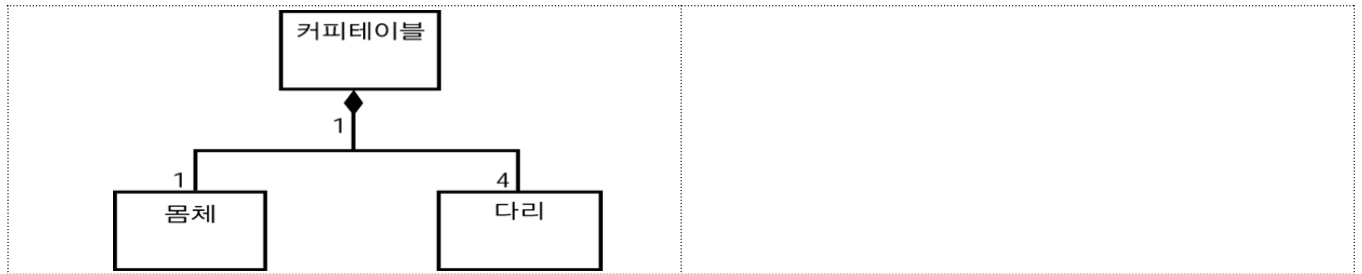
어떤 식당의 경우 한 끼 식사를 수프 혹은(or)샐러드, 그리고 메인 코스와 디저트로 구성하여 내놓는다.

두 컴포넌트 중 하나가 전체를 구성하는 일부가 될 수 있음을 나타낸 그림

(2) 복합연관

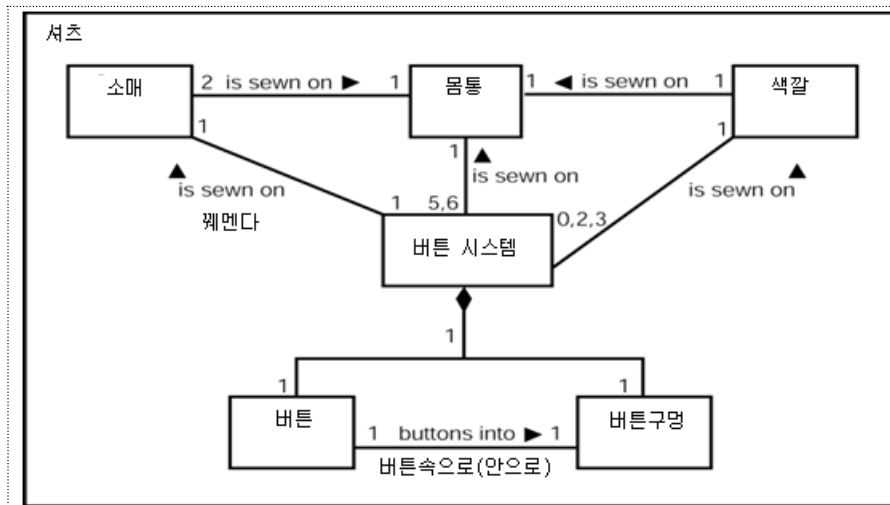
- 복합체(composite)는 강한 집합 연관에 의해 만들어진 클래스이다. 복합체에서 각 컴포넌트 클래스는 오직 하나의 전체 클래스에만 속할 수 있다.

복합연관에서는 각각의 컴포넌트 클래스가 오직 하나의 전체 클래스만을 구성하며, 안이 채워진 마름모꼴을 사용하여 나타낸다.



(3)복합체 구조 다이어그램

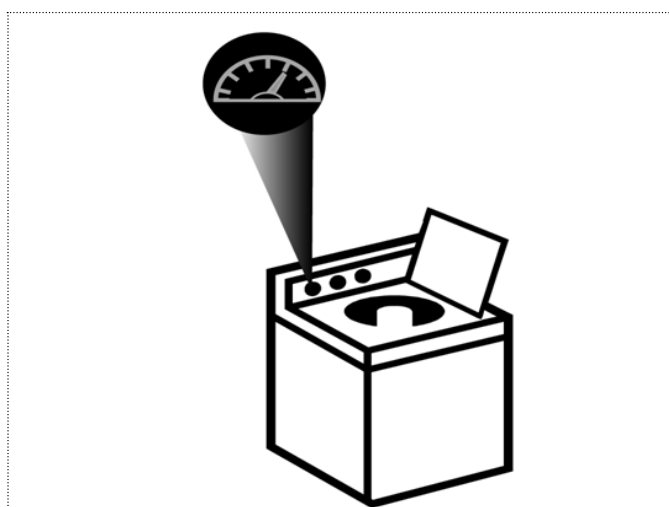
- 복합체는 클래스의 컴포넌트를 나타낸다고 하였다. 만약 클래스의 내부 구조를 보여주고 싶을 때에는 어떻게 하여야 할까? 바로 이때 이용할 수 있는 것이 UML2.0의 복합체 구조 다이어그램(composite structure diagram)이다.



복합체 구조 다이어그램에서는 클래스의 컴포넌트를 나타낼 때 전체 클래스 사각형 안에 다이어그램을 그려준다.

(4)인터페이스와 실체화

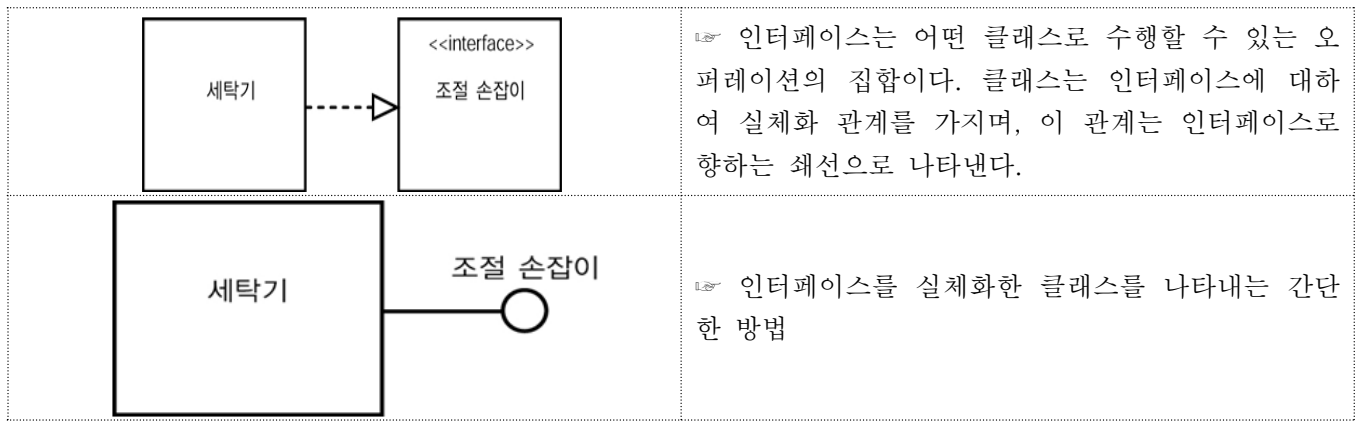
- 인터페이스란 클래스의 일정한 행동(behavior)을 나타내는 오퍼레이션의 집합으로써, 다른 클래스에서 사용될 수 있다.



세탁기에서 일을 시킬 수 있는 조절 손잡이(세탁기의 인터페이스)

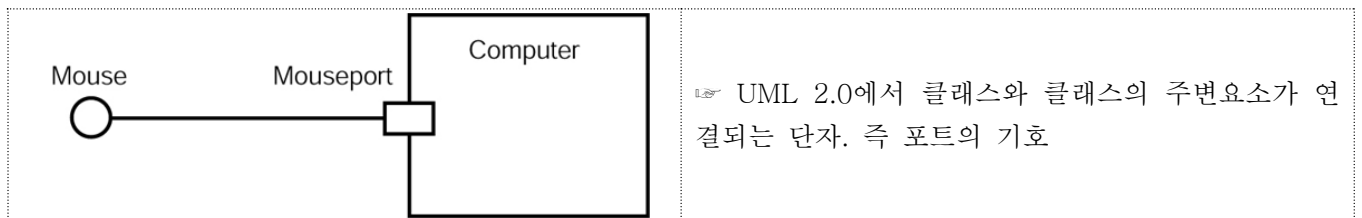
UML에서는 세탁기의 행동 중 일부가 조절 손잡이의 행동을 “실체화(realize)”한 것이라고 말한다. 클래스와 인터페이스가 가지는 이러한 관계를 실체화(realization) 관계라고 한다.

- 클래스와 인터페이스간의 실체화 관계를 나타내는 기호는 상속의 기호와 매우 흡사하다.



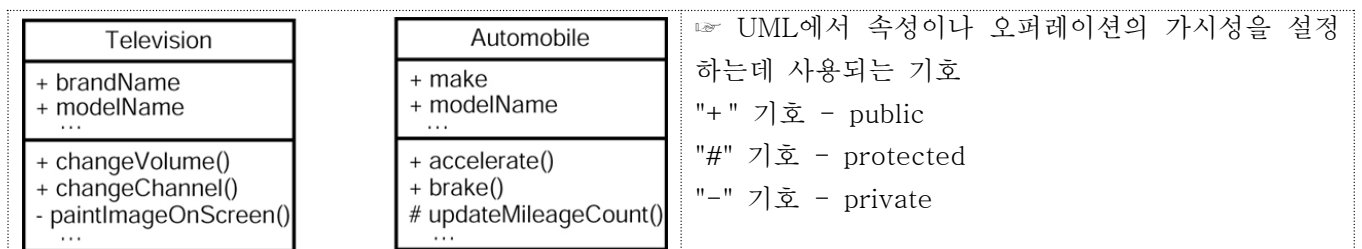
(5)인터페이스와 포트

- UML 2.0에서는 클래스와 인터페이스의 연결 단자를 그릴 수 있으므로 인터페이스의 개념을 좀 더 자세히 표현할 수가 있게 되었다. UML2.0에서는 포트같은 입력되는 연결단자를 그릴 수 있도록 기호를 제공한다.



(6)가시성

- 가시성(visibility)이란, 속성과 오퍼레이션에 적용되는 것으로, 해당 클래스(혹은 인터페이스)의 속성과 오퍼레이션을 들여다 볼 수 있는 범위를 말한다.
- 가시성을 나타내는 속성은 protected, public, private 세 가지다.
- public속성과 오퍼레이션은 다른 클래스가 마음껏 사용할 수 있다.
- protected속성과 오퍼레이션은 원래 클래스와 여기서 상속받은 클래스만이 사용할 수 있다.
- private속성과 오퍼레이션은 원래의 클래스만이 사용할 수 있다.
- 인터페이스를 실체화하기 위해서는 인터페이스안에 설정된 오퍼레이션들이 모두 public가시성을 가지고 있어야 한다.



(7)스코프

인스턴스 스코프(instance scope) - 각각의 인스턴스에 속한 속성과 오퍼레이션들이 각자의 값을 가지도록 되어 있다. (자바의 인스턴스 변수/ 인스턴스 메소드를 생각하면 된다.)

클래스 스코프(classifier scope) - 해당 클래스에 대해 유일한 속성값과 오퍼레이션 값을 가진다. (자바의 클래스 변수/ 클래스 메소드를 생각하자)

1. 집합연관과 복합연관의 차이는 무엇일까?
2. 실체화란 무엇인가? 상속과 무엇이 비슷하고, 무엇이 다른가?
3. 인터페이스를 통한 상호 작용은 어떻게 표현될까?
4. 가시성을 설정하는 세 가지를 들고, 각각에 대해 설명해 보자.

Part 6. 유스케이스 모델링

(1)유스케이스란?

- 사용자 시점에서 시스템을 모델링하는 역할
- 시스템 분석가가 사용자와 힘을 합쳐 시스템의 사용 방법을 결정하는데 도움을 주는 장치
- 쉽게 생각하면 시스템 사용에 대한 시나리오 집합
- 요구 사항을 알아내는 과정

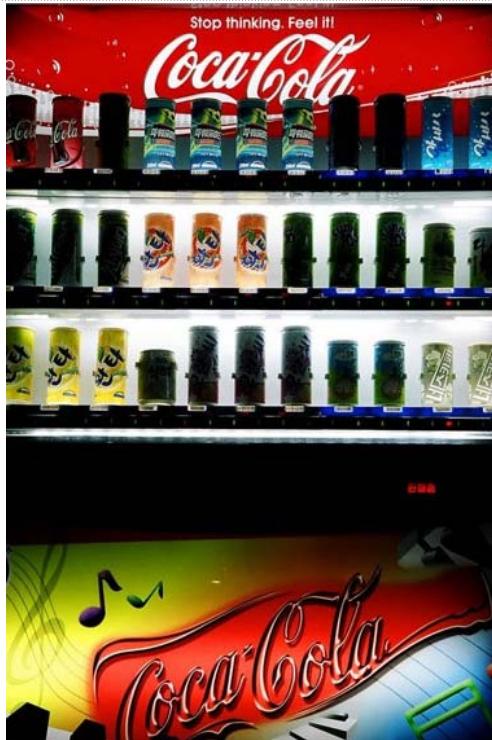
(2)왜 유스케이스가 중요한가?

- 유스케이스는 시스템을 사용하는 사용자에게서 정보를 얻어내는데 매우 유용하다.
- 유스케이스의 목적은 시스템 사용자를 시스템 분석과 설계의 초기단계에 포함시키는 것과 사용자들이 진정으로 도움을 받을 수 있는 시스템을 만들 확률을 높이는 것이다.

(3)유스케이스의 예제

- 유스케이스(use case)는 행위자(actor)가 관심을 가지고 있는 유용한 일을 달성하기 위한 시나리오의 집합을 명사한다.

☞ 음료수 자동 판매기의 경우
- 유스케이스는 “음료수 사기” 이다.



3-1)“음료수 사기” 유스 케이스

- 시나리오 시작은 자동 판매기에 돈을 넣는 것이겠다. 그다음 자신이 마실 음료수를 선택한다. 자동판매기는 선택된 음료수를 한 개 이상 가지고 있을 것이고, 돈을 넣은 사용자에게 음료수 한 캔을 떨어구 줄 것이다. 종료 조건은 음료수 한캔을 가진 상태가 될 것이다.

3-2)“음료수 없음”(out-of-selection) 시나리오

- 돈을 넣고 음료수구입 (by drinking water) 유스 케이스를 시작하고 마실 음료수를 선택한다. 사용자가 선택한 음료수가 다 떨어졌기 때문에 “없다(out-of-brand)”라는 메시지를 표시할 것이다. 그럼 다른 소다를 선택하게 하거나 투입한 돈을 반환받을 수 있도록 하는 옵션도 제공해야 한다. 이 때, 사용자는 다른 음료수를 선택하든지 아니면 돈을 돌려 받는다.

- 선행조건 “목마른 사용자” 종료 조건 “다른 음료수 혹은 반환된돈”이 될 것이다.

3-3)“돈이 맞지 않음(incorrect-amount-of-money)” 시나리오

- 자동 판매기에 돈을 넣고, 자신이 마실 음료수를 선택한다. 자동판매기에는 선택한 음료수가 있다고 가정하자

경우1) 자동 판매기에 거슬러 줄 돈이 있다면 음료수와 함께 거스름돈을 뱉어냄

경우2) 거스름돈이 부족한 경우에는 “거스름돈 없음” 메시지를 표시하면서 처음에 넣은 돈을 모두 돌려줌

- 선행조건은 “목마른 사용자” 종료조건은 “거스름돈과 음료수” 또는 “처음에 넣었던 돈”이 될 것이다.

(4)또다른 유스케이스를 추가 해보자

- 지금까지는 소비자 입장에서 음료수 자동판매기를 살펴보았는데, 다른 사용자가 바라 보았을 때 자동 판매기의 모습은 또 다르다.

- 공급원(supplier) - 자동판매기에 떨어진 음료수를 채운다.

- 수금원(collecotr) - 자동판매기에 모인 돈을 가지고 간다.

4-1)“다시채움(restock)”의 유스케이스

- 일정한 시간간격(여기선 약2주라고 하겠다)이 지났을 때 이 유스케이스를 시작한다.
- 자동판매기의 보안장치를 해제하고, 자동 판매기의 앞문을 열고 음료수 브랜드의 부족한 용량을 채우고 거스름돈의 잔액도 채운다. 그리고 나서 자동 판매기의 앞문을 닫고 보안 장치를 다시 건다.
- 선행조건은 “시간 간격의 흐름” 이고 종료조건은 “음료수 판매에 관련된 것을 다시 채운다” 이다.

(5)유스 케이스를 만들 때

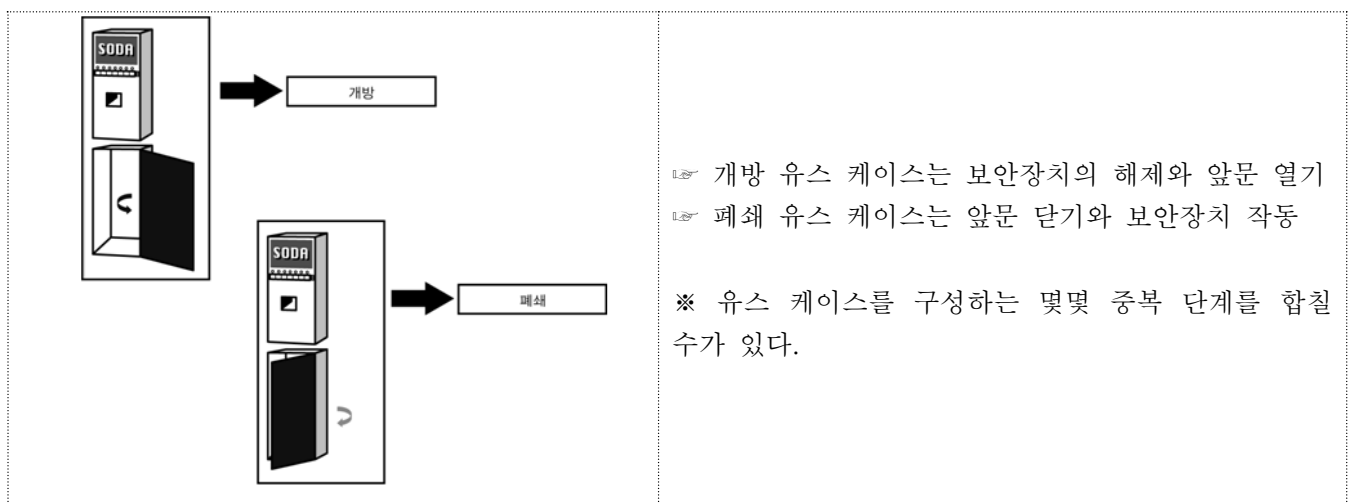
- 어떻게 구현할 것인지에 대하여 전혀 걱정을 하지 않는다.
- 음료수 자동판매기의 내부에 대해 신경도 쓰지 않았다.
- 음료수 자동판매기가 사용자에게 어떻게 보여지는 지에 대해서만 신경쓸 뿐이다.

※ 유스 케이스의 궁극적인 목적

- 음료수 자동판매기의 설계자와 개발자를 위한 자료
- 공장에서 나오는 음료수 자동판매기는 소비자, 보급원, 수금원이 원하는 유스케이스에 맞추어 이들이 사용하기 좋은 형태로 만들어짐.

(6)유스 케이스 포함하기

- 기존의 유스 케이스를 재사용하는 기법으로서 위에서 말한 자동판매기를 예로 들면 채우기 유스케이스와 수금 유스 케이스의 공통적인 진행단계는 보안 장치를 해제하고, 앞문을 열고, 앞문을 닫고, 보안장치를 작동 시키는 단계이다.
- 중복적인 진행 단계를 뽑아내어 유스 케이스를 따로 만든다.



(7) 유스 케이스 확장하기

- 기존의 유스 케이스에 몇 개의 진행 단계를 덧붙여서 새로운 유스 케이스를 만들어내면 유스케이스를 재사용할 수 있다.
- 유스 케이스 확장의 예

☞ 자동 판매기에 음료수 캔을 넣기 전에, 보급원은 잘 팔리는 음료수 브랜드와 잘 안팔리는 음료수 브랜

드를 체크한다고 가정하면 모든 브랜드의 음료수를 다시 채우는 대신에, 잘 팔리지 않는 브랜드의 음료수를 치우고 그 자리에 잘 팔리는 브랜드의 음료수를 (이것이 인기가 있을 것 같으니까) 넣음, 자동 판매기의 앞문에다가 고를 수 있는 브랜드들을 다시 정리하고 몇가지 단계를 추가하여 만든 유스케이스 “판매고에 맞추어 다시 채움(restock according to sales)”이라고 부를 수 있을 것이다.

(8)유스 케이스 분석은 이렇게 시작한다

- ☞ 초기 클래스 다이어그램을 그리기 위해 만난 의뢰인과의 대화로부터 시작.
- ☞ 사용자와 만나(될 수 있으면 사용자 그룹과 만나는 것이 좋음) 이야기할 때에는 설계하고자 하는 시스템을 가지고 어떤 일을 하는지를 모두 묻도록 함
- ☞ 유스케이스 각각에 대해 간단한 설명을 붙여 봄
 - 설명이 많으면 많을수록 사용자와 할 수 있는 대화의 밀도는 진해짐
- ☞ 유스 케이스는 시스템 개발 과정의 모든 단계를 끝낸다.

☞ 퀴즈

1. 유스 케이스를 시작 하도록 하는 개체를 무엇이라 부를까 ?
2. “유스 케이스 포함” 이란 무슨 의미일까?
3. “유스 케이스 확장” 이란 또 무슨 의미일까?
4. 유스케이스와 시나리오는 같은 것일까?

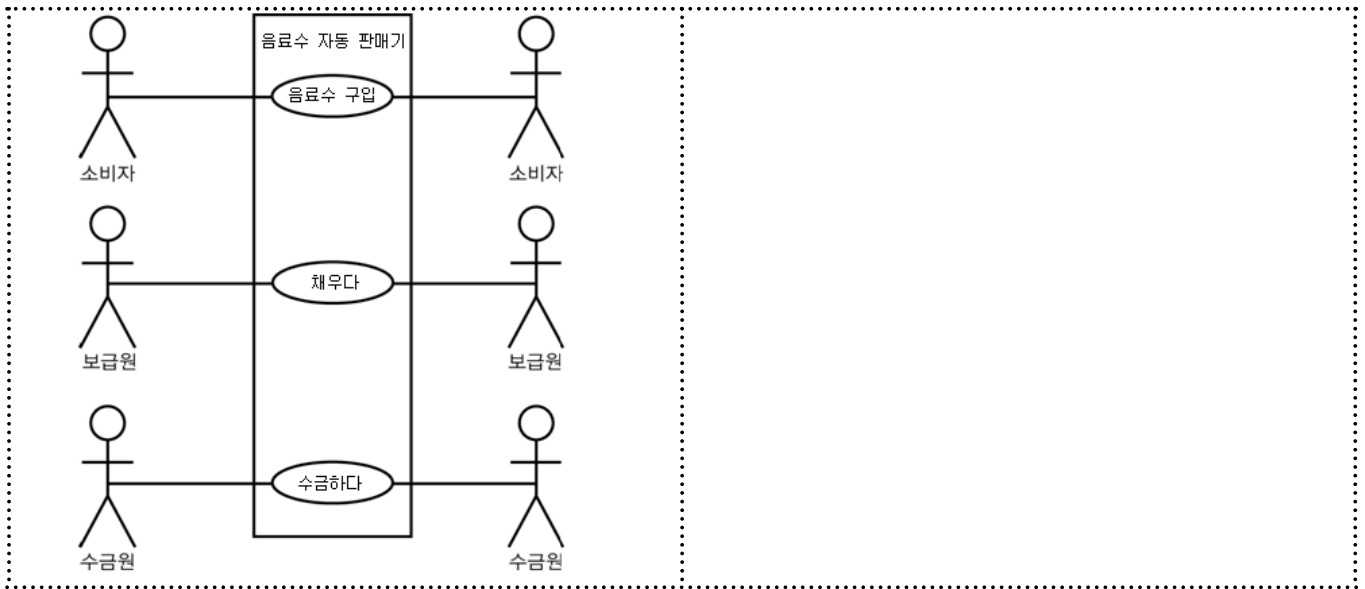
Part 7. 유스케이스 다이어그램

(1)유스 케이스 모델 나타내기



- ☞ 유스 케이스 모델에서 막대 인간은 행위자를 나타내며, 타원은 유스 케이스를 나타낸다.
- ☞ 둘 사이를 연결하는 실선은 행위자와 유스 케이스 사이에 교류가 이루어지고 있음을 나타낸다.

☞ Part 6에서 음료수 자동 판매기를 유스케이스 모델로 그린 결과



(2)시나리오의 진행 단계 나타내기

- 유스 케이스를 시작하는 행위자
- 유스 케이스를 위한 가정
- 유스 케이스가 시작하는데 필요한 선행 조건
- 시나리오의 진행 단계
- 유스 케이스가 끝나는데 필요한 종료 조건
- 유스 케이스의 결과를 받는 행위자

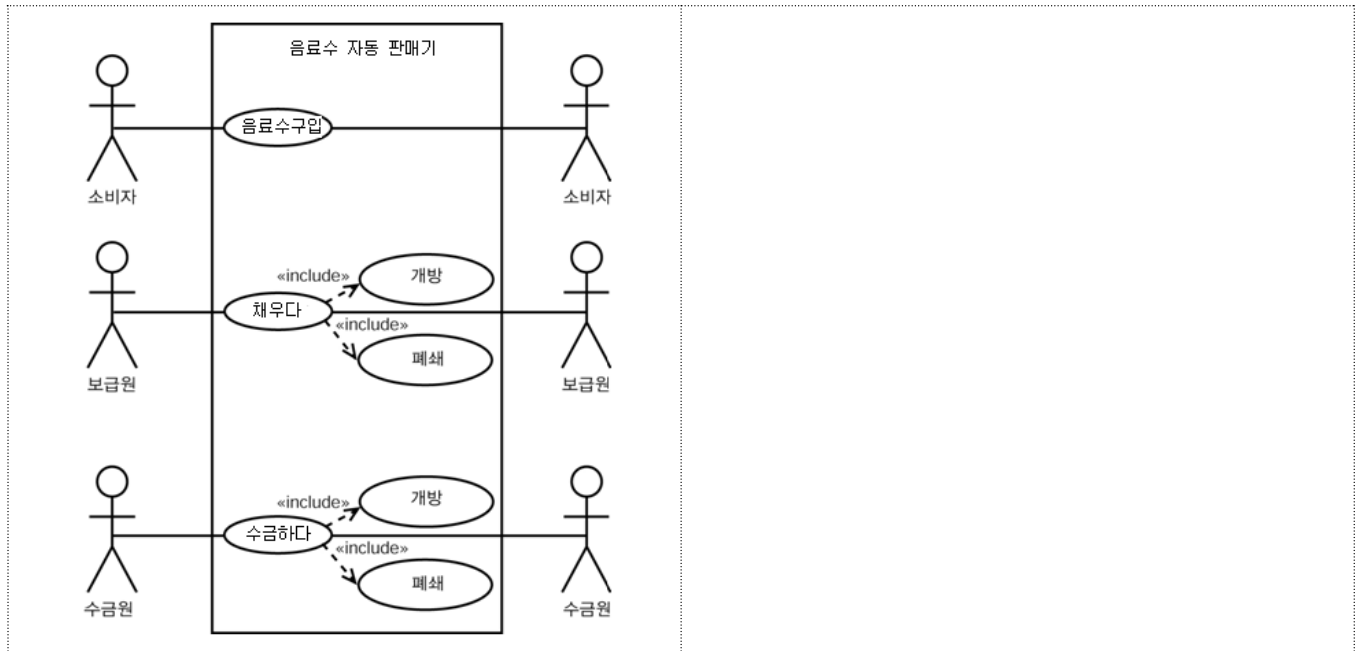
(3)유스 케이스 사이의 관계 나타내기

- 포함(inclusion)관계
 - 다른 유스 케이스에서 기존의 유스 케이스를 재사용하고 있는 관계
- 확장(extension)관계
 - 기존의 유스 케이스에 진행 단계를 추가로 하여 새로운 유스 케이스로 만들어낸 관계이다.
- 일반화(generalization)
 - 한 유스 케이스가 다른 유스 케이스를 상속한 관계
- 그룹화(grouping)
 - 여러 개의 유스 케이스를 조직화하는 단순한 방법

3-1)포함(inclusion)관계

☞ 포함 관계가 설정된 음료수 자동 판매기의 유스 케이스 모델

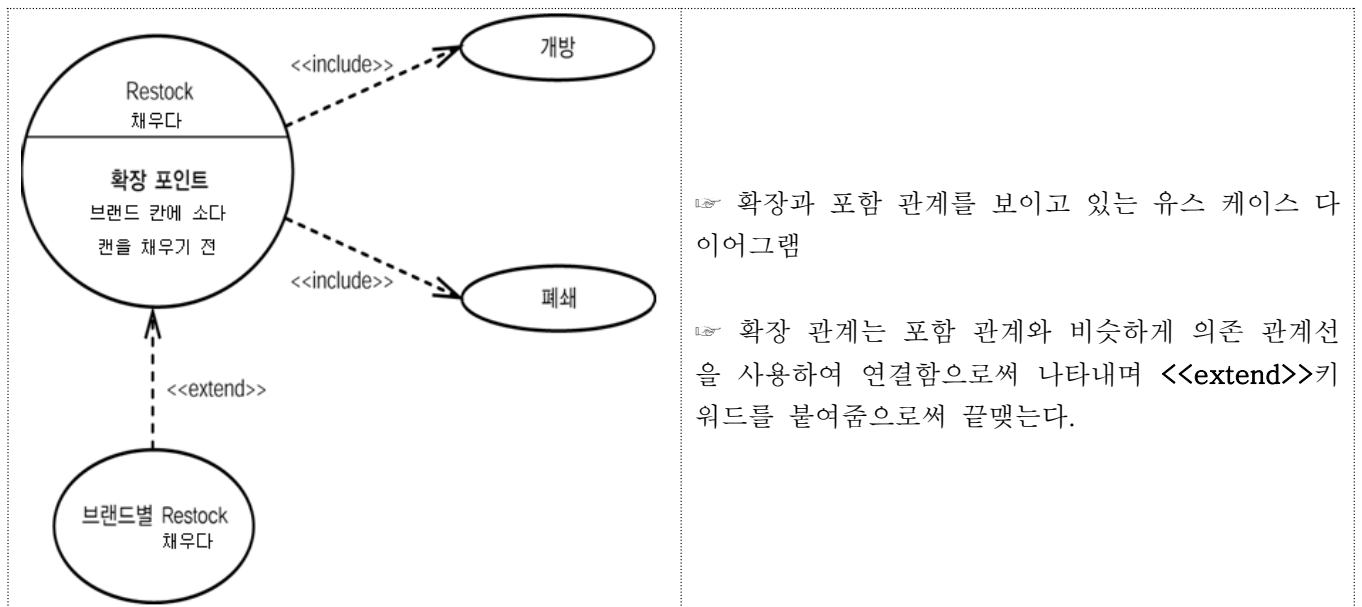
☞ 유스 케이스의 포함 관계를 나타낼 때에는 클래스 사이의 의존(dependency) 관계에 사용한 기호를 그대로 사용한다. 연결선 위에는 키워드<<include>>를 붙여준다.



3-2)확장(extension)관계

- 새로운 유스케이스를 만드는데 기본 구조가 된 유스케이스를 기본(base)유스케이스라고 하며, 새 유스 케이스는 기본 유스 케이스를 확장하였다고 한다.

- 유스 케이스의 확장은 기본 유스 케이스의 진행 단계에서 특정하게 지정된 위치에서만 발생할 수 있다. 이 위치를 확장포인트(extension point)라고 한다.

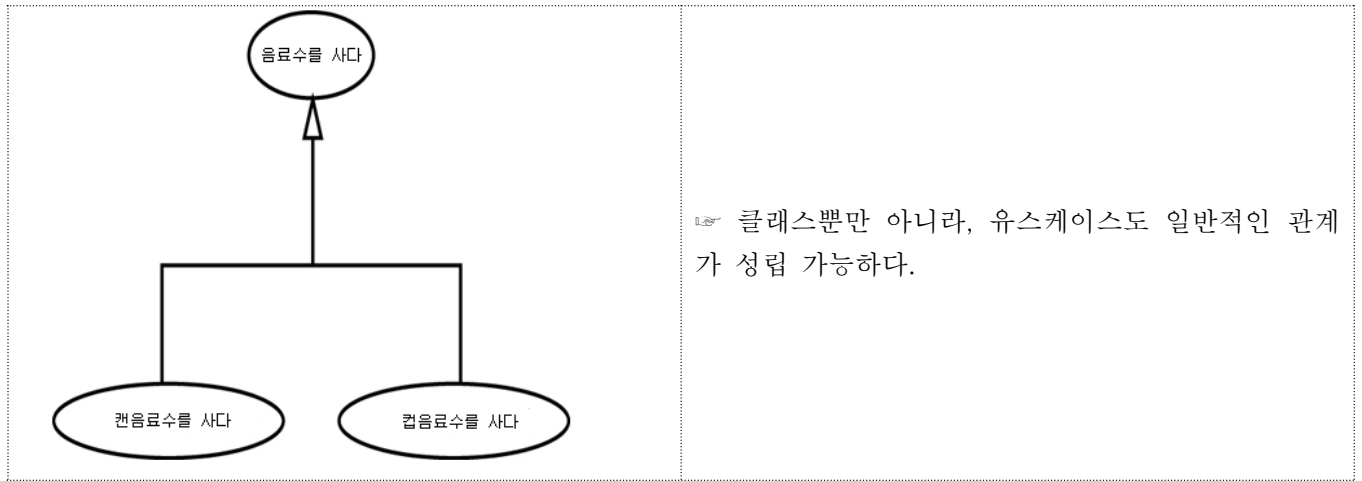


확장과 포함 관계를 보이고 있는 유스 케이스 다이어그램

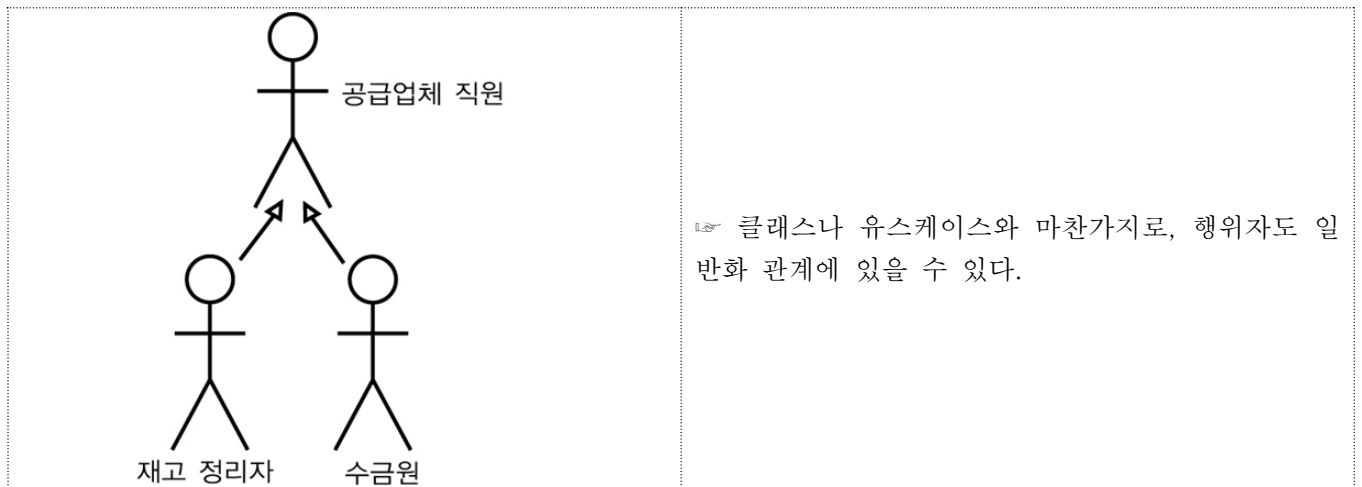
확장 관계는 포함 관계와 비슷하게 의존 관계선을 사용하여 연결함으로써 나타내며 <<extend>> 키워드를 붙여줌으로써 끝맺는다.

3-3)일반화(generalization)

- 클래스가 다른 클래스로부터 상속을 받을 수 있듯이 유스케이스도 상속이 가능하다. 유스케이스 상속의 경우, 자식 유스케이스는 부모 유스케이스가 가진 모든 행동과 의미를 물려받으며, 여기에 자신만의 행동을 추가할 수도 있다. 또한, 부모 유스케이스가 등장한 곳에는 항상 자식 유스케이스를 대신 놓을 수 있다.



- 일반화 관계는 행위자 사이에도 있을 수 있다.



3-4)그룹화

- 유스 케이스 다이어그램이 여러 개의 유스 케이스를 가지고 있는 형태로 나타날 수 있을때는 유스 케이스들을 조직화 하는 것이 좋다.

- 유스 케이스를 조직화 하는 가장 간단한 방법

■ 관련된 유스케이스를 하나의 패키지로 그룹화한다.

■ 패키지는 탭이 붙은 폴더 그림으로 나타내고, 이안에다가 관련된 유스케이스들을 넣으면 된다.

(4)시스템 분석 단계에서의 유스케이스 다이어그램

- 의뢰인과의 인터뷰로 시작

■ 인터뷰를 통해 시스템 분야(문제 해결 대상이 되는 도메인)에 대한 지식의 기초가 되는 클래스 다이어그램을 그리는 것

- 사용자와의 인터뷰

■ 행위자와 추상적 수준(상위 수준, high level)의 유스 케이스 다이어그램을 그려냄으로써 기능적인 요구사항을 정리

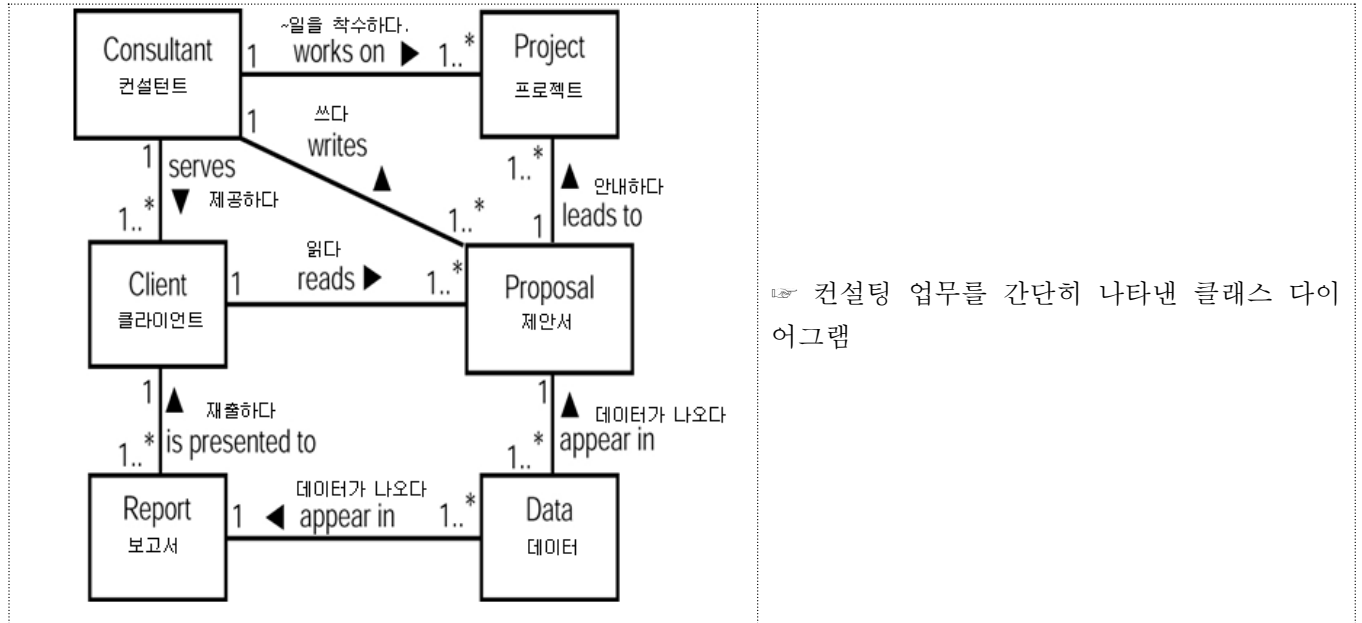
- 계속되는 사용자와의 인터뷰

■ 시스템 요구사항을 구체화

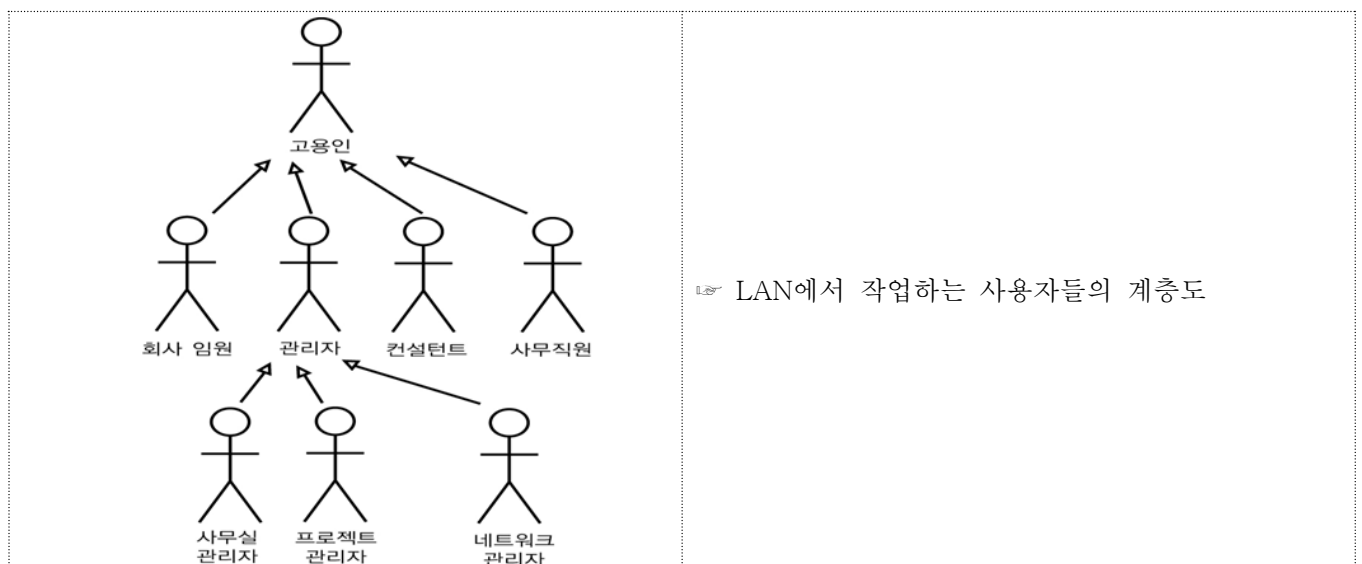
(5)유스 케이스 모델의 적용 예

- 여러분이 모 컨설팅 기업의 주문으로 LAN을 설계하는 일을 맡았다고 하자.
- LAN 선을 까는 일 뿐만 아니라 LAN에다가 구축해야 할 기능까지 파악해야 한다.
- 어떻게 시작해야 할까?

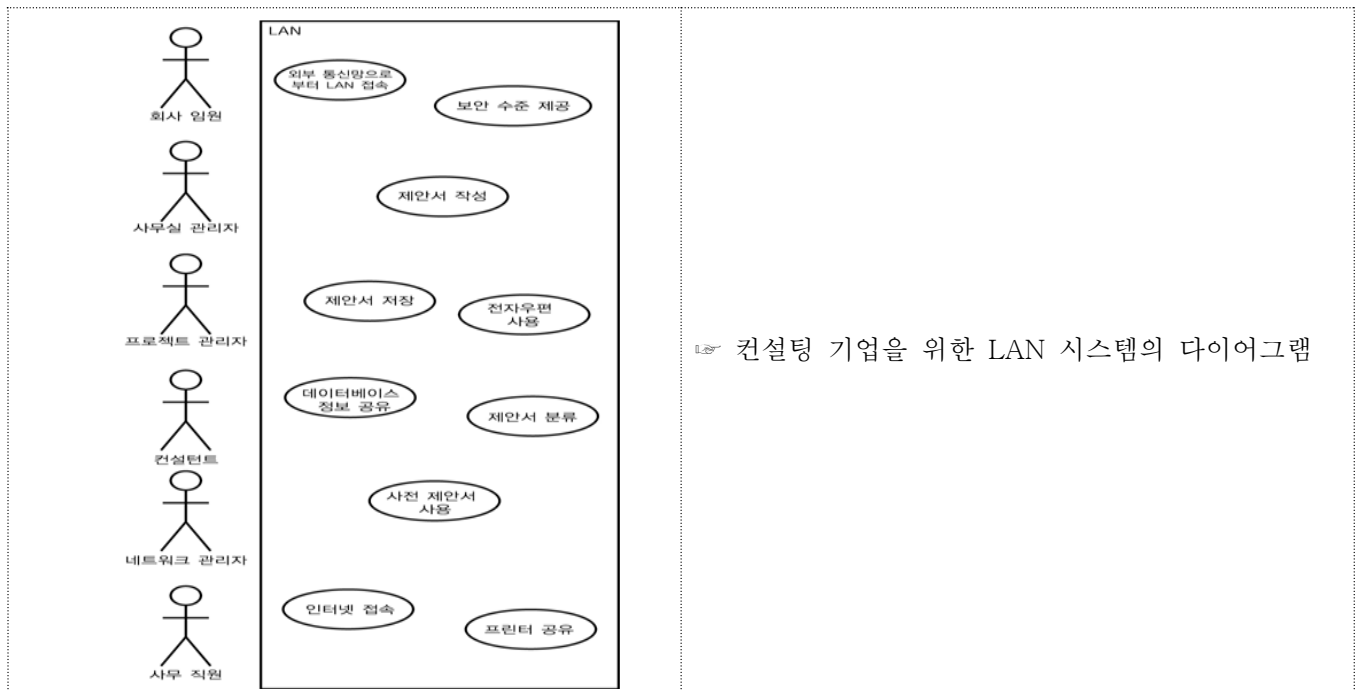
①컨설팅(consulting)이란 일이 어떤 것인가를 아는 것이 급선무 이기 때문에, 고객과의 인터뷰 목적은 이것을 나타낼 수 있는 클래스 다이어그램을 그리는 것이다.



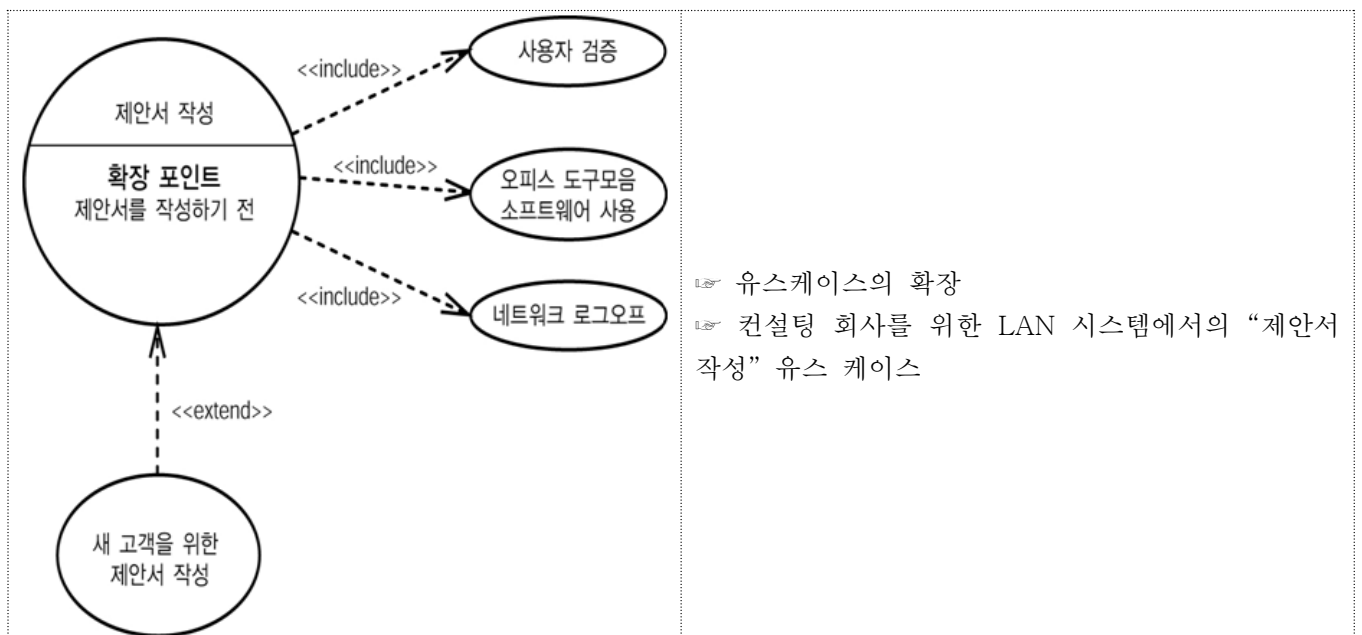
②이제는 시스템에 구축해 넣은 기능들에 대해 알아보기 위해 사용자에게로 관심을 돌려야한다.



③ 컨설팅 기업에서의 유스 케이스를 생각나는대로 써보자. 정리된 유스케이스와 상위 수준의 유스케이스 다이어그램으로 나타낼 수 있을 것이다. 이렇게 만든 유스 케이스들이 모여서 LAN의 기능적 요구사항을 이루게 되는 것이다.



④ 위에서 제시한 유스 케이스 다이어그램은 상당히 추상적이다. 이제는 다이어그램 내의 유스케이스 각각에 자세한 내용을 담을 수 있어야 한다.



☞ 퀴즈

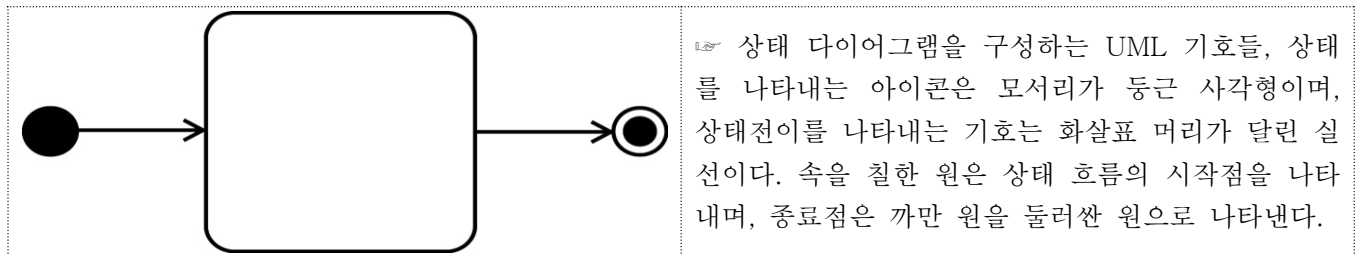
1. 유스 케이스를 시각화할 때 두 가지 좋은 점은?
2. 이번 장에서 공부한 일반화, 그룹화 그리고 유스케이스 사이의 관계를 설명해보자 그리고 유스 케이스를 그룹화할 필요가 있는 두 가지 상황도 이야기 해보자.
3. 클래스와 유스 케이스의 유사점은? 그리고 차이점은?
4. 포함한 확장은 어떻게 나타낼 수 있을까?

Part 8. 상태 다이어그램

(1)상태 다이어그램이란?

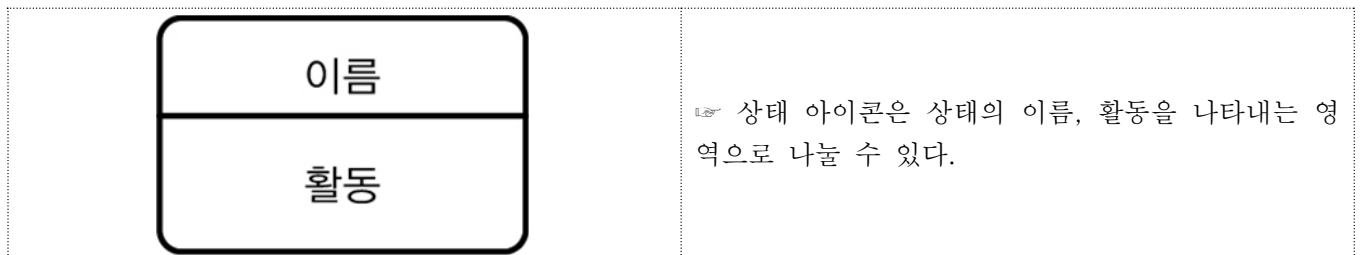
- 시스템 변화를 잡아낸다.
- 시스템이나 최소한 한 개 그룹의 클래스 혹은 객체의 행동을 모델링하는 이전의 다이어그램들과 달리, 상태 다이어그램은 “단일객체”의 상태를 나타낸다.
- 상태 다이어그램은 관습적으로 상태의 이름 중 첫 번째 문자는 대문자로 나타낸다. 가능하다면 이름의 끝을 ing(진행형)로 하는 것도 좋은 생각이다.

(2)상태 다이어그램을 구성하는 기호



(3)상태 아이콘에 넣는 정보

- 가장 위 부분에는 상태의 이름(이것은 영영을 나누든, 나누지 않는간에 반드시 써주어야 한다.), 중간 부분에는 상태 변수(state variable)가, 가장 아랫 부분에는 활동(activity)이 들어간다.

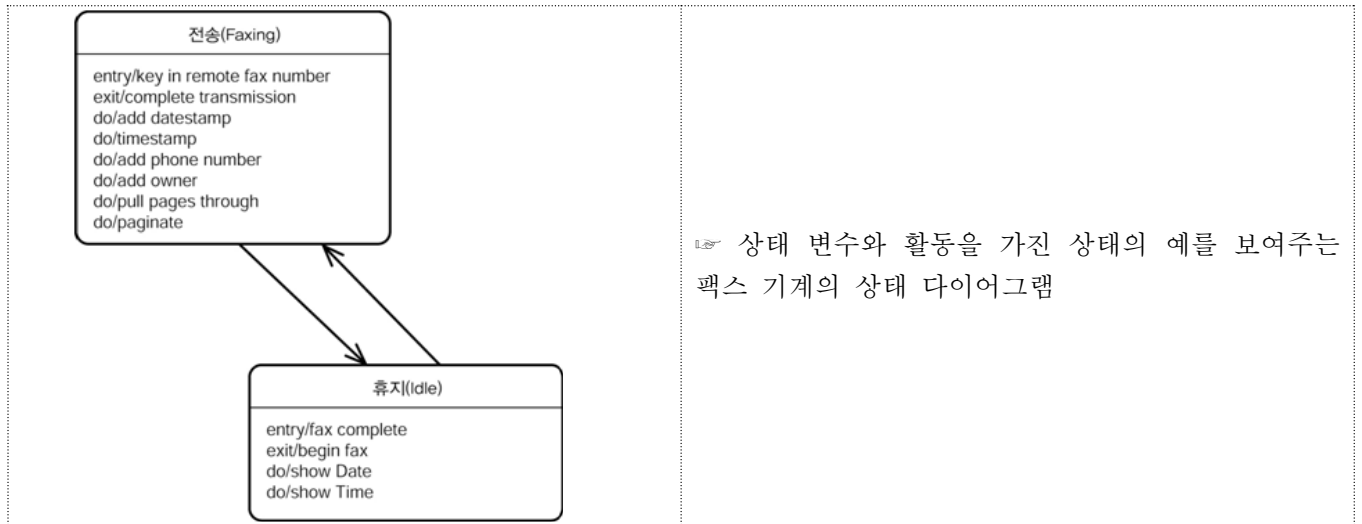


- 자주 쓰이는 세 가지 활동은 잘 알아보자.

- 진입 - 시스템이 상태로 들어갈 때 일어남
- 탈출 - 시스템이 상태에 서 빠져 나올 때 일어남
- 활동 - 시스템이 상태 안에 있는 동안 일어남

-팩스기계를 예로 들어보자. 팩스 기계가 팩스를 전송할 때(즉, 전송(Faxing) 상태에 있을 때) 기계는 팩스를 전송하기 시작한 날짜와 시간(상태 변수인“날짜(date)”와 ”시간(time)”의 각각의 값)을 체크하고, 전화번호와 주인의 이름(역시 상태 변수임“전화번호(phone number)”와 ”주인(owner)”의 각각의 값)을 체크해 둔다. 팩스 기계는 전송 상태에 있는 동안 팩스에 날짜 스탬프(add datestamp)와 시간스탬프를 붙이는 (add timestamp) 활동을 하며, 전화번호를 붙이고 (add phone number), 주인의 이름을 붙이는(add owner)활동도 동시에 수행한다. 이 상태에서 일어나는 다른 활동을 들어본다면 페이지를 끌어 당겨 넣고(pull pages through), 페이지를 매기며(paginate), 전송을 끝내는 일(complete taransmission)이 있을 것이다.

팩스 기계가 휴지(idle) 상태에 있는 동안 디스플레이에 날짜와 시간이 표시된다.



(4)상태 전이선에 추가되는 정보 : 사건과 동작

- 전이를 나타내는 선 위에도 정보를 추가할 수 있다.

■ 촉발사건(trigger event) - 전이가 일어나는 원인을 제공하는 사건

■ 동작(action) - 실제로 수행되어 상태 변화를 일으키는 연산

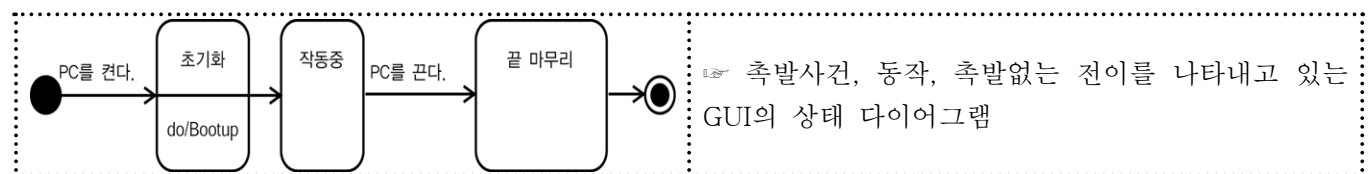
■ 출발없는 전이(triggerless transition) - 연관된 동작 없이 전이를 일으킬 수 있으며, 활동을 종료했기 때문에(사건 때문이 아니라) 일어나는 전이

GUI(그래픽 사용자 인터페이스)예

■ 초기화(initializing)

■ 작동중(working)

■ 끝 마무리(shutting down)



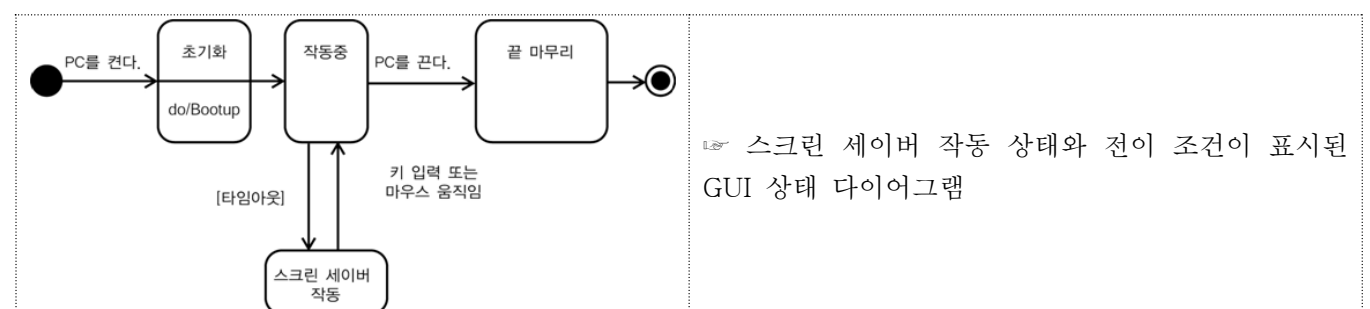
4-2)상태 전이선에 추가되는 정보 : 전이조건

※ 전이 조건의 예

- 컴퓨터를 켜놓기만 하고 아무 일도 하지 않고 있으면, 스크린 세이버가 작동하여 오랫동안 같은 픽셀이 모니터 화면에 남는 것을 막는다.

- 스크린 세이버가 작동하기까지의 시간 간격은 Windows 제어판에서 설정하며, 대개 10분이다

* 10분이라는 조건이 바로 전이 조건이다.



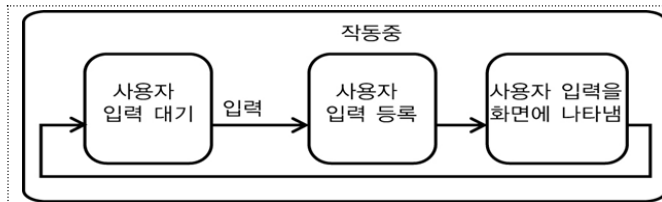
(5)하위 상태

- GUI의 예에서처럼 작동 중인 상태 안에 있는 동안에도 변화한다. 이러한 변화는 상태의 변화이며, 주어진 상태 내부에서 일어나는 것이기 때문에 **하위 상태(substate)**라는 이름으로 불린다. 하위 상태는 두 가지로 나뉜다. 하나는 **순차적(sequential)** 하위 상태이며, 또 하나는 **동시적(concurrent)**하위 상태이다.

5-1)순차적 하위 상태

- 순차적 하위 상태는 차례로 이어진다. 앞서 이야기한 GUI의 작동 중 상태 내에서 변화는 다음과 같은 순차적인 흐름으로 정리할 수 있다.

- 사용자 입력 대기
- 사용자 입력 등록
- 사용자 입력을 화면에 나타냄



GUI 작동중 상태 안의 순차적 하위 상태

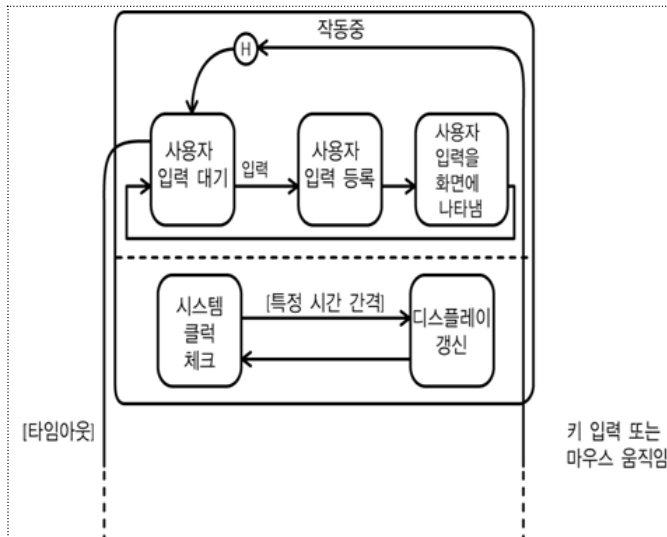
5-2)동시적 하위 상태

- GUI는 단지 입력만을 기다리고 있지는 않는다. 시스템 쿨럭도 체크하고 특정한 시간 간격이 지나면 애플리케이션의 디스플레이도 갱신해야 한다.

(6)이력 상태

- UML은 어떤 기호를 써서 복합 상태로 하여금 주어진 객체가 복합 상태를 벗어날 때 활성중인 하위 상태를 기억해 두도록 한다.

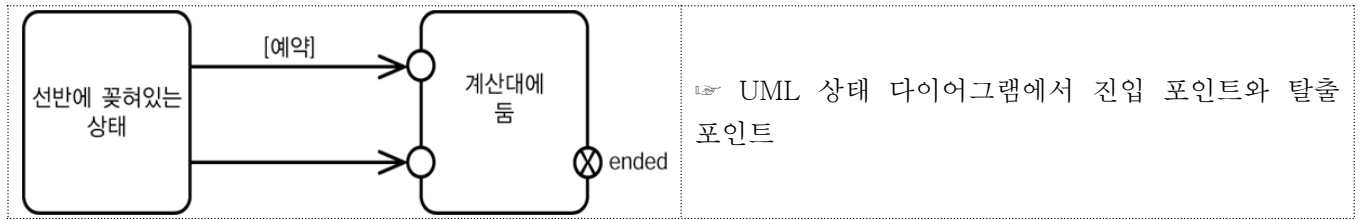
- 기호는 원으로 둘러싸인 “H”문자로서, 이 기호로 나타내는 상태를 **이력상태(history state)**라고 한다.



작은 원으로 둘러싼 H 문자로 나타내는 이력 상태는 주어진 객체가 해당 복합 상태를 벗어날 때 활성중인 하위 상태를 기억해 두도록 한다.

(7)UML 2.0에서 새로워진 것

- UML 2.0에서는 상태와 관련된 연결포인트(connection point)라는 새로운 기호를 제공한다. 상태에 진입하거나 상태에서부터 탈출하는 포인트를 의미한다.



(8)상태 다이어그램은 왜 중요할까?

- 한 객체에 일어날 수 있는 모든 변화의 양상을 잡아내어 모델링 가능(시스템 내의 객체 행동을 이해하는데 큰 도움)
- 개발자는 객체들이 어떻게 행동하는 지를 정확히 파악해야함

🌀 퀴즈

1. 상태 다이어그램의 클래스 다이어그램, 객체 다이어그램, 또는 유스케이스 다이어그램과 비교하여 근본적으로 다른 것은 무엇일까?
2. 촉발없는 전이란?

Part 9. 시퀀스 다이어그램

(1)시퀀스 다이어그램이란?

- 시퀀스 다이어그램(sequence diagram)이란, 객체(사각형으로 나타내며, 이름에 밑줄이 들어가 있는), 실선 화살표로 그려지는 메시지 그리고 수직 진행 상황을 나타내는 시간으로 구성되어 있다.

1-1)객체

- 객체는 시퀀스 다이어그램의 가장 윗 부분에 위치하며, 왼쪽에서 오른쪽으로 배열된다. 배열순서는 다이어그램을 간략하게 하는 방향으로 기준을 삼는다.

■ 생명선 - 각 객체로부터 아래로 뻗어가는 쇠선

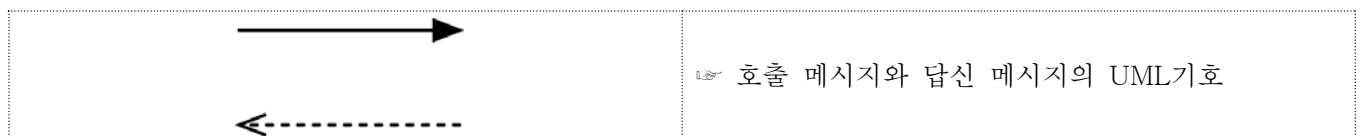
■ 실행 - 생명선을 따라 나타나는 좁다란 사각형



1-2)메시지

- 한 객체에서 다른 객체로 전송되는 메시지는 한 객체의 생명선에서 다른 객체의 생명선으로 이동하는 것을 의미한다. 객체는 자기 자신에게도 메시지를 보낼 수 있다. 즉 자신의 생명선에서 자신의 생명선으로 되돌아오도록 할 수 있다는 뜻이다.

■호출(call)메시지: 수신 객체가 오퍼레이션을 수행할 때까지 송신 객체는 기다리고 있다. 동기(synchronous) 메시지라고도 불린다.

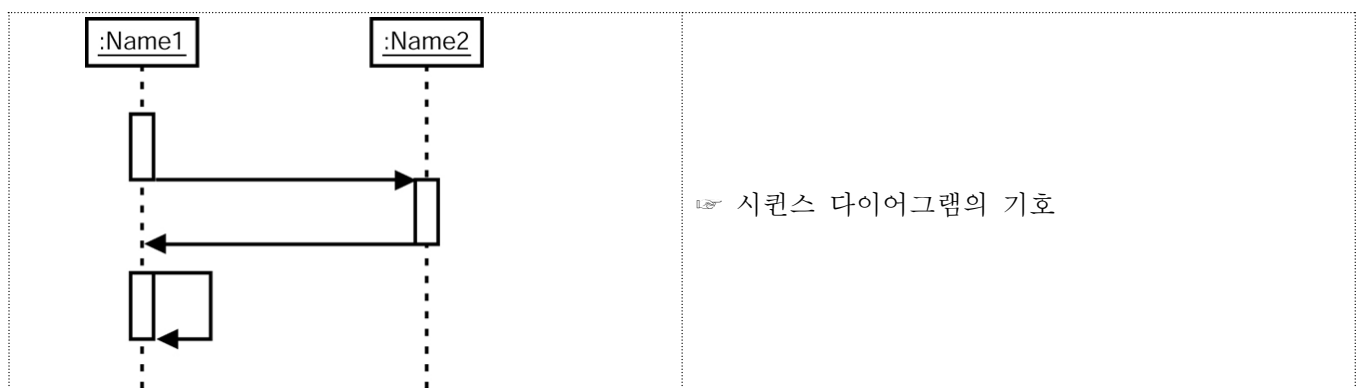


■비동기 메시지: 오퍼레이션이 완료되기를 기다리지는 않는다.



1-3)시간

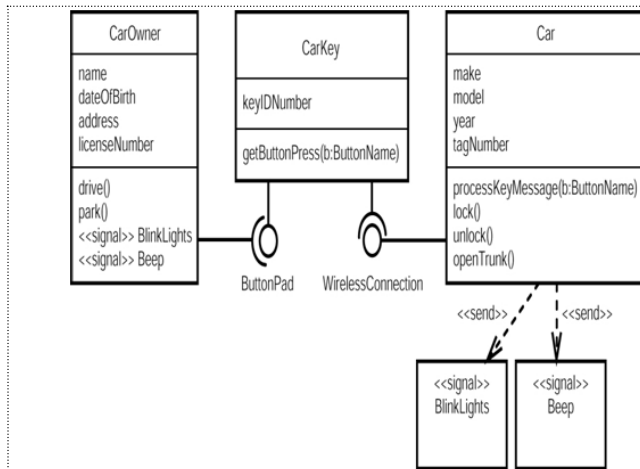
- 시퀀스 다이어그램은 시간을 수직 방향으로 나타낸다. 시간은 가장 윗부분에서 아래를 향해 흐르기 시작한다.



예) 자동차와 자동차 키

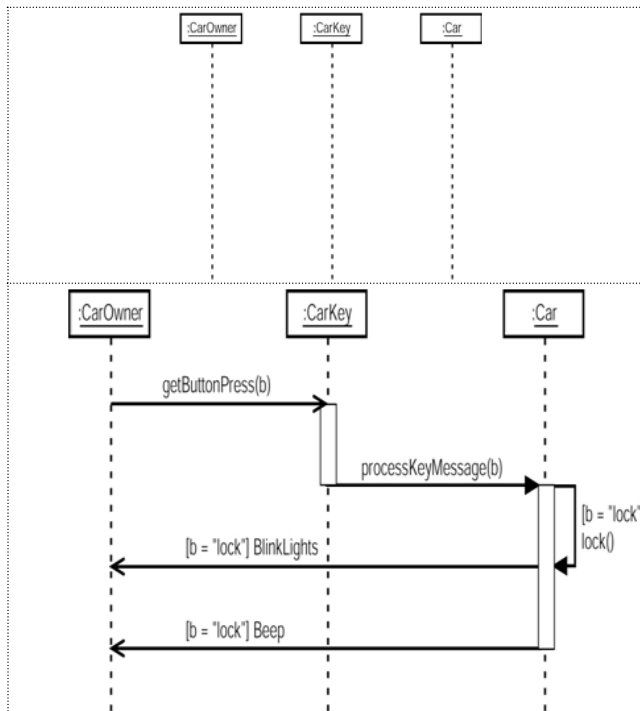
- 자동차키의 잠금(lock)버튼을 눌렀을 때 자동차는 스스로 잠긴다. 그리고 나서 라이트를 깜빡인다. 문이 잠겼다는 사실을 알리기위해 뽁! 소리를 낸다.

클래스 다이어그램



자동차 주인과 차키와 차 사이의 관계

시퀀스 다이어그램



세 개의 객체를 그리는 것으로 시작해 보자.

메시지가 시퀀스 다이어그램을 완성시킨다.

음료수 자동 판매기

프론트가 하는 일

- 음료수 선택을 받아들이고 돈을 받는다.
- 음료수 없음 또는 돈이 맞지 않음 같은 문구를 보여준다.
- 금전등록기에서 잔돈을 받아 고객이 사용 가능 하도록 한다.
- 돈을 돌려준다.
- 디스펜서로부터 음료수 캔을 받아 고객에게 준다

금전 등록기가 하는 일

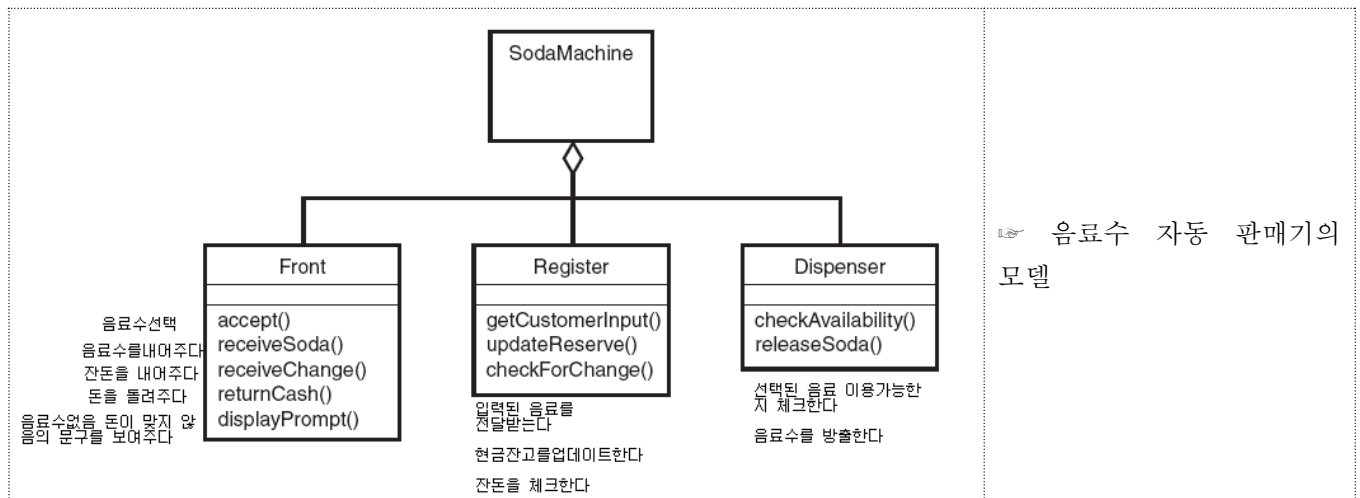
- 프론트로부터 고객의 입력(음료 선택과 돈)을 전달 받는다.
- 현금 잔고를 업데이트 한다.

■ 잔돈을 체크한다.

디스펜서가 하는 일

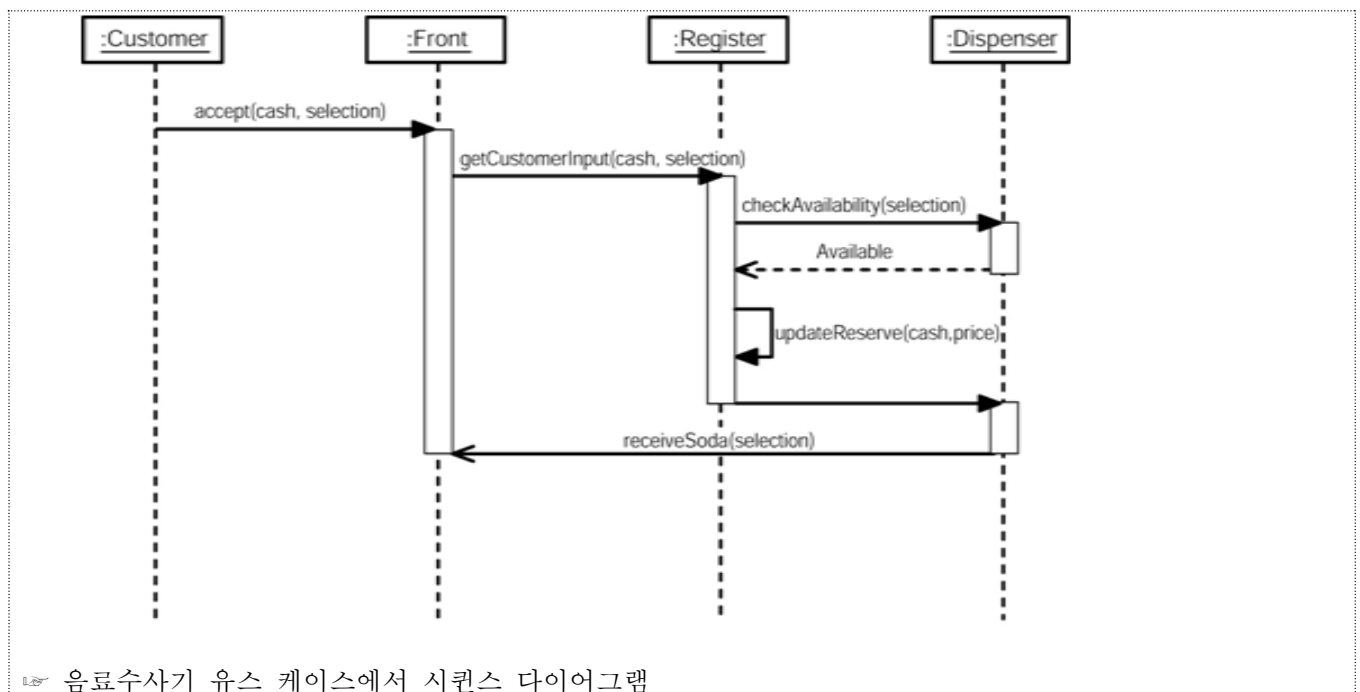
■ 선택된 음료가 이용 가능한지 체크한다.

■ 음료수 캔을 방출한다.

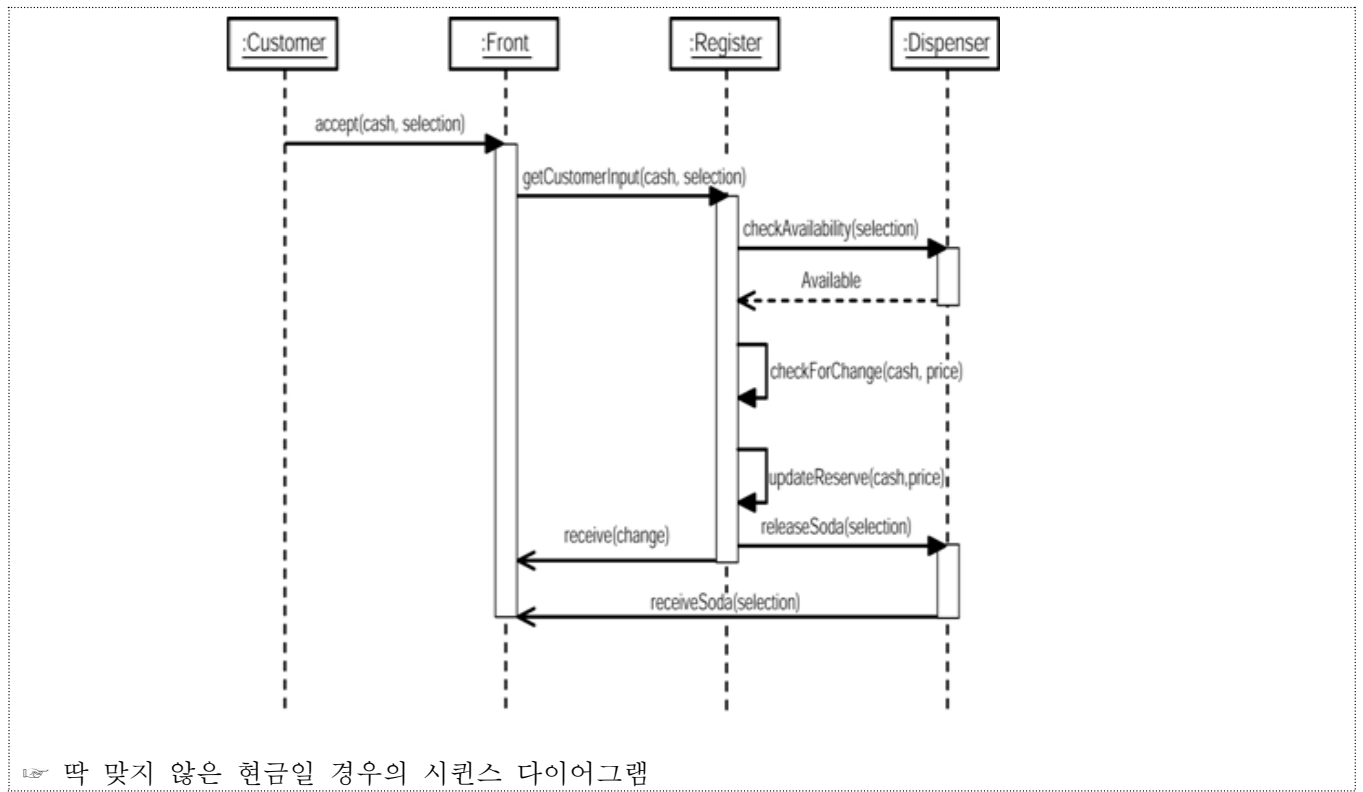
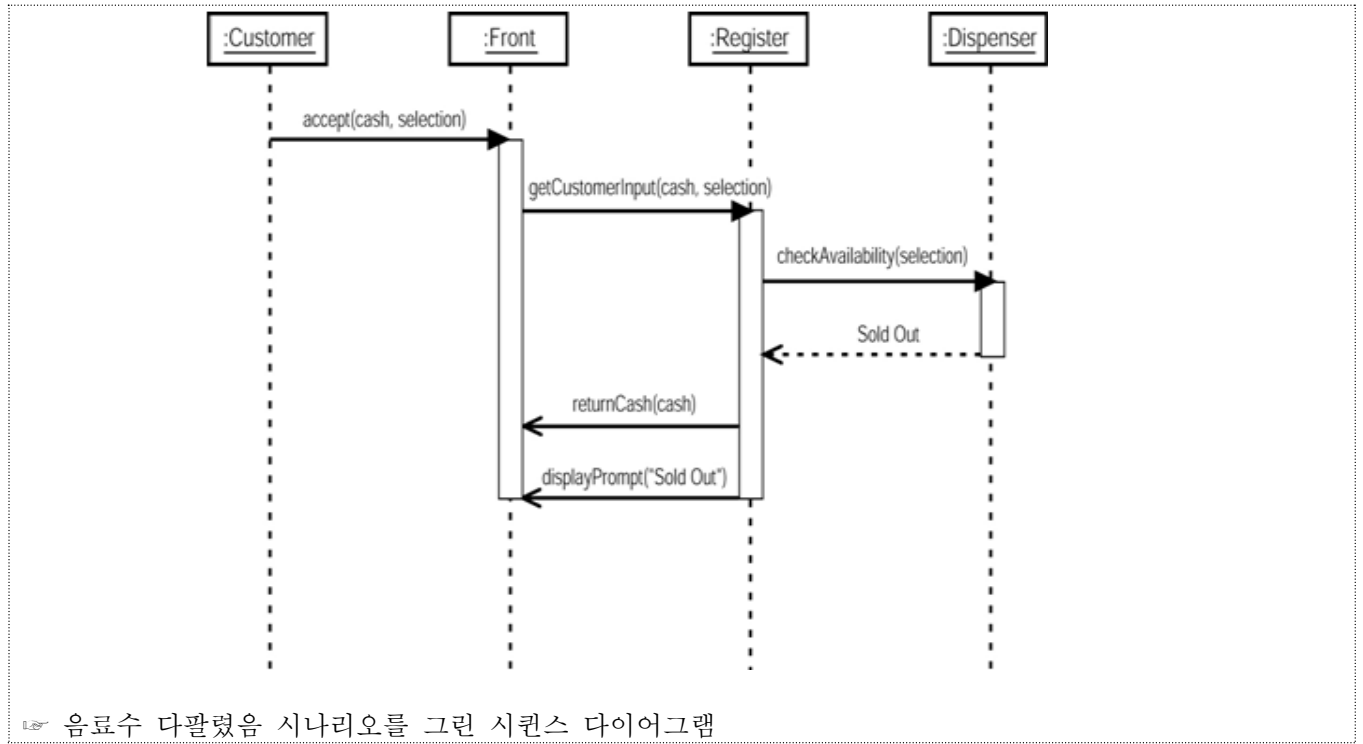


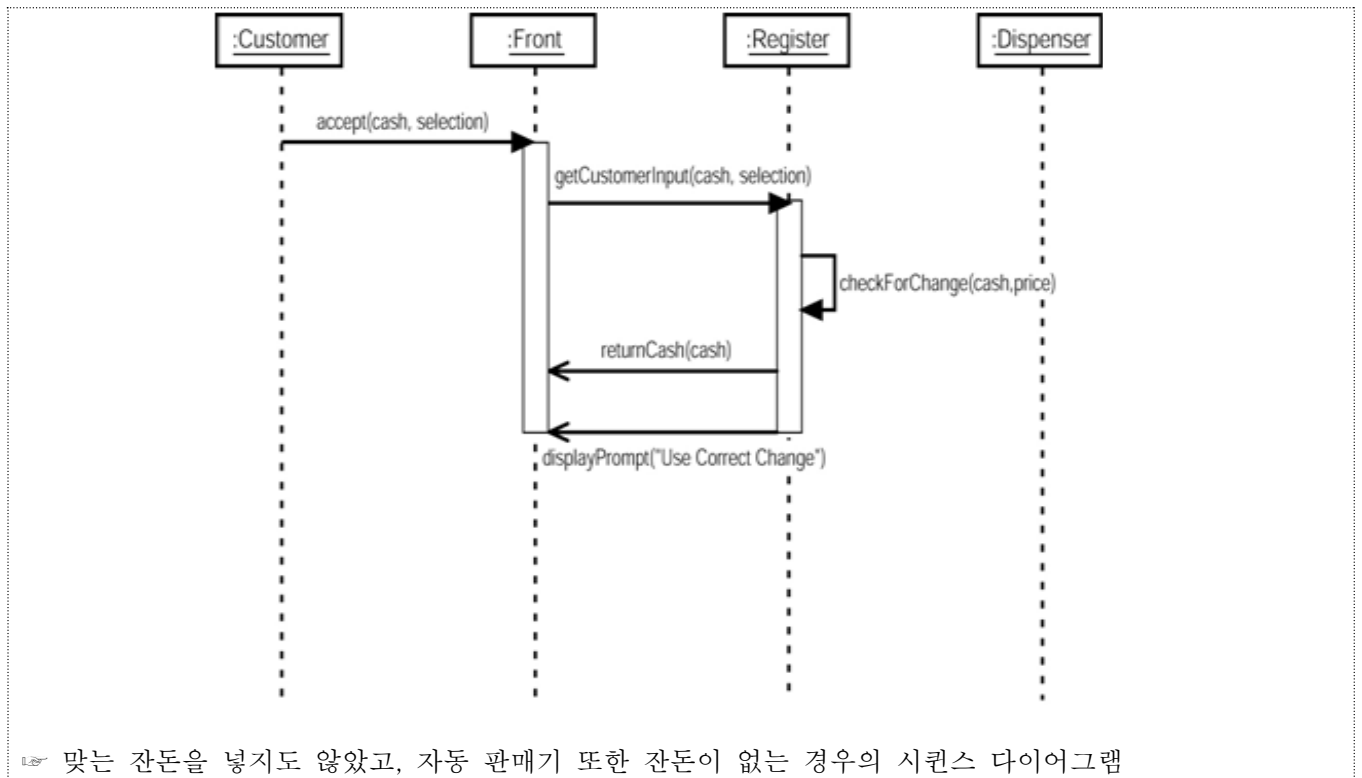
음료수 사기의 간단한 시나리오

1. 고객이 자동 판매기의 프론트에 있는 동전 투입구에 돈을 넣고 음료를 선택한다
2. 돈이 금전 등록기로 전달되어 스스로 업데이트 한다.
3. 지금 이것은 최고의 시나리오이기 때문에 선택한 음료는 이용 가능한 것이며, 디스펜서는 이 음료를 자동 판매기의 프론트로 방출한다.



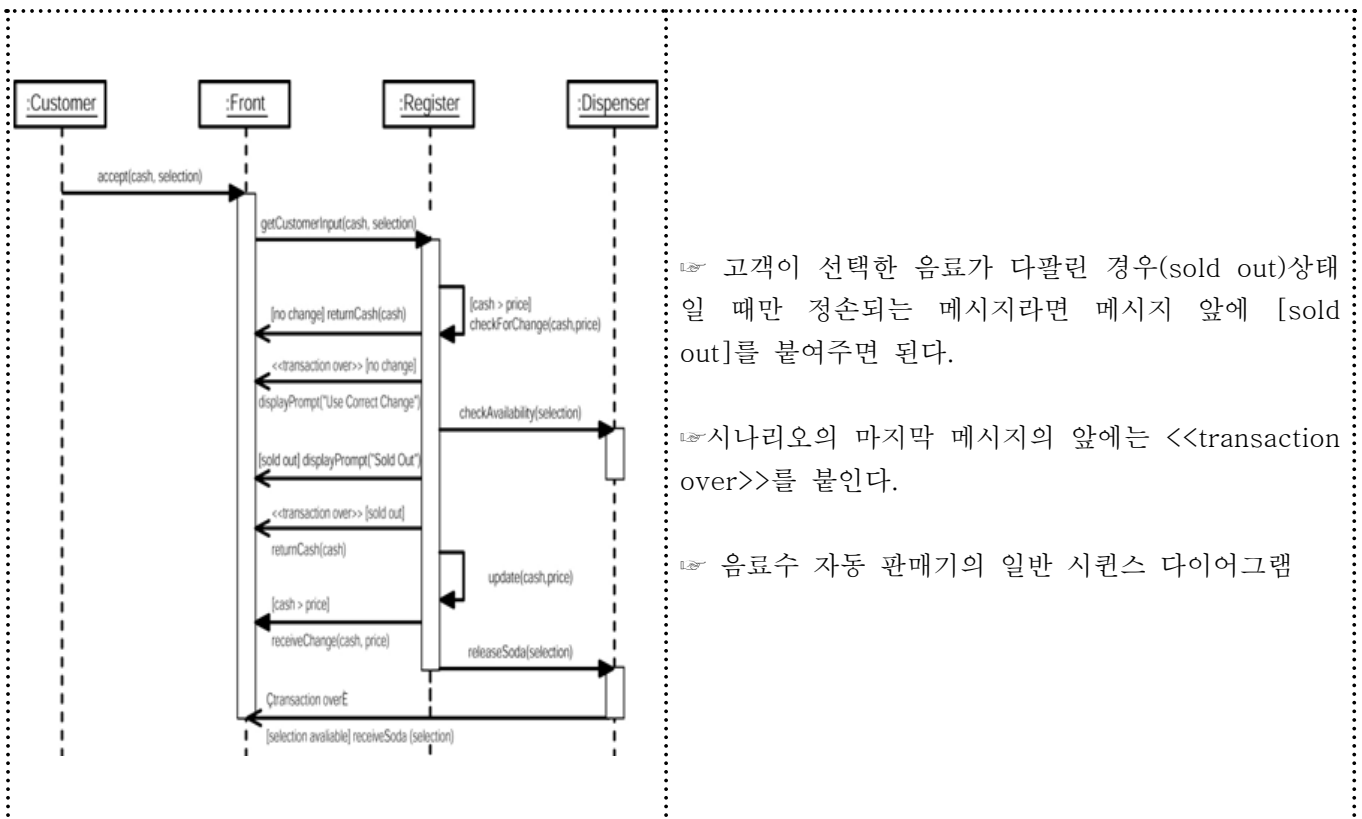
☞ 다른 시나리오로써 고객이 선택한 음료가 다 팔렸을 수도 있다. 이렇게 다 팔렸을 경우(sold out)의 시나리오를 그린 시퀀스 다이어그램이다.





(2) 시퀀스 다이어그램 : 일반 시퀀스 다이어그램

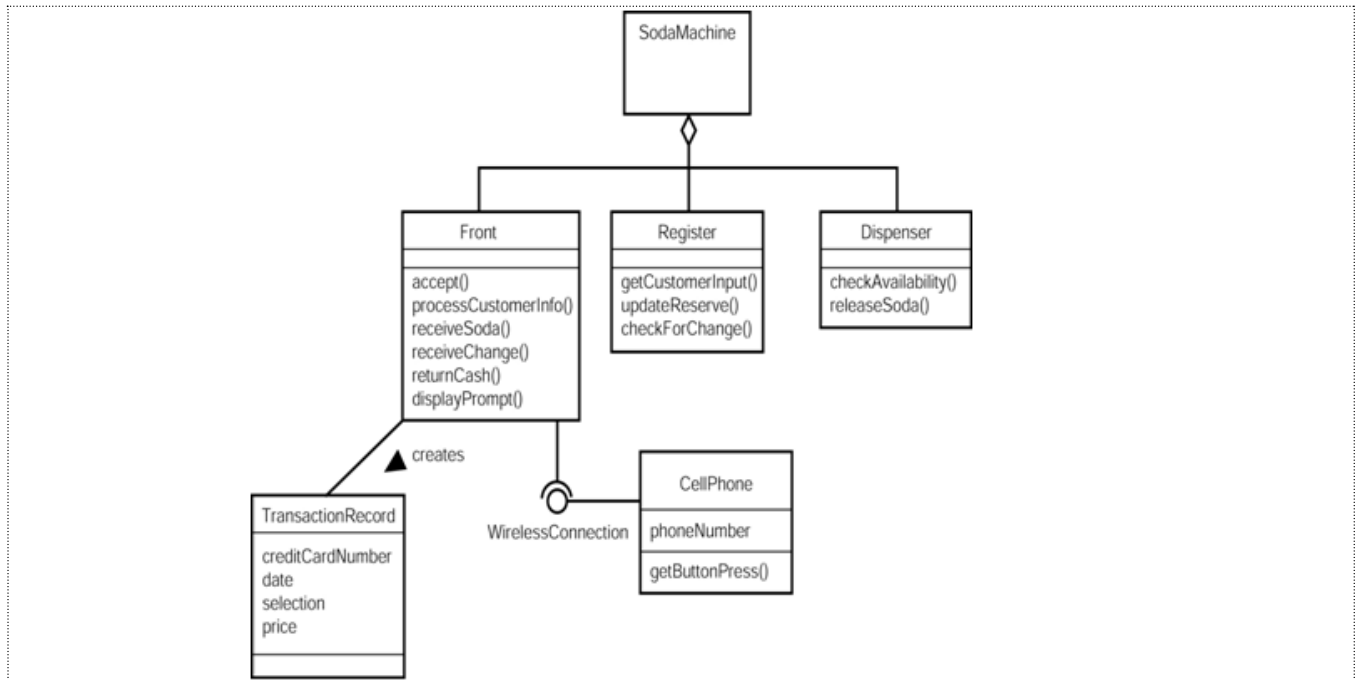
- 지금 까지는 하나의 시퀀스 다이어그램에 하나의 시나리오만을 그려왔다. 이러한 시퀀스 다이어그램을 **인스턴스 시퀀스 다이어그램(instance sequence diagram)**이라고 한다.
- 만약 하나의 시퀀스 다이어그램에 모든 시나리오를 다 포함시킨다면 그 시퀀스 다이어그램은 **일반 시퀀스 다이어그램(generic sequence diagram)**이 된다.



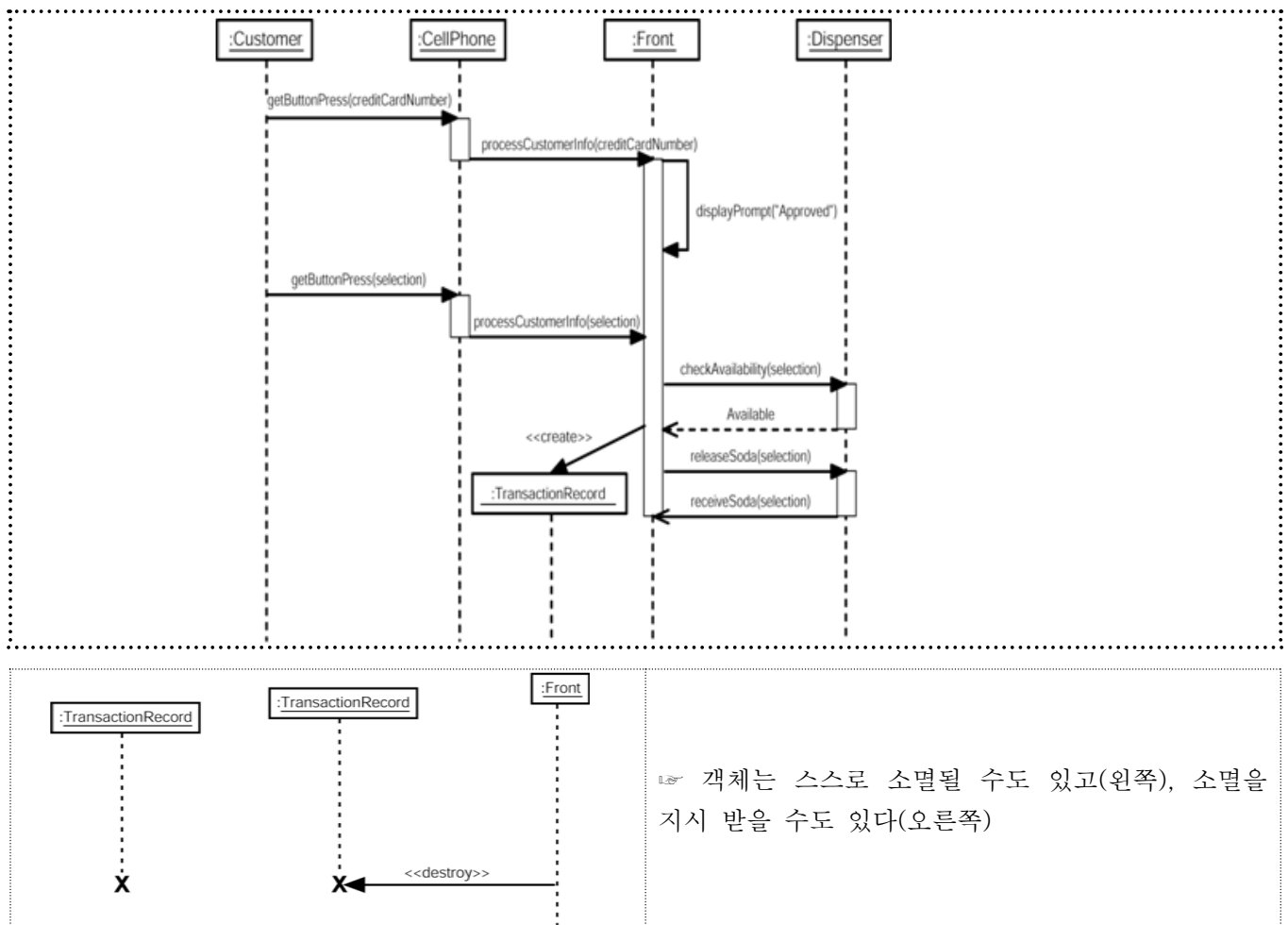
(3)시퀀스 내에서 객체 생성하기

☞시나리오

- 음료수자판기에서 돈을 이용하지 않고 휴대폰(신용카드)을 통해 음료수 자판기 이용하는 기술을 모델링하려고 한다. 우선 휴대폰을 이용해 음료수 자동판매기를 이용하는 클래스 다이어그램을 보자



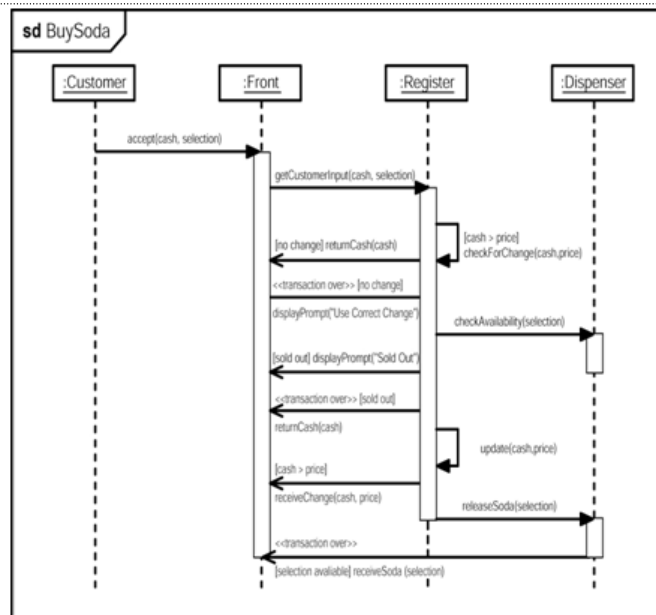
☞ 휴대폰(신용카드)를 통해 음료수 자동판매기를 이용하는 시나리오를 이용한 시퀀스 다이어그램



(4)시퀀스 프레임 넣기 : UML2.0의 시퀀스 다이어그램

- UML 2.0에서는 시퀀스 다이어그램을 좀 더 유용하게 사용할 수 있도록 추가된 것이 있다. 여러분은 시퀀스 다이어그램의 가장 자리를 선으로 긋고 왼쪽 위에 칸막이(compartment)를 둬으로써 시퀀스 다이어그램을 프레임에 넣을 수가 있게 된다. 왼쪽 위의 공간은 다이어그램을 구분지을 수 있는 정보가 들어간다.

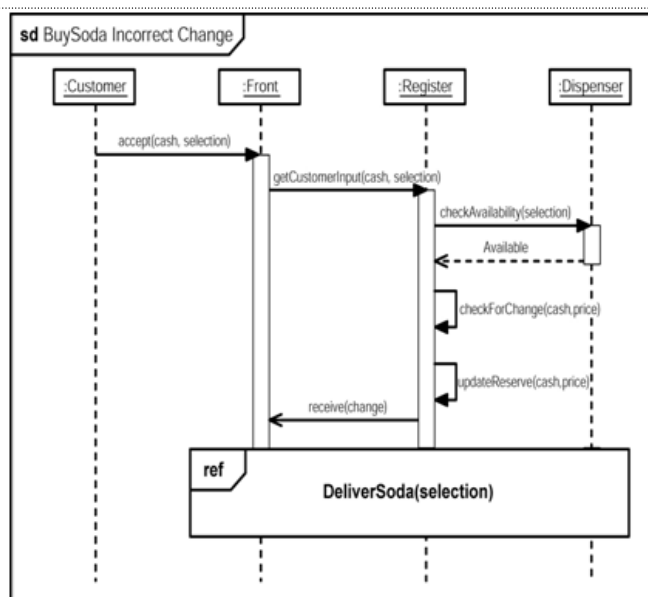
-그 정보들 중에는 우선 프레임 내의 다이어그램 형태를 나타내는 **조정자(operator)**가 포함한다.



UML 2.0에서 시퀀스 다이어그램 프레임 넣기

4-1)교류발생

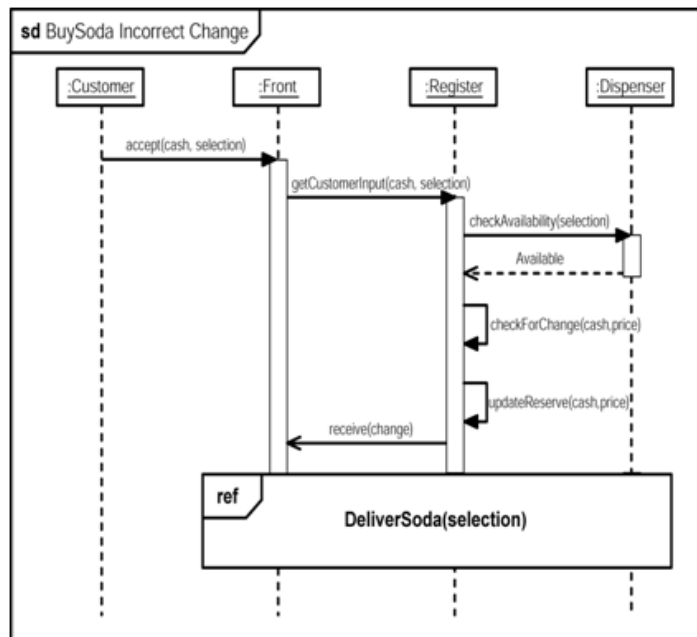
- 유스케이스의 시나리오를 일반 시퀀스 다이어그램이 아닌 인스턴스 시퀀스 다이어그램을 생성했다면 다이어그램에서 다이어그램으로 상당량 복사해야할 부분이 생길 것이다. 프레임은 한 시퀀스 다이어그램의 일부를 다른 다이어그램에서 쉽고 빠르게 재사용할 수 있도록 도와준다. 다이어그램의 한 부분은 프레임으로 묶고, 그 프레임의 좌 상단 부분에 마찬가지로 칸막이를 둔다. 그리고 새로운 다이어그램에 이 프레임(메시지와 생명선을 제외하고) 삽입한다. 이 특별한 프레임 부분은 **교류 발생(interaction occurrence)**이라고 하며 조정자는 **ref**로 표현한다.



시퀀스 다이어그램에서 교류 발생 부분 프레임 넣기

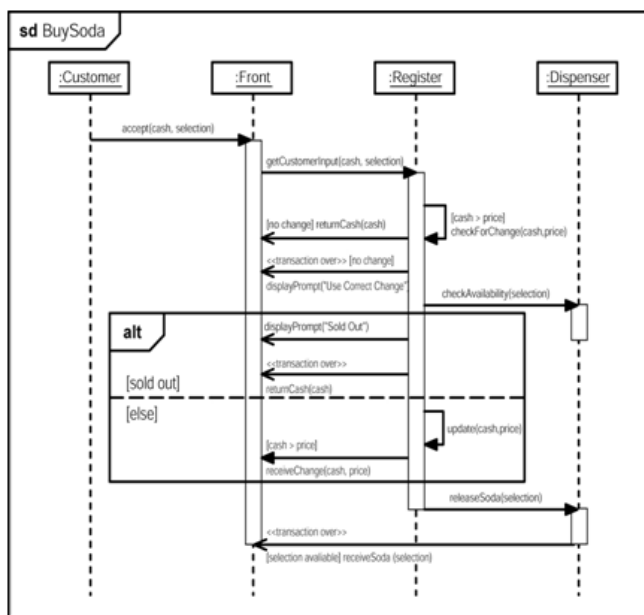
4-2)합쳐진 교류 조각

- 교류 발생은 교류 조각(interaction fragment : 시퀀스 다이어그램의 한 부분은 일컫는 UML 2.0의 일반적인 이름)의 특별한 케이스이다. 여러분은 이러한 교류 조각을 여러 방법으로 결합 시킬 수가 있고, 조정자로 결합 형태를 나타낸다. 조각 주변을 프레임으로 두르고 연결된 교류 조각을 점선으로 나눔으로써 표현한다.

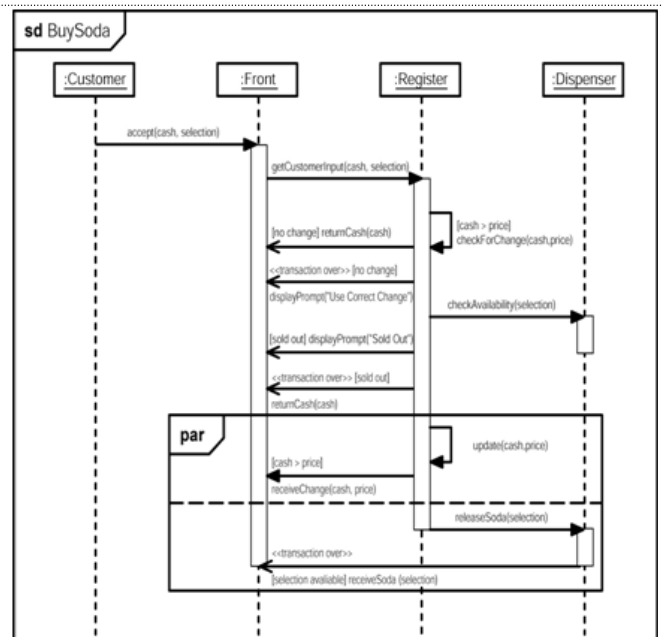


☞ 교류 발생 부분 재사용하기

- 가장 널리 쓰이는 결합 방법 표현에는 alt조정자와 par조정자, 두가지를 꼽는다.



☞ alt결합에서는 결합된 조각 중 특정 조건을 만족하는 하나만 처리된다. 전이 조건을 이용하여 처리되는 조각이 무엇인지 구분지어 준다.



☞ par조정자는 결합된 조각들이 서로 방해하지 않고 동시에 활동함을 나타낸다.

퀴즈

1. 동기 메시지와 비동기 메시지를 정의해 보자.
2. UML2.0에서 교류 조각(interaction fragment)은 무엇을 의미하는가?
3. 시퀀스 다이어그램에서 객체 생성은 어떻게 나타낼까?

Part 10. 통신 다이어그램

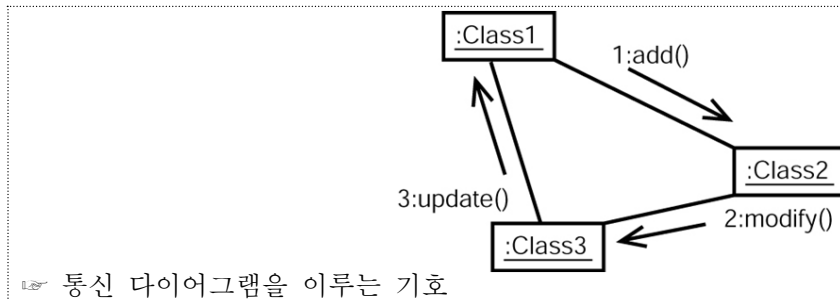
(1)통신 다이어그램이란?

- 객체 다이어그램은 객체 그 자체와 다른 객체와의 관계를 보여주는 그림이다. 통신 다이어그램은 객체 다이어그램을 확장한 것이다. 객체 사이의 연관 관계 뿐만 아니라, 각 객체들이 주고 받는 메시지들을 나타낸다. 혼란스러움을 막기 위해서 연관 관계의 이름을 적는 것은 대개 생략해버린다.

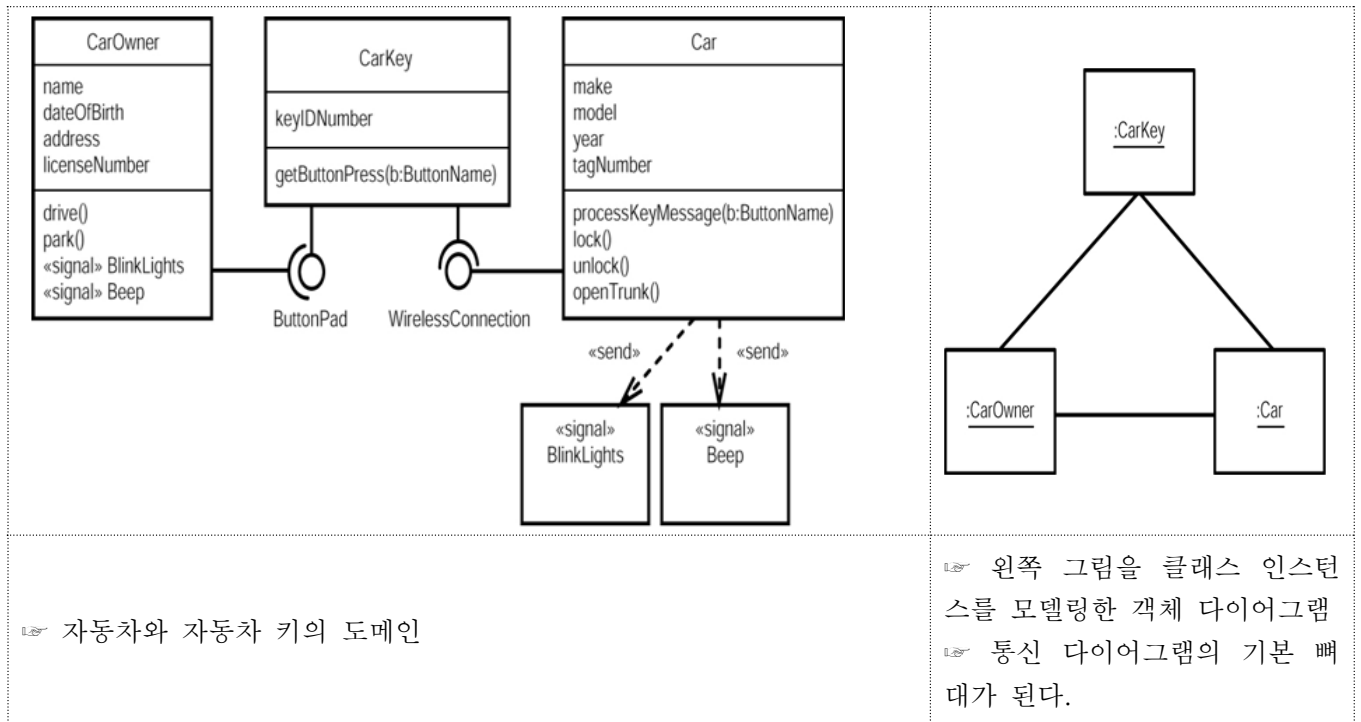
■ 객체 다이어그램 - 스크린샷

■ 통신 다이어그램 - 영화

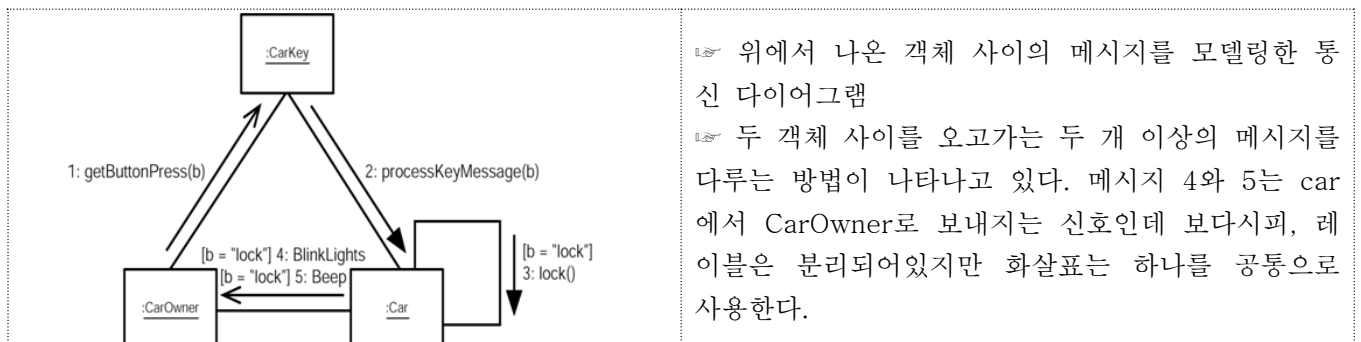
- 시퀀스 다이어그램을 통신 다이어그램으로 바꿀 수 있으며, 그 반대도 가능하다.



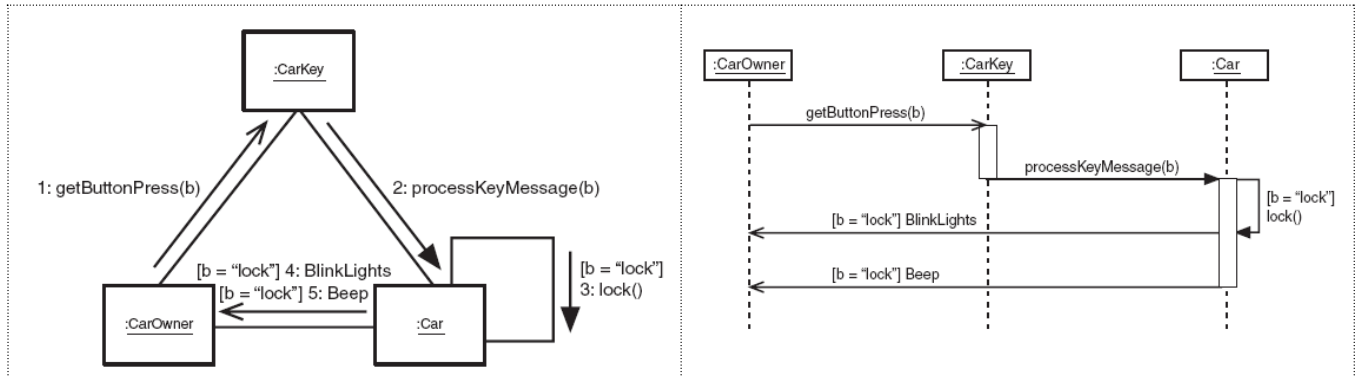
1-1)통신 다이어그램 예제 (자동차와 자동차 키)



1-2)통신 다이어그램(메시지 추가)

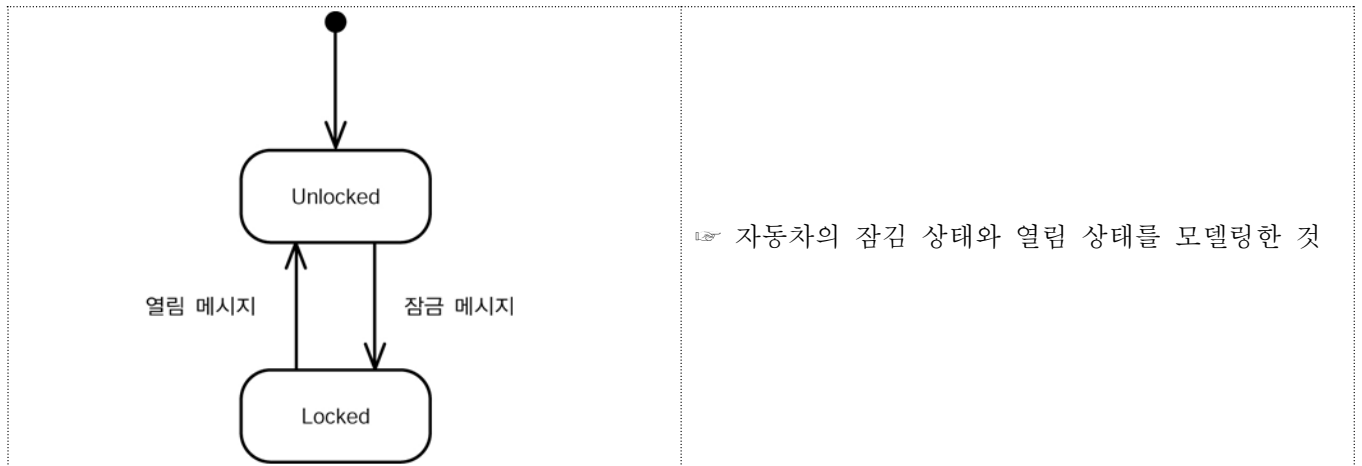


1-3)자동차와 자동차키에 대한 시퀀스 다이어그램과 통신 다이어그램



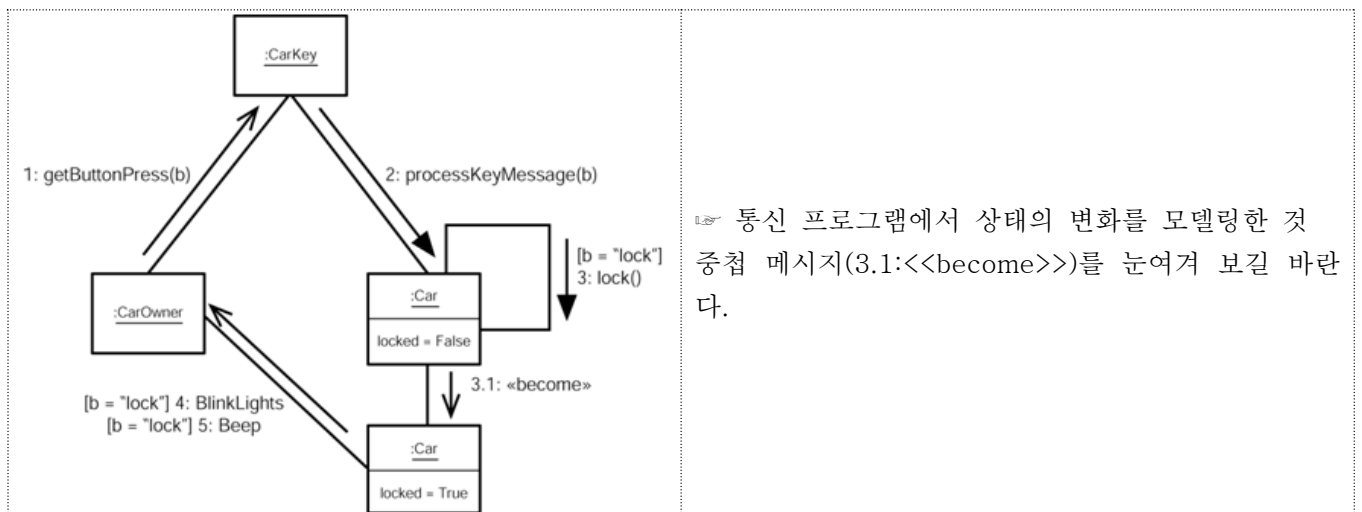
(2)상태 바꾸기와 중첩 메시지

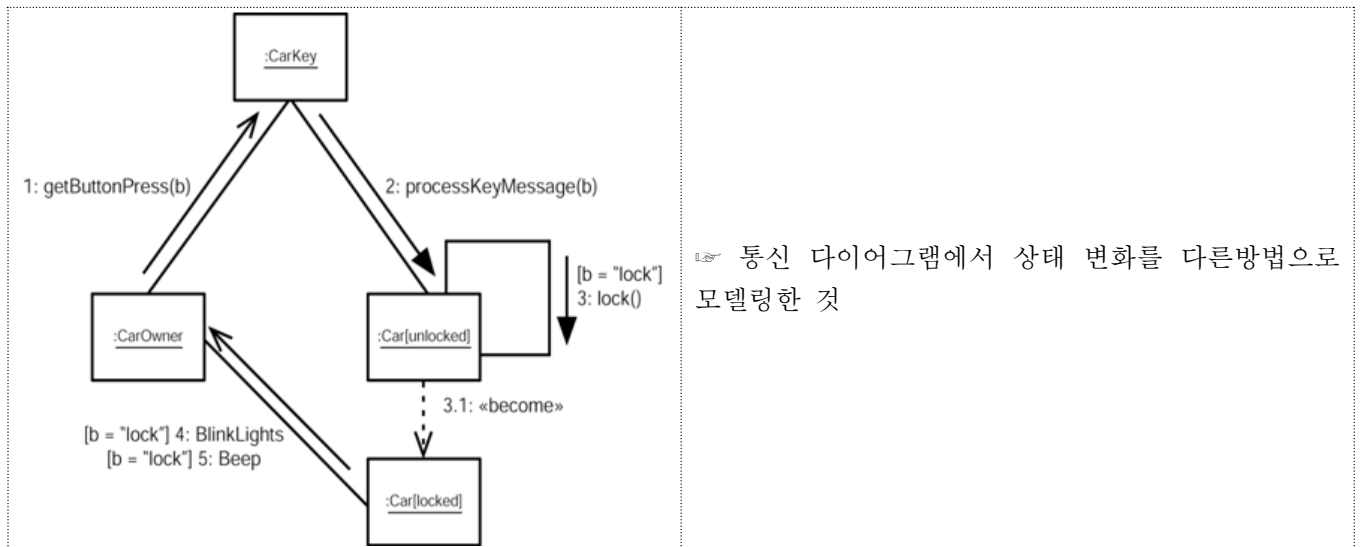
- car 객체가 true 또는 false의 값을 갖는 locked 속성을 가지고 있다고 가정하자



- 통신 다이어그램에서도 상태의 변화를 보여줄 수가 있다. 그러기 위해서는 우선 car 객체의 locked값을 보여준다. 그리고 새로운 값이 locked를 갖는 똑같은 car객체를 복사한다. 두 객체를 연결하고 첫 번째에서 두 번째 객체로 `<<become>>` 키워드를 레이블로 갖는 메시지를 보낸다.

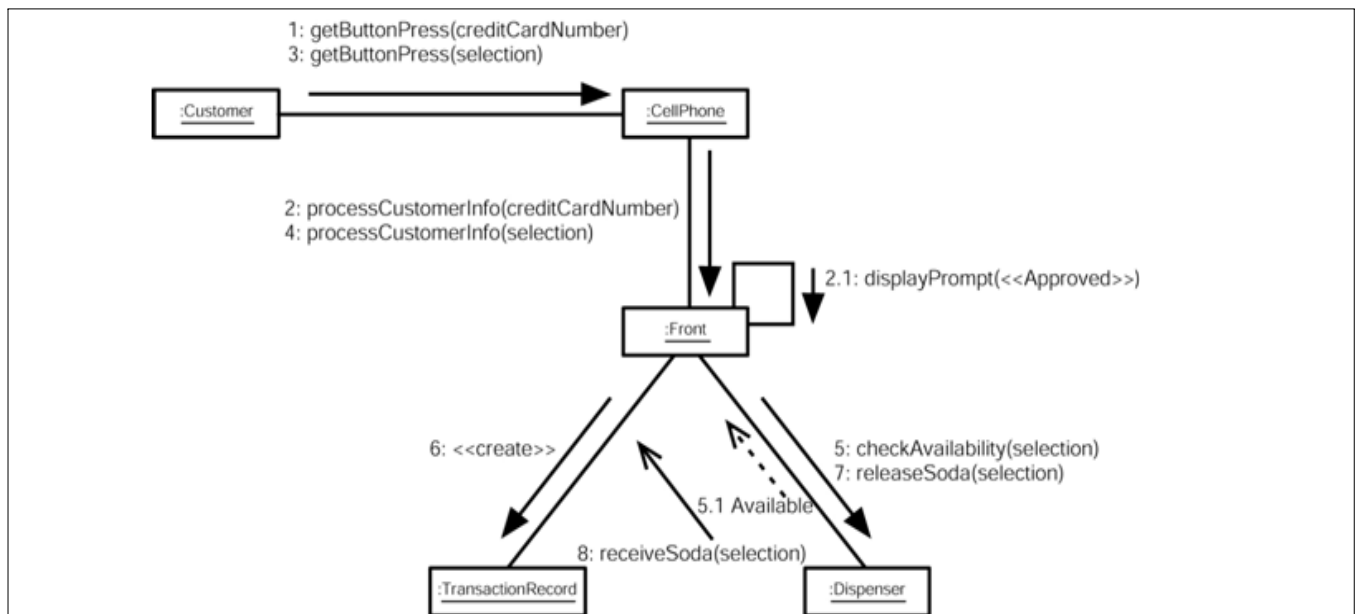
- 중첩 메시지 표현법 : 중첩될 대상의 메시지 숫자와 똑같은 숫자를 붙여주고 소수점을 붙인 후 중첩되는 메시지의 고유 숫자를 붙여 주면 된다.



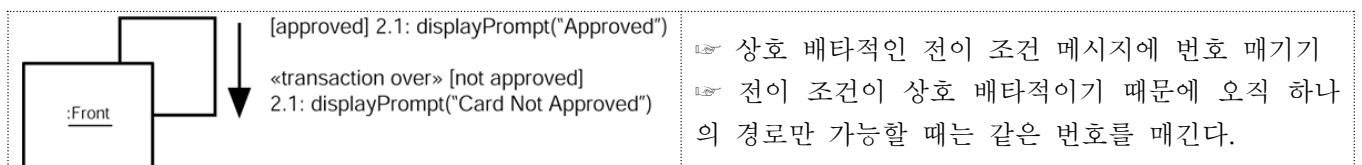


(3)객체의 생성

- 객체의 생성을 이야기 하기 위하여 다시 휴대폰으로 이용 가능한 음료수 자동 판매기의 예제로 다시 돌아가 보자. 생성된 객체는 트랜잭션 기록으로써 자동 판매기가 고객의 계정에서 가격을 지불 받을 수 있는 수단이다. 객체 생성을 모델로 나타내기 위해서 메시지 레이블에 <<create>>를 넣으면 된다.

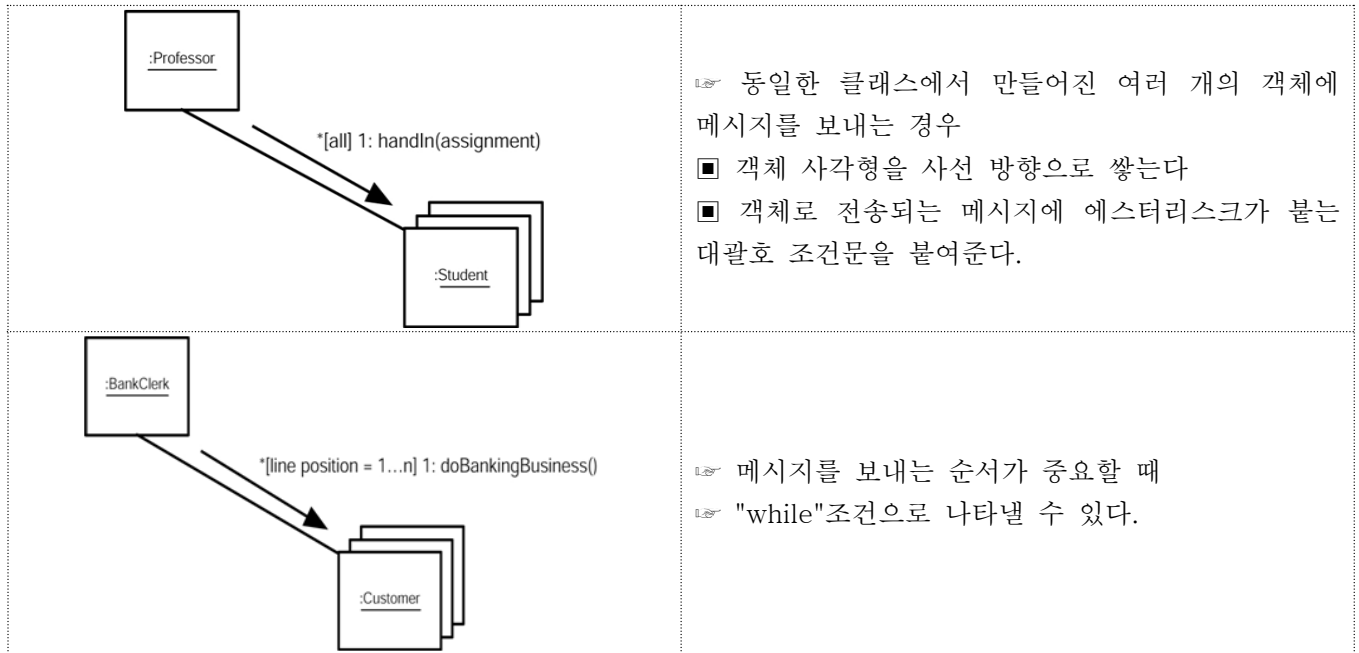


3-1)상호 배타적인 전이 조건 메시지에 번호 매기기

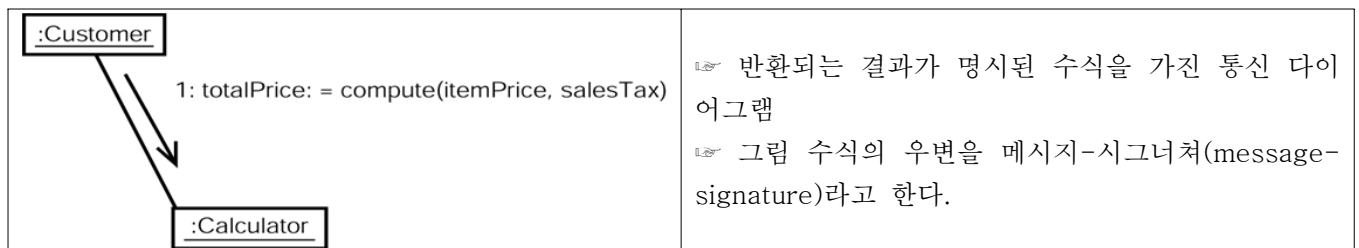


(4)그 외의 개념들

4-1)여러 객체로 메시지 전송하기



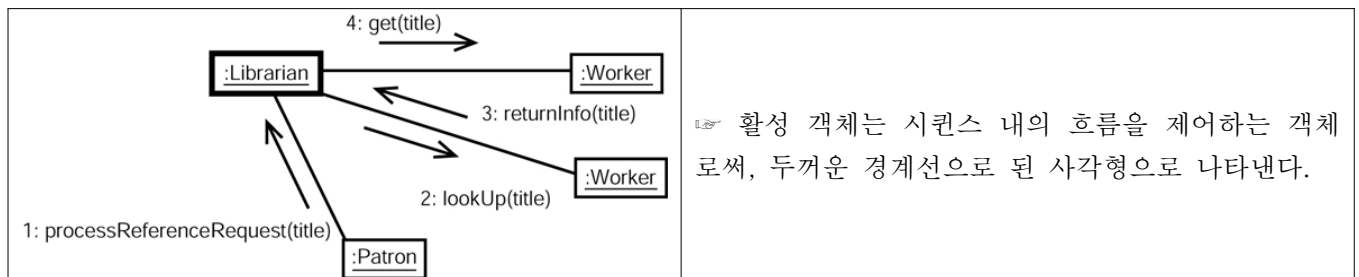
4-2)반환된 결과 나타내기



4-3)활성 객체

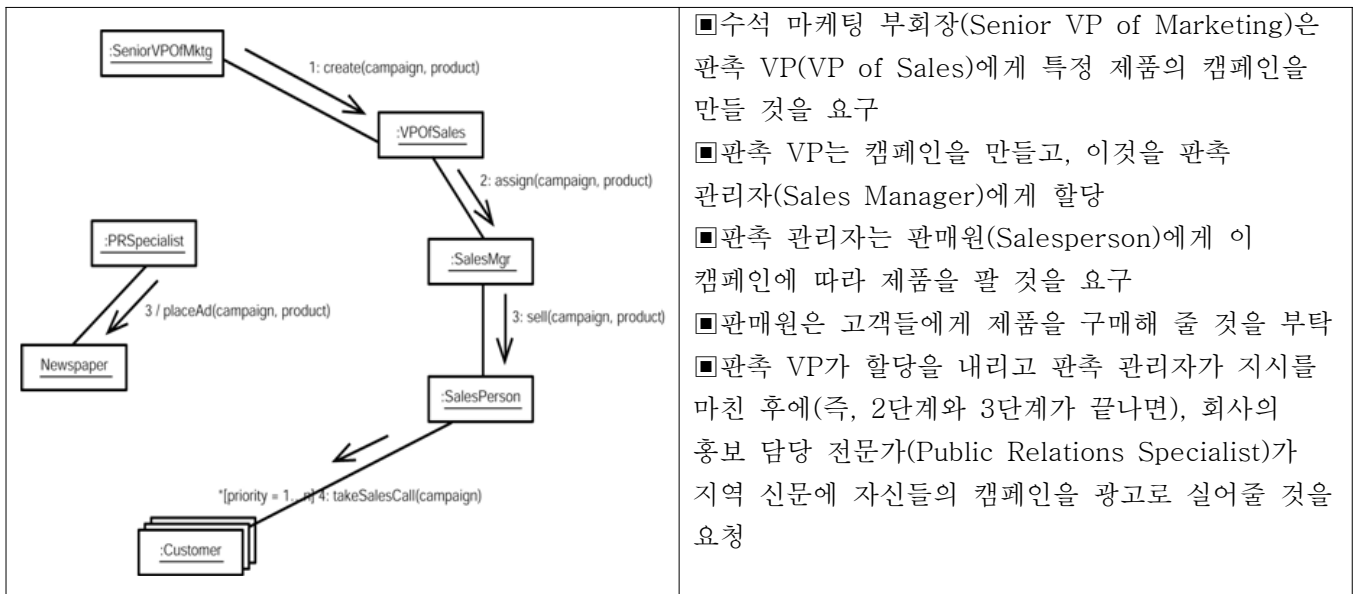
- 객체간의 교류에 있어서 특정한 객체가 흐름을 제어하는 경우가 있다. 흐름을 제어하는 이객체를 **활성 객체(active object)**라고 하며, 활성 객체는 **수동 객체(passive object)**에게 메시지를 보내며 다른 활성 객체들과 교류한다.

- 시스템 내에서 두 개 이상의 객체가 동시에 작동될 수 있는데, 이것을 **동시성(concurrency)**이라고 한다.



4-4)동기화

- 다른 몇 개의 메시지(비연속적으로 전송될 수도 있다.)를 받은 후에야 메시지를 전송하는 객체가 있을 수 있다. 즉, 이 객체는 다른 메시지들과 자신의 메시지를 **"동기화(synchronize)"**해야 한다.

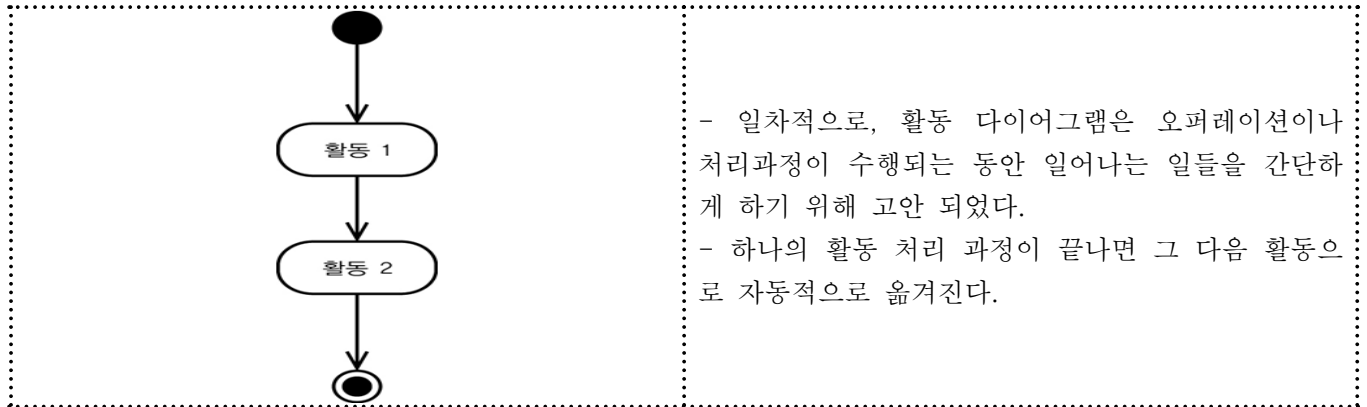


퀴즈

1. 통신 다이어그램에서는 메시지를 어떻게 나타낼까?
2. 통신 다이어그램에서는 순차적인 흐름 정보를 어떻게 나타낼까?
3. 상태의 변화는 어떻게 나타낼까?
4. 시퀀스 다이어그램과 통신 다이어그램의 “의미상 동등성”은 무슨 의미 일까?

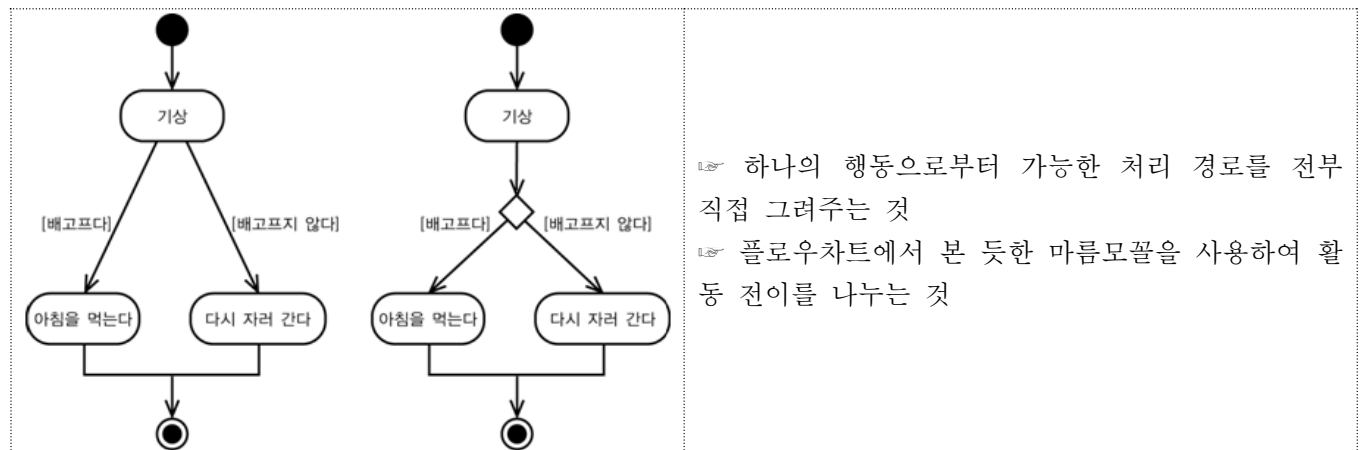
Part 11. 활동 다이어그램

(1) 활동 다이어그램이란?



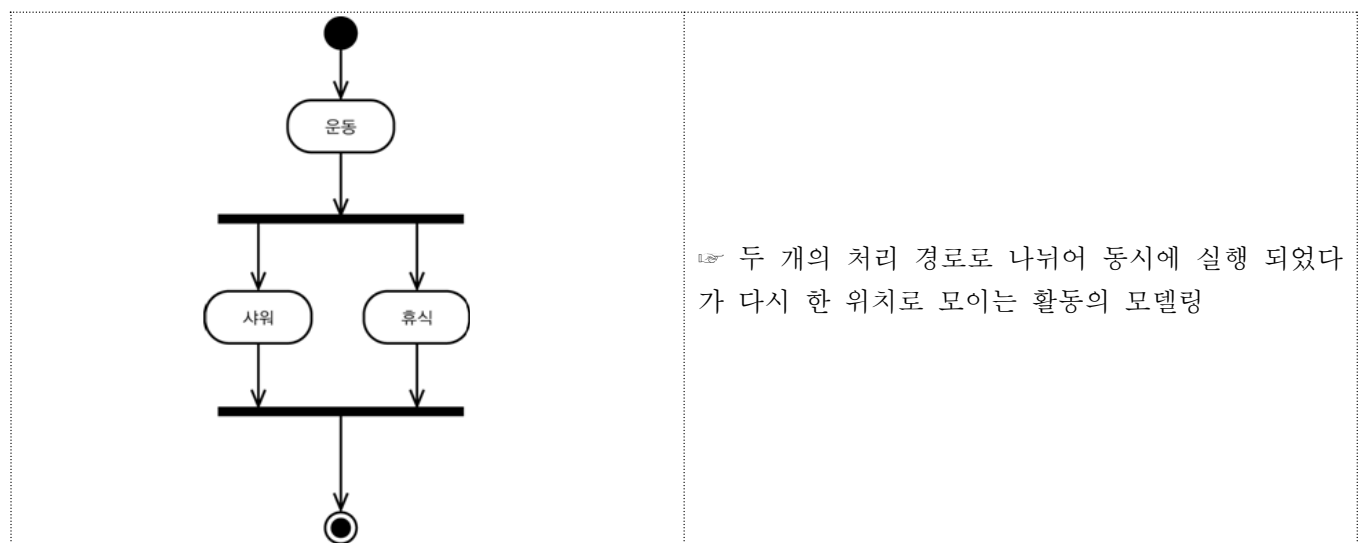
(2) 결정

- 활동이 진행되다 보면 결정이 필요한 지점이 반드시 있게 마련이다. 어떤 조건이냐에 따라 다른 처리 경로가 이어지며, 이렇게 해서 갈라진 두 개의 경로는 서로 배타적이다.
- 결정 위치는 두 가지 방법중 하나를 사용



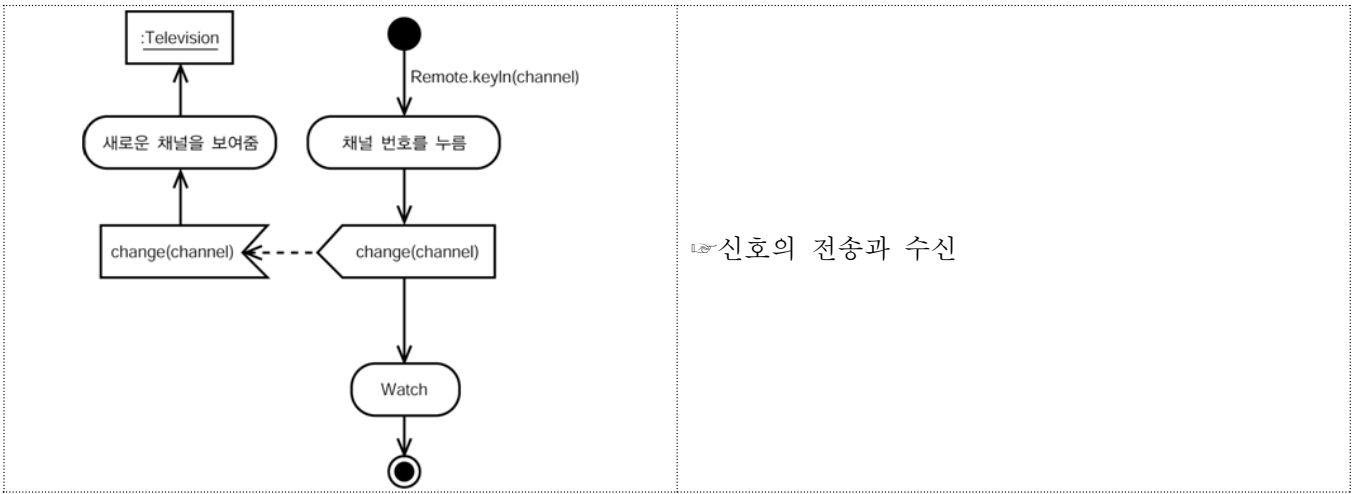
(3) 동시 경로

- 동시에 실행되었다가 하나로 모이는 두 개의 처리 경로로 활동 전이를 분리해야 할 경우



(4)신호

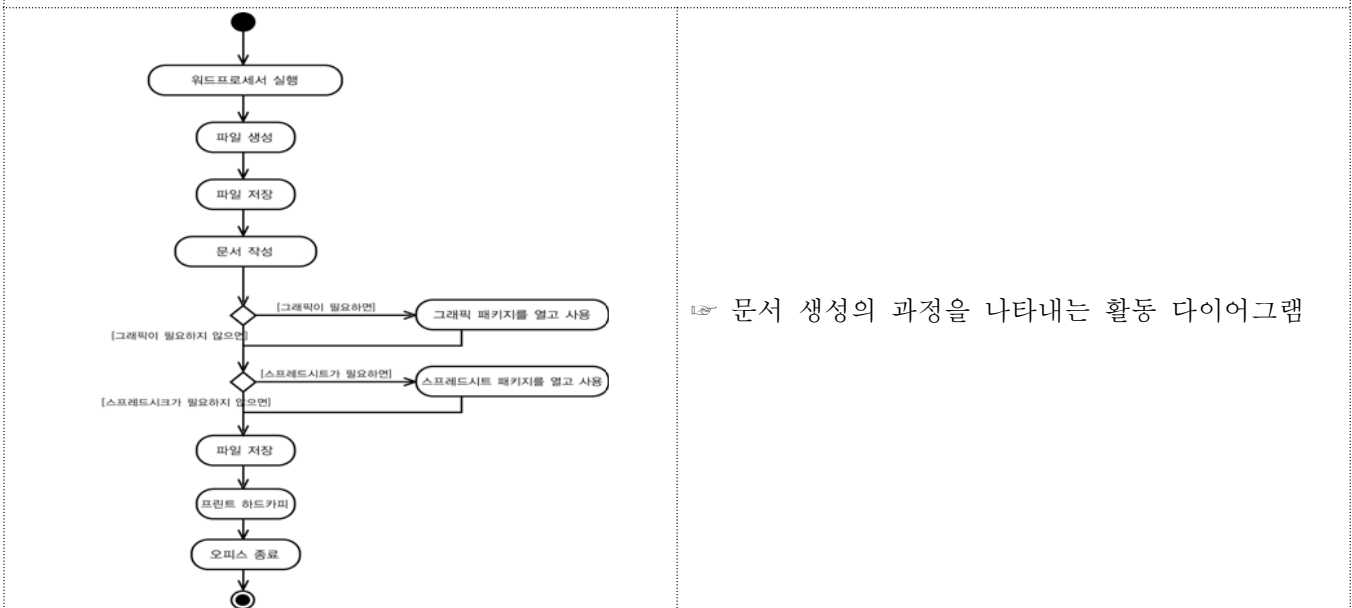
- 활동이 진행되는 도중에 신호(signal)를 보낼 수도 있다. 신호가 보내어지면 그 신호를 받은 쪽은 활동을 개시해야 한다. 신호의 전송을 뾰족한 오각형(convex pentagon)으로 나타내며- 신호의 수신은 쐐기 모양의 파인 다각형으로 나타낸다.
- UML에서는 뾰족한 오각형이 **출력사건(output event)**을 나타내며, 쐐기 모양의 파인 다각형은 **입력사건(input event)**을 나타낸다.



(5)활동 다이어그램 적용의 예

☞ 문서 만들기

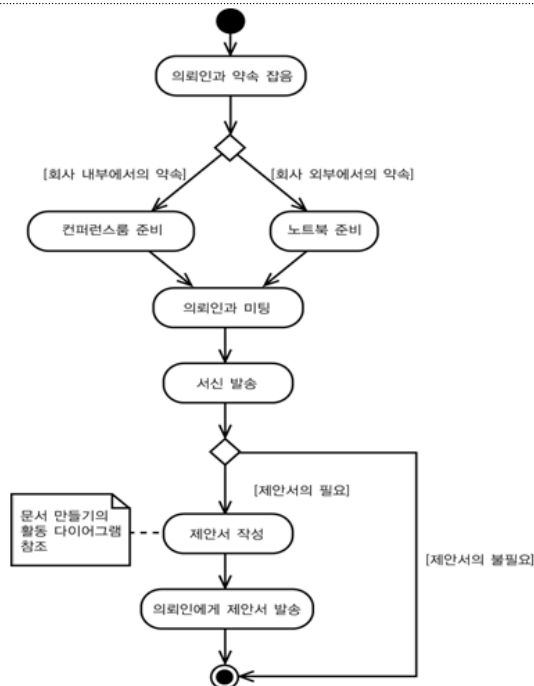
1. 워드프로세서를 실행시킨다.
2. 파일을 생성한다.
3. 디렉토리에 다른 이름으로 파일을 저장한다.
4. 문서를 작성한다.
5. 그래픽이 필요하면 그래픽 패키지를 열고 그림을 그린 다음, 문서에 붙여넣기를 시도한다.
6. 스프레드시트가 필요하면 스프레드시트 패키지를 열고 스프레드시트를 만든 다음, 문서에 붙여넣기를 시도한다.
7. 파일을 저장한다.
8. 이 문서를 프린터로 하드 카피한다.
9. 오피스 소프트웨어를 종료한다.



- 활동 다이어그램의 사용은 처리 과정에 속해 있는 각 활동의 책임이 누구에게 있는지를 나타내어 줄 수 있다는 점에서 더 빛을 발한다.

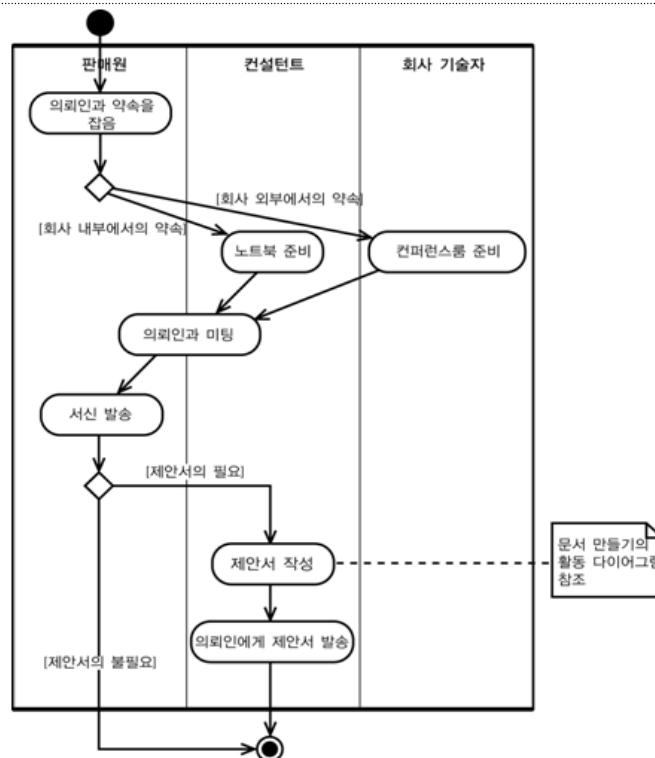
☞ 컨설팅 회사에서 새 의뢰인과 미팅하는 과정

1. 판매원이 의뢰인과 약속을 잡는다.
2. 약속이 컨설팅 회사의 건물에서 잡히면, 회사 내의 전문가가 프리젠테이션을 위한 컨퍼런스 룸을 준비한다.
3. 약속이 컨설팅 회사 건물 외부에서 잡히면, 컨설턴트는 자신의 노트북 컴퓨터 에다가 프리젠테이션을 준비한다.
4. 컨설턴트와 판매원이 의뢰인과 만날 시간과 장소를 정한다.
5. 판매원은 의뢰인에게 보낼 서신을 발송한다.
6. 미팅 후 문제점이 정리되면, 컨설턴트는 제안서를 작성하여 의뢰인에게 발송한다.



5-2)구획면

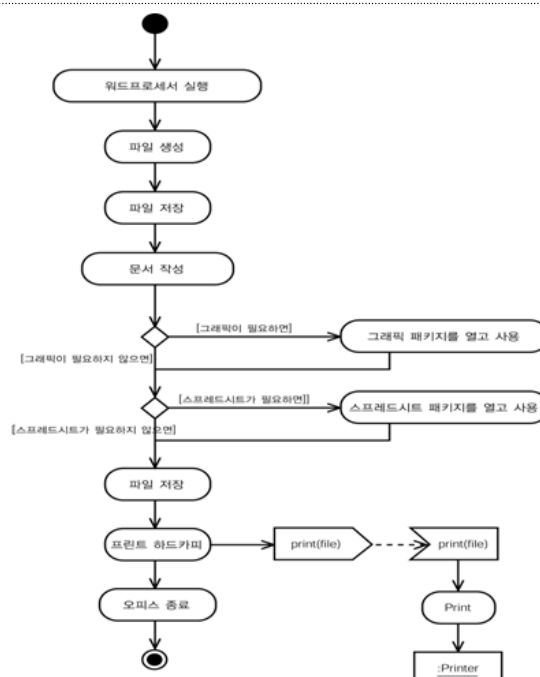
- 활동 다이어그램에는 역할(role)을 표시할 수 있다. 이렇게 하려면, 다이어그램을 **구획면(swimlane)**이라고 불리는 수직 구역으로 분할한다. 각각의 구획면에는 역할의 이름이 윗부분에 나오며, 각 역할에 대한 활동이 나타난다. 물론, 한 구획면에서 다른 구획면으로의 전이도 가능하다.



업무과정 활동 다이어그램에 구획면을 넣어 다시 그린 그림

(6)혼합 다이어그램

- 여러개의 다른 다이어그램을 필요에 따라 조합할 수 있음을 보이기 위하여 준비한 것으로, 이렇게 그린 다이어그램을 **혼합다이어그램(hybrid diagram)**이라고 한다.

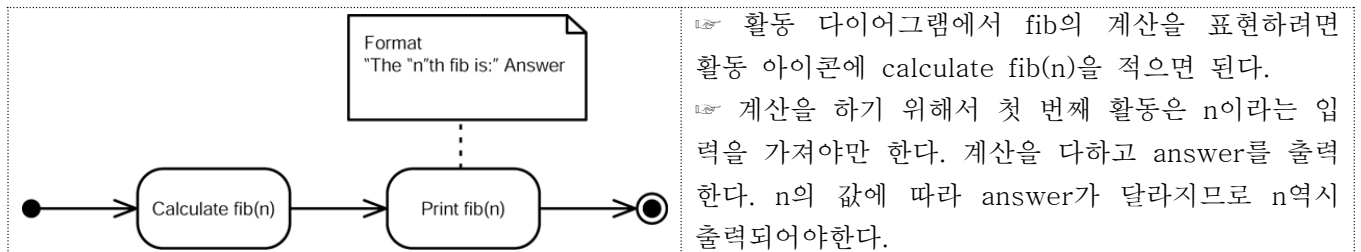


(7)UML 2.0에서 새로운 개념

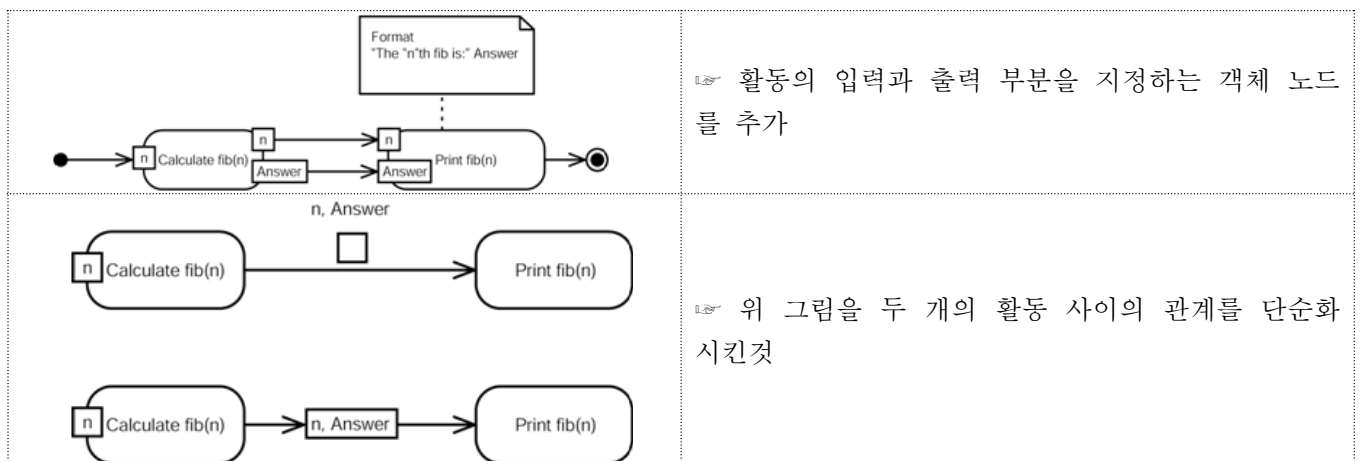
7-1)활동의 객체

- 객체노드(object node)를 이용해서 활동의 입력부분과 출력부분을 지정할 수 있다.

예제) 피보나치 수열이라고 불리는 1,1,2,3,5,8,13.... 의 수열을 본적이 있는가 피보나치 수열이며, 각 숫자의 "fib"라고 불린다. 즉, 첫째 fib는 fib(1)는 1이고 fib(2)는 1이며 fib(3)은 2다 각 숫자는 앞의 숫자 두 개를 더 하여 만들어지기 때문에 fib(8)은 21이된다.

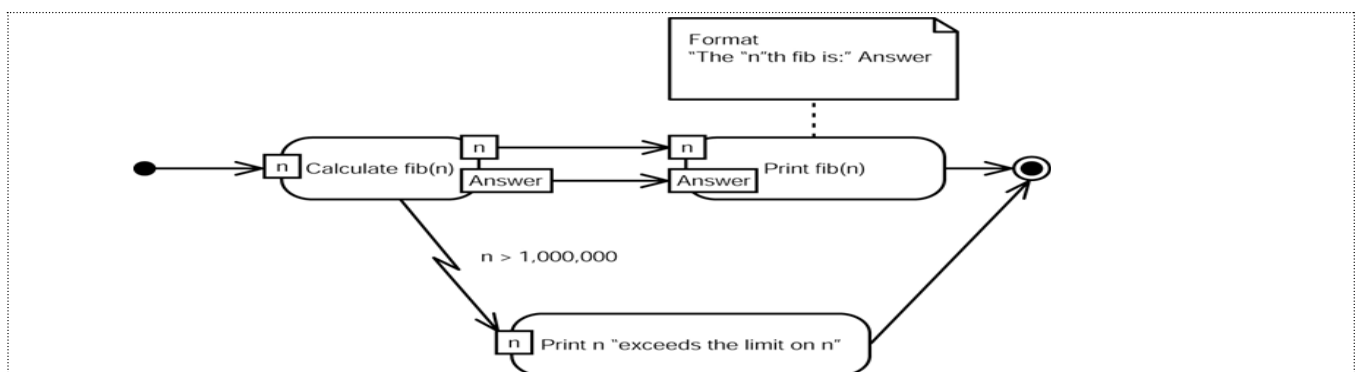


- 입력 부분을 보여주기 위해 작은 상자를 첫 번째 활동의 가장자리 왼쪽에 붙여주고 n이라고 써준다. 출력 부분은 보여주기 위해 작은 상자를 가장자리 오른쪽에 붙였다. 이 작은 상자들이 바로 객체 노드이다. 객체 노드는 두 번째 활동에서 입력 부분을 나타내기 위해서도 사용 되었다.



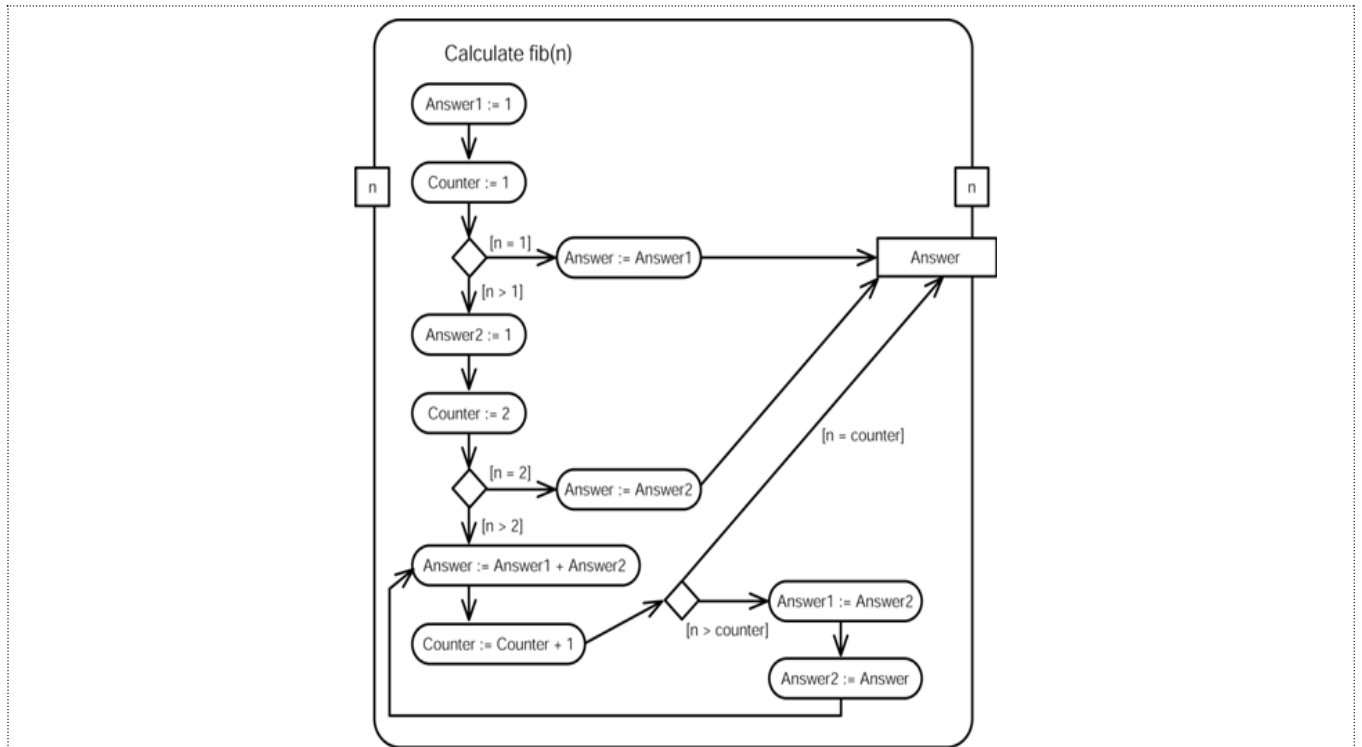
(8)예외 다루기

- 간혹 활동은 **예외상황**(정상시와 다르거나 능력 이상의 것을 시도하려는 상황)과 부딪치게 될 경우가 있다. 예를 들어, 피보나치 계산기가 100만이 넘어가면 계산이 불가능하다고 가정해보자, 땀만보다 큰 수를 n의 값으로 지정하면 “ 한계치를 넘었습니다.(exceed the limit on n)”를 출력한다. 이러한 내용을 활동 다이어그램으로 나타내기 위해서 번개와 닮은 화살표를 이용한다. 예외가 발생하는 활동에서 시작되어 예외가 발생하게 된 이유를 설명하는 활동으로 향하게 된다. 이러한 활동을 **예외 처리자(exception handler)**라고 부른다.

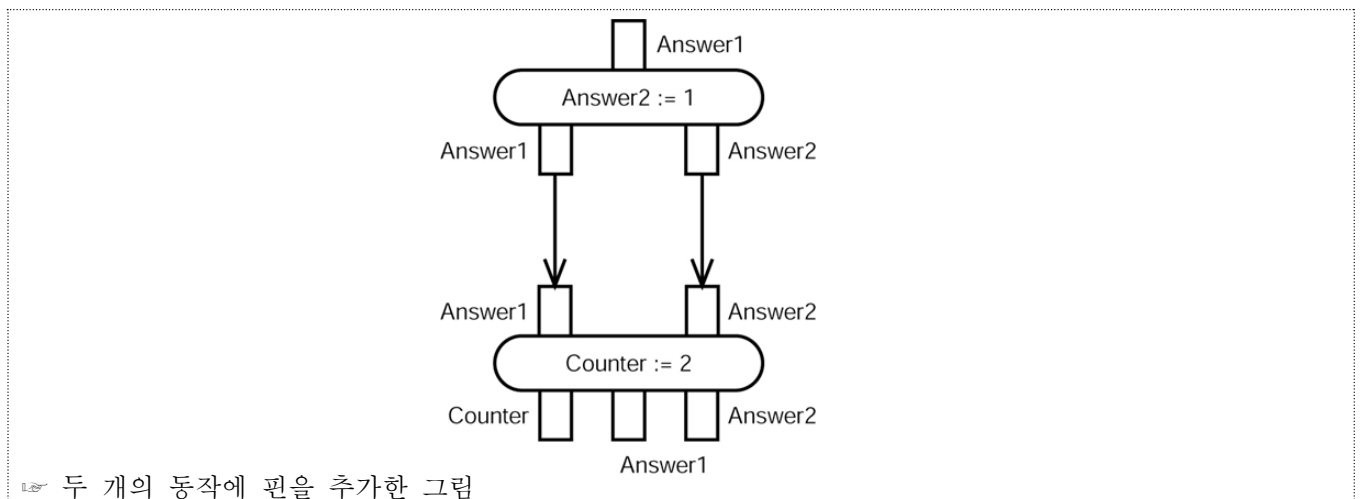


(9)활동 분해하기

- 하나의 활동은 많은 동작(action)으로 구성되어 있다. 동작의 아이콘은 활동의 아이콘과 같다. fib를 계산하는 모든 과정을 모델링 하기 위해서 몇 가지 변수가 필요하다. 먼저 오퍼레이션의 목적인 n번째까지의 계산을 판단하기 위해(counter)가 필요하고, 계산된 결과를 저장할 변수(answer)도 필요하다. 또한 두 숫자를 합쳐야 하므로 변수 두 개가 더 필요하다(answer1 answer2)

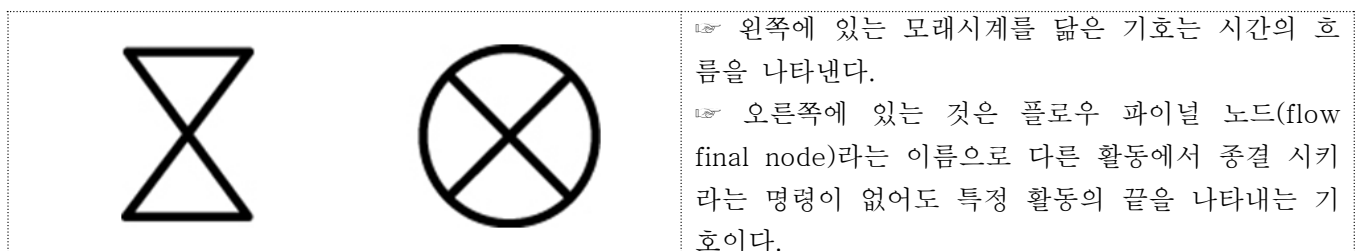


- 동작이 객체 노드를 갖는 것도 가능하며, 동작에 위치한 객체 노드를 **핀(pin)**이라고 부른다.



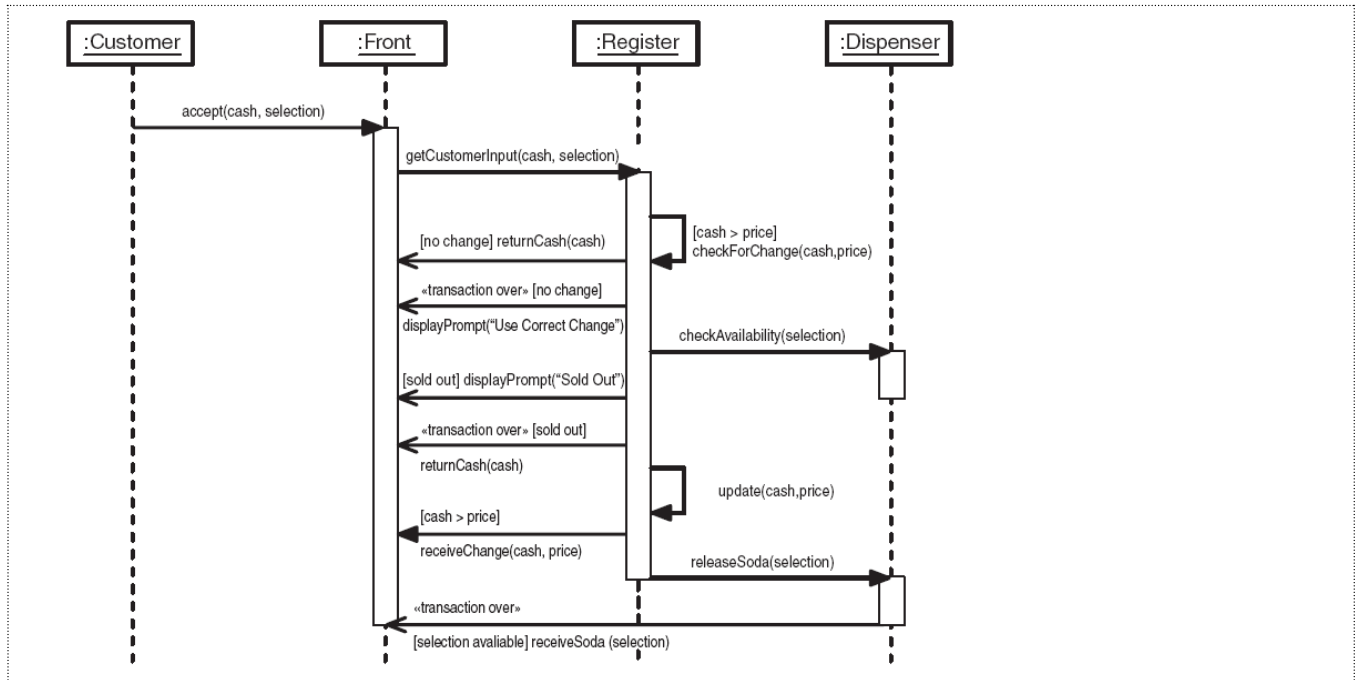
(10)시간 표시하기와 플로우 끝내기

- 활동 다이어그램을 좀더 매끄럽게만들어 주는 새로운 UML 기호 두 개를 더 보여주겠다.

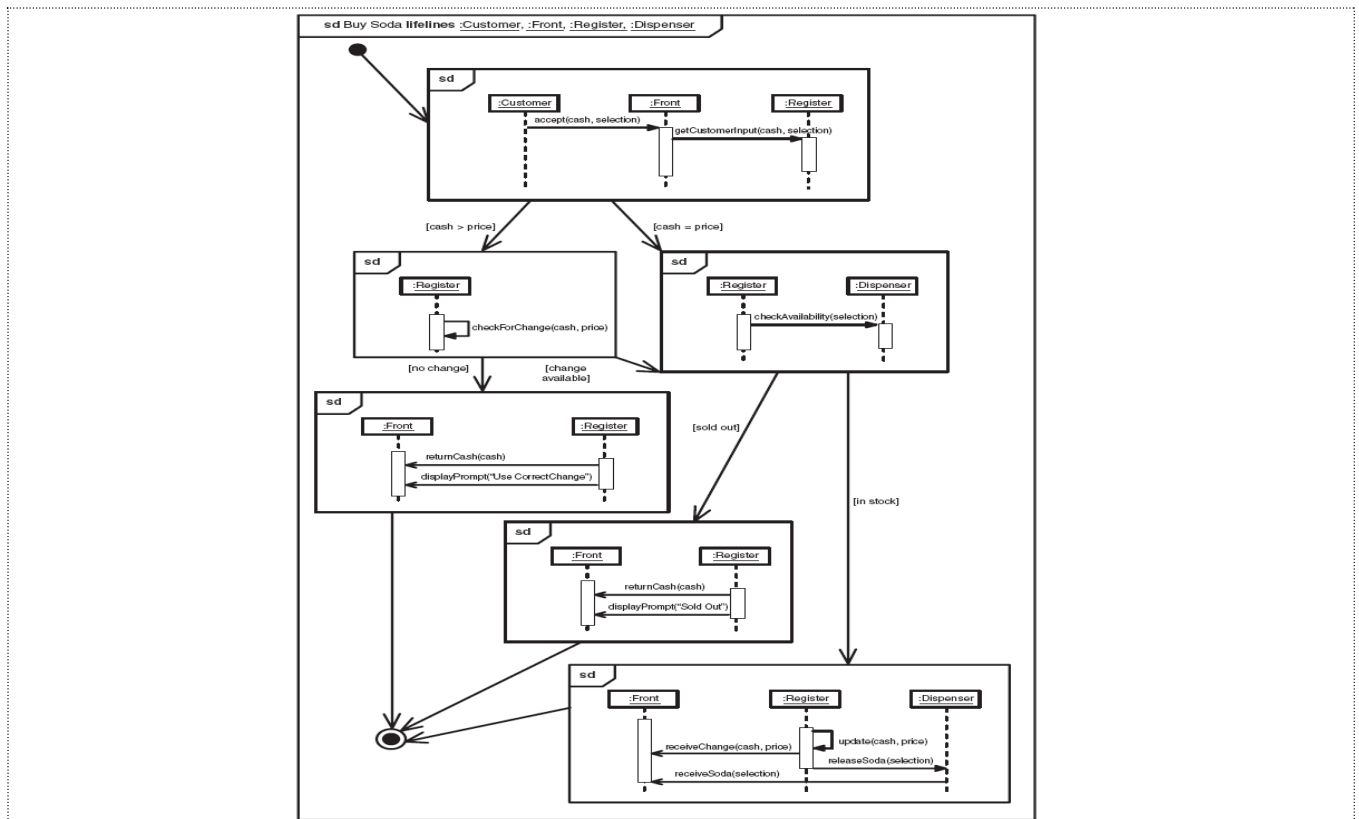


(11)교류 개요 다이어그램(interaction overview diagram)

- 교류 개요 다이어그램은 전체로 보면 하나의 활동 다이어그램인데, 각각의 활동 내부를 보면 교류 다이어그램으로 이루어져 있다.



- 이 다이어그램에서 집중해서 봐야할 곳은 다른 시퀀스 다이어그램에 연결되어 있는 각각의 시퀀스 다이어그램이다.



퀴즈

1. 결정위치 (decision point)를 나타내는 두 가지 방법은 무엇일까?
2. 구획면(swimlane)이란?
3. 신호 전송과 신호 수신은 어떻게 나타낼까?
4. 동작(action)은 무엇인가?

Part 12. 컴포넌트 다이어그램

(1)컴포넌트란?

- 소프트웨어 컴포넌트는 시스템을 이루는 물리적 요소이다. 컴포넌트는 컴퓨터 내에 있으며 다른 컴포넌트에 인터페이스를 제공한다.

컴포넌트와 컴포넌트 사이의 관계를 모델링하는 이유는 다음과 같다

- 의뢰인이 완성된 시스템의 구조와 완성된 시스템의 기능을 볼 수 있게 하기 위하여
- 개발자에게 작업할 구조를 구체적으로 알리기 위하여
- 문서와 도움말을 제공해야 하는 문서화 담당자들이 쉽게 이해할 수 있도록 하기 위하여
- 컴포넌트를 언제든지 **재사용**할 수 있게 하기 위하여

(2)컴포넌트와 인터페이스

- 객체는 자신의 행동을 어떻게 수행하는 지를 외부 세계와 다른 객체에게 보이지 않는다. (캡슐화 혹은 정보 은닉) 그대신, 객체는 다른 객체들이 자신의 오퍼레이션 수행을 요청할 수 있게 하기 위한 “얼굴(face)”을 제공해야 한다. 이 얼굴이 바로 객체의 인터페이스다.

2-1)인터페이스

- 인터페이스는 어떤 클래스가 수행하는 행동의 특성을 설정하는 오퍼레이션의 집합이다.
- 인터페이스는 속성을 가지고 있지 않고 오퍼레이션만 가지고 있는 클래스다.

2-2)컴포넌트와 인터페이스

※ 컴포넌트와 인터페이스에 대하여 기억해 둘 두가지

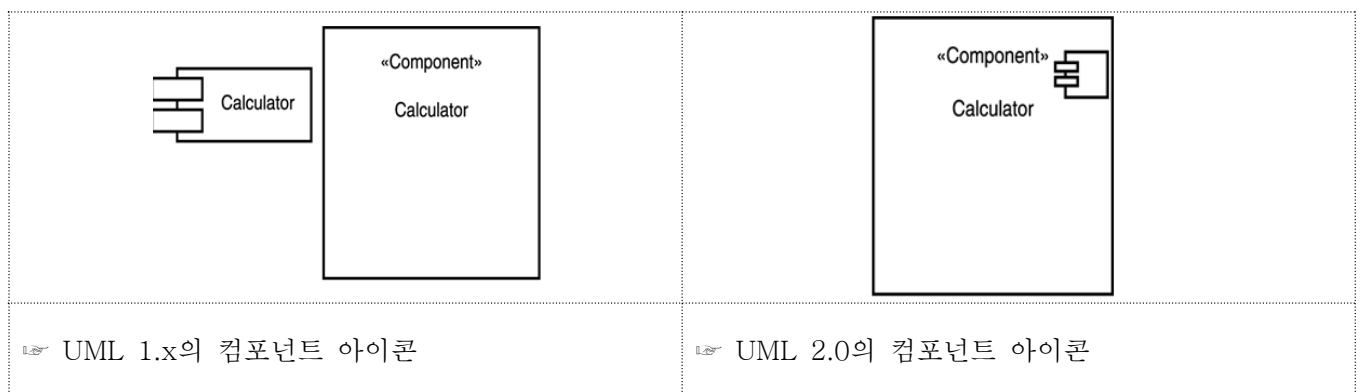
- 컴포넌트의 오퍼레이션은 그 컴포넌트의 인터페이스를 통해서만 사용할 수 있다.
 - 컴포넌트는 다른 컴포넌트들이 자신의 오퍼레이션을 호출하도록 자신의 인터페이스를 노출시킬 수 있다.
- ⇒ 서비스를 제공하는 컴포넌트 : 제공되는 인터페이스(provided interface)
- ⇒ 서비스를 액세스하는 컴포넌트 : 필수 인터페이스(required interface)

2-3)대체와 재사용

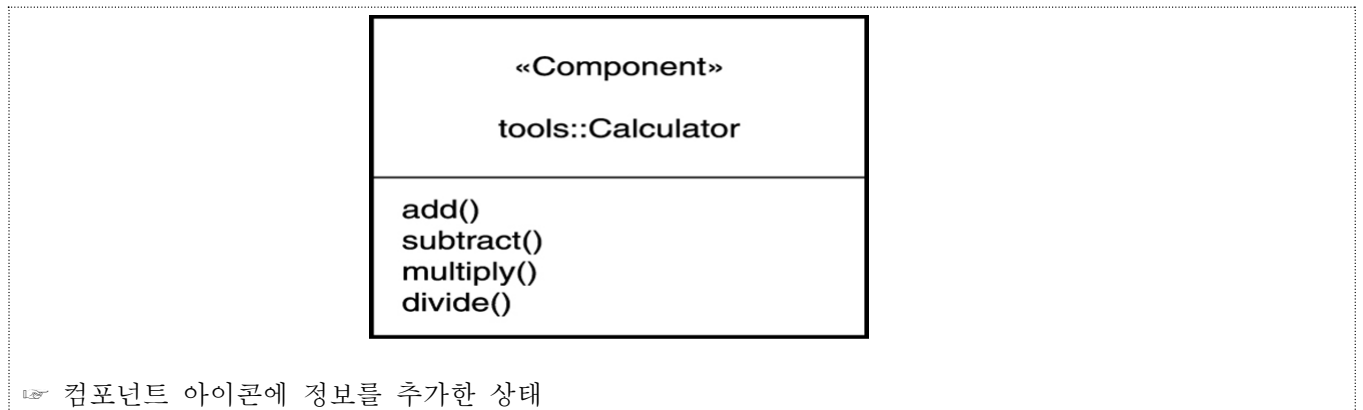
- 인터페이스는 컴포넌트 대체와 컴포넌트 재사용이라는 중요한 개념을 등장시킨다.
- 새로운 컴포넌트가 기존의 컴포넌트와 동일한 인터페이스를 가지고 있다면, 새 컴포넌트로 대체하여 재사용할 수 있다.

(3)컴포넌트 다이어그램이란?

- 컴포넌트 다이어그램은 컴포넌트, 인터페이스로 구성되어 있으며, 각각의 관계가 설정되어 있다.

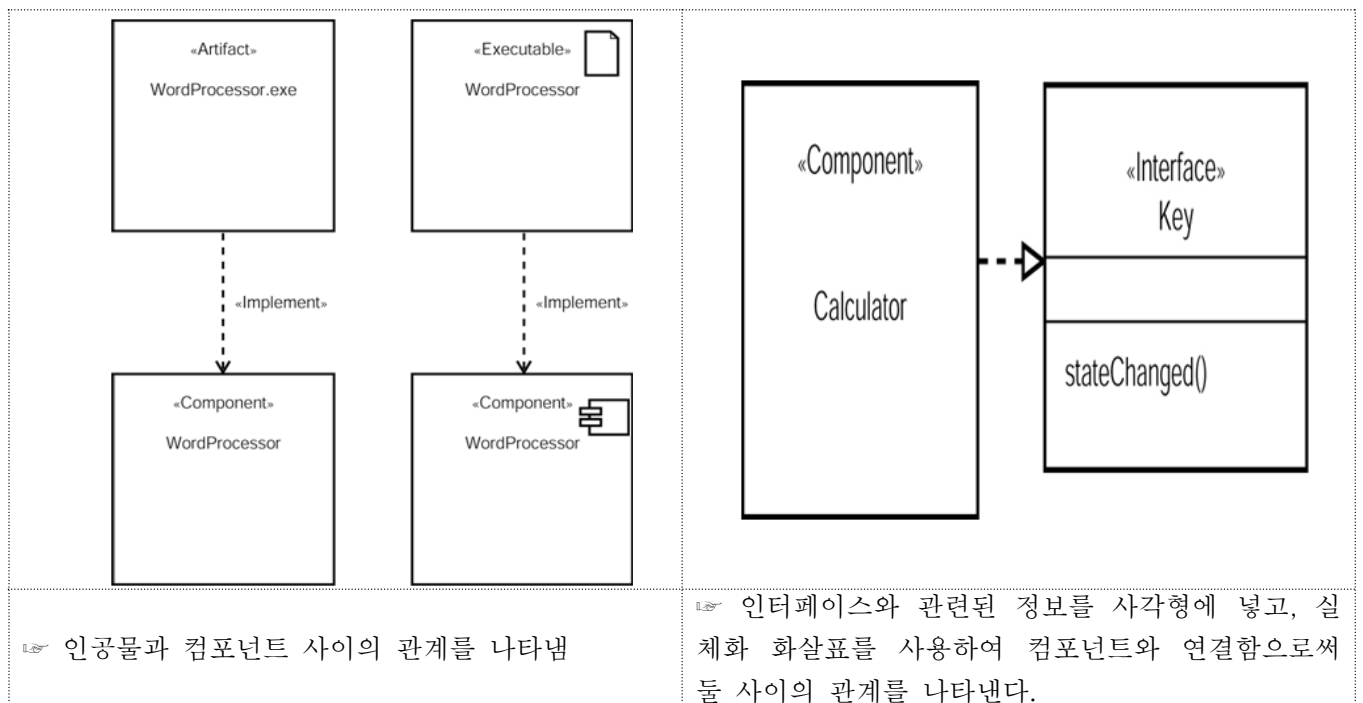


- 컴포넌트가 패키지에 포함되어 있다면 컴포넌트의 이름 앞에 패키지 이름을 붙일 수 있다

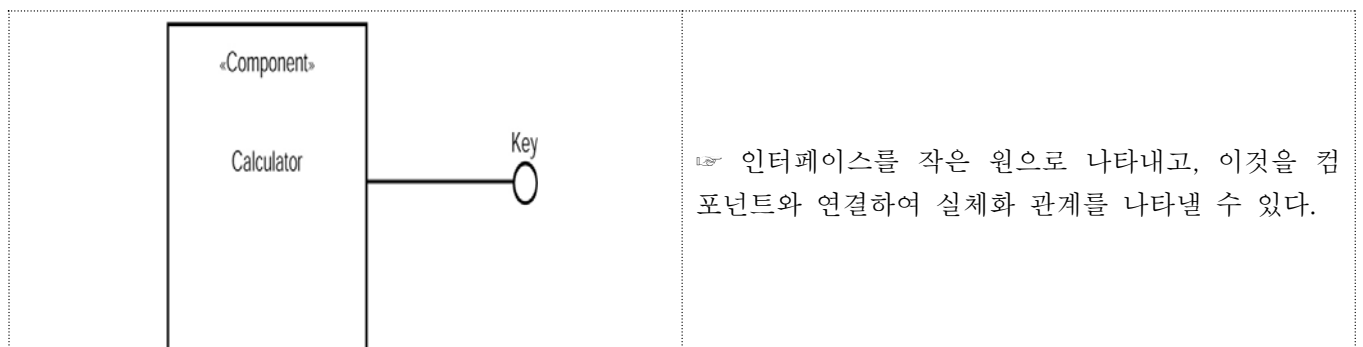


(3)인터페이스 나타내기

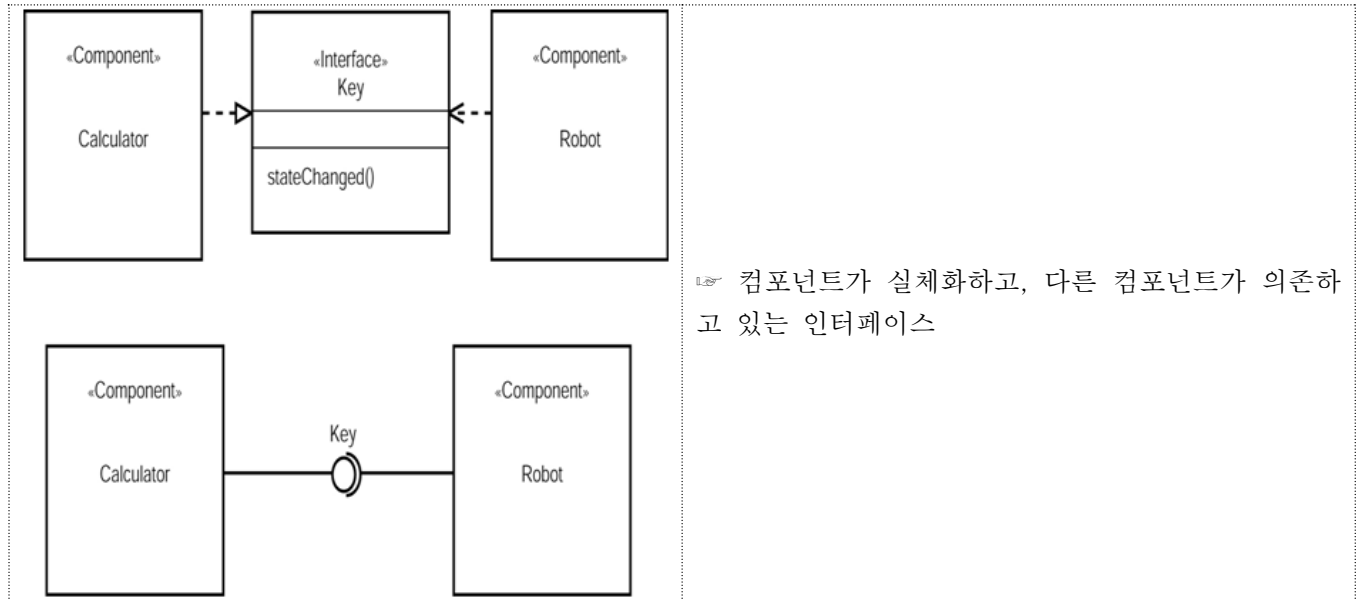
- 컴포넌트와 그 컴포넌트가 실체화한 인터페이스를 나타내는 방법은 두 가지이다. 첫째 방법은 인터페이스를 나타낼 때 그 인터페이스와 관련된 정보를 가진 사각형으로 그리는 것이다. 이사각형은 컴포넌트와 속이 빈 화살표 머리를 가진 채선으로 연결한다.



- 인터페이스를 나타내는 두 번째 방법으로써, 컴포넌트와 연결된 인터페이스가 작은 원으로 그려지고 있다.

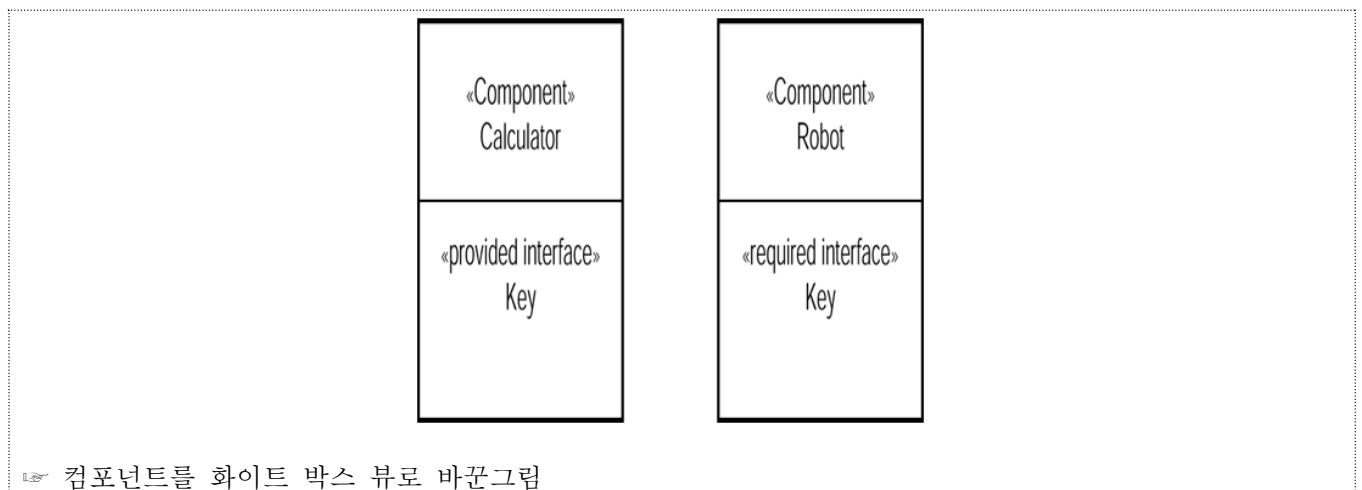


- 실체화 관계 뿐만 아니라 의존 관계도 나타낼 수 있다. 여기서 의존 관계는 컴포넌트와 필수 인터페이스 사이에 설정된다.

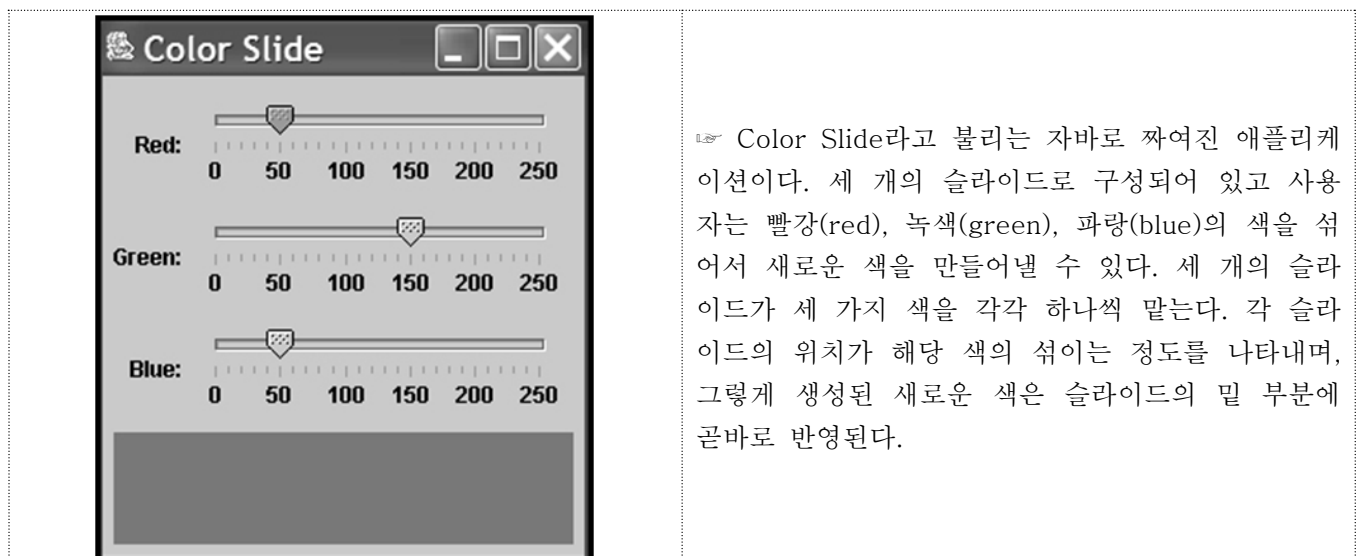


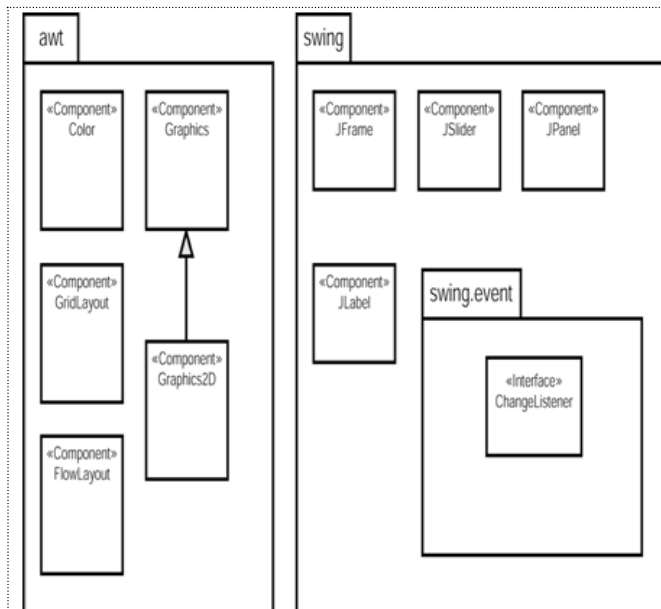
3-1)박스 - 블랙 앤 화이트

- 컴포넌트의 인터페이스를 모델링한 것을 UML에서는 외부의 뷰 또는 “블랙 박스”뷰라고 부른다.



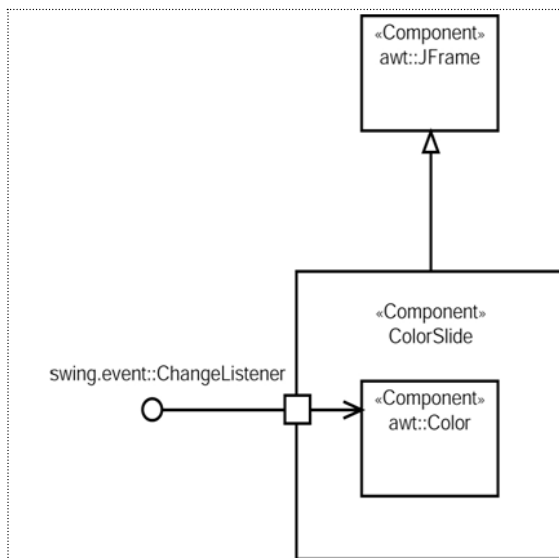
(4)컴포넌트 다이어그램 적용하기





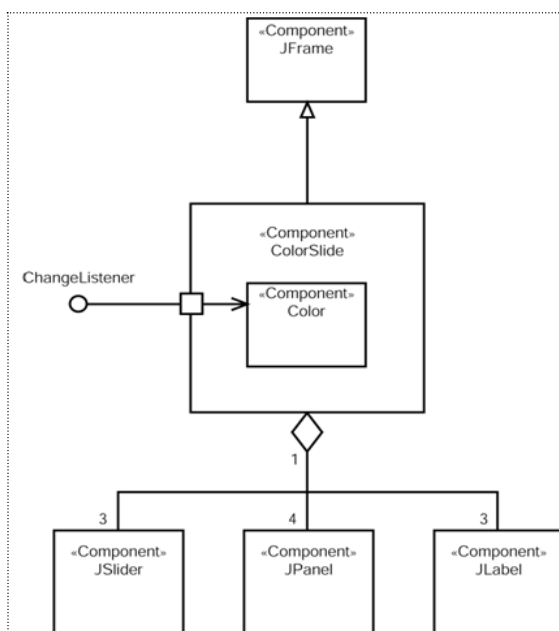
awt는 그래픽 사용자 인터페이스(GUI)를 컨트롤하고 보여주는 컴포넌트 그룹이다. 이 프로그램에서 가장 눈여겨보아야 할 컴포넌트는 Color, GridLayout, FlowLayout, Graphics, Graphics2D 컴포넌트이다.

다음 패키지 탭에 써 있는 swing(GUI를 추가할 수 있는 컴포넌트 그룹)이다. 패키지 내부의 각 컴포넌트가 무슨 역할을 맡았는지는 그 컴포넌트의 이름이 설명해 주고 있다. swing.event라고 레이블이 붙어 있는 패키지는 ChangeListener 인터페이스를 제공한다. 이 인터페이스는 GUI에 변화가 발생하기를 기다리는 역할을 한다. 즉 변화를 감지한다.



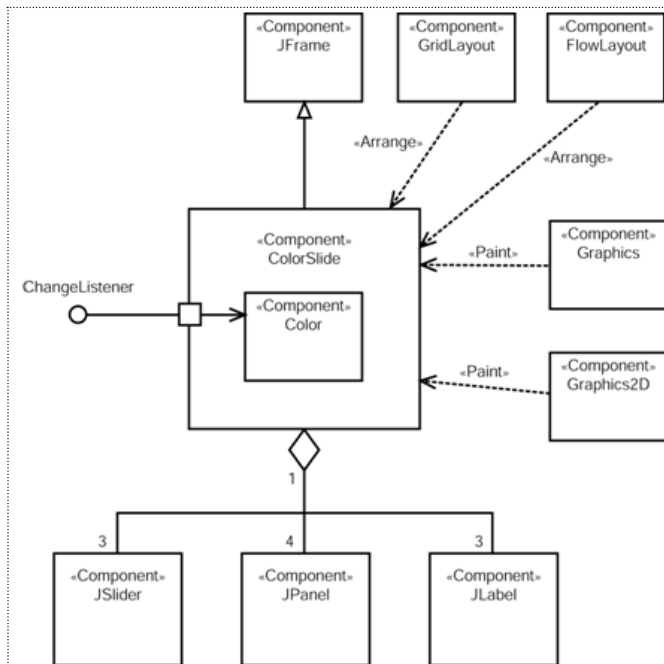
이번 예제의 컴포넌트의 분석 중에서 가장 초기 단계이다.

그림 내용을 잘 살펴보면 colorslide는 JFrame으로부터 상속되며, ChangeListener를 제공한다. ChangeListener는 ColorSlide를 이용하는 사용자를 위한 필수 인터페이스이다. ChangeListener와 ColorSlide 사이의 교류는 포트를 통하여 이루어지며, 그 교류를 통해 Color를 보내게 된다. UML 2.0에서는 볼과 소켓 연결을 **조립식 커넥터(assembly connector)**라고 하고, 화살표를 위임 커넥터(**delegation connector**)라고도 한다



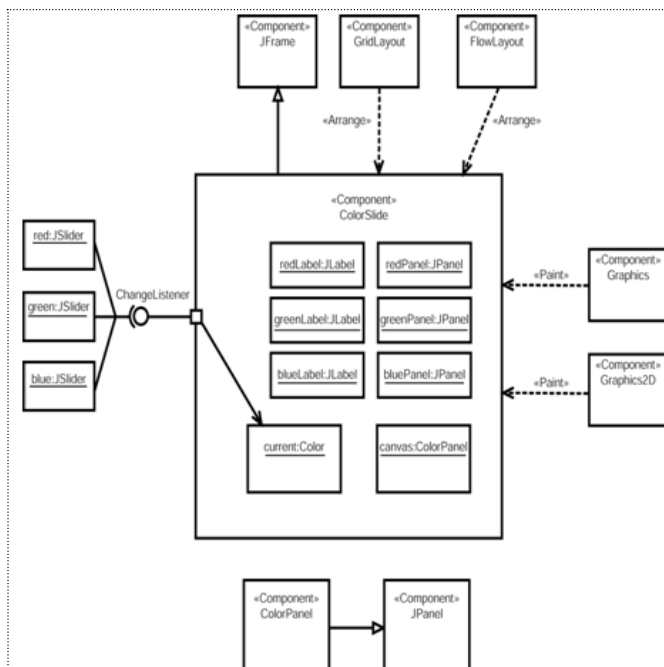
컴포넌트와 집합 연관에 있는 ColorSlide 애플리케이션

Colorslice는 JSlider, JPanel, JLabel 컴포넌트(다형성을 나타냄)와 집합연관에 있다. 프로그램이 빨강, 녹색, 파랑을 다루기 때문에 세 개의 슬라이드와 세 개의 레이블을 사용하고 있다. 또한, 각각의 슬라이드가 위치할 곳과 색이 섞인 결과를 보여주는 곳이 필요하기 때문에 4개의 패널이 필요하다.



GUI 컴포넌트를 배열하고 GUI를 표현하는 자바 컴포넌트의 추가

<<Arrange>> 키워드는 GridLayout과 FlowLayout이 패널, 슬라이드, 레이블을 배열하고 있음을 나타낸다. <<Paint>> 키워드는 Graphics와 Graphics2D의 연출을 의미한다.



슬라이드와 인터페이스 사이의 관계를 나타내고 있다.

ColorSlide 애플리케이션의 컴포넌트 - 객체모델링

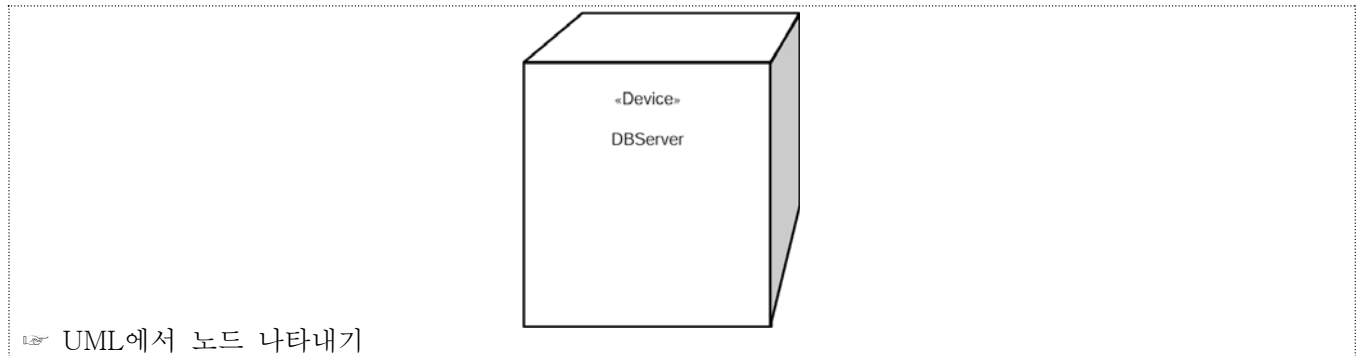
퀴즈

- 컴포넌트와 인공물 사이의 차이점은 무엇인가?
- 컴포넌트와 인터페이스 사이의 관계를 나타내는 방법 두 가지는?
- 제공되는 인터페이스란? 그리고 필수 인터페이스란?

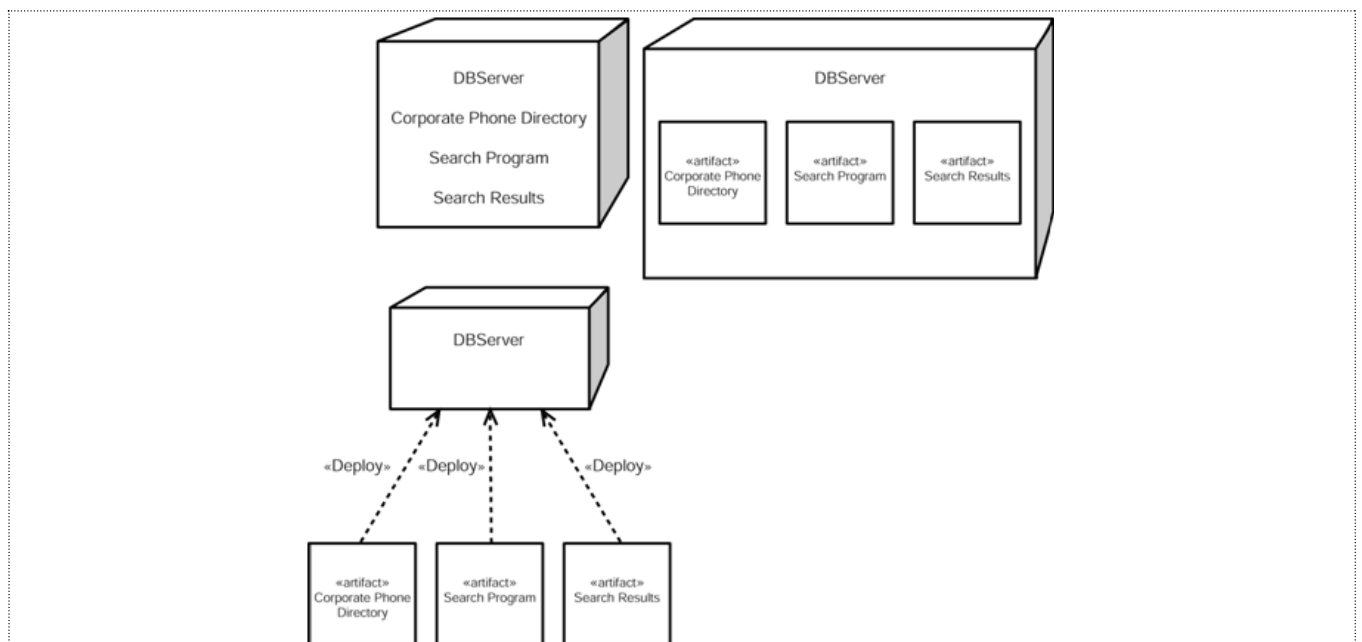
Part 13. 배포 다이어그램

(1) 배포 다이어그램이란?

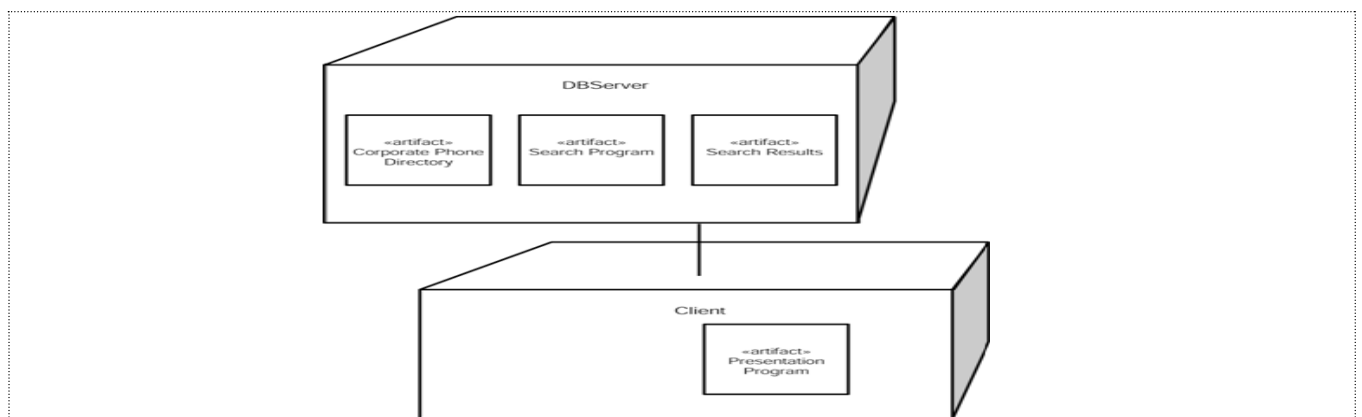
- 배포 다이어그램은 시스템 하드웨어 인공물이 어떻게 배포 되는지를 보여준다. 그리고 하드웨어가 다른 하드웨어에 어떻게 연결되는지도 보여준다. 주요 하드웨어 아이템은 **노드(node)**이다.
- UML 2.0에서는 인공물을 실행하는 노드를 **디바이스**라고 정의한다.



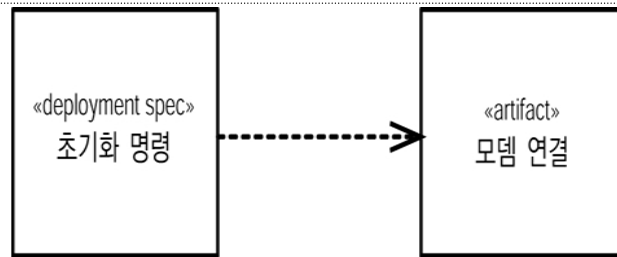
- 아래 그림은 노드로 배포된 인공물을 모델링하는 세 가지 방법을 보여준다.



- 두 개의 육면체 사이에 선을 그음으로써 두 노드를 접속(connection)시킬 수 있다. 노드 사이를 잇는 접속선은 꼭 통신선이나 케이블일 필요는 없다.



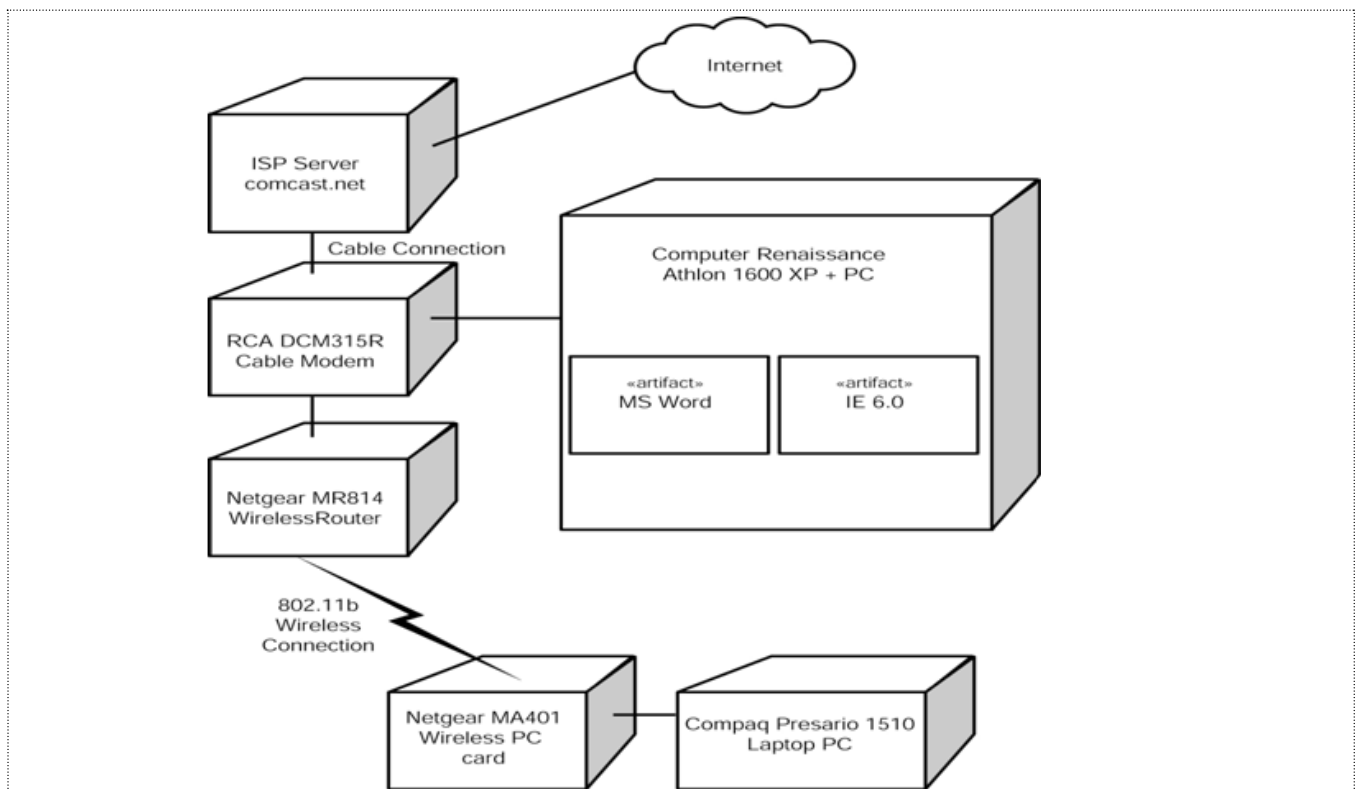
- UML 2.0이 인공물에 중점을 두었기 때문에 인공물에 관련된 개념들이 많이 생겨나게 되었다. 이러한 개념 중 하나가 배포 명세(deployment specification, 다른 인공물을 위해 매개 변수를 제공하는 인공물)이다. 모델 연결 때에 요청하는 초기 명령문이 좋은 예가 되겠는데, 이 명령문은 모델 특유의 특성을 위해 설정된 스트링 값이다.



☞ 배포 명세 및 배포 명세가 매개 변수를 보내는 인공물과의 관계를 나타낸 그림

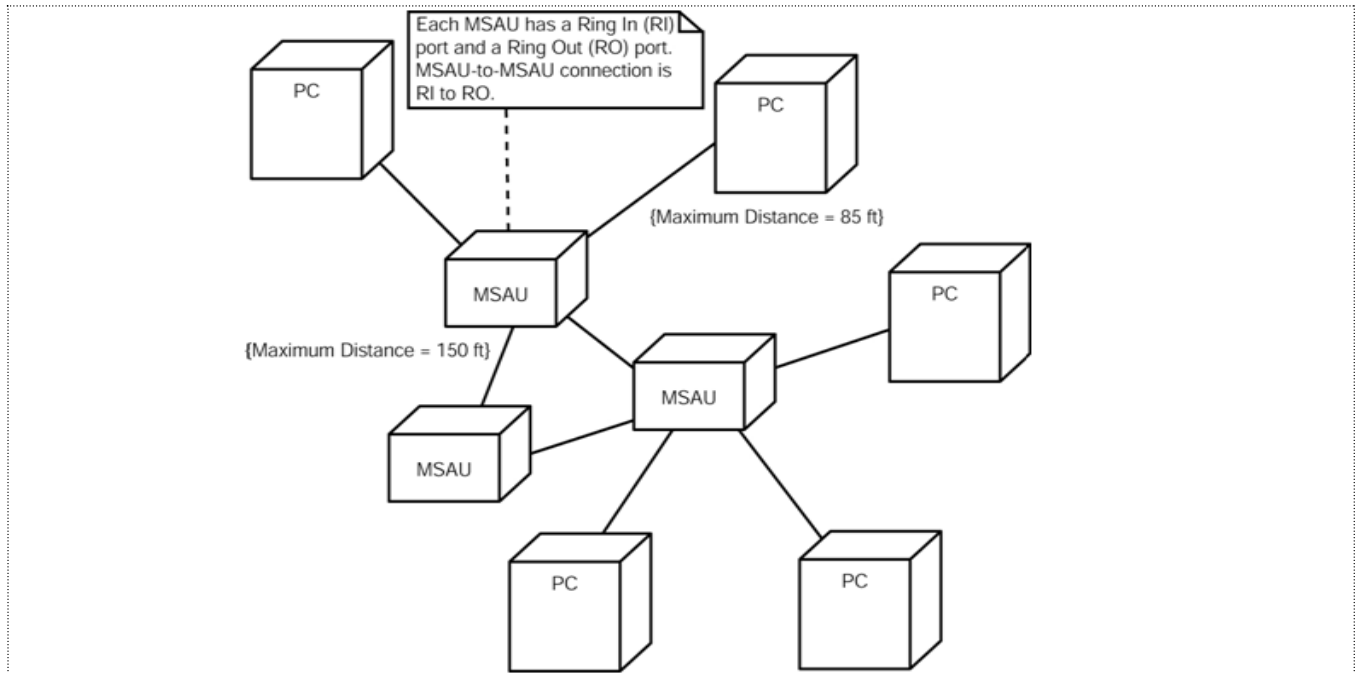
(2) 배포 다이어그램의 적용

2-1) 가정용 컴퓨터 시스템



2-2)토큰-링 네트워크

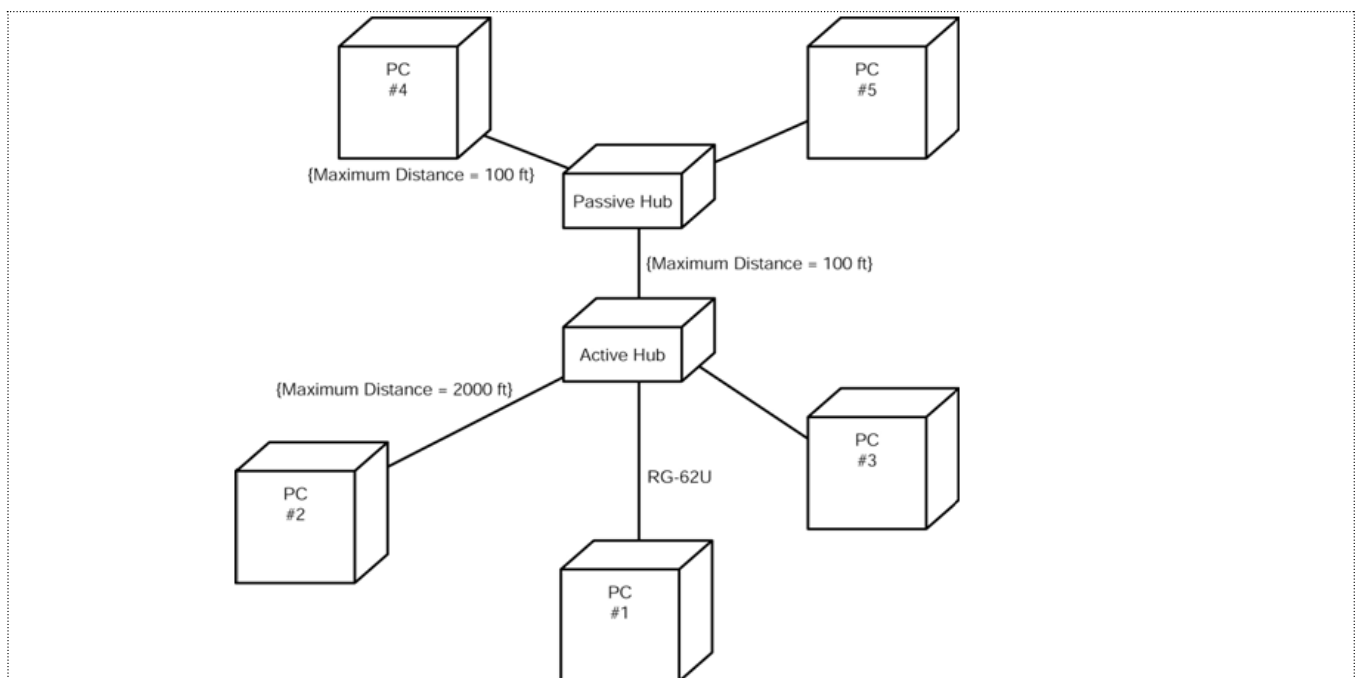
- 토큰-링(Token-ring) 네트워크에서 모든 컴퓨터는 중앙 다중 스테이션 액세스 유닛(MultiStation Access Unit : MSAU)으로 연결하는 네트워크 카드 인터페이스 카드(NICs)를 가지고 있다. 또한, MSAU끼리 연결되어 마치 고리(ring)같은 형태를 만든다. MSAU의 고리는 트래픽 관리자 역할을 하며, 토큰(token)이란 신호를 사용하여 각각의 컴퓨터로 하여금 자신이 정보를 언제 보낼지를 알 수 있도록 한다.



2-3)ARCnet

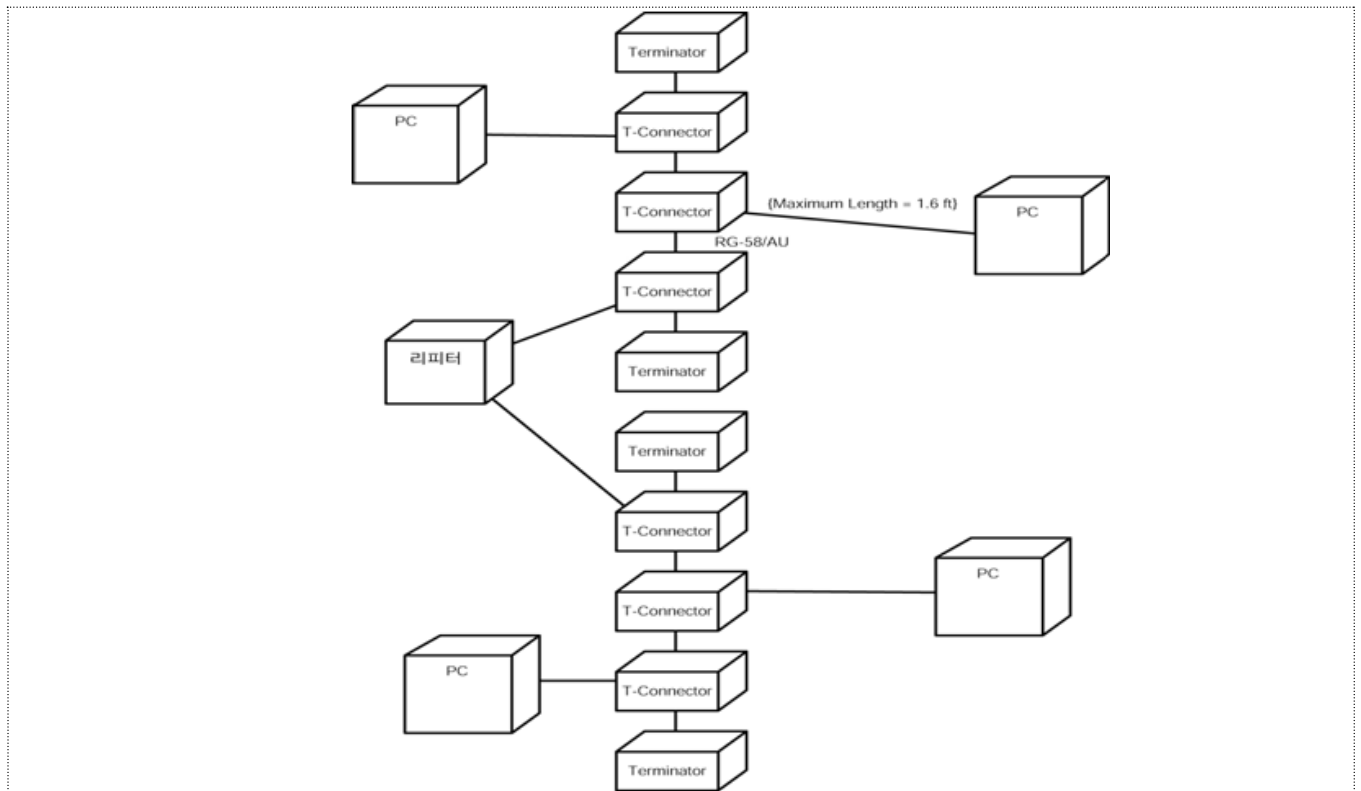
- ARCnet(Attached Resources Computing network)는 토큰링 네트워크와 같이 컴퓨터와 컴퓨터 사이에 토큰을 주고 받으록 하지만, 각각의 컴퓨터는 자신에게 할당된 번호를 가지고 있다는 점이 다르다. 이 번호의 순서에 따라 어느 컴퓨터가 토큰을 가질 지가 결정된다. 또한, 각각의 컴퓨터들은 액티브 허브(active hub) 혹은 패시브 허브(passive hub)에 연결되어 있다.

- 토큰-링 네트워크 내의 MSAU와 달리, ARCnet허브는 고리 사이로 토큰을 돌리지 않으며, 토큰을 주고 받는 것은 컴퓨터들이다.

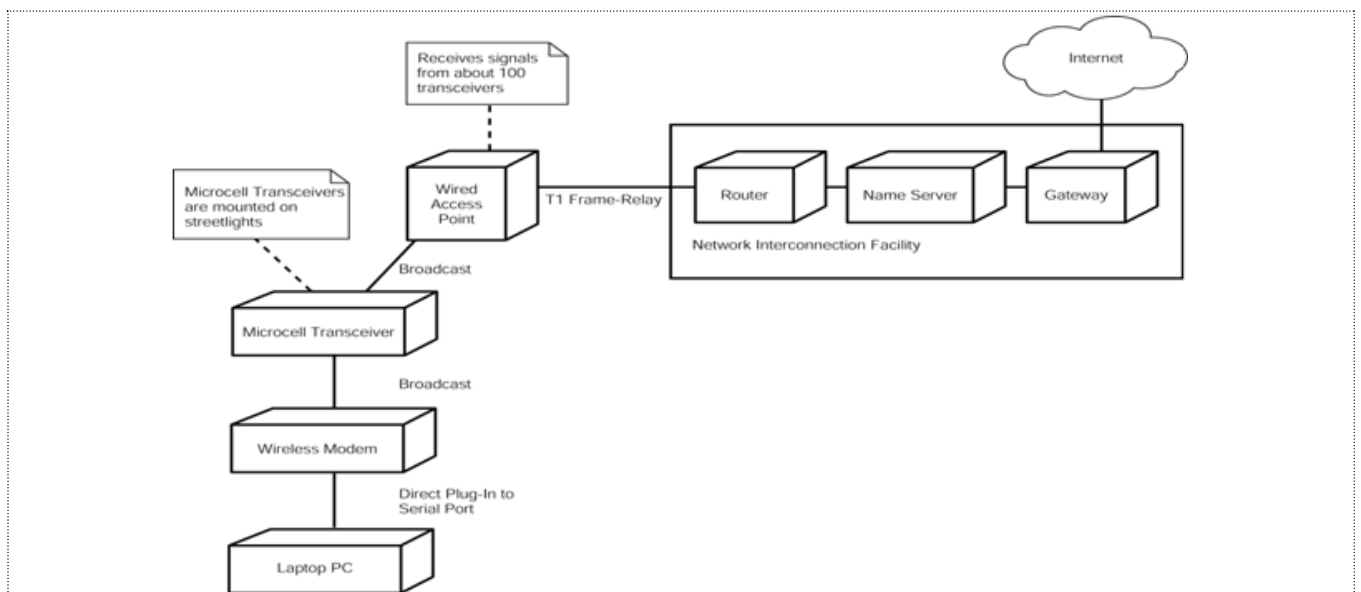


2-4)썸 이더넷

- 썸 이더넷(Thin ethernet)은 많이 사용하는 네트워크 형태이다. 컴퓨터는 T-커넥터로 불리는 접속 장치를 통해 네트워크 케이블에 접속한다. 한쪽의 네트워크 세그먼트(segment)는 리피터(repeater)라는 신호 증폭 장치를 통해 다른쪽 네트워크 구역과 연결될 수 있다.



2-5)리코켓 무선 네트워크



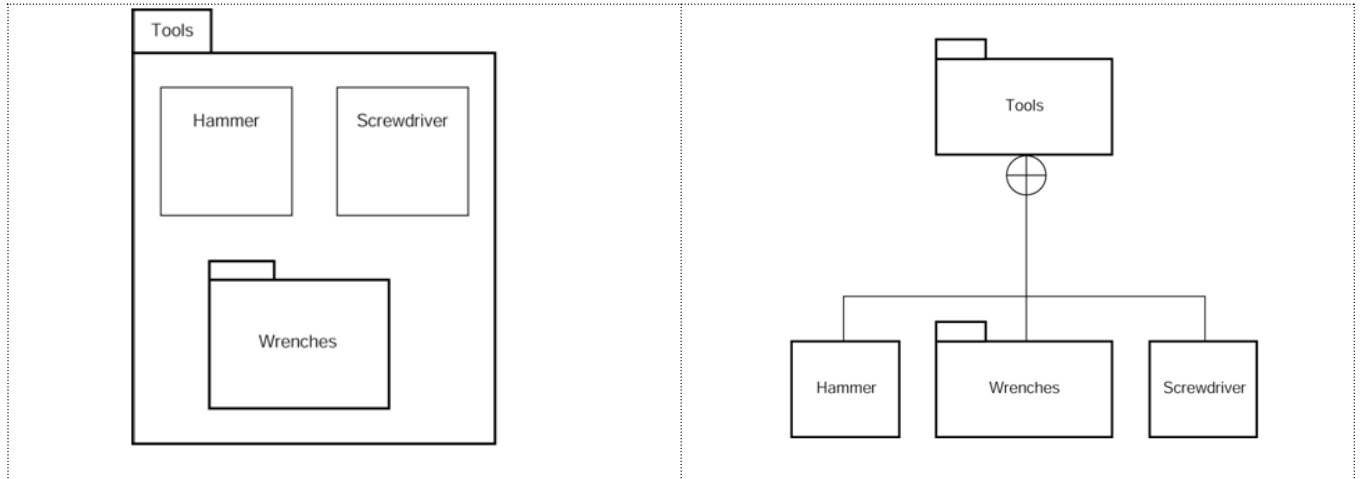
퀴즈

1. 배포 다이어그램에서 노드는 어떻게 나타낼까?
2. 노드에는 어떤 정보를 넣을 수 있을까?
3. 토큰-링 네트워크는 어떻게 구동될까?

Part 14. 패키지와의 UML의 구조

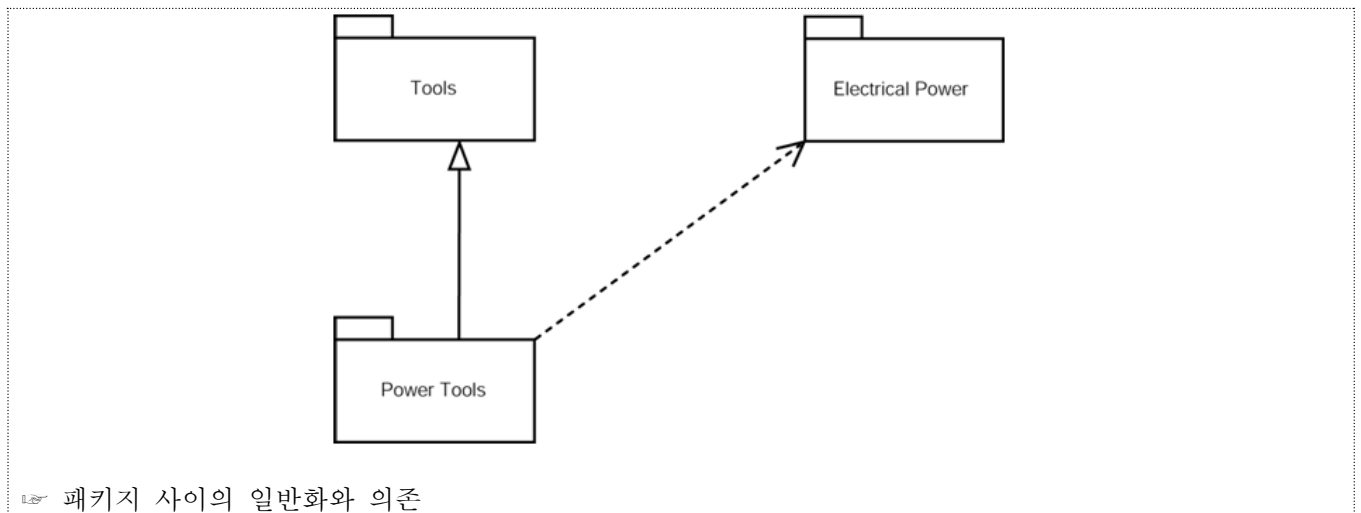
(1)패키지의 목적

- 패키지라는 이름은 다이어그램의 요소(클래스나 유스 p이스)들을 그룹화한다는 의미이다. 그룹화될 요소들을 탭 폴더 아이콘으로 둘러싸면 그 요소들은 패키지로 묶는 것이 된다. 패키지에 이름을 붙여주면 그룹에 이름을 붙여주는 것과 같다. UML에서는 그룹으로 묶인 요소들을 위해 패키지의 **네임스페이스(namespace)**를 제공한다.
- 패키지 내부 요소를 참조하기 위해서는 **패키지 이름 : 패키지요소** 표기법으로 사용한다. 이렇게 표기한 이름을 완전히 **한정된 이름(fully qualified name)**이라고 부른다.



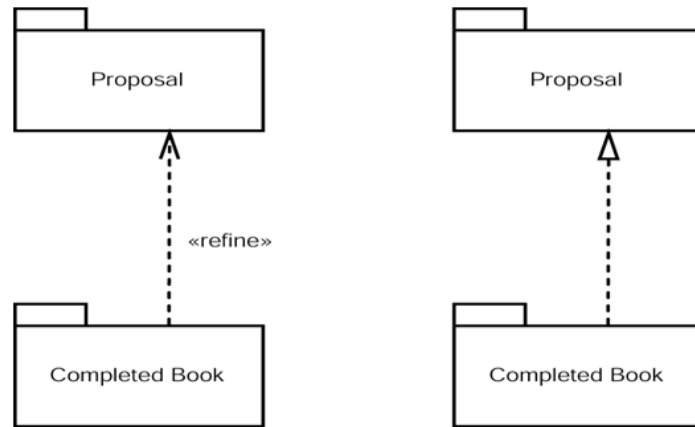
(2)패키지 사이의 관계

- 패키지가 다른 패키지와 관련을 맺는 방법에는 세 가지가 있다. 한패키지가 다른 패키지를 **일반화**할 수 있고, **의존**할 수 있고, **개량**(refine) 수 있다.



패키지 사이의 일반화와 의존

- 한 패키지의 구성요소가 다른 패키지의 구성요소와 같고, 좀 더 자세하게 이루어져 있다면 그 다른 패키지를 개량한다고 한다. 예를 들어, 여러분이 책을 쓴다면 보통 각 장(chapter)마다 간단하게 요약된 내용으로 시작을 할 것이다. 각 장의 요약 부분을 proposal이라는 패키지의 구성 요소라고 가정해보자. 또한, completed book이라는 패키지에 완성된 장이 구성요소로 들어 있다고 해보자. 이럴 때 completed book 패키지는 proposal 패키지의 개량이라고 할 수 있다.

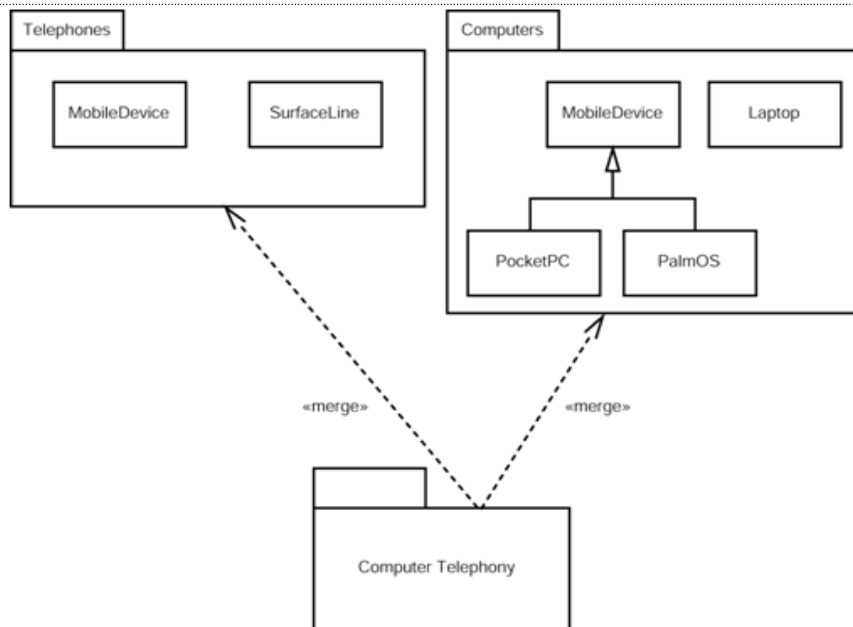


개량 관계를 나타내는 두 가지 방법

- 위의 그림 왼쪽에 있는 다이어그램은 의존의 한 종류인 개량을 나타낸다.(그러므로 점선 화살표를 사용하고 옆에 <<refine>>을 적어준다.)

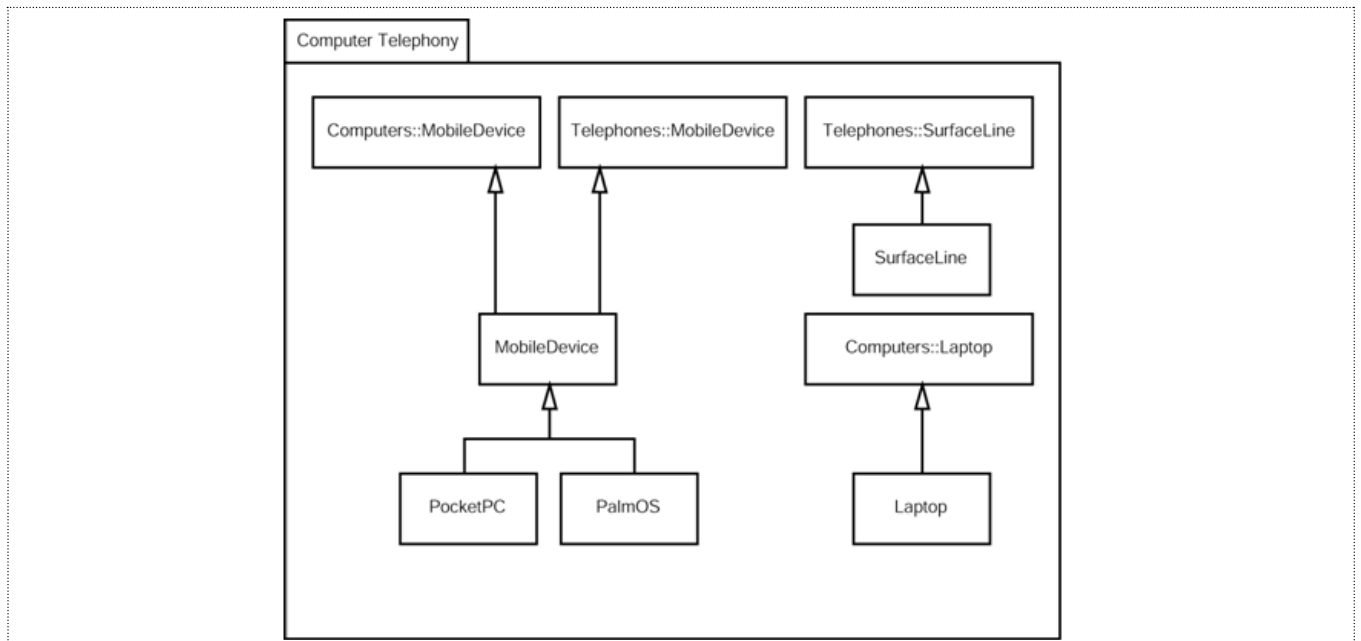
(3)패키지 합병하기

- 패키지는 다른 패키지와 합병할 수도 있다. 합병관계는 합병하는 패키지(source)와 합병되는 패키지(target) 사이에 이루어지는 의존 관계의 일종이다. 합병 결과로 원래의 패키지(source)가 변형된다.
- 예) 클래스로 구성된 Computer라고 불리는 패키지와 Telephones이라는 패키지가 있다. Computer Telephony는 이 둘을 합병한 패키지이다.



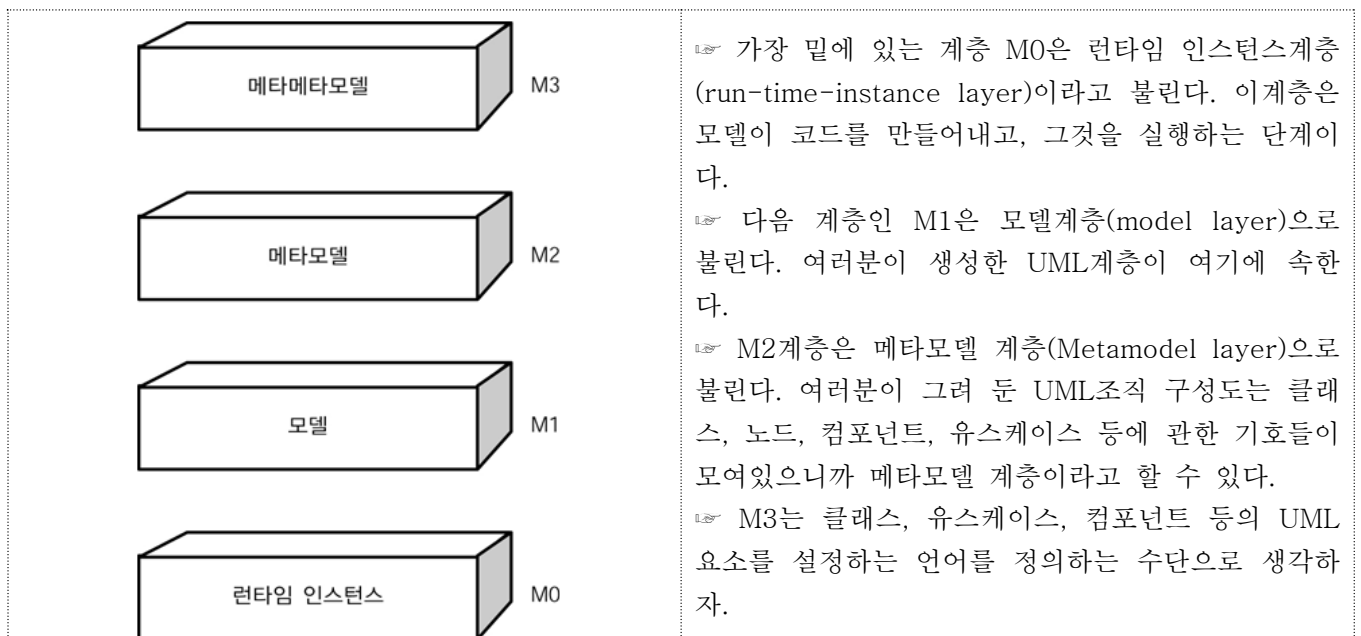
두 개의 패키지를 합병한 패키지 모델

- 합병은 Computer Telephony를 아래 그림처럼 변형 시킨다. 합병되는(target) 두 패키지의 모든 클래스가 임포트되어 있다. 완전히 한정된 이름에 의하면 Laptop과 surfaceline의 상속 관계가 합병되는(target)패키지 내에서 비롯됨을 알 수 있다.



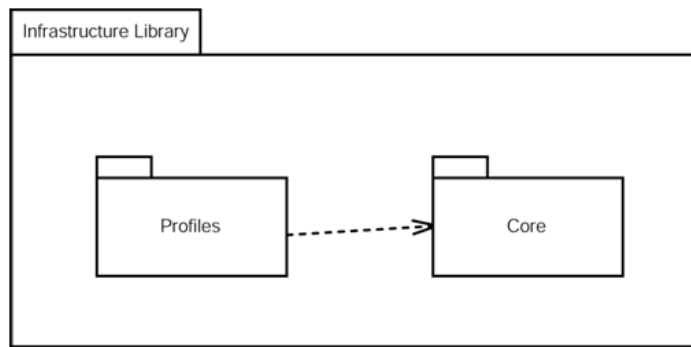
(4)UML의 근본 구조

- UML은 4계층 구조로 되어 있으며, 각 계층은 구성요소들이 가지고 있는 일반적인 성질에 따라 구분되었다.



(5)UML의 하부구조 패키지에 넣기

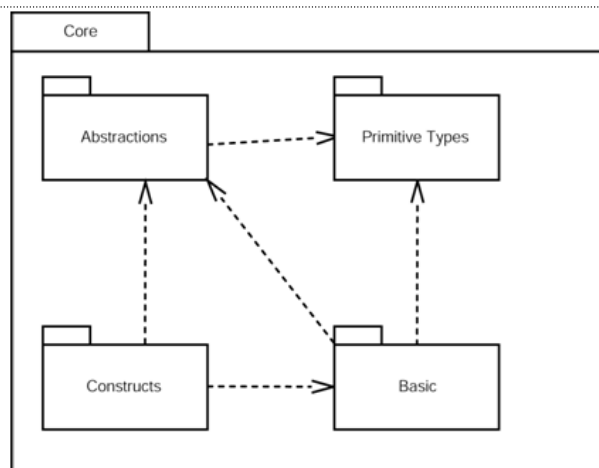
- UML의 근본 구조를 모델링 하기 위해 패키지를 사용
- 패키지 안에는 MOF로 쓰여진 클래스 다이어그램이 있다.



☞ Infranstructure Library는 Core와 Profile패키지를 소유한다.

(6)Core

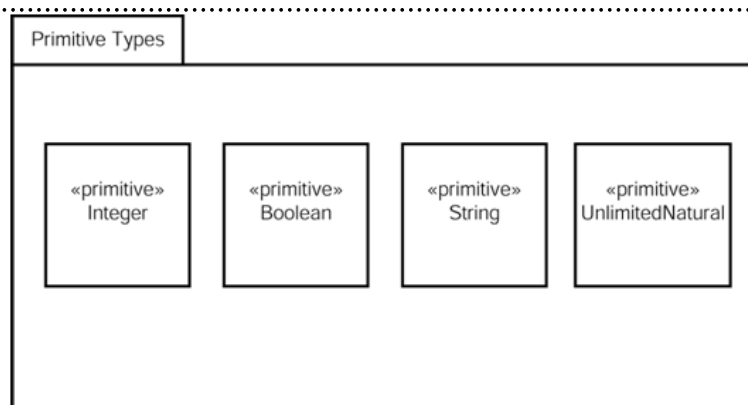
- Core는 아래 그림에서 보는거와 같이, 4개의 패키지를 소유한다.



☞ Core 패키지의 목록

6-1)기본타입

- 기본타입(Primitive Types)은 모델링 언어를 구축할 때 사용하는 데이터 타입이다. 이 패키지에서 타입에는 Integer, Boolean, String, UnlimitedNatural이 있다. 마지막 타입은 무한한 자연수를 의미하며, 무한을 의미하는 에스터리스크(*)로 표현한다.



☞ Infrastructure의 Core 패키지에 있는 기본 타입 패키지

6-2)추상

- Abstractions 패키지는 20개의 패키지를 가지고 있다. 1part ~ 13part까지 배웠던 개념들의 표현 설정 방법을 나타낸다. 이 패키지들중 Elements 패키지가 가장 기본적인 패키지이며, 하나의 추상 클래스를 가지고 있다.

6-3)기초

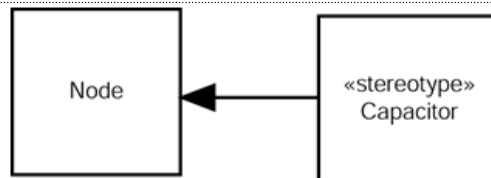
- Basic 패키지는 모델링을 하기 위한 걸음마의 일종이다. 클래스에 기초하여, 복잡한 모델링 언어를 개발하기 위한 초석이다. 만약 클래스(속성과 다른 클래스로부터 상속받은 능력), 매개변수(클래스의 오퍼레이션을 위해), 패키지, 특정 데이터 타입으로만 UML을 상상할 수 있다.

6-4)컨스트럭트

- Constructs 패키지는 수많은 Abstractions 패키지와 Basic 패키지에 의존한다. 이 두 패키지들로부터 요소들(클래스, 관계, 데이터 타입 같은)을 좀더 자세히 하기 위해 아이템을 결합한다. 예를 들어, 클래스의 속성과 고퍼레이션을 어떻게 보이게 할지 설명서를 포함하는 것과 같다. 이 패키지에서 클래스 사이의 관계(다중성 같은)에 추가할 수 있는 정보들을 찾을 수 있을 것이다.

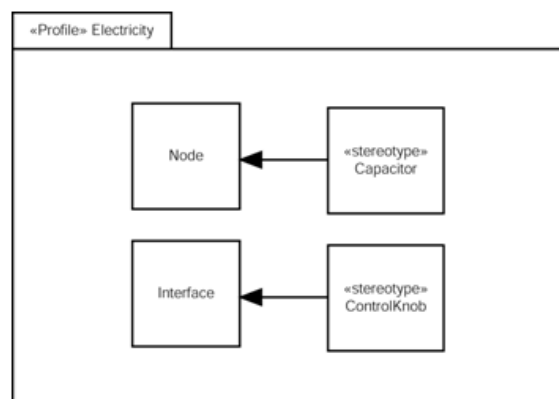
(7)프로파일(Profile)

- 이 패키지는 메타모델을 지식 영역의 특정 부분에 적응할 수 있도록 매커니즘을 제공한다. 하나의 적응이 하나의 분리된 Profile이다.
- Profile이 새로운 메타 모델을 구성하는게 아닌 Core패키지를 이용한 것이 된다.
- Profile은 이미 존재하는 메타모델을 약간 비틀고 꼬고 하는 것이라고 생각하면 된다. 마치 UML을 법이나 교육계의 모델에 적응시키는 것과 같다.
- **Extension** 과 **Stereotype** 이라는 메타 메타클래스를 소유한다는 뜻이다
- 스테레오타입을 생성하기 위한 형식적인 메커니즘을 지정



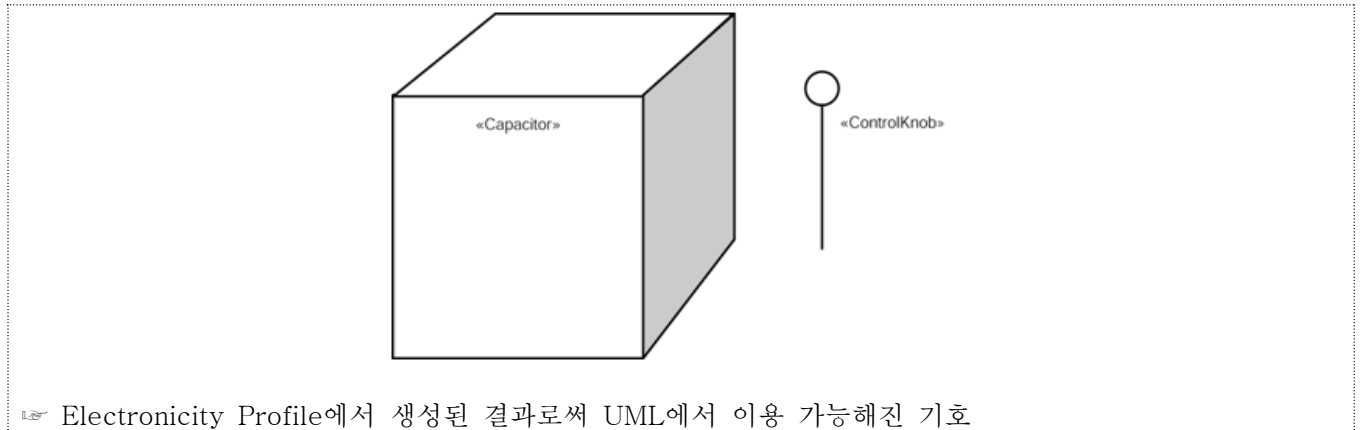
☞ Capacitor 스테레오 타입의 생성

☞ 짝 찬 삼각형 머리로 된 화살표는 메타클래스와 스테레오타입 사이의 확장 관계를 나타낸다.

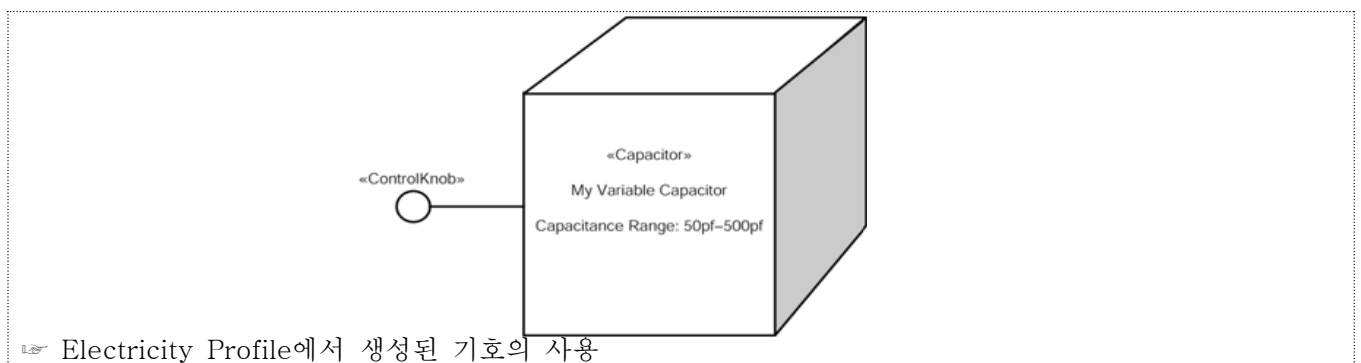


☞ 전기와 관련된 모델링을 위해 UML에 적응하는 Profile의 시작 단계

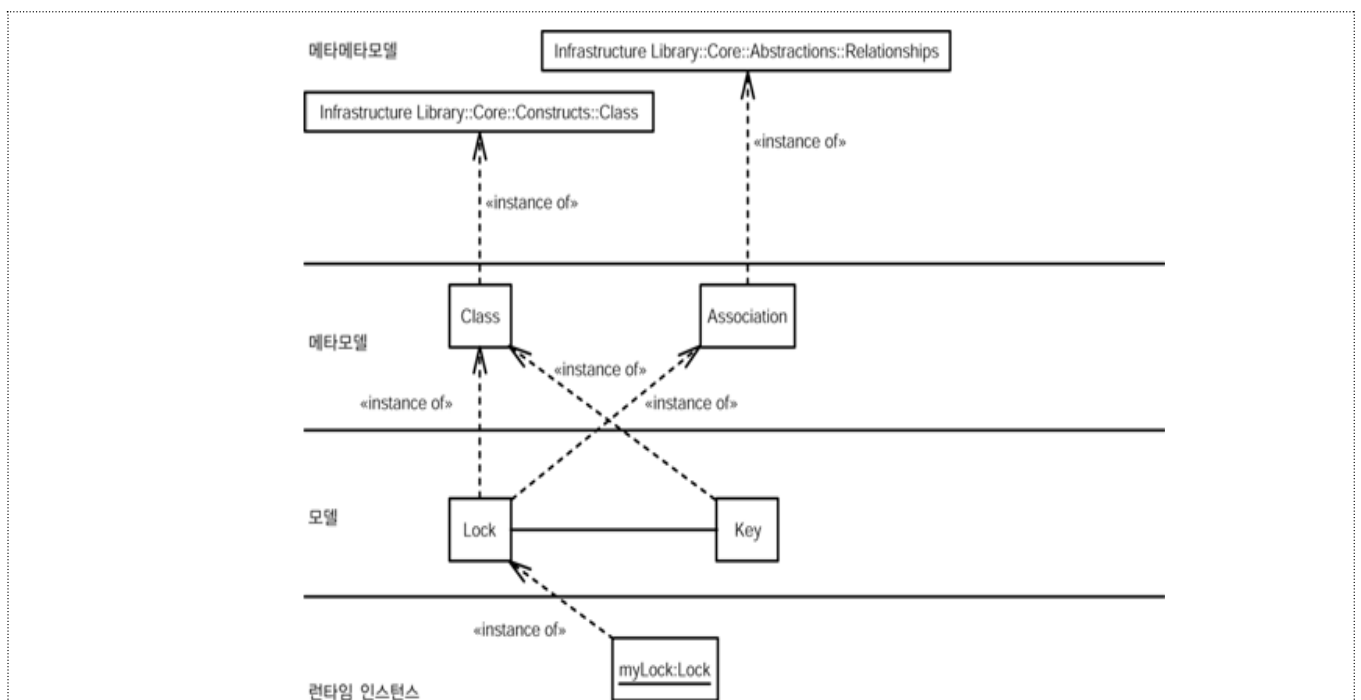
- 위 스테레오타입들이 만들어졌고, UML Electricity Profile에서(즉, 확장된 메타모델에서) 이용할 수 있는 기호가 되었다 .



- 좀더 실제적인 부분을 살펴보자. 모델에 이 기호를 사용했을 때의 모양은 다음과 같다.

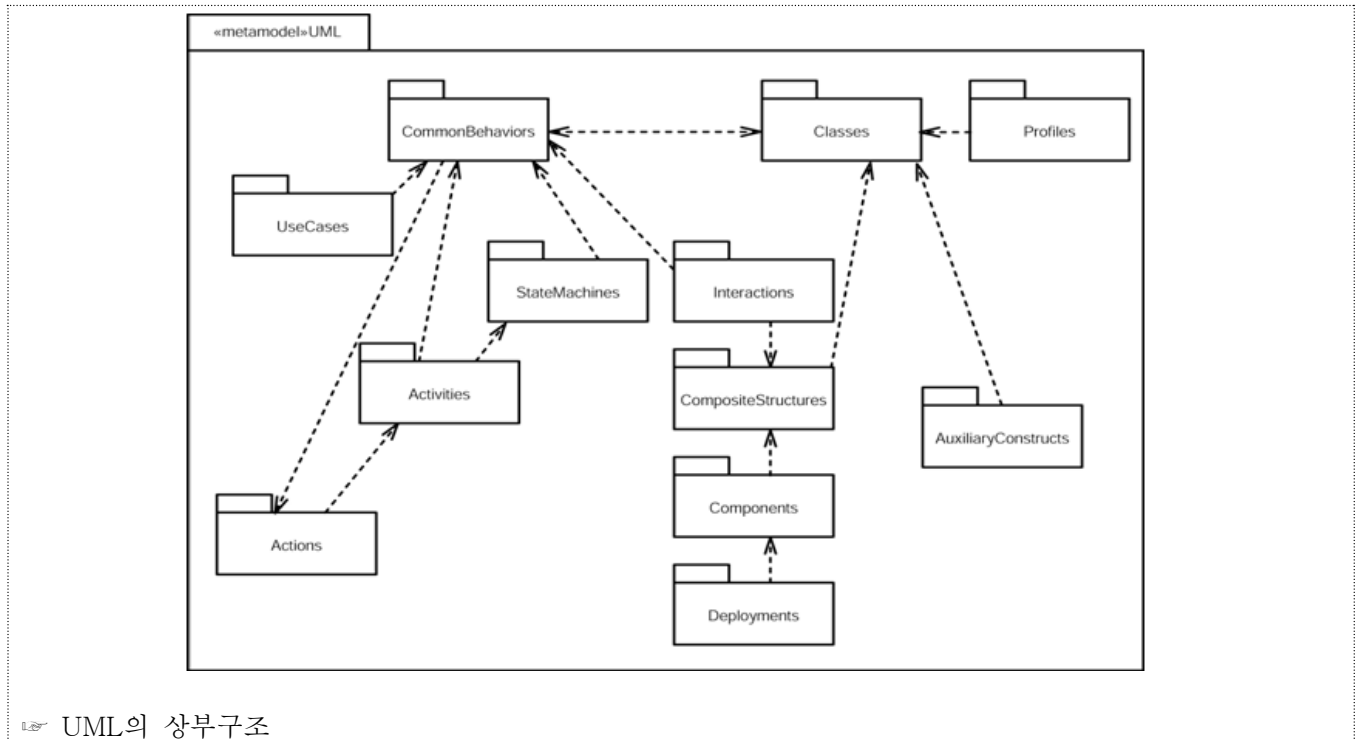


- 모델에서 클래스를 만들 때는 UML 클래스의 인스턴스를 생성하는 것이다. 다시 UML클래스는 메타메타모델에 있는 메타메타클래스의 인스턴스가 된다. 다른 부분을 보자. 런 타임 인스턴스는 여러분의 모델을 기본으로 한 코드의 결과물이다. 지금까지 공부한 내용을 4계층에 대해서 모든 것을 요약한 그림이다. 또한 메타 메타모델에 있는 몇 가지를 보여주고 있다.



(8)UML의 상부구조 패키지에 넣기

- 패키지 다이어그램이 UML의 근본 구조를 모델링하는 것과 마찬가지로, 패키지 다이어그램은 UML내부의 요소들(OMG가 UML의 상부구조라고 간주하는 것)을 모델링 한다.



- 위 그림을 보면 의존 관계의 화살표에서 두 가지 이상한 점(양방향 의존관계와 순환하는 의존관계)을 찾을 수 있을 것이다. 이 다이어그램은 두 패키지 사이의 관계를 의존 관계로 표현한 것이다. 즉 패키지의 요소중 하나라도 다른 패키지의 요소에 의존한다면 패키지도 의존 관계에 있다고 표현하였다.

(9)UML 확장 - 스테레오 타입

- "« »"로 나타나며 기존의 UML 요소를 확장하여 새로운 것을 만들기 위해 고안

■ 의존 관계

- 이미 만들어져 있는 스테레오타입의 대다수를 차지
- 클라이언트와 공급자 사이의 연관
- «import» «send» «instantiate»

■ 클래스

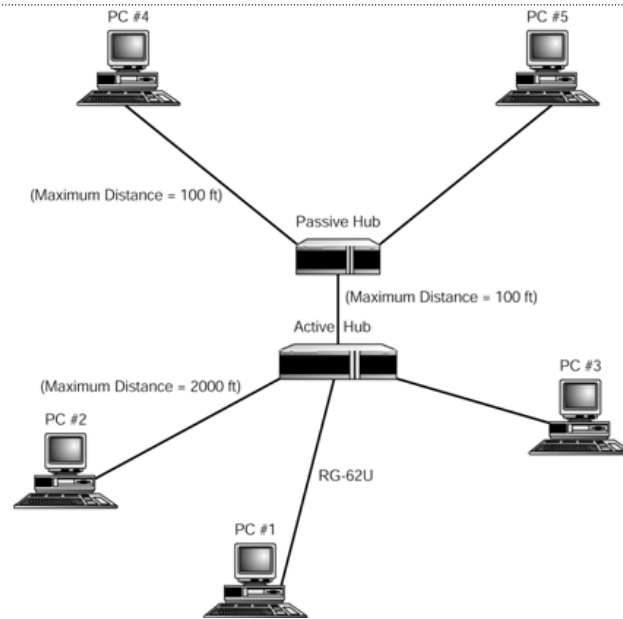
- «metaclass» «type» «utility» «create» «destroy»

■ 패키지

- «modelLibrary»
- «framework»

9-1)UML 확장 - 그래픽 스테레오 타입

- 의뢰인에게 확실한 의미를 전달하기 위하여 새로운 기호를 사용해야 할 경우



ARCnet그림을 멋있게 바꾼 결과

퀴즈

1. 메타모델(metamodel)이란?
2. 분류자(classifier)란?
3. UML을 확장하는 능력이 왜 중요할까?
4. UML 확장 매커니즘이란 무엇일까?

Part 15. 시스템 개발에 UML 적용

(1)바람직한 개발과정이란?

- 오늘날의 업무 환경에서 요구하는 복잡한 시스템을 개발하기 위해서는 팀 작업(team approach)이 필수적이다. 왜 팀 작업이 필수적일까? 한 사람이 업무 환경을 파악하고 문제를 이해하여 솔루션을 설계하고, 이것을 프로그램으로 옮겨서 하드웨어에 전개하고 연동시키기에는 지식이라는 것이 너무나 세분화되어 있기 때문이다.
- 팀에는 분석가가 필수적이다. 분석가는 의뢰인과 만나 그들의 문제를 파악하고, 솔루션을 설계하는 설계자와 그것을 코딩하는 프로그래머, 그리고 솔루션을 설치하는 시스템엔지니어와 계속적으로 의견을 주고 받는다. 시스템 개발 과정에서 각각의 모든 역할을 고려하여 적절히 이용하고, 각 단계에서 소요될 시간을 적당히 분배해야 한다. 그리고 작업 진척을 나타내는 작업 결과물(work-product)을 계속 내면서 다음 작업을 지정, 변경해 가야 한다.
- 개발과정에서 작업의 각 단계가 서로 떨어지지 않도록 해야한다. 그 대신, 각 단계에서 얻어진 작업 결과물을 피드백으로 계속 활용하여 새로운 노하우를 해당 작업 단계에 쉽게 끌어들이 수 있도록 해야 한다.

※ 바람직한 개발 과정을 다섯가지로 정리 해보자.

- 해결하고자 하는 문제를 개발팀이 명확하게 이해할 수 있도록 한다.
- 한 팀이 여러 가지 역할을 가진 멤버로 이루어질 수 있다는 것을 고려한다.
- 각각의 역할을 맡은 멤버간의 의사소통이 원활 하도록한다.
- 프로젝트 진행중에 각 단계간의 피드백을 수용하고 고려한다.
- 의뢰인에게 보여줄 수 있는 작업 결과물을 만들어내되, 불필요한 문서화는 하지 않는다.

GRAPPLE의 구조 (RAD³)

1. 요구사항 수집(Requirement Gathering)
2. 분석(Analysis)
3. 설계(Design)
4. 개발(Development)
5. 배포(Deployment)

(2)요구사항 수집

- 다섯 개의 진행 영역에다가 중요한 순서로 번호를 매겨 본다면, “요구사항 수집”이 가장 우선시 되어야 한다. 의뢰인이 무엇을 원하는지도 알지 못하는데, 어떻게 제대로 된 시스템을 구축할 수 있겠는가. 유스 케이스 분석을 또올릴지 모르겠지만 이것도 의뢰인이 가지고 있는 도메인의 핵심을 이해하지 않으면 아무 짝에도 쓸모가 없다.

2-1)업무 과정 파악

- 훌륭한 개발 작업은 의뢰인의 업무가 어떻게 진행되는 지를 이해하는 것부터 시작된다. 분석가는 의뢰인 혹은 의뢰인이 인정한 “관계자”를 만나서 해당 업무가 어떤 단계로 진행되는지에 대한 이야기를 나눈다. 여기서 얻어지는 귀중한 자료는 의뢰인의 도메인에 맞는 용어의 집합일 것이다. 분석가는 차후에 의뢰인을 만나 이야기할 때 이 어휘를 사용하게 된다.
- 이때 만들어지는 작업 결과물은 해당 업무 과정에서 단계와 결정 위치(decision points)를 정리하여 나타낸 활동 다이어그램이다.

2-2)도메인분석

- 이 동작은 앞서 알아본 “농구팀 감독 인터뷰” 과정과 비슷하다. 이 동작은 앞 동작과 같이 의뢰인이나 의뢰인이 인정한 관계자와 만나 이야기를 나눌 때 수행되며, 의뢰인의 도메인을 가능한 더욱 탄탄하게 파악하기 위함이 목적이다. 이 동작과 앞 동작은 개념적인 성격을 가지고 있고, 구축하고자 하는 구체적인 시스템 사항은 아직 나타나지 않는다. 분석가는 최대한 의뢰인에게 모든 것을 알아야한다.

- 분석가와 의뢰인이 대화가 진행되는 동안, 다른 팀 멤버는 쉬고 있을까? 아니다 필요하다면 대화에 같이 참여하여 워드프로세서 설치된 노트북으로 대화내용을 정리하고, 객체모델러는 추상적인 수준의 클래스 다이어그램을 그린다. 이러한 팀 멤버는 될 수 있으면 모두 참여 시키자.

2-3)연동 시스템 확인

- 어떤 시스템이든 다른 시스템들과 함께 동작하지 않는 좋은 시스템은 없다. 이 동작의 초기에 개발팀은 자신들의 만든 새로운 시스템이 의존할 시스템과 이 시스템과 잘 맞물리는 시스템을 찾아야한다. 시스템 엔지니어는 이동작의 진행 과정을 차근이 예의 주시한 다음, 배포 다이어그램을 작업 결과물로 만들어 낸다. 이 다이어그램에는 노드, 노드간의 접속, 상주(resident) 컴포넌트 그리고 컴포넌트간 의존 관계가 나타나 있다.

2-4)시스템 요구사항 파악

- JAD모임에는 의뢰인이 속한 조직체의 의사 결정권자와 잠재 사용자, 개발팀 멤버들이 참석한다. 의사 진행자(facilitator)는 모임의 진행을 맡는다. 이 사람의 임무는 의사 결정권자와 사용자들이 무엇을 원하는 지에 대한 의견을 끌어 내는 것이다. 개발팀 멤버 중에 최소한 둘은 의견을 노트하고, 객체 모델러는 앞서 그려 두었던 클래스 다이어그램을 손질한다.

- 여기서 만들어지는 작업 결과물은 패키지 다이어그램이다. 여기에는 추상적인 수준의 시스템 기능이 담겨 있다.

(3)분석

- 이 진행 영역에서 “요구사항 수집”진행 영역의 결과를 가지고 더욱 자세히 작업하여, 문제의 이해도를 높인다.

3-1)시스템 사용법의 이해

- 이 동작은 추상 수준의 유스케이스 분석 작업이다. 개발팀은 JAD모임에서 각 유스케이스를 시작하게 하는 행위자와 그 유스케이스의 결과를 얻는 행위자를 알아낸다. 모임 진행자는 각 참석자와 의견을 조정하고, 개발팀 중 둘은 내용을 노트한다. 프로젝트 몇 개를 더 수행한후에는 이 모임 진행자가 유스케이스 분석가가 될 확률이 크다.

-또한 개발팀은 유스케이스를 새로 만들어내고 추상 유스케이스도 만들어야한다. 이 동작에서 만들어지는 작업 결과물은 행위자와 유스 케이스간에 스트레오 타입을 가진 의존 관계를 나타내는 유스케이스 다이어그램이다.

3-2)유스 케이스에 살 붙이기

- 이 동작에서는 개발팀이 사용자와 작업을 계속하여 각 유스케이스를 구성하는 단체의 순서를 분석해낸다.

3-3)클래스 다이어그램 손질

- 객체 모델러는 회의 내용을 모두 듣고 자신의 클래스 다이어그램을 계속 고쳐간다. 이 때, 객체 모델러는 연관, 추상 클래스, 다중성, 일반화, 집합연관 등의 내용을 적어 넣어야 한다. 여기서 만들어지는 작업 결과물은 손질이 끝난 클래스 다이어그램이다.

3-4)객체의 상태 변화 분석

- 객체 모델러는(필요할 때마다) 상태 변화를 나타냄으로써 모델을 좀 더 자세히 만든다. 작업 결과물은 예상

대로 상태 다이어그램이다.

3-5)객체간 교류 정의

- 개발팀은 유스케이스 다이어그램과 손질이 끝난 클래스 다이어그램을 가지게 된다. 이제는 객체들이 어떻게 교류하는지 정의해야 한다. 객체 모델러는 시퀀스 다이어그램과 협력 다이어그램을 그린다. 여기에 상태 변화를 포함시켜도 좋다. 이 다이어그램들은 이 동작에서 작업 결과물이 된다.

3-6)연동 시스템과의 통합 분석

- 시스템 엔지니어는 앞서 이야기한 모든 단계와 보조를 맞추면서, 연동 시스템과의 통합에 관련한 세세한 사항을 분석하고 알아내야 한다. 어떤 통신 접속 방법을 사용할 것인지, 네트워크 구조는 어떤지 등 그리고 만일 이시스템이 데이터베이스를 액세스해야 한다면 데이터 베이스 분석가는 이 데이터베이스의 구조를 모두 결정해야한다.

(4)설계

- 이 진행 영역에서는 “분석”진행 영역에서 얻어낸 결과를 가지고 솔루션을 설계한다. “분석”과 “설계” 진행 영역은 설계가 완료될 때까지 자유롭게 오갈 수 있다. 어떤 방법론에서는 분석과 설계를 하나의 단계에 넣기도 한다.

4-1)객체 다이어그램의 개발과 손질

- 프로그래머는 클래스 다이어그램을 가지고 필요한 객체 다이어그램을 그린다. 각각의 오퍼레이션을 체크하고 여기에 맞는 활동 다이어그램을 그림으로써 객체 다이어그램은 점점 완전한 모습을 찾아간다. 활동 다이어그램은 “개발”진행 영역에서 대부분의 코딩 작업 기반을 제공한다. 여기에서 만들어지는 작업 결과물은 객체 다이어그램과 활동 다이어그램이다.

4-2)컴포넌트 다이어그램의 개발

- 프로그래머가 제구실을 하는 동작 단계가 바로 이 단계이다. 여기서 할 일은 컴포넌트를 시각화하고 각각의 의존 관계를 나타내는 것이다. 즉, 작업 결과물은 컴포넌트 다이어그램이다.

4-3)배포계획

- 컴포넌트 다이어그램이 만들어지고 나면, 시스템 엔지니어는 시스템 배포와 연동 시스템과의 통합을 계획하기 시작한다. 시스템 엔지니어는 배포 다이어그램에 컴포넌트를 적절히 배포해 그려 넣는다. 여기서 만들어지는 작업 결과물은 이전에 그린 배포 다이어그램의 일부가 될부분은 다이어그램이다.

4-4)사용자 인터페이스의 설계와 원형정의

- 사용자와 JAD모임을 한 번 더 연다. 비록 “설계”진행 영역의 일부이긴 하지만, 이 모임은 바로 이전에 가졌던 JAD 모임을 끝어서 여기까지 진행할 수 있다. 사용자 인터페이스는 모든 유스 케이스를 고려하여 만들어져야 한다. GUI분석가는 사용자와 함께 각각의 유스케이스에 맞는 화면 프로토타입을 종이에 그려낸다. 여기서 작업 결과물은 화면 원형을 잡은 스크린샷 이다.

4-5)시험 설계

- 유스 케이스는 소프트웨어를 시험하는데 설계의 용도로 쓰인다. 작업 결과물은 시험 스크립트가 된다.

4-6)문서화 시작

- 최종 사용자나 시스템 관리자를 위한 시스템 설명서를 작성하는 일은 지금 시작해도 결코 빠르다고 할 수 없다. 문서화 저능가는 시스템 설계자와 함께 스토리보드를 작성하기 시작하고 각 문서의 골격을 만든다. 이 때 만들어지는 작업 결과물은 문서의 골격이다.

4-7)시험 설계

- 유스케이스는 소프트웨어를 시험하는데 설계의 용도로 쓰인다. 즉 작업 결과물은 시험 스크립트가 된다.

4-8)문서화 시작

- 최종 사용자나 시스템 관리자를 위한 시스템 설명서를 작성하는 일은 지금 시작해도 결코 빠르다고 할 수 없다. 이 때 만들어지는 작업 결과물은 문서의 골격이다.

(5)개발

- 여기서부터 프로그래머가 나서야한다.

5-1)코드작성

- 프로그래머는 이미 완성된 클래스 다이어그램, 객체다이어그램, 활동다이어그램, 컴포넌트 다이어그램을 가지고 시스템에 필요한 코딩을 시작한다.

5-2)코드시험

- 시험전문가(개발자가 아니다)가 하는 작업으로써, 작성된 코드가 원하는대로 동작하는 지를 시험 스크립터를 사용하여 평가한다. 이 시험 결과가 작업 결과물이며, 해당 코드가 모든 시험을 통과할때까지 “ 코드작성 ” 단계에 이것을 반영하여(피드백으로)사용할수 있다.

5-3)사용자 인터페이스의 구축과 코드의 연결 및 시험

- 사용자의 확인을 거친 사용자 인터페이스를 직접 구현한다.

5-4)문서화 완료

- 문서화 전문가가 프로그래머와 손발을 맞추어 동시에 진행시키는 문서화 작업은 소프트웨어의 개발 완료와 동시에 끝난다.

(6)배포

- 개발 과정이 끝나면 하드웨어에 설치하여 다른 시스템과 연동시키는 일을 한다.

6-1)백업과 복구에 대한 계획

- 시스템 엔지니어는 시스템이 충돌했을 경우 어떤 조치를 취해야 하는지를 계획한다.

6-2)적절한 하드웨어에 최종 시스템 설치하기

- 시스템 엔지니어는(필요한 경우) 프로그래머의 도움을 받아 컴퓨터에 최종적으로 만들어진 소프트웨어를 설치한다.

6-3)설치된 시스템의 시험

- 시스템이 처음에 의도한 대로 동작하는지? 백업/복구 조치가 제대로 이루어지는지를 체크한다.

(7)발표

- 말그대로 발표를 한다.

🍷 퀴즈

1. JAD모임이란?

Part 16. 사례연구

(1)업무과정파악

- A사는 전세계적인 규모의 요식업 운영을 할려고 계획 중이다 그들은 회사를 설립하고 능력과 경험이 충분한 식당 지배인, 웨이터, 요리사 그리고 기타 스태프진을 고용하였다.

- 하지만 가장요한 것은 미래의 식당이 갖추어야 할 기반 기술이다. 이것이 제대로 갖추어지면 외식의 즐거움을 높여줄 그들만의 식당을 처음으로 열 수 있게 된다.

1-1)업무과정파악 - 고객에 대한 서빙(인터뷰)

분석가 : 나와 주셔서 감사합니다.

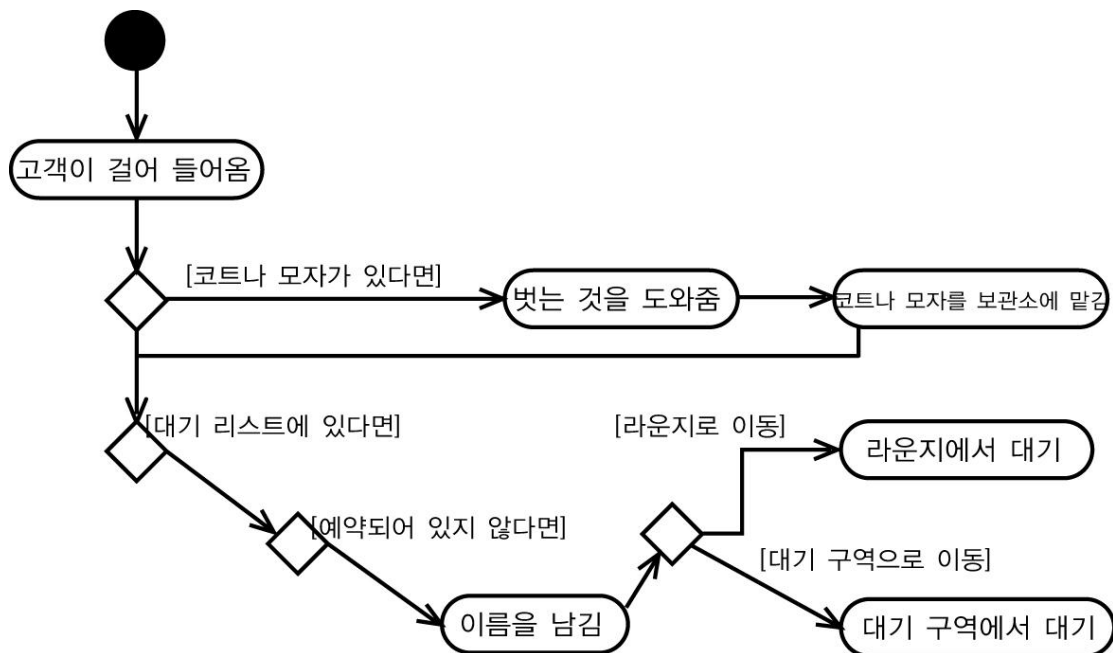
지배인 : 저도 기쁩니다. 정확하게 알고 싶으신 것이 무엇입니까

분석가 : 가장 기본적인 것부터 시작하지요. 고객이 식당안으로 걸어 들어갈 때 어떻게 됩니까?

지배인 : 다음과 같습니다. 만일 고객이 코트를 입고 계신다면 벗는 것을 도와드리고 식당의 보관소에 둡니다. 물론, 고객에게 코트 보관증을 드리지요. 모자의 경우도 마찬가지입니다. 그리고..

분석가 : 잠깐만요 식당에 대기 구역이 있다고 해보세요. 고객이 거기에 가서 이름을 남기지 않습니까? 혹은..

지배인 : 아닙니다. 저희는 고객이 최대한 편하게 느낄 수 있는 방향으로 일을 진행합니다. 그리고 대기 구역이 있다고 해도 신경 쓰지 않습니다.



☞ 식당 업무 과정인 “고객에 대한 서빙” 대한 활동 다이어그램의 시작 단계

다시 인터뷰로 돌아가면

분석가 : 대기 리스트에 있는 손님 차례가 되거나 예약 손님이 식당에 왔을 때 그분들은 앉힌다는거죠?

지배인 : 맞습니다. 하지만 생각하는 것만큼 간단하지 않습니다. 테이블이 준비되어 있어야하고, 물론 냇끝

해야하 합니다. 테이블 담당은 이전에 온 손님이 사용했던 테이블보를 치우고 새 손님을 받을 수 있게 세팅 합니다. 이것이 끝나면 주인이 손님들을 테이블로 모시고 웨이터를 부르지요

분석가 : 불려요 ?

지배인 : 예. 이과정은 복잡하지 않습니다. 왜냐하면 웨이터마다 맡은 서빙 구역이 있고, 언제 테이블이 준비되는지 대개 알고 있기 때문입니다. 웨이터들은 맡은 구역을 날아 다니듯이 식당 주인의 제스처를 재빨리 보고 따릅니다.

분석가 : 그다음에는요?

지배인 : 웨이터는 테이블 옆에 섭니다. ‘그’는 메뉴를 보여주고 손님이 음식을 결정하는 동안 마실 것을 주문할 것인지를 묻습니다. 다음에, 웨이터는 빵과 버터 쟁반을 가지고 다니며 물을 따라 주는 보조웨이터를 부릅니다. 누군가가 마실 것을 주문하면, ‘그’가 가져오는 거죠

분석가 : 잠깐만요 ‘그’라고 말씀하셨는데요, 테이블에서 시중드는 사람은 항상 남자입니까?

지배인 : 아닙니다. 여자일 수도있습니다.

분석가 : 그러면 ‘서빙담당’이라는 용어를 쓰면 어떨까요? 그리고 손님은 현재 마실 것을 주문할 기회를 두 번 가지는 것으로 파악하였습니다.

지배인 : 맞게 생각하셨네요. 만일 손님이 테이블이 비기까지 기다리면서 라운지에서 한잔 하고 있을 때 (테이블이 비었을 상황에 말이죠), 잔에 음료수가 아직 남아있다면 그것을 테이블로 가지고 가도록 하지요. 하지만 너무많이 마신분에게 이러한 서비스를 거부할 권리는 남겨두고 있습니다.

분석가 : 말씀 해주셔서 감사합니다. 이제, 테이블에서 메뉴를 고르는 부분으로 되돌아와 보죠.

지배인 : 네. 저희는 정규 메뉴에는 없는 “그날의 특별식”을 준비해 놓고 있습니다. 그리고 서빙담당은 손님에게 이것을 알려줍니다.

분석가 : 여기도 상당히 많은 일이 일어날 것 같은데요? 손님들이 서빙 담당에게 추천해 달라고 부탁하는 경우가 있을테고, 대개 서빙 담당은 솔직하게 말해 줄 것 같아요. 이음식이 저음식보다 좋은지, 나쁜지요 이것이 지배인님이 바라는 바인가요?

지배인 : 네. 그렇습니다. 사실, 서빙담당들은 모두 저희 식당에서 식사를 하죠, 음식마다 각자 의견을 가지고 있습니다. 정말 그들이 어떤 특정한 음식을 싫어할 경우에는, 저희입장에서는 그들의 그것을 손님에게 말해주기 전에 주방장에게 알려주었으면 좋겠습니다. 하지만 각자의 취향을 표현하는 것도 나쁘지 않지요. 물론, “ 이 음식은 악취가 장난아니예요”란 식으로 말하는 것을 좋아할 주인은 없습니다. 그러나, 다른 부분에 대한 의견은 얼마든지 표현할 수 있는 것 아니겠어요?

분석가 : 이해했습니다. 좋아요, 이저 정리해 봅시다. 손님 그리고.. 음 이 손님이 실질적인 손님 단체이죠? 코트를 벗고, 라운지에서 기다리고, 테이블이 비기를 기다리고, 자리가 생기면 앉고 마실 것을 주문할 수 있고 빵과 물을 서비스 받을 수 있고 그리고 메뉴를 보는 것입니다. 맞나요?

지배인 : 맞습니다. 서빙담당은 마실 것을 가지고 테이블로 되돌아오고요, 손님에게 메뉴 시간을 5분내지 10분정도 주고 기다리다가 다시 옵니다. 물론, 손님이 일찍 메뉴를 골랐을 때 는 더빨리오지요.

분석가 : 어떻게 일찍 되돌아올 지를 압니까?

지배인 : 어떻게든 낚새를 알아채야하지요. 서빙 담당은 주방으로 와서 주방장에게 무슨 말을 건네지 않는 한은 대개 해당 테이블의 구역에 있게 됩니다.

분석가 : 구역이요?

지배인 : 네. 서빙 담당들은 각자 몇 개씩의 테이블을 맡고 있습니다. 이 테이블들이 있는 곳이구역이죠. 한 구역은 흡연자용 구역으로 지정되고요, 나머지는 비흡연자용 구역입니다.

분석가 : 어떤 구역에 누가 일할지는 어떻게 결정하나요?

지배인 : 매번 돌려서 순서를 바꿉니다.

분석가 : 이제 서빙 과정으로 되돌아와 보죠. 손님은 원하는 음식을 메뉴에서 고르고, 서빙담당을 받아 적습니다 그리고...

지배인 : 그리고 주방장에게 알립니다. 선택사항은 주방장이 알아볼 수 있는 폼에 적지요

분석가 : 폼이라고요?

지배인 : 테이블 번호, 선택사항 그리고 시간(이건 상당히 중요합니다)입니다.

분석가 : 왜중요한가요?

지배인 : 주방은 대개(저희가 바라는 바이지요) 매우 바쁜 곳이고, 주방장은 주문이 들어온 순서에 따라 무엇을 만들어 둘 지에 관한 우선순위를 정해야 하기 때문입니다.

분석가 : 그것이 복잡해질 수 있나요?

지배인 : 실제로는 조금 그렇습니다.

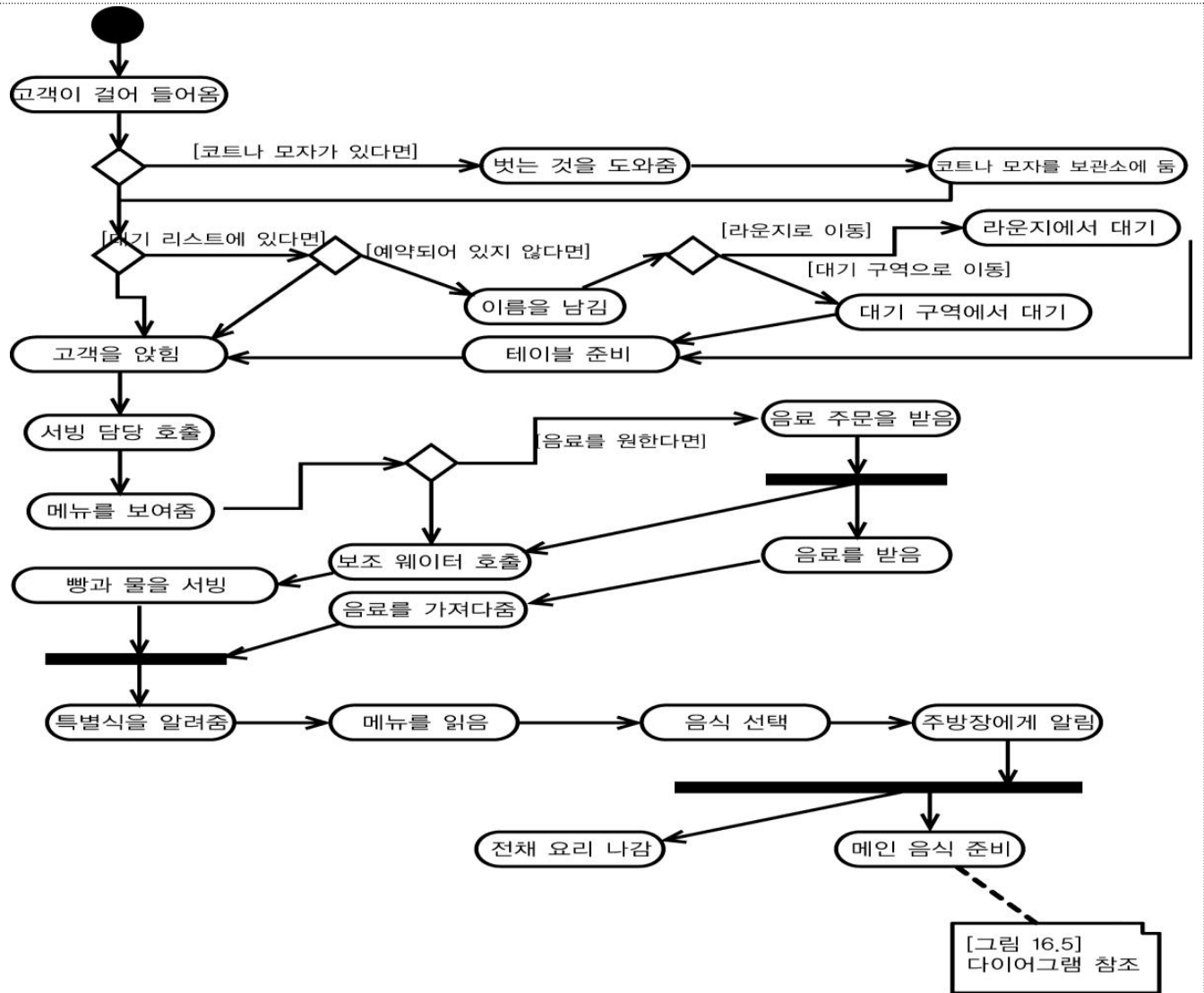
분석가 : 어떻게요?

지배인 : 대부분의 음식은 메인 코스 전에 전체 요리가 나가죠. 대부분의 손님들은 메인 코스가 식으면 별로 안 좋아합니다. 따라서 주방장은 전체요리를 준비합니다.(많은 부분들이 이미 만들어져 있습니다. 샐러드 같은거죠)그리고 서빙 담당은 이것을 가지고 나갑니다. 여기서의 '역경'은 한 테이블에 앉은 모든 사람에게 메인 코스 음식을 동시에 그리고 식지 않게 준비해야 한다는 것입니다. 제가 역견이랑 말을 쓴 이유는 테이블에 있는 사람들이 전체 요리를 각각 다른 시간에 다 먹기 때문입니다. 조정이 필요하죠

분석가 : 독립적으로 진행시켜야 한다는 것으로 들리는 군요. 이것이 조금 다르게 이야기 해볼까요? 주방장의 시점에서요.

지배인 : 좋습니다. 아주 좋은 생각 같아요

분석가 : 이제, 주방장이 메인 코스를 요리할 시점까지 온 것 같습니다. 지금까지의 과정을 다이어그램으로 나타내면 어떻게 될까요



지배인 : 제대로 보냈습니다. 어쨌든, 주방장은 메인 코스를 요리하고, 서빙 담당은 테이블에 앉은 손님들이 전체 요리를 다 먹었을 때 이것을 집어들고 갑니다. 손님은 음식을 먹고요. 서빙담당은 테이블 상황을 살피기 위해 최소한 한 번은 옵니다.

분석가 : 손님이 음식의 어떤 부분에 대해 불만을 표시 했다면요?

지배인 : 손님이 만족할 때까지 최선을 다합니다. 저희가 돈을 지불해야 하더라도요. 손님을 잃는 것보다 돈을 좀 쓰는게 낫습니다.

분석가 : 멋진 생각이십니다.

지배인 : 감사합니다. 손님의 식사가 끝나면, 서빙담당이 다시 와서 후식을 먹을 건지 묻습니다. 만일 먹겠다고 하면, 후식 메뉴를 보여주고 주문을 받지요. 만일 먹지 않겠다고 하면 커피를 드실것인지 못습니다. 커피를 마시겠다고 하면 커피와 커피 잔을 내어 오고 퍼키를 부어 줍니다. 손님이 아무것도 원하는 않을 경우에는 계산서를 가지고 옵니다. 몇 분이 흐른 다음, 현금또는 신용카드를 받지요 그리고 거스름돈이나 카드 영수증을 줍니다. 손님은 팁을 주고 자신의 코트를 다시 입은 다음 나갑니다.

분석가 : 끝났나요?

지배인 : 아닙니다. 서빙담당은 테이블에 담당을 불러서 테이블을 치우게 하고 다시 세팅해서 다음 손님을 받을 수 있도록 해야 하지요.

분석가 : 지금까지 손님에 관한 내용은 하나도 없는 것 같은데요. 짧지만 별도의 과정을 하나 더 생각해 두려고 합니다. 두가지를 묻고 싶네요 첫째로, 서빙담당은 손님이 식사를 다했는지 어떻게 압니까?

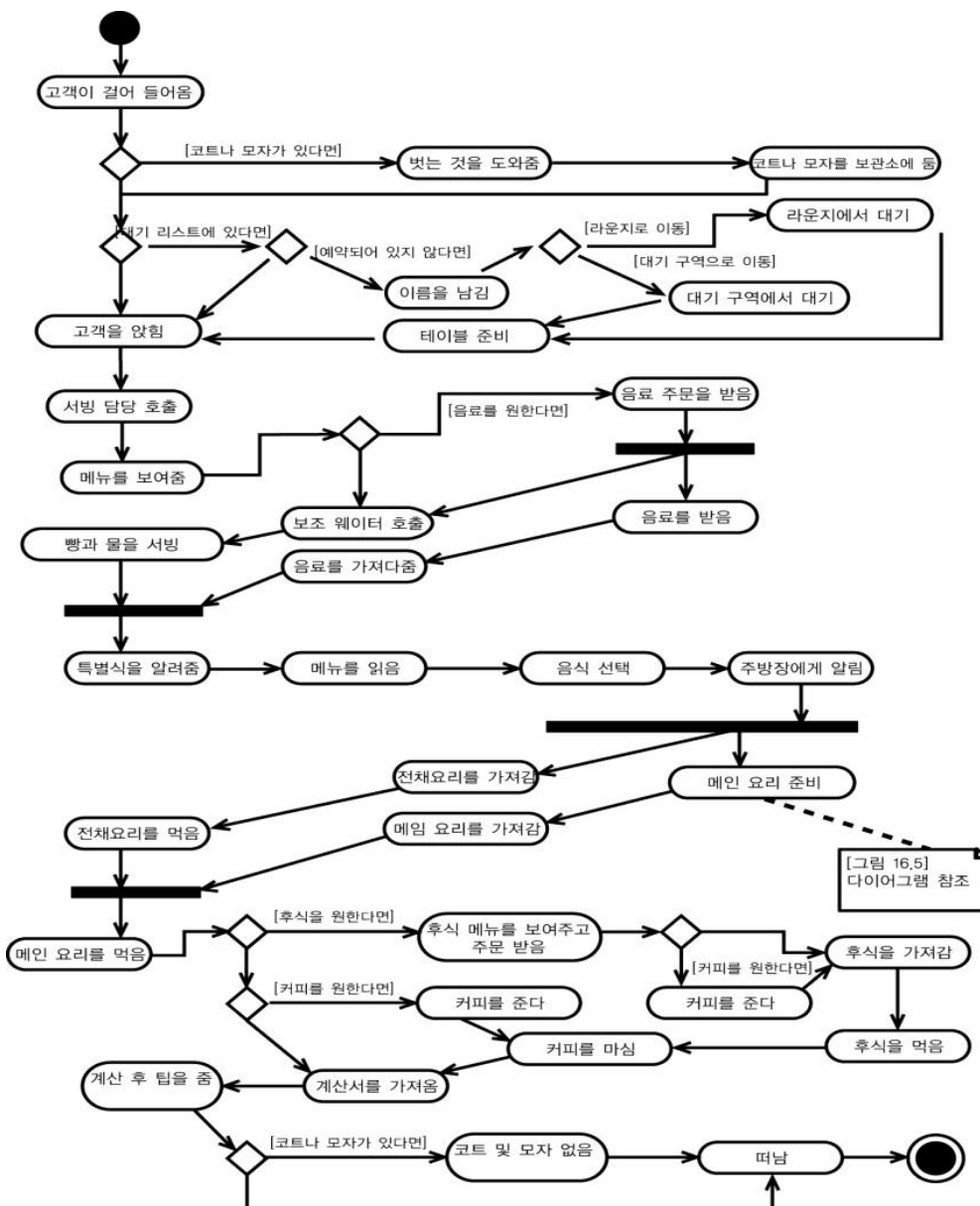
지배인 : 자신의 구역에 있으면서 맡은 테이블을 살펴 봅니다. 각자의 경험에 따라, 식사를 하는데 보통 시간이 얼마나 걸리는지 알고 있지요. 따라서 언제쯤 테이블 근처에 다가갈 지를 예상할 수 있습니다. 두 번째 질문은 뭐지요?

분석가 : 네 앞서서 지배인님께서서는 서빙 담당이 주방장에게 할 말이 있을 때 주방에 간다고 하셨는데요, 이런 경우가 왜 발생합니까?

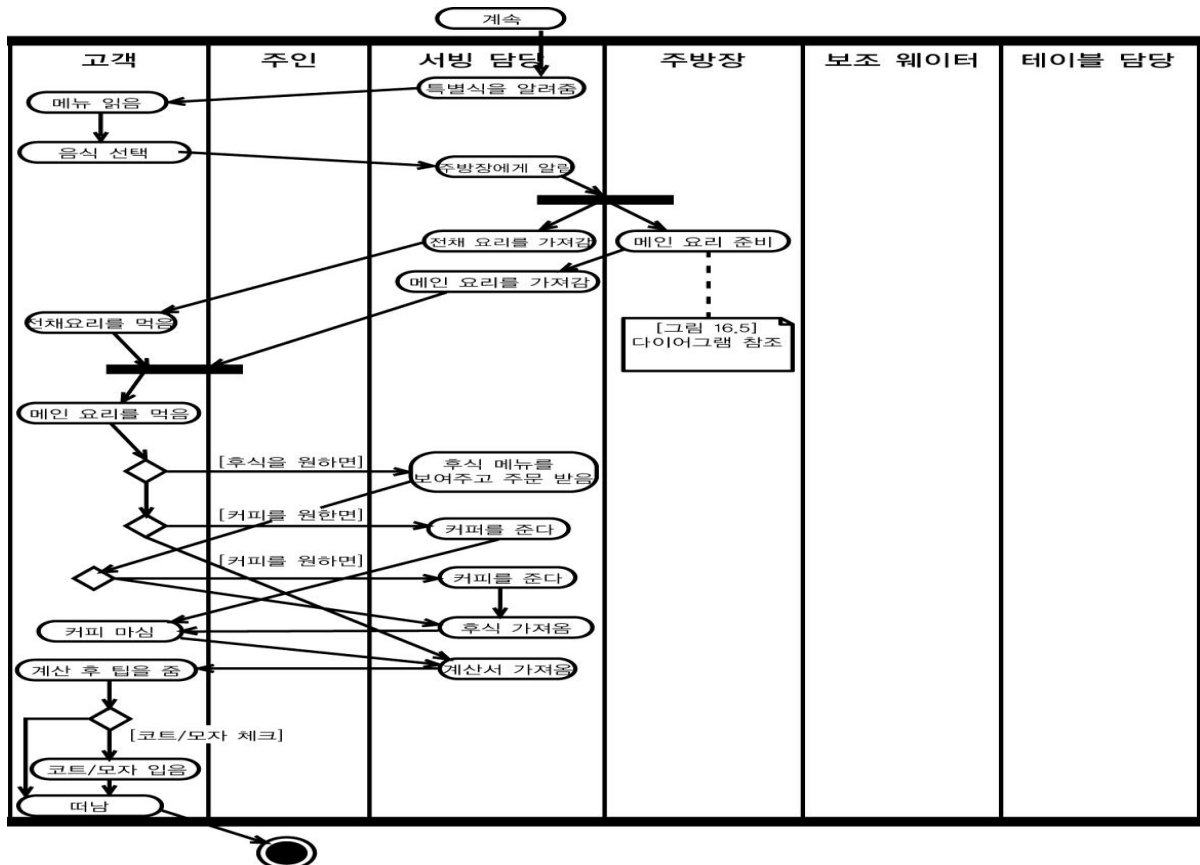
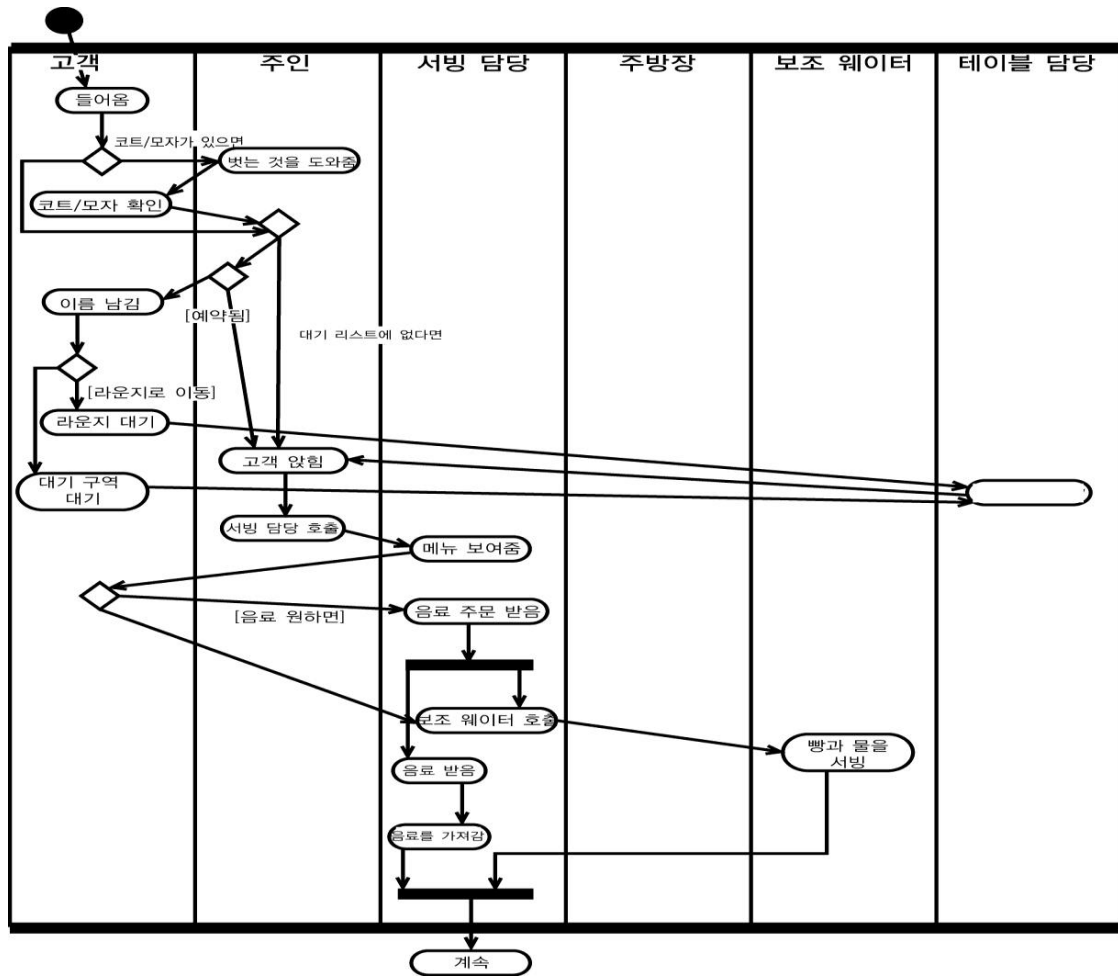
지배인 : 음식이 언제쯤 나오는지 손님이 알고 싶어하는 경우가 가끔 있습니다. 이 때 손님이 서빙 담당을 부르죠. 서빙 담당은 주방장에게 물어 볼 수밖에 없구요. 물어보고 난 다음에는 다시 손님에게 가서 말해 줍니다.

분석가 : 알고 계시겠지만, 저는 식당에서 손님을 맞이할 때 수행하는 일들을 전혀 몰랐습니다.

지배인 : 재미있네요. 그럴줄 알았어요 제게 모든 단계를 세세하게 물어주시기 전에는 저도 이게 얼마나 많은지 몰랐습니다.



- 손님에 대한 서빙 업무과정에 대한 구획면 다이어그램



음식 준비하기

분석가 : 저에 이야기했을 때 지배인님께서서는 대부분의 음식은 메인 코스 이전에 전체 요리를 제공한다고 하셨습니다. 그리고 대부분의 사람들은 메인 코스가 식으면 싫어한다고 하셨지요. 테이블에 앉아 있는 손님 모두에게 동시에 메인 코스를 대접하는 일과 음식이 식지 않도록 하는 일이 역경이라고 하셨고, 조정이 필요하다고 하셨습니다

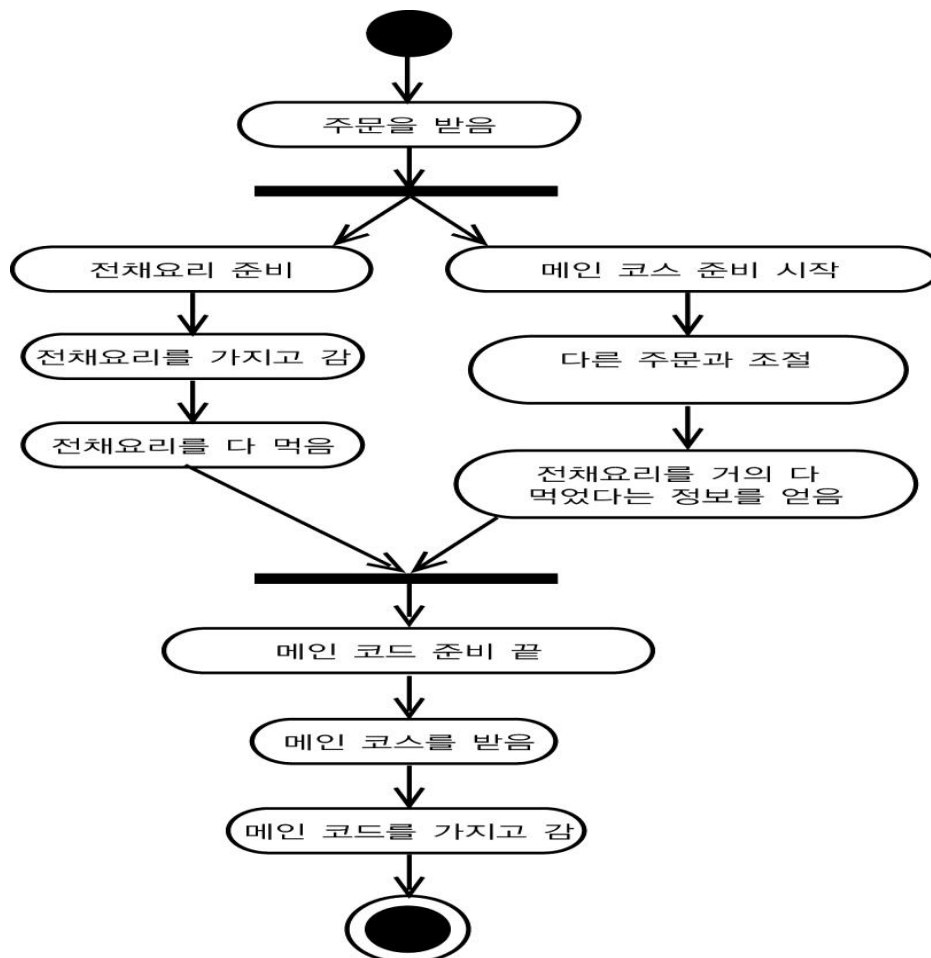
지배인 : 물론입니다. 테이블에 계신 손님들은 거의 항상 동시에 전체 요리를 다 드시는 일이 없습니다. 따라서 모든 분께 따뜻한 메인 코스를 대접할 때 조정이 필요하지요. 이 조정은 서빙 담당과 주방장 사이에 이루어집니다. 주방장은 서빙 담당으로부터 주문을 받아 전체 요리를 준비하고 메인 코스를 요리하기 시작하지요. 전체 요리가 다 만들어지면, 서빙담당이 주방으로 와서 이것을 가지고 테이블로 갑니다.

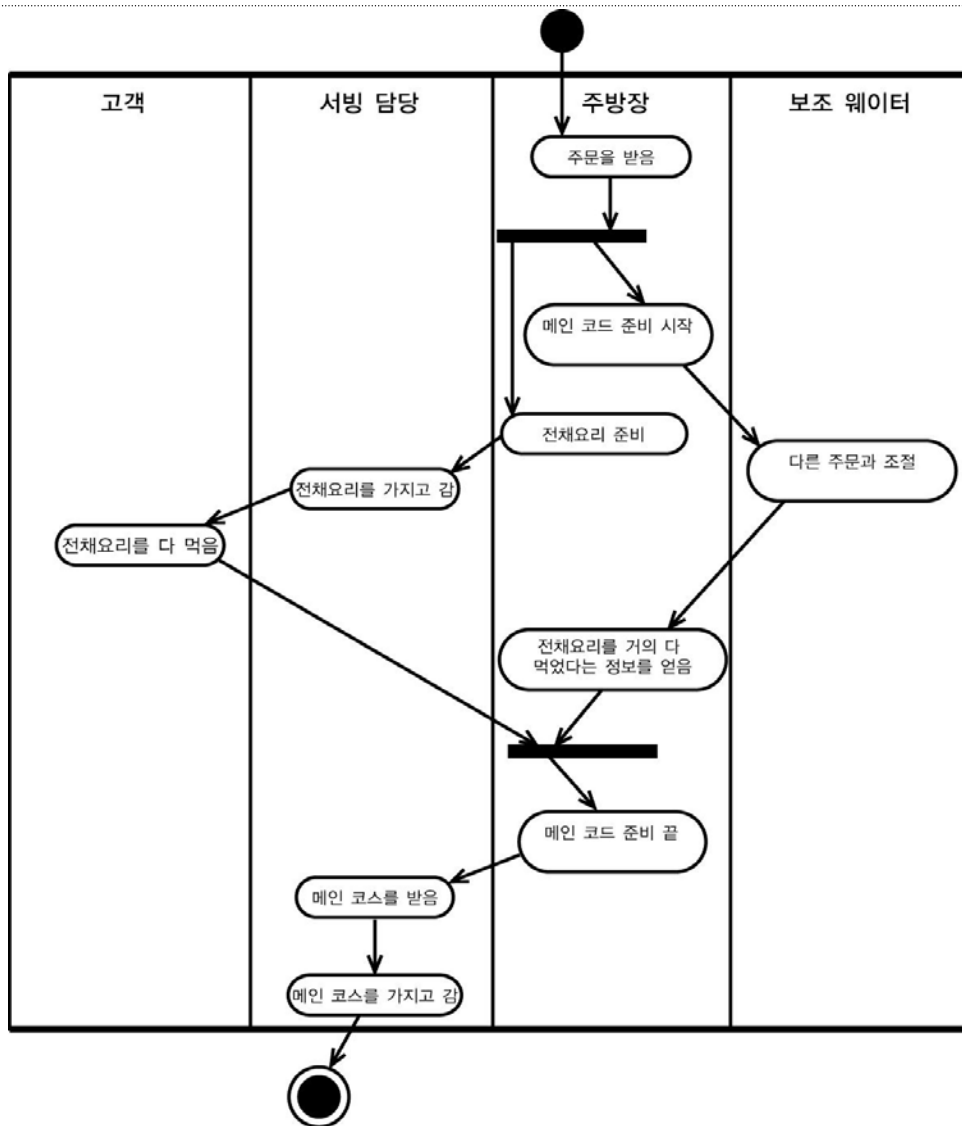
분석가 : 그리고 서빙담당은 ‘어찌어찌’하여 전체 요리가 완성되었다는 것을 알겠네요?

지배인 : 서빙담당은 가끔씩 주방을 둘러보니까 가능한 것이죠. 이제, 여기서 조정이 이루어집니다. 주방장은 서빙담당에게 전체 요리를 건네주고, 메인 코스 요리를 마지막으로 손질하기 전에 서빙 담당이 “테이블의 손님들이 전체 요리를 거의 다 먹었다”라고 알려줄 때까지 기다립니다. 서빙담당은 자신이 맡은 구역에 무뎠러 있으면서 테이블을 계속 예의 주시하지요. 적당한 때가 되면 주방으로 가서 주방장에게 “메인 코스를 낼 순서가 되었다”라고 알립니다. 이 때 주방장은 요리를 끝내지요. 능란한 주방장은 주방 보조들과 호흡을 맞추어서 여러 테이블에 앉은 손님들에게 대접할 요리 준비를 동시에 수행할 수 있습니다. 하여간 목표는 테이블의 손님들이 메인 코스를 먹을 준비가 되었을 때 바로 메인 코스를 대령하는 것이죠.

분석가 : 이것이 항상 정확하게 제때에 이루어지나요?

지배인 : 아뇨 꼭 그렇지는 않습니다. 하지만 경험과 상식이 조금 있다면 점점 정확하게 일을 할 수 있게 됩니다. 가끔 한 테이블에 있는 손님들 중 한 분이 메인 코스가 다 준비되었는데도 여전히 전체 요리를 먹고 있는 경우가 있긴 하지만 아주 사소한 경우입니다.





Part 17. 사례연구 - 도메인분석

(1)인터뷰 분석하기

- 업무과정 인터뷰의 상황에서 초기클래스 다이어그램을 만들어야 한다.
- 모델러는 인터뷰에서 나온 명사와 동사, 동사구를 놓치지않고 모아둔다.
- 명사 : 모델내의클래스 가되며, 그중 몇 개는 속성이 되기도한다.
- 동사와동사구 : 오퍼레이션 혹은 연관에 붙일 레이블을 만드는데 사용할 수 있다.

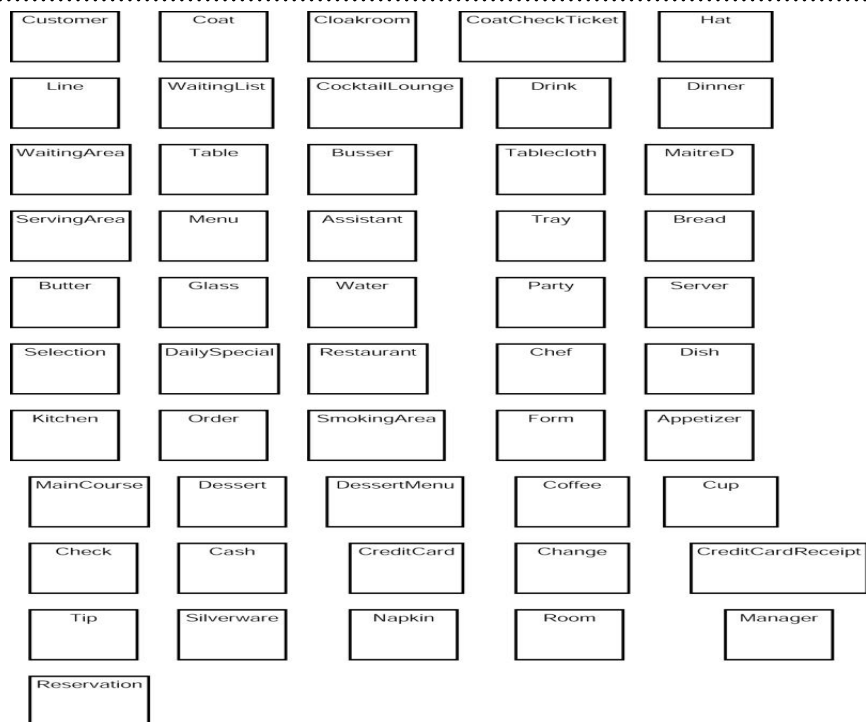
식당 지배인이 사용한 명사

customer,coat,cloakroom,coat-checkticket,hat,line,waitinglist,cocktailounge,drink,dinner,waitingarea,table,busser,tablecloth,maitred,,Ū,servingarea,menu,assistant,tray,bread,butter,glass,water,party,server,selection,dailyspecial,restaurant,chef,dish,kitchen,order,smokingarea,form,appetizer,maincourse,desse
rt,dessertmenu,coffee,cup,check,cash,creditcard,change,creditcardreceipt,tip,silverware,napkin,room,r
eservation,manager.

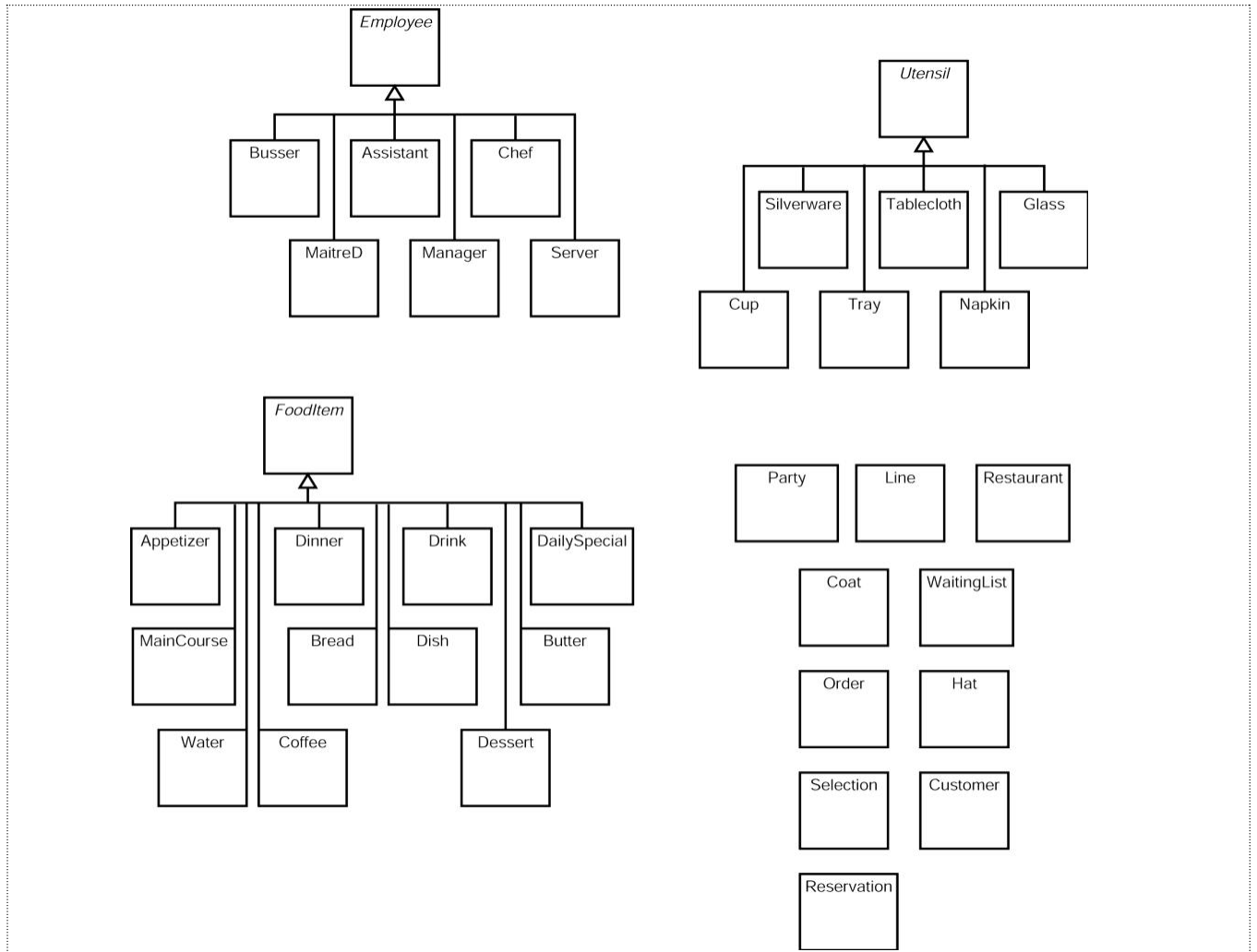
식당 지배인이 사용한 동사와 동사구

has,help,store,give,getinline,honor,seat,leave,sit,wait,comeup,getridof,set,walk,callfor,hover,see,gstur
e,show,ask,order,decide,callover,bring,pour,order,go,get,wait,bring,finish,reserve,refuse,recite,ask,rec
ommend,encourage,like,tell,express,look,comeback,drink,read,allow,makeaselection,getattention,getan
order,talk,assign,designate,determine,notify,write,give,prioritize,consistof,prepare,bring,finish,coordina
te,cook,pickup,eat,comeover,checkon,makesure,cost,losemoney,loseacustomer,comeby,ask,want,provi
de,takeanorder,pour,collect,leave,call,clean,getready,glance,anticipate,talk,comeout,summon,goback,fi
ndout,tell,provide,prefer,finish,coordinate,receive,check,rely,stay,keepaneyeon,clean,makesure,takeca
reof,huntaroundfor,remove,bundleup,fold,arrange,packup,send.

(2)초기 클래스 다이어그램 작성하기



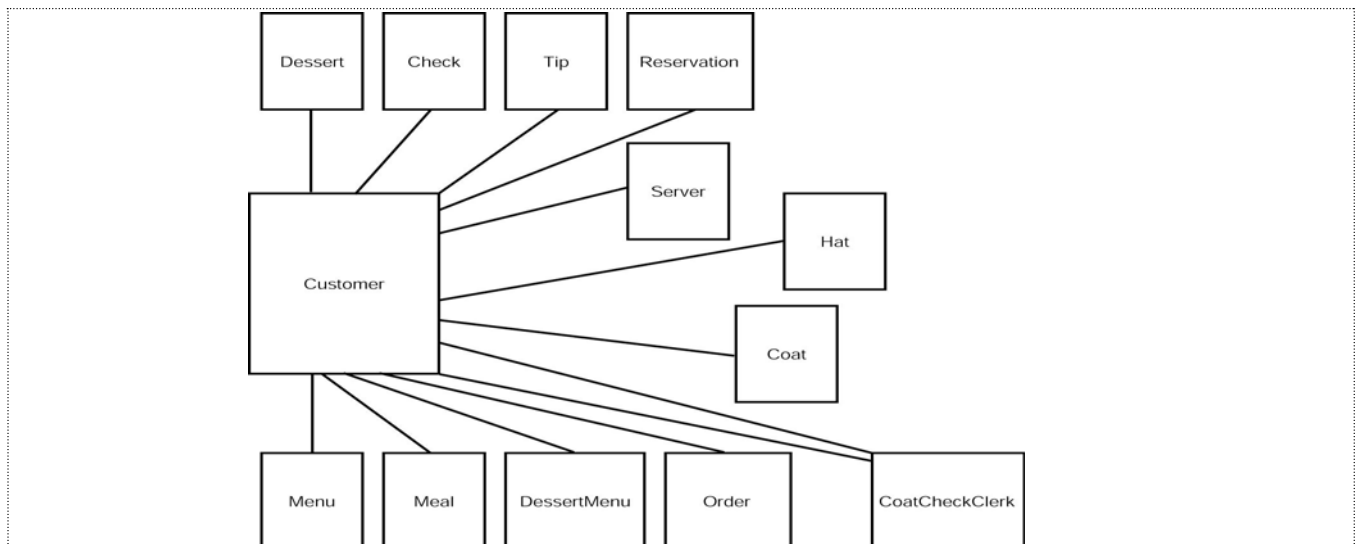
(3)클래스를 그룹화 하기



(4)연관 관계 짓기

- 연관짓기 쉬운 몇 개의 클래스에 대하여 연관관계를 맺어준 후 하나씩 늘려가면서 모든클래스에 대해연관을 짓는다.
- 집합연관과 복합연관을 지어주고, 동사와동사구를 클래스오퍼레이션으로 붙여준다.

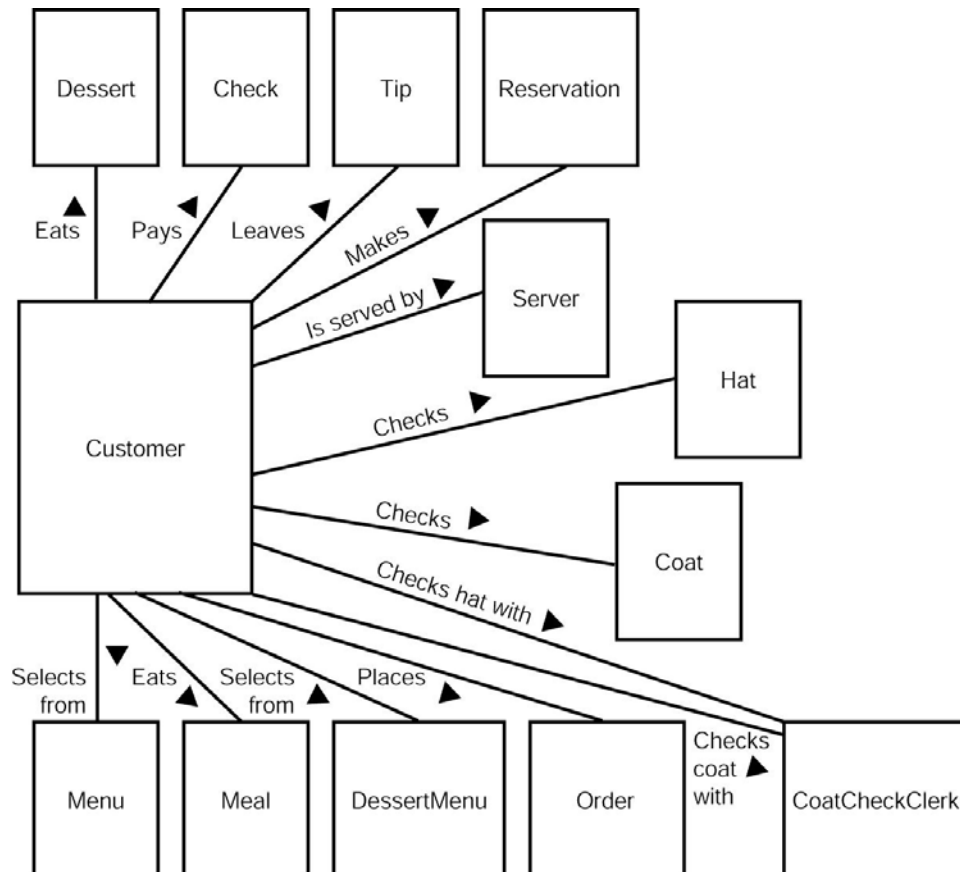
(5)Customer클래스에 대한 초기연관



5-1) 연관 관계의 성격을 나타내는 문구

- 손님이 예약한다(CustomermakesaReservation).
- 손님이 서빙담당에게 시중을 받는다(CustomerisservedbyaServer).
- 손님이 음식을 먹는다(CustomereatsaMeal).
- 손님이 후식을 먹는다(CustomereatsaDessert).
- 손님이 주문한다(CustomerplacesanOrder).
- 손님이 메뉴를 선택한다(Customer selects from a Menu).
- 손님이 후식메뉴를 선택한다(CustomerselectsfromaDessertMenu).
- 손님이 계산서에 대해 지불 한다(CustomerpaysaCheck).
- 손님이 팁을 준다(CustomerleavesaTip).
- 손님이 코트담당에게 코트를 맡긴다(CustomerchecksaCoatwithaCoatCheckClerk).
- 손님이 코트담당에게 모자를 맡긴다(CustomerchecksaHatwithaCoatCheckClerk).

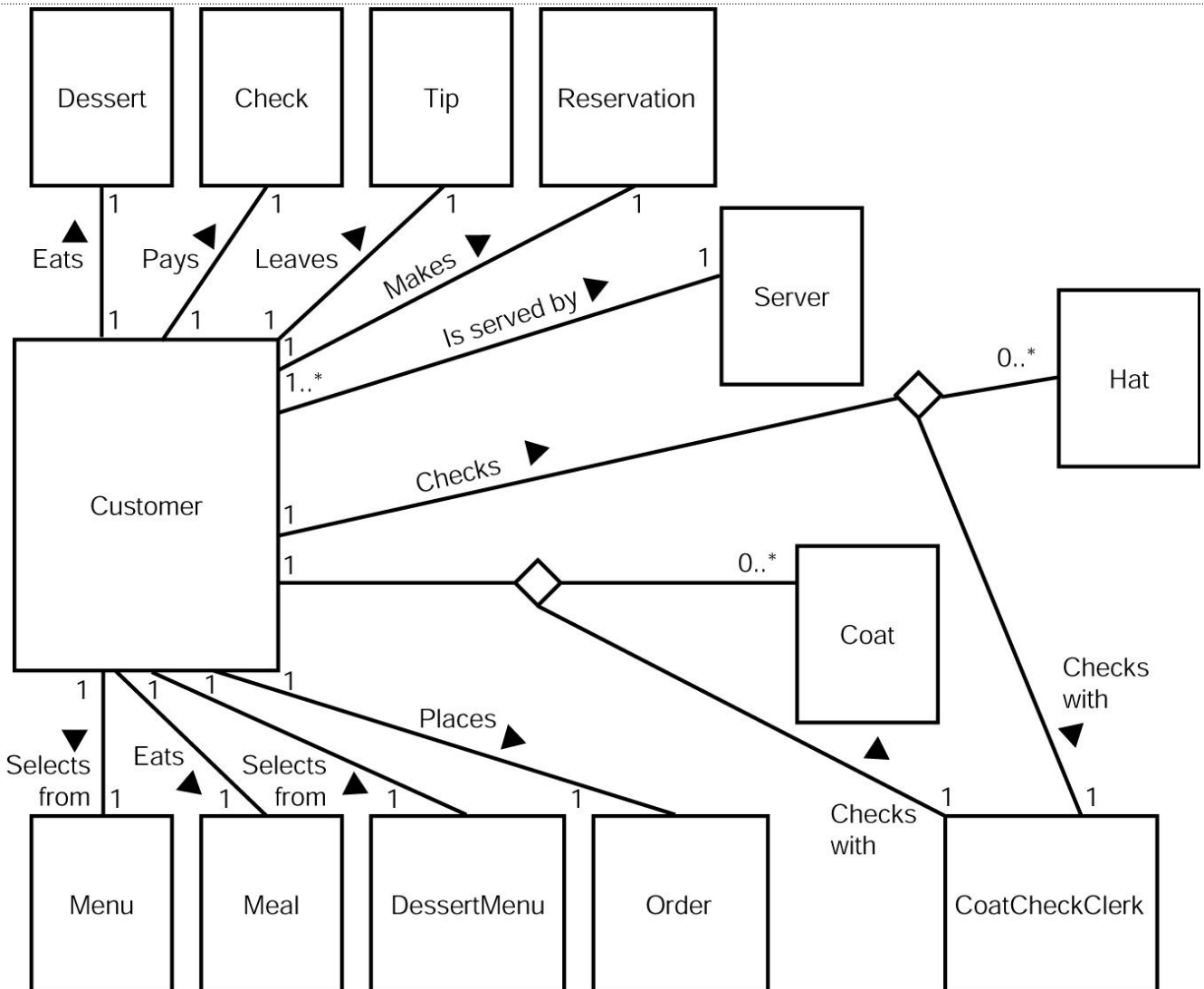
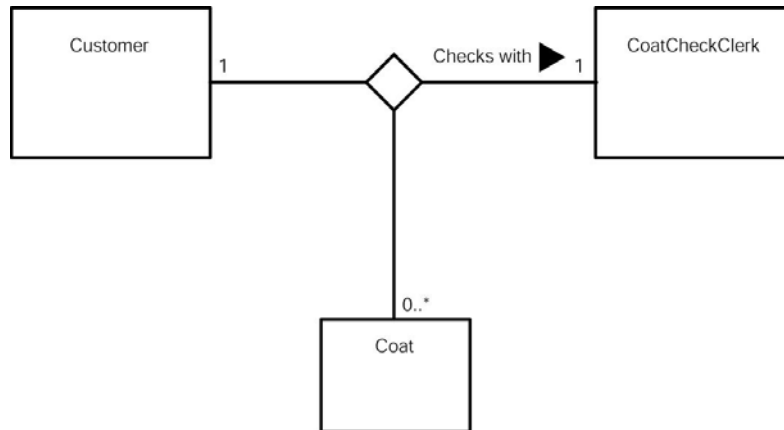
- Customer 클래스에 대한 레이블링된 연관



5-3)3원 연관

- “3원(ternart)”이라는 뜻은 세 개의 클래스가 연관에 포함 된다는 뜻이다. 이것을 나타내려면 연관된 클래스를 마름모꼴로 연결하고 마름모꼴 옆에 연관의 이름을 써준다.

예) 손님이 코트 담당에게 코를 맡긴다. 손님이 코트 담당에게 모자를 맡긴다.

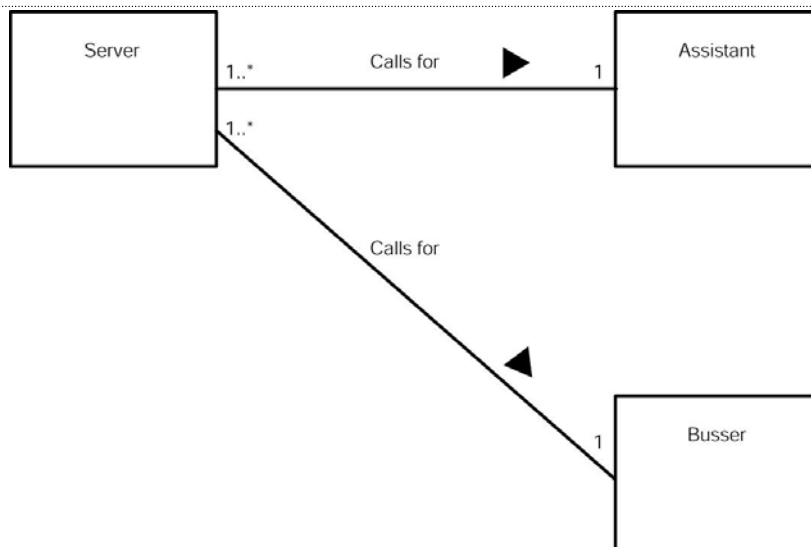


다중성과 레이블링이 모두 추가된 상태의 Customer클래스에 대한 연관이다.

(6)Server클래스에 대한 연관

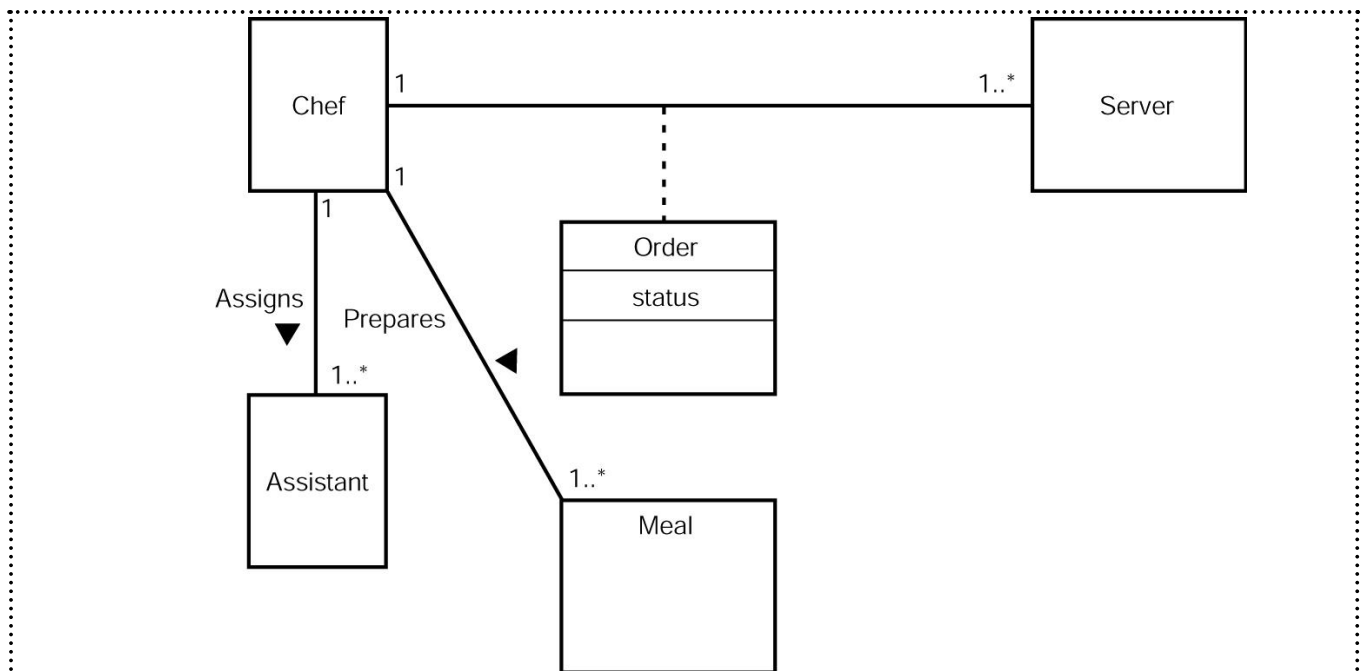
-Server에 관련된 대부분의 연관은 3원으로 표시할 수 있다.

- 서빙담당이 손님으로부터 주문을 받는다(ServertakesanOrderfromaCustomer).
- 서빙담당이 주방장에게 주문을 건넨다(ServertakesanOrdertoaChef).
- 서빙담당이 손님에게 음식을 내어 놓는다(ServerservesaCustomeraMeal).
- 서빙담당이 손님에게 후식을 내어 놓는다(ServerservesaCustomeraDessert).
- 서빙담당이 손님에게 메뉴를 가져온다(ServerbringsaCustomeraMenu).
- 서빙담당이 손님에게 후식 메뉴를 가져온다(ServerbringsaCustomeraDessertMenu).
- 서빙담당이 손님에게 계산서를 가져온다(ServerbringsaCustomeraCheck).
- 서빙담당이 손님으로부터 현금을 받는다(ServercollectsCashfromaCustomer).
- 서빙담당이 손님으로부터 신용카드를 받는다(ServercollectsaCreditCardfromaCustomer).

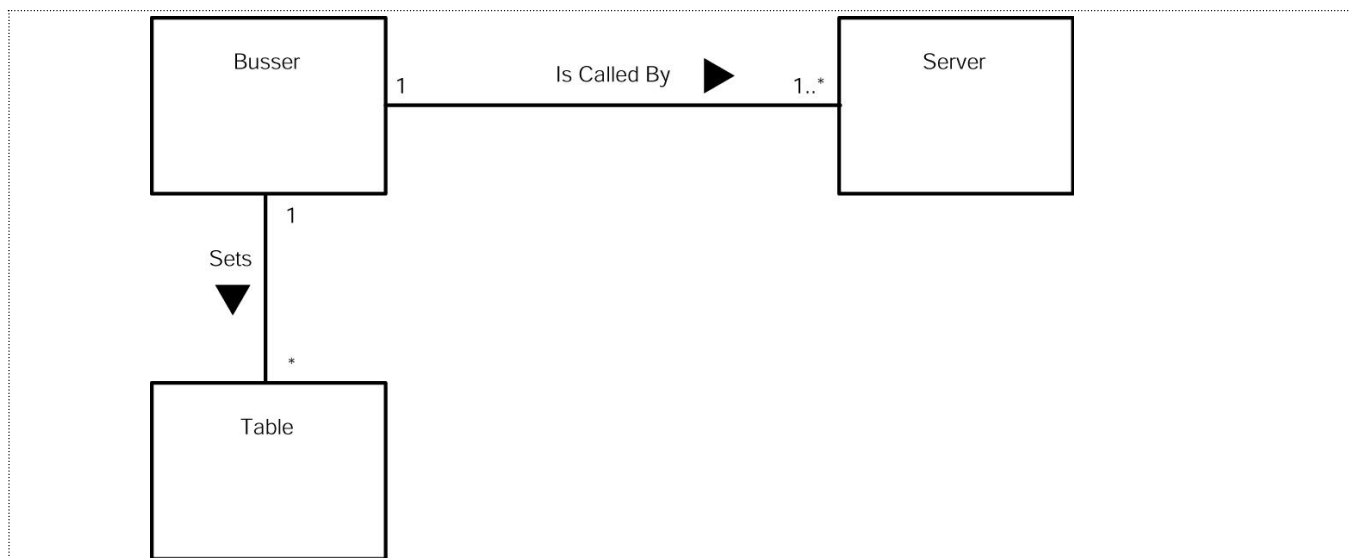


☞ Server에 대한 연관

6-1)Chef클래스에 대한 연관

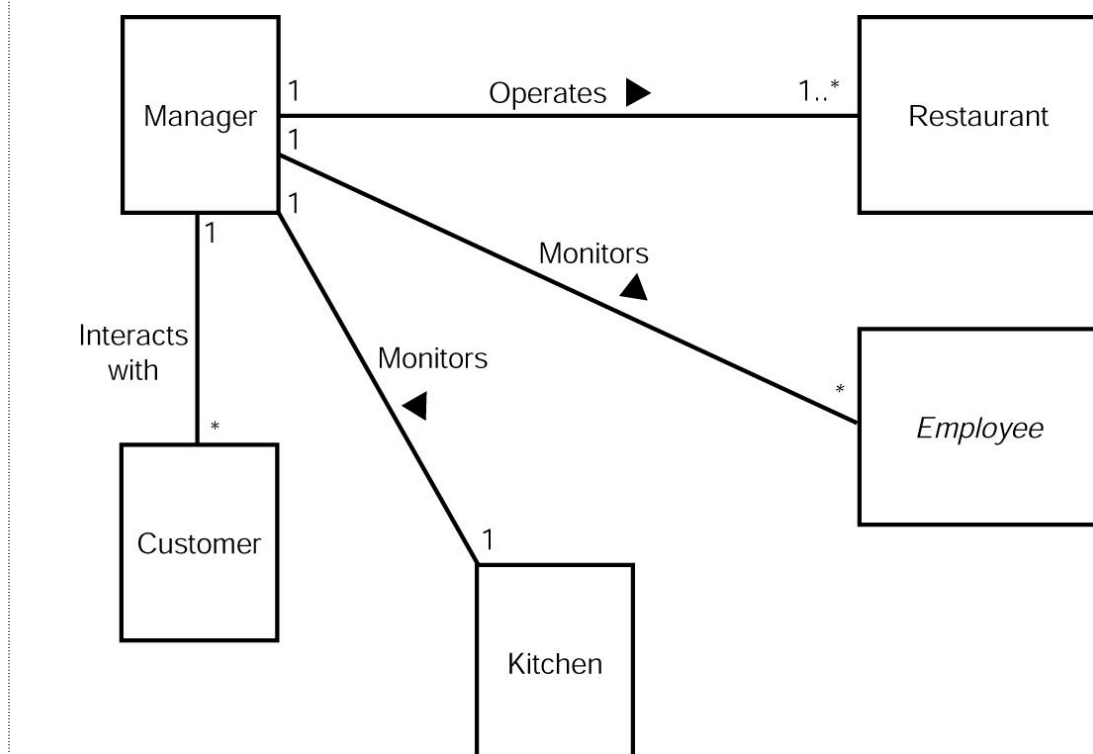


6-2)Busser 클래스에 대한 연관



6-3)Manager클래스에 대한 연관

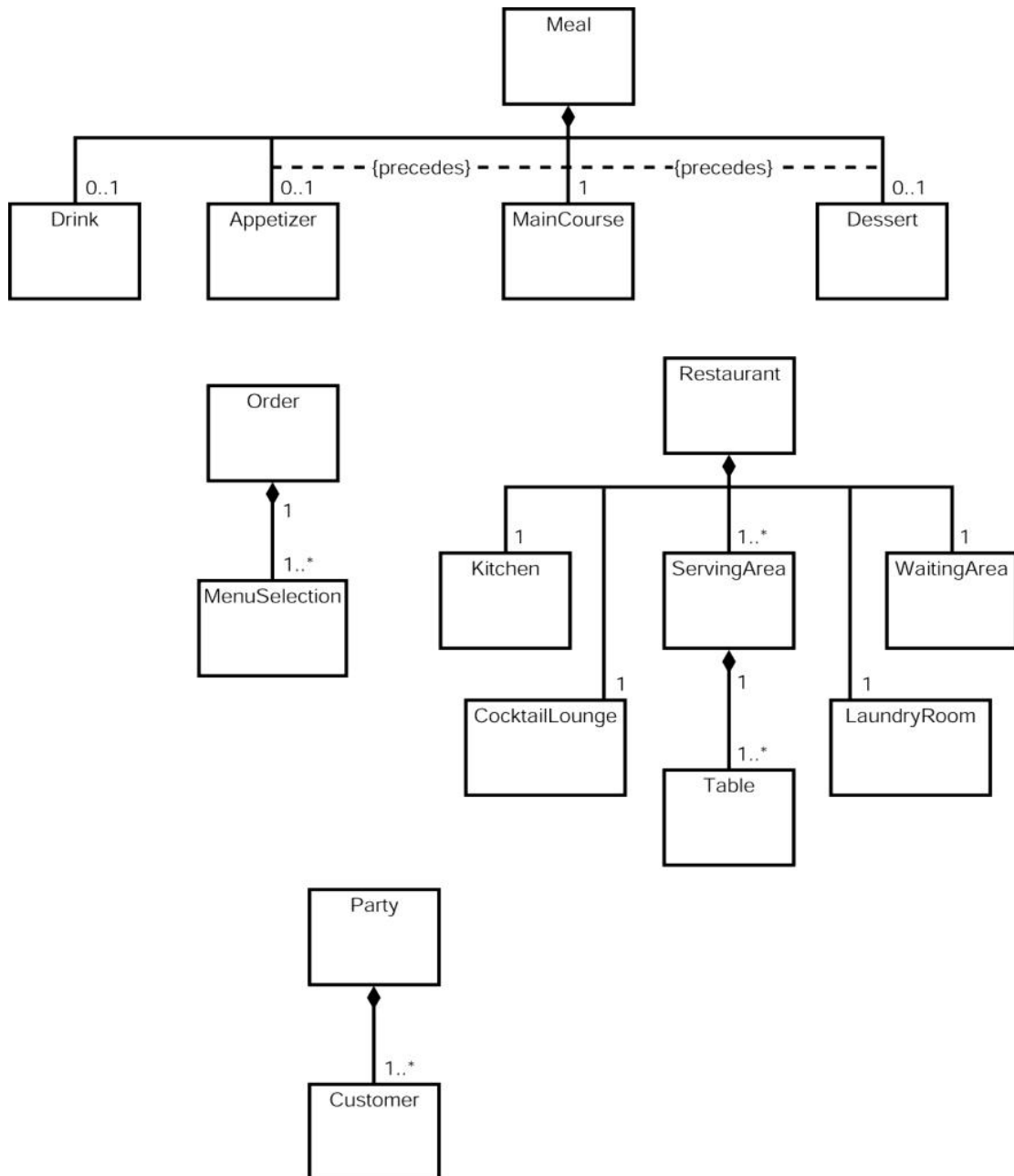
- 관리자가 식당을 운영한다(ManageroperatestheRestaurant).
- 관리자가 종업원을 살펴본다(ManagermonitorstheEmployees).
- 관리자가 주방을 살펴본다(ManagermonitorstheKitchen).
- 관리자가 손님과 이야기를 나눈다(ManagerinteractswiththeCustomer).



(7)집합연관과 복합연관 찾아내기

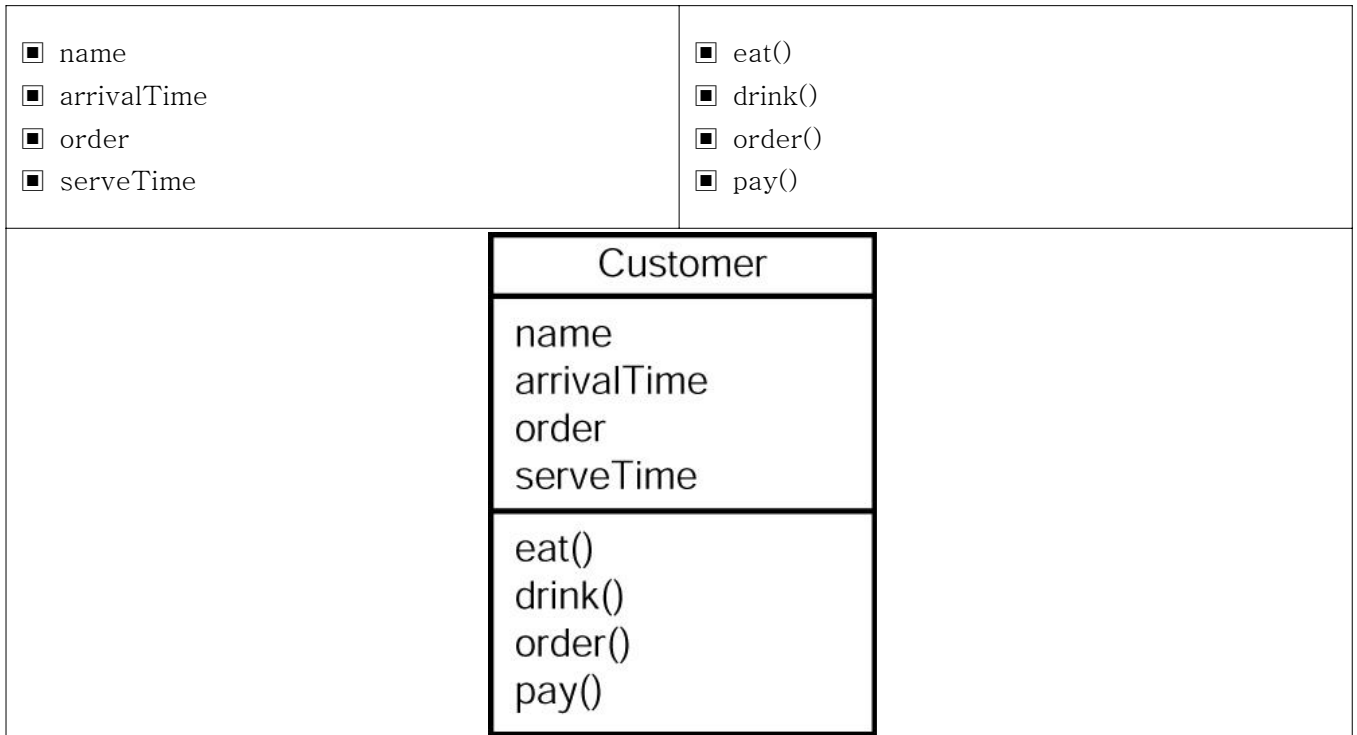
복합 연관을 찾아보면 다음과 같다.

- 주문은 한 개이상의 메뉴 선택으로 구성 되어있다(An Order consists of one or more MenuSelections).
- 식당은주방, 한개이상의서빙구역, 대기구역, 칵테일라운지, 세탁소로 구성되어있다(A Restaurant consists of a Kitchen, one or more Serving Areas, a CocktailLounge, and a LaundryRoom).
- 서빙구역은 한 개이상의 테이블로 구성 되어있다(A Serving Area consists of one or more Tables).
- 단체손님은 한명이상의 손님으로 구성되어있다(A Party consists of one or more Customers).



(8)클래스 다이어그램 속성과 오퍼레이션

- Customer

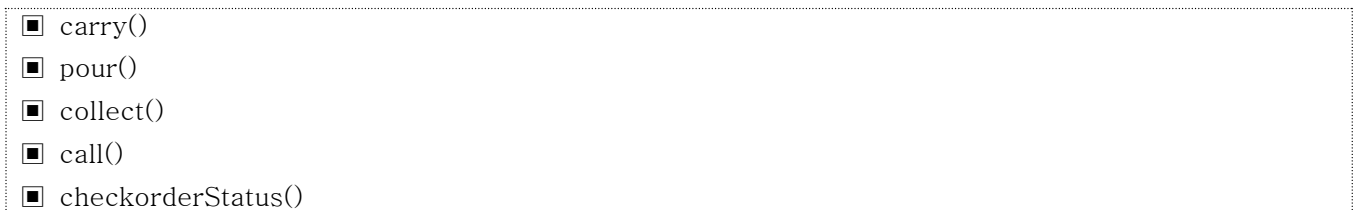


- Employee

- Server, Chef, Manager, Assistant는 모두 추상 클래스 Employee의 자식 클래스다. 따라서 Employee에 추가한 속성은 자식 클래스가 그대로 이어받게 된다.



-Server



-Chef



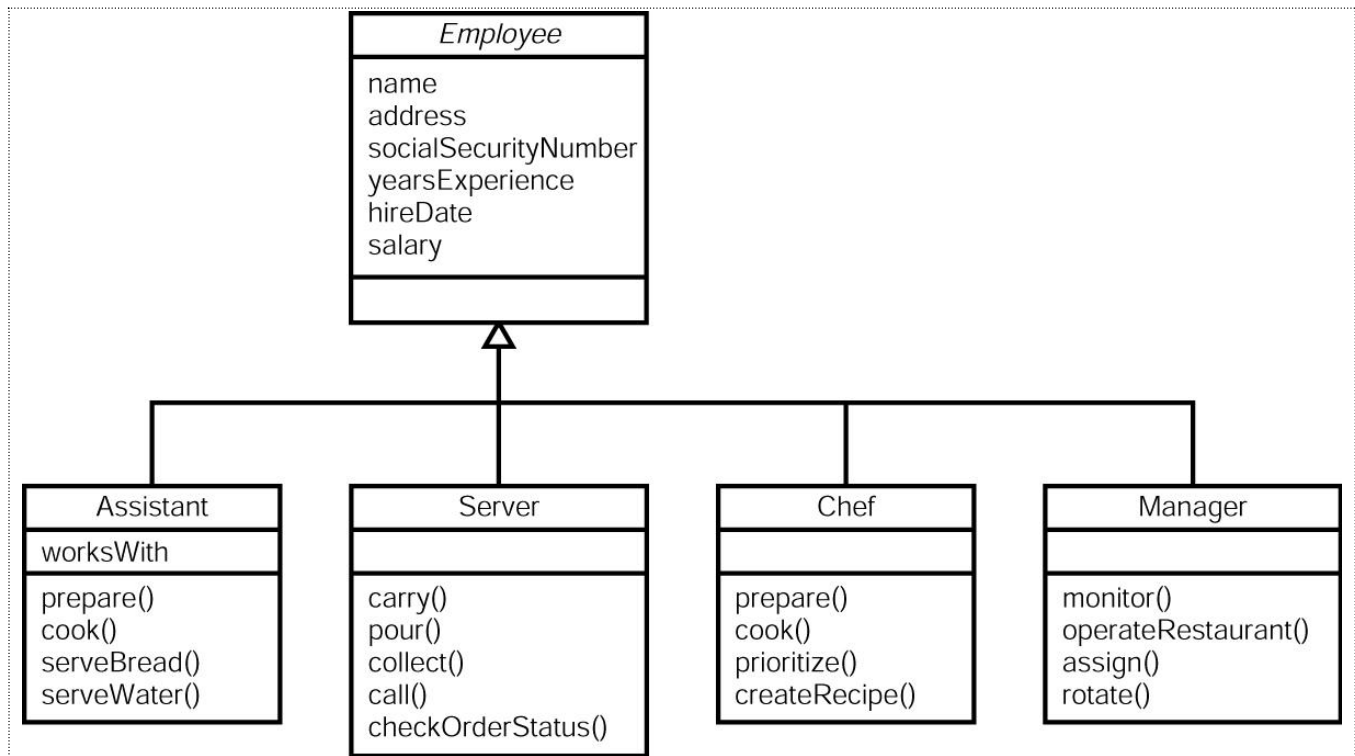
-Assistant

- prepare()
- cook()
- servBread()
- serveWater()

-Manager

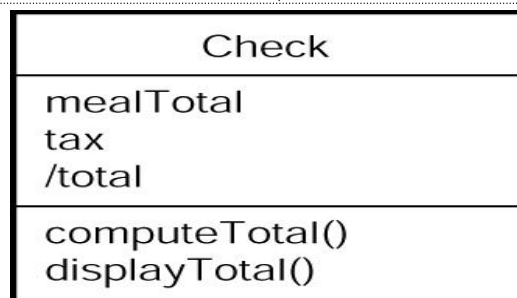
- monitor()
- operateRestaurant()
- addign()
- rotate()

- Employee와 그의 자식 클래스들



-Check 클래스

- | | |
|-------------|------------------|
| ■ mealTotal | ■ computeTotal() |
| ■ tax | ■ displayTotal() |
| ■ total | |



(9)모델 사전

- 모델링에서 사용된 모든 용어를 풀이 해놓은 일람표
- 인터뷰 결과나 업무관계, 그리고 도메인 분석 결과를 정리할 때 만들어 두도록 하자.
- 일관성을 유지할 수 있으며, 모호함을 피해 가는데 도움이 된다.

9-1)다이아그램의 조직화

- 하나의 거대한 다이어그램에 모든 세세한 사항을 그려 넣는 것은 별로 좋은 생각이 아니다.
- 특정 클래스들에 대한 세세한 사항을 그려야 한다면 별도의 다이어그램을 그려두면된다.
- 모델링도구는 대개 이런 것들을 적절하게 연결하여 다이어그램을 조직적으로 만들어준다.

Part 17. 사례연구 - 시스템요구사항 수집