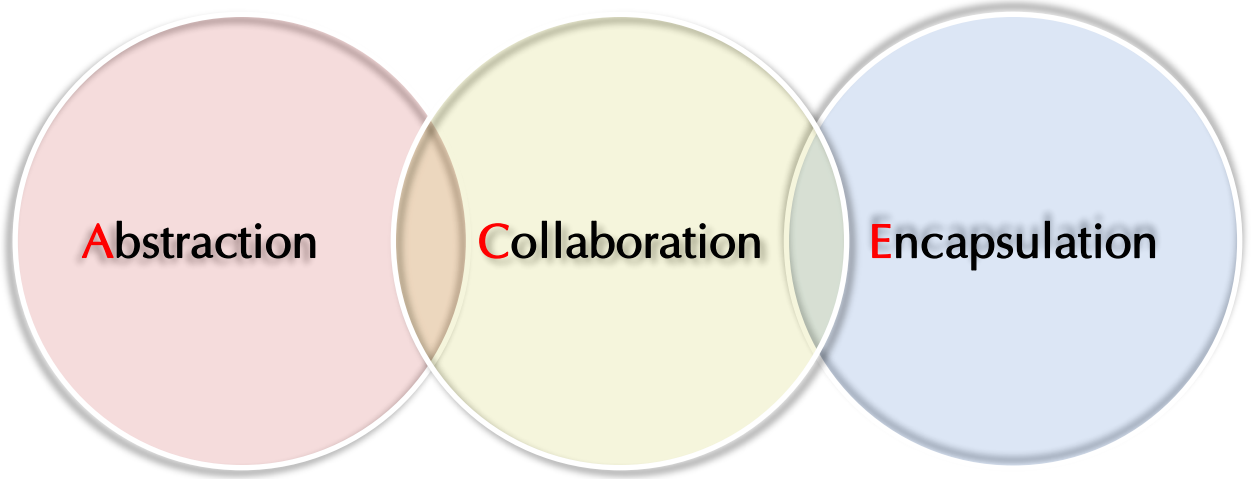
# Object Oriented Programming - Java

# 목차

# 1. 횡단보도(Crosswalk) 이야기

# OOP 특징

✓ 빛의 3원색은 빨강(Red), 녹색(Green), 청색(Blue) 입니다. 줄여서 RGB라고 하죠.
✓ 객체지향 프로그래밍의 3대 요소는 무엇일까? 여러 가지 의견들이 많이 있습니다.
✓ 추상화(abstraction), 상속(inheritance), 캡슐화(encapsulation), 다형성(polymorphism)등을 제시하기도 합니다.
✓ 우리는 추상화(Abstraction), 협업(Collaboration),캡슐화(Encapsulation) 세 가지를 제시합니다.



# OOP 특징 – 추상화의 힘

- ✓ 추상화는 공통적인 특징을 묶어 내고, 의미를 부여하는 사고 활동입니다.
- ✓ 추상화의 힘을 빌리면, 복잡하거나 숫자가 많은 대상을 필요한 만큼 간결하게 표현할 수 있습니다.
- ✓ Abstraction의 범주에 inheritance, polymorphism, operator overloading, operator overriding 등이 있습니다.
- ✓ 추상화 역량은 인간(human being)을 동물과 구별하는 대표적인 역량입니다. OOPL은 추상화를 지원합니다.

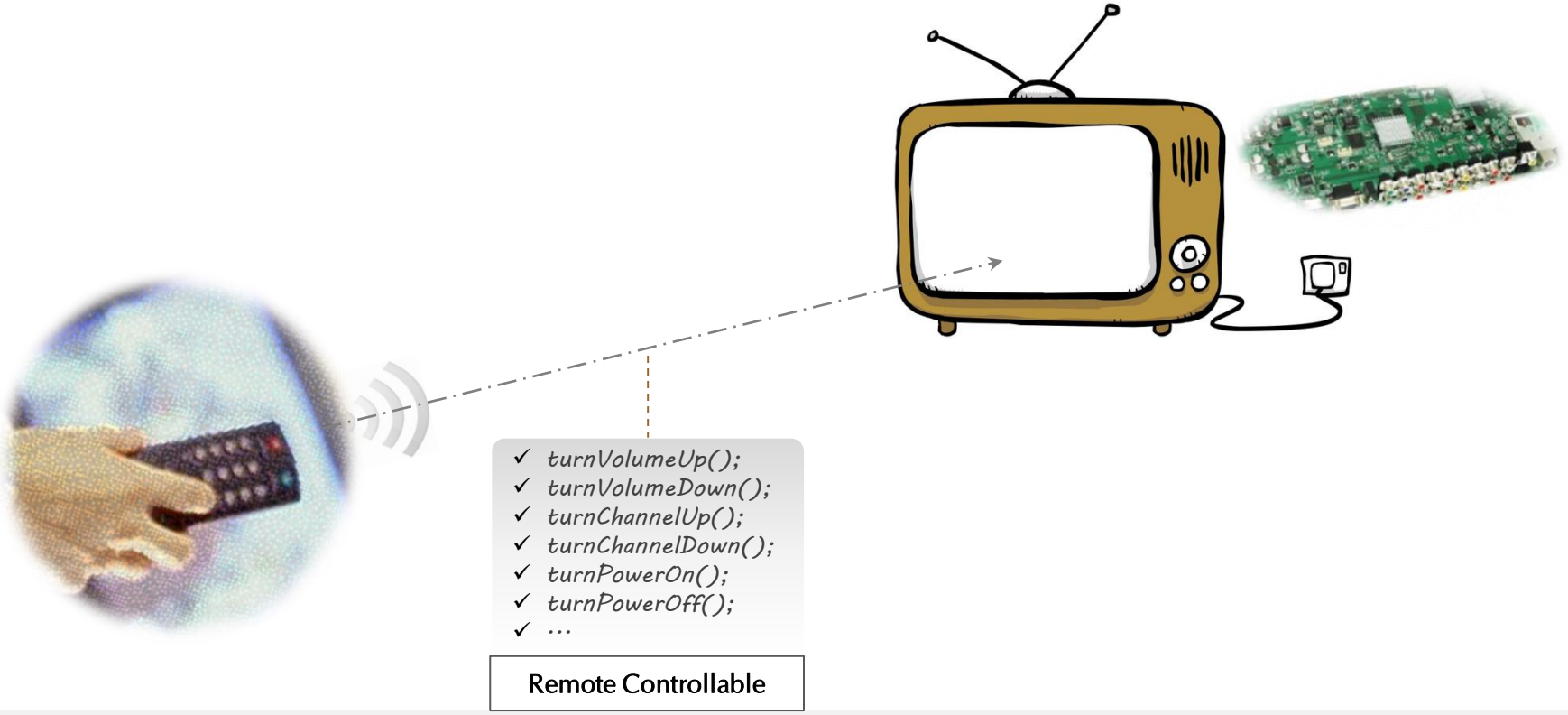
사육사가 {조류 우리}에 들러서 {새}들의 상태를 확인하다가, 홍학 우리에서 {다리를 다친 홍학}을 발견하고는 수의사에게 치료를 부탁했습니다.


*[출처] http://blog.sangwoodiary.com/entry/20100502-at-the-zoo*

# OOP 특징 – 캡슐화의 힘

✓ 어떤 기기가 동작하는 방식을 알아야 사용할 수 있다면 TV 조작조차 간단하지 않을 겁니다.
✓ 하지만, 우리는 리모컨이라는 작은 기기를 가지고, 소리가 커지는 원리를 몰라도, 쉽게 소리를 키울 수 있습니다.
✓ 복잡한 것은 안으로 숨기고, 밖으로는 간단한 인터페이스 만을 내 놓아서, 누구나 쉽게 사용해 주기 때문입니다.
✓ 프로그램에서도 복잡한 구조와 처리는 안쪽에 숨기고(== 캡슐화를 하고), 밖으로는 간단한 인터페이스를 내 놓아야 합니다.

Abstraction은 효율적인 캡슐화를 하는 방법을 제공합니다. 인터페이스를 실체화(realization)하는 관계는 추상화 계층을 이용하는 방식입니다.



# OOP 특징 – 협업(collaboration)

✓ OOP는 객체들의 상호작용으로 이루어집니다.

✓ 객체들의 협업을 위해서는 각각의 참여객체의 역할과 책임을 이해해야 합니다.

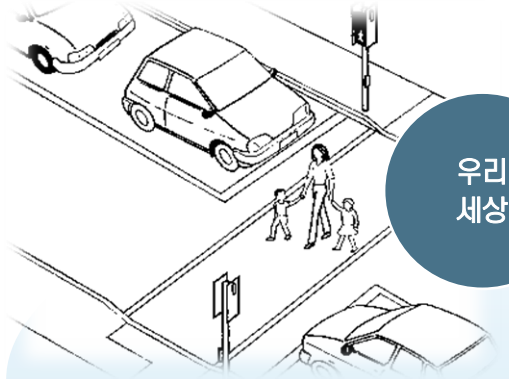
✓ OOP를 설계하는 과정은 결국 참여 객체들에게 어떤 역할과 책임을 주어야 할 지에 대한 고민의 연속입니다.

# 시스템 개발 – 연속적인 변환(transformation)

✓ 우리가 살고 있는 세상에 자주 만나는 불편함을 해소하기 위해 [SW intensive] 시스템을 개발합니다.

✓ 우리의 불편을 해결해 주는 시스템은 우리가 사용하는 표현으로부터 출발하여 자그마한 시스템 세계를 구성합니다.

✓ 자연어 → 모델링 언어 → 프로그래밍 언어를 거치면서 시스템 세계가 요구하는 특성을 점진적으로 반영합니다.



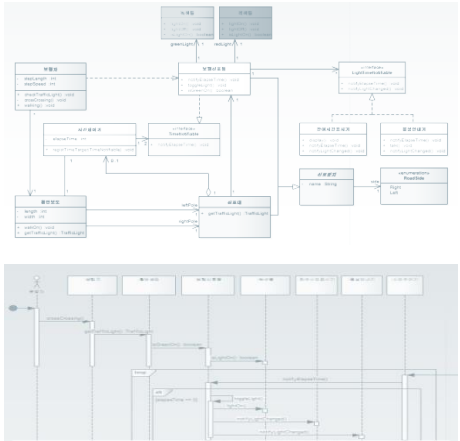
우리 세상

언어에서 언어로의 변환(Transformation)

시스템 세계

1. 행인이 횡단보도로 다가가며 보행자신호등을 확인한다. 적색등이 켜져있다. 녹색등을 기다리며 멈춰선다.
2. 시간이 흘러 녹색등이 켜지고, 또 시간이 흘러 적색등이 켜진다. 녹색등과 적색등이 교대로 켜진다.
3. 녹색등이 켜지면 음성 안내기는 "녹색등입니다. 건너가십시오."라는 안내를 하고 "뚜르르, 뚜르르,..."하고 알림음을 낸다. 5초가 남으면 "위험합니다. 건너지 마십시오."라는 안내를 하고, 0초가 남으면 알림음을 멈춘다.
4. 녹색등이 켜지면 남은시간표시기에 숫자가 20부터 시작하여 1초 단위로 줄어든다. 적색등이 켜지면 남은 시간 표시기는 꺼진다.
5. 행인은 횡단보도를 건너간다. 시간이 부족하면 달린다.

요구사항 – 자연언어



모델링 – UML

```
import com.nextree.omw.crossroad.Logger;

public class TrafficLight extends TrafficDevice {

    private GreenLight greenLight;
    private RedLight redLight;

    public TrafficLight(String name, RoadSide side) {
        super(name);
        super.setRoadSide(side);
        this.greenLight = new GreenLight();
        this.redLight = new RedLight();
        this.greenLight.off();
        this.redLight.on();
    }

    protected void toggleLight() {
        // 지정시간 뒤에
        if (isGreenOn()) {
            setRedOn();
        } else {
            setGreenOn();
        }
    }

    protected LightColor getLightColor() {
        if (isGreenOn()) {
            return LightColor.Green;
        } else if (isRedOn()) {
            return LightColor.Red;
        }
        throw new RuntimeException("신호등의 색상이 없습니다...");
    }
```

프로그램 – 프로그래밍언어

```
F9F9F9F9 F9F9F9F9 F9F9F9F9
E29189F9 F9F9F9F9 F9F9F9F9
F9F9F9F9 F9F9F9F9 F9F9F9F9
F9F9F9F9 F9F9F9F9 F9F9F9F9
F9F9F9F9 F9F9F9F9 F9F9F9F9
F9F9F9F9 F9F9F902 00F8F020
FCFCFCFC FCFCFCFC FCFCFCFC
FCFCFCFC FCFCFCFC FCFCFCFC
FCFCFCFC FCFCFCFC FCFCFCFC
FCFCFCFC FCFCFCFC FCFCFCFC
01F035EA 7E7E7E7E 7E7E7E7E
7E7E7E7E 7E7E7E7E 7E7E7E7E
7E7E7E7E 7E7E7E7E 7E7E7E7E
7E7E7E7E 7E7E7E7E 7E7E7E7E
7E7E7E7E 7E7E7E7E 008080F8
3F3F3F3F 3F3F3F3F 3F3F3F3F
3F3F3F3F 3F3F3F3F 3F3F3F3F
3F3F3F3F 3F3F3F3F 3F3F3F3F
```

실행코드 – 기계어

# [SW intensive] 시스템 개발

✓ “SW 개발”이라는 거대한 활동 이름 아래, 개발팀은 저마다의 방식으로 개발을 진행하고 있습니다.

*2014년 ~*

MSA (마이크로 서비스 아키텍처)

*2008년 ~*

SOA (서비스 지향 아키텍처)

*2002년 ~*

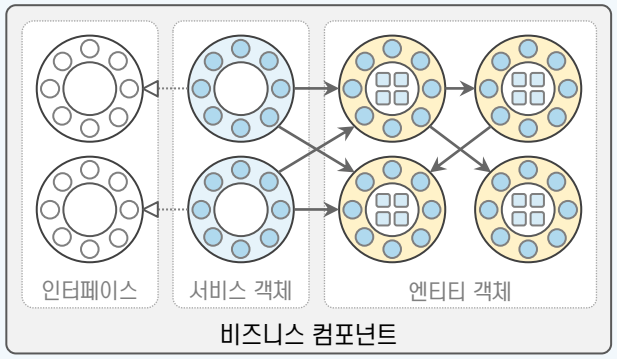
CBD (컴포넌트 기반 개발)

*1995년 ~*

OOAD (객체지향 분석 설계)

# [SW intensive] 시스템 개발

✓ “SW 개발”이라는 거대한 활동 이름 아래, 개발팀은 저마다의 방식으로 개발을 진행하고 있습니다.




**질서정연한** 객체들의 집합으로 구성한 비즈니스 컴포넌트

*2014년 ~*

MSA (마이크로 서비스 아키텍처)

*2008년 ~*

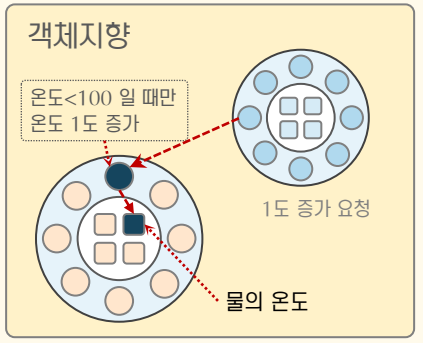
SOA (서비스 지향 아키텍처)

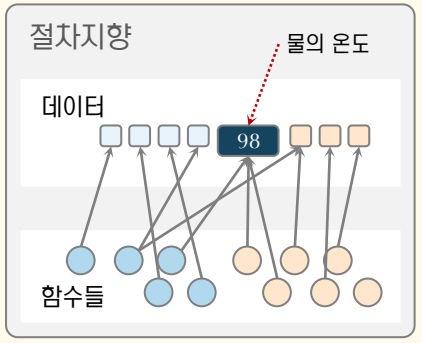
*2002년 ~*

CBD (컴포넌트 기반 개발)

*1995년 ~*

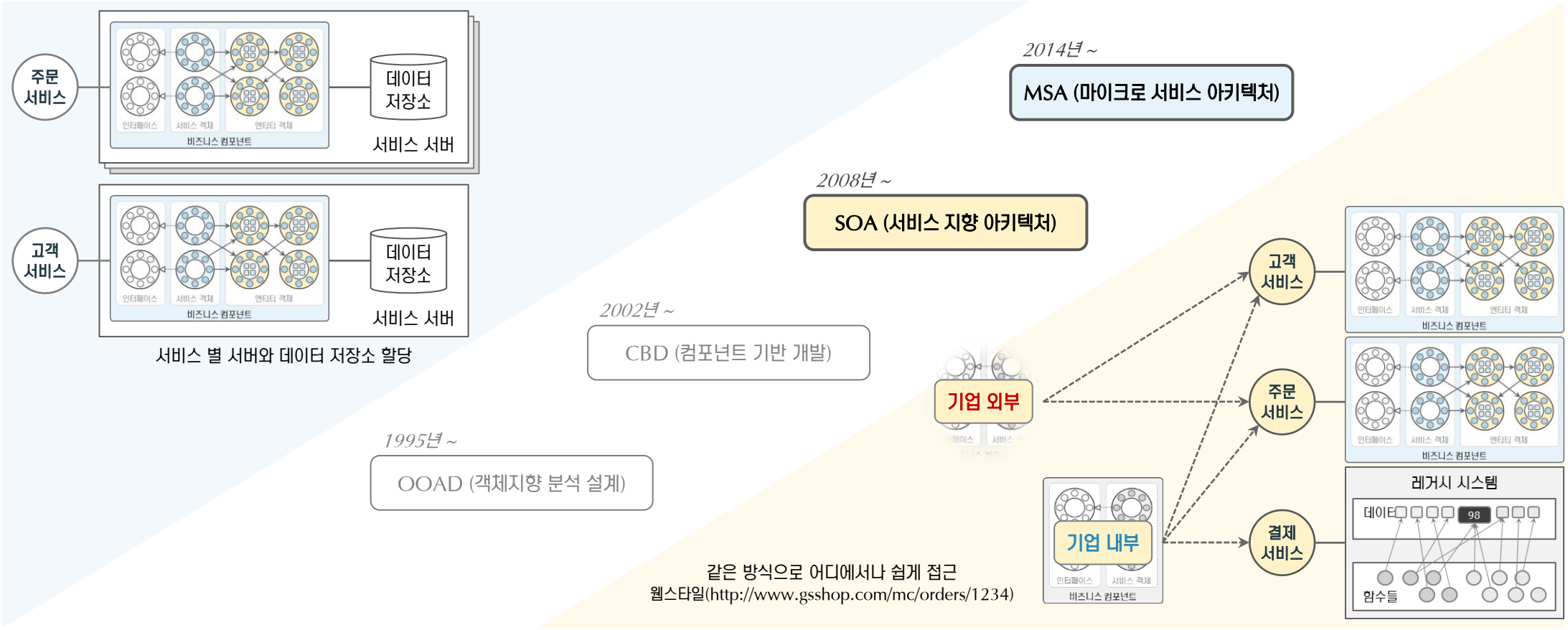
OOAD (객체지향 분석 설계)



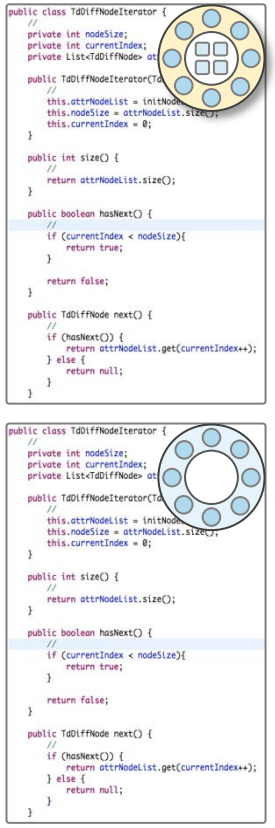
# [SW intensive] 시스템 개발

✓ “SW 개발”이라는 거대한 활동 이름 아래, 개발팀은 저마다의 방식으로 개발을 진행하고 있습니다.

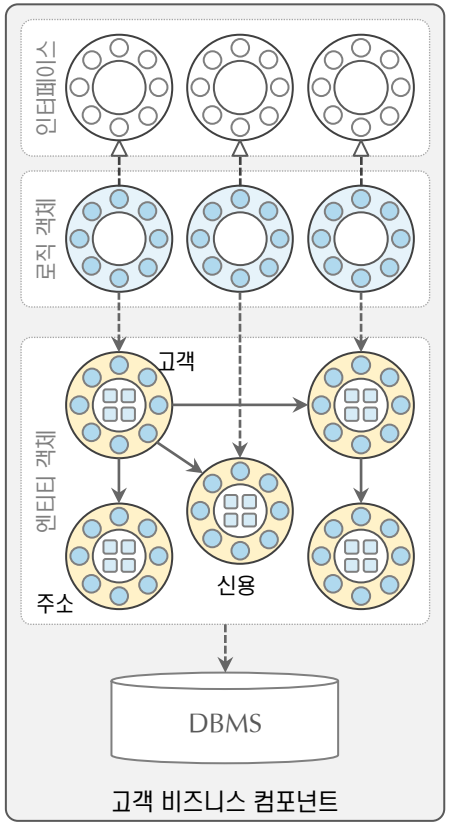


# [SW intensive] 시스템 개발

✓ “SW 개발”이라는 거대한 활동 이름 아래, 개발팀은 저마다의 방식으로 개발을 진행하고 있습니다.

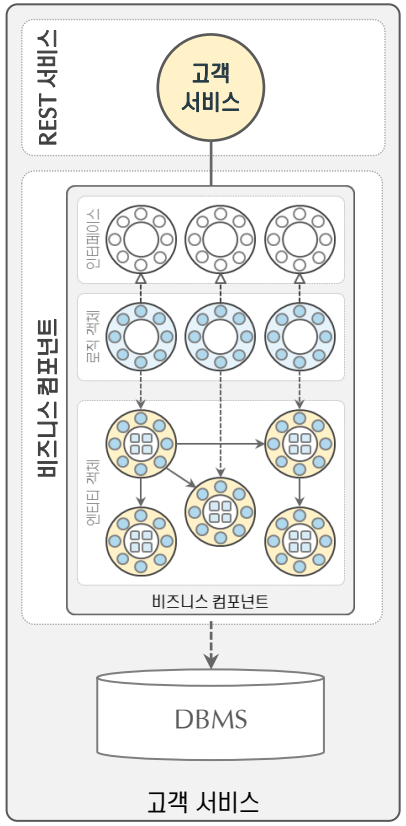


**질서정연한** 소스코드



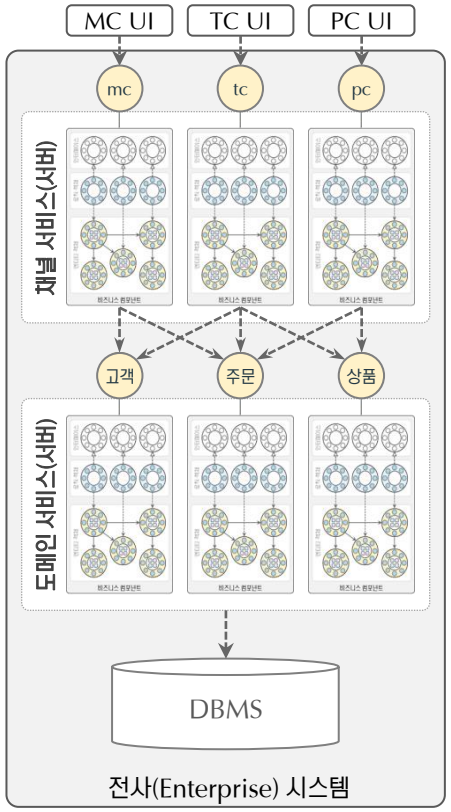

고객 비즈니스 컴포넌트

**질서정연한** 객체 집합으로 구성한 컴포넌트



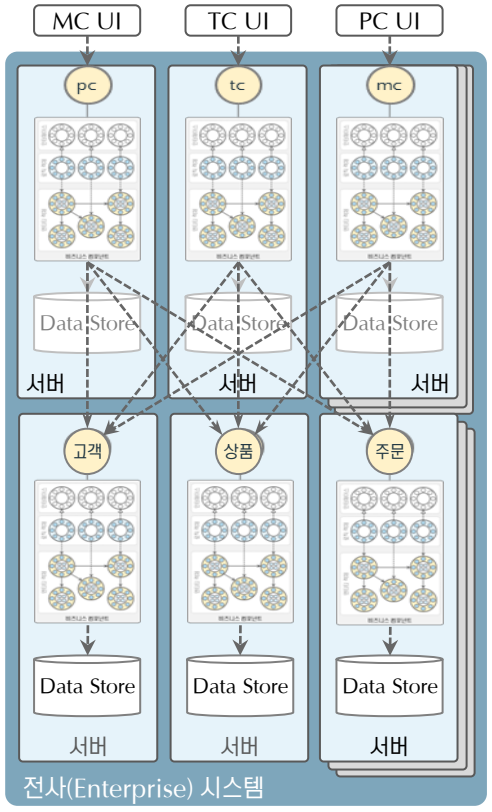

고객 서비스

**접근이 쉬운** REST 기반의 서비스




전사(Enterprise) 시스템

채널/도메인 서비스 기반 전사 시스템

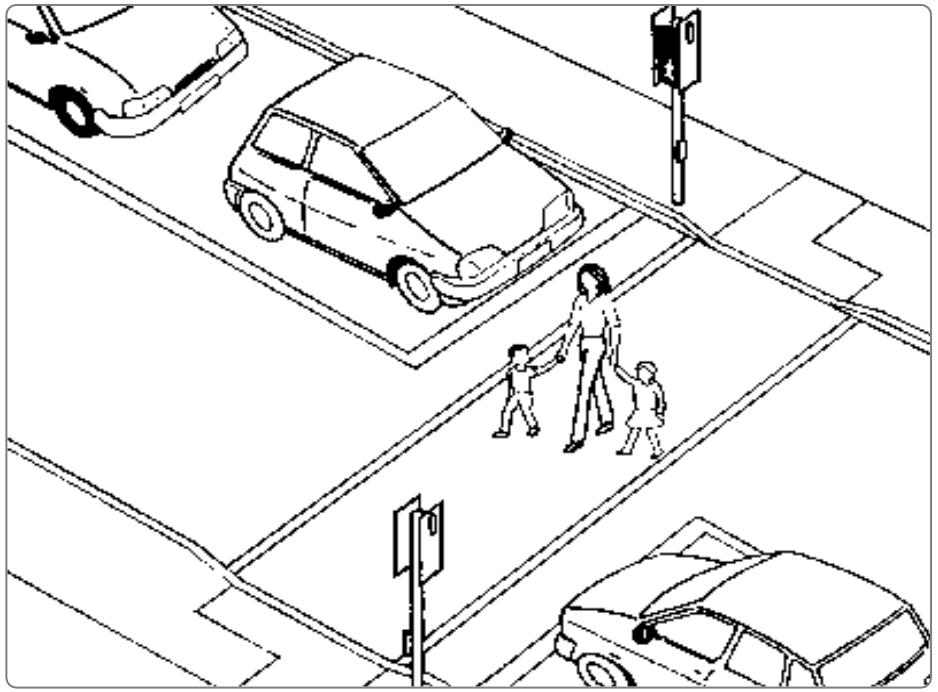



마이크로 서비스 기반 엔터프라이즈 시스템

# 1. 횡단보도(Crosswalk) 이야기

# 1.1 횡단보도 – 이야기(story)

- ✓ 생활 속에 자주 볼 수 있는 횡단보도 상황을  UML 모델로 표현하고 Java 언어로 프로그램을 작성합니다.
- ✓ Java 언어로 구현함으로써, 자연어(natural language), 모델링 언어, 프로그래밍 언어 간의 차이를 이해합니다.
- ✓ UML로 표현하는 모델을 검증하기 위한 수단으로 Java 프로그램을 활용합니다.
- ✓ 객체 지향의 핵심 개념인 추상화(abstraction)보다는  모델링의 핵심인 협업(collaboration)을 이해합니다.

| 현장 모습 | 현장 상황 이야기 |
| --- | --- |
|  | 한 여성이 횡단보도를 건너려고 횡단보도로 걸어간다. 신호등을 확인하고는 멈춰선다. 적색등에 불이 켜져 있다. 몇 초가 지난 후 녹색등에 불이 들어온다. 곧이어 음성 안내기의 안내소리가 들린다. “녹색등입니다. 건너가십시오.” 이어 알림음이 계속된다. “뚜루르, 뚜루르,…” 이 여성은 빠른 걸음으로 걷기 시작한다. 시선은 맞은편 신호등 바로 옆에 있는 LCD 남은시간표시기에 고정되어 있다. 20, 19, 18,… 1초 단위로 숫자가 줄어든다. 어느덧 남은시간표시기는 숫자 “5”를 보여준다. 곧바로 음성 안내기의 안내소리가 들려왔다. “위험합니다. 건너지 마십시오.” 이 여성은 급한 마음에 달리기 시작한다. 횡단보도를 막 건넜을 때, 신호기의 적색등에 불이 들어온다. LCD 남은시간표시기의 불이 꺼지며, 음성 안내기가 외친다. “적색등입니다. 건너지 마십시오.” 이어 계속되던 알림음도 멎었다. |

# 1.2 횡단보도 – 시나리오 조각

- ✓ 횡단보도 상황을 설명하는 긴 이야기는 우리가 현장에서 고객의 요구 사항을 서술한 것과 같습니다.
- ✓ 모든 큰 일은 작은 일로 나누어 처리합니다. 이 긴 이야기를 모델링과 구현하려 작은 시나리오 조각으로 나눕니다.
- ✓ 시나리오 조각은 특정 역할을 중심으로 하나의 완전한 의미를 가질 수 있도록 재구성합니다.
- ✓ 각 시나리오 조각은 객체 모델링 단위이며, 정적인 측면과 동적인 측면을 동시에 모델링합니다.

| 현장 상황 이야기 | 분석을 위한 시나리오 조각 |
|---|---|
| 한 여성이 횡단보도를 건너려고 횡단보도로 걸어간다. 신호등을 확인하고는 멈춰선다. 적색등에 불이 켜져 있다. 몇 초가 지난 후 녹색등에 불이 들어온다. 곧이어 음성 안내기의 안내소리가 들린다. "녹색등입니다. 건너가십시오." 이어 알림음이 계속된다. "뚜루르, 뚜루르,..." 이 여성은 빠른 걸음으로 걷기 시작한다. 시선은 맞은편 신호등 바로 옆에 있는 LCD 남은시간표시기에 고정되어 있다. 20, 19, 18,... 1초 단위로 숫자가 줄어든다. 어느덧 남은시간표시기는 숫자 "5"를 보여준다. 곧바로 음성 안내기의 안내소리가 들려왔다. "위험합니다. 건너지 마십시오." 이 여성은 급한 마음에 달리기 시작한다. 횡단보도를 막 건넜을 때, 신호기의 적색등에 불이 들어온다. LCD 남은시간표시기의 불이 꺼지며, 음성 안내기가 외친다. "적색등입니다. 건너지 마십시오." 이어 계속되던 알림음도 멎었다. | 1. 보행자가 횡단보도로 다가가며 보행자신호등을 확인한다. 적색등이 켜져있다. 녹색등을 기다리며 멈춰선다.<br>2. 시간이 흘러 녹색등이 켜지고, 또 시간이 흘러 적색등이 켜진다. 녹색등과 적색등이 교대로 켜진다. 녹색등으로 바뀌면 보행자는 건너 간다.<br>3. 녹색등이 켜지면 음성 안내기는 "녹색등입니다. 건나가십시오."라는 안내를 하고 "뚜르르, 뚜르르,..."하고 알림음을 낸다. 5초가 남으면 "위험합니다. 건너지 마십시오."라는 안내를 하고, 0초가 남으면 알림음을 멈춘다.<br>4. 녹색등이 켜지면 남은시간표시기에 숫자가 20부터 시작하여 1초 단위로 줄어든다. 적색등이 켜지면 남은 시간 표시기는 꺼진다.<br>5. 보행자는 횡단보도를 건너간다. |

# 1.3 실습 1-1 단계 – 시나리오

- ✓ 첫 시나리오 조각은 보행자가 횡단보도를 건너려고 신호등을 확인하는 내용입니다.
- ✓ 이 시나리오를 무대 위에서 연출해야 하는 연출자(director) 입장에서 생각해 봅니다.
- ✓ 이 연극을 무대에 올리기 위해서는 대본(script)이 필요합니다. 한 줄짜리 상황묘사를 바탕으로 대본을 작성합니다.
- ✓ 이런 대본을 작성하기 위해서는 상황을 충분히 이해하고 있어야 합니다.

**협업 이해 →** 도메인 객체 모델링 → 구현

1. 보행자가 횡단보도로 다가가며 보행자신호등을 확인한다. 적색등이 켜져있다. 녹색등을 기다리며 멈춰선다.

(막이 열리면 11번가의 모습이 보이고, 무대 한 가운데 횡단보도가 있다. 보행자인 민수가 등장한다.)

연출가 : 민수씨, 이 횡단보도(이름은 cross11)를 건너가세요.

민수 : 아, 이 횡단보도요? 알겠습니다. 먼저 보행자 신호등을 확인해야겠는데요. 횡단보도, 보행자 신호등 좀 주세요.

횡단보도 : (보행자 신호등을 건네며) 여기 있어요.

민수: 보행자신호등, 지금 녹색등이 켜져 있나요?

보행자신호등: 아니오, 지금은 적색등이 켜져 있습니다.

민수: 아, 그렇군요. 기다려야겠군요(멈춰 선다.)

# 1.3 실습 1-1 단계 –UML을 그려봅시다.

✓ **시나리오에 등장하는 등장인물-참여객체를 식별해봅니다.**

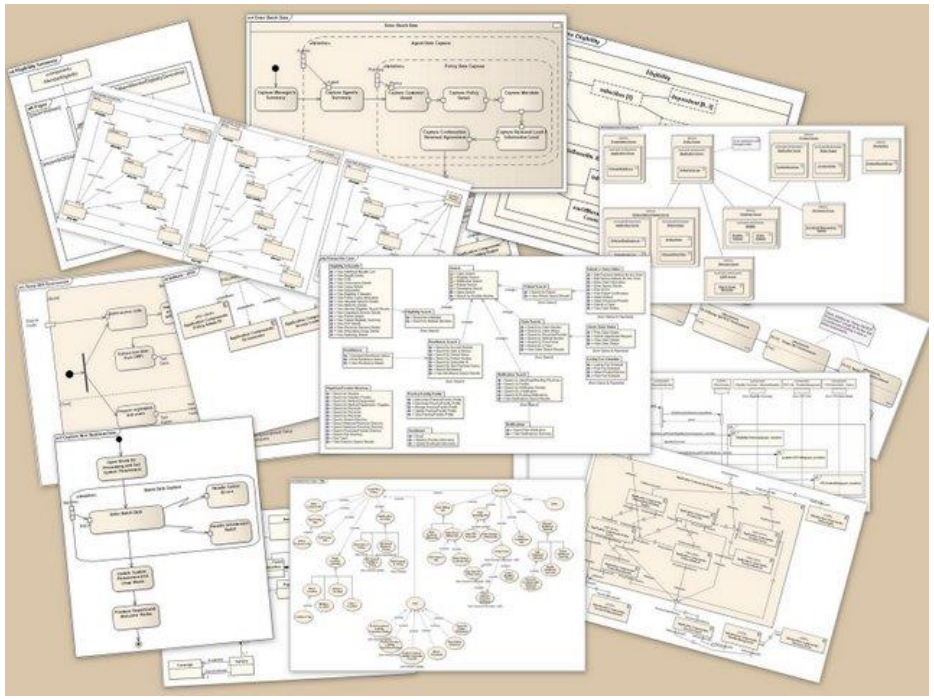
- 어떤 객체들이 식별될 수 있을까요?

✓ **역할과 책임을 나눠봅니다.**

- 각 객체는 어떤 행위를 할 수 있을까요?

협업 이해 → **도메인 객체 모델링** → 구현

1. 보행자가 횡단보도로 다가가며 보행자신호등을 확인한다. 적색등이 켜져있다. 녹색등을 기다리며 멈춰선다.

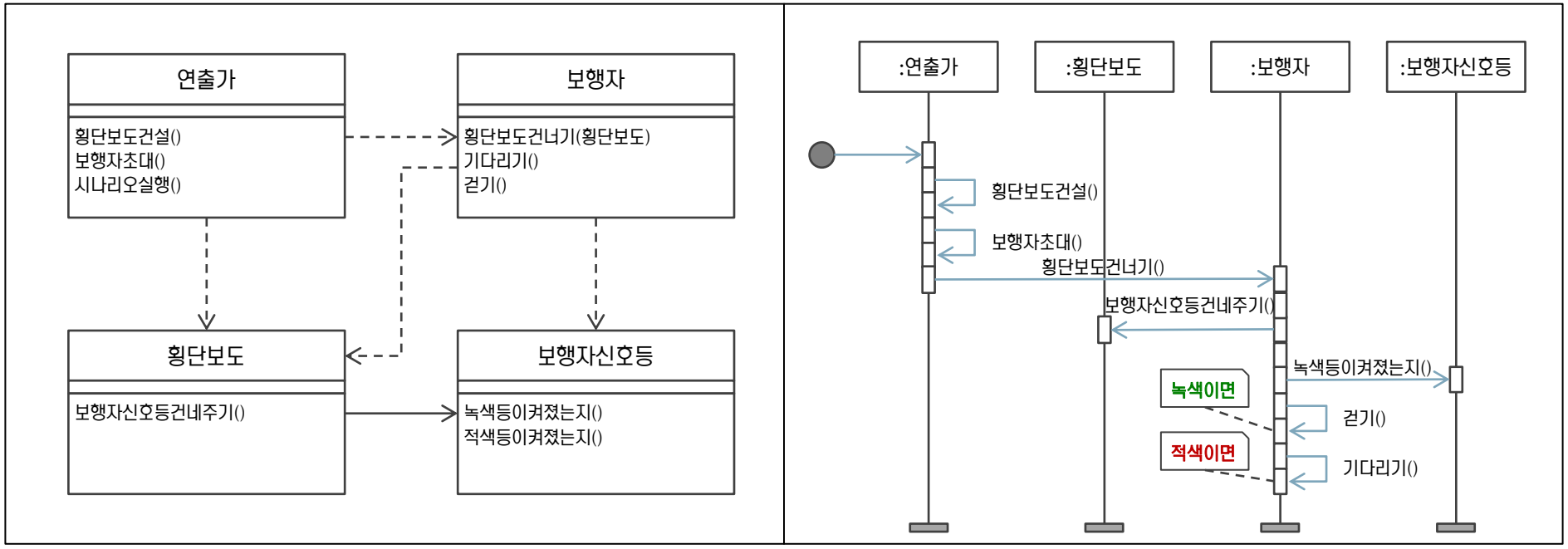


# 1.3 실습 1-1 단계 – UML 모델

- ✓ 시나리오에 참여하는 객체가 무엇인지를 찾는 것을 정적(static) 모델링 또는 구조적인 모델링이라고 합니다.
- ✓ 처리 흐름이나 객체/컴포넌트의 협업 등을 찾고자 정의 하는 것을 동적(dynamic) 모델링 또는 행위 모델링이라고 합니다.
- ✓ 횡단보도, [보행자]신호등, 적색등, 녹색등과 같은 클래스들을 찾을 수 있습니다.
- ✓ 각 객체들이 시나리오에 참여할 때, 어떤 역할과 책임을 가지고 참여하는 지 충분히 생각해 봅니다.

협업 이해 →**도메인 객체 모델링** → 구현

1. 보행자가 횡단보도로 다가가며 보행자신호등을 확인한다. 적색등이 켜져있다. 녹색등을 기다리며 멈춰선다.



# 1.4 실습 1-2 단계 - 시나리오

✓ 첫번째 시나리오를 좀 더 다듬고 사실적으로 표현할 수 있습니다.
✓ 이 가운데 보행자 신호등을 양쪽 모두 있다는 가정을 할 수 있고 이 내용을 시나리오에 추가하였습니다.
✓ 그리고 녹색등, 적색등의 경우 시나리오에 자주 등장되므로 속성이 아닌 객체로 식별될 수 있습니다.
✓ 또한 점점 참여하는 객체가 늘어남에 따라 이를 해결하는 방안도 필요합니다.

**협업 이해 →**도메인 객체 모델링 → 구현

1. 보행자가 횡단보도로 다가가며 보행자신호등을 확인한다. 적색등이 켜져있다. 녹색등을 기다리며 멈춰선다.

(막이 열리면 11번가의 모습이 보이고, 무대 한 가운데 횡단보도가 있다. 보행자인 민수가 횡단보도 왼쪽에서 등장한다.)

연출가 : 민수씨, 11번가 첫 번째 횡단보도(이름은 cross11)를 왼쪽에서 건너세요.
민수 : 아, 이 횡단보도요? 알겠습니다. 먼저 보행자 신호등을 확인해야겠는데요. 횡단보도, 보행자 신호등 좀 주세요.
횡단보도 : (오른쪽 보행자 신호등을 건네며) 여기 있어요.
민수: 보행자신호등, 지금 녹색등이 켜져 있나요?
보행자신호등: 잠시만요. 확인 좀 하구요. 녹색등, 너 지금 켜져 있니?
녹색등: 아뇨.
보행자신호등: (민수에게) 녹색등은 꺼져있네요.
민수: 잠시 기다려야겠네요.

# 1.3 실습 1-2 단계 –UML을 그려봅시다.

- ✓ 추가된 상황을 식별해봅니다.
- ✓ 참여객체의 속성도 역할을 가지면 객체로 승격할 수 있습니다.
- ✓ 자, 두번째 UML을 그려봅니다.

**협업 이해 →** 도메인 객체 모델링 → 구현

1. 보행자가 횡단보도로 다가가며 보행자신호등을 확인한다. 적색등이 켜져있다. 녹색등을 기다리며 멈춰선다.

(막이 열리면 11번가의 모습이 보이고, 무대 한 가운데 횡단보도가 있다. 보행자인 민수가 횡단보도 왼쪽에서 등장한다.)

연출가 : 민수씨, 11번가 첫 번째 횡단보도(이름은 cross11)를 왼쪽에서 건너세요.

민수 : 아, 이 횡단보도요? 알겠습니다. 먼저 보행자 신호등을 확인해야겠는데요. 횡단보도, 보행자 신호등 좀 주세요.

횡단보도 : (오른쪽 보행자 신호등을 건네며) 여기 있어요.

민수: 보행자신호등, 지금 녹색등이 켜져 있나요?

보행자신호등: 잠시만요. 확인 좀 하구요. 녹색등, 너 지금 켜져 있니?

녹색등: 아뇨.

보행자신호등: (민수에게) 녹색등은 꺼져있네요.

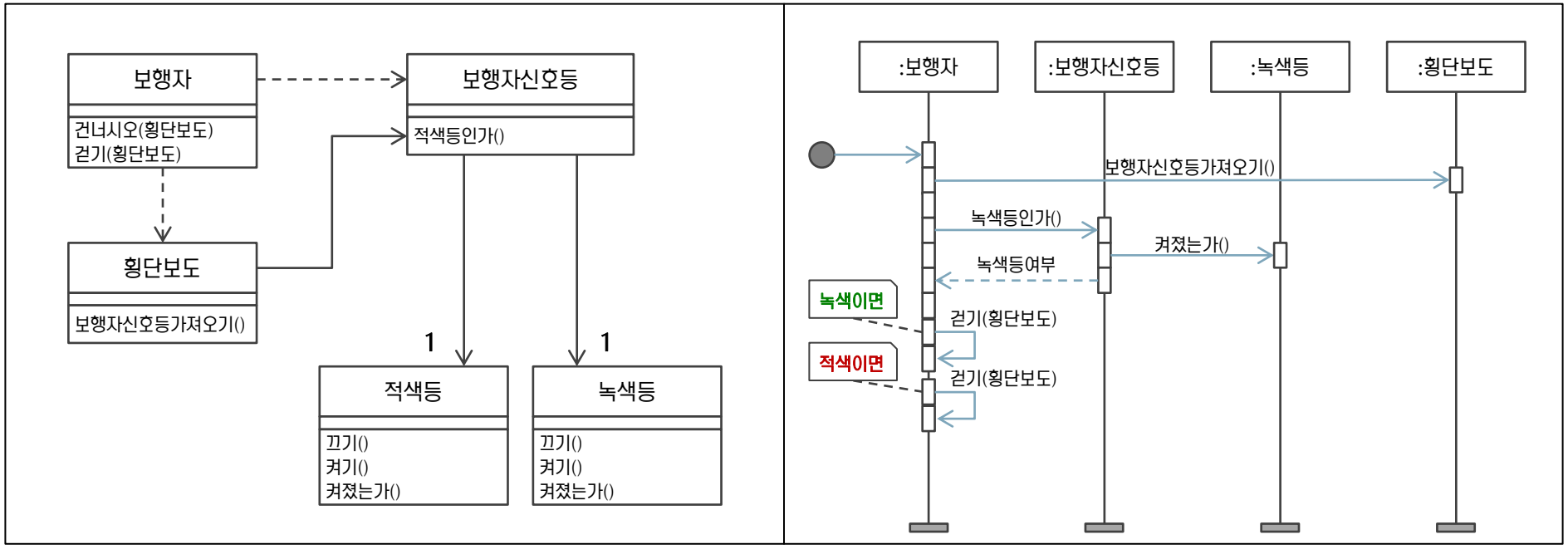
민수: 잠시 기다려야겠네요.

# 1.4 실습 1-2 단계 – UML 모델

- ✓ 시나리오에서 적색등과 녹색등이 식별되었습니다. 이를 모델에서 표현하는 방법은 다양합니다.
- ✓ 신호등 클래스를 두 종류로 식별하는 경우 코드의 중복 문제가 발생합니다.
- ✓ 코드 중복과 재활용 문제를 어떤 방법으로 풀어나가는가가 이번 시나리오의 핵심 목적입니다.
- ✓ 또한 객체간의 협업을 잘 파악하여 그 어떤 객체도 불만이 없도록 책임을 조율해야 합니다.

협업 이해 → 도메인 객체 모델링

1. 보행자가 횡단보도로 다가가며 보행자신호등을 확인한다. 적색등이 켜져있다. 녹색등을 기다리며 멈춰선다.



# 1.4 실습 1-2 단계 – UML 모델

- ✓ 시나리오에서 적색등과 녹색등이 식별되었습니다. 이를 모델에서 표현하는 방법은 다양합니다.
- ✓ 신호등 클래스를 두 종류로 식별하는 경우 코드의 중복 문제가 발생합니다.
- ✓ 코드 중복과 재활용 문제를 어떤 방법으로 풀어나가는가가 이번 시나리오의 핵심 목적입니다.
- ✓ 또한 객체간의 협업을 잘 파악하여 그 어떤 객체도 불만이 없도록 책임을 조율해야 합니다.

협업 이해 → 도메인 객체 모델링

1. 보행자가 횡단보도로 다가가며 보행자신호등을 확인한다. 적색등이 켜져있다. 녹색등을 기다리며 멈춰선다.

- **상황설정의 현실화 - 시나리오를 좀 더 사실적으로 만들어보자!**

1. 횡단보도에는 보행자 신호등이 양쪽에 있다. 보행자가 어느 쪽에서 건너는가에 따라 확인하는 대상(장치)가 달라진다. 흐름이 바뀌므로, 쪽(방향)은 매우 중요한 의미를 지닌다. 따라서, 이것을 독립된 개념으로 정의해야 한다.
2. 시나리오에는 녹색등과 적색등이 보행자 신호등의 속성으로 존재했다. 객체가 아닌 속성이므로 실체가 없다.
3. 참여객체 중 가장 많은 장식을 달고 있는 것은 횡단보도이다.(횡단보도가 보행자 신호등을 소유한다고 설정하였으므로). 이 횡단보도를 건설하는 buildCrossWalk가 Director라는 객체안에서 너무큰 비중을 차지하게 되어, 분리가 필요하다.

- **보행자 신호등이 적색등과 녹색등을 속성으로 가질 때의 문제점**

1. 보행자 신호등이 등을 끄고 켜는 일이 속성을 설정하거나 체크하는 것이므로 번거롭다.
2. 지문에는 녹색등, 적색등이 지속적으로 언급된다. 자주 등장한다는 것은 역할이 있다는 것이고, 객체가 될 자격이 있다는 것이다. 향후에는 노란등까지 언급되는데, 그에 대응하는 객체가 모델링 영역이나 프로그래밍 영역에는 존재하지 않는다.

- **새로운 개념에 대한 표현**

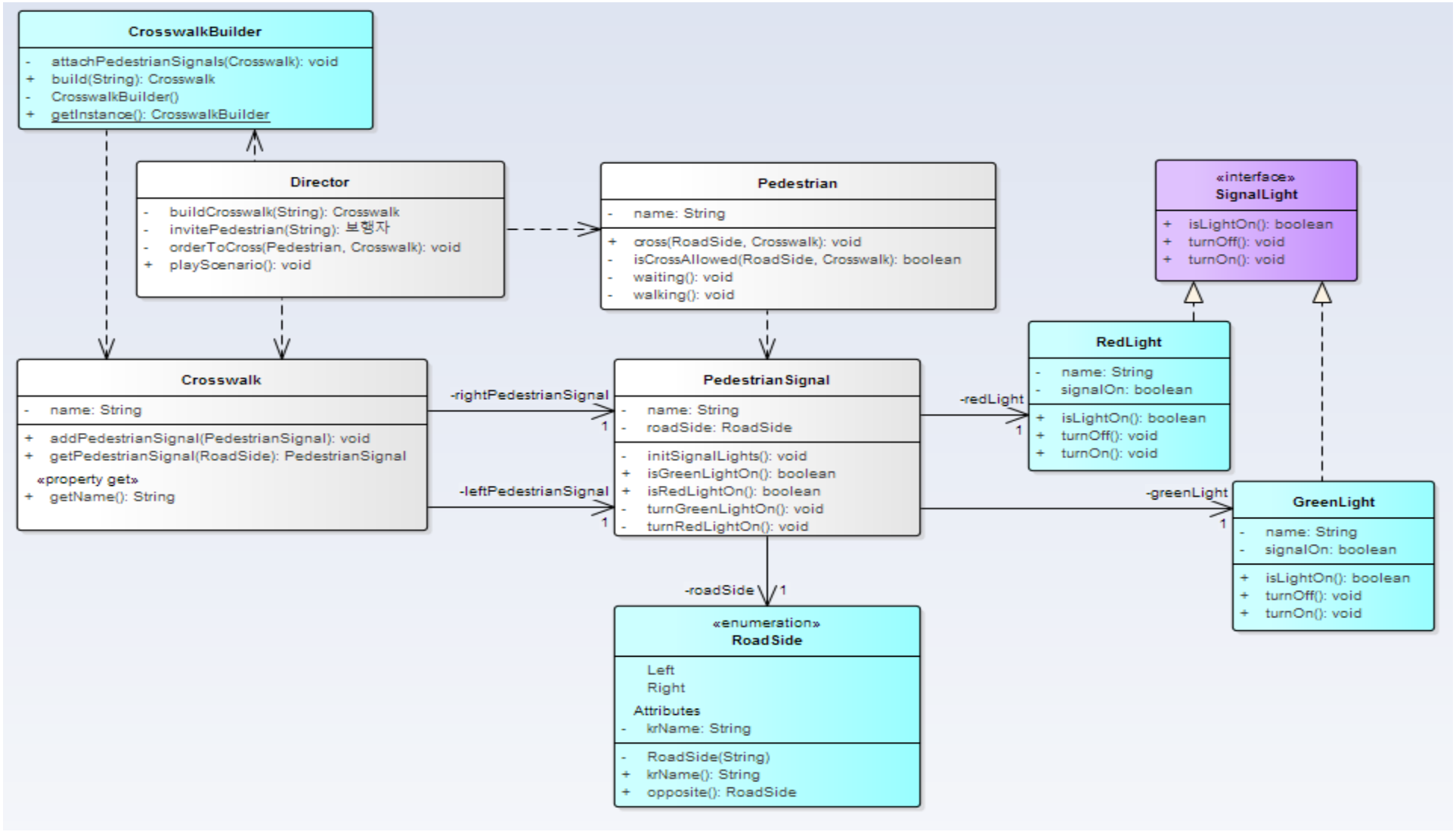
1. 보행자가 어느 쪽에서 건너는가에 따라 확인하는 대상(장치)가 달라진다. 따라서 "쪽(Side)"라는 개념은 시나리오의 흐름을 결정짓는 단서가 되므로 중요한 역할을 한다.
2. 개념의 타입을 신호등의 속성으로 둘 것인가? 속성은 왼쪽, 오른쪽을 가지고 있지만 보행자가 필요한 보행자신호등의 위치는 왼쪽, 오른쪽이라는 정적인 개념이 아니다. 단지 자신의 반대편이 어딘지만 판단하면 된다. 상수의 열거와 행위를 가져오기 위해 Enum 클래스로 선언하는 것이 효율적이다.

- **메소드의 규모가 커졌을 때의 해결 방법**

1. 보행자 신호등에 여러 속성이 추가되면서, 횡단보도를 건설(구성)하는 일은 더욱 어려워졌다. 시나리오 1.1에서는 보행자신호등 하나만 설정했으므로 연출자가 구성했지만 1.2에서는 방향이라는 개념이 생겨나면서 횡단보도에 두개의 신호등이 구성되어야 한다.
2. 따라서, 너무 많은 일을 하고 있는 메소드는 객체로 분리 시키는 작업이 필요하다.

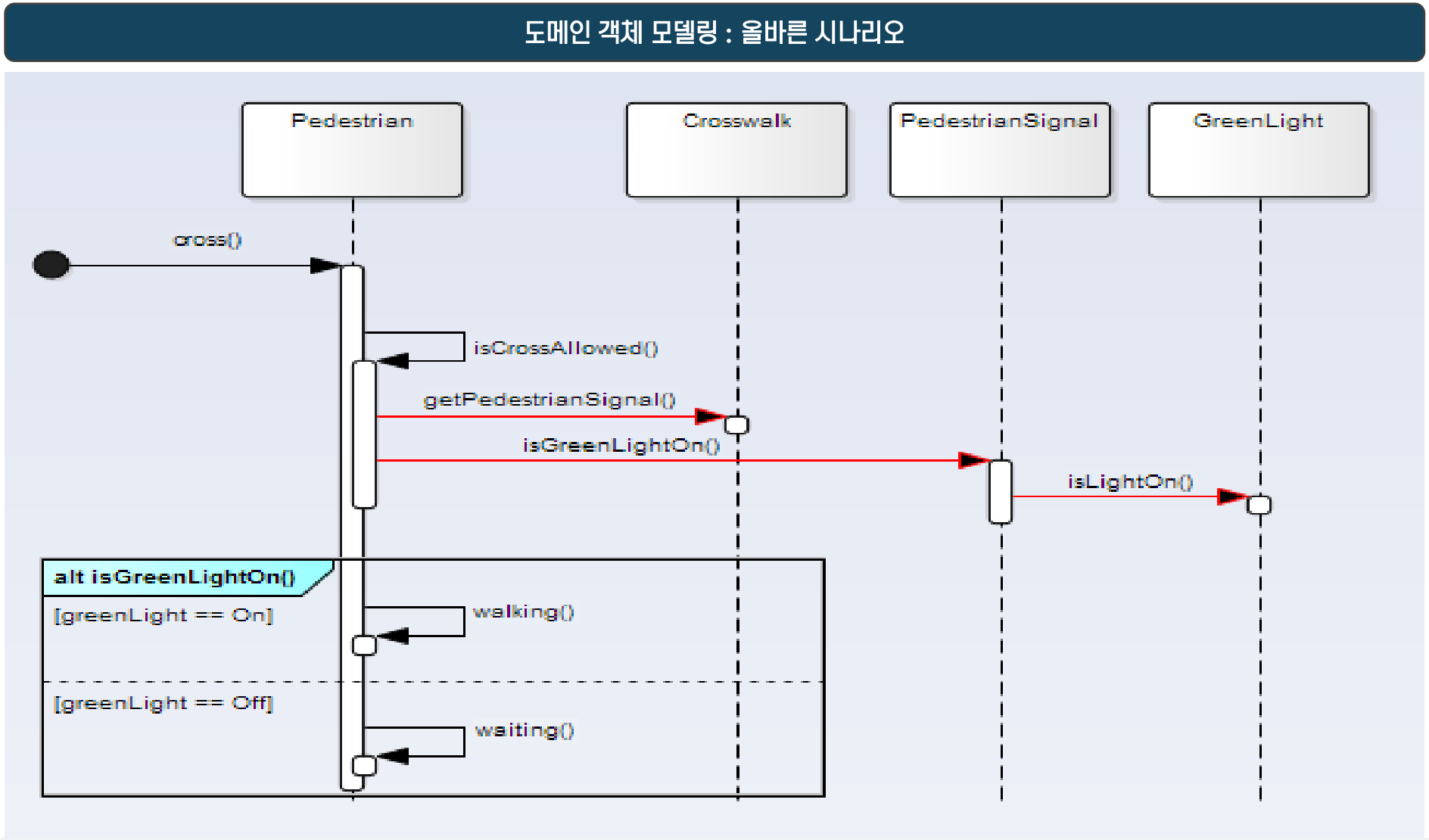
# 도메인 객체 모델링

✓ 클래스 다이어그램



# 도메인 객체 모델링

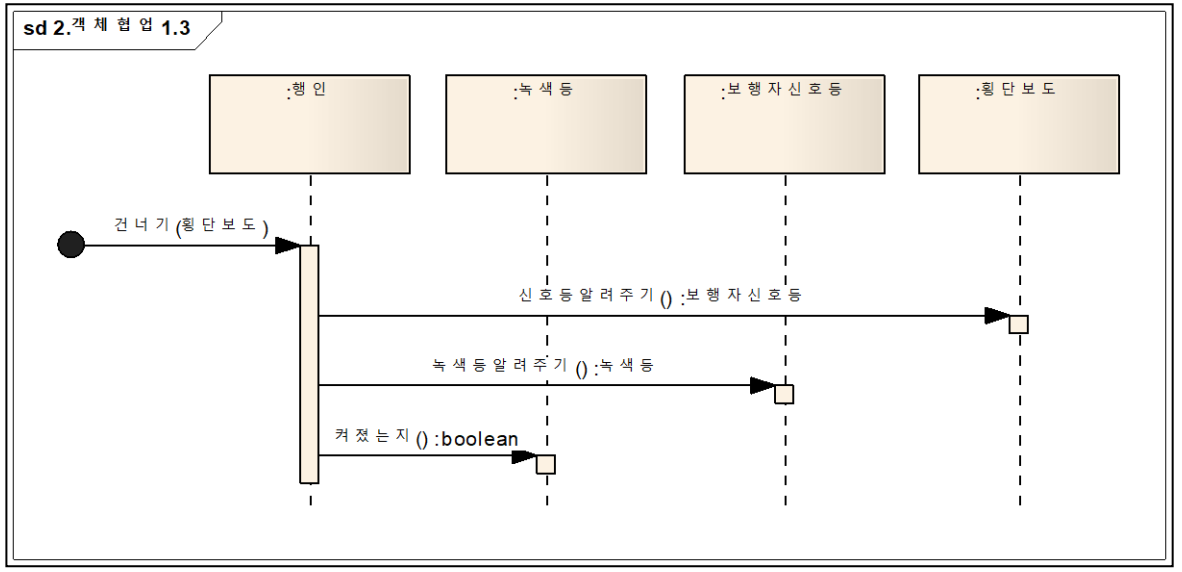
✓ 보행자 신호등 객체에게 신호등 확인 책임을 위임하는 시나리오

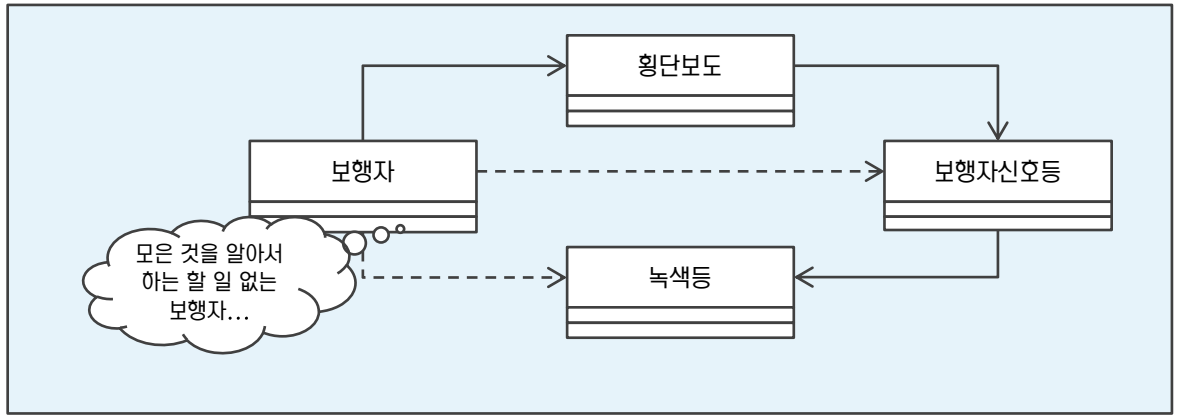


# 도메인 객체 모델링

## 1. 혼자 북치고 장구치는 보행자를 가진 시나리오

### 모델링 오류 Case 1. 혼자 북치고 장구치는 보행자




모든 것을 알아서 직접 수행하는 보행자를 위한 시나리오를 가정해보자. 모델링 실습 중에 가끔씩 발견하는 시나리오이다 .

보행자는 횡단보도로부터 보행자신호등을 얻어온다. 다음으로 보행자신호등으로부터 녹색등을 얻어온다. 그런 다음 녹색등에게 점등여부를 묻는다.

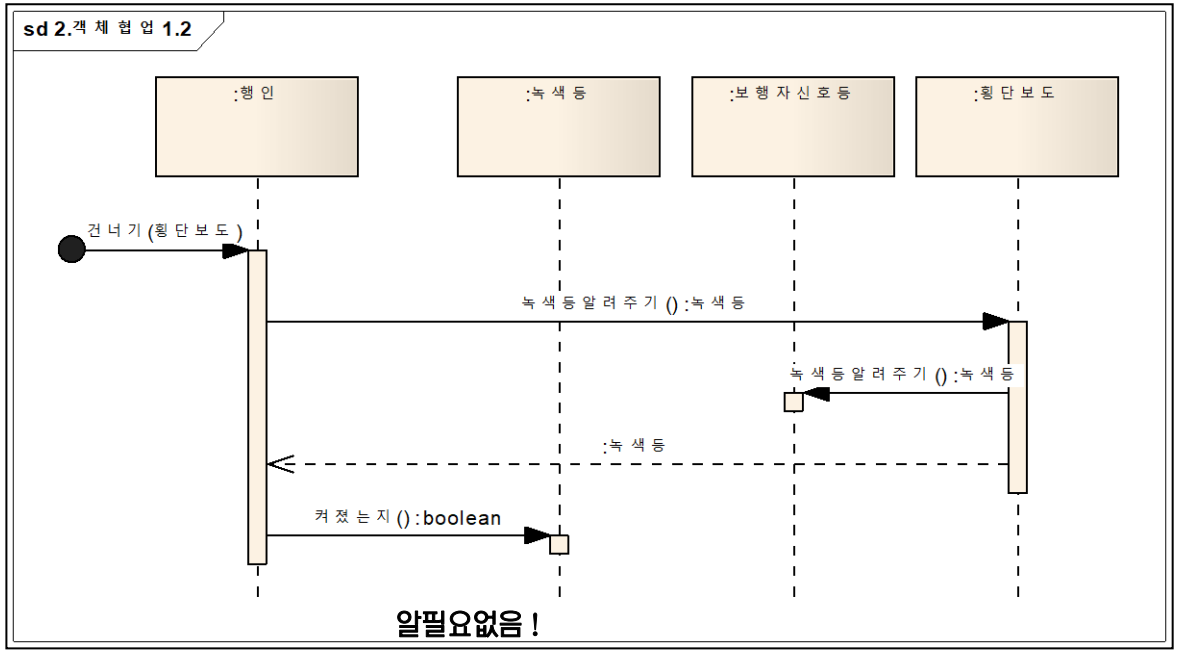
이 시나리오에서 횡단보도나 보행자신호등은 모두 Information Holder로서의 역할을 가지고 있다. 적어도 이 시나리오에서 두 객체의 중요성은 아주 낮어보인다. 보행자는 누가 무엇을 가지고 있는지 모두 알알고 있으며, 객체들이 가지고 있는 책임을 자세히 알고 있다. 할 일이 없거나 매우 피곤한 스타일의 관리자와 같다.

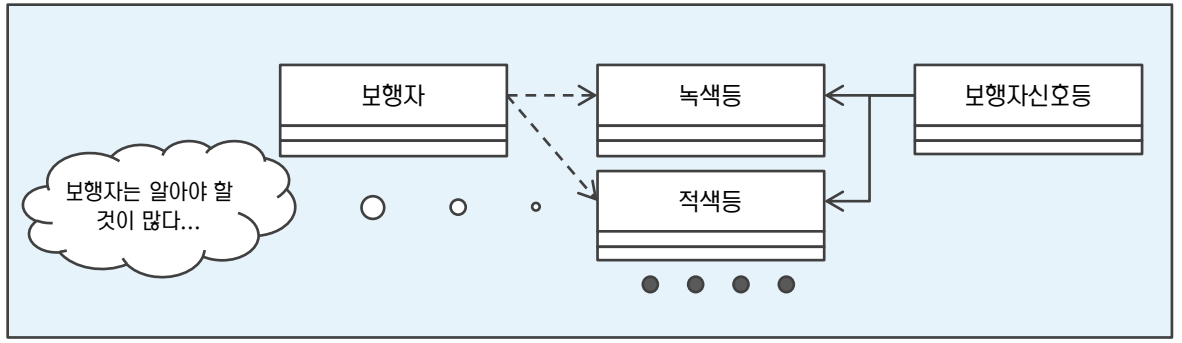
이 시나리오에서 횡단보도나 보행자신호등 모두 자신의 역할에 대해 불만스러워 할 것 같다. 보행자가 혼자서 설치고 있으니… 보행자 스스로도 아주 피곤할 것 같다…

# 도메인 객체 모델링

## 2. 개념없는 보행자 - 보행자신호등 객체를 단순한 Information Holder로 보는 시나리오

### 모델링 오류 Case 2. 개념없는 보행자



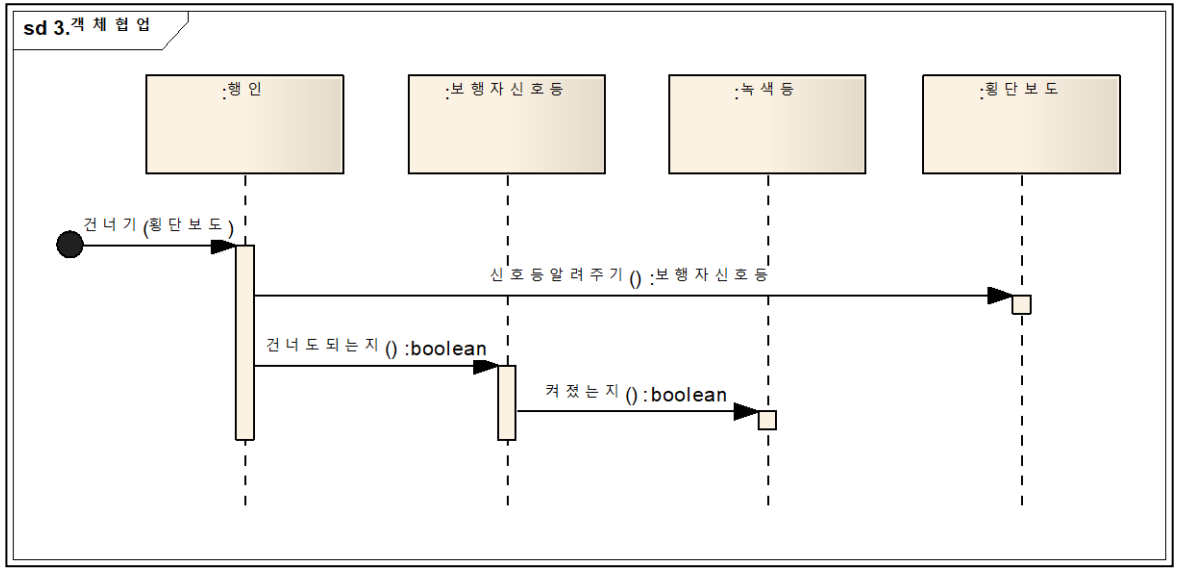

보행자는 횡단보도를 통해서 녹색등을 직접 얻은 다음 녹색등이 켜졌는지 물어본다. 이 시나리오에서 보행자신호등은 자신이 가지고 있던 녹색등을 요청에 의해 내어놓는다. 이 시나리오에서 보행자신호등의 역할은 녹색등을 가지고만 있는 Information Holder로서의 역할만을 하고 있다. 물론 다른 시나리오에서 녹색등을 시간의 지남에 따라 끄고 켜는 일을 하겠지만, 적어도 이 사나리오에서는 녹색등과의 대화가 없이 그저 가지고만 있다.

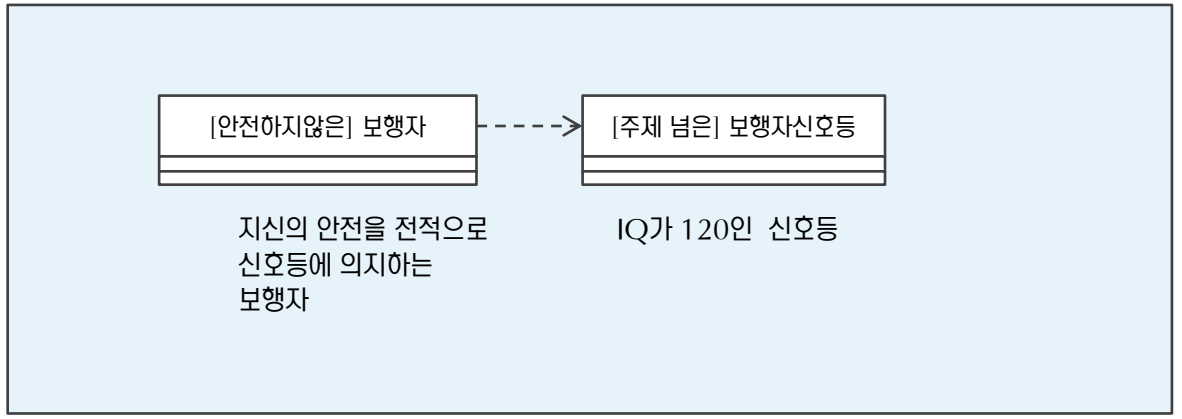
이 시나리오에서 보행자는 녹색등을 자세히 알아야 한다는 부담이 있다. 녹색등을 얻어와야 하고 녹색등과 직접 대화를 통해 정보를 알아내어야 한다. 만약에 등이 여러 종류가 있다면 보행자는 그만큼 많은 등을 알아야 한다. 보행자는 피곤할 것이며 따라서 이 시나리오에서는 행복하지 않다. 보행자신호등 역시 자신의 제어하에 있는 녹색등을 보행자에게 내어주는 것이 별로 반가운 일이 아니며, 녹색등에 대한 자신의 역할을 만족하지 못할 것이다. 횡단보도 역시 자신이 녹색등을 알아야 하는 문제가 있다.

# 도메인 객체 모델링

## 3. 사고 위험이 있는 보행자, 책임의 소재를 잘못 지정한 케이스

### 모델링 오류 Case 3. 사고 위험이 있는 보행자



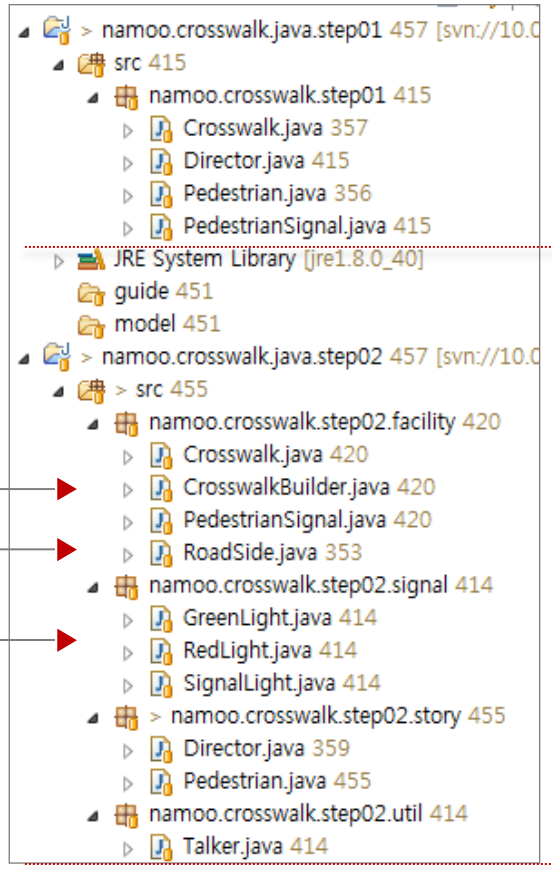

어떤 보행자가 보행자신호등에게 건너도 되는지에 대한 판단을 맡기는 시나리오를 생각해보자. 보행자는 보행자신호등의 판단에 따라 건너거나 대기한다.

시나리오을 이렇게 구성하여도 보행자가 횡단보도를 건너는 시나리오는 무리없이 진행될 수 있다. 횡단보도를 건넌다는 최종 목표를 무난히 달성할 수 있다. 하지만, 이러한 방식의 모델링(여기서는 시나리오 구성)은 도메인의 본질을 잘못 파악하여 모델에 반영하는 오류를 범하는 것이다.

도메인의 본질은 도메인 영역에서 협업을 하는 객체들의 역할과 책임수행을 기반으로 파악하여야 한다. 보행자신호등이 건너도 되는지에 대한 판단을 한다면 보행자를 위한 횡단보도 설계가 잘못될 수 있다. 즉, 보행자신호등이 원래 자신의 책임이 아닌 것을 수행함으로써 너무 복잡해지거나 자신이 하지 말아야 할 일을 수행할 수도 있다. 이러한 역할과 책임에 대한 오류는 모델링 작업 시 범할 수 있는 기본적인 오류이다.

# 1.4 실습 1-2 단계 – Java coding

- ✓ 시나리오를 두 단계로 나누어서 코딩합니다. 결과 코드는 매우 단순합니다.
- ✓ 주요 협업이 있을 때 마다, 요청과 응답 내용을 출력하여서, 어떤 협업이 있는 지 보여 줍니다.
- ✓ 차근차근 문제를 해결하기 위해 두 단계로 실습을 합니다. 첫 실습에서는 기본 협업 체계를 모델링하고 구현합니다.
- ✓ 둘째 실습에서는 프로그램이 갖추어야 할 패키지 틀을 갖추고, 한 수준 더 자세하게 구현합니다.




복잡객체(횡단보도)는 빌더가 필요합니다.

왼쪽/오른쪽을 정의합니다.

적색등/녹색등을 정의합니다. 공통 인터페이스를 정의합니다.

패키지를 정의하고 객체를 나눕니다..

namoo.crosswalk.step01

<KimPD:Director> 시나리오를 시작합니다. 큐~
<KimPD:Director> 횡단보도를 건설합니다.
<KimPD:Director> 보행자 minsoo 씨가 등장합니다.
<KimPD:Director> minsoo 씨 cross11 횡단보도를 건너 가세요.
<minsoo:Pedestrian> cross11를 건너가라구요? 알았어요.
<minsoo:Pedestrian> cross11 횡단보도, 보행자신호등을 주세요.
<cross11:Crosswalk> 보행자 신호등 여기 있습니다.
<minsoo:Pedestrian> pedSignal 보행자신호등, 녹색등이 켜져있나요?
<pedSignal:PedestrianSignal> 적색등이 켜져 있습니다.
<minsoo:Pedestrian> 적색등이군요. 녹색등이 들어올 때까지 기다려야겠군요.

namoo.crosswalk.step02

# 1.5 실습 2 단계(신호등변경) – 시나리오(1/2)

✓ 두번째 시나리오는 신호등이 주어진 시간에 따라 녹색등과 적색등이 교대로 켜지는 내용입니다.
✓ 타이머와 같이 시나리오에는 나와 있지 않지만 핵심적인 역할을 하는 개체를 식별하는 것이 핵심입니다.
✓ 시간을 제어하기 위한 타이머, 보행자 신호등 두개와 타이머를 관리하기 위한 신호제어기를 식별 할 수 있습니다.
✓ 신호제어기의 경우와 같이 당장은 역할이 미미하지만, 후에 여러가지 역할을 할 것으로 예상되는 개체도 있습니다.

**협업 이해 →** 도메인 객체 모델링 → 구현

2. 시간이 흘러 녹색등이 켜지고, 행인은 건너간다. 시간이 흘러 다시 적색등이 켜진다.

(막이 열리면 무대 위에 여러 장치들이 놓여 있다. 신호제어기, 타이머가 있고, 각 보행자신호등 별로 녹색등, 적색등 두개가 달려있다.)

신호제어기 : 자 지금부터 30초를 기준시간으로 하여 신호등을 변경할 겁니다. 우선 보행자 신호등, 신호등을 초기화하세요.
보행자신호등 : 예, 초기화합니다. 녹색등을 끄고, 적색등은 켭니다. 이제 준비되었어요.
신호제어기 : 타이머는 지금부터 1초 단위로 경과시간을 통지해 주세요.
타이머 : 예, 신호제어기, 1초 지났습니다.
신호제어기 : 아, 그래요. 우리의 기준 시간이 30초이니, 29초 남은 셈이군요.
왼쪽 보행자 신호기, 29초 남았습니다. 오른쪽 보행자 신호기, 29초 남았습니다.
왼쪽보행자신호기 : (아무런 반응이 없다.) 아, 그래요. 29초 남았군요.
오른쪽보행자신호기 : (아무런 반응이 없다.) 아, 그래요. 29초 남았군요.

... 위의 진행을 반복한다. (30초가 지났다.)

...(다음장에서 계속)

# 1.5 실습 2 단계(신호등변경) - 시나리오(2/2)

타이머 : 신호제어기 30초 지났습니다.
신호제어기 : 아, 그래요. 우리의 기준 시간이 30초이니, 0초 남은 셈이로군요.
왼쪽 보행자 신호기, 0초 남았습니다. 오른쪽 보행자 신호기, 0초 남았습니다.
왼쪽보행자신호기 : 아, 그래요. 시간이 다 되었군요. 신호등을 변경해야지. 녹색등, 지금 켜져있나요?
왼쪽녹색등 : 아뇨 꺼진상태인데요.
왼쪽보행자신호기 : 그래요. 그럼 녹색등은 켜구요. 적색등은 끄세요.
왼쪽녹색등 : 예, 켰습니다.
왼쪽적색등 : 예, 껐습니다.

오른쪽보행자신호기 : 아, 그래요. 0초나 남았다구요. 신호등 변경해야겠네요. 녹색등 지금 켜져있나요?
오른쪽녹색등 : 아뇨 꺼진상태인데요.
오른쪽보행자신호기 : 그래요. 그럼 녹색등은 켜구요, 적색등은 끄세요.
오른쪽녹색등 : 예, 켰습니다.
오른쪽적색등 : 예, 껐습니다.

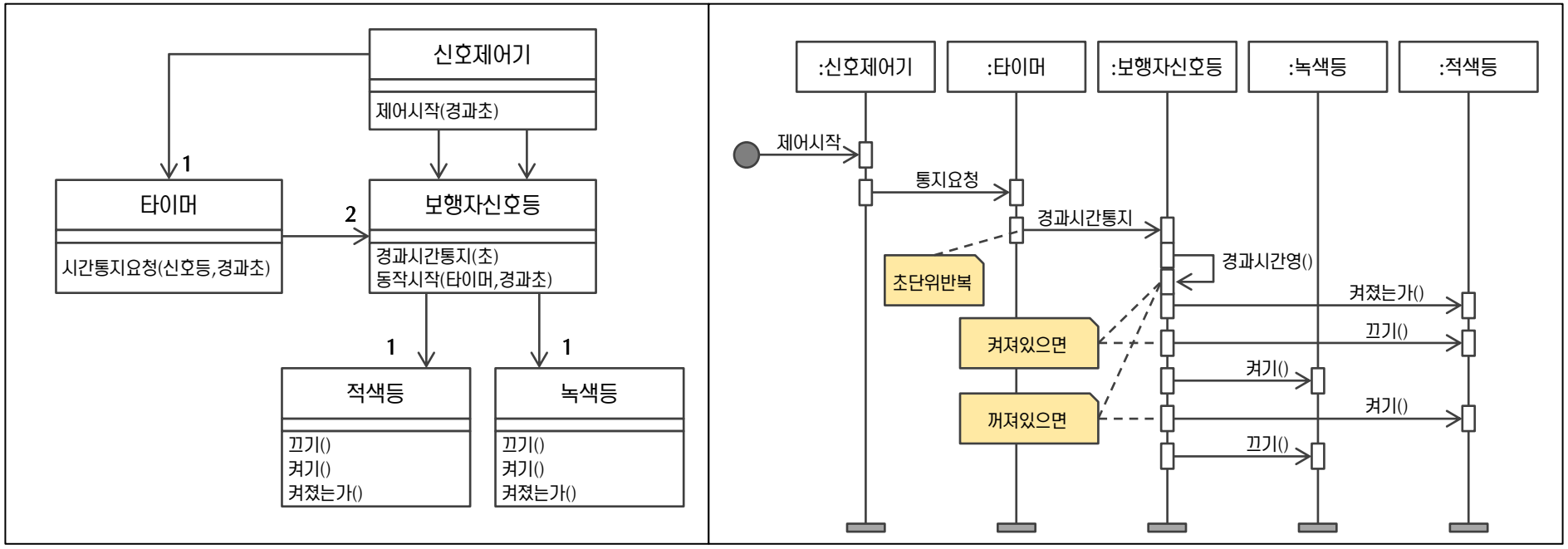
...(시나리오 끝)

# 1.5 실습 2 단계(신호등변경) – UML 모델

✓ 시나리오에 따라 참여하는 객체들을 식별하고 이를 클래스로 표현합니다.
✓ 클래스 다이어그램으로부터 협업 시나리오를 진행할 수 있습니다.
✓ 각 클래스의 오퍼레이션은 클래스 다이어그램을 작성하면서 식별되고 협업 시나리오를 전개하면서도 식별됩니다.

협업 이해 → 도메인 객체 모델링

2. 시간이 흘러 녹색등이 켜지고, 행인은 건너간다. 시간이 흘러 다시 적색등이 켜진다.

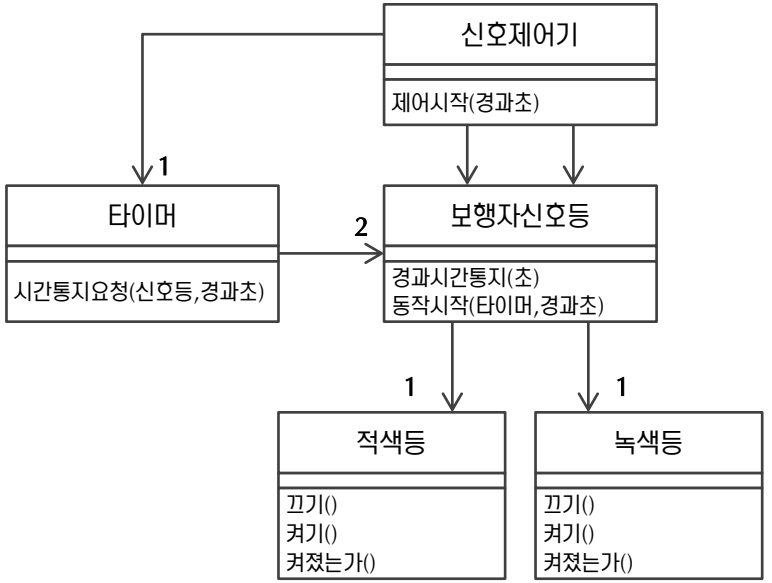


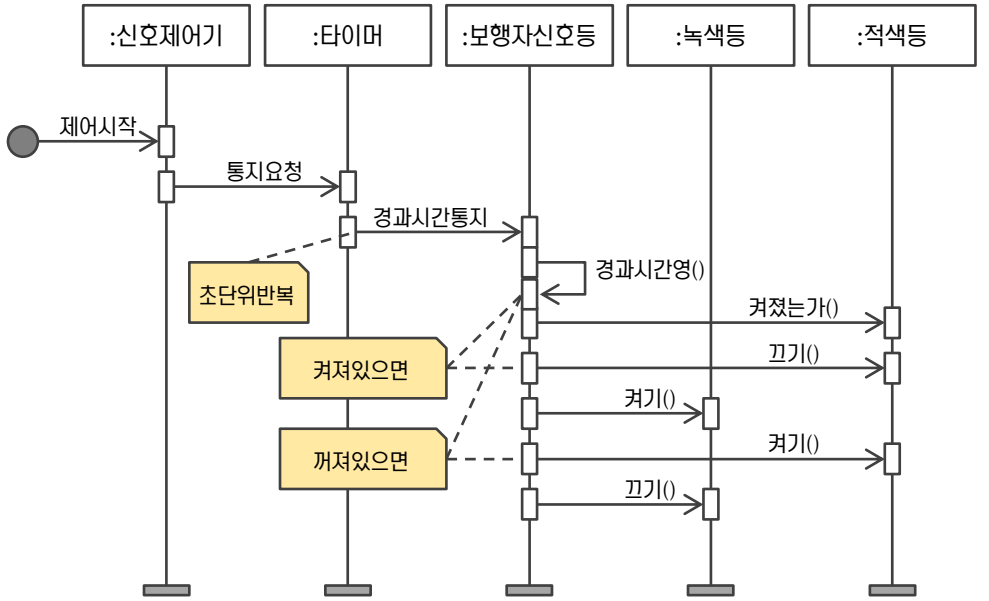
# 도메인 객체 모델링 2 – 신호등 변경

- ✓ 두번째 시나리오는 신호등이 주어진 시간에 따라 녹색등과 적색등으로 교대로 켜지는 내용임
- ✓ 시간을 제어하기 위해 타이머가 등장했고, 보행자 신호등 두 개와 타이머를 관리하는 신호제어기 등장함
- ✓ 지금 당장은 신호제어기의 역할이 미미하지만, 후에 시간대별 신호주기 조절 등과 같은 역할이 예상됨

## 도메인 객체 모델링

2. 시간이 흘러 녹색등이 켜지고, 또 시간이 흘러 적색등이 켜진다. 녹색등과 적색등이 교대로 켜진다.

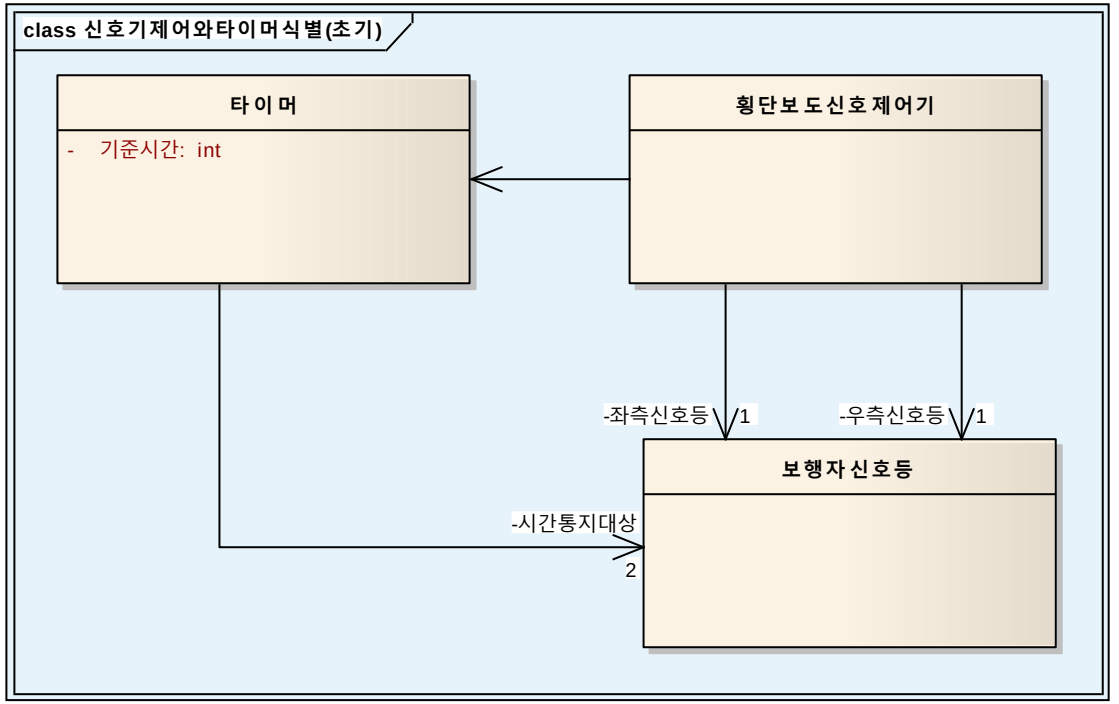


# 도메인 객체 모델링 2 – 모델링(3/7)

✓ 신호제어에 참여하는 클래스를 식별함

## 도메인 객체 모델링 – 객체식별 (3/4)



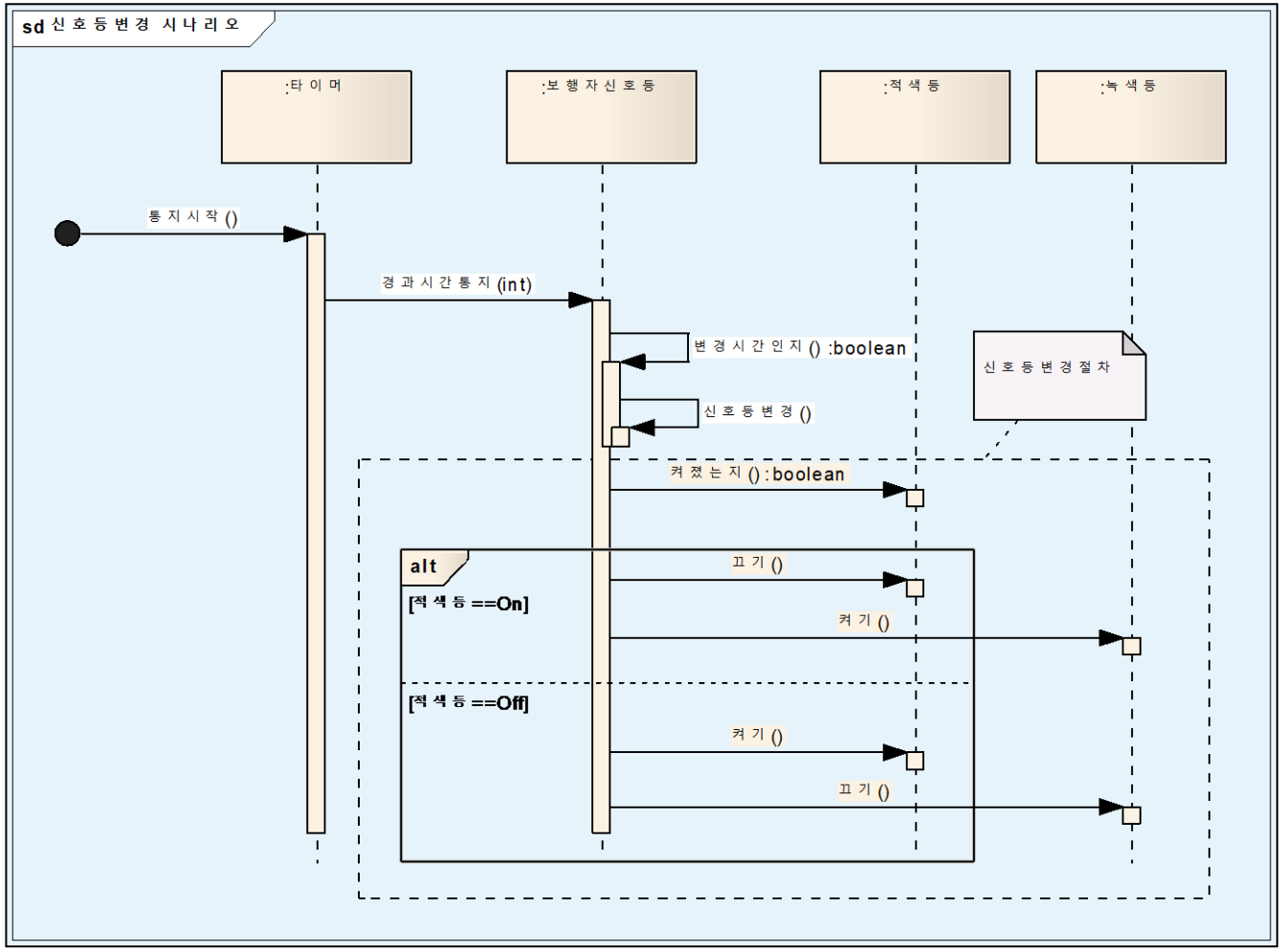

이번 시나리오 조각은 “시간이 흘러 녹색등이 켜지고, 또 시간이 흘러 적색등이 켜진다. 녹색등과 적색등이 교대로 켜진다.”로 어디에도 제어를 한다는 이야기가 없다. 하지만, 녹색등과 적색등이 교대로 켜지고 꺼지는 원리에 대해 파악하면서 그 속에는 시간에 의한 제어 메커니즘이 있음을 파악할 수 있다. 이렇듯 자연언어 속에 “암묵적인 합의에 의해 생략된” 정보를 찾는 과정이 모델링의 한 부분이다.

1. 보행자 신호등 제어와 관련하여 신호제어기를 식별한다. 이 신호제어기는 횡단보도용이므로 “횡단보도신호제어기”라고 구체적인 이름을 붙인다.
2. 신호제어기 입장에서 바라보면, 보행자신호등은 길 양쪽에 두 개가 있다. 그냥 좌측신호등, 우측신호등으로 식별한다.
3. 신호제어기는 경과시간을 측정을 위해 타이머 하나를 가진다.
4. 타이머 입장에서 보행자신호등은 “시간통지대상”일 뿐이다. 그리고 좌측,우측 구분없이 그냥 두 개의 대상에 대해 순차적으로 시간을 알려줄 뿐이다.
5. 앞에서 식별한 클래스와 새로 식별한 클래스 두 개를 가지고 객체 간의 협업 시나리오를 진행한다.
6. 협업 시나리오 진행을 통해 오퍼레이션을 식별하고, 관계를 조정한다.

# 도메인 객체 모델링 2 – 모델링(6/7)

✓ 보행자 신호등을 끄고 켜는 모델링

## 도메인 객체 모델링 – 객체 협업 시나리오 – 신호등 끄고 켜기



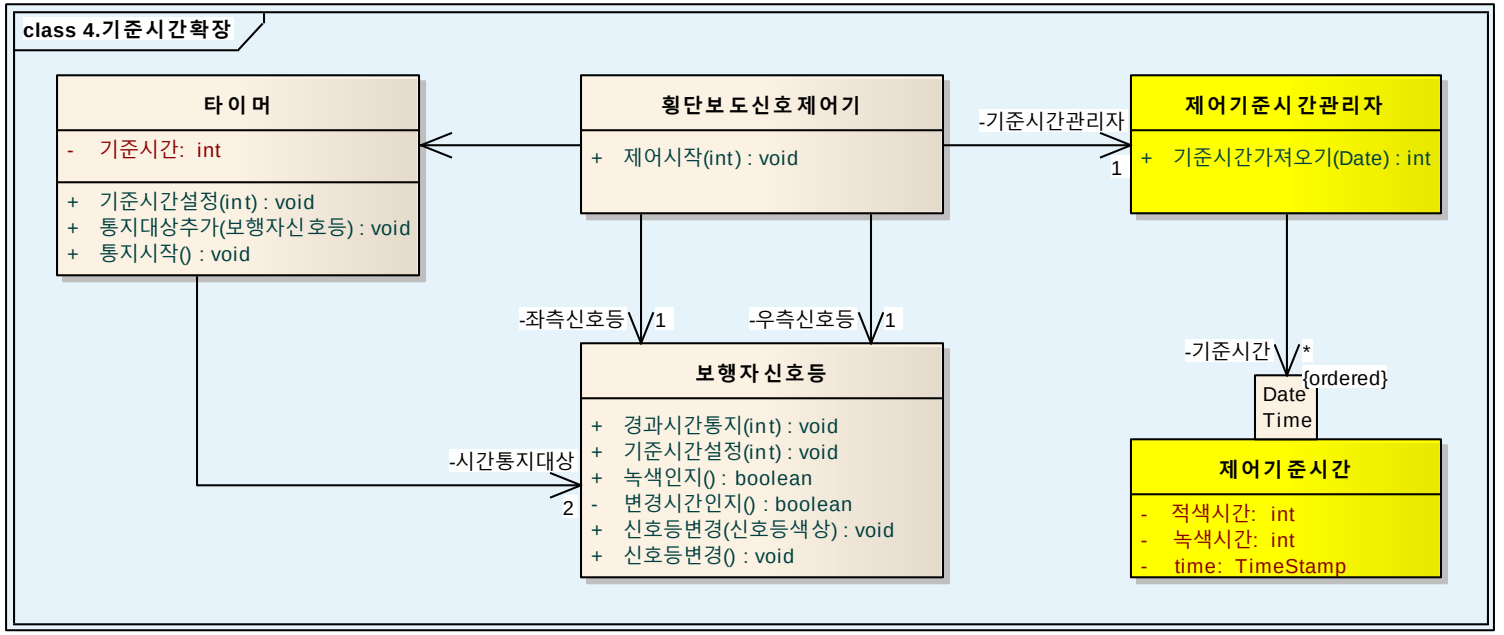

1. 타이머는 초단위로 보행자신호등에게 경과시간을 알려준다.
2. 보행자신호등은 기준시간과 경과시간을 비교하여 신호등 변경시간인지를 판단하고
3. 변경시간이면, 신호등을 변경한다.
4. 점선으로 표시한 블록 안은 신호등을 변경하는 절차를 보여준다.
5. 적색등이 켜져있으면, 적색등을 끄고, 녹색등을 켠다.
6. 반대로, 적색등이 꺼져있으면, 적색등을 켜고, 녹색등을 끈다.

[토론] 끄고 켜는 순서가 의미가 있을까? 있다면 어떤 의미가 있는가? 안전(safty) 관점에서 토의하시오.

# 도메인 객체 모델링 2 – 모델링(7/7)

✓ 제어 기준시간을 단순히 신호길이를 나타내는 초에서 , 제어기준시간 객체로 확장할 수 있음

## 도메인 객체 모델링 – 제어시간 확장



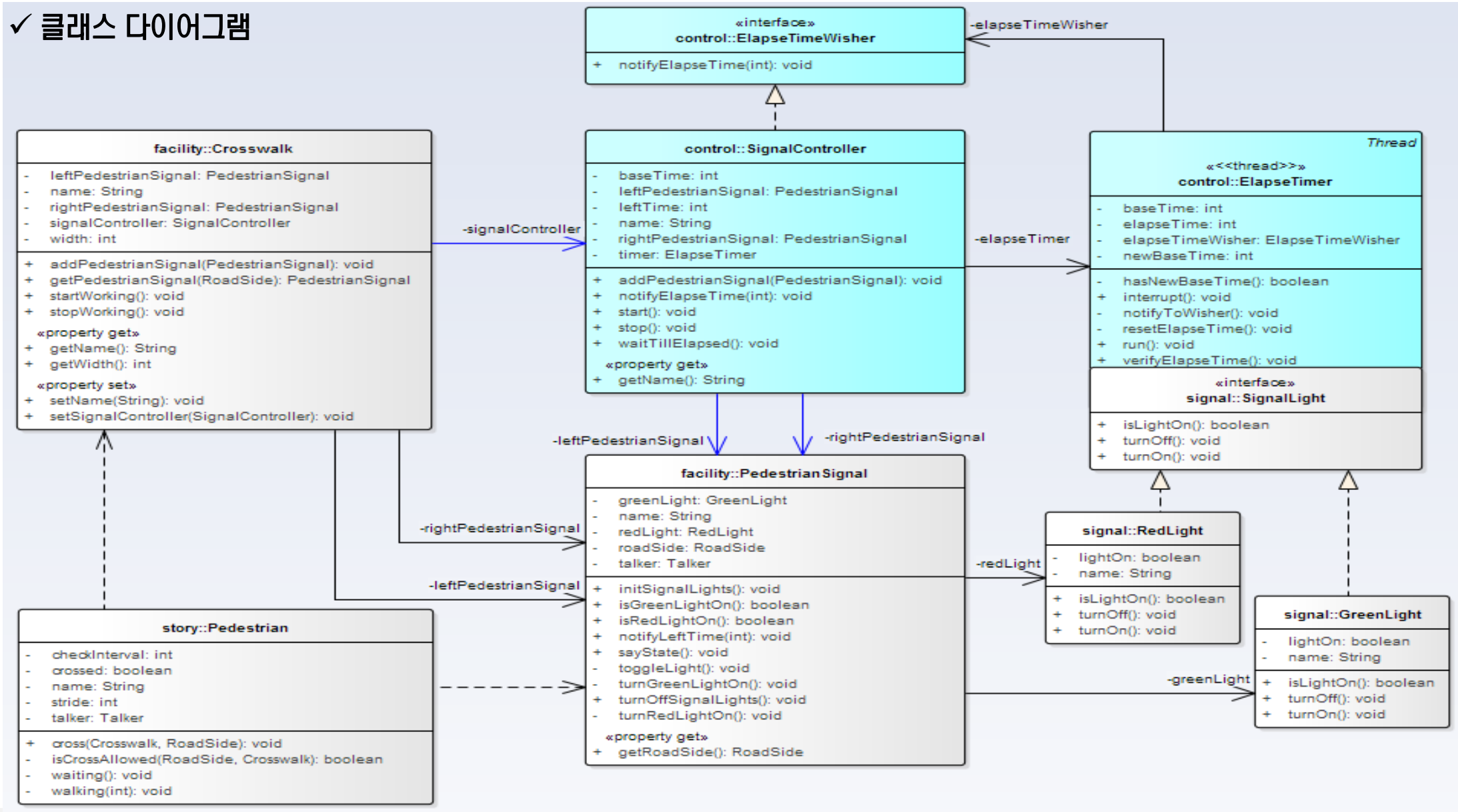

지금까지의 모델에서 제어 기준시간은 신호길이를 초로 표현한 것이었다. 하지만, 녹색등과 적색등의 신호길이가 다를 수 있고, 계절이나 시간에 따라 신호제어 기준시간이 달라진다고 했을 때, 표현할 수 없는 한계를 가지고 있다. 이것을 제어기준시간 객체로 확장할 수 있다.

1. 제어기준시간이 여러 개일 수 있으며, 그 중에서 날자와 시간으로 선택하여야 하므로 제어기준시간을 관리하는 관리자가 필요하다. 신호제어기는 제어기준시간관리자에게 요청시점의 날자와 시간을 기준으로 기준시간을 요청한다. 기준시간가져오기(Date) 오퍼레이션
2. 제어기준시간은 날자와 시간에 따라 서로 다를 수 있다. 만약에 제어기준시간을 객체로 확장한다면, 기준시간설정(제어기준시간) 으로 변경.

신호등 색상이 추가될 가능성은 얼마나 있으며, 그러한 가능성에 대한 확장을 고려해야 하는 지에 대해 토의한다.

# 도메인 객체 모델링 2 – 모델링(7/7)

✓ 클래스 다이어그램



# End of Document

✓ 묻고 답하기

✓ 토의

감사합니다...