1강. UML의 기초

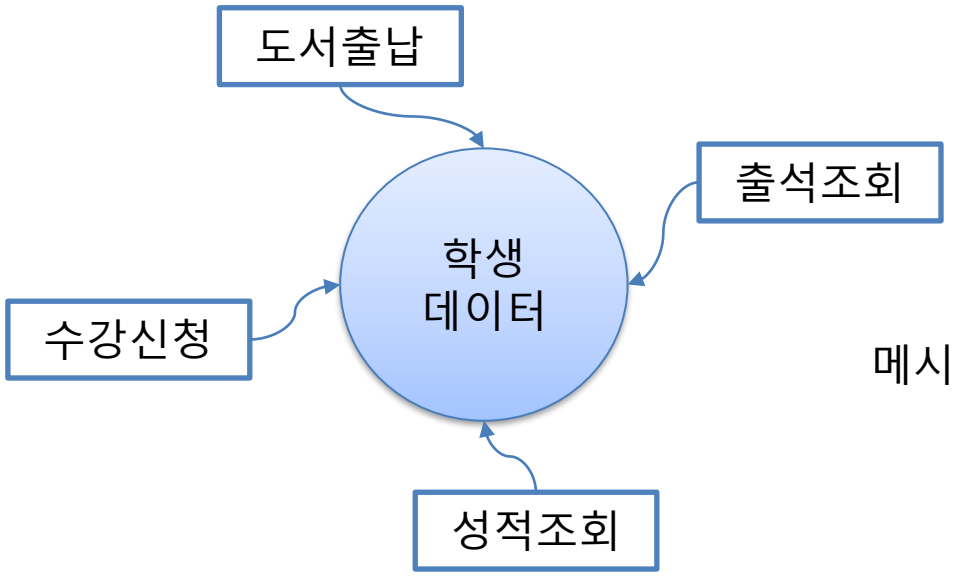
UI 설계

UML의 개요

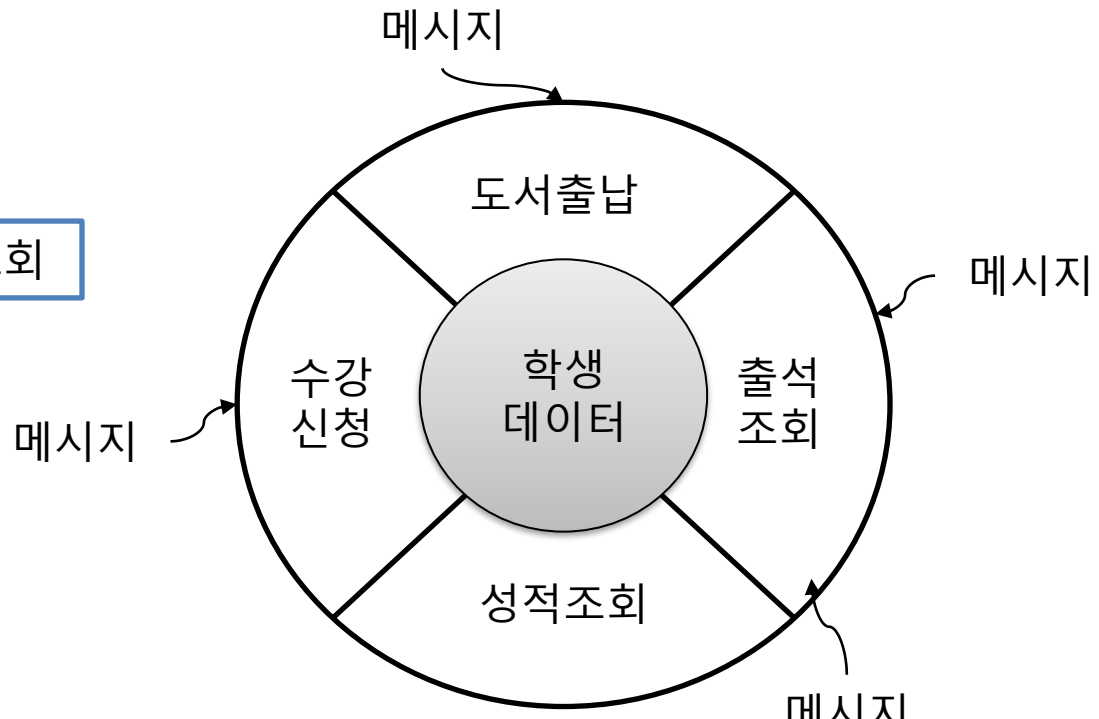
- UML(Unified Modeling Language)
  - 통합 모델링 언어
  - 시스템을 모델로 표현해 주는 대표적인 모델링 언어
  - UML은 시스템 개발자가 구축하고자 하는 소프트웨어를 코딩하기에 앞서 표준화되고 이해되기 쉬운 방법으로 소프트웨어를 설계하여 다른 사람들과 효율적으로 의사 소통할 수 있는 메커니즘을 제공한다.
  - 현재 UML은 객체지향 시스템 개발 분야에서 가장 우수한 모델링 언어이다.

UML의 개요

- 객체 지향 모델링
  - 객체지향 방법론에 의한 소프트웨어 개발이 대세가 된지 오래된다.
  - 이는 모든 문제점과 다양한 규모 및 복잡도를 갖는 대규모 시스템을 구축하는데 가치가 집중되었기 때문이다.



a) 구조적 방법



b) 객체지향 방법

UML의 개요

- 구조적 개발방법과 객체지향 개발 방법
  - 구조적 개발방법

    프로그램의 논리와 데이터를 분리해서 소프트웨어를 설계하기 때문에, 생명주기 각 단계에서 단계별로 자연스럽게 연결되지 않고, 프로그램 내부 기능들은 데이터와 복잡하게 얽혀있어 소프트웨어를 변경하는 일이 쉽지 않다.

  - 객체지향 개발 방법

    구조적 방법의 문제점을 극복하고 인간이 사고하는 방식대로 프로그램을 개발하려는 노력으로 탄생했다.

    객체지향이란 현실세계에 존재하는 실체 및 개념들을 객체라는 독립된 단위로 구성하고 이 객체들이 메시지를 통하여 상호작용함으로써 전체 시스템이 운영되는 개념이다.

    객체, 클래스, 메시지(기능)으로 구성된다.

UML의 개요

- UML 다이어그램의 종류
  - 구조 다이어 그램

    클래스 다이어그램, 객체 다이어 그램, 컴포넌트 다이어그램, 유스 케이스 다이어그램
  - 행위 다이어 그램

    활동 다이어그램, 상태 머신 다이어그램, 상호작용 다이어그램 등

- 개념, 명세, 구현
  - 문제 도메인(problem domain), 소프트웨어 설계 제안, 이미 완성된 소프트웨어 구현에 대한 다이어그램을 그릴때 UML을 사용한다.
  - 세 가지 차원을 개념, 명세, 구현이라는 말을 붙여 구현한다.

유스케이스

- 유스케이스(Use Case)

유스 케이스는 시스템의 동작 하나를 기술한 것이다. 방금 시스템에 특정한 일을 시킨 사용자의 관점에서 작성하며, 사용자가 보낸 자극 '하나 ' 에 대한 반응으로 시스템이 진행하는 '눈에 보이는' 이벤트의 흐름을 포착한다.

눈에 보이는 이벤트란 사용자가 볼 수 있는 이벤트를 뜻한다.

애플리케이션 제작시 사용자 요구사항 확인 툴(tool)로 사용함

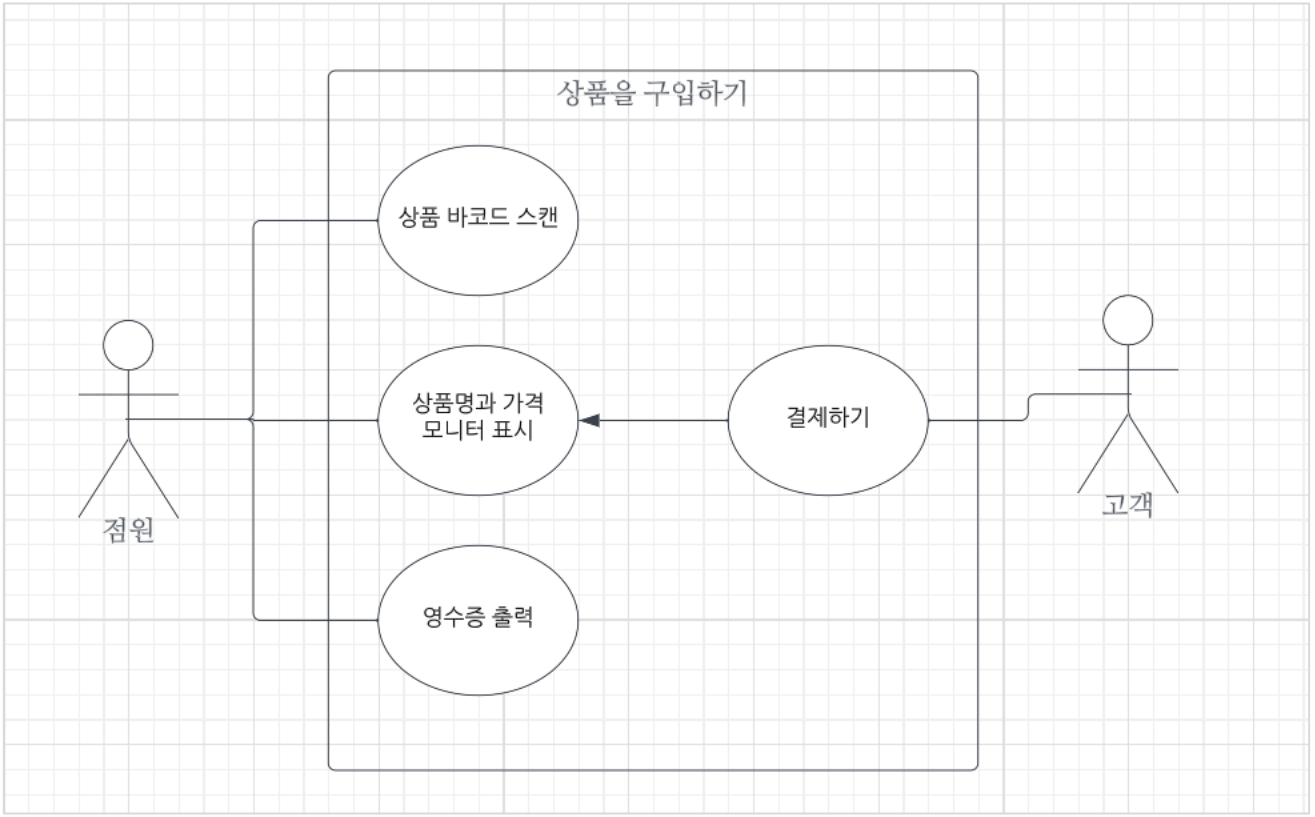
예) 판매시점관리(POS) 시스템

상품을 구입하기

1. 점원은 상품을 스캐너 위로 통과시킨다. 스캐너가 바코드를 읽는다.
2. 상품명과 가격이 고객쪽 화면에 표시된다.
3. 고객은 카드나 현금으로 가격을 결제한다.
4. 영수증이 출력된다.

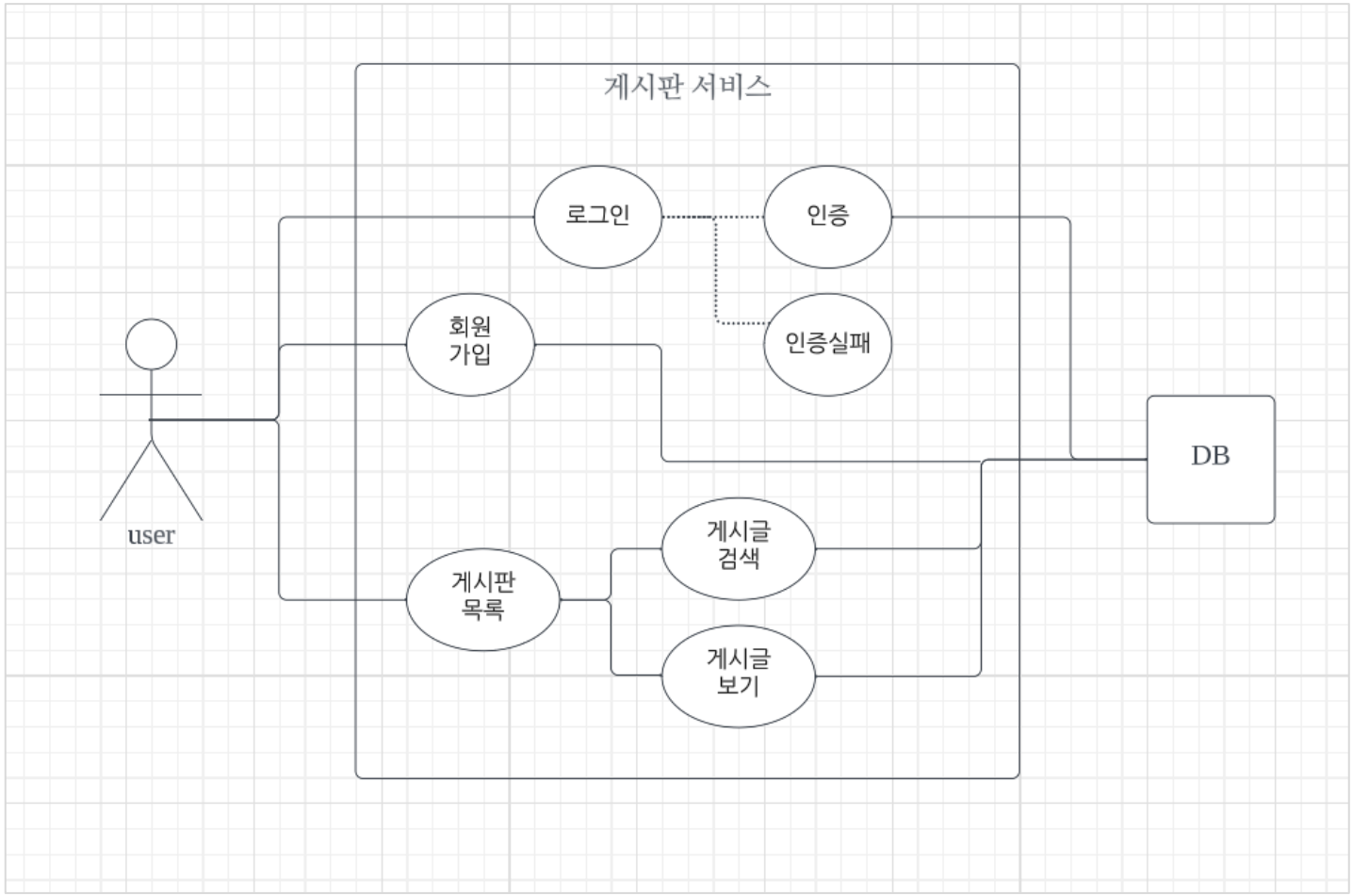
유스케이스

- 유스케이스(Use Case) 다이어그램



유스케이스

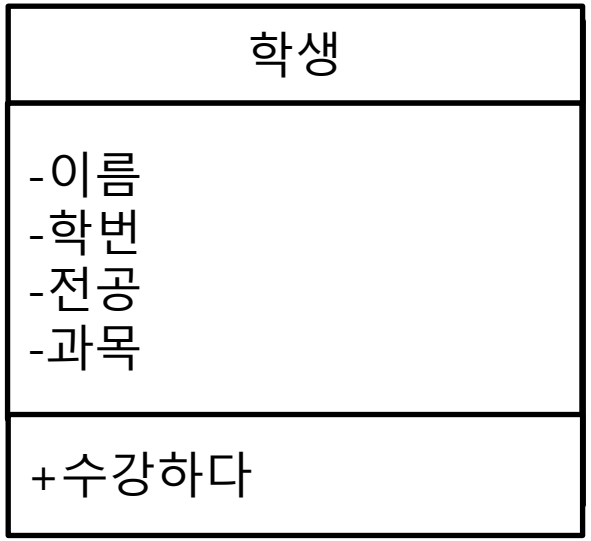
- 유스케이스(Use Case) 다이어그램



UML의 개요

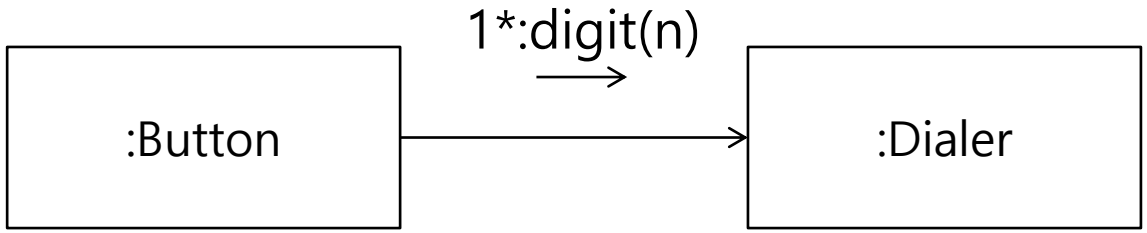
❖ 클래스

- 객체를 생성하는 설계도로 사물의 속성과 기능을 가지고 있다.
- UML 클래스의 표현



다이어그램으로 작업하기

- 시퀀스 다이어 그램
  - 휴대전화를 제어하는 소프트웨어

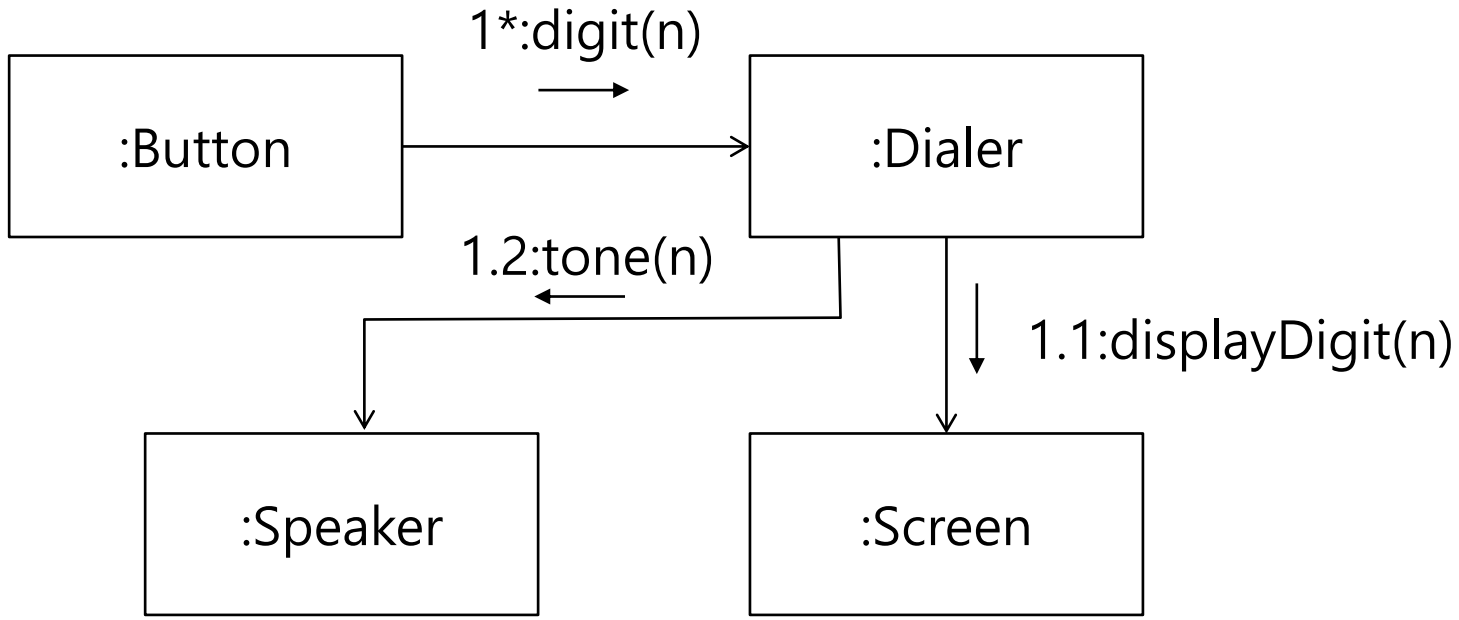


소프트웨어가 버튼이 눌릴 때마다 이를 감지해서 다이얼을 돌리는 일을 제어하는 객체에 메시지를 보낸다.

버튼 객체와 다이얼러 객체를 그리고 Button이 Dialer에 번호 메시지를 여러 개 보내는 것도 그린다. 별표(*)는 '여러 개 ' 를 의미한다.

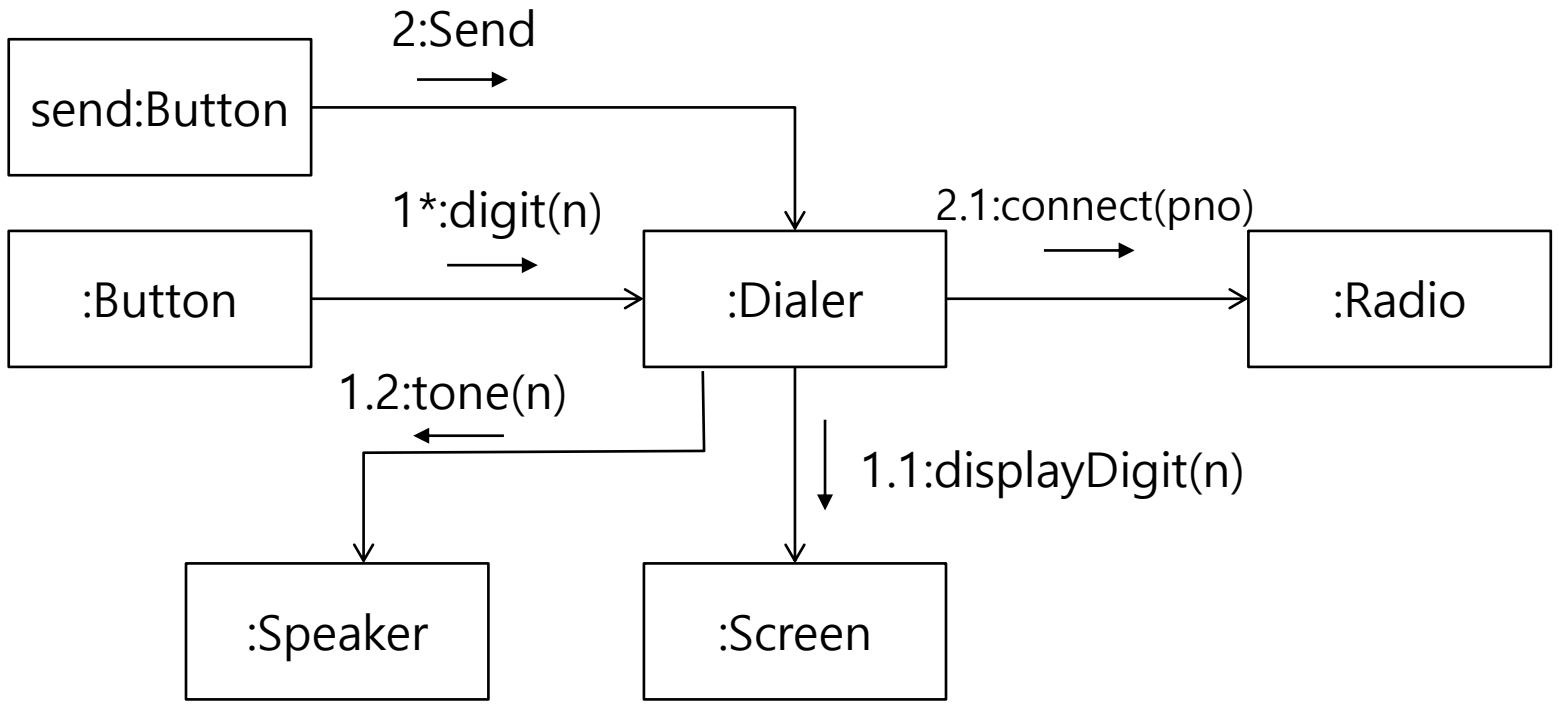
다이어그램으로 작업하기

번호 메시지를 받으면 Dialer는 무엇을 해야 할까? 화면에 번호를 표시해야 하니까 화면(Screen) 객체에 displayDigit 메시지를 보낼것이다.
그리고 스피커를 통해 어떤 톤을 들려주는 것도 좋다.
그러므로 Button이 스피커(Speaker) 객체에도 tone 메시지를 보내게 한다.



다이어그램으로 작업하기

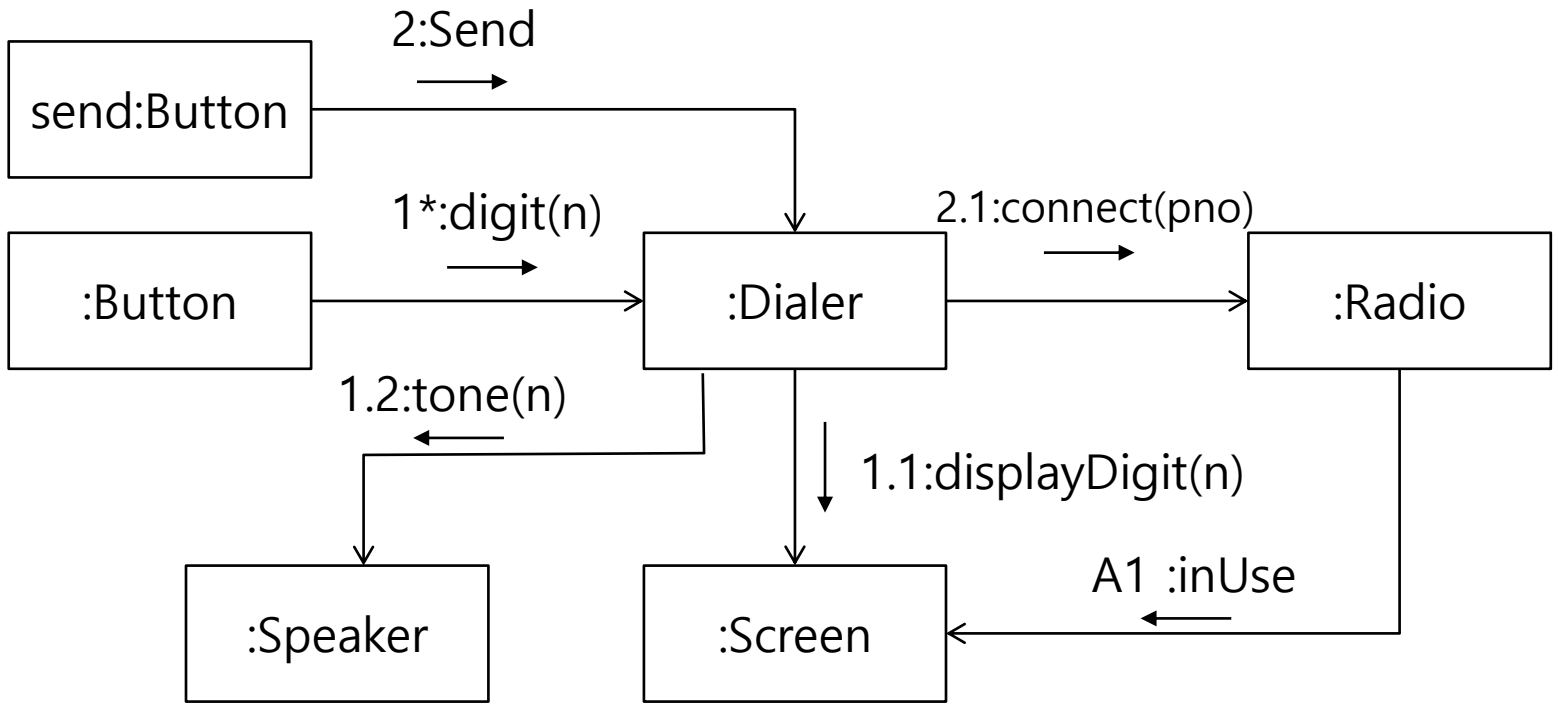
이 시점에서 우리는 셀 네트워크에 접속해서 사용자가 누른 전화번호를 전달하라고 휴대 전화의 무선(Radio) 부분에 말해야 한다.



다이어그램으로 작업하기

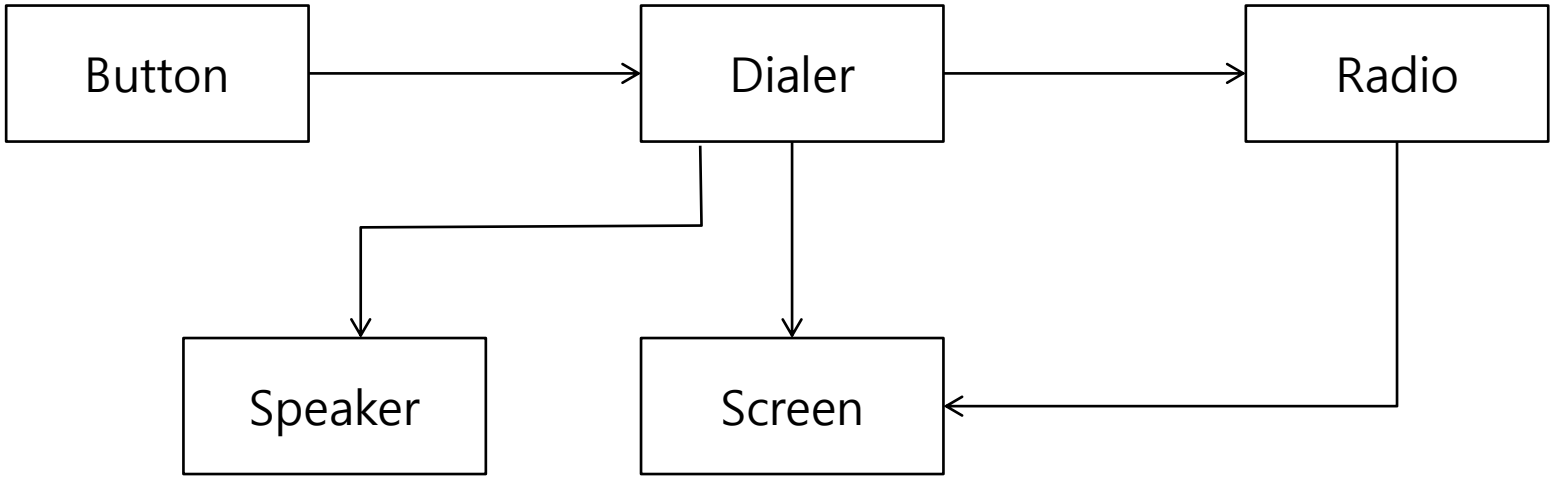
❖ 휴대전화의 협력 다이어그램

연결이 맺어지면 Radio는 화면 객체에 사용 중 지시자에 불을 켜라고 말할 수 있다. 그런데 이 메시지를 보낼 때는 다른 제어 스레드를 사용할 가능성이 굉장히 높다. 그럴때는 시퀀스 번호 앞에 글자를 붙여서 이 사실을 표현한다.



다이어그램으로 작업하기

- 휴대폰 클래스 다이어그램

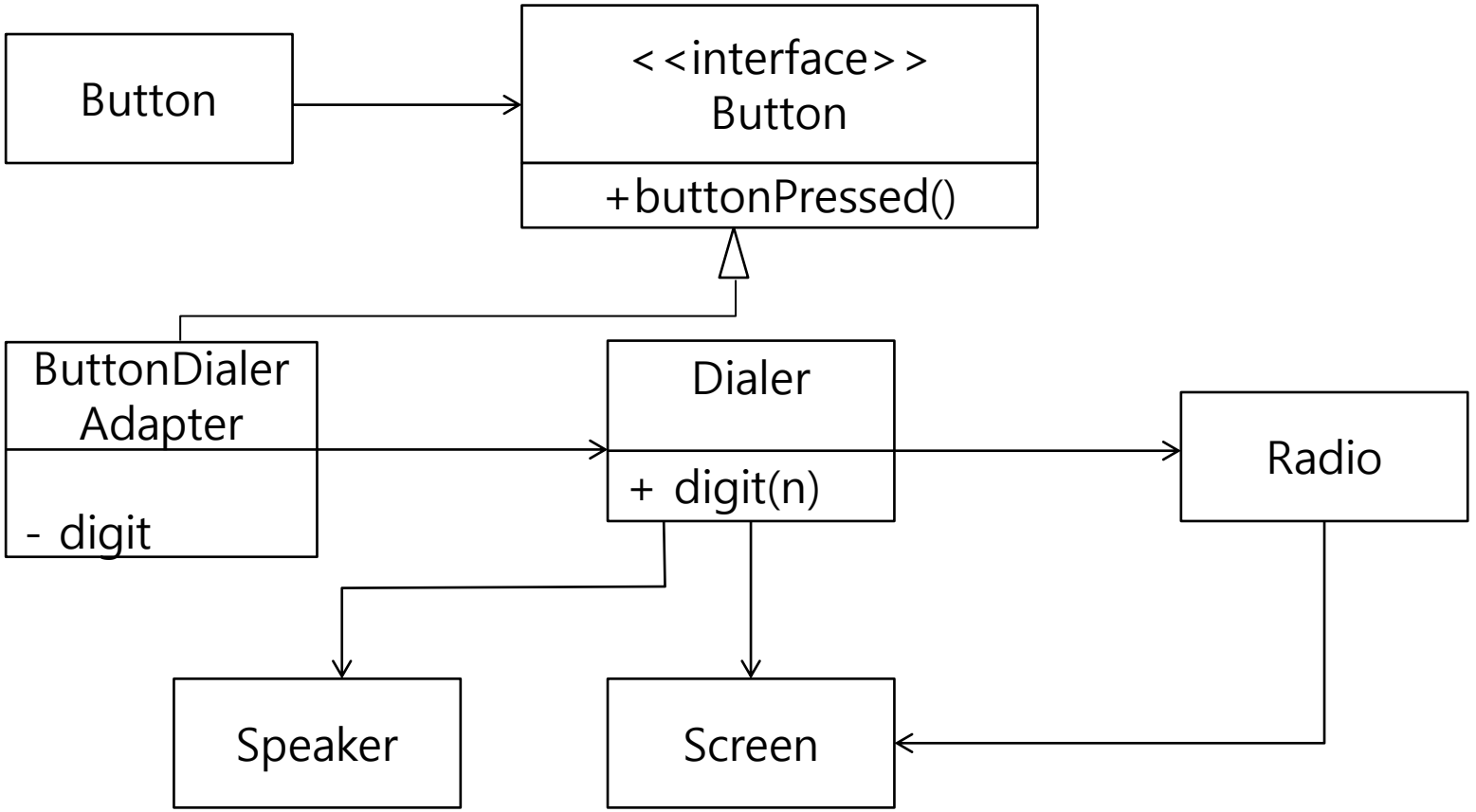


이 클래스 다이어그램에서 협력에 참여하는 객체마다 클래스가 하나씩 생기며, 협력의 연결(link)마다 연관이 하나씩 생긴다.

다이어그램으로 작업하기

① Button을 Dialer에서 분리하기

② Button과 Dialer를 이어 주는 어댑터 만들기



다이어그램으로 작업하기

◆ Button과 Dialer의 의존관계

Button 소스코드가 Dialer 소스코드를 언급하도록 하고 싶지 않다.

Button과 Dialer 사이에 인터페이스를 만든다

Button은 식별자 토큰을 하나씩 가진다.

Button 클래스는 자기가 눌렸다는 사실을 감지하면, ButtonListener 인터페이스의 ButtonPressed 메서드를 호출하면서 자기 식별자 토큰을 인자로 넘긴다.

이렇게 하면 Button이 Dialer에 의존하지 않게 할 수 있다.

◆ Dialer가 Button에 대하여...

Dialer가 ButtonListener에서 입력이 들어올것을 기대해야 하는가?

ButtonDialerAdapter는 ButtonListener 인터페이스를 구현한다.

이 어댑터의 buttonPressed 메서드가 호출될 때 이 어댑터는 Dialer에 digit(n) 메시지를 보낸다. Dialer에 전달할 숫자(n)는 어댑터가 기억하고 있다.

다이어그램으로 작업하기

- 클래스로 구현하기

```
public class Button {
    private Dialer dialer;

    public Button(Dialer dialer) {
        this.dialer = dialer;
    }
}
```

```
public class ButtonDialerAdapter implements ButtonListener{
    private int digit;
    private Dialer dialer;

    public ButtonDialerAdapter(int digit, Dialer dialer) {
        this.digit = digit;
        this.dialer = dialer;
    }

    @Override
    public void buttonPressed() {
        dialer.digit(digit);
    }
}
```