

# Mermaid Tutorials

소프트웨어융합학부

노기섭 교수

(kafa46@cju.ac.kr)

# Why do we need Diagrams

## ■ 왜 우리는 그림을 그리는가?

- 가장 기본적인 정보 표현
- 사람이 가장 쉽게 이해할 수 있는 대화 수단



이집트 상형문자(이미지 출처: <https://en.wikipedia.org/wiki/Hieroglyph>)

# SW에 그림이 필요한 이유

## ■ SW 설명을 어떻게 하나?

- 사용자, 발주자, 팀원, 경영진, 다른 프로그래머 등에게 SW를 설명해야 하는 상황
- 소스코드로 설명? 화이트보드에 글로 쓰면서 설명?

```
7 string sInput;  
8 int iLength, iN;  
9 double dblTemp;  
10 bool again = true;  
11  
12 while (again) {  
13     iN = -1;  
14     again = false;  
15     getline(cin, sInput);  
16     system("cls");  
17     stringstream(sInput) >> dblTemp;  
18     iLength = sInput.length();  
19     if (iLength < 4) {  
20         again = true;  
21         continue;  
22     } else if (sInput[iLength - 3] != '.') {  
23         again = true;  
24         continue;  
25     } while (++iN < iLength) {  
26         if (isdigit(sInput[iN])) {  
27             continue;  
28         } else if (iN == (iLength - 3)) {  
29
```

이미지 출처:

[https://www.oss.kr/oss\\_guide/show/ecb8ce8-9218-41f1-a92b-f2ab537f7eeb](https://www.oss.kr/oss_guide/show/ecb8ce8-9218-41f1-a92b-f2ab537f7eeb)



이미지 출처:

<https://entertain.daum.net/tv/119093/video/378058069>

# SW를 그림으로 표현하는 방법

## ■ 그림으로 표현하는 SW

- Flowchart, Class Diagram, Sequence Diagram, ER Diagram, ...

- SW공학 UML 에서 정의

- 직관적으로 SW 설명 가능

- 가능한 도구들

- StarUML: <https://staruml.io>

- draw.io: <https://draw.io>

→ 이 외에도 수십 가지 Tool 존재

- 지원 방식

- Drag & Drop
- WYSIWYG (What You See Is What You Get)

일일이 도형을 선택하고 연결  
UI 모양을 공부  
그림 파일로 저장/문서에 붙여넣기

SW변경되면 위 작업을 반복...  
(요구하는 사람은 minor 요구,  
하지만 Diagram 작업자는 죽을 맛)

그림을 재배치 해야 하는 경우라면?  
(이런 경우는 매우 빈번하게 발생...)

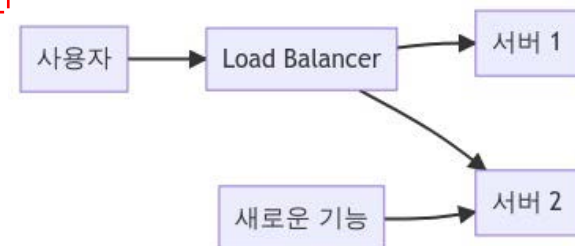
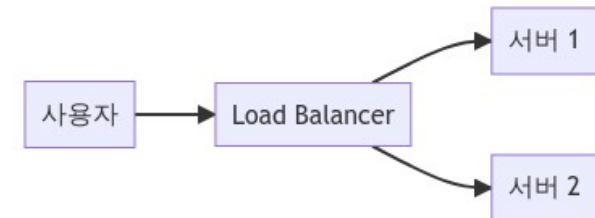
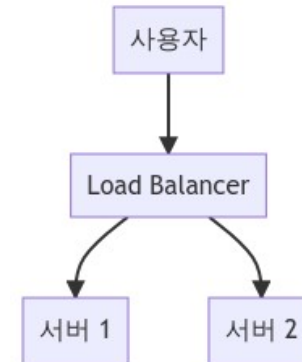
# Developer's Diagram Works

## ■ SW 개발자들이 하는 일

- 업무분석, 요구사항 분석
- 구현(코딩), 테스트, ...

- 위 관련 사항을 문서화
  - 작성 언어: HTML, Markdown 주로 사용
  - 문서 작성 시 즉시 diagram 제작/삽입
  - Diagram 자동 배치, Orientation 변경
  - 유지보수 편의성

이 모든 요구를 충족시킬 수 있는 도구 ➔ Mermaid




# Mermaid 소개

## ■ Mermaid

- Text-based
- Markdown-friendly
- Developer perspectives
- Javascript-based
- Support Extension
  - VS Code
  - Github

```
<script src="https://unpkg.com/mermaid/"></script>
<script>mermaid.initialize({startOnLoad:true});</script>

<div class="mermaid">
  graph LR
    A --- B
    B-->C[Success]
    B-->D(Fail)
</div>
```

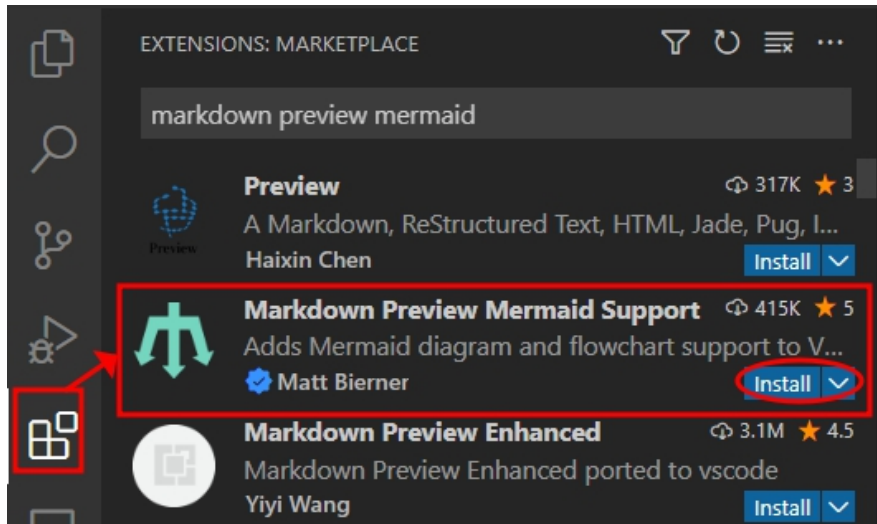


mermaid\_sample.html

```
```mermaid
graph LR
  A --- B
  B-->C[Success]
  B-->D(Fail)
```
```



# Mermaid with VS code



## 샘플 코드

```
```mermaid
flowchart
    A[설날 아침] -->|세뱃돈| B(쇼핑 가기)
    B --> C{물 사야하나?}
    C -->|선택1| D[노트북]
    C -->|선택2| E[스마트폰]
    C -->|선택3| F[자동차]
```
```

VS code:

Markdown 익스텐션 설치 후 사용

참고 문서: [https://kafa46.github.io/mermaid/docs/01\\_01\\_intro\\_to\\_mermaid.html](https://kafa46.github.io/mermaid/docs/01_01_intro_to_mermaid.html)

# Flowchart



# Elements of Flowchart

## ■ 노드(Node)

- [https://kafa46.github.io/mermaid/docs/02\\_03\\_flowchart\\_node.html](https://kafa46.github.io/mermaid/docs/02_03_flowchart_node.html)

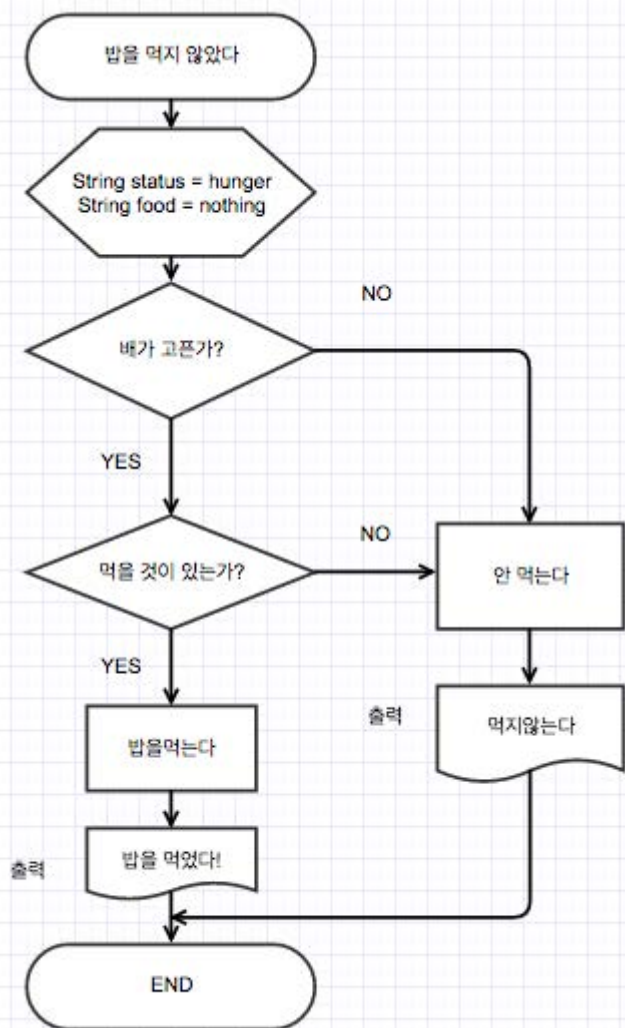
## ■ Graph

- [https://kafa46.github.io/mermaid/docs/02\\_04\\_flowchart\\_graph.html](https://kafa46.github.io/mermaid/docs/02_04_flowchart_graph.html)

## ■ 노드와 노드 연결

- [https://kafa46.github.io/mermaid/docs/02\\_05\\_flowchart\\_links.html](https://kafa46.github.io/mermaid/docs/02_05_flowchart_links.html)

# 실습



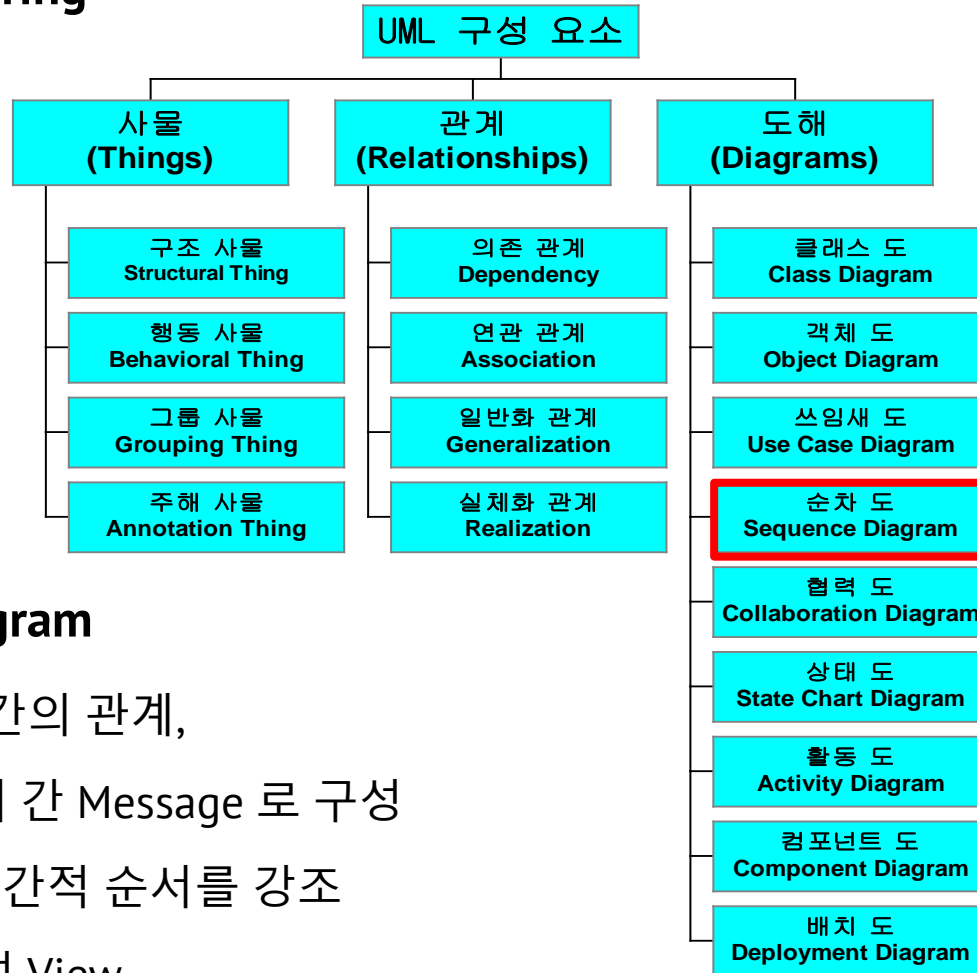
```

```mermaid
flowchart TB
  %% 좌우 배치일 경우 `LR` 적용
  a([밥을 먹지 않았다])
  b{{String status = hunger <br> String = nothing}}
  c{배가 고픈가?}
  d{먹을 것이 있는가?}
  e[밥을 먹는다]
  f[/밥을 먹었다/]
  g([End])
  a --> b --> c -->|YES| d -->|YES| e --> f --> g
  c -- NO --> h[안 먹는다]
  d -- NO --> h
  h --> i[/먹지 않는다/]
  i --> g
  ```
  
```

# Sequence Diagram

# Sequence Diagram - 이론

## ■ In SW Engineering



## ■ Sequence Diagram

- 객체와 객체 간의 관계,  
그리고 객체 간 Message 로 구성
- Message의 시간적 순서를 강조
- 시스템의 동적 View

# Elements of Sequence Diagram

## ■ 기본 문법

- [https://kafa46.github.io/mermaid/docs/03\\_01\\_sequence\\_basic\\_syntax.html](https://kafa46.github.io/mermaid/docs/03_01_sequence_basic_syntax.html)

## ■ 메시지

- [https://kafa46.github.io/mermaid/docs/03\\_02\\_sequence\\_messages.html](https://kafa46.github.io/mermaid/docs/03_02_sequence_messages.html)

## ■ Activation 표기

- [https://kafa46.github.io/mermaid/docs/03\\_03\\_sequence\\_activations.html](https://kafa46.github.io/mermaid/docs/03_03_sequence_activations.html)

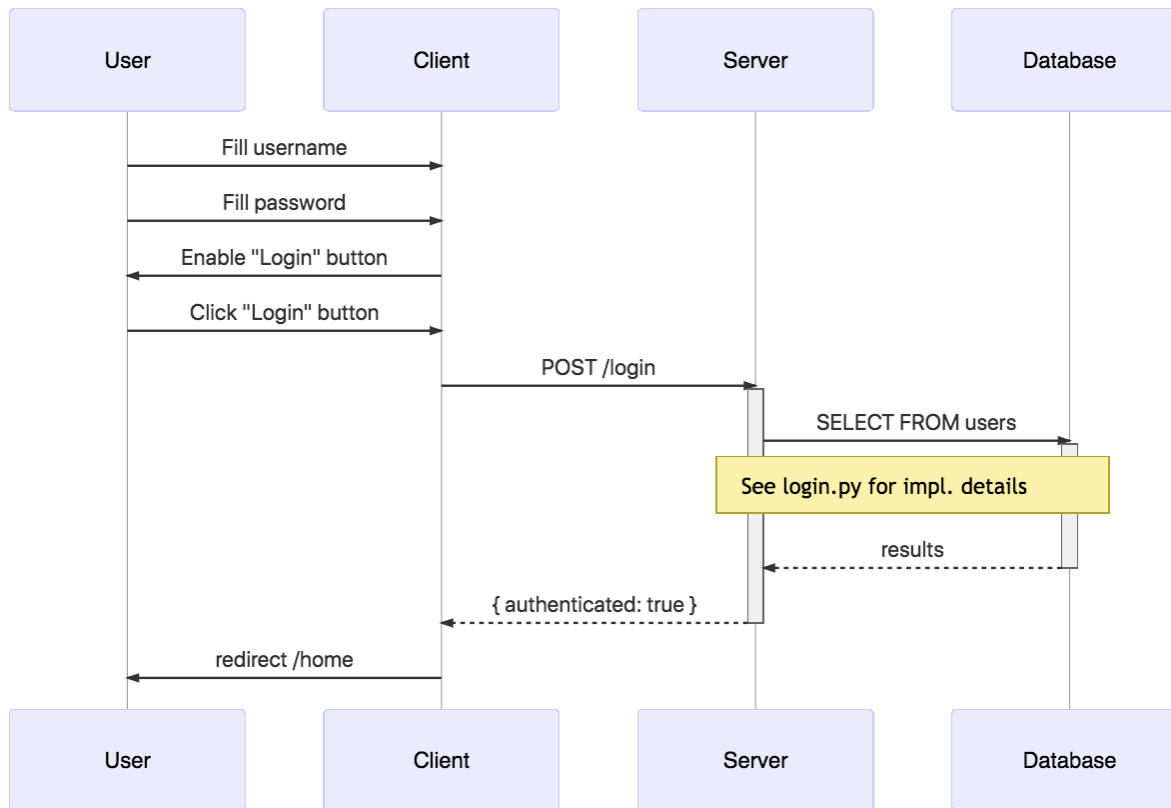
## ■ 반복(loop) 표현

- [https://kafa46.github.io/mermaid/docs/03\\_05\\_sequence\\_loops.html](https://kafa46.github.io/mermaid/docs/03_05_sequence_loops.html)

## ■ 병렬(parallel) 경로

- [https://kafa46.github.io/mermaid/docs/03\\_07\\_sequence\\_parallel.html](https://kafa46.github.io/mermaid/docs/03_07_sequence_parallel.html)

# 실습



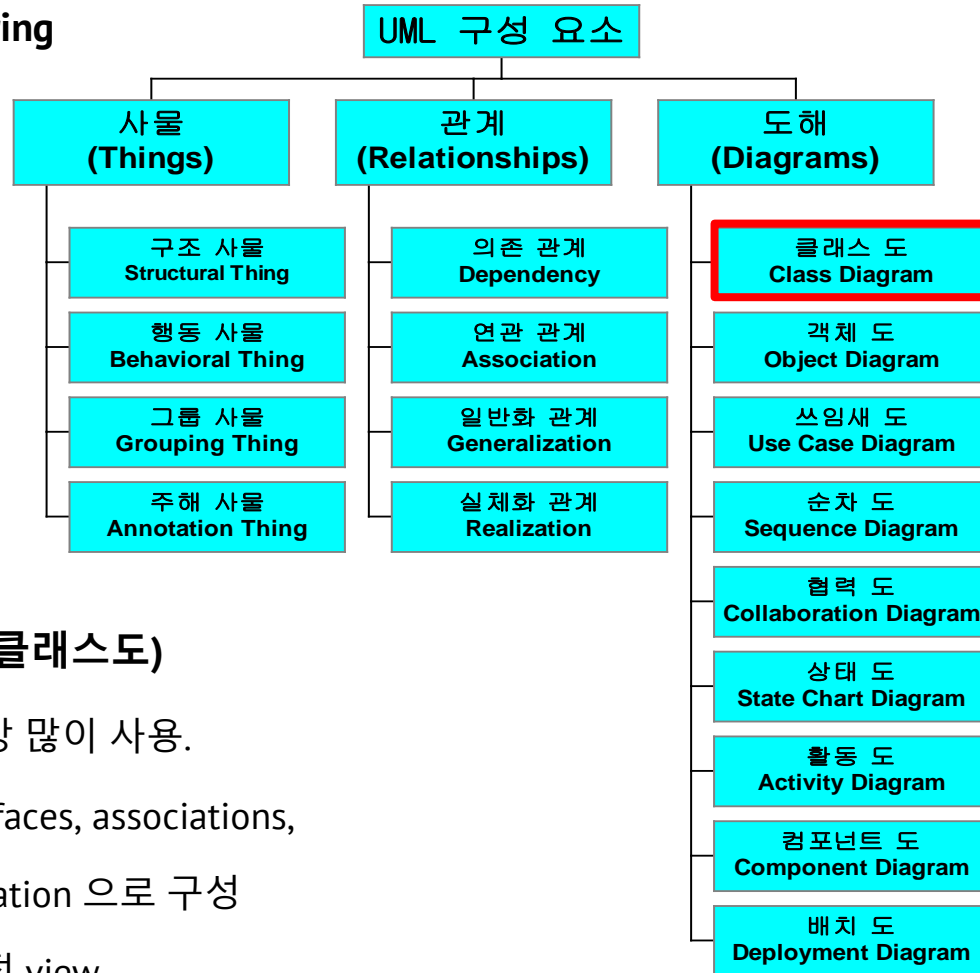
```
``mermaid
sequenceDiagram
    participant U as User
    participant C as Client
    participant S as Server
    participant DB as Database

    U ->> C : Fill username
    U ->> C : Fill password
    C ->> U : Enable "Login" button
    U ->> C : Click "Login" button
    C ->>+ S : POST /login
    S ->>+ DB : SELECT FROM users
    Note over S,DB: See login.py for
    impl. details
    DB -->>- S : results
    S -->>- C : { authenticated: true }
    C ->> U: redirect /home
    ``
```

# Class Diagram

# Class Diagram - 이론

## ■ In SW Engineering



## ■ Class diagram(클래스도)

- UML에서 가장 많이 사용.
- Classes, interfaces, associations, collaboration 으로 구성
- 시스템의 정적 view



## ■ 구성 요소

- [https://kafa46.github.io/mermaid/docs/04\\_01\\_class\\_defining\\_components.html](https://kafa46.github.io/mermaid/docs/04_01_class_defining_components.html)

## ■ 다중성

- [https://kafa46.github.io/mermaid/docs/04\\_02\\_class\\_cardinality.html](https://kafa46.github.io/mermaid/docs/04_02_class_cardinality.html)

## ■ Annotations & Comments

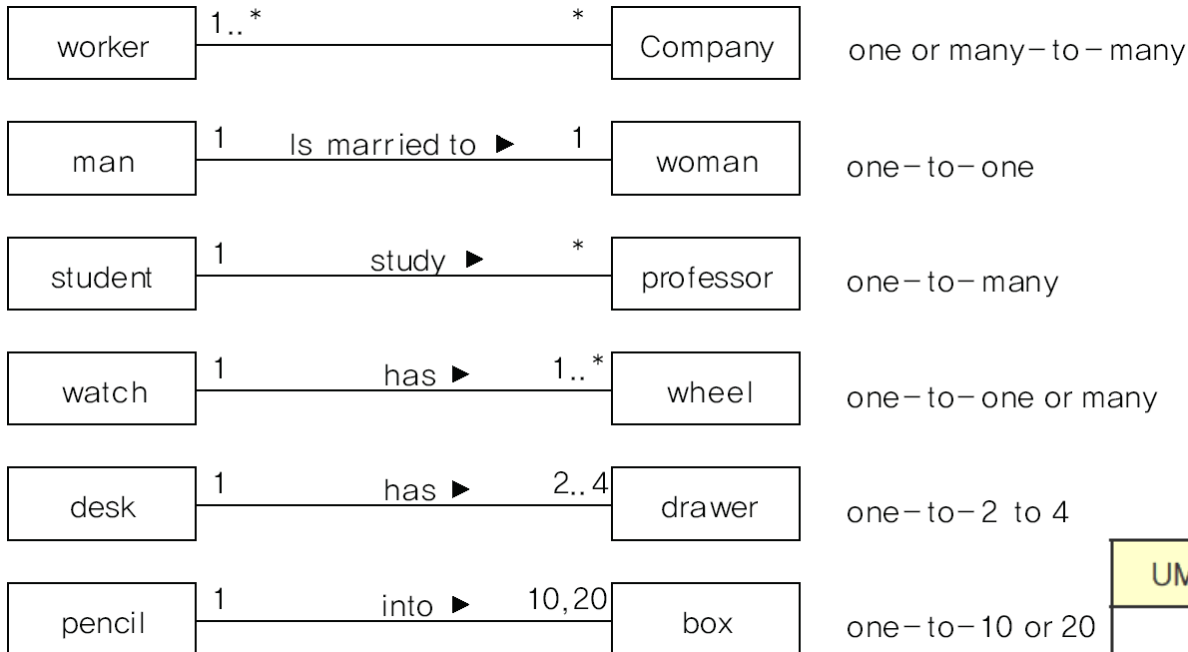
- [https://kafa46.github.io/mermaid/docs/04\\_03\\_class\\_annotations.html](https://kafa46.github.io/mermaid/docs/04_03_class_annotations.html)

## ■ 클래스 방향 설정

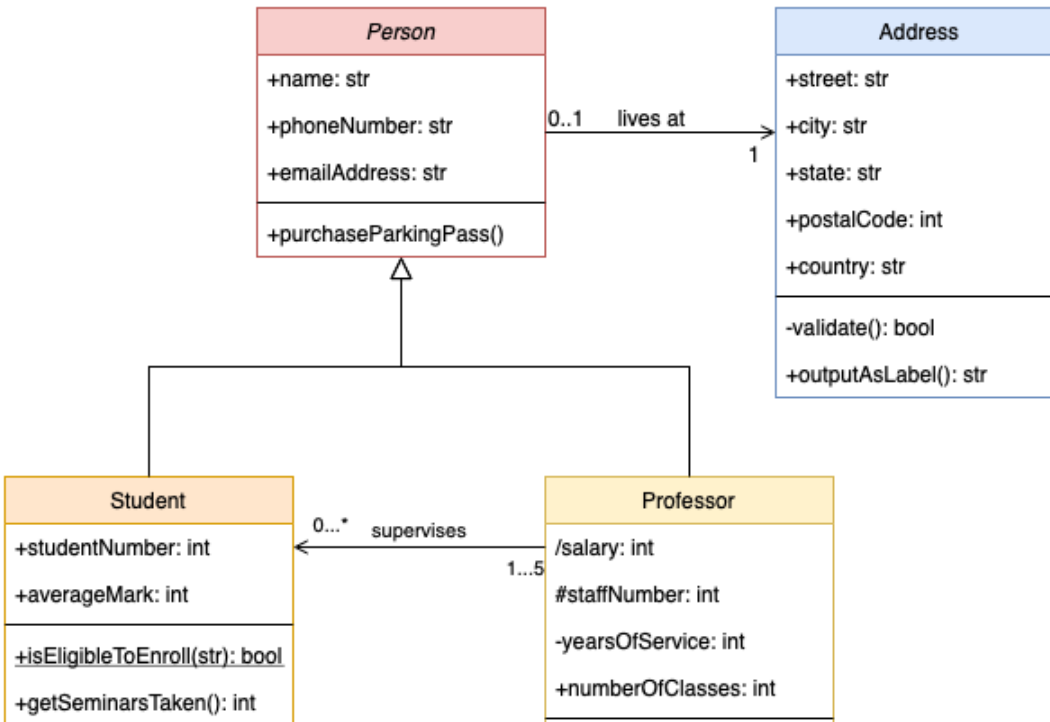
- [https://kafa46.github.io/mermaid/docs/04\\_04\\_class\\_directions.html](https://kafa46.github.io/mermaid/docs/04_04_class_directions.html)

# 클래스 다이어그램 - 다중성

## ■ 다중성 표현



| UML 다중성 | 의미          |
|---------|-------------|
| 1       | 정확히 1       |
| 0..1    | 0 이거나, 혹은 1 |
| *       | 무제한(0 포함)   |
| 1..*    | 적어도 하나 이상   |
| 2..6    | 2 부터 6 까지   |
| 2, 4    | 2 이거나, 혹은 4 |



```

classDiagram
    direction BT %% TB, BT, RL, LR 모두 가능

    class Person{
        <<abstract>>
        +name: str
        +phoneNumber: str
        +emailAddress: str
        +purchaseParkingPass()
    }

    class Student{
        +studentNumber: int
        +averageMark: int
        +isEligibleToEnroll(str) bool
        +getSeminarsTaken():int
    }

    class Professor{
        -salary: int
        #staffNumber: int
        -yearsOfService: int
        +numberOfClasses: int
    }

    class Address{
        +street: str
        +city: str
        +state: str
        +postalCode: int
        +country: str
        -validate() bool
        +outputAsLabel() str
    }

    Person <|-- Student
    Person <|-- Professor
    Student "0..*" <-- "1..5" Professor : supervise
    Person "0..1" --> "1" Address : lives in
    ...
    
```

# ER Diagram

# Elements of ER Diagram

## ■ ER (Entity-Relationship) 다이어그램

데이터베이스에서 데이터, 데이터 사이의 조건 및 관계, 제약사항들을 표현

(참고: [https://kafa46.github.io/mermaid/docs/05\\_00\\_er\\_diagram\\_intro.html](https://kafa46.github.io/mermaid/docs/05_00_er_diagram_intro.html))

- 개체(entity)

- 시스템을 구성하는 객체, 사람, 이벤트, 장소 등을 의미. 단수형 명사를 사용

- 관계(relationship)

- 개체와 개체가 어떻게 상호 작용하는지를 의미함

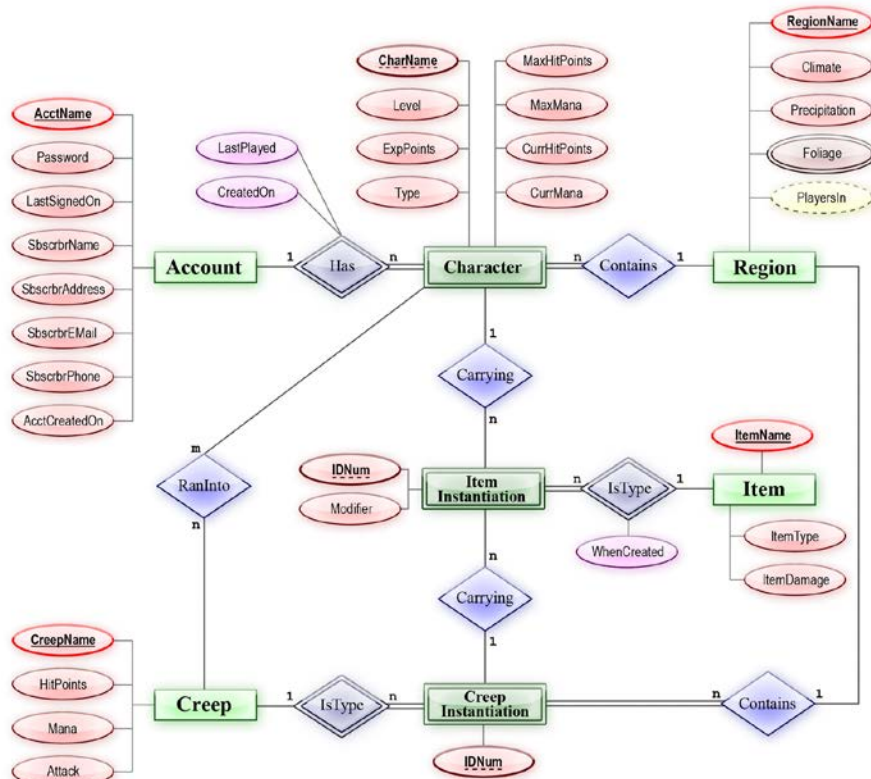
- 속성(attribute)

- 개체, 관계, 또는 다른 속성의 구체적 특성을 표현하며, 속성 이름에 밑줄이 있으면 Primary Key를 의미함.

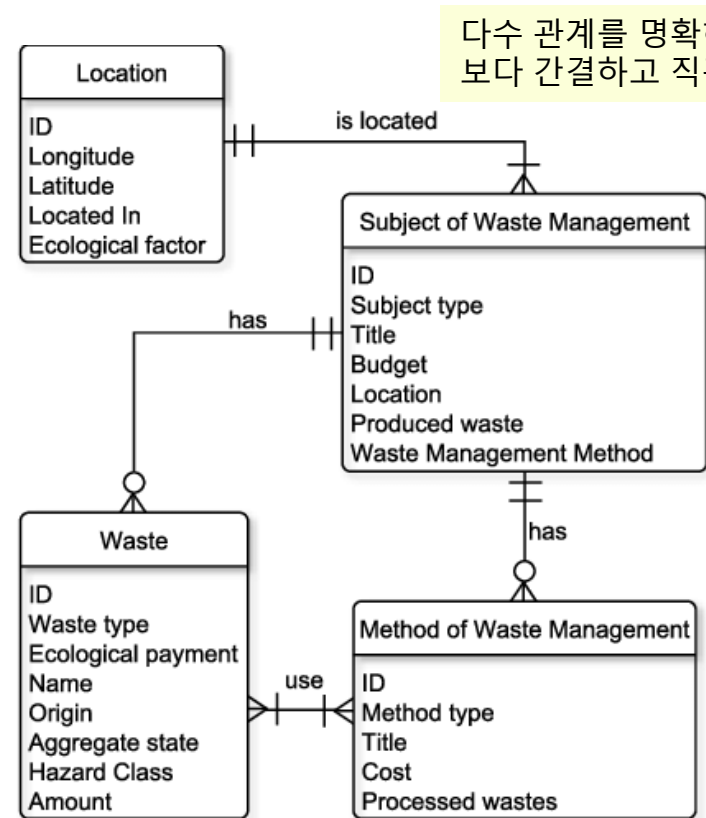
- 연결선(line)

- 개체, 관계, 속성을 연결하는 선

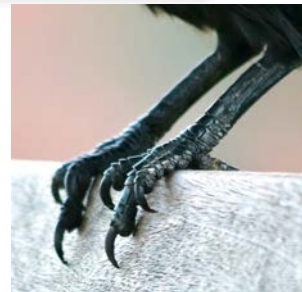
# ER Diagram 종류



전통적 방법



다수 관계를 명확히 표현  
보다 간결하고 직관적 표현



Crow Foot 방법

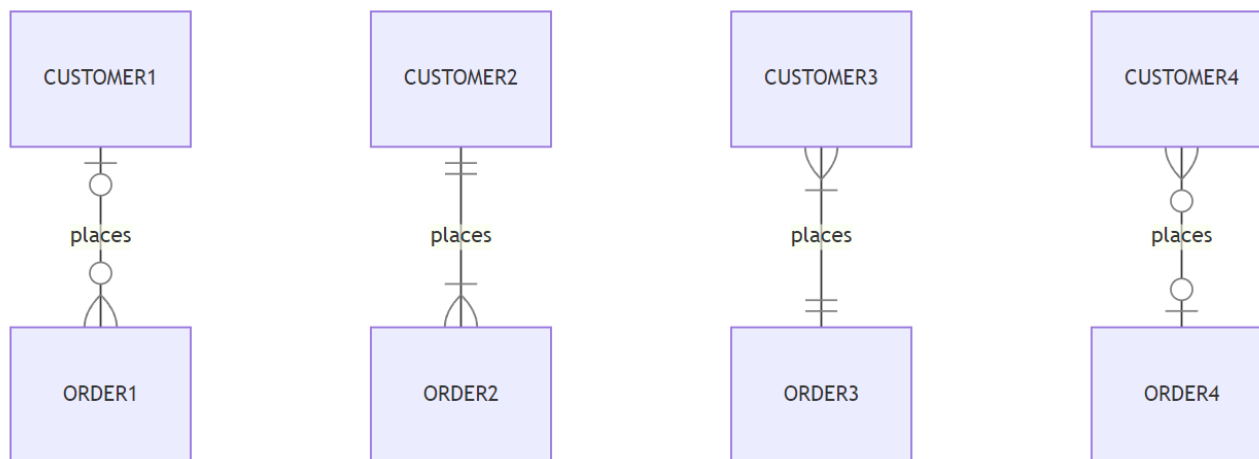
# 개체와 관계

| 표현                | 설명                            |
|-------------------|-------------------------------|
| 고리(ring)          | 숫자 0을 의미함                     |
| 실선(dash)          | 숫자 1을 의미함                     |
| 까마귀 발 (crow foot) | 다수(many) 또는 무한(infinite)을 의미함 |
| 고리와 실선            | 0 또는 1                        |
| 실선과 실선            | 정확히 1                         |
| 고리와 까마귀 발         | 0개 이상                         |
| 실선과 까마귀 발         | 1개 이상                         |

```

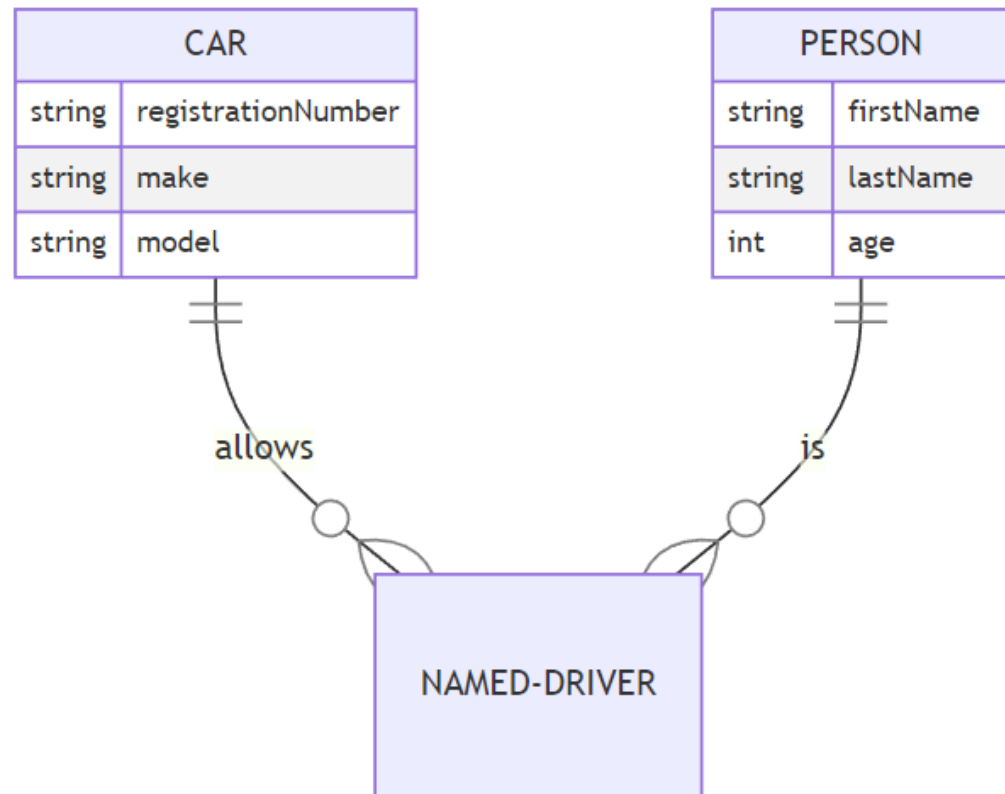
mermaid
erDiagram
    CUSTOMER ||--o{ ORDER : places
    
```

| 관계선 왼쪽 | 관계선 오른쪽 | 의미          |
|--------|---------|-------------|
| o      | o       | 0 또는 1      |
|        |         | 정확히 1       |
| }o     | o{      | 0이상 (무한대까지) |
| }      | {       | 1 이상(무한대까지) |



# 속성(attributes) 설정

```
```mermaid
erDiagram
    CAR ||--o{ NAMED-DRIVER : allows
    CAR {
        string registrationNumber
        string make
        string model
    }
    PERSON ||--o{ NAMED-DRIVER : is
    PERSON {
        string firstName
        string lastName
        int age
    }
}
```

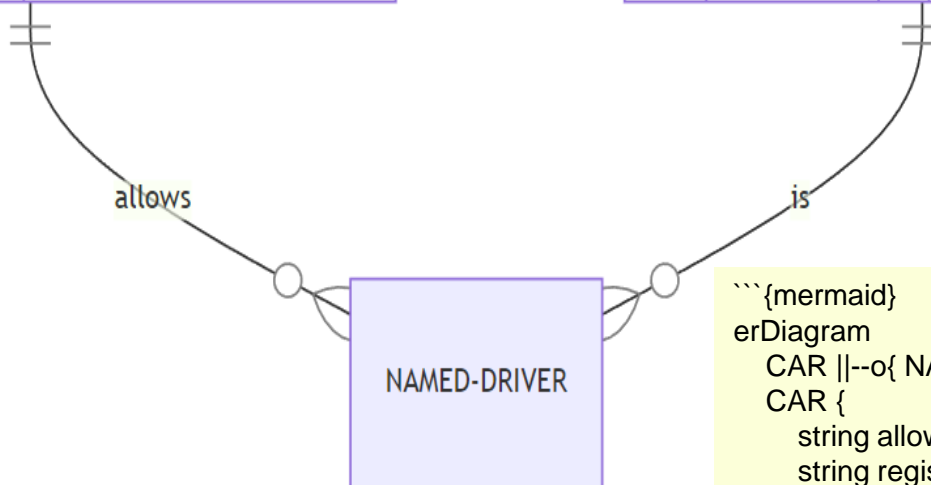




# Key와 주석(Comments)

CAR			
string	allowedDriver	FK	The license of the allowed driver
string	registrationNumber		
string	make		
string	model		

PERSON			
string	driversLicense	PK	The drive license number
string	firstName		
string	lastName		
int	age		



```
```{mermaid}
erDiagram
    CAR ||--o{ NAMED-DRIVER : allows
    CAR {
        string allowedDriver FK "The license of the allowed driver"
        string registrationNumber
        string make
        string model
    }
    PERSON ||--o{ NAMED-DRIVER : is
    PERSON {
        string driversLicense PK "The driver license number"
        string firstName
        string lastName
        int age
    }
    ```
```

소프트웨어  
꼰대 강의!



수고하셨습니다 ..^..