[ 클래스 다이어그램 ]

1. 클래스 다이어그램의 구성요소 정리

[ 클래스 ]

|  |  |
| --- | --- |
| 형식 | 설명 |
| Class Name  - attributeName : type  + operationName() : type | 보통 이름, 멤버 변수, 멤버 함수의 3구역으로 나누어서 표기함. 멤버 변수와 멤버 함수는 생략 가능, 이름은 생략 불가 |

[ 멤버 변수 ]

|  |  |
| --- | --- |
| 형식 | |
| + AttributeName : TypeName [\*] | |
| 표시 여부 | + : Public  - : Private  # : Protected  ~ : Package |
| 형식이 존재 하지 않음 | : TypeName 생략 |
| 다중성 | [\*] (단, 1이면 생략) |
| final 키워드 사용 상수 | { readOnly } |
| static | 해당 변수 밑에 밑줄 |

[ 멤버 함수 ]

|  |  |
| --- | --- |
| 형식 | |
| + OperationName(parameter1 : Type 1[\*], …) : ReturnType [\*] | |
| 표시 여부 | + : Public  - : Private  # : Protected  ~ : Package |
| virtual 선언 | 1) *이탤릭체 표기*  2) ReturnType 뒤에 { abstract } 표기 |
| 반환 형식 | : ReturnType 정의되어 있지 않을 경우 생략 |
| 다중성 | [\*] (단, 1이면 생략) |
| static | 해당 함수 밑에 밑줄 |

[ 다중성 ]

하나의 인스턴스에 연관된 다른 쪽 클래스의 가능한 인스턴스의 수 의미

|  |  |
| --- | --- |
| 다중성 | \* == : 무제한  1, 2, 4 : 1, 2, 4  1..\* : 1 이상  5..10 : 5 ~ 10  1,2…4 : 1, 2 ~ 4 |

[ 스테레오 타입 ]

UML 제공 요소 이외의 추가적인 확장 요소 표기. 길러멧 사이에 작성. 이름 부분에 작성됨.

|  |  |
| --- | --- |
| 순수 가상 클래스 | ≪ interface ≫ |
| 추상 클래스 | ≪ abstract ≫, { abstract } |

[ 관계 ]

|  |  |  |
| --- | --- | --- |
| 형식 | 이름 | 설명 |
| https://upload.wikimedia.org/wikipedia/commons/thumb/9/93/Uml_classes_en.svg/300px-Uml_classes_en.svg.png  https://upload.wikimedia.org/wikipedia/commons/thumb/9/93/Uml_classes_en.svg/300px-Uml_classes_en.svg.png | Association (연관) | 한 클래스가 다른 클래스에서 제공하는 기능 사용(멤버 변수). 방향성이 없는 연관은 실선으로, 방향성이 있는 연관은 열린 화살표가 있는 실선으로 표기 |
| https://upload.wikimedia.org/wikipedia/commons/thumb/9/93/Uml_classes_en.svg/300px-Uml_classes_en.svg.png | Inheritance / Generalization  (상속 / 일반화) | 부모 클래스와 자식 클래스 간의 상속 관계. 실선과 비어있는 삼각형을 부모 클래스로 연결 |
| https://upload.wikimedia.org/wikipedia/commons/thumb/9/93/Uml_classes_en.svg/300px-Uml_classes_en.svg.png | Realization / Implementation  (실현 / 구현) | 인터페이스의 함수를 오버라이딩하여 실제 기능으로 구현한 관계. 점선과 비어있는 삼각형을 인터페이스 쪽으로 연결 |
| https://upload.wikimedia.org/wikipedia/commons/thumb/9/93/Uml_classes_en.svg/300px-Uml_classes_en.svg.png | Dependency  (의존, 종속) | 서버와 클라이언트의 관계처럼 한 요소의 정의를 변경하면 다른 요소가 변경될 수 있는 두 요소 사이에 존재함. 지역 변수나 함수의 매개변수로 사용됨. 점선과 열린 화살표로 표기 |
| https://upload.wikimedia.org/wikipedia/commons/thumb/9/93/Uml_classes_en.svg/300px-Uml_classes_en.svg.png | Aggregation  (집합) | 관계의 일부 또는 전체를 나타내는 연관(has). 클래스가 다른 클래스에서 컨테이너 등으로 포함되어 있고, 수명 주기가 다른 클래스(외부 생성, 소멸). 실선과 빈 마름모로 표기 |
| https://upload.wikimedia.org/wikipedia/commons/thumb/9/93/Uml_classes_en.svg/300px-Uml_classes_en.svg.png | Composition  (구성) | 관계의 일부 또는 전체를 나타내는 연관(own). 클래스가 다른 클래스에서 컨테이너 등으로 포함되어 있고, 수명 주기가 같은 클래스(내부 생성, 소멸). 실선과 채워진 마름모로 표기 |

+) 클래스 다이어그램 작성시 주의사항

- 클래스명 지을 때 앞에 대문자 C or 프로젝트명 or 아예 안붙이는게 나을수도 있음

- 클래스 구현시 실제모습에 기반하여 작성하는 것도 좋음

ex) 경기에서 심판의 위치는 코트 안 => 코트 클래스에 위치 or 관련되게 작성될 수 있음

- 컨텐츠는 싱글턴 쓰지 않는 것이 좋음

-- 전역변수와 같은 맥락

-- 관리자에서만 or 거의 사용하지 않는 것이 좋음

-- 시스템이나 명확한 곳에서만 사용

ex) Input Manager, Sound Manager(래핑해서 사용될 수 있음)

- 클래스 다이어그램을 그릴때, 연관 관계 고려시 사람 내부에 소유하고 있는가로 판단하면 쉬움

ex) 사람이 빌딩을 내부에 가지고 있음(x)

사람이 빌딩 주소/아이디를 내부에 가지고 있음(o)

[ 이름 ]

[ 소제목 ]

- 내용

[ 이름 ]

[ 소제목 ]

- 내용