UML(Unified Modeling Language)

소프트웨어 공학에서 사용되는 표준화된 범용 모델링 언어로 소프트웨어 개념을 다이어그램으로 그리기 위해 사용하는 시각적인 표기법

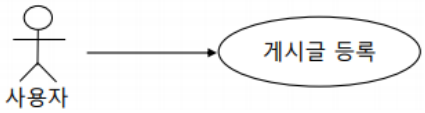
1P 모두 볼것!!

유스케이스 다이어그램(Usecase Diagram)

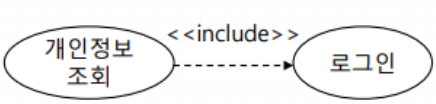
Actor와 시스템이 수행하는 활동간의 관계를 표시하며, 시스템의 기능적인 요구사항을 설명하기 위한 도구

유스케이스 다이어그램 관계

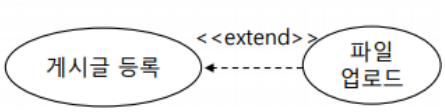
- **연관 관계**: 액터와 유스케이스 간의 상호 작용 관계



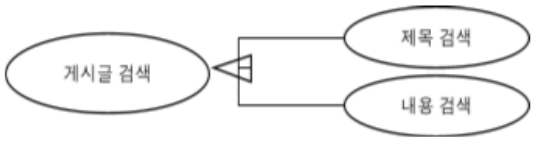
- **포함 관계**: 유스케이스를 실행하 기 위하여
  반드시 선행 실행되어야 하는 유스케이스와의 관계



- **확장 관계**: 유스케이스를 실행함으로써
  선택적으로 실행되는 유스케이스와의 관계



- **일반화 관계** : 유사한 유스케이스 또는 액터들을 모아
  그들을 추상화하여
  유스케이스/액터를 연결시켜 그룹핑하는 것



클래스 다이어그램(Class Diagram)

정적(구조) 다이어그램으로 UML모델링에서 가장 일반적으로 사용됨.

시스템의 구조와 구조 간 상호 관계를 나타내며

시스템의 논리적 및 물리적 구성요소 설계 시 주로 활용

클래스 다이어그램 관계

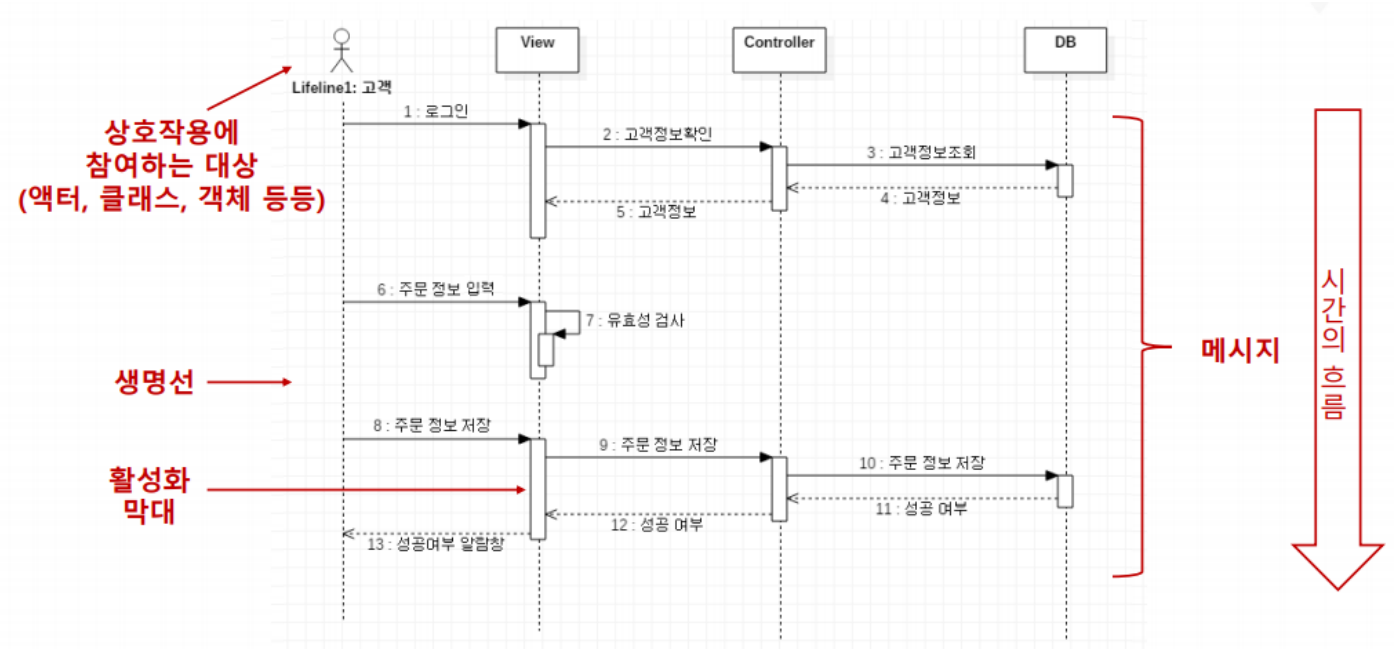
- **의존 관계(Dependency)** : 클래스간의 지역변수에서 참조하는 관계를 의미함

- **일반화 관계(Generalization)** : 부모 클래스 - 자식 클래스 상속 관계(extends)

- **실체화 관계(Realization)** : 부모 인터페이스 - 자식 클래스 상속 관계(implements)

시퀀스 다이어그램(Sequence Diagram)

동적(행위) 다이어그램으로 상호작용 다이어그램의 일부분이며 시스템 내부에서 동작하는 객체들 사이의 주고 받는 메시지를 시간 순서에 따라서 표현하는 도구이다.



시퀀스 다이어그램 구성

- **객체(Object)** : 객체는 위쪽의 사각형 박스 안에 밑줄 친 이름으로 표시하여, 아래쪽으로 생명선을 가지고 있다.

- **생명선(Lifeline)** : 객체에서 아래로 뻗어 나가는 연쇄선을 의미하며 시간의 흐름에 따라 발생하는 이벤트를 표시한다.

- **실행(Activation)** : 이벤트로 인해 객체의 오퍼레이션이 실행되고 있음을 나타내며 직사각형(활성화 막대)으로 표시하며, 직사각형이 길수록 수행 시간이 길다는 의미를 가진다.

- **메시지(Message)** : 객체 간 상호 작용은 메시지 교환으로 이루어지며, 다른 객체로 메시지를 전달하여 전달받은 객체의 오퍼레이션을 수행한다. 시간 수행 순서는 위에서 아래로 표시한다.

Datadase

한 조직에 필요한 정보를 여러 응용 시스템에서 공용할 수 있도록

논리적으로 연관된 데이터를 모으고 중복되는 데이터를 최소화하여 구조적으로 통합/저장해놓은 것

DBMS(DataBase Management System)

데이터베이스에서 데이터를 추출, 조작, 정의, 제어 등을 할 수 있게 해주는 데이터베이스 전용 관리 프로그램.

JDBC(Java Database Connectivity)

자바 언어에서 DB에 접근할 수 있게 해주는 Programming API. (java.sql 패키지)

Java - DBMS연동에 필요한 메소드를 Connection 인터페이스를 통해 제공함.

웹서버(Web Server)

웹 브라우저(Web Client)의 요청을 받아 html 파일이나 이미지(jpg, gif 등),

자바스크립트의 정적인 콘텐츠를 제공.

ex) Apache Web Server

웹 애플리케이션 서버(WAS, Web Application Server)

서버단(Server Level)에서 애플리케이션이 동작할 수 있는 환경을 제공하고,

안정적인 트랜잭션 처리 및 관리, 다른 시스템 간의 애플리케이션 연동을 지원하는

동적 서버 콘텐츠를 수행, 제공.

ex) Tomcat

MVC Model1 패턴

- JSP에서 로직과 출력을 처리하는 패턴으로

요청 시 JSP에서 비즈니스 로직을 직접 처리하고 그 결과를 바로 클라이언트에게 출력(응답)함.

- 장점 : 구조가 단순하고 JSP 페이지 하나에 작성하기 때문에 구현이 쉬움.

- 단점 :

1) 비즈니스 로직과 출력을 위한 VIEW가 하나로 합쳐져 있어 JSP가 복잡해짐.

2) 프론트엔드와 백엔드 코드가 같이 작성되기 때문에 협업에 좋지 않음.

MVC Model2 패턴

- 역할 별 작업이 가능하도록 분담하는 설계 패턴으로

Model은 비즈니스 로직 (Service, DAO, VO),

View는 클라이언트의 요청과 요청한 정보의 응답 결과를 보여주는 화면(JSP, HTML),

Controller는 클라이언트의 요청을 전달받아 응답처리를 위한 Service를 호출하고 결과를 View에 전송한다.

- 장점 :

1) View, Controller가 분리되어 있어 구현이 쉬움.

2) 각 레이어가 독립적으로 구성되어 있어 개발 및 유지 보수에 좋음

- 단점 : 초기 구조 설계 단계에서 많은 시간이 걸림.