모델2 기반의 MVC 패턴

목차

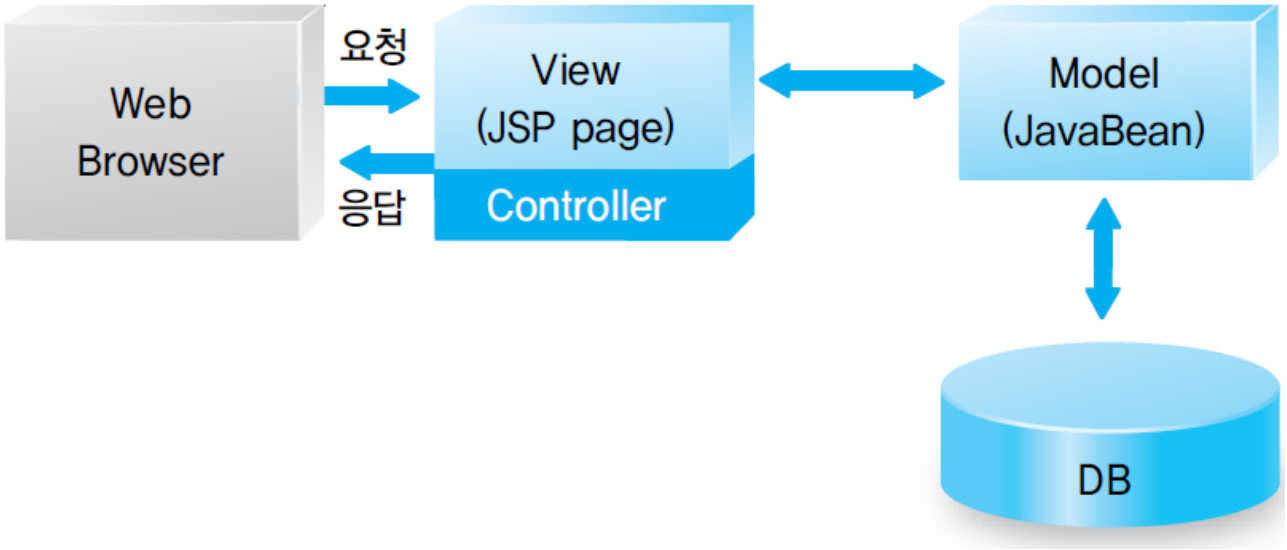
1. **모델 2와 MVC 패턴의 개요**
    1. 모델1 vs 모델2
    2. MVC 패턴(Model-View-Controller pattern)
2. **컨트롤러인 서블릿에 사용자의 요청을 명령어로 전달 - 커맨드 패턴**
    1. 요청 파라미터로 명령어를 전달하는 방법
    2. 요청 URL 자체를 명령어로 사용하는 방법

1. 모델 2와 MVC 패턴의 개요

모델 1 vs. 모델 2

모델 1 구조

- 웹 브라우저의 요청(request)을 받아들이고, 웹 브라우저에 응답(response)하는 것을 JSP페이지가 단독으로 처리하는 구조

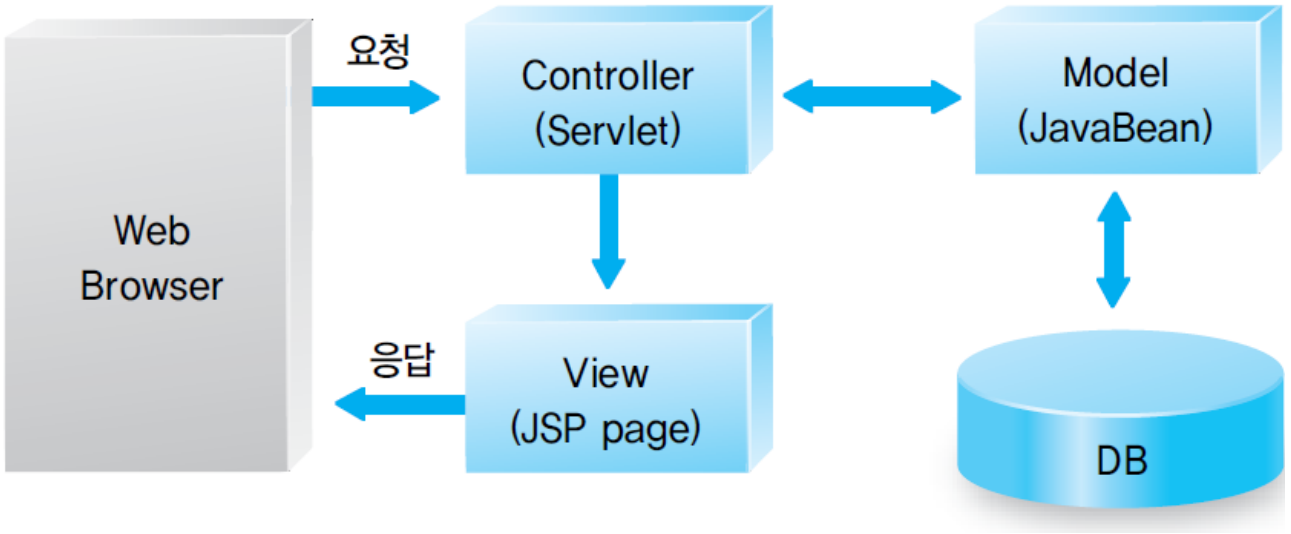


1. 모델 2와 MVC 패턴의 개요

모델 1 vs. 모델 2

모델 2 구조

- 요청(request) 처리, 데이터 접근(data access), 비즈니스 로직(business logic)을 포함하고 있는 컨트롤러와 뷰는 엄격히 구분



1. 모델 2와 MVC 패턴의 개요

- **MVC 패턴(Model-View-Controller pattern)**
  - 전통적인 GUI(Graphic User interface) 기반의 애플리케이션을 구현하기 위한 디자인 패턴
  - MVC 구조는 사용자의 입력을 받아서 입력에 대한 처리를 하고, 그 결과를 다시 사용자에게 표시하기 위한 최적화된 설계를 제시

1. 모델 2와 MVC 패턴의 개요

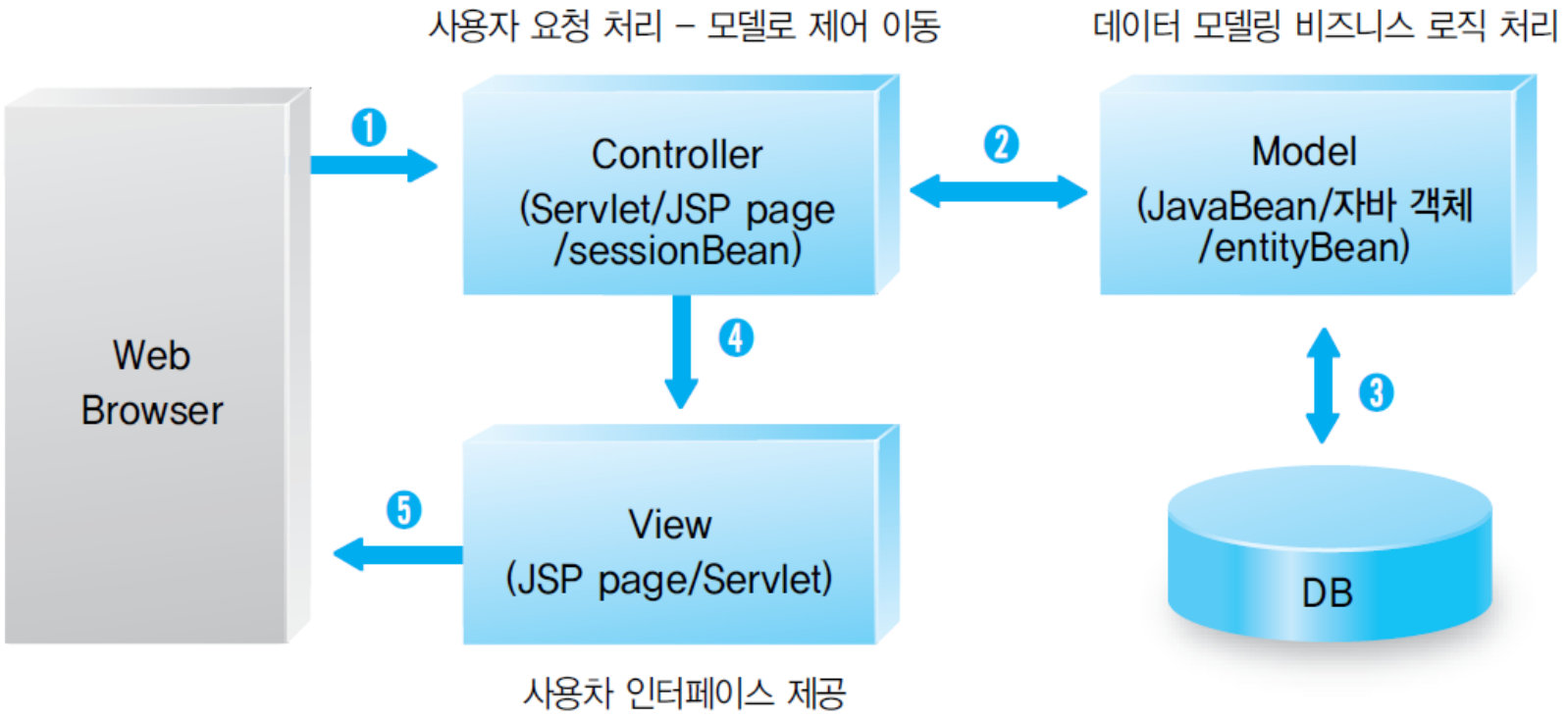
- **MVC 패턴(Model-View-Controller pattern)**
  - MVC 패턴의 요소
    - 모델(Model) : 로직을 가지는 부분
      - DB와의 연동을 통해서 데이터를 가져와 어떤 작업을 처리
      - 처리한 작업의 결과를 데이터로서 DB에 저장하는 일을 처리
      - 자바빈(JavaBean), 처리 로직인 자바 클래스(Java class)가 이에 해당

1. 모델 2와 MVC 패턴의 개요

- **MVC 패턴(Model-View-Controller pattern)**
  - MVC 패턴의 요소
    - 뷰(View) : 화면에 내용을 표시
      - 정보를 보여주는 역할만을 담당
      - JSP 페이지가 이에 해당
    - 컨트롤러(Controller) : 어플리케이션의 흐름을 제어
      - 사용자의 요청을 받아서 모델(Model)에 넘겨줌
      - 모델(Model)이 처리한 작업의 결과를 뷰(View)로 보냄
      - 서블릿(Servlet)이 이에 해당

1. 모델 2와 MVC 패턴의 개요

- **MVC 패턴(Model-View-Controller pattern)**
  - MVC 패턴의 구조



1. 모델 2와 MVC 패턴의 개요

- **MVC 패턴(Model-View-Controller pattern)**
  - MVC 패턴의 구성 요소
    - 컨트롤러(Controller) : 서블릿(Servlet)
      - 웹브라우저의 요청을 받는 진입점
      - 사용자의 요청을 받아서 요구사항을 분석 후 로직 처리를 모델로 보냄
      - 로직의 처리 결과를 모델로부터 받아서, 사용자에게 응답하기 위해 뷰로 보냄

1. 모델 2와 MVC 패턴의 개요

- **MVC 패턴(Model-View-Controller pattern)**
  - MVC 패턴의 구성 요소
    - 컨트롤러(Controller)의 작업 처리 과정

① 웹 브라우저의 요청을 받음

- 웹브라우저의 요청은 서블릿의 서비스 메소드인 doGet() 또는 doPost()메소드가 받음

② 웹 브라우저가 요구하는 작업을 분석

- 사용자가 요구한 작업에 맞는 로직이 실행되도록, 웹브라우저의 요구 작업을 분석

③ 모델을 사용해서 요청한 작업을 처리

- 요청한 작업에 해당하는 로직을 처리

1. 모델 2와 MVC 패턴의 개요

- **MVC 패턴(Model-View-Controller pattern)**
  - MVC 패턴의 구성 요소
    - 컨트롤러(Controller)의 작업 처리 과정

④ 로직 처리 결과를 request객체의 속성에 저장

- request 객체의 속성에 처리 결과를 저장
- 처리 결과는 같은 request 객체 영역에서 공유

⑤ 적당한 뷰(JSP페이지)를 선택 후 해당 뷰로 포워딩(forwarding).

- 처리 결과를 저장한 request객체를 뷰로 전달

1. 모델 2와 MVC 패턴의 개요

- **MVC 패턴(Model-View-Controller pattern)**
  - MVC 패턴의 구성 요소
    - 뷰(View) : JSP 페이지
      - 요청에 대한 응답 결과를 표시
      - JSP 페이지의 request는 컨트롤러(Controller)인 서블릿(Servlet)과 같은 객체로 공유
        - ${requestScope.result} 또는 request.getAttribute("result")와 같이 사용해서 결과를 화면에 표시

1. 모델 2와 MVC 패턴의 개요

- **MVC 패턴(Model-View-Controller pattern)**
  - MVC 패턴의 구성 요소
    - 모델(Model) : 자바빈
      - 컨트롤러(Controller)가 넘겨준 로직 처리
      - 모델(Model)의 작업 처리과정
        - ① 컨트롤러(Controller)의 요청을 받음
        - ② 모델에서 로직을 처리
        - ③ 처리한 로직의 결과를 컨트롤러(Controller)로 반환

1. 모델 2와 MVC 패턴의 개요

- **MVC 패턴(Model-View-Controller pattern)**
  - MVC 패턴의 구성 요소
    - 모델(Model) : 자바빈
      - 컨트롤러(Controller)가 넘겨준 로직 처리
      - 모델(Model)의 작업 처리과정
        - ① 컨트롤러(Controller)의 요청을 받음
        - ② 모델에서 로직을 처리
        - ③ 처리한 로직의 결과를 컨트롤러(Controller)로 반환

2. 컨트롤러인 서블릿에 사용자의 요청을 명령어로 전달 - 커맨트 패턴

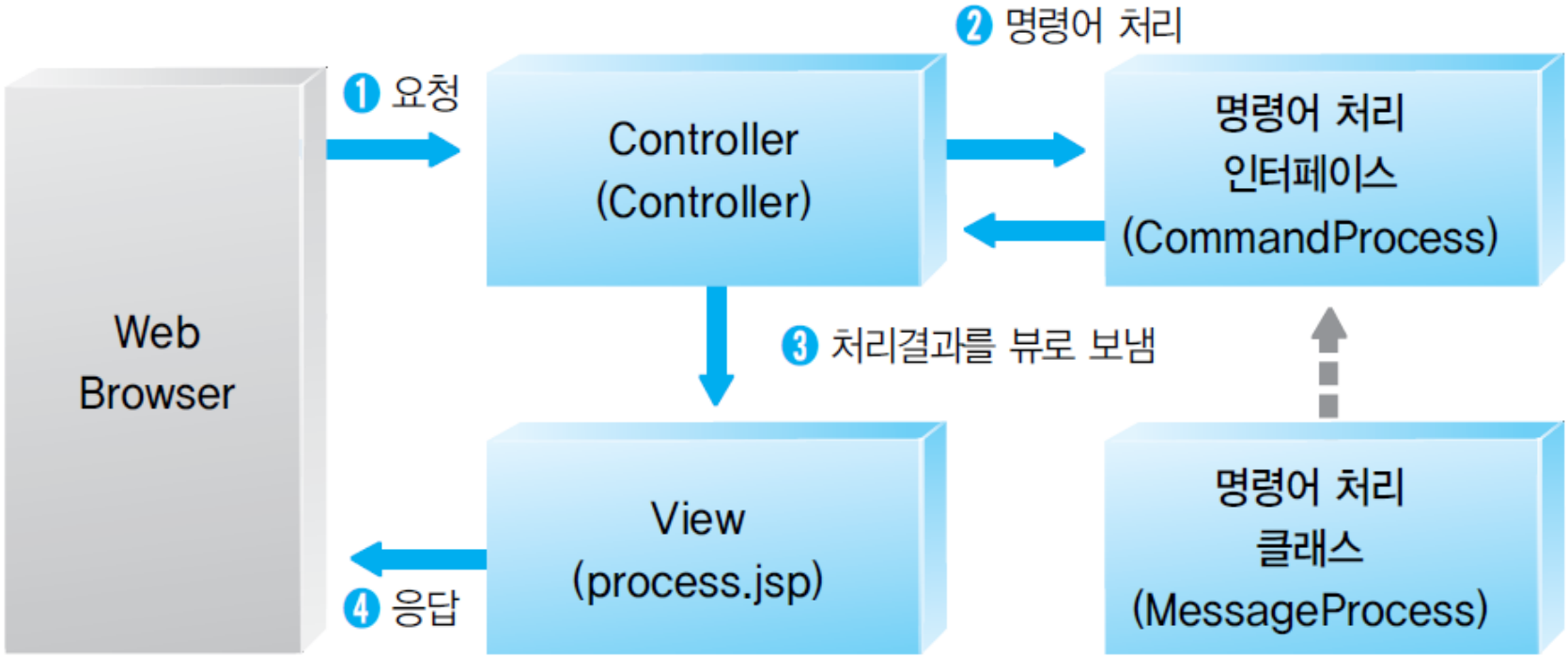
- **개요**
  - 사용자가 어떤 요청을 했는지 판단하기 위한 가장 일반적인 방법이 명령어로서 사용자의 요청을 전달
  - 명령어와 로직을 연결하는 properties 매핑 파일이 필요

2. 컨트롤러인 서블릿에 사용자의 요청을 명령어로 전달 - 커맨트 패턴

- **요청 파라미터로 명령어를 전달하는 방법**
  - 컨트롤러인 서블릿에 요청 파라미터를 정보를 덧붙여서 사용
    - http://localhost:8080/studyjsp/MessageContoller?message=aaa
  - 간편하긴 하나 명령어가 파라미터로 전달되게 되면 정보가 웹 브라우저를 통해 노출

2. 컨트롤러인 서블릿에 사용자의 요청을 명령어로 전달 - 커맨트 패턴

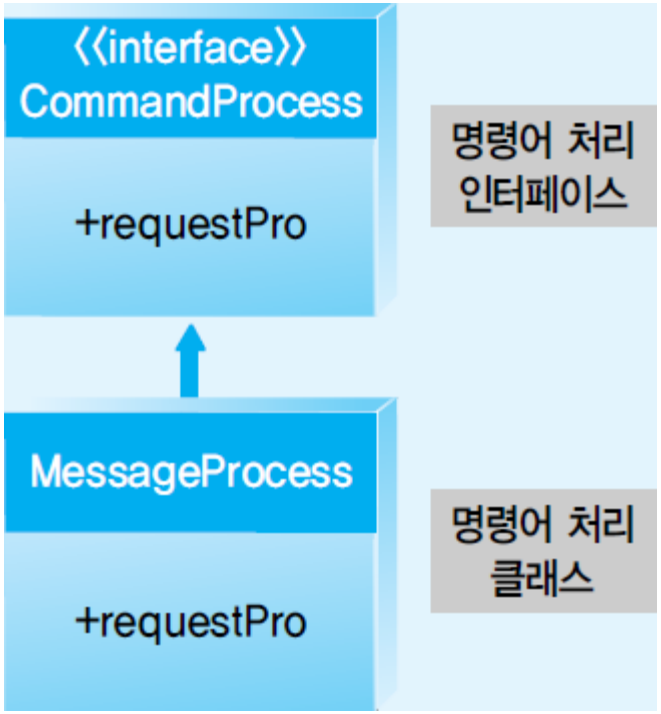
- **요청 파라미터로 명령어를 전달하는 방법**



▲ 요청 파라미터로 명령어를 전달하는 예제의 구조

2. 컨트롤러인 서블릿에 사용자의 요청을 명령어로 전달 - 커맨트 패턴

- 요청 파라미터로 명령어를 전달하는 방법



- 슈퍼 인터페이스인 Command Process와 명령어 처리 클래스인 MessageProcess 간의 관계

2. 컨트롤러인 서블릿에 사용자의 요청을 명령어로 전달 - 커맨트 패턴

- **요청 URI 자체를 명령어로 사용하는 방법**
  - 사용자가 요청한 URI 자체를 명령어로 사용하는 방법
    - http://127.0.0.1:8080/studyjsp/ch17/test.do
  - 요청되는 URI가 실제 페이지가 아니고 명령어이므로 악의적인 명령어로부터 사이트가 보호되며, 요청되는 URL이 좀 더 자연스러워진다는 장점

참고 사항 - UML 다이어그램

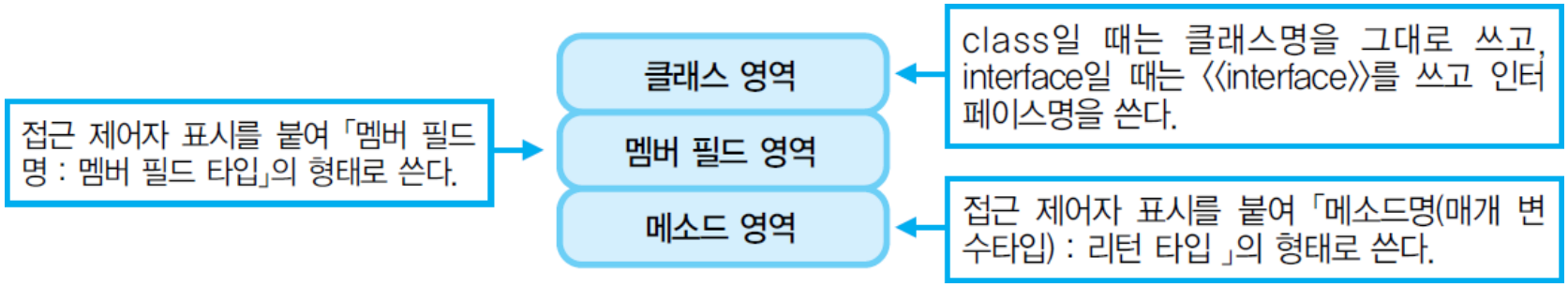
- UML(Unified Modeling Language) 다이어그램 중 하나인 클래스 다이어그램(class diagram)을 사용해서 클래스의 구조를 표시
- 클래스 다이어그램에는 하나의 클래스의 구조를 표현하는 것과 클래스간의 관계를 표현하는 것이 있음

참고 사항 - UML 다이어그램

- **하나의 클래스의 구조를 표현**
  - 다이어그램에 클래스와 클래스의 멤버인 멤버 필드, 메소드를 기술
  - 클래스 다이어그램은 사각형 안에 클래스 영역, 멤버 필드 영역, 메소드 영역이 각각 나뉘어짐
  - 멤버 필드나 메소드의 접근 제어자를 기호로 표시
    - 「+」는 public, 「-」는 private, 「#」은 protected이고, 아무 표시도 없는 것은 default 접근 제어자

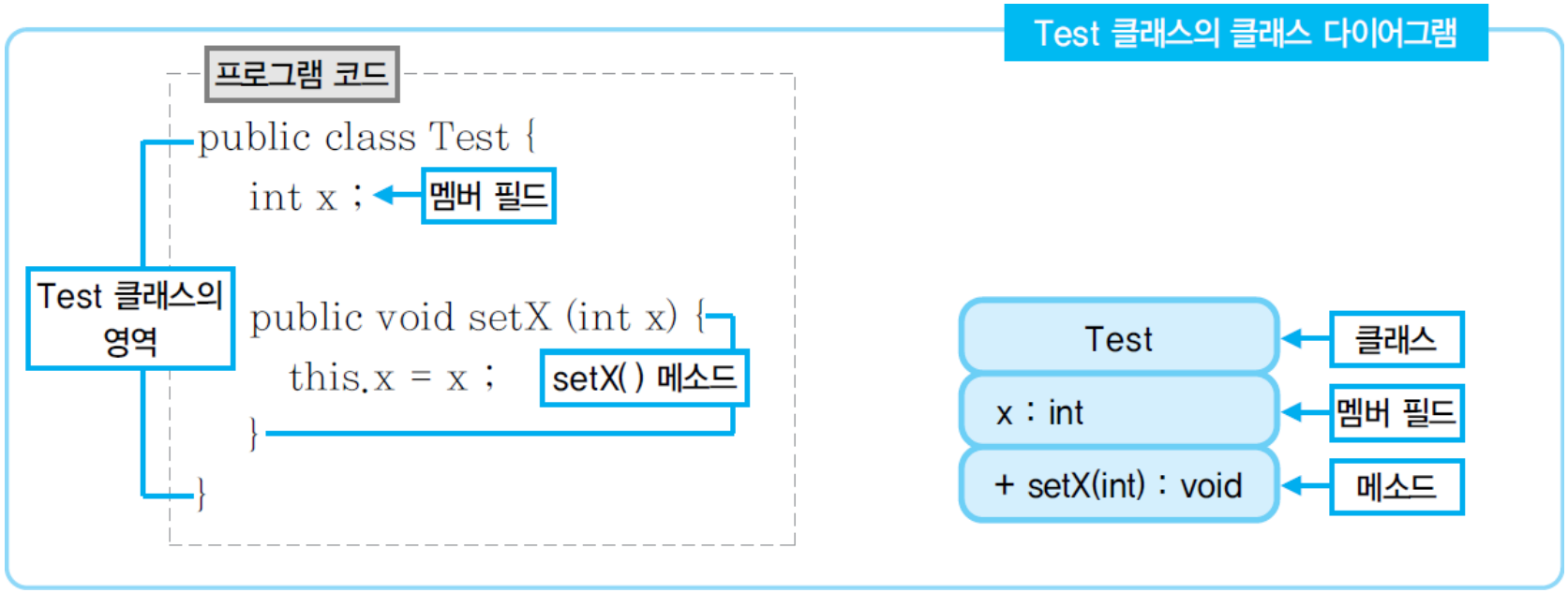
참고 사항 - UML 다이어그램

- **하나의 클래스의 구조를 표현**



참고 사항 - UML 다이어그램

- 하나의 클래스의 구조를 표현

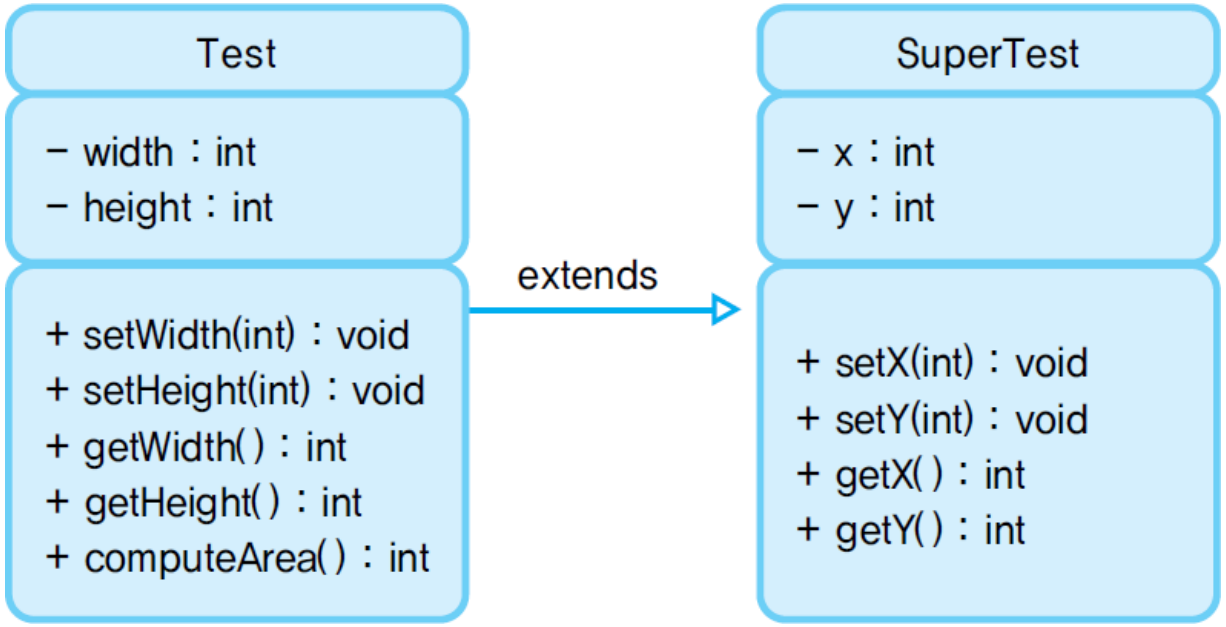


참고 사항 - UML 다이어그램

- **하나의 클래스의 구조를 표현**
  - 추상 클래스(abstract class)나 추상 메소드(abstract method)와 같이 abstract 키워드를 사용한 경우, 기울임꼴(italic : 이탤릭)로 표시
  - static 키워드를 사용한 메소드나 멤버 필드에는 밑줄(underline : 언더라인)을 표시

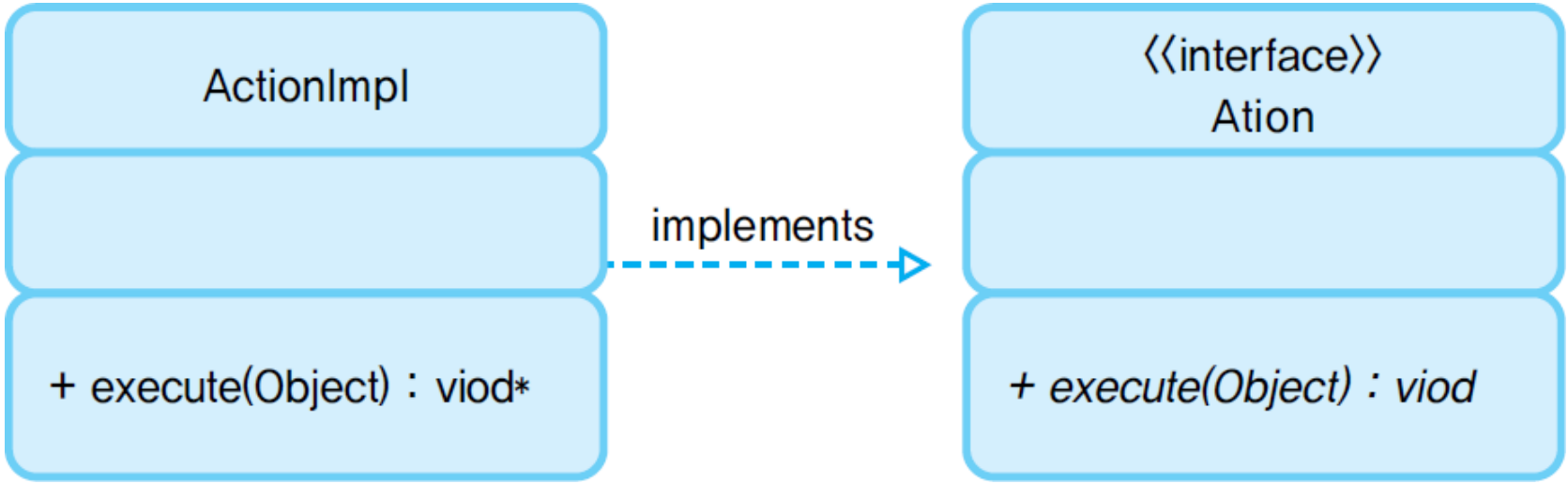
참고 사항 - UML 다이어그램

- **클래스간의 관계를 표현**
  - 상속(extends): 상속의 표시는 음영이 없는 직선 화살표로 나타내며, 화살촉이 있는 부분이 상속을 해주는 클래스이고 반대편이 상속을 받는 클래스



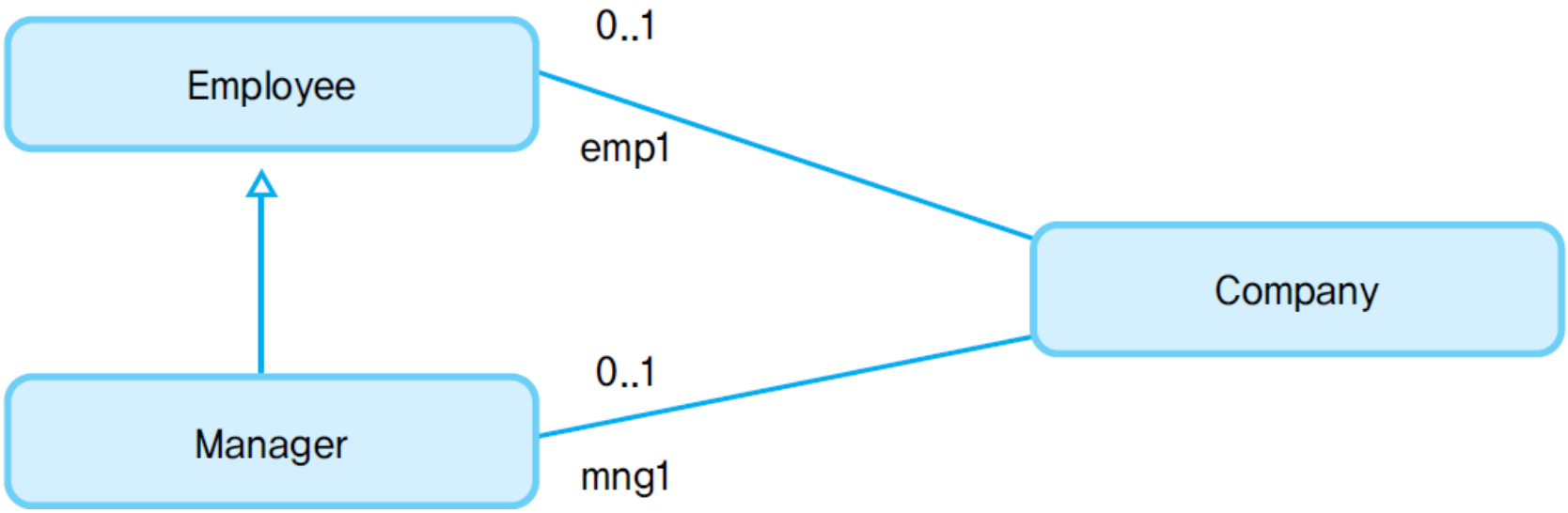
참고 사항 - UML 다이어그램

- **클래스간의 관계를 표현**
  - 임플리먼츠(implements) : implements 표시는 음영이 없는 점선 화살표로 나타낸다. 인터페이스는 <<interface>>를 기입



참고 사항 - UML 다이어그램

- **클래스간의 관계를 표현**
  - 객체들 간의 포함 관계 : 하나의 객체를 생성할 때 다른 객체를 같이 생성
    - 단일 객체 생성

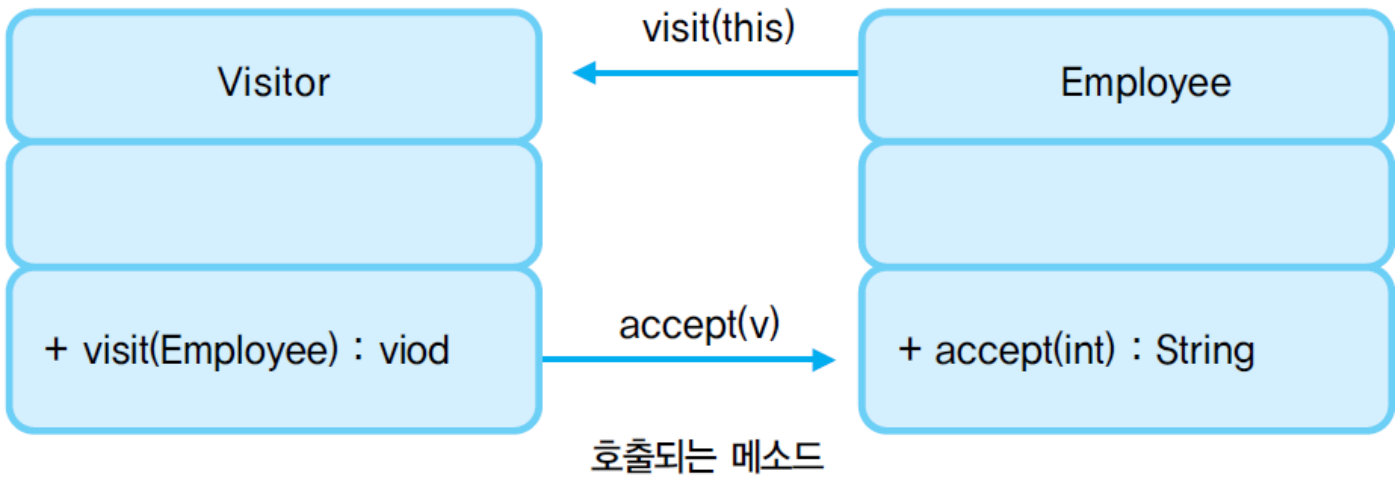


참고 사항 - UML 다이어그램

- **클래스간의 관계를 표현**
  - 객체들 간의 포함 관계 : 하나의 객체를 생성할 때 다른 객체를 같이 생성
    - 다중 객체 생성

참고 사항 - UML 다이어그램

- **클래스간의 관계를 표현**
  - 메소드 호출 관계와 주석



Visitor는 accept(v)를
호출해서 Employee에
접근한다.

주석