# Spring framework

## - spring CORE

1강 - 스프링의 핵심원리

Lectured by Soongu Hong

# 1. 스프링 프레임워크란?

# * 스프링의 핵심

- 스프링은 자바 언어 기반의 프레임워크! (단순히 웹 애플리케이션 개발에 국한되지 않음)
- 스프링은 강력한 객체 지향 애플리케이션을 만들 수 있게 도와주는 도구!

# 강력한 객체 지향 애플리케이션이 뭔데???

유연하고 변경이 쉬운 것?

위키백과, 우리 모두의 백과사전.

**객체 지향 프로그래밍**(영어: Object-Oriented Programming, OOP)은 컴퓨터 프로그래밍의 패러다임 중 하나이다. **객체 지향 프로그래밍은 컴퓨터 프로그램을 명령어의 목록으로 보는 시각에서 벗어나 여러 개의 독립된 단위, 즉 "객체"들의 모임으로 파악하고자 하는 것이다. 각각의 객체는 메시지를 주고받고, 데이터를 처리할 수 있다.**

객체 지향 프로그래밍은 프로그램을 유연하고 변경이 쉽게 만들기 때문에 대규모 소프트웨어 개발에 많이 사용된다. 또한 프로그래밍을 더 배우기 쉽게 하고 소프트웨어 개발과 보수를 간편하게 하며, 보다 직관적인 코드 분석을 가능하게 하는 장점이 있다. 그러나 지나친 프로그램의 객체화 경향은 실제 세계의 모습을 그대로 반영하지 못한다는 비판을 받기도 한다.

# 유연하고 변경이 쉽다 - 다형성

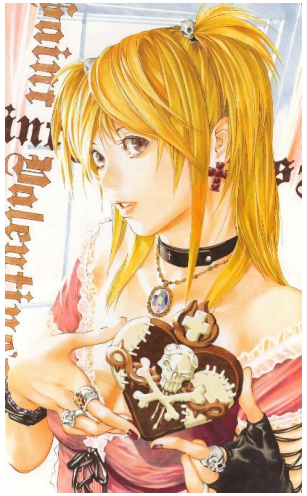
객체지향 세상에서는 모든 객체를

역할과 구현체로 구분한다.

# CAST SCHEDULE

| 일자 | 요일 | 시간 | 라이토 | 엘(L) | 렘 | 류크 | 미사 |
|---|---|---|---|---|---|---|---|
| 4월 20일 | 수 | 14:30 | 홍광호 | 김성철 | 김선영 | 서경수 | 케이 |
| 4월 22일 | 금 | 19:30 | 홍광호 | 김성철 | 장은아 | 서경수 | 케이 |
| 4월 23일 | 토 | 14:00 | 고은성 | 김성철 | 김선영 | 강홍석 | 장민제 |
| 4월 24일 | 일 | 14:00 | 고은성 | 김성철 | 장은아 | 서경수 | 케이 |
| 4월 26일 | 화 | 19:30 | 홍광호 | 김성철 | 장은아 | 서경수 | 케이 |
| 4월 27일 | 수 | 19:30 | 고은성 | 김성철 | 김선영 | 강홍석 | 케이 |
| 4월 30일 | 토 | 14:00 | 홍광호 | 김성철 | 김선영 | 강홍석 | 케이 |
| 5월 1일 | 일 | 14:00 | 홍광호 | 김성철 | 장은아 | 서경수 | 장민제 |
| 5월 5일 | 목 | 14:00 | 고은성 | 김성철 | 장은아 | 강홍석 | 장민제 |
| 5월 5일 | 목 | 19:00 | 홍광호 | 김성철 | 장은아 | 강홍석 | 케이 |
| 5월 7일 | 토 | 14:00 | 홍광호 | 김성철 | 장은아 | 강홍석 | 장민제 |
| 5월 8일 | 일 | 14:00 | 홍광호 | 김성철 | 김선영 | 서경수 | 케이 |

역할



구현체


구현체

# 역할 - 인터페이스

# 구현 - 클래스

컴포넌트들을 역할과 구현으로 구분 지으면
유연하며 변경이 쉬운 프로그램이 만들어진다.

# 2. 객체 지향 설계 원칙 (SOLID 원칙)

# SOLID

클린코드로 유명한 로버트 마틴이 좋은 객체 지향 설계의 5가지 원칙을 정리

- SRP: 단일 책임 원칙(single responsibility principle)
- OCP: 개방-폐쇄 원칙 (Open/closed principle)
- LSP: 리스코프 치환 원칙 (Liskov substitution principle)
- ISP: 인터페이스 분리 원칙 (Interface segregation principle)
- DIP: 의존관계 역전 원칙 (Dependency inversion principle)

# 1. SRP - 단일 책임 원칙

# (Single Responsibility Principle)

- 하나의 클래스는 하나의 책임만 가져야 한다.
- SRP를 적용하면 책임 영역이 확실해지기 때문에 변경의 연쇄작용에서 자유로워질 수 있습니다.
- 또한 코드의 가독성 향상, 유지보수 용이라는 이점도 누릴 수 있습니다.

# 2. OCP - 개방 폐쇄 원칙

# (Open Closed Principle)

- 확장에는 열려 있으나 변경에는 닫혀 있어야 한다.
- 요구사항의 변경이나 추가사항이 발생했을 때 기존 구성요소에는 수정이 일어나지 말아야 하며, 기존 구성요소를 쉽게 확장하여 재사용할 수 있어야 한다는 것입니다.
- 이를 가능케 하는 중요 메커니즘은 다형성과 추상화입니다.

# 3. LSP - 리스코프 치환 원칙

# (Liskov Subtitution Principle)

- 객체는 프로그램의 정확성을 깨뜨리지 않으면서 하위 타입 인스턴스로 변경할 수 있어야 한다.
- 하위 타입 인스턴스는 다른 인스턴스로 교체될 때 상위 타입의 규약을 지켜야 한다는 뜻입니다.
- 예를 들면 자동차(하위타입 인스턴스)를 바꿀 때 아반떼가 소나타로 바뀌어도 반드시 accelarator의 규약은 속도가 증가해야 한다는 인터페이스의 규약을 지켜야 한다는 것입니다.

# 4. ISP - 인터페이스 분리 원칙

# (Interface Segregation Principle)

- 특정 클라이언트를 위한 여러 개의 인터페이스가 하나의 범용 인터페이스보다 낫다.
- SRP가 클래스의 단일 책임을 의미한다면, ISP는 인터페이스의 단일 책임을 의미합니다.
- 예를 들면 자동차 인터페이스에는 운전에 관한 기능들과 정비에 관한 기능들이 모두 명세 되어 있는 것 보다는 모든 운전자가 모두 자가정비를 하는 것이 아니기 때문에 운전 인터페이스와 정비 인터페이스로 따로 분리하는 것이 좋습니다.

# 5. DIP - 의존관계 역전 원칙

# (Dependency Inversion Principle)

- 구현 클래스에 의존하지 말고 인터페이스에 의존하라.
- 객체 지향 설계에서는 반드시 클라이언트는 역할에 의존하지 않고 구현체에 의존하는 순간 변경이 아주 어려워집니다.

스프링 프레임워크는 아주 쉽게 **SOLID** 원칙을 지키면서 개발할 수 있게 해준다!

# 3. 제어의 역전 (Inversion Of Control)

# 제어의 역전 - IoC
# (Inversion of Control)

- 기존 프로그램은 클래스 내부에서 자신이 필요한 다른 클래스의 객체를 직접 생성하여 연결하거나 실행했습니다.
- 반면에 객체 생성을 다른 클래스에게 위임하여 객체 생성의 제어권을 넘기는 것을 제어의 역전이라고 부릅니다.
- 스프링 프레임워크는 객체 생성의 제어권을 스프링 컨테이너가 전임하기 때문에 IoC Framework라고 부를 수 있습니다.