[ 05 ] AOP(Aspect Oriented Programming)

1. AOP란?
   * 프로그래밍을 하다 보면, 공통적인 기능이 많이 발생합니다. 이러한 공통 기능을 모든 모듈에 적용하기 위한 방법으로 상속을 통한 방법이 있습니다. 상속도 좋은 방법이기는 하지만 몇 가지 문제가 있습니다. 우선 JAVA에서는 다중 상속이 불가하므로 다양한 모듈에 상속기법을 통한 공통 기능 부여는 한계가 있습니다. 그리고, 기능 구현부분에 핵심 기능 코드와 공통 기능 코드가 섞여 있어 효율성이 떨어집니다.
   * 위의 상속을 통한 방법에 한계가 있어 AOP가 등장하게 되었습니다.
   * 예를 들어 쇼핑몰 A와 쇼핑몰 B가 있다고 가정해 보겠습니다. 만일 A는 구매 시 고객의 포인트 점수를 올려주지만, B의 경우 직접 결재 시 가격을 낮춰주는 정책이 존재한다고 생각해 봅시다. 이 경우 사용자가 하는 행위는 ‘쇼핑몰에서 물건을 구입한다’이지만 부가적으로 A사와 B사의 정책 차이 때문에 코드를 변경해 줘야만 합니다.
   * AOP는 사전적 의미로는 ‘측면 혹은 관점’을 의미하지만, 실제로 프로그램 개발에서 의미하는 것은 ‘비즈니스 로직은 아니지만 반드시 해야 하는 작업. 필요한 기능들’로 해석될 수 있습니다
   * **AOP방법은 핵심 기능과 공통 기능을 분리 시켜놓고, 공통 기능을 필요로 하는 핵심 기능들에서 사용하는 방식 입니다.**
   * 특정 시스템 내에서 이러한 예로는 보안이나 성능 모니터링과 같은 작업을 들 수 있습니다. 보안 검증이 된 사람에 의해서만 특정 비즈니스 로직이 이뤄져야 하지만 그 자체가 고객의 비즈니스는 아닙니다. 그 자체가 목적은 아니고 시스템의 완성도를 높여주는 역할을 합니다.
2. AOP와 관련된 용어
   * **Aspect** : 공통 기능. 예를 들어 로깅같은 기능 자체에 대한 용어
   * **Advice** : 공통기능을 구현한 객체 메
   * **Join Point** : 핵심기능. 공통 기능을 적용할 수 있는 대상.
   * **PointCuts** : Join Point의 부분으로 실제 Advice를 적용해야 되는 부분
   * Proxy : Advice가 적용되었을 때 만들어지는 객체

Weaving : Advice와 target이 결합되어서 프록시 객체를 만드는 과정(advice를 핵심기능에 적용하는

Ch05\_aop

Student.java

Porm.xml 3개 추가

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.7.4</version>

</dependency>

-

<dependency>

<groupId>cglib</groupId>

<artifactId>cglib</artifactId>

<version>2.2</version>

</dependency>

-

<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->

<dependency>

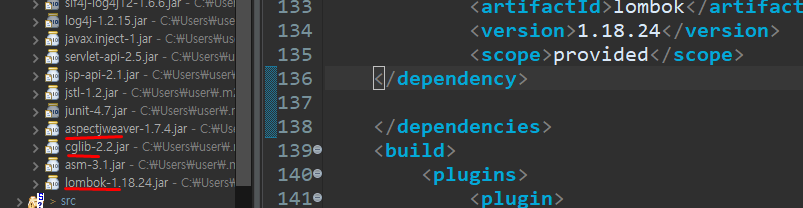
<groupId>org.projectlombok</groupId>

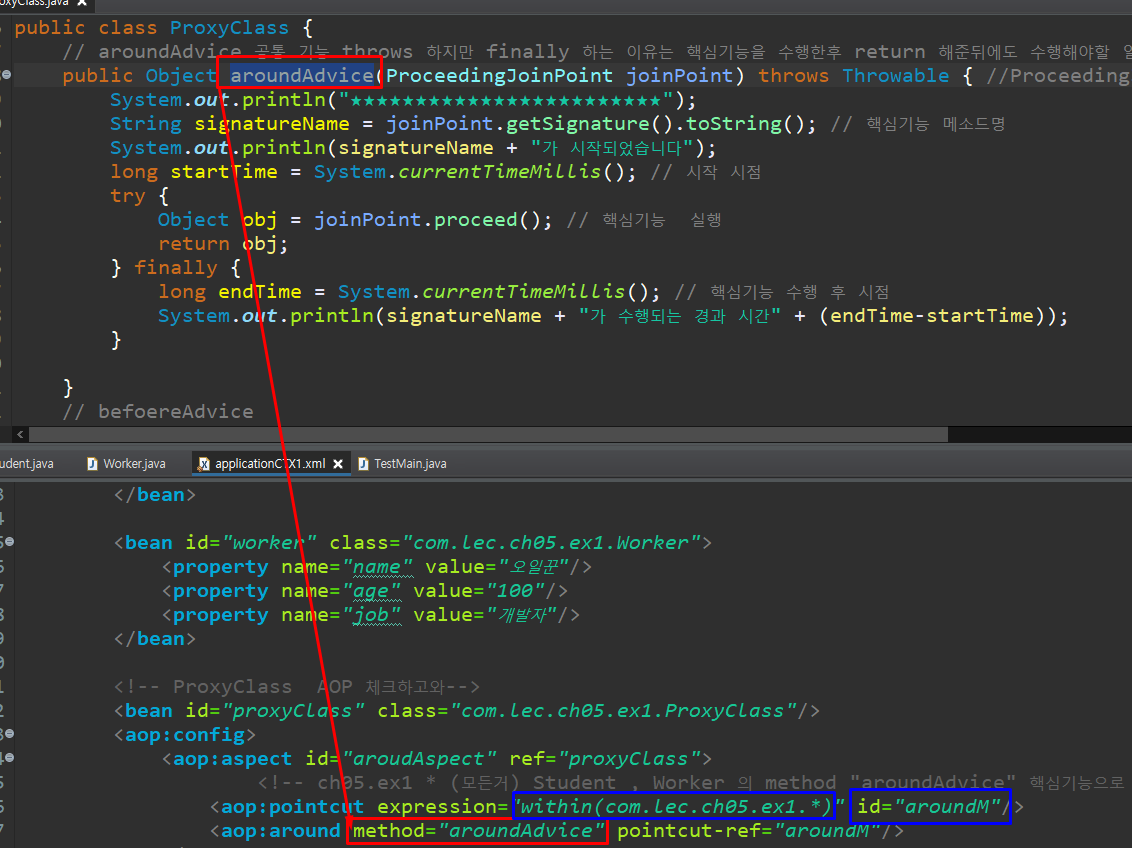
<artifactId>lombok</artifactId>

<version>1.18.24</version>

<scope>provided</scope>

</dependency>





* Within

1. 스프링에서 AOP 구현
   * AOP의 구현은 proxy를 이용한다.
   * Weaving 방식은 두가지
     1. XML을 이용
     2. @Aspect 어노테이션 이용

* <aop :around> : 앞 뒤로 실행
* 