* @Component : Bean Configuration 파일에 Bean을 등록하지 않아도 자동으로 등록된다.
* @Bean : 개발자가 Class의 코드를 수정할 수 없는 경우

같은 클래스 타입의 Bean을 여러 개 등록할 경우

* @Component : 개발자가 Class의 코드를 수정할 수 있는 경우
* @Component 를 사용하면 Bean Configuration File에 Bean 을 정의하지 않아  
    
  도 자동으로 등록된다.

**<AOP>**

* Aspect Oriented Programming : 관점 지향 프로그래밍
* 하나의 프로그램을 관점(혹은 관심사)라는 논리적인 단위로 분리하여 관리하는 개념
* 관심사를 통해 Spring Framework가 어떤 메서드가 호출되는지 관심있게 지켜보다가 특정 메서드가 호출되면 자동으로 메서드 전과 후에 다른 메서드가 호출 될 수 있도록 한다.

<용어>

* Joint Point : 모듈이 삽입되어 동작하게 되는 특정 위치(메서드 호출 등)
* Point Cut : 다양한 Joint Point 중에 어떤 것을 사용할지 선택
* Advice : Joint Point에 삽입되어 동작할 수 있는 코드
* Weaving : Advice를 핵심 로직 코드에 적용하는 것
* Aspect : Point Cut + Advice

<Advice 종류>

* before : 메서드 호출 전에 동작하는 Advice
* after-returning : 예외 없이 호출된 메서드의 동작이 완료되면 동작하는 Advice
* after-throwing : 호출된 메서드 동작 중 예외가 발생했을 때 동작하는 Advice
* after : 예외 발생 여부에 관계없이 호출된 메서드의 동작이 완료되면 동작하는 Advice
* around : 메서드 호출 전과 후에 동작하는 Advice

<Execution 명시자>

* Pointcut을 저정할 때 사용하는 문법
* execution(접근제한자 리턴타입 클래스이름 메서드이름(매개변수))
* 접근 제한자 : public만 지원된다.
* 리턴타입 : 메서드의 매개변수 타입
* 클래스 이름 : 패키지를 포함한 클래스 이름
* 메서드 이름 : 메서드의 이름
* 매개변수 : 매개변수의 형태
* \* : 하나의 모든 것을 의미한다.
* .. : 개수 상관없이 모든 것을 의미한다.