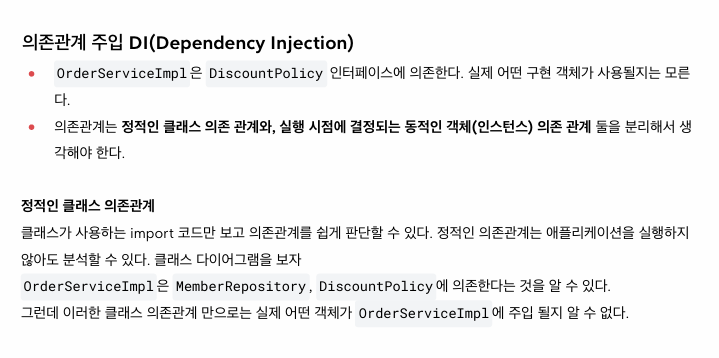


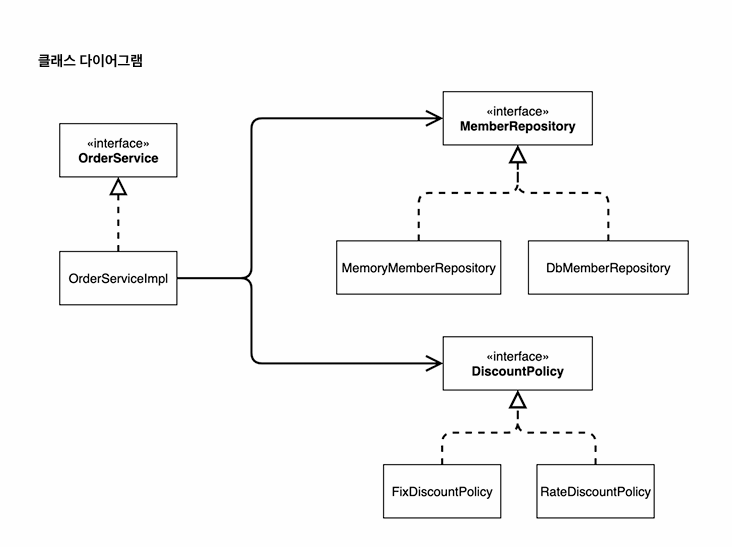
제어의 역전 : 개발자가 객체를 생성하고 연결하고 실행하는 것이 아닌, 프레임워크 같은 대리자가 객체 생성 연결 주입을 대신 해주는 것을 의미한다. (제어권이 뒤바뀐다.)

프레임 워크가 내가 작성한 코드를 제어하고, 대신 실행하면 그것은 프레임워크다

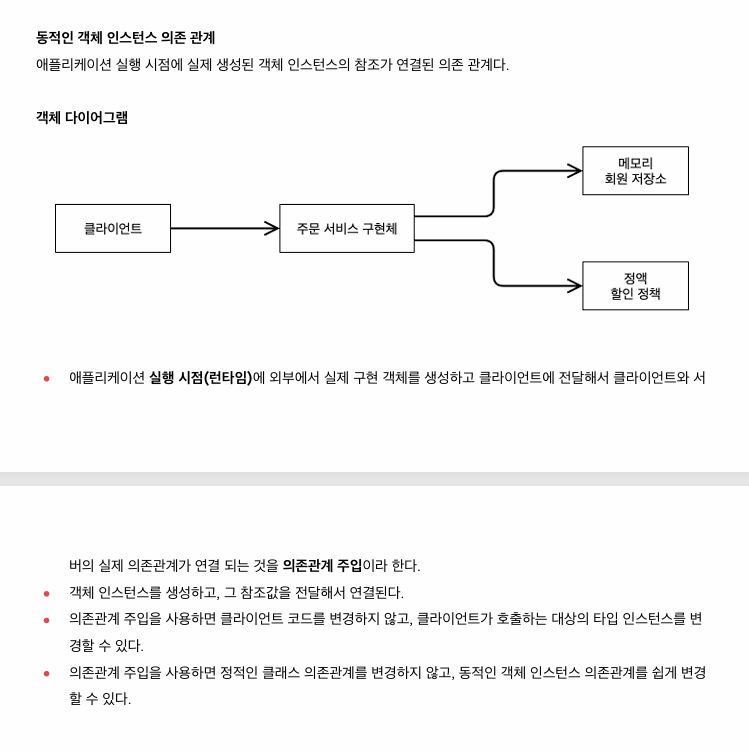
: ex ) JUNIT > Annotation을 작성하면, 우리가 작성한 코드 실행의 제어를 가지고 대신 실행 해줌. 또한 beforeEach와 같이 생명주기 관리를 한다. junit의 고유한 라이플 사이클 속에서 내가 만든 로직을 중간에 실행한다. 제어권을 개발자가 아닌 junit이 가지고 있다.



클래스다이어그램을 보고 알 수 있는 것처럼 어플리케이션을 실행하지 않고, import 코드만 보고 어떤 의존 관계를 가지고 있는지 알 수 있으면 정적인 클래스 의존관계를 의미한다.



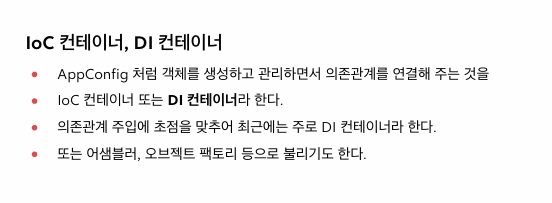
상속이든, interface 구현이든 화살표 방향으로 의존하고 있다는 뜻이다.

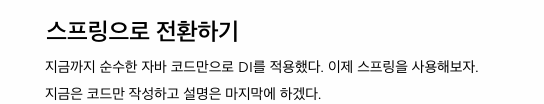


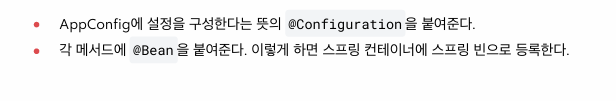
실제 실행 시켜봐야 구현체가 무엇이 올지 안다면, 동적인 객체 인스턴스 의존 관계라고 한다.

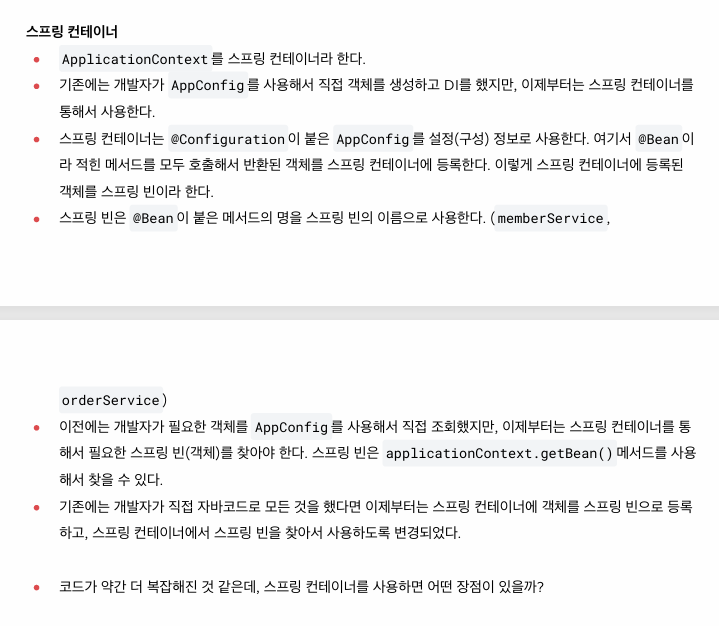
구현체는 애플리케이션을 실행할 때마다 동적으로 바뀌게 된다.

정적인 클래스 의존관계를 변경하지 않는다는 것 : 코드를 손대서 import를 변경(구현체에 의존하고 있는 경우라면, 사용하는 구현체의 종류에 따라 import 내용이 달라지게 됨)









@Bean(name = “ㅇㅇㅇ”)과 같이 bean의 이름을 사용자가 임의로 설정할 수도 있지만, 관례상 스프링 컨테이너에서 호출하는 메소드의 이름을 따르는 것이 좋다.