2️⃣

Section2. 스프링 핵심 원리 이해1 - 예제 만들기

프로젝트 생성

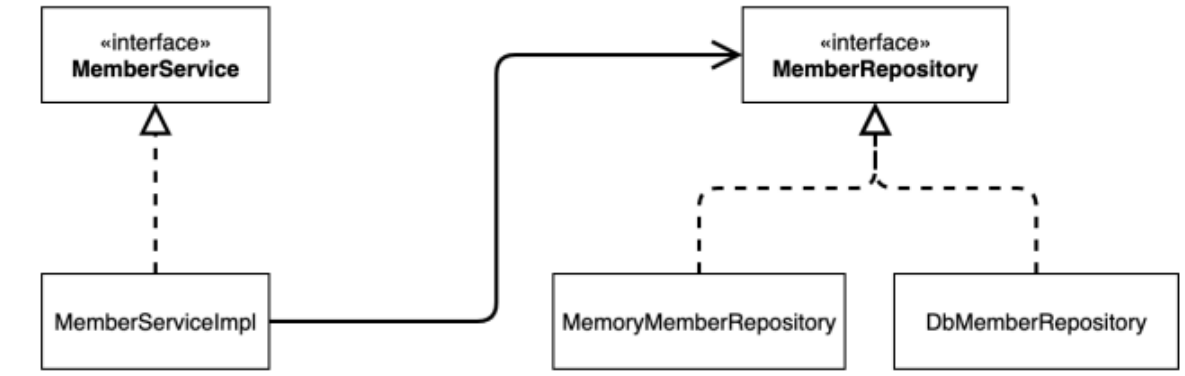
- Gradle
- groupId: hello
- artifactId: core
- Dependencies: 선택 X

비즈니스 요구사항과 설계

- 회원
  - 회원가입, 회원조회
  - 회원등급 - 일반, VIP
  - 회원 데이터는 자체 DB 구축, 외부 시스템 연동 (미확정)
- 주문, 할인 정책
  - 회원은 상품 주문 가능
  - 회원 등급에 따른 할인 정책 적용
  - 할인 정책은 VIP는 1,000원(고정 금액) 할인 ← 변경 가능성 큼

회원 도메인 설계

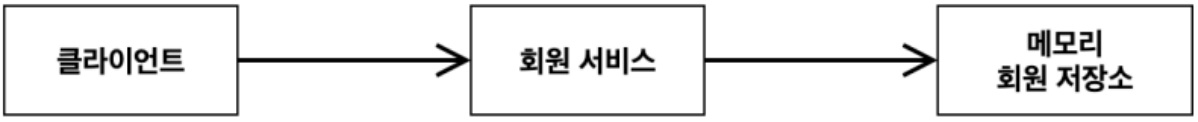
회원 클래스 다이어그램



회원 도메인 개발

> 💡 실습 코드 작성

- 사용자 저장 메서드 `save()`
- 사용자 조회(id) 메서드 `findMember()`



`MemberService` → `MemberServiceImpl` → `MemberRepository` → `MemoryMemberRepository`

회원 도메인 실행과 테스트

```
// 순수 자바 코드로 생성된 MemberApp
import hello.core.member.Grade;
import hello.core.member.Member;
import hello.core.member.MemberService;
import hello.core.member.MemberServiceImpl;

public class MemberApp {
    public static void main(String[] args) {
        MemberService memberService = new MemberServiceImpl()
        Member member = new Member(1L, "memberA", Grade.VIP);
        memberService.join(member);
```

```
        Member findMember = memberService.findMember(1L);
        System.out.println("new member = " + member.getName()
        System.out.println("find Member = " + findMember.getN
    }
}
```

- 해당 코드의 설계 상 문제점은?
  - 다른 저장소로 변경할 때 OCP 원칙은 지켜지는가?
  - DIP 잘 지켜지고 있는가?

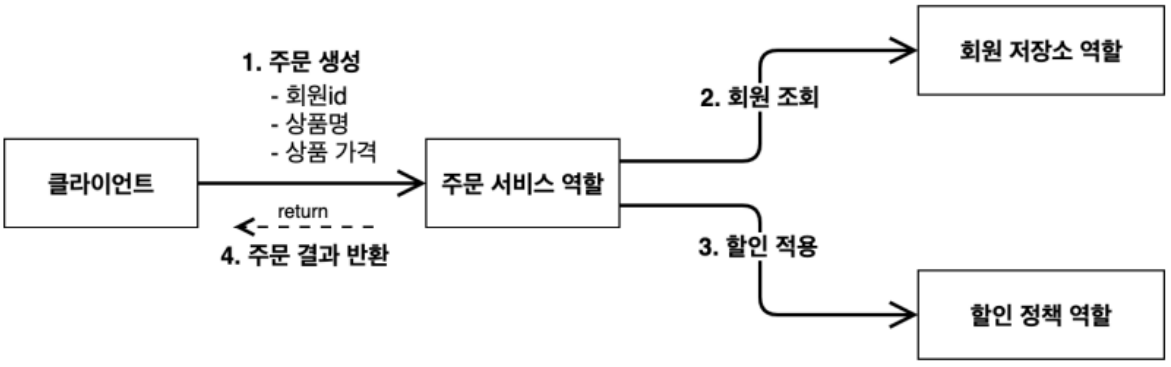
→ '의존관계가 인터페이스 뿐만 아니라 구현까지 모두 의존하는 문제점이 있음'

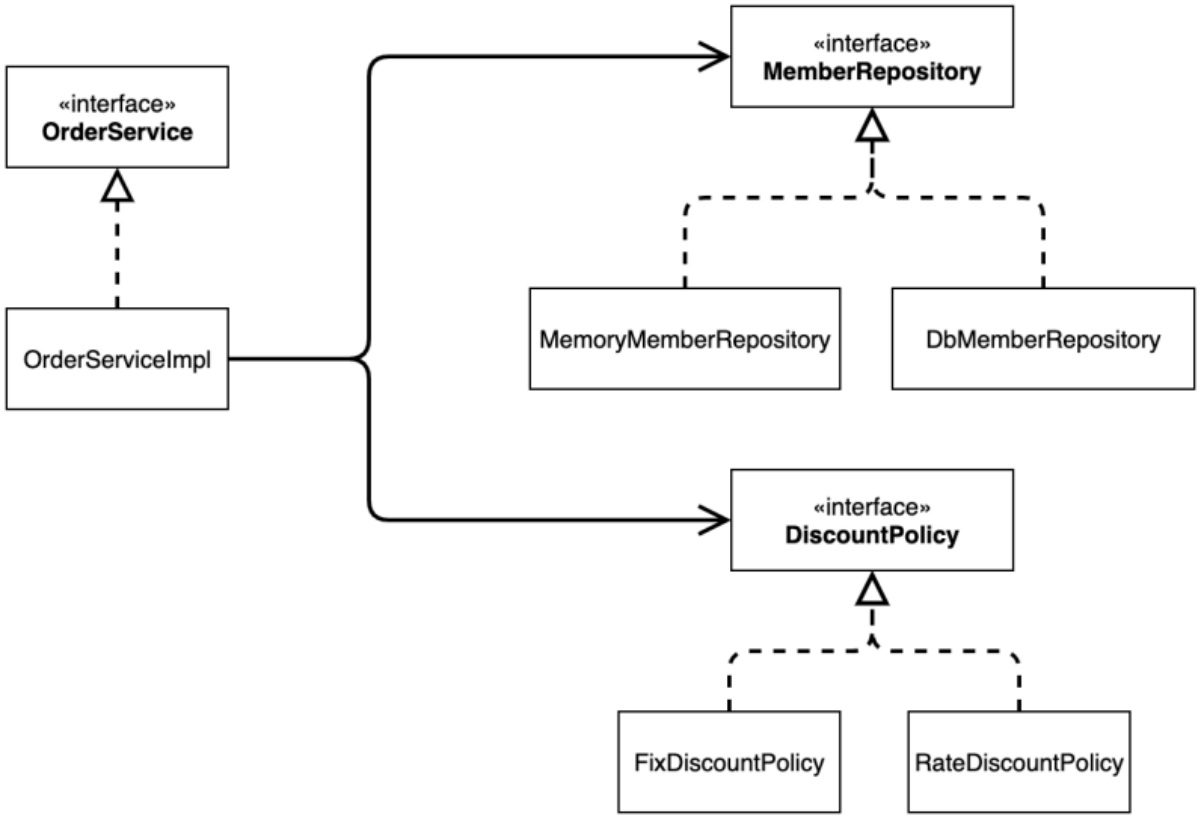
실습코드를 보면 `MemberRepository memberRepository = new MemoryMemberRepository;`

즉, `MemberServiceImpl` 은 추상화, 구현 모두에게 의존하고 있음

주문과 할인 도메인 설계

실습 코드 작성





주문과 할인 도메인 개발

> 💡 실습 코드 작성

`Order createOrder(Long memberId, String itemName, int itemPrice)`

- 위 실습 코드의 설계는 Order 자체는 할인 정책과, 사용자의 정보를 알고있지 않다.
  - 잘 설계된 코드 ← 단일 책임의 원칙

주문과 할인 도메인 실행과 테스트

> 💡 실습 코드 작성