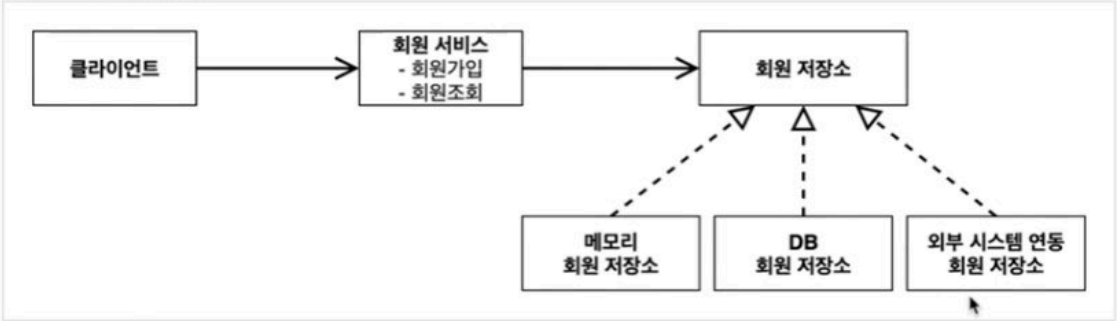
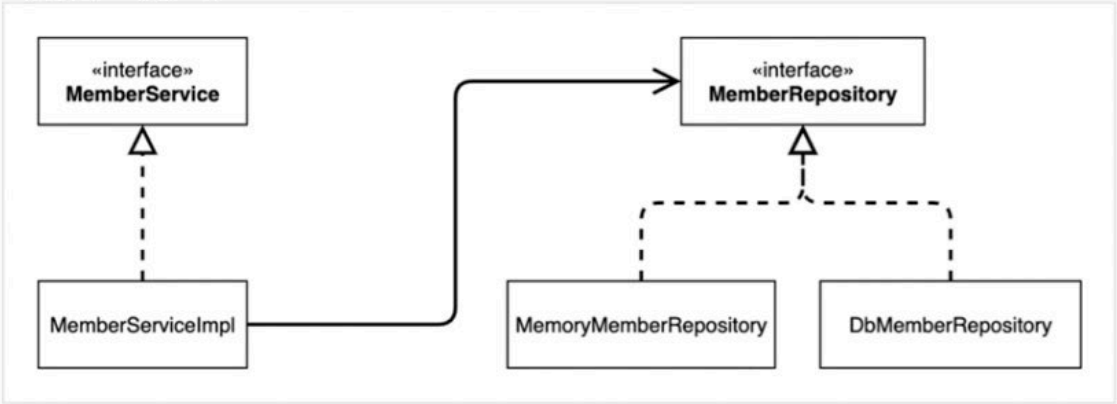
H2 회원 도메인 설계

- 회원 도메인 요구사항
  - 회원을 가입하고 조회할 수 있다.
  - 회원은 일반과 VIP 두 가지 등급이 있다.
  - 회원 데이터는 자체 DB를 구축할 수 있고, 외부 시스템과 연동할 수 있다. (미확정)

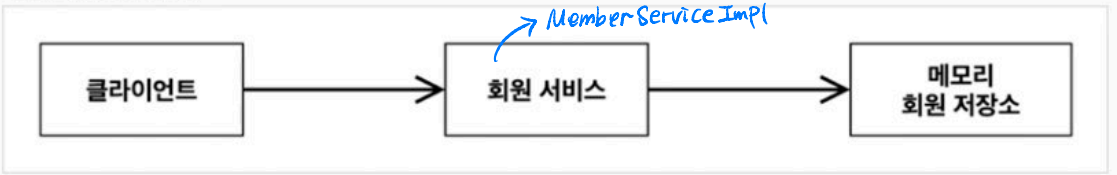
① *회원 도메인 협력 관계* (기획자도 같이 볼 수 있음)



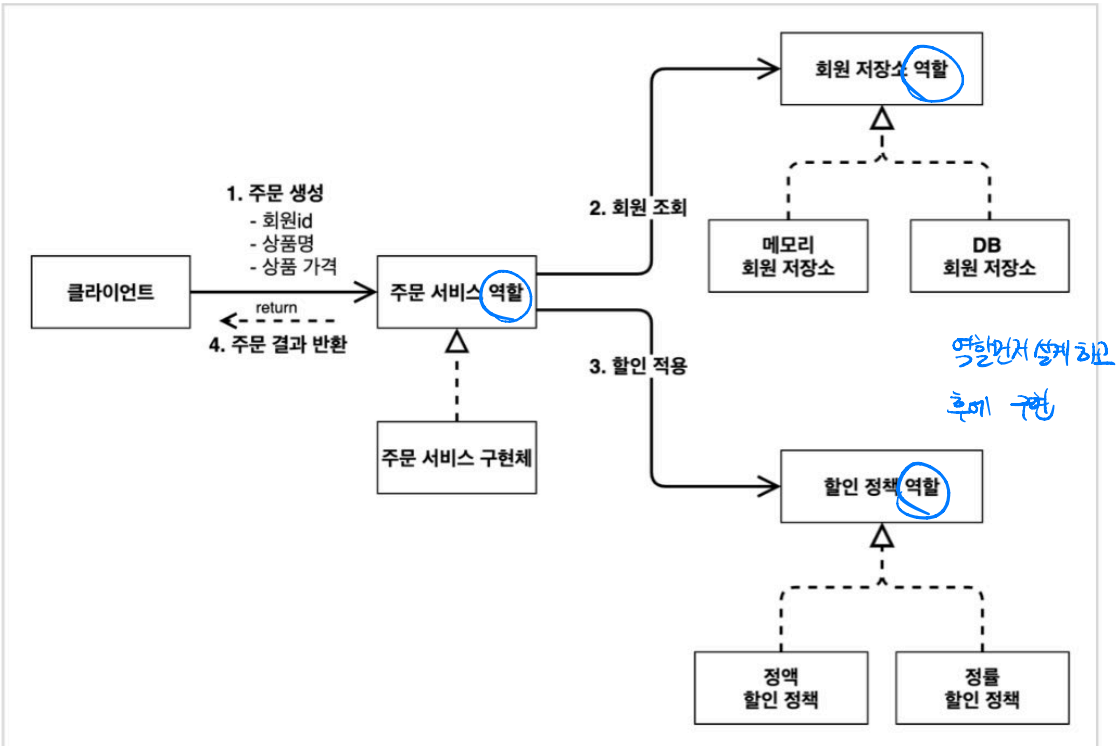
② *회원 클래스 다이어그램* (개발자가 구체화)



③ *회원 객체 다이어그램* (서버가 떠서 클라이언트가 실제 사용하는 것)

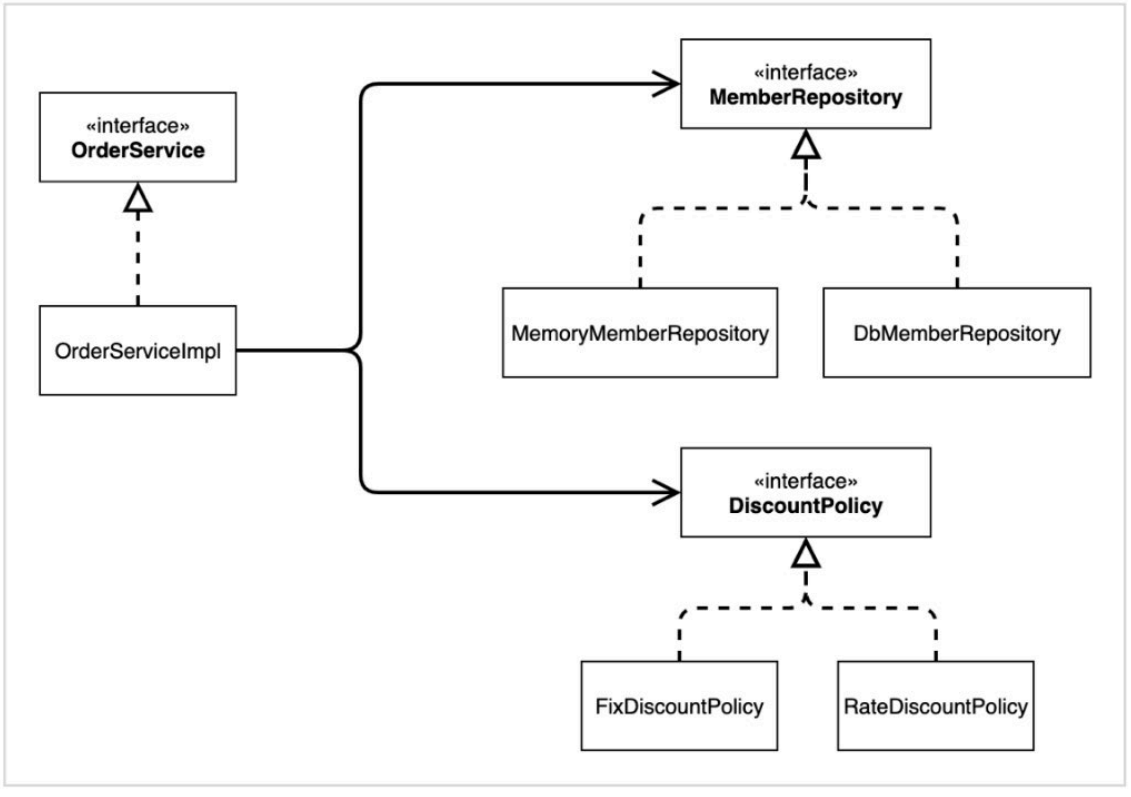


① 도메인 협력 관계

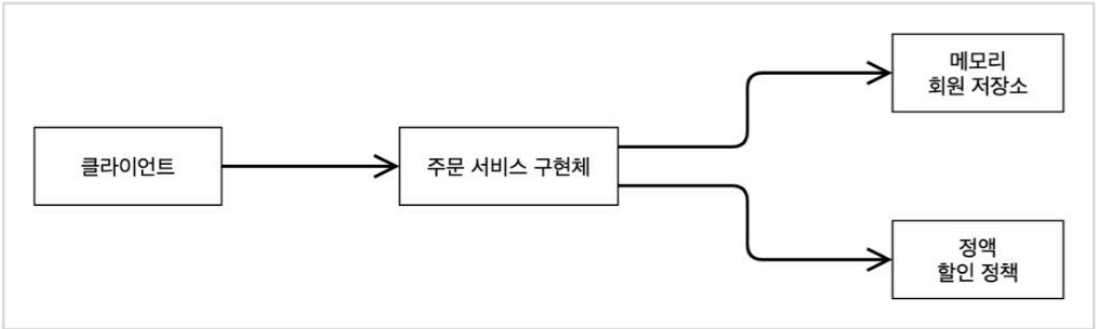


역할과 구현을 분리해서 자유롭게 구현 객체를 조립할 수 있게 설계했다. 덕분에 회원 저장소는 물론이고, 할인 정책도 유연하게 변경할 수 있다.

② 주문 도메인 클래스 다이어그램

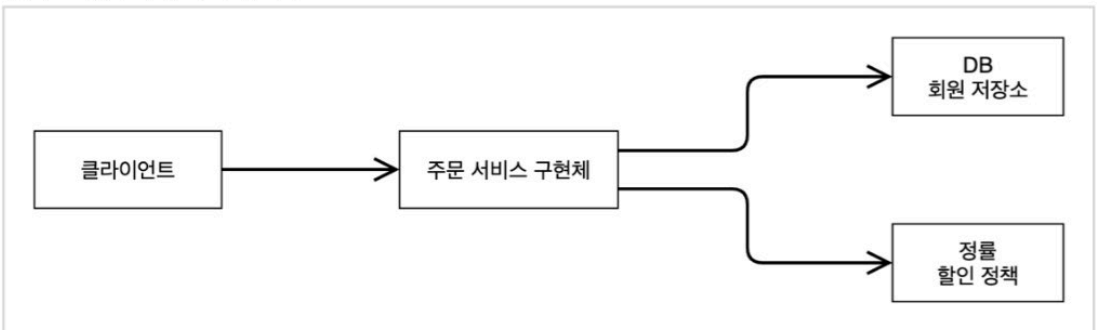


주문 도메인 객체 다이어그램1



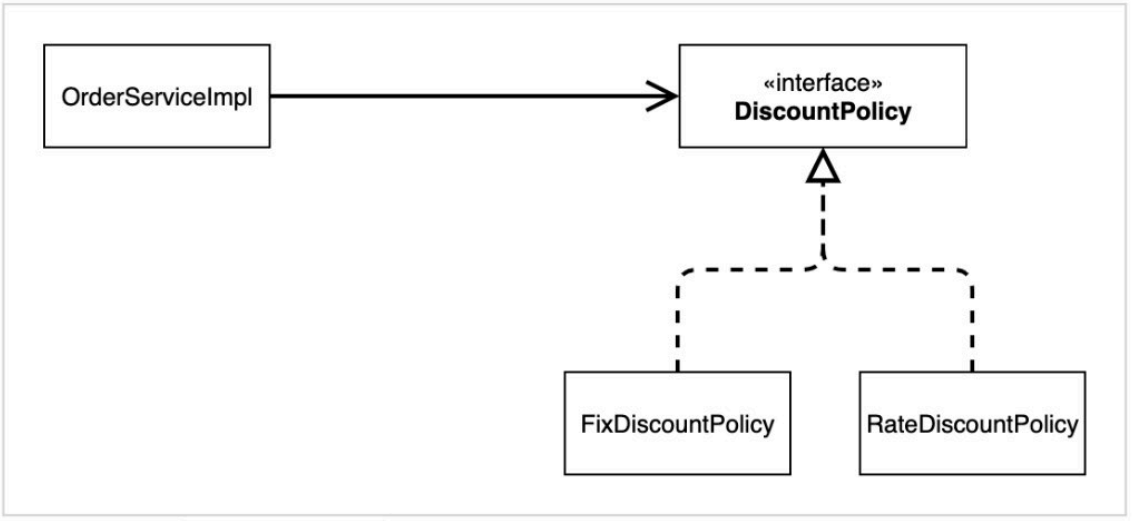
회원을 메모리에서 조회하고, 정액 할인 정책(고정 금액)을 지원해도 주문 서비스를 변경하지 않아도 된다.
역할들의 협력 관계를 그대로 재사용 할 수 있다.

주문 도메인 객체 다이어그램2



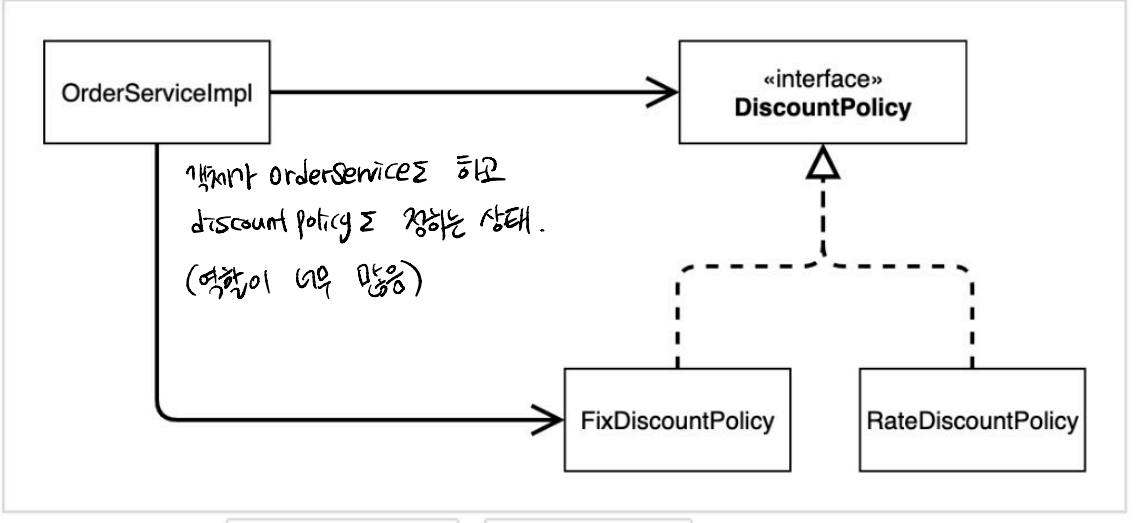
회원을 메모리가 아닌 실제 DB에서 조회하고, 정률 할인 정책(주문 금액에 따라 % 할인)을 지원해도 주문 서비스를 변경하지 않아도 된다.
협력 관계를 그대로 재사용 할 수 있다.

기대했던 의존관계



지금까지 단순히 `DiscountPolicy` 인터페이스만 의존한다고 생각했다.

실제 의존관계



잘보면 클라이언트인 `OrderServiceImpl`이 `DiscountPolicy` 인터페이스 뿐만 아니라

AppConfig 등장

- 애플리케이션의 전체 동작 방식을 구성(config)하기 위해, **구현 객체를 생성**하고, **연결**하는 책임을 가지는 별도의 설정 클래스를 만들자.

AppConfig

```
package hello.core;

import hello.core.discount.FixDiscountPolicy;
import hello.core.member.MemberService;
import hello.core.member.MemberServiceImpl;
import hello.core.member.MemoryMemberRepository;
import hello.core.order.OrderService;
import hello.core.order.OrderServiceImpl;

public class AppConfig {

    public MemberService memberService() {
        return new MemberServiceImpl(new MemoryMemberRepository());
    }

    public OrderService orderService() {
        return new OrderServiceImpl(
                new MemoryMemberRepository(),
                new FixDiscountPolicy());
    }

}
```

① 생성

② 생성 + 주입

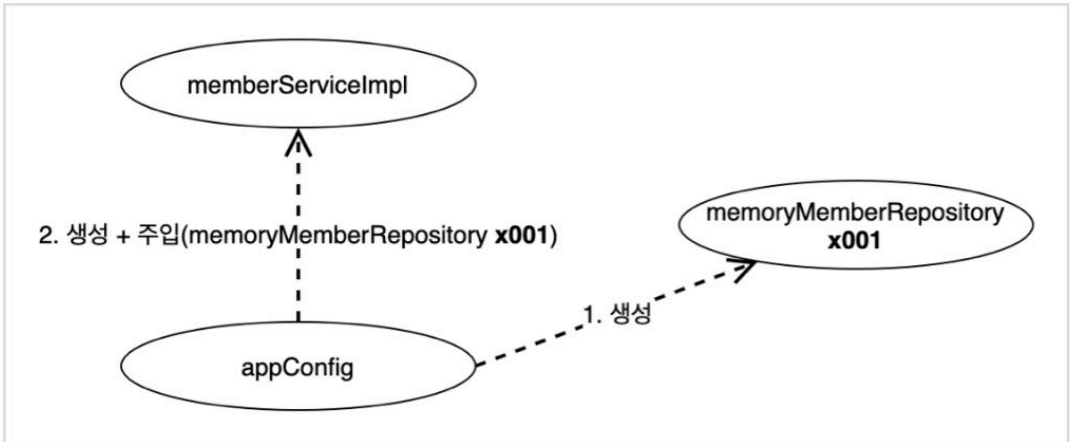
MemberServiceImpl - 생성자 주입

```
package hello.core.member;

public class MemberServiceImpl implements MemberService {

    private final MemberRepository memberRepository; //

    //생성자
    public MemberServiceImpl(MemberRepository memberRepository) {
        this.memberRepository = memberRepository;
    }

    public void join(Member member) {
        memberRepository.save(member);
    }

    public Member findMember(Long memberId) {
        return memberRepository.findById(memberId);
    }

}
```

★ 객체가 인터페이스만 알고있는 상태.
DIP 위반 해결.

외부에서 repository (종류) 들어옴.

- 설계 변경으로 `MemberServiceImpl`은 `MemoryMemberRepository`를 의존하지 않는다!


memberServiceImpl
2. 생성 + 주입(memoryMemberRepository x001)
memoryMemberRepository
x001
1. 생성
appConfig