

레거시와 리팩토링.

백명석 & 최범균 강사



The Red.

TABLE OF CONTENT.

- 1 레거시
- 2 레거시 분석
- 3 레거시에 테스트 코드 만들기
 - 방법1 : 범위를 좁혀서 테스트 작성
 - 방법2 : 대체 구현을 이용해서 테스트 작성
 - 방법3 : 범위를 넓혀서 테스트 작성
- 4 리팩토링
- 5 정리

레거시



레거시 코드

몇 가지 정의

- 오래되었지만 여전히 사용되는 것
- 테스트가 없는 코드
- 모든 코드가 레거시

레거시에 대한 느낌

하기 싫음, 부담됨

월급 원천

레거시 덕분에 회사가 굴러 감

레거시의 혼한 특징



수정 공포

레거시 코드를 이해하는데
많은 노력 필요

이해가 부족한 상태에서
코드를 수정해야 하는 상황 많음
기능 변경 요구, 일정 압박

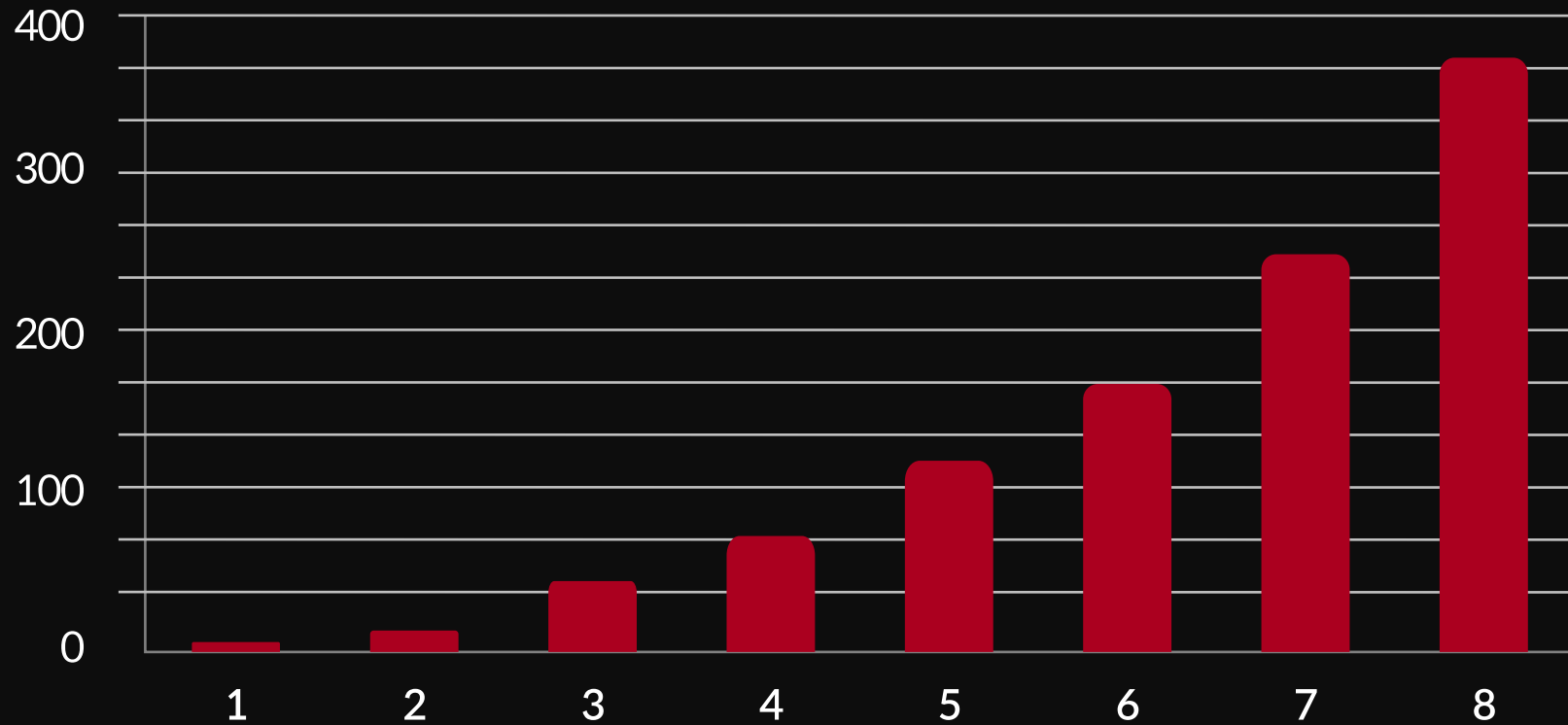
악순환

코드 덩태기로 두려움 회피

- 복붙 → 중복 증가
- if-else 블록 추가 → 복잡함 증가
 - 긴 메서드, 긴 클래스 유발

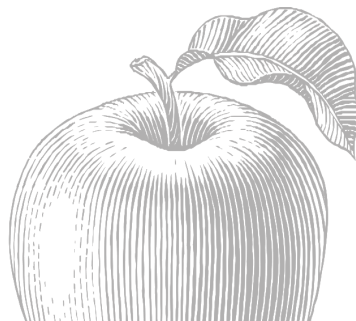
레거시 변경에 대한 두려움

악순환과 개발비용



악순환 줄이기

- ✓ 악순환을 줄이려면 코드 변경 비용을 낮춰야 함
- ✓ 변경 비용을 낮추려면 변경하기 쉬운 구조로 점진적으로 리팩토링 해야 함
- ✓ 리팩토링해도 이전과 동일하게 동작해야 함
- ✓ 이전과 동일하게 동작하는지 확인할 수 있는 테스트가 필요함
- ✓ 테스트를 만들려면 기능이 어떻게 동작하는지 분석해야 함
- ✓ 즉 악순환을 줄이려면 레거시를 분석하고 테스트를 만들고 리팩토링 해야 함



레거시 분석



한계

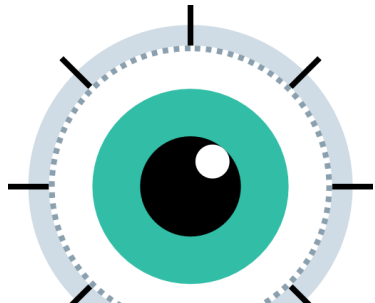
코드 전반을
머릿속에
보관하기 어려움

모니터로
긴 코드를 보는 것도
어려움

코드를
이해하기 위한
보조 수단 필요

코드 분석에 도움이 되는 보조 수단

코드 시각화



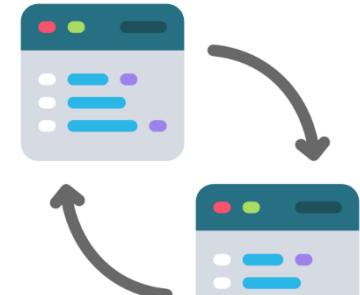
코드 출력
+ 형광펜



함께 코드 보기



스크래치
리팩토링



코드 분석에 도움이 되는 보조 수단

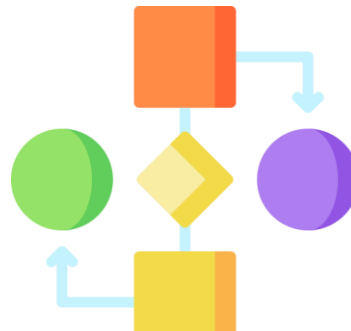
코드 시각화

다이어그램을 사용해서 코드 흐름을 시각화

실행 흐름 이해에 많은 도움

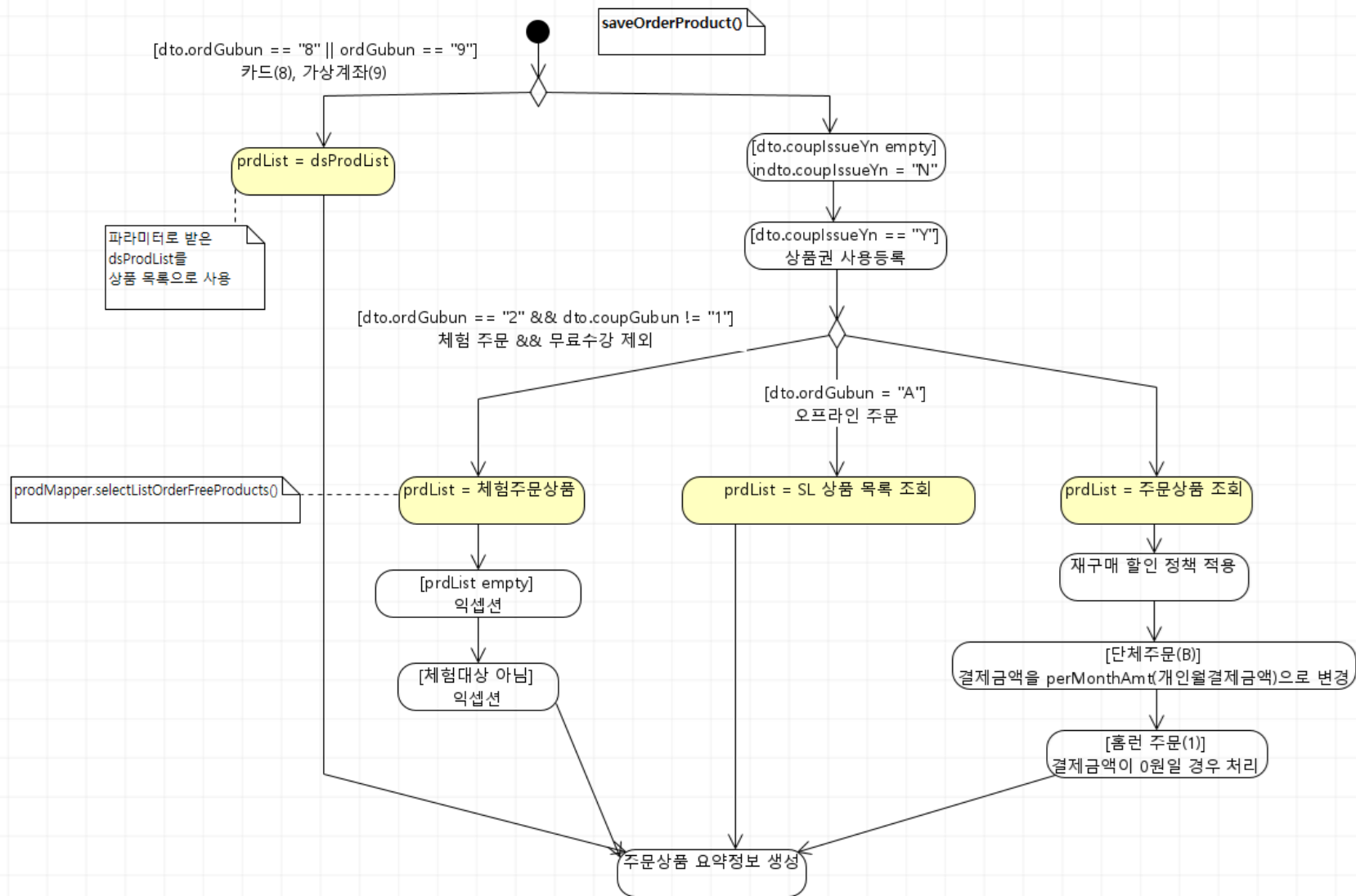
시각화를 위한 몇 가지 표기법

- 액티비티 다이어그램: 단계적인 코드 실행 흐름
- 시퀀스 다이어그램: 구성 요소간 연동 흐름
- 클래스, 메서드, 필드, 함수 등 구성요소 간 의존/호출 관계 그래프



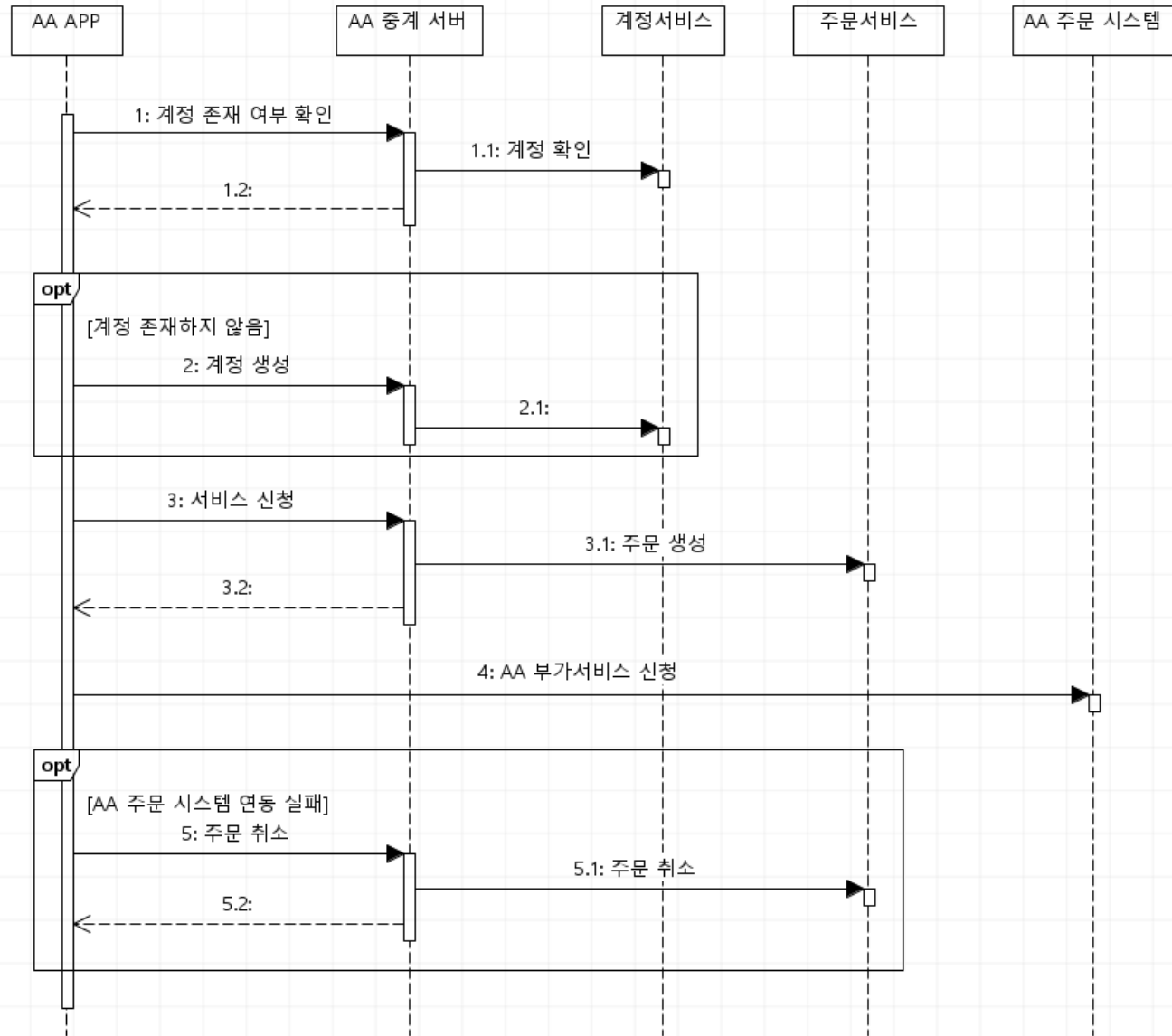
액티비티 다이어그램 예시

- 코드 구조 시각화
- 조건 분기 추적 용이
- 논리적인 코드 블록 도출



시퀀스 다이아그램 예시

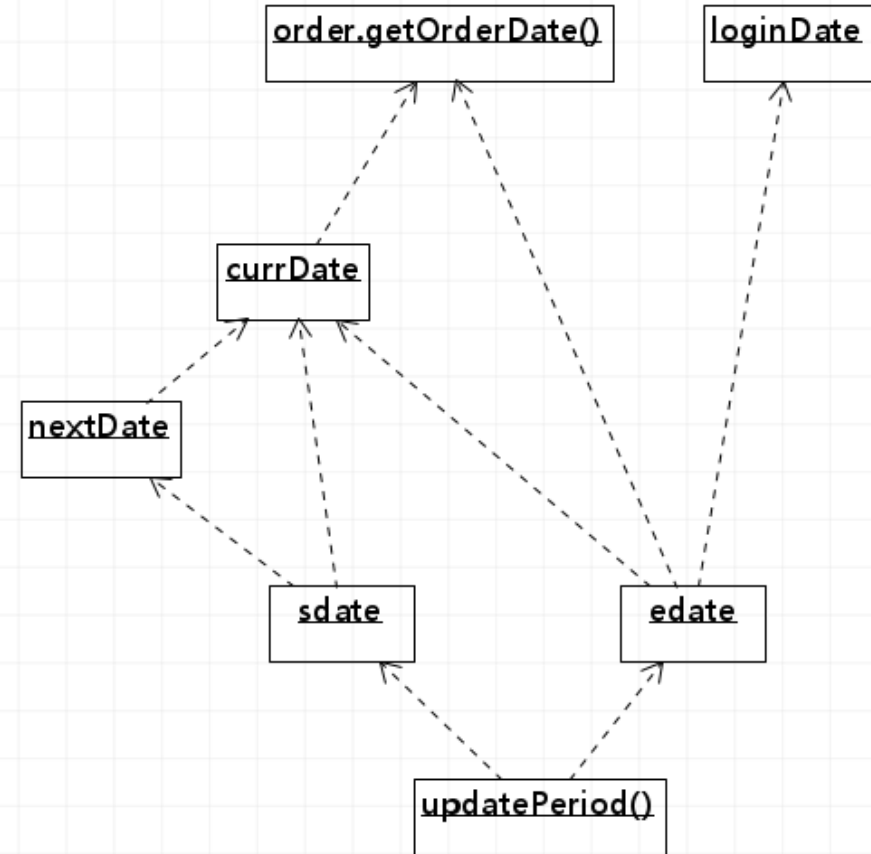
- 요소 간 호출 관계
- 시간 흐름에 따른 실행 순서
- 비동기 실행 표현



의존 그래프

클래스/변수/필드/메서드 간 의존 관계를
그림으로 표현

- 영향도 분석에 도움
- 클래스 D, 객체 D, 피처 스케치 등 사용



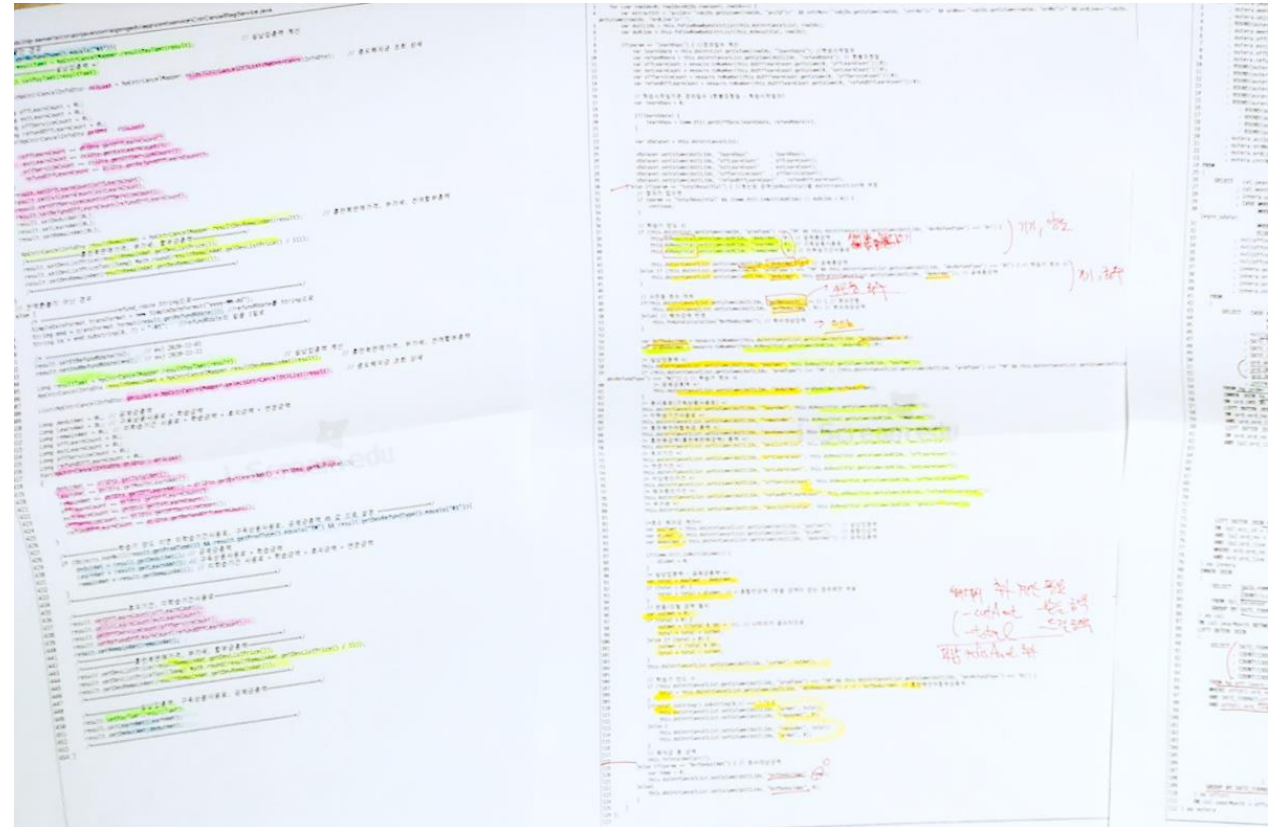
코드 분석에 도움이 되는 보조 수단

코드 출력

여러 메서드의 코드를
펼쳐 놓고 볼 수 있음

가능하다면 A3에 출력

형광펜, 화살표 등을 사용해서
관계 표시



코드 분석에 도움이 되는 보조 수단

함께 코드 보며 분석하기

각자의 이해가 모여 더 큰 이해

상호 지식 보완

같이 모여 코드 보기

- 3명 이하: 3명이 모여 앉을 수 있는 의자
- 4명 이상: 회의실, 큰 모니터
- 또는 화면 공유/화상 미팅



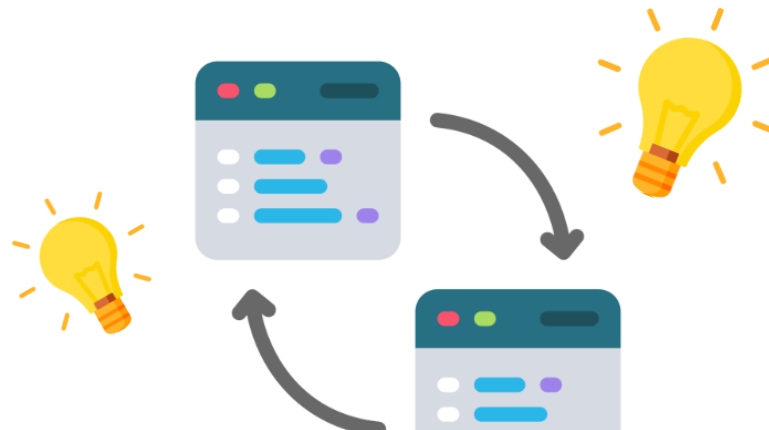
코드 분석에 도움이 되는 보조 수단

스크래치 리팩토링

코드 이해 목적으로 진행하는 리팩토링

- 리팩토링 과정에서 코드의 의미를 이해
- 실제 리팩토링하는 것이 목적 아님

함께 모여 하면 더 효과적



레거시에 테스트 코드 만들기

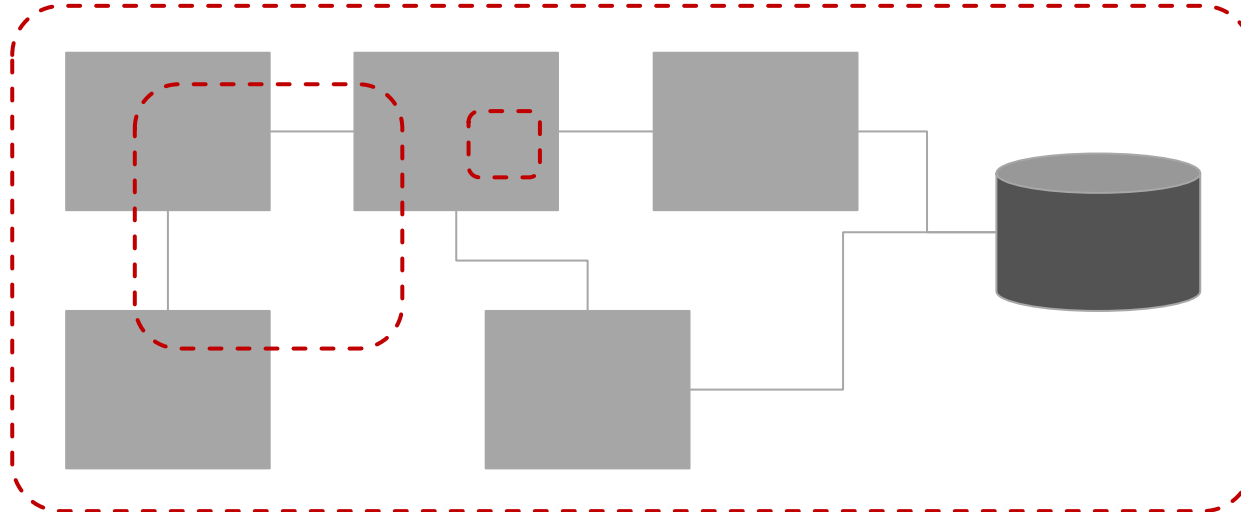


레거시에 테스트 코드 만들기

방법1 범위를 좁혀서 테스트 작성

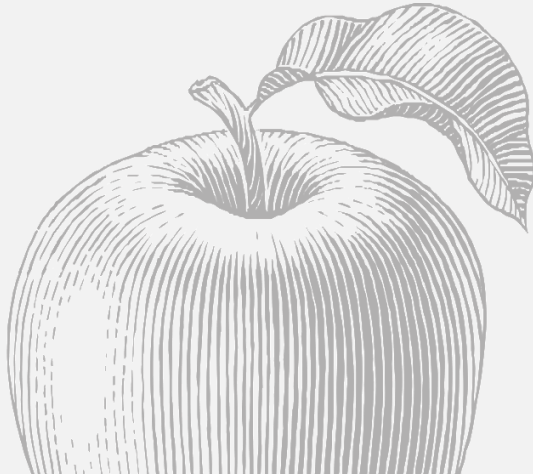
방법2 대체 구현을 이용해서 테스트 작성

방법3 범위를 넓혀서 테스트 작성



“

레거시에 테스트 코드 만들기 방법 1

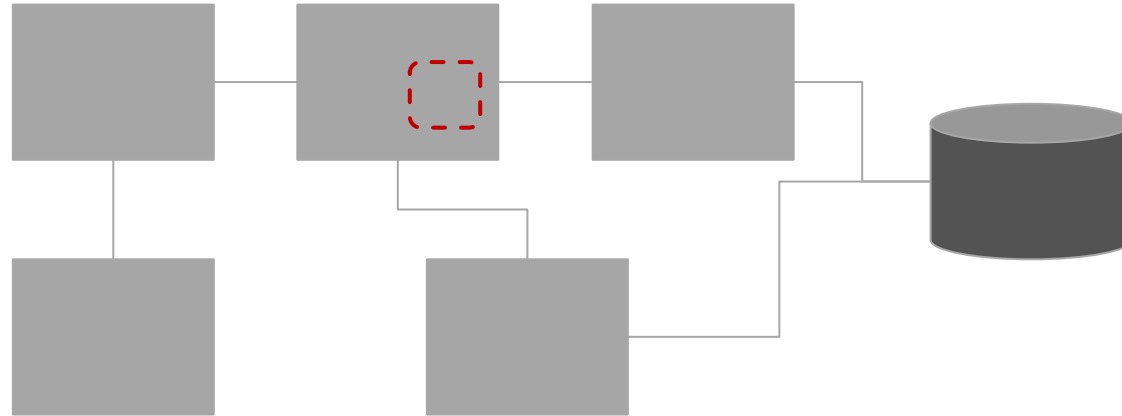


레거시에 테스트 코드 만들기 - 방법 1

범위를 좁혀서 테스트 만들기

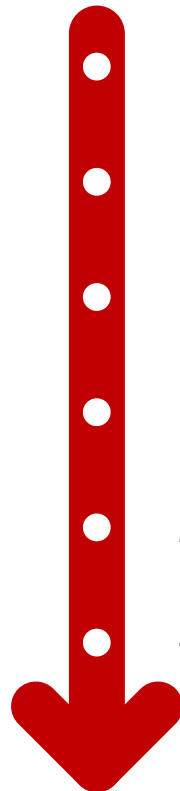
테스트 만들 대상을 기존 코드와 분리해서 테스트 작성

- 일부 코드/로직/기능만 테스트하고 싶을 때 사용
- 새로 추가한 코드만 테스트하고 싶을 때 사용
- 예: 계산 로직 분리



레거시에 테스트 코드 만들기 - 방법 1

진행과정



- 테스트할 대상/기능/로직 확인
- 테스트 대상 코드를 한 곳으로 모음
- 테스트 대상 코드의 입력 파라미터와 결과 값 확인
- 필요하면 입력과 결과를 위한 타입 추가
- 그런 뒤 해당 코드를 별도 메서드/클래스로 분리
- 분리한 대상에 테스트 코드 추가

레거시에 테스트 코드 만들기 - 방법 1

예: 계산 로직 테스트 코드 만들기 - 테스트 대상 찾기

```
public void provideServicePeriod(Long ordNo, LocalDate loginDate){
    Order order = getOrder(ordNo);
    if (order.getGubun().equals("A")) {
        LocalDate edate = YearMonth.from(order.getOrderDate().atEndOfMonth());
        updatePeriod(period, order.getOrderDate(), edate);
    } else {
        LocalDate edate = null;
        if (order.getUnit().equals("D")) {
            edate = loginDate.plusDays(order.getQty());
        } else if (order.getUnit().equals("M")) {
            edate = loginDate.plusMonths(order.getQty());
        }
        updatePeriod(ordNo, loginDate, edate);
    }
    ...
}
```


레거시에 테스트 코드 만들기 - 방법 1

테스트 대상 모으기

```
public void provideServicePeriod(Long ordNo, LocalDate loginDate) {  
    Order order = getOrder(ordNo);  
    LocalDate sdate = null;  
    LocalDate edate = null;  
    if (order.getGubun().equals("A")) {  
        sdate = order.getOrderDate();  
        edate = YearMonth.from(order.getOrderDate()).atEndOfMonth();  
    } else {  
        sdate = loginDate;  
        if (order.getUnit().equals("D")) {  
            edate = loginDate.plusDays(order.getQty());  
        } else if (order.getUnit().equals("M")) {  
            edate = loginDate.plusMonths(order.getQty());  
        }  
    }  
    updatePeriod(ordNo, sdate, edate);  
}
```

레거시에 테스트 코드 만들기 - 방법 1

입력, 결과 값

```
public void provideServicePeriod(Long ordNo, LocalDate loginDate) {  
    Order order = getOrder(ordNo);  
    LocalDate sdate = null;  
    LocalDate edate = null;  
    if (order.getGubun().equals("A")) {  
        sdate = order.getOrderDate();  
        edate = YearMonth.from(order.getOrderDate().atEndOfMonth());  
    } else {  
        sdate = loginDate;  
        if (order.getUnit().equals("D")) {  
            edate = loginDate.plusDays(order.getQty());  
        } else if (order.getUnit().equals("M")) {  
            edate = loginDate.plusMonths(order.getQty());  
        }  
    }  
    updatePeriod(ordNo, sdate, edate);  
}
```

레거시에 테스트 코드 만들기 - 방법 1

입력, 결과 값 확인

```
public void provideServicePeriod(Long ordNo, LocalDate loginDate) {  
    Order order = getOrder(ordNo);  
    LocalDate sdate = null;  
    LocalDate edate = null;  
    if (order.getGubun().equals("A")) {  
        sdate = order.getOrderDate();  
        edate = YearMonth.from(order.getOrderDate()).atEndOfMonth();  
    } else {  
        sdate = loginDate;  
        if (order.getUnit().equals("D")) {  
            edate = loginDate.plusDays(order.getQty());  
        } else if (order.getUnit().equals("M")) {  
            edate = loginDate.plusMonths(order.getQty());  
        }  
    }  
    updatePeriod(ordNo, sdate, edate);  
}
```

입력

- order : Order
- loginDate : LocalDate

결과

- sdate : LocalDate
- edate : LocalDate

레거시에 테스트 코드 만들기 - 방법 1

필요하면 타입 추가

```
public void provideServicePeriod(Long ordNo, LocalDate loginDate) {  
    Order order = getOrder(ordNo);  
    Period period = null;  
    if (order.getGubun().equals("A")) {  
        period = Period.of(order.getOrderDate(),  
            YearMonth.from(order.getOrderDate().atEndOfMonth());  
    } else {  
        if (order.getUnit().equals("D")) {  
            period = Period.of(loginDate, loginDate.plusDays(order.getQty()));  
        } else if (order.getUnit().equals("M")) {  
            period = Period.of(loginDate, loginDate.plusMonths(order.getQty()));  
        }  
    }  
    updatePeriod(ordNo, period.getSdate(), period.getEdate());  
}
```

결과

- sdate : LocalDate
- edate : LocalDate

결과 위한 타입 추가

```
public class Period {  
    private LocalDate sdate;  
    private LocalDate edate;  
    ...  
}
```

레거시에 테스트 코드 만들기 - 방법 1

코드 분리

```
public void provideServicePeriod(Long ordNo, LocalDate loginDate) {
    Order order = getOrder(ordNo);
    Period period = PeriodCalculator.calculate(order, loginDate);
    updatePeriod(ordNo, period.getSdate(), period.getEdate());
}
```

```
public static class PeriodCalculator {
    public static Period calculate(Order order, LocalDate loginDate) {
        Period period = null;
        if (order.getGubun().equals("A")) {
            period = Period.of(order.getOrderDate(),
                YearMonth.from(order.getOrderDate().atEndOfMonth());
        } else {
            if (order.getUnit().equals("D")) {
                period = Period.of(loginDate, loginDate.plusDays(order.getQty()));
            } else if (order.getUnit().equals("M")) {
                period = Period.of(loginDate, loginDate.plusMonths(order.getQty()));
            }
        }
        return period;
    }
}
```

레거시에 테스트 코드 만들기 - 방법 1

테스트 코드 추가

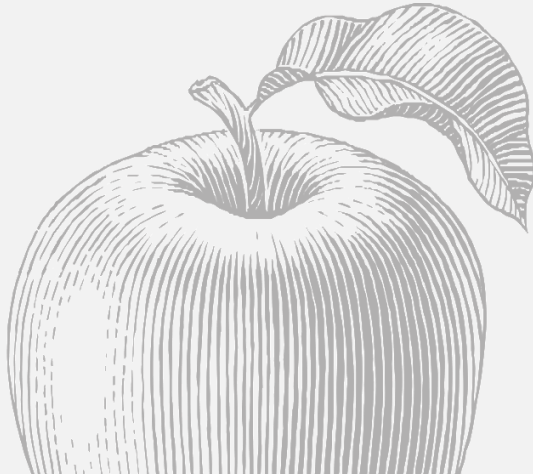
분리한 대상에 대한 테스트 코드 추가

다양한 케이스를 추가하면서 테스트 커버리지를 높임

```
public class PeriodCalculatorTest {  
  
    @Test  
    void gubunA() {  
        Order order = Order.builder()  
            .gubun("A")  
            .orderDate(LocalDate.of(2021, 5, 5))  
            .build();  
        Period p = PeriodCalculator.calculate(order, LocalDate.of(2021, 5, 6));  
        assertThat(p.getSdate()).isEqualTo(LocalDate.of(2021, 5, 6));  
        assertThat(p.getEdate()).isEqualTo(LocalDate.of(2021, 5, 31));  
    }  
}
```

“

레거시에 테스트 코드 만들기 방법 2



레거시에 테스트 코드 만들기 - 방법 2

대체 구현을 사용해서 테스트 만들기

테스트 대상이 사용(의존)하는 객체/기능이 존재할 때 사용

의존 대상의 구현을 대체할 대역을 만들어서 테스트 작성

필요한 것 두 가지

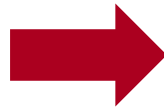
- 의존 대상 : 대역을 만들 수 있는 구조로 변경 필요
- 테스트 대상 : 대역을 사용할 수 있는 구조로 변경 필요

레거시에 테스트 코드 만들기 - 방법 2

대역을 만들 수 있는 구조로 변경: 방법 1/3

의존하는 구현 코드를 새 타입으로 이동 → 새 타입 사용

```
public class Some {  
    public void doSome() {  
        ...  
        doAny code 1  
        doAny code 2  
        ...  
    }  
}
```



```
public class Any {  
    public void doAny() {  
        doAny code1  
        doAny code2  
    }  
}
```

```
public class Some {  
    private Any any;  
  
    public void some() {  
        ...  
        any.doAny();  
        ...  
    }  
}
```

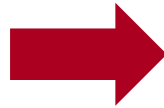
레거시에 테스트 코드 만들기 - 방법 2

대역을 만들 수 있는 구조로 변경: 방법 2/3

의존 대상에서 인터페이스 추출 → 인터페이스 사용

클래스에 대한 대역을 쉽게 생성할 수 있어 상대적으로 사용 빈도 낮지만
설계 관점에서 구조 변화가 필요할 때 종종 사용

```
public class AnyOther {  
    public void doAny() { ... }  
    public void doOther() { ... }  
}  
  
public class Some {  
    private AnyOther anyOther;  
    public void doSome() {  
        ...  
        anyOther.doAny();  
        ...  
    }  
}
```



```
public interface Any {  
    void doAny();  
}  
  
public class Some {  
    private Any any;  
    public void doSome() {  
        ...  
        any.doAny();  
        ...  
    }  
}
```

```
public class AnyOther  
    implements Any {  
    public void doAny() { ... }  
    public void doOther() { ... }  
}
```

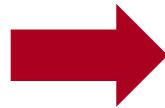
레거시에 테스트 코드 만들기 - 방법 2

대역을 만들 수 있는 구조로 변경: 방법 3/3

테스트 대상에서 구현을 대체할 부분을 **protected** 메서드로 분리

대체 구현을 제공할 하위 클래스에서 메서드 재구현

```
public class Any {  
    public void doAny() {  
        ...  
        code1 // 대체 필요 부분  
        ...  
    }  
}
```



```
public class Any {  
    public void doAny() {  
        ...  
        doCode1();  
        ...  
    }  
  
    protected void doCode1() {  
        code1;  
    }
```

```
public class AnyDouble extends Any {  
  
    @Override  
    protected void doCode1() {  
        // 대체 구현  
    }
```

레거시에 테스트 코드 만들기 - 방법 2

테스트 대상이 대역을 사용할 수 있는 구조로 변경

“

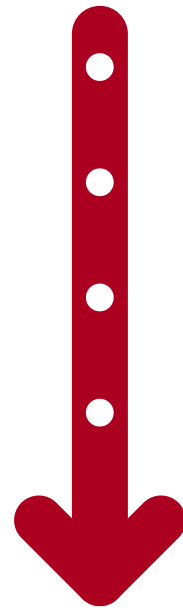
테스트 대상에
의존 대상 주입 가능하게

생성자나 세터 메서드 이용



레거시에 테스트 코드 만들기 - 방법 2

진행과정



• 테스트 대상이 의존(사용)하는 코드 확인

• 대역을 생성할 수 있는 구조로 변경

• 테스트 대상이 대역을 사용할 수 있도록 구조 변경

• 대역을 사용해서 테스트 코드 작성

예제코드

테스트 대상

- createAccount 메서드
- 주요 로직 검증 목적

의존 대상 (구현 대체 후보)

- RestTemplate과 관련 코드
- ServiceAccountRepository

```
@Service
public class CreateAccountService {
    @Autowired
    private ServiceAccountRepository serviceAccountRepository;
    @Value("${check.url}")
    private String checkUrl;
    private RestTemplate restTemplate = new RestTemplate();

    @Transactional
    public Long createAccount(String email) {
        if (StringUtils.isEmpty(email)) throw new AppException("오류");
        ResponseEntity<AuthApiResponse> authResp =
            restTemplate.getForEntity(
                checkUrl + "?email=" + email,
                AuthApiResponse.class);
        AuthApiResponse resp = authResp.getBody();
        if (resp.getCode() == AuthApiCode.DUP)
            throw new DupException();
        ServiceAccount acc = ServiceAccount.builder()
            .accStatus("R").email(email).build();
        serviceAccountRepository.save(acc);
        return acc.getAcclId();
    }
}
```

의존 대상 1: RestTemplate 및 관련 코드

이메일 중복 여부 확인에 사용

외부 연동 포함

구현 대체 대상

```
@Service
public class CreateAccountService {
    @Autowired
    private ServiceAccountRepository serviceAccountRepository;
    @Value("${check.url}")
    private String checkUrl;
    private RestTemplate restTemplate = new RestTemplate();

    @Transactional
    public Long createAccount(String email) {
        if (StringUtils.isEmpty(email)) throw new AppException("입력오류");
        ResponseEntity<AuthApiResponse> authResp =
            restTemplate.getForEntity(
                checkUrl + "?email=" + email,
                AuthApiResponse.class);
        AuthApiResponse resp = authResp.getBody();
        if (resp.getCode() == AuthApiCode.DUP)
            throw new DupException();
        ServiceAccount acc = ServiceAccount.builder()
            .accStatus("R").email(email).build();
        serviceAccountRepository.save(acc);
        return acc.getAcclId();
    }
}
```

관련 코드를 모아서 새 타입 생성

```
@Service
public class CreateAccountService {
    @Autowired
    private ServiceAccountRepository serviceAccountRepository;
    @Value("${check.urk}")
    private String checkUrl;
    private RestTemplate restTemplate = new RestTemplate();

    @Transactional
    public Long createAccount(String email) {
        if (StringUtils.isEmpty(email)) throw new ApplicationException("오류");
        ResponseEntity<AuthApiResponse> authResp =
            restTemplate.getForEntity(
                checkUrl + "?email=" + email,
                AuthApiResponse.class);
        AuthApiResponse resp = authResp.getBody();
        if (resp.getStatusCode() == AuthApiCode.DUP)
            throw new DupException();
        ServiceAccount acc = ServiceAccount.builder()
            .accStatus("R").email(email).build();
        serviceAccountRepository.save(acc);
        return acc.getAcclId();
    }
}
```



```
@Service
public class EmailChecker {
    @Value("${check.urk}")
    private String checkUrl;
    private RestTemplate restTemplate = new RestTemplate();

    public void checkDuplicate(String email) {
        ResponseEntity<AuthApiResponse> authResp =
            restTemplate.getForEntity(
                checkUrl + "?email=" + email,
                AuthApiResponse.class);
        AuthApiResponse resp = authResp.getBody();
        if (resp.getStatusCode() == AuthApiCode.DUP)
            throw new DupException();
    }
}
```


테스트 대상이 새 타입 사용하도록 변경

```
@Service
public class CreateAccountService {
    @Autowired
    private ServiceAccountRepository serviceAccountRepository;
    @Autowired
    private EmailChecker emailChecker;

    @Transactional
    public Long createAccount(String email) {
        if (StringUtils.isEmpty(email)) throw new AppException("오류");
        emailChecker.checkDuplication(email);
        ServiceAccount acc = ServiceAccount.builder()
            .accStatus("R").email(email).build();
        serviceAccountRepository.save(acc);
        return acc.getAcclId();
    }
}
```

의존 대상 2 : ServiceAccountRepository

인터페이스 타입 + 메서드가 몇 개 없는 리포지토리
→ 가짜 구현 대역

```
public interface ServiceAccountRepository
    extends Repository<ServiceAccount, Long> {
    Optional<ServiceAccount> findById(Long id);

    @Query(value = "update service_acc ...생략",
        nativeQuery = true)
    void cancelConfirm3(@Param("cntrlId") Long cntrlId);

    void save(ServiceAccount acc);
}
```

```
public class MemoryServiceAccountRepository
    implements ServiceAccountRepository {
    private long nextId = 1;
    private Map<Long, ServiceAccount> values = new HashMap<>();
    public Optional<ServiceAccount> findById(Long id) {
        return Optional.ofNullable(values.get(id));
    }

    public void cancelConfirm3(Long cntrlId) {
        throw new RuntimeException("not supported");
    }

    public void save(ServiceAccount acc) {
        acc.setAcId(nextId++);
        values.put(acc.getAcId(), acc);
    }
}
```

테스트 대상이 대역을 사용할 수 있도록 구조 변경

1 생성자 사용

```
@Service
public class CreateAccountService {
    private ServiceAccountRepository serviceAccountRepository;
    private EmailChecker emailChecker;

    public CreateAccountService(
        ServiceAccountRepository serviceAccountRepository,
        EmailChecker emailChecker) {
        this.serviceAccountRepository = serviceAccountRepository;
        this.emailChecker = emailChecker;
    }
}
```

2 세터 사용

```
@Service
public class CreateAccountService {
    @Autowired
    private ServiceAccountRepository serviceAccountRepository;
    @Autowired
    private EmailChecker emailChecker;

    public void setServiceAccountRepository(
        ServiceAccountRepository serviceAccountRepository) {
        this.serviceAccountRepository = serviceAccountRepository;
    }

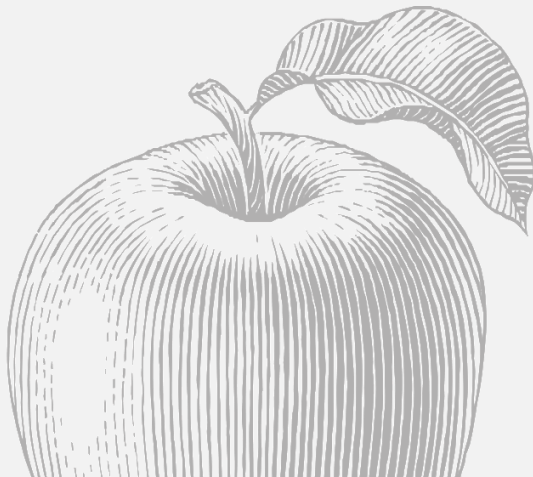
    public void setEmailChecker(EmailChecker emailChecker) {
        this.emailChecker = emailChecker;
    }
}
```

대역을 사용해서 테스트 작성

```
class CreateAccountServiceTest {  
    private EmailChecker mockEmailChecker = mock(EmailChecker.class);  
    private ServiceAccountRepository memoryRepo = new MemoryServiceAccountRepository();  
    private CreateAccountService service;  
  
    @BeforeEach  
    void setUp() {  
        service = new CreateAccountService(memoryRepo, mockEmailChecker);  
    }  
  
    @Test  
    void emailDup() {  
        willThrow(new DupException()).given(mockEmailChecker).checkDuplication("a@a.com");  
  
        assertThatCode(() -> service.createAccount("a@a.com")).isInstanceOf(DupException.class);  
    }  
  
    @Test  
    void newAccountCreated() {  
        Long newId = service.createAccount("a@a.com");  
  
        ServiceAccount newAcc = memoryRepo.findById(newId).get();  
        assertThat(newAcc.getEmail()).isEqualTo("a@a.com");  
        assertThat(newAcc.getAccStatus()).isEqualTo("R");  
    }  
}
```

“

레거시에 테스트 코드 만들기 방법 3



레거시에 테스트 코드 만들기 - 방법 3

범위를 넓혀서 테스트 만들기

좁은 범위 테스트가 어려운 경우

- 의존 대상이 많아 특정 범위만 테스트 만들기 어려움
- 테스트를 만들기 위해 변경해야 하는 코드가 너무 많음
- 코드 의미를 알 수 없어 테스트 대상 범위를 좁히기 어려움
- 일부 로직이 쿼리에 위치함

범위를 가능한 넓혀 다양한 구성 요소 간 연동을 포함하는 테스트 코드 작성

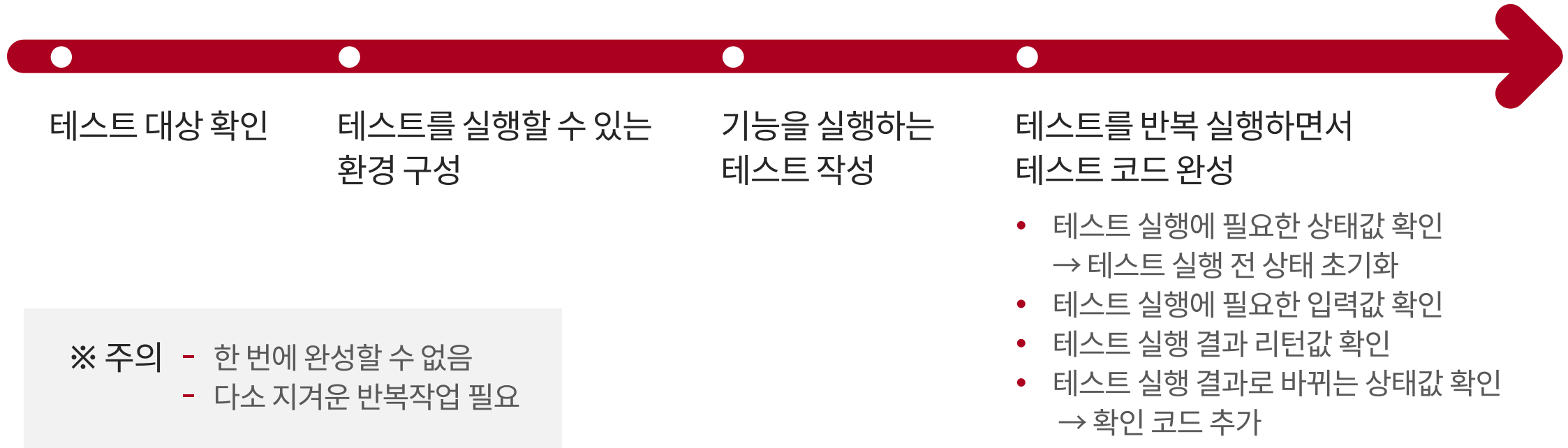
- DB 연동/외부 연동을 포함하기도 함
- API 또는 제공 기능 단위로 통합 테스트

주요 목적

- 레거시 코드 동작 이해 (Characterization Test)
- 리팩토링 사전 작업

레거시에 테스트 코드 만들기 - 방법 3

진행과정



레거시에 테스트 코드 만들기 - 방법 3

예: 동작 이해하기 위한 테스트

환불 확정을
처리하는 로직

- 여러 모델 변경
 - 계약 상태 변경
 - 환불 신청 상태 변경
 - 서비스 기간 상태 변경
 - 계정 상태 변경
 - 등등등
- 일부 로직이 쿼리에 포함

컨트롤러는 서비스의
한 메서드를 호출

서비스의 메서드를 테스트 대상으로 선택

스프링 프레임워크 사용

컨트롤러

서비스

리포지토리

리포지토리

리포지토리

레거시에 테스트 코드 만들기 - 방법 3

코드 일부 분석

```
public class ContractCancelService {  
    ...  
    @Transactional  
    public void saveCntrCancel(CntrCancelInfoDto dto) {  
        String status = contractCancelRepository.findByCntrId(dto.getCntrId()).get().getCnlStatus();  
        if ("2".equals(status) && "Y".equals(dto.getConfirmYn())) {  
            throw new AppException("이미 확정");  
        } else if ("3".equals(status) && !"Y".equals(dto.getConfirmYn())) {  
            throw new AppException("이미 완료");  
        }  
        if ("Y".equals(dto.getConfirmYn()) && dto.getConfirmUserId() == null) {  
            dto.setProcType("confirm");  
            dto.setConfirmUserId(dto.getUserId());  
            updateCancelStatus(dto);  
        } else if (!"Y".equals(dto.getConfirmYn()) && dto.getApproverId() == null) {  
            dto.setProcType("approve");  
            dto.setApproverId(dto.getUserId());  
            updateCancelStatus(dto);  
        }  
    }  
}
```

```
private void updateCancelStatus(CntrCancelInfoDto dto) {
    if ("confirm".equals(dto.getProcType())) {
        servicePeriodHRepository.cancelConfirm1(dto.getCntrId());
        servicePeriodRepository.cancelConfirm2(dto.getCntrId());
        serviceAccountRepository.cancelConfirm3(dto.getCntrId());
    }
    String cnlStatus = null;
    String cntrStatus = null;
    if ("confirm".equals(dto.getProcType())) {
        cnlStatus = "2";
        cntrStatus = "12";
    } else if ("approve".equals(dto.getProcType())) {
        cnlStatus = "3";
        cntrStatus = "13";
    }
    ContractCancel contractCancel = contractCancelRepository.findById(dto.getCntrId())
        .orElseThrow(() -> new AppException("계약 취소 없음"));
    contractCancel.setCnlStatus(cnlStatus);
    if (dto.getConfirmUserId() != null) {
        contractCancel.setConfirmUserId(dto.getConfirmUserId());
    }
    if (dto.getApproverId() != null) {
        contractCancel.setApproverId(dto.getApproverId());
    }

    Contract contract = contractRepository.findById(dto.getCntrId()).orElseThrow(() -> new AppException("계약 없음"));
    contract.setCntrStatus(cntrStatus);
}
}
```

레거시에 테스트 코드 만들기 - 방법 3

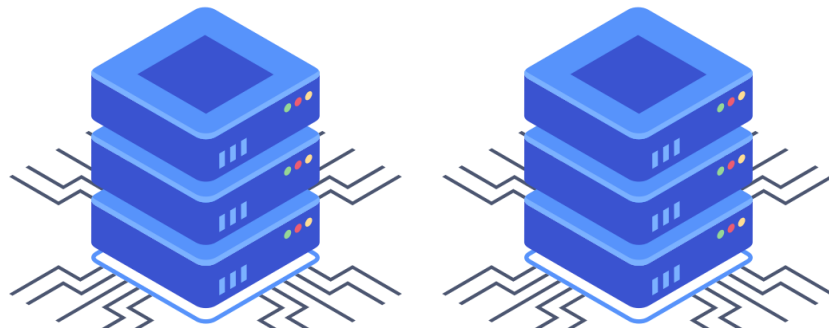
테스트 환경 준비

로컬에 운영환경과 동일한 DBMS 설치

- 특정 DB에서만 동작하는 전용 쿼리 실행 위함
- 테스트 실행 후 변경된 데이터 확인 용이
- 다른 개발자와 동일한 데이터를 변경하지 않기 위함

필요한 DB 테이블 생성

- 방법1: 로컬에 미리 테이블을 생성, 변경되면 반영
- 방법2: 테스트를 실행할 때마다 테이블 초기화
(삭제하고 재생성)



레거시에 테스트 코드 만들기 - 방법 3

환경 준비

테스트 데이터 초기화를 위한 보조 클래스

- 각 테스트 메서드를 실행할 때마다 테스트 실행 환경을 정리하기 위함

테스트 대상 코드를 보면서 당장 알게 된 테이블을 초기화 대상으로 추가

- 테스트를 진행하면서 점진적으로 추가

```
@Component
@Profile("localdb")
public class GivenHelper {
    private JdbcTemplate jdbcTemplate;

    public GivenHelper(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    public void clearAll() {
        jdbcTemplate.execute("truncate table cntr");
        jdbcTemplate.execute("truncate table cntr_cancel");
        ...
    }
}
```

레거시에 테스트 코드 만들기 - 방법 3

최초 테스트 코드 작성

테스트 대상을 실행할 수 있는
기반 작업

- 테스트 대상 자체에서 익셉션이 발생할 때까지 테스트 반복 실행
- 이를 통해 테스트 실행 환경(DB 연결 등) 완료



```
@SpringBootTest
@ActiveProfiles("localdb")
class ContractCancelServiceIT {
    @Autowired GivenHelper givenHelper;
    @Autowired ContractCancelService contractCancelService;

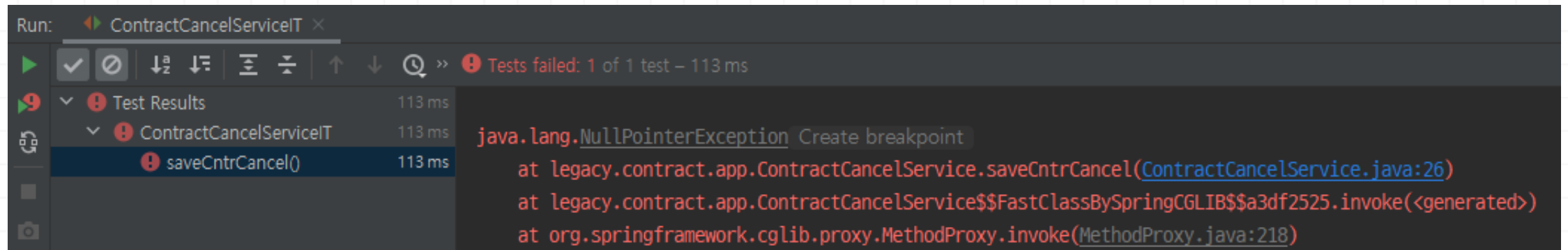
    @BeforeEach
    void setUp() { givenHelper.clearAll(); }

    @Test
    void saveCntrCancel() {
        contractCancelService.saveCntrCancel(null);
    }
}
```

레거시에 테스트 코드 만들기 - 방법 3

테스트 실행과 에러 - 원인 제거 - 테스트 보완 반복

예: 테스트 실행 결과



```
Run: ContractCancelServiceIT x
Tests failed: 1 of 1 test - 113 ms
Test Results 113 ms
  ContractCancelServiceIT 113 ms
    saveCntrCancel() 113 ms
java.lang.NullPointerException Create breakpoint
at legacy.contract.app.ContractCancelService.saveCntrCancel(ContractCancelService.java:26)
at legacy.contract.app.ContractCancelService$$FastClassBySpringCGLIB$$a3df2525.invoke(<generated>)
at org.springframework.cglib.proxy.MethodProxy.invoke(MethodProxy.java:218)
```

레거시에 테스트 코드 만들기 - 방법 3

원인 분석과 에러 대응 안 도출

dto.getCntrId() 값 사용

findById가 리턴한
Optional<ContractCancel>이 존재해야 함ContractCancel의
getCnlStatus()가
"2"나 "3"이면 안 됨확정 기능 테스트라
dto.getConfirmYn()이
"Y"여야 함dto.getUserId()
값이 필요함dto.getConfirmUserId()는
null이어야 함

```
public void saveCntrCancel(CntrCancelInfoDto dto) {  
    String status = contractCancelRepository.findById(dto.getCntrId()).get().getCnlStatus();  
    if ("2".equals(status) && "Y".equals(dto.getConfirmYn())) {  
        throw new AppException("이미 확정");  
    } else if ("3".equals(status) && !"Y".equals(dto.getConfirmYn())) {  
        throw new AppException("이미 완료");  
    }  
    if ("Y".equals(dto.getConfirmYn()) && dto.getConfirmUserId() == null) {  
        dto.setProcType("confirm");  
        dto.setConfirmUserId(dto.getUserId());  
        updateCancelStatus(dto);  
    } else if (!"Y".equals(dto.getConfirmYn()) && dto.getApproverId() == null) {  
        dto.setProcType("approve");  
        dto.setApproverId(dto.getUserId());  
        updateCancelStatus(dto);  
    }  
}
```

레거시에 테스트 코드 만들기 - 방법 3

테스트 코드 수정

```
@SpringBootTest
@ActiveProfiles("localdb")
class ContractCancelServiceIT {

    ...
    @Autowired
    private ContractCancelRepository contractCancelRepository;

    @Test
    void saveCntrCancel() {
        contractCancelRepository.save(
            ContractCancel.builder().cntrId(1L).cnlStatus("").build()
        );

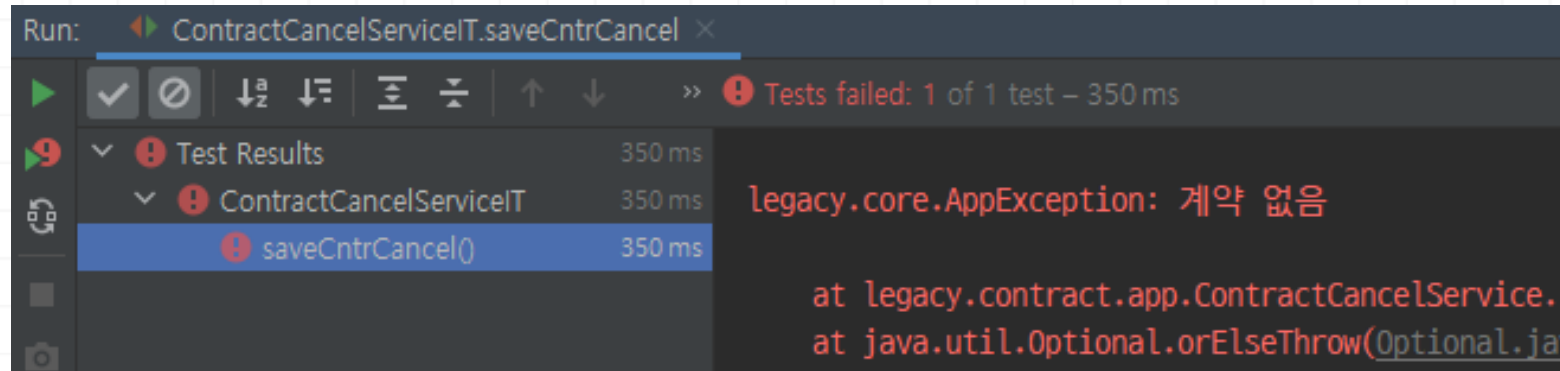
        CntrCancelInfoDto dto = new CntrCancelInfoDto();
        dto.setCntrId(1L);
        dto.setConfirmYn("Y");
        dto.setUserId(5L);
        contractCancelService.saveCntrCancel(dto);
    }
}
```

ContractCancel 존재
cnlStatus는 2나 3 안 됨

dto.getCntrId() 사용
dto.getConfirmYn()는 "Y"
dto.getUserId() 값 필요
dto.getConfirmUserId()는 null

레거시에 테스트 코드 만들기 - 방법 3

테스트 실행



findById가 리턴한
Optional<Contract>가 존재해야 함

```
private void updateCancelStatus(CntrCancelInfoDto dto) {  
    ...  
    Contract contract = contractRepository.findById(dto.getCntrId()).orElseThrow() -> new AppException("계약 없음");  
    contract.setCntrStatus(cntrStatus);  
}
```

레거시에 테스트 코드 만들기 - 방법 3

테스트 코드 수정

```
...
@Autowired
private ContractRepository contractRepository;

@Test
void saveCntrCancel() {
    contractCancelRepository.save(
        ContractCancel.builder().cntrlId(1L).cnlStatus("1").build()
    );
    contractRepository.save(
        Contract.builder().cntrlId(1L).build()
    );

    CntrCancelInfoDto dto = new CntrCancelInfoDto();
    dto.setCntrlId(1L);
    dto.setConfirmYn("Y");
    dto.setUserId(5L);
    contractCancelService.saveCntrCancel(dto);
}
```

Contract 존재

레거시에 테스트 코드 만들기 - 방법 3

테스트 실행

cntr 테이블 acc_id가
null이면 안 됨

```
Run: ContractCancelServiceIT.saveCtrCancel x
Tests failed: 1 of 1 test - 266 ms
Test Results
  ContractCancelServiceIT
    saveCtrCancel()
      2021-05-18 22:11:45.276 WARN 18960 --- [main] o.h.engine.jdbc.spi.SqlExceptionHelper : SQL Error: 1048, SQLState: 23000
      2021-05-18 22:11:45.276 ERROR 18960 --- [main] o.h.engine.jdbc.spi.SqlExceptionHelper : (conn=17) Column 'acc_id' cannot be null
      org.springframework.dao.DataIntegrityViolationException: could not execute statement; SQL [n/a]; constraint [null]; nested exception is org.hi
      at org.springframework.orm.jpa.vendor.HibernateJpaDialect.convertHibernateAccessException(HibernateJpaDialect.java:276)
      at org.springframework.orm.jpa.vendor.HibernateJpaDialect.translateExceptionIfPossible(HibernateJpaDialect.java:233)
      at org.springframework.orm.jpa.JpaTransactionManager.doCommit(JpaTransactionManager.java:566)
      at org.springframework.transaction.support.AbstractPlatformTransactionManager.processCommit(AbstractPlatformTransactionManager.java:743)
      at org.springframework.transaction.support.AbstractPlatformTransactionManager.commit(AbstractPlatformTransactionManager.java:711)
      at org.springframework.transaction.interceptor.TransactionAspectSupport.commitTransactionAfterReturning(TransactionAspectSupport.java:654)
      at org.springframework.transaction.interceptor.TransactionAspectSupport.invokeWithinTransaction(TransactionAspectSupport.java:407)
      at org.springframework.transaction.interceptor.TransactionInterceptor.invoke(TransactionInterceptor.java:119)
      at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
      at org.springframework.dao.support.PersistenceExceptionTranslationInterceptor.invoke(PersistenceExceptionTranslationInterceptor.java:137)
      at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
      at org.springframework.data.jpa.repository.support.CrudMethodMetadataPostProcessor$CrudMethodMetadataPopulatingMethodInterceptor.invoke(CrudMethodMetadataPostProcessor$CrudMethodMetadataPopulatingMethodInterceptor.java:186)
      at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
      at org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke(ExposeInvocationInterceptor.java:97)
      at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
      at org.springframework.aop.framework.JdkDynamicAopProxy.invoke(JdkDynamicAopProxy.java:215) <1 internal line>
      at legacy.contract.app.ContractCancelServiceIT.saveCtrCancel(ContractCancelServiceIT.java:27) <31 internal lines>
```

레거시에 테스트 코드 만들기 - 방법 3

테스트 코드 수정

Contract의 acclId 필수
Contract의 cntrStatus 필수

```
@Test
void saveCntrCancel() {
    contractCancelRepository.save(
        ContractCancel.builder().cntrlId(1L).cnlStatus("").build()
    );
    contractRepository.save(
        Contract.builder().cntrlId(1L).acclId(10L).cntrStatus("").build()
    );

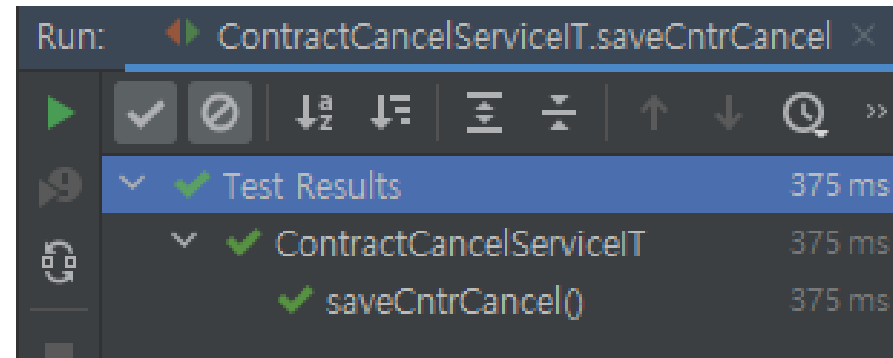
    CntrCancelInfoDto dto = new CntrCancelInfoDto();
    dto.setCntrlId(1L);
    dto.setConfirmYn("Y");
    dto.setUserId(5L);
    contractCancelService.saveCntrCancel(dto);
}
```

레거시에 테스트 코드 만들기 - 방법 3

테스트 실행

테스트가 통과되면 다음 중 선택

- 실행 결과를 확인하는 코드 추가
- 필요한 상황이 없는지 더 조사



레거시에 테스트 코드 만들기 - 방법 3

결과 확인 코드 추가

```

...
    dto.setProcType("confirm");
    dto.setConfirmUserId(dto.getUserId());
    updateCancelStatus(dto);
...

if ("confirm".equals(dto.getProcType())) {
    cnlStatus = "2";
    cntrStatus = "12";
    ...

ContractCancel contractCancel =
    contractCancelRepository.findById(dto.getCntrId())
        .orElseThrow(() -> new ApplicationException("계약 취소 없음"));
contractCancel.setCnlStatus(cnlStatus);
if (dto.getConfirmUserId() != null) {
    contractCancel.setConfirmUserId(dto.getConfirmUserId());
}

```

```

@Test
void saveCntrCancel() {
    ...생략

    CntrCancelInfoDto dto = new CntrCancelInfoDto();
    dto.setCntrId(1L);
    dto.setConfirmYn("Y");
    dto.setUserId(5L);
    contractCancelService.saveCntrCancel(dto);

    ContractCancel cc =
        contractCancelRepository.findById(1L).get();
    assertThat(cc.getCnlStatus()).isEqualTo("2");
    assertThat(cc.getConfirmUserId()).isEqualTo(5L);
}

```

레거시에 테스트 코드 만들기 - 방법 3

테스트 작성과 지식

테스트 코드를 만들고 통과시키는 과정에서 알게 된 사실

환불 확정을 실행하려면
적어도 다음 두 데이터가 필요
(상황, GIVEN)

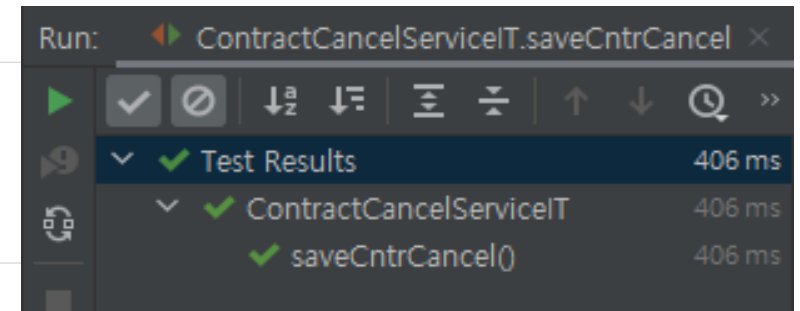
- ContractCancel: cnlStatus가 2나 3이 아님
- Contract가 존재해야 함

환불 확정을 요청할 때
다음 데이터 필요
(실행, WHEN)

- 확정할 계약 ID: cntrlId
- confirmYn 값은 Y
- userId 값이 필요

확정 실행하면
상태가 다음과 같이 바뀜
(결과, THEN)

- ContractCancel
 - cnlStatus가 2가 됨
 - confirmUserId가 요청할 때 전달한 userId로 설정됨



레거시에 테스트 코드 만들기 - 방법 3

결과 확인 코드 추가

```
String cnlStatus = null;
String cntrStatus = null;
if ("confirm".equals(dto.getProcType())) {
    cnlStatus = "2";
    cntrStatus = "12";
```

```
...
Contract contract = contractRepository.findById(dto.getCntrId())
    .orElseThrow(() -> new ApplicationException("계약 없음"));
contract.setCntrStatus(cntrStatus);
```

```
@Test
void saveCntrCancel() {
    ...생략
```

```
CntrCancelInfoDto dto = new CntrCancelInfoDto();
dto.setCntrId(1L);
dto.setConfirmYn("Y");
dto.setUserId(5L);
contractCancelService.saveCntrCancel(dto);
```

```
ContractCancel cc =
    contractCancelRepository.findById(1L).get();
assertThat(cc.getCnlStatus()).isEqualTo("2");
assertThat(cc.getConfirmUserId()).isEqualTo(5L);
```

```
Contract cont =
    contractRepository.findById(1L).get();
assertThat(cont.getCntrStatus()).isEqualTo("12");
}
```


레거시에 테스트 코드 만들기 - 방법 3

쿼리에 숨겨진 상황이나 결과 찾기

```
private void updateCancelStatus(CntrCancelInfoDto dto) {  
    if ("confirm".equals(dto.getProcType())) {  
        servicePeriodHRepository.cancelConfirm1(dto.getCntrId());  
        servicePeriodRepository.cancelConfirm2(dto.getCntrId());  
        serviceAccountRepository.cancelConfirm3(dto.getCntrId());  
    }  
}
```

service_period 값을 사용하는 service_period_h 생김

service_period 필요

```
@Query(value = "insert into service_period_h (cntr_id, chg_seq, period_cd, start_dt, end_dt) " +  
    "select sp.cntr_id, sp.chg_seq, sp.period_cd, sp.start_dt, sp.end_dt " +  
    "from service_period sp " +  
    "join cntr_cancel cc on cc.cntr_id = sp.cntr_id and cc.cnl_status = '1' " +  
    "where sp.cntr_id = :cntrId",  
    nativeQuery = true)  
void cancelConfirm1(@Param("cntrId") Long cntrId);
```

cntr_cancel의 cnl_status가 1이어야 함

레거시에 테스트 코드 만들기 - 방법 3

새로 알게된 테이블을 초기화 대상으로 추가

```
@Component
@Profile("localdb")
public class GivenHelper {
    private JdbcTemplate jdbcTemplate;

    public GivenHelper(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    public void clearAll() {
        jdbcTemplate.execute("truncate table cntr");
        jdbcTemplate.execute("truncate table cntr_cancel");
        jdbcTemplate.execute("truncate table service_period");
        jdbcTemplate.execute("truncate table service_period_h");
    }
}
```

레거시에 테스트 코드 만들기 - 방법 3

테스트 코드 수정

ServicePeriod 필요
ContractCancel의 cnlStatus는 1
ServicePeriodH 생김

```
...
@Autowired
ServicePeriodRepository servicePeriodRepository;
@Autowired
ServicePeriodHRepository servicePeriodHRepository;

@Test
void saveCntrCancel() {
    contractCancelRepository.save(
        ContractCancel.builder().cntrlId(1L).cnlStatus("1").build()
    );
    servicePeriodRepository.save(
        ServicePeriod.builder()
            .id(new ServicePeriodId(1L, 2L)).periodCd(1000)
            .build()
    );
    ...
    contractCancelService.saveCntrCancel(dto);
    ...
    ServicePeriodH sph = servicePeriodHRepository
        .findById(new ServicePeriodHId(1L, 2L)).get();
    assertThat(sph.getPeriodCd()).isEqualTo(1000);
}
```

테스트 코드 완성

정상적인 경우를 확인할 수 있는 수준까지 테스트 코드 완성
이후 점진적으로 예외 상황을 포함한 다양한 경우를 확인하는 코드 추가

정상적인 경우에 대한 테스트 코드를 만드는 시간도 적지 않게 걸림
상황을 만들기 위한 코드 작성도 많은 노력 필요

한 번에 완벽한 테스트를 만든다는 욕심은 버릴 것



테스트를 점진적으로 완성한 뒤

“

다음은 리팩토링



리팩토링



리팩토링

리팩토링

변경 비용을 낮추려면
이해/변경이 쉬운 구조로
점진적으로 리팩토링 해야 함

- 레거시에서 흔히 발생하는 상황
: 코드 이해에 2시간, 수정은 5분
- 리팩토링을 통해 코드 이해 시간을 줄일 필요

테스트가 있다면 과감하게
리팩토링 가능

- 코드 변경 후에 전과 동일하게 동작함을 확인
- Characterization Test

테스트가 없어도 필요하면
리팩토링 진행

- 현재의 위험 회피 → 미래에 더 큰 위험
- 예: 이름 변경, if절 조건 반전

리팩토링

미사용 코드 삭제

```
// TODO 삭제대상 2021-05-19 00으로 대체함
// someDeletingCode
// anyDeletingCode
```

주석으로 되어 있는 코드
삭제하기

- 나중에 쓸 지 몰라서 남겨둘 필요 없음
- 주석 처리한 날짜를 기록하고 일정 기간 뒤에 삭제

미사용 파라미터 삭제

미사용 메서드 삭제

미사용 변수 삭제

미사용 클래스 삭제

미사용 로직 삭제



리팩토링

매직 넘버

숫자, 문자열 리터럴 → 의미있는 이름 부여

상수, 열거형 등 사용



리팩토링

이름 변경

```
@Query(value = "insert into service_period_h (cntr_id, chg_seq, period_cd, start_dt, end_dt) " +  
  "select sp.cntr_id, sp.chg_seq, sp.period_cd, sp.start_dt, sp.end_dt " +  
  "from service_period sp " +  
  "join cntr_cancel cc on cc.cntr_id = sp.cntr_id and cc.cnl_status = '1' " +  
  "where sp.cntr_id = :cntrId",  
  nativeQuery = true)
```

servicePeriodHRepository.**cancelConfirm1**(dto.getCntrId())



servicePeriodHRepository.**addHistoryFromContactPeriod**(dto.getCntrId())

- 클래스, 메서드, 파라미터, 변수 이름을 의미에 맞게 변경
- 가장 쉽게 할 수 있는 리팩토링

리팩토링

변수 선언과 사용

```
int count;
```

... (10줄 코드, 이 사이에 count 변수 사용 X)

```
count = countsByCondition(lst, cond);
```



... (10줄 코드)

```
int count = countsByCondition(lst, cond);
```

- 변수는 사용 직전 위치로 이동
- 변수 선언 위치와 사용 위치가 멀리 떨어져 있으면 코드 이해에 부담 증가

리팩토링

변수 제거

```
int count = countsByCondition(lst, cond);  
if (count > 0) {  
  
}
```



```
count = countsBySome(lst);  
if (count == 0) {  
  
}
```

```
if (countsByCondition(lst, cond) > 0) {  
  
}  
  
if (countsBySome(lst) == 0) {  
  
}
```

- 긴 코드에서 값이 바뀌는 변수가 많을수록 코드 추적 어려움
- 필요하지 않은 변수는 가능하면 제거
- 한 변수를 여러 의미로 사용하지 않기

리팩토링

if 줄이기

```
if (조건) {  
    ...긴코드  
    ...긴코드  
} else {  
    return;  
}
```



```
if (!조건) {  
    return;  
}  
...긴코드  
...긴코드
```

- if-else에서 if가 길고 else가 짧은 경우 역조건을 사용해서 구조 단순화
- 중첩된 if 제거로 코드 복잡도 감소
- 보호절(guard clause), 빠른 리턴(early return)

리팩토링

메서드 분리

같은 조건의 if-else가 몇 군데 출현

일부 비슷하게 동작하는 두 기능을 한 메서드에서 구현했는지 확인

두 기능을 한 메서드에서 구현했다면 점진적으로 메서드 분리

개략적인 순서

- 두 기능 중 한 기능을 위한 메서드 추가
- 이 메서드는 내부에서 기존 메서드를 호출
- 기존 메서드를 호출하는 코드가 새 메서드를 호출하도록 변경
- 기존 메서드의 코드를 새 메서드로 이동
- 기존 메서드 이름을 변경
- 분리 완료 후 추가 개선 진행

메서드 이름 지을 때
참고하면 좋은 기준

- public 메서드: 무엇을 하는지 표현하는 이름(추상적)
- private 메서드: 하위 작업을 구체적으로 표현하는 이름

리팩토링

메서드 분리 예 : 확정과 완료 두 기능이 혼재

```
public void saveCntrCancel(CntrCancelInfoDto dto) {
    String status = contractCancelRepository.findById(dto.getCntrId()).get().getCnlStatus();
    if ("2".equals(status) && "Y".equals(dto.getConfirmYn())) {
        throw new AppException("이미 확정");
    } else if ("3".equals(status) && !"Y".equals(dto.getConfirmYn())) {
        throw new AppException("이미 완료");
    }
    if ("Y".equals(dto.getConfirmYn()) && dto.getConfirmUserId() == null) {
        dto.setProcType("confirm");
        dto.setConfirmUserId(dto.getUserId());
        updateCancelStatus(dto);
    } else if (!"Y".equals(dto.getConfirmYn()) && dto.getApproverId() == null) {
        dto.setProcType("approve");
        dto.setApproverId(dto.getUserId());
        updateCancelStatus(dto);
    }
}
```

리팩토링

메서드 분리 예 : 새 메서드 추가

```

public void confirmCancel(CntrCancelInfoDto dto) {
    this.saveCntrCancel(dto);
}

public void saveCntrCancel(CntrCancelInfoDto dto) {
    String status = contractCancelRepository.findById(dto.getCntrId()).get().getCnlStatus();
    if ("2".equals(status) && "Y".equals(dto.getConfirmYn())) {
        throw new AppException("이미 확정");
    } else if ("3".equals(status) && !"Y".equals(dto.getConfirmYn())) {
        throw new AppException("이미 완료");
    }
    if ("Y".equals(dto.getConfirmYn()) && dto.getConfirmUserId() == null) {
        dto.setProcType("confirm");
        dto.setConfirmUserId(dto.getUserId());
        updateCancelStatus(dto);
    } else if (!"Y".equals(dto.getConfirmYn()) && dto.getApproverId() == null) {
        dto.setProcType("approve");
        dto.setApproverId(dto.getUserId());
        updateCancelStatus(dto);
    }
}
}

```


리팩토링

메서드 분리 예 : 새 메서드를 호출하게 코드 변경

```
contractCancelService.saveCntrCancel(dto);
```



```
if ("Y".equals(dto.getConfirmYn())) {  
    contractCancelService.confirmCancel(dto);  
} else {  
    contractCancelService.saveCntrCancel(dto);  
}
```

리팩토링

메서드 분리 예 : 기존 메서드의 코드를 새 메서드로 이동

```
public void confirmCancel(CntrCancelInfoDto dto) {  
    String status = contractCancelRepository.findById(dto.getCntrId()).get().getCnlStatus();  
    if ("2".equals(status) && "Y".equals(dto.getConfirmYn())) {  
        throw new AppException("이미 확정");  
    }  
    if ("Y".equals(dto.getConfirmYn()) && dto.getConfirmUserId() == null) {  
        dto.setProcType("confirm");  
        dto.setConfirmUserId(dto.getUserId());  
        updateCancelStatus(dto);  
    }  
}  
  
public void saveCntrCancel(CntrCancelInfoDto dto) {  
    String status = contractCancelRepository.findById(dto.getCntrId()).get().getCnlStatus();  
    if ("3".equals(status) && !"Y".equals(dto.getConfirmYn())) {  
        throw new AppException("이미 완료");  
    }  
    if (!"Y".equals(dto.getConfirmYn()) && dto.getApproverId() == null) {  
        dto.setProcType("approve");  
        dto.setApproverId(dto.getUserId());  
        updateCancelStatus(dto);  
    }  
}
```

리팩토링

메서드 분리 예 : 기존 메서드 이름 변경

```
public void saveCtrCancel(CtrCancelInfoDto dto) {  
    ...  
}
```

```
if ("Y".equals(dto.getConfirmYn())) {  
    contractCancelService.confirmCancel(dto);  
} else {  
    contractCancelService.saveCtrCancel(dto);  
}
```



```
public void completeCancel(CtrCancelInfoDto dto) {  
    ...  
}
```

```
if ("Y".equals(dto.getConfirmYn())) {  
    contractCancelService.confirmCancel(dto);  
} else {  
    contractCancelService.completeCancel(dto);  
}
```

리팩토링

메서드 분리 예 : 코드 정리

```
public void confirmCancel(CntrCancelInfoDto dto) {  
    String status = contractCancelRepository.findById(dto.getCntrId()).get().getCnlStatus();  
    if ("2".equals(status) && "Y".equals(dto.getConfirmYn())) {  
        throw new AppException("이미 확정");  
    }  
    if ("Y".equals(dto.getConfirmYn()) && dto.getConfirmUserId() == null) {  
        dto.setProcType("confirm");  
        dto.setConfirmUserId(dto.getUserId());  
        updateCancelStatus(dto);  
    }  
}
```



```
public void confirmCancel(CntrCancelInfoDto dto) {  
    String status = contractCancelRepository.findById(dto.getCntrId()).get().getCnlStatus();  
    if ("2".equals(status)) {  
        throw new AppException("이미 확정");  
    }  
    if (dto.getConfirmUserId() == null) {  
        dto.setProcType("confirm");  
        dto.setConfirmUserId(dto.getUserId());  
        updateCancelStatus(dto);  
    }  
}
```

메서드 실행 전에
dto.getConfirmYn() 확인해서
confirmCancel() 호출

```
if ("Y".equals(dto.getConfirmYn())) {  
    contractCancelService.confirmCancel(dto);  
} else {  
    contractCancelService.completeCancel(dto);  
}
```

리팩토링

클래스 분리

```
public class MemberService {  
    private MemberDao dao;  
  
    public void create(MemberDto dto) {  
        ...  
    }  
  
    ... 메서드 많음  
}
```



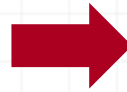
```
public class CreateMemberService {  
    private MemberDao dao;  
  
    public void create(MemberDto dto) {  
        ...  
    }  
}  
  
public class MemberService {  
    private MemberDao dao;  
  
    ... 메서드 많음  
}
```

- 클래스가 커지면
 - 어느 순간부터 큰 클래스의 복잡도 > 클래스를 분리해서 증가하는 복잡도
 - 코드 분석 어려움 증가
- 그래서 일부 기능을 별도 클래스로 분리
- 분리 과정에서 발생하는 중복은 분리 후에 정리

리팩토링

메서드로 추출

```
public Long createAccount(String email) {  
    if (StringUtils.isEmptyOrWhitespace(email)) {  
        throw new AppException("잘못된 이메일");  
    }  
    ResponseEntity<AuthApiResponse> authResp =  
        restTemplate.getForEntity(...생략);  
    AuthApiResponse resp = authResp.getBody();  
    if (resp.getCode() == AuthApiCode.DUP) {  
        throw new DupException();  
    }  
    ServiceAccount acc = ServiceAccount.builder()  
        .accStatus("R").email(email).build();  
    serviceAccountRepository.save(acc);  
    return acc.getAcclId();  
}
```



```
public Long createAccount(String email) {  
    validate(email);  
    checkEmailDuplicate(email);  
    ServiceAccount acc = ServiceAccount.builder()  
        .accStatus("R").email(email).build();  
    serviceAccountRepository.save(acc);  
    return acc.getAcclId();  
}  
  
private void validate(String email) {  
    ...  
}  
  
private void checkEmailDuplication(email) {  
    ResponseEntity<AuthApiResponse> authResp =  
        restTemplate.getForEntity(...생략);  
    ...  
}
```

- 코드 일부를 메서드로 빼냄
- 의도가 드러나는 메서드 이름 사용
- 코드 가독성을 높여줌

리팩토링

클래스로 추출

```
public void provideServicePeriod(Long ordNo, LocalDate loginDate) {  
    Order order = getOrder(ordNo);  
    Period period = null;  
    if (order.getGubun().equals("A")) {  
        period = Period.of(order.getOrderDate(), YearMonth.from(...생략);  
    } else {  
        if (order.getUnit().equals("D")) {  
            period = Period.of(...생략);  
        } else if (order.getUnit().equals("M")) {  
            period = Period.of(...생략;  
        }  
    }  
    updatePeriod(period, period.getSdate(), period.getEdate());  
}
```



```
public class PeriodRule {  
    private Order order;  
  
    public PeriodRule(Order order) {  
        this.order = order;  
    }  
  
    public Period getPeriod(LocalDate loginDate) {  
        Period period = null;  
        ...생략  
        return period;  
    }  
}  
  
public void provideServicePeriod(Long ordNo,  
    LocalDate loginDate) {  
    Order order = getOrder(ordNo);  
    Period period =  
        new PeriodRule(order).getPeriod(loginDate);  
    updatePeriod(period, period.getSdate(),  
        period.getEdate());  
}
```

- 메서드 일부 또는 메서드 전체를 별도 클래스로 추출
- 필드나 변수를 생성자나 메서드로 전달
- 테스트 용이성 증가

리팩토링

파라미터 값 정리

메서드에서 사용하는 값만
파라미터로 받기

- DTO 형태로 넘어오는 파라미터는 사용하지 않는 프로퍼티 제거
- 또는 사용하는 값만 담고 있는 새 타입 사용

사용하지 않는 파라미터 값은 코드 분석을 어렵게 함

새 타입 이용해서
파라미터로 정리하는 순서

1. 메서드 상단에 새 타입을 이용한 객체 생성
2. 메서드가 새 타입 객체를 사용할 때까지 다음 반복
 - 메서드에서 사용하는 파라미터 프로퍼티를 새 타입 객체에 추가
 - 메서드에서 새 타입 객체의 프로퍼티를 사용하게 변경
3. 새 타입 객체를 생성하는 부분을 뺀 나머지를 별도 public 메서드로 추출
4. 메서드 호출을 인라인(inline) 처리
5. 과정 3에서 추출한 메서드 이름을 원래 메서드 이름으로 변경

정리



정리 맺음말

레거시 리팩토링과 테스트 만들기는 절로 되지 않음

많이 연습한 뒤에
돈 받는 일에서
시도해야 함

연습없이 덤비면 “하지 말랬잖아!”, “안 된다고 했잖아” 등의 반응

의도적으로 수련할 것

- 혼자서 연습
 - 경험을 가진 동료로부터 배우기 (짜코딩 요청 등)
 - 회사에 없다면 외부 멘토에게 도움 요청 (노트북 들고 찾아가기)
-

레거시를 대할 때는
마음 다짐이 필요

“개선할 거리가 있다! 해 보자!”