ATDD - PendingOrder

ATDD Cycle

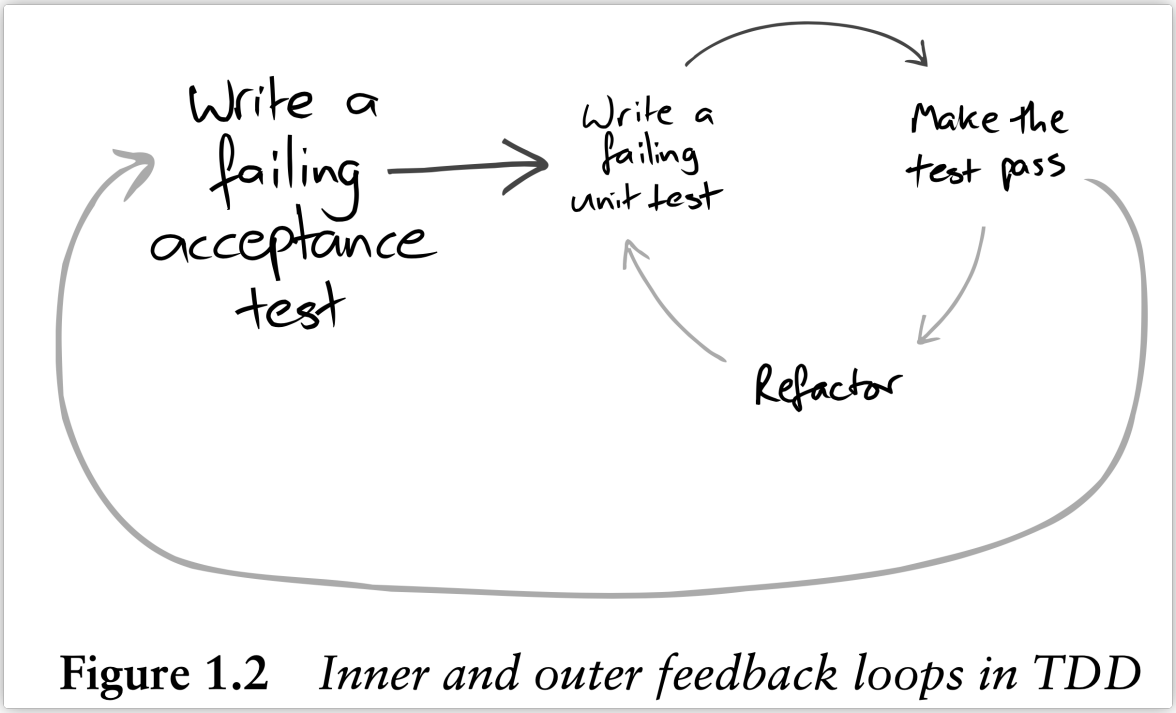

Figure 1.2 *Inner and outer feedback loops in TDD*

0. Create Project

spring initializer로

- gradle project, java
- java11
- dependencies
    - lombok
    - Spring Web

1. Create Failing Integration Test

```
@SpringBootTest
@AutoConfigureMockMvc
```

```
class OrderApiIntegrationTest {
    @Autowired
    private MockMvc mockMvc;
    @Autowired
    private ObjectMapper objectMapper;

    @Test
    void createPendingOrder() throws Exception {
        // 3. Arrange
        long productId = 1L;
        int quantity = 1;
        PendingOrderRequest request = PendingOrderRequest.create(productId, quantity);
        PendingOrderResponse expectedResponse = PendingOrderResponse.create(productId, quantity);

        // 2. Act
        MockHttpServletResponse response = mockMvc.perform(MockMvcRequestBuilders.post("/orders/pendingOrder")
                .contentType(MediaType.APPLICATION_JSON)
                .content(objectMapper.writeValueAsString(request)))
                .andReturn().getResponse();

        // 1. Assert
        assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
        PendingOrderResponse pendingOrderResponse = objectMapper.readValue(response.getContentAsString(), PendingOrderResponse.class);
        assertThat(pendingOrderResponse.getId()).isGreaterThan(PendingOrderResponse.NOT_GENERATED_YET);
    }
}
```

2. Add OrderApi

```
@RestController
public class OrderApi {
    @PostMapping("/orders/pendingOrder")
    public ResponseEntity<PendingOrderResponse> createPendingOrder(@RequestBody PendingOrderRequest request) {
        PendingOrderResponse response = new PendingOrderResponse(request.getProductId(), request.getQuantity());
        return ResponseEntity.ok(response);
    }
}
```

이제 Acceptance Test는 올바른 이유로 실패

- OrderApi에서 CreateOrderService가 필요해짐
- Acceptance Test가 올바른 이유로 실패하면 Acceptance Test를 park할 차례이다.
- 이게 TDD의 Double Loop
  - 처음에는 Acceptance Test(Outer Loop)로 시작
  - Acceptance Test가 올바른 이유로 실패하면 TDD의 Inner Loop(Unit Test)로 들어감
  - Unit Test가 모두 성공하면 이 Acceptance Test도 성공할 것을 기대하면서...

3. CreateOrderServiceTest

```
class CreateOrderServiceTest {
    private PendingOrderRepository pendingOrderRepository;
    private CreateOrderService createOrderService = new CreateOrderServiceImpl(pendingOrderRepository);

    @Test
    void createPendingOrder() {
        long productId = 1L;
        int quantity = 2;
        PendingOrderRequest request = new PendingOrderRequest(productId, quantity);
        PendingOrder pendingOrder = createOrderService.createPendingOrder(request);
        assertThat(pendingOrder.getId()).isPositive();
    }

    private static class CreateOrderServiceImpl implements CreateOrderService {
        private final PendingOrderRepository pendingOrderRepository;

        private CreateOrderServiceImpl(PendingOrderRepository pendingOrderRepository) {
            this.pendingOrderRepository = pendingOrderRepository;
        }

        @Override
        public PendingOrder createPendingOrder(PendingOrderRequest request) {
```

```
            PendingOrder pendingOrder = PendingOrder.create(request.getProductId(), request.getQuantity());
            return pendingOrderRepository.save(pendingOrder);
        }
    }
}
```

4. Make it pass

```
class CreateOrderServiceTest {
    private PendingOrderRepository pendingOrderRepository = new PendingOrderRepositoryMemoryImpl();
    private CreateOrderService createOrderService = new CreateOrderServiceImpl(pendingOrderRepository);

    @Test
    void createPendingOrder() {
        long productId = 1L;
        int quantity = 2;
        PendingOrderRequest request = new PendingOrderRequest(productId, quantity);
        PendingOrder pendingOrder = createOrderService.createPendingOrder(request);
        assertThat(pendingOrder.getId()).isPositive();
    }

    private static class CreateOrderServiceImpl implements CreateOrderService {
        private final PendingOrderRepository pendingOrderRepository;

        private CreateOrderServiceImpl(PendingOrderRepository pendingOrderRepository) {
            this.pendingOrderRepository = pendingOrderRepository;
        }

        @Override
        public PendingOrder createPendingOrder(PendingOrderRequest request) {
            PendingOrder pendingOrder = PendingOrder.create(request.getProductId(), request.getQuantity());
            return pendingOrderRepository.save(pendingOrder);
        }
    }

    private class PendingOrderRepositoryMemoryImpl implements PendingOrderRepository {

        private final AtomicLong pendingOrderId = new AtomicLong(1);

        @Override
        public PendingOrder save(PendingOrder pendingOrder) {
            pendingOrder.assignId(nextId());
            return pendingOrder;
        }

        private long nextId() {
            return pendingOrderId.getAndIncrement();
        }
    }
}
```

4.1 move inner class to upper level

- @Service, @Repositry Annotation 추가

5. Acceptance Test가 성공하는지 확인

```java
@RestController
public class OrderApi {
    private final CreateOrderService createOrderService;

    public OrderApi(CreateOrderService createOrderService) {
        this.createOrderService = createOrderService;
    }

    @PostMapping("/orders/pendingOrder")
    public ResponseEntity<PendingOrderResponse> createPendingOrder(@RequestBody PendingOrderRequest request) {
        PendingOrder pendingOrder = createOrderService.createPendingOrder(request);
        PendingOrderResponse response = PendingOrderResponse.create(pendingOrder);
        return ResponseEntity.ok(response);
    }
}
```

참고

메소드 생성시 body에서 throws exception하도록 변경하기