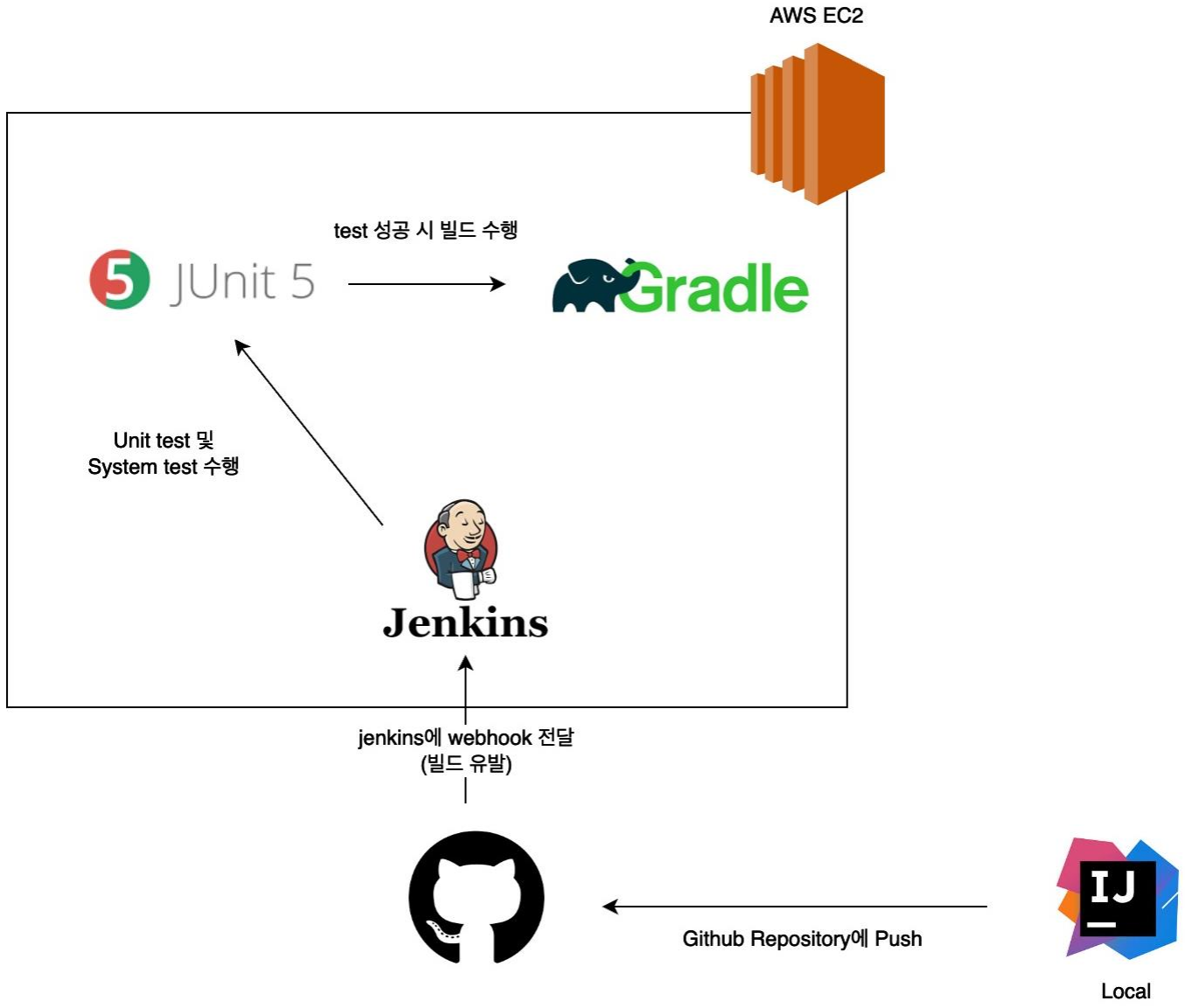
Distrubuted Vending Machine(DVM)

- OOPT Stage 2050/2060

Team 3
202012334 김건우
202012337 김범준
202012338 김상준
202012364 차우정

1. 개발 환경 (CI/CD, UT)



2. UT 및 System Test 시나리오 및 결과

1) Unit-Test

1. 재고관리

```java
import Stock.StockDB;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

beomzun +1 *
public class TestTest {

    new *
    @DisplayName("successMultiplyTest")
    @Test
    void modifyStock(){
        StockDB stockDB = new StockDB( filename: "stock.txt");
        stockDB.modifyStock( itemCode: 1,  itemAmount: 0);
    }
    new *
    @DisplayName("successMultiplyTest")
    @Test
    void restoreStock(){
        StockDB stockDB = new StockDB( filename: "stock.txt");
        boolean b = stockDB.restoreStock( itemCode: 1,  itemAmount: 10);
        assertThat(b).isEqualTo( expected: true);
    }
    new *
    @DisplayName("successMultiplyTest")
    @Test
    void deductStock(){
        StockDB stockDB = new StockDB( filename: "stock.txt");
        boolean b = stockDB.deductStock( itemCode: 1,  itemAmount: 9);
        assertThat(b).isEqualTo( expected: true);
    }

    new *
    @DisplayName("successMultiplyTest")
    @Test
    void get(){
        StockDB stockDB = new StockDB( filename: "stock.txt");
        int b = stockDB.get(1);
        assertThat(b).isEqualTo( expected: 1);
    }
```

```java
package test;

import static org.assertj.core.api.Assertions.assertThat;

import Stock.Reservaion.ReservationDB;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

class Reservation {

    3 usages
    ReservationDB res = new ReservationDB();

    @DisplayName("putReservation")
    @Test
    void putReservation(){
        boolean  i = res.putReservation( code: "1234",  itemCode: 1,  itemAmount: 10);
        assertThat(i).isEqualTo( expected: true);
    }


    @DisplayName("findCode")
    @Test
    void findCode(){
        boolean  i = res.findCode("1234");
        assertThat(i).isEqualTo( expected: true);
    }
    @DisplayName("expireReservation")
    @Test
    void expireReservation(){
        boolean  i = res.expireReservation( code: "1234");
        assertThat(i).isEqualTo( expected: true);
    }

}
```

2. 결제 관리

```java
package test;
import OutActor.BankActor;
import static org.assertj.core.api.Assertions.assertThat;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
public class payTest {
    @Test
    void requestPay(){
        BankActor hu = new BankActor();
        //cardid correct, enough money
        assertThat(hu.requestPay( cardid: "111111",  price: 10000)).isEqualTo( expected: 0);
        //cardid isn't correct.
        assertThat(hu.requestPay( cardid: "1234",  price: 10000)).isEqualTo( expected: 1);
        //cardid is correct, not enough money
        assertThat(hu.requestPay( cardid: "222222",  price: 1000)).isEqualTo( expected: 2);
    }
}
```

3. 새로운 인증코드 생성

```java
package test;

import static org.assertj.core.api.Assertions.assertThat;
import static org.junit.jupiter.api.Assertions.assertNotEquals;


import org.junit.jupiter.api.Test;
public class ControllerTest {

    @Test
    void createCode(){
        Controller con = new Controller();
        String a = con.createCode();
        String b = con.createCode();
        assertNotEquals(a, b);
    }
}
```

4. 네트워크

설정

```
C:\Users\frien\OneDrive - konkuk.ac.kr\FRIENDSHIP\2024-1\OOAD\DVM>java -jar build\libs\DVM-1.0-SNAPSHOT.jar
Hello world!
Case : 1
Your dvmId : Team1
Your position x : 11
Your position y : 11
Stock DB Filename : stock.txt
Enter command (send <id> <message>  ¦ quit ¦ ask <itemCode> ¦
 ¦ renew ¦ prepay <id> <item code> <amount>code <insert code>¦ show <item code>): Invalid command.
Enter command (send <id> <message>  ¦ quit ¦ ask <itemCode> ¦
 ¦ renew ¦ prepay <id> <item code> <amount>code <insert code>¦ show <item code>):
```

```
C:\Users\frien\OneDrive - konkuk.ac.kr\FRIENDSHIP\2024-1\OOAD\DVM>java -jar build\libs\DVM-1.0-SNAPSHOT.jar
Hello world!
Case : 1
Your dvmId : Team2
Your position x : 23
Your position y : 45
Stock DB Filename : stock2.txt
Enter command (send <id> <message>  ¦ quit ¦ ask <itemCode> ¦
 ¦ renew ¦ prepay <id> <item code> <amount>code <insert code>¦ show <item code>): Invalid command.
Enter command (send <id> <message>  ¦ quit ¦ ask <itemCode> ¦
 ¦ renew ¦ prepay <id> <item code> <amount>code <insert code>¦ show <item code>): |
```

Broadcast

송신

```
Enter command (send <id> <message>  ¦ qui
 ¦ renew ¦ prepay <id> <item code> <amou
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
[SocketManager] broadcast success
[SocketManager] broadcast to :Team2
[SocketManager] broadcast to :Team1
```

수신

```
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"8"
,"item_num":7,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"12
","item_num":0,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"13
","item_num":0,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"2"
,"item_num":0,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"5"
,"item_num":0,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"1"
,"item_num":0,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"15
","item_num":0,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"3"
,"item_num":0,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"11
","item_num":0,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"7"
,"item_num":6,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"14
","item_num":0,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"4"
,"item_num":0,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"9"
,"item_num":8,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"16
","item_num":0,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"10
","item_num":9,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"17
","item_num":0,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"6"
,"item_num":5,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"18
","item_num":0,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"19
","item_num":0,"coor_x":23,"coor_y":45}}
receive Task message : {"msg_type":"resp_stock","src_id":"Team2","dst_id":"Team1","messageContent":{"item_code":"20
","item_num":0,"coor_x":23,"coor_y":45}}
```

반영

```
Positions:
  Team2: (23,45)
  Team1: (11,11)
Stocks:
  Stock Data 0:
  Stock Data 1:
    Team2: 0
    Team1: 3
  Stock Data 2:
    Team2: 0
    Team1: 6
  Stock Data 3:
    Team2: 0
    Team1: 3
  Stock Data 4:
    Team2: 0
    Team1: 6
  Stock Data 5:
    Team2: 0
    Team1: 6
  Stock Data 6:
    Team2: 5
    Team1: 0
  Stock Data 7:
    Team2: 6
    Team1: 0
  Stock Data 8:
    Team2: 7
    Team1: 0
  Stock Data 9:
    Team2: 8
    Team1: 0
  Stock Data 10:
    Team2: 9
    Team1: 0
  Stock Data 11:
    Team2: 0
    Team1: 0
  Stock Data 12:
    Team2: 0
    Team1: 0
  Stock Data 13:
    Team2: 0
    Team1: 0
  Stock Data 14:
    Team2: 0
    Team1: 0
  Stock Data 15:
    Team2: 0
    Team1: 0
  Stock Data 16:
    Team2: 0
    Team1: 0
  Stock Data 17:
    Team2: 0
    Team1: 0
  Stock Data 18:
    Team2: 0
    Team1: 0
  Stock Data 19:
```

선결제 요청

실패

```
 ¦ renew ¦ prepay <id> <item code> <amount>code <insert code>¦ show <item code>): prepay Team2 2 1
receive Task message : {"msg_type":"resp_prepay","src_id":"Team2","dst_id":"Team1","messageContent":{"
item_code":"2","item_num":1,"coor_x":0,"coor_y":0,"availability":"F"}}
Prepay Message sent : Team2 : (2 , 1)
Enter command (send <id> <message>  ¦ quit ¦ ask <itemCode> ¦
 ¦ renew ¦ prepay <id> <item code> <amount>code <insert code>¦ show <item code>): |
```

성공

```
Enter command (send <id> <message>  ¦ quit ¦ ask <itemCode> ¦
 ¦ renew ¦ prepay <id> <item code> <amount>code <insert code>¦ show <item code>): prepay Team2 6 3
receive Task message : {"msg_type":"resp_prepay","src_id":"Team2","dst_id":"Team1","messageContent":{"
item_code":"6","item_num":3,"coor_x":0,"coor_y":0,"availability":"T"}}
Prepay Message sent : Team2 : (6 , 3)
Enter command (send <id> <message>  ¦ quit ¦ ask <itemCode> ¦
 ¦ renew ¦ prepay <id> <item code> <amount>code <insert code>¦ show <item code>): |
```

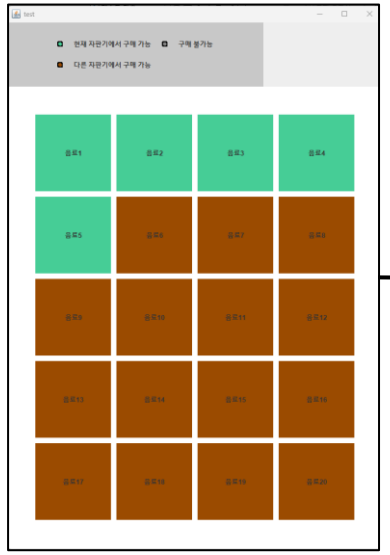
2) System-Test (LAN)

Test Number	Use Case	Test Description	Pass/Fail
1	Pay Item	User가 구매하고자 하는 음료의 결제 금액을 정확하게 계산하여 User에게 출력하였는지, 결제하기 위해 입력한 카드 정보를 DVM이 정상적으로 수신하였는지 확인한다.	Pass
2	Request Payment	User의 카드 정보와 잔액에 따라서 결제 과정이 정상적으로 진행되는지 확인한다.	Pass
3	Refund	Bank에게 환불 요청을 정상적으로 송신하는지 확인한다.	Pass
4	Fail Payment	Bank의 결제 실패로 인한 결제 실패 응답을 정상적으로 수신하는지 확인한다.	Pass
5	View Fail Payment	User에게 결제 실패 및 후속 절차를 정상적으로 안내하는지 확인한다.	Pass
6	View Item	User에게 DVM에서 판매하는 전체 메뉴를 정상적으로 출력하는지 확인한다.	Pass
7	Renew All Stock	현재 DVM에서 결제 가능한 전체 음료의 재고를 정상적으로 파악하였는지 확인한다.	Pass
8	Deduct Stock	현재 DVM에서 제공 가능한 음료의 개수를 결제한 음료의 개수만큼 차감하는지 확인한다.	Pass
9	Modify Stock	DVM의 음료 재고가 올바른 방법으로 접근한 admin에 의해 정상적으로 보충되었는지 확인한다.	Pass
10	Reserve Stock	수신한 선결제 요청 msg에서 요구하는 음료의 개수만큼 현재 DVM에서 제공 가능한 음료의 재고를 차감하는지 확인한다.	Pass
11	Response Stock Info	수신한 msg에서 확인한 음료의 개수와 현재 DVM의 해당 음료의 재고에 따라 정상적으로 응답을 보내는지 확인한다.	Pass
12	Pre-Pay Item	User의 선결제 요청이 정상적으로 요청되었는지 확인한다.	Pass
13	Response to Pre-Pay	Other DVM으로부터 수신한 선결제 응답에 따라 User에게 정상적으로 선결제 응답을 보내는지 확인한다.	Pass
14	Ask Stock Info	All Other DVM에게 위치 정보와 음료의 재고를 정상적으로 요청하는지 확인한다.	Pass

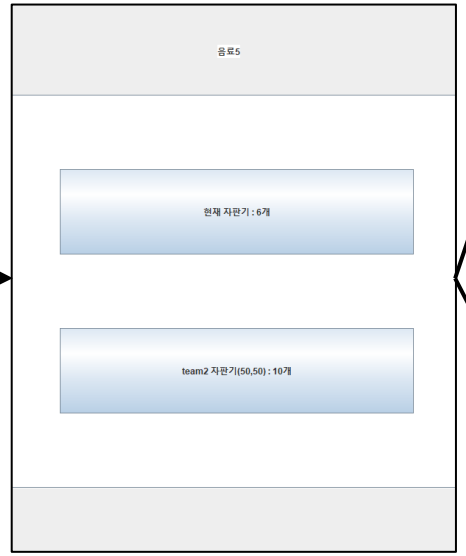
Test Number	Use Case	Test Description	Pass/Fail
15	Fail Pre-Pay	선결제 실패 응답에 따른 후속 절차를 정상적으로 진행하는지 확인한다.	Pass
16	View Fail Pre-Pay	User에게 선결제 실패를 정상적으로 안내하는지 확인한다.	Pass
17	Request Pre-Pay	User가 요청한 음료의 개수를 제공 가능한 가장 가까운 DVM에게 음료 종류와 개수, 인증 코드를 정상적으로 송신하는지 확인한다.	Pass
18	Create Code	인증 코드를 정상적으로 생성하는지 확인한다.	Pass
19	View Code	User에게 인증 코드를 정상적으로 출력하는지 확인한다.	Pass
20	Receive Code	Other DVM으로부터 음료 종류와 개수, 인증 코드를 정상적으로 수신하여 저장했는지 확인한다.	Pass
21	Get Reserve Item	User가 입력한 인증코드가 올바른 경우, 해당 인증 코드와 함께 저장한 음료를 개수만큼 제공하는지 확인한다.	Pass
22	Expire Code	인증코드를 정상적으로 제거하는지 확인한다.	Pass
23	Select Item	User가 선택한 음료를 DVM이 정상적으로 인식했는지 확인한다.	Pass
24	View Closet DVM	Broadcast의 결과로 수신한 정보를 바탕으로 현재 DVM에서 가장 가까운 DVM을 정확하게 계산하여 User에게 안내하였는지 확인한다.	Pass
25	Serve Item	User에게 음료를 정상적으로 제공하였는지 확인한다.	Pass

3. 시스템 동작 화면

초기화면

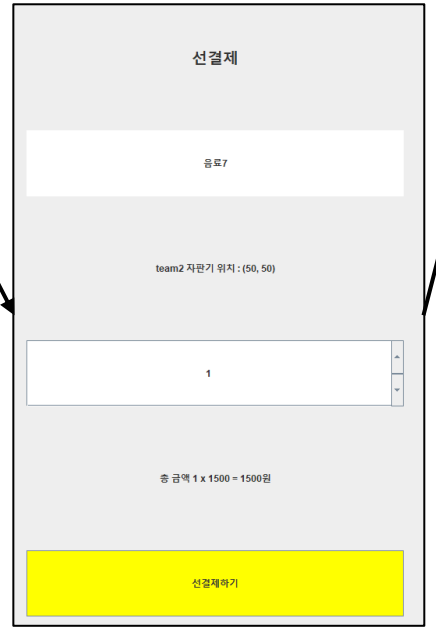


음료 선택 시

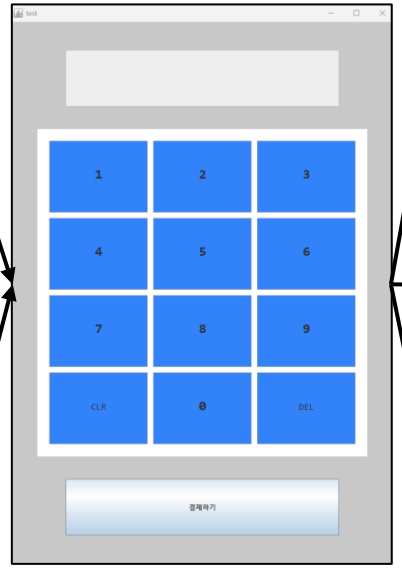


결제

선결제



카드 입력



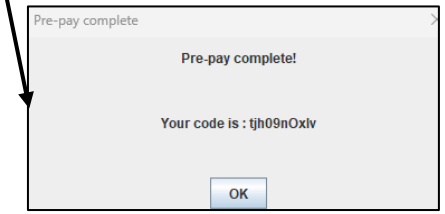
결제 성공

```
오후 4:20:39: Executing ':UI.main()'...

> Task :compileJava UP-TO-DATE
> Task :processResources NO-SOURCE
> Task :classes UP-TO-DATE
> Task :UI.main()
Served 2 amount(s) of item #1!
```

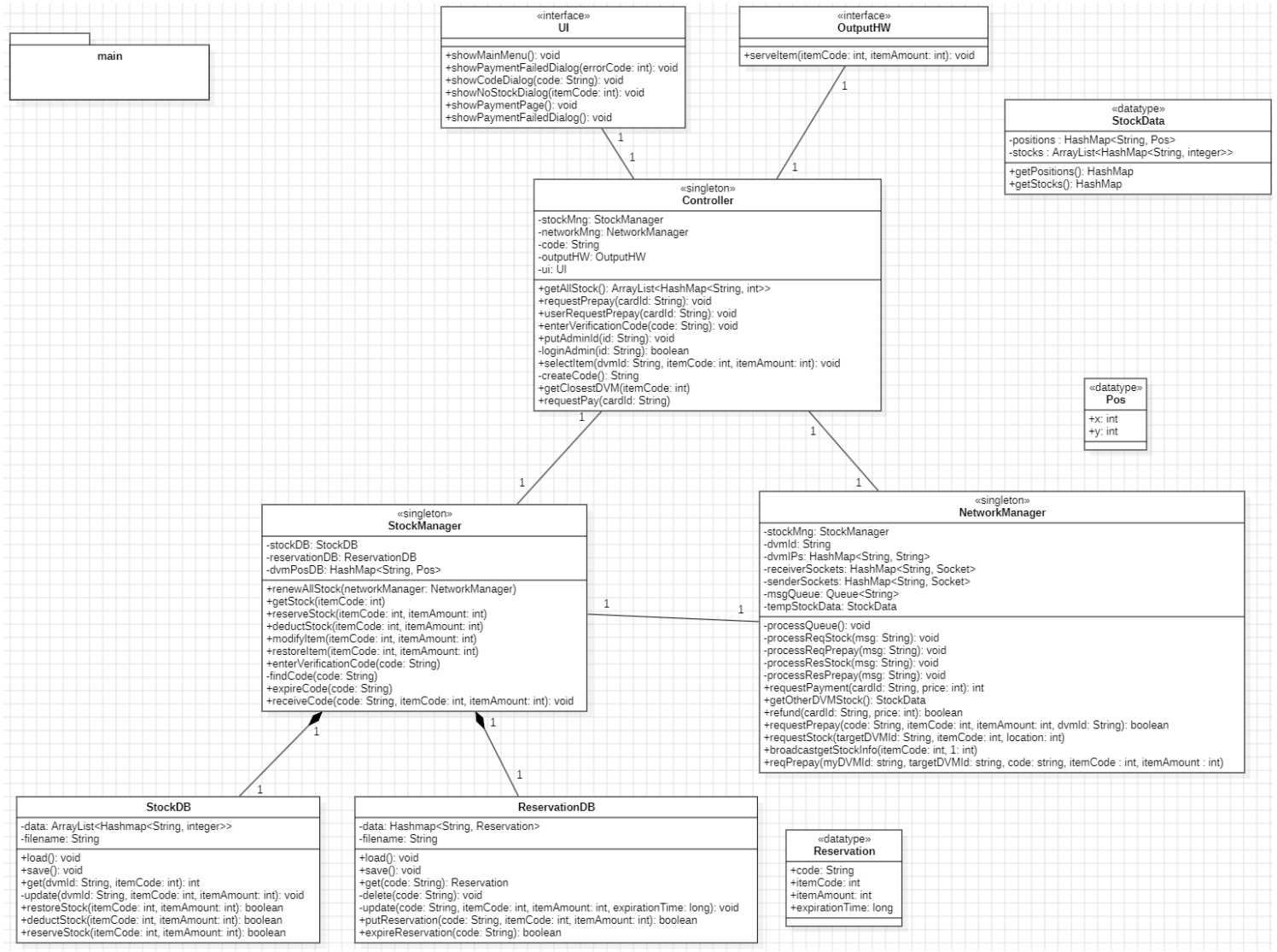
결제 실패

선결제 성공



4. OOD (2040) 에서 변경/수정된 부분

1. DCD (2040)

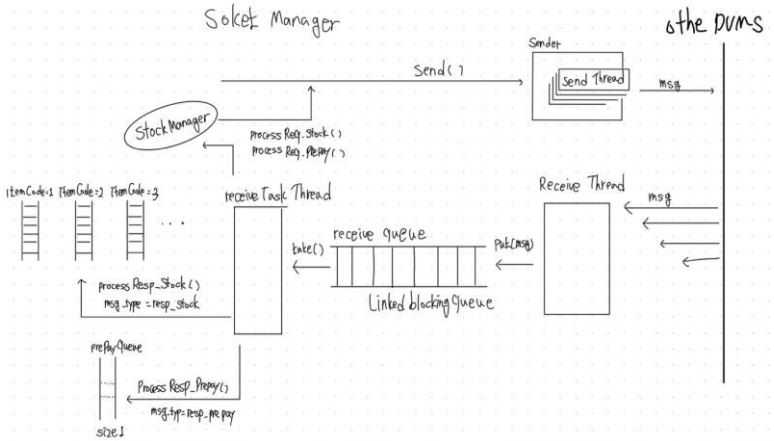


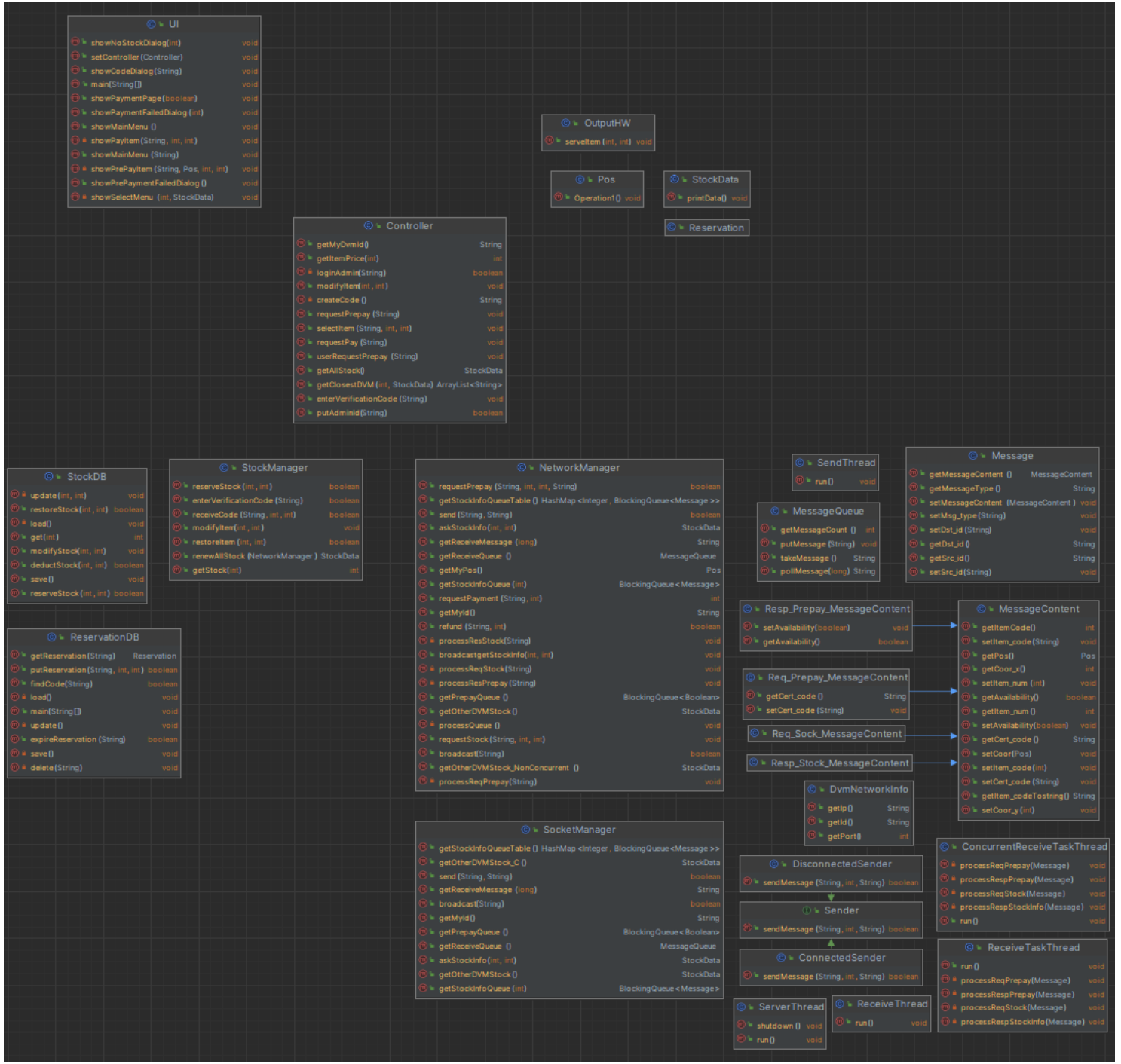
4. OOD (2040) 에서 변경/수정된 부분

1. DCD (2050)

- **Network를 자세히 고려하지 않음**
  - Manager 단일 클래스로 해결시도
  - Msg, Thread 등 이외의 클래스들 필요

네트워크에 대한 이해가 부족, 2040 반영X





4. OOD (2040) 에서 변경/수정된 부분

1. SD

1) 변경되지 않은 부분

#1	Pay item
#2	Request payment
#3	Refund
#4	Restore Stock
#5	Fail Payment
#8	Deduct stock
#9	Modify stock
#10	Reserve stock
#11	Response stock info
#12	Response to Pre-Pay
#13	Ask stock info.
#15	Fail Pre-Pay
#16	Request Pre-Pay
#18	Create code
#20	Receive code
#21	Get reserve item
#22	Expire code
#23	Select item
#24	View Closet DVM
#25	Serve item

2) 변경된 부분

#6	View Item
#7	Renew all stock
#14	Pre-Pay item
#17	View Fail Pre-Pay
#19	View code

변경 사유)

1. 데이터 종속 여부
2. Network 구현
3. UI 구현

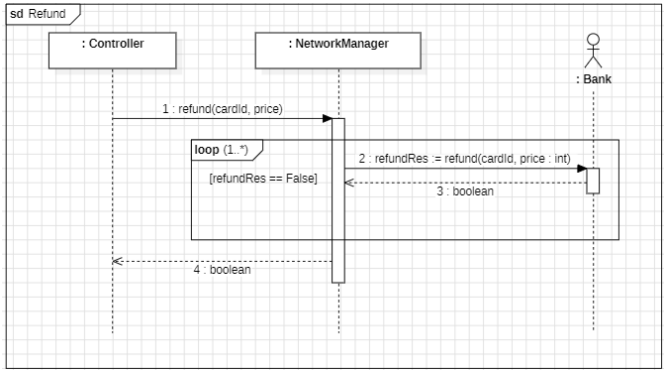
1) 변경되지 않은 부분

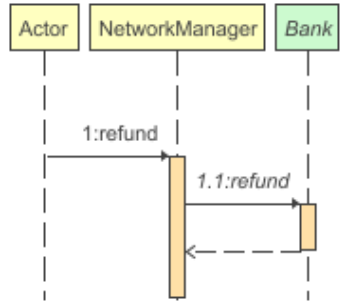
- SD 그대로 개발 진행
- UC 활용의 좋은 예

Activity 2041. Define Real Use Cases #11

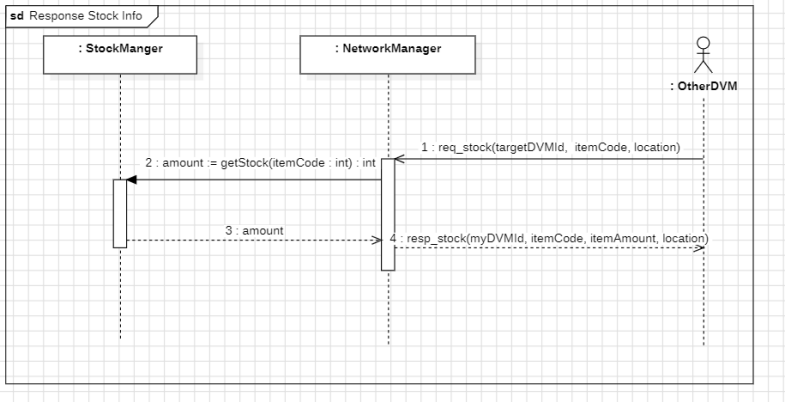
Use Case	Response stock info
Actor	Other DVM
Purpose	재고 확인요청에 대한 응답 msg를 전송한다.
Overview	Other DVM에서 특정 음료의 개수를 msg를 통해 전달 받는다. 해당 음료의 개수가 재고에 충분히 있다면 해당 개수를 msg로 보낸다. 만약 관리자가 재고를 수정하고 있거나, 재고 변경이 금지되는 작업을 수행 중 이라면, 무조건 0을 msg로 보낸다.
Type	primary and Essential
Cross Reference	Function : R5.1 Use Cases : Renew all stock
Pre-Requisites	현재 DVM의 재고를 가져올 수 있어야한다. 비동기적으로 프로토콜 요청을 처리할 수 있어야한다.
Typical Courses of Events	(A) : Actor, (S) : System 1.(A) Other DVM에서 음료에 대한 재고 확인 요청을 보낸다 2.(S) 해당 요청 음료의 재고와 요청 개수를 비교한다.(A1) 3.(S) 재고의 개수를 응답 msg에 담아 전송한다.
Alternative Courses of Events	A1. 재고가 부족하다면, 재고 0을 보낸다.
Exceptional Courses of Events	N/A

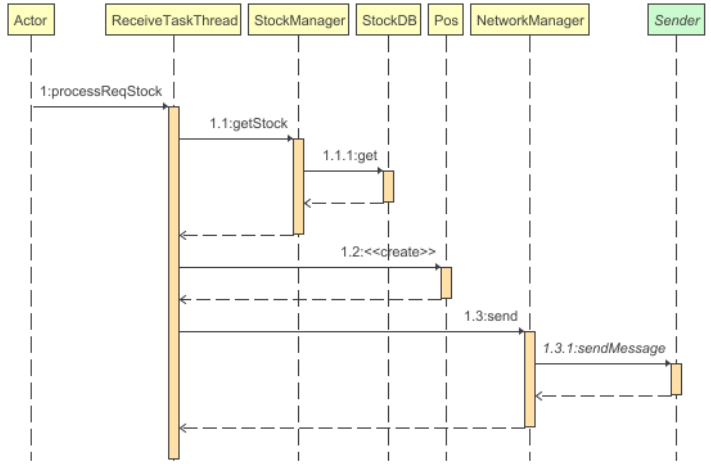
#3 Refund



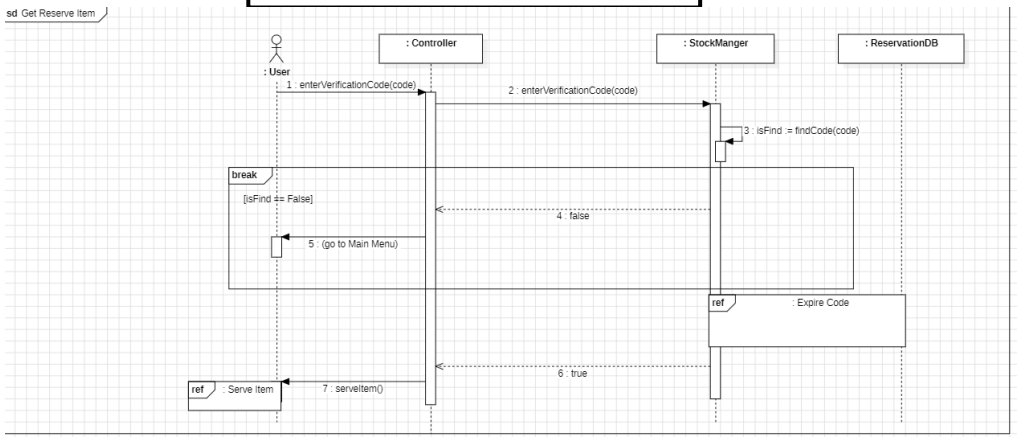


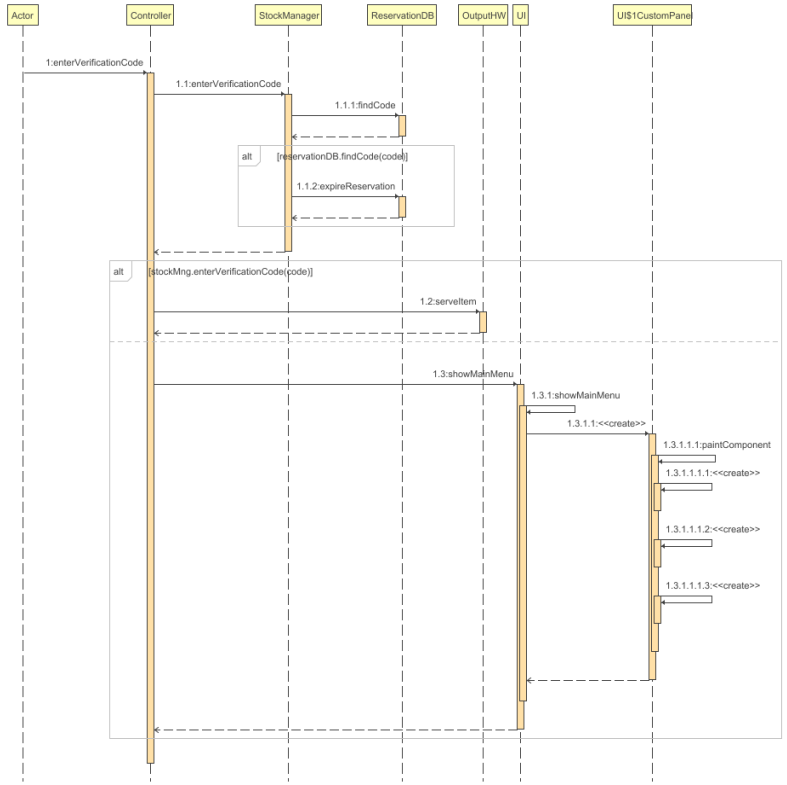
#11 Response Stock Info



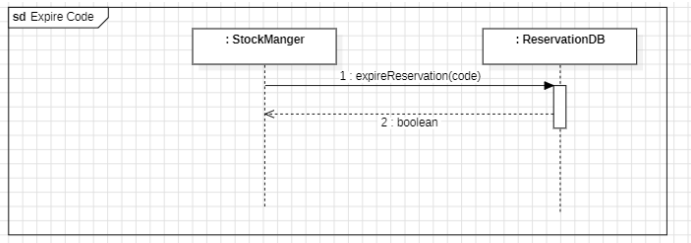


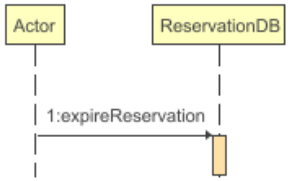
#21 Get Reserve Item



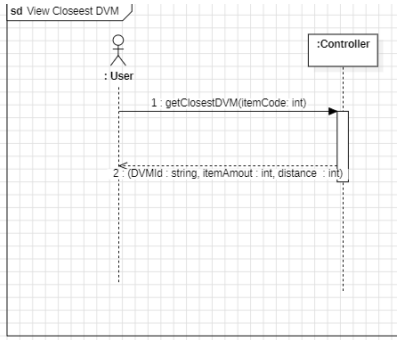


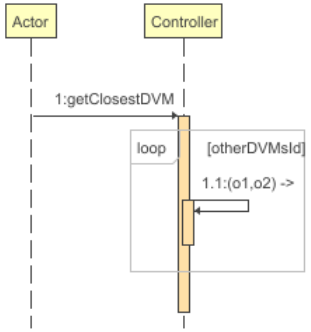
#22 Expire Code





#24 View Closest DVM

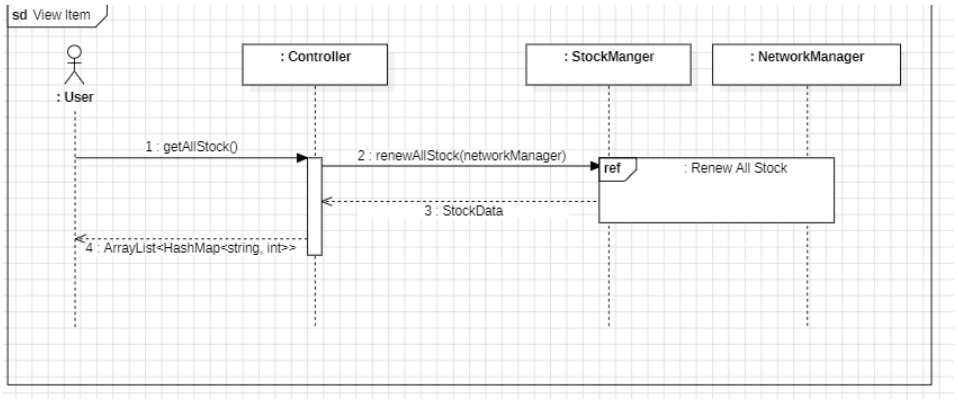


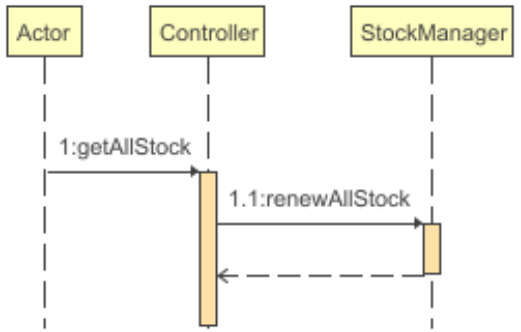


2) 변경 된 부분 – 1 (#6, #7, #14)

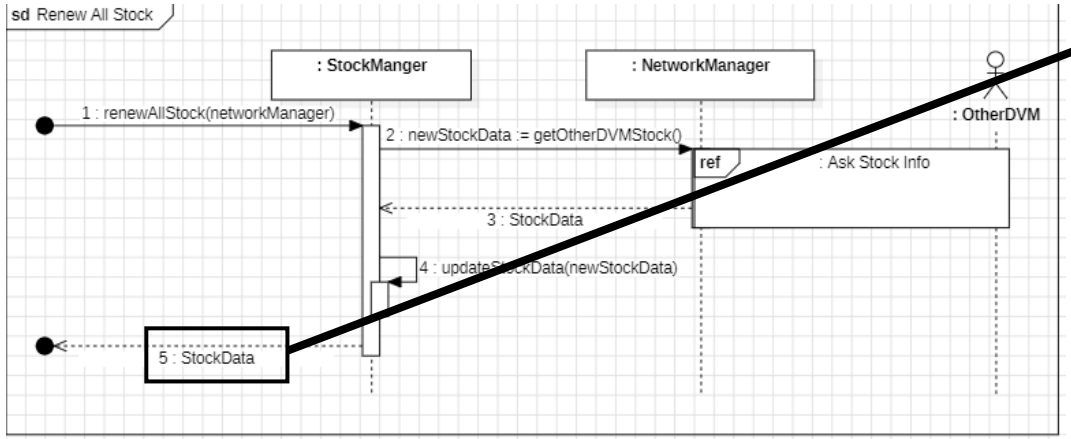
변경 사유
- 데이터 종속 여부

#6 View Item





#7 Renew All Stock



#14 Ask Stock Info에서 처리

- StockData 객체 생성 위해서는 DVMId 필요
- StockManger에서 DVMId 미보유

StockData
- 음료 정보, for Controller

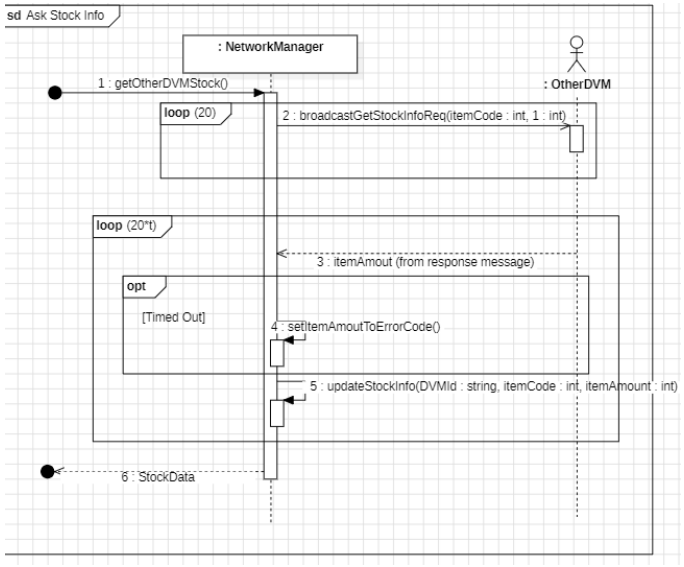
- NetworkManger, DVMId 보유
- StockManger, DVMId 미보유

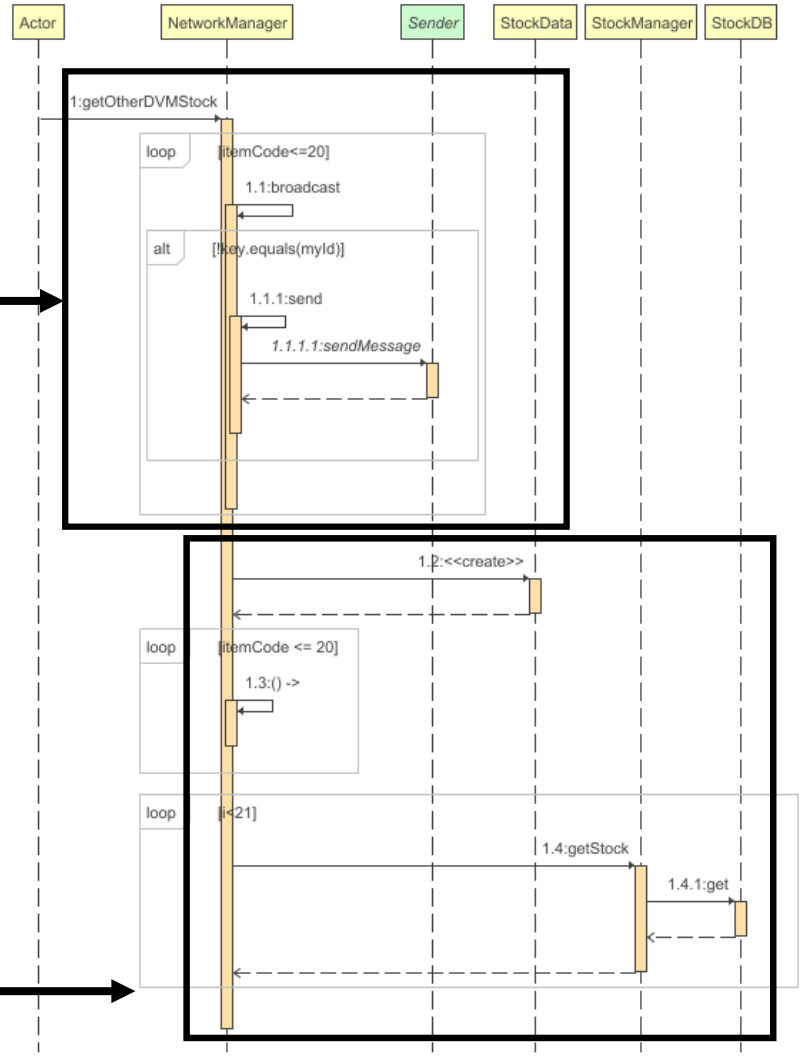
StockManger -> NetworkManger

⬇

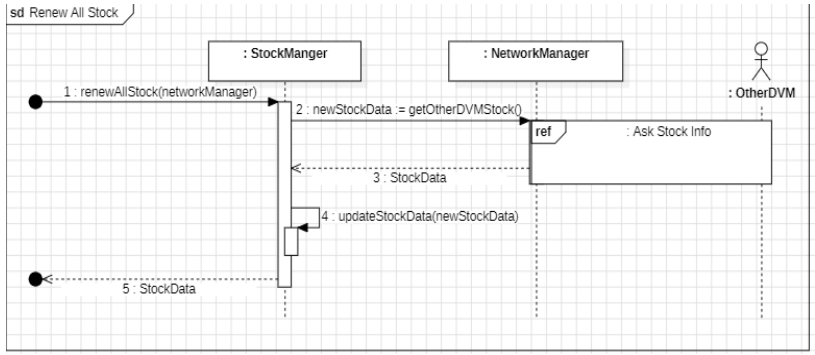
NetworkManger -> StockManger

#14 Ask Stock Info





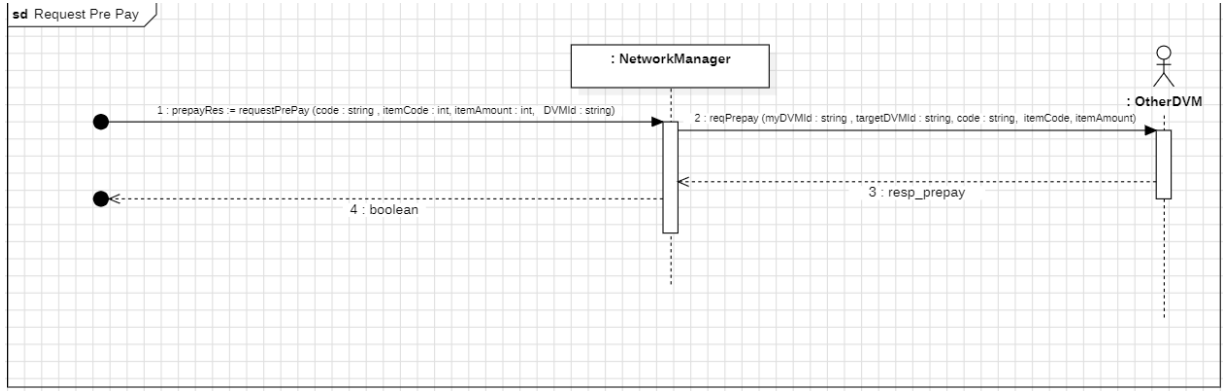
#7 Renew All Stock

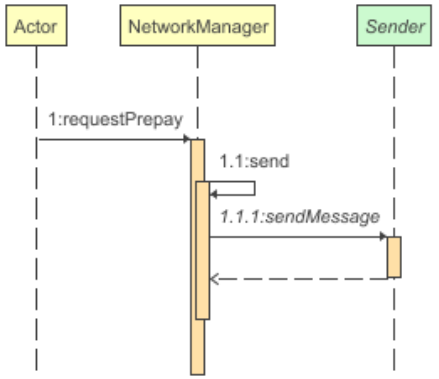


2) 변경 된 부분 -2 (#17)

변경 사유
- Network 구현, NetworkManger 역할 분배

#17 Request Prepay

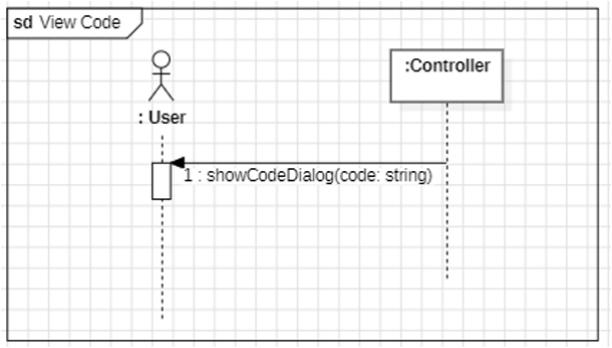


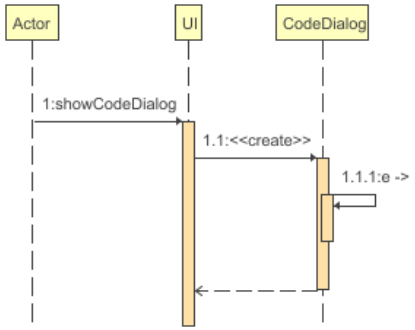


2) 변경 된 부분 -2 (#19)

변경 사유
- UI 구현

#19 View Code

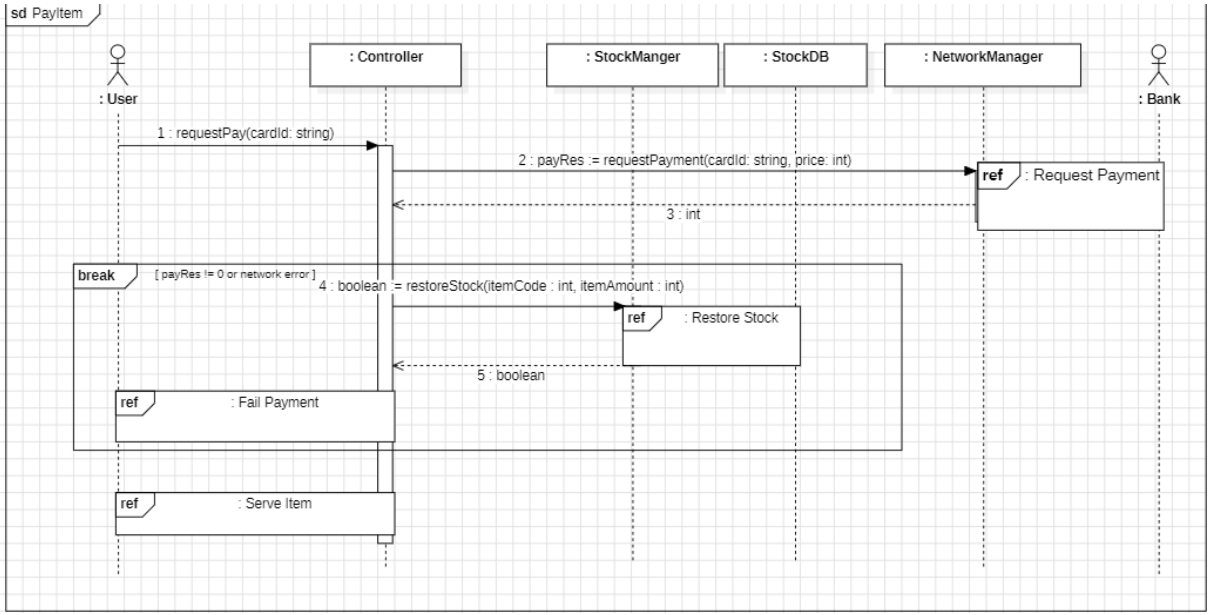




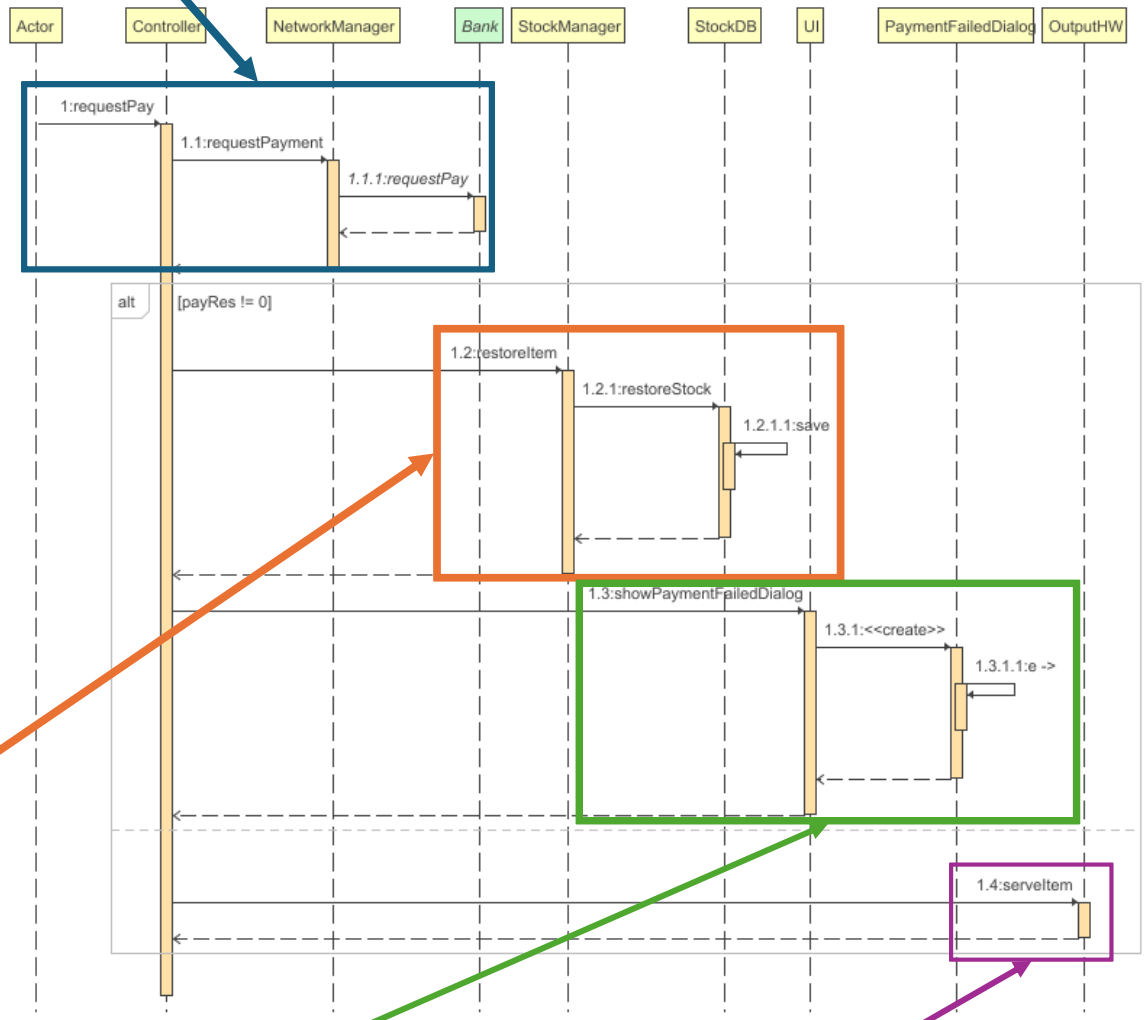
5. 구현 시 예상보다 어려웠던 점

UC의 무분별한 분해

#1 PayItem



Sequence Diagram #2 Request Payment

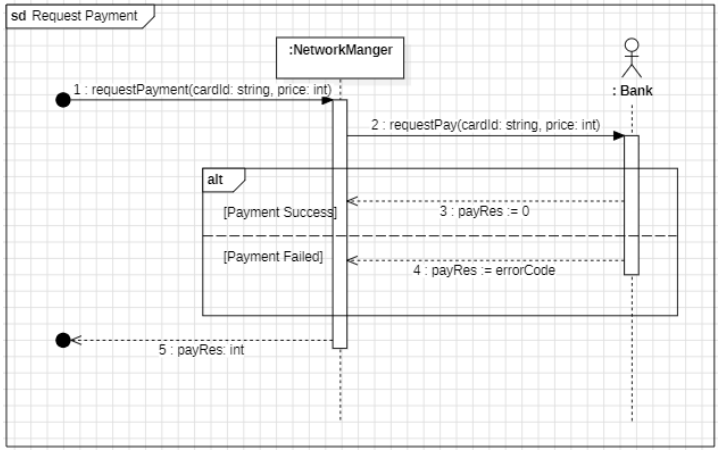


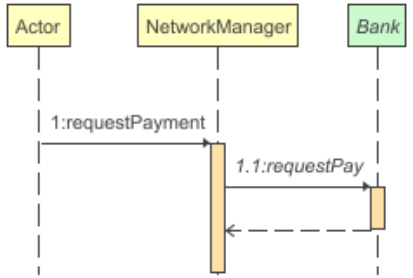
Sequence Diagram #4 Restore Stock

Sequence Diagram #5 Fail Payment

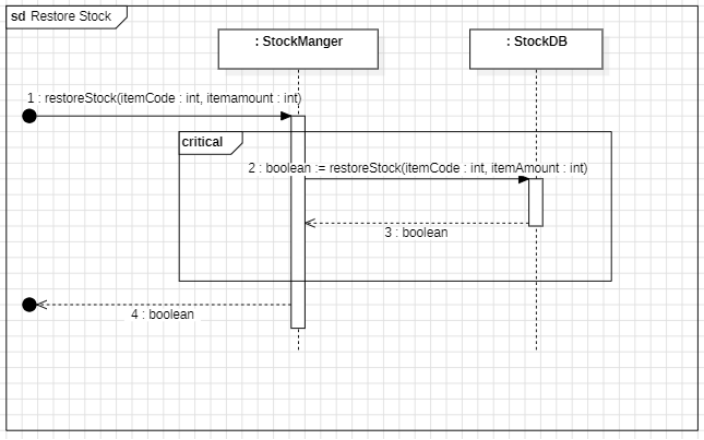
Sequence Diagram #25 Serve Item

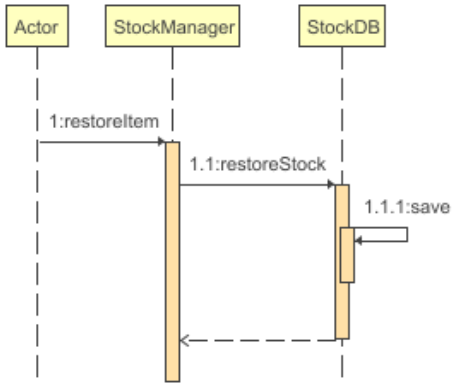
Sequence Diagram #2 Request Payment



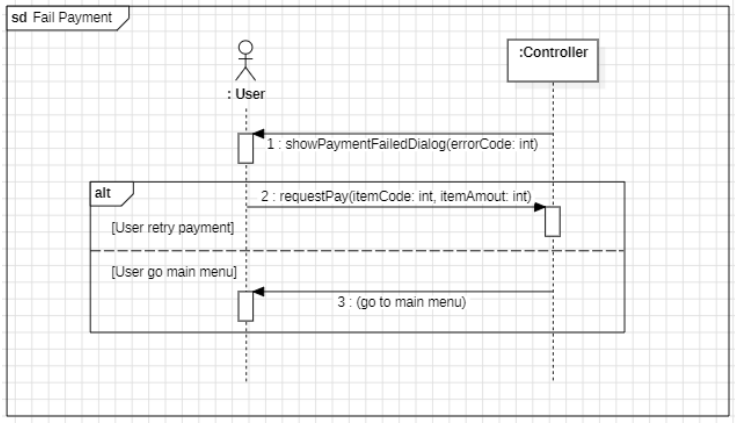


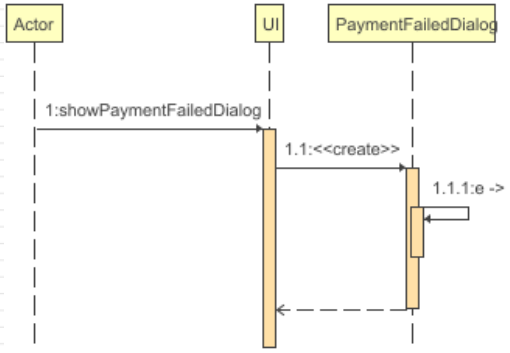
Sequence Diagram #4 Restore Stock



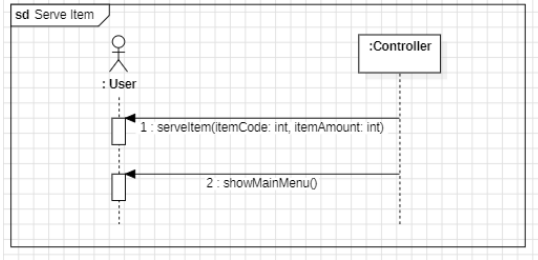


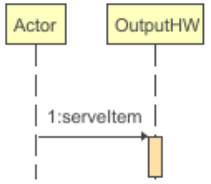
Sequence Diagram #5 Fail Payment



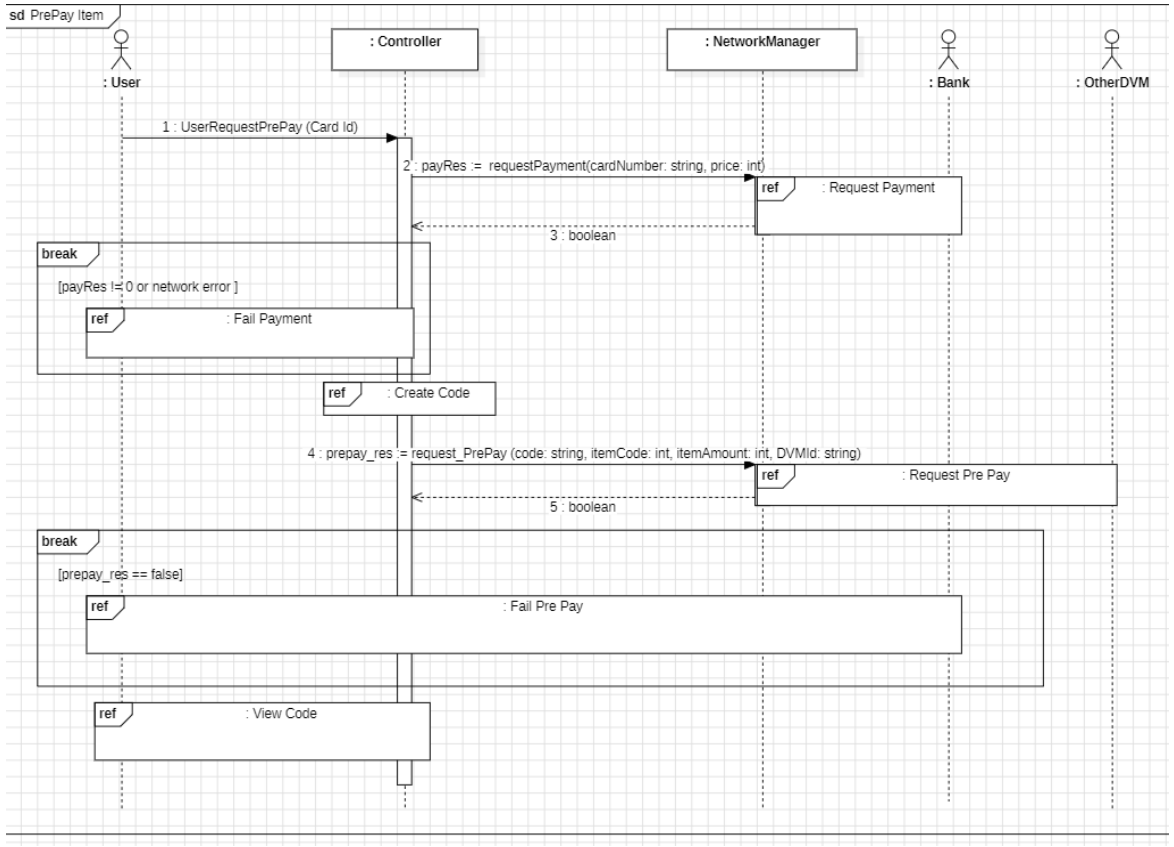


Sequence Diagram #25 Serve Item

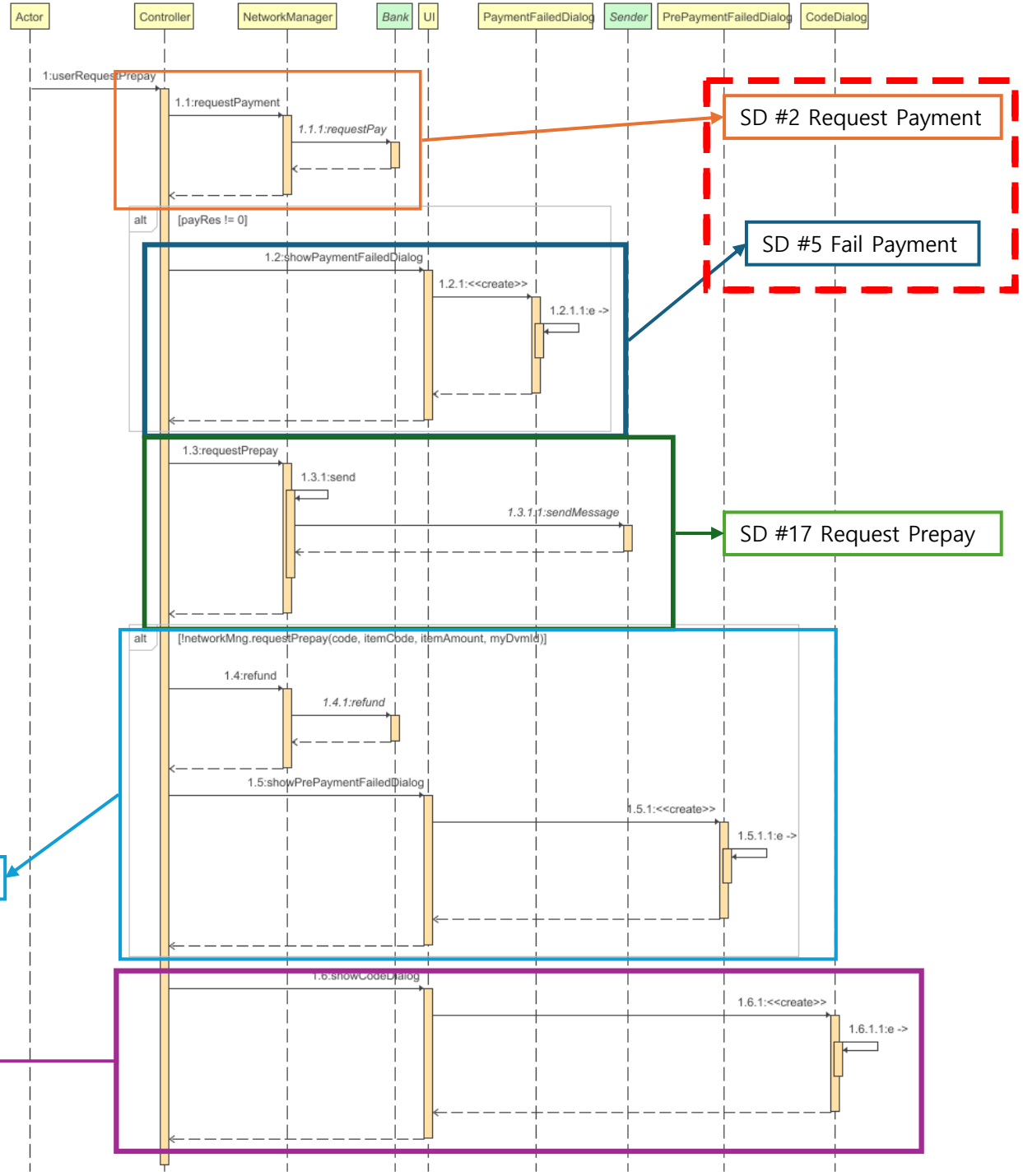




#12 Prepay Item



UC 분해 장점
- **같은 기능**을 하는 **Method** 찾기 쉬움
- **개발 및 UT에 편리**



Sequence Diagram #2 Request Payment

Sequence Diagram #5 Fail Payment

Sequence Diagram #10 Restore Stock

Sequence Diagram #25 Serve Item

Activity 2031. Define Essential Use Cases

결제
1. Pay item
2. Request payment
3. Refund
5. Fail payment

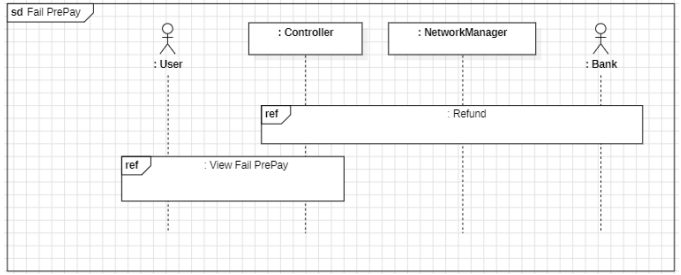
선결제 요청
11. Response stock info.
12. Response to Pre-Pay
13. Ask stock info.
14. Pre-Pay item
15. Fail Pre-Pay
16. Request Pre-Pay
17. View Fail Pre-Pay

선결제 처리
18. Create code
19. View code
20. Receive code
21. Get reserve item
22. Expire code

재고 관리
4. Restore stock
6. View Item
7. Renew all stock
8. Deduct stock
9. Modify stock
10. Reserve stock

메뉴
23. Select item
24. View Closet DVM
25. Serve item

#15 Fail Prepay

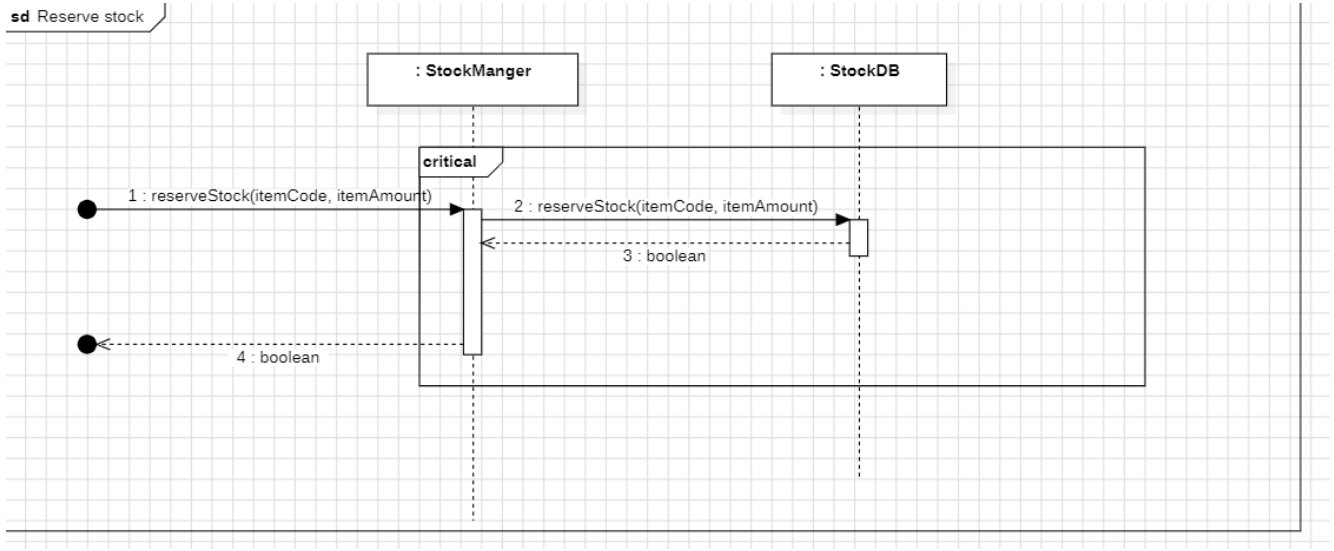


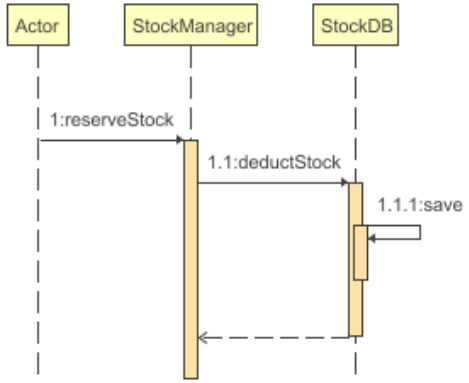
UC 분해 단점

- OOAD : 객체의 communication을 통해 service를 제공.
- FR를 놓치지 않기 위해, UC를 분해.
- 하나의 UC를 위해 여러 UC들이 Ref로 필요
- UC들 간의 traceability가 부족, 전체의 흐름을 파악하는데 어려움 -> 개발에 혼동.

- 아쉬운 부분(Traceability)

#10 Reserve Stock





Activity 2041. Define Real Use Cases #10

Use Case	Reserve stock
Actor	Event-based
Purpose	선결제 요청이 들어온 경우 요청 개수만큼 제외
Overview	비동기식으로 들어오는 선결제 요청 msg를 받은 경우, Other DVM에서 요청한 음료의 개수 만큼 해당 음료의 재고를 차감한다.
Type	Primary and Essential
Cross Reference	System Functions : R3.2 Use Case : Response to Pre-pay
Pre-Requisites	선결제 요청 msg를 받은 상태 선결제 요구 재고 이상의 item이 현재 DVM에 있는 상태.
Typical Courses of Events	1.: Actor, (S) : System 2.(S) 해당 msg에서 요구하는 음료의 개수를 확인 3.(S) 해당 음료의 재고만큼 현재 DVM에서 차감 (E1)
Alternative Courses of Events	N/A
Exceptional Courses of Events	E1 : 해당 음료의 재고가 음수가 되는 경우, 요청을 보낸 DVM에게 제공이 불가능 하다는 것을 알림.

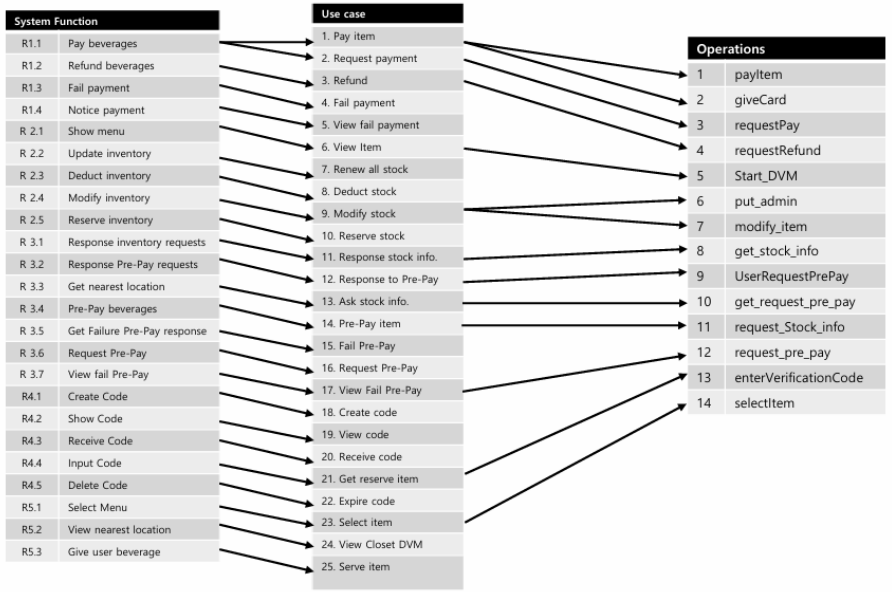
혼동

- Reserve : 미리 재고 차감 | 선결제 재고 차감
- 개발 중 UC 참고, 번호를 이용
- Traceability 미흡.

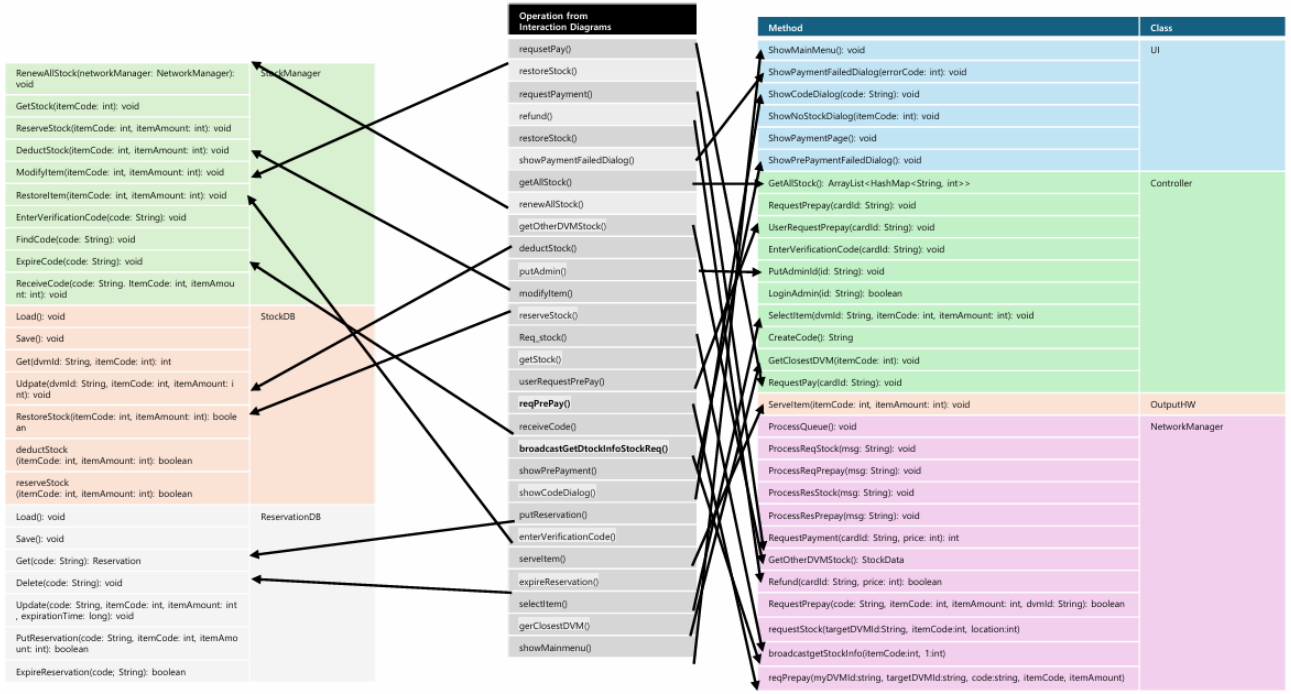
- 아쉬운 부분(Traceability)

2040 – 복잡한 Traceability

Activity 2039. Analyze Traceability Analysis

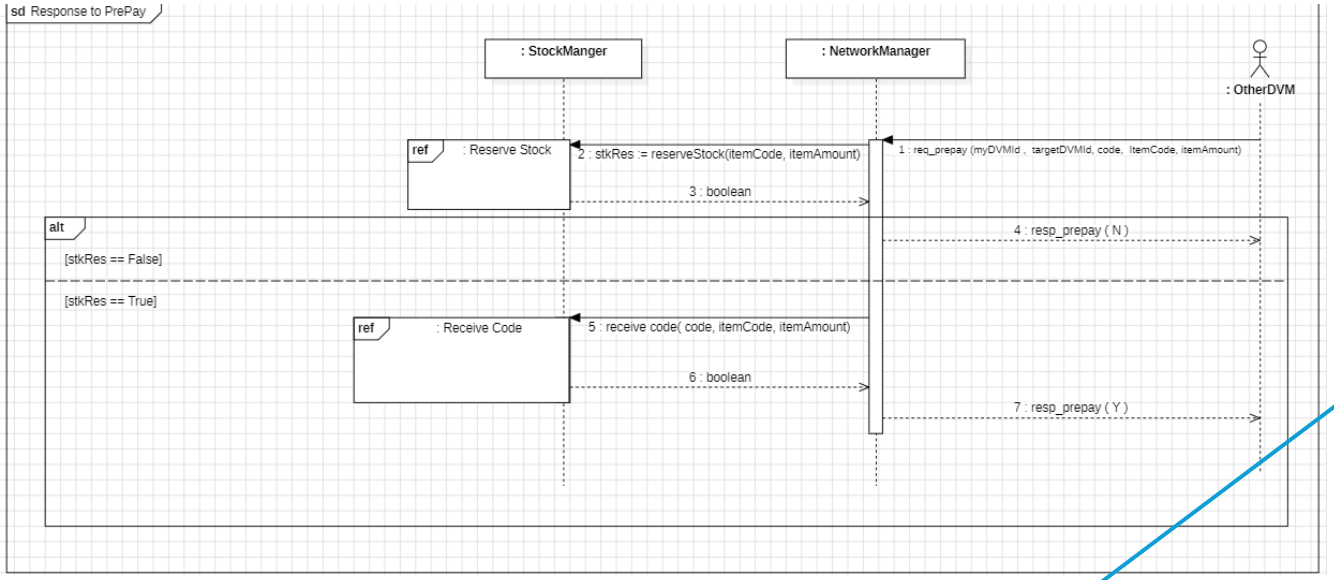


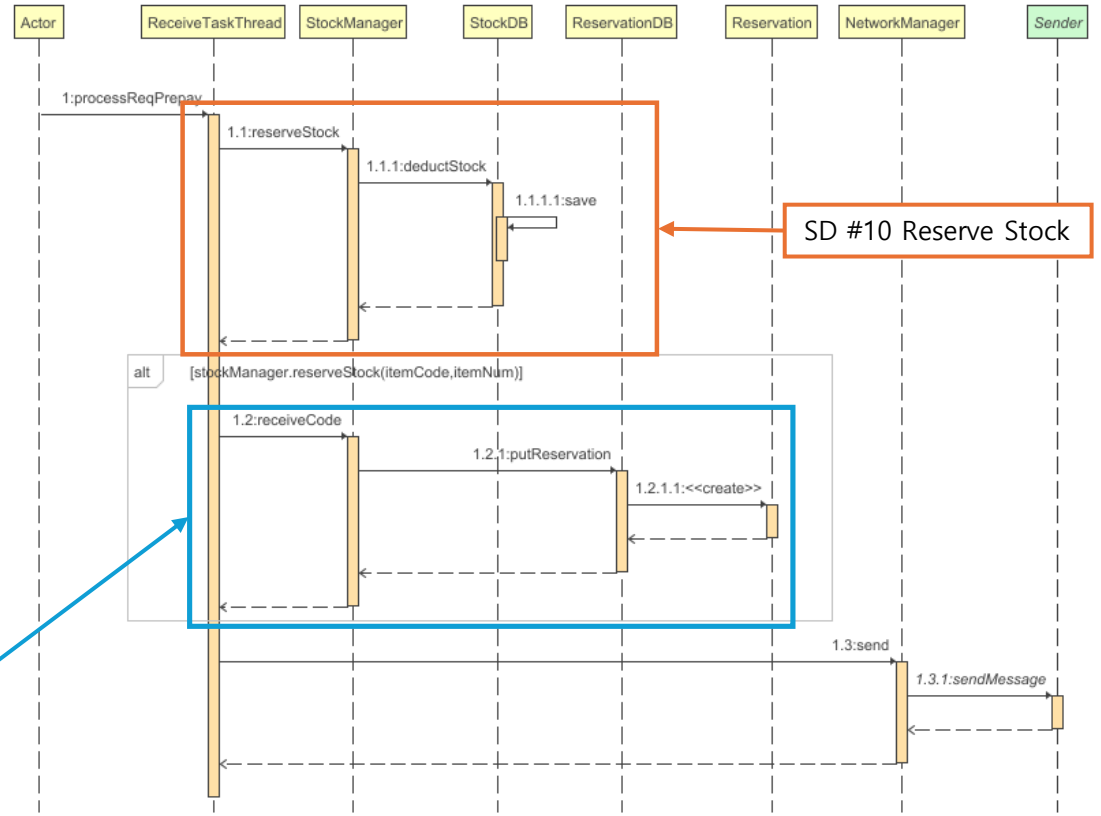
Design Traceability Analysis



Glossary의 필요성

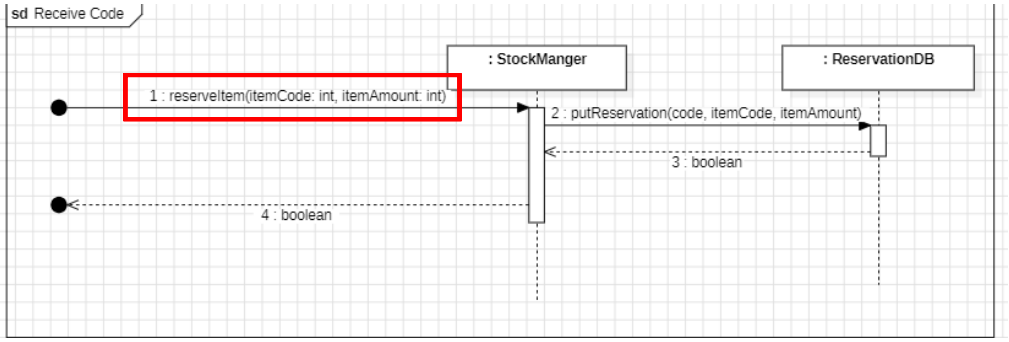
#13 Response to Prepay

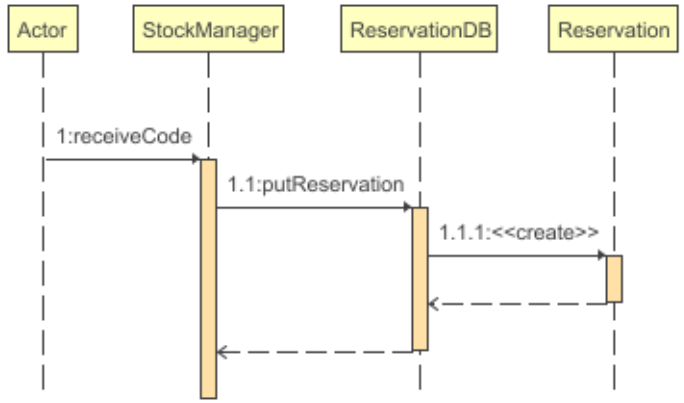




단어 혼동

#20 Receive Code





Glossary의 필요성

Activity 1004. Record terms in Glossary

Term	Description
DVM	Distributed Vending Machine의 약자, 네트워크로 연결된 분산 자판기를 의미.
결제	사용자가 DVM을 통해 음료를 구매하기 위해 카드 정보를 입력하고 DVM은 그 정보를 은행사에게 보내어 결제를 확인하는 전체 과정을 의미.
선결제	사용자가 현재 사용하고 있는 DVM에 없으며, 다른 DVM에 존재하는 음료를 구매하고자 할 때 사용하는 결제 시스템으로, 해당 기능을 통해 가장 가까운 다른 DVM에 음료를 예약할 수 있음.
msg	Message의 줄임말로 DVM이 네트워크 상으로 연결된 다른 DVM에게 전송하거나 전송 받을 때 사용되는 데이터 단위.
인증 코드	선결제 요청 시 다른 DVM에서 보내는 msg 중 하나로, 인증 코드와 같은 의미. 선결제를 통해 다른 DVM에서 음료를 구매한 경우, 발급되며, 해당 코드를 통해 발급된 DVM에서 음료를 받을 수 있음. Code와 동일시하여 사용.
최신화	사용자가 결제를 통해 음료를 구매한 경우와 다른 DVM에서 선결제를 요청한 경우 재고에서 해당 개수를 빼야 함.
Broadcast	네트워크로 연결 된 모든 DVM에게 msg를 전송하는 방식
transaction	두 개 이상의 개체 간에 발생하는 상호작용이나 거래를 나타내는 용어. 선결제 요청과 해당 자판기의 결제, 다른 DVM에서 해당 DVM으로의 선결제가 동시에 일어나는 경우, 발생.
rollback	트랜잭션의 일부 또는 전체가 실패하거나 취소되어야 할 때, 이전 상태로 되돌리는 것을 의미.
Admin	DVM을 관리하는 사람, 관리자
User	DVM을 사용하는 사람, 이용자
Bank	DVM과의 결제를 관리하는 Actor,

Commentary: Glossary (Data Dictionary)

In its simplest form, the **Glossary** is a list of noteworthy terms and their definitions. It is surprisingly common that a term, often technical or particular to the domain, will be used in slightly different ways by different stakeholders; this needs to be resolved to reduce problems in communication and ambiguous requirements.

> *Suggestion*
>
> Start the Glossary early. I'm reminded of an experience working with simulation experts, in which the seemingly innocuous, but important, word "cell" was discovered to have slippery and varying meanings among the group members.

The goal is not to record all possible terms, but those that are unclear, ambiguous, or which require some kind of noteworthy elaboration, such as format information or validation rules.

Glossary as Data Dictionary

In the UP, the Glossary also plays the role of a **data dictionary,** a document that records data about the data—that is, **metadata.** During inception the glossary should be a simple document of terms and descriptions. During elaboration, it may expand into a data dictionary.

The Glossary is not only for atomic terms such as "product price." It can and should include composite elements such as "sale" (which includes other elements, such as date and location), and nicknames used to describe a collection of data transmitted between actors in the use cases. For example, in the *Process Sale* use case, consider the following statement:

> System sends payment authorization request to an external Payment Authorization Service, and requests payment approval.

"Payment authorization request" is a nickname for an aggregate of data, which needs to be explained in the Glossary.

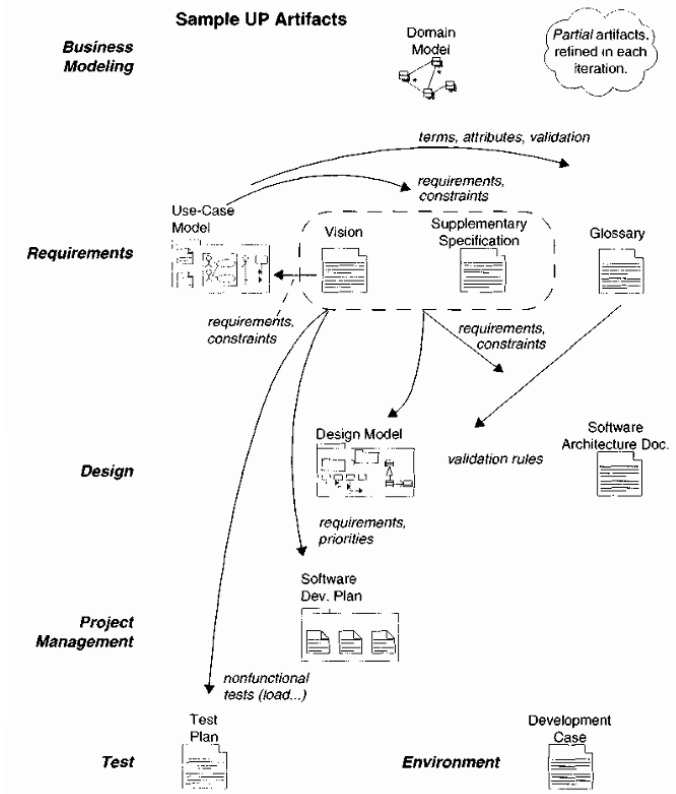


Figure 7.1 Sample UP artifact influence.

정리)

FR을 찾기 위해 UC 분해

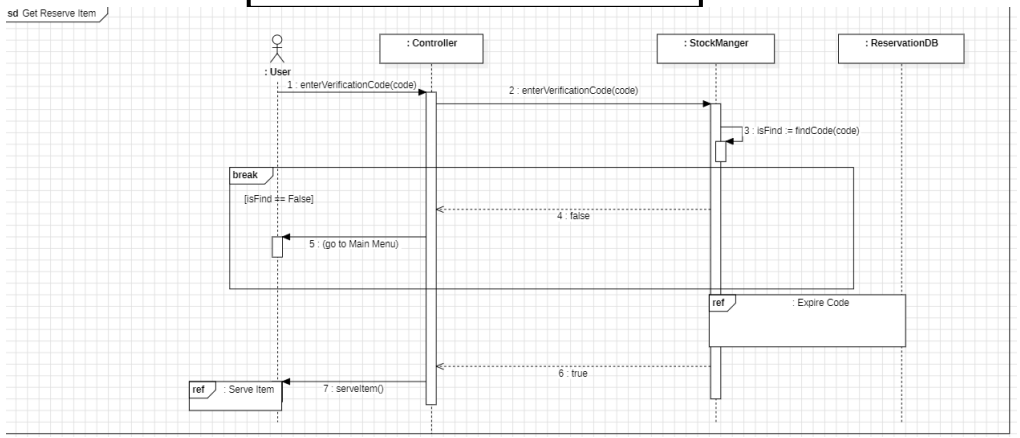
- **Traceability**, **Glossary**에 대해 상세히 기술

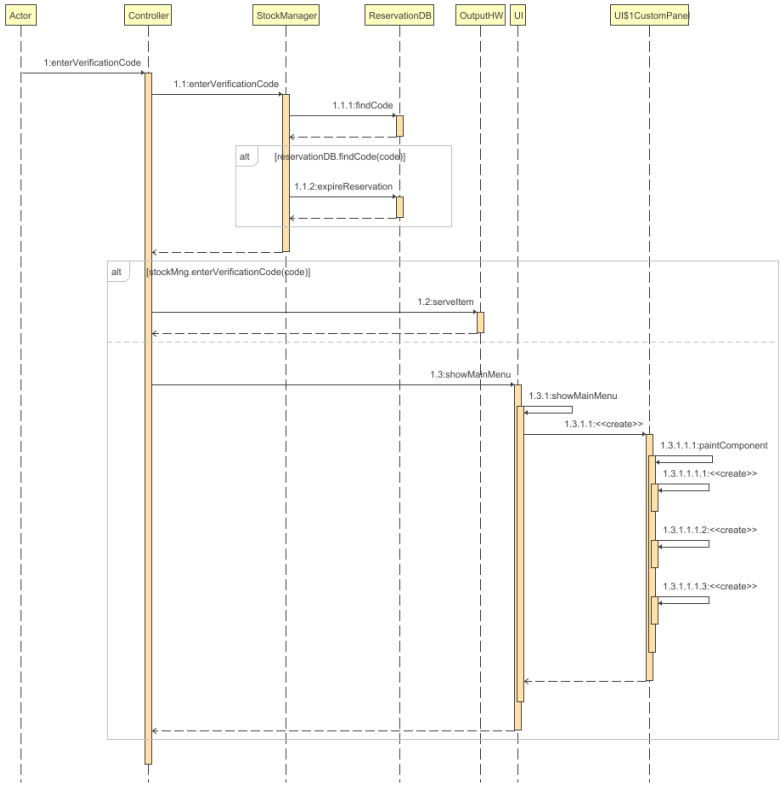
- > 개발 및 유지보수에 도움

6. 구현 시 예상보다 쉬웠던 점

- SD, DCD 완료 -> 구현은 간편

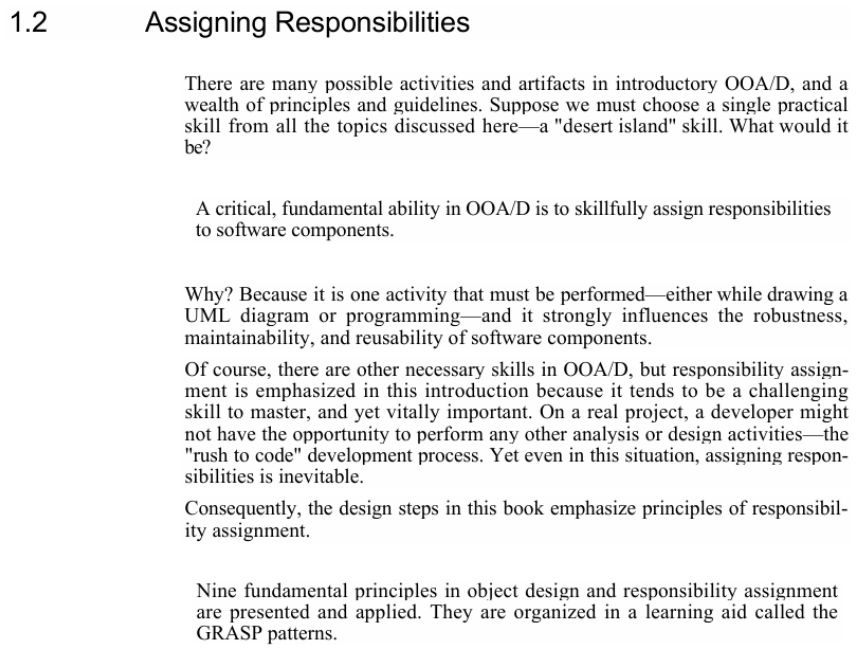
#21 Get Reserve Item





7. 객체지향개발방법론의 장단점 및 개인적인 소감

장점	단점
- 불필요한 필드, 메서드 등이 프로젝트에 포함되지 않음. - 무분별한 Getter, Setter 남발하지 않음. - 구현 전에 필요한 모든 요소를 분석, 구현 시작 후 테스트까지 오랜 시간이 걸리지 않음 - 예상치 못한 버그로 시간을 보내는 일이 적음.	- 설계에 많은 시간을 투자, 구현 단계에서 맞지 않는 부분이 발생하면 되돌아가야 함. - 구현 전 분석 및 설계 단계에서 필요하는 작업이 많음.



1.2 Assigning Responsibilities

There are many possible activities and artifacts in introductory OOA/D, and a wealth of principles and guidelines. Suppose we must choose a single practical skill from all the topics discussed here—a "desert island" skill. What would it be?

A critical, fundamental ability in OOA/D is to skillfully assign responsibilities to software components.

Why? Because it is one activity that must be performed—either while drawing a UML diagram or programming—and it strongly influences the robustness, maintainability, and reusability of software components.

Of course, there are other necessary skills in OOA/D, but responsibility assignment is emphasized in this introduction because it tends to be a challenging skill to master, and yet vitally important. On a real project, a developer might not have the opportunity to perform any other analysis or design activities—the "rush to code" development process. Yet even in this situation, assigning responsibilities is inevitable.

Consequently, the design steps in this book emphasize principles of responsibility assignment.

Nine fundamental principles in object design and responsibility assignment are presented and applied. They are organized in a learning aid called the GRASP patterns.