DVM OOPT 3rd Cycle

객체지향개발방법론 3팀

권민혁, 노성준, 이서준

목차

1. Resolution 여부 - 2nd cycle 에서의 1차 피드백

BRUTE FORCE TEST

Test	Test #	Description	Expected output	Test output	P / F
상품 목록, 메뉴 출력	1 - 1	ID, 상품명, 가격, 재고, 메뉴 정상 출력 여부 확인	ID, 상품명, 가격, 재고, 메뉴	ID, 상품명, 가격, 메뉴	F
	1 – 2	1을 눌러 음료 선택	음료 선택 메뉴로 이동	음료 선택 메뉴로 이동	P
	1 - 3	0을 눌러 종료	프로그램 종료	프로그램 종료	P
	1 - 4	<0 && >2의 정수 입력	에러 메시지 출력, 메뉴 선택으로 복귀	에러 메시지 출력, 메뉴 선택으로 복귀	P
	1 - 5	실수 입력	에러 메시지 출력, 메뉴 선택으로 복귀	에러 메시지 무한 출력	F
	1 - 6	문자 입력	에러 메시지 출력, 메뉴 선택으로 복귀	프로그램 종료	F
	1 - 7	문자열 입력	에러 메시지 출력, 메뉴 선택으로 복귀	프로그램 종료	F
음료 선택	2 - 1	유효하지 않은 정수 ID 입력	에러 메시지 출력, 음료 선택으로 복귀	수량 입력으로 진행	F
	2 - 2	유효하지 않은 실수 ID 입력	에러 메시지 출력, 음료 선택으로 복귀	에러 메시지 무한 출력	F
	2 - 3	유효하지 않은 문자 ID 입력	에러 메시지 출력, 음료 선택으로 복귀	에러 메시지 무한 출력	F
	2 - 3	유효하지 않은 문자열 ID 입력	에러 메시지 출력, 음료 선택으로 복귀	에러 메시지 무한 출력	F
	2 - 4	유효한 음료 ID 입력	수량 입력으로 진행	수량 입력으로 진행	P

1. Resolution 여부 - 2nd cycle 에서의 1차 피드백

BRUTE FORCE TEST

Test	Test #	Description	Expected output	Test output	P / F
수량 선택	3 - 1	0 입력	에러 메시지 출력, 수량 선택으로 복귀	에러 메시지 출력, 메뉴 선택으로 복귀	P
	3 - 2	음수 입력	에러 메시지 출력, 수량 선택으로 복귀	에러 메시지 출력, 메뉴 선택으로 복귀	P
	3 - 3	재고 초과 정수 입력	가장 가까운 다른 자판기 위치 안내	가장 가까운 다른 자판기 위치 안내	P
	3 - 4	1 이상, 재고 이내의 정수 입력	카드 번호 입력 메뉴 이동	카드 번호 입력 메뉴 이동	P
	3 - 5	1 이상, 재고 이내의 실수 입력	에러 메시지 출력, 수량 선택으로 복귀	소수점 자리가 카드 번호로 입력	F
	3 - 6	문자 입력	에러 메시지 출력, 수량 선택으로 복귀	에러 메시지 무한 출력	F
카드 번호 입력	4 - 1	잔고가 충분한 상태의 유효한 카드 번호 입력	카드 결제 성공, 카드 잔고 차감	카드 결제 성공, 카드 잔고 차감	P
	4 - 2	잔고가 부족한 상태의 유효한 카드 번호 입력	카드 결제 실패, 메뉴 선택으로 복귀	카드 결제 실패, 메뉴 선택으로 복귀	P
	4 - 3	유효하지 않은 카드 번호 입력	3회 실패 시 에러 메시지 출력 후 메뉴 선택으로 복귀	3회 실패 시 에러 메시지 출력 후 메뉴 선택으로 복귀	P
	4 - 4	유효하지 않은 카드 번호 1~2회 입력 후 잔고가 충분한 상태의 유효한 카드 번호 입력	카드 결제 성공, 카드 잔고 차감	카드 결제 성공, 카드 잔고 차감	P
	4 - 5	유효하지 않은 카드 번호 1~2회 입력 후 잔고가 충분하지 않은 유효한 카드 번호 입력	카드 결제 실패, 메뉴 선택으로 복귀	카드 결제 실패, 메뉴 선택으로 복귀	P

#Test Result = 10/23 = 43.4%

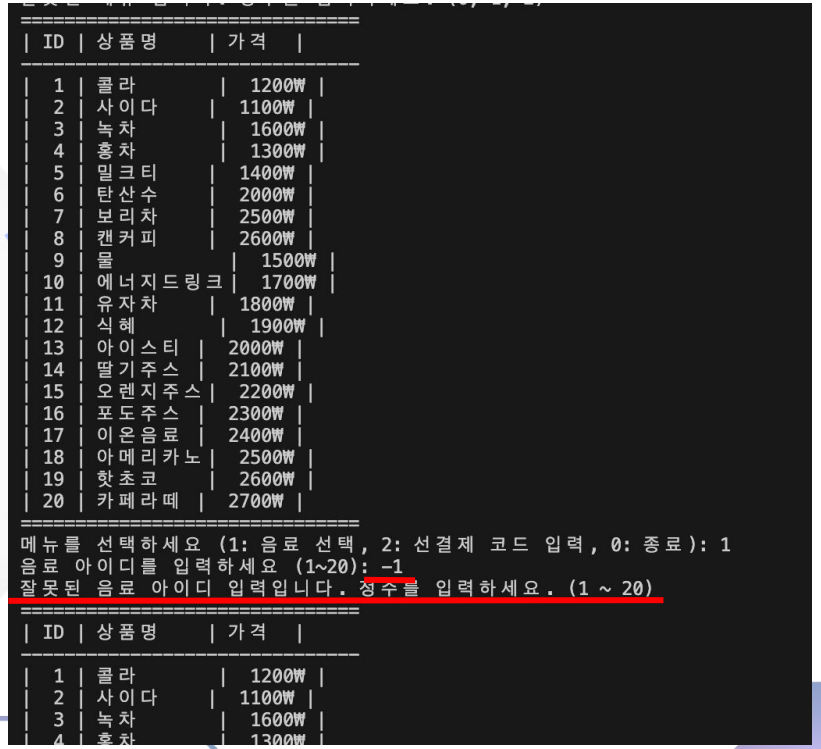
1. Resolution 여부 - 3rd cycle 에서의 2차 피드백

BRUTE FORCE TEST

Test	Test #	Description	Expected output	Test output	P / F
상품 목록, 메뉴 출력	1 – 1	실수 입력	에러 메시지 출력, 메뉴 선택으로 복귀	에러 메시지 출력, 메뉴 선택으로 복귀	P
	1 – 2	문자 입력	에러 메시지 출력, 메뉴 선택으로 복귀	에러 메시지 출력, 메뉴 선택으로 복귀	P
	1 – 3	문자열 입력	에러 메시지 출력, 메뉴 선택으로 복귀	에러 메시지 출력, 메뉴 선택으로 복귀	P
음료 선택	2 – 1	유효하지 않은 정수 ID 입력	에러 메시지 출력, 음료 선택으로 복귀	에러 메시지 출력, 프로그램 종료	F
	2 – 2	유효하지 않은 실수 ID 입력	에러 메시지 출력, 음료 선택으로 복귀	에러 메시지 출력, 음료 선택으로 복귀	P
	2 – 3	유효하지 않은 문자 ID 입력	에러 메시지 출력, 음료 선택으로 복귀	에러 메시지 출력, 음료 선택으로 복귀	P
	2 – 4	유효하지 않은 문자열 ID 입력	에러 메시지 출력, 음료 선택으로 복귀	에러 메시지 출력, 음료 선택으로 복귀	P
수량 선택	3 – 1	1 이상, 재고 이내의 실수 입력	에러 메시지 출력, 수량 선택으로 복귀	에러 메시지 출력, 수량 선택으로 복귀	P
	3 - 2	문자 입력	에러 메시지 출력, 수량 선택으로 복귀	에러 메시지 출력, 수량 선택으로 복귀	P

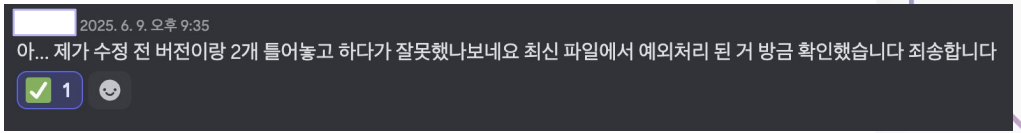
1. Resolution 여부 - 3rd cycle 에서의 2차 피드백

Test	Test#	Description	Expected output	Test output
음료 선택	2-1	유효하지 않은 정수 ID 입력	에러 메시지 출력, 메뉴 선택으로 복귀	수량 입력으로 진행



유효하지 않은 정수 ID 입력될 경우
-> 에러 메시지 출력, 메뉴 선택으로 복귀

2rd cycle에서 수정한 부분이었는데,
검증팀에서 구버전으로 테스트해 발생한 오류



1. Resolution 여부 - 3rd cycle 에서의 2차 피드백

BRUTE FORCE TEST

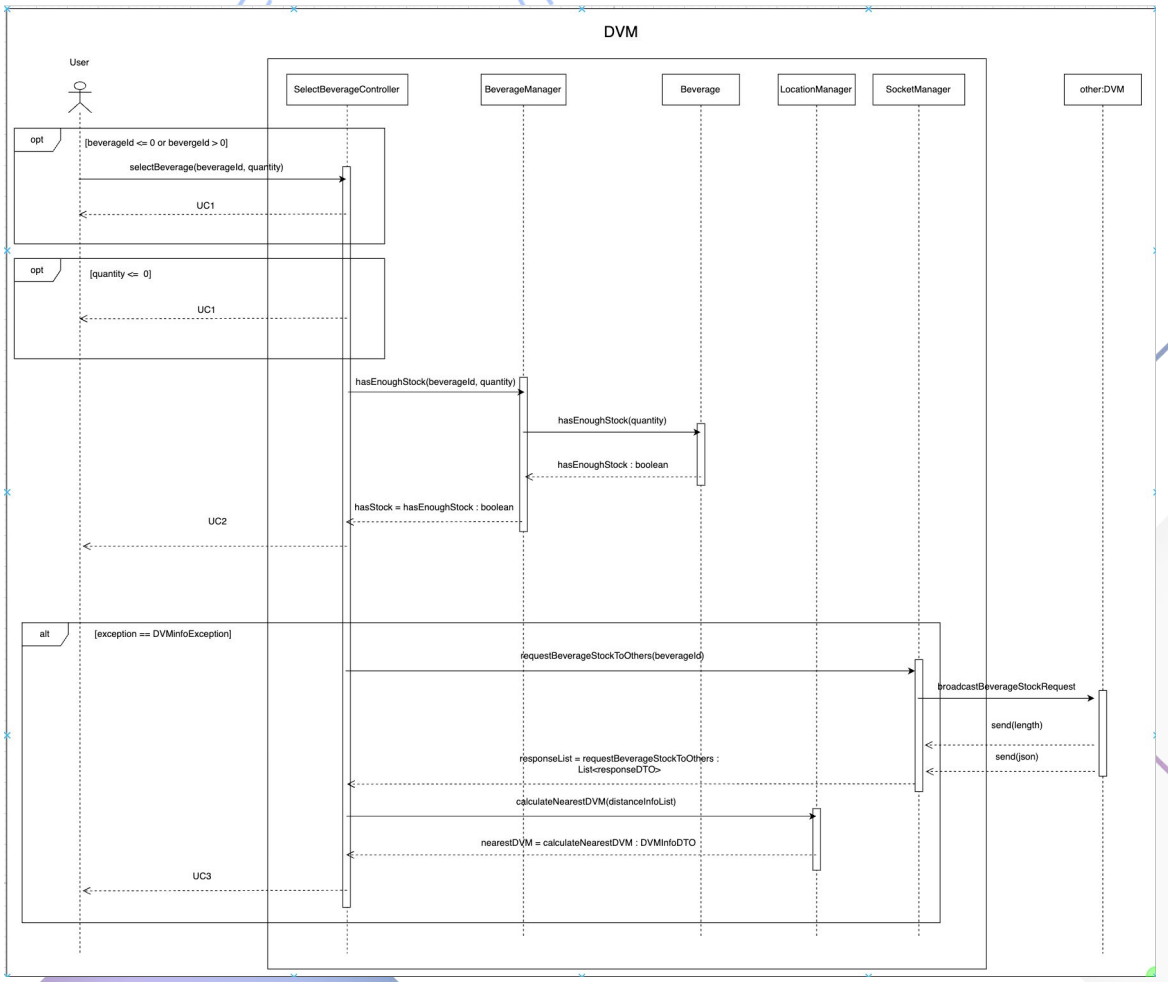
Test	Test #	Description	Expected output	Test output	P / F
선결제	4-1	재고 없는 음료 선택	재고 없음 출력, 선결제 진행	재고 없음 출력, 선결제 진행	P
	4-2	선결제 시도	인근 자판기 위치 출력	인근 자판기 위치 출력	P
	4-3	선결제 수량에 문자, 문자열, 실수 입력	에러 메시지 출력, 메뉴 선택으로 복귀	에러 메시지 출력, 메뉴 선택으로 복귀	P
	4-4	인근 자판기 재고 이내 입력	카드 번호 입력	카드 번호 입력	P
	4-5	선결제 카드번호 입력	결제 성공 혹은 오류 및 메뉴로 복귀	결제 성공 혹은 오류 및 메뉴로 복귀	P
	4-6	인근 자판기 재고를 초과하는 입력	재고 없음 출력, 메뉴 선택으로 복귀	재고 없음 출력, 메뉴 선택으로 복귀	P
	4-7	인근 자판기 재고 전체 선결제 후 즉시 구매(선결제 코드x) 시도	모든 재고가 이미 선결제되어 재고 없음 출력, 메뉴 선택으로 복귀	모든 재고가 이미 선결제되어 재고 없음 출력, 메뉴 선택으로 복귀	P
	4-8	정상 인증코드 입력	제품 제공	제품 제공	P
	4-9	잘못된 인증코드 입력	에러 메시지 출력, 메뉴 선택으로 복귀	에러 메시지 출력, 메뉴 선택으로 복귀	P

#Test Result = 17/18 = 94.5%

2. Usecase 별 동작화면

UC1. 음료 선택

1. (U) 사용자는 원하는 음료의 [ID, 원하는 수량]을 입력한다.
2. (D) DVM은 사용자가 입력한 음료 ID가 유효한지 확인한다.
3. (D) DVM은 사용자가 입력한 음료 ID에 해당하는 음료의 재고가 사용자로부터 받은 수량보다 크거나 같은지 확인한다.
4. (D) 음료의 재고가 있다면 사용자에게 결제 요청 화면을 보여준다.
5. (D) 음료의 재고가 없다면 다른 모든 DVM에게 해당 음료의 재고를 요청하여 확인한다.
6. (D) DVM은 자신과 가장 가깝고, 사용자가 구매하고자 하는 음료의 재고가 있는 자판기의 위치를 계산한 후, 선결제 요청 화면을 보여준다.



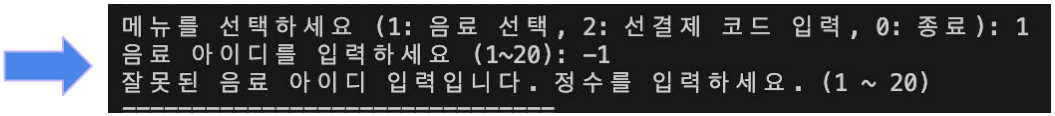
2. Usecase 별 동작화면

메뉴 출력

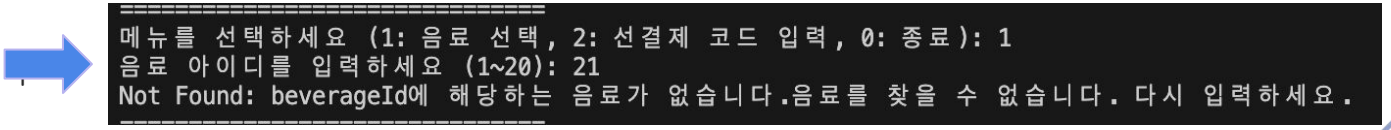
```
→  build git:(develop) ./sonar_scanner_example 2 9002
================================
| ID | 상품명      | 가격   |
--------------------------------
|  1 | 콜라        |  1200₩ |
|  2 | 사이다      |  1100₩ |
|  3 | 녹차        |  1600₩ |
|  4 | 홍차        |  1300₩ |
|  5 | 밀크티      |  1400₩ |
|  6 | 탄산수      |  2000₩ |
|  7 | 보리차      |  2500₩ |
|  8 | 캔커피      |  2600₩ |
|  9 | 물          |  1500₩ |
| 10 | 에너지드링크|  1700₩ |
| 11 | 유자차      |  1800₩ |
| 12 | 식혜        |  1900₩ |
| 13 | 아이스티    |  2000₩ |
| 14 | 딸기주스    |  2100₩ |
| 15 | 오렌지주스  |  2200₩ |
| 16 | 포도주스    |  2300₩ |
| 17 | 이온음료    |  2400₩ |
| 18 | 아메리카노  |  2500₩ |
| 19 | 핫초코      |  2600₩ |
| 20 | 카페라떼    |  2700₩ |
================================
메뉴를 선택하세요 (1: 음료 선택, 2: 선결제 코드 입력, 0: 종료):
```

2. Usecase 별 동작화면

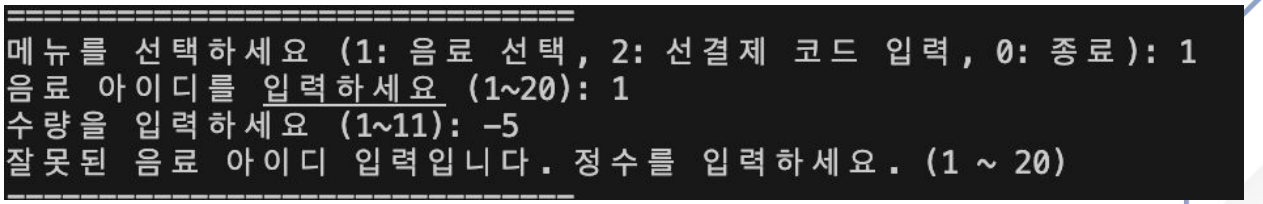
음료 아이디가 음수인 경우



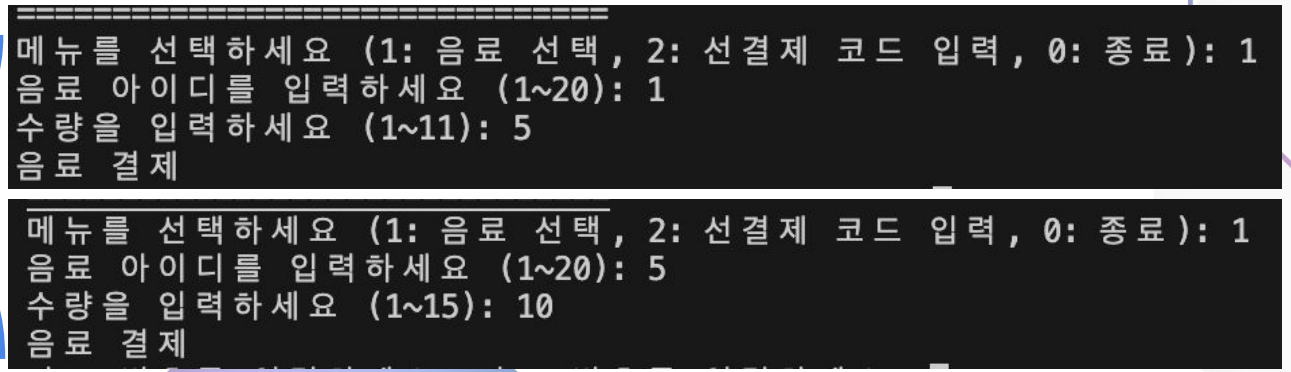
음료 아이디가 20 초과인 경우



음료 수량이 음수인 경우



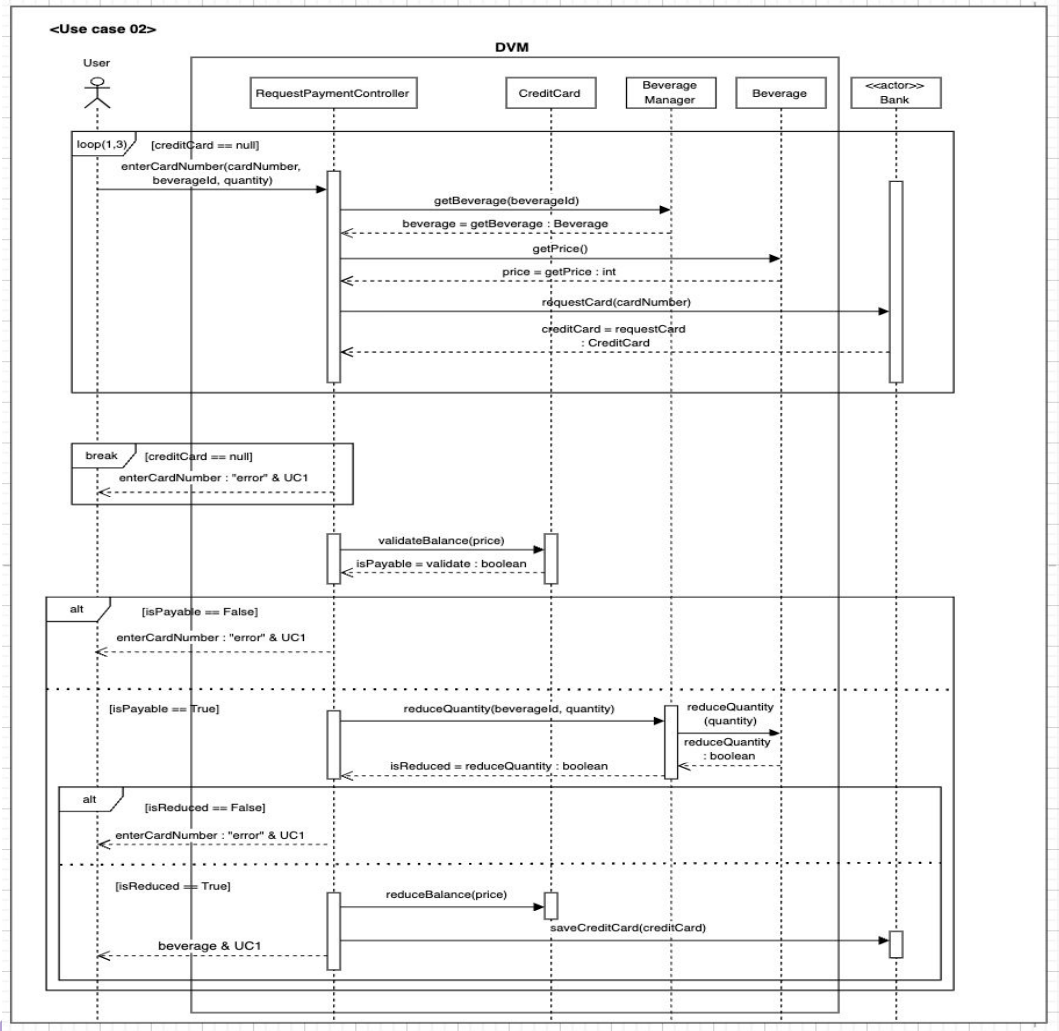
음료 수량 동적 출력



2. Usecase 별 동작화면

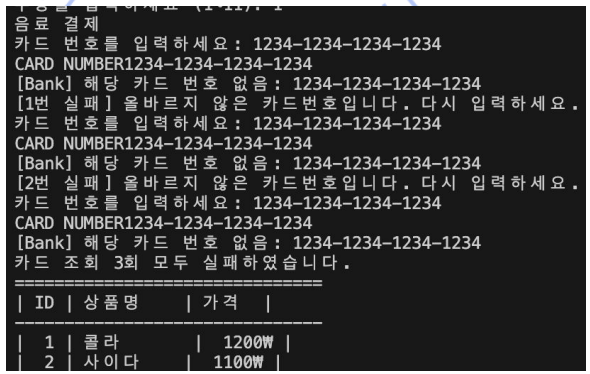
UC2. 결제 요청

1. (U) 사용자는 카드 번호를 입력한다.
2. (D) DVM은 사용자가 입력한 카드 번호로 Bank에게 카드를 요청한다.
3. (B) Bank는 카드 번호가 유효하다면, 카드 번호에 해당하는 카드를 반환한다.
4. (D) DVM은 카드의 잔액이 음료 구매가보다 크거나 같을 경우, 음료의 재고를 반환하고 카드의 잔액을 음료 구매가만큼 차감한다.
5. (D) DVM은 사용자에게 음료를 제공한다.

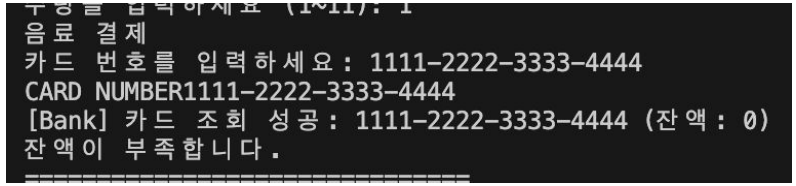


2. Usecase 별 동작화면

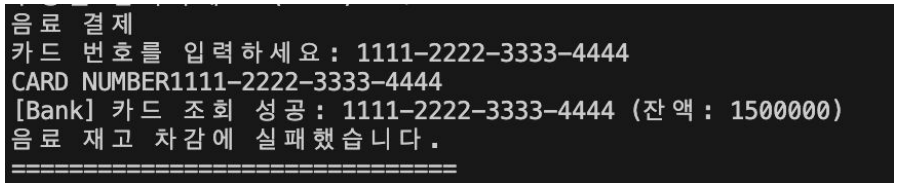
유효하지 않은
카드 번호 3번 입력



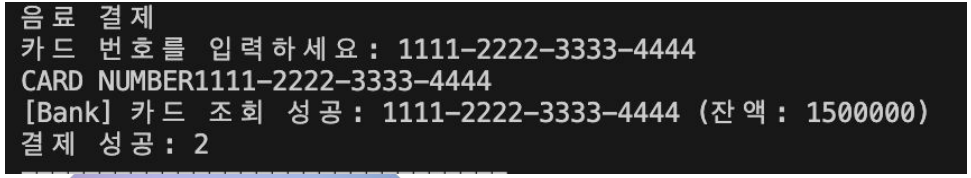
카드 잔액 부족



재고 차감 실패
(선결제로 인한)



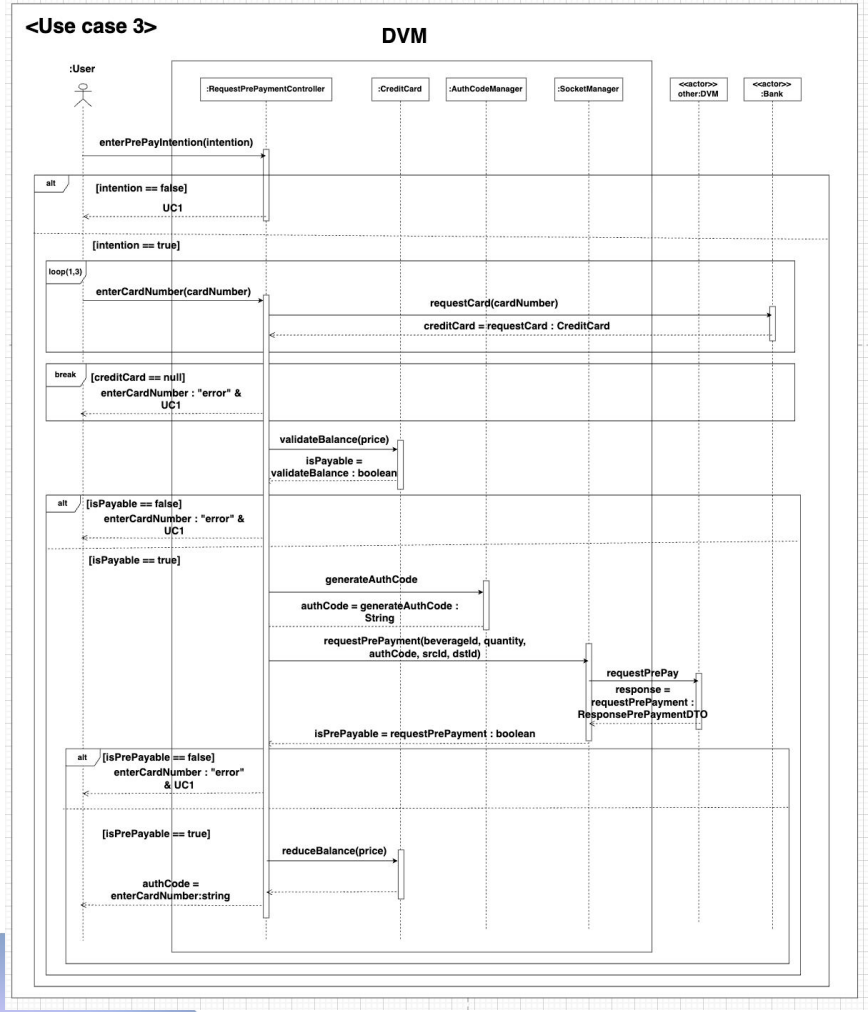
결제 성공



2. Usecase 별 동작화면

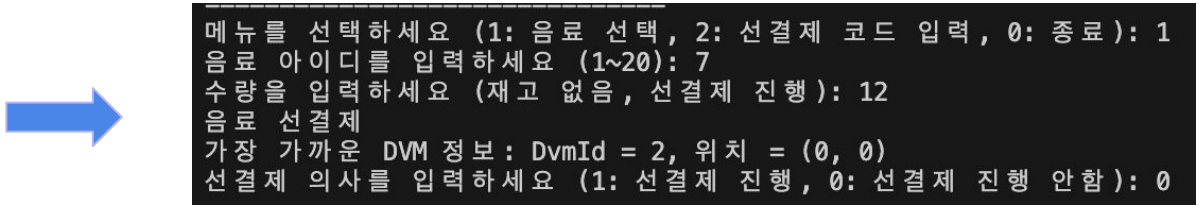
UC3. 선결제 요청

1. (U) 사용자는 선결제 의사를 입력한다.
2. (U) 선결제하고자 하는 사용자는 카드 번호를 입력한다.
3. (D) DVM은 사용자가 입력한 카드 번호로 Bank 에게 카드를 요청한다.
4. (B) Bank는 카드 번호가 유효하다면, 카드 번호에 해당하는 카드를 반환한다.
5. (D) DVM은 카드의 잔액이 음료 구매가보다 크거나 같을 경우, 인증코드를 생성한다.
6. (D) DVM은 사용자가 선결제하고자 하는 다른 DVM에게 생성한 인증코드와 선결제하고자 하는 음료 id, 수량 정보를 포함해서 선결제 요청을 보낸다.
7. (D) DVM은 다른 DVM에게 선결제 성공 응답을 받으면, 사용자에게 인증코드를 제공한다.

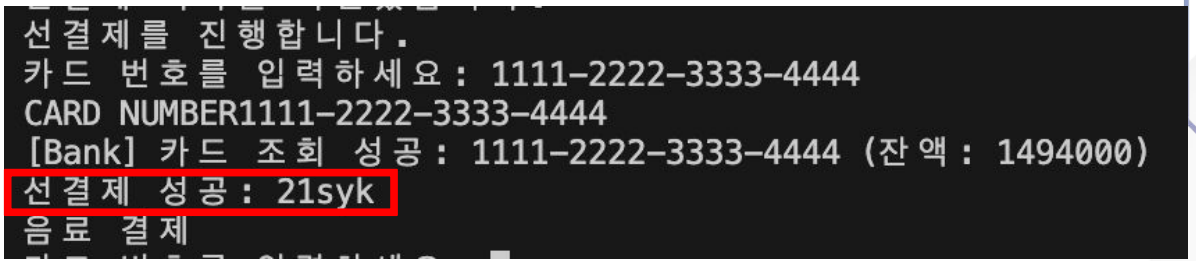


2. Usecase 별 동작화면

선결제 의사 없음



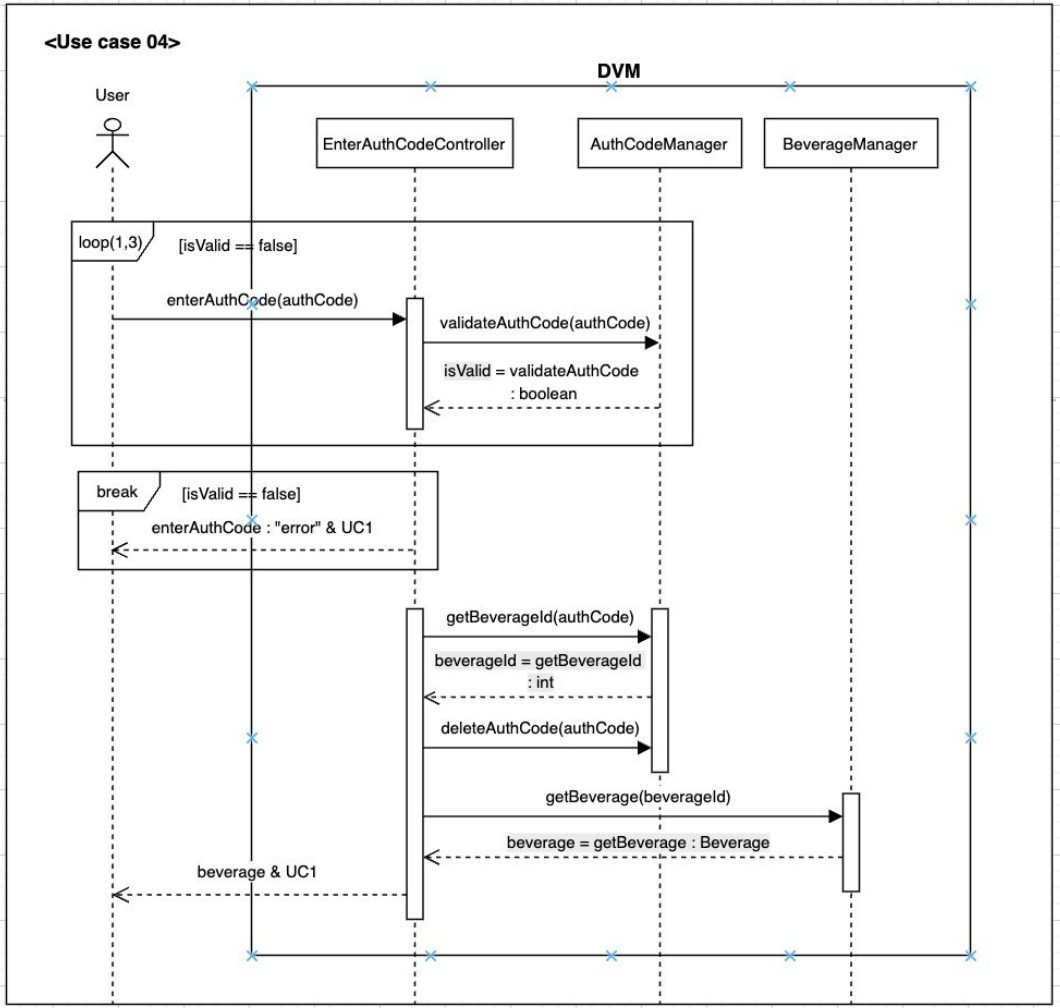
선결제 성공 인증 코드 발급



2. Usecase 별 동작화면

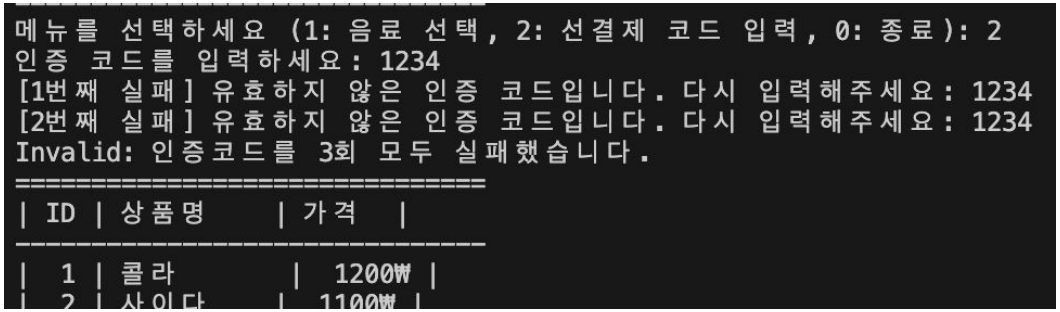
UC4. 인증 코드 입력 후 음료 수령

1. (U) 사용자는 인증 코드를 입력한다
2. (D) DVM은 사용자가 입력한 인증코드가 유효한지 검사한다.
3. (D) 인증코드가 유효하다면 DVM은 사용자에게 음료를 제공한다.

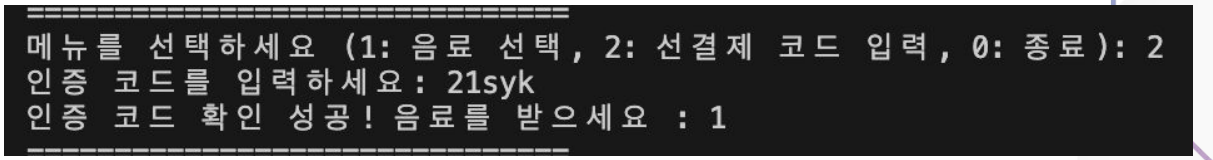


2. Usecase 별 동작화면

유효하지 않은
인증코드 3회
입력



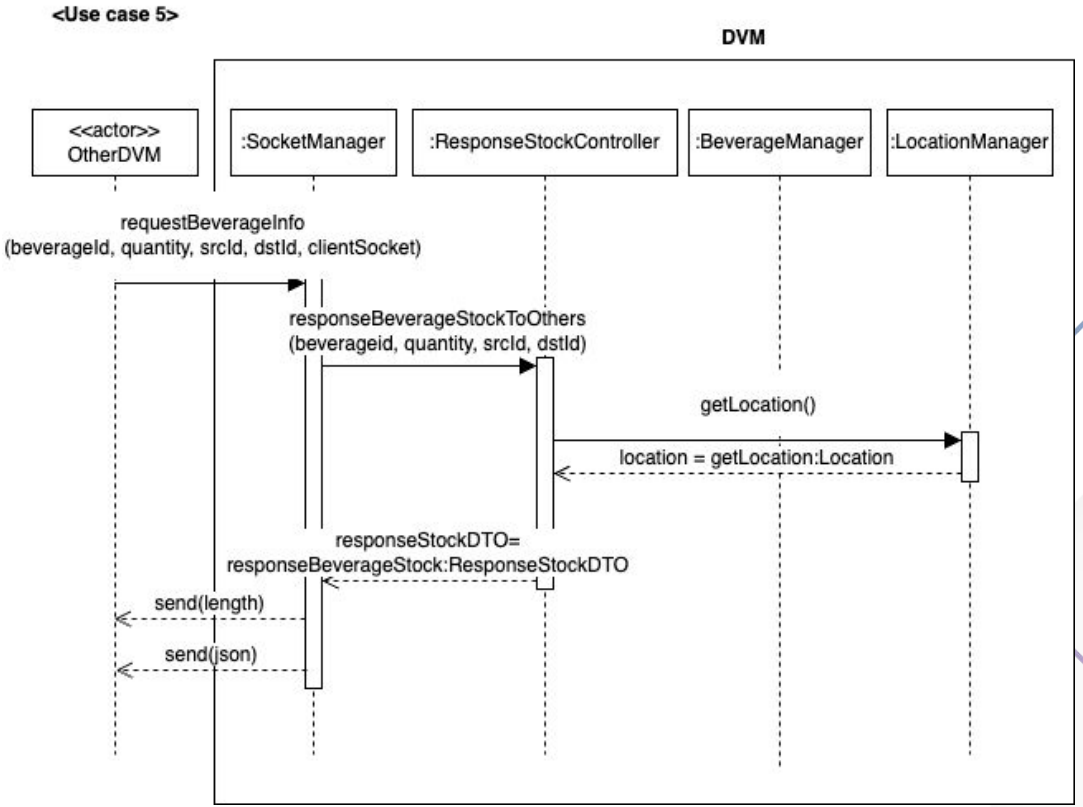
인증코드로 음료
수령 성공



2. Usecase 별 동작화면

UC5. 선결제시 재고 확인 응답

1. (O) 다른 DVM은 DVM에게 특정 음료에 대한 재고 확인 요청을 전송한다.
2. (D) DVM은 다른 DVM으로부터 재고 확인 요청 메시지를 수신한다.
3. (D) DVM은 재고 확인 요청에 포함된 음료의 재고를 확인한다.
4. (D) DVM은 음료의 재고와 자신의 위치 정보를 응답한다.



2. Usecase 별 동작화면

재고 확인 요청, 응답에 대한 json data 로그 화면

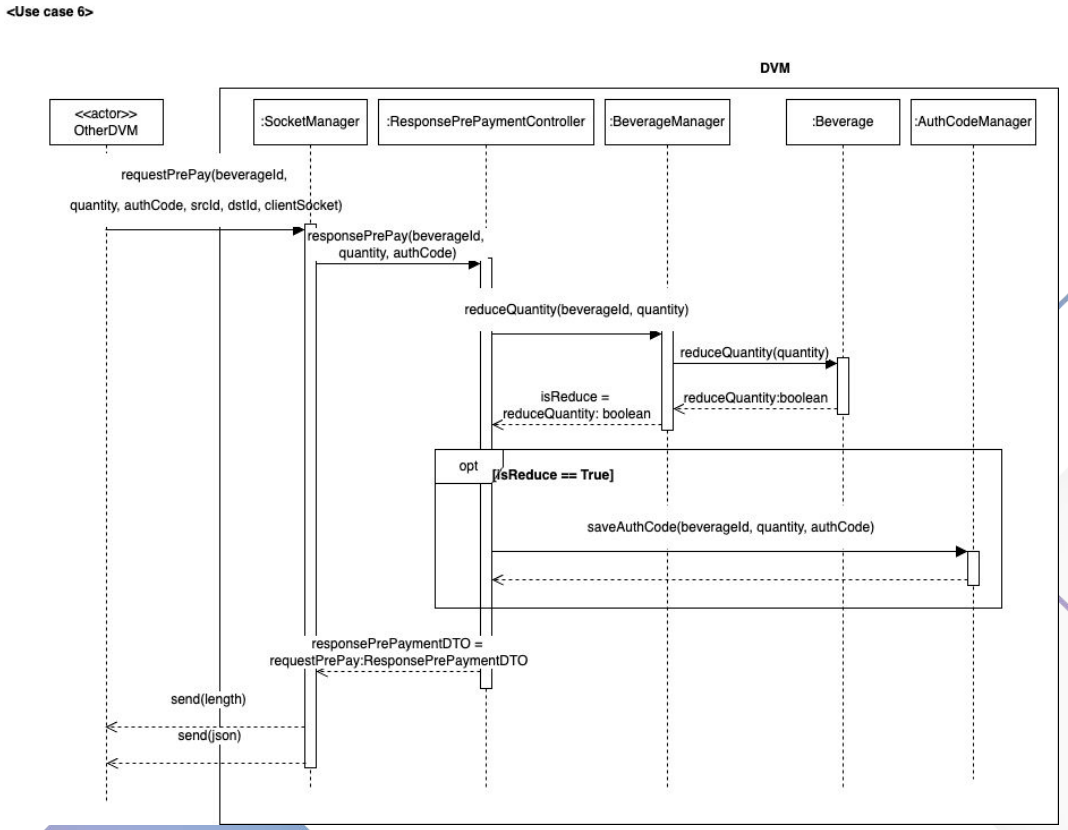
```
4 98
{"dst_id":1,"msg_content":{"item_code":7,"item_num":2},"msg_type":"req_stock","src_id":1804312176}
```

```
<<<<<<<<<<<receive data>>>>>>>>>>>>
4 121
{"dst_id":1804312176,"msg_content":{"coor_x":0,"coor_y":0,"item_code":7,"item_num":2},"msg_type":"resp_stock","src_id":1}
```

2. Usecase 별 동작화면

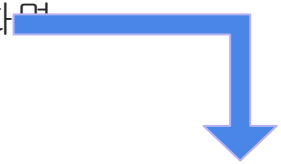
UC6. 선결제 요청 응답

1. (O) 다른 DVM은 DVM에게 특정 음료에 대한 선결제를 요청한다.
2. (D) DVM은 선결제 요청에 포함된 음료의 재고를 확인한다.
3. (D) DVM은 해당 음료의 재고가 있다면 재고를 차감하고, 선결제 요청에 포함된 인증 코드를 저장한 후, 선결제 성공을 응답한다.



2. Usecase 별 동작화면

선결제 요청,응답에 대한 json data 로그 화면



```
<<<<<<<<<<<receive data>>>>>>>>>>>>
4 110
{"dst_id":1,"msg_content":{"cert_code":"bg6gs","item_code":7,"item_num":2},"msg_type":"req_prepay","src_id":2}
```

```
<<<<<<<<<<<receive data>>>>>>>>>>>>
4 111
{"dst_id":2,"msg_content":{"availability":true,"item_code":7,"item_num":2},"msg_type":"resp_prepay","src_id":1}
```

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
1	AuthCodeManager	validateAuthCode	ValidateAuthCode_등록된 코드_True반환	미리 저장된 코드 "ABC12", "ABC34"에 대해 validateAuthCode가 true를 반환하는지 확인	true 반환
2	AuthCodeManager	validateAuthCode	ValidateAuthCode_없는코드_False반환	존재하지 않는 코드에 대해 NotFoundException이 발생하는지 확인	NotFoundException 발생
3	AuthCodeManager	getBeverageId	GetBeverageId_등록된코드_Id반환	미리 저장된 코드로 getBeverageId 호출 시 올바른 음료 ID를 반환하는지 확인	1, 2 반환
4	AuthCodeManager	getBeverageId	GetBeverageId_없는코드_exception_반환	존재하지 않는 코드로 getBeverageId 호출 시 NotFoundException이 발생하는지 확인	NotFoundException 발생
5	AuthCodeManager	generateAuthCode	GenerateAuthCode_생성된_인증코드_길이는_5	generateAuthCode로 생성된 코드의 길이가 5인지 확인	길이 5
6	AuthCodeManager	generateAuthCode	GenerateAuthCode_영어와_숫자만포함	생성된 코드가 영문 대소문자와 숫자로만 구성되어 있는지 확인	모두 영숫자 (isalnum)

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
7	AuthCodeManager	deleteAuthCode	DeleteAuthCode_등록된코드_성공	기존에 저장된 코드를 삭제한 뒤, 다시 조회하면 NotFoundException이 발생하는지 확인	삭제 후 조회 시 NotFoundException 발생
8	AuthCodeManager	deleteAuthCode	DeleteAuthCode_없는코드_exception	존재하지 않는 코드를 삭제하려 할 때 NotFoundException이 발생하는지 확인	NotFoundException 발생
9	AuthCodeManager	saveAuthCode	SaveAuthCode_새로운코드_저장후_조회성공	새로운 코드 "XYZ99"를 저장한 뒤 validateAuthCode와 getBeverageId가 정상 동작하는지 확인	true 반환, ID=7 반환
10	AuthCodeManager	deleteAuthCode	DeleteAuthCode_하나삭제_다른코드유효	"ABC12"만 삭제한 뒤 "ABC34"는 여전히 유효한지 확인	"ABC34" -> true 반환, ID=2
11	AuthCodeManager	saveAuthCode	SaveAuthCode_중복키_기존값유지	기존에 "ABC12"로 저장된 값이 덮어쓰이지 않고 유지되는지 확인	ID=1 반환 (덮어쓰기 없이 유지)

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
12	Beverage	getId	Beverage_getId_성공	생성자에 전달한 id(42)가 getId()로 반환되는지 확인	42 반환
13	Beverage	getPrice	Beverage_getPrice_성공	생성자에 전달한 price(1500)가 getPrice()로 반환되는지 확인	1500 반환
14	Beverage	hasEnoughStock	hasEnoughStock_0이면_TRUE반환	stock=5, 요청량 0일 때 true 반환	true 반환
15	Beverage	hasEnoughStock	hasEnoughStock_재고보다 작으면_TRUE반환	stock=5, 요청량 3일 때 true 반환	true 반환
16	Beverage	hasEnoughStock	hasEnoughStock_재고보다_1만큼_작아도_TRUE반환_경계값_테스트	stock=5, 요청량 4(경계값-1)일 때 true 반환	true 반환
17	Beverage	hasEnoughStock	hasEnoughStock_재고와같으면_TRUE반환	stock=5, 요청량 5(경계값)일 때 true 반환	true 반환

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
18	Beverage	hasEnoughStock	hasEnoughStock_재고보다 크면_FALSE반환	stock=5, 요청량 8일 때 false 반환	false 반환
19	Beverage	hasEnoughStock	hasEnoughStock_재고보다 _1만큼_커도_FALSE반환_ 경계값_테스트	stock=5, 요청량 6일 때 false 반환	false 반환
20	Beverage	hasEnoughStock	hasEnoughStock_음수요청 이면_TRUE반환	stock=5, 요청량 -1일 때 (현재 구현상) true 반환	true 반환
21	Beverage	reduceQuantity	reduceQuantity_0이면_TR UE반환	stock=7, 감소량 0일 때 true 반환	true 반환
22	Beverage	reduceQuantity	reduceQuantity_재고보다_ 작으면_TRUE반환	stock=7, 감소량 5일 때 true 반환	true 반환
23	Beverage	reduceQuantity	reduceQuantity_재고보다_ 1만큼_작아도_TRUE반환	stock=7, 감소량 6(경계값-1)일 때 true 반환	true 반환

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
24	Beverage	reduceQuantity	reduceQuantity_재고와_같으면_TRUE반환	stock=7, 감소량 7(경계값)일 때 true 반환	true 반환
25	Beverage	reduceQuantity	reduceQuantity_재고보다_크면_FALSE반환	stock=7, 감소량 10일 때 false 반환	false 반환
26	Beverage	reduceQuantity	reduceQuantity_재고보다_1만큼_커도_FALSE반환	stock=7, 감소량 8(경계값+1)일 때 false 반환	false 반환

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
27	BeverageManager	getBeverage	GetBeverage_성공_Beverage반환	유효한 ID(1)로 호출 시 Beverage 객체 반환	1 반환
28	BeverageManager	getBeverage	GetBeverage_실패_문자입력_예외	문자형 리터럴 'a'(=97) 입력 시 예외 발생	NotFoundException 던짐
29	BeverageManager	getBeverage	GetBeverage_실패_음수입력_예외	음수 ID(-1) 입력 시 예외 발생	NotFoundException 던짐
30	BeverageManager	getBeverage	GetBeverage_실패_0입력_예외	ID 0 입력 시 예외 발생	NotFoundException 던짐
31	BeverageManager	getBeverage	GetBeverage_실패_초과입력_예외	범위를 벗어난 ID(999) 입력 시 예외 발생	NotFoundException 던짐
32	BeverageManager	hasEnoughStock	HasEnoughStock_0인경우_TRUE반환	ID 1, 요청량 0일 때 true 반환	true 반환

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
33	BeverageManager	hasEnoughStock	HasEnoughStock_재고보다 작을경우_TRUE반환	ID 1, 요청량 3일 때 true 반환	true 반환
34	BeverageManager	hasEnoughStock	HasEnoughStock_재고와같을경우_TRUE반환	ID 2, 요청량 10일 때 true 반환	true 반환
35	BeverageManager	hasEnoughStock	HasEnoughStock_재고보다 클경우_FALSE반환	ID 2, 요청량 20일 때 false 반환	false 반환
36	BeverageManager	hasEnoughStock	HasEnoughStock_음수요청이면_TRUE반환	ID 1, 요청량 -5일 때 (현재 구현상) true 반환	true 반환
37	BeverageManager	hasEnoughStock	HasEnoughStock_존재하지않는ID이면_예외	존재하지 않는 ID(999,0,-1) 요청 시 예외 발생	NotFoundException 던짐
38	BeverageManager	reduceQuantity	ReduceQuantity_0인경우_TRUE반환	ID 1, 감소량 0일 때 true 반환	true 반환

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
39	BeverageManager	reduceQuantity	ReduceQuantity_재고보다작을경우_TRUE반환	ID 1, 감소량 3일 때 true 반환	true 반환
40	BeverageManager	reduceQuantity	ReduceQuantity_재고와같을경우_TRUE반환	ID 2, 감소량 10일 때 true 반환	true 반환
41	BeverageManager	reduceQuantity	ReduceQuantity_음수요청이면_FALSE반환	ID 1, 감소량 -2일 때 false 반환	false 반환
42	BeverageManager	reduceQuantity	ReduceQuantity_존재하지않는ID이면_예외	존재하지 않는 ID(999,0,-1) 요청 시 예외 발생	NotFoundException 던짐

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
43	Bank	requestCard	RequestCard_있는번호_조회성공	DB에 존재하는 카드 번호로 호출 시 유효한 CreditCard 객체 반환	유효한 카드인 card->getCardNumber() == "1234-5678-9012-3456" 반환
44	Bank	requestCard	RequestCard_없는번호_조회실패	DB에 없는 카드 번호로 호출 시 예외 발생	NotFoundException 던짐
45	Bank	saveCreditCard	SaveCreditCard_기존카드_잔액업데이트	기존 DB 카드의 잔액 변경 후 저장하고, 조회 시 업데이트된 잔액 반환	requestCard()->getBalance() == 20000 반환
46	Bank	saveCreditCard	SaveCreditCard_신규카드_추가후조회	DB에 없는 신규 카드 저장 후 requestCard 가능 확인	card->getCardNumber() == "1111-2222-3333-4444", getBalance() == 3000 반환
47	Bank	requestCard	RequestCard_파일열기실패_예외발생	DB 파일이 없을 때 requestCard 호출 시 예외 발생	FileOpenException 던짐

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
48	CreditCard	validateBalance	ValidateBalance_0_성공	price=0일 때 validateBalance(0)가 true 반환되는지 확인	true 반환
49	CreditCard	validateBalance	ValidateBalance_유효한 값_성공	price=3000일 때 validateBalance(3000)가 true 반환되는지 확인	true 반환
50	CreditCard	validateBalance	ValidateBalance_경곗값 _max_성공	price=5000(잔액과 동일)일 때 validateBalance(5000)가 true 반환되는지 확인	true 반환
51	CreditCard	validateBalance	ValidateBalance_경계값 _5001_예외	price=5001(잔액 초과)일 때 validateBalance(5001)가 false 반환되는지 확인	false 반환
52	CreditCard	validateBalance	ValidateBalance_초과값 _예외	price=99999(잔액 훨씬 초과)일 때 validateBalance(99999)가 false 반환되는지 확인	false 반환
53	CreditCard	reduceBalance	ReduceBalance_잔액감 소	reduceBalance(1000) 호출 후 잔액이 4000으로 감소했는지 validateBalance(4000)로 확인	validateBalance(4000) → true 반환

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
56	Location	getX	getX_정상작동	생성자에 전달한 x(7)가 getX()로 반환되는지 확인	7 반환
57	Location	getY	getY_정상작동	생성자에 전달한 y(-3)가 getY()로 반환되는지 확인	-3 반환
58	Location	distanceTo	distanceTo_같은좌표_0 반환	동일 좌표(5,5) 간 거리가 0.0인지 확인	0.0 반환
59	Location	distanceTo	distanceTo_가로이동_거리계산	(10,2) → (4,2) 수평 이동 시 거리 6.0인지 확인	6.0 반환
60	Location	distanceTo	distanceTo_세로이동_거리계산	(-1,8) → (-1,3) 수직 이동 시 거리 5.0인지 확인	5.0 반환
61	Location	distanceTo	distanceTo_대각선이동_거리계산	(0,0) → (3,4) 대각선 이동 시 피타고라스 5.0인지 확인	5.0 반환

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
54	CreditCard	isValid	IsValid_있는번호_성공	유효한 카드번호일 때 isValid()가 true 반환되는지 확인	true 반환
55	CreditCard	isValid	IsValid_없는번호_실패	빈 문자열 카드번호일 때 isValid()가 false 반환되는지 확인	false 반환

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
62	Location	distanceTo	distanceTo_음수좌표_거리계산	(-2,-3) → (1,1) 음수 좌표 간 거리 √(9+16)=5.0인지 확인	5.0 반환
63	Location	distanceTo	distanceTo_큰차이_거리계산	(1000,-1000) → (-500,200) 간 거리(√(1500²+(-1200)²))인지 확인	expected 변수 값 반환 (≈1920.937)
64	LocationManager	calculateNearest	calculateNearest_단일요소_반환	responseList에 단일 요소 전달 시 해당 요소의 DVMInfoDTO 반환 확인	X=3, Y=4, prePaymentDvmId=100 반환
65	LocationManager	calculateNearest	calculateNearest_여러요소_가장가까운반환	여러 요소 중 최단 거리인 요소 반환 확인	X=1, Y=1, prePaymentDvmId=2 반환
66	LocationManager	calculateNearest	calculateNearest_음수좌표_반환	음수 좌표를 가진 요소 반환 확인	X=-3, Y=-4, prePaymentDvmId=42 반환

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
67	SelectBeverage Controller	selectBeverage	재고충분_성공	beverageId, quantity 값에 문제가 없고, 선택한 음료의 재고가 충분할 경우, 아무 예외 없이 리턴	exception not throw
68	SelectBeverage Controller	selectBeverage	잘못된BeverageId_예외 발생	입력받은 beverageId 값이 1 ~ 20 사이의 값이 아닌 경우, exception 발생	InvalidException 발생
69	SelectBeverage Controller	selectBeverage	잘못된Quantity_예외발생	입력받은 quantity 값이1 이상이 아닐 경우, exception 발생	InvalidException 발생
70	SelectBeverage Controller	selectBeverage	BeverageNotFound_InvalidException으로변환	입력받은 beverageId 값이 잘못된 beverageId 인 경우, exception 발생	InvalidException 발생
71	SelectBeverage Controller	selectBeverage	재고부족_DVMInfoException 발생	입력받은 beverageId 값에 해당하는 음료의 재고가 quantity보다 작을 경우, broadcast 통신을 통해 선결제 대상인 dvm을 계산 후, 선결제 대상의 정보를 담은 exception 발생	DVMInfoException 발생

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
72	RequestPaymentController	enterCardNumber	성공_유효카드_잔액충분_재고차감성공	유효하고, 잔액이 충분한 카드를 입력할 경우, 음료 반환 성공	음료 반환 성공
73	RequestPaymentController	enterCardNumber	카드3회실패_예외발생	카드번호를 3회 모두 잘못 입력할 경우, exception 발생	CardNotFoundException 발생
74	RequestPaymentController	enterCardNumber	잔액부족_예외발생	입력받은 카드번호의 잔액이 부족할 경우, exception 발생	InsufficientBalanceException 발생
75	RequestPaymentController	enterCardNumber	재고차감실패_예외발생	결제하려는 음료의 재고가 사용자가 입력한 quantity 값보다 작을 경우, exception 발생	BeverageReductionException 발생

3. 테스트 내용 설명

Number	Class	MethodName	testMethodName	Description	Result
76	RequestPrePaymentController	responsePrePay	responsePrePay_잘못된_카드번호_예외	선결제시 잘못된 카드번호 입력으로 인한 실패	InvalidException 발생
77	RequestPrePaymentController	responsePrePay	responsePrePay_잔액부족_예외	선결제시 잔액부족으로 인한 실패	FailedToPrePayException 발생
78	RequestPrePaymentController	responsePrePay	responsePrePay_선결제_성공	선결제 성공	랜덤한 authCode 반환
79	EnterAuthCodeController	EnterAuthCode	EnterAuthCode_유효한_음료반환	올바른 인증코드를 입력 후 음료 반환	beverageId 1반환
80	EnterAuthCodeController	EnterAuthCode	EnterAuthCode_잘못된 코드_예외발생	잘못된 인증코드를 3회 입력 후 예외발생	InvalidException 발생

4. 정적 분석 결과

Static Analysis

File	Line	Severity	Review
Beverage.cpp	36	Medium	원인: 멤버 변수를 수정하지 않는 함수에 const 키워드가 선언되지 않았음. 대처: 기능이 아닌 Notation의 문제이므로 수정을 요구하지 않음.
BeverageManager.h	6	Medium	원인: 소멸자가 제대로 호출되지 않아 자원 누수가 발생할 수 있음 대처: 아직 자원 관리가 중요하지 않아 문제가 될 시 수정 요구.
Bank.h	14	Medium	원인: 소멸자가 제대로 호출되지 않아 자원 누수가 발생할 수 있음 대처: 아직 자원 관리가 중요하지 않아 문제가 될 시 수정 요구.
CreditCard.h	5	Medium	원인: 소멸자가 제대로 호출되지 않아 자원 누수가 발생할 수 있음 대처: 아직 자원 관리가 중요하지 않아 문제가 될 시 수정 요구.
LocationManager.h	10	Medium	원인: 소멸자가 제대로 호출되지 않아 자원 누수가 발생할 수 있음 대처: 아직 자원 관리가 중요하지 않아 문제가 될 시 수정 요구.
SocketManager.h	23	Medium	원인: 소멸자가 제대로 호출되지 않아 자원 누수가 발생할 수 있음 대처: 아직 자원 관리가 중요하지 않아 문제가 될 시 수정 요구.
EnterAuthCodeController.h	9	Medium	원인: 소멸자가 제대로 호출되지 않아 자원 누수가 발생할 수 있음 대처: 아직 자원 관리가 중요하지 않아 문제가 될 시 수정 요구.
RequestPaymentController.h	9	Medium	원인: 소멸자가 제대로 호출되지 않아 자원 누수가 발생할 수 있음 대처: 아직 자원 관리가 중요하지 않아 문제가 될 시 수정 요구.

4. 정적 분석 결과

Static Analysis

File	Line	Severity	Review
RequestPrePaymentController.h	13	Medium	원인: 소멸자가 제대로 호출되지 않아 자원 누수가 발생할 수 있음 대처: 아직 자원 관리가 중요하지 않아 문제가 될 시 수정 요구.
main.cpp	174	Medium	원인: try-catch 문의 중첩 대처: 가독성의 문제이므로 수정을 요구하지 않음.
	251~260	Medium	원인: delete의 부적절한 사용으로 인한 메모리 낭비 대처: 각 모듈의 헤더파일을 수정하면 해결되기 때문에 main.cpp은 수정을 요구하지 않음.
	23	Low	원인: SonarQube에서는 for loop보다 std:: 사용을 권장 대처: 수정을 요구하지 않음.
	176	Low	원인: Parameter의 미사용 지적 대처: 삭제가 가독성 저하를 일으킬 수 있으므로 수정을 요구하지 않음.

4. 정적 분석 결과

```
class Bank {
private:
    list<CreditCard> cards;

public:
    Bank() = default;
    ~Bank() {
        for (CreditCard& card : cards) {
            delete &card;
        }
        cards.clear();
    };
    virtual CreditCard* requestCard(string cardNumber);
    virtual void saveCreditCard(CreditCard creditCard);
};
```

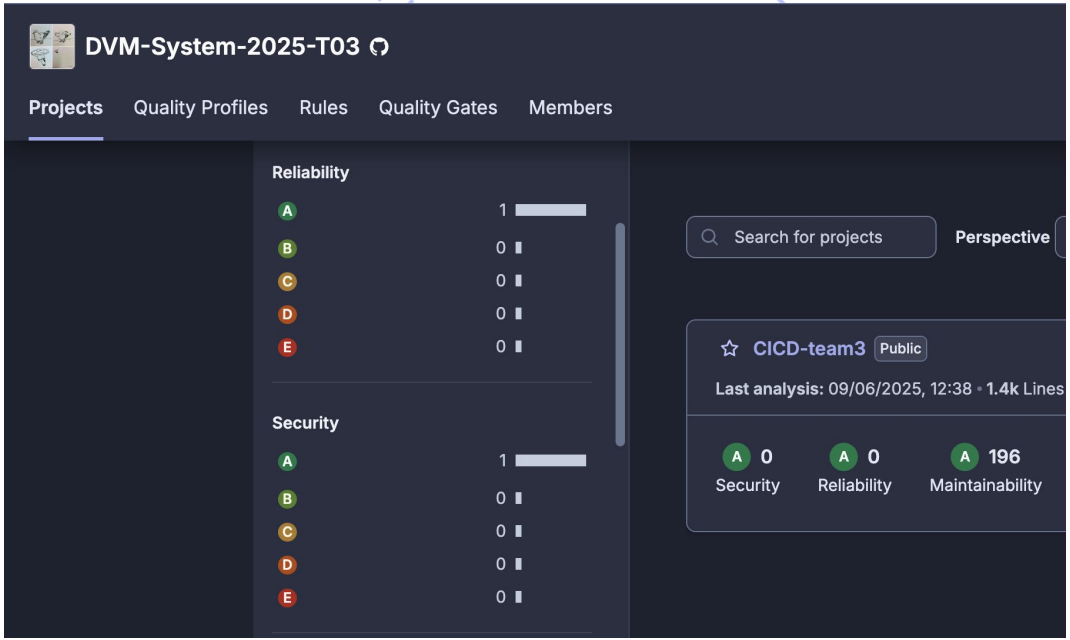
```
class SocketManager {
private:
    ResponseStockController *responseStockController;
    ResponsePrePaymentController *responsePrePaymentController;
    map<int,pair<string,int>> otherDVMInfo;

    map<int,int> otherDVMSockets;
    int serverSocket;
    int serverPort;
    int srcId;

    void connectOtherDVMSocket();
    void openServerSocket();
    void waitingRequest();


public:
    SocketManager() = default;
    SocketManager(int srcId, int serverPort);
    ~SocketManager() {
        for (auto& socket : otherDVMSockets) {
            close(socket.second);
        }
        close(serverSocket);
    }
```

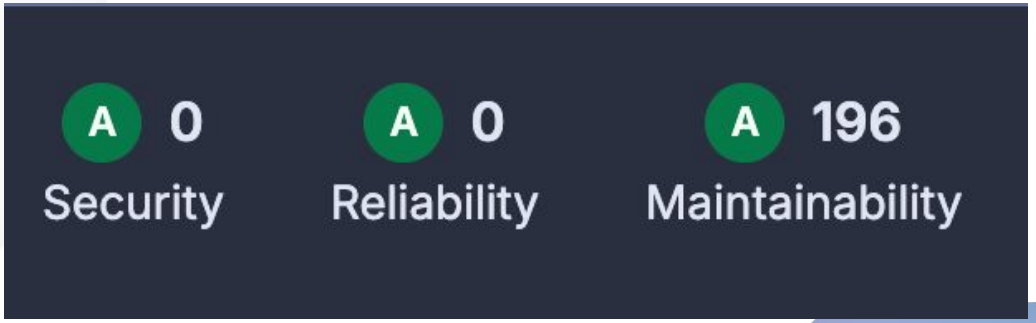
4. 정적 분석 결과

```
31  int main(int argc, char* argv[]) {
32
33      SocketManager* socketManager = new SocketManager(atoi(argv[1]), atoi(argv[2]));
34      AuthCodeManager* authCodeManager = new AuthCodeManager();
35      Bank* bank = new Bank();
36      BeverageManager* beverageManager = new BeverageManager();
37      LocationManager* locationManager = new LocationManager(0, 0);   // 현재 DVM 위치 (0, 0) -> 임시로 설정
38
39      SelectBeverageController* selectBeverageController = new SelectBeverageController(locationManager, beverageManager, socketManager);
40      RequestPrePaymentController* requestPrePaymentController = new RequestPrePaymentController(authCodeManager, bank, socketManager, beverageManager);
41      EnterAuthCodeController* enterAuthCodeController = new EnterAuthCodeController(beverageManager, authCodeManager);
42      ResponsePrePaymentController* responsePrePaymentController = new ResponsePrePaymentController(beverageManager, authCodeManager);
43      RequestPaymentController* requestPaymentController = new RequestPaymentController(beverageManager, bank);
44      ResponseStockController* responseStockController = new ResponseStockController(locationManager, beverageManager);
45
46      socketManager->setController(responseStockController, responsePrePaymentController);
```

```
31  int main(int argc, char* argv[]) {
95      while (true) {
249     }
250
251     delete socketManager;
252     delete authCodeManager;
253     delete bank;
254     delete beverageManager;
255     delete locationManager;
256     delete selectBeverageController;
257     delete requestPrePaymentController;
258     delete enterAuthCodeController;
259     delete responsePrePaymentController;
260     delete requestPaymentController;
261     delete responseStockController;
262
263
264     return 0;
265 }
```

4. 정적 분석 결과





-> 소멸자 등을 수정하며 Reliability를 A로 만들 수 있었습니다.

5. OOAD를 수행한 소감

이름	소감
노성준	평소 소프트웨어 아키텍처, 객체지향설계, 클린코드에 관심이 많았지만, OOAD 과정처럼 분석, 디자인 단계에서부터 여러 diagram들을 그리면서 소프트웨어를 개발해본 적은 처음이었습니다. 이번 팀프로젝트를 통해 초기 설계과정에서 많은 리소스를 들이며 고생하니, 이후 구현과정이 비교적 수월해지는 것을 체감할 수 있었습니다. 앞으로 다른 프로젝트 진행 시에 바로 구현에 들어가는 것이 아니라, 설계단계에서 좀 더 시간을 쓰는 습관을 가지는 것도 좋을 것 같다는 생각이 듭니다.
권민혁	코드를 짤 때, 항상 객체지향적으로 설계해야지 다짐하면서도, 마감기한 내에 기능을 구현하기에만 급급하여 지키기 쉽지 않았습니다. 이번 기회에 OOAD 과정을 통해 객체지향적으로 설계해보면서 항상 숙제로 남겨왔던 객체지향적 코드를 직접 경험할 수 있었습니다. 설계과정에서 수많은 다이어그램을 그리면서 '이렇게까지 꼼꼼하게 해야할까' 생각하며 배보다 배꼽이 큰 것 같은 느낌을 받았습니다. 그런데 코드를 구현하면서 객체지향에서 꼼꼼한 설계의 장점을 몸소 느낄 수 있었습니다.
이서준	OOAD 단계를 따라 개발해 본 것은 처음이었습니다. 구현 전에 문서를 작성하면서 팀 내 컨벤션이 통일되고, 협업 효율도 높아졌습니다. 다만, OOAD를 한 번만 수행하다 보니 설계에서 놓치는 부분이 많고 시간이 많이 들었던 점은 아쉬웠습니다. 나중에 다른 프로젝트에서는 전체 과정보다 핵심 단계만 선택적으로 적용하는 것도 좋은 방법이라 생각합니다.

감사합니다.