

Distribute Vending Machine (DVM)

OOPT 2050&60 2nd Cycle Presentation

Team 2

202111308 손승우

202111340 이수민

202111240 강찬욱

빌드 및 실행 방법 소개

- 사전준비
 - Cmake (3.15 버전 이상)
 - C++ 컴파일러 (MinGW)
 - CLion (IDE로 실행 시)

빌드 및 실행 방법 소개

- CLI로 빌드/실행 방법

1. 새 빌드 디렉터리 생성 후 이동

mkdir build

cd build

2. MinGW용 CMake 빌드 생성

cmake .. -G "MinGW Makefiles"

3. 빌드 실행

mingw32-make # 또는 make

./main.exe #main 실행

./MyTests.exe #Test 실행

빌드 및 실행 방법 소개

- Clion (IDE) 으로 빌드/실행방법
 1. 프로젝트 열기
File -> Open -> 프로젝트 루트의 CMakeLists.txt 선택
 2. Cmake 설정 확인
우측 하단의 Cmake 창에서 로드된 Cmake 설정 확인
 3. 빌드
메뉴에서 Build -> Build Project
빌드 로그는 하단 Build 탭에 출력
 4. 실행 설정
상단 실행 구성 드롭다운에서 생성된 실행 타겟 선택
Run -> Run 'main'

Act. 2051. Implement Class & Method Def.

Class 1. Display

Type	Class
Name	Display
Purpose	사용자에게 정보를 시각적으로 출력하고 사용자의 입력을 처리한다.
Overview (Class)	음료 목록, 재고 상태, 결제 진행 상황, 오류 메시지 등을 사용자에게 보여주는 기능을 담당하며 사용자 입력을 받은 기능을 담당한다.
Cross Reference	Functions: Use Case: 1. 음료 선택, 2. 선결제 물품 수령, 3. 음료 구매, 4. 선결제

Act. 2051. Implement Class & Method Def.

Class 1. Display

Type	Method
Name	printMenu
Purpose	자판기의 화면에 음료 메뉴를 출력한다.
Cross Reference	Use Case: 1. 음료 선택
Input (Method)	없음
Output (Method)	없음 (UI 에 출력)
Abstract operation (Method)	현재 재고 정보를 바탕으로 음료 목록을 출력
Exceptional Courses of Events	없음

Act. 2051. Implement Class & Method Def.

Class 1. Display

Type	Method
Name	selectMenu
Purpose	사용자의 음료 선택을 Display 로부터 전달 받는다.
Cross Reference	Use Case: 1. 음료 선택
Input (Method)	ItemType: String
Output (Method)	없음 (UI 에 출력)
Abstract operation (Method)	선택된 ItemType을 Inventory.selectItem() 으로 전달한다.
Exceptional Courses of Events	1. 입력 형식이 잘못되었을 경우 재입력 2. 존재하지 않는 음료 코드일 경우 재입력 3. 유효하지 않은 수량일 경우 재입력 4. 사용자가 제품을 선택했을 때, 선택한 자판기 또는 다른 자판기에 해당 제품의 재고가 없을 경우, 사용자에게 품절을 안내함

Act. 2051. Implement Class & Method Def.

Class 1. Display

Type	Method
Name	inputCertCode
Purpose	사용자가 인증 코드를 입력할 수 있도록 입력 화면을 제공한다.
Cross Reference	Use Case: 2. 선결제 물품 수령
Input (Method)	code: String
Output (Method)	없음 (UI 에 출력)
Abstract operation (Method)	입력 필드 표시 및 입력 값 전달 처리
Exceptional Courses of Events	1. 존재하지 않는 코드 입력 시 안내 후 초기 화면으로 돌아감 2. 코드 형식(알파벳(대소문자 모두 가능) 혹은 숫자로 구성된 8자리 문자열)이 잘못되었을 경우 안내 후 초기 화면으로 돌아감 3. 인증 실패 시, 실패 원인에 따라 다른 메시지 출력

Act. 2051. Implement Class & Method Def.

Class 1. Display

<pre>std::pair<int, int> PrepaymentHandler::PrePaymentCheck(std::string code) { std::optional<CodeInfo> info = certificationCodeRepository->findByCode(code); if(info != std::nullopt) { EraseCode(code); return {info->itemCode, info->itemNum}; } return {-1, -1}; } void PrepaymentHandler::EraseCode(std::string code) { certificationCodeRepository->deleteByCode(code); } /** * @brief 코드가 유효한지 검사하고, 유효하면 인증코드 리포지토리에서 삭제한다. */</pre>	<pre>172 174 ↗ » 173 175 □ 174 176 175 177 176 178 177 179 178 180 » 179 181 180 182 181 183 182 184 □ 183 185 184 186 185 187 ↗ 186 188 187 189</pre>	<pre>std::pair<int, int> PrepaymentHandler::PrePaymentCheck(std::string code) { if (!IsValidCode(code)) { return { -1, -1}; } std::optional<CodeInfo> info = certificationCodeRepository->findByCode(code); if(info != std::nullopt) { EraseCode(code); return { info->itemCode, info->itemNum}; } return { -1, -1}; } void PrepaymentHandler::EraseCode(std::string code) { certificationCodeRepository->deleteByCode(code); }</pre>
---	--	--

<pre>bool PrepaymentHandler::IsValidCode(std::string code){ const std::string charset = "abcdefghijklmnopqrstuvwxyz" "ABCDEFGHIJKLMNOPQRSTUVWXYZ" "0123456789"; for (char c : code) { if (charset.find(c) == std::string::npos) { return false; // charset에 없는 문자가 있으면 false 반환 } } return true; // 모든 문자가 charset에 포함되면 true 반환 }</pre>	<pre>189 194 ↗ » 190 195 □ 191 196 192 197 193 198 194 199 195 200 196 201 197 202 198 203 199 204 200 205 201 206 202 207</pre>	<pre>bool PrepaymentHandler::IsValidCode(std::string code){ if (code.size() != 8) { return false; } const std::string charset = "abcdefghijklmnopqrstuvwxyz" "ABCDEFGHIJKLMNOPQRSTUVWXYZ" "0123456789"; for (char c : code) { if (charset.find(c) == std::string::npos) { return false; // charset에 없는 문자가 있으면 false 반환 } } }</pre>
---	--	---

Act. 2051. Implement Class & Method Def.

Class 1. Display

```
void OutputFormatter::showCertificateResult(bool result){  
    std::cout << "\n[인증 실패] 유효하지 않은 코드입니다.\n";  
    std::cout << "-----\n";  
}
```

>> 99 99 ☐ 
100 100
101 101
>> 102 102

```
void OutputFormatter::showCertificateResult(int result){  
    if (result == 1) {  
        std::cout << "\n[인증 실패] 인증 코드의 형식이 잘못되었습니다.\n";  
        std::cout << "-----\n";  
    }  
    else if (result == 2) {  
        std::cout << "\n[인증 실패] 존재하지 않는 코드입니다.\n";  
        std::cout << "-----\n";  
    }  
}
```

Act. 2051. Implement Class & Method Def.

Class 1. Display

Type	Method
Name	paymentMenu
Purpose	사용자로부터 음료 결제 요청을 받고 결제 프로세스를 시작한다.
Cross Reference	Use Case: 3. 음료 결제
Input (Method)	accept: String
Output (Method)	없음 (UI 에 출력)
Abstract operation (Method)	음료 결제 요청에 대한 화면 출력 및 처리
Exceptional Courses of Events	1. 유효하지 않은 입력 값 전달 시 재입력

Act. 2051. Implement Class & Method Def.

Class 1. Display

Type	Method
Name	paymentRequest
Purpose	사용자로부터 카드 정보를 입력받는다.
Cross Reference	Use Case: 3. 음료 결제, 4. 선결제
Input (Method)	card: String
Output (Method)	없음 (UI 에 출력)
Abstract operation (Method)	카드 정보 입력 화면 출력 및 처리
Exceptional Courses of Events	1. 카드 정보가 포맷(총 16자리 4자리씩 하이픈으로 구분된 문자열)에 맞지 않은 경우 오류 처리 후 초기 화면으로 돌아감 ex) 1234-1234-1234-1234 2. 해당 카드 정보가 없을 경우 오류 처리 후 초기 화면으로 돌아감

Act. 2051. Implement Class & Method Def.

Class 1. Display

```
#include "InputTimeoutHandler.h"
#include <iostream>
#include <chrono>

InputTimeoutHandler::InputTimeoutHandler() : stopFlag(false) {
    inputThread = std::thread(&InputTimeoutHandler::inputLoop, this);
}

InputTimeoutHandler::~InputTimeoutHandler() {
    stopFlag = true;
    if (inputThread.joinable()) inputThread.join();
}

std::optional<std::string> InputTimeoutHandler::getInput(int timeout_sec) {
    std::unique_lock<std::mutex> lock(mtx);

    if (cv.wait_for(lock, std::chrono::seconds(timeout_sec), [&]() { return !inputQueue.empty(); })) {
        std::string result = inputQueue.front();
        inputQueue.pop();
        return result;
    } else {
        return std::nullopt;
    }
}

void InputTimeoutHandler::inputLoop() {
    while (!stopFlag) {
        std::string line;
        if (!std::getline(std::cin, line)) {
            stopFlag = true;
            break;
        }

        {
            std::lock_guard<std::mutex> lock(mtx);
            inputQueue.push(line);
        }
        cv.notify_one();
    }
}
```

```
1  #pragma once
2
3  #include <string>
4  #include <queue>
5  #include <mutex>
6  #include <condition_variable>
7  #include <thread>
8  #include <optional>
9  #include <atomic>
10
11 class InputTimeoutHandler {
12 public:
13     InputTimeoutHandler();
14     ~InputTimeoutHandler();
15
16     // 타임아웃 기반 입력 요청 (초 단위)
17     std::optional<std::string> getInput(int timeout_sec);
18
19 private:
20     void inputLoop();
21
22     std::queue<std::string> inputQueue;
23     std::mutex mtx;
24     std::condition_variable cv;
25     std::thread inputThread;
26     std::atomic<bool> stopFlag;
27 };
```

- input 입력 후 일정 초가 지나면 input을 끝내도록 구성
다만, getline 특성상 thread를 분리하여도 input 관련기능이 겹치는 이슈가 발생하여 해당 기능 제외 필요

Act. 2051. Implement Class & Method Def.

Class 1. Display

Type	Method
Name	displayState
Purpose	input 에 따른 화면을 출력한다.
Cross Reference	Use Case: 2. 선결제 물품 수령, 3. 음료 결제, 4. 선결제
Input (Method)	code: String
Output (Method)	없음 (UI 에 출력)
Abstract operation (Method)	code 내용에 따른 화면 출력
Exceptional Courses of Events	1. 유효하지 않은 입력 시 오류 처리

Act. 2051. Implement Class & Method Def.

Class 2. Inventory

Type	Class
Name	Inventory
Purpose	자판기의 음료 재고 상태를 관리한다.
Overview	음료의 재고 수량을 저장하고, 추가, 차감, 조회 등의 재고 관련 작업을 수행하는 기능을 담당한다.
Cross Reference	Functions: Use Case: 1. 음료 선택, 2. 선결제 물품 수령, 3. 음료 구매, 4. 선결제, 5. 음료 추가

Act. 2051. Implement Class & Method Def.

Class 2. Inventory

Type	Method
Name	isAvailable
Purpose	특정 음료 코드의 재고가 충분한지 확인한다.
Cross Reference	Use Case: 1. 음료 선택, 3. 음료 결제, 4. 선결제
Input (Method)	code: int, qty: int
Output (Method)	bool
Abstract operation (Method)	음료 코드 유효성 확인 후 재고 수량 비교
Exceptional Courses of Events	없음

Act. 2051. Implement Class & Method Def.

Class 2. Inventory

Type	Method
Name	decreaseStock
Purpose	결제 성공 시 재고 수량을 차감하고 변경 사항을 저장한다.
Cross Reference	Use Case: 3. 음료 결제, 4. 선결제
Input (Method)	code: int, qty: int
Output (Method)	없음
Abstract operation (Method)	재고 가능 여부 확인 후 차감 및 DB 파일 반영
Exceptional Courses of Events	재고 부족 시 아무 작업도 수행하지 않음

Act. 2051. Implement Class & Method Def.

Class 3. Prepayment Handler

Type	Class
Name	PrepaymentHandler
Purpose	선결제 처리 및 다른 DVM과의 재고/선결제 요청 통신을 담당한다.
Overview (Class)	재고 부족 시 다른 자판기(DVM)들의 재고를 조회하고, 선결제 요청을 전송 및 응답을 처리하며, 인증 코드의 생성, 저장, 검증, 삭제까지의 전체 선결제 흐름을 관리한다. 인증 코드를 기반으로 선결제 음료 수량을 가능하게 하는 핵심 로직을 포함한다.
Cross Reference	Functions: Use Case: 2. 선결제 물품 수령, 4. 선결제 처리

Act. 2051. Implement Class & Method Def.

Class 3. Prepayment Handler

Type	Method
Name	findAvailableDVM
Purpose	가장 가까운 DVM 중 재고가 충분한 곳을 탐색한다.
Cross Reference	Use Case: 4. 선결제
Input (Method)	beverage: Beverage*, qty: int
Output (Method)	ResponseStock*
Abstract operation (Method)	브로드캐스트 요청 후 응답 중에서 조건을 만족하는 DVM 탐색
Exceptional Courses of Events	유효한 응답이 없을 경우 nullptr 반환

Act. 2051. Implement Class & Method Def.

Class 3. Prepayment Handler

Type	Method
Name	prepaymentRequest
Purpose	타 DVM에 선결제 요청을 전송하고 결과를 수신한다.
Cross Reference	Use Case: 4. 선결제
Input (Method)	beverage: Beverage*, qty: int, dstIp: string, dstId: string
Output (Method)	pair<bool, string> (성공여부, 인증코드)
Abstract operation (Method)	인증 코드 생성 후 dstip에 요청 메시지를 전송한다. 이후 응답 파싱하여 결과를 반환한다.
Exceptional Courses of Events	JSON 파싱 실패 혹은 거절 응답 시 실패 처리

Act. 2051. Implement Class & Method Def.

Class 3. Prepayment Handler

Type	Method
Name	PrePaymentCheck
Purpose	입력된 인증 코드에 해당하는 음료 정보를 조회한다.
Cross Reference	Use Case: 2. 선결제 물품 수령
Input (Method)	code: string
Output (Method)	pair<int, int> (itemCode, itemNum)
Abstract operation (Method)	인증 코드 유효성 확인 후 정보 조회 및 코드 삭제
Exceptional Courses of Events	인증 코드가 존재하지 않으면 {-1, -1} 반환

Act. 2051. Implement Class & Method Def.

Class 3. Prepayment Handler

Type	Method
Name	handlePrepaymentRequest
Purpose	타 DVM의 선결제 요청을 처리한다.
Cross Reference	없음
Input (Method)	certCode: string, itemCode: int, qty: int
Output (Method)	bool
Abstract operation (Method)	재고 확인 후 전달 받은 인증 코드 저장하고 재고를 차감한다.
Exceptional Courses of Events	재고 부족 시 false 반환

Act. 2051. Implement Class & Method Def.

Class 4. Purchase Handler

Type	Class
Name	PurchaseHandler
Purpose	사용자의 카드 정보를 기반으로 결제를 수행하고, 결제 성공 시 재고를 차감하는 역할을 한다.
Overview (Class)	결제 시스템과 연동하여 사용자의 결제 요청을 처리하고, 결제가 성공한 경우 해당 음료의 재고를 Inventory에서 차감하는 로직을 담당한다. 음료 구매의 핵심 결제 흐름을 제어하는 기능 클래스이다.
Cross Reference	Functions: Use Case: 3. 음료 구매, 4. 선결제

Act. 2051. Implement Class & Method Def.

Class 4. Purchase Handler

Type	Method
Name	purchase
Purpose	결제를 진행한다.
Cross Reference	Use Case: 3. 음료 결제
Input (Method)	cardNumber: string, beverage: Beverage*, qty: int
Output (Method)	pair<bool, string> (결제 성공 여부, 상태 메시지)
Abstract operation (Method)	카드 정보와 가격을 결제 시스템에 전달하고 결제 성공 시 재고를 차감한다.
Exceptional Courses of Events	1. 결제 실패 시 재고를 차감하지 않고 실패 메시지 반환 2. 연결된 다른 DVM에서 재고 없음 또는 응답 없음 시 각 상황에 맞는 오류메시지 반환

Act. 2051. Implement Class & Method Def.

Class 4. Purchase Handler

```
PurchaseHandler::PurchaseHandler(PaymentSystem *p, Inventory *i):paymentSystem(p), inventory(i) {  
}  
  
std::pair<bool, std::string> PurchaseHandler::purchase(std::string cardNumber, Beverage* beverage, int qty) {  
    std::pair<bool, std::string> result = paymentSystem->connectSystem(cardNumber, beverage->getPrice()*qty);  
    if (!result.first) {  
        return result;  
    }  
    inventory->decreaseStock(beverage->getCode(), qty);  
    return result;  
}
```

```
11 11  
12 12  
13 13  
14 14  
15 15  
16 16  
17 17  
18 18  
19 19  
20 20  
21 21  
22 22  
23 23  
24 24  
25 25  
26 26  
27 27  
28 28  
29 29  
30 30  
31 31  
32 32  
  
PurchaseHandler::PurchaseHandler(PaymentSystem *p, Inventory *i):paymentSystem(p), inventory(i) {  
}  
  
std::pair<bool, std::string> PurchaseHandler::purchase(std::string cardNumber, Beverage* beverage, int qty) {  
    if (!isValidCardFormat(cardNumber)) {  
        return { false, y."카드 형식이 잘못되었습니다."};  
    }  
  
    std::pair<bool, std::string> result = paymentSystem->connectSystem( cardNumber, price: beverage->getPrice()*qty);  
    if (!result.first) {  
        return result;  
    }  
    inventory->decreaseStock(beverage->getCode(), qty);  
    return result;  
}  
  
bool PurchaseHandler::isValidCardFormat(const std::string& cardNumber) {  
    // 허용 포맷: 16자리 숫자 or "1234-5678-9012-3456" 형식  
    std::regex simpleFormat( p: "^\\d{16}$");  
    std::regex hyphenFormat( p: "\\d{4}-\\d{4}-\\d{4}-\\d{4}$");  
    return std::regex_match(cardNumber, simpleFormat) || std::regex_match(cardNumber, hyphenFormat);  
}
```

Act. 2051. Implement Class & Method Def.

Class 5. BroadCast Impl.

Type	Class
Name	BroadCastImpl
Purpose	DVM간 네트워크 메시지를 전송하고 응답을 수신하는 통신 기능을 담당한다.
Overview (Class)	TCP 소켓을 이용해 특정 IP와 포트로 메시지를 전송하고, 응답을 수신하는 기능을 제공한다. 선결제 시 다른 DVM의 재고 상태를 조회하거나, 선결제 요청 메시지를 전달할 때 사용된다.
Cross Reference	Functions: Use Case: 4. 선결제

Act. 2051. Implement Class & Method Def.

Class 5. BroadCast Impl.

Type	Method
Name	broadCast
Purpose	지정된 IP와 포트로 TCP 메시지를 전송하고 응답을 수신한다.
Cross Reference	Use Case: 4. 선결제
Input (Method)	ip: string, port: int, message: string
Output (Method)	string
Abstract operation (Method)	Winsock을 초기화하고 소켓 생성 및 연결을 진행한다. 이후 메시지를 전송하고 응답을 수신 후 string으로 반환한다.
Exceptional Courses of Events	Winsock 초기화 실패, 소켓 생성 실패, 연결 실패, 응답 수신 실패 시 빈 문자열 반환

Act. 2051. Implement Class & Method Def.

Class 6. Payment System Impl.

Type	Class
Name	PaymentSystemImpl
Purpose	카드 번호와 결제 금액을 기반으로 결제 승인 여부를 판단하고, 잔액을 갱신한다.
Overview (Class)	DB를 이용한 카드 시스템에 접근하여 결제 관련 시스템을 처리하는 클래스
Cross Reference	Functions: Use Case: 3. 음료 구매, 4. 선결제

Act. 2051. Implement Class & Method Def.

Class 6. Payment System Impl.

Type	Method
Name	connectSystem
Purpose	카드시스템 연결 (구현은 DB 읽기)
Cross Reference	Use Case: 3. 음료 결제
Input (Method)	cardNumber: string, price: int
Output (Method)	pair<bool, string> (결제 성공 여부, 상태 메시지)
Abstract operation (Method)	결제 시스템 파일을 읽어 카드 정보를 조회하고 금액을 차감한다. 이후 파일을 갱신하고 결과를 반환한다.
Exceptional Courses of Events	1. 파일 열기 실패 2. 카드 정보 없음 3. 잔액 부족

Act. 2051. Implement Class & Method Def.

Class 7. DVM Server

Type	Class
Name	DVMServer
Purpose	외부 DVM으로부터 들어오는 재고 조회 및 선결제 요청을 처리하는 TCP 서버 역할을 수행한다.
Overview (Class)	TCP 기반 서버로 동작하며, 다른 DVM에서 송신한 메시지를 수신하고, 자판기의 재고 정보 또는 선결제 가능 여부에 따라 응답 메시지를 전송한다. 클라이언트 요청을 수신한 후 Inventory, PrepaymentHandler와 협력하여 실제 로직을 수행하고 응답을 구성한다.
Cross Reference	없음

Act. 2051. Implement Class & Method Def.

Class 7. DVM Server

Type	Method
Name	run
Purpose	자판기 서버를 시작하고 클라이언트의 연결을 대기한다.
Cross Reference	없음
Input (Method)	없음
Output (Method)	없음
Abstract operation (Method)	소켓을 열고 클라이언트 연결 수락 및 처리를 반복한다.
Exceptional Courses of Events	1. 소켓 바인드/리스닝/연결 실패 시 에러 메시지 출력

Act. 2051. Implement Class & Method Def.

Class 7. DVM Server

Type	Method
Name	handleClient
Purpose	클라이언트로부터 받은 요청을 파싱하고 처리한 뒤 응답을 전송한다.
Cross Reference	없음
Input (Method)	clientFd: int (클라이언트 소켓 FD)
Output (Method)	없음
Abstract operation (Method)	요청 수신 시 DVM 거리 구분 (local/Remote) , 응답속도 및 시도한 우선순위에 따른 응답을 전송한다.
Exceptional Courses of Events	1. 요청 JSON 파싱 오류 2. 소켓 오류 발생 3. 다른 DVM에 응답 전송 실패 시 code 저장과 음료 수량 감소에 대한 롤백 처리

Act. 2051. Implement Class & Method Def.

Class 7. DVM Server

```
response = makeResponseRepay(jsonReq),
}

// 응답 전송
send(clientFd, response.c_str(), response.size(), 0);
}

// 연결 종료
close(clientFd);

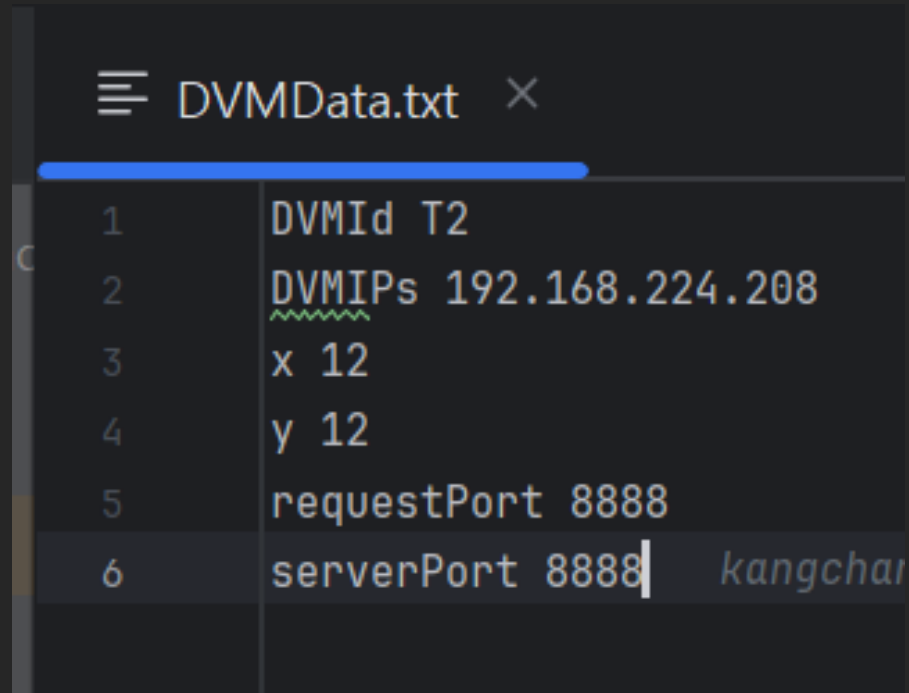
tring DVMServer::makeResponseStock(json jsonReq) {
    string itemCode = jsonReq["msg_content"]["item_code"];
    // 응답 JSON 구성
    json resp;
    resp["msg_type"] = "resp_stock";
    resp["msg_content"] = DVMServer::makeResponseStock(itemCode); // 응답 내용 구성
}

31 31 response = makeResponseRepay(jsonReq),
32 32 }
>> 33 33 else {
>> 34 34     response = R("{\"error\": \"Unknown msg_type\"}");
35 35 }
36 36
37 37 // 응답 전송 (전송 실패 시 로그 출력)
38 38 if (send(clientFd, buf: response.c_str(), len: response.size(), flags: 0) == -1) {
39 39     perror("ErrMsg: "[DVMServer] 응답 전송 실패");
40 40
41 41     string itemCode = jsonReq["msg_content"]["item_code"];
42 42     int qty = jsonReq["msg_content"]["item_num"];
43 43     string certCode = jsonReq["msg_content"]["cert_code"];
44 44     prepaymentHandler->rollBackPrepaymentRequest(certCode, itemCode, stoi(itemCode), qty);
45 45 }
46 46 }
```

```
bool PrepaymentHandler::rollBackPrepaymentRequest(string certCode, int itemCode, int qty) {
    //인증 코드 삭제
    certificationCodeRepository->deleteByCode(certCode);
    //재고 증가
    inventory->increaseStock(itemCode, qty);
    return true;
}
```

Act. 2051. Implement Class & Method Def.

Class 7. DVM Server



```
≡ DVMDData.txt ×
1 DVMId T2
2 DVMIPs 192.168.224.208
3 x 12
4 y 12
5 requestPort 8888
6 serverPort 8888 kangchar
```

- 다른 DVM 연결 시 다른 DVM 의 IP 주소를 /src/DVM/database/DVMDData.txt 파일에 DVMIPs attribute 에 추가하면 연결할 수 있다. (공백을 기준으로 구분)

Act. 2052. Implements Windows CLI

```
=====
      2팀 DVM System
=====
1. 음료 구매
2. 선결제 음료 수령
=====
> 메뉴를 선택해주세요:|
```

```
> 메뉴를 선택해주세요:1

===== 음료 선택 메뉴 =====
코드 | 이름           | 가격   | 수량
-----
01  | 콜라           | 1500 원 | 4 개
02  | 사이다         | 1400 원 | 5 개
03  | 녹차           | 1300 원 | 1 개
04  | 홍차           | 1300 원 | 2 개
05  | 밀크티        | 1800 원 | 6 개
06  | 탄산수         | 1200 원 | 7 개
07  | 보리차         | 1100 원 | 7 개
08  | 캔커피         | 1600 원 | 0 개
09  | 물             | 1000 원 | 0 개
10  | 에너지드링크  | 2000 원 | 0 개
11  | 유자차         | 1400 원 | 0 개
12  | 식혜          | 1500 원 | 0 개
13  | 아이스티      | 1300 원 | 0 개
14  | 딸기주스       | 1700 원 | 0 개
15  | 오렌지주스    | 1700 원 | 0 개
16  | 포도주스       | 1700 원 | 0 개
17  | 이온음료       | 1500 원 | 0 개
18  | 아메리카노    | 2000 원 | 0 개
19  | 핫초코        | 1800 원 | 0 개
20  | 카페라떼      | 2000 원 | 0 개
=====

> 음료 코드와 수량을 입력하세요 (예: 01 2):|
```

```
> 음료 코드와 수량을 입력하세요 (예: 01 2):01 1
```

[선택 완료]:
콜라 1개가 선택되었습니다.

최종 선택 목록:
- 콜라 1개

총 금액: 1500원

```
> 결제를 진행하시겠습니까? (예: Y or N):
```

Act. 2052. Implements Windows CLI

> 결제를 진행하시겠습니까? (예: Y or N):Y

결제 단계로 진행합니다...

```
=====
      결제 진행
=====
```

카드를 삽입해주세요...

11111

카드 결제 중입니다...

[결제 승인] 결제가 성공적으로 완료되었습니다.

음료를 준비 중입니다...

[음료 제공] 콜라 1개가 제공되었습니다.

```
=====
감사합니다. 또 이용해주세요!
=====
```

> 음료 코드와 수량을 입력하세요 (예: 01 2):11 1

[재고 부족] 유자차는(은) 현재 0개 남아 있습니다.

[재고 부족] '유자차'는 현재 0개 남아 있습니다.

[재고 확인 중...] 다른 DVM들에 정보 요청 중입니다...

[재고 확인 완료] '유자차 1개'는 아래 DVM에서 수령 가능합니다.
위치: x좌표 12 y좌표 12 (T2)

총 금액: 1400원

> 선결제를 진행하시겠습니까? (예: Y or N):Y

Act. 2052. Implements Windows CLI

```
=====
```

결제 진행

```
=====
```

카드를 삽입해주세요...

11111

카드 결제 중입니다...

[결제 승인] 결제가 성공적으로 완료되었습니다.

발급된 인증코드: AnpVfiiW

수령 위치: x좌표 12 y좌표 12 (T2)

※ 해당 위치에서 인증코드를 입력하면 음료를 받을 수 있습니다.

```
=====
```

감사합니다. 또 이용해주세요!

```
=====
```

```
=====
```

선결제 물품 수령

```
=====
```

인증 코드를 입력해주세요 :

AnpVfiiW

초기 화면으로 돌아갑니다.

```
-----
```

음료를 준비 중입니다...

[음료 제공] 유자차 1개가 제공되었습니다.

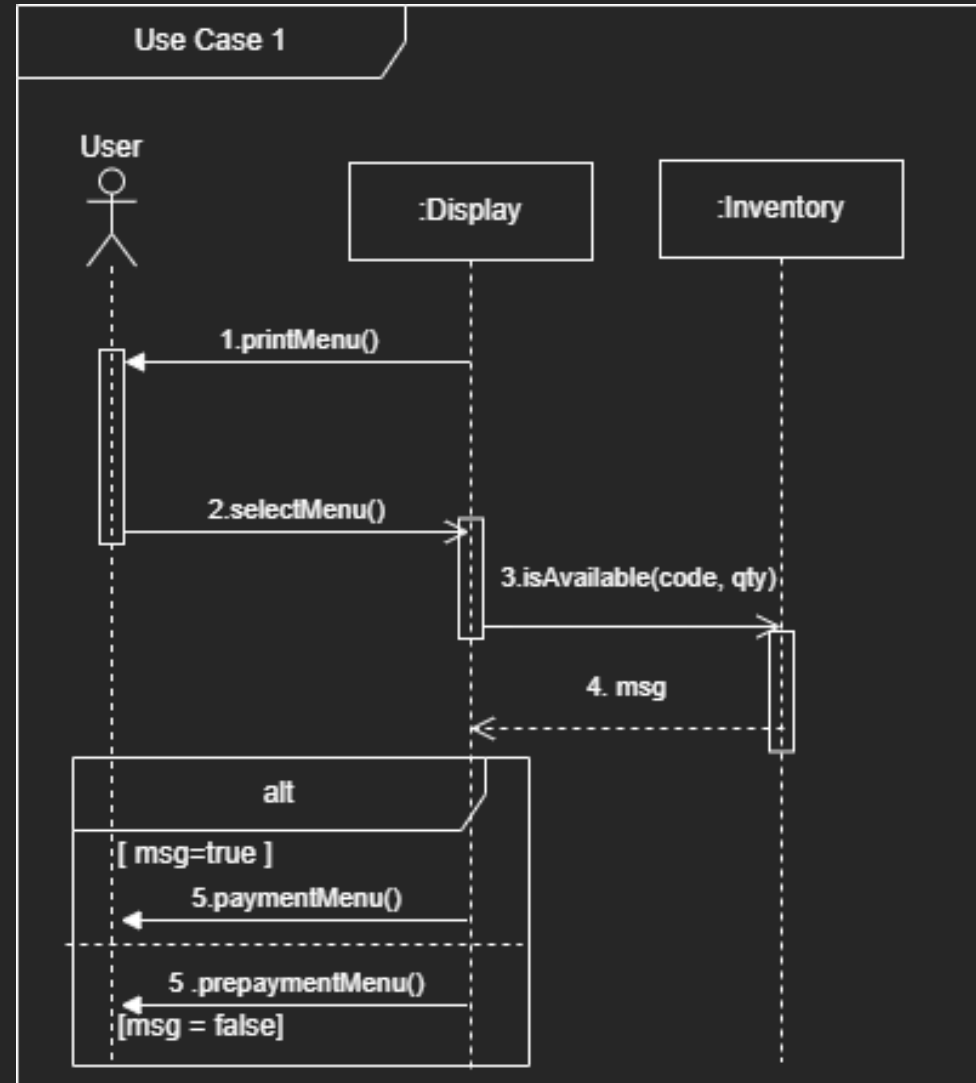
```
=====
```

감사합니다. 또 이용해주세요!

```
=====
```

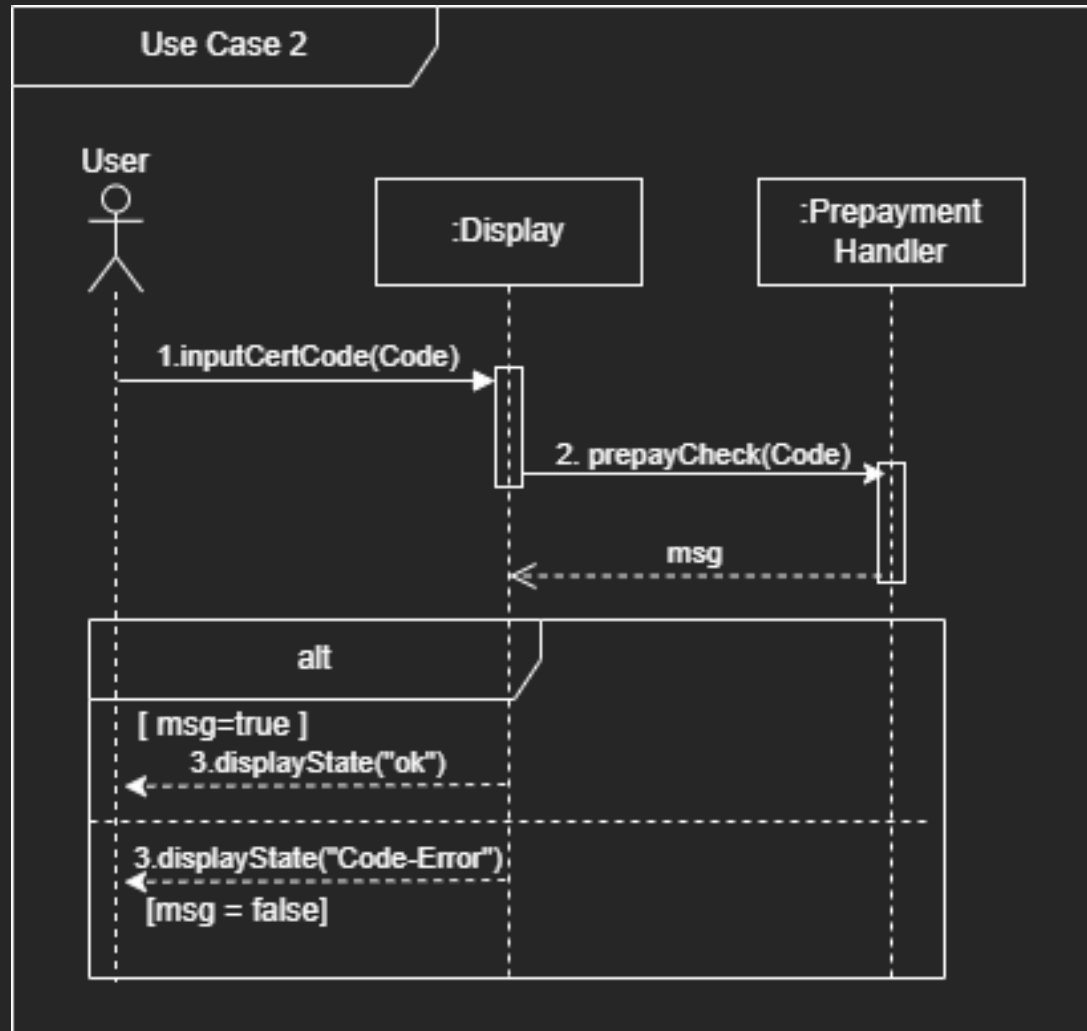
Act. 2053. Implement Reports

1. Sequence Diagram



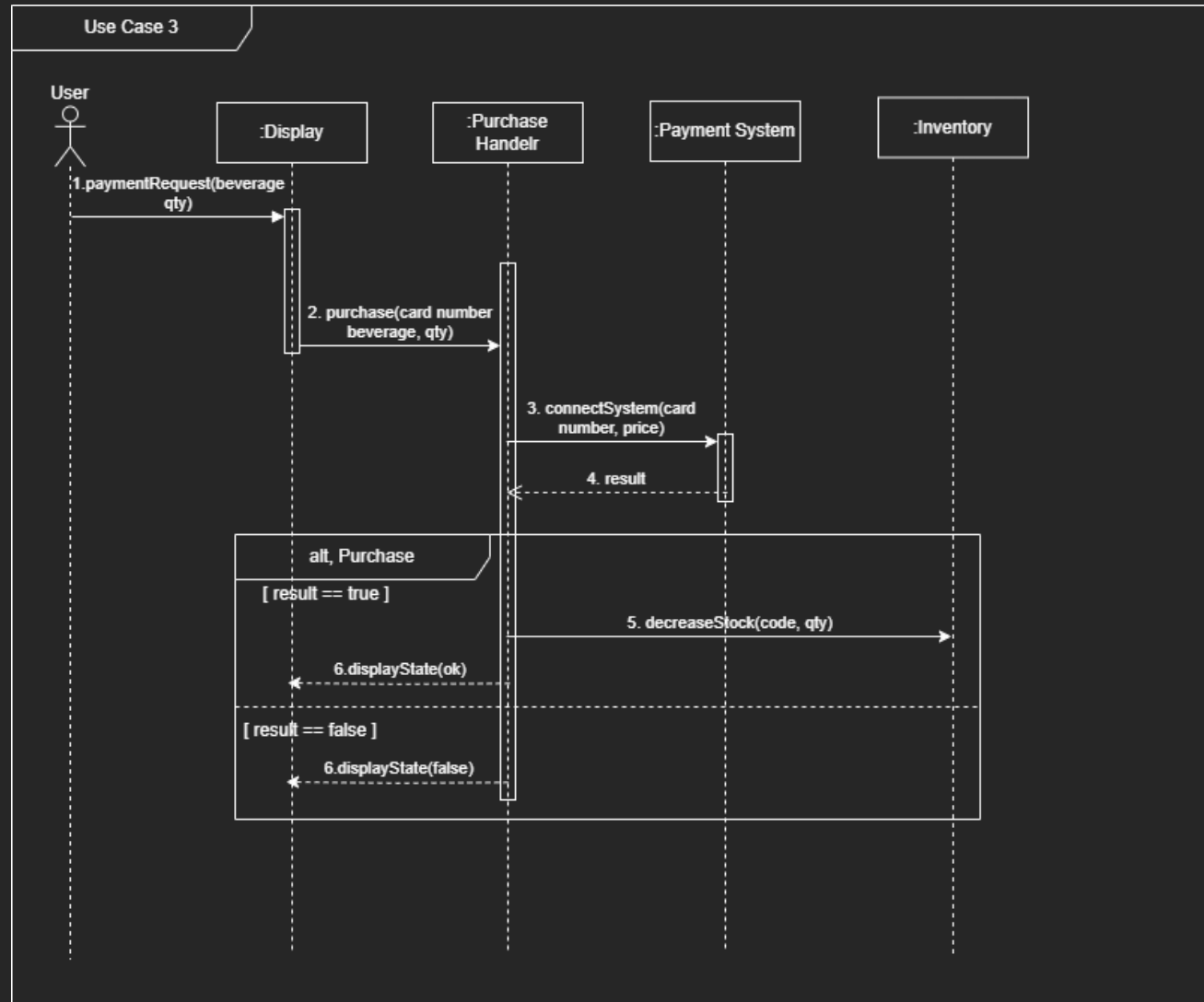
Act. 2053. Implement Reports

1. Sequence Diagram



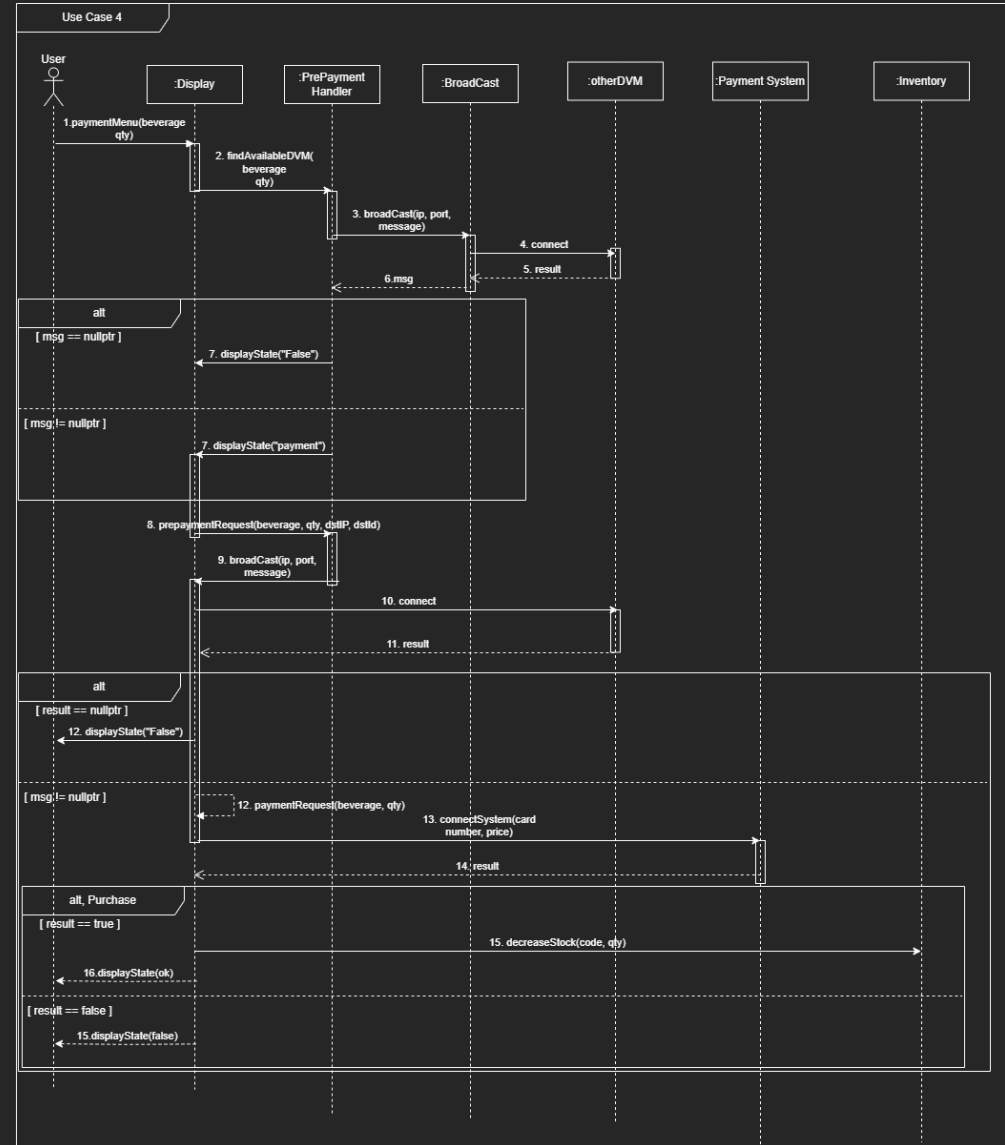
Act. 2053. Implement Reports

1. Sequence Diagram



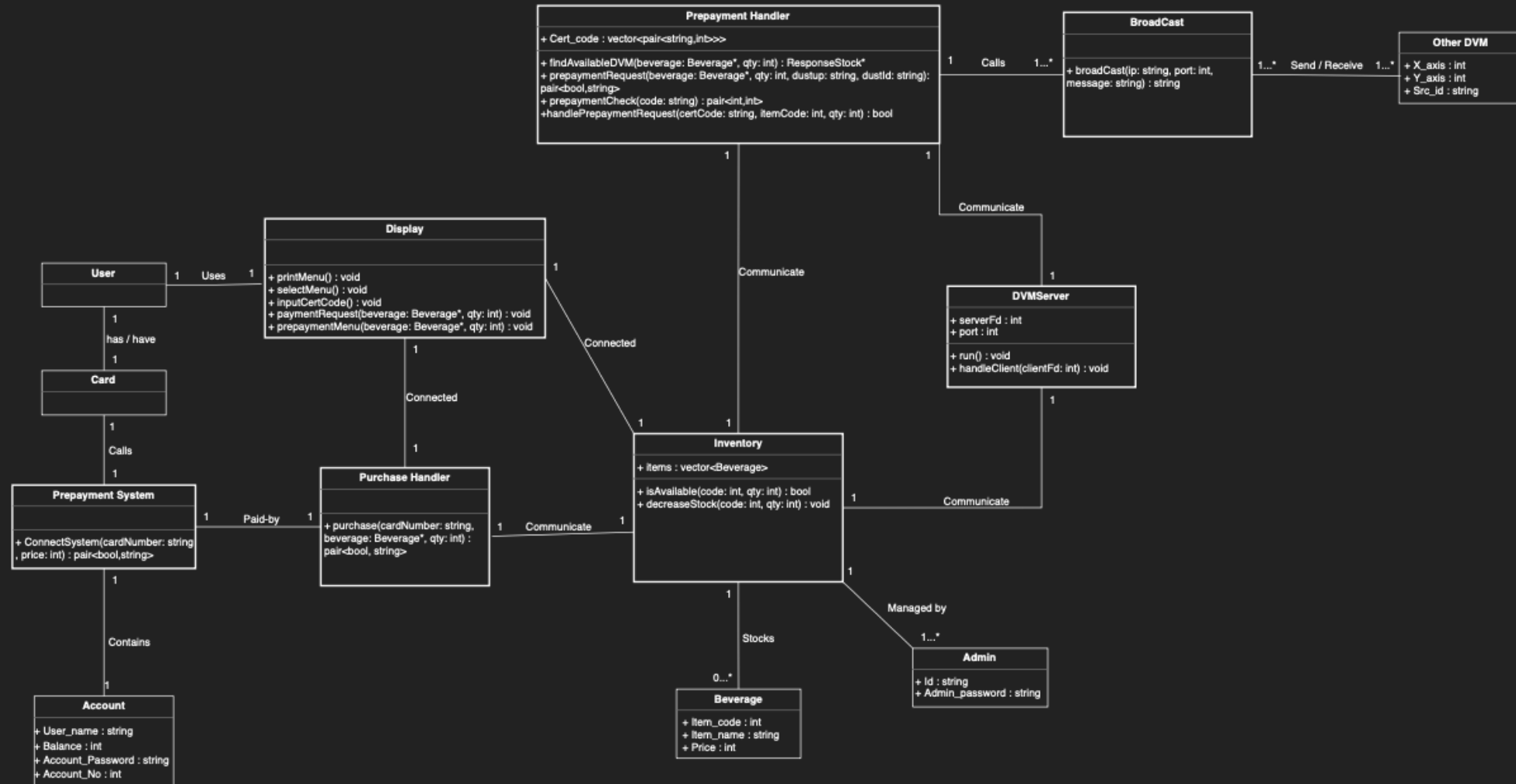
Act. 2053. Implement Reports

1. Sequence Diagram



Act. 2053. Implement Reports

2. Class Diagram



Act. 2061. Unit Testing

- Unit 테스트 대상
 - Application Layer에 존재하는 도메인 로직과 비즈니스 로직
 - 화면 출력 및 외부 시스템 연결은 제외
- Unit 테스트 방법
 - Google Test 프레임워크를 활용하여 주요 로직에 대한 Unit 테스트 작성
 - 실제 의존 객체 대신 Fake 객체 (FakePaymentSystem, FakeBroadcast 등) 를 사용하여 독립성 확보
 - EXPECT_EQ, EXPECT_TRUE 등을 통해 정상 동작 및 예외 상황 검증

Act. 2061. Unit Testing(gtest)

Inventory Unit Test

Num	Test Case Name/ Related Function	Description	Condition	Expect	Pass/Fail
UT-1.1	LoadInventory_Success/ getBeverage()	BeverageRepository에서 음료 7종을 로드하여 Inventory에 정상적으로 저장되는지 확인	음료 7종 을 저장하는 FakeRepository 사용	itemCount() == 7, 각 이름 일치	Pass
UT-1.2	GetBeverage_InvalidCode_ReturnsNullptr/ getBeverage()	유효하지 않은 코드 입력 시 nullptr 반환 확인	code == 0 or 9999	nullptr 반환	Pass
UT-1.3	IsAvailable_WorksCorrectly/ isAvaliable()	충분/부족 재고 및 유효하지 않은 코드에 대해 isAvailable 동작 확인	qty == 5 (충분), qty == 15 (부족), code==99, code==0 (잘못된 코드)	true / false 정확히 반환	Pass
UT-1.4	IsValidCode_WorksCorrectly/ isValidCode()	inventory에 존재하는 음료코드인 경우만 true 반환되는지 확인	code == 100 code == -1, code == 0 code == 1, code ==2, code ==3	유효한 코드 외 false, 유효한 코드 시 true	Pass

Act. 2061. Unit Testing (gtest)

Inventory Unit Test

Num	Test Case Name/ Related Function	Description	Condition	Expect	Pass/Fail
UT-1.5	DecreaseStock_Updates_Repository_And_Inventory/ decreaseStock()	재고 차감 시 Inventory 내부 상태와 Repository에 반영되는지 확인	code == 1, qty == 3	stock 3 감소, updateCalled == true	Pass
UT-1.6	DecreaseStock_Fail_Invalid_Code/ decreaseStock()	코드가 유효하지 않을 경우 재고 차감이 발생하지 않는지 확인	code == 0	updateCalled == false	Pass
UT-1.7	DecreaseStock_Fail_Not_Enough_Quantity/ decreaseStock()	보유 수량보다 많은 수량 차감 시 실패 처리 여부 확인	code == 1, qty == 100	updateCalled == false	Pass

Act. 2061. Unit Testing(gtest)

Prepayment Unit Test

Num	Test Case Name/ Related Function	Description	Condition	Expect	Pass/Fail
UT-2.1	MakeRequestStockMessage_FormatsCorrectJson/ makeRequestStockMessage()	JSON 형식이 올바르게 생성되는지 검증	itemCode = 5, qty = 3	"05", "3" 포함된 JSON 반환	Pass
UT-2.2	MakeRequestStockMessage_TwoDigitItemCode/ makeRequestStockMessage()	itemCode가 두 자리일 때 padding 없이 처리되는지 확인	itemCode = 12	"12"로 그대로 출력	Pass
UT-2.3	MakeRequestPrepaymentMessage_GeneratesCorrectJson/ makeRequestPrepaymentMessage()	요청 메시지 JSON의 항목 구성 확인	certCode, itemCode=5, qty=2	msg_type == "req_prepay" 포함	Pass
UT-2.4	MakeRequestPrepaymentMessage_TwoDigitItemCode/ makeRequestPrepaymentMessage()	itemCode가 10 이상일 때 그대로 출력되는지 확인	itemCode = 15	"15" 출력	Pass

Act. 2061. Unit Testing (gtest)

Prepayment Unit Test

Num	Test Case Name/ Related Function	Description	Condition	Expect	Pass/ Fail
UT-2.5	FindDistance_ReturnsCorrect Distance/ findDistance()	거리 계산 함수의 계산 유효성 확인	DVM 위치 (3,4), 대상 (0,0)	5.0 반환	Pass
UT-2.6	GenerateCertificationCode_Re turnsCorrectLengthAndChars et/ generateCertificationCode()	생성된 인증 코드에 대한 길이와 문자 유효성 확 인	length = 10	길이 = 10, 모든 문자는 유효	Pass
UT-2.7	PrepaymentRequest_Success_ ReturnsTrue/ prepaymentRequest()	타 DVM에 대한 선결제 가능 여부 응답이 availability = "T" 인 경우	availability = "T"	True	Pass
UT-2.8	PrepaymentRequest_Failure_R eturnsFalse/ prepaymentRequest()	타 DVM에 대한 선결제 가능 여부 응답이 availability = "F" 인 경우	availability = "F"	false	Pass

Act. 2061. Unit Testing (gtest)

Prepayment Unit Test

Num	Test Case Name/ Related Function	Description	Condition	Expect	Pass/ Fail
UT-2.9	PrepaymentRequest_InvalidMsgType_ReturnsFalse/ prepaymentRequest()	타 DVM 의 msg_type이 올바르지 않을 경우의 검증	msg_type == "wrog_msg"	false 반환	Pass
UT-2.10	PrepaymentRequest_InvalidJson_ReturnsFalse/ prepaymentRequest()	타 DVM 의 요청 형식이 json 이 아닐 경우의 검증	{invalid json}	false 반환	Pass
UT-2.11	FindAvailableDVM_ReturnsClosestAvailableDVM/ findAvaliableDVM()	여러 응답 중 가장 가까운 DVM 선택하는지 검증	2개 DVM 응답, 거리 다름	가까운 DVM 반환	Pass
UT-2.12	FindAvailableDVM_AllInsufficientStock_ReturnsNullptr/ findAvaliableDVM()	모든 DVM 재고 부족 시	모든 DVM 응답에 대해 item_num < request qty	nullptr 반환	Pass

Act. 2061. Unit Testing(gtest)

Prepayment Unit Test

Num	Test Case Name/ Related Function	Description	Condition	Expect	Pass/ Fail
UT-2.13	HandlePrepaymentRequest_Success/ handlePrepaymentRequest()	타 DVM의 선결제 요청 성공 시	qty = 2, 재고 충분	true 반환 인증코드 저장됨 수량 감소	Pass
UT-2.14	HandlePrepaymentRequest_StockUnavailable_Fails/ handlePrepaymentRequest()	타 DVM의 선결제 요청 시에 수량이 없을 경우	qty = 20	False 반환 인증코드 저장 X 수량 유지	Pass
UT-2.15	PrePaymentCheck_Success/ prePaymentCheck()	선결제 물품 수령 시 유효한 코드일 경우	VALID123 미리 저장.	itemCode, qty 반환 certCode삭제됨	Pass
UT-2.16	PrePaymentCheck_CodeNotFound/ prepaymentCheck()	선결제 물품 수령 시 유효하지 않은 코드일 경우	UNKNOWN 입력	-1, -1 반환	Pass

Act. 2061. Unit Testing(gtest)

Purchase Unit Test

Num	Test Case Name/ Related Function	Description	Condition	Expect	Pass/ Fail
UT-3.1	Purchase_Success/ purchase()	결제 시스템이 성공 응답을 주었을 때 재 고 차감 및 결제 성공 메시지 확인	카드번호: "1234-5678", 수량: 2, 결제 성공	result.first == true, result.second == "결제 성공", 수량 2만큼 감소	Pass
UT-3.2	Purchase_Fail_Insufficient Balance/ purchase()	결제 시스템에서 잔액 부족 응답을 줄 경 우 재고 차감 없이 실패 처리	카드번호: "1234-5678", 수량: 2, 결제 실패	result.first == false, result.second == "잔액 부족", 수량 유지	Pass

Act. 2061. Unit Testing (gtest)

Result of Unit Test

