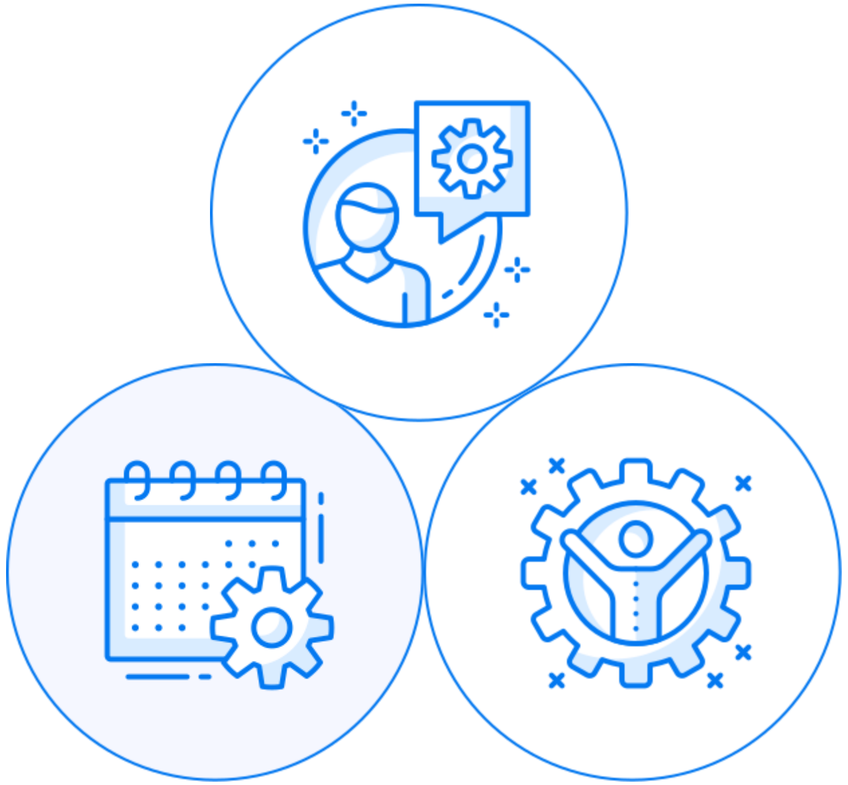
객체지향개발방법론

OOPT 2050

1st Cycle



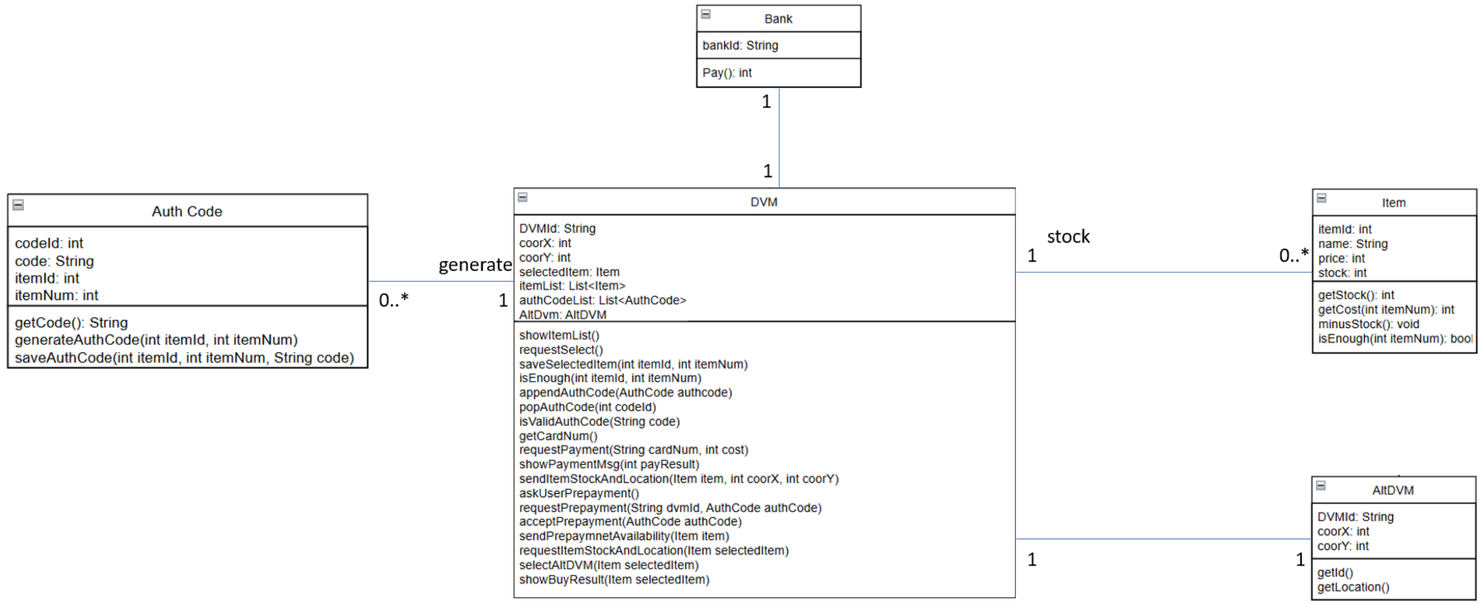
팀원 : 201912315 김성유 | 201411282 양대경 | 202011396 황정승

https://github.com/brian770/OOPT_development

목차

1. OOD (2040) 에서 변경/수정된 부분 정리
2. 구현 내용
3. 빌드 및 실행 방법 소개
4. Unit Test

1. OOD (2040) 에서 변경/수정된 부분 정리

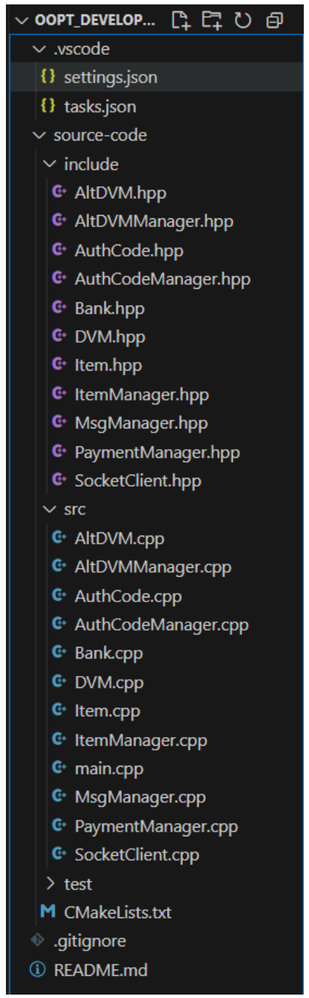


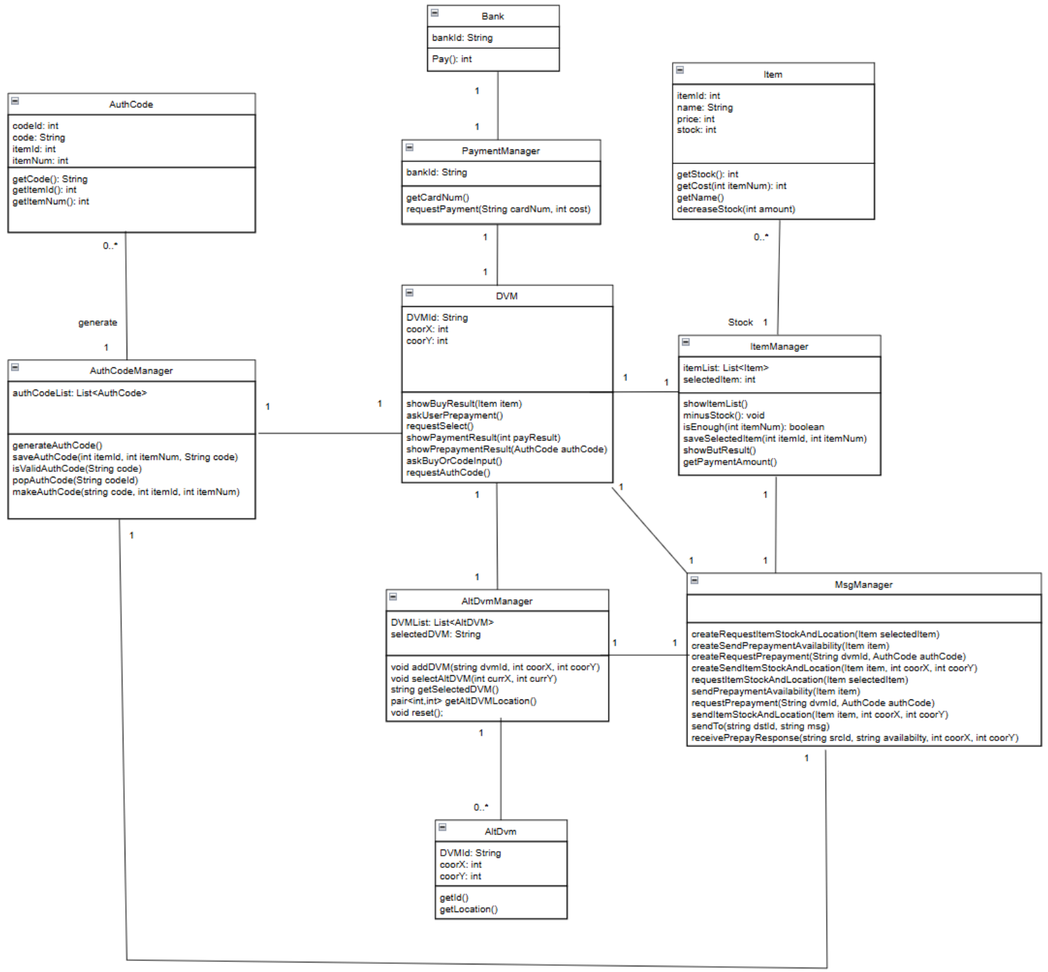
기존 클래스 다이어그램 문제점

- DVM 객체에 너무 많은 기능이 집중됨
- 각 객체를 관리하는 위치가 불분명함(유지보수성 악화)
- 새로운 객체 생성 및 연결이 어려움 (확장성 악화)
- 각 객체의 사용의도에 따라 불필요한 메소드가 포함됨(값만 받아오면 되는데 다른 메소드 존재)

-> 즉, 절차지향 프로그래밍에 가까움

1. OOD (2040) 에서 변경/수정된 부분 정리





- main.cpp 에서 DVM 생성자 호출
- DVM은 각 Manager 객체와 상호작용
  - ItemManager: Item 객체 관리
  - AltDVMManager: 대안 자판기 선정 및 AltDVM 객체 관리
  - PaymentManager: 결제 진행 및 Bank 객체와 상호작용
  - AuthCodeManager: AuthCode 객체 관리
  - MsgManager: 타 자판기와 통신할 때 주고받는 메시지 관리
- main.cpp에서 서버 주소 및 포트 번호를 입력하여 SocketClient 객체 생성 및 서버와 연결(서버 미구현으로 인한 미완성)

2. 구현 내용

DVM.hpp

```cpp
#ifndef DVM_H
#define DVM_H

#include "ItemManager.hpp"
#include "PaymentManager.hpp"
#include "AuthCodeManager.hpp"
#include "MsgManager.hpp"
#include "AltDVMManager.hpp"
#include <string>
#include <algorithm>


class DVM {
private:
    std::string DVMId;
    int coorX;
    int coorY;

public:
    DVM(const std::string& id, int x, int y);
    std::string askUserPrepayment();
    std::pair<int, int> requestSelect();
    void showPaymentResult(int payResult);
    void showPrepaymentResult(const std::string authCode, std::pair<int,int> location);

    // 추가 메서드
    bool askBuyOrCodeInput();
    std::string requestAuthCode();
};

#endif // DVM_H
```

- 필드
  - 자판기 id
  - 자판기 x 좌표
  - 자판기 y 좌표
- 메소드
  - DVM 생성자
  - 선결제 의사 확인
  - 음료수 선택 입력 받기
  - 결제 결과 표시
  - 선결제 결과 표시
  - 음료수 선택 또는 코드 입력 선택 요청
  - 인증 코드 입력 받기

2. 구현 내용

DVM.cpp

```cpp
#include "DVM.hpp"
#include <iostream>

using namespace std;

DVM::DVM(const string& id, int x, int y)
    : DVMId(id), coorX(x), coorY(y) {
        ItemManager itemManager;
        PaymentManager paymentManager;
        AuthCodeManager authCodeManager;
        AltDVMManager altDVMManager(itemManager);
        MsgManager msgManager(itemManager, authCodeManager, altDVMManager, "1");
        bool buyOrCodeInput = askBuyOrCodeInput();
        // 음료수 구매
        if(buyOrCodeInput){
            itemManager.showItemList();
            itemManager.saveSelectedItem(requestSelect());
            // 재고가 충분한 경우
            if(itemManager.isEnough()){
                int payResult = paymentManager.requestPayment(itemManager.getPaymentAmount());
                showPaymentResult(payResult);
                // 결제 성공한 경우
                if(payResult == 1){
                    itemManager.minusStock();
                    itemManager.showBuyResult();
                }
                // 결제 실패한 경우
                else return;
            }
            // 재고가 부족한 경우
            else{
                msgManager.requestItemStockAndLocation();
                altDVMManager.selectAltDVM(x,y);
                // 선결제 O
                if(askUserPrepayment().compare("y") == 0){
                    string authCode = authCodeManager.generateCode();
                    string dvmID = altDVMManager.getSelectedDVM();
                    msgManager.requestPrepayment(dvmID, authCodeManager.makeAuthCode(
                        authCode, itemManager.getSelectedItemId(),itemManager.getSelectedItemNum()));
                    showPrepaymentResult(authCode, altDVMManager.getAltDVMLocation());
                }
                // 선결제 X
                else{
                    string authCode = "noprepayment";
                    string dvmID = altDVMManager.getSelectedDVM();
                    showPrepaymentResult(authCode, altDVMManager.getAltDVMLocation());
                }
            }
        }
        // 인증코드 입력
        else{
            const string code = requestAuthCode();
            // 인증코드 확인
            if(authCodeManager.isValidAuthCode(code)) {
                itemManager.saveSelectedItem(authCodeManager.popAuthCode(code));
            } else {
                cout << "해당 인증코드가 존재하지 않습니다." << endl;
                return;
            }
            // 인증코드 객체에 담겨있는 정보로 음료수 제공
            itemManager.showBuyResult();
        }
    }
```

- DVM 생성자 구현

2. 구현 내용

```cpp
// 선결제 여부 묻기
// 음료수의 재고가 부족할 때
string DVM::askUserPrepayment() {
    string answer;
    bool isAvailable = false;
    while(!isAvailable){
        cout << "선결제 하시겠습니까? Yes(Y|y) No(N|n): ";
        cin >> answer;
        transform(answer.begin(), answer.end(), answer.begin(), ::tolower);
        if(answer.compare("y") == 0 || answer.compare("n") == 0){
            isAvailable = true;
        }
        else {
            cout << "Y|y 또는 N|n으로 입력해주세요." << endl;
        }
    }
    return answer;
}
```

- 선결제 의사 확인 구현

```cpp
// 결제 결과 출력
void DVM::showPaymentResult(int payResult) {
    switch (payResult) {
        case 1:
            cout << "결제 성공하였습니다." << endl;
            break;
        case 2:
            cout << "카드 정보가 불일치합니다." << endl;
            break;
        case 3:
            cout << "계좌에 돈이 부족합니다." << endl;
            break;
        default:
            cout << "알 수 없는 오류가 발생했습니다." << endl;
            break;
    }
}
```

- 결제 결과 표시 구현

```cpp
// 항목 선택 요청
pair<int,int> DVM::requestSelect() {
    pair<int,int> answer;
    bool isAvailable = false;

    while(!isAvailable){
        cout << "음료수 번호를 입력해주세요: ";
        cin >> answer.first;
        if(answer.first <= 20 && answer.first >= 1){
            isAvailable = true;
        }
        else{
            cout << "음료수 번호는 1 ~ 20 사이의 값입니다." << endl;
        }
    }
    isAvailable = false;

    while(!isAvailable){
        cout << "수량을 입력해주세요: ";
        cin >> answer.second;
        if(answer.second <= 99 && answer.second >= 1){
            isAvailable = true;
        }
        else{
            cout << "음료수 수량은 1~99 까지만 선택 가능합니다." << endl;
        }
    }
    return answer;
}
```

- 음료수 선택 입력 받기 구현

2. 구현 내용

```
// 선결제 결과 출력
void DVM::showPrepaymentResult(const string authCode, pair<int,int> location) {
    // 선결제 O
    if(authCode.compare("noprepayment")!=0){
        cout << "인증코드는 " << authCode << "입니다." << endl;
        if(location.first == -1) cout << "대안자판기가 없습니다." << endl;
        else cout << "대안 자판기의 위치는 " << location.first << "," << location.second << " 입니다." << endl;
    }
    // 선결제 X
    else{
        if(location.first == -1) cout << "대안자판기가 없습니다." << endl;
        else cout << "대안 자판기의 위치는 " << location.first << "," << location.second << " 입니다." << endl;
    }
}
```

• 선결제 결과 표시 구현

```
// 음료수 구매일 경우 true 반환
bool DVM::askBuyOrCodeInput(){
    string answer;
    bool isAvailable = false;
    while(!isAvailable){
        cout << "음료수 구매(1) | 인증코드 입력(2): ";
        cin >> answer;
        if(answer.compare("1") == 0 || answer.compare("2") == 0){
            isAvailable = true;
        }
        else {
            cout << "1 또는 2로 입력해주세요." << endl;
        }
    }
    return answer.compare("1") == 0;
}
```

• 음료수 선택 또는 코드 입력 선택 요청 구현

```
string DVM::requestAuthCode(){
    string answer;
    bool isAvailable = false;
    while(!isAvailable){
        cout << "인증코드를 입력해주세요(5자리):  ";
        cin >> answer;
        if(answer.length()==5){
            isAvailable = true;
        }
        else {
            cout << "5자리 인증코드를 입력해주세요." << endl;
        }
    }
    return answer;
}
```

• 인증 코드 입력 받기 구현

2. 구현 내용

ItemManager.hpp

```hpp
#ifndef ITEMMANAGER_H
#define ITEMMANAGER_H

#include <vector>
#include "Item.hpp"

class ItemManager {
private:
    std::vector<Item> itemList;
    int selectedItemId;
    int selectedItemNum;

public:
    // 생성자
    ItemManager();

    // 메서드
    void showItemList() const;
    void saveSelectedItem(std::pair<int, int> saveInfo); // 매개변수 추가
    bool isEnough() const;
    void minusStock();

    // 추가 메서드
    void showBuyResult();
    int getPaymentAmount();

    // 구글테스트용 메서드
    int getSelectedItemId();
    int getSelectedItemNum();
    std::vector<Item> getItemList();
};

#endif // ITEMMANAGER_H
```

- 필드
  - item 객체를 담고 있는 벡터
  - 사용자가 고른 음료수의 id를 가지고 있는 변수
  - 사용자가 고른 음료수의 수량을 가지고 있는 변수
- 메소드
  - ItemManager 생성자
  - 메뉴 출력
  - 사용자가 선택한 음료수 정보 저장
  - 사용자가 선택한 음료수의 재고가 충분한지 확인
  - 음료수 재고 감소
  - 사용자에게 음료수 제공
  - 결제 금액 계산 및 반환
  - 사용자가 선택한 음료수의 id 반환
  - 사용자가 선택한 음료수의 수량 반환

2. 구현 내용

ItemManager.cpp

```cpp
ItemManager::ItemManager()
    : selectedItemId(0), selectedItemNum(0) {
    // 초기 아이템 등록
    itemList.emplace_back(1, "콜라", 1500, 10);
    itemList.emplace_back(2, "사이다", 1400, 8);
    itemList.emplace_back(3, "녹차", 1300, 5);
    itemList.emplace_back(4, "홍차", 1350, 6);
    itemList.emplace_back(5, "밀크티", 1800, 7);
    itemList.emplace_back(6, "탄산수", 1200, 10);
    itemList.emplace_back(7, "보리차", 1100, 9);
    itemList.emplace_back(8, "캔커피", 1600, 8);
    itemList.emplace_back(9, "물", 1000, 20);
    itemList.emplace_back(10, "에너지드링크", 2000, 6);
    itemList.emplace_back(11, "유자차", 1500, 4);
    itemList.emplace_back(12, "식혜", 1400, 5);
    itemList.emplace_back(13, "아이스티", 1350, 6);
    itemList.emplace_back(14, "딸기주스", 1700, 7);
    itemList.emplace_back(15, "오렌지주스", 1700, 8);
    itemList.emplace_back(16, "포도주스", 1700, 7);
    itemList.emplace_back(17, "이온음료", 1300, 10);
    itemList.emplace_back(18, "아메리카노", 1900, 5);
    itemList.emplace_back(19, "핫초코", 1800, 6);
    itemList.emplace_back(20, "카페라떼", 2000, 4);
}
```

• ItemManager 생성자 구현

```cpp
void ItemManager::saveSelectedItem(pair<int, int> saveInfo) {
    selectedItemId = saveInfo.first;
    selectedItemNum = saveInfo.second;
}
```

• 사용자가 선택한 음료수 정보 저장 구현

```cpp
void ItemManager::showItemList() const {
    for(int id=0; id<itemList.size(); id++){
        cout << "ID: " << id+1
                << ", Name: " << itemList[id].getName()
                << ", Price: " << itemList[id].getCost(1)
                << ", Stock: " << itemList[id].getStock() << endl;
    }
}
```

• 메뉴 출력 구현

2. 구현 내용

```
bool ItemManager::isEnough() const {
    if(itemList[selectedItemId-1].getStock() >= selectedItemNum)
        return true;
    else
        return false;
}
```

- 사용자가 고른 수량만큼 재고가 있는지 확인

```
void ItemManager::minusStock() {
    itemList[selectedItemId-1].decreaseStock(selectedItemNum);
}
```

- 사용자가 고른 수량만큼 재고 감소

```
void ItemManager::showBuyResult() {
    cout << itemList[selectedItemId-1].getName() << " " << selectedItemNum << "개 구매되었습니다." << endl;
}
```

- 사용자가 구매한 음료수 제공

```
int ItemManager::getPaymentAmount(){
    return itemList[selectedItemId-1].getCost(selectedItemNum);
}
```

- 결제 금액 계산 및 반환 구현

2. 구현 내용

PaymentManager.hpp

```hpp
#ifndef PaymentManager_HPP
#define PaymentManager_HPP

#include "Bank.hpp"
#include <string>

class PaymentManager {
private:
    std::string pmId;

public:
    // 생성자
    PaymentManager();

    // 카드번호를 가져오는 메서드
    std::string getCardNum();

    // 결제를 요청하는 메서드 (카드번호를 내부적으로 호출해서 사용)
    // 1: 성공, 2: 카드 정보 안 맞음, 3: 계좌에 돈이 부족함, -1: 예외상황 발생
    int requestPayment(int cost);
};

#endif // PaymentManager_HPP
```

- 필드
  - PaymentManager ID
- 메소드
  - PaymentManager 생성자
  - 사용자에게서 카드 번호를 입력받음
  - Bank에 결제 요청

2. 구현 내용

PaymentManager.cpp

```
PaymentManager::PaymentManager() : pmId("temp") {
}
```

• PaymentManager 생성자 구현

```
string PaymentManager::getCardNum() {
    string cardNum;
    bool isAvailable = false;
    while(!isAvailable){
        cout << "카드 번호를 입력해주세요(5자리): ";
        cin >> cardNum;
        if(cardNum.length() == 5){
            isAvailable = true;
        }
        else {
            cout << "5자리 카드번호로 입력해주세요." << endl;
        }
    }
    return cardNum;
}
```

• 사용자에게서 카드 번호를 입력받아서 반환

```
int PaymentManager::requestPayment(int cost) {
    Bank bank;
    return bank.pay(getCardNum(), cost);
}
```

• getCardNum을 인자로 넘겨서 카드 번호를 저장하지 않고 은행에 결제 요청을 하는 코드 구현

2. 구현 내용

Item.hpp

```
#ifndef ITEM_H
#define ITEM_H

#include <string>

class Item {
private:
    int itemId;
    std::string name;
    int cost;
    int stock;

public:
    // 생성자
    Item(int itemId, const std::string& name, int price, int stock);

    // Getter
    int getStock() const;
    int getCost(int itemNum) const;
    std::string getName() const;

    // 추가 메서드
    void decreaseStock(int amount);
};

#endif // ITEM_H
```

- 필드
  - 음료수의 ID
  - 음료수의 이름
  - 음료수의 가격
  - 음료수의 재고
- 메소드
  - Item 생성자
  - 음료수의 재고를 반환
  - 음료수의 가격 반환
  - 음료수의 이름 반환
  - 음료수의 재고 감소

2. 구현 내용

Item.cpp

```cpp
// 생성자 정의
Item::Item(int itemId, const string& name, int cost, int stock)
    : itemId(itemId), name(name), cost(cost), stock(stock) {}

// 재고 반환
int Item::getStock() const {
    return stock;
}

// 수량에 따른 총 비용 계산
int Item::getCost(int itemNum) const {
    return cost * itemNum;
}

string Item::getName() const {
    return name;
}

// 추가 메서드
void Item::decreaseStock(int amount) {
    if (stock >= amount) {
        stock -= amount;
    }
}
```

Item 생성자 구현

Stock을 반환하는 코드 구현

구매하려는 개수를 매개변수로 입력받아서 총 가격 반환

음료수의 이름 반환

음료수의 재고가 amount보다 많으면 amount만큼 차감

2. 구현 내용

Bank.hpp

```hpp
#ifndef BANK_HPP
#define BANK_HPP

#include <string>
#include <map>

class Bank {
private:
    std::string bankId;
    std::map<std::string, int> account;
public:
    // 생성자
    Bank();

    // 결제 메서드
    int pay(std::string cardNum, int cost);

    // 테스트용 메서드
    std::map<std::string, int> getAccount();
};

#endif // BANK_HPP
```

- 필드
  - 은행의 ID
  - 은행 내부적으로 저장되어있는 카드번호와 잔고
- 메소드
  - Bank 생성자
  - cardNum과 비용을 매개변수로 받아서 결제

2. 구현 내용

Bank.cpp

```cpp
Bank::Bank()
: bankId("신한은행"){
    account.insert(pair<string, int>("12345", 10000)); // test input
}

// 결제 처리 메서드
// 1: 결제 성공, 2: 카드 정보 없음, 3: 잔액 부족, -1: 예외상황 발생
// 예외 상황이 있나?
int Bank::pay(const string cardNum, int cost) {
    for(auto& iter : account){
        if(iter.first == cardNum){
            if(iter.second >= cost){
                iter.second -= cost; // 잔액 차감
                return 1; // 결제 성공
            }
            else return 3; // 잔액 부족
        }
    }
    return 2;
}
```

Bank 생성자

cardNum을 입력받으면 해당하는 cardNum이 은행 내부의 계좌를 저장해둔 배열에 있는지 확인 후에 cardNum이 있고 잔고가 충분할 경우에 잔액 차감 및 1 반환

2. 구현 내용

AuthCodeManager.hpp

```hpp
#ifndef AUTHCODEMANAGER_H
#define AUTHCODEMANAGER_H

#include <vector>
#include "AuthCode.hpp"

class AuthCodeManager {
private:
    std::vector<AuthCode> authCodeList;

public:
    // 생성자
    AuthCodeManager();

    // 메서드
    std::string generateCode();
    AuthCode makeAuthCode(const std::string& code, int itemId, int itemNum);
    void saveAuthCode(const std::string& code, int itemId, int itemNum);
    bool isValidAuthCode(const std::string& code);
    std::pair<int,int> popAuthCode(const std::string& code);

};

#endif // AUTHCODEMANAGER_H
```

- 필드
  - AuthCode 객체 vector
- 메소드
  - AuthCodeManager 생성자
  - 전달용 인증코드 생성
  - 전달받은 인증코드 저장
  - 동일 인증코드 존재 여부 확인
  - 사용된 인증코드 삭제 및 반환

2. 구현 내용

AuthCodeManager.cpp

```cpp
#include "AuthCodeManager.hpp"
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <algorithm>

using namespace std;

// 생성자
AuthCodeManager::AuthCodeManager()
    : authCodeList() {
    srand((unsigned int)time(NULL));
    authCodeList.emplace_back("12345", 1, 1);
}

string AuthCodeManager::generateCode() {
    // 인증코드 생성
    srand((unsigned int)time(NULL));

    // 숫자와 대소문자로 구성된 5자리 문자열 저장할 변수
    char code_str[6];
    int type; // 0: 숫자, 1: 소문자, 2: 대문자

    for (int i = 0; i < 5; i++) {
        type = rand() % 3; // 0, 1, 2 중 랜덤으로 선택
        if (type == 0) {
            code_str[i] = '0' + rand() % 10; // 숫자
        } else if (type == 1) {
            code_str[i] = 'a' + rand() % 26; // 소문자
        } else {
            code_str[i] = 'A' + rand() % 26; // 대문자
        }
    }

    code_str[5] = '\0'; // 문자열 종료 문자 추가

    return string(code_str); // string으로 변환
}
```

• AuthCodeManager 생성자 구현

```cpp
// 전달용 인증코드 생성
AuthCode AuthCodeManager::makeAuthCode(const string& code, int itemId, int itemNum) {
    return AuthCode(code, itemId, itemNum);
}
```

• 전달용 인증코드 생성 구현

```cpp
// 전달받은 인증코드 저장
void AuthCodeManager::saveAuthCode(const string& code, int itemId, int itemNum) {
    authCodeList.emplace_back(code, itemId, itemNum);
}
```

• 전달받은 인증코드 저장 구현

```cpp
// 해당 인증코드가 존재하는지 확인
bool AuthCodeManager::isValidAuthCode(const string& code) {
    for (const auto& authCode : authCodeList) {
        if (authCode.getCode() == code) {
            return true;
        }
    }
    return false;
}
```

• 동일 인증코드 존재 여부 확인 구현

```cpp
// 사용된 인증 코드 삭제
pair<int,int> AuthCodeManager::popAuthCode(const string& code) {
    for (auto iter = authCodeList.begin(); iter != authCodeList.end(); ++iter) {
        if (iter->getCode() == code) {
            AuthCode temp = *iter;
            authCodeList.erase(iter);
            return make_pair(temp.getItemId(), temp.getItemNum());
        }
    }
    return make_pair(0,0);
}
```

• 사용된 인증코드 삭제 및 반환 구현

2. 구현 내용

AuthCode.hpp

```cpp
#ifndef AUTHCODE_H
#define AUTHCODE_H

#include <string>

class AuthCode {
private:
    std::string code;
    int itemId;
    int itemNum;

public:
    // 생성자
    AuthCode(const std::string& code, int itemId, int itemNum);

    // Getter
    std::string getCode() const;
    int getItemId();
    int getItemNum();
};

#endif // AUTHCODE_H
```

- 필드
  - 인증코드
  - 음료수 ID
  - 음료수 수량
- 메소드
  - AuthCode 생성자
  - 인증코드 반환
  - 음료수 ID 반환
  - 음료수 수량 반환

2. 구현 내용

AuthCode.cpp

```cpp
#include "AuthCode.hpp"

using namespace std;

// 생성자 정의
AuthCode::AuthCode(const string& code, int itemId, int itemNum)
    : code(code), itemId(itemId), itemNum(itemNum) {}
```

• AuthCode 생성자 구현

```cpp
// 인증코드 반환
string AuthCode::getCode() const{
    return code;
}
```

• 인증코드 반환 구현

```cpp
// 음료수 수량 반환
int AuthCode::getItemNum(){
    return itemNum;
}
```

• 음료수 수량 반환 구현

```cpp
// 음료수 번호 반환
int AuthCode::getItemId(){
    return itemId;
}
```

• 음료수 ID 반환 구현

2. 구현 내용

AltDVM.hpp

```cpp
#pragma once
#include <string>
#include <utility>

class AltDVM {
private:
    std::string DVMId;
    int coorX;
    int coorY;

public:
    AltDVM(const std::string& id, int x, int y);
    std::string getId() const;
    std::pair<int, int> getLocation() const;
};
```

- 필드
  - 대안 자판기 Id
  - 대안 자판기 x좌표
  - 대안 자판기 y좌표
- 메소드
  - 대안 자판기 생성자
  - 대안 자판기 id반환
  - 대안 자판기 좌표 반환

2. 구현 내용

AltDVM.cpp

```cpp
#include "AltDVM.hpp"

using namespace std;

AltDVM::AltDVM(const string& id, int x, int y)
    : DVMId(id), coorX(x), coorY(y) {}

string AltDVM::getId() const {
    return DVMId;
}

pair<int, int> AltDVM::getLocation() const {
    return {coorX, coorY};
}
```

- AltDVM 생성자 구현
- 대안 자판기 id반환 구현
- 대안 자판기 좌표 반환 구현

2. 구현 내용

AltDVMManager.hpp

```cpp
#pragma once
#include <vector>
#include <string>
#include "AltDVM.hpp"
#include "ItemManager.hpp"

class AltDVMManager {
private:
    std::vector<AltDVM> DVMList;
    std::string selectedDVMId;
    ItemManager itemManager;

public:
    AltDVMManager(ItemManager im);
    void addDVM(const std::string& dvmId, int coorX, int coorY, const std::string& availability);
    void selectAltDVM(int currX, int currY); // 현재 자판기 위치를 받아 선택
    std::string getSelectedDVM() const;
    std::pair<int,int> getAltDVMLocation() const;
    void reset();

    // google test용 메서드
    std::vector<AltDVM> getAltDVMList();
};
```

- 필드
  - AltDVM 객체를 담고 있는 vevtor
  - 선정된 대안 자판기 Id
  - Item 정보를 관리하는 ItemManager 객체
- 메소드
  - AltDVMManager 생성자
  - 대안 자판기 리스트에 저장
  - 가장 가까운 대안자판기 선정
  - 선택된 대안 자판기 id 반환
  - 선택된 대안자판기 좌표 반환
  - 대안 자판기 초기화
  - 저장된 대안 자판기 반환

2. 구현 내용

AltDVMManager.cpp

```cpp
AltDVMManager::AltDVMManager(ItemManager im) : itemManager(im) {

}

void AltDVMManager::addDVM(const string& dvmId, int coorX, int coorY, const string& availability) {
    if (availability == "T") {
        DVMList.emplace_back(dvmId, coorX, coorY);
    }
}

void AltDVMManager::selectAltDVM(int currX, int currY) {
    if (DVMList.empty()) return;

    int minDist = numeric_limits<int>::max();
    for (auto& dvm : DVMList) {
        auto [x, y] = dvm.getLocation();
        int dist = (currX - x) * (currX - x) + (currY - y) * (currY - y);
        if (dist < minDist) {
            minDist = dist;
            selectedDVMId = dvm.getId();
        }
    }
}
```

- AltDVMManager 생성자 구현
- 대안 자판기 리스트에 저장 구현
- 가장 가까운 대안자판기 선정 구현

2. 구현 내용

AltDVMManager.cpp

```cpp
string AltDVMManager::getSelectedDVM() const {
    return selectedDVMId;
}

void AltDVMManager::reset() {
    DVMList.clear();
    selectedDVMId.clear();
}

std::pair<int,int> AltDVMManager::getAltDVMLocation() const{
    for(int iter = 0; iter<DVMList.size(); iter++){
        if(DVMList[iter].getId().compare(selectedDVMId)==0){
            return DVMList[iter].getLocation();
        }
    }
    return {-1,-1};
}

std::vector<AltDVM> AltDVMManager::getAltDVMList()
{
    return DVMList;
}
```

- 선택된 대안 자판기 id 반환 구현
- 대안 자판기 초기화 구현
- 선택된 대안자판기 좌표 반환 구현
- 저장된 대안 자판기 반환 구현

2. 구현 내용

MsgManager.hpp

```cpp
#pragma once
#include "ItemManager.hpp"
#include "AuthCodeManager.hpp"
#include "AltDVMManager.hpp"
#include <string>

class MsgManager {
private:
    ItemManager itemManager;
    AuthCodeManager authCodeManager;
    AltDVMManager altDvmManager;
    std::string myId;

public:
    MsgManager(ItemManager im, AuthCodeManager am, AltDVMManager adm, const std::string& selfId);

    std::string createRequestItemStockAndLocation();
    std::string createPrepaymentAvailability(const std::string& dstId);
    std::string createRequestPrepayment(const std::string& dvmId, AuthCode authCode);
    std::string createItemStockAndLocation(const std::string& dstId, int coorX, int coorY);

    void requestItemStockAndLocation();
    void sendPrepaymentAvailability(const std::string& dstId);
    void requestPrepayment(const std::string& dvmId, AuthCode authCode);
    void sendItemStockAndLocation(const std::string& requesterId, int coorX, int coorY);

    void sendTo(const std::string& dstId, const std::string& msg);
    void receivePrepayResponse(const std::string& srcId, const std::string& availability, int coorX, int coorY);
};
```

- 필드
  - Item 정보를 관리하는 객체
  - AuthCode를 관리하는 객체
  - AltDVM을 관리하는 객체
  - 현재 자판기 Id
- 메소드
  - MsgManager 생성자
  - Item재고 및 좌표 요청 메시지 생성
  - 선 결제 가능 여부 메시지 생성
  - 대안 자판기에 선 결제 요청 메시지 생성
  - 현재 자판기의 재고와 좌표를 담은 메시지 생성
  - Item 재고 및 좌표 요청
  - 선 결제 가능 여부 메시지 전송
  - 대안 자판기에 선 결제 요청
  - 현재 자판기의 재고와 좌표를 담은 메시지 전송
  - 메시지를 해당 자판기에 전송
  - 다른 자판기로부터 받은 선 결제 메시지 처리

2. 구현 내용

MsgManager.cpp

```cpp
MsgManager::MsgManager(ItemManager im, AuthCodeManager am, AltDVMManager adm, const string& selfId)
    : itemManager(im), authCodeManager(am), altDvmManager(adm), myId(selfId) {}

// item 재고와 자판기 위치를 요청하는 메세지 생성
string MsgManager::createRequestItemStockAndLocation() {
    int itemId = itemManager.getSelectedItemId();
    int itemNum = itemManager.getSelectedItemNum();
    ostringstream oss;
    oss << "{\n"
        << "  \"msg_type\": \"req_stock\",\n"
        << "  \"src_id\": \"" << myId << "\",\n"
        << "  \"dst_id\": \"0\",\n"
        << "  \"msg_content\": {\n"
        << "    \"item_code\": \"" << setw(2) << setfill('0') << itemId << "\",\n"
        << "    \"item_num\": \"" << setw(2) << setfill('0') << itemNum << "\"\n"
        << "  }\n}";
    return oss.str();
}

// 선결제 가능 여부 응답 메세지 생성
string MsgManager::createPrepaymentAvailability(const string& dstId) {
    int itemId = itemManager.getSelectedItemId();
    int itemNum = itemManager.getSelectedItemNum();
    bool available = itemManager.isEnough();
    ostringstream oss;
    oss << "{\n"
        << "  \"msg_type\": \"resp_prepay\",\n"
        << "  \"src_id\": \"" << myId << "\",\n"
        << "  \"dst_id\": \"" << dstId << "\",\n"
        << "  \"msg_content\": {\n"
        << "    \"item_code\": \"" << setw(2) << setfill('0') << itemId << "\",\n"
        << "    \"item_num\": \"" << setw(2) << setfill('0') << itemNum << "\",\n"
        << "    \"availability\": \"" << (available ? "T" : "F") << "\"\n"
        << "  }\n}";
    return oss.str();
}
```

- MsgManager 생성자
- Item재고 및 좌표 요청 메시지 생성 구현
  (추후 JSON형식으로 변경 예정)
- 선 결제 가능 여부 메시지 생성
  (추후 JSON형식으로 변경 예정)

2. 구현 내용

MsgManager.cpp

```cpp
// 선결제 가능 여부 요청 메세지 생성
string MsgManager::createRequestPrepayment(const string& dvmId, AuthCode authCode) {
    int itemId = authCode.getItemId();
    int itemNum = authCode.getItemNum();
    string code = authCode.getCode();
    ostringstream oss;
    oss << "{\n"
        << "  \"msg_type\": \"req_prepay\",\n"
        << "  \"src_id\": \"" << myId << "\",\n"
        << "  \"dst_id\": \"" << dvmId << "\",\n"
        << "  \"msg_content\": {\n"
        << "    \"item_code\": \"" << setw(2) << setfill('0') << itemId << "\",\n"
        << "    \"item_num\": \"" << setw(2) << setfill('0') << itemNum << "\",\n"
        << "    \"cert_code\": \"" << code << "\"\n"
        << "  }\n}";
    return oss.str();
}

// item 재고와 자판기 위치를 응답하는 메세지 생성
string MsgManager::createItemStockAndLocation(const string& dstId, int coorX, int coorY) {
    int itemId = itemManager.getSelectedItemId();
    int itemNum = itemManager.getSelectedItemNum();
    ostringstream oss;
    oss << "{\n"
        << "  \"msg_type\": \"resp_stock\",\n"
        << "  \"src_id\": \"" << myId << "\",\n"
        << "  \"dst_id\": \"" << dstId << "\",\n"
        << "  \"msg_content\": {\n"
        << "    \"item_code\": \"" << setw(2) << setfill('0') << itemId << "\",\n"
        << "    \"item_num\": \"" << setw(2) << setfill('0') << itemNum << "\",\n"
        << "    \"coor_x\": \"" << coorX << "\",\n"
        << "    \"coor_y\": \"" << coorY << "\"\n"
        << "  }\n}";
    return oss.str();
}
```

- 대안 자판기에 선 결제 요청 메시지 생성 구현
  (추후 JSON형식으로 변경 예정)

- 현재 자판기의 재고와 좌표를 담은 메시지 생성 구현
  (추후 JSON형식으로 변경 예정)

2. 구현 내용

MsgManager.cpp

```cpp
void MsgManager::receivePrepayResponse(const string& srcId, const string& availability, int coorX, int coorY) {
    altDvmManager.addDVM(srcId, coorX, coorY, availability);
}

// item 재고와 자판기 위치를 요청하는 메세지
void MsgManager::requestItemStockAndLocation() {
    sendTo("0", createRequestItemStockAndLocation());
}

// 선결제 가능 여부 응답 메세지
void MsgManager::sendPrepaymentAvailability(const string& dstId) {
    sendTo(dstId, createPrepaymentAvailability(dstId));
}

// 선결제 가능 여부 요청 메세지
void MsgManager::requestPrepayment(const string& dvmId, AuthCode authCode) {
    sendTo(dvmId, createRequestPrepayment(dvmId, authCode));
}

// item 재고와 자판기 위치를 응답하는 메세지
void MsgManager::sendItemStockAndLocation(const string& requesterId, int coorX, int coorY) {
    string msg = createItemStockAndLocation(requesterId, coorX, coorY);
    sendTo(requesterId, msg);
}

//간단 테스트용
void MsgManager::sendTo(const std::string& dstId, const std::string& msg) {
    std::cout << " sendTo " << dstId << "\n" << msg << "\n";
}
```

- 다른 자판기로부터 받은 선 결제 메시지 처리 구현
- Item 재고 및 좌표 요청
- 선 결제 가능 여부 메시지 전송 구현
- 대안 자판기에 선 결제 요청 구현
- 현재 자판기의 재고와 좌표를 담은 메시지 전송 구현
- 메시지를 해당 자판기에 전송(테스트용)
  (추후 socket사용시 변경 예정)

2. 구현 내용

SocketClient.hpp

```hpp
#ifndef SOCKET_CLIENT_HPP
#define SOCKET_CLIENT_HPP

#include <string>
#include <winsock2.h>

class SocketClient {
private:
    WSADATA wsaData;
    SOCKET clientSocket;
    sockaddr_in serverAddr;
    std::string serverIp;
    int serverPort;

public:
    SocketClient(const std::string& ip, int port);
    ~SocketClient();

    bool connectToServer();
    bool sendMessage(const std::string& msg);
    std::string receiveMessage();
    void closeConnection();
};

#endif // SOCKET_CLIENT_HPP
```

- 필드
  - 원속 초기화 정보
  - 클라이언트 소켓
  - 서버 주소 정보(IP, 포트 번호 포함)
  - 접속할 서버의 IP 주소
  - 접속할 서버의 포트 번호
- 메소드
  - SocketClient 생성자
  - SocketClient 소멸자
  - 서버에 TCP 연결 시도
  - 메시지 전송
  - 메시지 수신
  - 소켓 연결 종료

2. 구현 내용

SocketClient.cpp

```cpp
#include "SocketClient.hpp"
#include <iostream>

using namespace std;

SocketClient::SocketClient(const string& ip, int port)
    : serverIp(ip), serverPort(port), clientSocket(INVALID_SOCKET) {
    WSAStartup(MAKEWORD(2, 2), &wsaData);
}
```

• SocketClient 생성자 구현

```cpp
SocketClient::~SocketClient() {
    closeConnection();
    WSACleanup();
}
```

• SocketClient 소멸자 구현

```cpp
bool SocketClient::connectToServer() {
    clientSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (clientSocket == INVALID_SOCKET) {
        cerr << "소켓 생성 실패\n";
        return false;
    }

    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(serverPort);
    serverAddr.sin_addr.s_addr = inet_addr(serverIp.c_str());

    if (connect(clientSocket, (SOCKADDR*)&serverAddr, sizeof(serverAddr)) == SOCKET_ERROR) {
        cerr << "서버 연결 실패\n";
        closesocket(clientSocket);
        return false;
    }

    return true;
}
```

• 서버에 TCP 연결 시도 구현

2. 구현 내용

SocketClient.cpp

```cpp
bool SocketClient::sendMessage(const string& msg) {
    int sendResult = send(clientSocket, msg.c_str(), static_cast<int>(msg.size()), 0);
    return sendResult != SOCKET_ERROR;
}
```

• 메시지 전송 구현

```cpp
string SocketClient::receiveMessage() {
    char buffer[4096];
    int bytesReceived = recv(clientSocket, buffer, sizeof(buffer), 0);
    if (bytesReceived > 0) {
        return string(buffer, bytesReceived);
    }
    return "";
}
```

• 메시지 수신 구현

```cpp
void SocketClient::closeConnection() {
    if (clientSocket != INVALID_SOCKET) {
        closesocket(clientSocket);
        clientSocket = INVALID_SOCKET;
    }
}
```

• 소켓 연결 종료 구현

3. 빌드 및 실행 방법 소개

CMakeLists.txt

```
# dvm 라이브러리 정의 - 필요한 .cpp 전부 포함
add_library(dvm
    src/AuthCode.cpp
    src/AuthCodeManager.cpp
    src/Bank.cpp
    src/DVM.cpp
    src/Item.cpp
    src/ItemManager.cpp
    src/PaymentManager.cpp
    src/AltDVM.cpp
    src/AltDVMManager.cpp
    src/MsgManager.cpp
    src/SocketClient.cpp
)
```

```
# 실행 파일 (메인 프로그램)
add_executable(dvm_main src/main.cpp)
target_link_libraries(dvm_main PRIVATE dvm)

# 테스트 설정
enable_testing()

add_executable(testCode test/testCode.cpp)
target_link_libraries(testCode PRIVATE dvm gtest gtest_main)

add_test(NAME testCode COMMAND testCode)
```

3. 빌드 및 실행 방법 소개

```
j3rry3@hwangjeongseung-ui-MacBookAir OOPT_development % cd source-code
j3rry3@hwangjeongseung-ui-MacBookAir source-code % mkdir build
j3rry3@hwangjeongseung-ui-MacBookAir source-code % cd build
```

```
j3rry3@hwangjeongseung-ui-MacBookAir build % cmake ..
CMake Warning (dev) at /opt/homebrew/share/cmake/Modules/FetchContent.cmake:1373 (message):
  The DOWNLOAD_EXTRACT_TIMESTAMP option was not given and policy CMP0135 is
  not set.  The policy's OLD behavior will be used.  When using a URL
  download, the timestamps of extracted files should preferably be that of
  the time of extraction, otherwise code that depends on the extracted
  contents might not be rebuilt if the URL changes.  The OLD behavior
  preserves the timestamps from the archive instead, but this is usually not
  what you want.  Update your project to the NEW behavior or specify the
  DOWNLOAD_EXTRACT_TIMESTAMP option with a value of true to avoid this
  robustness issue.
Call Stack (most recent call first):
  CMakeLists.txt:7 (FetchContent_Declare)
This warning is for project developers.  Use -Wno-dev to suppress it.

-- Configuring done (0.2s)
-- Generating done (0.0s)
-- Build files have been written to: /Users/j3rry3/OOPT_development/source-code/build
```

3. 빌드 및 실행 방법 소개

```
j3rry3@hwangjeongseung-ui-MacBookAir build % make
[  4%] Building CXX object CMakeFiles/dvm.dir/src/AuthCode.cpp.o
[  8%] Building CXX object CMakeFiles/dvm.dir/src/AuthCodeManager.cpp.o
[ 12%] Building CXX object CMakeFiles/dvm.dir/src/Bank.cpp.o
[ 16%] Building CXX object CMakeFiles/dvm.dir/src/DVM.cpp.o
[ 20%] Building CXX object CMakeFiles/dvm.dir/src/Item.cpp.o
[ 25%] Building CXX object CMakeFiles/dvm.dir/src/ItemManager.cpp.o
[ 29%] Building CXX object CMakeFiles/dvm.dir/src/PaymentManager.cpp.o
[ 33%] Building CXX object CMakeFiles/dvm.dir/src/AltDVM.cpp.o
[ 37%] Building CXX object CMakeFiles/dvm.dir/src/AltDVMManager.cpp.o
[ 41%] Building CXX object CMakeFiles/dvm.dir/src/MsgManager.cpp.o
[ 45%] Building CXX object CMakeFiles/dvm.dir/src/SocketClient.cpp.o
[ 50%] Linking CXX static library libdvm.a
[ 50%] Built target dvm
[ 54%] Building CXX object CMakeFiles/dvm_main.dir/src/main.cpp.o
[ 58%] Linking CXX executable dvm_main
[ 58%] Built target dvm_main
[ 62%] Building CXX object _deps/googletest-build/googletest/CMakeFiles/gtest.dir/src/gtest-all.cc.o
[ 66%] Linking CXX static library ../../../lib/libgtest.a
[ 66%] Built target gtest
[ 70%] Building CXX object _deps/googletest-build/googletest/CMakeFiles/gtest_main.dir/src/gtest_main.cc.o
[ 75%] Linking CXX static library ../../../lib/libgtest_main.a
[ 75%] Built target gtest_main
[ 79%] Building CXX object CMakeFiles/testCode.dir/test/testCode.cpp.o
[ 83%] Linking CXX executable testCode
ld: warning: ignoring duplicate libraries: 'lib/libgtest.a'
[ 83%] Built target testCode
[ 87%] Building CXX object _deps/googletest-build/googlemock/CMakeFiles/gmock.dir/src/gmock-all.cc.o
[ 91%] Linking CXX static library ../../../lib/libgmock.a
[ 91%] Built target gmock
[ 95%] Building CXX object _deps/googletest-build/googlemock/CMakeFiles/gmock_main.dir/src/gmock_main.cc.o
[100%] Linking CXX static library ../../../lib/libgmock_main.a
[100%] Built target gmock_main
```

```
j3rry3@hwangjeongseung-ui-MacBookAir build % ./dvm_main
_deps/       bin/         CMakeFiles/  dvm_main*    lib/         testCode*
```

4. Unit Test

Item.hpp 관련 테스트 코드

```
// ----- Item.hpp 관련 테스트 코드 -----
// 음료수 이름 출력
TEST(ItemTest, GetItemName) {
    EXPECT_EQ(itemManager.getItemList()[0].getName(), "콜라");
}

// 음료수 가격 출력
TEST(ItemTest, GetItemCost) {
    EXPECT_EQ(itemManager.getItemList()[0].getCost(1), 1500);
}

// 음료수 수량 출력
TEST(ItemTest, GetItemStock) {
    EXPECT_EQ(itemManager.getItemList()[0].getStock(), 10);
}
// 재고 감소
TEST(ItemTest, DecreaseItemStock) {
    Item item(1, "콜라", 1500, 5);
    item.decreaseStock(1);
    EXPECT_EQ(item.getStock(), 4);
}
// ----- Item.hpp 관련 테스트 코드 -----
```

ItemManager.hpp 관련 테스트 코드

```
// ----- ItemManager.hpp 관련 테스트 코드 -----
// selectedItemId 반환
TEST(ItemManagerTest, SaveSelectedItemId){
    itemManager.saveSelectedItem(make_pair(1,1));
    EXPECT_EQ(itemManager.getSelectedItemId(), 1);
}

// selectedItemNum 반환
TEST(ItemManagerTest, SaveSelectedItemNum){
    itemManager.saveSelectedItem(make_pair(1,1));
    EXPECT_EQ(itemManager.getSelectedItemNum(), 1);
}

// 재고 확인 TRUE
TEST(ItemManagerTest, IsEnoughTrue){
    itemManager.saveSelectedItem(make_pair(1,1));
    EXPECT_TRUE(itemManager.isEnough());
}

// 재고 확인 FALSE
TEST(ItemManagerTest, IsEnoughFalse){
    itemManager.saveSelectedItem(make_pair(1,50));
    EXPECT_FALSE(itemManager.isEnough());
}

// 재고 확인 FALSE
TEST(ItemManagerTest, MinusStock){
    itemManager.saveSelectedItem(make_pair(1,1));
    itemManager.minusStock();
    EXPECT_EQ(itemManager.getItemList()[0].getStock(), 9);
}

// 재고 확인 FALSE
TEST(ItemManagerTest, GetPaymentAmount){
    itemManager.saveSelectedItem(make_pair(1,1));
    EXPECT_EQ(itemManager.getPaymentAmount(), 1500);
}
// ----- ItemManager.hpp 관련 테스트 코드 -----
```

4. Unit Test

Bank.hpp 관련 테스트 코드

```
// ----- Bank.hpp 관련 테스트 코드 -----
// 결제 성공
TEST(BankTest, PaySuccess) {
    EXPECT_EQ(bank.pay("12345", 5000), 1);
}

// 카드 정보 안맞음
TEST(BankTest, CardNumNotValid) {
    EXPECT_EQ(bank.pay("12343", 5000), 2);
}

// 계좌에 돈이 부족함
TEST(BankTest, MoneyLack) {
    EXPECT_EQ(bank.pay("12345", 50000), 3);
}

// Bank에 계좌정보 저장해둔 map에 잘 저장되는지
TEST(BankTest, GetAccount) {
    auto drinks = bank.getAccount();
    for (const auto& [cardNum, money] : drinks) {
        EXPECT_EQ(cardNum, "12345");
    }
}

TEST(BankTest, GetMoney) {
    auto drinks = bank.getAccount();
    for (const auto& [cardNum, money] : drinks) {
        EXPECT_EQ(money, 10000);
    }
}
// ----- Bank.hpp 관련 테스트 코드 -----
```

PaymentManager.hpp 관련 테스트 코드

```
// ----- PaymentManager.hpp 관련 테스트 코드 -----
TEST(PaymentManagerTest, GetCardNum) {
    std::istringstream fakeInput("12345\n");
    std::streambuf* originalCin = std::cin.rdbuf();  // 원래 cin 백업
    std::cin.rdbuf(fakeInput.rdbuf());               // cin을 가짜로 교체

    string result = paymentManager.getCardNum();

    EXPECT_EQ(result, "12345");

    std::cin.rdbuf(originalCin);  // 원래 cin 복원
}

// ----- PaymentManager.hpp 관련 테스트 코드 -----
```

4. Unit Test

AuthCode.hpp 관련 테스트 코드

```
// ----- AuthCode.hpp 관련 테스트 코드 -----
TEST(AuthCodeTest, getCode){
    AuthCode authCode("12345",1,3);
    EXPECT_EQ(authCode.getCode(), "12345");
}

TEST(AuthCodeTest, getItemId){
    AuthCode authCode("12345",1,3);
    EXPECT_EQ(authCode.getItemId(), 1);
}

TEST(AuthCodeTest, getItemNum){
    AuthCode authCode("12345",1,3);
    EXPECT_EQ(authCode.getItemNum(), 3);
}
// ----- AuthCode.hpp 관련 테스트 코드 -----
```

AuthCodeManager.hpp 관련 테스트 코드

```
// ----- AltDVMManager.hpp 관련 테스트 코드 -----
TEST(AltDVMManagerTest, AddDVM1){
    altDVMManager.addDVM("5", 5, 6, "T");
    EXPECT_EQ(altDVMManager.getAltDVMList()[3].getId(), "5");
}

TEST(AltDVMManagerTest, AddDVM2){
    altDVMManager.addDVM("5", 5, 6, "T");
    EXPECT_EQ(altDVMManager.getAltDVMList()[3].getLocation(), make_pair(5,6));
}

TEST(AltDVMManagerTest, SelectAltDVM){
    altDVMManager.selectAltDVM(1,1);
    EXPECT_EQ(altDVMManager.getSelectedDVM(), "2");
}

TEST(AltDVMManagerTest, GetAltDVMLocation){
    altDVMManager.selectAltDVM(1,1);
    EXPECT_EQ(altDVMManager.getAltDVMLocation(), make_pair(1,3));
}

TEST(AltDVMManagerTest, Reset){
    altDVMManager.reset();
    EXPECT_EQ(altDVMManager.getAltDVMList().size(), 0);
}
// ----- AltDVMManager.hpp 관련 테스트 코드 -----
```

4. Unit Test

AltDVM.hpp 관련 테스트 코드

```
// ----- AltDVM.hpp 관련 테스트 코드 -----
TEST(AltDVMTest, GetID){
    AltDVM altDVM("2", 3, 4);
    EXPECT_EQ(altDVM.getId(), "2");
}
TEST(AltDVMTest, GetLocationX){
    AltDVM altDVM("2", 3, 4);
    EXPECT_EQ(altDVM.getLocation().first, 3);
}

TEST(AltDVMTest, GetLocationY){
    AltDVM altDVM("2", 3, 4);
    EXPECT_EQ(altDVM.getLocation().second, 4);
}
// ----- AltDVM.hpp 관련 테스트 코드 -----
```

AltDVMManager.hpp 관련 테스트 코드

```
// ----- AltDVMManager.hpp 관련 테스트 코드 -----
TEST(AltDVMManagerTest, AddDVM1){
    altDVMManager.addDVM("5", 5, 6, "T");
    EXPECT_EQ(altDVMManager.getAltDVMList()[3].getId(), "5");
}

TEST(AltDVMManagerTest, AddDVM2){
    altDVMManager.addDVM("5", 5, 6, "T");
    EXPECT_EQ(altDVMManager.getAltDVMList()[3].getLocation(), make_pair(5,6));
}

TEST(AltDVMManagerTest, SelectAltDVM){
    altDVMManager.selectAltDVM(1,1);
    EXPECT_EQ(altDVMManager.getSelectedDVM(), "2");
}

TEST(AltDVMManagerTest, GetAltDVMLocation){
    altDVMManager.selectAltDVM(1,1);
    EXPECT_EQ(altDVMManager.getAltDVMLocation(), make_pair(1,3));
}

TEST(AltDVMManagerTest, Reset){
    altDVMManager.reset();
    EXPECT_EQ(altDVMManager.getAltDVMList().size(), 0);
}
// ----- AltDVMManager.hpp 관련 테스트 코드 -----
```

4. Unit Test

DVM.hpp 관련 테스트 코드

```
// ----- DVM.hpp 관련 테스트 코드 -----
TEST(DVMTest, AskBuyOrCodeInput) {
    DVM dvm("12",1,1);
    std::istringstream fakeInput("1\n");
    std::streambuf* originalCin = std::cin.rdbuf();  // 원래 cin 백업
    std::cin.rdbuf(fakeInput.rdbuf());               // cin을 가짜로 교체

    bool result = dvm.askBuyOrCodeInput();

    EXPECT_TRUE(result);

    std::cin.rdbuf(originalCin);  // 원래 cin 복원
}
// ----- DVM.hpp 관련 테스트 코드 -----
```

4. Unit Test

Test code 실행 화면

```
j3rry3@hwangjeongseung-ui-MacBookAir build % ./testCode
Running main() from /Users/j3rry3/OOPT_development/source-code/build/_deps/googletest-src/googletest/src/gtest_main.cc
[==========] Running 33 tests from 9 test suites.
[----------] Global test environment set-up.
[----------] 4 tests from ItemTest
[ RUN      ] ItemTest.GetItemName
[       OK ] ItemTest.GetItemName (0 ms)
[ RUN      ] ItemTest.GetItemCost
[       OK ] ItemTest.GetItemCost (0 ms)
[ RUN      ] ItemTest.GetItemStock
[       OK ] ItemTest.GetItemStock (0 ms)
[ RUN      ] ItemTest.DecreaseItemStock
[       OK ] ItemTest.DecreaseItemStock (0 ms)
[----------] 4 tests from ItemTest (0 ms total)

[----------] 6 tests from ItemManagerTest
[ RUN      ] ItemManagerTest.SaveSelectedItemId
[       OK ] ItemManagerTest.SaveSelectedItemId (0 ms)
[ RUN      ] ItemManagerTest.SaveSelectedItemNum
[       OK ] ItemManagerTest.SaveSelectedItemNum (0 ms)
[ RUN      ] ItemManagerTest.IsEnoughTrue
[       OK ] ItemManagerTest.IsEnoughTrue (0 ms)
[ RUN      ] ItemManagerTest.IsEnoughFalse
[       OK ] ItemManagerTest.IsEnoughFalse (0 ms)
[ RUN      ] ItemManagerTest.MinusStock
[       OK ] ItemManagerTest.MinusStock (0 ms)
[ RUN      ] ItemManagerTest.GetPaymentAmount
[       OK ] ItemManagerTest.GetPaymentAmount (0 ms)
[----------] 6 tests from ItemManagerTest (0 ms total)
```

THANK YOU FOR WATCHING