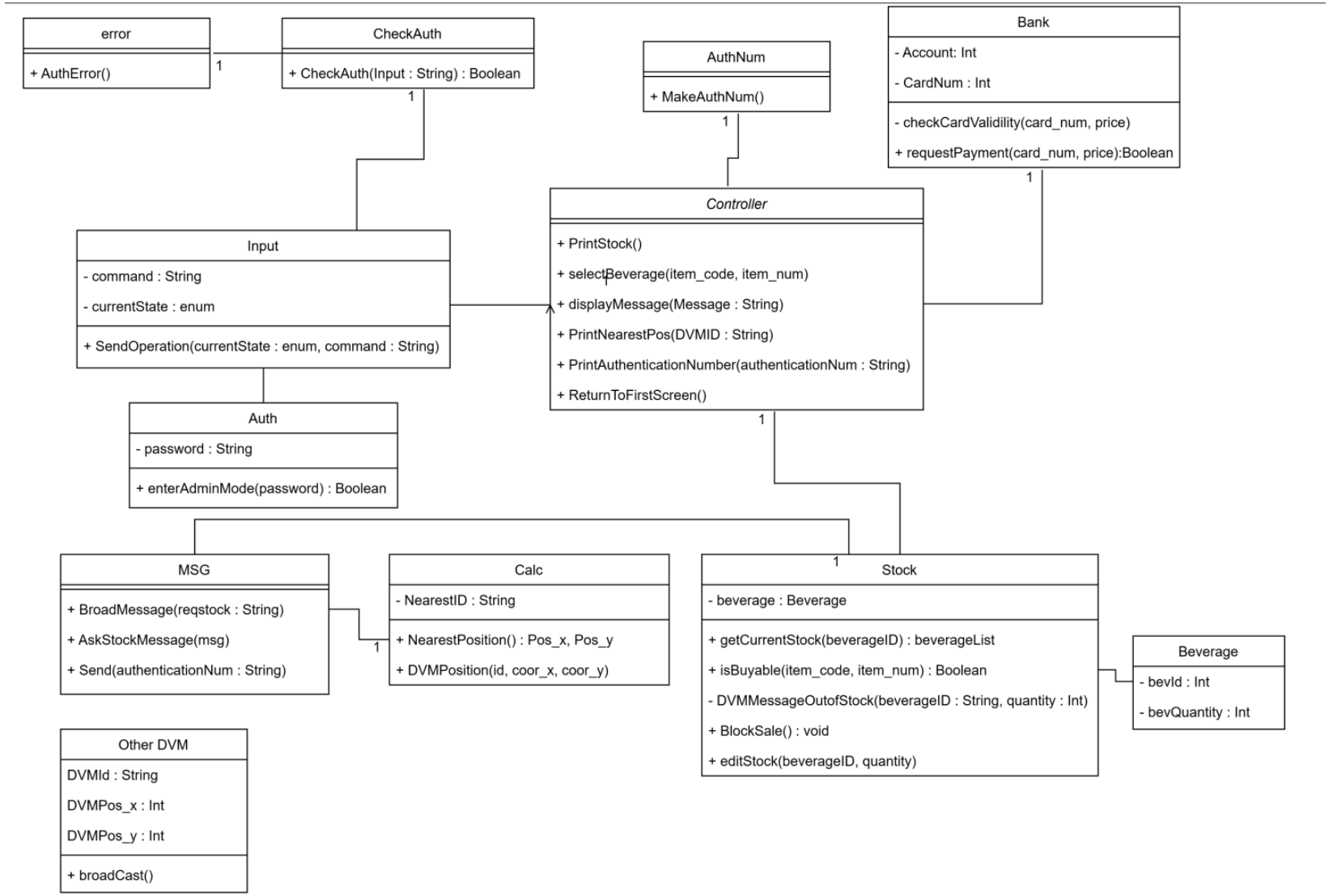
OOPT 2050 & 2060

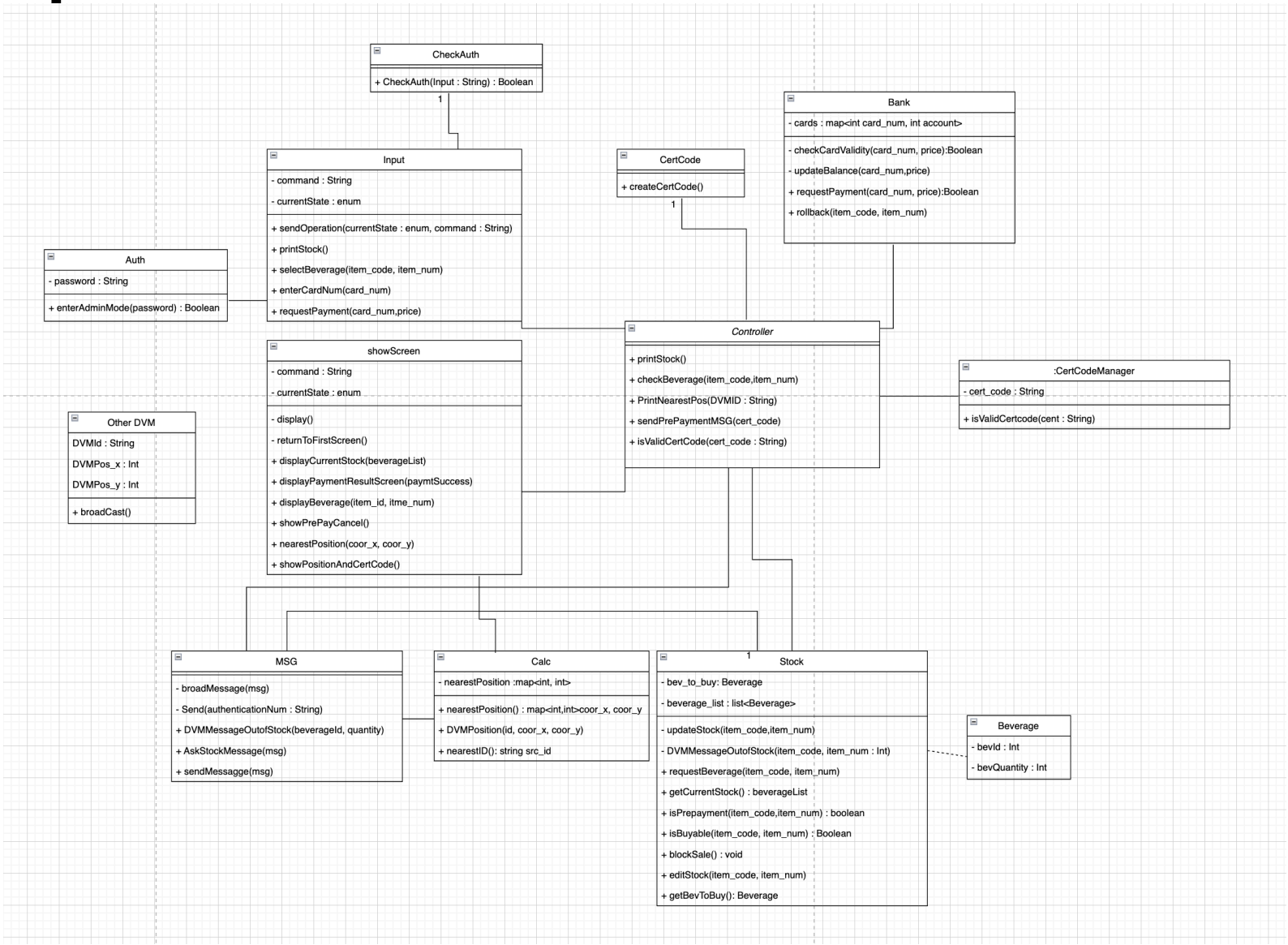
202112348 조영탁
202211282 김종혁
202312373 정석현

목차

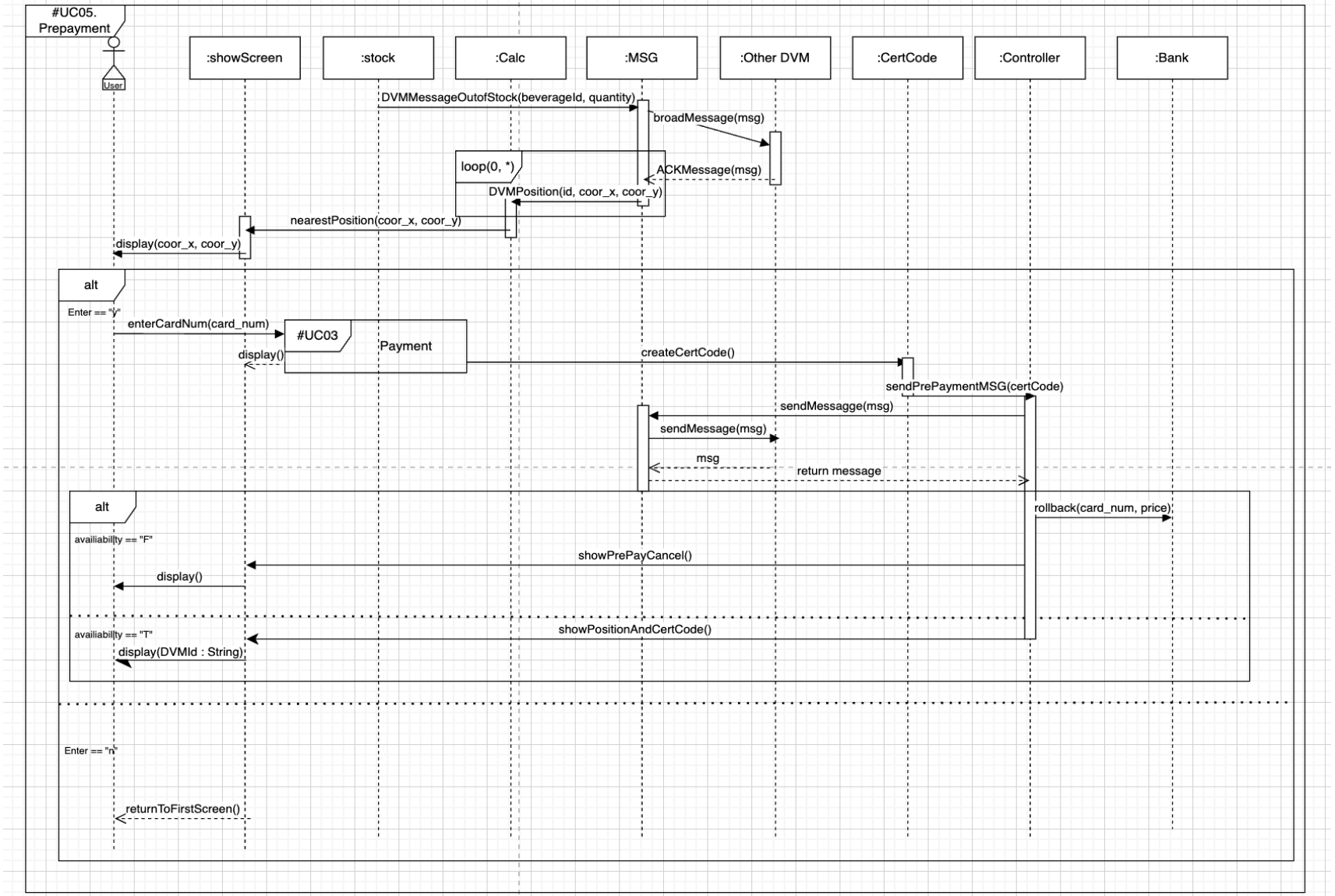
2051. Implement Class & Methods Definition



2051. Implement Class & Methods Definition



2051. Implement Class & Methods Definition



2051. Implement Class & Methods Definition

Type	Class
Name	PendingBeverage
Purpose	구매중인 음료의 정보를 저장
Overview	구매 시작 후 음료 반환 또는 선결제 완료 시점까지 현재 처리 중인 음료의 정보를 저장
Abstract Operation	#UC02 Select Beverage->isPrePayment(itme_code, item_num) 이후 Stock 클래스 Update Stock에서 폐기
Exceptional Course of Events	N/A

2051. Implement Class & Methods Definition

Type	Class
Name	Controller
Purpose	요청에 따른 분기 처리
Overview	구매 시작 후 음료 반환 또는 선결제 완료 시점까지 현재 처리 중인 음료의 정보를 저장
Abstract Operation	#UC02 Select Beverage->isPrePayment(itme_code, item_num) 이후 Stock 클래스 Update Stock에서 폐기
Exceptional Course of Events	

2051. Implement Class & Methods Definition

```
void Controller:: runShowScreenCommand(CommandType cmd){
    CommandType type = cmd;

    switch (type) {
        case CommandType::DISPLAY_CURRENT_STOCK:
            showScreen.displayCurrentStock(stock.getCurrentStock());
            break;
        case CommandType::DISPLAY_FIRST_SCREEN:
            showScreen.displayFirstScreen();
            break;
            // showScreen.displayPrepayLocation(coor_x :int ,coor_y : int);
            break;
        case CommandType::DISPLAY_ENTER_CARD_NUM:
            showScreen.displayEnterCardNum();
            break;
        case CommandType::DISPLAY_PAYMENT_RESULT_SCREEN:
            // showScreen.displayPaymentResultScreen(paymentSuccess : bool);
            break;
        case CommandType::DISPLAY_BEVERAGE:
            // showScreen.displayBeverage(item_id: int, item_num :int);
            break;
        case CommandType::DISPLAY_CERT_CODE_ENTER:
            showScreen.displayCertCodeEnter();
            break;
        case CommandType::DISPLAY_CERT_CODE_FAILED:
            showScreen.displayCertCodeFailed();
            break;
        case CommandType::DISPLAY_INVALID_ID_RANGE:
            showScreen.displayInvalidIdRange();
            break;
    }
}
```

2051. Implement Class & Methods Definition

Type	Class
Name	InMemoryDB
Purpose	테이블 정보를 저장하는 싱글톤 클래스
Overview	N/A
Cross Reference	“Select Beverage”, “Get Beverage” “Prepayment”
Input	N/A
Output	Appropriate table
Abstract Operation	재고 작업 및 거리 계산시 InMemoryDB에 저장된 Table 접근
Exceptional Course of Events	N/A

2051. Implement Class & Methods Definition

```
#pragma once
#include "PositionTable.hpp"
#include "BeverageTable.hpp"
#include "CertCodeTable.hpp"

class InMemoryDB {
private:
    InMemoryDB() = default;
    InMemoryDB(const InMemoryDB&) = delete;
    InMemoryDB& operator=(const InMemoryDB&) = delete;

public:
    static InMemoryDB& instance();
    PositionTable positionTable;
    BeverageTable beverageTable;
    CertCodeTable certcodeTable;

    void initPosition();
    void initBeverage();
};
```

2052. Implements Windows

```
displayFirstScreen
------------------------------
어서오세요.
시작하려면 's'를 입력하세요.
(선결제 인증코드 입력을 원하시면 인증코드를 입력하세요...)
------------------------------
```

Name	displayFirstScreen
Responsibility	시스템을 가동하면 나오는 화면
Type	CLI
Cross Referencses	"Select Beverage"
Notes	입력받은 값에 따라 다음 화면으로 넘어간다
Pre-Conditions	N/A
Post-Conditions	N/A

2052. Implements Windows

```
displayBeverage
------------------------------
음료수 명 : 사이다, 갯수: 5
안녕히가십시오.
------------------------------
```

Name	displayBeverage
Responsibility	사용자가 얻게 된 음료수와 수량 출력
Type	CLI
Cross Referencses	
Notes	초기화면으로 돌아감
Pre-Conditions	선결제 인증코드 입력 or 음료 구매 완료
Post-Conditions	N/A

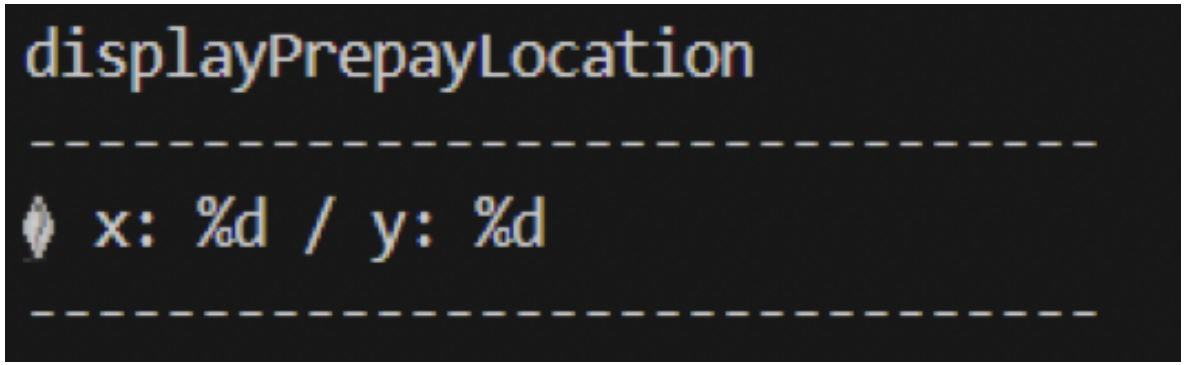
2052. Implements Windows

```
------------------------------
(1) 콜라 : 99개 (2) 사이다 : 99개 (3) 녹차 : 99개 (4) 홍차 : 99개 (5) 밀크티 : 99개
(6) 탄산수 : 99개 (7) 보리차 : 99개 (8) 캔커피 : 0개 (9) 물 : 0개 (10) 유자차 : 0개
(11) 에너지 드링크 : 0개 (12) 식혜 : 0개 (13) 딸기주스 : 0개 (14) 오렌지 주스 : 0개 (15) 포도 주스 : 0개
(16) 이온 음료 : 0개 (17) 아메리카노 : 0개 (18) 핫초코 : 0개 (19) 카페라떼 : 0개
음료 ID와 갯수를 입력해주세요. (e.g. 3 2)

------------------------------
```

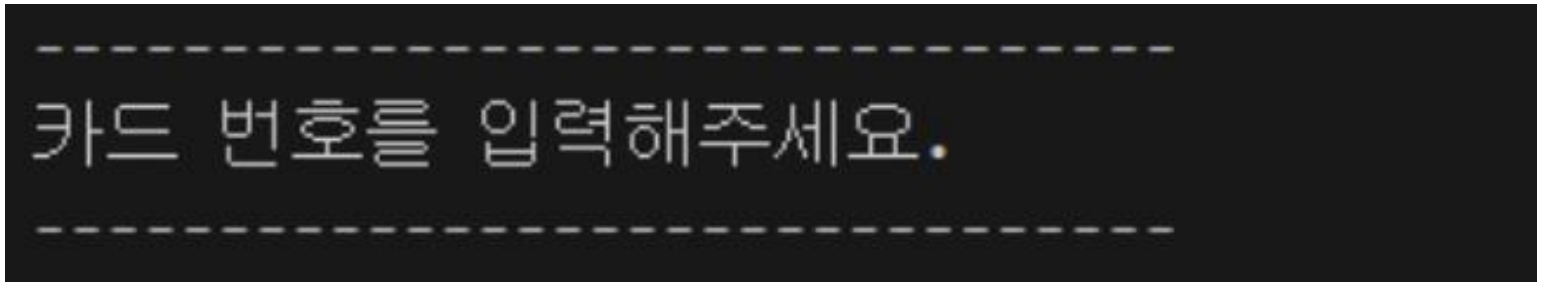
Name	displayCurrentStock
Responsibility	현재 자판기에서 보유한 음료수를 보여준다
Type	CLI
Cross Referencses	" Select Beverage"
Notes	"(음료코드) 음료명 : <음료갯수>개 "가 순서대로 출력
Pre-Conditions	초기 화면에서 's'를 눌러 진입
Post-Conditions	N/A

2052. Implements Windows



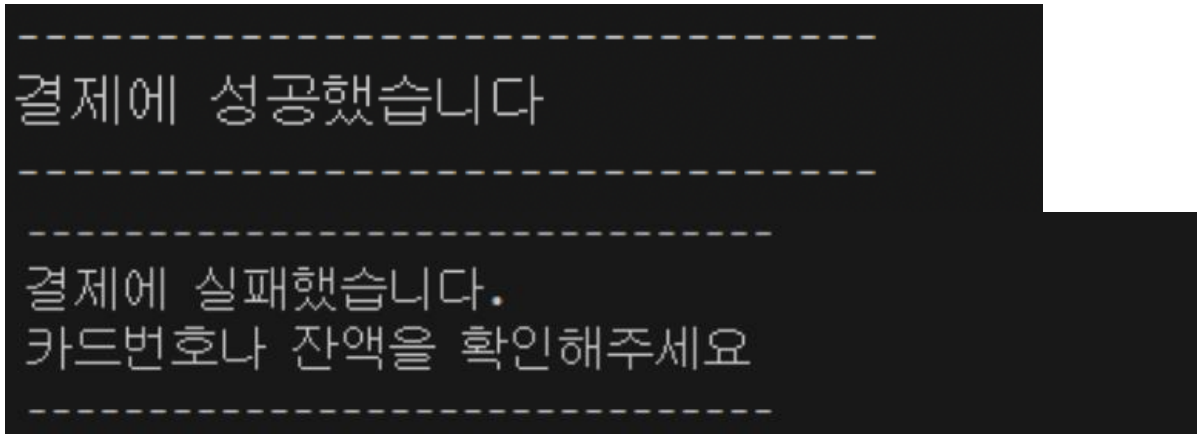
Name	displayPrePayLocation
Responsibility	선결제를 진행할 자판기의 위치가 출력된다
Type	CLI
Cross Referencses	"PrePayment"
Notes	x,y 좌표가 화면에 출력된다.
Pre-Conditions	displayCurrentStock화면에서 구매할 수 없는 음료 입력
Post-Conditions	y/n 입력에 따라 결제요구 출력 또는 초기화면으로 복귀

2052. Implements Windows



Name	displayEnterCardNum
Responsibility	카드 입력 요구 문구를 출력한다
Type	CLI
Cross Referencses	" Payment"
Notes	N/A
Pre-Conditions	유효한 음료 선택 or 선결제 수락
Post-Conditions	음료 반환 or 인증번호 출력

2052. Implements Windows



Name	DisplayPaymentResult
Responsibility	결제
Type	CLI
Cross Referencses	“Payment”
Notes	N/A
Pre-Conditions	displayEnterCardNum
Post-Conditions	이후 초기화면으로 복귀

2055. Write Unit Test Code

```cpp
// 테스트 픽스처 클래스
class BankTest : public ::testing::Test {
protected:
    Bank bank;

    void injectCard(int card_num, int balance) {
        auto ptr = reinterpret_cast<std::map<int, int>*>(
            reinterpret_cast<char*>(&bank) + offsetof(Bank, cards));
        (*ptr)[card_num] = balance;
    }

    int getBalance(int card_num) {
        auto ptr = reinterpret_cast<std::map<int, int>*>(
            reinterpret_cast<char*>(&bank) + offsetof(Bank, cards));
        return (*ptr)[card_num];
    }
};
```

```cpp
TEST_F(BankTest, ConstructorGeneratesCards) {
    // 기본 생성자 동작만 확인 (5개 생성되지만 비공개라 직접 확인 불가)
    SUCCEED();
}

TEST_F(BankTest, RequestPayment_SuccessAndFail) {
    int card_num = 1234;
    injectCard(card_num, 5000);

    // 충분한 잔고 -> true
    EXPECT_TRUE(bank.reqeustPayment(card_num, 1000));
    EXPECT_EQ(getBalance(card_num), 5000 - 1000 - 1000); // 내부 updateBalance 중복 호출됨

    // 부족한 잔고 -> false
    EXPECT_FALSE(bank.reqeustPayment(card_num, 10000));
}

TEST_F(BankTest, RollbackRestoresBalance) {
    int card_num = 5678;
    injectCard(card_num, 3000);

    EXPECT_TRUE(bank.reqeustPayment(card_num, 1000));
    EXPECT_EQ(getBalance(card_num), 3000 - 1000 - 1000);

    bank.rollback(card_num, 1000);
    EXPECT_EQ(getBalance(card_num), 3000 - 1000);  // 롤백 후 복구 확인
}
```

2055. Write Unit Test Code

```cpp
class BeverageTest : public ::testing::Test {
protected:
    Beverage bev;

    void SetUp() override {
        // 비공개 멤버 초기화를 위해 reinterpret_cast 사용
        auto ptr = reinterpret_cast<int*>(
            reinterpret_cast<char*>(&bev) + offsetof(Beverage, bevId));
        *ptr = 42;

        ptr = reinterpret_cast<int*>(
            reinterpret_cast<char*>(&bev) + offsetof(Beverage, bevQuantity));
        *ptr = 10;
    }
};
```

```cpp
TEST_F(BeverageTest, IsSameIdReturnsTrueForSameId) {
    EXPECT_TRUE(bev.isSameId(42));
}

TEST_F(BeverageTest, IsSameIdReturnsFalseForDifferentId) {
    EXPECT_FALSE(bev.isSameId(99));
}

TEST_F(BeverageTest, IsEnoughReturnsTrueWhenQuantitySufficient) {
    EXPECT_TRUE(bev.isEnough(5));
}

TEST_F(BeverageTest, IsEnoughReturnsFalseWhenQuantityInsufficient) {
    EXPECT_FALSE(bev.isEnough(15));
}

TEST_F(BeverageTest, ReduceBeverageReducesQuantity) {
    bev.reduceBeverage(3);
    EXPECT_EQ(bev.getBevQuantity(), 7);
}

TEST_F(BeverageTest, ReduceBeverageDoesNotGoNegative) {
    bev.reduceBeverage(15);
    EXPECT_EQ(bev.getBevQuantity(), 10); // 변화 없음
}

TEST_F(BeverageTest, GetBevIdReturnsCorrectId) {
    EXPECT_EQ(bev.getBevId(), 42);
}

TEST_F(BeverageTest, GetBevQuantityReturnsCorrectQuantity) {
    EXPECT_EQ(bev.getBevQuantity(), 10);
}
```

2055. Write Unit Test Code

```
class CalcTest : public ::testing::Test {
protected:
    void SetUp() override {
        // 매 테스트 전 DB 초기화
        auto& db = InMemoryDB::instance();
        db.initPosition();

        // 예시 위치 데이터 삽입
        calc.DVMPosition("A", 5, 5);
        calc.DVMPosition("B", 2, 1);
        calc.DVMPosition("C", 10, 10);
    }

    Calc calc;
};
```

```
// nearestID() 테스트
TEST_F(CalcTest, TestNearestID) {
    std::string nearest = calc.nearestID();
    EXPECT_EQ(nearest, "B"); // (2,1)이 (1,1)에 가장 가까움
}

// nearestPosition() 테스트
TEST_F(CalcTest, TestNearestPositionCoordinates) {
    std::map<int, int> pos = calc.nearestPosition();
    EXPECT_EQ(pos.begin()->first, 2);   // x
    EXPECT_EQ(pos.begin()->second, 1);  // y
}

// DVMPosition 삽입 동작 테스트
TEST_F(CalcTest, TestInsertPosition) {
    auto& db = InMemoryDB::instance();
    calc.DVMPosition("D", 7, 7);
    auto* pos = db.positionTable.find("D");

    ASSERT_NE(pos, nullptr);
    EXPECT_EQ(pos->coor_x, 7);
    EXPECT_EQ(pos->coor_y, 7);
}
```

2055. Write Unit Test Code

```cpp
// createCertCode가 호출되면 toString()으로 5자리 코드가 생성되어야 함
TEST(CertCodeTest, CreateCertCodeGeneratesFiveCharString) {
    CertCode cert;
    cert.createCertCode();
    std::string code = cert.toString();

    // 길이는 정확히 5자
    EXPECT_EQ(code.length(), 5);

    // 비어 있으면 안 됨
    EXPECT_FALSE(code.empty());

    for (char c : code) {
        EXPECT_TRUE(std::isalnum(c));
    }
}

// 여러 인스턴스가 서로 다른 값을 갖는지 테스트
TEST(CertCodeTest, DifferentInstancesProduceDifferentCodes) {
    CertCode cert1;
    CertCode cert2;

    cert1.createCertCode();
    cert2.createCertCode();

    EXPECT_NE(cert1.toString(), cert2.toString());
}
```

```cpp
// createCertCode() 여러번 호출 시 값이 바뀌는지 테스트
TEST(CertCodeTest, CreateCertCodeGeneratesNewValueEachTime) {
    CertCode cert;
    cert.createCertCode();
    std::string first = cert.toString();
    cert.createCertCode();
    std::string second = cert.toString();

    EXPECT_NE(first, second);  // 랜덤이므로 값 다름
}

// 호출 이전에는 빈 값인지 테스트
TEST(CertCodeTest, ToStringReturnsEmptyBeforeGeneration) {
    CertCode cert;
    std::string result = cert.toString();

    EXPECT_TRUE(result.empty());
}

// 생성된 코드가 유효한 charset만 사용하는지 테스트
TEST(CertCodeTest, GeneratedCodeUsesValidCharset) {
    CertCode cert;
    cert.createCertCode();
    std::string code = cert.toString();

    const std::string validChars =
        "0123456789"
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
        "abcdefghijklmnopqrstuvwxyz";

    for (char c : code) {
        EXPECT_NE(validChars.find(c), std::string::npos);  // validChars 내에 존재해야 함
    }
}
```

2060. Unit Testing

```
Test project /Users/jonnykim/CICD-team4/build
      Start  1: CertCodeTest.CreateCertCodeGeneratesFiveCharString
 1/19 Test  #1: CertCodeTest.CreateCertCodeGeneratesFiveCharString ..........   Passed    0.01 sec
      Start  2: CertCodeTest.CreateCertCodeGeneratesNewValueEachTime
 2/19 Test  #2: CertCodeTest.CreateCertCodeGeneratesNewValueEachTime ........   Passed    0.00 sec
      Start  3: CertCodeTest.ToStringReturnsEmptyBeforeGeneration
 3/19 Test  #3: CertCodeTest.ToStringReturnsEmptyBeforeGeneration ...........   Passed    0.00 sec
      Start  4: CertCodeTest.GeneratedCodeUsesValidCharset
 4/19 Test  #4: CertCodeTest.GeneratedCodeUsesValidCharset ..................   Passed    0.00 sec
      Start  5: CertCodeTest.DifferentInstancesProduceDifferentCodes
 5/19 Test  #5: CertCodeTest.DifferentInstancesProduceDifferentCodes ........   Passed    0.00 sec
      Start  6: BankTest.ConstructorGeneratesCards
 6/19 Test  #6: BankTest.ConstructorGeneratesCards ..........................   Passed    0.00 sec
      Start  7: BankTest.RequestPayment_SuccessAndFail
 7/19 Test  #7: BankTest.RequestPayment_SuccessAndFail ......................   Passed    0.00 sec
      Start  8: BankTest.RollbackRestoresBalance
 8/19 Test  #8: BankTest.RollbackRestoresBalance ............................   Passed    0.00 sec
      Start  9: BeverageTest.IsSameIdReturnsTrueForSameId
 9/19 Test  #9: BeverageTest.IsSameIdReturnsTrueForSameId ...................   Passed    0.00 sec
      Start 10: BeverageTest.IsSameIdReturnsFalseForDifferentId
10/19 Test #10: BeverageTest.IsSameIdReturnsFalseForDifferentId .............   Passed    0.00 sec
      Start 11: BeverageTest.IsEnoughReturnsTrueWhenQuantitySufficient
11/19 Test #11: BeverageTest.IsEnoughReturnsTrueWhenQuantitySufficient ......   Passed    0.00 sec
      Start 12: BeverageTest.IsEnoughReturnsFalseWhenQuantityInsufficient
12/19 Test #12: BeverageTest.IsEnoughReturnsFalseWhenQuantityInsufficient ...   Passed    0.00 sec
      Start 13: BeverageTest.ReduceBeverageReducesQuantity
13/19 Test #13: BeverageTest.ReduceBeverageReducesQuantity ..................   Passed    0.00 sec
      Start 14: BeverageTest.ReduceBeverageDoesNotGoNegative
14/19 Test #14: BeverageTest.ReduceBeverageDoesNotGoNegative ................   Passed    0.00 sec
      Start 15: BeverageTest.GetBevIdReturnsCorrectId
15/19 Test #15: BeverageTest.GetBevIdReturnsCorrectId .......................   Passed    0.00 sec
      Start 16: BeverageTest.GetBevQuantityReturnsCorrectQuantity
16/19 Test #16: BeverageTest.GetBevQuantityReturnsCorrectQuantity ...........   Passed    0.00 sec
      Start 17: CalcTest.TestNearestID
17/19 Test #17: CalcTest.TestNearestID .....................................   Passed    0.01 sec
      Start 18: CalcTest.TestNearestPositionCoordinates
18/19 Test #18: CalcTest.TestNearestPositionCoordinates .....................   Passed    0.00 sec
      Start 19: CalcTest.TestInsertPosition
19/19 Test #19: CalcTest.TestInsertPosition ................................   Passed    0.00 sec

100% tests passed, 0 tests failed out of 19
```