객체지향개발방법론

Distributed Vending Machine : MS129 Vending Machine

Stage 2050/2060
1st Cycle Development

201710560 컴퓨터공학부 정의재
201711315 컴퓨터공학부 신원세
201714164 컴퓨터공학부 박서영
201914173 컴퓨터공학부 김현웅

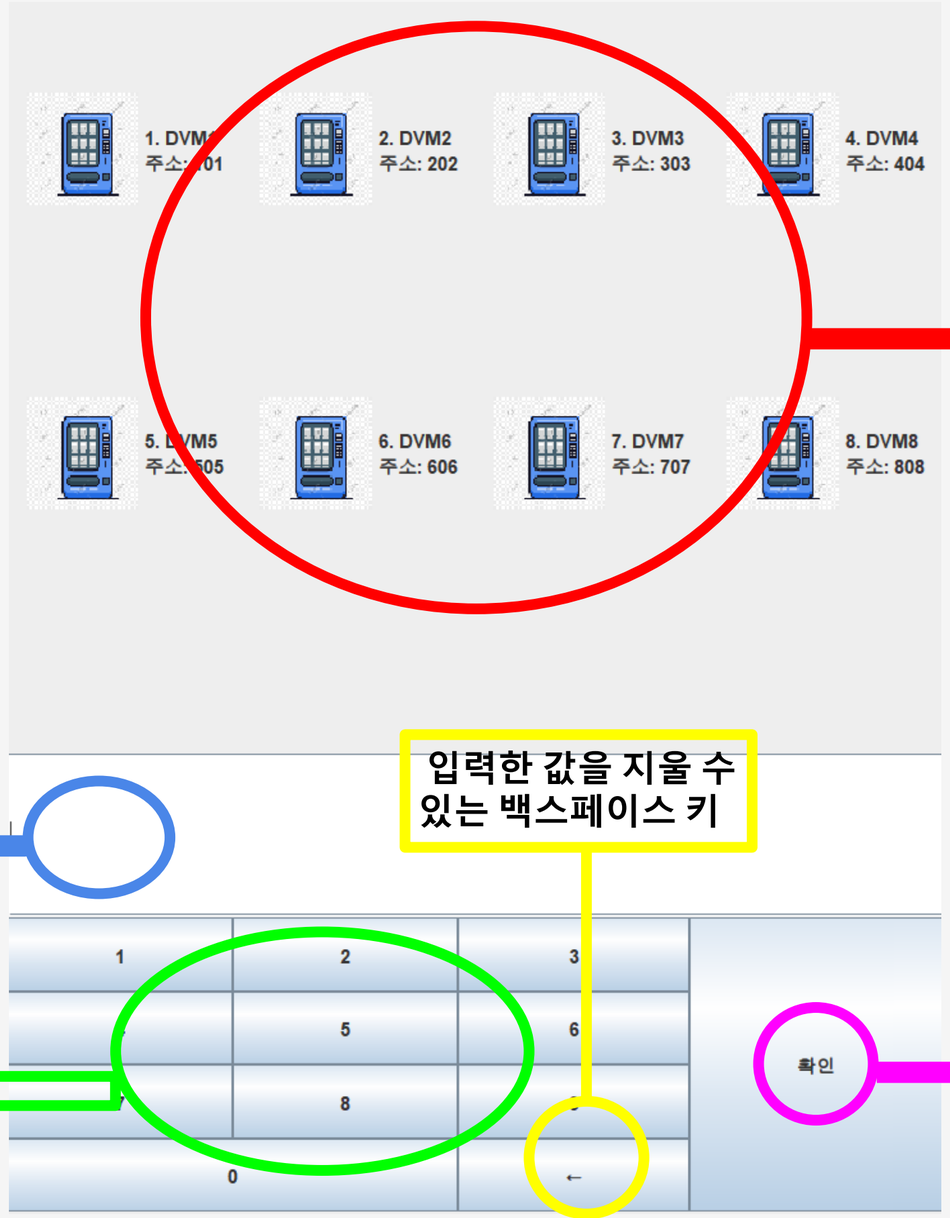
목차

1. 사용 메뉴얼

MS129 DVM은 기존 자판기의 불편함을 개선하기 위해 고안된 자판기로, DVM 간의 네트워크 통신을 통해 자판기가 가지고 있는 음료수의 재고 확인을 돕고 사용자에게 선결제의 편리함을 제공한다.

총 8대의 DVM이 존재하며, 각 DVM은 오직 7종류의 음료만을 가지고 있다. 사용자가 이용하고 있는 DVM이 원하는 음료를 가지고 있지 않다면 선결제를 통해 그 음료에 대한 계산을 미리 하고 발급받은 코드를 통해 다른 자판기에서 해당 음료를 구매할 수 있다.

1. 사용 메뉴얼



DVM의 메인 스크린으로 DVM 선택, 음료 선택, 결제 방법 선택 등의 시스템 상 진행상황과 사용자에게 알리는 메시지를 출력하는 화면

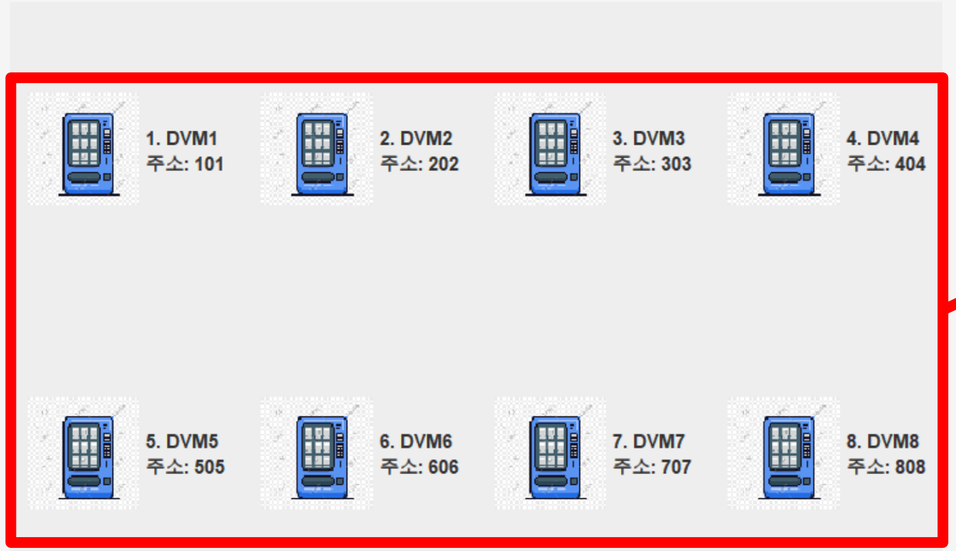
사용자가 다이얼패드를 통해 입력한 값이 나타나는 화면

입력한 값을 지울 수 있는 백스페이스 키

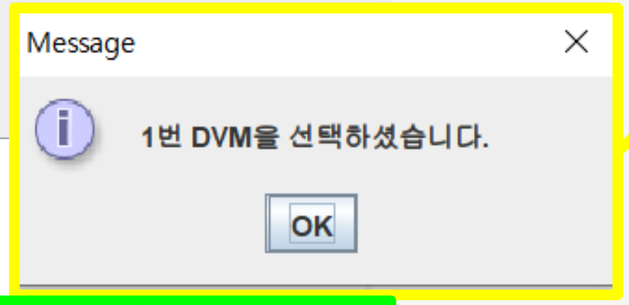
사용자가 값을 입력할 수 있는 다이얼 패드

사용자가 다이얼패드를 통해 입력한 값을 저장하는 키

1. 사용 메뉴얼

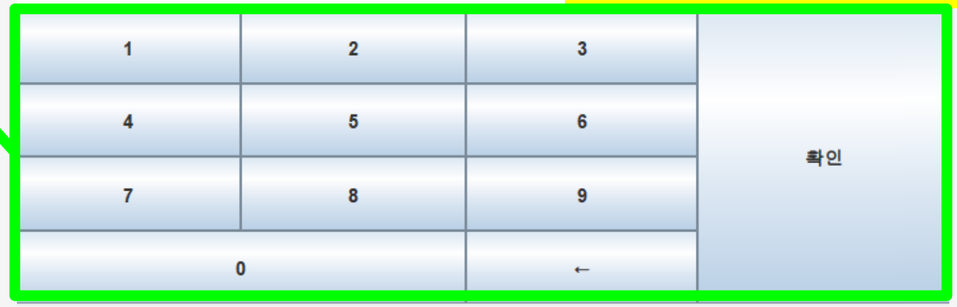


1) 총 8개의 DVM들이 DVM 이름, 주소와 함께 출력돼있다. 사용자들은 이 DVM들 중 하나의 DVM을 선택하여야 한다.



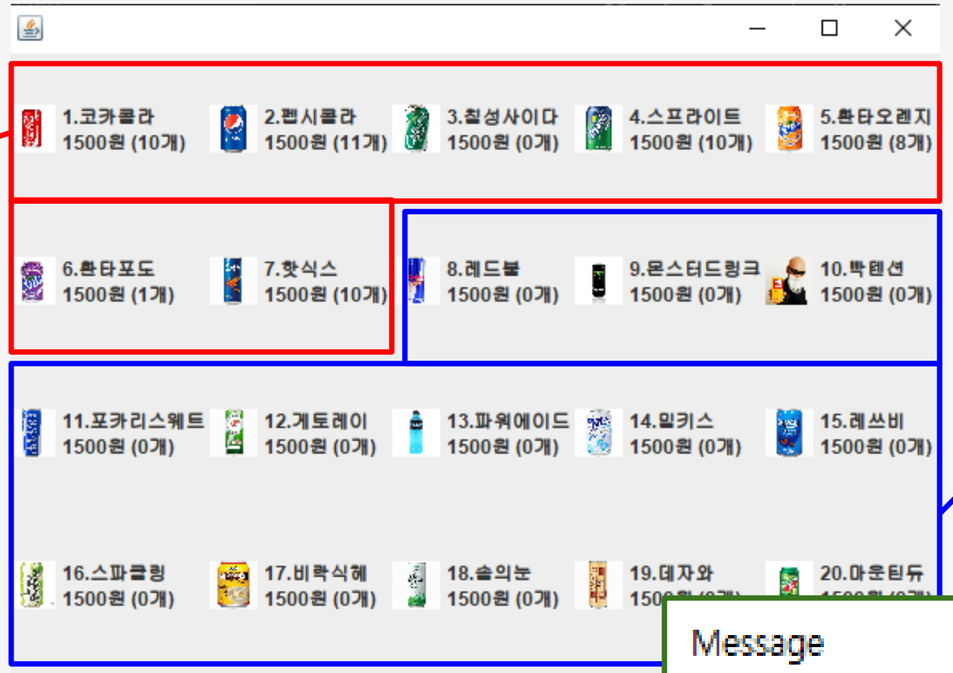
3) 사용자가 어떤 DVM을 선택하였는지 DVM의 번호와 함께 출력해주는 메시지

2) 사용자가 DVM을 선택할 수 있는 키패드이다. 사용자는 키패드를 터치하여 번호를 입력할 수 있다. 번호 입력후 확인을 누른다.

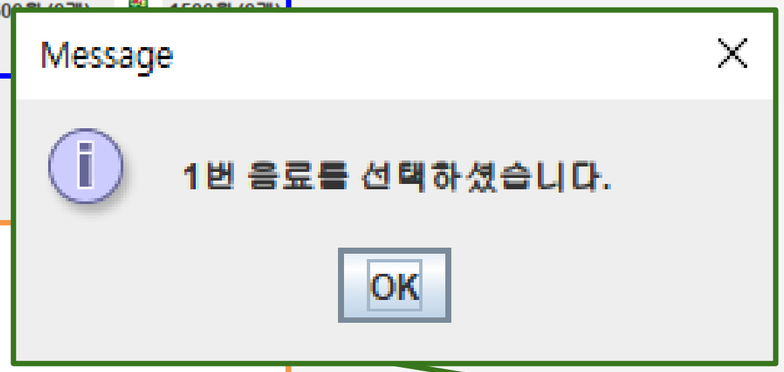


1. 사용 메뉴얼

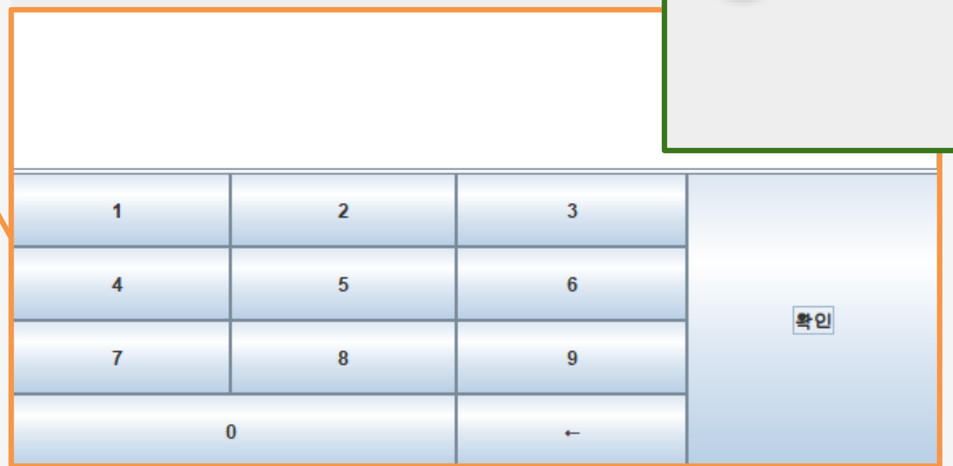
1)1번 - 7번
현재 DVM
판매메뉴 제공



1)8번 - 20번 까지
현재 DVM 에서
판매하지 않는 메뉴제공
: 선택 가능



2) Dial Button 클릭으로
음료 번호 버튼 선택 후, 확인 버튼 클릭

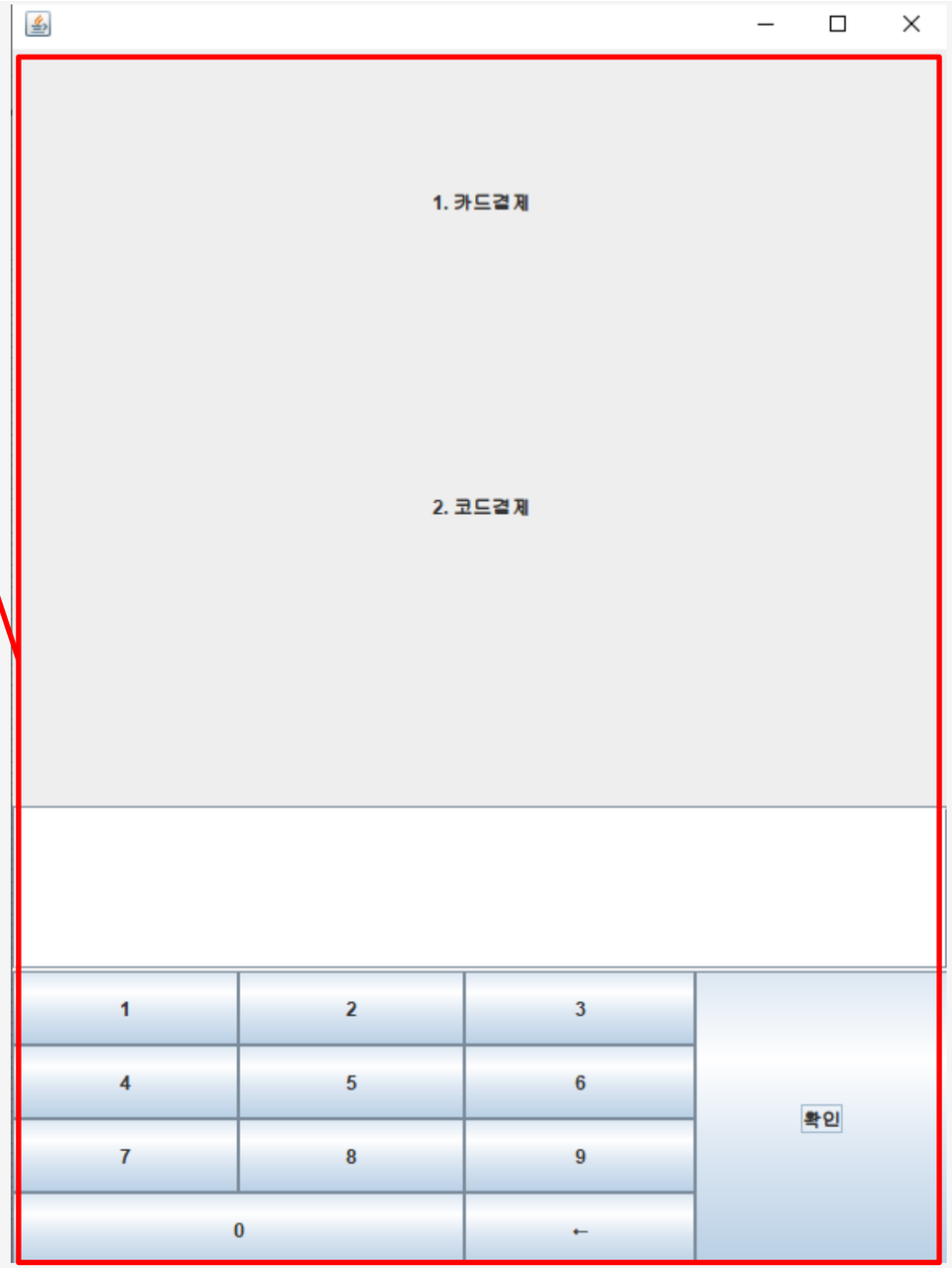


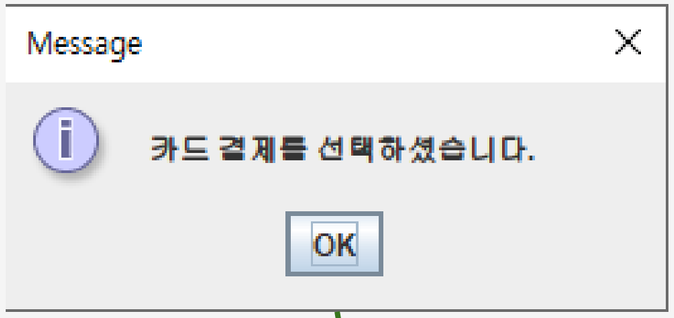
3) 음료 선택 시, 팝업
OK 버튼 클릭-> 다음 화면 이동

1. 사용 메뉴얼

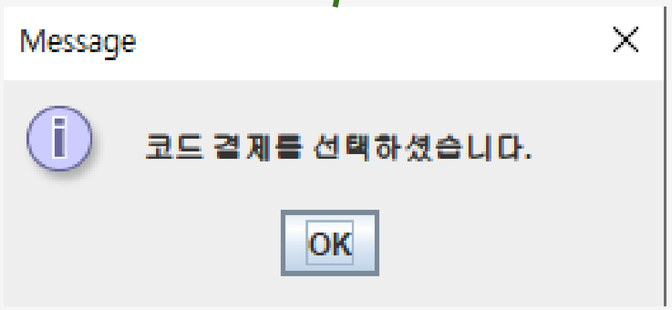
1)결제 방법 메뉴
2가지 제공

: Dial Button사용하여,
1 또는 2 버튼 클릭 후
확인 버튼 클릭





2) 결제 방법 선택 후
팝업 발생 시,
OK 버튼 클릭 -> 다음화면 이동

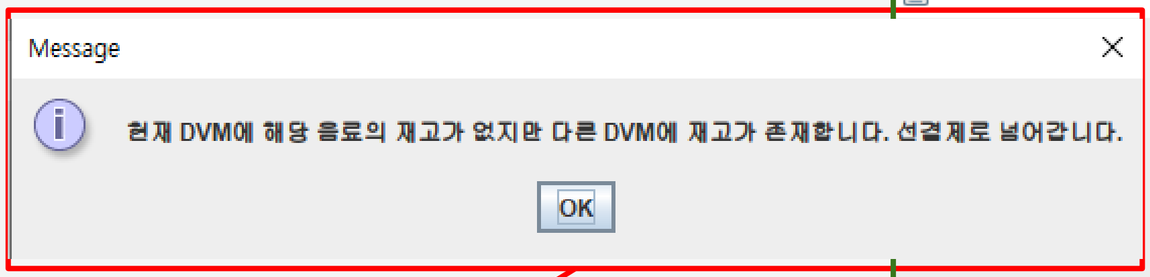


1. 사용 메뉴얼

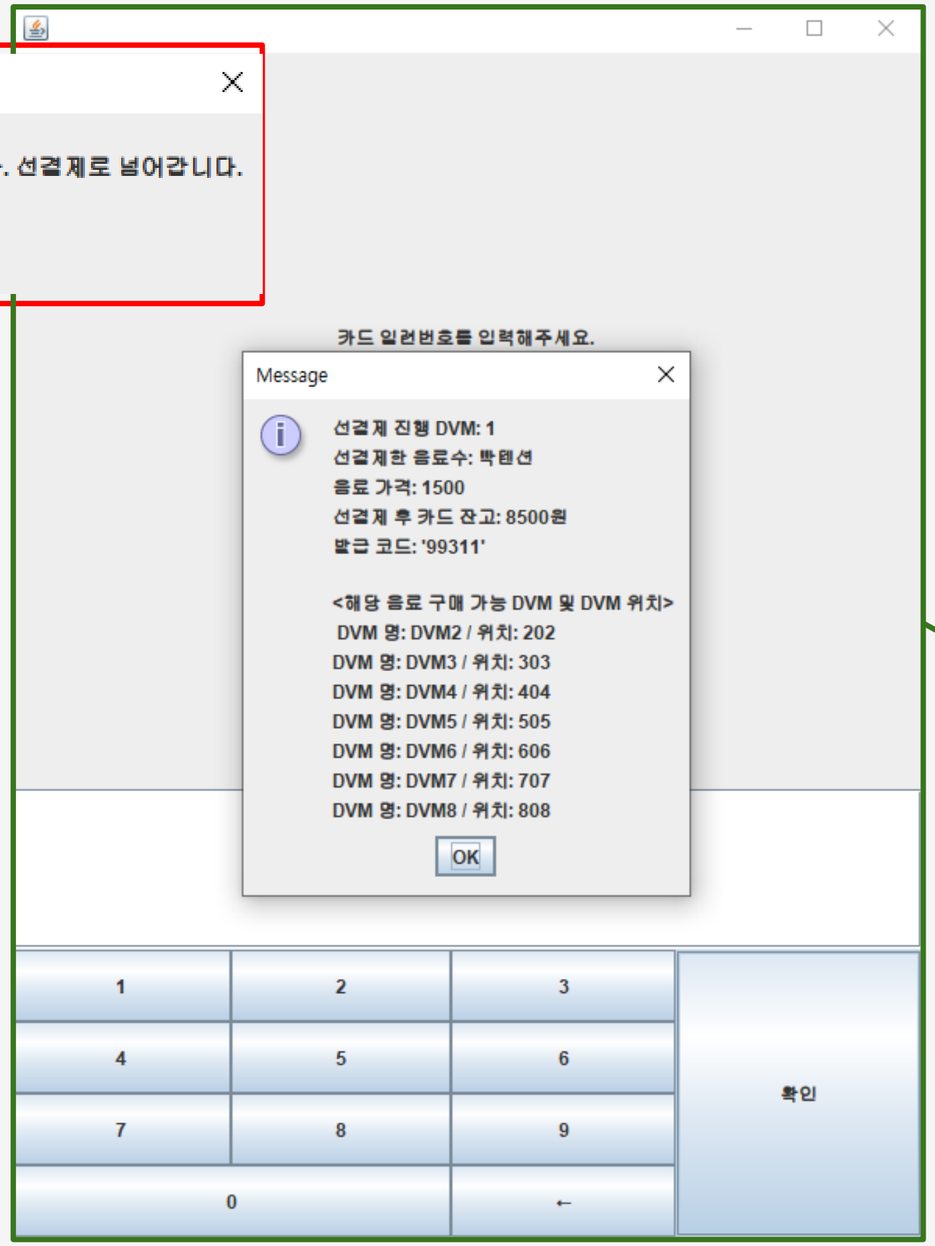
Dial Button 사용하여
결제할 카드의 번호 버튼 선택 후,
확인 버튼 클릭

결제 가능 카드번호
: 12341234 (잔액 10000원)
: 11111111 (잔액 0원)
: 10000000 (잔액 10000원)

1. 사용 메뉴얼

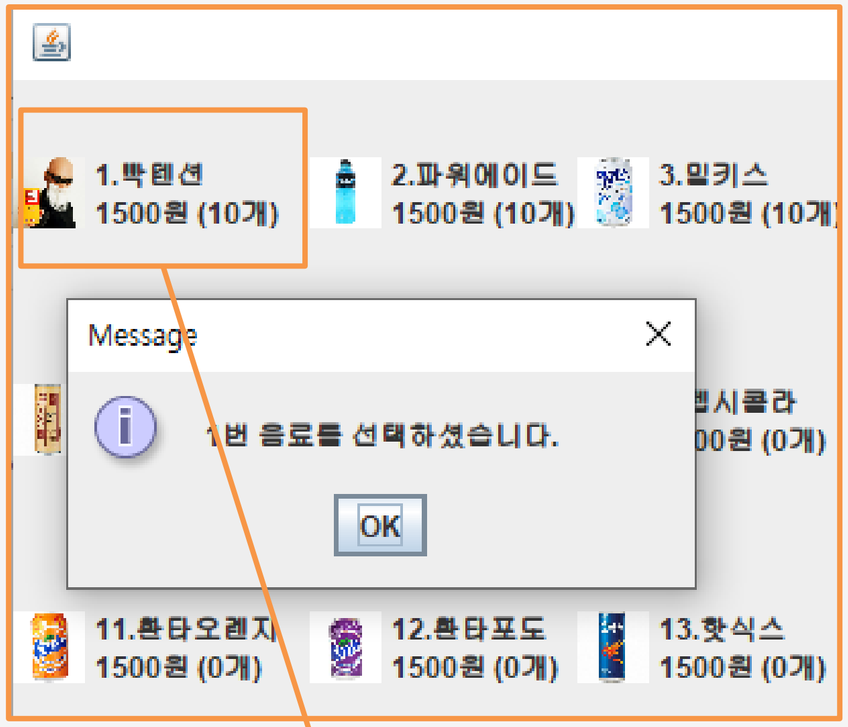


1) (8~20)번 음료를 선택시, 네트워크를 통해 다른 DVM에 재고 확인 후 재고가 있다면 선결제로 넘어감

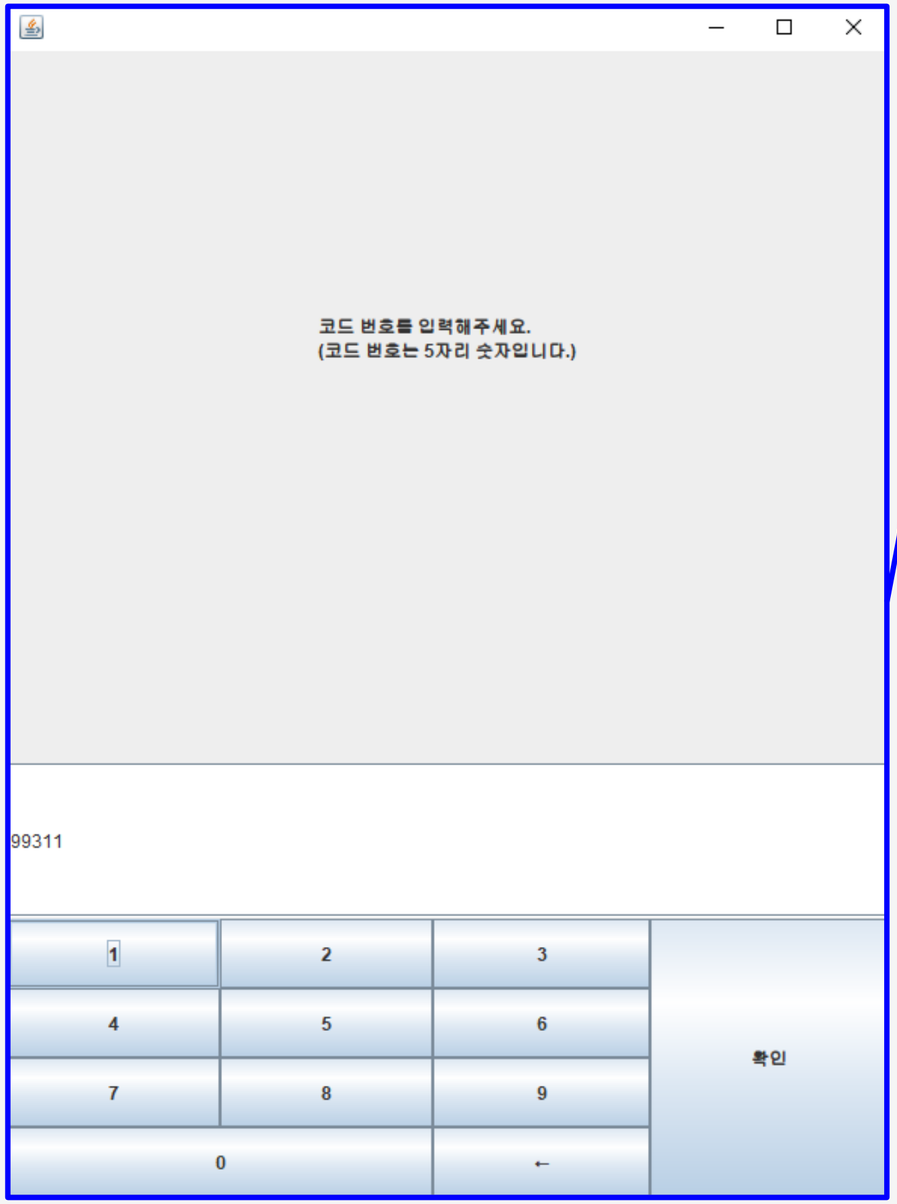


2) 카드결제 진행 후 발급된 인증코드와 네트워크를 통해 구매가능한 DVM들의 위치 출력

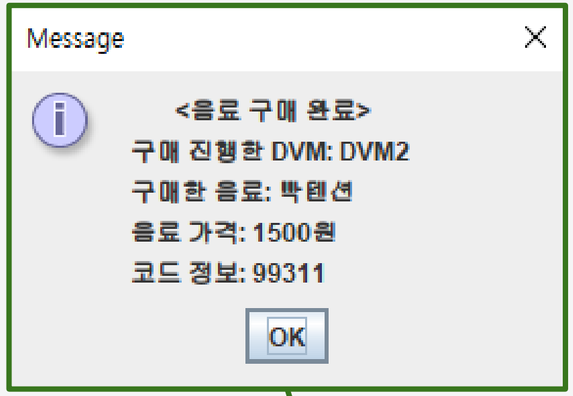
1. 사용 메뉴얼



1) 위치정보를 바탕으로 DVM2로 이동해 선결제 했던 음료수를 찾음



2)선결제때 발급받았던 코드 '99311'를 입력함



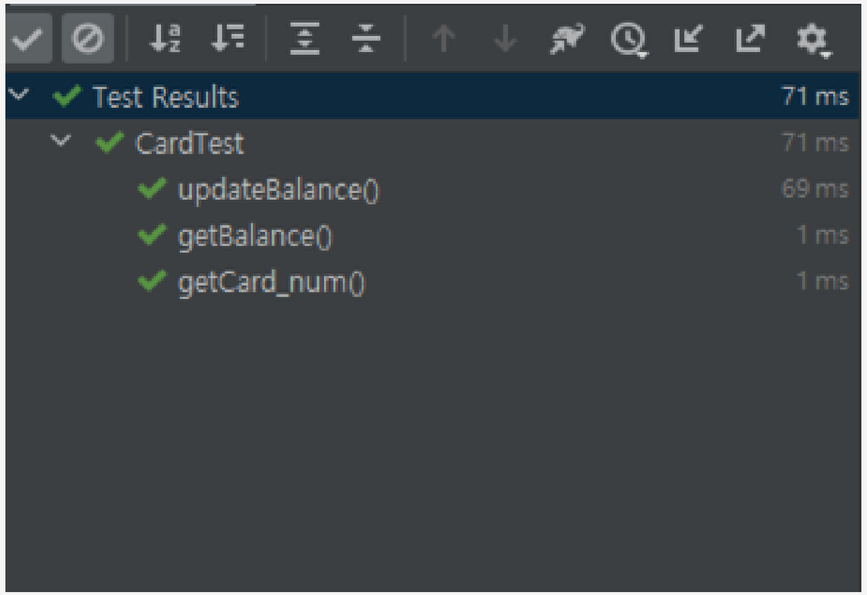
3) 성공적으로 구매 완료, 사용된 코드는 삭제됨

2. 데모 영상

https://youtu.be/fuhzgaLJMx4

3. 유닛 테스트

1. Card

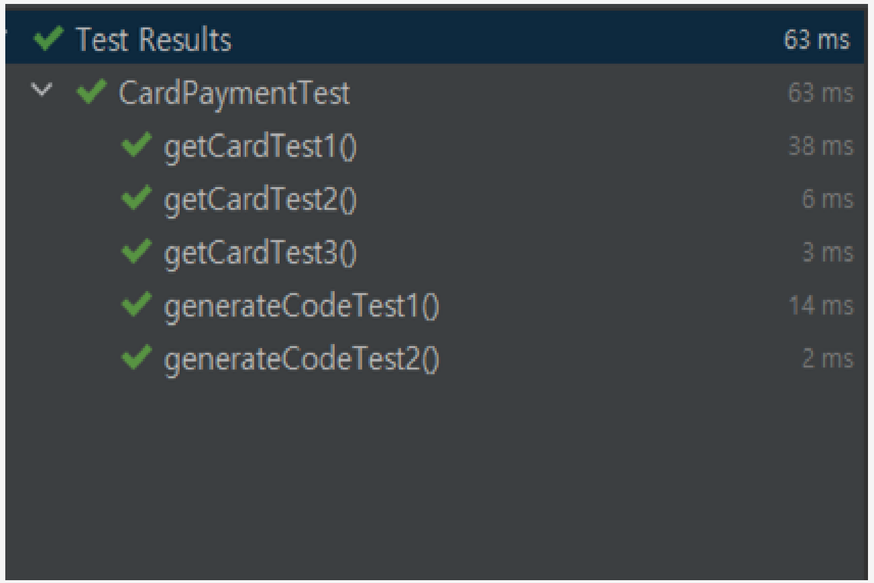



```java
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class CardTest {

    Card card1 = new Card( card_num: 0,  balance: 11000);
    Card card2 = new Card( card_num: 1, balance: 12000);
    Card card3 = new Card( card_num: 2, balance: 13000);
    Card card4 = new Card( card_num: 10, balance: 14000);
    Card card5 = new Card( card_num: 1000, balance: 15000);
    Card card6 = new Card( card_num: 10000, balance: 16000);
    Card card7 = new Card( card_num: 100000, balance: 17000);
    Card card8 = new Card( card_num: 1000000, balance: 18000);
    Card card9 = new Card( card_num: 10000000, balance: 19000);
    Card card10 = new Card( card_num: 100000000, balance: 20000);


    @Test
    void getCard_num() { // 카드 번호가 올바르게 전달되는가
        assertEquals( expected: 0,card1.getCard_num());

        assertEquals( expected: 1,card2.getCard_num());

        assertEquals( expected: 2, card3.getCard_num());

        assertEquals( expected: 10, card4.getCard_num());

        assertEquals( expected: 1000, card5.getCard_num());

        assertEquals( expected: 10000, card6.getCard_num());

```

```java
@Test
void updateBalance() {
    card1.updateBalance( price: 1500);
    assertEquals( expected: 9500,card1.getBalance());

    card2.updateBalance( price: 1500);
    assertEquals( expected: 10500,card2.getBalance());

    card3.updateBalance( price: 1500);
    assertEquals( expected: 11500, card3.getBalance());

    card4.updateBalance( price: 1500);
    assertEquals( expected: 12500, card4.getBalance());

    card5.updateBalance( price: 1500);
    assertEquals( expected: 13500, card5.getBalance());

    card6.updateBalance( price: 1500);
    assertEquals( expected: 14500, card6.getBalance());

    card7.updateBalance( price: 1500);
    assertEquals( expected: 15500, card7.getBalance());

    card8.updateBalance( price: 1500);
    assertEquals( expected: 16500, card8.getBalance());

    card9.updateBalance( price: 1500);
    assertEquals( expected: 17500, card9.getBalance());

    card10.updateBalance( price: 1500);
    assertEquals( expected: 18500, card10.getBalance());
}
```

3. 유닛 테스트

2. CardPayment




```java
@Test
void generateCodeTest1() {
    Drink drink_info = new Drink( name: "음료수1",  price: 1000,  stock: 1,  imgURL: null);
    int generatedCodeNum = (int) (Math.random() * (99999 - 10000 + 1)) + 10000;
    Code generatedCode = new Code(generatedCodeNum, drink_info);
    assertTrue( condition: generatedCodeNum > 10000);
    assertTrue( condition: generatedCodeNum < 100000);
}
```

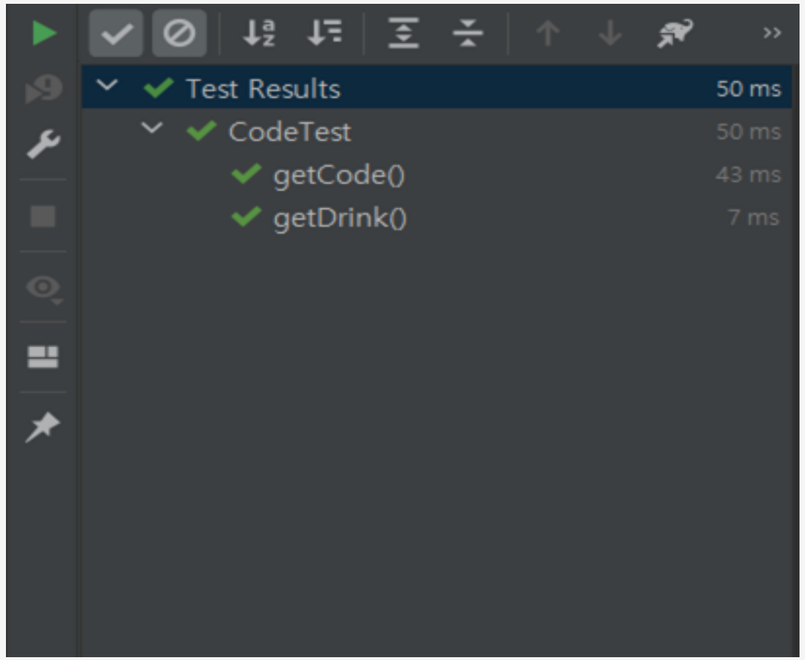
```java
@Test
void generateCodeTest2() {
    Drink drink_info = new Drink( name: "음료수1",  price: 1000,  stock: 1,  imgURL: null);
    int generatedCodeNum = (int) (Math.random() * (99999 - 10000 + 1)) + 10000;
    Code generatedCode = new Code(generatedCodeNum, drink_info);
    assertEquals(Code.class, generatedCode.getClass());
}
```

```java
@Test
void getCardTest1() {
    Card card1 = new Card( card_num: 12341234,  balance: 10000);
    Card card2 = new Card( card_num: 11111111,  balance: 0);

    ArrayList<Card> tempList = new ArrayList<>();
    tempList.add(card1);
    tempList.add(card2);
    ArrayList<Card> basicCardList = tempList;
    assertTrue( condition: basicCardList.size() == 2);

}
```

```java
@Test
void getCardTest3() {
    ArrayList<Card> basicCardList;
    int[] basicCardNameList = {12341234, 11111111, 10000000};
    Card card1 = new Card(basicCardNameList[0],  balance: 10000);
    Card card2 = new Card(basicCardNameList[1],  balance: 0);
    Card card3 = new Card(basicCardNameList[2],  balance: 10000);
    ArrayList<Card> tempList = new ArrayList<>();
    tempList.add(card1);
    tempList.add(card2);
    tempList.add(card3);
    basicCardList = tempList;           //미리 초기화 되어있는 basicCardList


    int card_num = 12341234;            //Customer 가 입력한 카드번호

    Card myCard = null;
    for (Card card : basicCardList) {
        if (card.getCard_num() == card_num)
            myCard = card;

    }
    assertEquals(myCard.getClass(), Card.class);
}
```

3. 유닛 테스트

3. Code

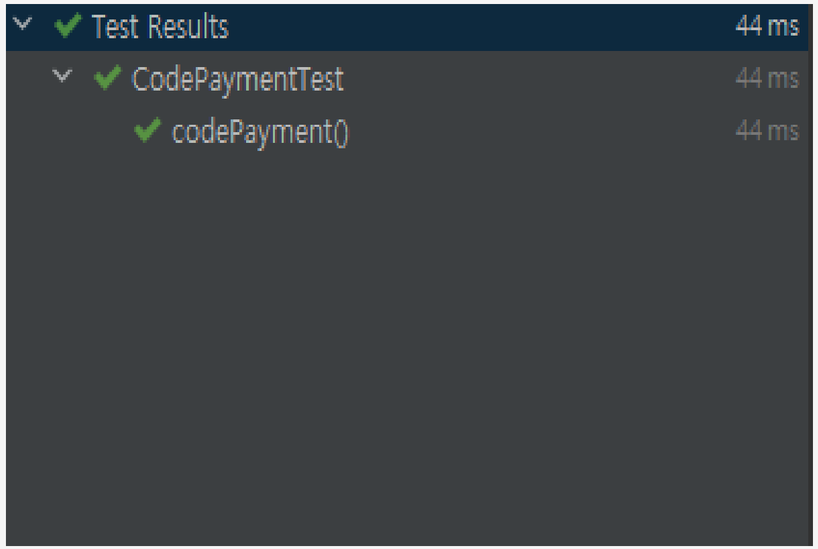



```java
@Test
void getCode() { // 코드가 맞게 반환되는지
    assertEquals( expected: 11111, code1.getCode());

    assertEquals( expected: 22222, code2.getCode());

    assertEquals( expected: 33333, code3.getCode());

    assertEquals( expected: 44444, code4.getCode());

    assertEquals( expected: 55555, code5.getCode());

    assertEquals( expected: 66666, code6.getCode());

    assertEquals( expected: 77777, code7.getCode());

    assertEquals( expected: 88888, code8.getCode());

    assertEquals( expected: 99999, code9.getCode());

    assertEquals( expected: 00000, code10.getCode());
}
```

```java
@Test
void getDrink() { // 음료 정보가 맞게 반환되는지

    assertEquals(drink1, code1.getDrink());

    assertEquals(drink2, code2.getDrink());

    assertEquals(drink3, code3.getDrink());

    assertEquals(drink4, code4.getDrink());

    assertEquals(drink5, code5.getDrink());

    assertEquals(drink6, code6.getDrink());

    assertEquals(drink7, code7.getDrink());

    assertEquals(drink8, code8.getDrink());

    assertEquals(drink9, code9.getDrink());

    assertEquals(drink10, code10.getDrink());
}
```

3. 유닛 테스트

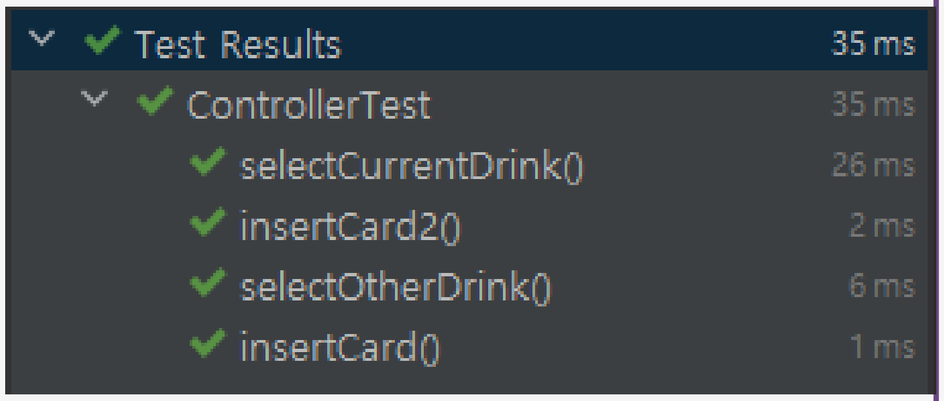
4. CodePayment




```java
@Test
void codePayment() { // drink_info를 반환
    assertEquals(drink_info1, code_info1.getDrink());

    assertEquals(drink_info2, code_info2.getDrink());

    assertEquals(drink_info3, code_info3.getDrink());

    assertEquals(drink_info4, code_info4.getDrink());

    assertEquals(drink_info5, code_info5.getDrink());

    assertEquals(drink_info6, code_info6.getDrink());

    assertEquals(drink_info7, code_info7.getDrink());

    assertEquals(drink_info8, code_info8.getDrink());

    assertEquals(drink_info9, code_info9.getDrink());

    assertEquals(drink_info10, code_info10.getDrink());
}
```

3. 유닛 테스트

5. Controller



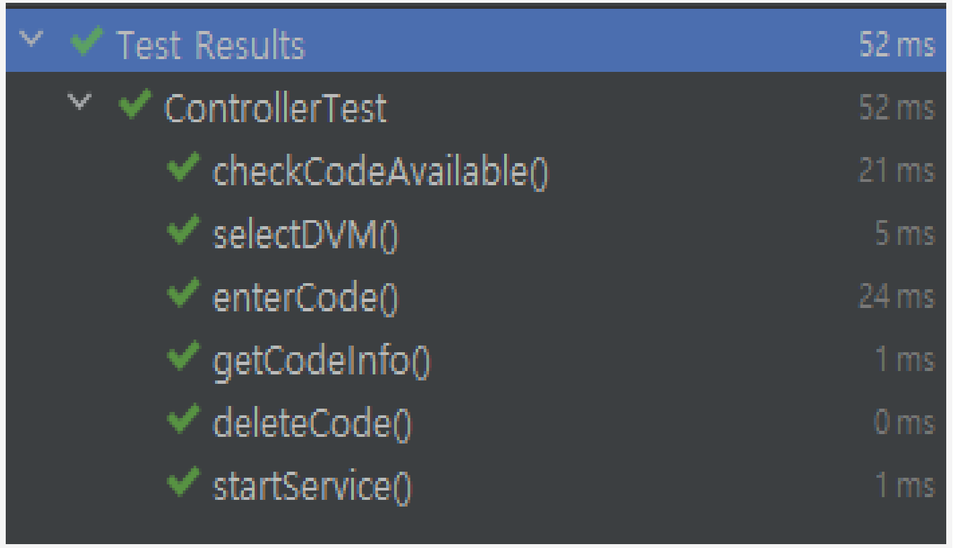

```java
@Test
void selectCurrentDrink() {
    otherDVMs = new OtherDVMs();
    DVM testDVM = otherDVMs.getDVM( index: 0);
    boolean current_stock;
    int currentDVMIndex = 0;
    int testDialNum = 3;
    if(testDVM.getDrink_list().get(testDialNum).getStock() > 0){
        current_stock = true;
    }
    else
        current_stock = false;
    assertEquals( expected: true,current_stock);
}
```

```java
@Test
void selectOtherDrink() {
    otherDVMs = new OtherDVMs();

    DVM currentDVM2 = otherDVMs.getDVM( index: 0);
    Drink selected_drink = currentDVM2.getDrink_list().get(2);   // 1번 DVM 칠성사이다
    ArrayList<DVM> accessible_DVM_list = otherDVMs.checkOtherDVMsStock(selected_drink, currentDVM2);
    assertEquals( expected: 2,accessible_DVM_list.size());

    DVM currentDVM3 = otherDVMs.getDVM( index: 1);
    Drink selected_drink2 = currentDVM3.getDrink_list().get(12);   // 2번 DVM 핫식스
    ArrayList<DVM> accessible_DVM_list2 = otherDVMs.checkOtherDVMsStock(selected_drink2, currentDVM3);
    assertEquals( expected: 2,accessible_DVM_list2.size());
    DVM testDVM2 = otherDVMs.getDVM( index: 1);
    DVM testDVM3 = otherDVMs.getDVM( index: 2);
    DVM testDVM6 = otherDVMs.getDVM( index: 5);
    DVM testDVM8 = otherDVMs.getDVM( index: 7);
    ArrayList<DVM> access_test_list = new ArrayList<>(Arrays.asList(testDVM2,testDVM3
            ,testDVM6,testDVM8));
    currentDVMIndex = 0;
    int testDialNum = 10; //1번 DVM 빡탠션
    assertTrue( condition: testDialNum>=8 && testDialNum <=20);
    DVM currentDVM = otherDVMs.getDVM(currentDVMIndex);
    selected_drink = currentDVM.getDrink_list().get(testDialNum-1);
    System.out.println(selected_drink.getName());
    accessible_DVM_list = otherDVMs.checkOtherDVMsStock(selected_drink,currentDVM);
    assertEquals(access_test_list,accessible_DVM_list);
}
```

```java
@Test
void insertCard() {
    int testCardNum = 12341234;
    boolean isPrepayment = false;
    boolean test_card_avail = cardPayment.getCard_available(testCardNum);
    assertEquals( expected: true, test_card_avail);
    Card testCard = cardPayment.getCard(testCardNum);
    assertEquals( expected: 10000,testCard.getBalance());
}
@Test
void insertCard2() {
    otherDVMs = new OtherDVMs();
    int testCardNum = 12341234;
    boolean isPrepayment = true;
    Card testCard = cardPayment.getCard(testCardNum);
    Drink testDrink = new Drink( name: "코카콜라", price: 1000, stock: 10, imgURL: "");
    Code code = cardPayment.generateCode(testDrink);
    assertNotNull(code.getCode());
}
```

3. 유닛 테스트

5. Controller




```java
@Test
void enterCode() {
    otherDVMs = new OtherDVMs();
    currentDVMIndex = 1;
    Code testCode = new Code( code: 12345,new Drink( name: "코카콜라", price: 1000, stock: 10, imgURL: ""));
    code_list.add(testCode);
    Drink testDrink = codePayment.codePayment(testCode);
    assertEquals( expected: "코카콜라",testDrink.getName());
    String testResult = otherDVMs.requestDrink(testDrink, currentDVMIndex);
    assertNotEquals( unexpected: "",testResult);
}
```

```java
@Test
void getCodeInfo() {
    code_list= new ArrayList<Code>();
    Drink testDrink = new Drink( name: "코카콜라", price: 1000, stock: 10, imgURL: "");
    Code testCode = new Code( code: 12345,testDrink);
    Code testCode2 = new Code( code: 54321,testDrink);
    code_list.add(testCode); code_list.add(testCode2);
    int testCodeNum = 12345;
    for(Code code : code_list){
        if(code.getCode() == testCodeNum) {
            assertEquals( expected: true, (code.getCode() == testCodeNum));
        }
    }
}
```

```java
@Test
void deleteCode() {
    code_list= new ArrayList<Code>();
    Drink testDrink = new Drink( name: "코카콜라", price: 1000, stock: 10, imgURL: "");
    Code testCode = new Code( code: 12345,testDrink);
    Code testCode2 = new Code( code: 54321,testDrink);
    code_list.add(testCode);
    code_list.add(testCode2);
    int testCodeNum = 12345;
    for (Code code : code_list) {
        if(testCodeNum == code.getCode()){
            code_list.remove(code);
        }
    }
    assertEquals( expected: 1,code_list.size());
}
```

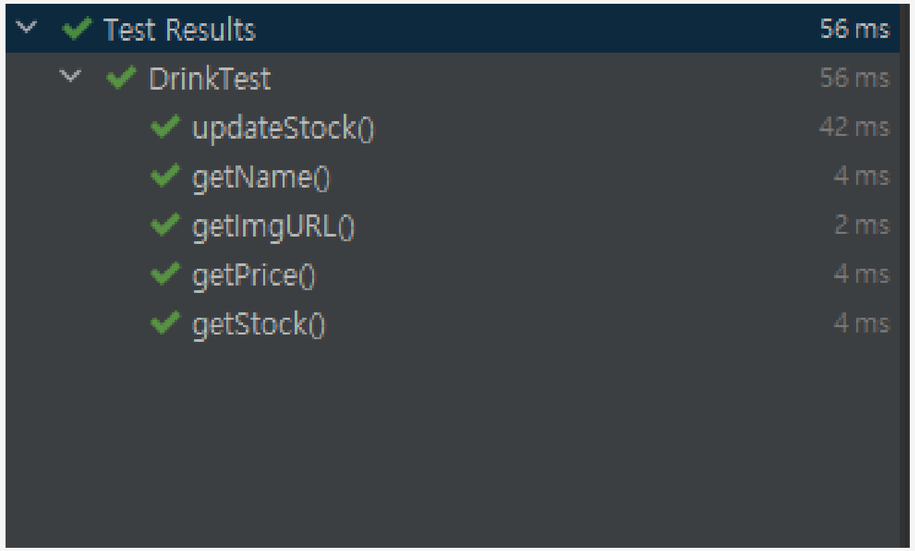
```java
@Test
void checkCodeAvailable() {
    Drink testDrink = new Drink( name: "코카콜라", price: 1000, stock: 10, imgURL: "");
    Code testCode = new Code( code: 12345,testDrink);
    code_list.add(testCode);

    int testCodeNum = 12345;
    for (Code code : code_list) {
        assertEquals(testCodeNum,code.getCode());
    }
}
```

```java
@Test
void selectDVM() {
    int testNum = 1;
    currentDVMIndex = testNum-1;
    otherDVMs = new OtherDVMs();
    DVM testDVM = otherDVMs.getDVM(currentDVMIndex);
    System.out.println(testDVM.getId());
    assertEquals( expected: 0,testDVM.getId());
}
```

```java
@Test
void startService() {
    otherDVMs = new OtherDVMs();
    int testDVMListSize = 8;

    ArrayList<DVM> dvmList = otherDVMs.getDVMList();
    assertEquals(testDVMListSize,dvmList.size());
    int testAddress = 101;
    for(int i=0; i<dvmList.size(); i++){
        ArrayList<Integer> std = new ArrayList<Integer>();
        assertEquals(i,dvmList.get(i).getId());
        std.add(dvmList.get(i).getId());
        assertEquals(testAddress,dvmList.get(i).getAddress());
        testAddress += 101;
        std.add(dvmList.get(i).getAddress());
    }
}
```

3. 유닛 테스트

6. Drink




```java
@Test
void getName() {
    assertEquals( expected: "코카콜라", drink1.getName());

    assertEquals( expected: "펩시콜라", drink2.getName());

    assertEquals( expected: "칠성사이다", drink3.getName());

    assertEquals( expected: "스프라이트", drink4.getName());

    assertEquals( expected: "환타오렌지", drink5.getName());

    assertEquals( expected: "환타포도", drink6.getName());

    assertEquals( expected: "핫식스", drink7.getName());

    assertEquals( expected: "레드불", drink8.getName());

    assertEquals( expected: "몬스터드링크", drink9.getName());

    assertEquals( expected: "빡텐션", drink10.getName());
}
```

```java
@Test
void getPrice() {
    assertEquals( expected: 1500, drink1.getPrice());

    assertEquals( expected: 1500, drink2.getPrice());

    assertEquals( expected: 1500, drink3.getPrice());

    assertEquals( expected: 1500, drink4.getPrice());

    assertEquals( expected: 1500, drink5.getPrice());

    assertEquals( expected: 1500, drink6.getPrice());

    assertEquals( expected: 1500, drink7.getPrice());

    assertEquals( expected: 1500, drink8.getPrice());

    assertEquals( expected: 1500, drink9.getPrice());

    assertEquals( expected: 1500, drink10.getPrice());
}
```

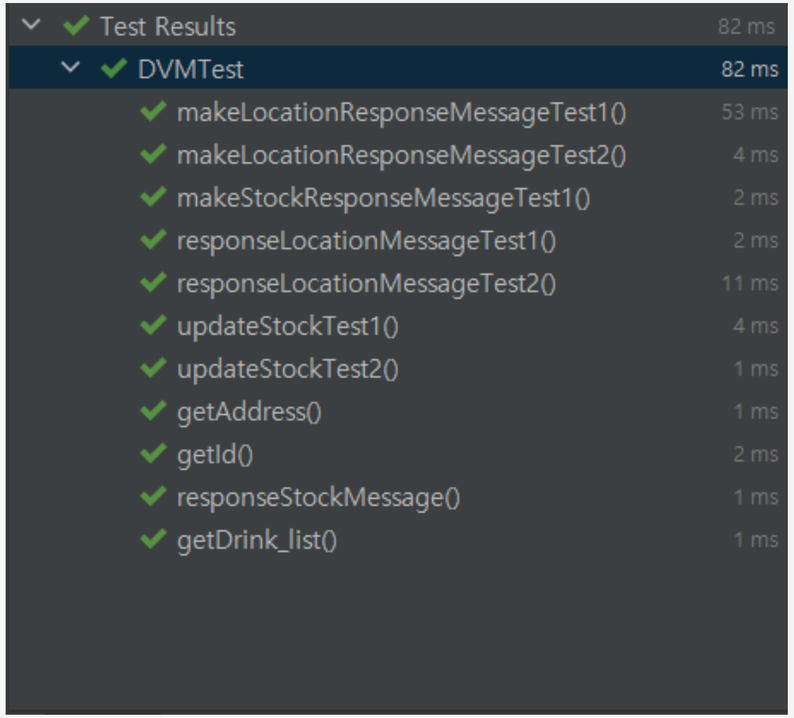
```java
@Test
void getStock() {
    assertEquals( expected: 10, drink1.getStock());

    assertEquals( expected: 11, drink2.getStock());

    assertEquals( expected: 0, drink3.getStock());

    assertEquals( expected: 10, drink4.getStock());

    assertEquals( expected: 8, drink5.getStock());

    assertEquals( expected: 1, drink6.getStock());

    assertEquals( expected: 10, drink7.getStock());

    assertEquals( expected: 0, drink8.getStock());

    assertEquals( expected: 0, drink9.getStock());

    assertEquals( expected: 0, drink10.getStock());
}
```

```java
@Test
void updateStock() {
    drink1.updateStock();
    assertEquals( expected: 9, drink1.getStock());

    drink2.updateStock();
    assertEquals( expected: 10,drink2.getStock());

    drink3.updateStock(); // 재고 없는거임 예외
    assertEquals( expected: -1,drink3.getStock());

    drink4.updateStock();
    assertEquals( expected: 9,drink4.getStock());

    drink5.updateStock();
    assertEquals( expected: 7,drink5.getStock());

    drink6.updateStock();
    assertEquals( expected: 0,drink6.getStock());

    drink7.updateStock();
    assertEquals( expected: 9,drink7.getStock());

    drink8.updateStock(); // 재고 없는거임 예외
    assertEquals( expected: -1,drink8.getStock());

    drink9.updateStock(); // 재고 없는거임 예외
    assertEquals( expected: -1,drink9.getStock());

    drink10.updateStock(); // 재고 없는거임 예외
    assertEquals( expected: -1,drink10.getStock());
}
```

```java
@Test
void getImgURL() {
    assertEquals( expected: "src/main/resources/image/1.jpg",drink1.getImgURL());

    assertEquals( expected: "src/main/resources/image/2.jpg",drink2.getImgURL());

    assertEquals( expected: "src/main/resources/image/3.jpg",drink3.getImgURL());

    assertEquals( expected: "src/main/resources/image/4.jpg",drink4.getImgURL());

    assertEquals( expected: "src/main/resources/image/5.jpg",drink5.getImgURL());

    assertEquals( expected: "src/main/resources/image/6.jpg",drink6.getImgURL());

    assertEquals( expected: "src/main/resources/image/7.jpg",drink7.getImgURL());

    assertEquals( expected: "src/main/resources/image/8.jpg",drink8.getImgURL());

    assertEquals( expected: "src/main/resources/image/9.jpg",drink9.getImgURL());

    assertEquals( expected: "src/main/resources/image/10.jpg",drink10.getImgURL());
}
```

3. 유닛 테스트

7. DVM

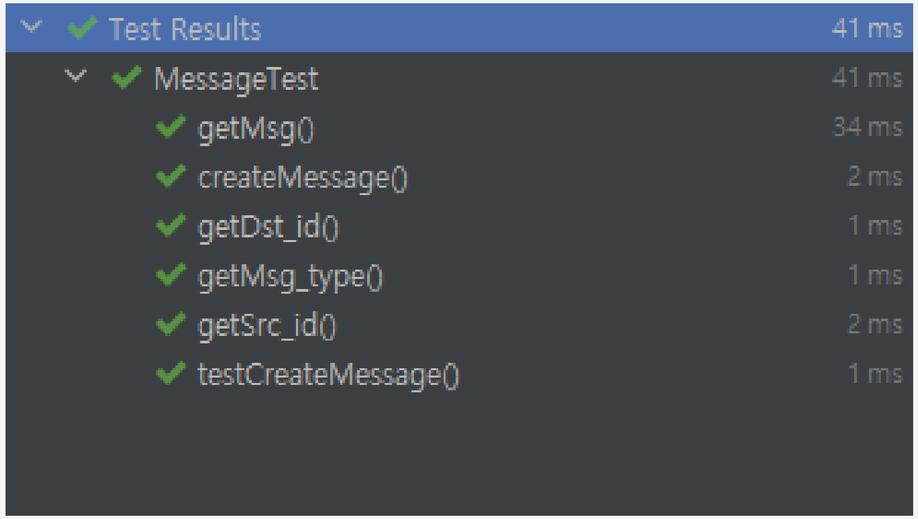



```java
@Test
void responseLocationMessageTest1() {
    Message message = new Message();
    message = message.createMessage( src_id: 1,  dst_id: 2, MsgType.RESPONSE_LOCATION,  msg: "101");

    int address = -1;
    int msg_type = 5;  //RESPONSE_LOCATION
    String msg = message.getMsg();
    if (msg_type == MsgType.RESPONSE_LOCATION) {
        address = Integer.parseInt(msg);
    }
    assertEquals( expected: 101, address);
    assertNotEquals( unexpected: -1, address);
}

@Test
void responseLocationMessageTest2() {
    Message message = new Message();
    message = message.createMessage( src_id: 1,  dst_id: 2, MsgType.RESPONSE_LOCATION,  msg: "101");

    int address = -1;
    int msg_type = 4;  //not RESPONSE_LOCATION
    String msg = message.getMsg();
    if (msg_type == MsgType.RESPONSE_LOCATION) {
        address = Integer.parseInt(msg);
    }
    assertEquals( expected: -1, address);
}
```

```java
@Test
void getDrink_list() {
    ArrayList<Drink> drinkArrayList = new ArrayList<>(); // 전체 음료수 리스트
    drinkArrayList.add(new Drink( name: "코카콜라",  price: 1500,  stock: 10,  imgURL: "src/main/resource
    drinkArrayList.add(new Drink( name: "펩시콜라",  price: 1500,  stock: 11,  imgURL: "src/main/resource
    drinkArrayList.add(new Drink( name: "칠성사이다",  price: 1500,  stock: 0,  imgURL: "src/main/resourc
    drinkArrayList.add(new Drink( name: "스프라이트",  price: 1500,  stock: 10,  imgURL: "src/main/resour
    drinkArrayList.add(new Drink( name: "환타오렌지",  price: 1500,  stock: 8,  imgURL: "src/main/resourc

    ArrayList<Drink> drink_list = drinkArrayList;

    assertEquals(drink_list.getClass(),ArrayList.class );
    assertEquals(drink_list.get(0).getClass(), Drink.class);
}
```

3. 유닛 테스트

8. Message




```
@Test
void getSrc_id() {
    Message msg2 = msg1.createMessage( src_id: 1, dst_id: 1, msg_type: 1);
    assertEquals( expected: 1,msg2.getSrc_id());
    Message msg4 = msg3.createMessage( src_id: 10, dst_id: 1, msg_type: 1);
    assertEquals( expected: 10,msg4.getSrc_id());
    Message msg6 = msg5.createMessage( src_id: 100, dst_id: 1, msg_type: 1);
    assertEquals( expected: 100,msg6.getSrc_id());
    Message msg8 = msg7.createMessage( src_id: 1000, dst_id: 1, msg_type: 1);
    assertEquals( expected: 1000,msg8.getSrc_id());
    Message msg10 = msg9.createMessage( src_id: 10000, dst_id: 1, msg_type: 1);
    assertEquals( expected: 10000,msg10.getSrc_id());
}
```

```
@Test
void getDst_id() {
    Message msg2 = msg1.createMessage( src_id: 1, dst_id: 1, msg_type: 1);
    assertEquals( expected: 1,msg2.getDst_id());
    Message msg4 = msg3.createMessage( src_id: 10, dst_id: 10, msg_type: 1);
    assertEquals( expected: 10,msg4.getDst_id());
    Message msg6 = msg5.createMessage( src_id: 100, dst_id: 100, msg_type: 1);
    assertEquals( expected: 100,msg6.getDst_id());
    Message msg8 = msg7.createMessage( src_id: 1000, dst_id: 1000, msg_type: 1);
    assertEquals( expected: 1000,msg8.getDst_id());
    Message msg10 = msg9.createMessage( src_id: 10000, dst_id: 10000, msg_type: 1);
    assertEquals( expected: 10000,msg10.getDst_id());
}
```

```
@Test
void getMsg_type() {
    Message msg2 = msg1.createMessage( src_id: 1, dst_id: 1, msg_type: 1);
    assertEquals( expected: 1,msg2.getMsg_type());
    Message msg4 = msg3.createMessage( src_id: 10, dst_id: 10, msg_type: 2);
    assertEquals( expected: 2,msg4.getMsg_type());
    Message msg6 = msg5.createMessage( src_id: 100, dst_id: 100, msg_type: 3);
    assertEquals( expected: 3,msg6.getMsg_type());
    Message msg8 = msg7.createMessage( src_id: 1000, dst_id: 1000, msg_type: 4);
    assertEquals( expected: 4,msg8.getMsg_type());
    Message msg10 = msg9.createMessage( src_id: 10000, dst_id: 10000, msg_type: 5);
    assertEquals( expected: 5,msg10.getMsg_type());
}
```

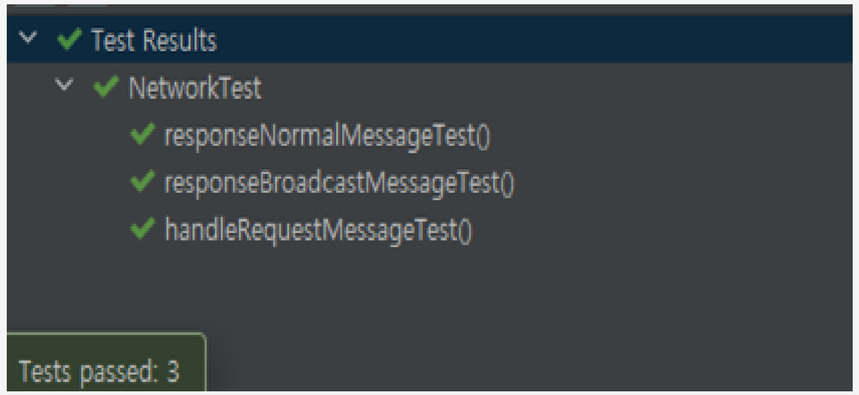
```
@Test
void getMsg() {
    Message msg2 = msg1.createMessage( src_id: 1, dst_id: 1, msg_type: 1, msg: "안녕하세요");
    assertEquals( expected: "안녕하세요",msg2.getMsg());
    Message msg4 = msg3.createMessage( src_id: 10, dst_id: 10, msg_type: 2, msg: "반갑습니다");
    assertEquals( expected: "반갑습니다",msg4.getMsg());
    Message msg6 = msg5.createMessage( src_id: 100, dst_id: 100, msg_type: 3, msg: "잘부탁드립니다");
    assertEquals( expected: "잘부탁드립니다",msg6.getMsg());
    Message msg8 = msg7.createMessage( src_id: 1000, dst_id: 1000, msg_type: 4, msg: "MS129");
    assertEquals( expected: "MS129",msg8.getMsg());
    Message msg10 = msg9.createMessage( src_id: 10000, dst_id: 10000, msg_type: 5, msg: "DVM");
    assertEquals( expected: "DVM",msg10.getMsg());
}
```

```
@Test
void createMessage() {
    Message 메시지1 = new Message();
    메시지1.setDst_id(1);
    메시지1.setSrc_id(1);
    메시지1.setMsg_type(1);
    메시지1.setMsg("");
    assertEquals(메시지1,메시지1);
    Message 메시지2 = new Message();
    메시지2.setDst_id(14321);
    메시지2.setSrc_id(14351);
    메시지2.setMsg_type(12);
    메시지2.setMsg("");
    assertEquals(메시지2,메시지2);
    Message 메시지3 = new Message();
    메시지3.setDst_id(1463);
    메시지3.setSrc_id(143);
    메시지3.setMsg_type(1234);
    메시지3.setMsg("");
    assertEquals(메시지3,메시지3);
    Message 메시지4 = new Message();
    메시지4.setDst_id(58671);
    메시지4.setSrc_id(1658);
    메시지4.setMsg_type(165);
    메시지4.setMsg("");
    assertEquals(메시지4,메시지4);
    Message 메시지5 = new Message();
    메시지5.setDst_id(179087);
    메시지5.setSrc_id(65581);
    메시지5.setMsg_type(165);
    메시지5.setMsg("");
    assertEquals(메시지5,메시지5);
}
```

```
@Test
void testCreateMessage() {
    Message 메시지1 = new Message();
    메시지1.setDst_id(1);
    메시지1.setSrc_id(1);
    메시지1.setMsg_type(1);
    메시지1.setMsg("안녕하세요");
    assertEquals(메시지1,메시지1);
    Message 메시지2 = new Message();
    메시지2.setDst_id(14321);
    메시지2.setSrc_id(14351);
    메시지2.setMsg_type(12);
    메시지2.setMsg("반갑습니다");
    assertEquals(메시지2,메시지2);
    Message 메시지3 = new Message();
    메시지3.setDst_id(1463);
    메시지3.setSrc_id(143);
    메시지3.setMsg_type(1234);
    메시지3.setMsg("잘부탁드립니다");
    assertEquals(메시지3,메시지3);
    Message 메시지4 = new Message();
    메시지4.setDst_id(58671);
    메시지4.setSrc_id(1658);
    메시지4.setMsg_type(165);
    메시지4.setMsg("MS129");
    assertEquals(메시지4,메시지4);
    Message 메시지5 = new Message();
    메시지5.setDst_id(179087);
    메시지5.setSrc_id(65581);
    메시지5.setMsg_type(165);
    메시지5.setMsg("DVM");
    assertEquals(메시지5,메시지5);
}
```

3. 유닛 테스트

9. Network

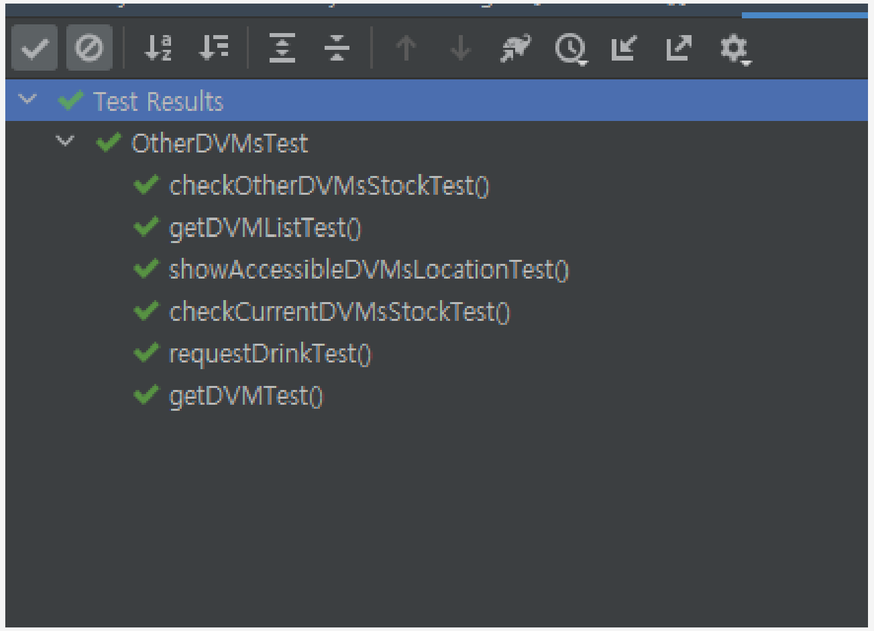



```java
@Test
void handleRequestMessageTest(){
    ArrayList<Drink> drinks = new ArrayList<>();
    Drink drink = new Drink( name: "펩시콜라",  price: 0,  stock: 3,  imgURL: "");
    drinks.add(drink);
    ArrayList<DVM> dvms = new ArrayList<>();
    DVMc dvm1 = new DVMc(drinks,  id: 1,  address: 101);
    DVMc dvm2 = new DVMc(drinks,  id: 2,  address: 202);
    dvms.add(dvm1);
    dvms.add(dvm2);
    Network network = new Network(dvms);

    Object o = network.handleRequestMessage(message.createMessage( src_id: 1,  dst_id: 2, MsgType.REQUEST_STOCK,  msg: "펩시콜라"));
    assertEquals(Integer.class, o.getClass());
    assertEquals( expected: 3, (int)o);

    Object o2 = network.handleRequestMessage(message.createMessage( src_id: 1,  dst_id: 0, MsgType.REQUEST_STOCK,  msg: "펩시콜라"));
    assertEquals(ArrayList.class, o2.getClass());
    assertEquals( expected: 101, ((ArrayList<DVM>)o2).get(0).getAddress());


}
```

```java
@Test
void responseBroadcastMessageTest() {
    ArrayList<Drink> drinks = new ArrayList<>();
    Drink drink = new Drink( name: "펩시콜라",  price: 0,  stock: 3,  imgURL: "");
    drinks.add(drink);
    ArrayList<DVM> dvms = new ArrayList<>();
    DVMc dvm1 = new DVMc(drinks,  id: 1,  address: 101);
    DVMc dvm2 = new DVMc(drinks,  id: 2,  address: 202);
    dvms.add(dvm1);
    dvms.add(dvm2);
    Network network = new Network(dvms);

    int stock = network.responseBroadcastMessage(message.createMessage( src_id: 2,  dst_id: 1, MsgType.RESPONSE_STOCK,  msg: "3"));
    assertEquals( expected: 3, stock);


}

@Test
void responseNormalMessageTest() {
    ArrayList<Drink> drinks = new ArrayList<>();
    Drink drink = new Drink( name: "펩시콜라",  price: 0,  stock: 3,  imgURL: "");
    drinks.add(drink);
    ArrayList<DVM> dvms = new ArrayList<>();
    DVMc dvm1 = new DVMc(drinks,  id: 1,  address: 101);
    DVMc dvm2 = new DVMc(drinks,  id: 2,  address: 202);
    dvms.add(dvm1);
    dvms.add(dvm2);
    Network network = new Network(dvms);

    int stock = network.responseNormalMessage(message.createMessage( src_id: 2,  dst_id: 1, MsgType.RESPONSE_STOCK,  msg: "3"));
    assertEquals( expected: 3, stock);

    int address = network.responseNormalMessage(message.createMessage( src_id: 2,  dst_id: 1, MsgType.RESPONSE_LOCATION,  msg: "101"));
    assertEquals( expected: 101, address);
}
```

3. 유닛 테스트

9. OtherDVMs



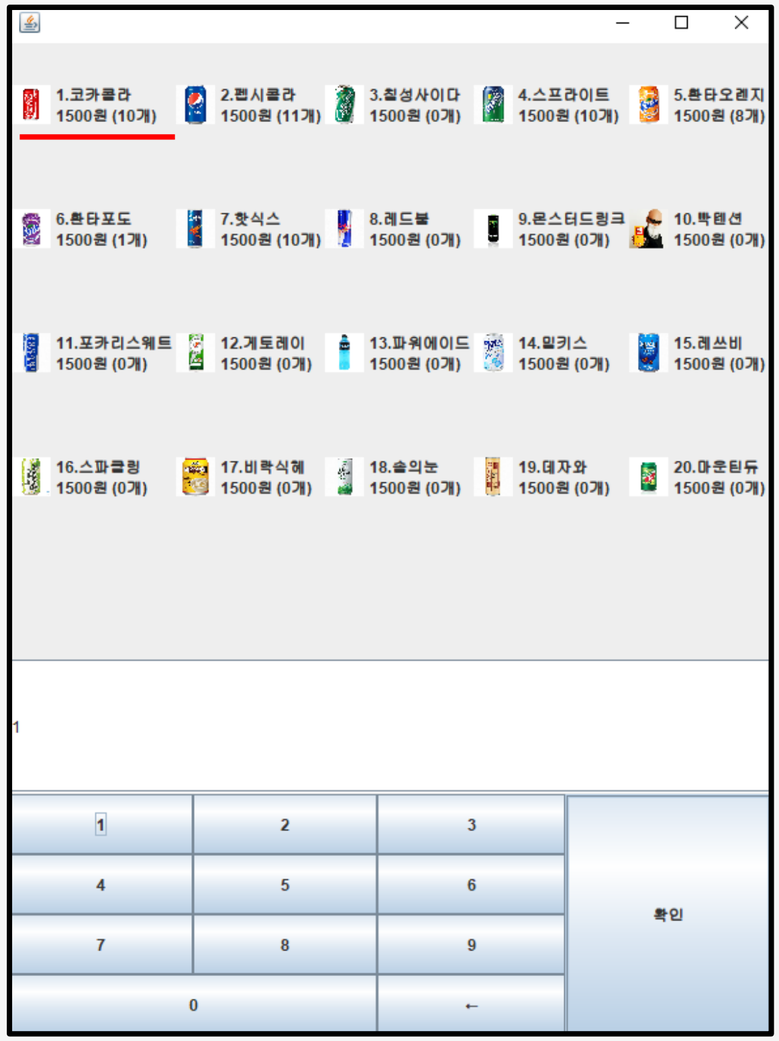

```java
class OtherDVMsTest {
    Controller controller = new Controller();
    OtherDVMs otherDVMs = new OtherDVMs();
    @Test
    void getDVMTest() {
        DVM dvm = otherDVMs.getDVM( index: 0);
        assertEquals( expected: 0, dvm.getId());
    }

    @Test
    void getDVMListTest() {
        ArrayList<DVM> dvmList = otherDVMs.getDVMList();
        assertEquals( expected: 8, dvmList.size());
        assertEquals( expected: 7, dvmList.get(dvmList.size() - 1).getId());
    }

    @Test
    void checkCurrentDVMsStockTest() {
        Drink drink = new Drink( name: "빡텐션", price: 0, stock: 0, imgURL: "");
        DVM currentDVM = otherDVMs.getDVM( index: 0);

        boolean b = otherDVMs.checkCurrentDVMsStock(drink, currentDVM);
        assertFalse(b);
    }

    @Test
    void checkOtherDVMsStockTest() {
        Drink drink = new Drink( name: "빡텐션", price: 0, stock: 0, imgURL: "");

        DVM currentDVM = otherDVMs.getDVM( index: 0);
        ArrayList<DVM> dvms = otherDVMs.checkOtherDVMsStock(drink, currentDVM);

        assertNotEquals(dvms.size(), actual: 0);
    }
```

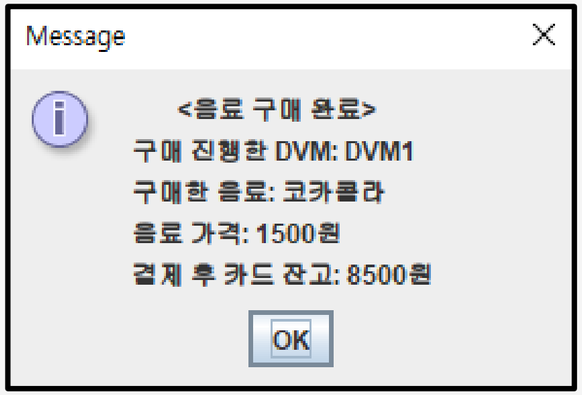
4. 시스템 테스트

No	Test 항목	Description
1	Select drink test	선택한 음료로 결제 진행이 잘 되는지 확인한다.
2		범위 밖의 음료를 선택한다.
3		정해진 가격대로 정확히 출력되는지 확인한다.
4	Proceed Card-Payment test	사용 가능한 카드를 입력한다.
5		사용 불가능한 카드를 입력한다.
6		카드의 잔액이 충분할 때만 결제가 진행되는지 확인한다.
7		카드의 잔액이 부족할 경우 잔액 부족 메시지를 출력하는지 확인한다.
8	Check current machine stock test	현재 자판기의 재고가 정확한지 확인한다.
9		현재 자판기에서 재고가 부족할 때 재고가 부족하다는 메시지를 출력하는 지 확인한다.
10	Check other DVMs stock test	자기 외의 다른 자판기의 재고가 정확하게 전달되는지 확인한다.
11		다른 자판기의 재고도 모두 비운 뒤 재고 확인이 정확히 이루어지는지 확인한다.
12	DVMs location test	재고가 있는 자판기의 위치를 올바르게 출력하는 지 확인한다.
13	Make code test	랜덤으로 생성된 인증코드가 기존의 인증코드와 중복되지 않는지 확인한다.
14		화면에 출력되는 인증코드가 사용자가 선결제해서 생성된 인증코드와 일치하는지 확인한다.
15	Enter code test	코드를 생성한 후 생성한 코드와 같은 코드를 입력해본다.
16		코드를 생성한 후 생성한 코드와 다른 코드를 입력해본다.
17		생성한 코드가 없는 채로 코드 입력을 시도해본다.
18	Check code test	유효하지 않은 인증코드를 입력해본다
19		이미 사용한 인증코드를 입력해본다
20	Check drink test	제공할 음료가 사용자가 결제한 음료와 일치하는지 확인한다

4. 시스템 테스트

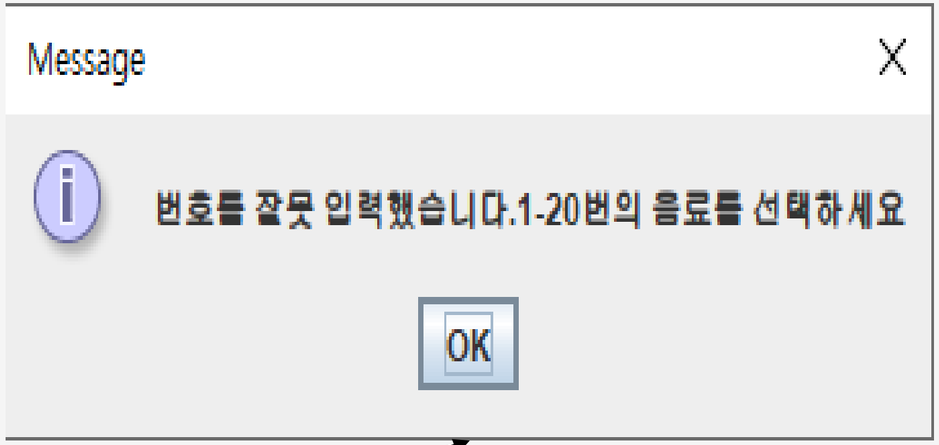
1. 선택한 음료로 결제진행이 잘 되는지 확인한다. - **성공**





4. 시스템 테스트

2. 범위 밖의 음료를 선택한다
   -> **실패 팝업 발생되어야 함** - **성공**



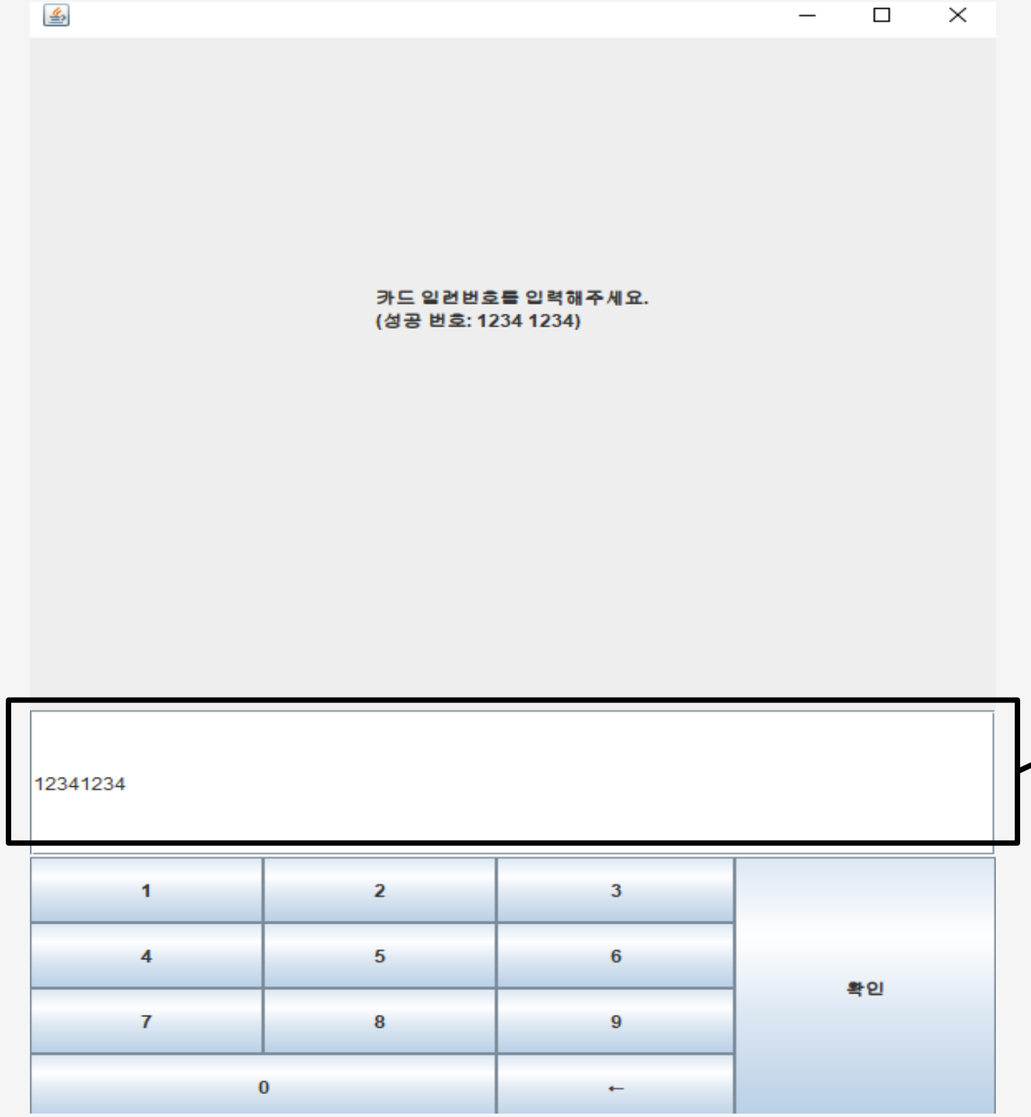
22번 음료 선택 시

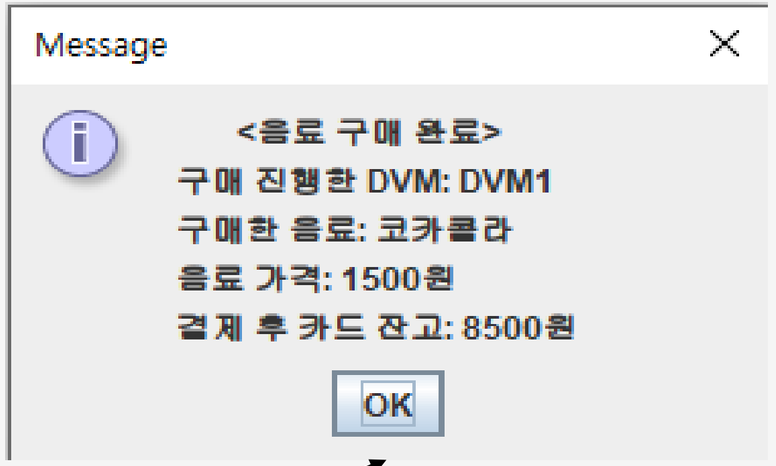
4. 시스템 테스트

3. 정해진 가격대로 정확히 출력되는지 확인한다.- **성공**

4. 시스템 테스트

4. 사용 가능한 카드를 입력한다.- **성공**

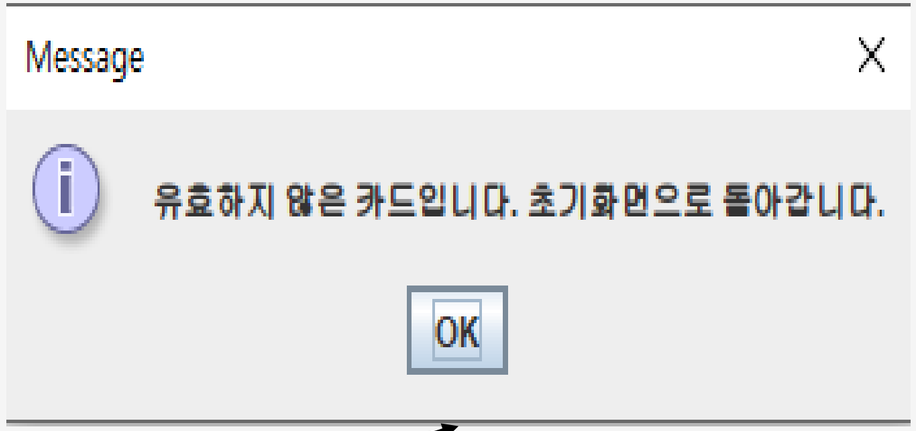




사용가능한 카드 번호
12341234 사용

4. 시스템 테스트

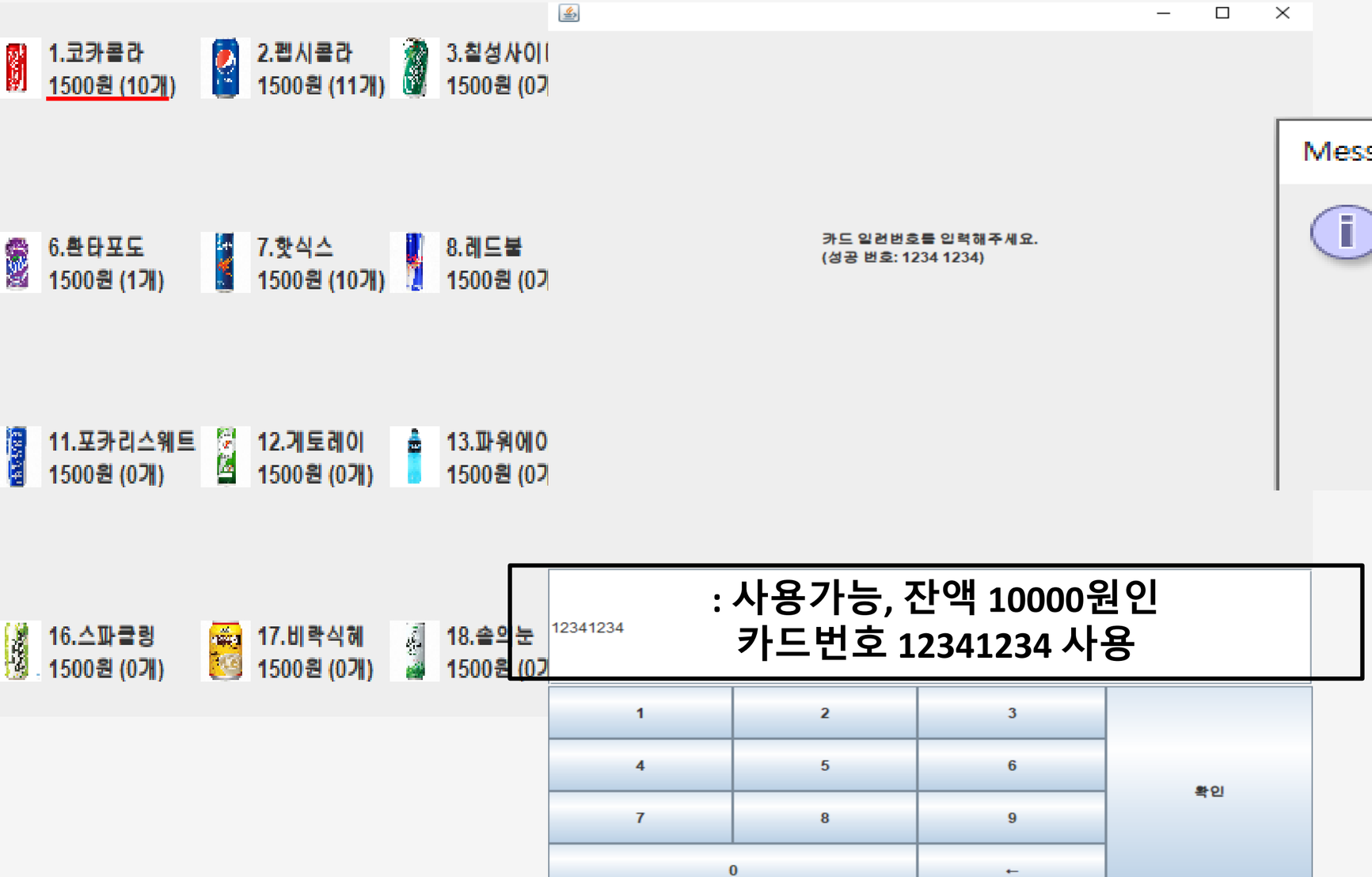
5. 사용 불가능한 카드를 입력한다.
   -> **결제 실패 메시지가 발생해야 함 - 성공**

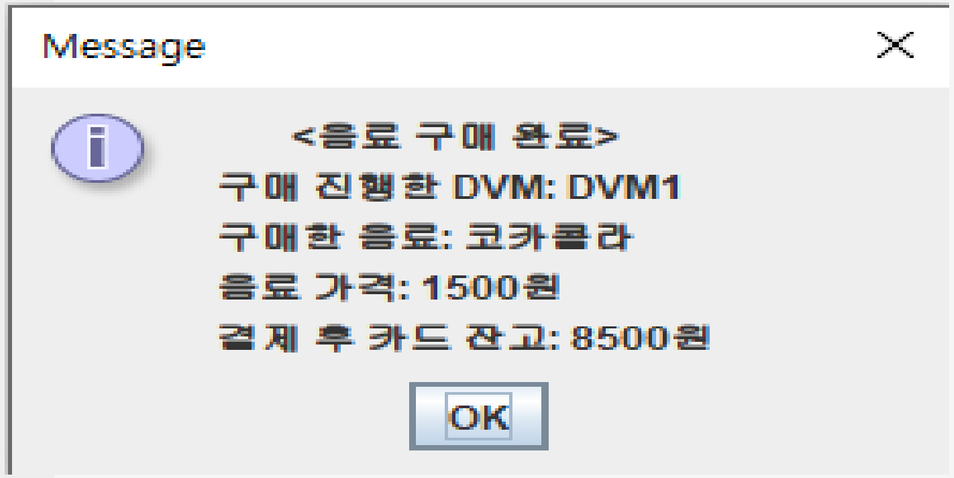


사용불가능한 카드 번호
99999999 사용

4. 시스템 테스트

6. 카드 잔액이 충분할 때만 결제가 진행되는지 확인한다.- **성공**

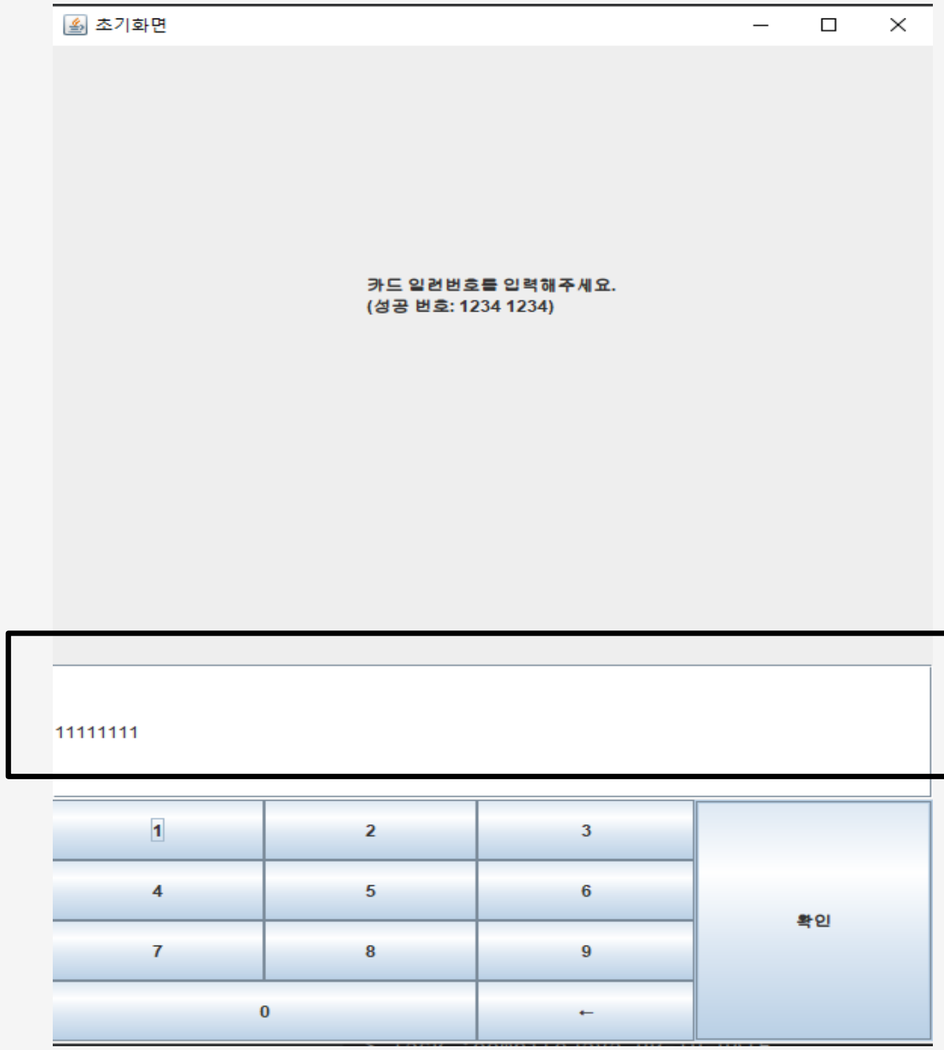


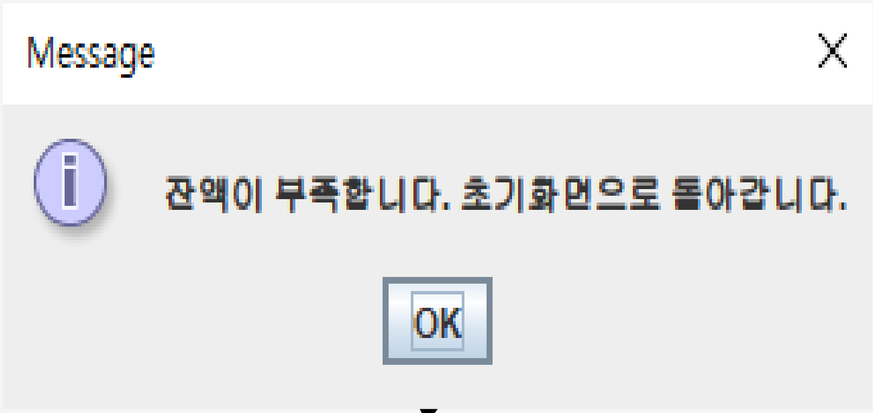


: 사용가능, 잔액 10000원인 카드번호 12341234 사용

4. 시스템 테스트

7. 카드 잔액이 부족할 경우 잔액 부족 메세지를 출력하는지 확인한다.- **성공**





: 사용가능, 잔액 0원인
카드번호 11111111 사용

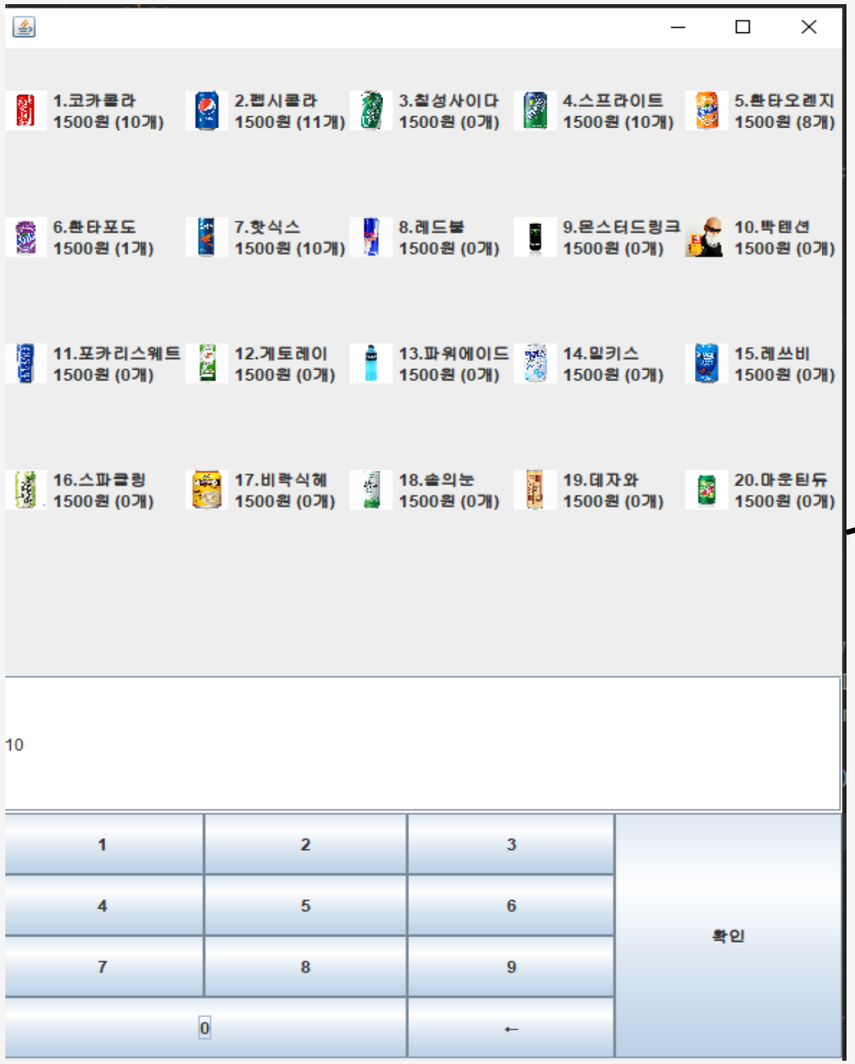
4. 시스템 테스트

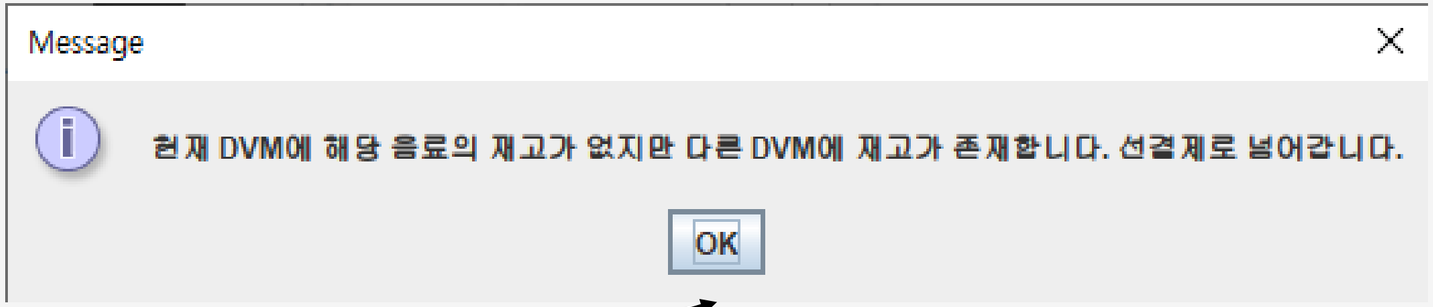
8. 현재 자판기의 재고가 정확한지 확인한다. - **성공**

: 결제 진행 후 다시 구매하고자 할 때,
판매된 재고 반영

4. 시스템 테스트

9. 현재 자판기에서 재고가 부족할 때 재고가 부족하다는 메시지를 출력하는지 확인한다. .- **성공**

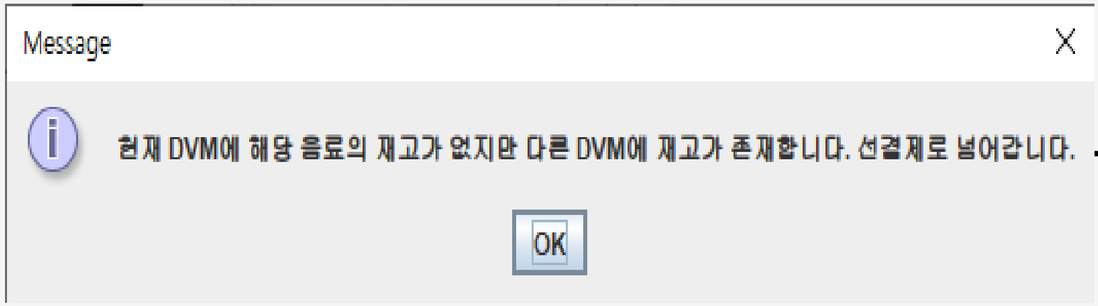


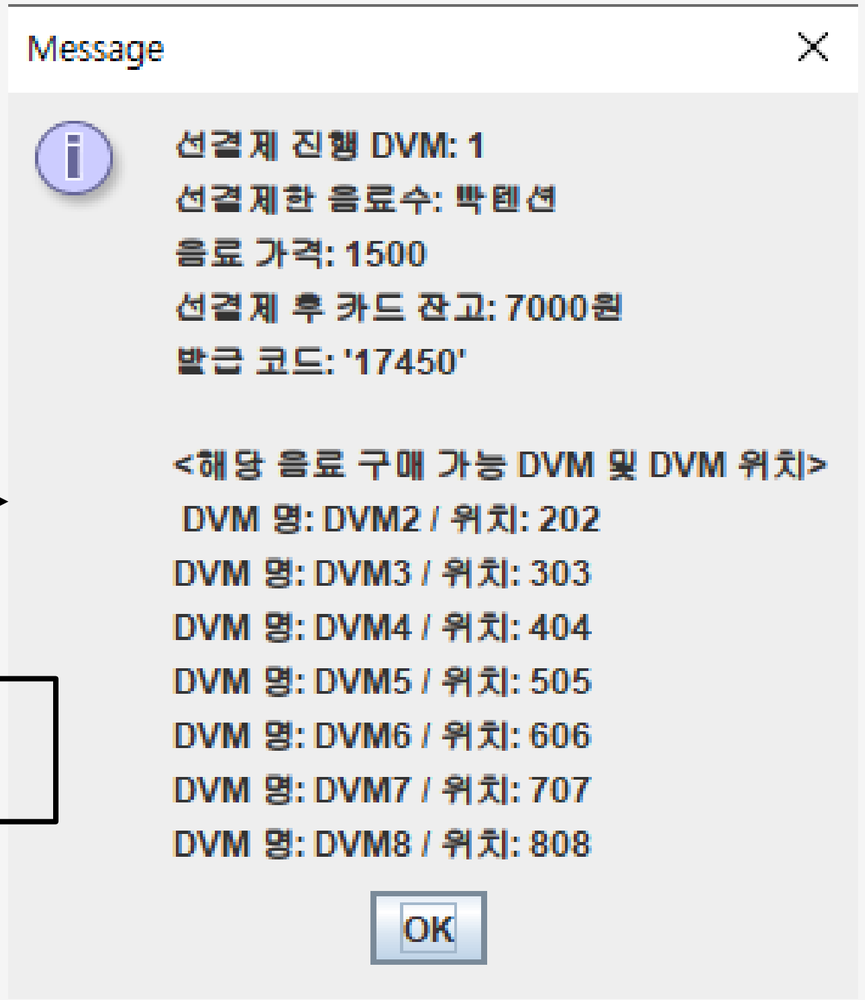


현재 자판기에 0개인 10번 음료 선택

4. 시스템 테스트

10. 다른 자판기의 재고가 정확히 전달되는지 확인한다.- **성공**

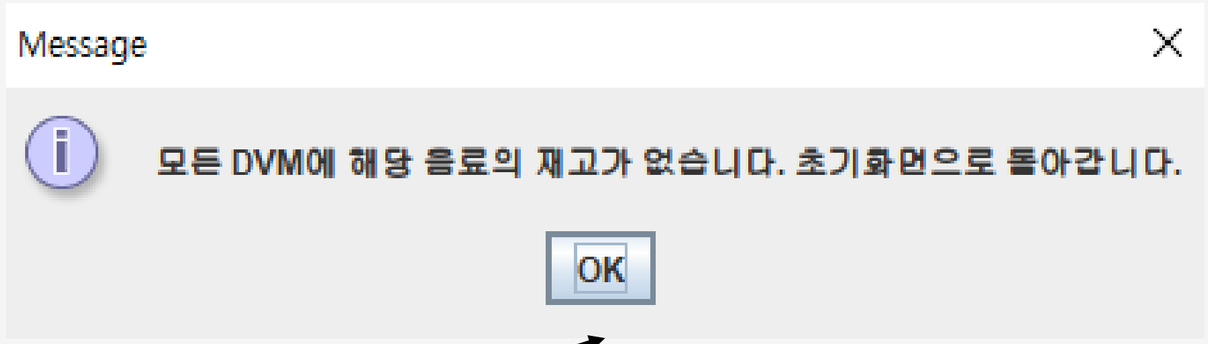




현재 자판기에 0개인 10번 음료 선택
: 다른 DVM 재고 확인 후 선결제

4. 시스템 테스트

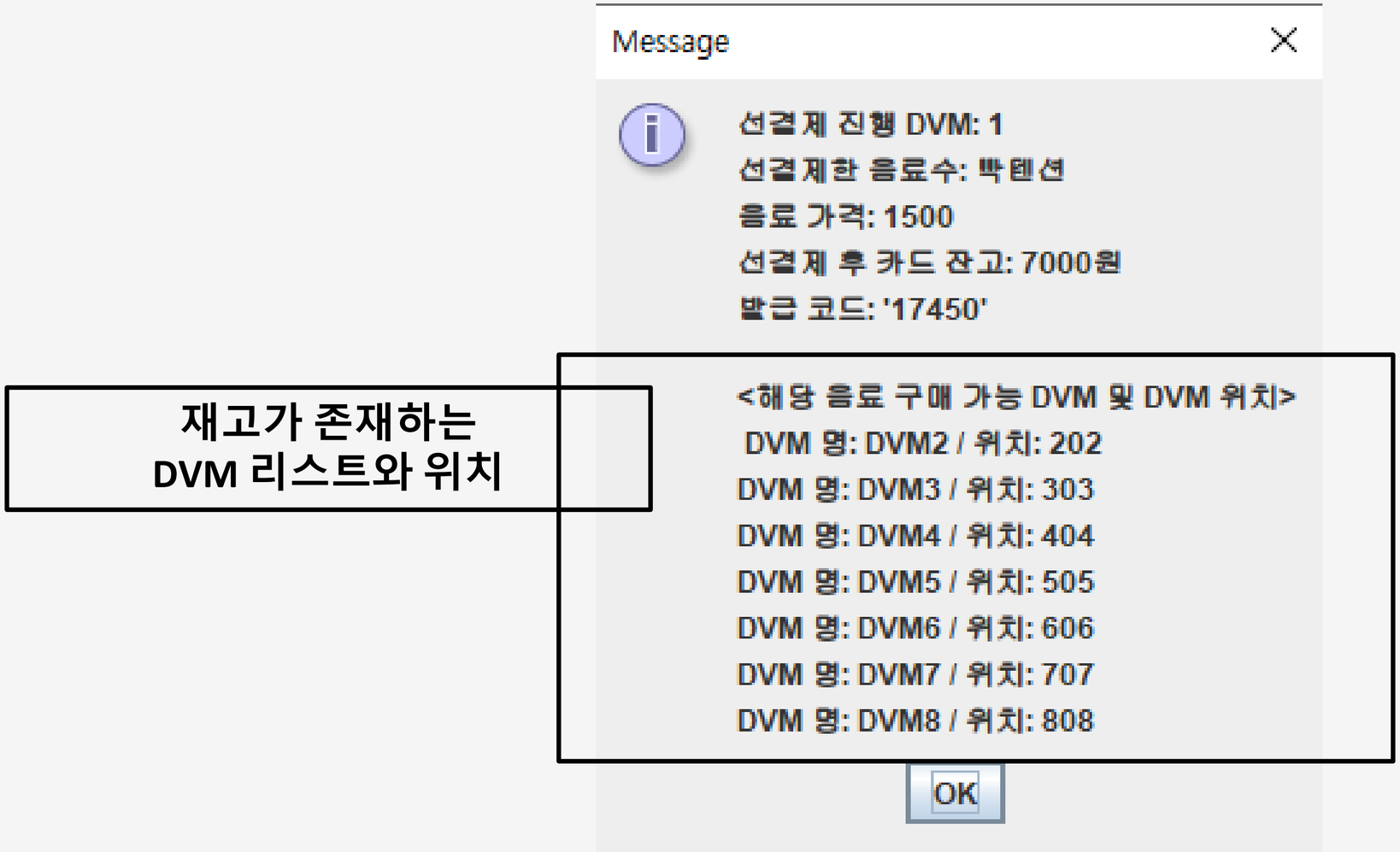
11. 다른 자판기의 재고도 모두 비운 뒤, 재고 확인이 정확히 이루어지는지 확인한다..- **성공**



모든 자판기에 재고가 없는
3번 음료 선택

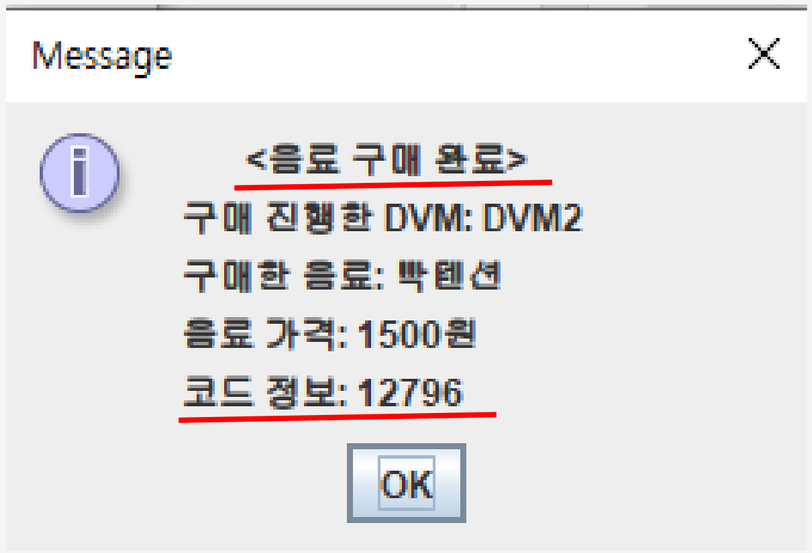
4. 시스템 테스트

12. 재고가 있는 자판기의 위치를 올바르게 출력하는지 확인한다.- **성공**



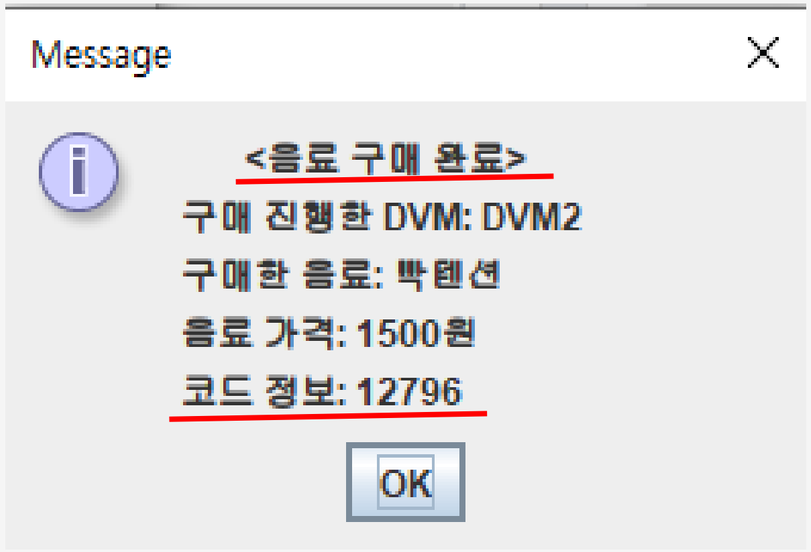
4. 시스템 테스트

13. 랜덤으로 생성된 인증코드가 기존의 코드들과 중복되지 않는지 확인한다.- **성공**



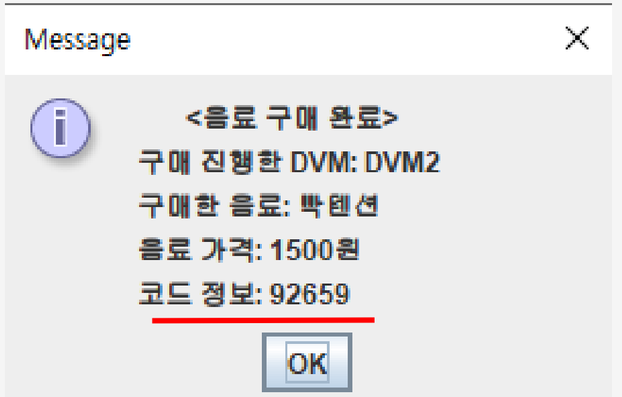
4. 시스템 테스트

14. 화면에 출력되는 인증코드가 사용자가 선결제해서 생성된 인증코드와 일치하는지 확인한다.- **성공**



4. 시스템 테스트

15. 코드를 생성한 후, 같은 코드를 코드결제 시 입력해본다.- **성공**

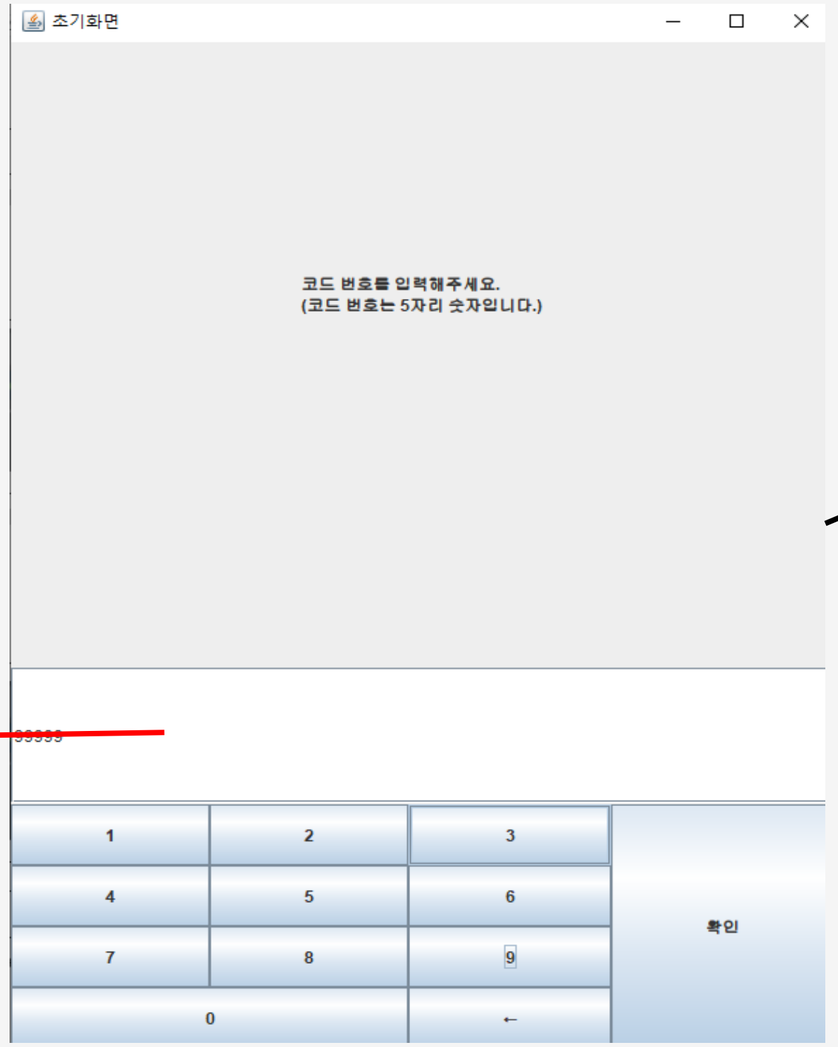


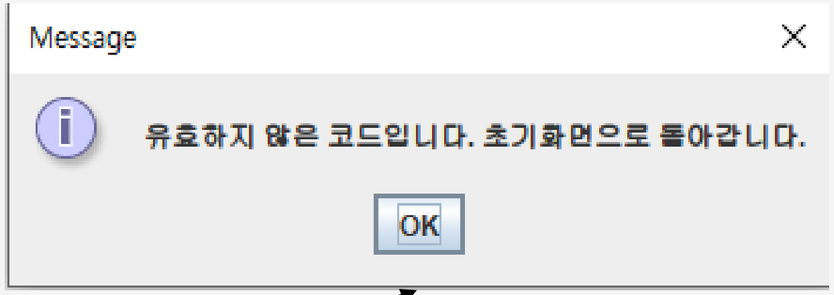
생성된 코드 92659 입력 시,
구매 완료

4. 시스템 테스트

16. 코드를 생성한 후, 다른 코드를 코드결제 시 입력해본다.

-> 코드 결제 실패가 나야함 - 성공

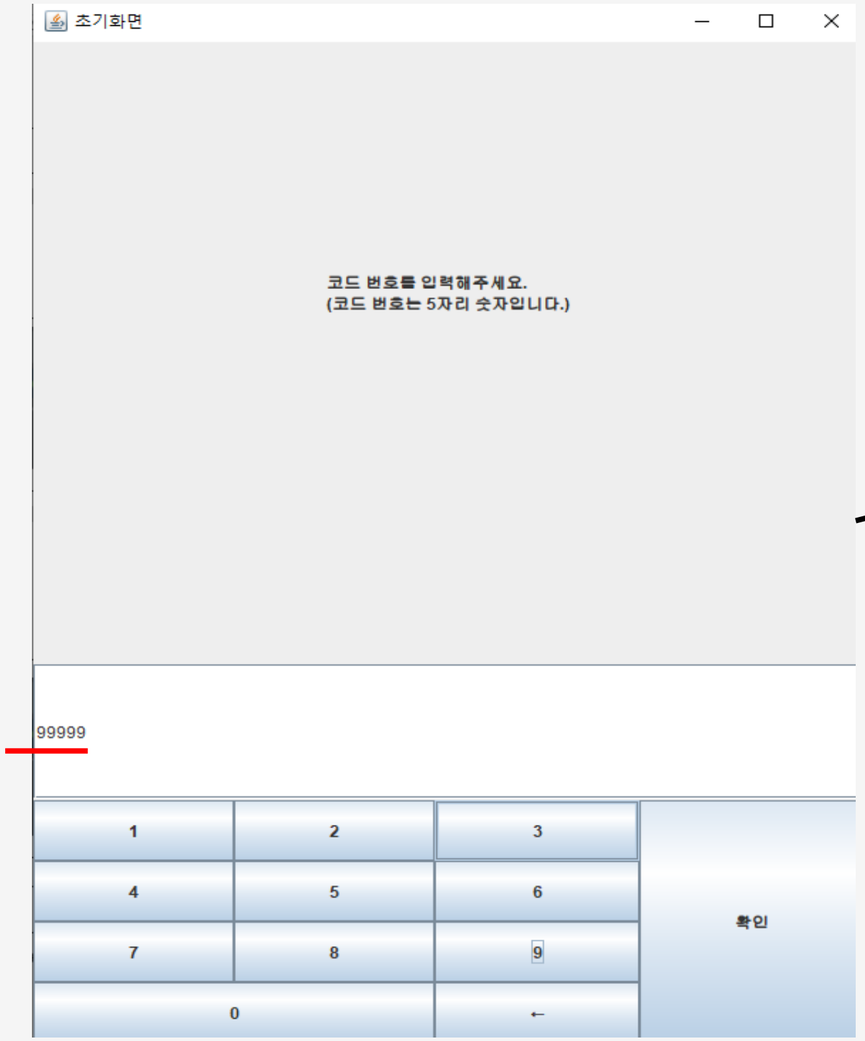


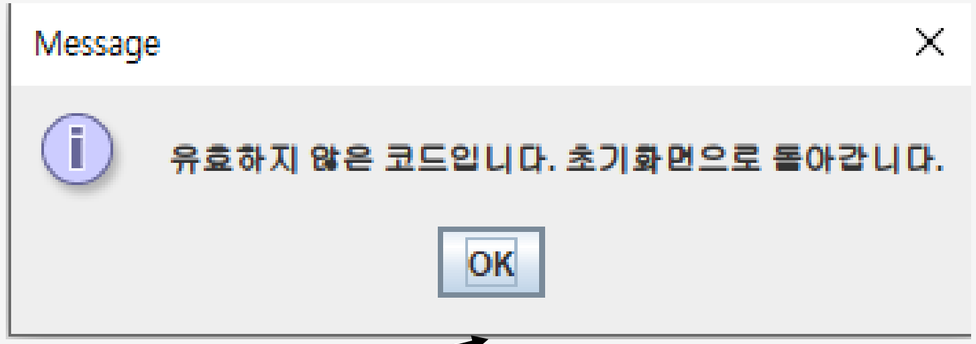


생성된 코드 50728과 다른 99999 입력 시

4. 시스템 테스트

17. 생성한 코드가 없을 때 코드입력을 시도해본다
-> 결제 실패가 나야함 - 성공

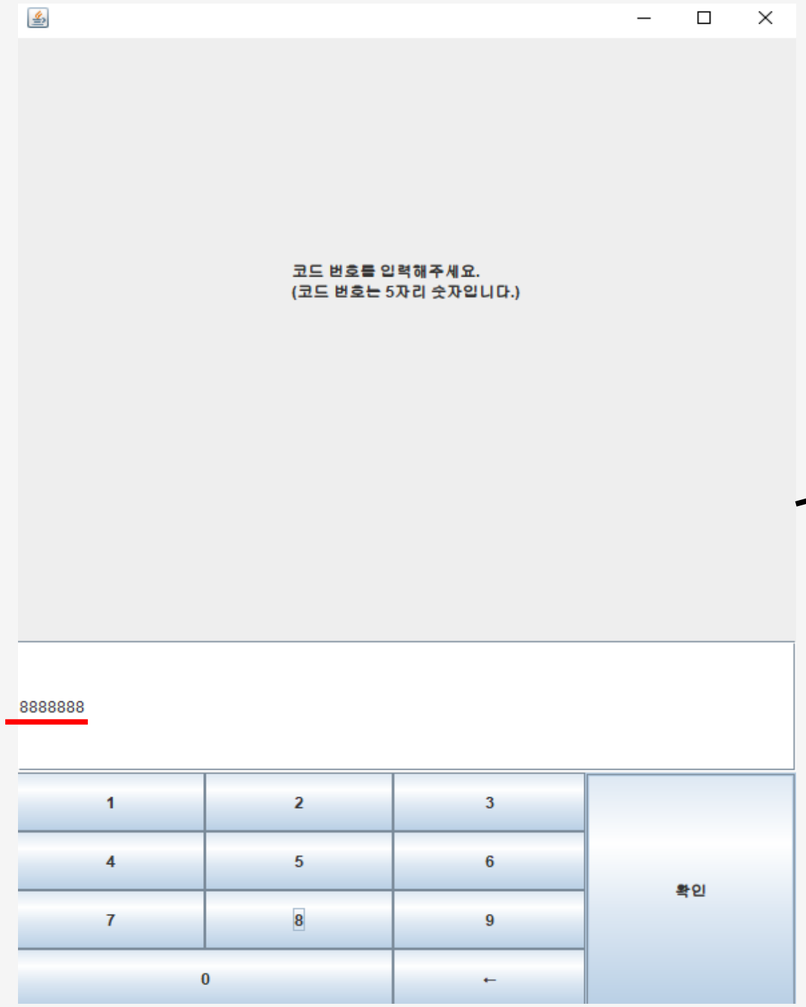


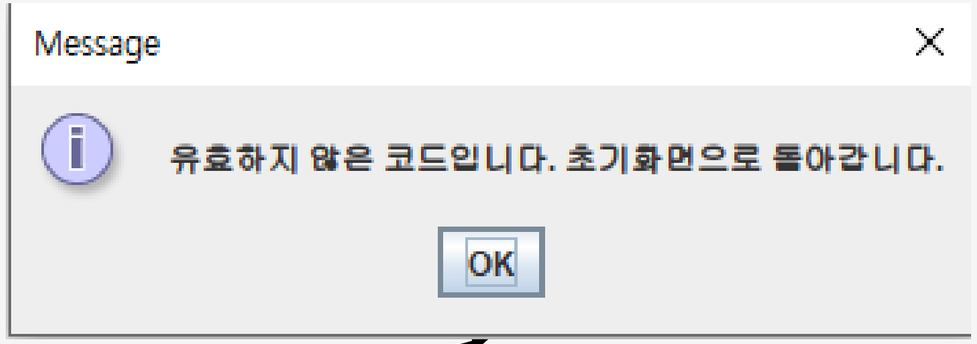


생성된 코드가 없을 때
99999라는 코드 입력 시

4. 시스템 테스트

18. 유효하지 않은 인증코드를 입력해본다.
-> 결제 실패가 나야 함 - 성공

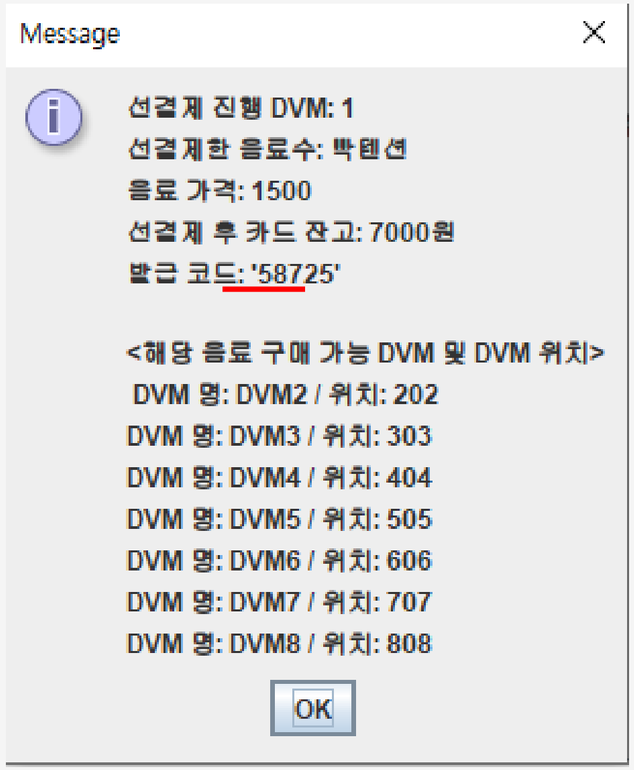




4. 시스템 테스트

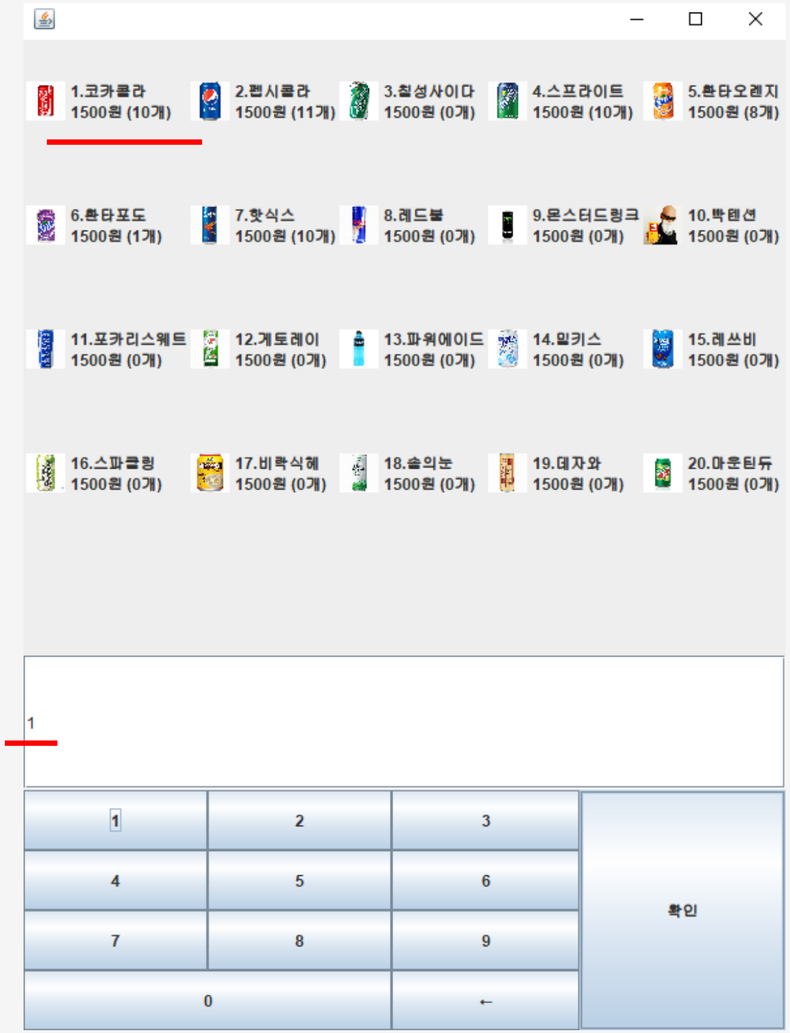
19. 이미 사용한 인증코드를 입력해본다.

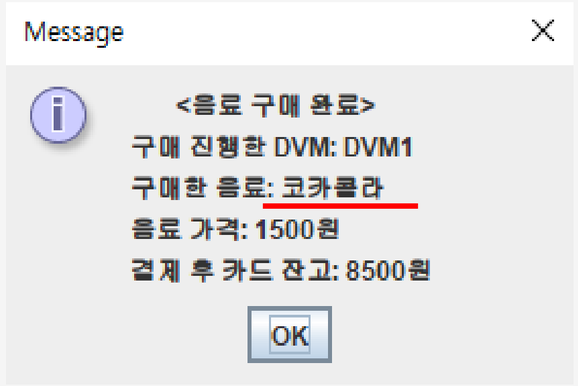
-> 결제 실패가 나야 함



4. 시스템 테스트

20. 제공할 음료가 사용자가 결제한 음료와 일치하는지 확인한다.- **성공**

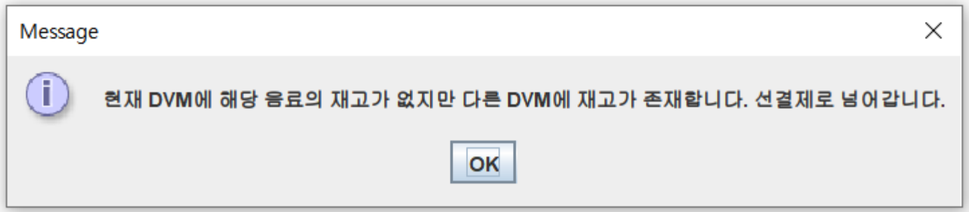




5. DVM간 네트워크용 스텁(Stub)

[Stock] BroadCast Message

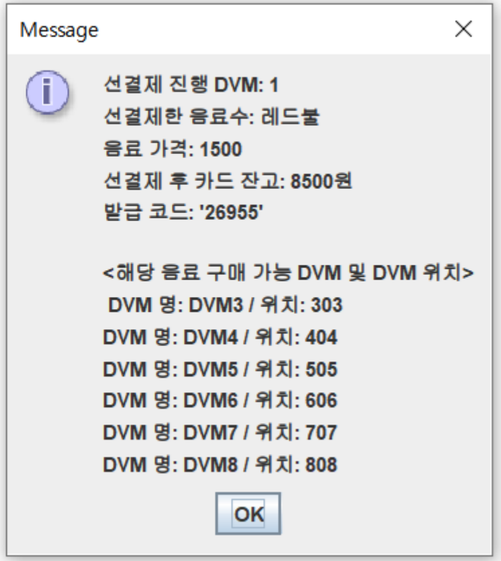
```
> Task :Test2.main()
ResopnseBroadCastMessage == src_id: 0, dst_id: 0, msg_type: 2, msg: 0
ResopnseBroadCastMessage == src_id: 1, dst_id: 0, msg_type: 2, msg: 0
ResopnseBroadCastMessage == src_id: 2, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 3, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 4, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 5, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 6, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 7, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 0, dst_id: 0, msg_type: 2, msg: 0
ResopnseBroadCastMessage == src_id: 1, dst_id: 0, msg_type: 2, msg: 0
ResopnseBroadCastMessage == src_id: 2, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 3, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 4, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 5, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 6, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 7, dst_id: 0, msg_type: 2, msg: 10
```



5. DVM간 네트워크용 스텁(Stub)

[Location] Request Message / Response Message

```
ResopnseBroadCastMessage == src_id: 2, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 3, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 4, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 5, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 6, dst_id: 0, msg_type: 2, msg: 10
ResopnseBroadCastMessage == src_id: 7, dst_id: 0, msg_type: 2, msg: 10
ResopnseNormalMessage == src_id: 2, dst_id: 0, msg_type: 2, msg: 303
requestLocationMessage == src_id: 0,dst_id: 2, msg_type: 2, msg: 303
ResopnseNormalMessage == src_id: 3, dst_id: 0, msg_type: 2, msg: 404
requestLocationMessage == src_id: 0,dst_id: 3, msg_type: 2, msg: 404
ResopnseNormalMessage == src_id: 4, dst_id: 0, msg_type: 2, msg: 505
requestLocationMessage == src_id: 0,dst_id: 4, msg_type: 2, msg: 505
ResopnseNormalMessage == src_id: 5, dst_id: 0, msg_type: 2, msg: 606
requestLocationMessage == src_id: 0,dst_id: 5, msg_type: 2, msg: 606
ResopnseNormalMessage == src_id: 6, dst_id: 0, msg_type: 2, msg: 707
requestLocationMessage == src_id: 0,dst_id: 6, msg_type: 2, msg: 707
ResopnseNormalMessage == src_id: 7, dst_id: 0, msg_type: 2, msg: 808
requestLocationMessage == src_id: 0,dst_id: 7, msg_type: 2, msg: 808
```



감사합니다