Konkuk Univ.

Distributed Vending Machine
: MS129 Vending Machine

3rd Cycle Development

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

Redmine Issue Tracking

일감 추적

	진행중	완료됨	합계
결함	8	168	176
새기능	0	1	1
지원	0	0	0

모든 일감 보기 | 요약 | 달력 | Gantt 챠트

일감

새 일감만들기

검색조건

상태 진행중 검색조건 추가

유형 이다 결함

옵션

적용 지우기 저장

#	유형	상태	우선순위	제목	담당자	변경
232	결함	의견	보통	[2nd Cycle] [Check Style] Nested If Depth		2021/06/09 20:25
218	결함	의견	보통	[2nd Cycle] [PMD] StubTest - L86		2021/06/09 21:04
214	결함	의견	보통	[2nd Cycle] [PMD] DVM.java - L24		2021/06/09 20:26
213	결함	의견	보통	[2nd Cycle] [PMD] Card.java - L19		2021/06/09 20:09
209	결함	의견	보통	[2nd Cycle] [findbugs] StubTest.java - L11		2021/06/09 20:53
182	결함	의견	보통	[2nd Cycle] [findbugs] Controller.java - L84		2021/06/09 20:29
100	결함	의견	보통	[Stage 1009 - 7.1 - 8] 17번에서 앞의 내용에서는 DVM이 다른 DVM의 위치를 알고 있는것처럼 묘사되므로, 이 Test Case의 필요성이 없다고 판단		2021/05/25 07:01
94	결함	의견	보통	[Stage 1009 - 7.1 - 2] 단계 중간에 이전 화면이나 초기화면으로 돌아가는 기능이 필요		2021/05/25 06:45

[2nd Cycle][Stage 2131-6] Ref 수정 필요

6. 카드 결제를 진행한다

Use Case	6. 카드 결제를 진행한다.
Actor	Customer
Type	Evident
Cross Reference	Functions: R3.2, R3.4, R6.1 Use cases : "카드 결제를 진행한다", "결제를 취소한다", "선결제를 진행한다"
Pre-Requisites	- Customer 가 결제 방법 중 카드 결제를 선택해야 한다.
Typical Courses of Events	(A): Actor, (S): System 1. (A) Customer 는 카드 결제를 위해, 카드를 결제 단말기에 꽂는다. 2. (S) Customer 가 결제 단말기에 사용한 카드가 결제를 할 수 있는 종류의 카드인지 확인한다. 3. (S) 결제할 음료의 가격을 전달받는다. 4. (S) 결제 단말기에 사용한 카드의 잔액이 음료의 가격을 결제할 수 있는지 확인한다. 5. (S) 카드 결제를 완료한다. 6. (S) 재고를 업데이트한다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2. 카드가 결제를 할 수 없는 종류의 카드인 경우, "결제를 취소한다."를 invoke 한다. Line 3. 카드의 잔액이 음료의 가격을 결제할 수 없을 경우, "결제를 취소한다"를 invoke 한다.

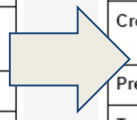
6. 카드 결제를 진행한다

Use Case	6. 카드 결제를 진행한다.
Actor	Customer
Type	Evident
Cross Reference	Functions: R3.2, R3.4 Use cases : "카드 결제를 진행한다", "결제를 취소한다"
Pre-Requisites	- Customer 가 결제 방법 중 카드 결제를 선택해야 한다.
Typical Courses of Events	(A): Actor, (S): System 1. (A) Customer 는 카드 결제를 위해, 카드를 결제 단말기에 꽂는다. 2. (S) Customer 가 결제 단말기에 사용한 카드가 결제를 할 수 있는 종류의 카드인지 확인한다. 3. (S) 결제할 음료의 가격을 전달받는다. 4. (S) 결제 단말기에 사용한 카드의 잔액이 음료의 가격을 결제할 수 있는지 확인한다. 5. (S) 카드 결제를 완료한다. 6. (S) 재고를 업데이트한다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2. 카드가 결제를 할 수 없는 종류의 카드인 경우, "결제를 취소한다."를 invoke 한다. Line 3. 카드의 잔액이 음료의 가격을 결제할 수 없을 경우, "결제를 취소한다"를 invoke 한다.

[2nd Cycle][Stage 2131-1] Ref 수정 필요

1. 서비스를 시작한다.

Use Case	1. 서비스를 시작한다.
Actor	System
Type	Hidden
Cross Reference	Functions: R 1.1, R 1.2 Use cases: "서비스를 시작한다", "DVM 선택을 진행한다"
Pre-Requisites	N/A
Typical Courses of Events	(A): Actor, (S): System 1. (S) 화면에 전체 DVM 목록을 보여준다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A



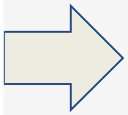
1. 서비스를 시작한다.

Use Case	1. 서비스를 시작한다.
Actor	System
Type	Hidden
Cross Reference	Functions: R 1.1 Use cases: "서비스를 시작한다"
Pre-Requisites	N/A
Typical Courses of Events	(A): Actor, (S): System 1. (S) 화면에 전체 DVM 목록을 보여준다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A

[2nd Cycle][Stage 2131-4] Ref 수정 필요

4. 현재 자판기에서 판매하지 않는 음료수를 선택한다

Use Case	4. 현재 자판기에서 판매하지 않는 음료수를 선택한다.
Actor	Customer
Type	Evident
Cross Reference	Functions: R1.2, R2.2, R6.1 Use cases : "DVM 선택을 진행한다","현재 자판기에서 판매하지 않는 음료수를 선택한다","선결제를 진행한다"
Pre-Requisites	- 전체 20가지의 음료 종류를 알아야 한다.
Typical Courses of Events	(A1): Actor, (S): System 1. (A1) Customer 가 현재 자판기에서 판매하지 않는 음료수를 선택한다.. 2. (S) "선결제를 진행한다"를 invoke한다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A



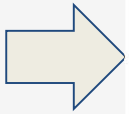
4. 현재 자판기에서 판매하지 않는 음료수를 선택한다

Use Case	4. 현재 자판기에서 판매하지 않는 음료수를 선택한다.
Actor	Customer
Type	Evident
Cross Reference	Functions: R2.2 Use cases : ,"현재 자판기에서 판매하지 않는 음료수를 선택한다"
Pre-Requisites	- 전체 20가지의 음료 종류를 알아야 한다.
Typical Courses of Events	(A1): Actor, (S): System 1. (A1) Customer 가 현재 자판기에서 판매하지 않는 음료수를 선택한다.. 2. (S) "선결제를 진행한다"를 invoke한다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A

[2nd Cycle][Stage 2131-6] Ref 수정 필요

3. 현재 자판기에 판매하는 음료수를 선택한다

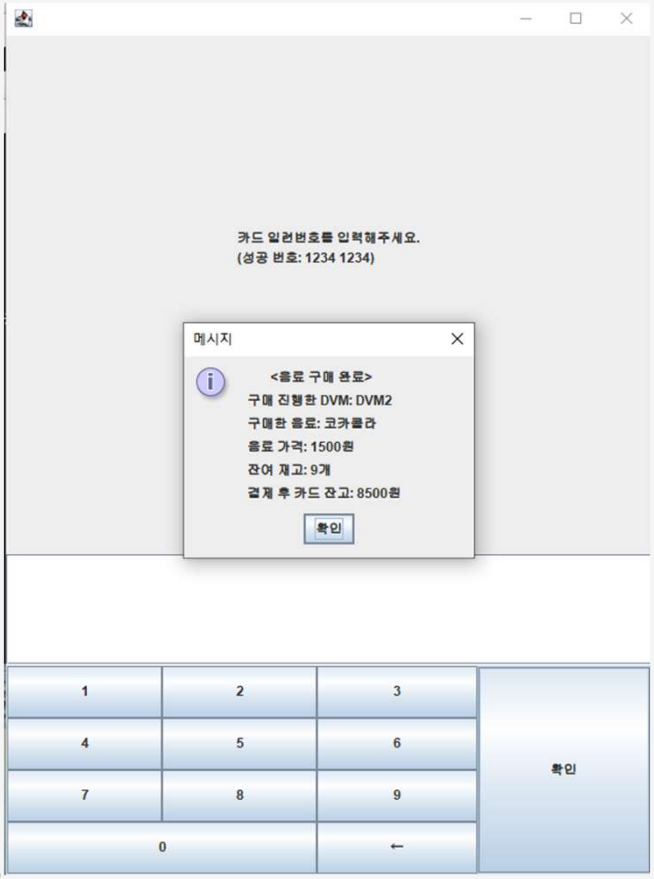
Use Case	3. 현재 자판기에서 판매하는 음료수를 선택한다.
Actor	Customer
Type	Evident
Cross Reference	Functions: R1.2, R2.1, R3.1, R4.1 Use cases : "DVM 선택을 진행한다","현재 자판기에 판매하는 음료수를 선택한다", "결제 방법 중 하나를 선택한다", "현재 자판기의 재고를 확인한다"
Pre-Requisites	- 전체 20가지의 음료 종류를 알아야 한다.
Typical Courses of Events	(A1): Actor, (S): System 1. (A1) Customer 가 현재 자판기에 판매하는 음료수를 선택한다. 2. (S) "현재 자판기의 재고를 확인한다"를 invoke하여 선택한 음료수의 재고를 확인한다. 3. (S) "결제 방법 중 하나를 선택한다"를 invoke 한다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line2. 현재 자판기의 재고가 없다면 재고가 없다는 메시지를 출력 후 "선결제를 진행한다"를 invoke한다.

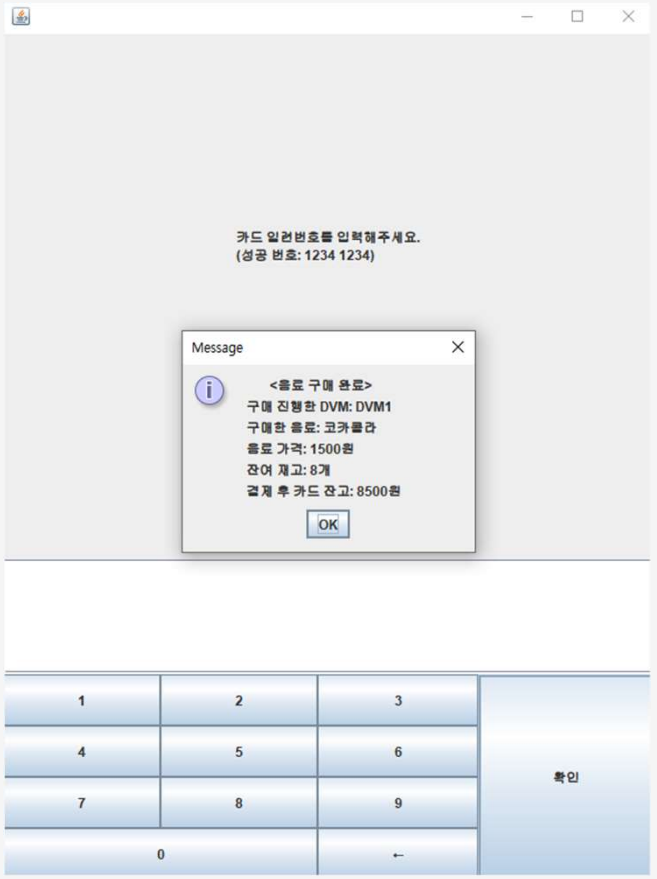


3. 현재 자판기에 판매하는 음료수를 선택한다

Use Case	3. 현재 자판기에서 판매하는 음료수를 선택한다.
Actor	Customer
Type	Evident
Cross Reference	Functions: R2.1, R4.1 Use cases :,"현재 자판기에 판매하는 음료수를 선택한다", "현재 자판기의 재고를 확인한다"
Pre-Requisites	- 전체 20가지의 음료 종류를 알아야 한다.
Typical Courses of Events	(A1): Actor, (S): System 1. (A1) Customer 가 현재 자판기에 판매하는 음료수를 선택한다. 2. (S) "현재 자판기의 재고를 확인한다"를 invoke하여 선택한 음료수의 재고를 확인한다. 3. (S) "결제 방법 중 하나를 선택한다"를 invoke 한다.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line2. 현재 자판기의 재고가 없다면 재고가 없다는 메시지를 출력 후 "선결제를 진행한다"를 invoke한다.

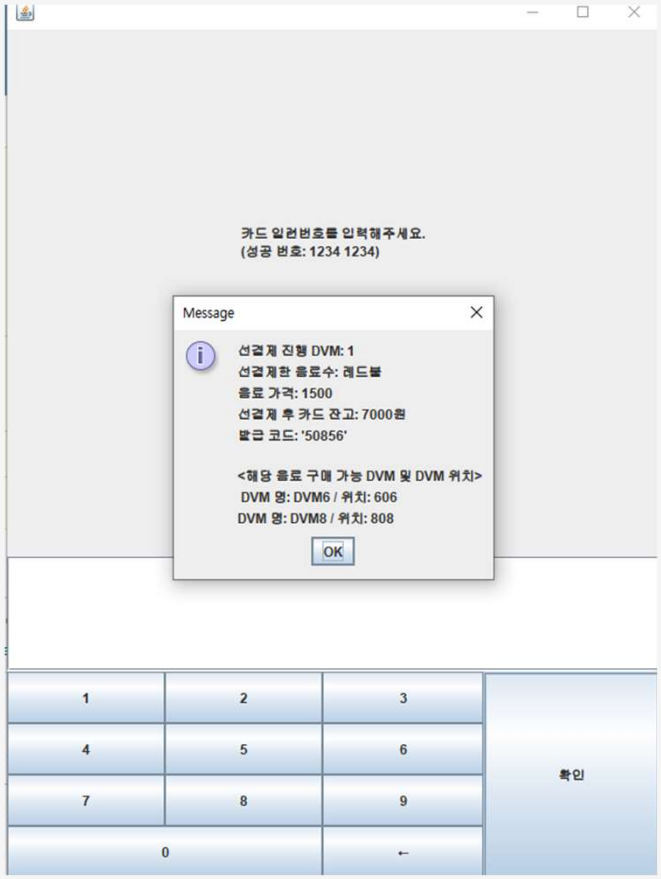
[2nd Cycle][CPT - Test Case 20] 구매한 DVM 번호가 잘못 나옴





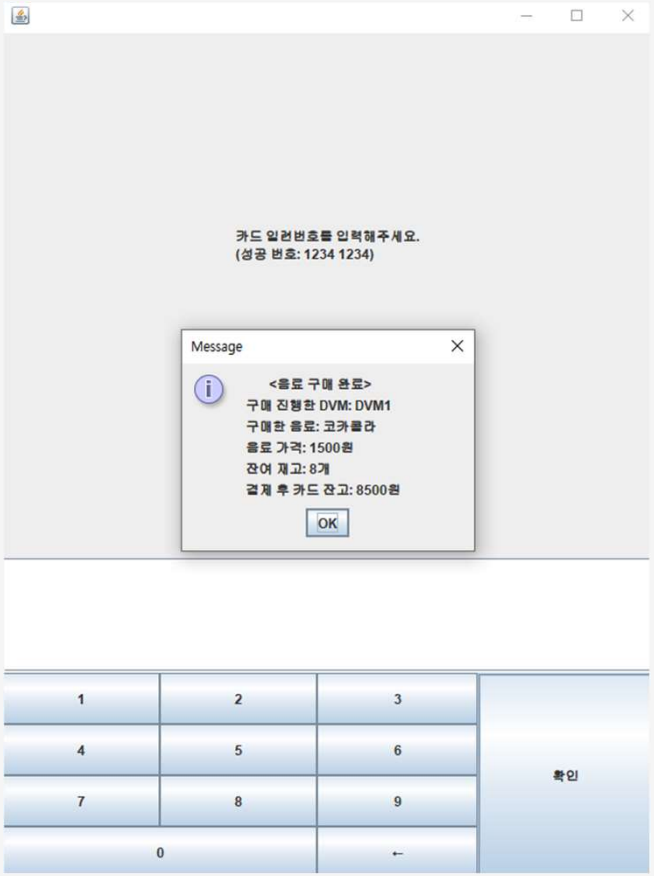
MS129

[2nd Cycle][CPT - Test Case 21] 구매한 DVM 번호가 잘못 나옴



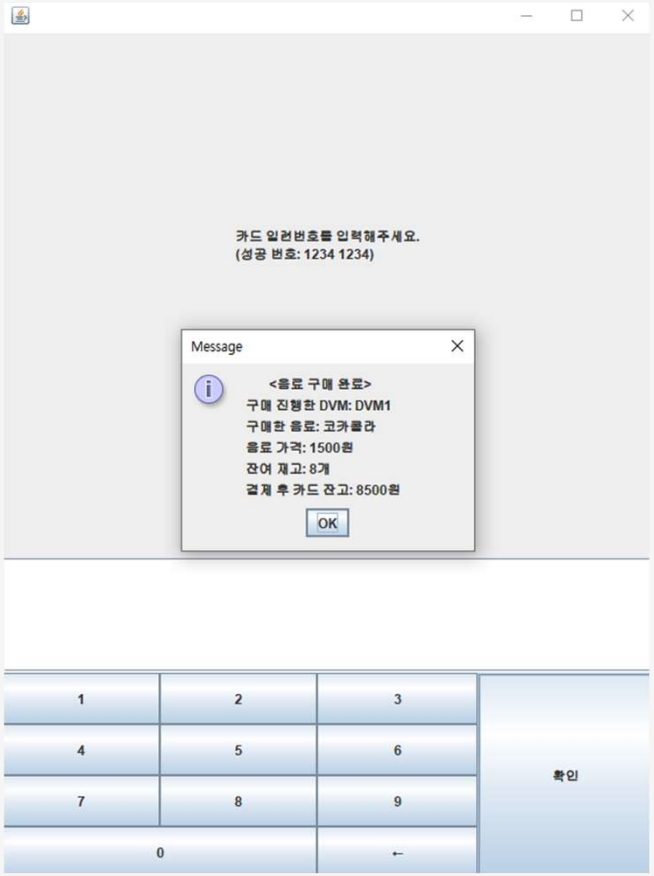
MS129

[2nd Cycle][CPT - Test Case 25] 구매한 DVM 번호가 잘못 나옴



MS129

[2nd Cycle][CPT - Test Case 26] 구매한 DVM 번호가 잘못 나옴



MS129

[2nd Cycle][findbugs] DVM1.java ~ DVM8.java switch문에서 default case missing

```java
public void run() {
    try {
        serverSocket = new ServerSocket(8000 + getDVMId());
        System.out.println("[DVM" + getDVMId() + "] SERVER ON");

        while (true) {
            receive_socket= serverSocket.accept();
            //System.out.println("Client connected");

            objectInputStream = new ObjectInputStream(receive_socket.getInputStream());
            Message msg = (Message) objectInputStream.readObject();

            System.out.println("[DVM" + getDVMId() + "] DVM" + msg.getSrc_id()
                    + "로부터 메시지 수신(유형: " + msg.getMsg_type() + ", 내용: " + msg.getMsg()+ ")");
            int type = msg.getMsg_type();
            switch (type) {
                case MsgType.REQUEST_STOCK:
                    responseStockMessage(msg);
                    break;
                case MsgType.REQUEST_LOCATION:
                    responseLocationMessage(msg);
                    break;
                case MsgType.DRINK_SALE_CHECK:
                    responseSaleMessage(msg);
                    break;
            }
            receive_socket.close();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```java
public void run() {
    try {
        serverSocket = new ServerSocket(8000 + getDVMId());
        System.out.println("[DVM" + getDVMId() + "] SERVER ON");

        while (true) {
            receive_socket= serverSocket.accept();
            //System.out.println("Client connected");

            objectInputStream = new ObjectInputStream(receive_socket.getInputStream());
            Message msg = (Message) objectInputStream.readObject();

            System.out.println("[DVM" + getDVMId() + "] DVM" + msg.getSrc_id()
                    + "로부터 메시지 수신(유형: " + msg.getMsg_type() + ", 내용: " + msg.getMsg()+ ")");
            int type = msg.getMsg_type();
            switch (type) {
                case MsgType.REQUEST_STOCK:
                    responseStockMessage(msg);
                    break;
                case MsgType.REQUEST_LOCATION:
                    responseLocationMessage(msg);
                    break;
                case MsgType.DRINK_SALE_CHECK:
                    responseSaleMessage(msg);
                    break;
                default:
                    System.out.println("잘못된 메시지 유형입니다.");
            }
            receive_socket.close();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

[2nd Cycle][findbugs] DVM1.java ~ DVM8.java : final field to static final field

설명

Unread field: DVM1.STUB_TEST_ID; should this field be static?

```
16      -     private final int STUB_TEST_ID = 999;
        16    +     private static final int STUB_TEST_ID = 999;
```

[2nd Cycle][findbugs] DVM1.java ~ DVM8.java : thread.run to start()

설명

DVM1.main(String[]) explicitly invokes run on a thread(did you mean to start it instead?)

```
    DVM1 dvm1 = new DVM1(drinkArrayList, 1, 101);
    dvm1.run();
    dvm1.start();
}

public void run() {
```

설명

Call the method Thread.start() to execute the content of the run() method in a dedicated thread.

하위 일감

[2nd Cycle][findbugs]MyFrame L268~403/ Stubtest switch default case 추가

```
394                 case 6: // 재고 있는 DVM 위치 출력
395                     //showAccessibleDVMList(pScreen);
396                     stage = 0;
397                     pScreen.removeAll();
398                     showAllDVMList(pScreen);
399                     pScreen.updateUI();
400                     break;
401                 default:
402                     System.out.println("잘못된 접근입니다.(MyFrame Stage 관련)");
403             }
```

[2nd Cycle][PMD] Message.java - Generate Constructor

[2nd Cycle] [PMD] Message.java - L55 - 2

HyeJi Yeom이(가) 9일 전에 추가함. 일분 이하 전에 수정됨.

상태:	완료
우선순위:	보통
담당자:	-
범주:	code smell&bug

설명

Each class should declare at least one constructor.

```
    public Message() {

    }
    public Message(int src_id, int dst_id, int msg_type, String msg) {
        this.src_id = src_id;
        this.dst_id = dst_id;
        this.msg_type = msg_type;
        this.msg = msg;
    }
```

[2nd Cycle][PMD] CardPayment.java - 랜덤 코드 생성 nextint 사용으로 변경

```
    public Code generateCode(Drink selected_drink){
        //5자리 코드 생성 후, 음료 객체랑 합쳐서 코드 객체 생성
        drink_info = selected_drink;
        Random random = new Random();
        int generatedCodeNum = (int)(random.nextInt( bound: 99999)+10000);
        Code generatedCode = new Code(generatedCodeNum, drink_info);
        return generatedCode;
    }
```

[2nd Cycle][PMD] Code.java/Drink.java/Card.java : private field could be made final

```
1  public class Code {
2
3      private int code;
-      private Drink drink;
4  +   private final Drink drink;
```

```
2  public class Drink {
3
-      private String name;
4  +   private final String name;
5      private int price;
6      private int stock;
-      private String imgURL;
7  +   private final String imgURL;
```

```
1  public class Card {
-      private int card_num;
2  +   private final int card_num;
```

[2nd Cycle][PMD] CodePayment.java : 자동으로 컴파일러가 생성해주는 생성자 제거

```
//    public CodePayment() {
//    }
//    public CodePayment(Code code_info, Boolean isCodeAvailable) {
//        code_info = this.code_info;
//        isCodeAvailable = this.isCodeAvailable;
//    }
```

[2nd Cycle][PMD] Controller.java/CardPayment.java : Unused field 삭제

```
//    private Card card_info;
    private OtherDVMs otherDVMs;
    private Drink selected_drink;
    private CardPayment cardPayment = new CardPayment();
    private CodePayment codePayment = new CodePayment();

    public ArrayList<Code> getCodeList() { return code_list; }
    public int getCurrentDVMIndex() { return currentDVMIndex; }
    public ArrayList<DVM> getAccessible_DVM_list() { return accessible_DVM_list; }
//    public Card getCard_info(){
//        return card_info;
//    }
```

[2nd Cycle] [Check Style] Avoid Star Import

```
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JComponent;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;
import java.awt.GridLayout;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Image;

import java.util.ArrayList;

public class MyFrame extends JFrame {
```

[2nd Cycle] [Check Style] Visibility Modifier

```
private Card card_info;
private Drink drink_info;
private boolean isPrePayment;
private ArrayList<Card> basicCardList;
```

```
public class Controller {
    private ArrayList<Code> code_list = new ArrayList<~>();
    private int currentDVMIndex;
    private ArrayList<DVM> accessible_DVM_list;
    private Card card_info;
    private OtherDVMs otherDVMs;
    private Drink selected_drink;
    private CardPayment cardPayment = new CardPayment();
    private CodePayment codePayment = new CodePayment();

    public ArrayList<Code> getCodeList() { return code_list; }
    public int getCurrentDVMIndex() { return currentDVMIndex; }
    public ArrayList<DVM> getAccessible_DVM_list() { return accessible_DVM_list; }
    public Card getCard_info() { return card_info; }
    public OtherDVMs getOtherDVMs() { return otherDVMs; }
    public Drink getSelected_drink() { return selected_drink; }
    public CardPayment getCardPayment() { return cardPayment; }
    public CodePayment getCodePayment() { return codePayment; }
```

[2nd Cycle] [Check Style] Visibility Modifier

```
public class DVM1 extends Thread implements DVM {

    private ArrayList<Drink> drink_list;
    private int id;
    private int address;
    private ServerSocket serverSocket = null;
    private Socket receive_socket = null;
    private ObjectInputStream objectInputStream = null;
    private ObjectOutputStream objectOutputStream = null;
    private static final int STUB_TEST_ID = 999;

    public ServerSocket getServerSocket() { return serverSocket; }
    public Socket getReceive_socket() { return receive_socket; }
    public ObjectInputStream getObjectInputStream() { return objectInputStream; }
    public ObjectOutputStream getObjectOutputStream() { return objectOutputStream; }
```

[2nd Cycle] [Check Style] Visibility Modifier

```
public class MyFrame extends JFrame {

    private int stage = 0;
    private int inputNum = 0;
    private int inputTemp = 0;
    private JTextField inputText = new JTextField( text: "          ", SwingConstants.CENTER); //
    private String[] num_list = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "0", "←", "확인"};
    private JButton[] dialButton_list = new JButton[12];
    private ArrayList<JLabel> labelList = new ArrayList<JLabel>();

    public int getStage() { return stage; }
    public int getInputNum() { return inputNum; }
    public int getInputTemp() { return inputTemp; }
    public JTextField getInputText() { return inputText; }
    public String[] getNum_list() { return num_list; }
    public JButton[] getDialButton_list() { return dialButton_list; }
    public ArrayList<JLabel> getLabelList() { return labelList; }
```

[2nd Cycle] [Check Style] Visibility Modifier

```
    /////////////// panel 들 ///////////////
    private JPanel pDial = new JPanel();
    private JPanel panelDown = new JPanel();
    private JPanel pTemp = new JPanel();
    private JPanel pScreen = new JPanel();
    private JPanel pInput = new JPanel();

    public JPanel getpDial() { return pDial; }
    public JPanel getPanelDown() { return panelDown; }
    public JPanel getpTemp() { return pTemp; }
    public JPanel getpScreen(){
        return pScreen;
    }
    public JPanel getpInput() { return pInput; }
```

[2nd Cycle] [Check Style] Visibility Modifier

```
/////////////// gridBagLayout 편하게 사용하려고 전역으로 선언 ///////////////
private GridBagLayout grid = new GridBagLayout();
private GridBagConstraints gbc = new GridBagConstraints();

public GridBagLayout getGrid() { return grid; }
public GridBagConstraints getGbc() { return gbc; }
// controller 객체
private Controller controller = new Controller();

public Controller getController() { return controller; }
```

[2nd Cycle] [Check Style] Visibility Modifier

```
public class Network extends Thread {
    private ArrayList<DVM> dvmList;
    public ArrayList<DVM> getDvmList(){
        return dvmList;
    }
```

```
public class OtherDVMs extends Thread{
    private ArrayList<DVM> dvmList =new ArrayList<DVM>();
    private Network network;

    public ArrayList<DVM> getDvmList() { return dvmList; }
    public Network getNetwork() { return network; }
```

[2nd Cycle] [Check Style] Unused Imports

```
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
```

```
import java.util.ArrayList;

public class CardPayment {
```

```
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.ArrayList;

public class Network extends Thread {
```

```
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;

public class DVM1 extends Thread implements DVM {
```

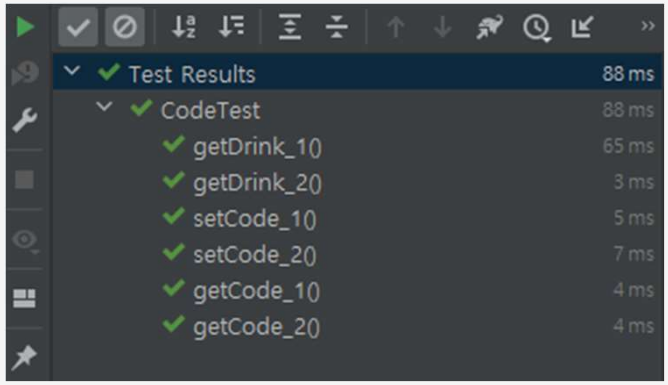
```
import java.util.ArrayList;

public class OtherDVMs extends Thread{
```

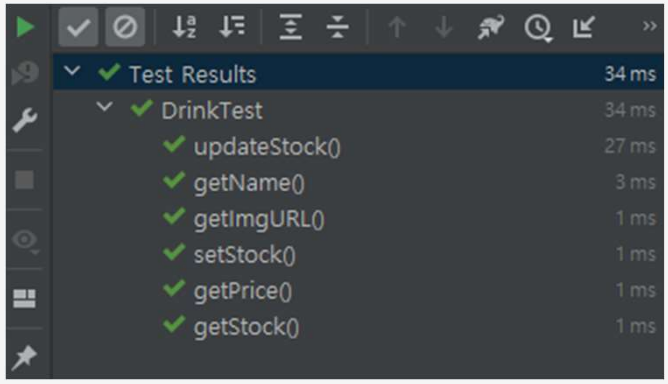
지난 Code Coverage

TestCase 추가 작성

	Coverage on New Code	Uncovered Lines on New Code	Uncovered Conditions on New Code
Card.java	100%	0	0
CardPayment.java	77.1%	4	4
Code.java	75.0%	2	0
CodePayment.java	40.0%	6	0
Controller.java	5.1%	80	32
Drink.java	90.5%	2	0
DVM.java	–	0	0
DVM1.java	8.8%	110	24
DVM2.java	9.2%	107	22
DVM3.java	8.5%	108	22
DVM4.java	8.5%	108	22
DVM5.java	8.5%	108	22
DVM6.java	8.5%	108	22
DVM7.java	8.5%	108	22
DVM8.java	8.5%	108	22
Message.java	100%	0	0
MsgType.java	0.0%	1	0
MyFrame.java	–	0	0
Network.java	30.9%	69	7
OtherDVMs.java	92.0%	15	5
StubTest.java	–	0	0
Test2.java	–	0	0

Code Test + Drink Test

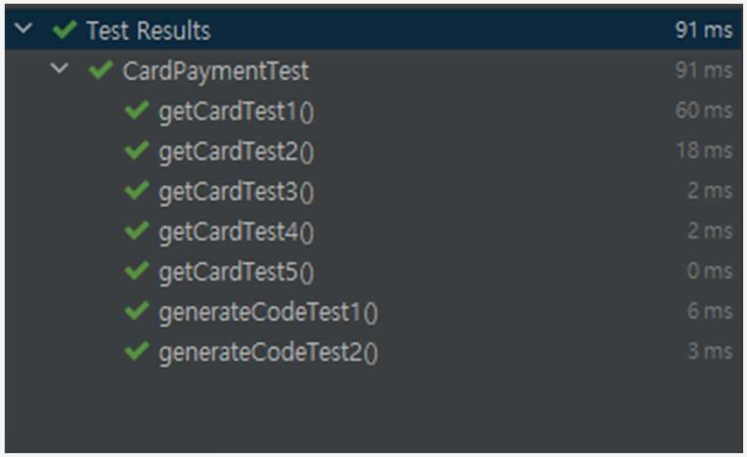





```java
@Test
void getCode_1() { // 코드가 맞게 반환되는지
    assertEquals( expected: 11111, code1.getCode());
    assertEquals( expected: 22222, code2.getCode());
    assertEquals( expected: 33333, code3.getCode());
    assertEquals( expected: 44444, code4.getCode());
    assertEquals( expected: 55555, code5.getCode());
}
@Test
void getCode_2() {
    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());
}
@Test
void setCode_1(){
    int testCode = 12345;
    code1.setCode(testCode);
    assertEquals(testCode, code1.getCode());

    int testCode2 = 54321;
    code2.setCode(testCode2);
    assertEquals(testCode2, code2.getCode());
}

@Test
void setCode_2(){
    int testCode2 = 54321;
    int testCode3 = 11111;
    code3.setCode(testCode2);
    assertNotEquals(testCode3, code3.getCode());
}

@Test
void getDrink_1() { // 음료 정보가 맞게 반환되는지
```

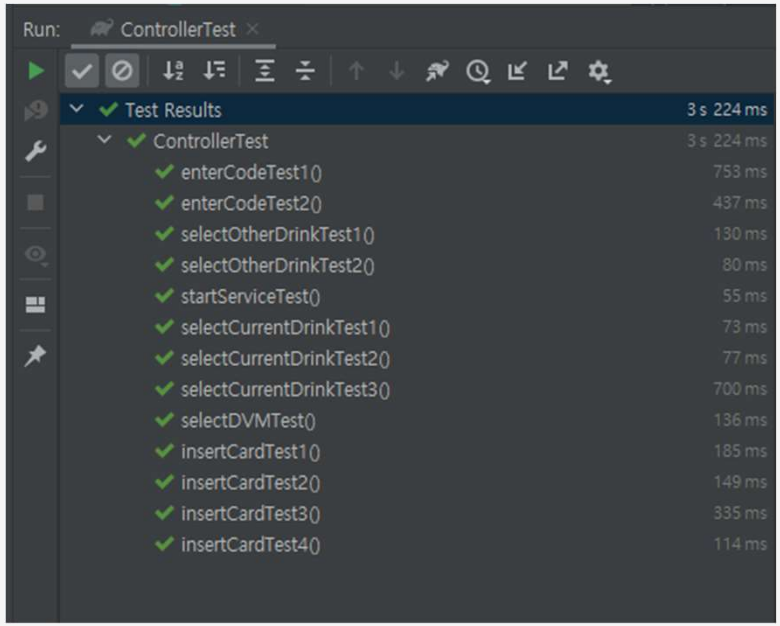
```java
@Test
void setStock(){
    drink1.setStock(8);
    assertEquals( expected: 8,drink1.getStock());

    drink2.setStock(8);
    assertEquals( expected: 8,drink2.getStock());

    drink3.setStock(3);
    assertEquals( expected: 3,drink3.getStock());

}
@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());
}

@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());
```

CardPayment Test




```java
@Test
void getCardTest5() { // 카드 넘버 확인
    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;
    int[] card_numList;

    for (int i = 0; i < 3; i++) {
        int temp_num = basicCardList.get(i).getCard_num();
        assertEquals(temp_num, basicCardList.get(i).getCard_num());
    }
}
```

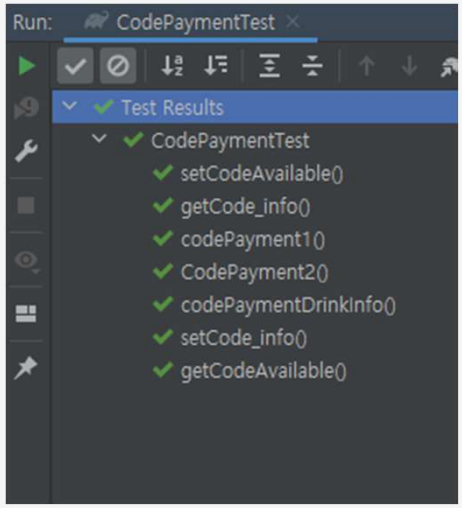
Controller Test




```java
@Test // 다른 DVM에 재고가 있는 경우
void selectOtherDrinkTest1() throws IOException {
    final int EMPTY_ALL_STOCK = 0; // 모든 DVM의 재고가 0임
    final int CUR_IN_STOCK = 1;    // 현재 DVM에 재고가 있음
    final int OTHER_IN_STOCK = 2;  // 다른 DVM에 재고가 있음
    controller.selectDVM( num: 1);

    int result = controller.selectOtherDrink( dialNum: 20);

    assertEquals(OTHER_IN_STOCK, result);
}

@Test // 모든 DVM에 재고가 없는 경우
void selectOtherDrinkTest2() throws IOException {
    final int EMPTY_ALL_STOCK = 0; // 모든 DVM의 재고가 0임
    final int CUR_IN_STOCK = 1;    // 현재 DVM에 재고가 있음
    final int OTHER_IN_STOCK = 2;  // 다른 DVM에 재고가 있음
    controller.selectDVM( num: 1);

    int result = controller.selectOtherDrink( dialNum: 19);

    assertEquals(EMPTY_ALL_STOCK, result);
}

@Test // 1. 결제 가능카드 + 재고 존재하는 음료 결제 시도
void insertCardTest1() {
    controller.selectDVM( num: 1); // 1번 DVM 선택
    controller.selectCurrentDrink( dialNum: 1); // 코카콜라 선택

    String result = controller.insertCard( card_num: 12341234, isPrepayment: false);

    String expectedResult = "       <음료 구매 완료>" +
            "\n구매 진행한 DVM: DVM1"
            + "\n구매한 음료: 코카콜라"
            + "\n음료 가격: 1500원"
            + "\n잔여 재고: 9개"
            + "\n결제 후 카드 잔고: "
            + "8500원";
    assertEquals(expectedResult, result);
}
```


MS129

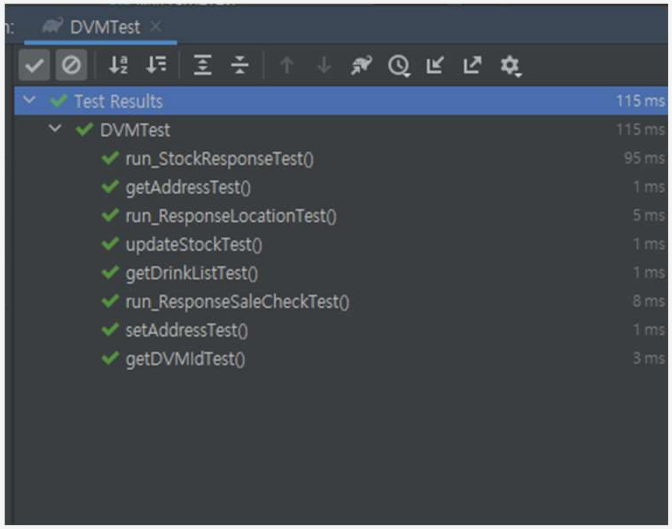
Codepayment Test




```java
@Test
void setCode_info() {
    CodePayment codePayment = new CodePayment();
    codePayment.setCode_info(code_info1);
    assertEquals(code_info1,codePayment.getCode_info());

    codePayment.setCode_info(code_info2);
    assertEquals(code_info2,codePayment.getCode_info());

    codePayment1.setCode_info(code_info4);
    assertEquals(code_info4, codePayment1.getCode_info());

}

@Test
void getCodeAvailable() {
    codePayment1.setCode_info(code_info3);
    codePayment1.setCodeAvailable(true);
    assertTrue(codePayment1.getCodeAvailable());
}

@Test
void setCodeAvailable() {
    codePayment1.setCodeAvailable(false);
    assertEquals( expected: false, codePayment1.getCodeAvailable());
}
```

```java
@Test
void CodePayment2(){
    CodePayment cp1 = new CodePayment();
    CodePayment cp2 = new CodePayment();
    cp1.setCodeAvailable(true);
    cp1.setCode_info(code_info1);
    cp2.setCodeAvailable(false);
    cp2.setCode_info(code_info2);

    assertEquals( expected: true, cp1.getCodeAvailable());
    assertEquals( expected: false, cp2.getCodeAvailable());

    assertEquals(code_info1, cp1.getCode_info());
    assertEquals(code_info2, cp2.getCode_info());
}

@Test
void codePaymentDrinkInfo(){
    CodePayment cp1 = new CodePayment(code_info1,  isCodeAvailable: true);
    assertEquals(drink_info1,cp1.codePayment(code_info1));
}
```

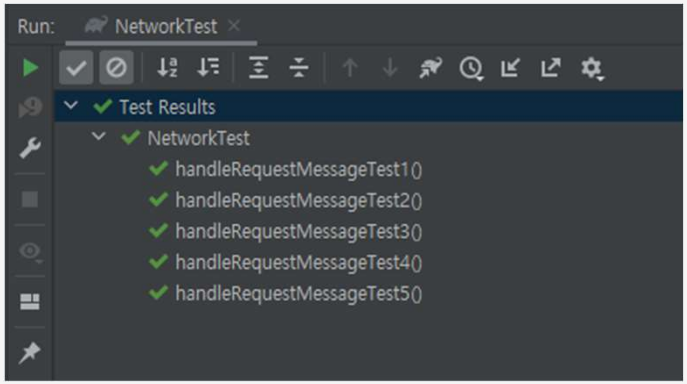
DVM test




```java
@Test
void run_ResponseSaleCheckTest() throws IOException {
    int SOURCE_ID = 999;
    initDrinkList();
    DVM dvm = new DVM(drinkList, id: 8, address: 808);
    dvm.setServerPort();
    Socket socket = new Socket( host: "localhost", port: 8000 + dvm.getDVMId());
    Socket socket2 = new Socket( host: "localhost", port: 8000 + dvm.getDVMId());
    DummySocket dummySocket = new DummySocket();
    Message message = new Message();
    message.createMessage(SOURCE_ID, dst_id: 8, MsgType.DRINK_SALE_CHECK, msg: "레드불");
    dummySocket.sendSocket(socket, message);
    Message message2 = new Message();
    message2.createMessage(SOURCE_ID, dst_id: 8, msg_type: 999);
    dummySocket.sendSocket(socket2, message2);
    dvm.run();
    Message receivedMessage = dummySocket.receiveSocket(socket);
    dvm.closeServerPort();

    assertEquals( expected: "19", receivedMessage.getMsg());
    assertEquals(MsgType.DRINK_SALE_RESPONSE, receivedMessage.getMsg_t
    assertEquals(SOURCE_ID, receivedMessage.getDst_id());
    assertEquals(dvm.getDVMId(), receivedMessage.getSrc_id());
}
```

```java
@Test
void getAddressTest(){
    Drink drink = new Drink( name: "코카콜라", price: 1500, stock: 10, imgURL: "src/main/resources/image/1.
    drinkList.add(drink);
    DVM dvm = new DVM(drinkList, id: 8, address: 808);
    int address = dvm.getAddress();
    assertEquals( expected: 808, address);
}

@Test
void setAddressTest(){
    Drink drink = new Drink( name: "코카콜라", price: 1500, stock: 10, imgURL: "src/main/resources/image/1.
    drinkList.add(drink);
    DVM dvm = new DVM(drinkList, id: 8, address: 808);
    int newAddress = 404;
    dvm.setAddress(newAddress);
    assertEquals(newAddress, dvm.getAddress());
}

@Test
void updateStockTest(){
    Drink drink = new Drink( name: "코카콜라", price: 1500, stock: 10, imgURL: "src/main/resources/image/1.
    drinkList.add(drink);
    DVM dvm = new DVM(drinkList, id: 8, address: 808);
    dvm.updateStock(drink);
    assertEquals( expected: 9, dvm.getDrink_list().get(0).getStock());
```

Network test




```java
@Test // 1. REQUEST_STOCK, dst_id != 0 인 경우
void handleRequestMessageTest1(){
    Network network = new Network(dvmList);
    Message message = new Message();
    message.createMessage( src_id: 999,  dst_id: 1, MsgType.REQUEST_STOCK,  msg: "코카콜라");

    int stock =  (int) network.handleRequestMessage(message);

    assertEquals( expected: 10, stock);
}

@Test // 2. REQUEST_STOCK, dst_id == 0 인 경우
void handleRequestMessageTest2(){
    Network network = new Network(dvmList);
    Message message = new Message();
    message.createMessage( src_id: 999,  dst_id: 0, MsgType.REQUEST_STOCK,  msg: "코카콜라");

    ArrayList<DVM> dvmList = (ArrayList<DVM>) network.handleRequestMessage(message);

    assertEquals( expected: 8, dvmList.size());
}

@Test // 3. REQUEST_LOCATION 인 경우
void handleRequestMessageTest3(){
    Network network = new Network(dvmList);
    Message message = new Message();
```

Overall Review

4학년이 제공한 CTIP 환경에 대한 소감 :

장점 : 코드가 빌드되고 그 결과를 개발자들에게 피드백해주는 과정에 소요되는 시간이 감소됨
수동적인 반복 작업들을 줄일 수 있음
품질 도구들을 활용하여 코드 품질 검토가 가능하며, 테스트 코드 커버리지 증가에도 영향을 줄 수 있음

단점 : 필요하지 않다고 느끼는 피드백들이 종종 있음, 아무래도 툴을 이용한 검사이기 때문에 사람이라면 잡지 않을 사소한 문제점까지 잡아냄

개선방향 : 중복되는 피드백들이 종종 있었는데, 이를 통합해서 보여줬으면 좋겠음.

UP (OOPT)를 기반으로 SW 개발한 방식에 대한 소감 :

장점 : 체계적으로 서비스를 구현할 수 있는게 가장 큰 장점, 구현하고자 하는 방향이 무엇인지에 대해 계속 생각할 수 있음.

단점 : 반복적인 작업과 많은 문서량으로 인해 지칠 때가 있음, 또한 어느 한 단계가 잘못 설계된다면 다시 돌아가서 전의 단계들을 수정해야하므로 항상 신중함이 요구됨

개선방향 : 개발 과정에서 앞선 설계의 수정이 필요함을 알게 되었을 때 체계적으로 연관된 문서수정을 할 수 있도록 수정 단계별 메뉴얼이 있으면 좋겠음
프로젝트 참여자들의 협동,의사소통에 대한 가이드가 따로 없는 것이 개선되면 좋겠음

THANK YOU

감사합니다.