DVM System

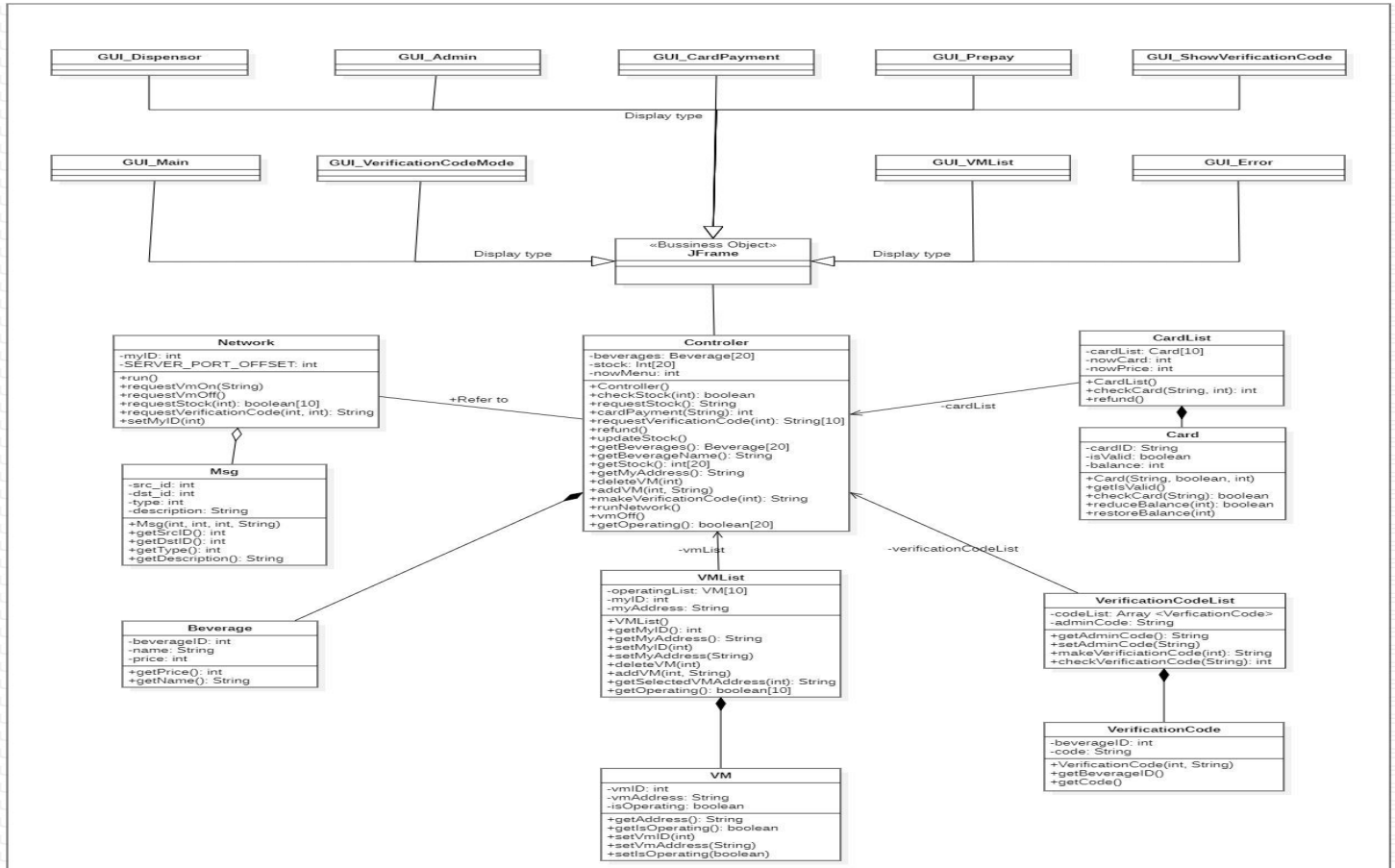
1st Cycle development

Team 2
201810286 박혜린
201811264 변장훈
201311312 정인욱

INDEX

01 Refine Class Diagram

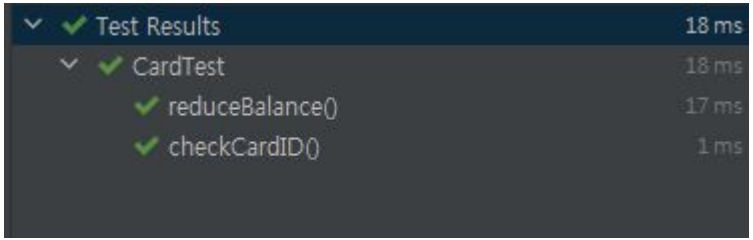
1 Refine Class Diagram



02 Write Unit Test Code & Result

2 Write Unit Test Code & Result

- **Class Card**




```java
import ...

class CardTest {
    @Test
    void checkCardID() {
        Card card = new Card( cardID: "cardID", isValid: false, balance: 1000);

        assertEquals(card.checkCardID( cid: "123"), actual: false);
        assertNotNull(card.checkCardID( cid: "123"));
        assertSame(card.checkCardID( cid: "123"), actual: false);

        assertEquals(card.checkCardID( cid: "cardID"), actual: true);
        assertNotNull(card.checkCardID( cid: "cardID"));
        assertSame(card.checkCardID( cid: "cardID"), actual: true);
    }
    @Test
    void reduceBalance() {
        Card card = new Card( cardID: "cardID", isValid: false, balance: 1000);

        assertEquals(card.reduceBalance( amount: 1500), actual: false);
        assertSame(card.reduceBalance( amount: 1500), actual: false);
        assertNotNull(card.reduceBalance( amount: 1500));

        assertEquals(card.reduceBalance( amount: 500), actual: true);
        assertSame(card.reduceBalance( amount: 500), actual: true);
        assertNotNull(card.reduceBalance( amount: 500));

        assertEquals(card.reduceBalance( amount: -1), actual: true);
        assertSame(card.reduceBalance( amount: -1), actual: true);
        assertNotNull(card.reduceBalance( amount: -1));
    }
}
```

2 Write Unit Test Code & Result

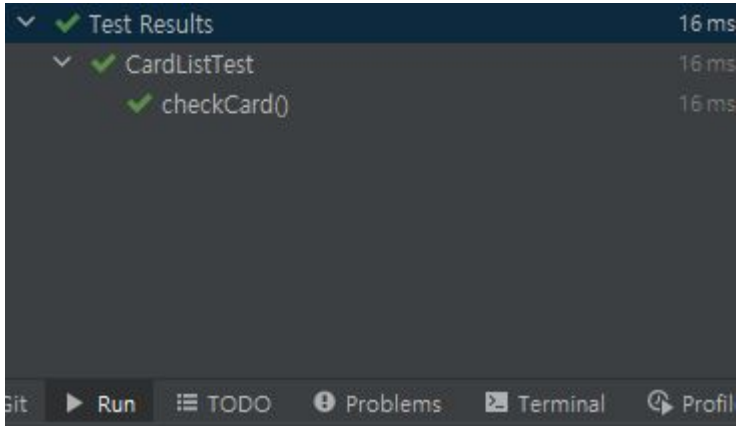
- **Class CardList**

```java
package tests;

import ...

class CardListTest {

    @Test
    void checkCard() {
        CardList cardList = new CardList();
        /* 해당하는 카드가 없는 경우 */
        assertEquals(cardList.checkCard( cardNum: "123451234512", nowPrice: 423), actual: 1);
        assertEquals(cardList.checkCard( cardNum: "52431", nowPrice: 4123), actual: 1);
        assertEquals(cardList.checkCard( cardNum: "622435252534", nowPrice: 42523), actual: 1);
        /* 해당하는 카드가 있지만 유효하지 않을 경우 */
        assertEquals(cardList.checkCard( cardNum: "444444444444", nowPrice: 2000), actual: 2);
        assertEquals(cardList.checkCard( cardNum: "666666666666", nowPrice: 0), actual: 2);
        assertEquals(cardList.checkCard( cardNum: "888888888888", nowPrice: 30000), actual: 2);
        /* 해당하는 카드가 있고 유효하지만 잔액이 없는 경우 */
        assertEquals(cardList.checkCard( cardNum: "111111111111", nowPrice: 999000), actual: 3);
        assertEquals(cardList.checkCard( cardNum: "777777777777", nowPrice: 1000), actual: 3);
        assertEquals(cardList.checkCard( cardNum: "999999999999", nowPrice: 50000), actual: 3);
        /* 해당하는 카드가 있고, 유효하고, 잔액이 충분한 경우 */
        assertEquals(cardList.checkCard( cardNum: "111111111111", nowPrice: 500), actual: 0);
        assertEquals(cardList.checkCard( cardNum: "555555555555", nowPrice: 9200), actual: 0);
    }
}
```



2 Write Unit Test Code & Result

- **Class Controller**

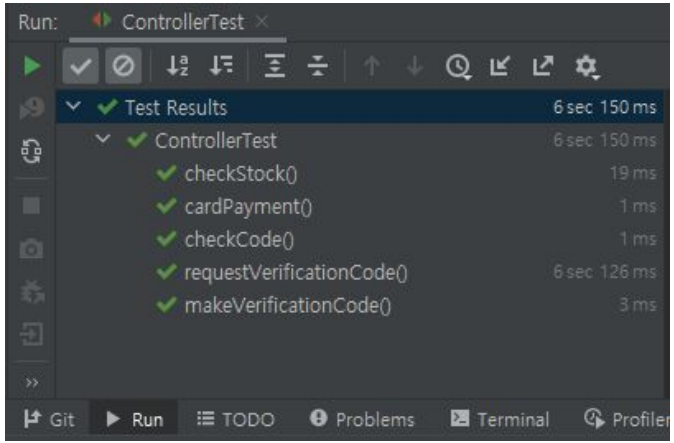
```
package tests;

import ...

class ControllerTest {
    Controller controller = new Controller();
    @Test
    void checkStock() {

        assertEquals(controller.checkStock( beverageID: 0), actual: true);
        assertEquals(controller.checkStock( beverageID: 1), actual: true);
        assertEquals(controller.checkStock( beverageID: 2), actual: true);
        assertEquals(controller.checkStock( beverageID: 3), actual: true);
        assertEquals(controller.checkStock( beverageID: 4), actual: true);
        assertEquals(controller.checkStock( beverageID: 5), actual: true);
        assertEquals(controller.checkStock( beverageID: 6), actual: true);
        assertEquals(controller.checkStock( beverageID: 10), actual: false);
        assertEquals(controller.checkStock( beverageID: 14), actual: false);
        assertEquals(controller.checkStock( beverageID: 19), actual: false);
    }

    @Test
    void cardPayment() {
        controller.checkStock( beverageID: 0);
        assertEquals(controller.cardPayment( cardID: "111111111111"), actual: 0);
        controller.checkStock( beverageID: 1);
        assertEquals(controller.cardPayment( cardID: "111111111111"), actual: 0);
        controller.checkStock( beverageID: 2);
        assertEquals(controller.cardPayment( cardID: "131313131313"), actual: 1);
        controller.checkStock( beverageID: 3);
```



2 Write Unit Test Code & Result

- **Class VerificationCodeList**

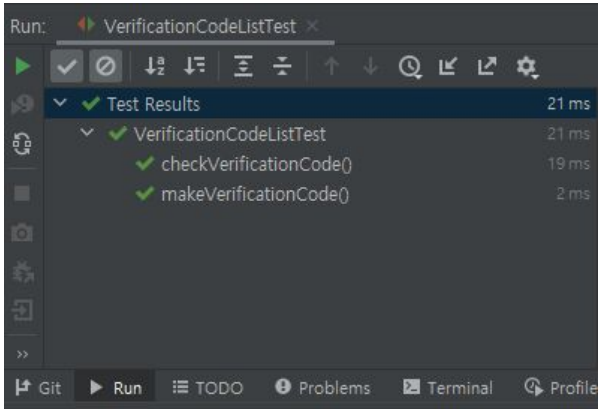
```
package tests;

import ...

class VerificationCodeListTest {
    VerificationCodeList verificationCodeList = new VerificationCodeList();
    @Test
    void makeVerificationCode() {
        assertNotNull(verificationCodeList.makeVerificationCode( beverageID: 1));
        assertNotNull(verificationCodeList.makeVerificationCode( beverageID: 2));
        assertNotNull(verificationCodeList.makeVerificationCode( beverageID: 3));
        assertNotNull(verificationCodeList.makeVerificationCode( beverageID: 6));
        assertNotNull(verificationCodeList.makeVerificationCode( beverageID: 8));
        assertNotNull(verificationCodeList.makeVerificationCode( beverageID: 10));
        assertNotNull(verificationCodeList.makeVerificationCode( beverageID: 14));
        assertNotNull(verificationCodeList.makeVerificationCode( beverageID: 18));
        assertNotNull(verificationCodeList.makeVerificationCode( beverageID: 19));
    }

    @Test
    void checkVerificationCode() {
        assertEquals(verificationCodeList.checkVerificationCode("123"), actual: -1);
        assertEquals(verificationCodeList.checkVerificationCode("153445"), actual: -1);
        assertEquals(verificationCodeList.checkVerificationCode("14523"), actual: -1);
        assertEquals(verificationCodeList.checkVerificationCode("12"), actual: -1);

        assertEquals(verificationCodeList.checkVerificationCode(verificationCodeList.makeVerificationCode( beverageID: 0)), actual: 0);
        assertEquals(verificationCodeList.checkVerificationCode(verificationCodeList.makeVerificationCode( beverageID: 1)), actual: 1);
        assertEquals(verificationCodeList.checkVerificationCode(verificationCodeList.makeVerificationCode( beverageID: 2)), actual: 2);
        assertEquals(verificationCodeList.checkVerificationCode(verificationCodeList.makeVerificationCode( beverageID: 3)), actual: 3);
```



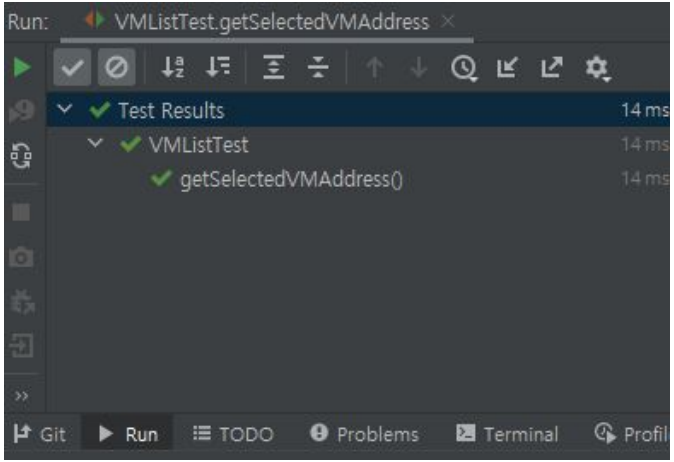
2 Write Unit Test Code & Result

- **Class VMList**

```java
package tests;

import ...

class VMListTest {
    VMList vmList = new VMList();
    @Test
    void getSelectedVMAddress() {
        assertEquals(vmList.getSelectedVMAddress( index: 0),  actual: null);
        assertEquals(vmList.getSelectedVMAddress( index: 1),  actual: null);
        vmList.addVM( vmID: 1, address: "편의점");
        assertEquals(vmList.getSelectedVMAddress( index: 0),  actual: "편의점");
        assertEquals(vmList.getSelectedVMAddress( index: 3),  actual: null);
        assertEquals(vmList.getSelectedVMAddress( index: 4),  actual: null);
        assertEquals(vmList.getSelectedVMAddress( index: 5),  actual: null);
        vmList.addVM( vmID: 5, address: "건대입구역");
        assertEquals(vmList.getSelectedVMAddress( index: 4),  actual: "건대입구역");
        assertEquals(vmList.getSelectedVMAddress( index: 6),  actual: null);
        vmList.addVM( vmID: 8, address: "인천공항");
        assertEquals(vmList.getSelectedVMAddress( index: 7),  actual: "인천공항");
        assertEquals(vmList.getSelectedVMAddress( index: 9),  actual: null);
    }
}
```



03 System Testing

3 System Testing

num	TEST	description	Use Case	System Function	Pass
1	카드 결제 기능	유효하지 않은 카드로 결제가 가능한지 확인한다.	Card Payment	R 1.1	P
2	카드 결제 기능	잔액이 없는 카드로 결제가 가능한지 확인한다.	Card Payment	R 1.1	P
3	카드 결제 기능	정상 결제된 경우 잔액이 메뉴의 가격만큼 차감되었는지 확인한다.	Card Payment	R 1.1	P
4	카드 결제 기능 (선결제)	유효하지 않은 카드로 결제가 가능한지 확인한다.	Card Payment(Prepay)	R 1.2	P
5	카드 결제 기능 (선결제)	잔액이 없는 카드로 결제가 가능한지 확인한다.	Card Payment(Prepay)	R 1.2	P
6	카드 결제 기능 (선결제)	정상 결제된 경우 잔액이 메뉴의 가격만큼 차감되었는지 확인한다.	Card Payment(Prepay)	R 1.2	P
7	재고 관리 기능	실제 재고와 측정된 재고값을 비교한다.	Manage Stock	R 2.1	P
8	관리자 모드 진입 기능	올바른 관리자 코드를 입력했을때 관리자 모드로 전환하는지 확인한다.	Input Admin Code	R 2.2	P
9	관리자 모드 진입 기능	올바르지 않은 관리자 코드를 입력했을때 오류 문구가 출력되고 메뉴 선택 대기 상태로 전환하는지 확인한다.	Input Admin Code	R 2.2	P

10	다른 자판기 재고 여부 응답 기능	다른 자판기로부터 재고 여부 확인 메시지를 수신한 경우 해당 메뉴의 재고 여부를 메시지로 응답하는지 확인한다.	Respond Stock	R 2.3	P
11	메뉴 선택 기능	재고가 있는 메뉴를 선택했을때 카드 결제 대기 상태로 전환하는지 확인한다.	Select Menu	R 3.1	P
12	메뉴 선택 기능	재고가 없는 메뉴를 선택했을때 다른 자판기로 재고 여부 메시지가 전송되는지 확인하는지 확인한다.	Select Menu	R 3.1	P
13	인증코드 입력 기능	유효한 인증코드를 입력한 경우 해당 음료가 제공되는지 확인한다.	Input Verification Code	R 3.2	P
14	인증코드 입력 기능	유효하지 않은 인증코드를 입력한 경우 오류 문구가 출력되고 메뉴 선택 대기 상태로 전환하는지 확인한다.	Input Verification Code	R 3.2	P
15	다른 자판기 선택 기능	자판기를 선택했을때 Card Payment가 실행되는지 확인한다.	Select Other DVM	R 3.3	P
16	자판기 시작 메시지 응답 기능	다른 자판기로부터 자판기 시작 메시지를 수신한 경우 해당 자판기에게 작동중임을 알리는 메시지를 전송하는지 확인한다.	Respond Vm On	R 4.1	P
17	자판기 시작 메시지 응답 기능	다른 자판기로부터 자판기 시작 메시지를 수신한 경우 VMlist에 해당 자판기를 추가하는지 확인한다.	Respond Vm On	R 4.1	P

18	자판기 종료 기능	자판기 종료를 선택할 경우 VMlist의 다른 자판기들에게 자판기 종료 메시지를 전송하는지 확인한다.	Request Vm Off (vmlist)	R 4.2	P
19	자판기 종료 메시지 응답 기능	다른 자판기로부터 자판기 종료 메시지를 수신한 경우 VMlist에서 해당 자판기가 삭제되는지 확인한다.	Respond Vm Off	R 4.3	P
20	작동중임을 알려주는걸 수신하는 기능	다른 자판기로부터 작동중임을 알리는 메시지를 수신한 경우 해당 자판기가 VMlist에서 추가되는지 확인한다.	Receive Operating	R 4.4	P
21	재고 여부 수신 기능	다른 자판기로부터 재고 여부 메시지 (재고 있음)를 수신한 경우 해당 자판기가 선결제 선택 자판기에 포함되었는지 확인한다.	Receive Stock	R 4.5	P
22	재고 여부 수신 기능	다른 자판기로부터 재고 여부 메시지 (재고 없음)를 수신한 경우 해당 자판기가 선결제 선택 자판기에 제외되었는지 확인한다.	Receive Stock	R 4.5	P
23	환불 수신 기능	다른 자판기로부터 환불 요청 메시지를 수신한 경우 결제했던 카드의 잔액이 정상적으로 복구되는지 확인한다.	Respond Refund	R 4.6	P
24	생성된 인증코드 수신 기능	다른 자판기로부터 생성된 인증코드를 메시지로 수신할 경우 해당 인증코드를 30초간 정상 출력하는지 확인한다.	Receive Verification Code	R 4.7	P

25	인증코드 생성 응답 기능	다른 자판기로부터 인증코드 생성 요청 메시지를 수신할 경우 & 선결제 대상 음료의 재고가 1 이상일 경우 인증코드를 생성하여 해당 자판기에게 전송하는지 확인한다.	Respond Verification Code	R 4.8	P
26	인증코드 생성 응답 기능	다른 자판기로부터 인증코드 생성 요청 메시지를 수신할 경우 & 선결제 대상 음료의 재고가 0일 경우 해당 자판기에게 환불 요청 메시지를 전송하는지 확인한다.	Respond Verification Code	R 4.8	P
27	자판기 설정	입력한 ID가 1부터 10 범위 내의 정수가 아닐 경우 프로그램이 종료되는지 확인한다.	Set DVM Setting	R 4.9	P
28	자판기 설정	입력한 주소의 길이가 10 초과일 경우 프로그램이 종료되는지 확인한다.	Set DVM Setting	R 4.9	P
29	자판기 설정	입력한 주소가 null일 경우 프로그램이 종료되는지 확인한다.	Set DVM Setting	R 4.9	P
30	자판기 설정	입력한 관리자 코드에 숫자 이외 다른 문자가 포함되어 있을 때 프로그램이 종료되는지 확인한다.	Set DVM Setting	R 4.9	P
31	자판기 설정	입력한 관리자 코드의 길이가 6자리가 아닐 경우 프로그램이 종료되는지 확인한다.	Set DVM Setting	R 4.9	P

04 Traceability Analysis

4 Traceability Analysis

Essential Use Case	S-Link
Card Payment	S1,S2
Card Payment (Prepay)	S1,S3
Manage Stock	S4
Input Admin Code	S5
Select Menu	S6,S7
Input Verification Code	S9
Select Other DVM	S10
Respond Stock	S7,S11
Set DVM Setting	S8,S12
Respond Vm On	S12,S13
Request Vm Off (vmlist)	S14,S15
Respond Vm Off	S15
Receive Operating	S13
Receive Stock	S7
Respond Refund	S16
Receive Verification Code	S17
Respond Verification Code	S3,S17

SID	Operation in Sequence Diagram	M_Link
S1	inputCard()	M12, M13, M17, M18, M19, M22, M23, M46, M49, M50, M51
S2	offerBeverage()	M13
S3	requestVerificationCode()	M1,M7,M8,M9,M10,M11,M32, M55, M57
S4	setBeverage()	M20
S5	inputAdminCode()	M53
S6	selectMenu()	M4, M12, M15, M16,M38,M39,M40,M41
S7	requestStock()	M1,M7,M8,M9,M10,M11,M32
S8	inputVmInfo()	M2,M6,M14,M21,M28,M33,M34,M35,M37,M42,M43,M44,M45,M48, M54
S9	inputCode()	M56, M58, M59
S10	selectPrePayDVM()	M5, M17
S11	sendStock()	M8,M9,M10,M11
S12	requestVmOn()	M1,M7, M24,M26,M31,M32,M33,M37,M43,M44
S13	sendOperating()	M8,M9,M10,M11,M26,M30,M32
S14	selectVmOff()	M3,M39,M41
S15	requestVmOff()	M1,M7,M8,M9,M10,M11,M25,M29,M32,M36,M44
S16	requestRefund()	M7,M8,M9,M10,M11,M32,M47, M52
S17	sendVerificationCode()	M8,M9,M10,M11,M27

4 Traceability Analysis

MID	Method	Class
M1	run()	Network
M2	requestVmOn(String)	Network
M3	requestVmOff()	Network
M4	requestStock(int) : boolean[10]	Network
M5	requestVerificationCode(int,int) : String	Network
M6	setMyID(int)	Network
M7	Msg(int,int,int,String)	Msg
M8	getSrcID() : int	Msg
M9	getDstID() : int	Msg
M10	getType() : int	Msg
M11	getDescription() : String	Msg
M12	getPrice() : int	Beverage
M13	getName() : String	Beverage
M14	Controller()	Controler
M15	checkStock(int) : boolean	Controler
M16	requestStock() : String	Controler
M17	cardPayment(String) : int	Controler
M18	requestVerificationCode(int) : String	Controler
M19	refund()	Controler
M20	updateStock()	Controler
M21	getBeverage() : Beverage[20]	Controler
M22	getBeverageName() : String	Controler
M23	getStock() : int[20]	Controler
M24	getMyAddress() : String	Controler
M25	deleteVM(int,String)	Controler
M26	addVM(int,String)	Controler
M27	makeVerificationCode(int) : String	Controler
M28	runNetwork()	Controler
M29	vmOff()	Controler
M30	getOperating() : boolean[20]	Controler
M31	VMList()	VMList
M32	getMyID() : int	VMList

Traceability Analysis

M33	getMyAddress() : String	VMList
M34	setMyID(int)	VMList
M35	setMyAddress(String)	VMList
M36	deleteVM(int)	VMList
M37	addVM(int,String)	VMList
M38	getSelectedVMAddress(int) : String	VMList
M39	getOperating() : boolean[10]	VMList
M40	getAddress() : String	VM
M41	getIsOperating() : boolean	VM
M42	setVmID(int)	VM
M43	setVmAddress(String)	VM
M44	setIsOperating(boolean)	VM
M45	CardList()	CardList
M46	checkCard(String,int) : int	CardList
M47	refund()	CardList
M48	Card(String,boolean,int)	Card

M49	getIsValid()	Card
M50	checkCard(String) : boolean	Card
M51	reduceBalance(int) : boolean	Card
M52	restoreBalance(int)	Card
M53	getAdminCode() : String	VerificationCodeList
M54	setAdminCode(String)	VerificationCodeList
M55	makeVerificationCode(int) : String	VerificationCodeList
M56	checkVerificationCode(String) : int	VerificationCodeList
M57	VerificationCode(int,String)	VerificationCode
M58	getBeverageID()	VerificationCode
M59	getCode()	VerificationCode

05 시연 영상

https://www.youtube.com/watch?v=o4T7llqfXAU

감사합니다