

객체지향 개발 방법론

# DVM Project 001

#4조



201710240

이해림



201811217

이해인



201711836

송호영

# Index

Activity 2051. Implement Class & Methods Definitions

Activity 2055. Write Unit Test Code

Activity 2061. Unit Testing

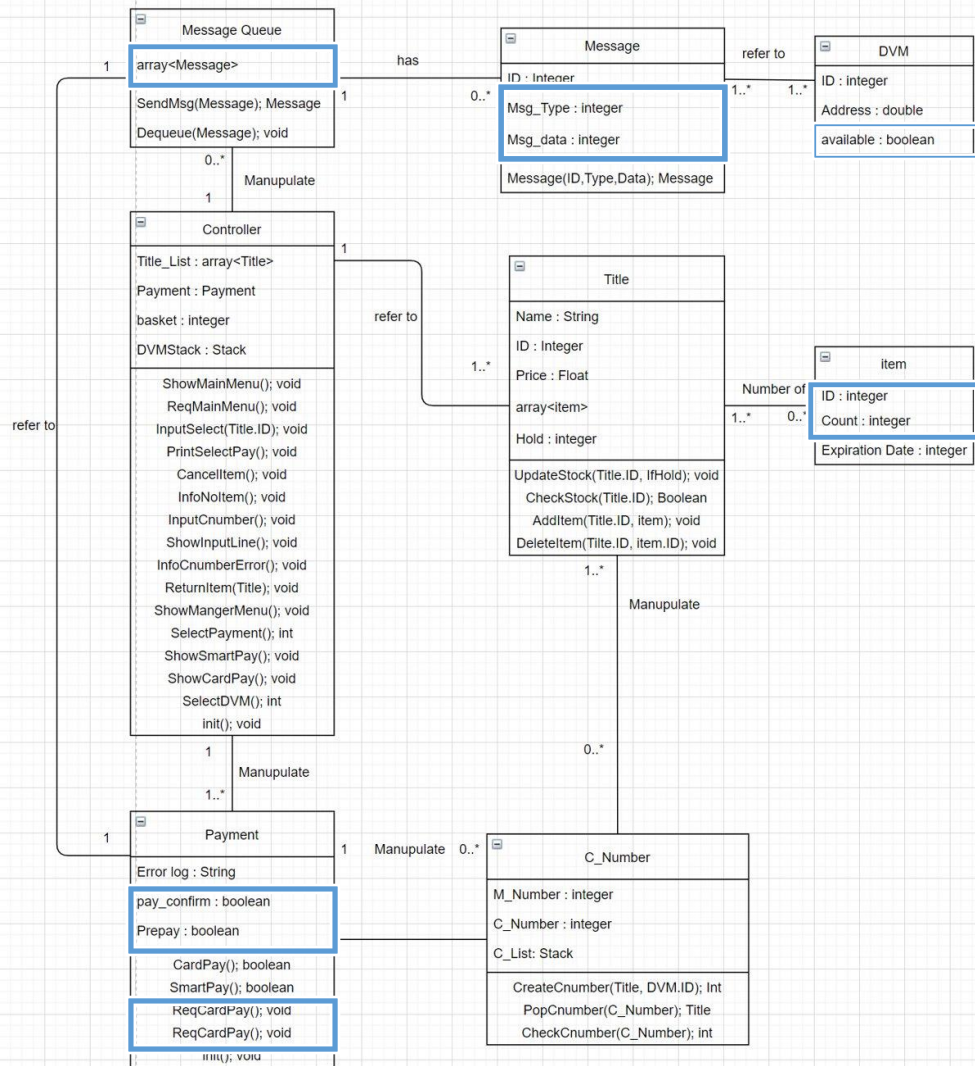
Activity 2063. System Testing

Activity 2066. Testing Traceability Analysis

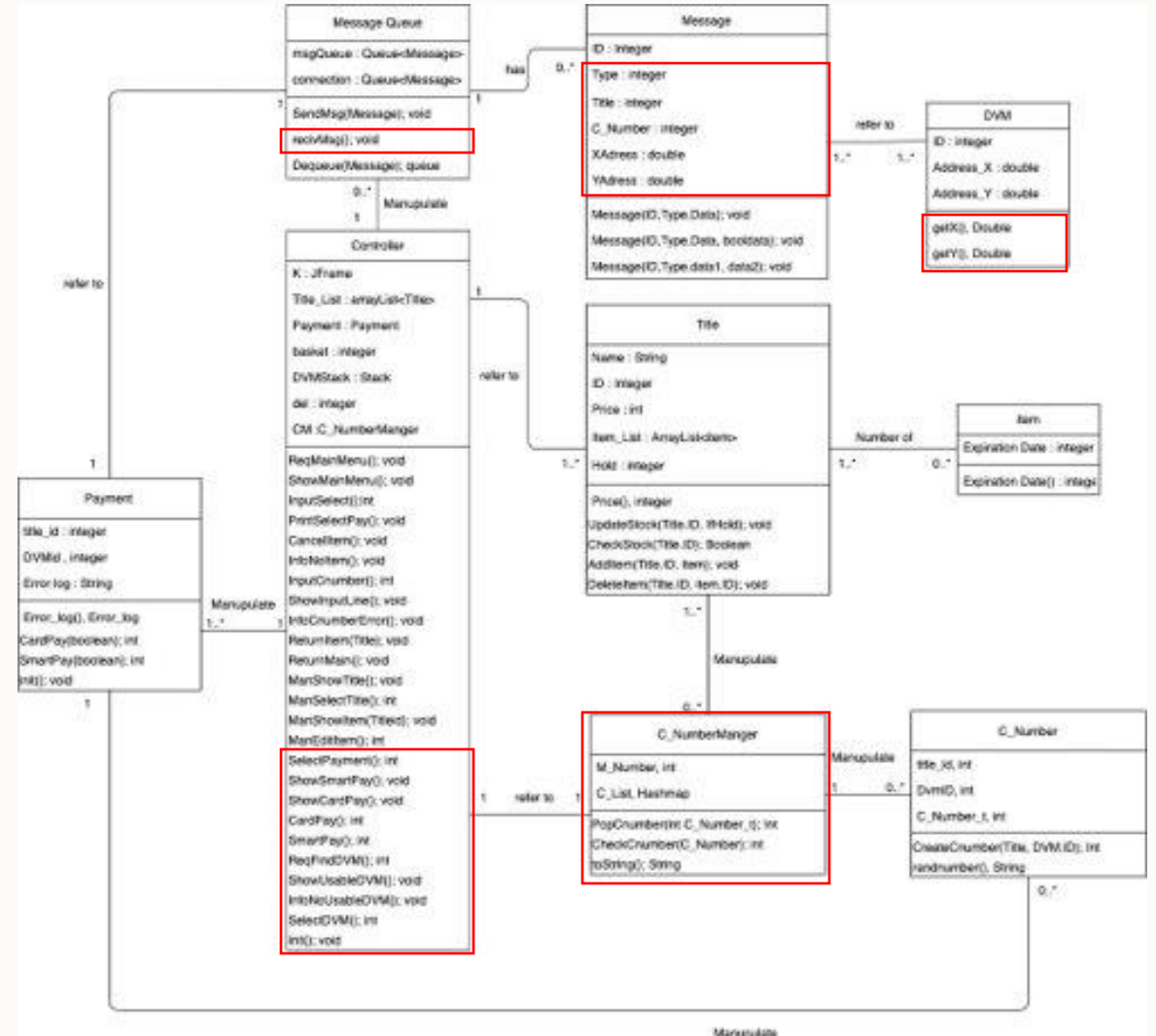
Demonstration video

# Activity 2051. Implement Class & Methods Definitions

수정 전

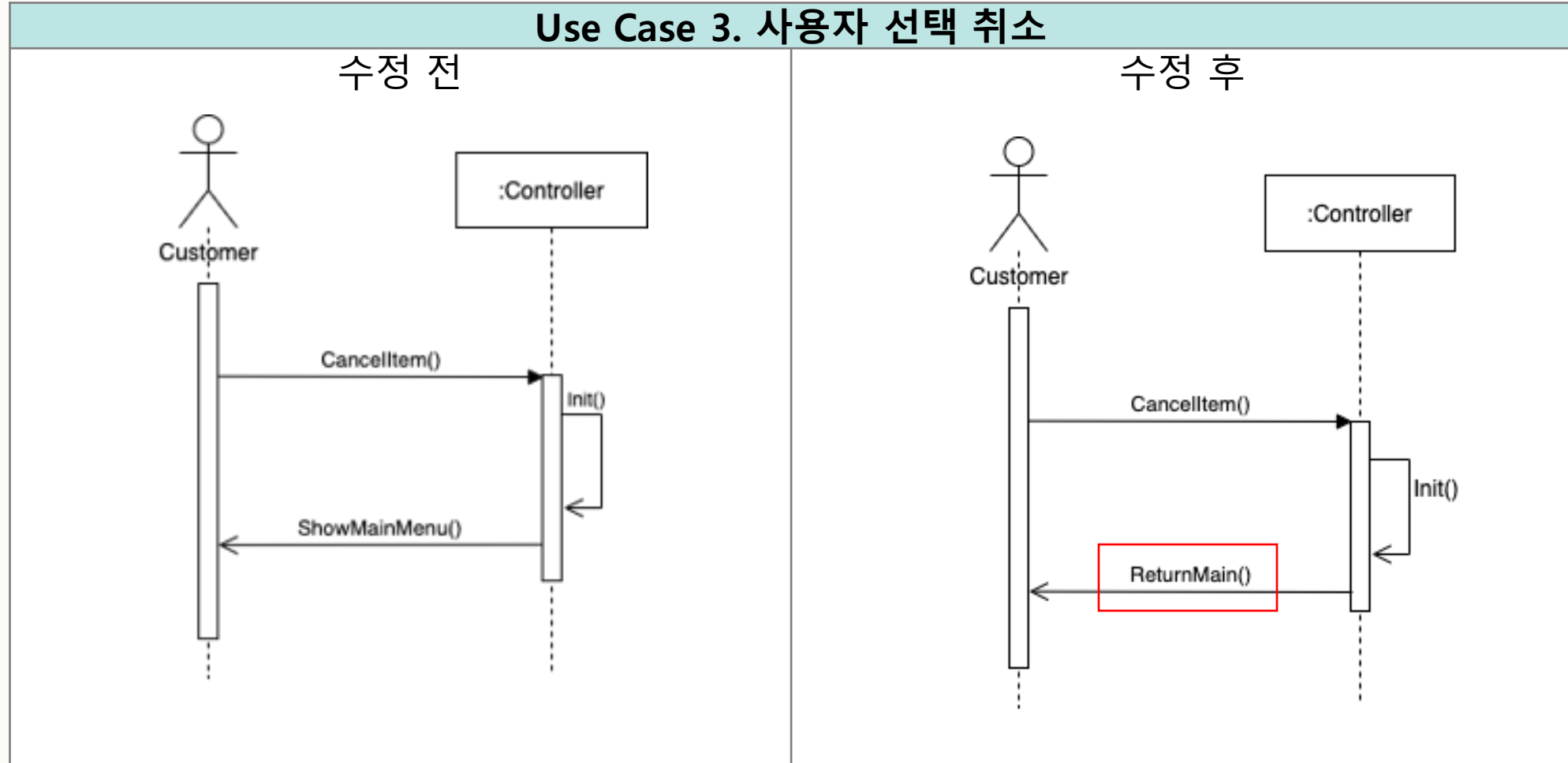


수정 후

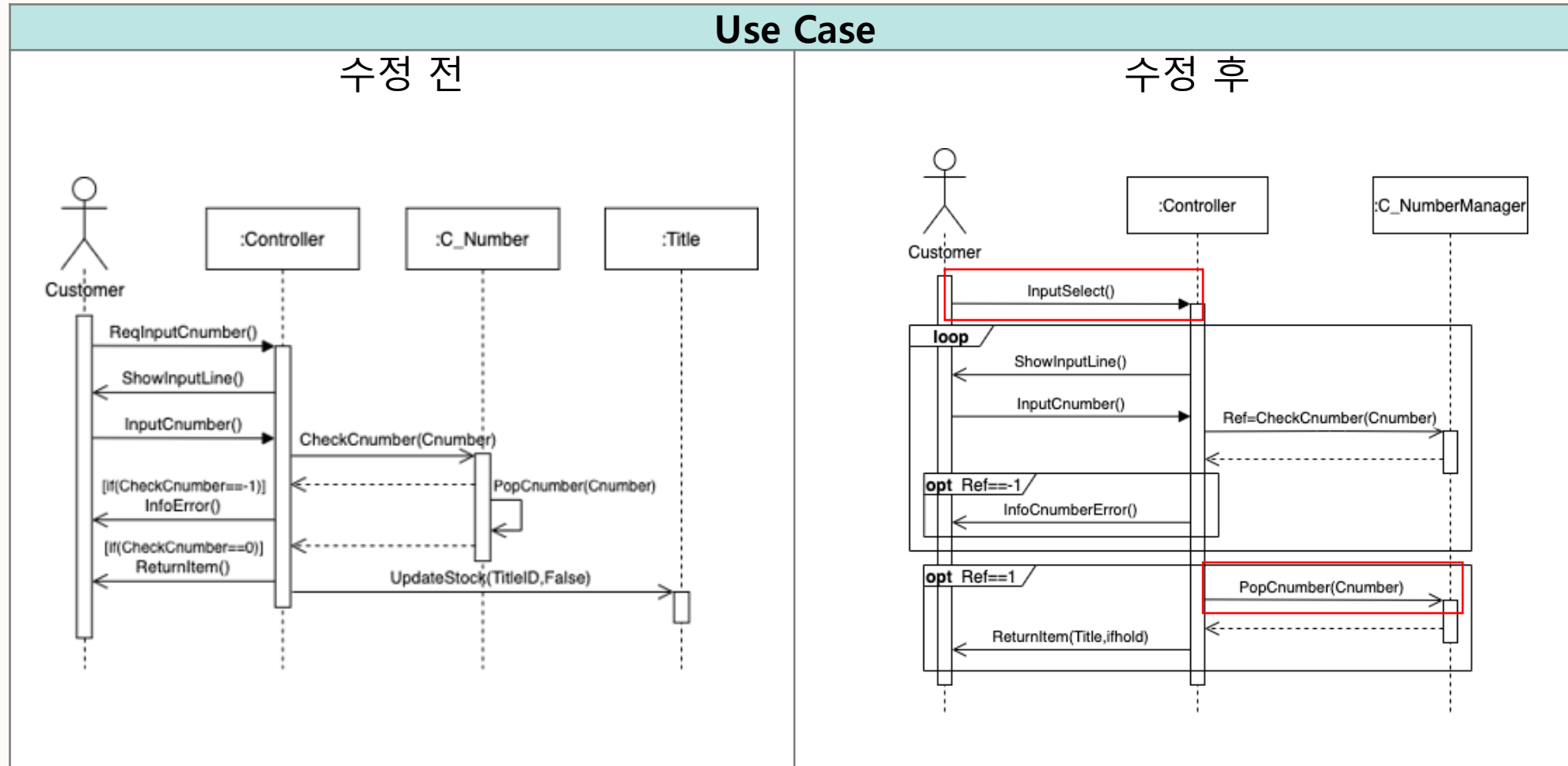


# Activity 2051. Implement Class & Methods Definitions

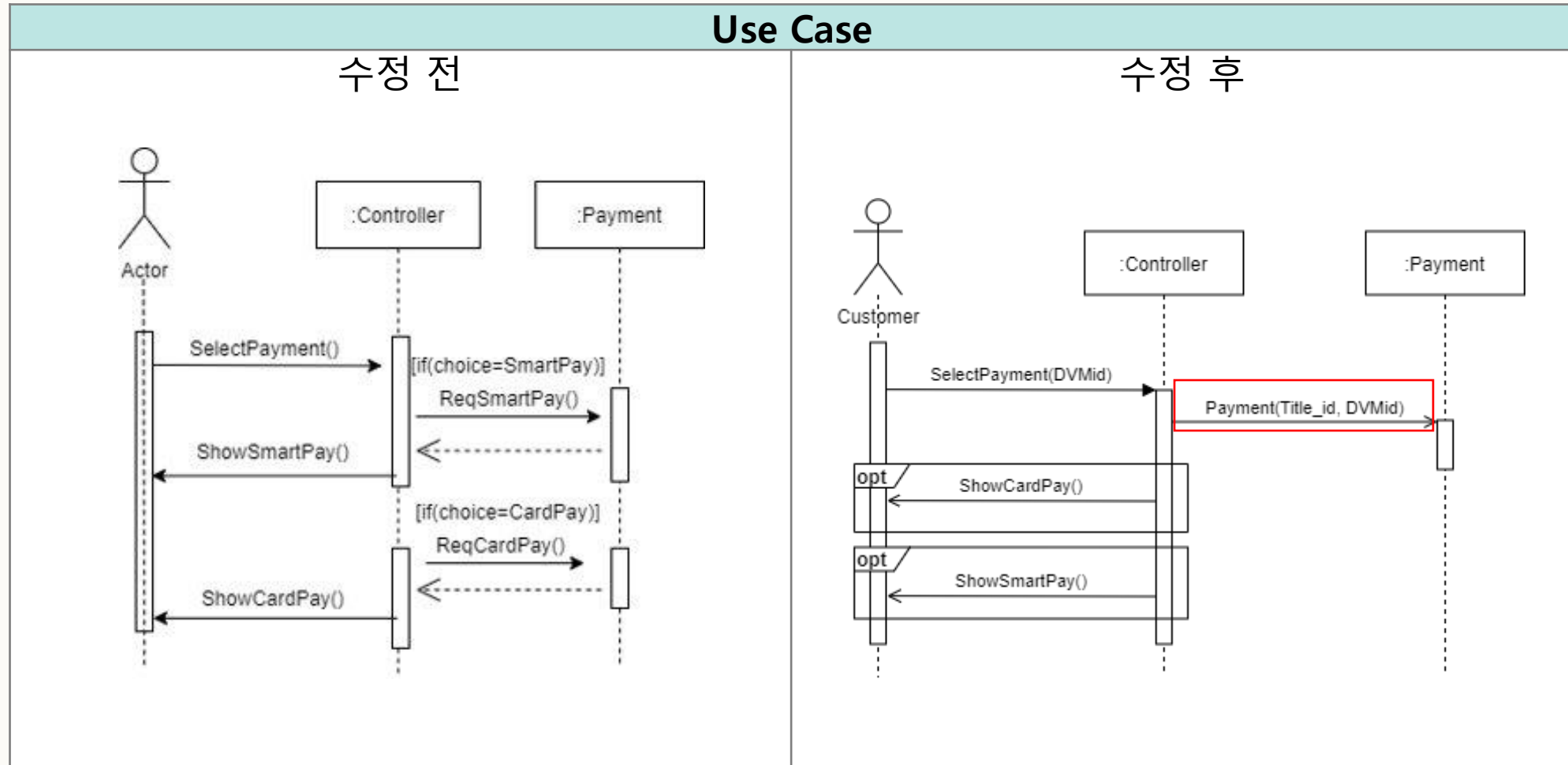
## Use Case 3. 사용자 선택 취소



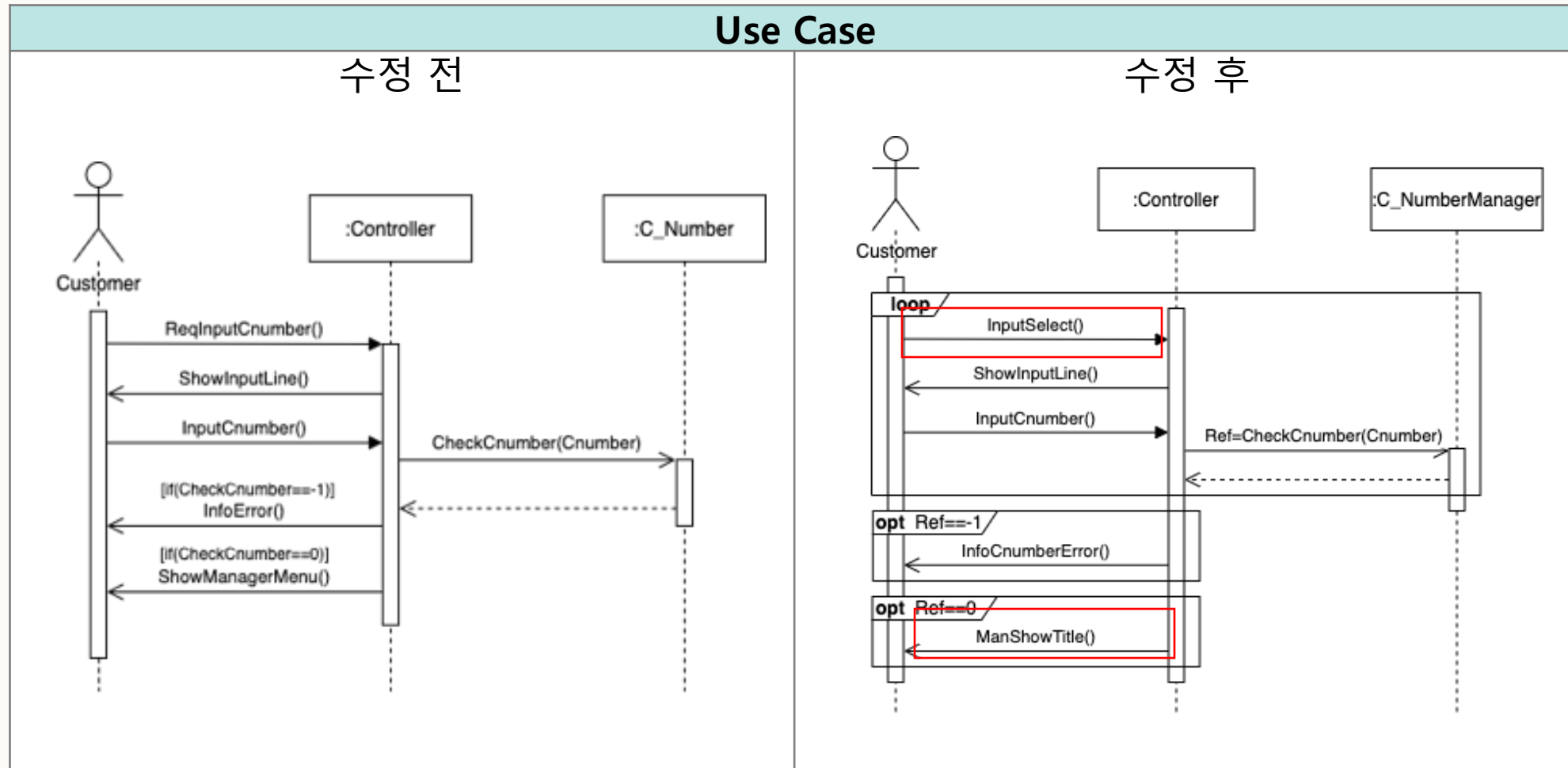
# Activity 2051. Implement Class & Methods Definitions



# Activity 2051. Implement Class & Methods Definitions



# Activity 2051. Implement Class & Methods Definitions

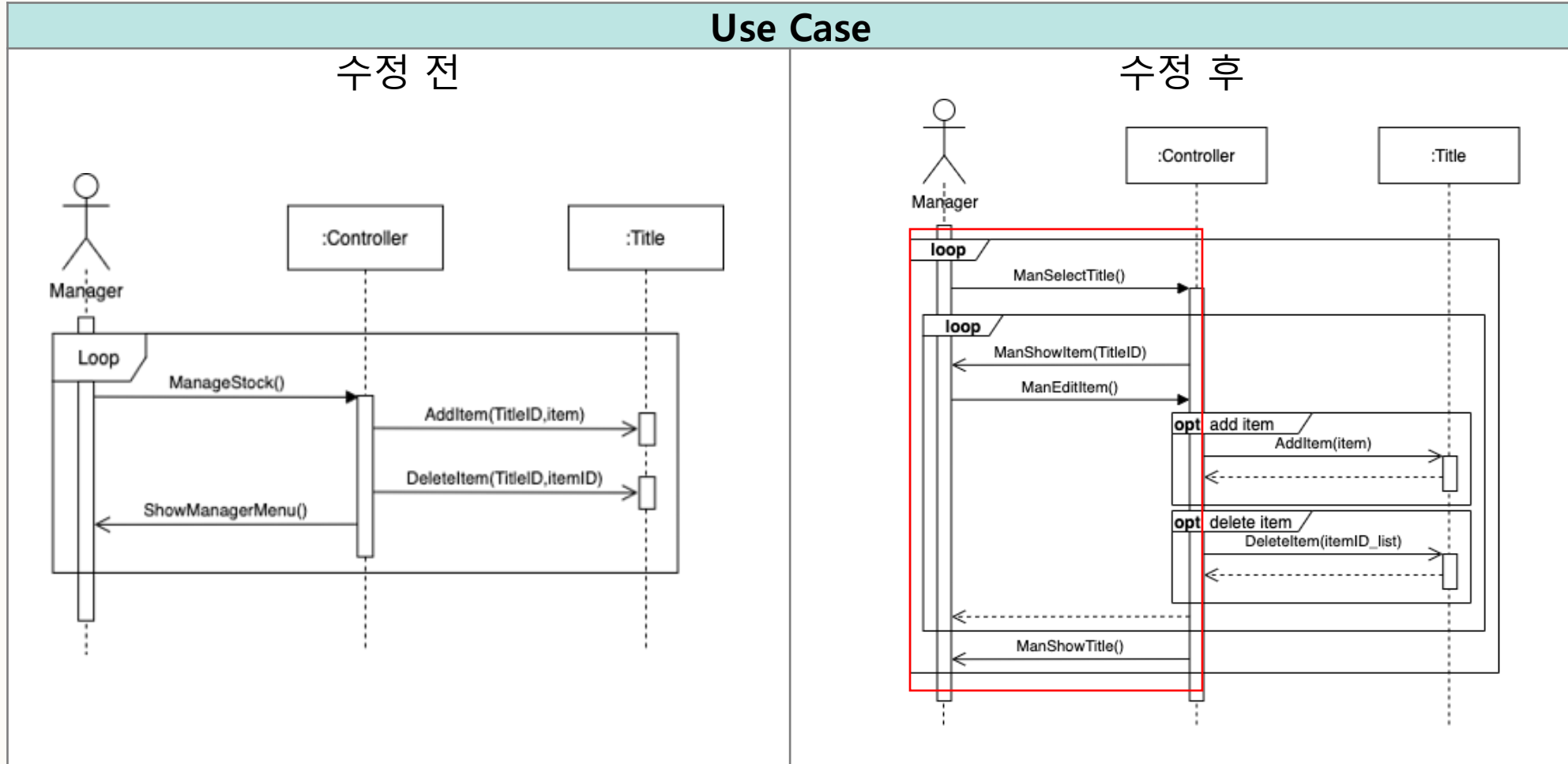


# Activity 2051. Implement Class & Methods Definitions

## Use Case

수정 전

수정 후





# Activity 2051. Implement Class & Methods Definitions

Name	Controller
Type	Class
Purpose	사용자에게 메뉴를 표시
Overview	N/A
Cross Reference	R1.1, R1.2.1, R1.2.2, R1.2.3, R1.3, R1.4, R1.5, R2.1, R2.4, R3.1.2 "자판기 음료 종류 출력", "구매할 음료 입력", "사용자 선택 취소", "사용자 인증번호/바코드 입력", "재고가 부족한 제품 안내", "구매가능한 자판기 안내", "사용자 자판기 선택", "결제 수단 목록 출력", "인증번호/바코드 출력"
Input	TitleID, DVMid
Output	Integer , void
Exceptional Courses of Events	N/A

# Activity 2051. Implement Class & Methods Definitions

Name	Title
Type	Class
Purpose	재고를 관리하고 그에 대한 정보를 저장하는 클래스
Overview	N/A
Cross Reference	R1.1, R1.2.1, R1.3, R1.4, R2.3.1, R2.3.2, R3.1.1, R3.2, R4.2, R4.4 "자판기 음료 종류 출력", "구매할 음료 입력", "재고가 부족한 제품 안내", "구매가능한 자판기 안내", "음료 전달", "재고 수량 업데이트", "인증번호/바코드 생성", "재고변경 및관리"
Input	Id, IfHold, item, ilst
Output	Name, Item_List, boolean
Exceptional Courses of Events	N/A

# Activity 2051. Implement Class & Methods Definitions

Name	Item
Type	Class
Purpose	재고 각각의 정보를 저장하는 클래스
Overview	N/A
Cross Reference	R1.1, R1.2.1, R1.3, R1.4, R3.1.1, R3.2, R4.2, R4.4 "자판기 음료 종류 출력", "구매할 음료 입력", "재고가 부족한 제품 안내", "구매가능한 자판기 안내", "음료 전달", "재고 수량 업데이트", "인증번호/바코드 생성", "재고변경 및관리"
Input	N/A
Output	Expiration_date
Exceptional Courses of Events	N/A

# Activity 2051. Implement Class & Methods Definitions

Name	Payment
Type	Class
Purpose	진행되는 결제정보를 컨트롤러에 전달하는 클래스
Overview	N/A
Cross Reference	R2.1, R2.3.1, R2.3.2, R2.4, R3.1.1 "결제 수단 목록 출력", "간편 결제", "카드 결제", "결제 취소", "인증번호, 바코드 출력"
Input	Title_id, DVMid, Pay_confirm
Output	Error_log, CN, integer
Exceptional Courses of Events	N/A

# Activity 2051. Implement Class & Methods Definitions

Name	C_NumberManager
Type	Class
Purpose	인증번호 구매 진행시 저장된 인증번호를 확인하는 클래스
Overview	N/A
Cross Reference	R1.2.3, R3.1.1, R3.1.2 "사용자 인증번호/바코드 입력", "인증번호/바코드 생성", 인증번호/바코드 출력"
Input	Title_id, id, C_Number_t
Output	R, integer, String
Exceptional Courses of Events	N/A

# Activity 2051. Implement Class & Methods Definitions

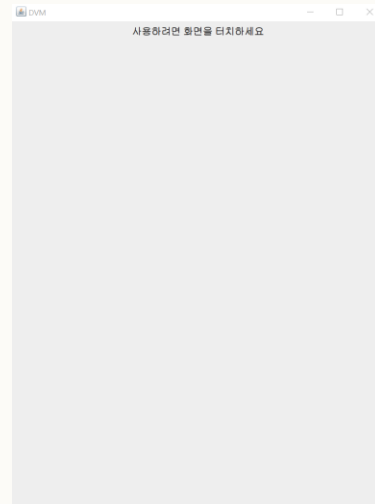
Name	C_Number
Type	Class
Purpose	인증번호를 생성하는 클래스
Overview	N/A
Cross Reference	R3.1.1, R3.1.2 "인증번호/바코드 생성", 인증번호/바코드 출력"
Input	Title_id, DvmID
Output	C_Number_t, numStr
Exceptional Courses of Events	N/A

# Activity 2051. Implement Class & Methods Definitions

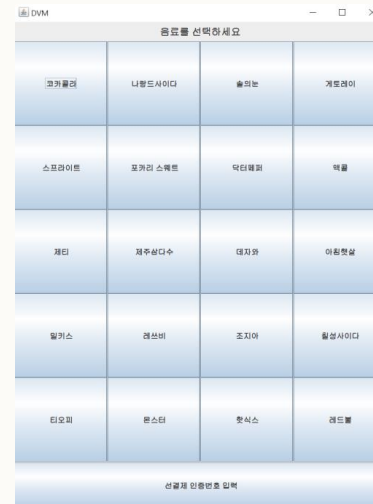
Name	Message_Queue
Type	Class
Purpose	메세지를 수신/발신하고 큐에 저장하는 클래스
Overview	N/A
Cross Reference	R1.4, R3.1.1, R4.3, R4.4, R4.5.1, R4.5.2 "구매가능한 자판기 안내", "인증번호/바코드 생성", "위치 정보 확인", "재고 정보 확인", "메시지 발신", "메시지 수신"
Input	Message, type
Output	temp1
Exceptional Courses of Events	N/A

# Activity 2052. Implement Windows (사용자 매뉴얼)

첫 실행화면

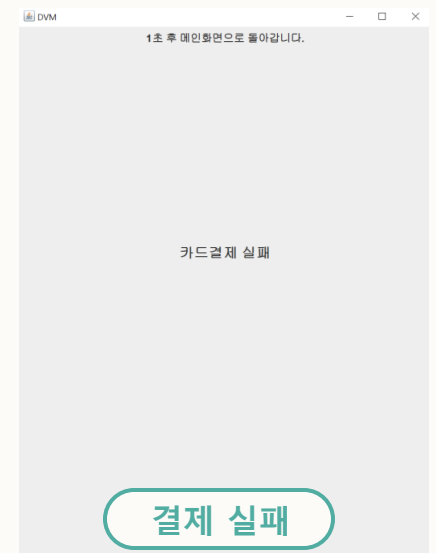


Touch!



재고가 있는 제품 선택

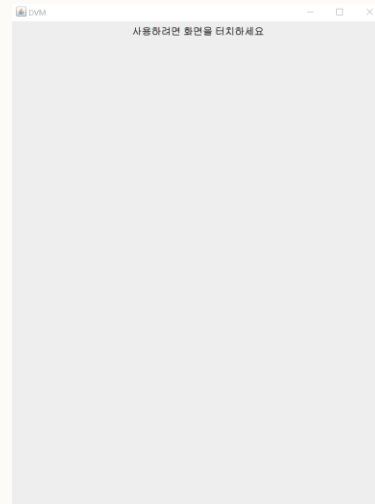
카드결제



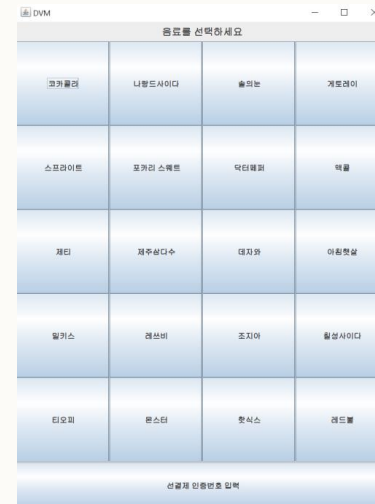


# Activity 2052. Implement Windows (사용자 매뉴얼)

첫 실행화면



Touch!



재고가 있는 제품 선택

간편결제



Touch!

간편결제

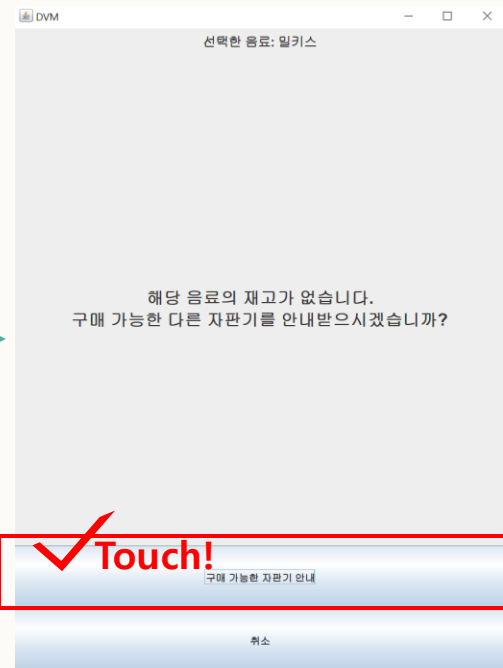
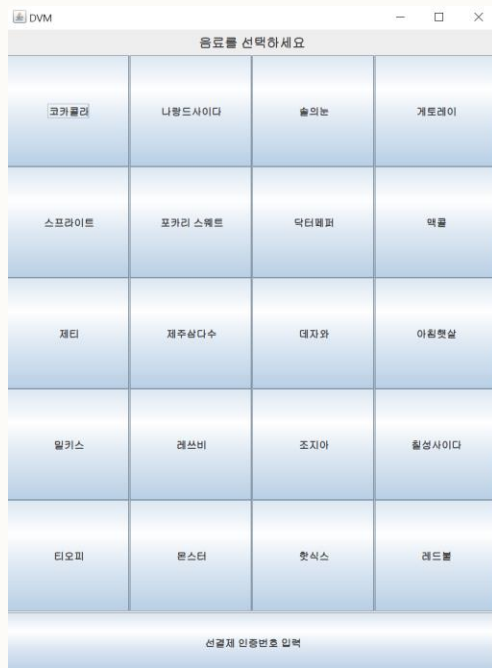
취소

결제 성공

결제 실패

# Activity 2052. Implement Windows (사용자 매뉴얼)

## 재고가 존재하지 않는 제품 선택



# Activity 2052. Implement Windows (사용자 매뉴얼)

## 재고가 존재하지 않는 제품 선택



선결제 성공



선결제 실패

# Activity 2052. Implement Windows (사용자 매뉴얼)

선결제 된 인증번호 입력

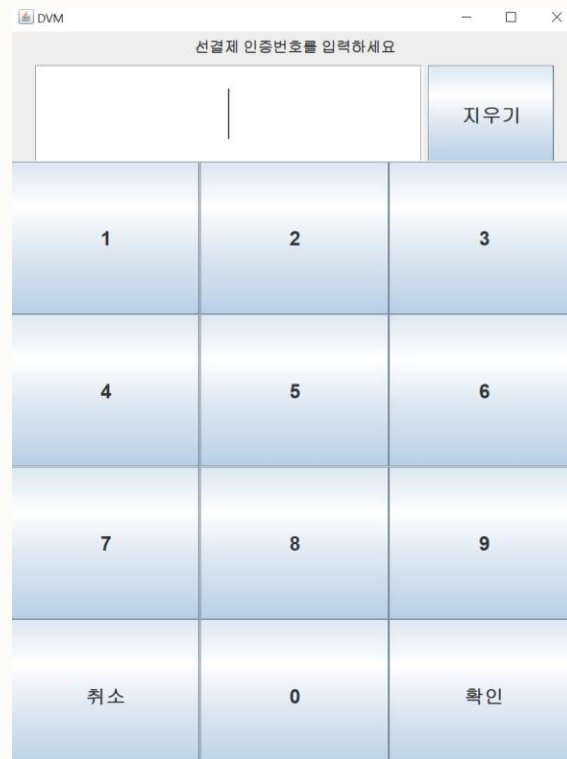
The screenshot shows a window titled "선결제 인증번호를 입력하세요" (Enter pre-payment authentication code). The window contains a text input field with the value "117345" and a "지우기" (Erase) button. Below the input field is a numeric keypad with buttons for digits 1-9, 0, and "취소" (Cancel). The "확인" (Confirm) button is located at the bottom right of the keypad.



The screenshot shows a window titled "2초 후 메인화면으로 돌아갑니다." (Returns to the main screen in 2 seconds). The window contains a confirmation message: "말키스 음료가 나왔습니다." (Malke's drink is ready).

# Activity 2052. Implement Windows (사용자 매뉴얼)

존재하지 않는 인증번호 입력



The screenshot shows a window titled 'DVM' with the subtitle '선결제 인증번호를 입력하세요'. It features a text input field with a vertical cursor and a '지우기' (Erase) button. Below the input field is a 4x3 grid of buttons containing the digits 1 through 9, '취소' (Cancel), and '확인' (Confirm).



The screenshot shows a window titled 'DVM' with the subtitle '2초 후 이전화면으로 돌아갑니다.' (Returns to the previous screen in 2 seconds). The main text reads '해당 인증번호에 대한 선결제 정보를 확인할 수 없습니다. 다시 입력하세요.' (Cannot check pre-payment information for the corresponding authentication number. Please re-enter it.) and there is a '확인' (Confirm) button at the bottom.

# Activity 2052. Implement Windows (사용자 매뉴얼)

관리자 모드

관리자 고유번호 입력

DVM 선결제 인증번호를 입력하세요

1	2	3
4	5	6
7	8	9
취소	0	확인

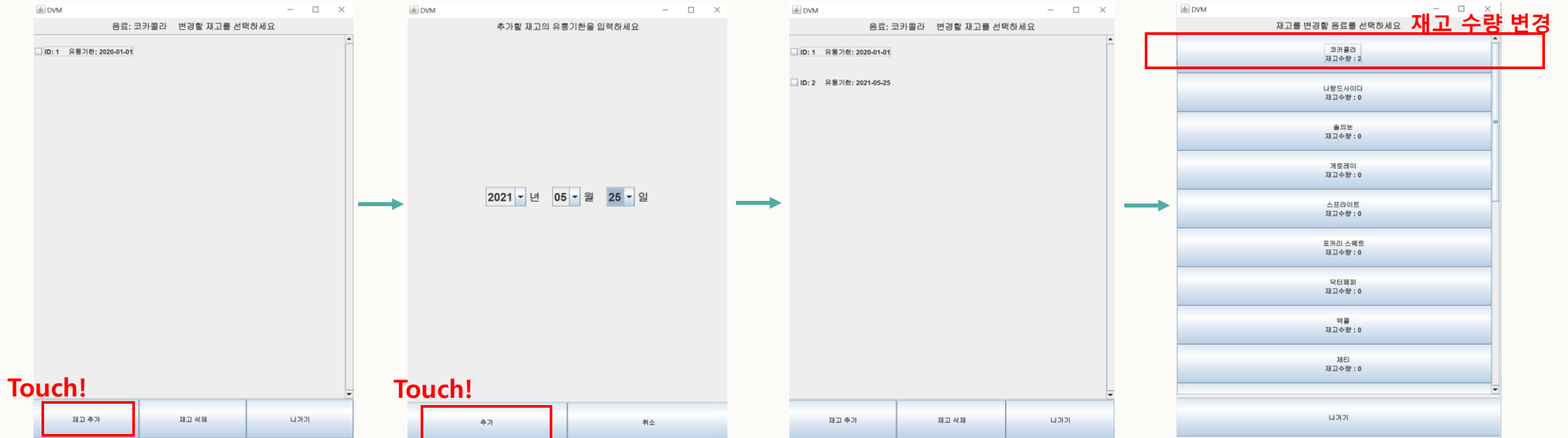


DVM 재고를 변경할 음료를 선택하세요

코카콜라 재고수량 : 1
나랑드사이다 재고수량 : 0
솔의눈 재고수량 : 0
게토레이 재고수량 : 0
스프라이트 재고수량 : 0
포카리 스웨트 재고수량 : 0
덕터비피 재고수량 : 0
맥콜 재고수량 : 0
제티 재고수량 : 0
나가기

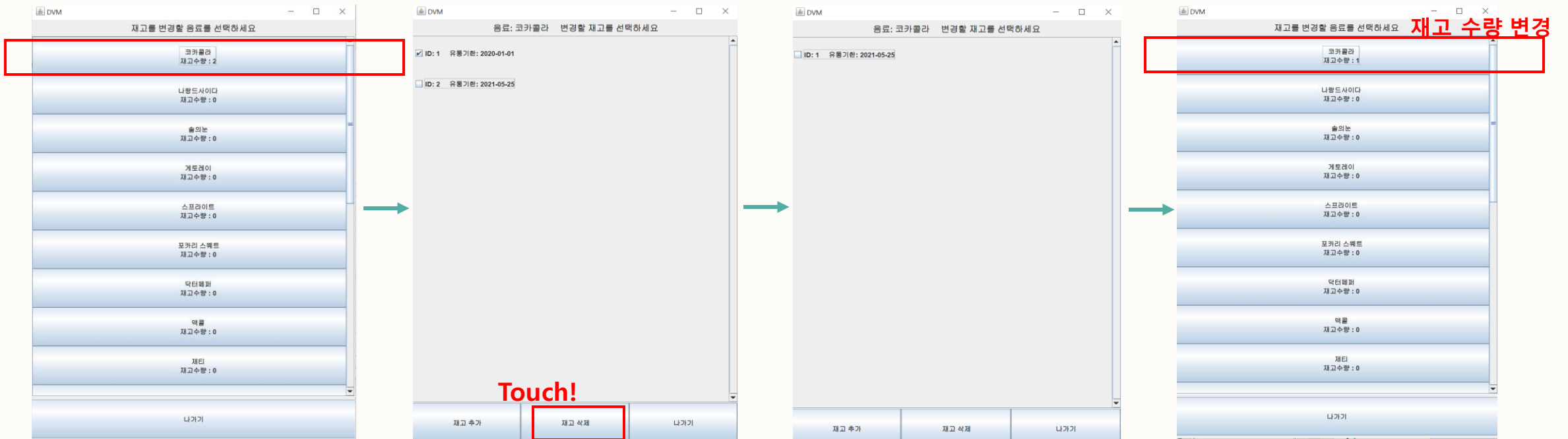
# Activity 2052. Implement Windows (사용자 매뉴얼)

## 재고 추가



# Activity 2052. Implement Windows (사용자 매뉴얼)

## 재고 삭제





# Activity 2055. Write Unit Test Code

Class:Title

```
1 package Logic;
2
3 import org.junit.Assert;
4 import org.junit.Test;
5
6 import java.util.ArrayList;
7
8 import static org.junit.Assert.*;
9
10 public class TitleTest {
11     private final Title t = new Title( Name: "코카콜라", Price: 700);
12
13     @Test
14     public void testAddItem() {
15         t.AddItem(new Item( Expiration_Date: 20201125));
16         t.AddItem(new Item( Expiration_Date: 20201125));
17         t.AddItem(new Item( Expiration_Date: 20201125));
18         t.AddItem(new Item( Expiration_Date: 20201125));
19         t.AddItem(new Item( Expiration_Date: 20201125));
20         int ExpectedResult = 5;
21         int ActualResult = t.getItem_List().size();
22         Assert.assertEquals(ExpectedResult, ActualResult);
23     }
```

```
25     @Test
26     public void testDeleteItem() {
27         t.AddItem(new Item( Expiration_Date: 20201125));
28         t.AddItem(new Item( Expiration_Date: 20201125));
29         t.AddItem(new Item( Expiration_Date: 20201125));
30         t.AddItem(new Item( Expiration_Date: 20201125));
31         t.AddItem(new Item( Expiration_Date: 20201125));
32         ArrayList<Integer> delitemlist = new ArrayList<>();
33         delitemlist.add(0);
34         delitemlist.add(1);
35         delitemlist.add(2);
36         t.DeleteItem(delitemlist);
37         int ExpectedResult = 2;
38         int ActualResult = t.getItem_List().size();
39         Assert.assertEquals(ExpectedResult, ActualResult);
40     }
41
42     @Test
43     public void testItem_List() {
44         t.AddItem(new Item( Expiration_Date: 20201025));
45         t.AddItem(new Item( Expiration_Date: 20201125));
46         t.AddItem(new Item( Expiration_Date: 20201125));
47         t.AddItem(new Item( Expiration_Date: 20201125));
48         t.AddItem(new Item( Expiration_Date: 20201125));
49         int ExpectedResult = 20201025;
50         int ActualResult = t.getItem_List().get(0).getExpiration_Date();
51         Assert.assertEquals(ExpectedResult, ActualResult);
52     }
```

# Activity 2055. Write Unit Test Code

Class:Title

```
54     @Test
55     public void testUpdateStock() {
56         t.AddItem(new Item( Expiration_Date: 20201025));
57         t.AddItem(new Item( Expiration_Date: 20201125));
58         t.AddItem(new Item( Expiration_Date: 20201125));
59         t.AddItem(new Item( Expiration_Date: 20201125));
60         t.AddItem(new Item( Expiration_Date: 20201125));
61         t.UpdateStock( id: 1,  IfHold: true);
62         int ExpectedResult = 1;
63         int ActualResult = t.getHold();
64         Assert.assertEquals(ExpectedResult, ActualResult);
65         ExpectedResult = 4;
66         ActualResult = t.getItem_List().size();
67         Assert.assertEquals(ExpectedResult, ActualResult);
68         t.UpdateStock( id: 1,  IfHold: false);
69         ExpectedResult = 1;
70         ActualResult = t.getHold();
71         Assert.assertEquals(ExpectedResult, ActualResult);
72         ExpectedResult = 4;
73         ActualResult = t.getItem_List().size();
74         Assert.assertEquals(ExpectedResult, ActualResult);
75         t.UpdateStock( id: 0,  IfHold: true);
76         ExpectedResult = 0;
77         ActualResult = t.getHold();
78         Assert.assertEquals(ExpectedResult, ActualResult);
79         ExpectedResult = 4;
80         ActualResult = t.getItem_List().size();
81         Assert.assertEquals(ExpectedResult, ActualResult);
82         t.UpdateStock( id: 0,  IfHold: false);
83         ExpectedResult = 0;
84         ActualResult = t.getHold();
85         Assert.assertEquals(ExpectedResult, ActualResult);
86         ExpectedResult = 3;
```

```
83         ExpectedResult = 0;
84         ActualResult = t.getHold();
85         Assert.assertEquals(ExpectedResult, ActualResult);
86         ExpectedResult = 3;
87         ActualResult = t.getItem_List().size();
88         Assert.assertEquals(ExpectedResult, ActualResult);
89     }
90
91     @Test
92     public void testCheckStock() {
93         t.AddItem(new Item( Expiration_Date: 20201025));
94         assertTrue(t.CheckStock());
95         t.UpdateStock( id: 0,  IfHold: false);
96         assertFalse(t.CheckStock());
97     }
98 }
```

# Activity 2055. Write Unit Test Code

---

Class:Item

```
1 package Logic;
2
3 import org.junit.Assert;
4 import org.junit.Test;
5
6 import static org.junit.Assert.*;
7
8 public class ItemTest {
9
10     Item item = new Item( Expiration_Date: 20201125);
11     ;
12
13     @Test
14     public void testGetExpiration_Date() {
15         item.setExpiration_Date(20211026);
16         int ExpectedResult = 20211026;
17         int ActualResult = item.getExpiration_Date();
18         Assert.assertEquals(ExpectedResult, ActualResult);
19     }
20 }
```

# Activity 2055. Write Unit Test Code

## Class:Payment

```
1 package Logic;
2
3 import org.junit.Assert;
4 import org.junit.Test;
5
6 import static org.junit.Assert.*;
7
8 public class PaymentTest {
9
10     Payment payment = new Payment( title_id: 1, DVMid: 1);
11     ;
12
13     @Test
14     public void testCardPay() {
15         int ActualResult = payment.CardPay( Pay_confirm: true);
16         Assert.assertTrue( condition: ActualResult > 0);
17         int ExpectedResult = -3;
18         ActualResult = payment.CardPay( Pay_confirm: false);
19         Assert.assertEquals(ExpectedResult, ActualResult);
20         payment = new Payment( title_id: 1, DVMid: 0);
21         ExpectedResult = 0;
22         ActualResult = payment.CardPay( Pay_confirm: true);
23         Assert.assertEquals(ExpectedResult, ActualResult);
24         ExpectedResult = -3;
25         ActualResult = payment.CardPay( Pay_confirm: false);
26         Assert.assertEquals(ExpectedResult, ActualResult);
27     }
28 }
```

```
29
30 @Test
31 public void testSmartPay() {
32     int ActualResult = payment.SmartPay( Pay_confirm: true)
33     Assert.assertTrue( condition: ActualResult > 0);
34     int ExpectedResult = -3;
35     ActualResult = payment.SmartPay( Pay_confirm: false);
36     Assert.assertEquals(ExpectedResult, ActualResult);
37     payment = new Payment( title_id: 1, DVMid: 0);
38     ExpectedResult = 0;
39     ActualResult = payment.SmartPay( Pay_confirm: true);
40     Assert.assertEquals(ExpectedResult, ActualResult);
41     ExpectedResult = -3;
42     ActualResult = payment.SmartPay( Pay_confirm: false);
43     Assert.assertEquals(ExpectedResult, ActualResult);
44 }
45
46 @Test
47 public void testInit() {
48     payment = new Payment( title_id: 1, DVMid: 1);
49     payment.init();
50     int ExpectedResult = -1;
51     int ActualResult = payment.getTitle_id();
52     Assert.assertEquals(ActualResult, ExpectedResult);
53     ExpectedResult = -1;
54     ActualResult = payment.getDVMid();
55     Assert.assertEquals(ExpectedResult, ActualResult);
56     assertNull(payment.getError_log());
57 }
```

# Activity 2055. Write Unit Test Code

---

Class:C\_Number

```
1 package Logic;
2
3 import org.junit.Assert;
4 import org.junit.Test;
5
6 import static org.junit.Assert.*;
7
8 public class C_NumberTest {
9
10     private C_Number CN = new C_Number( title_id: 1, id: 1);
11
12     @Test
13     public void testCreateCNumber() {
14         int ExpectedResult = CN.CreateCNumber( title_id: 1, DvmID: 1);
15         int ActualResult = CN.getC_Number_t();
16         Assert.assertEquals(ExpectedResult, ActualResult);
17     }
18 }
```

# Activity 2055. Write Unit Test Code

Class: C\_NumberManager

```
1 package Logic;
2
3 import org.junit.Assert;
4 import org.junit.Test;
5
6 import static org.junit.Assert.*;
7
8 public class C_NumberManagerTest {
9     private C_NumberManager CM= new C_NumberManager();
10    private C_Number Cn=new C_Number( title_id: 1, id: 1);
11
12    @Test
13    public void testPopCnumber() {
14        Cn.setC_Number_t(971125);
15        CM.AddCnumber(Cn);
16        int ExpectedResult=Cn.getTitle_id();
17        int ActualResult=CM.PopCnumber( C_Number_t: 971125);
18        Assert.assertEquals(ExpectedResult,ActualResult);
19    }
```

```
21    @Test
22    public void testCheckCnumber() {
23        Cn.setC_Number_t(999999);
24        CM.setM_Number(971125);
25        CM.AddCnumber(Cn);
26        int ExpectedResult=1;
27        int ActualResult=CM.CheckCnumber( C_Number_t: 999999);
28        Assert.assertEquals(ExpectedResult,ActualResult);
29        ExpectedResult=0;
30        ActualResult=CM.CheckCnumber( C_Number_t: 971125);
31        Assert.assertEquals(ExpectedResult,ActualResult);
32        ExpectedResult=-1;
33        ActualResult=CM.CheckCnumber( C_Number_t: 888888);
34        Assert.assertEquals(ExpectedResult,ActualResult);
35    }
36
37    @Test
38    public void testAddCnumber() {
39        Cn.setC_Number_t(971125);
40        CM.AddCnumber(Cn);
41        int ExpectedResult = 1;
42        int ActualResult = CM.getC_List().size();
43        Assert.assertEquals(ExpectedResult,ActualResult);
44    }
45 }
```

# Activity 2055. Write Unit Test Code

Class: DVM

```
1 package Logic;
2
3 import org.junit.Assert;
4 import org.junit.Test;
5
6 import static org.junit.Assert.*;
7
8 public class DVMTest {
9     private DVM dvm = new DVM( ID: 1, X: 1.0, Y: 1.0);
10
11     @Test
12     public void testGetID() {
13         dvm.setID(7);
14         int ExpectedResult = 7;
15         int ActualResult = dvm.getID();
16         Assert.assertEquals(ExpectedResult, ActualResult);
17     }
18
19     @Test
20     public void testGetAddress_X() {
21         dvm.setAddress_X(1.23456);
22         Double ExpectedResult = 1.23456;
23         Double ActualResult = dvm.getAddress_X();
24         Assert.assertEquals(ExpectedResult, ActualResult);
25     }
26
27     @Test
28     public void testGetAddress_Y() {
29         dvm.setAddress_Y(2.45678);
30         Double ExpectedResult = 2.45678;
31         Double ActualResult = dvm.getAddress_Y();
32         Assert.assertEquals(ExpectedResult, ActualResult);
33     }
34 }
```

```
35     public void testGetCurrentID() {
36         dvm.CurrentID=2;
37         Assert.assertEquals( expected: 2, dvm.CurrentID);
38     }
39 }
```

# Activity 2055. Write Unit Test Code

Class:Message

```
1 package Logic;
2
3 import org.junit.Assert;
4 import org.junit.Test;
5
6 import static org.junit.Assert.*;
7
8 public class MessageTest {
9     Message msg = new Message( myid: 1);
10
11     @Test
12     public void testSetmsg1() {
13         msg.setmsg( id: 1, type: 1, data: 1);
14         Assert.assertEquals( expected: 1, msg.getTargetID());
15         Assert.assertEquals( expected: 1, msg.getType());
16         Assert.assertEquals( expected: 1, msg.getTitle());
17     }
18
19     @Test
20     public void testSetmsg2() {
21         msg.setmsg( id: 1, type: 2, data: true);
22         Assert.assertEquals( expected: 1, msg.getTargetID());
23         Assert.assertEquals( expected: 2, msg.getType());
24         Assert.assertTrue(msg.isBoolData());
25     }
26
27     @Test
28     public void testSetmsg3() {
29         msg.setmsg( id: 1, type: 3);
30         Assert.assertEquals( expected: 1, msg.getTargetID());
31         Assert.assertEquals( expected: 3, msg.getType());
32     }
```

```
34     @Test
35     public void testSetmsg4() {
36         msg.setmsg( id: 1, type: 4, data1: 1.0, data2: 1.0);
37         Assert.assertEquals( expected: 1, msg.getTargetID());
38         Assert.assertEquals( expected: 4, msg.getType());
39         Assert.assertEquals(Double.toString( d: 1.0), Double.toString(msg.getxAddress()));
40         Assert.assertEquals(Double.toString( d: 1.0), Double.toString(msg.getyAddress()));
41     }
42
43     @Test
44     public void testSetmsg5() {
45         msg.setmsg( id: 1, type: 5, data1: 1, data2: 971125);
46         Assert.assertEquals( expected: 1, msg.getTargetID());
47         Assert.assertEquals( expected: 5, msg.getType());
48         Assert.assertEquals( expected: 1, msg.getTitle());
49         Assert.assertEquals( expected: 971125, msg.getC_Number());
50     }
51 }
52 }
```



# Activity 2055. Write Unit Test Code

Class:Message\_Queue

```
1 package Logic;
2
3 import org.junit.Assert;
4 import org.junit.Test;
5
6 import java.net.InetAddress;
7
8 public class Message_QueueTest {
9     private Message msg = new Message( myid: 1);
10    private Message_Queue queue = new Message_Queue();
11
12    @Test
13    public void testMsgSend() {
14        msg.setTargetID(1);
15        msg.setType(2);
16        msg.setBoolData(true);
17        Message_Queue.MsgSend(msg);
18        msg.setType(3);
19        Message_Queue.MsgSend(msg);
20    }
21
22    @Test
23    public void testDequeue() {
24        Controller c = new Controller();
25        c.getTitle_List().get(0).AddItem(new Item( Expiration_Date: 20201125));
26        msg.setTargetID(1);
27        msg.setType(5);
28        msg.setTitle(1);
29        msg.setC_Number(971125);
30        Message_Queue.msgQueue.offer(msg);
31        Message_Queue.Dequeue();
32    }
33 }
```

# Activity 2055. Write Unit Test Code

Class:Controller

```
1 package Logic;
2
3 import GUI.*;
4 import org.junit.Assert;
5 import org.junit.Before;
6 import org.junit.Ignore;
7 import org.junit.Test;
8
9 import javax.swing.*;
10
11 import static org.junit.Assert.*;
12
13 public class ControllerTest {
14     private Controller c = new Controller();
15
16     @Ignore
17     @Before
18     public void initController() {
19         c = new Controller();
20         c.setK(new JFrame());
21     }
22
23     @Ignore
24     @Test
25     public void inputSelect() {
26         int del;
27         c.setK(new MainMenu(c.getTitle_List()));
28         del = c.InputSelect();
29         Assert.assertEquals( expected: -2, del);
30         c.getK().setVisible(false);
31     }
```

```
32
33     @Ignore
34     @Test
35     public void showInputLine() {
36         int del;
37         c.setK(new MainMenu(c.Title_List));
38         del = c.InputSelect();
39         Assert.assertEquals( expected: 0, del);
40         c.getK().setVisible(false);
41     }
42
43     @Ignore
44     @Test
45     public void inputCnumber() {
46         int del;
47         c.setK(new InputLine());
48         del = c.InputCnumber();
49         Assert.assertEquals( expected: 0, del);
50         c.getK().setVisible(false);
51     }
52
53     @Ignore
54     @Test
55     public void infoCnumberError() {
56         int del;
57         c.setK(new InputLine());
58         del = c.InputCnumber();
59         Assert.assertEquals( expected: 1, del);
60         c.getK().setVisible(false);
61     }
```

# Activity 2055. Write Unit Test Code

## Class:Controller

```
65 ▶ public void manShowTitle() {
66     c.ManShowTitle();
67     Assert.assertEquals( expected: 0, c.getDel());
68     c.getK().setVisible(false);
69 }
70
71 @Ignore
72 @Test
73 ▶ public void manSelectTitle() {
74     c.ManShowTitle();
75     Assert.assertEquals( expected: 0, c.getDel());
76     c.getK().setVisible(false);
77 }
78
79 @Ignore
80 @Test
81 ▶ public void manShowItem() {
82     c.Title_List.get(0).AddItem(new Item( Expiration_Date: 20250101));
83     c.ManShowItem( TitleID: 1);
84     Assert.assertEquals( expected: 0, c.getDel());
85     c.getK().setVisible(false);
86 }
87
88 @Ignore
89 @Test
90 ▶ public void manEditItem() {
91     c.Title_List.get(0).AddItem(new Item( Expiration_Date: 20250101));
92     c.ManShowItem( TitleID: 1);
93     Assert.assertEquals( expected: 2, c.getTitle_List().get(0).getItem_List().size());
94     c.ManShowItem( TitleID: 1);
95     Assert.assertEquals( expected: 0, c.getTitle_List().get(0).getItem_List().size());
96     c.getK().setVisible(false);
97 }
```

```
99 @Ignore
100 @Test
101 ▶ public void printSelectPay() {
102     c.setK(new MainMenu(c.Title_List));
103     c.Title_List.get(0).AddItem(new Item( Expiration_Date: 20201125));
104     c.InputSelect();
105     Assert.assertEquals( expected: 1, c.Title_List.get(0).getItem_List().size());
106     c.getK().setVisible(false);
107 }
108
109 @Ignore
110 @Test
111 ▶ public void selectPayment() {
112     c.setK(new PaymentMenu(new Title( Name: "test", Price: 700));
113     c.setBasket(1);
114     c.SelectPayment( DVMid: 0);
115     Assert.assertEquals( expected: 0, c.getDel());
116     c.getK().setVisible(false);
117 }
118
119 @Ignore
120 @Test
121 ▶ public void cancelItem() {
122     c.setBasket(1);
123     c.setPayment(new Payment( title_id: 1, DVMid: 0));
124     c.CancelItem();
125     Assert.assertEquals( expected: -666, c.getBasket());
126     Assert.assertTrue(c.DVMStack.isEmpty());
127     Assert.assertNull(c.getPayment());
128     c.getK().setVisible(false);
129 }
```

# Activity 2055. Write Unit Test Code

## Class:Controller

```
131 | @Ignore
132 | @Test
133 | public void showCardPay() {
134 |     c.setPayment(new Payment( title_id: 1, DVMMid: 0));
135 |     c.ShowCardPay();
136 |     c.getK().setVisible(false);
137 | }
138 |
139 | @Ignore
140 | @Test
141 | public void cardPay() {
142 |     c.setK(new CardPayUI());
143 |     c.Title_List.get(0).AddItem(new Item( Expiration_Date: 20201125));
144 |     c.setBasket(1);
145 |     c.setPayment(new Payment( title_id: 1, DVMMid: 0));
146 |     c.CardPay();
147 |     Assert.assertEquals( expected: 0, c.getTitle_List().get(0).getItem_List().size());
148 |     c.getK().setVisible(false);
149 | }
150 |
151 | @Ignore
152 | @Test
153 | public void showSmartPay() {
154 |     c.setPayment(new Payment( title_id: 1, DVMMid: 0));
155 |     c.ShowSmartPay();
156 |     c.getK().setVisible(false);
157 | }
158 |
```

```
159 | @Ignore
160 | @Test
161 | public void smartPay() {
162 |     c.setK(new SmartPayUI());
163 |     c.Title_List.get(0).AddItem(new Item( Expiration_Date: 20201125));
164 |     c.setBasket(1);
165 |     c.setPayment(new Payment( title_id: 1, DVMMid: 0));
166 |     c.SmartPay();
167 |     Assert.assertEquals( expected: 0, c.Title_List.get(0).getItem_List().size());
168 |     c.getK().setVisible(false);
169 | }
170 |
171 | @Ignore
172 | @Test
173 | public void returnItem() {
174 |     c.Title_List.get(0).AddItem(new Item( Expiration_Date: 20201125));
175 |     c.setBasket(1);
176 |     c.setPayment(new Payment( title_id: 1, DVMMid: 0));
177 |     c.ReturnItem(c.Title_List.get(0), IfHold: false);
178 |     Assert.assertEquals( expected: 0, c.Title_List.get(0).getItem_List().size());
179 |     c.getK().setVisible(false);
180 | }
181 |
182 | @Ignore
183 | @Test
184 | public void printCnumber() {
185 |     c.setBasket(1);
186 |     c.setPayment(new Payment( title_id: 1, DVMMid: 1));
187 |     c.PrintCnumber(c.Title_List.get(0), DVMMid: 1, Cnumber: 66666);
188 |     c.getK().setVisible(false);
189 | }
```

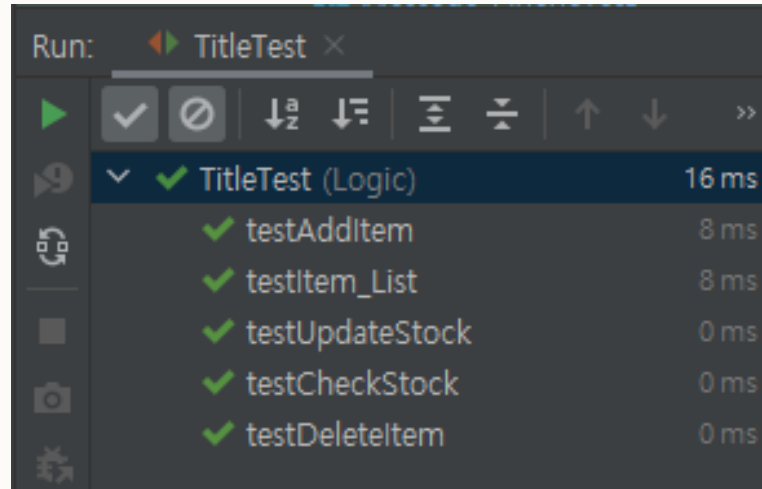
# Activity 2055. Write Unit Test Code

Class:Controller

```
191     @Ignore
192     @Test
193     public void infoNoItem() {
194         c.setBasket(1);
195         c.infoNoItem();
196         c.getK().setVisible(false);
197     }
198
199     @Ignore
200     @Test
201     public void reqFindDVM() {
202         c.setBasket(1);
203         c.infoNoItem();
204         c.getK().setVisible(false);
205     }
206
207     @Test
208     public void testInit() {
209         c.setBasket(10);
210         c.setPayment(new Payment(c.getBasket(), DVMid: 1));
211         c.init();
212         int ExpectedResult = -666;
213         int ActualResult = c.getBasket();
214         Assert.assertEquals(ExpectedResult, ActualResult);
215         Assert.assertNull(c.getPayment());
216     }
217 }
```

# Activity 2061. Unit Testing

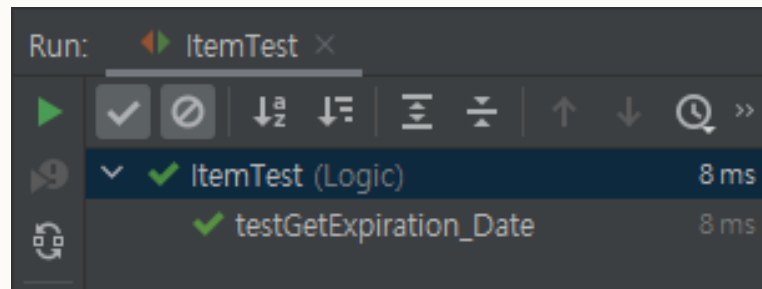
## TitleTest



Run: TitleTest

Test Name	Duration
TitleTest (Logic)	16 ms
testAddItem	8 ms
testItem_List	8 ms
testUpdateStock	0 ms
testCheckStock	0 ms
testDeleteItem	0 ms

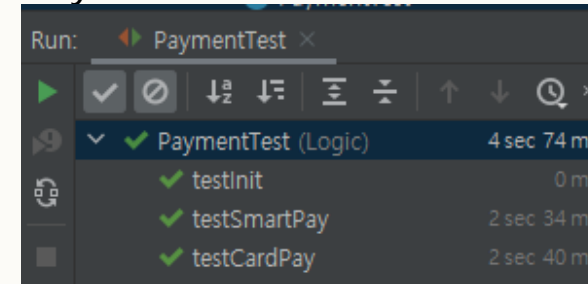
## ItemTest



Run: ItemTest

Test Name	Duration
ItemTest (Logic)	8 ms
testGetExpiration_Date	8 ms

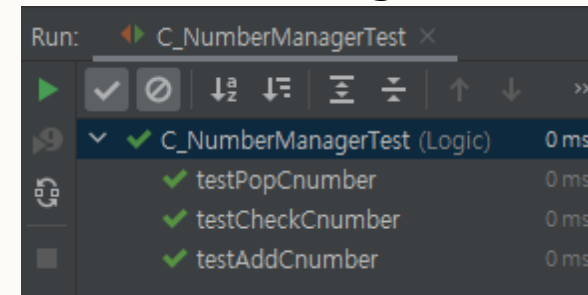
## PaymentTest



Run: PaymentTest

Test Name	Duration
PaymentTest (Logic)	4 sec 74 ms
testInit	0 ms
testSmartPay	2 sec 34 ms
testCardPay	2 sec 40 ms

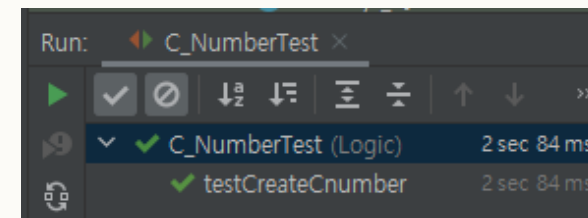
## C\_NumberManagerTest



Run: C\_NumberManagerTest

Test Name	Duration
C_NumberManagerTest (Logic)	0 ms
testPopCNumber	0 ms
testCheckCNumber	0 ms
testAddCNumber	0 ms

## C\_NumberTest



Run: C\_NumberTest

Test Name	Duration
C_NumberTest (Logic)	2 sec 84 ms
testCreateCNumber	2 sec 84 ms

# Activity 2061. Unit Testing

## DVMTest

Run: DVMTest x

Test Name	Duration
✓ DVMTest (Logic)	8 ms
✓ testGetID	8 ms
✓ testGetAddress_X	0 ms
✓ testGetAddress_Y	0 ms

## MessageTest

Run: MessageTest x

Test Name	Duration
✓ MessageTest (Logic)	10 sec 220 ms
✓ testSetmsg1	2 sec 73 ms
✓ testSetmsg2	2 sec 24 ms
✓ testSetmsg3	2 sec 29 ms
✓ testSetmsg4	2 sec 42 ms
✓ testSetmsg5	2 sec 52 ms

## Message\_QueueTest

Run: Message\_QueueTest x

Test Name	Duration
✓ Message_QueueTest (Logic)	4 sec 170 ms
✓ testMsgSend	4 sec 108 ms
✓ testDequeue	62 ms

## ControllerTest

Run: ControllerTest x

Test Name	Duration
✓ ControllerTest (Logic)	2 min 35 sec
✓ manShowItem	3 sec 20 ms
✓ showSmartPay	3 sec 358 ms
✓ manShowTitle	1 sec 841 ms
✓ infoCnumberError	5 sec 598 ms
✓ infoNoItem	3 sec 686 ms
✓ smartPay	5 sec 591 ms
✓ printSelectPay	6 sec 50 ms
✓ inputSelect	30 sec 43 ms
✓ selectPayment	4 sec 673 ms
✓ init	0 ms
✓ manEditItem	22 sec 425 ms
✓ cardPay	6 sec 337 ms
✓ showCardPay	3 sec 748 ms
✓ showInputLine	7 sec 539 ms
✓ printCnumber	5 sec 103 ms
✓ manSelectTitle	7 sec 157 ms
✓ returnItem	3 sec 33 ms
✓ reqFindDVM	15 sec 859 ms
✓ cancelItem	2 ms
✓ inputCnumber	19 sec 720 ms

# Activity 2061. Unit Testing

---

## Test Summary

<b>23</b> tests	<b>0</b> failures	<b>0</b> ignored	<b>0.099s</b> duration
--------------------	----------------------	---------------------	---------------------------

**100%**  
successful

Packages

Classes

Package	Tests	Failures	Ignored	Duration	Success rate
<a href="#">Logic</a>	23	0	0	0.099s	100%



## Activity 2063. System Testing

Ref	Use Case name	Test Description	P/F
R1.1	1. 자판기 음료 종류 출력	- 20가지 메뉴가 정상적으로 출력되는지 확인	P
R1.2.1	2. 구매할 음료 입력	- 사용자가 선택한 음료가 정확히 기록되는지 확인	P
R.1.2.2	3. 사용자 선택 취소	- 사용자 선택 취소 시 메인 화면으로 돌아가는 지 확인	P
R1.2.3	4. 사용자 인증번호/바코드 입력	- 다른 인증번호가 입력된 경우 오류 메시지를 출력하는지 확인	P
		- 올바른 인증번호가 입력된 경우 보관된 제품 중 인증번호에 맞는 제품을 전달하는지 확인	P
R1.3	5. 재고가 부족한 제품 안내	- 재고가 부족한 제품을 정확히 출력하는지 확인	P
R1.4	6. 구매 가능한 자판기 안내	- 정확한 위치정보를 출력하는지 확인	P
		- 지정된 물품의 재고가 정확히 존재하는지 확인	P
		- 지정된 물품의 재고가 존재하는 자판기가 모두 출력되는지 확인	P

## Activity 2063. System Testing

Ref	Use Case name	Test Description	P/F
R1.5	7. 사용자 자판기 선택	- 사용자가 선택한 DVM이 정확히 기록되는 지 확인	P
R2.1	8. 결제 수단 목록 출력	- 사용자가 선택한 음료가 정확히 목록화되어 있는지 확인	P
R2.2	9. 결제 수단 결정	- 사용자가 선택한 결제 수단에 따른 결제 화면으로 넘어가는지 확인	P
R2.3.1	10. 간편 결제	- 사용자가 선택한 제품들이 정상적으로 합산되어 계산 되는지 확인	P
R2.3.2	11. 카드 결제		
R2.4	12. 결제 취소	- 결제 취소 시 메인 화면으로 돌아가는 지 확인	P
R3.1.1	13. 인증번호/바코드 생성	- 인증번호/바코드가 중복되지 않게 생성되는지 확인	P
R3.1.2	14. 인증번호/바코드 출력	- 생성된 인증번호/바코드가 정상적으로 출력되는지 확인	P

## Activity 2063. System Testing

Ref	Use Case name	Test Description	P/F
R3.2	15. 음료 전달	- 사용자가 선택한 제품들이 합산되어 전달 되는지 확인	P
R3.3	16. 재고 수량 업데이트	- 사용자 이용 이후 변화된 재고가 정확히 변화하는지 확인	P
R4.1	17. 관리자 메뉴 출력	- 정확한 절차가 시행됨에 따라 메뉴가 출력 되는지 확인	P
R4.2	18. 재고 변경/관리	- 관리자의 동작에 따라 재고의 수량이 잘 변경되는지 확인	P
R4.3	19. 위치 정보 확인	- 정확한 위치 정보를 반환하는지 확인	P
R4.4	20. 재고 정보 확인	- 정확한 재고 정보를 반환하는지 확인	P
R4.5.1	21. 메시지 발신	- DVM네트워크를 통한 메시지가 정확히 목표 DVM으로 전송되는지 확인	P
R4.5.2	22. 메시지 수신	- DVM네트워크를 통한 메시지가 정확히 목표 DVM으로 수신되는지 확인	P

# Activity 2067. Testing Traceability Analysis

System Function
R1.1 자판기 음료 종류 출력
R1.2.1 구매할 음료 입력
R1.2.2 사용자 선택 취소
R1.2.3 사용자 인증번호 / 바코드 입력
R1.3 재고가 부족한 제품 안내
R1.4 구매 가능한 자판기 안내
R1.5 사용자 자판기 선택
R2.1 결제 수단 목록 출력
R2.2 결제 수단 결정
R2.3.1 간편 결제
R2.3.2 카드 결제
R2.4 결제 취소
R3.1.1 인증번호/바코드 생성
R3.1.2 인증번호/바코드 출력
R3.2 음료 전달
R3.3 재고 수량 업데이트
R4.1 관리자 메뉴 출력
R4.2 재고 변경/관리
R4.3 위치 정보 확인
R4.4 재고 정보 확인
R4.5.1 메세지 발신
R4.5.2 메세지 수신

Use Case
1. 자판기 음료 종류 출력
2. 구매할 음료 입력
3. 사용자 선택 취소
4. 사용자 인증번호 / 바코드 입력
5. 재고가 부족한 제품 안내
6. 구매 가능한 자판기 안내
7. 사용자 자판기 선택
8. 결제 수단 목록 출력
9. 결제 수단 결정
10. 간편 결제
11. 카드 결제
12. 결제 취소
13. 인증번호/바코드 생성
14. 인증번호/바코드 출력
15. 음료 전달
16. 재고 수량 업데이트
17. 관리자 메뉴 출력
18. 재고 변경/관리
19. 위치 정보 확인
20. 재고 정보 확인
21. 메세지 발신
22. 메세지 수신

System Operation
ReqMainMenu()
InputSelect()
CancelItem()
InputCnNumber()
ReqFindDVM()
SelectDVM()
SelectPayment()
SmartPay()
CardPay()
CancelPay()
ManSelectTitle()
ManEditItem()
SendMessage()

Method	Class
1. ReqMainMenu(); void	Controller
2. ShowMainMenu(); void	
3. InputSelect(Title.ID); int	
4. PrintSelectPay(); void	
5. CancelItem(); void	
6. InfoNoItem(); void	
7. InputCnNumber(); int	
8. ShowInputLine(); void	
9. InfoCnNumberError(); void	
10. PrintCnNumber(); void	
11. ReturnItem(Title); void	
12. ReturnMain(); void	
13. ManShowTitle(); void	
14. ManSelectTitle();int	
15. ManShowItem(); void	
16. ManEditItem(); int	
17. SelectPayment(); int	
18. ShowSmartPay(); void	
19. ShowCardPay(); void	
20. CardPay(); int	
21. SmartPay(); int	
22. ReqFindDVM(); int	
23. ShowUsableDVM();void	
24. InfoNoUsableDVM();void	
25. SelectDVM(); int	
26. init(); void	
27. Error_log(); String	
28. SmartPay(); int	
29. CardPay(); int	
30. init(); void	
31. PopCnNumber(C_Number_t), int	
32. CheckCnNumber(C_Number); int	
33. AddCnNumber(C_Number); void	
32. toString(), Stiring	
33. CreateCnNumber(Title); int	
34. randnumber(), String	
35. Item_List(); ArrayList<Item>	
36. AddItem(Title.ID, Item); void	
37. DeleteItem(i_list); void	
38. UpdateStock(Title.ID, IfHold); void	
39. CheckStock(Title.ID);boolean	
40. getExpiration Date() integer	
41. getID(); integer	
42. getAddress_X; double	
43. getAddress_y; double	
44. RecivMsg(); void	
44. SendMsg(); void	
27. Dequeue();void	
28.setMsg(ID, Type,...); void	

Class
Controller
Payment
C_NumberManger
C_Number
Title
item
DVM
MessageQueue
Message

Unit Test
inputSelect(Title.ID);
printSelectPay();
cancelItem();
infoNoItem();
inputCnNumber();
showInputLine();
infoCnNumberError();
printCnNumber();
returnItem(Title);
manShowTitle();
manSelectTitle();
manShowItem();
manEditItem();
selectPayment();
showSmartPay();
showCardPay();
cardPay();
smartPay();
reqFindDVM();
testInit();
testSmartPay();
testCardPay();
testInit();
testPopCnNumber();
testCheckCnNumber();
testAddCnNumber();
testCreateCnNumber();
testAddItem();
testDeleteItem();
testItem_List();
testUpdateStock();
testCheckStock();
testGetExpirationDate();
testGetID();
testGetAddress_X();
testGetAddress_Y();
testGetCurrentID();
testMsgSend();
testDequeue();
testSetMsg1();
testSetMsg2();
testSetMsg3();
testSetMsg4();
testSetMsg5();

# Demonstration Video

---

<https://youtu.be/1VZMeKxbdWw>