

# A NEW C.S.E. ATM

OOPT Stage 2050 & 2060

**Project Team**

T3

**Date**

2018-05-22

-----  
**Team Information**

201311299 이원오

201311301 이재규

201311309 전홍준

# INDEX

- 1. Activity 2051. Implement Class & Methods Definitions**
- 2. Activity 2052. Implement Windows**
- 3. Activity 2055. Write Unit Test Code**
- 4. Activity 2061. Unit Testing**
- 5. Activity 2063. System Testing**
- 6. Activity 2067. Testing Traceability Analysis**

## 1. Activity 2051. Implement Class & Methods Definitions

<b>Type</b>	Class
<b>Name</b>	ATM
<b>Purpose</b>	ATM을 전체적으로 관리하며, Admin과 User Data의 정보를 전달받는 Class
<b>Overview</b>	N/A
<b>Cross Reference</b>	Deposit, Withdraw, Transfer, CheckBalance, RobberyReport, AddCash, CheckCash
<b>Exceptional Course of Event</b>	N/A

<b>Type</b>	Class
<b>Name</b>	User_Data
<b>Purpose</b>	User의 Data 저장과 관리를 담당하는 Class로 File I/O를 통해 File을 관리하는 Class
<b>Overview</b>	N/A
<b>Cross Reference</b>	Deposit, Withdraw, Transfer, CheckBlanace, RobberyReport
<b>Exceptional Course of Event</b>	N/A

<b>Type</b>	Class
<b>Name</b>	Admin
<b>Purpose</b>	관리자로서 ATM기기내의 실 잔액량을 추가하고 확인하면서 관리하는 Class
<b>Overview</b>	N/A
<b>Cross Reference</b>	AddCash, CheckCash
<b>Exceptional Course</b>	N/A

<b>of Event</b>	
-----------------	--

<b>Type</b>	Method
<b>Name</b>	saveInfo
<b>Purpose</b>	변경된 정보의 저장을 담당하는 method
<b>Overview</b>	N/A
<b>Cross Reference</b>	Deposit, Withdraw, Transfer, CheckBalance, RobberyReport
<b>Exceptional Course of Event</b>	N/A

<b>Type</b>	Method
<b>Name</b>	find
<b>Purpose</b>	Account data 들을 User_Data로 객체로 정보를 가져오는 역할
<b>Overview</b>	N/A
<b>Cross Reference</b>	Deposit, Withdraw, Transfer, CheckBalance, RobberyReport
<b>Exceptional Course of Event</b>	해당 Account가 없는 경우 null 반환

<b>Type</b>	Method
<b>Name</b>	getMenu
<b>Purpose</b>	초기화면에서 어떤 선택을 할건지 입력 받는 함수
<b>Overview</b>	N/A
<b>Cross Reference</b>	Deposit, Withdraw, Transfer, CheckBalance, RobberyReport
<b>Exceptional Course of Event</b>	해당되는 번호가 입력되지 않을 경우 에러 확인과 함께 다시 실행.

<b>Type</b>	Method
<b>Name</b>	getAccount
<b>Purpose</b>	사용자로부터 계좌번호를 입력받는 함수
<b>Overview</b>	N/A
<b>Cross Reference</b>	Deposit, Withdraw, Transfer, CheckBalance, RobberyReport
<b>Exceptional Course of Event</b>	N/A

<b>Type</b>	Method
<b>Name</b>	getTransferAccount
<b>Purpose</b>	사용자로부터 이체할 계좌번호를 받는 메소드
<b>Overview</b>	N/A
<b>Cross Reference</b>	Transfer
<b>Exceptional Course of Event</b>	N/A

<b>Type</b>	Method
<b>Name</b>	getAmount
<b>Purpose</b>	사용자로부터 금액을 입력받는 메소드.
<b>Overview</b>	N/A
<b>Cross Reference</b>	Deposit, Withdraw, Transfer, CheckBalance
<b>Exceptional Course of Event</b>	금액의 자리수가 8자리 이상이거나 숫자가 아닐 경우, -1을 리턴

<b>Type</b>	Method
-------------	--------

<b>Name</b>	CheckAmountError
<b>Purpose</b>	금액의 양이 적절한지 판정해주는 메소드
<b>Overview</b>	N/A
<b>Cross Reference</b>	Deposit, Withdraw, Transfer
<b>Exceptional Course of Event</b>	금액이 100만원을 초과하거나 음수일 경우 false 반환

<b>Type</b>	Method
<b>Name</b>	getPw
<b>Purpose</b>	사용자로부터 패스워드를 입력받는 메소드
<b>Overview</b>	N/A
<b>Cross Reference</b>	Withdraw, Transfer, CheckBalance, RobberyReport
<b>Exceptional Course of Event</b>	N/A

<b>Type</b>	Method
<b>Name</b>	getAdminPw
<b>Purpose</b>	관리자모드에서 관리자 비밀번호를 받는 메소드
<b>Overview</b>	N/A
<b>Cross Reference</b>	changeAdmin
<b>Exceptional Course of Event</b>	N/A

<b>Type</b>	Method
<b>Name</b>	getAdminAccount
<b>Purpose</b>	관리자모드에서 관리자 아이디를 받는 메소드
<b>Overview</b>	N/A
<b>Cross Reference</b>	changeAdmin
<b>Exceptional Course of Event</b>	N/A

<b>Type</b>	Method
<b>Name</b>	getCashAmount
<b>Purpose</b>	관리자 모드에서 입력할 현금의 양을 입력받는 메소드
<b>Overview</b>	N/A
<b>Cross Reference</b>	changeAdmin
<b>Exceptional Course of Event</b>	입력값이 숫자가 아니거나 자리수가 8자리 이상일 경우 에러 반환

<b>Type</b>	Method
<b>Name</b>	getRobberyAccount
<b>Purpose</b>	사용자로부터 도난 신고할 계좌를 입력받는 메소드
<b>Overview</b>	N/A
<b>Cross Reference</b>	RobberyReport
<b>Exceptional Course of Event</b>	N/A

## 2. Activity 2052. Implement Windows

<b>Name</b>	Press "입금", "출금", "잔액확인", "계좌이체", "도난신고", "admin"
<b>Responsibilities</b>	GUI의 입금, 출금, 잔액확인, 계좌이체, 도난신고, admin 버튼 을 누른다
<b>Type</b>	GUI
<b>Cross Reference</b>	R0
<b>Notes</b>	GUI의 입금, 출금, 잔액확인, 계좌이체, 도난신고, admin 버튼 을 누른다 메인 화면이다
<b>Pre-Conditions</b>	N/A
<b>Post-Conditions</b>	입금, 출금, 잔액확인, 계좌이체, 도난신고, admin의 화면을 보여준다

<b>Name</b>	Press "입금" 다음 단계에 있는 txt box 안에 값 입력 및 "확인" 버튼 클릭
<b>Responsibilities</b>	전체적인 "입금" 단계 절차 및 실행 확인
<b>Type</b>	GUI
<b>Cross Reference</b>	R1.1, R1.2
<b>Notes</b>	전체적인 "입금" 단계 절차 및 실행 확인
<b>Pre-Conditions</b>	N/A
<b>Post-Conditions</b>	"입금" 첫 단계에서 마지막 단계 까지 순차적으로 실행 끝나고 첫 초기 단계로 복귀

<b>Name</b>	Press "출금" 다음 단계에 있는 txt box 안에 값 입력 및 "확인" 버튼 클릭
<b>Responsibilities</b>	전체적인 "출금" 단계 절차 및 실행 확인
<b>Type</b>	GUI
<b>Cross Reference</b>	R2.1, R2.2
<b>Notes</b>	전체적인 "입금" 단계 절차 및 실행 확인
<b>Pre-Conditions</b>	N/A
<b>Post-Conditions</b>	"입금" 첫 단계에서 마지막 단계 까지 순차적으로 실행 끝나고 첫 초기 단계로 복귀

<b>Name</b>	Press "계좌이체" 다음 단계에 있는 txt box 안에 값 입력 및 "확인" 버튼 클릭
<b>Responsibilities</b>	전체적인 "계좌이체" 단계 절차 및 실행 확인
<b>Type</b>	GUI
<b>Cross Reference</b>	R3.1, R3.2, R3.3
<b>Notes</b>	전체적인 "계좌이체" 단계 절차 및 실행 확인
<b>Pre-Conditions</b>	N/A
<b>Post-Conditions</b>	"계좌이체" 첫 단계에서 마지막 단계 까지 순차적으로 실행 끝나고 첫 초기 단계로 복귀

<b>Name</b>	Press "잔액확인" 다음 단계에 있는 txt box 안에 값 입력 및 "확인" 버튼 클릭
<b>Responsibilities</b>	전체적인 "잔액확인" 단계 절차 및 실행 확인
<b>Type</b>	GUI
<b>Cross Reference</b>	R4.1, R4.2

<b>Notes</b>	전체적인 "잔액확인" 단계 절차 및 실행 확인
<b>Pre-Conditions</b>	N/A
<b>Post-Conditions</b>	"잔액확인" 첫 단계에서 마지막 단계 까지 순차적으로 실행 끝나고 첫 초기 단계로 복귀

<b>Name</b>	Press "도난신고" 다음 단계에 있는 txt box 안에 값 입력 및 "확인" 버튼 클릭
<b>Responsibilities</b>	전체적인 "도난신고" 단계 절차 및 실행 확인
<b>Type</b>	GUI
<b>Cross Reference</b>	R5.1, R5.2
<b>Notes</b>	전체적인 "도난신고" 단계 절차 및 실행 확인
<b>Pre-Conditions</b>	N/A
<b>Post-Conditions</b>	"도난신고" 첫 단계에서 마지막 단계 까지 순차적으로 실행 끝나고 첫 초기 단계로 복귀

<b>Name</b>	Press "admin" 다음 단계에 있는 txt box 안에 값 입력 및 "확인" 버튼 클릭
<b>Responsibilities</b>	"admin" 단계 안에서 "AddCash", "CheckCash"를 할 수 있는 화면 출력
<b>Type</b>	GUI
<b>Cross Reference</b>	R6
<b>Notes</b>	관리자로서 "admin" 기능을 수행 하는 화면 출력
<b>Pre-Conditions</b>	N/A
<b>Post-Conditions</b>	AddCash, CheckCash 할 수 있는 화면 출력

<b>Name</b>	Press "AddCash" 다음 단계에 있는 txt box 안에 값 입력 및 "확인" 버튼 클릭
<b>Responsibilities</b>	전체적인 "AddCash" 단계 절차 및 실행 확인
<b>Type</b>	GUI
<b>Cross Reference</b>	R6.1.1, R6.2.1

<b>Notes</b>	ATM 기기에 실 현금 잔액량 추가
<b>Pre-Conditions</b>	N/A
<b>Post-Conditions</b>	"AddCash" 첫 단계에서 마지막 단계 까지 순차적으로 실행 끝나고 첫 초기 단계로 복귀

<b>Name</b>	Press "CheckCash" 다음 단계에 있는 txt box 안에 값 입력 및"확인" 버튼 클릭
<b>Responsibilities</b>	전체적인 "CheckCash" 단계 절차 및 실행 확인
<b>Type</b>	GUI
<b>Cross Reference</b>	R6.2.1, R6.2.2
<b>Notes</b>	ATM 기기에 실 현금 잔액량 확인
<b>Pre-Conditions</b>	N/A
<b>Post-Conditions</b>	"CheckCash" 첫 단계에서 마지막 단계 까지 순차적으로 실행 행 끝나고 첫 초기 단계로 복귀

### 3. Activity 2055. Write Unit Test Code

#### ATM Testing Class

```
1 package ATM;
2
3 import static org.junit.Assert.*;
4
5 import java.io.IOException;
6
7 import org.junit.Test;
8
9 public class ATMtesting {
10
11     @Test
12     public void testGetMenu_1() {
13         ATM test = new ATM();
14         int t = 1;
15         int tes = test.getMenu();
16         assertEquals(tes,t);
17     }
18
19     @Test
20     public void testGetAccount() {
21         ATM test = new ATM();
22         String account = "940621";
23         String test_acc = test.getAccount();
24         assertEquals(account, test_acc);
25     }
26
27     @Test
28     public void testGetTransferAccount() {
29         ATM test = new ATM();
30         String account = "940621";
31         String test_acc = test.getTransferAccount();
32         assertEquals(account, test_acc);
33     }
34
35     @Test
36     public void testGetAmount() {
37         ATM test = new ATM();
38         int amount = 20000;
39         int test_amount = test.getAmount();
40         assertEquals(amount, test_amount);
41     }
42
43     @Test
44     public void testCheckAmountError() { // 20000원을 넣었을 때 오류가 안뜨는지 확인
45         ATM test = new ATM();
46         boolean isRight;
47         int amount = 20000;
48         isRight = test.checkAmountError(amount);
49         assertEquals(isRight, true);
50     }
51 }
```

```
51
52 @Test
53 public void testGetPw() {
54     ATM test = new ATM();
55     String pw = "4839";
56     String test_pw = test.getPw();
57     assertEquals(pw, test_pw);
58 }
59
60 @Test
61 public void testGetAdminAccount() {
62     ATM test = new ATM();
63     String id = "sklee08";
64     String test_id = test.getAdminAccount();
65     assertEquals(id, test_id);
66 }
67
68 @Test
69 public void testGetAdminPw() {
70     ATM test = new ATM();
71     int pw = 1234;
72     int test_pw = test.getAdminPw();
73     assertEquals(pw, test_pw);
74 }
75
76 @Test
77 public void testGetAdminMenu() {
78     ATM test = new ATM();
79     int menu = 1;
80     int test_menu = test.getAdminMenu();
81     assertEquals(menu, test_menu);
82 }
83
84 @Test
85 public void testGetCashAmount() {
86     ATM test = new ATM();
87     int camount = 20000;
88     int test_camount = test.getCashAmount();
89     assertEquals(camount, test_camount);
90 }
91
92 @Test
93 public void testGetRobberyAccount() {
94     ATM test = new ATM();
95     String raccout = "940621";
96     String test_raccout = test.getRobberyAccount();
97     assertEquals(raccout, test_raccout);
98 }
99
100 @Test
101 public void testGetRobberyPw() {
102     ATM test = new ATM();
103     String rpw = "4839";
104     String test_rpw = test.getRobberyPw();
105     assertEquals(rpw, test_rpw);
106 }
107
```

```
107  
108 @Test  
109 public void testCalculate() throws IOException {  
110     ATM test = new ATM();  
111     String account = "940621";  
112     int amount = 20000;  
113     User_Data d = test.find(account);  
114     boolean isCal = test.calculate(d, amount);  
115     assertEquals(isCal, true);  
116 }  
117  
118 @Test  
119 public void testFind() throws IOException {  
120     ATM test = new ATM();  
121     String account = "940621";  
122     User_Data d = test.find(account);  
123     assertEquals(account, d.getAccountNum());  
124 }  
125  
126 }  
127
```

## ATM Testing Class

```
1 package ATM;
2
3 import static org.junit.Assert.*;
4
5
6
7 public class Admin_testing {
8
9     @Test
10    public void testAdmin_cash() {
11        Admin test = new Admin();
12        boolean cashCheck;
13        if(test.getAdminCash() == 5000000)
14        {
15            cashCheck = true;
16        }
17        else
18            cashCheck = false;
19        assertEquals(true, cashCheck);
20    }
21
22    @Test
23    public void testCheckAdminID() {
24        Admin test = new Admin();
25        boolean idCheck = test.CheckAdminID("sklee08");
26        assertEquals(true, idCheck);
27    }
28
29    @Test
30    public void testCheckAdminPW() {
31        Admin test = new Admin();
32        boolean pwCheck = test.CheckAdminPW(1234);
33        assertEquals(true, pwCheck);
34    }
35
36    @Test
37    public void testAdminCalculate() {
38        Admin test = new Admin();
39        test.AdminCalculate(200000);
40        assertEquals(5200000, test.getAdminCash());
41    }
42
43    @Test
44    public void testGetAdminCash() {
45        Admin test = new Admin();
46        int test_cash = test.getAdminCash();
47        assertEquals(test_cash, 5000000);
48    }
49
50 }
51
```

## User\_Data Testing Class

```
package ATM;

import static org.junit.Assert.*;

public class User_Data_testing {

    @Test
    public void testCheckAccountPw() {
        User_Data test = new User_Data("940621", "4839", null, null, null);
        String account = "940621";
        String pw = "4839";
        boolean isCheck = test.checkAccountPw(account, pw);
        assertEquals(true, isCheck);
    }

    @Test
    public void testCheckBank() {
        User_Data test = new User_Data(null, null, "국민", null, null);
        String bank = "국민";
        boolean isCheck = test.checkBank(bank);
        assertEquals(true, isCheck);
    }
}
```

## 4. Activity 2061. Unit Testing

### ATM Testing Class

Finished after 35.126 seconds

Runs: 14/14

Errors: 0

Failures: 0

- ✓ ATM.ATMtesting [Runner: JUnit 4] (35.106 s)
  - ✓ testGetTransferAccount (2.845 s)
  - ✓ testGetCashAmount (1.908 s)
  - ✓ testGetPw (7.672 s)
  - ✓ testFind (0.003 s)
  - ✓ testGetAccount (3.116 s)
  - ✓ testGetAdminPw (3.876 s)
  - ✓ testCalculate (0.001 s)
  - ✓ testGetRobberyAccount (6.855 s)
  - ✓ testGetAdminAccount (1.828 s)
  - ✓ testCheckAmountError (0.001 s)
  - ✓ testGetAmount (1.878 s)
  - ✓ testGetRobberyPw (3.039 s)
  - ✓ testGetAdminMenu (1.073 s)
  - ✓ testGetMenu\_1 (1.011 s)

### Admin Testing Class

Finished after 0.031 seconds

Runs: 5/5

Errors: 0

Failures: 0

- ✓ ATM.Admin\_testing [Runner: JUnit 4] (0.000 s)
  - ✓ testAdminCalculate (0.000 s)
  - ✓ testCheckAdminID (0.000 s)
  - ✓ testCheckAdminPW (0.000 s)
  - ✓ testAdmin\_cash (0.000 s)
  - ✓ testGetAdminCash (0.000 s)

# User\_Data Testing Class

Finished after 0.031 seconds

Runs: 2/2

Errors: 0

Failures: 0

- ATM.User\_Data\_testing [Runner: JUnit 4] (0.000 s)
  - testCheckAccountPw (0.000 s)
  - testCheckBank (0.000 s)

## 5. Activity 2063. System Testing

Test Number	Test 항목	Description	Use case	System function
1	입금 버튼 시험	입금 화면이 제대로 잘 나오는지 확인	Deposit	R1
2	출금 버튼 시험	출금 화면이 제대로 잘 나오는지 확인	Withdraw	R2
3	계좌이체 버튼 시험	계좌이체 화면이 제대로 잘 나오는지 확인	Transfer	R3
4	잔액확인 버튼 시험	잔액확인 화면이 제대로 잘 나오는지 확인	CheckBalance	R4
5	도난신고 버튼 시험	도난신고 화면이 제대로 잘 나오는지 확인	RobberyReport	R5
6	Admin 버튼시험	Admin 화면이 제대로 잘 나오는지 확인	Admin	R6
7	Add Cash 버튼 시험	Add Cash 화면이 제대로 잘 나오는지 확인	AddCash	R6.1
8	Check Cash 버튼 시험	Check Cash 화면이 제대로 잘 나오는지 확인	CheckCash	R6.2

메뉴를 입력하시오 : (0:종료/1:입금 /2:출금 /3:조회/4:이체/5:관리자/6:도난신고)1  
계좌번호를 입력하시오 : 9711111  
Password : 7435  
금액을 입력하시오 : 20000  
잔액은 6664213  
성공!  
메뉴를 입력하시오 : (0:종료/1:입금 /2:출금 /3:조회/4:이체/5:관리자/6:도난신고)1  
계좌번호를 입력하시오 : 9711110  
cannot find the account  
메뉴를 입력하시오 : (0:종료/1:입금 /2:출금 /3:조회/4:이체/5:관리자/6:도난신고)1  
계좌번호를 입력하시오 : 9711111  
Password : 7444  
wrong pw!  
메뉴를 입력하시오 : (0:종료/1:입금 /2:출금 /3:조회/4:이체/5:관리자/6:도난신고)1  
계좌번호를 입력하시오 : 9711111  
Password : 7435  
금액을 입력하시오 : 20000  
잔액은 6684213  
성공!  
메뉴를 입력하시오 : (0:종료/1:입금 /2:출금 /3:조회/4:이체/5:관리자/6:도난신고)2  
계좌번호를 입력하시오 : 9711111  
Password : 7435  
금액을 입력하시오 : 20000  
잔액은 6664213  
성공!  
메뉴를 입력하시오 : (0:종료/1:입금 /2:출금 /3:조회/4:이체/5:관리자/6:도난신고)2  
계좌번호를 입력하시오 : 9711111  
Password : 7435  
금액을 입력하시오 : 7777777  
금액초과!  
메뉴를 입력하시오 : (0:종료/1:입금 /2:출금 /3:조회/4:이체/5:관리자/6:도난신고)

## 6. Activity 2067. Testing Traceability Analysis

