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

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

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

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

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

Type	Method
Name	find
Purpose	Account data 들을 User_Data로 객체로 정보를 가져오는 역할
Overview	N/A
Cross Reference	Deposit, Withdraw, Transfer, CheckBalance, RobberyReport
Exceptional Course of Event	N/A

2. Activity 2052. Implement Windows

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

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

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

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

Name	Press "잔액확인" 다음 단계에 있는 txt box 안에 값 입력 및 "확인" 버튼 클릭
Responsibilities	전체적인 "잔액확인" 단계 절차 및 실행 확인
Type	GUI
Cross Reference	R4.1, R4.2
Notes	전체적인 "잔액확인" 단계 절차 및 실행 확인
Pre-Conditions	N/A
Post-Conditions	"잔액확인" 첫 단계에서 마지막 단계 까지 순차적으로 실행 끝나고 첫 초기 단계로 복귀

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

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

Name	Press "AddCash" 다음 단계에 있는 txt box 안에 값 입력 및 "확인" 버튼 클릭
Responsibilities	전체적인 "AddCash" 단계 절차 및 실행 확인
Type	GUI
Cross Reference	R6.1.1, R6.2.1
Notes	ATM 기기에 실 현금 잔액량 추가
Pre-Conditions	N/A
Post-Conditions	"AddCash" 첫 단계에서 마지막 단계 까지 순차적으로 실행 끝나고 첫 초기 단계로 복귀

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

3. Activity 2055. Write Unit Test Code

ATM Testing Class

```java
package ATM;

import static org.junit.Assert.*;

import java.io.IOException;

import org.junit.Test;

public class ATMtesting {

    @Test
    public void testGetMenu_1() {
        ATM test = new ATM();
        int t = 1;
        int tes = test.getMenu();
        assertEquals(tes,t);
    }

    @Test
    public void testGetAccount() {
        ATM test = new ATM();
        String account = "940621";
        String test_acc = test.getAccount();
        assertEquals(account, test_acc);
    }

    @Test
    public void testGetTransferAccount() {
        ATM test = new ATM();
        String account = "940621";
        String test_acc = test.getTransferAccount();
        assertEquals(account, test_acc);
    }

    @Test
    public void testGetAmount() {
        ATM test = new ATM();
        int amount = 20000;
        int test_amount = test.getAmount();
        assertEquals(amount, test_amount);
    }

    @Test
    public void testCheckAmountError() {    // 20000원을 넘었을 때 오류가 안뜨는지 확인
        ATM test = new ATM();
        boolean isRight;
        int amount = 20000;
        isRight = test.checkAmountError(amount);
        assertEquals(isRight, true);
    }
```

```
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 raccount = "940621";
96          String test_raccount = test.getRobberyAccount();
97          assertEquals(raccount, test_raccount);
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
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

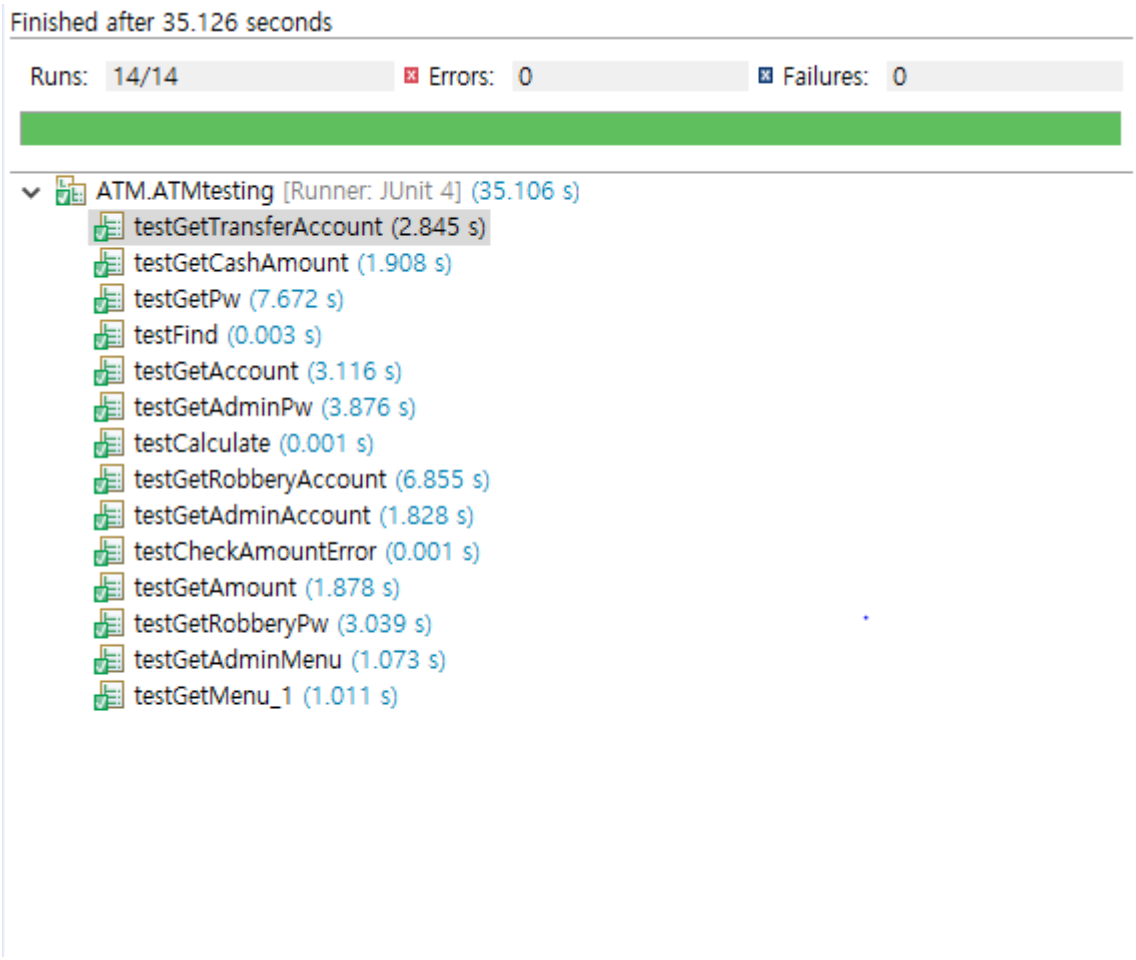
```java
package ATM;

import static org.junit.Assert.*;

public class Admin_testing {

    @Test
    public void testAdmin_cash() {
        Admin test = new Admin();
        boolean cashCheck;
        if(test.getAdminCash() == 5000000)
        {
            cashCheck = true;
        }
        else
            cashCheck = false;
        assertEquals(true, cashCheck);
    }

    @Test
    public void testCheckAdminID() {
        Admin test = new Admin();
        boolean idCheck = test.CheckAdminID("sklee08");
        assertEquals(true, idCheck);
    }

    @Test
    public void testCheckAdminPW() {
        Admin test = new Admin();
        boolean pwCheck = test.CheckAdminPW(1234);
        assertEquals(true, pwCheck);
    }

    @Test
    public void testAdminCalculate() {
        Admin test = new Admin();
        test.AdminCalculate(200000);
        assertEquals(5200000, test.getAdminCash());
    }

    @Test
    public void testGetAdminCash() {
        Admin test = new Admin();
        int test_cash = test.getAdminCash();
        assertEquals(test_cash, 5000000);
    }

}
```

User_Data Testing Class

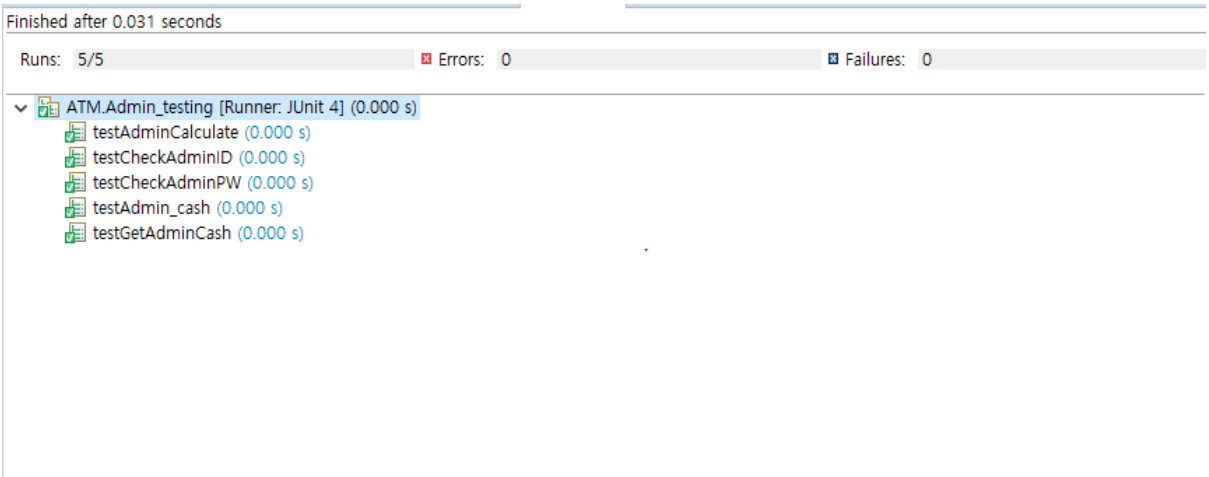
```java
package ATM;

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

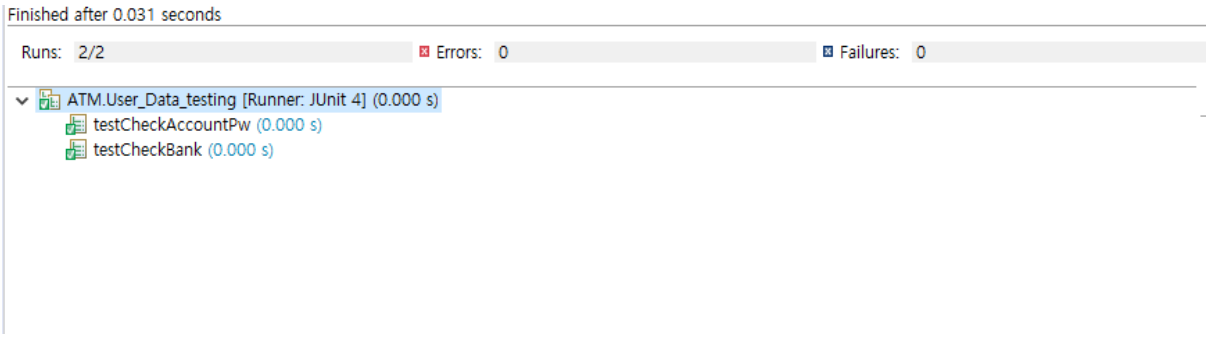
ATM Testing Class



Admin Testing Class



User_Data Testing Class



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 : 971111
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 : 971110
cannot find the account
메뉴를 입력하시오 : (0:종료/1:입금 /2:출금 /3:조회/4:이체/5:관리자/6:도난신고)1
계좌번호를 입력하시오 : 971111
Password : 7444
wrong pw!
메뉴를 입력하시오 : (0:종료/1:입금 /2:출금 /3:조회/4:이체/5:관리자/6:도난신고)1
계좌번호를 입력하시오 : 971111
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 : 971111
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 : 971111
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