Software Modeling & Analysis

Global ATM System

-system test & static test respond

Project Team

1 Team

Date

2018-03-31

Team Information

201311287 엄현식

201311318 최정헌

201611293 전다윤

목차

1 Specification

1.1 Stage 1000 Planning

1.1.1 1003.Define Requirements 수정

1.1.2 1004 Record Terns in Glossary

실제 ATM 기기를 만든다고 설계했으며 , 컴퓨터안 interface 이기 때문에 실제 지급할 수 없지만 , 설계상 문제는 아니라 판단

1.1.3 1008 Define Business Concept Models

Business Concept Model 은 Object 를 뽑아보는 것일 뿐 , 다음 단계에 들어가지 않아도 된다고 배워서 문제가 되지 않음.

1.2 Stage 2030 Analysis

1.2.1 2031. Describe Use Case

1.2.1.1 Check ,Deposit ,Withdraw ,Issue TrafficCard 수정

1.2.1.2 Update

Stage 2040 Describe real use-case : Update – setBalance 등 method 로, 실제 설명하지 않은 부분 "불일치문제가 있다" 고 쓰여져 있다.

1.2.1.3 Status Alarm

관리자에게 알람을 보내는 목적으로 구현을 진행했고 , 만약 testing 과정에서 알람이 보내주지 않았다면 ,Specific review 가 아닌 system testing review 말해야 되는 false 가 일어난 부분이다.

1.2.2 2033. Define Domain Model

Domain 모델에 나온 모든 부분을 사용하지 않아도 된다고 배움 수정될 필요 X

1.2.3 2034. Refind Glossary

*Clossary 라 쓰여 있는데 , Glossary 를 뜻하는 거라 판단

Update , Verrify Sufficient Fund , Status Alarm -다음단계에서 사용하지 않더라도 , use-case 에 나온 부분, 어떤 의미인지 설명이 필요하다 판단.

History , date , senser countyid – domain medel 에서 무엇을 의미하는지 설명 되지 않아, 설명이 필요하다 판단.

1.2.4 Define System Sequence Diagrams

1.2.4.1 Deposit

실제 현금 투입구를 개방할 수 없어 구현에 없지만 , 실제 ATM 에 필요한 과정이라 판단.

1.2.5 2036. Define Operation Contracts

1.2.5.1 selectService 수정

1.2.5.2 insertCash 수정

1.2.5.3 end(void)

Stage 1000 에서 추가할 필요 X

-Management use-case 안 필요한 method 이다.

-insertCash() , selectService 등 대부분의 것들이 Stage 1000 에 정의 되어 있지 않은데 , 언급한 이유를 알수 없다.

1.2.6 2039 Traceability Analysis 수정

1.3 Stage 2040 Design

1.3.1 2041 Design Real Use-case – 중복

1.2.1 에서 언급

1.3.2 2144. Define Interaction Diagrams

Return 값에 alarm 은 메일 알람을 보낸다는 뜻으로 쓴것이다.

readItem- 생명선이 왜 끝까지 이어져야 하는지 판단하기 어렵다.

사용자가 넣은 item 을 읽어주는 method 인고, 읽고 ATM 안 읽은 정보를 저장하면 끝나야 한다고 판단하였다.

1.3.2.1 Check

Sequence Diagram 과 흐름이 다르다

-> 2030 sequence diagram 과 흐름이 같다 / 무엇이 다르다고 하는지 알 수 없다.

loop 문 , opt 같은 경우에는 몇 개인지 상세하게 적지 않아도 된다고 하셨었기 때문에 저렇게 작성한 것이고 , 굳이 고치지 않기로 판단

->다른 interaction diagram 도 같은 이유로 수정하지 않음 .

readItem 전에 말한 내용 중복

객체들의 정보 수정/반영(?) -> method 인자값을 뜻한다 판단 -수정

Confirm 도 비밀번호를 맞게 입력하면 , 끝나기 때문에 , printReceipt 까지 이어질 필요가 없다고 판단

실제 프로그램에서 atm 이 아닌 gui controller 가 printReceipt , getBalance 를 사용함. 비록 , atm 안에서 하지는 않지만 전체적 순서는 diagram 과 같다.

1.3.2.2 Deposit

외화를 나누는 분기 -atm 은 들어온 현금을 읽는 역할만 하고 ,gui 에서 하나의 금액을 선택하도록 한다.

Deposit(money)의 반환 값은 money 가 아니라 Boolean type 으로 true/false 이므로 수정할 필요가 없다.

유효한 현금인지는 insertCash 에서 판단하므로 따로 나타내지 않는다-수정할 필요 없다 판단.

printReceipt 의 pre – condition 수정 -중복

1.3.2.3 withDraw

loop 문 조건 – check 와 중복

readItem 반환값인 languageMode 는 user 에게 보내는게 아니기 때문에 수정 x

selectService 는 result 값에 따라 바뀌지 않는다.

13,12 상관관계 수정

잔고 부족 검사하는 부분은 누락되지 않았다. (알람기능을 사용하지 못해서 누락되었다고 판단했을 거라 추측)

printReceipt 조건문 삭제 -> 다른 interaction diagram 도 모두 잘못되있어 수정

printReceipt 에 receiptAmount 변화하는 부분 code 안에 존재 -명시할 필요 없다 판단.

1.3.2.4 Transfer

예외 부분은 명시할 필요 X

Confirm->inputPassword 와 같은 의미 이다.

inputTransfer->destAccount 존재

printReceipt 도 출력하는 부분 존재

->transfer interaction diagram 에 존재하는 부분이 모두 없다고 언급

1.3.2.5 IssueTrafficCard

Confirm 이 user 가 password 를 입력하는 부분이다.

-뒤에 내용도 모두 같다.

dataRange 정하는 부분 존재.

명세표 출력을 선택하면 printReceipt – 존재한다.

setDataRange 는 카드의 유효기간을 설정한다. 생명선이 이어져야 할 필요 x

⇨ transfer interaction diagram 에 존재하는 부분이 모두 없다고 언급

1.3.2.6 Management

Sequence diagram 과 흐름이 같다.

End 되기 전 관리자 제어하는 부분도 존재한다.

프로그램상 존재한다.

⇨ 이부분에 관해 없다고 판단하여 test 를 진행하지 않은 것으로 보임

1.3.3 2045 Define Design Class Diagrams – 수정

1.4 Stage 2050 Implementation

1.4.1 2051 Implement Class & Method

모든 cross Reference 수정

Update 와 같은 use-case 는 존재한다.

1.4.1.1 insertCash

Update 는 존재하는 use-case

나머지는 모두 수정

1.4.1.2 printReceipt &setDataRange - 수정

1.4.1.3 end -이전 단계에서 나옴. 수정할 필요 X

1.4.1.4 confirm& writeData 수정

1.4.1.5 set_balance()

이 메소드는 계좌 잔액을 바꾸는 method 로서 , 입금 , 출금 , 교통카드발급에 쓰이며 계좌의 잔고를 증감 시키므로 , purpose 의 증감시킨다는 틀리지 않다.

1.4.1.6 add_link -구체적 목적 설명 -> 수정

1.4.1.7 waitReadItem -> item 입력을 기다린다는 뜻이다.- 수정할 필요 X

1.4.1.8 selectNation – 오타 수정

1.4.2 System test 후 수정

2 Test Report review

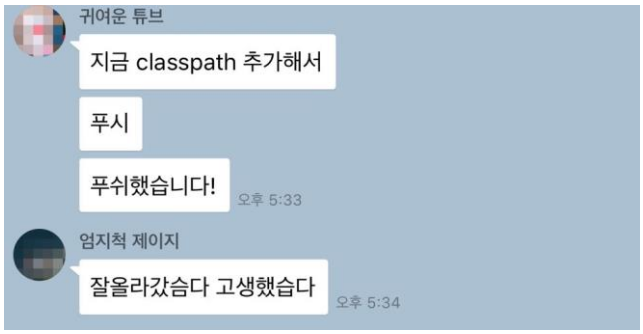
Testing 한 결과 , 0% 로 아무것도 되지 않는 다 답변하였다.

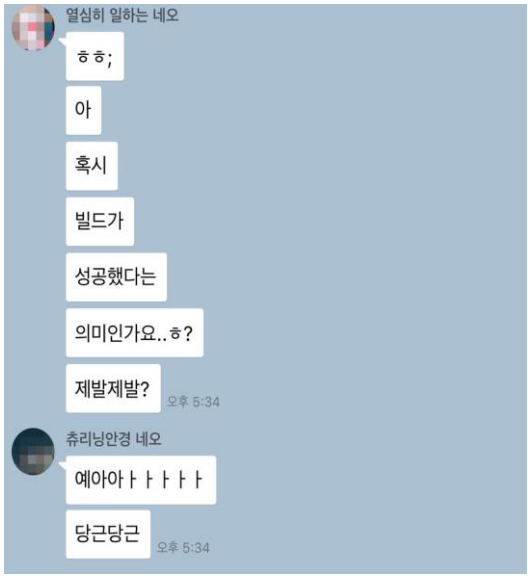
Error 가 난 이유

2.1 Path 설정 오류(해결 시 25% success)

환경에 따라 구동이 안될 수 있는 상황 및 테스팅 환경, 테스팅 방법을 위한 메뉴얼을 제공하였다.

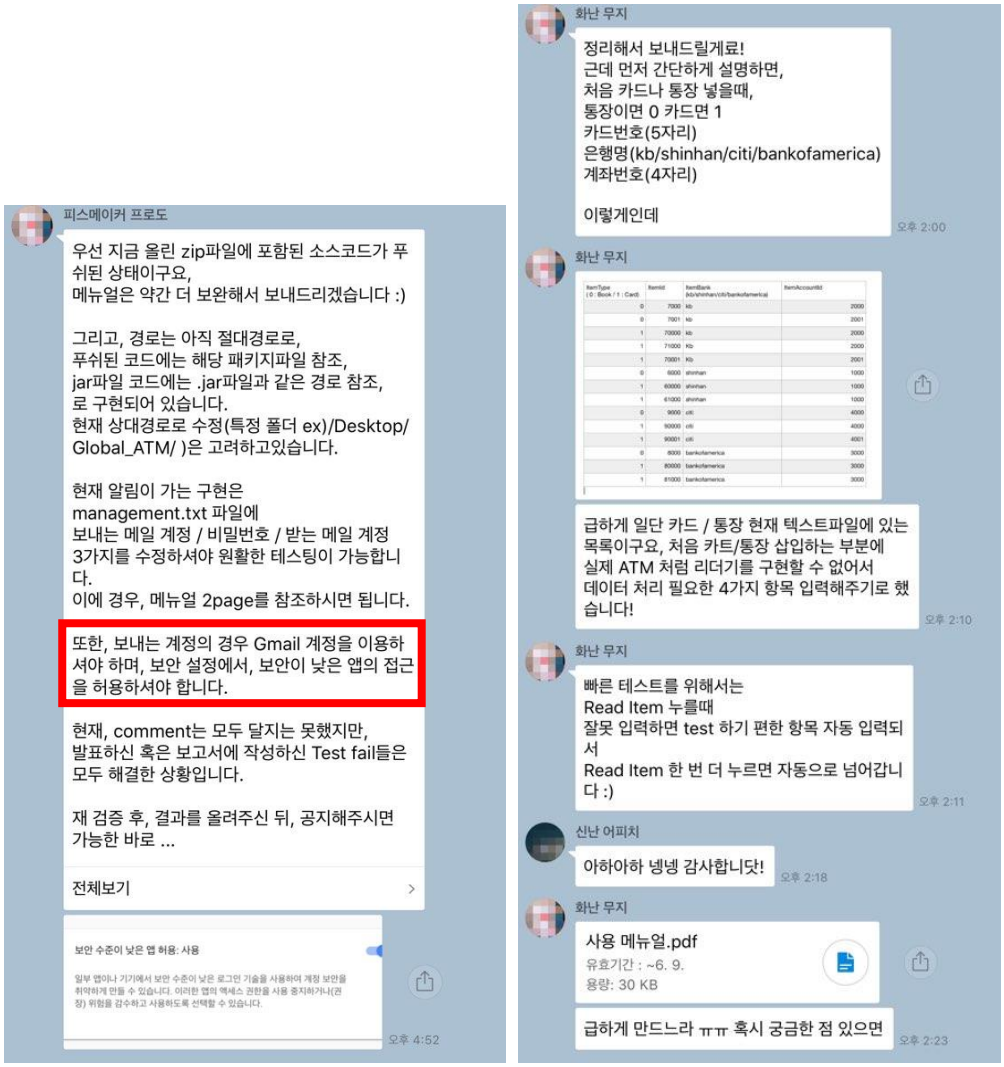
FileNotFoundException 오류 및 빌드 오류를 수정하여 push 하였고 성공적으로 빌드되었다는 답변을 받았습니다.





알람 기능 오류 (해결 시 25% -> 78% success)

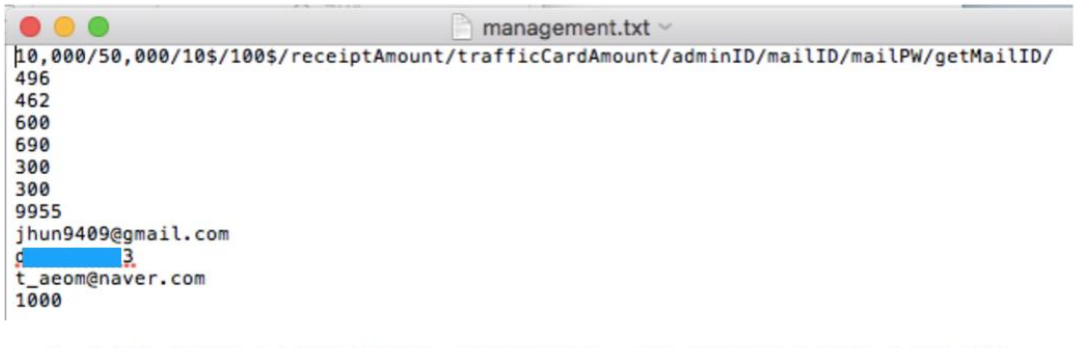
Google e-mail(smtp)을 통해 관리자에게 알람을 보내는 시스템을 이용하여서 몇가지 실행하기전, 필요한 매뉴얼이 있었고, 그에 관해 제공해 주었다.



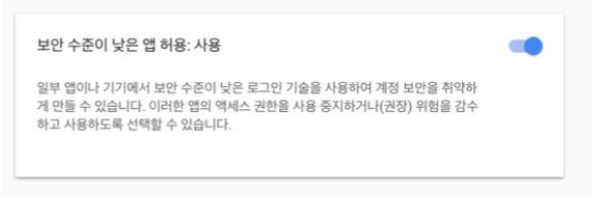
그러나, 결과적으로 , 알람 기능이 제대로 작성되지 않았다는 테스팅 결과를 받았다.

*Manual

8. management.txt



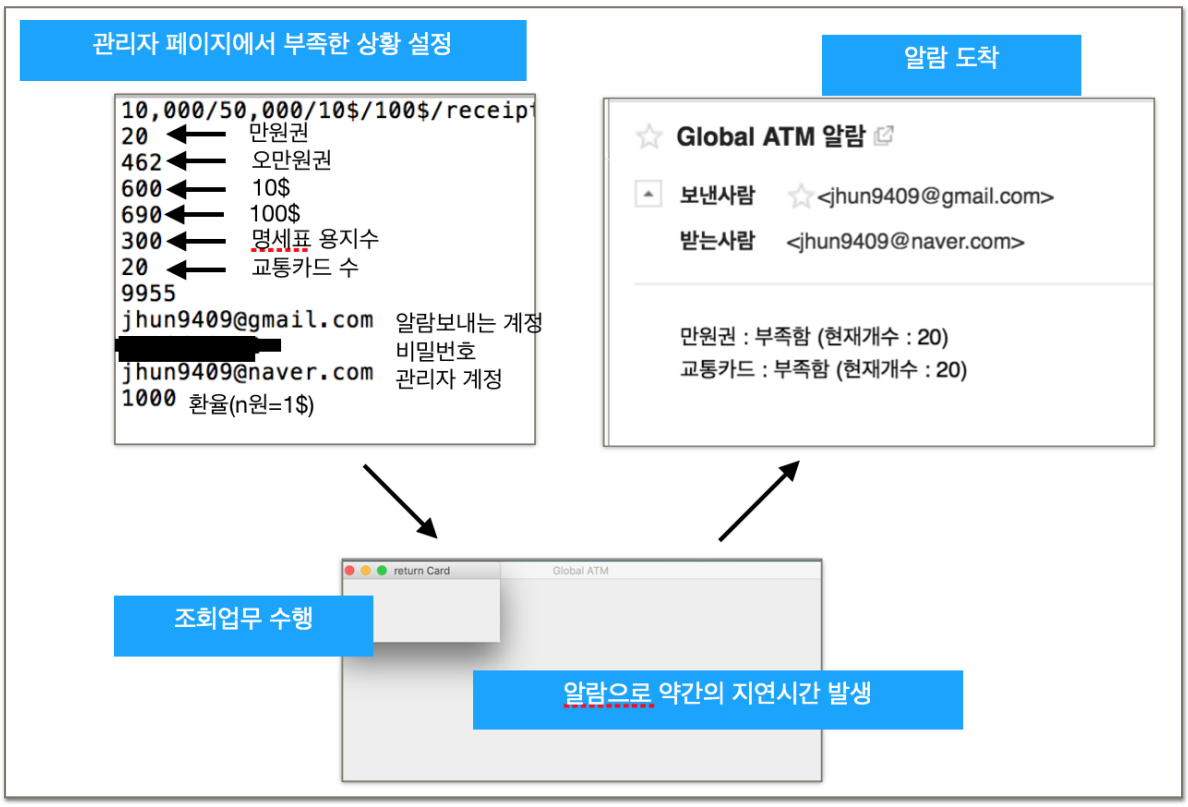
1. ATM이 소지중인 1만 원 지폐 (MAX : 1000(입금불가) , 900~999(과다) 0~99(과소) 알림 발생)
2. ATM이 소지중인 5만 원 지폐 (MAX : 1000(입금불가) , 900~999(과다) 0~99(과소) 알림 발생)
3. ATM이 소지중인 10$ 지폐 (MAX : 1000(입금불가) , 900~999(과다) 0~99(과소) 알림 발생)
4. ATM이 소지중인 100$ 지폐 (MAX : 1000(입금불가) , 900~999(과다) 0~99(과소) 알림 발생)
5. ATM이 소지중인 명세표 용지 (0~99(과소) 알림 발생)
6. ATM이 소지중인 교통카드 (0~99(과소) 알림 발생)
7. Management.txt 접근 Key
8. 알림 보내는 mail address
   - Gmail 계정을 이용하셔야 하며, 보안 설정에서, 보안이 낮은 앱의 접근을 허용하셔야 합니다.



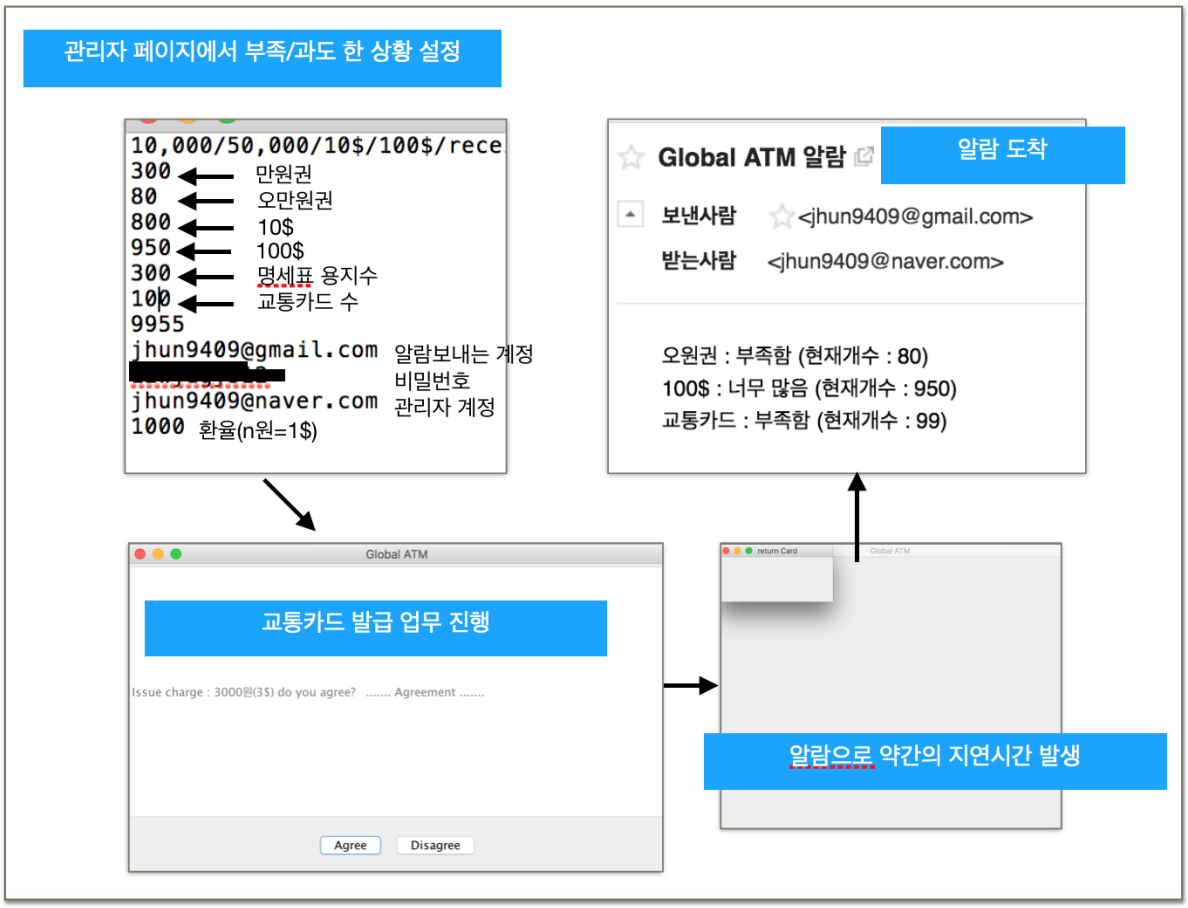
9. 알림 보내는 mail pw
10. 알림 받는 mail address
11. 학용 (N원 : 1$)

2.2 실제 개발팀에서 실행한 과정 및 결과

만원권(20 개) / 교통카드 수(20 개) 부족한상황에서 조회 업무 진행할 때 알림 도착



오만원권 (80 개) / 100$ (950 개) / 교통카드 수 (100 개) 일때 교통카드 발급 업무 진행



Manual 과 메시지를 통해 관리자 알람을 보내기 위해 관리자 txt 파일 및 구글계정 설정을 변경해야 하는데 , 이 부분을 제대로 숙지 하지 못해서 에러가 난 것으로 판단할 수 있었다.

2.3 예외 처리 (해결 시 78% -> 100% success)

실제 예외처리 에러는 소프트웨어검증팀 발표 2 일 전인 6/2 일날 수정사항을 전달했으나 , 이미 검증을 마친 후여서 소프트웨어검증팀 발표에 적용되지 않았다.

2.3.1 GUI 화면 출력 오류 -수정

2.3.2 여러 화폐가 동시에 입금되지 않는 오류

문서부터 여러 종류의 화폐는 동시에 입금되지 않는다고 정의 되어 있다.

2.3.3 숫자 입력 시 불편함 -수정

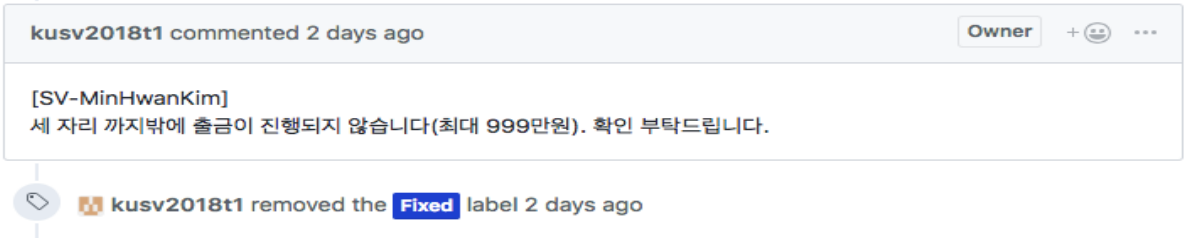
2.3.4 여러 화폐가 동시에 입금되지 않는 오류

문서부터 여러 종류의 화폐는 동시에 입금되지 않는다고 정의 되어 있다.

2.3.5 만원 이하의 돈 , 10 달러 이하의 돈은 처음부터 출금할 수 없다고 설정

3 Pairwise Testing Report Review

3.1 1000 만원 이상 출금 불가

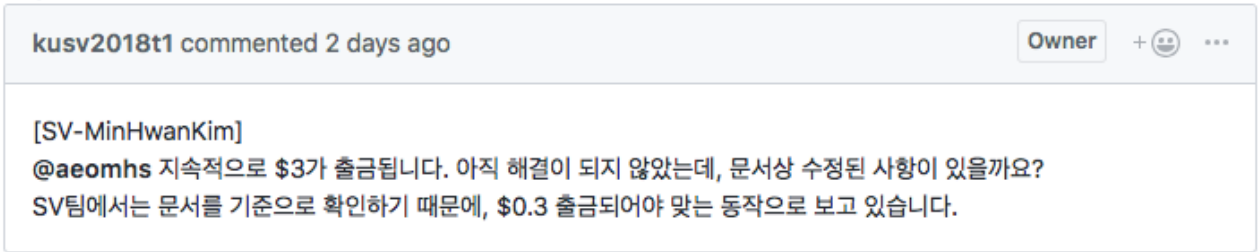


1000 만원 이상을 출금 가능하게 한다는 내용은 은행 정책에 관한 문제인데,

실제(현실) ATM 기기에서는 1 회 출금할 경우, 최고 100 만원까지 가능하며, 1000 만원 이상 출금 불가능합니다.

저희 프로젝트는 이 부분은 문제점이 아닌 당연한 예외 처리라고 생각하고, 때문에 문서에도 따로 명시하지 않고 구현하였습니다.

3.2 카드발급 수수료 오류



문서 상 어디에도 0.3 달러가 수수료로 표시되어 있지 않으며 ,구현에서도 3$(즉 문서상 환율로 3000 원) 가 출금되는게 맞습니다.

3.3 금액 인출

금액을 선택한 뒤, 표시되는 거래 결과창에는 인출된 지폐와는 상관없이 해당 계좌의 국적에 맞게 금액이 표시되도록 구현했습니다. 즉, 거래결과창에는 외국 은행의 경우 달러로 표시하는 것이 맞고, 인출된 돈은 하드웨어 적으로 잘 처리되었다고 가정했습니다

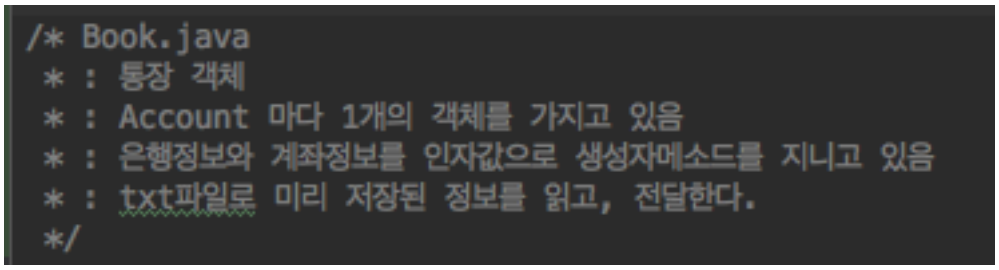
4 Code Scroll review

4.1 주석

4.1.1 문제

클래스의 설명 주석이 없거나, 설명 주석 내용에 모든 태그가 포함되어있지 않음	JAVA_71	높음	Book.java	8	Item	Item
메서드의 설명 주석이 없거나, 설명 주석 내용에 모든 태그가 포함되어있지 않음	JAVA_72	높음	Book.java	16	Book	Item
메서드의 설명 주석이 없거나, 설명 주석 내용에 모든 태그가 포함되어있지 않음	JAVA_72	높음	Book.java	54	Book	Item
메서드의 설명 주석이 없거나, 설명 주석 내용에 모든 태그가 포함되어있지 않음	JAVA_72	높음	Book.java	58	Book	Item

4.1.2 해결(주석 추가)



4.2 Exceoption

4.2.1 문제

메서드 (printStackTrace)가 사용됨.	JAVA_44	매우 높음	Book.java	50	Book.Book(String, int)	Item
catch 절에서 exception(java.io.FileNotFoundException)이 그대로 출력됨.	JAVA_21	높음	ATM.java	96	ATM.ATM()	ATM

4.2.2 해결

모두 printStackTrace()로 대체 (오류 확인을 위한 콘솔 메시지 출력 유지)

4.3 스타일 개선

4.3.1 문제

숫자 상수를 사용함(for loop의 counter로 - 1이나 0, 1을 사용하는 경우 제외)	Sun_24	매우 낮음	Book.java	18	Book.Book(String, int)	Item

4.3.2 해결 -스타일 개선

```
//parsing index Accountid  Bookid   Bookpwd
private int PARSING_AID = 4;
private int TAB_SIZE = 2;
private int ID_SIZE = 4;
private int PWD_SIZE = 4;
private int PARSING_BID = PARSING_AID + TAB_SIZE + ID_SIZE;
private int PARSING_BPWD = PARSING_BID + TAB_SIZE + PWD_SIZE;
```

```
while((getStr = br.readLine()) != null) {
    //finding..
    if(getStr.substring(0, PARSING_AID).equals(String.valueOf(aid))) {
        bid = Integer.parseInt(getStr.substring(PARSING_AID + TAB_SIZE, PARSING_BID));
        bpwd = Integer.parseInt(getStr.substring(PARSING_BID + TAB_SIZE, PARSING_BPWD));
        break;
    }
}
```

4.4 C 스타일-사전협의 부족으로(관련 문제 언급 X)

해당 소스 파일이 C 스타일 주석으로 시작하지 않음	Sun_03	매우 낮음	Book.java	1	Item	Item
한 줄에 여러 식별자를 선언함	Sun_07	매우 낮음	Book.java	12	Book	Item

4.5 "숫자 상수 사용 금지"

숫자 상수의 경우 final 변수를 두어 따로 선언하여 사용하도록 수정

```
//input password
private char [] input_pw = new char[4];
private int pw_i;

//insert cash
private char [] input_cash = new char[3];
private int cash_i;
private String ptn_s;

//input Transfer accountid
private char [] input_id = new char[4];
private int id_i;

//input Date Range
private char [] input_date = new char[2];
private int date_i;
```

```
//Size
private final int PWD_SIZE = 4;
private final int CASH_STRING = 3;
private final int AID_SIZE = 4;
private final int TC_DATE_SIZE = 2;

//input password
private char [] input_pw = new char[PWD_SIZE];
private int pw_i;

//insert cash
private char [] input_cash = new char[CASH_STRING];
private int cash_i;
private String ptn_s;

//input Transfer accountid
private char [] input_id = new char[AID_SIZE];
private int id_i;

//input Date Range
private char [] input_date = new char[TC_DATE_SIZE];
private int date_i;
```

```
//parsing index Accountid  Bookid   Bookpwd
private final int PARSING_AID = 4;
private final int TAB_SIZE = 2;
private final int ID_SIZE = 4;
private final int PWD_SIZE = 4;
private final int PARSING_BID = PARSING_AID + TAB_SIZE + ID_SIZE;
private final int PARSING_BPWD = PARSING_BID + TAB_SIZE + PWD_SIZE;
```

```
//Index Num (Bank.txt)
private final int NAME_INDEX = 0;
private final int AID_INDEX = 1;
private final int BALANCE_INDEX = 2;
private final int TCID_INDEX = 3;
```

```
//parsing index Accountid  Bookid   Bookpwd
private final int PARSING_AID = 4;
private final int TAB_SIZE = 2;
private final int ID_SIZE = 5;
private final int PWD_SIZE = 4;
private final int PARSING_CID = PARSING_AID + TAB_SIZE + ID_SIZE; // 4 + 2 + 5 = 11
private final int PARSING_CPWD = PARSING_CID + TAB_SIZE + PWD_SIZE; // 10 + 2 + 4 = 17
```

```
//Bank Index
private final int BANK_TYPE_NUM = 4;
private final int BANK_KB = 0;
private final int BANK_SH = 1;
private final int BANK_CT = 2;
private final int BANK_BA = 3;
private Bank[] bank = new Bank[BANK_TYPE_NUM];
private int usingBankID;

//ATM Item "Traffic Card"
private TrafficCard tCard;
private int trafficCardAmount;

//ATM Item "Cash"
private final int MAX_CASH = 1000;
private final int CASH_TYPE_NUM = 4;
private final int CASH_TYPE_MAN_WON = 0;
private final int CASH_TYPE_0_MAN_WON = 1;
private final int CASH_TYPE_TEN_DOLLAR = 2;
private final int CASH_TYPE_HUNDRED_DOLLAR = 3;
private int[] cashAmount = new int[CASH_TYPE_NUM];
```

```
//Transaction Mode
private final int READY_MODE = 0;
private final int CHECK_MODE = 1;
private final int DEPOSIT_MODE = 2;
private final int WITHDRAW_MODE = 3;
private final int TRANSFER_MODE = 4;
private final int ISSUE_MODE = 5;
private int transactionMode = READY_MODE;

//Account Nation (KOR / ENG)
private final int ACC_KOR = 0;
private final int ACC_ENG = 1;
private int accountNation;
```

```
//withdraw Cash type
private final int CASH_KOR = 0;
private final int CASH_ENG = 1;
private int cashNation;

//Fee
private final int FEE_KOR = 1000;
private final int FEE_ENG = 1;

//$ Type
private final int TEN_DOLLAR = 10;
private final int HUNDRED_DOLLAR = 100;
//won Type
private final int MAN_WON = 10000;
private final int OMAN_WON = 50000;
```

4.6 GUI

GUI 의 경우, 미리 맞춰져 있는 스타일을 유지하기로 함 (대부분 위배 GUI 에서 발생)

이 외에서 위배 된 경우, 수정함

4.7 생성자에서 초기화해주는 전역변수를 제외하고 모두 수정

```java
public ATM() {
    try {
        if(path){
            bootATM = new File(path_1);
        }
        else{
            bootATM = new File(path_2);
        }
        fr = new FileReader(bootATM);
        br = new BufferedReader(fr);

        //mention delete
        br.readLine();

        //declare getStr
        String getStr;
```

```java
public ATM() {

    //declare getStr
    String getStr;
```

4.8 메서드 주석 없는 것은 추가 / 스타일 통일

```java
/* Initialization
 * Loading Text file "management.txt"
 * Setting attributes
 */
public ATM() {...}

/* Method
 * Description : 읽힌 아이템의 계좌 정보를 통해 유효한 값인지 확인한다.
 */
public int readItem(int itemType, int itemID, String bankID, int accountID) {...}

/* Method
 * Description : 고객이 선택한 서비스 정보를 저장한다.
 */
public void selectService(int service) {...}

/* Method
 * Description : 출금시 어떤 지폐(원/달러)를 원하는지 선택한 정보를 저장한다.
 */
public int selectNation(int nation) {...}

/* Method
 * Description : 고객이 입력한 비밀번호를 확인한다.
 */
public boolean confirm(int pwd) { return bank[usingBankID].confirm(pwd); }

/* Method
 * Description : 고객이 입금한 지폐를 계산한다.
 */
public int insertCash(String[] bill) {...}
```

```java
/* Initialization
 * Loading Text file "Bank.txt"
 * Setting attributes
 */
public Bank(String _bankName) {...}

/* Method
 * Description : 해당 계좌 정보가 유효한지 검사한다.
 */
public boolean validCheck(int _itemType, int _itemID, int _accountID) {...}

/* Method
 * Description : 비밀번호가 올바른지 검사한다.
 */
public boolean confirm(int _pwd) {...}

/* Method
 * Description : 계좌에 발급하는 교통카드 id를 연동한다.
 */
public boolean linkAccount(int _tcid) {...}

/* Method
 * Description : 계좌 잔액을 읽어온다.
 */
public int getBalance() {...}

/* Method
 * Description : 교통 카드 발급을 진행한다.
 */
public boolean chargeTrafficCard(int _money) { return this.withdraw(_money); }

/* Method
 * Description : 출금을 진행한다.
 */
public boolean withdraw(int _money) {...}

/* Method
 * Description : 입금을 진행한다.
 */
public boolean deposit(int _money) {...}

/* Method
 * Description : 송금 대상의 계좌가 유효한지 검사한다.
 */
public String checkAccount(String _bankID, int _accountID) {...}

/* Method
 * Description : 송금을 진행한다.
 */
```

```java
/* Method
 * Description : 출금 / 송금시 거래할 금액을 계산한다.
 */
public int enterAmount(int money) {...}

/* Method
 * Description : 거래 후, 잔액을 가져오고, ATM의 수정된 소지물 정보를 저장한다.
 */
public int getBalance() {...}

/* Method
 * Description : 명세표 출력시 명세표 용지를 계산한다.
 */
public boolean printReceipt(){...}

/* Method
 * Description : 고객이 원하는 교통카드 사용 날짜를 계산한다.
 */
public boolean setDataRange(int date_range) {...}

/* Method
 * Description : 교통카드 발급시 고객의 동의 여부를 판단한다.
 */
public boolean agreement() {...}

/* Method
 * Description : 송금시 수신 계좌가 유효한지 검사한다.
 */
public String destAccount(String bankID, int accountID) {
    return bank[usingBankID].checkAccount(bankID, accountID);
}
```

```java
// Bank init
public Bank(String _bankName) {...}

// valid check
public boolean validCheck(int _itemType, int _itemID, int _accountID) {...}

// confirm [modified] only pwd
public boolean confirm(int _pwd) {...}

// Link Account
public boolean linkAccount(int _tcid) {...}

// get balance
public int getBalance() {...}

// charge Traffic Card [modified]
public boolean chargeTrafficCard(int _money) { return this.withdraw(_money); }

// withdraw [modified] too much same as charge Traffic Card
public boolean withdraw(int _money) {...}

// deposit
public boolean deposit(int _money) {...}

// check Account [modified] bank ID string
public String checkAccount(String _bankID, int _accountID) {...}

// transfer [modified] boolean
public boolean transfer(int _money) {...}
```

4.9 "*"를 이용한 import 문 사용-> 수정

5 Summary

1~3 문서 , 코드 일치-수정

4 -GUI 에서는 예외처리를 제외하고는 어떤 연산도 하지 않는다.

정상적인 계산을 하기 위해서는 올바른 값들을 (System Operation 들에게) 보내주기 위해 그에 대한 거름망 역할을 GUI 에서 해주는 것이다.

5 -OOAD 를 통해 부족하지만, 최대한 객체 지향적으로 접근했다.

어떤 부분이 왜 객체 지향적이지 않은 지 알려주지 않아 대략 난감하다.

6-문서와 일치시키다 보니, 몇 좋지 않은 Interface 가 구현되기도 했다.

예를 들면, 송금, 출금시 System Operation 이 동일하여, 송금도 단위가 고정되는 등,

하지만 연산이 잘못 되어 고객에게 피해가 가는 상황이 발생되지는 않았다.

7-코드 상에 존재한다.

System 작동 시, 별도의 GUI Frame 이 존재한다

기준이 사전에 토의되지 않았다.

고객이 사용하는데 필요한 기능 및 설명이 나와있다.