Software Modeling & Analysis

Global ATM System

-Stage 2050 Construct & Stage 2060 Testing-

Project Team
1 Team

Date
2018-05-22

Team Information
201311287 엄현식
201311318 최정헌
201611293 전다윤

목차

1. Activity 2051. Implement Class &Methods Definitions

1.1. ATM

Type	Class
Name	ATM
Purpose	User 가 해당 시스템을 사용할 수 있도록 한다.
Overview	-
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.1.4 , R.2.0 , R.2.1, R.2.2, R.3.0 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw” , “IssueTrafficCard”,“Management” , “Update” , “Verify Sufficient Fund” , “Status Alarm”
Exceptional Course of Events	-

1.1.1. readItem

Type	Method
Name	readItem
Purpose	User 가 읽힌 Item 의 정보를 가져와 해당 계좌를 찾는다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw”
Input (Method)	Int itemType , int itemID , String bankID , int accountID
Output (Method)	int
Abstract operation (Method)	Itemtype(card/book), itemID(cid/bid) , 은행 이름 , 계좌 번호를 입력하면 해당 은행을 찾고 은행에게 가능한 계좌인지 찾으라 한다. 해당 계좌를 usingAccountID 로 설정한다. 해당계좌가 한국계좌면 0, 외국 계좌면 1 을 반환한다
Exceptional Course of Events	잘못된 item, 은행일 경우 , -1 을 반환한다.

1.1.2. selectService

Type	Method
Name	selectService
Purpose	User 가 선택한 서비스를 제공한다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw”
Input (Method)	int service
Output (Method)	Void
Abstract operation (Method)	계좌조회(check/1) , 입금(deposit/2) , 출금(withdraw/3) , 송금(transfer/4) 교통카트 발급(issueTrafficCard/5) 중 하나의 서비스를 선택하는 것
Exceptional Course of Events	

1.1.3. selectNation

Type	Method
Name	selectNation
Purpose	User 가 출금시, 거래할 지폐종류(원/달러)를 선택한다.
Cross Reference	System Function : R.1.1 Use cases : “Deposit”
Input (Method)	int nation
Output (Method)	int
Abstract operation (Method)	출금 서비스에서 원(0)/달러(1) 중 어떤 것을 선택할 것인지 선택하고 nation 을 return 한다.
Exceptional Course of Events	

1.1.4. Confirm

Type	Method
Name	Confirm
Purpose	User 가 해당 Account 본인임을 인증한다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 Use cases : “Check” , “Deposit” , “Transfer”

Input	Int pwd
Output	boolean
Abstract operation	입력한 비밀번호가 맞으면 true 를 , 틀리면 false 를 return 한다.
Exceptional Course of Events	-

1.1.5. insertCash

Type	**Method**
Name	insertCash
Purpose	User 가 입금하려는 금액을 넣는다.
Cross Reference	System Function :, R.1.3 ,R.2.0 , R.2.1,R.2.2 Use cases :, "Withdraw" , "Update", "Status Alarm"
Input	money
Output	int
Abstract operation	지폐 code 배열을 받고 그에 알맞은 돈을 bank 에 입금해준다.
Exceptional Course of Events	ATM 기기안 현금이 정한 값 많으면 돈을 더 못 넣으니 return false 를 하게 된다.

1.1.6. enterAmount

Type	**Method**
Name	enterAmount
Purpose	거래할 금액을 입력한다
Cross Reference	System Function : R.1.1 , R.1.2, R.2.0 , R.2.1, R.2.2 Use cases : "Deposit" , "Transfer" , "Update" , "Verify Sufficient Fund" ,
Input	Int money
Output	int
Abstract operation	돈을 입력 받고 서비스(출금/송금)에 맞는 bank method 를 실행한다.
Exceptional Course of Events	출금의 경우, ATM 기기안 현금이 필요한 양보다 없으면 false 를

	리턴하게 된다.

1.1.7. getBalance

Type	Method
Name	getBalance
Purpose	잔액을 보여준다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3, R.2.1 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw”
Input	
Output	int
Abstract operation	지금 읽고있는 계좌의 잔액을 보여준다.
Exceptional Course of Events	

1.1.8. printReceipt

Type	Method
Name	printRecipt
Purpose	명세표를 출력한다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 ,R.3.0 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw” , “Status Alarm”
Input	
Output	boolean
Abstract operation	Parameter wants == true 이면 , balance 를 bank 로부터 받아온다.
Exceptional Course of Events	

1.1.9. setDataRange

Type	Method

Name	setDataRange
Purpose	교통카드 이용 날짜를 설정한다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.1.4 , R.2.0 , R.2.1, R.2.2, Use cases : “IssueTrafficCard”, “Update” , “Verify Sufficient Fund” , “Status Alarm”
Input	Int date_range
Output	boolean
Abstract operation	Traffic Card 를 가져와 유효기간(date_range)를 set 해준다.
Exceptional Course of Events	-

1.1.10. agreement

Type	Method
Name	Agreement
Purpose	교통 카드 발급 약관을 보여주고, 서명을 받는다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.1.4 , R.2.0 , R.2.1, R.2.2, Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw” , “IssueTrafficCard”, “Update” , “Verify Sufficient Fund” , “Status Alarm”
Input	
Output	boolean
Abstract operation	account 와 trafficCard 를 연동 시켜 주고 , chargeTrafficCard 를 한다.
Exceptional Course of Events	-

1.1.11. destAccount

Type	Method
Name	destAccount
Purpose	송금할 대상을 설정한다.
Cross Reference	System Function : R.1.2, R.2.0 , R.2.1, R.2.2 Use cases :“Transfer” , “Update” , “Verify Sufficient Fund” , “Status Alarm”
Input	String bankID , int accountID

Output	String
Abstract operation	input 값을 보고 그 계좌의 이름을 return 받는다.
Exceptional Course of Events	존재하지 않는 계좌일 경우 , null 값을 return 한다.

1.1.12.end

Type	**Method**
Name	end
Purpose	관리자가 작업을 끝내면 다시 파일을 읽어 atm 정보를 업데이트 한다.
Cross Reference	System Function : R.3.0 Use cases :"Management"
Input	
Output	Void
Abstract operation	관리자가 작업을 끝내면 다시 파일을 읽어 atm 정보를 업데이트 한다.
Exceptional Course of Events	-

1.1.13.checkResource

Type	**Method**
Name	checkResource
Purpose	ATM 내부 현금 / 교통카드 / 명세표용지 양을 체크한다.
Cross Reference	System Function : R.2.2 Use cases : "Status Alarm"
Input	
Output	
Abstract operation	현금(cashAmount) , 교통카드(trafficCardAmount) , 명세표 종이(receiptAmount)가 부족할 경우 , 관리자에게 알람을 보낸다.
Exceptional Course of Events	-

1.1.14.getATMadminID()

Type	Method
Name	getATMadminID
Purpose	관리자 ID 를 return 해준다.
Cross Reference	System Function : R.3.0 Use cases : “Management”
Input	
Output	int
Abstract operation	
Exceptional Course of Events	-

1.2. Bank

Type	Class
Name	Bank
Purpose	ATM 으로부터 User 가 거래하는데 필요한 정보를 제공, 업데이트한다.
Overview (class)	
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.1.4 , R.2.0 , R.2.1, R.2.2, Use cases : “Check” , “Deposit” , “IssueTrafficCard”, “Transfer” , “Withdraw” , “Update” , “Verify Sufficient Fund” , “Status Alarm”
Exceptional Course of Events	-

1.2.1. loadItem

Type	Method
Name	loadItem
Purpose	계좌에 접근하기 위한 기본 세팅을 한다 (파일 데이터 읽어오기)
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.1.4 , R.2.0 , R.2.1 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw” , “Update” , “Verify Sufficient Fund”
Input (Method)	
Output (Method)	

Abstract operation (Method)	
Exceptional Course of Events	-

1.2.2. WriteData

Type	Method
Name	WriteData
Purpose	계좌파일에 바뀐 정보를 쓴다 (파일 데이터 쓰기)
Cross Reference	System Function : R.1.1 , R.1.2 , R.1.3 , R.1.4 , R.2.0 , R.2.1 Use cases : "Deposit" , "Transfer" , "Withdraw", "Update", "IssueTrafficCard" , "Verify Sufficient Fund"
Input (Method)	
Output (Method)	
Abstract operation (Method)	
Exceptional Course of Events	-

1.2.3. vaildCheck

Type	Method
Name	vaildCheck
Purpose	유효한 계좌 / Item 인지 확인한다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.1.4 Use cases : "Check" , "Deposit" , "Transfer" , "Withdraw" , "IssueTrafficCard"
Input (Method)	int _itemType, int _itemID, int _accountID
Output (Method)	boolean
Abstract operation (Method)	_itemType 인자를 통해 카드인지 통장인지 구분한다.
Exceptional Course of Events	유효한 계좌가 없다면 false 를 반환한다.

1.2.2. Confirm

Type	Method
Name	Confirm
Purpose	입력한 비밀번호가 맞는지 확인한다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.4 Use cases : “Check” , “Deposit” , “Transfer”, “IssueTrafficCard” ,
Input (Method)	Int _pwd
Output (Method)	boolean
Abstract operation (Method)	
Exceptional Course of Events	불러온 계좌와 해당 비밀번호가 일치하지 않으면 false 를 반환한다.

1.2.3. getBalance

Type	Method
Name	getBalance
Purpose	해당 계좌의 잔고를 불러온다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.1.4 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw” , “Update” , “IssueTrafficCard”
Input (Method)	
Output (Method)	Int //현재 계좌 잔고를 반환
Abstract operation (Method)	
Exceptional Course of Events	-

1.2.4. checkAccount

Type	Method
Name	checkAccount
Purpose	송금할 대상이 유효한지 확인한 후 대상 계좌 주인의 이름을 반환한다.

Cross Reference	System Function : R.1.3 Use cases : “Transfer”
Input (Method)	String _bankID, int _accountID
Output (Method)	String
Abstract operation (Method)	
Exceptional Course of Events	송금 대상이 유효하지 않으면 null 을 반환한다.

1.2.5. linkAccount

Type	Method
Name	linkAccount
Purpose	교통카드와 계좌를 연동시킨다.
Cross Reference	System Function : R.1.4 Use cases : “IssueTrafficCard”
Input (Method)	Int _tcid
Output (Method)	boolean
Abstract operation (Method)	
Exceptional Course of Events	-현재 계좌가 유효하지 않으면 false 를 반환한다.

1.2.6. transfer

Type	Method
Name	transfer
Purpose	송금한다.
Cross Reference	System Function : R.1.3 Use cases : “Transfer”
Input (Method)	Int _money
Output (Method)	boolean
Abstract operation (Method)	같은 은행이라면 파일에 데이터를 두 번 덮어쓰지 않도록 한 객체 데이터를 변경한다.

Exceptional Course of Events	내 계좌의 잔액이 부족하거나 유효하지 않으면 false 를 반환한다.

1.2.7. withdraw

Type	Method
Name	withdraw
Purpose	출금한다.
Cross Reference	System Function : R.1.2 0 Use cases : “Deposit”
Input (Method)	Int money
Output (Method)	boolean
Abstract operation (Method)	내 계좌의 잔고가 충분하면 빠지는 금액만큼 차감하여 파일에 저장한다.
Exceptional Course of Events	잔고가 충분하지 않으면 false 를 반환한다.

1.2.8. deposit

Type	Method
Name	deposit
Purpose	입력된 금액만큼 계좌 잔고를 증가 시킨다.
Cross Reference	System Function : R.1.1 Use cases : “Deposit”
Input (Method)	Int money
Output (Method)	boolean
Abstract operation (Method)	
Exceptional Course of Events	계좌가 유효하지 않으면 false 를 반환한다.

1.3. Account

Type	Class
Name	Account
Purpose	User 가 해당 서비스를 이용하기 위해서 등록한 계좌이다.

Overview (class)	
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.1.4 , R.2.0 , R.2.1 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw” , “IssueTrafficCard”, “Update” , “Verify Sufficient Fund”
Exceptional Course of Events	-

1.3.1. getItemID

Type	Method
Name	getItemID
Purpose	카드나 통장의 ID 를 가져온다
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.1.4 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw” , “IssueTrafficCard”
Input (Method)	Int type
Output (Method)	int
Abstract operation (Method)	
Exceptional Course of Events	-

1.3.2. get_aid

Type	Method
Name	get_aid
Purpose	Account 의 id 를 return 한다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.1.4 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw” , “IssueTrafficCard”
Input (Method)	
Output (Method)	int

Abstract operation (Method)	
Exceptional Course of Events	-

1.3.3. getPwd

Type	Method
Name	getPwd
Purpose	해당 계좌의 비밀번호를 가져온다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.4 Use cases : “Check” , “Deposit” , “Transfer” , “IssueTrafficCard”,
Input (Method)	
Output (Method)	Int[]
Abstract operation (Method)	
Exceptional Course of Events	-

1.3.3. getBalance

Type	Method
Name	Get_Balance
Purpose	해당 계좌의 잔고를 가져온다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.1.4 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw” , “IssueTrafficCard”
Input (Method)	
Output (Method)	int
Abstract operation (Method)	
Exceptional Course of Events	-

1.3.4. set_balance

Type	Method
Name	set_balance
Purpose	해당 계좌의 잔고를 증감한다.
Cross Reference	System Function : R.1.1 , R.1.2 , R.1.3 , R.1.4 , R.2.0 , R.2.1 Use cases : “Deposit” , “Transfer” , “Withdraw” , “IssueTrafficCard”, “Update” , “Verify Sufficient Fund”
Input (Method)	
Output (Method)	
Abstract operation (Method)	
Exceptional Course of Events	-

1.3.5. addLink

Type	Method
Name	addLink
Purpose	해당 계좌에 교통카드 정보를 입력한다. ->계좌에 교통카드를 연동하여 교통카드를 사용할 경우 , 계좌 안의 잔액으로 교통비가 빠져나가게 하기 위한 목적이다.
Cross Reference	System Function : R.1.4 Use cases :“IssueTrafficCard”
Input (Method)	
Output (Method)	boolean
Abstract operation (Method)	
Exceptional Course of Events	-

1.3.6. get_name

Type	Method
Name	get_name
Purpose	해당 계좌의 이름을 가져온다.

Cross Reference	System Function : R.1.2 Use cases : “Transfer”
Input (Method)	
Output (Method)	String
Abstract operation (Method)	
Exceptional Course of Events	-

1.4. Card

Type	Class
Name	Card
Purpose	User 가 해당 서비스를 이용하기 위해 소지하고 있어야하는 Item 이다.
Overview (class)	
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.2.0 , R.2.1 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw” , “Update” , “Verify Sufficient Fund”
Exceptional Course of Events	-

1.4.1. getCid

Type	Method
Name	getCid
Purpose	해당 카드번호를 가져온다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw”
Input (Method)	
Output (Method)	int
Abstract operation (Method)	

Exceptional Course of Events	-

1.4.2. getCpwd

Type	Method
Name	getCpwd
Purpose	해당 카드비밀번호를 가져온다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 Use cases : “Check” , “Deposit” , “Transfer”
Input (Method)	
Output (Method)	int
Abstract operation (Method)	
Exceptional Course of Events	-

1.5. Book

Type	Class
Name	Book
Purpose	User 가 해당 서비스를 이용하기 위해 소지하고 있어야하는 Item 이다.
Overview (class)	
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.2.0 , R.2.1 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw” , “Update” , “Verify Sufficient Fund”
Exceptional Course of Events	-

1.5.1. getBid

Type	Method
Name	getBpwd
Purpose	해당 통장번호를 가져온다.

Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw””
Input (Method)	
Output (Method)	int
Abstract operation (Method)	
Exceptional Course of Events	-

1.5.2. getBpwd

Type	Method
Name	getBpwd
Purpose	해당 통장 비밀번호를 가져온다.
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 Use cases : “Check” , “Deposit” , “Transfer”
Input (Method)	
Output (Method)	int
Abstract operation (Method)	
Exceptional Course of Events	-

1.6. TrafficCard

Type	Class
Name	TrafficCard
Purpose	User 가 교통카드발급을 할 경우, 해당 계좌와 연동된다.
Overview (class)	
Cross Reference	System Function : R.1.4 Use cases : “IssueTrafficCard”
Exceptional Course of Events	-

1.6.1. getTcid

Type	Method
Name	get_tcid
Purpose	해당 교통 카드 번호를 가져온다.
Cross Reference	System Function : R.1.4 Use cases : “IssueTrafficCard”
Input (Method)	
Output (Method)	int
Abstract operation (Method)	
Exceptional Course of Events	-

1.6.2. setDateRange

Type	Method
Name	setDateRange
Purpose	해당 교통 카드의 이용 기간을 설정한다.
Cross Reference	System Function : R.1.4 Use cases : “IssueTrafficCard”
Input (Method)	
Output (Method)	
Abstract operation (Method)	
Exceptional Course of Events	-

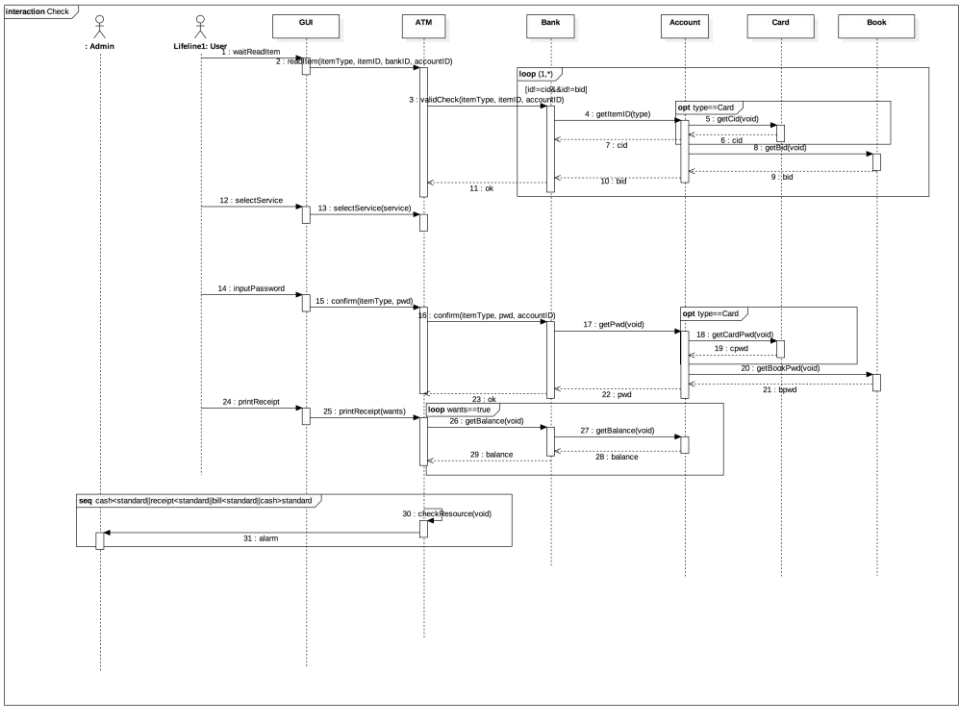
1.6.3. setAccountID

Type	Method
Name	setAccountID
Purpose	해당 교통카드를 계좌와 연동한다.

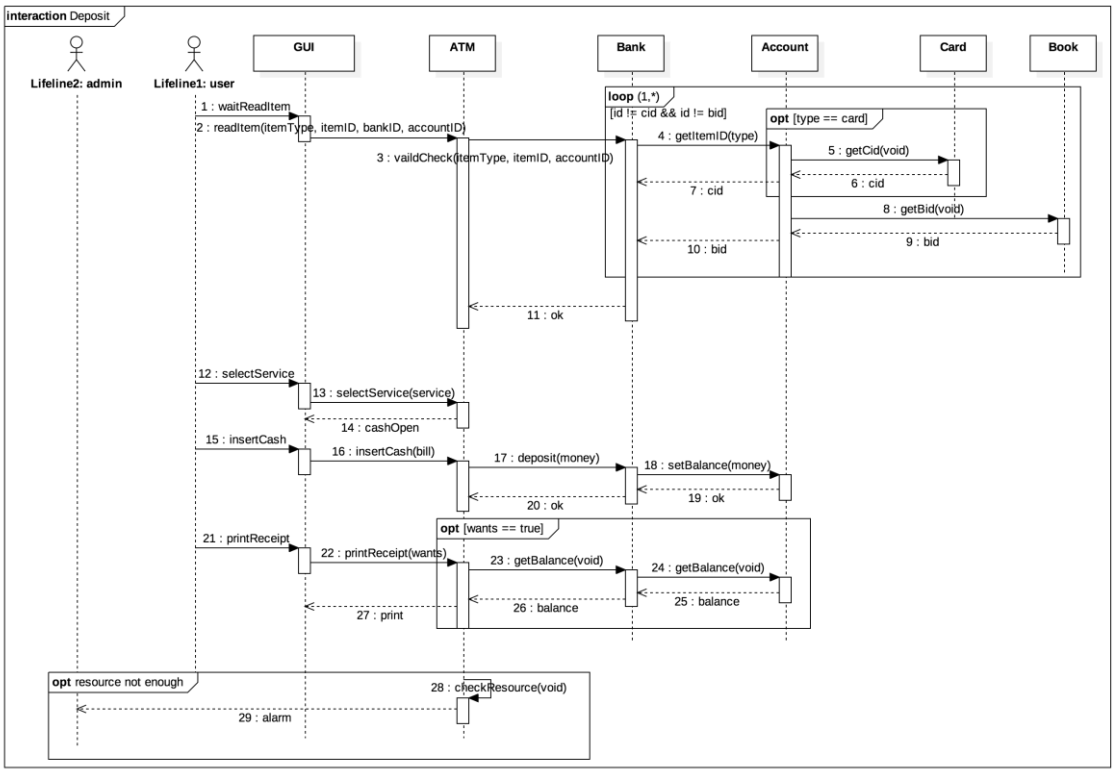
Cross Reference	System Function : R.1.4 Use cases : “IssueTrafficCard”
Input (Method)	
Output (Method)	
Abstract operation (Method)	
Exceptional Course of Events	-

2. Activity 2052. Implements Windows

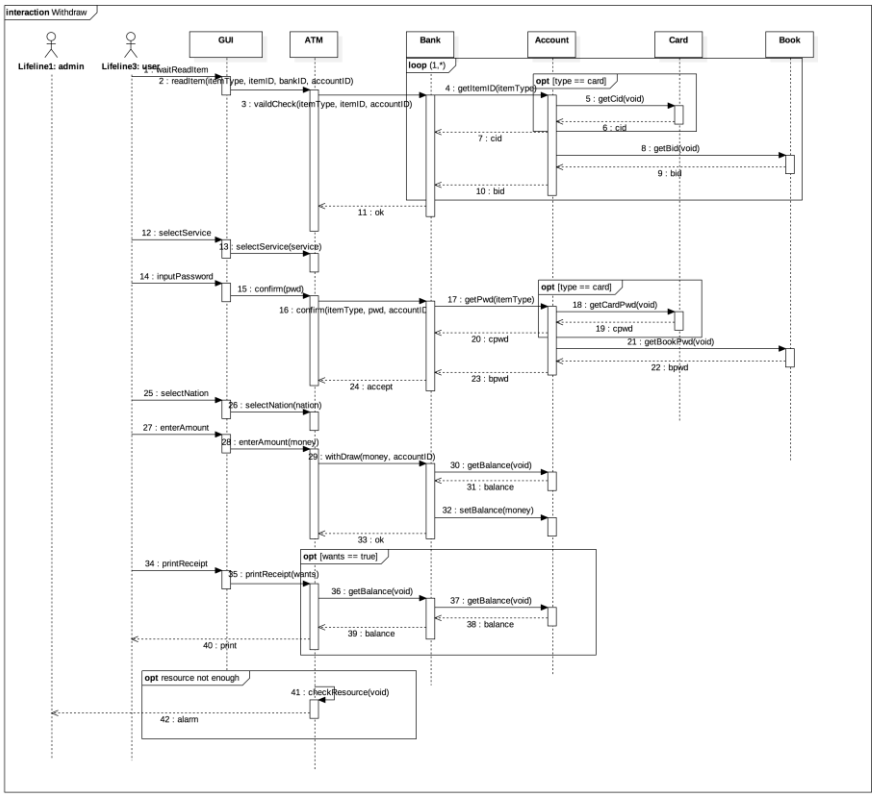
- Check



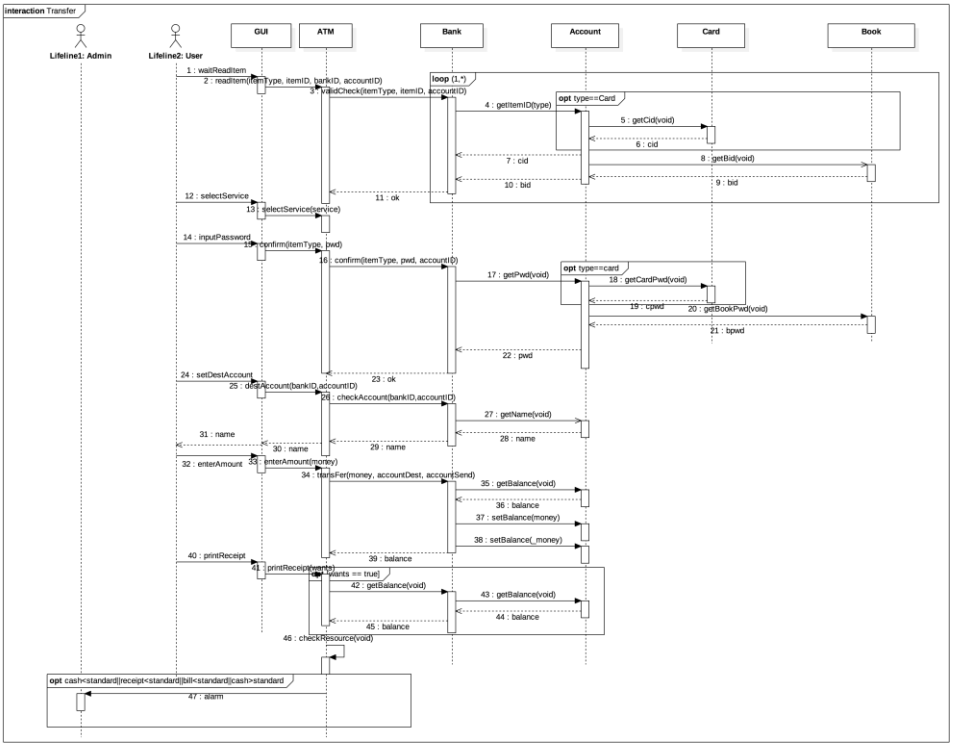
- Deposit



- Withdraw

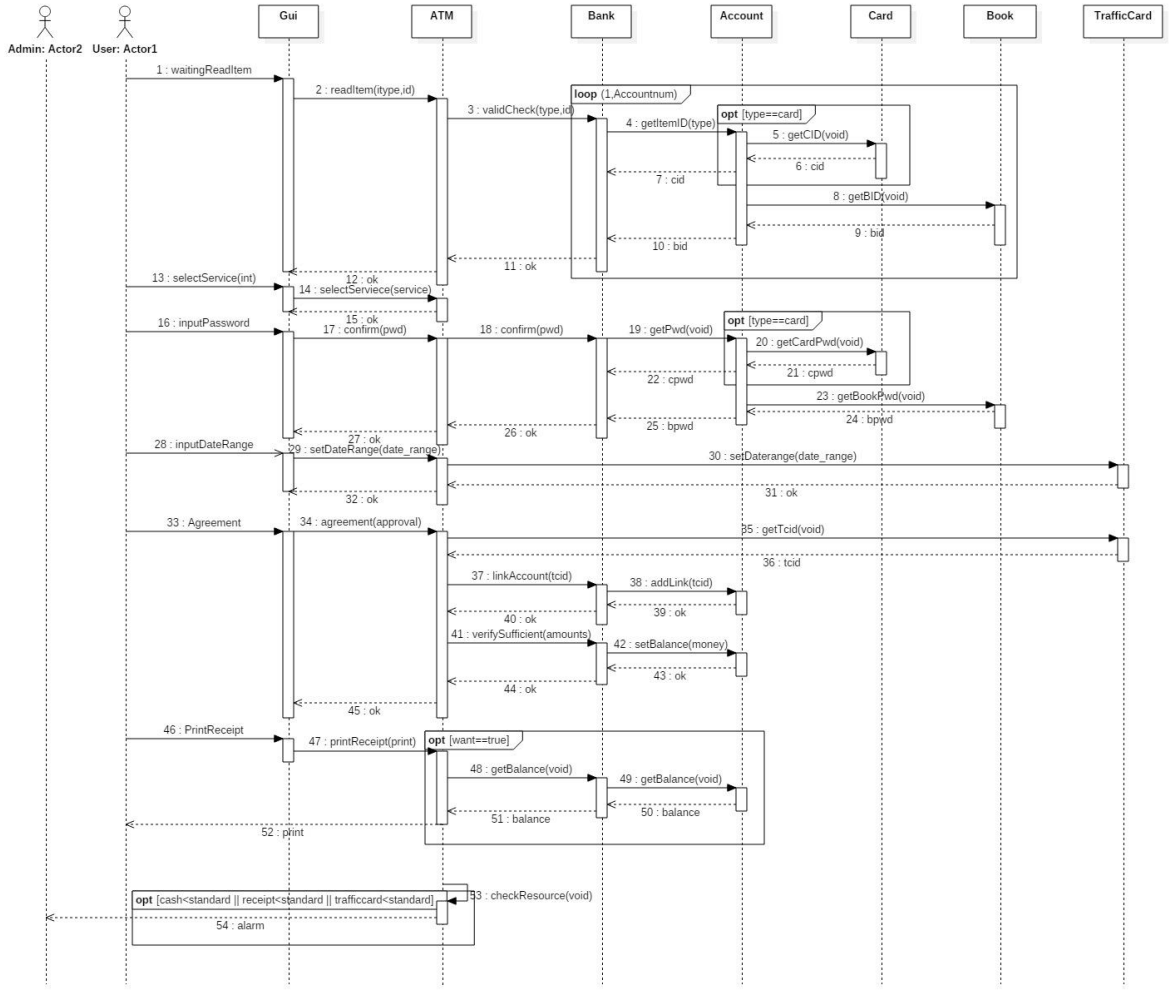


- Transfer



-

IssueTrafficCard



2.1. waitReadItem

Name	waitReadItem
Resposiblities	User 가 Item 을 인식시키는 것을 기다린다.
Type	GUI
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.1.4 , R.2.0 , R.2.1, R.2.2, R.3.0 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw” , “Management” , “Update” , “Verify Sufficient Fund” , “Status Alarm”
Notes	-
PreConditions	ATM 실행 상태
PostConditions	User 의 Item 값 입력

2.2. selectService

Name	selectService
Responsibilities	User 가 원하는 Service 를 선택한다.
Type	GUI
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.1.4 , R.2.0 , R.2.1, R.2.2, R.3.0 Use cases : “Check” , “Deposit” , “Transfer” , “Withdraw” , “Management” , “Update” , “Verify Sufficient Fund” , “Status Alarm”
Notes	-
Pre-Conditions	User 의 유효한 Item 인식
Post-Conditions	User 가 원하는 서비스 선택

2.3. inputPassword

Name	inputPassword
Responsibilities	User 가 비밀번호를 입력한다.
Type	GUI
Cross Reference	System Function : R.1.0 , R.1.2 , R.1.3 , R.1.4 , R.2.0 , R.2.1, R.2.2, R.3.0 Use cases : “Check” , “Transfer” , “Withdraw” , “Management” , “Update” , “Verify Sufficient Fund” , “Status Alarm”
Notes	
Pre-Conditions	서비스 선택
Post-Conditions	비밀번호 입력

2.4. insertCash

Name	insertCash
Responsibilities	입금하고자하는 금액을 입력한다.
Type	GUI
Cross Reference	System Function : R.1.1 , R.2.1, R.3.0 Use cases : “Check” , , “Update” , “Status Alarm”
Notes	

Pre-Conditions	입금 서비스 선택
Post-Conditions	금액 입력

2.5. selectNation

Name	**selectNation**
Resposiblities	출금하고자하는 지폐 종류를 선택한다.
Type	GUI
Cross Reference	System Function : R.1.3 Use cases : “Withdraw”
Notes	-
Pre-Conditions	출금 서비스 선택
Post-Conditions	원 / 달러 선택

2.6. enterAmount

Name	enter**Amount**
Resposiblities	거래하고자 하는 금액 입력
Type	GUI
Cross Reference	System Function : R.1.2 , R.1.3 Use cases : “Transfer” , “Withdraw”
Notes	-
Pre-Conditions	출금 / 송금 서비스 선택
Post-Conditions	거래 금액 입력

2.7. setDestAccount

Name	**inputTransfer**
Resposiblities	송금할 대상 계좌 정보 입력
Type	GUI

Cross Reference	System Function : R.1.2 Use cases : "Transfer"
Notes	-
Pre-Conditions	송금 서비스 선택
Post-Conditions	송금 대상 계좌 정보 입력

2.8. printReceipt

Name	**printReceipt**
Resposiblities	거래 내역 확인 및 명세표 출력
Type	GUI
Cross Reference	System Function : R.1.0 , R.1.1 , R.1.2 , R.1.3 , R.1.4 , R.2.0 , R.3.0 Use cases : "Check" , "Deposit" , "Transfer" , "Withdraw" , "Management" , "Status Alarm"
Notes	-
Pre-Conditions	거래 (서비스) 진행 완료
Post-Conditions	거래 정보 출력 및 명세표 출력 여부 확인

2.9. inputRangeDate

Name	**inputRangeDate**
Resposiblities	교통카드 이용 날짜 범위 입력
Type	GUI
Cross Reference	System Function : R.1.5 Use cases : "IssueTrafficCard"
Notes	-
Pre-Conditions	교통카드 발급 서비스 선택
Post-Conditions	교통카드 이용 날짜 범위 입력

2.10. Agreement

Name	**agreement**

Resposiblities	교통카드 발급 관련 약관 확인
Type	GUI
Cross Reference	System Function : R.1.5 Use cases : “IssueTrafficCard”
Notes	-
Pre-Conditions	교통카드 발급 서비스 선택
Post-Conditions	교통 카드 발급 약관 확인

3. Activity 2055. Write Unit Test Code

3.1. ATM

```
package ATM;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class ATMTest {

    public static ATM atm = new ATM();

    @org.junit.jupiter.api.Test
    void readItem() {
        assertEquals( expected: true, atm.readItem( itemType: 1,  itemID: 40000, bankID: "shinhan" ,  accountID: 1000));
    }

    @org.junit.jupiter.api.Test
    void selectNation() {
        assertEquals( expected: 1,atm.selectNation(1));
    }

    @org.junit.jupiter.api.Test
    void confirm() {
        assertEquals( expected: true,atm.confirm( pwd: 2230));
    }
```

```java
@org.junit.jupiter.api.Test
void insertCash() {
    String[] bill = new String[7];
    bill[0] = "0000";
    bill[1] = "0001";
    bill[2] = "0010";
    bill[3] = "0011";
    bill[4] = "0100";
    bill[5] = "0101";
    bill[6] = "0110";
    assertEquals( expected: true,atm.insertCash(bill));
}

@org.junit.jupiter.api.Test
void enterAmount() {
    atm.selectService(3);
    assertEquals( expected: true,atm.enterAmount( money: 10000));
}

@org.junit.jupiter.api.Test
void printReceipt() { assertEquals( expected: 0,atm.printReceipt( wants: true)); }

@org.junit.jupiter.api.Test
void destAccount() {
    assertEquals( expected: "park",atm.destAccount( bankID: "kb", accountID: 1000));
```

```java
    @Test
    void selectService() {
        atm.selectService(1);

    }

    @Test
    void setDataRange() { atm.setDataRange(10); }

    @Test
    void readManagementItem() { atm.readManagementItem( adminID: 123); }

    @Test
    void end() { atm.end(); }


}
```

3.2. Bank

```java
package BankSystem;

import ...

class BankTest {

    //make test Bank
    Bank bTest = new Bank( _bankName: "shinhan");

    @Test
    void validCheck() {
        //itemtype : 0 (book) , itemID : 7000 accountID : 1000
        assertEquals( expected: true,bTest.validCheck( _itemType: 0, _itemID: 7000, _accountID: 1000));
    }

    @Test
    void confirm() {
        //bTest.confirm is true
        //before the item should be checked
        this.validCheck();
        assertTrue(bTest.confirm( _pwd: 1230));
    }
```

```java
    @Test
    void linkAccount() {
        //link Account
        //before the item should be checked
        this.validCheck();
        bTest.linkAccount( _tcid: 9000);
    }

    @Test
    void getBalance() {
        //before the item should be checked
        this.validCheck();
        assertEquals( expected: 44000,bTest.getBalance());
    }

    @Test
    void chargeTrafficCard() {
        //it can chargeTrafficCard balance>3000
        //before the item should be checked
        this.validCheck();
        assertTrue(bTest.chargeTrafficCard( _money: 3000));
    }
```

```
    @Test
    void withdraw() {
        //before the item should be checked
        this.validCheck();
        //it can withdraw 1000won -> balance = 28000
        assertTrue(bTest.withdraw( _money: 10000));
    }

    @Test
    void deposit() {
        //before the item should be checked
        this.validCheck();
        //insert money 10000 -> balance =38000
        assertTrue(bTest.deposit( _money: 10000));
    }

    @Test
    void checkAccount() {
        //check kb account 1001 name : choi
        assertEquals( expected: "choi",bTest.checkAccount( _bankID: "kb", _accountID: 1001));
    }
```

```
    @Test
    void transfer() {
        //before the item should be checked and destAccount also checked
        this.validCheck();
        this.checkAccount();
        //same because destaccount ==using account
        assertTrue(bTest.transfer( _money: 10000));
    }
}
```

3.3. Account

```java
package BankSystem;

import ...

class AccountTest {
    //test -make Account
    Account accountTest = new Account( _bankID: "shinhan",  _name: "park",  _aid: 1000,  _bal: 36000,  _tcid: 0);

    @Test
    void get_name() {
        //test name : park
        assertEquals( expected: "park",accountTest.get_name());
    }

    @Test
    void get_aid() {
        //test get aid
        assertEquals( expected: 1000,accountTest.get_aid());
    }

    @Test
    void get_balance() {
        //test get_balance
        assertEquals( expected: 36000,accountTest.get_balance());
    }
```

```java
    @Test
    void get_tcid() {
        //test get_tcid
        assertEquals( expected: 0,accountTest.get_tcid());
    }

    @Test
    void set_balance() {
        accountTest.set_balance(10000);
    }

    @Test
    void getItemID() {
        //park -aid = 1000 -> card id = 40000 , 41000
        int cardID[] = new int[2];
        cardID[0] = 40000;
        cardID[1] = 41000;
        assertArrayEquals(cardID,accountTest.getItemID( type: 1));
        //park -aid = 1000 -> book id = 7000
        int bookID[] = new int[1];
        bookID[0] = 7000;
        assertArrayEquals(bookID,accountTest.getItemID( type: 0));
    }
```

```
    @Test
    void getPwd() {
        //get book pwd = 1230
        int bookPwd[] = new int[1];
        bookPwd[0] = 1230;
        assertArrayEquals(bookPwd,accountTest.getPwd( type: 0));

        //get card pwd : 2230,3230
        int cardPwd[] = new int[2];
        cardPwd[0] = 2230;
        cardPwd[1] = 3230;
        assertArrayEquals(cardPwd,accountTest.getPwd( type: 1));
    }

    @Test
    void addLink() {
        //tcid 9000 -addLink
        assertEquals( expected: true,accountTest.addLink( _tcid: 9000));
    }
}
```

3.4. Card

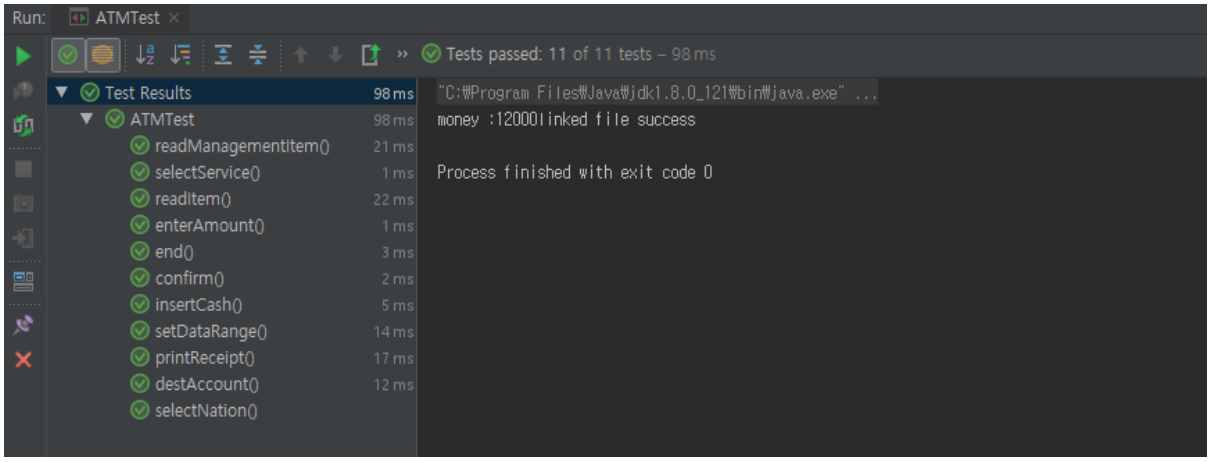
```
package Item;

import ...

class CardTest {

    //Card class test를 위한 class
    //shinhan Card , accountID 1000 Test book id : 4000 book pwd :  2230 cardindex : 0
    Card cTest = new Card( bank: "shinhan", aid: 1000, index: 0);


    @Test
    void getCid() {
        //run Card class getCid() -> return card id -> expect 4000
        assertEquals( expected: 40000,cTest.getCid());
    }

    @Test
    void getCpwd() {
        //run Card class getCid() -> return card pwd -> expect 2230
        assertEquals( expected: 2230,cTest.getCpwd());
    }
}
```

3.5. Book

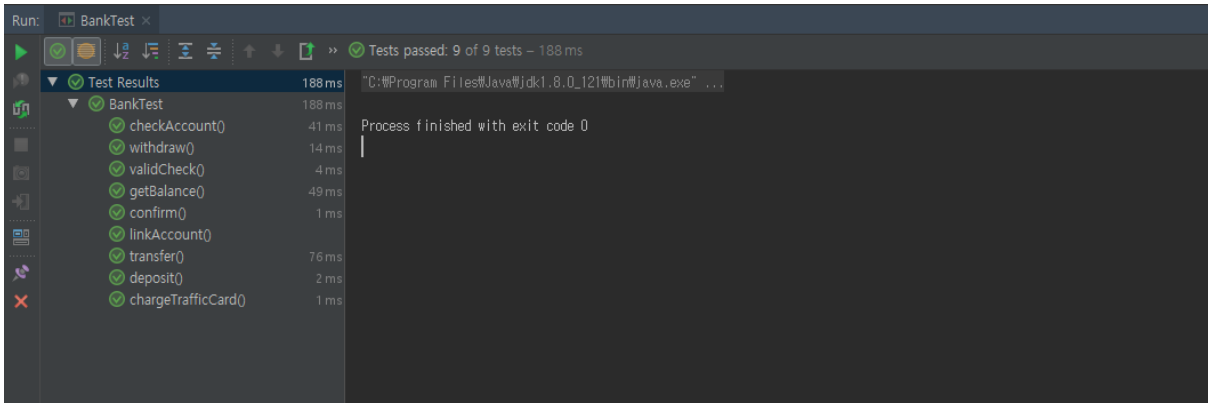
```java
package Item;

import ...

class BookTest {

    //book class test를 위한 class
    //shinhan book , accountID 1010 Test book id : 7000 book pwd :  1230
    Book bTest = new Book( bank: "shinhan", aid: 1000);

    @Test
    void getBid() {
        //run Book class getBid() -> return book ID -> expect 7000
        assertEquals( expected: 7000,bTest.getBid());
    }

    @Test
    void getBpwd() {
        //run Book class getBid() -> return book pwd -> expect 1230
        assertEquals( expected: 1230 , bTest.getBpwd());
    }
}
```

4. Activity 2061. Unit Testing

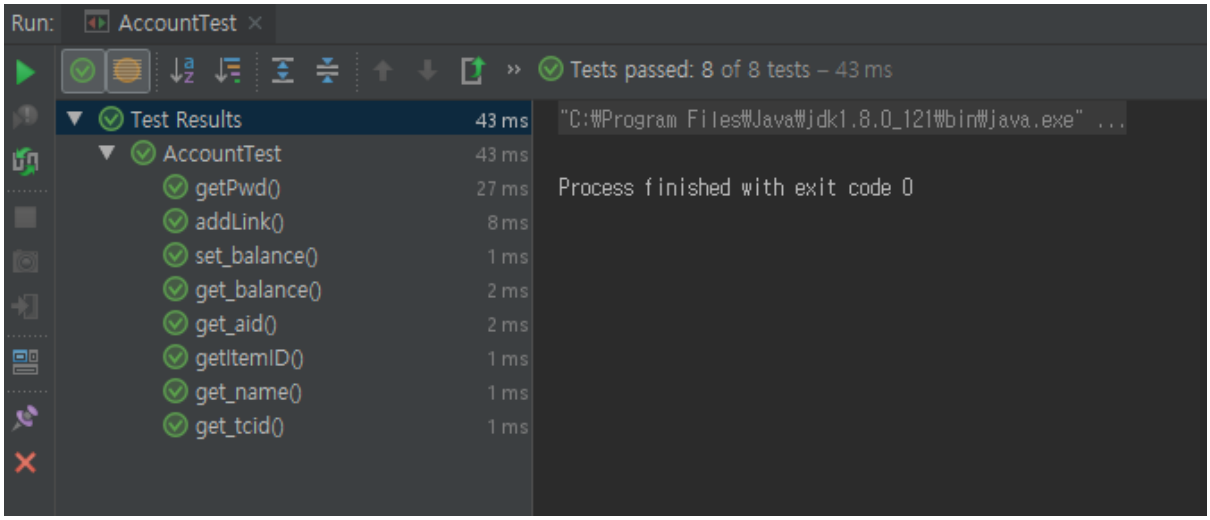
4.1. ATM



4.2. Bank

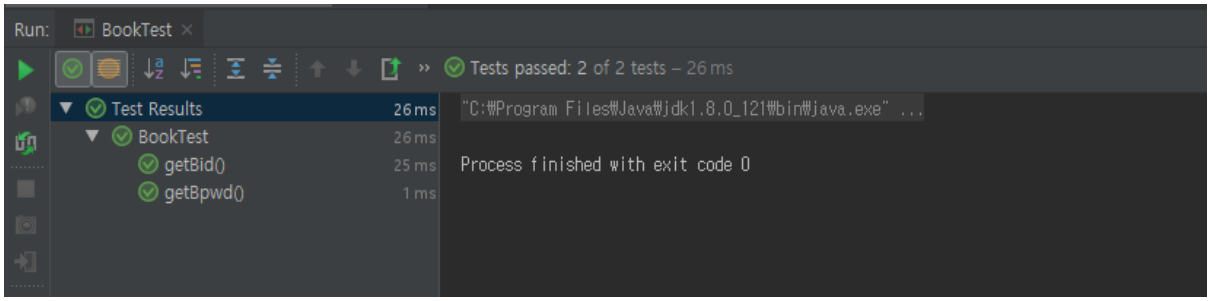


4.3. Account



4.4. Card

4.5. Book



5. Activity 2063. System Testing

Test Number	Test 항목	Description	Use Case	System Function	P/F
1-1	Read Item Test	사용가능한 한국계좌 Item 정보를 입력 했을 때 서비스 선택창으로 전환되는지 확인	-		T

1-2	Read Item Test	사용가능한 해외계좌 Item 정보를 입력 했을 때 서비스 선택창으로 전환되는지 확인	-		T
1-3	Read Item Test	사용 불가능한 Item 정보를 입력 했을 때 잘못된 item 이라고 다른 item 을 넣어주라는 창으로 바뀌는 지 확인	-		T
2-1	서비스 선택 test	계좌조회 서비스 선택했을 때 , 비밀번호 입력 창으로 전환되는지 확인	Check	R.1.0	T
2-2	서비스 선택 test	입금 서비스 선택했을 때 , 금액 입력 안내와 금액을 투입하라는 메시지 창으로 전환되는지 확인	Deposit	R.1.2	T
2-3	서비스 선택 test	출금 서비스 선택했을 때 , 비밀번호 입력 창으로 전환되는지 확인	Withdraw	R.1.1	T
2-4	서비스 선택 test	송금 서비스 선택했을 때 , 비밀번호 입력 창으로 전환되는지 확인	Transfer	R.1.2	T
2-5	서비스 선택 test	교통카드 발급 서비스를 선택 했을 때, 비밀번호 입력창으로 전환되는지 확인	Issue TrafficCard	R.1.3	T
2-6	서비스 선택 test	취소를 선택 했을 때, 카드를 반환하고 , 초기화면으로 돌아갈수 있는 창으로 변환			T
3-1	비밀번호 test	계좌조회 서비스를 선택한 경우 누른 다음 올바른 비밀번호를 입력했을 때 , 명세 결과창으로 전환되는지 확인	Check	R.1.0	T
3-2	비밀번호 test	출금 서비스를 선택한 경우 누른 다음 올바른 비밀번호를 입력했을 때 , 출금 지폐종류 확인 창으로 전환되는지 확인	Deposit	R.1.2	T
3-3	비밀번호 test	송금 서비스를 선택한 경우 누른 다음 올바른 비밀번호를 입력했을 때 , 송금 계좌 입력창으로 전환되는지 확인	Transfer	R.1.3	T
3-4	비밀번호 test	교통카드 발급 서비스를 선택한 경우 누른 다음 올바른 비밀번호를 입력했을 때 , 교통카드 사용기간 입력창으로 전환되는지 확인	Issue TrafficCard	R.1.4	T
3-5	비밀번호 test	잘못된 비밀번호를 입력한 경우, item 을 반환하고 ,초기화면으로 돌아갈 수 있는 창으로 변환	-		T

3-6	비밀번호 test	취소를 선택한 경우, 카드를 반환하고 초기화면으로 돌아갈 수 있는 창으로 변환			T
4-1	명세표 출력 test	거래결과가 화면에 출력되고 명세표 출력을 원한다고 선택하면, 명세표와 카드가 반환되고 초기화면으로 전환되는지 확인	-		T
4-2	명세표 출력 test	명세표 출력을 원하지 않는다고 선택하면, item 이 반환되고 초기화면으로 전환되는지 확인	-		T
5-1	입금 test	지폐를 투입 했을 때, 명세 결과 창으로 전환되는지 test	Deposit	R.1.1	T
5-2	입금 test	취소를 선택한 경우, 카드를 반환하고 초기화면으로 돌아갈 수 있는 창으로 변환			T
6-1	금액 입력 test	출금 서비스 선택의 경우 계좌잔고가 충분할때, 입력한 금액 만큼 출금되는지 확인하고 명세 결과 창으로 전환되는지 test	Withdraw	R.1.2	T
6-2	금액 입력 test	송금 서비스 선택의 경우 계좌잔고가 충분할때, 입력한 금액 만큼 송금되는지 확인하고 명세 결과 창으로 전환되는지 test	Transfer	R.1.3	T
6-3	금액 입력 test	송금 서비스 선택의 경우 계좌잔고가 부족할때, 카드를 반환하고 초기화면으로 돌아갈 수 있는 창으로 변환			T
7-1	송금 test	송금 서비스 선택에서 올바른 송금 대상의 은행과 계좌를 입력했을 때 , 송금 대상의 이름을 보여주고 금액량 입력창으로 전환되는지 test	Transfer	R.1.3	T
7-2	송금 test	송금 서비스 선택에서 잘못된 송금 대상의 은행과 계좌를 입력했을 때 , 경고창이 뜨고 초기화면으로 돌아갈 수 있는 창으로 변환	Transfer	R.1.3	T
7-3	송금 test	취소를 선택한 경우, item 를 반환하고 초기화면으로 돌아갈 수 있는 창으로 변환			T

8-1	교통카드 발급 test	교통카드 사용기간을 입력 했을 때 , 계좌 연동 및 카드 비용결제 승인 창으로 전환 되는지 test	Issue TrafficCard	R.1.4	T
8-2	교통카드 발급 test	계좌 연동 및 교통카드 비용 결제 승인 했을 때 , 계좌에 교통카드 비용 보다 많은 금액이 들어있는 경우에만 , 교통카드 발급이 올바르게 이루어 지는지 확인하고 , 명세결과 창으로 전환되는지 test	Issue TrafficCard	R.1.4	T
8-3	교통카드 발급 test	계좌 연동 및 교통카드 비용 결제 승인 했을 때 , 계좌에 교통카드 비용 보다 적은 금액이 들어있는 경우 , 초기화면으로 돌아갈 수 있는 창으로 변환	Issue TrafficCard	R.1.4	T
8-4	교통카드 발급 test	취소를 선택한 경우, 카드를 반환하고 초기화면으로 돌아갈 수 있는 창으로 변환	Issue TrafficCard	R.1.4	T
9-1	Management test	Management 창에서 올바른 adminID 를 입력한 경우 , atm 안 amount 를 수정할 수 있는 창으로 바뀐다.	Manageme nt	R.3.0	T
9-2	Management test	Management 창에서 잘못된 adminID 를 입력한 경우 , wrongid 라고 뜬다.	Manageme nt	R.3.0	T