Software Verification 2nd System Test

2nd Testing

- System Test

Project Team

T6

201311265 김상원

201214150 정성철

201210908 김성일

Date

2018-06-01

목차

1. Specification Review

- 1차 시스템 테스트 결과와 개정된 stage 1000, 2030, 2040, 2050 문서를 심층 분석하였음.

1.1 Stage 1000 문서 분석

- 시스템에 대한 컴퓨터 상에서의 시뮬레이션이므로 모든 입력은 키보드와 마우스를 통해 이루어진다.
- ATM에 들어있는 현금은 무한대라고 가정한다.

➔ 키보드 입력에 대한 구현이 완료 되었고 ATM에 들어있는 현금에 대한 가정이 추가되었다.

-Functional Requirements 관련

R6.1	Check Validation	Hidden

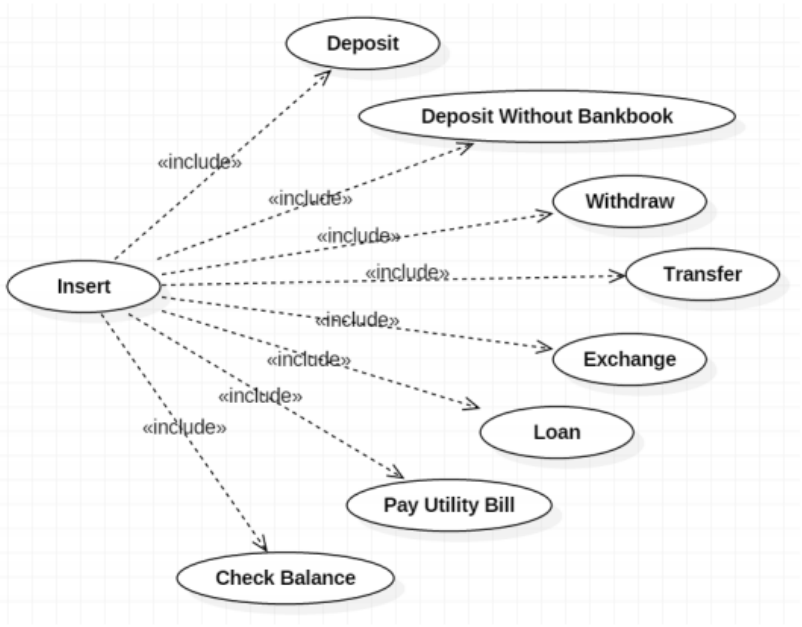
➔print error와 Check error 가 지워지고 새로운 Check Validation이 추가되었다.

-

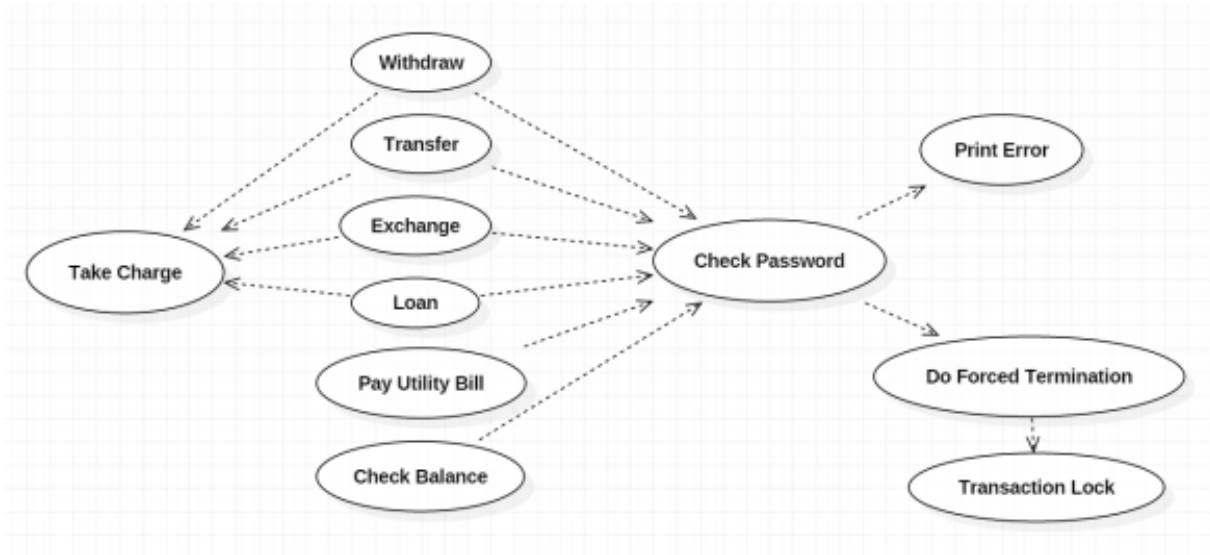
Use Cases by Event

Print Transaction Receipt	Print Error	Do Forced Termination
Take Charge	Check Password	Transaction Lock

➔ Use case by Event 에서 중복되던 case 가 지워졌다.



➔ Insert의 포함 관계에 대해 상세하게 명시하여 수정하였다.



➔ exchange의 check Password에 대한 화살표가 추가되었다.

Use Case	3. Withdraw
Actors	Customer
Description	-Customer can withdraw cash from account by using check card or bankbook. -Customer can only withdraw money in unit of 10000₩ and 50000₩ under balance. -Customer needs to choose the number of 50000₩. -Customer needs to know password of an account. -If customer successfully withdraws, ATM requests Offer to update database.(Hidden)

➔ 시스템 보고서 v1 내용이 반영되지 않았다.

1.2 Stage 2030 문서 분석

Activity 2031. Define Essential Use Cases

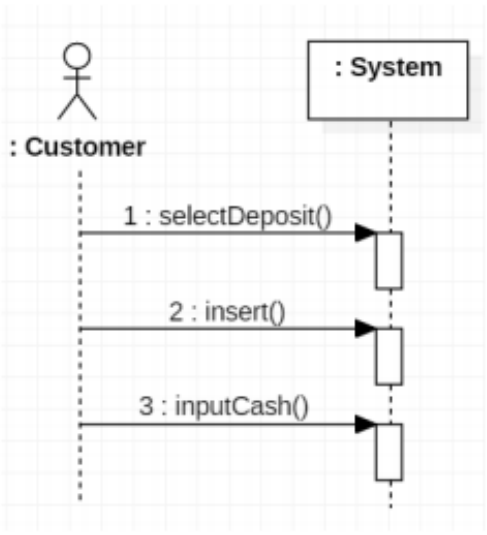
Use Case	1. Deposit
Actor	Customer
Purpose	Deposit cash into account by bankbook or check card, or credit card
Overview	Customer inputs card or bankbook, and cash, to deposit cash into account or credit card.
Type	Primary and Essential
Cross Reference	Functional Requirements : R1.1, R2.1, R2.2, R2.3, R2.4, R6.1, R6.2 Use case : Insert, Print Transaction Receipt, Print Error, Do Forced Termination, Check Validation, Update Database
Pre-Requisites	(N/A)

➔ 시스템 보고서 v1 내용이 반영되지 않았다.

Use Case	12. Do Forced Termination
Actor	(None)
Purpose	Immediately end transaction when error occurs 3 times
Overview	System ends transaction automatically and immediately when error occurs 3 times.
Type	Primary and Essential
Cross Reference	Functional Requirements : R1.1. R1.2, R1.3, R1.4, R1.5, R1.6, R1.7, R1.8, R2.1, R2.3, R2.4, R5.1, R6.2 Use Case : Deposit, Deposit Without Bankbook, Withdraw, Transfer, Exchange, Loan, Pay Utility Bill, Check Balance, Insert, Print Error, Transaction Lock, Update Database
Pre-Requisites	Use case "Print Error" occurred 3 times

➔ 3회 이상 오류에서 3회발생시로 Use case에 대한 명세가 보다 명확해졌다.

Activity 2035. Define System Sequence Diagrams

Use Case : Deposit	
1. Customer selects deposit menu from the basic screen. 2. System prints out "Insert check card or bankbook or credit card". 3. Customer inserts credit card, check card or bankbook("Insert"). 4. Invoke "Check Validation". If inserted method is valid, ask customer to input cash. 5. Customer Inputs cash in unit of 10000₩ and 50000₩. 6. System checks amount of cash inputted. 7. If total amount of money is correct, invoke "Print transaction receipt". 8. If Transaction Receipt is successfully printed, invoke "Update Database".	

➔ 시스템 보고서 v1 내용이 반영되지 않았다.

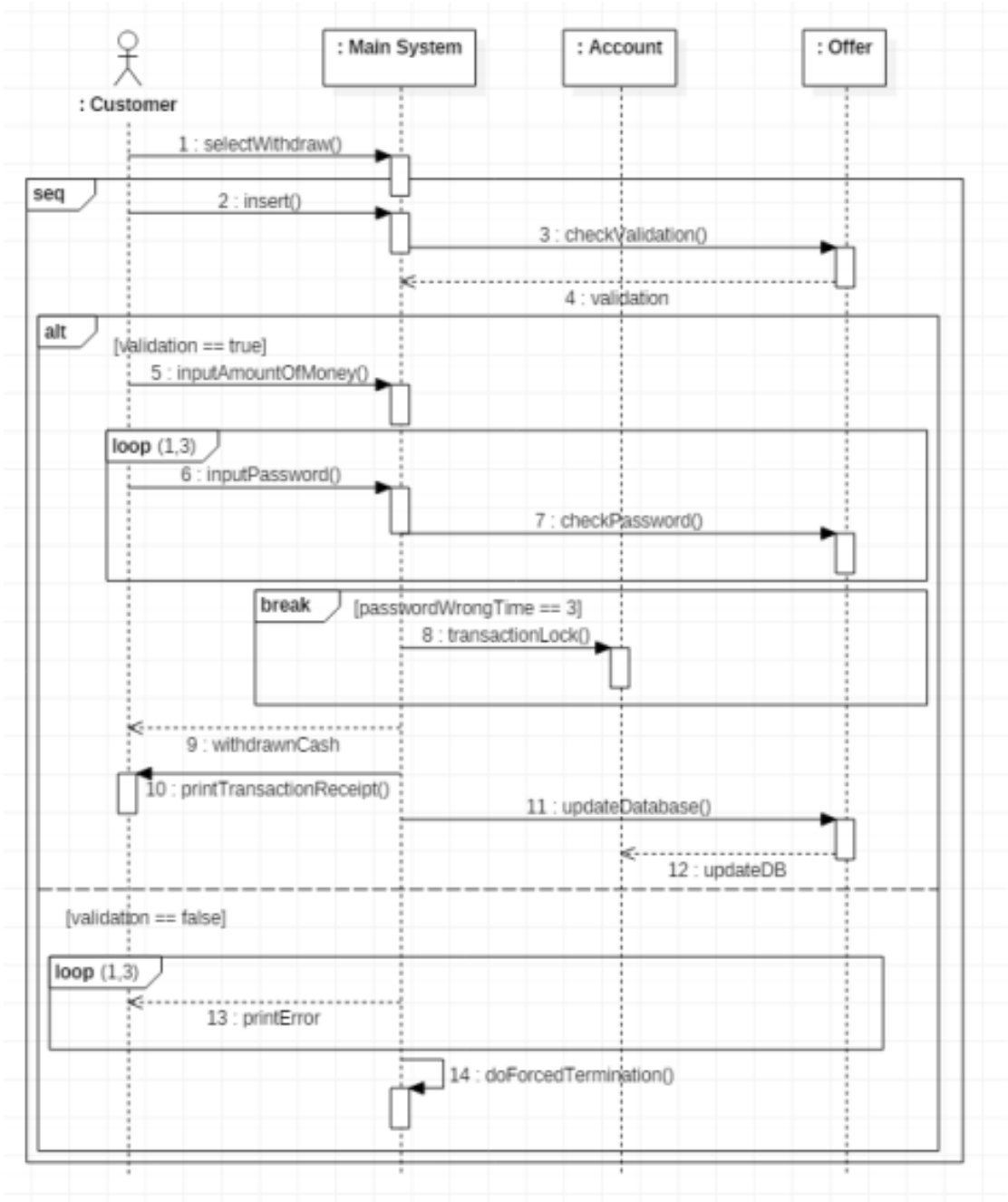
Activity 2039. Analyze (2030) Traceability Analysis

Ref.#	Function	Use Case	Operation
R1.1	Deposit	Deposit	1. selectDeposit
R1.2	Deposit Without Bankbook	Deposit Without Bankbook	2. selectDepositWithoutBankbook
R1.3	Withdraw	Withdraw	3. selectWithdraw
R1.4	Transfer	Transfer	4. selectTransfer
R1.5	Exchange	Exchange	5. selectExchange
R1.6	Loan	Loan	6. selectLoan
R1.7	Pay Utility Bill	Pay Utility Bill	7. selectPayUtilityBill
R1.8	Check Balance	Check Balance	8. selectCheckBalance
R2.1	Insert	Insert	9. selectForeignCurrency
R2.2	Print Transaction Receipt	Print Transaction Receipt	10. insert
R2.3	Print Error	Print Error	11. inputCash
R2.4	Do Forced Termination	Do Forced Termination	12. inputAccountNumber
R3.1	Take Charge	Take Charge	13. inputAmountOfMoney
R4.1	Check Password	Check Password	14. inputPassword
R5.1	Transaction Lock	Transaction Lock	15. Validation
R6.1	Check Validation	Check Validation	
R6.2	Update Database	Update Database	

➔ 모든 Function 과 Use case Operation의 관계들이 그려졌지만 정확한 관계가 보여지지 않는다. 추가적인 수정이 필요하다..

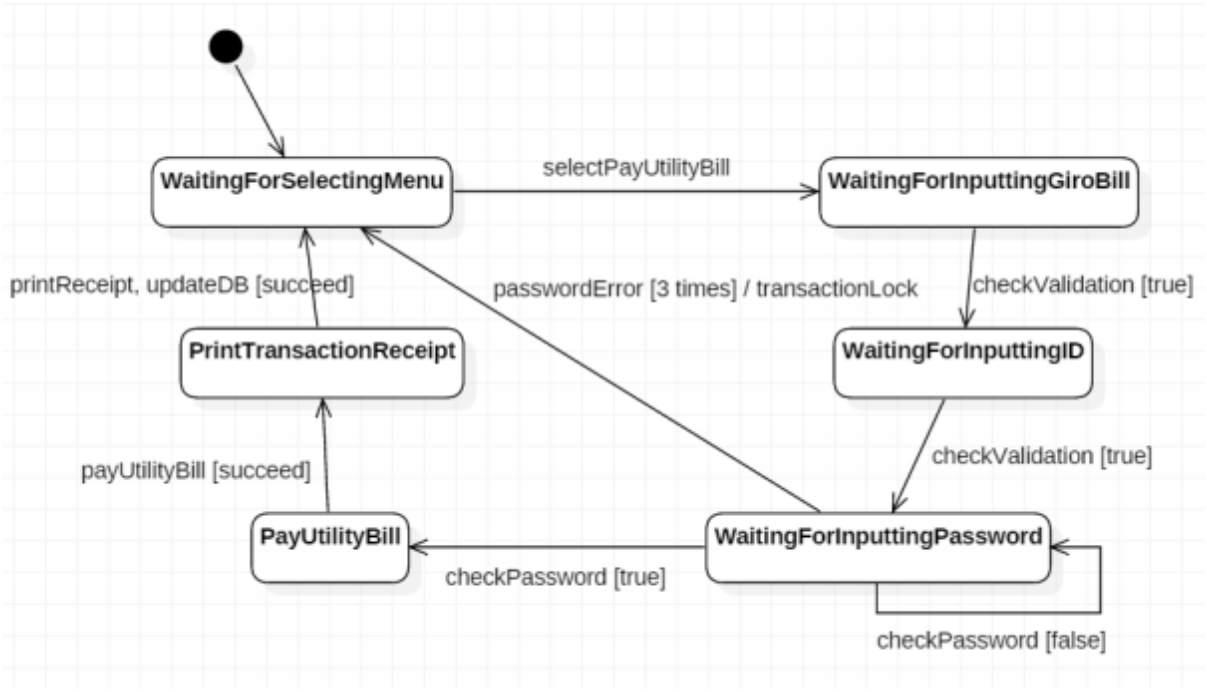
1.3 Stage 2040 문서 분석

3. Withdraw



➔ Password checking에 대한 부분과 Password 오류 발생시에 대한 명시가 추가되었다.

- Pay Utility Bill



➔ Password 3회 오류 발생시에 transctionLock에 대한 명시가 추가되었다.

1.4 Stage 2050 문서 분석

Type	Method
Name	checkBalance()
Purpose	거래내역을 확인한다
Cross Reference	Usecase: 8 Functional: R1.8
Input(Method)	
Output(Method)	void
Abstract operation(Method)	최근 계좌의 거래내역을 출력한다.
Exceptional Courses of Events	-

➔ CheckBalance 명시되어 있으나 실제 프로그램에선 거래내역조회 라는 이름으로 명시되어 있다. 따라서 checkBalance 가 아닌 View transaction history 로 바꾸어 표현해야한다.

2. Brute force testing

2.1 brute force test case

- 문서상 수정으로 기존 test case와 맞지 않는 부분을 중심으로 Brute force test을 위한 리스트를 수정하였다.

No.	테스트 설명
1	입금 페이지 안내 문구 확인
2	무통장 입금 해보기
3	거래 정지 기능 작동 확인
4	통장으로 대출 실행 시 안내 문구 유무 확인
5	키보드 입력 확인
6	대출 한도 기능 확인
7	아주 큰 금액을 입력
8	대출 상환 여부 확인
9	한도를 초과한 대출 실행 시 안내 문구 적절성 여부
10	대출 실행 후 GUI 상태 확인
11	자기 자신에게 송금 해보기
12	10000원 단위가 아닌 돈을 입금해보기
13	출금 단계 적절성 확인
14	지폐로 출금이 불가능한 금액 출금해보기
15	거래 정지된 계좌에서 출금해보기
16	5만원 이상 출금시 지폐 종류 개수 출력 확인
17	대출, 송금, 환전시 수수료 처리 여부 확인
18	신용 등급 조회
19	명세서 출력 여부 확인
20	환전에서 1만, 10만, 100만 버튼 작동 확인

➔ 총 20개의 Brute force 테스트 케이스를 선정하고 각 테스트 케이스에 따른 Expected output을 도출했다.

2.2 Test execution

No.	Expected output	P/F
1	입금 페이지에서 사용 가능 지폐 단위 등에 대한 안내 문구가 표시된다.	P
2	현대 카드로 은행 변경 후 금액 1만원을 무통장 입금한다.	F
3	비밀 번호를 3회 이상 틀린 후 출금을 진행하면 거래 정지 안내문구와 함께 출금이 진행이 되지 않는다.	P
4	통장으로 대출을 실행하면 통장은 대출을 진행할 수 없다는 안내가 나온다.	P
5	키보드로 시스템에 입력을 할 수 있다.	P
6	한도가 50만원인 계좌에 대출을 6번 연속 10만원씩 진행하면 6번째 시도 시 한도가 초과되었다는 안내와 함께 대출이 되지 않는다.	P
7	금액 한도에 대한 안내 등 적절한 조치와 함께 프로그램의 오류가 발생하지 않는다.	P
8	대출금이 있는 계좌에 돈을 입금하면 입금 금액만큼 대출액이 줄어든다.	P
9	한번에 정해진 한도를 초과하는 대출을 실행하면 한도가 초과라는 안내와 함께 대출이 되지 않는다.	P
10	대출을 정상적으로 실행하면 메인 메뉴 창으로 돌아간다.	P
11	보내는 계좌와 받는 계좌가 똑같다는 안내와 함께 진행되지 않는다.	P
12	단위가 맞지 않는다는 안내가 나온다.	P
13	출금이 출금 금액 입력 후 비밀번호를 입력하도록 진행된다.	P
14	단위가 맞지 않는다는 안내가 나온다.	F
15	거래가 정지된 계좌라고 안내되면서 출금이 중지된다.	P
16	출금 결과 페이지에서 5만원권, 1만원권이 몇 장씩 나오는지 안내된다.	P
17	정해진 수수료 정책에 따라 처리가 된다.	F
18	신용 등급이 올바르게 제공된다.	F
19	따로 확인할 수 있는 명세표가 어느 형태로든 제공된다.	F
20	1만, 10만, 100만이 잘 표시된다.	F

➔ 총 20개의 test case 중 6개의 test case가 실패하였다. 실제 실행 결과와 피드백을 'redmine'의 일감으로 등록하여 개발자가 확인할 수 있도록 하였다.

➔ 1차 System test 결과 0개 성공에서 14개 성공으로 많이 개선된 모습을 볼 수 있다.

➔ 14/20 > 70% success

3. Category Partitioning Test

3.1 Testable units

Group	Category
Configuration	Program mode
	Availability
Data type	Address input type
	Bill input type
	Address length
	Bill range
Check	Check sender address
	Check loan limit
	Check password
	Check receiver address
	Check balance

➔ 3개의 group과 11개의 category를 도출했다.

3.2 Representative classes of value

Category	Representative classes of value	Ref #
Program mode	**Deposit**	**101**
	Deposit without bankbook	**102**
	Withdraw	**103**
	Loan	**104**
	Transfer	**105**
	Exchange	**106**
	Pay utility bill	**107**
	Check balance	**108**
Availability	**Available**	**111**
	Unavailable	**112**
Address length	**Valid length**	**201**
	Short address	**202**
	Long address	**203**
Bill range	**Valid unit (10000, 50000)**	**211**
	0 < 10000	**212**
	10000 < 50000	**213**
	50000 < 100000	**214**

	Over integer range	**215**
Address input type	**Number**	**221**
	Not number	**222**
Bill input type	**Number**	**231**
	Not number	**232**
Check sender address	**Exist address**	**401**
	Non-exist address	**402**
Check loan limit	**Input < limit**	**411**
	Input > limit	**412**
Check password	**Correct password**	**421**
	Incorrect password	**422**
Check receiver address	**Exist address**	**431**
	Non-exist address	**432**
	Same with sender address	**433**
Check balance	**Enough money**	**441**
	Not enough money	**442**

➔ 각 category별로 representative classes of value를 뽑고 번호를 매겼다.

3.3 Generate Test case

➔ 만들어진 category를 통해 총 46080개의 test case를 생성했다.

3.3.1 Single Constraint 적용

Category	Representative classes of value
Address input type	Not number
Bill input type	Not number
Address length	Short address
	Long address
Bill range	Over integer range
Check sender address	Non-exist address
Check loan limit	Input > limit
Check password	Incorrect password
Check receiver address	Non-exist address
Check balance	Not enough money
Program mode	Check balance

➔ 기존 46080개의 test case가 총 123개로 줄어들었다.

3.3.2 Error Constraint 적용

Category	Representative classes of value
Check receiver address	Same with sender address

➔ 기존 123개의 test case가 총 68개로 줄어들었다.

3.3.3 If-Property Constraints 적용

Category	Representative classes of value	Constraints
Program mode	**Deposit**	**[property deposit]**
	Deposit without bankbook	**[property noBankbook]**
	Withdraw	**[property withdraw]**
	Loan	**[property loan]**
	Transfer	**[property transfer]**
	Exchange	**[property exchange]**
	Pay utility bill	**[property utility]**
Availability	**Available**	**[property AB]**
	Unavailable	**[property UAB]**
Bill input type	**Number**	**[if !utility]**
	Not number	**[if !utility]**
Address length	**Valid length**	**[property VA]**
Bill range	**Valid unit (10000, 50000)**	**[if !utility] [property VM]**
	0 < 10000	**[if !utility]**
	10000 < 50000	**[if !utility]**
	50000 < 100000	**[if !utility]**
Check sender address	**Exist address**	**[if VA && (transfer || exchange || utility || loan || withdraw)] [property ESA]**
Check loan limit	**Input < limit**	**[if (VM && ESA) && loan]**
Check password	**Correct password**	**[if ESA && !deposit && !noBankbook] [property CP]**
Check receiver address	**Exist address**	**[if VA && (transfer || deposit || noBankbook)] [property ERA]**
Check balance	**Enough money**	**[if CP && (withdraw || transfer || utility || exchange)]**

➔ 기존 68개의 test case가 총 62개로 줄어들었다.

3.4 Testing Result

ref #	sequence	description	P/F
1	108	계좌 잔액을 조회한다.	P
2	222	숫자가 아닌 문자로 계좌번호에 입력한다.	P
3	232	숫자가 아닌 문자로 입금금액을 입력한다.	P
4	202	기준 길이보다 짧은 계좌번호를 입력한다.	P
5	203	기준 길이보다 긴 계좌번호를 입력한다.	P
6	215	금액 입력란에 int 타입이 표현가능한 범위 이상의 값을 입력한다.	P
7	402	송신 계좌 번호에 존재하지 않는 계좌 번호를 입력한다.	P
8	412	대출 금액에 대출 한도보다 큰 금액을 입력한다.	P
9	422	틀린 비밀번호를 입력한다.	P
10	432	수신 계좌 번호에 존재하지 않는 계좌 번호를 입력한다.	F
11	433	송신 계좌 번호와 같은 계좌 번호를 수신 계좌 번호로 입력한다.	P
12	442	잔액이 충분하지 않은 계좌에 출금, 송금, 환전을 실행한다.	P
13	101, 111, 201, 211, 221, 231, 431	입금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 적절한 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	F
14	101, 111, 201, 212, 221, 231, 431	입금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 0 원에서 10000 원 사이의 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	F
15	101, 111, 201, 213, 221, 231, 431	입금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 10000 원에서 50000 원 사이의 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	F
16	101, 111, 201, 214, 221, 231, 431	입금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 50000 원에서 100000 원 사이의 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	F
17	101, 112, 201, 211, 221, 231, 431	입금거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 적절한 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	F
18	101, 112, 201, 212, 221, 231, 431	입금거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 0 원에서 10000 원 사이의 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	F
19	101, 112, 201, 213, 221, 231, 431	입금거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 10000 원에서 50000 원 사이의 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	F
20	101, 112, 201, 214, 221, 231, 431	입금거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 50000 원에서 100000 원 사이의 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	F
21	102, 111, 201, 211, 221, 231, 431	무통장 입금 거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 적절한 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	F
22	102, 111, 201, 212, 221, 231, 431	무통장 입금 거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 0 원에서 10000 원 사이의 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	F
23	102, 111, 201, 213	무통장 입금 거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 10000 원에서 50000 원 사이의 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	F

	221, 231,, 431		
24	102, 111, 201, 214, 221, 231, 431	무통장 입금 거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 50000 원에서 100000 원 사이의 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	**F**
25	102, 112, 201, 211, 221, 231, 431	무통장 입금 거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 적절한 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	**F**
26	102, 112, 201, 212, 221, 231, 431	무통장 입금 거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 0 원에서 10000 원 사이의 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	**F**
27	102, 112, 201, 213, 221, 231, 431	무통장 입금 거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 10000 원에서 50000 원 사이의 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	**F**
28	102, 112, 201, 214, 221, 231, 431	무통장 입금 거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 50000 원에서 100000 원 사이의 금액을 입력하고 존재하는 송신 계좌번호를 입력한다.	**F**
29	103, 111, 201, 211, 221, 231, 401, 421, 441	출금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 정확한 비밀번호를 입력하고 충분한 잔액이 있는 상태에서 적절한 금액을 입력하고 출금한다.	**P**
30	103, 111, 201, 212, 221, 231, 401, 421, 441	출금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 정확한 비밀번호를 입력하고 충분한 잔액이 있는 상태에서 0 원에서 10000 원 사이의 금액을 입력하고 출금한다.	**P**
31	103, 111, 201, 213, 221, 231, 401, 421, 441	출금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 정확한 비밀번호를 입력하고 충분한 잔액이 있는 상태에서 10000 원에서 50000 원 사이의 금액을 입력하고 출금한다.	**P**
32	103, 111, 201, 214, 221, 231, 401, 421, 441	출금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 정확한 비밀번호를 입력하고 충분한 잔액이 있는 상태에서 50000 원에서 100000 원 사이의 금액을 입력하고 출금한다.	**P**
33	103, 112, 201, 211, 221, 231, 401, 421, 441	출금거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 정확한 비밀번호를 입력하고 충분한 잔액이 있는 상태에서 적절한 금액을 입력하고 출금한다.	**P**
34	103, 112, 201, 212, 221, 231, 401, 421, 441	출금거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 정확한 비밀번호를 입력하고 충분한 잔액이 있는 상태에서 0 원에서 10000 원 사이의 금액을 입력하고 출금한다.	**P**
35	103, 112, 201, 213, 221, 231, 401, 421, 441	출금거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 정확한 비밀번호를 입력하고 충분한 잔액이 있는 상태에서 10000 원에서 50000 원 사이의 금액을 입력하고 출금한다.	**P**
36	103, 112, 201, 214, 221, 231, 401, 421, 441	출금거래를 진행하면서 거래 정지된 계좌에 적절한 계좌번호와 정확한 비밀번호를 입력하고 충분한 잔액이 있는 상태에서 50000 원에서 100000 원 사이의 금액을 입력하고 출금한다.	**P**
37	104, 111, 201, 211, 221, 231, 401, 411,	대출거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 비밀번호와 적절한 금액을 입력하고 대출을 받는다.	**P**

	421		
38	104, 111, 201, 212, 221, 231, 401, 411, 421	대출거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 비밀번호와 0 원에서 10000 원 사이의 금액을 입력하고 대출을 받는다.	**F**
39	104, 111, 201, 213, 221, 231, 401, 411, 421	대출거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 비밀번호와 10000 원에서 50000 원 사이의 금액을 입력하고 대출을 받는다.	**F**
40	104, 111, 201, 214, 221, 231, 401, 411, 421	대출거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 비밀번호와 50000 원에서 100000 원 사이의 금액을 입력하고 대출을 받는다.	**F**
41	104, 112, 201, 211, 221, 231, 401, 411, 421	대출거래를 진행하면서 거래 정지된 계좌에 적절한 비밀번호와 적절한 금액을 입력하고 대출을 받는다.	**F**
42	104, 112, 201, 212, 221, 231, 401, 411, 421	대출거래를 진행하면서 거래 정지된 계좌에 적절한 비밀번호와 0 원에서 10000 원 사이의 금액을 입력하고 대출을 받는다.	**F**
43	104, 112, 201, 213, 221, 231, 401, 411, 421	대출거래를 진행하면서 거래 정지된 계좌에 적절한 비밀번호와 10000 원에서 50000 원 사이의 금액을 입력하고 대출을 받는다.	**F**
44	104, 112, 201, 214, 221, 231, 401, 411, 421	대출거래를 진행하면서 거래 정지된 계좌에 적절한 비밀번호와 50000 원에서 100000 원 사이의 금액을 입력하고 대출을 받는다.	**F**
45	105, 111, 201, 211, 221, 231, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 정확한 상대방 계좌번호를 입력하고 충분한 잔액이 있는 상태에서 적절한 금액을 입력하고 송금을 한다.	**F**
46	105, 111, 201, 212, 221, 231, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 정확한 상대방 계좌번호를 입력하고 충분한 잔액이 있는 상태에서 0 원에서 10000 원사이의 금액을 입력하고 송금을 한다.	**F**
47	105, 111, 201, 213, 221, 231, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 정확한 상대방 계좌번호를 입력하고 충분한 잔액이 있는 상태에서 10000 원에서 50000 원사이의 금액을 입력하고 송금을 한다.	**F**
48	105, 111, 201, 214, 221, 231, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 정확한 상대방 계좌번호를 입력하고 충분한 잔액이 있는 상태에서 50000 원에서 100000 원사이의 금액을 입력하고 송금을 한다.	**F**
49	105, 112, 201, 211, 221, 231, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지된 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 정확한 상대방 계좌번호를 입력하고 충분한 잔액이 있는 상태에서 적절한 금액을 입력하고 송금을 한다.	**F**

50	105, 112, 201, 212, 221, 231, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지된 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 정확한 상대방 계좌번호를 입력하고 충분한 잔액이 있는 상태에서 0 원에서 10000 원사이의 금액을 입력하고 송금을 한다.	**F**
51	105, 112, 201, 213, 221, 231, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지된 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 정확한 상대방 계좌번호를 입력하고 충분한 잔액이 있는 상태에서 10000 원에서 50000 원사이의 금액을 입력하고 송금을 한다.	**F**
52	105, 112, 201, 214, 221, 231, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지된 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 정확한 상대방 계좌번호를 입력하고 충분한 잔액이 있는 상태에서 50000 원에서 100000 원사이의 금액을 입력하고 송금을 한다.	**F**
53	106, 111, 201, 211, 221, 231, 401, 421, 441	환전거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 적절한 금액을 입력하고 환전을 받는다.	**F**
54	106, 111, 201, 212, 221, 231, 401, 421, 441	환전거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 0 원에서 10000 원사이의 금액을 입력하고 환전을 받는다.	**F**
55	106, 111, 201, 213, 221, 231, 401, 421, 441	환전거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 10000 원에서 50000 원사이의 금액을 입력하고 환전을 받는다.	**F**
56	106, 111, 201, 214, 221, 231, 401, 421, 441	환전거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 50000 원에서 100000 원사이의 금액을 입력하고 환전을 받는다.	**F**
57	106, 112, 201, 211, 221, 231, 401, 421, 441	환전거래를 진행하면서 거래 정지된 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 적절한 금액을 입력하고 환전을 받는다.	**F**
58	106, 112, 201, 212, 221, 231, 401, 421, 441	환전거래를 진행하면서 거래 정지된 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 0 원에서 10000 원사이의 금액을 입력하고 환전을 받는다.	**F**
59	106, 112, 201, 213, 221, 231, 401, 421, 441	환전거래를 진행하면서 거래 정지된 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 10000 원에서 50000 원사이의 금액을 입력하고 환전을 받는다.	**F**
60	106, 112, 201, 214, 221, 231, 401, 421, 441	환전거래를 진행하면서 거래 정지된 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 50000 원에서 100000 원사이의 금액을 입력하고 환전을 받는다.	**F**
61	107, 111, 201, 221, 401, 421, 441	공과금거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 충분한 잔액이 있는 상태에서 적절한 금액을 입력하고 지불한다.	**P**
62	107, 112, 201, 221, 401, 421, 441	공과금거래를 진행하면서 거래 정지된 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 충분한 잔액이 있는 상태에서 적절한 금액을 입력하고 지불한다.	**P**

➔ 총 62개의 test case를 실행하였고 그 결과 22개의 test case는 pass하였고, 40개의 test case는 Fail하였다.

➔ 20/62 >32 % PASS

➔ 실행 결과와 피드백을 'redmine'의 일감으로 등록하여 개발자가 확인할 수 있도록 하였다.

4. Pairwise Test

4.1 Testable unit

동작모드: deposit, no_bank, withdraw, loan, transfer, exchange, utility, check_bal
계좌상태: avail, n/a
계좌형식: num, non-num
금액형식: num, non-num
계좌길이: vaild, short, long
금액범위: valid, 0~10K, 10~50K, 500~100K
송금계좌: exist, non-exi
대출한계: in<lim, in>lim
비밀번호: correct, incorrect
받는계좌: exist, non-exi
계좌잔고: enough, not_eno
if [동작모드] in {"deposit", "no_bank", "check_bal"} then [송금계좌] = "non-exi";
if [동작모드] in {"withdraw", "loan", "exchange", "utility"} then [받는계좌] = "non-exi";
if [동작모드] not in {"loan"} then [대출한계] = "in<lim";
if [동작모드] in {"utility"} then [금액형식] = "";

4.2 Test Case

동작모드	계좌상태	계좌형식	금액형식	계좌길이	금액범위	송금계좌	대출한계	비밀번호	받는 계좌	계좌잔고
deposit	n/a	num	non-num	vaild	0~10K	non-exi	in<lim	correct	non-exi	enough
withdraw	avail	non-num	num	short	0~10K	exist	in<lim	incorrect	non-exi	not_eno
transfer	avail	num	non-num	long	500~100K	exist	in<lim	correct	exist	not_eno
loan	n/a	non-num	num	long	valid	non-exi	in>lim	incorrect	non-exi	enough
deposit	avail	num	num	vaild	valid	non-exi	in<lim	incorrect	exist	not_eno
loan	n/a	num	non-num	short	10~50K	exist	in>lim	correct	non-exi	not_eno
exchange	avail	non-num	non-num	vaild	10~50K	exist	in<lim	incorrect	non-exi	enough
no_bank	n/a	non-num	num	short	10~50K	non-exi	in<lim	correct	exist	enough
check_bal	avail	non-num	non-num	short	valid	non-exi	in<lim	correct	exist	enough

exchange	n/a	non-num	num	short	500~100K	non-exi	in<lim	incorrect	non-exi	not_eno
exchange	avail	num	num	long	valid	exist	in<lim	correct	non-exi	enough
loan	avail	non-num	num	vaild	0~10K	non-exi	in>lim	correct	non-exi	enough
no_bank	avail	num	non-num	vaild	500~100K	non-exi	in<lim	incorrect	non-exi	enough
transfer	n/a	non-num	num	short	valid	non-exi	in<lim	incorrect	non-exi	enough
deposit	n/a	non-num	non-num	long	500~100K	non-exi	in<lim	correct	non-exi	not_eno
withdraw	n/a	num	non-num	long	10~50K	non-exi	in<lim	correct	non-exi	enough
transfer	n/a	non-num	num	vaild	10~50K	exist	in<lim	correct	exist	enough
withdraw	avail	num	non-num	vaild	valid	exist	in<lim	incorrect	non-exi	not_eno
check_bal	n/a	num	num	vaild	10~50K	non-exi	in<lim	incorrect	non-exi	not_eno
check_bal	n/a	num	non-num	long	500~100K	non-exi	in<lim	correct	non-exi	not_eno
no_bank	avail	non-num	non-num	long	0~10K	non-exi	in<lim	correct	exist	not_eno
transfer	avail	non-num	num	long	0~10K	non-exi	in<lim	incorrect	exist	not_eno
withdraw	n/a	non-num	num	short	500~100K	exist	in<lim	correct	non-exi	not_eno
deposit	n/a	non-num	non-num	short	10~50K	non-exi	in<lim	correct	exist	enough
loan	n/a	num	num	long	500~100K	non-exi	in<lim	incorrect	non-exi	enough
no_bank	avail	non-num	non-num	long	valid	non-exi	in<lim	correct	exist	not_eno
check_bal	avail	num	num	vaild	0~10K	non-exi	in<lim	correct	non-exi	not_eno
loan	avail	non-num	num	vaild	500~100K	exist	in>lim	correct	non-exi	enough
exchange	n/a	num	num	vaild	0~10K	exist	in<lim	incorrect	non-exi	not_eno

4.3 Testing result

동작모드	계좌상태	계좌형식	금액형식	계좌길이	금액범위	송금계좌	대출한계	비밀번호	받는 계좌	계좌잔고	P/F
deposit	n/a	num	non-num	vaild	0~10K	non-exi	in<lim	correct	non-exi	enough	P
withdraw	avail	non-num	num	short	0~10K	exist	in<lim	incorrect	non-exi	not_eno	F
transfer	avail	num	non-num	long	500~100K	exist	in<lim	correct	exist	not_eno	F
loan	n/a	non-num	num	long	valid	non-exi	in>lim	incorrect	non-exi	enough	F
deposit	avail	num	num	vaild	valid	non-exi	in<lim	incorrect	exist	not_eno	F
loan	n/a	num	non-num	short	10~50K	exist	in>lim	correct	non-exi	not_eno	P
exchange	avail	non-num	non-num	vaild	10~50K	exist	in<lim	incorrect	non-exi	enough	P
no_bank	n/a	non-num	num	short	10~50K	non-exi	in<lim	correct	exist	enough	P
check_bal	avail	non-num	non-num	short	valid	non-exi	in<lim	correct	exist	enough	F
exchange	n/a	non-num	num	short	500~100K	non-exi	in<lim	incorrect	non-exi	not_eno	F
exchange	avail	num	num	long	valid	exist	in<lim	correct	non-exi	enough	F
loan	avail	non-num	num	vaild	0~10K	non-exi	in>lim	correct	non-exi	enough	P
no_bank	avail	num	non-num	vaild	500~100K	non-exi	in<lim	incorrect	non-exi	enough	F

transfer	n/a	non-num	num	short	valid	non-exi	in<lim	incorrect	non-exi	enough	F
deposit	n/a	non-num	non-num	long	500~100K	non-exi	in<lim	correct	non-exi	not_eno	F
withdraw	n/a	num	non-num	long	10~50K	non-exi	in<lim	correct	non-exi	enough	F
transfer	n/a	non-num	num	vaild	10~50K	exist	in<lim	correct	exist	enough	P
withdraw	avail	num	non-num	vaild	valid	exist	in<lim	incorrect	non-exi	not_eno	P
check_bal	n/a	num	num	vaild	10~50K	non-exi	in<lim	incorrect	non-exi	not_eno	P
check_bal	n/a	num	non-num	long	500~100K	non-exi	in<lim	correct	non-exi	not_eno	F
no_bank	avail	non-num	non-num	long	0~10K	non-exi	in<lim	correct	exist	not_eno	F
transfer	avail	non-num	num	long	0~10K	non-exi	in<lim	incorrect	exist	not_eno	F
withdraw	n/a	non-num	num	short	500~100K	exist	in<lim	correct	non-exi	not_eno	F
deposit	n/a	non-num	non-num	short	10~50K	non-exi	in<lim	correct	exist	enough	F
loan	n/a	num	num	long	500~100K	non-exi	in<lim	incorrect	non-exi	enough	F
no_bank	avail	non-num	non-num	long	valid	non-exi	in<lim	correct	exist	not_eno	F
check_bal	avail	num	num	vaild	0~10K	non-exi	in<lim	correct	non-exi	not_eno	P
loan	avail	non-num	num	vaild	500~100K	exist	in>lim	correct	non-exi	enough	P
exchange	n/a	num	num	vaild	0~10K	exist	in<lim	incorrect	non-exi	not_eno	P

➔ 'Pict' 툴을 활용하여 pairwise 테스트를 진행하였다. 29개의 test case에 대해 testing을 진행하여 총 11개 test case가 pass하였고, 18개의 test case는 fail했다.

➔ 11/29 > 37% PASS

5. Overall

5.1 System test result

➔ Brute force test: 14/20 = 70% PASS

➔ Category-partition test: 20/62 = 32% PASS

➔ Pairwise test: 11/29 = 37% PASS

5.2 Summary

➔ 새로운 GUI가 생성되는 BUG가 여전히 존재하고 있는 등 1st system test 결과가 반영되지 않은 부분이 여전히 존재하고 있다.

➔ 명세서 등의 문서에 기술된 내용과 여전히 상이한 기능들이 남아있다.

➔ 여전히 많은 테스트 케이스에서 Fail 하는 경우가 많은데 세세한 부분의 부족이 원인이라기보다는 오히려 주요 기능 중 하나가 미흡하여 연관된 모든 케이스가 Fail 하는 모습이 보이고 있다.