Software Verification 1st System Test

1st Testing

- System Test

Project Team

T6

201311265 김상원

201214150 정성철

201210908 김성일

Date

2018-05-27

목차

1. Specification Review

- 실제 시스템 테스트 수행 전 Stage 1000, 2030, 2040 문서를 심층 분석하였음

1.1 Stage 1000 문서 분석

- Non-functional Requirements 관련 [page 3]

4. Non-functional Requirements

- 프로그램 시뮬레이션이므로 모든 입력은 키보드를 통해 이루어진다.

➔ 실제 구현된 프로그램은 키보드 입력이 되지 않으며 모든 동작이 마우스를 통해 동작합니다. 프로그램 특성상 많은 숫자 입력을 해야 하기 때문에 키보드 입력이 전혀 되지않는 부분은 사용자 경험상 상당히 좋지 않다고 볼 수 있습니다. 또 문서상 명시된 가장 기초 요구사항이 불만족 되면서 사실상 이후의 모든 검증이 Fail이라고 생각됩니다.

- Functional Requirements 관련 [page 6~7]

R6	Print Error	Hidden
R6.1	Check Error	Hidden

➔ 문서에 적시된 Functional requirements에 명시되지않은 'Check Error'라는 함수가 System Functions에 명시되어 있습니다. Functional requirements에 추가적인 명시가 필요합니다.

- Use case 관련 [Page10 ~11]

Use cased by Actor

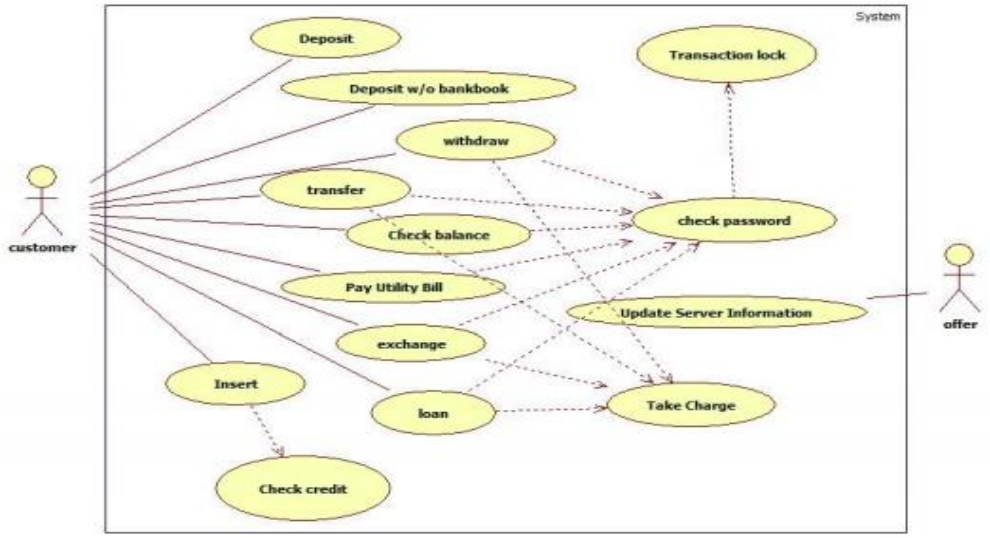
Deposit	Withdraw	Transfer	Check balance	Pay utility bill
Exchange	Insert	Loan	Deposit without bankbook	

Use cased by Event

Request Customer's data	Print error message	Transaction lock	Check validation	Pay utility bill
Take charge	Print transaction receipt	Proceed forced Termination	Check credit	Check password

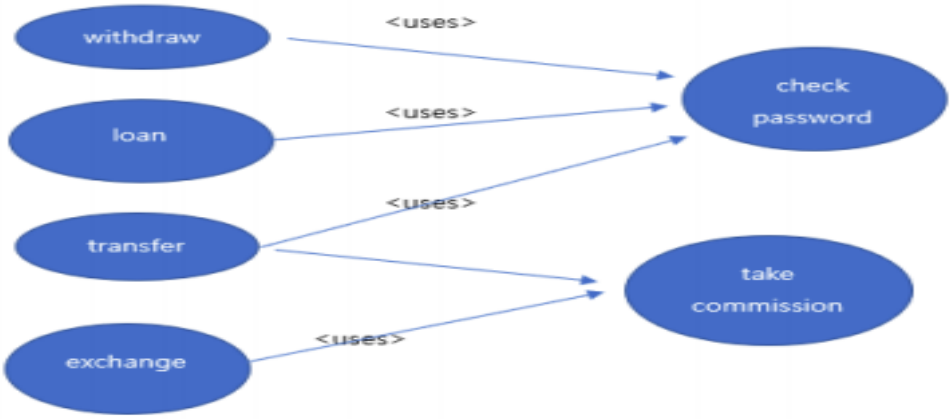
➔ Use case by actor와 Use case by Event에 동일한 use case가 중복되어 있습니다.

- Use case 관련 [page 13]



➔ 명시된 Use case 'Insert'는 다른 Use case에 포함되는 하위 개념으로 보는 것이 적절하다고 생각됩니다.

- Identify the relationships between Use-case 관련 [page 12]



➔ Use case 'exchange'는 문서에 따르면 환전을 하는 동작이므로 계좌에 접근하는 동작입니다. 따라서 Use case 'check password'와 <uses>의 relation을 가진다고 볼 수 있으므로 화살표 추가가 필요합니다.

- Use case Description 관련 [page 14]

Use Case	2. Withdraw
Actors	Customer
Description	-Customer can withdraw cash from account by using card or bankbook. -Customer can only withdraw money in unit of 10000₩ and 50000₩ under balance. -Customer has withdrawing limit at a day. -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 server information.(Hidden)

➔ Use case 'Withdraw'의 description에 Stage 1003에 명시되어 있지 않은 'limit at a day'라는 개념이 새로 등장합니다. Stage 1003에 내용을 추가하여 일관성을 유지해야 합니다.

1.2 Stage 2030 문서 분석

- Use case description 관련 [page 11]

Use Case	12. Forced Termination
Actor	(None)
Purpose	End transaction when error occurs more than 3 times
Overview	System ends transaction automatically when error except password error occurs more than 3 times.
Type	Primary and Essential
Cross Reference	Functional Requirements : R2.3, R2.4, R2.5, R3.3 Use Case : Print Error, Transaction Lock, Check Password
Pre-Requisites	Error occurred more than 3 times

➔ Use case description 상 명시된 Pre-Requisites 'Error occurred more than 3 times'는 실제로 일어날 수 없다고 판단되는 상황입니다.

➔ 심지어 실제 테스트 중 프로그램 작성 오류로 해당 상황이 수십 번 발생 했음에도 해당 Use case가 작동하지 않음. -> reliable 하지 않음.

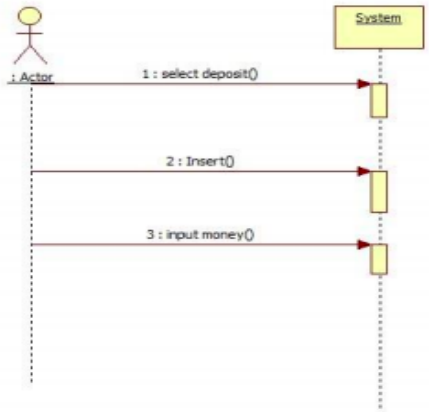
- Use case Description 관련 [page 12]

Use Case	15. Insert
Actor	Customer
Purpose	Check validation of information inputted
Overview	System checks validation of basic information inputted after selecting menu from basic screen.
Type	Primary and Essential
Cross Reference	Functional Requirements : R1.1, R1.2, R1.4, R1.5, R1.6, R1.7, R1.8, R2.3, R3.1, R3.2 Use Case : Deposit, Withdraw, Transfer, Exchange, Loan, Check Balance, Pay Utility Bill, Print Error, Request Customer's Data

➔ Use case 'Insert'는 Use case 이름을 보면 뭔가를 투입한다는 의미를 가지는데 시뮬레이션 프로그램이기 때문에 물리적으로 투입하는 것이 아니므로 현재의 애매모호함을 해결하면서 실제 기능과 좀 더 매칭 되는 description으로 수정이 필요해 보입니다.

- System Sequence Diagram 관련 [page 21]

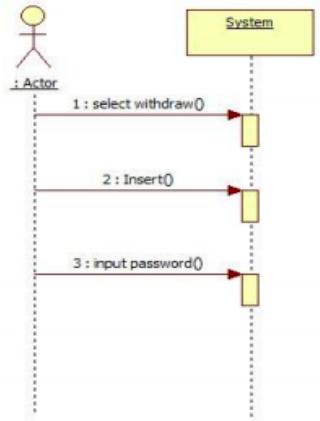
Activity 2035. Define System Sequence Diagrams

Use Case : Deposit	
1. Customer selects deposit menu from the basic screen. 2. System prints out "Input check card or bankbook or credit card". 3. Customer inputs credit card, check card or bankbook. 4. If user information is valid(from "Insert" and "Request Customer's Data"), 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, system invokes "Print transaction receipt". 8. (S) If Transaction Receipt is successfully printed, ask Offer to update server information.	

➔ 좌측 sequence에 명시된 8번 Transaction Receipt의 화살표가 존재하지 않습니다. 이에 따라 해당 단계에 대한 해석이 애매모호 해졌습니다.

- System Sequence Diagram 관련 [page 21]

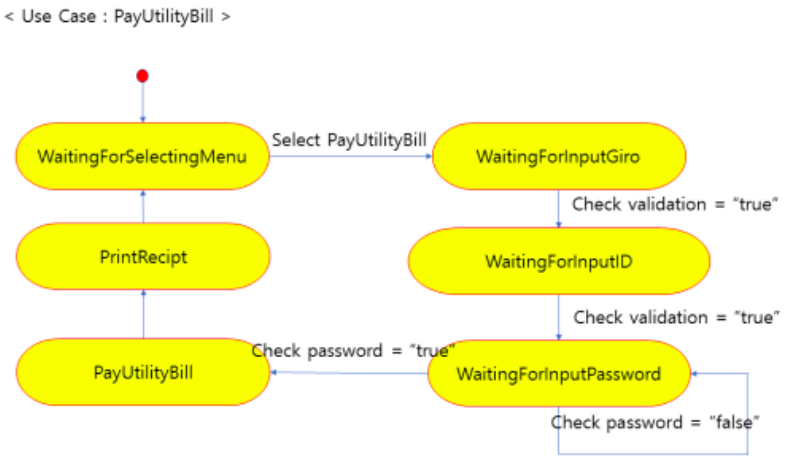
Use Case : Withdraw
1. Customer selects withdraw menu from the basic screen. 2. System prints out "Input check card or bankbook". 3. Customer inputs check card or bankbook. 4. If user information is valid(from "Insert"), System asks customer to input amount of money to withdraw. 5. Customer inputs amount of money in unit of 10000₩ and 50000₩ to withdraw from account. 6. System asks for password for the account. 7. Customer inputs password for the account. 8. Invoke "Check Password". If password is correct, System counts numbers of bills. 9. Invoke "Request Customer's Data". If balance is enough, System withdraws cash and invokes "Print Transaction Receipt". 10. If Transaction Receipt is successfully printed, System asks Offer to update server information.



➔ 좌측 sequence에 명시된 9번 Request Customer's Data의 화살표가 존재하지 않습니다. 이에 따라 해당 단계에 대한 해석이 애매모호 해졌습니다.

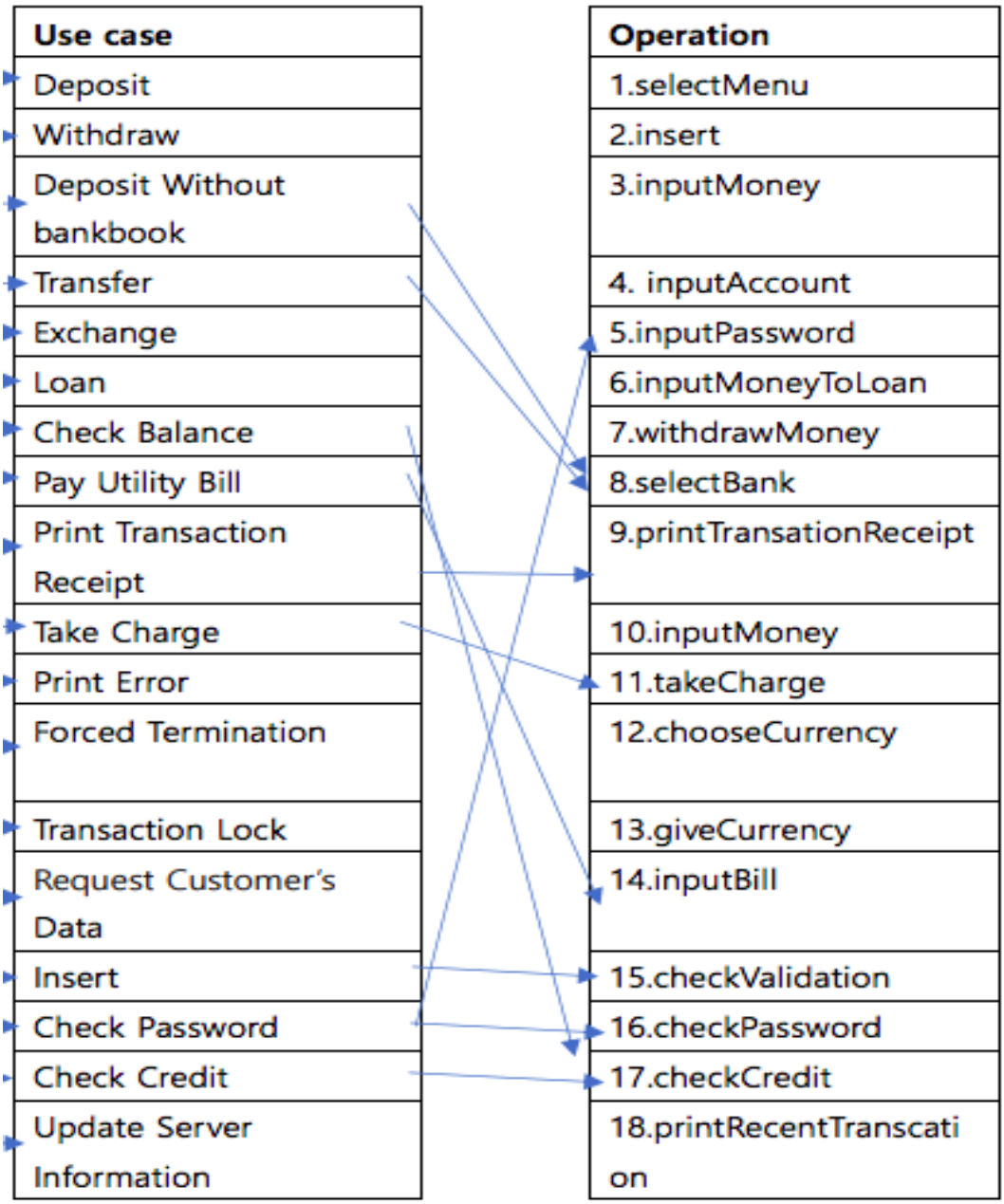
➔ 전반적으로 System의 응답에 관련된 Diagram 도식이 나타나 있지않고 상당히 함축적으로 Diagram이 그려져 있어 정확한 해석이 어렵습니다.

- State diagram 관련 [page 35]



➔ 문서에 지속적으로 등장하는 Use case 'Forced termination'에 대한 state diagram이 존재하지 않습니다.

- Traceability analysis 관련 [page 40]

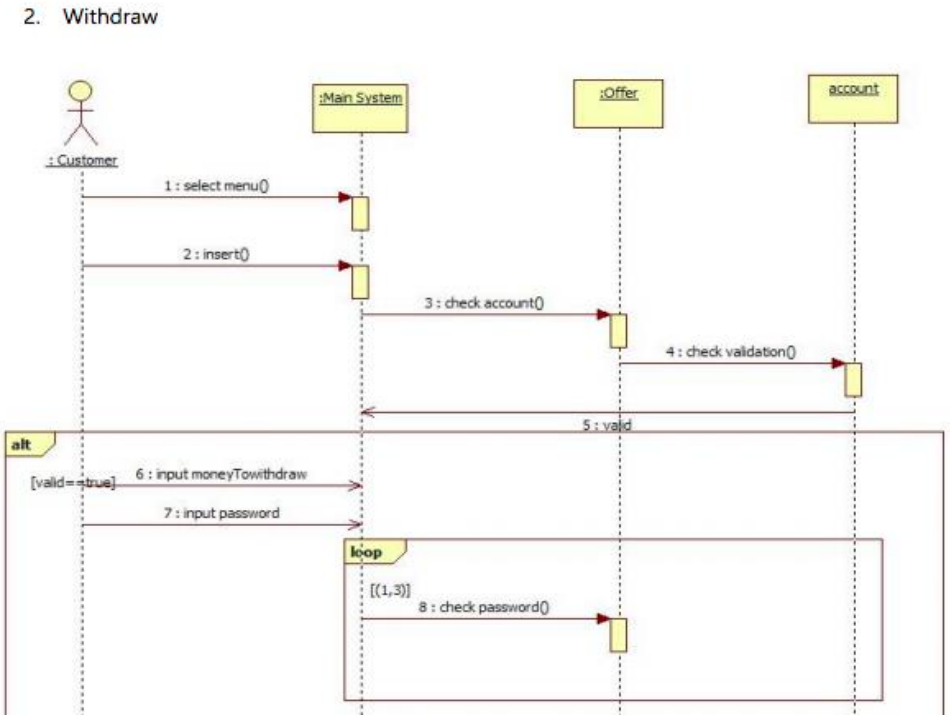


➔ 전반적으로 모든 Use case와 Operation들 간의 연결이 잘못되어 있거나 누락되어 있습니다. 최대한 빠짐없이 모든 연결을 이으면 화살표가 분간이 안 가는 지경으로 그려지는 것으로 알고 있습니다.

➔ 전반적으로 문서의 내용이 구체적이지 않고 애매모호한 부분이 많습니다. 이전 Stage문서와 맞지않는 부분도 일부 존재하고 심지어 본 문서 내에서도 일관성이 맞지 않는 부분이 발견되는 점에서 전반적인 문서 수정이 필요해 보입니다.

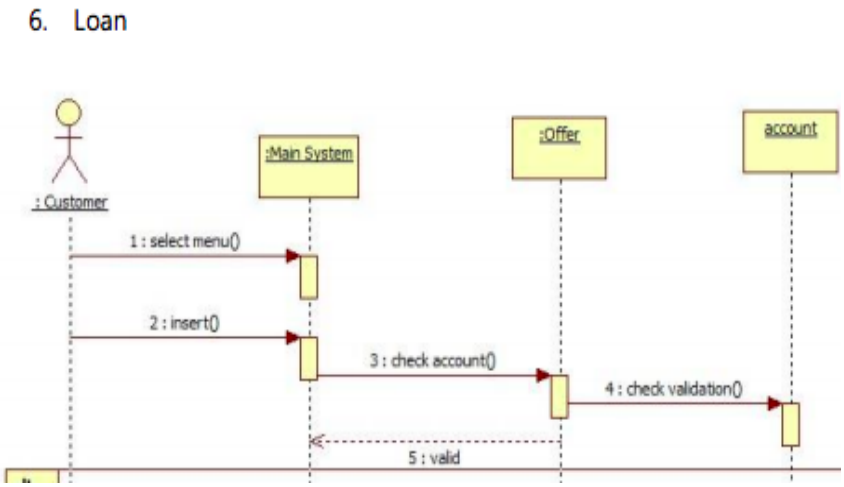
1.3 Stage 2040 문서 분석

- Interaction Diagram 관련 [page 23]



➔ 기존 문서에 명시되어 있는 Account를 통한 password확인 작업부분이 빠져 있습니다. Offer, account간 상호작용도 명시하여 모호함을 해결해야 합니다.

- Interaction Diagram 관련 [page 23]



➔ Account에 대한 validation을 진행하므로 Offer에서 바로 valid 응답이 오는 것이 아니라 account와 offer간의 상호작용을 명시해야 합니다.

2. Brute force testing

2.1 brute force test case

- 각 팀원들의 예상한 오류 발생 가능성이 높은 부분을 중심으로 Brute force test을 위한 리스트를 작성하였습니다.

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

➔ Specification 문서의 기능 요약과 GUI를 바탕으로 총 19개의 Brute force 테스트 케이스를 선정하고 각 테스트 케이스에 따른 Expected output을 도출했습니다.

2.2 Test execution

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

➔ 총 19개의 test case 중 19개의 test case가 실패하였습니다. 실제 실행 결과와 피드백을 'redmine'의 일감으로 등록하여 개발자가 확인할 수 있도록 하였습니다.

➔ 0/19 > 0% success

3. Category Partitioning Test

3.1 Testable units

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

➔ 3개의 group과 9개의 category를 도출했습니다.

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 type	**Valid format**	**201**
	Short address	**202**
	Long address	**203**
Bill type	**Valid unit (10000, 50000)**	**211**
	0 < 10000	**212**
	10000 < 50000	**213**
	50000 < 100000	**214**
	Over integer range	**215**

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 매겼습니다.

3.3 Generate Test case

➔ 만들어진 category를 통해 총 11520개의 test case를 생성했습니다.

3.3.1 Single Constraint 적용

Category	Representative classes of value
Address type	Short address
	Long address
Bill type	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 11520개의 test case가 총 121개로 줄어들었습니다.

3.3.2 Error Constraint 적용

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

➔ 기존 121개의 test case가 총 66개로 줄어들었습니다.

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]
Address type	Valid format	[property VA]
Bill type	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 66개의 test case가 총 60개로 줄어들었습니다.

3.4 Testing Result

ref #	sequence	description	P/F
1	108	계좌 잔액을 조회한다.	F
2	202	기준 길이보다 짧은 계좌번호를 입력한다.	P
3	203	기준 길이보다 긴 계좌번호를 입력한다.	P
4	215	금액 입력란에 int타입이 표현가능한 범위 이상의 값을 입력한다.	F
5	402	송신 계좌 번호에 존재 하지 않는 계좌 번호를 입력한다.	F
6	412	대출 금액에 대출 한도보다 큰 금액을 입력한다.	P
7	422	틀린 비밀번호를 입력한다.	F
8	432	수신 계좌 번호에 존재 하지 않는 계좌 번호를 입력한다.	F
9	433	송신 계좌 번호와 같은 계좌 번호를 수신 계좌 번호로 입력한다.	P
10	442	잔액이 충분하지 않은 계좌에 출금, 송금, 환전을 실행한다.	F
11	101, 111, 201, 211, 431	입금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 적절한 금액을 입력하고 ...	P
12	101, 111, 201, 212, 431	입금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 0원에서 10000원 사이의 금액을...	F
13	101, 111, 201, 213, 431	입금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 10000원에서 50000원 사이의 금액을 ...	F
14	101, 111, 201, 214, 431	입금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 50000원에서 100000원 사이의 금액을 ...	F
15	101, 112, 201, 211, 431	입금거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 적절한 금액을 입력하고 존재하는...	F

16	101, 112, 201, 212, 431	입금거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 0원에서 ...	F
17	101, 112, 201, 213, 431	입금거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 10000원에서 50000원 ...	F
18	101, 112, 201, 214, 431	입금거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 50000원에서 100000원 ...	F
19	102, 111, 201, 211, 431	무통장 입금 거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 적절한 금액을 ...	P
20	102, 111, 201, 212, 431	무통장 입금 거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 0원에서 10000원 ...	F
21	102, 111, 201, 213, 431	무통장 입금 거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 10000원에서 ...	F
22	102, 111, 201, 214, 431	무통장 입금 거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 50000원에서 ...	F
23	102, 112, 201, 211, 431	무통장 입금 거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 적절한 금액을 입력...	F
24	102, 112, 201, 212, 431	무통장 입금 거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 0원에서 10000원 ...	F
25	102, 112, 201, 213, 431	무통장 입금 거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 10000원에서 50000원 ...	F
26	102, 112, 201, 214, 431	무통장 입금 거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 50000원에서 100000원 ...	F
27	103, 111, 201, 211, 401, 421, 441	출금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 정확한 비밀번호를 입력...	P
28	103, 111, 201, 212, 401, 421, 441	출금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 정확한 비밀번호를 ...	F
29	103, 111, 201, 213, 401, 421, 441	출금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 정확한 비밀번호를 ...	F
30	103, 111, 201, 214, 401, 421, 441	출금거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 계좌번호와 정확한 비밀번호를 ...	F

31	103, 112, 201, 211, 401, 421, 441	출금거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 정확한 비밀번호를 입력하고 ...	F
32	103, 112, 201, 212, 401, 421, 441	출금거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 정확한 비밀번호를 입력하고 ...	F
33	103, 112, 201, 213, 401, 421, 441	출금거래를 진행하면서 거래가 정지된 계좌에 적절한 계좌번호와 정확한 비밀번호를 입력하...	F
34	103, 112, 201, 214, 401, 421, 441	출금거래를 진행하면서 거래 정지된 계좌에 적절한 계좌번호와 정확한 비밀번호를 입력하...	F
35	104, 111, 201, 211, 401, 411, 421	대출거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 비밀번호와 적절한 금액을 입력하고 ...	P
36	104, 111, 201, 212, 401, 411, 421	대출거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 비밀번호와 0원에서 10000원 사이의 ...	F
37	104, 111, 201, 213, 401, 411, 421	대출거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 비밀번호와 10000원에서 50000원...	F
38	104, 111, 201, 214, 401, 411, 421	대출거래를 진행하면서 거래 정지되지 않은 계좌에 적절한 비밀번호와 50000원에서 100000원 ...	F
39	104, 112, 201, 211, 401, 411, 421	대출거래를 진행하면서 거래 정지된 계좌에 적절한 비밀번호와 적절한 금액을 입력하고 대출을...	F
40	104, 112, 201, 212, 401, 411, 421	대출거래를 진행하면서 거래 정지된 계좌에 적절한 비밀번호와 0원에서 10000원 사이의 금액을 ...	F
41	104, 112, 201, 213, 401, 411, 421	대출거래를 진행하면서 거래 정지된 계좌에 적절한 비밀번호와 10000원에서 50000원 사이의...	F
42	104, 112, 201, 214, 401, 411, 421	대출거래를 진행하면서 거래 정지된 계좌에 적절한 비밀번호와 50000원에서 100000원 사이의...	F
43	105, 111, 201, 211, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 ...	P
44	105, 111, 201, 212, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 ...	F
45	105, 111, 201, 213, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 ...	F

46	105, 111, 201, 214, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 ...	F
47	105, 112, 201, 211, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지된 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 정확한 ...	F
48	105, 112, 201, 212, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지된 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 정확한 ...	F
49	105, 112, 201, 213, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지된 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 정확한 ...	F
50	105, 112, 201, 214, 401, 421, 431, 441	송금거래를 진행하면서 거래 정지된 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 정확한 ...	F
51	106, 111, 201, 211, 401, 421, 441	환전거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 ...	F
52	106, 111, 201, 212, 401, 421, 441	환전거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 ...	F
53	106, 111, 201, 213, 401, 421, 441	환전거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 ...	F
54	106, 111, 201, 214, 401, 421, 441	환전거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 ...	F
55	106, 112, 201, 211, 401, 421, 441	환전거래를 진행하면서 거래 정지된 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 ...	F
56	106, 112, 201, 212, 401, 421, 441	환전거래를 진행하면서 거래 정지된 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 ...	F
57	106, 112, 201, 213, 401, 421, 441	환전거래를 진행하면서 거래 정지된 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 ...	F
58	106, 112, 201, 214, 401, 421, 441	환전거래를 진행하면서 거래 정지된 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 ...	F
59	107, 111, 201, 401, 421, 441	공과금거래를 진행하면서 거래 정지되지 않은 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 ...	F
60	107, 112, 201, 401, 421, 441	공과금거래를 진행하면서 거래 정지된 계좌에 정확한 내 계좌번호와 정확한 비밀번호를 입력하고 충분한 ...	F

ref #	P/F
1	F
2	P
3	P
4	F
5	F
6	P
7	F
8	F
9	P
10	F
11	P
12	F
13	F
14	F
15	F
16	F
17	F
18	F
19	P
20	F
21	F
22	F
23	F
24	F
25	F
26	F
27	P
28	F
29	F
30	F

ref#	P/F
31	F
32	F
33	F
34	F
35	P
36	F
37	F
38	F
39	F
40	F
41	F
42	F
43	P
44	F
45	F
46	F
47	F
48	F
49	F
50	F
51	F
52	F
53	F
54	F
55	F
56	F
57	F
58	F
59	F
60	F

➔ 총 60개의 test case를 실행하였고 그 결과 9개의 test case는 pass하였고, 51개의 test case는 Fail하였습니다.

➔ 9/60 > 15% PASS

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

4. Pairwise Test

4.1 Testing Result

	동작모드	계좌상태	계좌번호	금액입력	송금계좌	대출한계	비밀번호	받는계좌	계좌잔고	
1	deposit	unavailable	vaild	0~10K	non-exist	input<limit	correct	non-exist	enough	Fail
2	transfer	available	short	10~50K	exist	input<limit	incorrect	exist	not_enough	Pass
3	deposit without bankbook	available	long	500~100K	non-exist	input<limit	correct	exist	enough	Pass
4	pay utility bill	unavailable	long	valid	non-exist	input<limit	incorrect	non-exist	not_enough	Fail
5	exchange	unavailable	short	500~100K	exist	input<limit	correct	non-exist	not_enough	Fail
6	loan	available	vaild	valid	exist	input>limit	incorrect	non-exist	enough	Fail
7	deposit	unavailable	vaild	10~50K	non-exist	input<limit	correct	exist	not_enough	Fail
8	loan	unavailable	short	0~10K	non-exist	input>limit	correct	non-exist	not_enough	Fail
9	withdraw	unavailable	long	10~50K	exist	input<limit	incorrect	non-exist	enough	Fail
10	withdraw	available	vaild	500~100K	non-exist	input<limit	correct	non-exist	not_enough	Fail
11	withdraw	available	short	0~10K	exist	input<limit	incorrect	non-exist	enough	Fail
12	transfer	unavailable	vaild	valid	non-exist	input<limit	correct	exist	enough	Fail
13	deposit without bankbook	unavailable	short	0~10K	non-exist	input<limit	incorrect	non-exist	not_enough	Fail
14	exchange	unavailable	long	valid	non-exist	input<limit	incorrect	non-exist	enough	Pass
15	withdraw	unavailable	short	valid	exist	input<limit	incorrect	non-exist	not_enough	Fail
16	deposit	available	long	500~100K	non-exist	input<limit	incorrect	exist	not_enough	Pass
17	deposit without bankbook	unavailable	vaild	10~50K	non-exist	input<limit	incorrect	exist	not_enough	Fail
18	transfer	unavailable	long	0~10K	exist	input<limit	incorrect	exist	not_enough	Fail
19	transfer	available	long	500~100K	non-exist	input<limit	correct	non-exist	enough	Fail
20	pay utility bill	available	vaild	0~10K	exist	input<limit	correct	non-exist	enough	Fail
21	loan	available	long	10~50K	exist	input>limit	correct	non-exist	not_enough	Pass
22	deposit without bankbook	available	vaild	valid	non-exist	input<limit	correct	exist	not_enough	Fail
23	loan	unavailable	vaild	500~100K	exist	input>limit	correct	non-exist	enough	Fail
24	pay utility bill	available	short	500~100K	non-exist	input<limit	incorrect	non-exist	not_enough	Fail
25	deposit	available	short	valid	non-exist	input<limit	incorrect	exist	enough	Pass
26	exchange	available	vaild	10~50K	exist	input<limit	correct	non-exist	not_enough	Fail
27	exchange	unavailable	vaild	0~10K	exist	input<limit	incorrect	non-exist	not_enough	Fail
28	pay utility bill	available	vaild	10~50K	non-exist	input<limit	correct	non-exist	not_enough	Fail
29	loan	available	short	valid	non-exist	input<limit	correct	non-exist	not_enough	Pass
										계: 7/29

➔ 'Pict' 툴을 활용하려 pairwise 테스트를 진행하였습니다. 29개의 test case에 대해 testing을 진행하여 총 7개 test case가 pass하였고, 22개의 test case는 fail했습니다.

➔ 7/29 > 24% PASS

5. Overall

5.1 System test result

➔ Brute force test: 0/19 = 0% PASS

➔ Category-partition test: 9/60 = 15% PASS

➔ Pairwise test: 7/29 = 24% PASS

5.2 Summary

➔ 문서와는 전혀 다른 설계

➔ 애매모호한 명세표

➔ 필요없는 GUI

➔ 비밀번호 오류시 새로운 GUI 생성되는 bug가 존재