OOAD 2nd cycle

|Payback ATM|

Mun gi tae / Han sang min

Chart

System Action.

- ***Category Partitioning Testing***
- ***Brute Force Testing***
- ***Document***

Static Analyze Action.

- ***PMD***
- ***Sonar Qube***

ETC.

- ***Final Traceability Analyze***
- ***Opinion***

Mode	key	Category
이체	1	이체 금액
		송금할 계좌
출금	2	출금 금액
입금	3	입금 금액
잔액조회	4	-
사용횟수	5	-

Category	Values	Key
로그인 계좌	범위 이내에 있고, DB에 있는 값	1
	범위 이내에 있고, DB에 없는 값	2
	오버플로우/언더플로우 값	3
	음수값	4
	기타 String	5
비밀번호	범위 이내에 있고, DB에 있는 값	1
	범위 이내에 있고, DB에 없는 값	2
	오버플로우/언더플로우 값	3
	음수값	4
	기타 String	5
송금할 계좌	범위 이내에 있고, DB에 있는 값	1
	범위 이내에 있고, DB에 없는 값	2
	오버플로우/언더플로우 값	3
	음수값	4
	기타 String	5
이체 금액	이체범위 안의 값	1
	음수값	2
	오버플로우/언더플로우 값	3
	기타 String	4
출금 금액	출금범위 안의 값	1
	음수값	2
	오버플로우/언더플로우 값	3
	기타 String	4
입금 금액	입금범위 안의 값	1
	음수값	2
	오버플로우/언더플로우 값	3
	기타 String	4
영수증 여부	허용된 대문자, 소문자	1
	허용되지 않은 대문자, 소문자	2
	"true"/"false"	3
	기타 String	4
페이백 여부	허용된 대문자, 소문자	1
	허용되지 않은 대문자, 소문자	2
	"true"/"false"	3
	기타 String	4

Category Partitioning Testing

Test	Description		Pre-Test result	Fix management	Test result (2nd cycle)
1	이체 관련	영수증: 허용된 대문자 소문자 페이백: 허용된 대문자 소문자	failed	페이백은 숫자만 받는다	ignored
2		영수증: 허용된 대문자 소문자 페이백: 허용되지 않은 대문자 소문자	failed	페이백은 숫자만 받는다	ignored
3	로그인 계좌: 범위 내에 있고 DB에 있는 값	영수증: 허용된 대문자 소문자 페이백: true false 삽입	failed	페이백은 숫자만 받는다	ignored
4		영수증: 허용된 대문자 소문자 페이벡: 기타 string	failed	페이백은 숫자만 받는다	ignored
5	비밀 번호: 범위 내에 있고 DB에 있는 값	영수증: 허용되지 않은 대,소문자 페이백: 허용된 대문자 소문자	failed	영수증자체의 허용되지 않는 대소문자는 구별하지만 역시나 페이벡은 숫자만 받는다	ignored
6	송금 계좌: 범위 내에 있고 DB에 있는 값	영수증: 허용되지 않은 대,소문자 페이백: 허용되지 않은 대문자 소문자	failed	대소문자는 구별하지만 페이백은 숫자만 받는다	ignored
7	이체금액: 이체 범위 안의 값	영수증: 허용되지 않은 대,소문자 페이백: true false 삽입	failed	영수증자체의 허용되지 않는 대소문자는 구별하지만 역시나 페이백은 숫자만 받는다	ignored
8		영수증: 허용되지 않은 대,소문자 페이벡:기타 string	failed	영수증자체의 허용되지 않는 대소문자는 구별하지만 역시나 페이백은 숫자만 받는다	ignored
9		영수증: true false 삽입 시 페이백: 허용된 대문자, 소문자	failed	영수증을 true/false로 입력 받는다는 전제 없다. 페이벡은 숫자만 받는다	ignored

60	(Key = 2.5.0.0.0.0.0.0.0.)		Passed		Passed
61	(Key = 3.1.0.0.0.0.1.1.1.)		Passed		Passed
62	입금 관련 로그인 계좌: 범위 이내에 있고, DB에 있는 값	페이백: 허용되지 않은 대문자, 소문자	failed	페이벡은 숫자만 받는다	ignored
63		페이벡: "true" /" false"	failed	페이벡은 숫자만 받는다	ignored
64	입금 금액: 입금 범위 안의 값 영수증: 허용된 대문자 소문자	페이백: 기타 String	failed	페이벡은 숫자만 받는다	ignored
65	(Key = 3.1.0.0.0.0.1.2.1.)		passed		passed
66	(Key = 3.1.0.0.0.0.1.2.2.)		Passed		Passed
67	(Key = 3.1.0.0.0.0.1.2.3.)		passed		Passed

15.8%→56.6%→100%

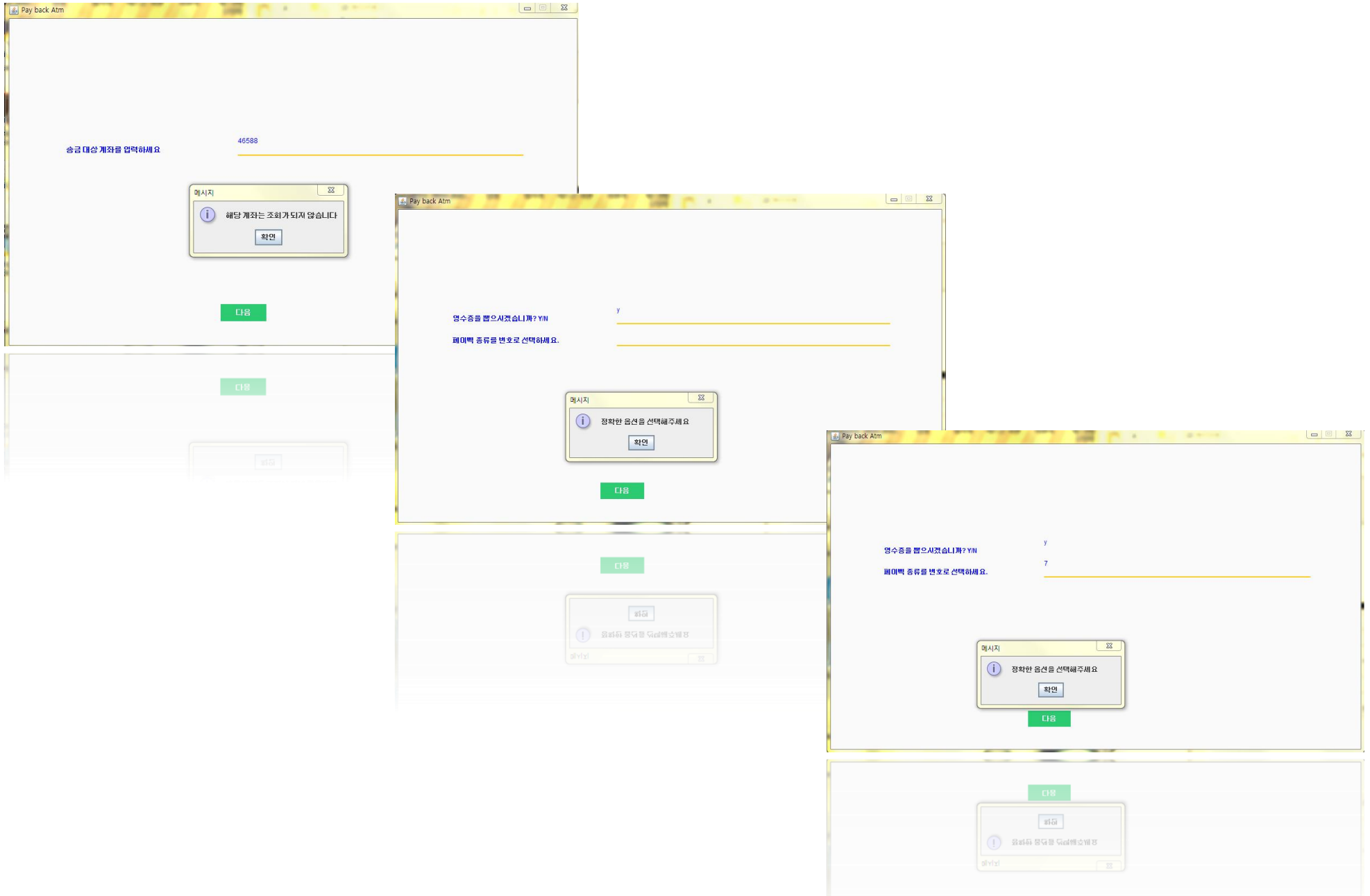
6/38 47/83 83/83

Brute Force Testing

Test Case#	Test Case	Pre-test result	Fix management	Test result
1	페이백 종류 범위값 내 번호 입력	Pass		Pass
2	페이벡 종류 입력하지 않음	Fail	예외처리	Pass
3	페이벡 종류 범위값 외 번호 입력	Fail	예외처리	Pass
4	Printstatement의 입력으로 임의의string	Pass		Pass
5	로그인 계좌와 동일한 계좌로 송금 요청	Pass		Pass
6	입금/출금/송금 금액에 음수 값 입력	Pass		Pass
7	100000이상의 입금을 수행하고 DB파일 잔액변화유무 확인	Pass		Pass
8	DB 파일 한도를 음수 값으로 설정한 후 프로그램 실행	Pass		Pass
9	영수증 (y/n)입력하지 않음	Pass		Pass
10	이체시 DB에 없는 계좌번호값 입력	Fail	예외처리	Pass
11	페이벡 종류 영수증 출력값 둘다 입력하지 않음	Pass		Pass
12	입금 금액 입력시 특정값 입력	Pass		Pass
13	입력 값이 잘못된 경우에 다음으로 진행하는데 (Fail인 상태) 그때 영수증 페이백 입력	Fail	예외처리	Pass
14	수수료 발생 유무 확인	Pass		Pass

7%→64%→100%

Brute Force Testing



User 명세에 대하여

검증팀에서는 User를 Customer와 Admin을 구별하여 명확하게 표기하라고 요구함

User는 Customer 한 명이기 때문에 명세로 표현할 필요가 없음. (Admin 정의 안함)

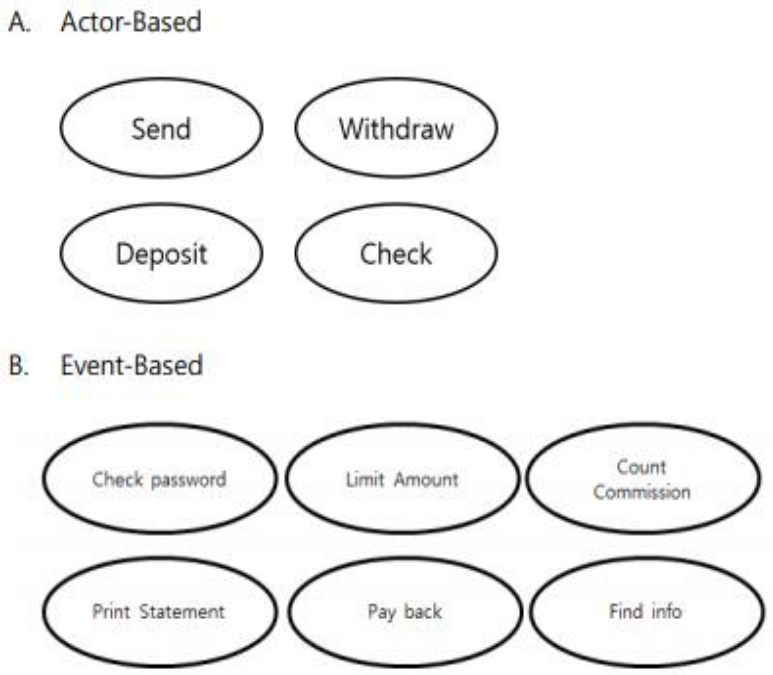
Interaction Diagram

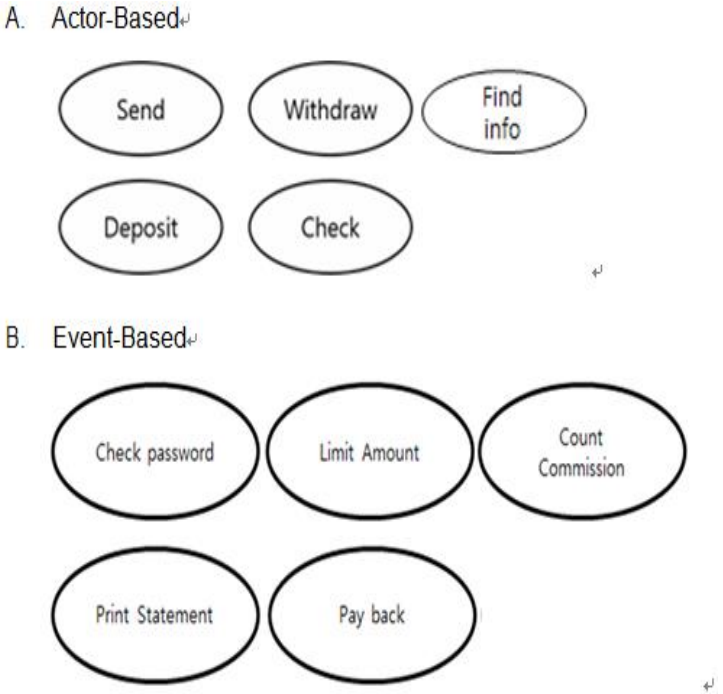
수정되어야 할 부분(선 표시 올바르게)에 대해 요구함

요구사항대로 수정함

Based에 대하여

> 검증팀에서 Find Info 라는 Use-Case는 사용자의 계좌번호 입력에 따라 실행 되어지는 것이라 보고 Event Based가 아닌 Actor Based로 수정하길 권유





Find Error.

All class and interfaces have to belong in named package

Each class should be declare at least one constructor

Avoid unused variables

It is somewhat confusing to have filed name same as a class name

Methods should not contain underscores

Header comment requirement required

To avoid mistakes add a comment at the beginning of the “commission” filed

if you want a default access modifier

Variables should be start with a lowercase character

Use explicit scoping instead of the default package private level

PMD report

Problems found

#	File	Line	Problem
1	Account.java	2	All classes and interfaces must belong to a named package
2	Account.java	2	Each class should declare at least one constructor
3	Account.java	2	headerCommentRequirement Required
4	Account.java	3	Avoid unused private fields such as 'Account'.
5	Account.java	3	Field Account has the same name as a method
6	Account.java	3	It is somewhat confusing to have a field name matching the declaring class name
7	Account.java	3	Variables should start with a lowercase character, 'Account' starts with uppercase character.
8	Account.java	3	fieldCommentRequirement Required
9	Account.java	4	Field Password has the same name as a method
10	Account.java	4	Variables should start with a lowercase character, 'Password' starts with uppercase character.
11	Account.java	4	fieldCommentRequirement Required
12	Account.java	5	Only variables that are final should contain underscores (except for underscores in standard prefix/suffix), 'Total_Amount' is not final.
13	Account.java	5	Variables should start with a lowercase character, 'Total_Amount' starts with uppercase character.
14	Account.java	5	fieldCommentRequirement Required
15	Account.java	6	Only variables that are final should contain underscores (except for underscores in standard prefix/suffix), 'Limit_Amount' is not final.
16	Account.java	6	Variables should start with a lowercase character, 'Limit_Amount' starts with uppercase character.
17	Account.java	6	fieldCommentRequirement Required

792	PrintStatement.java	45	Local variable 'c' could be declared final
793	Send.java	2	All classes and interfaces must belong to a named package
794	Send.java	2	Avoid short class names like Send
795	Send.java	2	Each class should declare at least one constructor
796	Send.java	2	headerCommentRequirement Required
797	Send.java	3	Found non-transient, non-static member. Please mark as transient or provide accessors.
798	Send.java	3	To avoid mistakes add a comment at the beginning of the commission field if you want a default access modifier
799	Send.java	3	Use explicit scoping instead of the default package private level
800	Send.java	3	fieldCommentRequirement Required
801	Send.java	4	Variables should start with a lowercase character, 'Amount' starts with uppercase character.
802	Send.java	4	fieldCommentRequirement Required
803	Send.java	5	Only variables that are final should contain underscores (except for underscores in standard prefix/suffix). 'Receiver_Account' is not final.
804	Send.java	5	Variables should start with a lowercase character, 'Receiver_Account' starts with uppercase character.
805	Send.java	5	fieldCommentRequirement Required
806	Send.java	7	Method names should not contain underscores
807	Send.java	7	Parameter 'amount' is not assigned and could be declared final
808	Send.java	7	To avoid mistakes add a comment at the beginning of the get_Amount method if you want a default access modifier
809	Send.java	7	Use explicit scoping instead of the default package private level
810	Send.java	13	Method names should not contain underscores
811	Send.java	13	To avoid mistakes add a comment at the beginning of the send_Amount method if you want a default access modifier
812	Send.java	13	Use explicit scoping instead of the default package private level
813	Withdraw.java	2	All classes and interfaces must belong to a named package
814	Withdraw.java	2	Each class should declare at least one constructor
815	Withdraw.java	2	headerCommentRequirement Required
816	Withdraw.java	3	Variables should start with a lowercase character, 'Amount' starts with uppercase character.
817	Withdraw.java	3	fieldCommentRequirement Required
818	Withdraw.java	5	Method names should not contain underscores
819	Withdraw.java	5	Parameter 'amount' is not assigned and could be declared final
820	Withdraw.java	5	To avoid mistakes add a comment at the beginning of the get_Amount method if you want a default access modifier
821	Withdraw.java	5	Use explicit scoping instead of the default package private level
822	Withdraw.java	11	Method names should not contain underscores
823	Withdraw.java	11	To avoid mistakes add a comment at the beginning of the send_Amount method if you want a default access modifier
824	Withdraw.java	11	Use explicit scoping instead of the default package private level
825	Withdraw.java	13	Local variable 'commission' could be declared final

Solution.

검증팀에서 제공한 환경에서 만들다 보니 디폴트 패키지 안에서 코딩

인터넷에 찾아보니 정적 분석 tool 마다 다르게 잡는다고 나와 있음
(꼭 필수적인 상황이 아니라고 생각하여 수정하지 않았다.)

사용 되고 있지 않은 변수를 찾아 모두 제거
클래스 이름과 변수 이름이 대문자 소문자까지 똑같은 것이 있어 변수를 소문자로 변경

```
public class Account extends Bank{
    private static long Account;
    public static long password;
    public static long total_Amount;
    public static long limit_Amount;
    public static long use_frequency;
```

```
//controller에 유저의 정보를 넘겨주기위한 변수들을 저장하고 있는 class이다
public class Account extends Bank{
    private static long account;
    public static long password;
    public static long total_Amount;
    public static long limit_Amount;
    public static long use_frequency;
```

```
//controller에 유저의 정보를 넘겨주기위한 변수들을 저장하고 있는 class이다
public class Account extends Bank{
    private static long account;
    public static long password;
    public static long total_Amount;
    public static long limit_Amount;
    public static long use_frequency;
```

Solution.

```
public class Controller {
    public static long User_Account;
    public static long Receiver_Account;
    private static long Password;
    private static String Category;
    public static long Input_Amount;
    private static long User_id;
    private static long User_password;
    private static long User_limit;
    private static long User_Totalmoney;
    private static long User_frequency;
    private static long Receiver_id;
    private static long Receiver_password;
    private static long Receiver_limit;
    private static long Receiver_Totalmoney;
    private static long Receiver_frequency;
    private static String U_bank;
    private static String R_bank;
```

```
import java.io.IOException;
// 전반적인 값들의 toss를 담당하는 클라스이다 여기에 유저와 리
public class Controller {
    public static long user_Account;
    public static long receiver_Account;
    private static long password;
    private static String category;
    public static long input_Amount;
    private static long user_id;
    private static long user_password;
    private static long user_limit;
    private static long user_Totalmoney;
    private static long user_frequency;
    private static long receiver_id;
    private static long receiver_password;
    private static long receiver_limit;
    private static long receiver_Totalmoney;
    private static long receiver_frequency;
    private static String u_bank;
    private static String r_bank;
    Scanner s=new Scanner(System.in);
```

Find Error.

3개의 버그와 35개의 취약점을 가지고 있다고 분석 (코드의 중복도 높지 않다는 것을 확인)

Bugs	Weakness	New Bugs	New Weakness
3	35	3	35
3	35	3	35

Code Reduplication	Duplicated blocks	Code Reduplication New Code lines
2.7 %	2	2.7 % 1.3K New Code Lines
2.7 %	2	2.7 % 1.3K New Code Lines

그러나 어떤 곳이 버그를 유발하며 어떤 곳이 취약한지를 파악불가

Final Traceability Analyze

Use Case		Operation in Sequence Diagrams		Methods	Class		Unit Test
1. Find Info		1. Get Account		Get Account(account_id: int): void	Controller		Input Account ID
2. Check Password		2. Category		Get Receiver_Account(account_id: int): void			Load Info
3. Limited Amount		3. Input Password		Input Password(password: int): void			Input Password
4. Count Commission		4. Get Receiver Account		Check Password(password: int): boolean			Check Password
5. Send		5. Input Amount		Input Amount(amount: int): void			Get Amount
6. Withdraw		6. Print Remain Amount		Check Limit(amount: int): Boolean			Get Total(total_amount)
7. Deposit		7. Get Answer		Category(category: String): void			Limited Amount
8. Check Remain				Find Info(account_id : int): void	Bank		Get Answer
9. Print Statement				Load Info(): void			Print Statement
10. Payback				Call Func(category: String): void			Check Payback
				Make Func(category: String): void			Choose Gift Code
				Update Account(User_id: int,Ch_amount: int):void			
				Count Commission(amount: int): int	Commission		
				Get Commission(amount: int): void			
				Check Current Time(): int			
				Get Total(total_amount)			
				Check Payback(frequency: int): void	Payback		
				Check Frequency(frequency: int): Boolean			
				Get Gift Code(G_code: int): void			
				Add Gift Code(G_code: int): void			
				Choose Gift Code(): void			
				Get Answer(answer: String): Boolean	Statement		
				Print Statement(): void			
				Send Amount(amount: int): int	Send/Withdraw		
				Show Amount(): void	Check Remain		
				Check Total Amount(): int			
				Print Total Amount(total_amount: int): void			

Opinion