Software Modeling & Analysis

[OOPT stage 2nd cycle]

No.	T7
Subject	Software Modeling & Analysis
Professor	JUNBEOM YOO
Team Member	201615007 문기태
	201410621 한상민

Chart.

System Action. Category Partitioning Testing

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

Test #	description		Pre-Test result	Fix management	Test result
1.					
2.					
3.	**이체 관련** 로그인 계좌: 범위 내에 있고 DB에 있는 값 비밀 번호: 범위 내에 있고 DB에 있는 값 송금 계좌:	영수증: 허용된 대문자 소문자 페이백: 허용된 대문자 소문자	**failed**	페이벡은 숫자만 받는다	**ignored**
4.		영수증: 허용된 대문자 소문자 페이백: 허용되지 않은 대문자 소문자	**failed**	페이벡은 숫자만 받는다	**ignored**
5.		영수증: 허용된 대문자 소문자 페이백: true false 삽입	**failed**	페이벡은 숫자만 받는다	**ignored**
6.		영수증: 허용된 대문자 소문자 페이벡: 기타 string	**failed**	페이벡은 숫자만 받는다	**ignored**
7.		영수증: 허용되지 않은 대,소문자 페이백: 허용된 대문자 소문자	**failed**	영수증자체의 허용되지 않는 대소문자는 구별하지만 역시나 페이벡은 숫자만 받는다	**ignored**
8.		영수증: 허용되지 않은 대,소문자 페이백: 허용되지 않은	**failed**	대소문자는 구별하지만 페이벡은 숫자만 받는다	**ignored**

	범위 내에 있고 DB에 있는 값 이체금액: 이체 범위 안의 값	대문자 소문자			
9.		영수증: 허용되지 않은 대,소문자 페이백: true false 삽입	**failed**	영수증자체의 허용되지 않는 대소문자는 구별하지만 역시나 페이벡은 숫자만 받는다	**ignored**
10.		영수증: 허용되지 않은 대,소문자 페이벡:기타 string	**failed**	영수증자체의 허용되지 않는 대소문자는 구별하지만 역시나 페이벡은 숫자만 받는다	**ignored**
11.		영수증: true false 삽입 시 페이백: 허용된 대문자, 소문자	**failed**	영수증을 true/false로 입력 받는다는 전제가 보고서에 없었고 또한 없을예정이며 페이벡은 숫자만 받는다	**ignored**
12.		영수증: true false 삽입 시 페이벡: 허용되지 않은 대문자, 소문자	**failed**	영수증을 true/false로 입력 받는다는 전제가 보고서에 없었고 또한 없을예정이며 페이벡은 숫자만 받는다	**ignored**
13.		영수증: true false 삽입 시 페이벡: “true”/“false”	**failed**	영수증을 true/false로 입력 받는다는 전제가 보고서에 없었고 또한 없을예정이며 페이벡은 숫자만 받는다	**ignored**
14.		영수증: true false 삽입 시 페이벡: 기타 String	**failed**	영수증을 true/false로 입력 받는다는 전제가 보고서에 없었고 또한 없을예정이며 페이벡은 숫자만 받는다	**ignored**
15.		영수증:기타 string 페이벡: 허용된 대문자, 소문자	**failed**	영수증 자체는 기타 string에 대한 예외처리를 하지만 페이벡은 숫자만	**ignored**

				받는다	
16.		영수증:기타 string 페이벡: 허용되지 않은 대문자, 소문자	**failed**	영수증 자체는 기타 string에 대한 예외처리를 하지만 페이벡은 숫자만 받는다	**ignored**
17.		영수증:기타 string 페이벡: "true"/"false"	**failed**	영수증 자체는 기타 string에 대한 예외처리를 하지만 페이벡은 숫자만 받는다	**ignored**
18.		영수증:기타 string 페이벡: 기타 String	**failed**	영수증 자체는 기타 string에 대한 예외처리를 하지만 페이벡은 숫자만 받는다	**ignored**
19.	(Key = 1.1.1.1.2.0.0.0.0.)		**Passed**		**Passed**
20.	(Key = 1.1.1.1.3.0.0.0.0.)		**Passed**		**Passed**
21.	(Key = 1.1.1.1.4.0.0.0.0.)		**Passed**		**Passed**
22.	(Key = 1.1.1.2.0.0.0.0.0.)		**Passed**		**Passed**
23.	(Key = 1.1.1.3.0.0.0.0.0.)		**Passed**		**Passed**
24.	(Key = 1.1.1.4.0.0.0.0.0.)		**Passed**		**Passed**
25.	(Key = 1.1.1.5.0.0.0.0.0.)		**Passed**		**Passed**
26.	(Key = 1.1.2.0.0.0.0.0.0.)		**Passed**		**Passed**
27.	(Key = 1.1.3.0.0.0.0.0.0.)		**Passed**		**Passed**
28.	(Key = 1.1.4.0.0.0.0.0.0.)		**Passed**		**Passed**
29.	(Key = 1.1.5.0.0.0.0.0.0.)		**Passed**		**Passed**
30.	(Key = 1.2.0.0.0.0.0.0.0.)		**Passed**		**Passed**
31.	(Key = 1.3.0.0.0.0.0.0.0.)		**Passed**		**Passed**
32.	(Key = 1.4.0.0.0.0.0.0.0.)		**Passed**		**Passed**
33.	(Key = 1.5.0.0.0.0.0.0.0.)		**Passed**		**Passed**
34.	출금 관련 로그인 계좌: 범위 이내에 있고, DB에 있는 값 비밀번호: 범위 이내에 있고, DB	영수증: 허용된 대문자 소문자 페이백: 허용된 대문자 소문자	**failed**	페이벡은 숫자만 받는다	**ignored**
35.		영수증: 허용된 대문자 소문자 페이백: 허용되지 않은 대문자 소문자	**failed**	페이벡은 숫자만 받는다	**ignored**
36.		영수증: 허용된 대문자	**failed**	페이벡은 숫자만	**ignored**

	에 있는 값	소문자 페이백: true false 삽입		받는다	
37.	출금 금액: 출금범위 안의 값	영수증: 허용된 대문자 소문자 페이벡: 기타 string	**failed**	페이벡은 숫자만 받는다	**ignored**
38.		영수증: 허용되지 않은 대,소문자 페이백: 허용된 대문자 소문자	**failed**	영수증자체의 허용되지 않는 대소문자는 구별하지만 역시나 페이벡은 숫자만 받는다	**ignored**
39.		영수증: 허용되지 않은 대,소문자 페이백: 허용되지 않은 대문자 소문자	**failed**	영수증자체의 허용되지 않는 대소문자는 구별하지만 역시나 페이벡은 숫자만 받는다	**ignored**
40.		영수증: 허용되지 않은 대,소문자 페이백: true false 삽입	**failed**	영수증자체의 허용되지 않는 대소문자는 구별하지만 역시나 페이벡은 숫자만 받는다	**ignored**
41.		영수증: 허용되지 않은 대,소문자 페이벡:기타 string	**failed**	영수증자체의 허용되지 않는 대소문자는 구별하지만 역시나 페이벡은 숫자만 받는다	**ignored**
42.		영수증: true false 삽입 시 페이백: 허용된 대문자, 소문자	**failed**	영수증을 true/false로 입력 받는다는 전제가 보고서에 없었고 또한 없을예정이며 페이벡은 숫자만 받는다	**ignored**
43.		영수증: true false 삽입 시 페이벡: 허용되지 않은 대문자, 소문자	**failed**	영수증을 true/false로 입력 받는다는 전제가 보고서에 없었고 또한 없을예정이며 페이벡은 숫자만 받는다	**ignored**
44.		영수증: true false 삽입 시 페이벡: "true"/"false"	**failed**	영수증을 true/false로 입력 받는다는 전제가 보고서에 없었고	**ignored**

				또한 없을예정이며 페이벡은 숫자만 받는다	
45.		영수증: true false 삽입 시 페이벡: 기타 String	**failed**	영수증을 true/false로 입력 받는다는 전제가 보고서에 없었고 또한 없을예정이며 페이벡은 숫자만 받는다	**ignored**
46.		영수증:기타 string 페이벡: 허용된 대문자, 소문자	**failed**	영수증 자체는 기타 string에 대한 예외처리를 하지만 페이벡은 숫자만 받는다	**ignored**
47.		영수증:기타 string 페이벡: 허용되지 않은 대문자, 소문자	**failed**	영수증 자체는 기타 string에 대한 예외처리를 하지만 페이벡은 숫자만 받는다	**ignored**
48.		영수증:기타 string 페이벡: "true"/"false"	**failed**	영수증 자체는 기타 string에 대한 예외처리를 하지만 페이벡은 숫자만 받는다	**ignored**
49.		영수증:기타 string 페이벡: 기타 String	**failed**	영수증 자체는 기타 string에 대한 예외처리를 하지만 페이벡은 숫자만 받는다	**ignored**
50.	(Key = 2.1.1.0.0.2.0.0.0.)		**Passed**		**Passed**
51.	(Key = 2.1.1.0.0.3.0.0.0.)		**Passed**		**Passed**
52.	(Key = 2.1.1.0.0.4.0.0.0.)		**Passed**		**Passed**
53.	(Key = 2.1.2.0.0.0.0.0.0.)		**Passed**		**Passed**
54.	(Key = 2.1.3.0.0.0.0.0.0.)		**Passed**		**Passed**
55.	(Key = 2.1.4.0.0.0.0.0.0.)		**Passed**		**Passed**
56.	(Key = 2.1.5.0.0.0.0.0.0.)		**Passed**		**Passed**
57.	(Key = 2.2.0.0.0.0.0.0.0.)		**Passed**		**Passed**
58.	(Key = 2.3.0.0.0.0.0.0.0.)		**Passed**		**Passed**
59.	(Key = 2.4.0.0.0.0.0.0.0.)		**Passed**		**Passed**
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 입금 금액: 입금 범위 안의 값 영수증: 허용된 대문자 소문자	페이벡: 허용되지 않은 대문자, 소문자	**failed**	페이벡은 숫자만 받는다	**ignored**
63.		페이벡: "true"/"false"	**failed**	페이벡은 숫자만 받는다	**ignored**
64.		페이벡: 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**
68.	입금 관련 로그인 계좌: 범위 이내에 있고, DB에 있는 값 입금 금액: 입금 범위 안의 값 영수증: 허용되지 않은 대문자 소문자	페이벡: 기타 string	**failed**	페이벡은 숫자만 받는다	**ignored**
69.	(Key = 3.1.0.0.0.0.1.3.1.)		**passed**		**passed**
70.	(Key = 3.1.0.0.0.0.1.3.2.)		**passed**		**passed**
71.	(Key = 3.1.0.0.0.0.1.3.3.)		**passed**		**passed**
72.	(Key = 3.1.0.0.0.0.1.3.4.)		**passed**		**passed**
73.	(Key = 3.1.0.0.0.0.1.4.1.)		**passed**		**passed**
74.	(Key = 3.1.0.0.0.0.1.4.2.)		**passed**		**passed**
75.	(Key = 3.1.0.0.0.0.1.4.3.)		**passed**		**passed**
76.	(Key = 3.1.0.0.0.0.1.4.4.)		**passed**		**passed**
77.	(Key = 3.1.0.0.0.0.2.0.0.)		**passed**		**passed**

78.	(Key = 3.1.0.0.0.0.3.0.0.)		passed		passed
79.	(Key = 3.1.0.0.0.0.4.0.0.)		passed		passed
80.	(Key = 3.2.0.0.0.0.0.0.0.)		passed		passed
81.	(Key = 3.3.0.0.0.0.0.0.0.)		passed		passed
82.	(Key = 3.4.0.0.0.0.0.0.0.)		passed		passed
83.	(Key = 3.5.0.0.0.0.0.0.0.)		passed		passed

System Action. Test result respond by developer:

첫 번째 테스트 케이스대한 모든 테스트를 통과한 다소 억지스러운 면모의 테스트케이스가 등장하여 pass percentage를 낮췄다. 애초에 첫 번째 테스트에서 설정된 테스트 케이스에 대한 것만을 생각하며 완벽히 수정하였고 두 번째 테스트 케이스가 추가로 등장했지만 우리 컨셉과 맞지 않는 부분들이 전부였으므로 수정하지 않고 그냥 그대로 두었다. (ignore 처리)

System Action. Brute Force Testing

Test Case#	Test Case	Pre-test result	Fix management	Test result
1	페이벡 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 음수값 입력	Pass		Pass
7	100000이상의 입금을 수행하고 DB파일 잔액 변화유무 확인	Pass		Pass
8	DB 파일 한도를 음수값으로 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	입력값이 잘못된 경우에 다음으로 진행하는데 (이미Fail인상태) 그때 영수증 페이벡 입력	Fail	예외처리	Pass
14	수수료 발생 유무 확인	Pass		Pass

System Action. Document

검증팀에서 요구한 문서 수정사항에 대한 대응

1. User 명세에 대하여
   A. 검증팀에서는 User를 Customer와 Admin을 구별하여 명확하게 표기하라고 요구함
   B. 그러나 애당초 Stage 1000에서 User는 Customer 한 명이기 때문에 명세로 표현할 필요가 없음. (Admin 정의 안함)
2. Interaction Diagram에 대하여
   A. 수정되어야 할 부분(선 표시 올바르게)에 대해 요구함
   B. 요구사항대로 수정함
3. 메뉴에 '사용횟수조회' 추가에 대하여
   A. 검증팀에서 메뉴에 사용횟수 조회라는 메뉴를 추가할 것을 요구함
   B. 애초에 사용횟수 조회는 payback에 대한 것으로 시스템 내부적으로 필요한 것이기 때문에 만들 필요가 없다고 판단했다. (Ignore처리)

Static Analyze Action. PMD

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

오류 취합.

1) All class and interfaces have to belong in named package

2) Each class should be declare at least one constructor

3) Avoid unused variables

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

5) Methods should not contain underscores

6) Header comment requirement required

7) To avoid mistakes add a comment at the beginning of the "commission" filed if you want a default access modifier

8) Variables should be start with a lowercase character

9) Use explicit scoping instead of the default package private level

대응 방안

1) 검증팀에서 제공한 환경에서 만들다 보니 디폴트 패키지 안에서 코딩을 하였다. 그러나 기능상의 문제와 단일패키지 이므로 수정할 필요성을 느끼지 못한다

2) 인터넷에 찾아보니 이것은 정적 분석 tool 마다 다르게 잡는다고 나와 있으므로 꼭 필수적인 상황이 아니라고 생각하여 수정하지 않았다.

3) 사용 되고 있지 않은 변수를 찾아 모두 지웠다

4) 클래스 이름과 변수 이름이 대문자 소문자까지 똑 같은 것이 있어 변수를 전부 소문자로 바꾸었다.

```
//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;
```

5) 함수명에 "_" 들어가면 안 된다고 나와있지만 기능상의 문제는 없고 오히려 가독성이 증가한다고 판단하여 수정하지 않았다.

6) 각 클래스에 대한 주석을 추가해달라는 요구였다 따라서 각 클래스마다 주석을 달아주었다.

```
//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;
```

7) 실수를 방지하려 각 클라스 인스턴스를 만들 때 앞에 주석을 달라고 하지만 그럴 필요가 없으므로 수정하지 않았다

8) 변수를 초기에 대문자로 시작하는 것이 많아서 전부다 소문자로 바꾸었다.

```
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);
```

9) getter setter 함수를 만들어서 사용하라는 권유이지만 저 문제에 해당되는 변수들은 딱히 외부에서 접근이 필요 없는 것이기 때문에 수정 하지 않았다.

Static Analyze Action. Sonar Qube

분석 결과:

1) 3개의 버그와 35개의 취약점을 가지고 있다고 분석 되었으며 코드의 중복도 그다지 높지 않다는 것을 확인하였다.

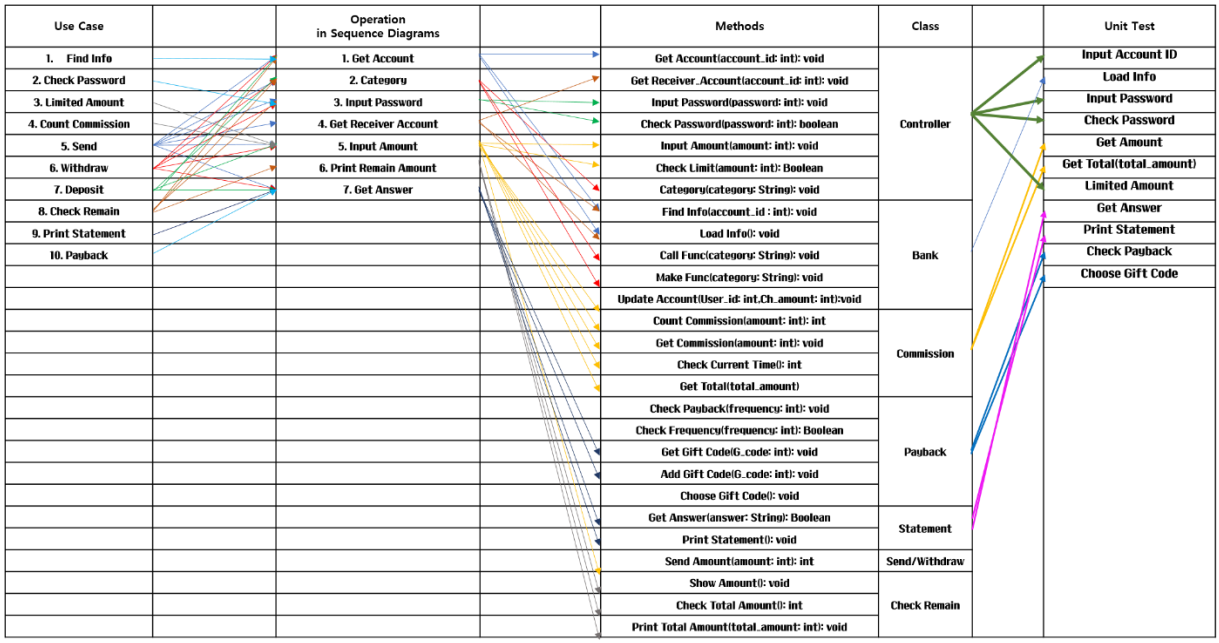
Bugs	Weakness	New Bugs	New Weakness
3	35	3	35

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

대응 방안

1) 어떤 것이 버그이고 어떤 곳이 취약한지에 대한 피드백을 받지 못하여 어느 곳이 버그를 유발하는지 찾지 못하였다.

Final Trace Ability



After Thoughts. 소감

T7. 문기태

프로그램 개발측면) 먼저 ATM이라는 소프트웨어가 이렇게 고려요소와 생각해야 될 사항들이 많은 줄 몰랐다. Stage 1000부터 사용자부터 정의하여 Use Case를 만드는 단계에서 사용자 측면에서 만드는 소프트웨어의 중요성과 세심함에 대해서 깨닫게 되었다. 그리고 Stage1000부터 분석하고 디자인을 하다 보니 실제 구현단계에서 바로 만들 때보다 수월하고 짜임새 있게 구성할 수 있다는 것을 깨달았다. 많은 클래스가 있어도 이미 정의한 Use Case와 Interaction Diagram으로 전체적인 파악이 가능했고 그들이 어떻게 상호작용하는지 한 눈에 알 수 있어서 좋았다.

팀원과의 교류 측면) 모든 팀들이 3~4인으로 구성된 반면 3인팀에서 1명이 중도 휴학하는 바람에 2인팀으로 해결하면서 다른 팀들보다 많은 어려움이 있었다. 그 과정에서 인력의 중요성도 알 수 있었고 프로그램의 개발에서는 많은 사람의 피드백과 도움이 필요하다는 것을 절실하게 깨달았다. 또한 팀원이 단계 도중에서 몇 일간 연락을 받지 않거나 포기하고 싶다며 상호교류가 잘 되지 않았을 때 스스로 만들어내면서 어려움이 있었지만 다시 설득하고 같이 하는 과정에서 무엇보다 개발하는데 있어 팀원들과의 마음도 맞아야 하는게 중요하다는 것을 깨닫게 되었다.

개발시간측면) 시간이 빠듯하긴 했다. 그렇다 보니 천천히 고민해보고 여러가지 상황을 고려해볼 수 있는 시간이 많이 주어지지 않아 아쉽긴 했다. 그러나 오히려 촉박한 시간으로 빠르게 생각하고 판단하는 능력이 많이 길러졌다고 생각한다.

T7. 한상민

프로그램 개발 측면) OOPT라는 거 자체가 1000단계부터 마지막까지 쭉 훑어보고 다시 원상태로 되돌아 와서 수정해 나가며 다시 진행하는 것으로 알고 있다. 하지만 대학교 수업이라는 한계 때문에 대략 4달동안 waterfall과 비슷한 방식으로 진행이 된 거 같아 아쉬움 감이 남아있다. 또한 수업을 진행하면서 어떠한 결과를 만들자고 할 때 확실히 사전에 밑바탕으로 기초공사를 해놓고 그 위에서 일을 진행하는 것과 한 번에 그냥 맨땅에서 일을 진행 했을 때의 차이점을 알게 됐다

팀원과의 교류 측면) 초반에 3명이었던 조가 한 명이 개인적인 사정으로 조를 나가게 되면서 많이 힘들지 않을까 하는 예상이 들었다. 하지만 의외로 철저한 분업화로 인한 일의 진행에 있어서의 편리성이 생겨서 생각보다 그리 고전하지는 않았던 것 같았다. 하지만 이렇게 철저한 분업화가 이루어지다 보니 한쪽이 다른 상대방의 일의 결과에 대해서 100%이해하지 못하고 그 다음 단계로 넘어가는 일이 발생하였으며. 서로의 결과에 대한 자체 피드백이 없었다. 시간이 좀 더 주어졌으면 서로 이해하고 훨씬 양질의 결과를 생산 할 수 있지 않았을까 하는 아쉬움이 남는다.

개발시간 측면) 솔직히 처음부터 구현을 하라고 했으면 더 크고 방대한 기능의 atm을 만들 수 있을 거 같지만 이 소프트웨어 모델링의 수업이 결과에 대한 질을 따지는게 아니라 그 과정에 있어서 흐름이 어떻게 흘러가는지에 대한 것을 배우는 과목이라 생각하고 있다. 따라서 이러한 간단한 기능의 ATM을 만드는 것 마저도 4달이라는 시간이 걸리지 않았나라는 생각이 든다.