JackPot ATM
OOPT 3rd Cycle

2018.06.11

Team 5

201112052 방민석

201312259 백만일

201211383 조영래

목차

1. System Testing Response - Stage 1000. Modified Part

1.1. Stage 1000 Planning1001. Define Draft Plan
2. Project Objective
- Jackpot 기능의 악용가능성 (ex. 한사람이 만원씩 지속적으로 출금)을 제거
-> 이후 단계에서 관련 명세가 존재하지 않음
이후 단계에서 시간 차가 있는 출금만 카운트하거나,
5 만원 이상 출금만 카운트하는 식 등의 명확한 방법으로 명세 추가 필요

Modified : 1일 1/1000 의 확률로 Jackpot 당첨이 가능하며, 수수료 1,200원을 도입하여 지속 출금 50번만 하여도 Jackpot 으로 얻을 수 있는 수익보다 ATM 수익이 높아지기 때문에 ATM 기기 입장에서 이득을 얻을 수 있다고 판단하여 추가로 기능구현을 하지 않았으며, 이에따라 문서를 수정하였음.

1. System Testing Response - Stage 2030. Modified Part

2031. Define Essential Use Cases
2010 까지는 존재하는 '수수료'와 '명세서 출력'에 대한 서술이 모두 사라짐
1. Withdraw
- "ATM의 잔고와 출금 금액을 비교한다"는 내용 추가해야 함
- Typical Courses of Events 항목에 13번이 두 개임.
- 수수료에 대한 서술 추가 필요

Modified : 수수료에 대한 내용 모두 추가, 명세서에 대한 내용은 1000 부터 제거함을 2nd cycle 발표에서 공지

Typical Courses of Events	(A):Actor, (S):System 1.(A) User가 출금 버튼을 누른다. 2.(S) 계좌 입력 화면을 출력한다. 3.(A) User는 계좌를 입력한다. 4.(S) 유효한 계좌인지 확인한다. 5.(S) 출금하고자 하는 금액 입력 화면을 출력한다. 6.(A) 출금하고자 하는 금액을 입력한다. 7.(S) 비밀번호 입력 화면을 출력한다. 8.(A) 비밀번호를 입력한다. 9.(S) 비밀번호가 일치하는지 확인한다. 10(S) ATM 잔고와 출금금액을 비교한다. 11.(S) User의 잔고와 출금금액+수수료 를 비교한다. 12.(S) User의 잔고를 출금금액+수수료만큼 감소시킨다. 13.(S) ATM 잔액을 감소시킨다.

1. System Testing Response - Stage 2030. Modified Part

2034. Refine Glossary
- 2040단계에서 View의 부재로 인해 GUI_Interface 관련 항목 수정 필요

Modified : 이 단계는 View(GUI 에 대한것을 고려하지 않고 시스템 내부적으로 사용자와 interaction 만을 고려하며 뒤의 2040 GUI 를 다루었기 때문에 변경된 클래스 네임만을 반영

수정 전 :

GUI_Interface	Class	사용자로부터 메뉴를 입력 받아 처리하고 컨트롤 하는 클래스

수정 후 :

System_control	Class	사용자로부터 메뉴를 입력 받아 처리하고 컨트롤 하는 클래스

1. System Testing Response - Stage 2030. Modified Part

2038. Refine System Test Case
- 1009 에서 정의한 System test case 와 항목 및 내용이 완전히 달라짐
-> 관련된 test case 끼리 묶고 넘버링 필요

Modified : Refine System Test Case 이기 때문에 기존 1000 단계에서 구체적이지 못한 항목에서 구체적으로 분석된 프로그램을 기반으로 새롭게 테스트 케이스를 변경하고 관련된 test case 는 N-1, N-2… 로 명시하여 묶음

1-1	input_menu Test	사용자가 입력한 해당 메뉴 화면으로 올바르게 진입하는지 확인한다.	R1.2 ,R.2, R.3, R.4, R.5
2-1	input_ID	사용자가 입력한 계좌 혹은 카드번호가 유효하지 않으면 오류메시지를 출력하는지 확인한다.	R1.2, R.2,R.3,R.4
2-2	input_ID	사용자가 입력한 계좌 혹은 카드번호가 유효하면 다음 단계로 넘어가는지 확인한다.	R1.2, R.2,R.3,R.4
3-1	input_PW	사용자가 입력한 비밀번호가 계좌번호의 비밀번호가 아니면 오류메시지를 출력하는지 확인한다	R1.2, R.3, R.4

1. System Testing Response - Stage 2040. Modified Part

2041. Design Real Use Cases
3. remittance
- Exceptional Courses of Events 에 2030 단계에서 존재하는
"E5: 자기 자신의 계좌에는 송금할 수 없다." 는 내용 누락.

Modified : 구현상에서 추가된 예외 상황에 대한 명시 추가

Exceptional Courses of Events	E1: 잘못된 계좌를 입력받으면 에러 메시지를 출력한다 E2: 비밀번호가 틀릴 경우 에러 메시지를 출력한다 E3: 송금 계좌가 잘못된 경우 에러 메시지를 출력한다 E4: 계좌 잔고가 송금액보다 적을 경우 에러 메시지를 출력한다 E5: 자기 자신의 계좌에는 송금할 수 없다

1. System Testing Response - Stage 2050. Modified Part

2051. Implement Class & Method Definitions
Class Definitions – Manager
1. Manager.checkID
- Abstract Operation 오타: inputMID > inputID, ManagerIID > ManagerID
2. Manager.CheckPW
- Abstract Operation 오타 : inputMPW > inputPW

Response : MID,MPW는 오타가 아닌 매니저의 정보임을 나타내기 위해 M을 붙여 표현한 것이므로 수정하지 않음

1. System Testing Response - Stage 2050. Modified Part

Class Definitions – Bank
1. SearchID
- 명세와 소스코드의 output 값의 자료형 불일치
2. getAccountBalance
- 명세와 소스코드의 output 값의 자료형 불일치

Modified : 소스코드와 일치하도록 문서상의 내용들을 수정함

Type	Method
Name	searchID
Purpose	Bank 내부에 개설된 통장 혹은 카드의 정보를 찾는 메소드
Overview	N/A
Cross Reference	withdraw, deposit,remittance, view_account_detail
Input	inputID:int
Output	int
Abstract Operation	Account 에 있는 checkID 를 호출
Exceptional Course of Event	N/A

1. System Testing Response - Stage 2050. Modified Part

Class Definitions – Account
1. Account.checkID
- Abstract Operation 에서 '찾음'이라는 표현이 모호함.
2. searchCardID
- Abstract Operation 에서 '찾음'이라는 표현이 모호함.

Modified : '찾음' 이라는 표현을 함수의 기능에 따라 좀 더 구체적으로 수정함

Type	Method
Name	Account.checkID
Purpose	사용자로부터 입력된 ID 계좌번호를 찾는 메소드
Overview	N/A
Cross Reference	withdraw, deposit,remittance, view_account_detail
Input	inputID: int
Output	int
Abstract Operation	inputID와 일치하는 계좌번호를 찾은 경우와 못 찾은 경우 각각 다른 값 반환
Exceptional Course of Event	N/A

1. System Testing Response - Stage 2050. Modified Part

Class Definitions – Account
3. balanceAccount
- Abstract Operation 에서 '변화시킨다'는 표현이 증가인지 감소인지 불분명

Response : input으로 받는 inputMoney가 양수와 음수 모두 들어올 수 있기에 불분명하지 않다고 판단해서 수정하지 않음

1. System Testing Response – Brute Force Testing

4.1. Testing Result

Test Suite	Test Case	Result
Manager	관리자 프로세스에서 5번 반복하여 입/출금 할 수 있다.	Passed
Manager	관리자 프로세스에서 10번 반복하여 입/출금 할 수 있다.	Passed
User	프로그램 초기 실행 시 난수가 제대로 생성된다.	Passed
User	출금 10번 이내에 Jackpot이 터지는 지 확인한다.	Passed
User	Jackpot이 터지면 해당 User에게 5만원을 추가 인출하며 ATM의 잔고에서 5만원이 빠져나간다.	Passed
User	1 일 1 명만 Jackpot 이 터진다.	Passed
User	출금 시에만 Jackpot 이 터진다.	Passed
User	하나의 통장으로 여러 번 출금할 시 Jackpot 당첨이 여러 번 되지 않게 한다.	Failed
User	모든 입력에 대한 기기의 반응은 1초 이내로 이루어진다.	Passed
User	거래는 1분 이내에 이루어져야 한다.	Passed
User	사용자가 보기 편한 화면을 제공해야 한다.	Passed
User	세 번 이상 연속적으로 출금할 수 있다.	Passed
User	세 번 이상 연속적으로 입금할 수 있다.	Passed
User	세 번 이상 연속적으로 송금할 수 있다.	Passed
User	세 번 이상 연속적으로 조회할 수 있다.	Passed
User	출금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Passed
User	입금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Passed
User	송금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Passed
Manager	ATM 잔고가 일정수준(50 만원) 이하가 되면 잔고부족 메시지를 출력한다.	Passed
User	출금 시 수수료 1200 원을 더하여 출금한다.	Passed
User	송금 시 수수료 1200 원을 더하여 출금한다.	Passed
User	입금 시 ATM 잔고 지폐 별 500 개를 초과하면 오류 메시지를 출력한다.	Passed
User	입금 시 계좌 잔고 10 억원을 초과하면 오류 메시지를 출력한다.	Passed
User	송금 시 상대방 계좌 잔고 10 억원을 초과하면 오류 메시지를 출력한다.	Failed
User	Jackpot이 터질 때 ATM 잔고가 부족하면 사용자의 계좌로 5만원을 입금하고 알림 메시지를 출력한다.	Passed
User	복수의 화폐에서 최대치(500 매)를 넘어선 매수를 입금하려고 하면, 초과된 모든 지폐에 대해 오류 메시지를 출력한다.	Failed
Manager	복수의 화폐에서 최대치(500 매)를 넘어선 매수를 입금하려고 하면, 초과된 모든 지폐에 대해 오류 메시지를 출력한다.	Failed

23/27 = 85% Pass

문서 변경으로 인하여 이전 Case에서 8개 추가됨

1. System Testing Response – Brute Force Testing

Test Suite	Test Case	How to fix	Developer Result
User	여러 화폐의 최대 수량(500)을 동시에 넘었을 때, 복수의 오류 메시지를 출력하지 않는다	각각의 화폐 수량 초과에 대하여 오류메시지를 출력하도록 변경	pass
User	같은 계좌로 출금을 여러 번 반복했을 때, 다시 Jackpot이 터진다.	잭팟 당첨 횟수에 대한 제한은 없으므로 같은 계좌라고 하더라도 당첨 당일만 아니라면 당첨이 가능함-> 수정하지 않음	예외
User	송금 시 상대방 계좌 잔고 10억원을 초과하면 잘못된 오류 메시지가 출력된다.	송금시, 입금계좌 10억 초과시 "일반계좌 최대 입금한도는 10억입니다" 메시지를 출력	pass
Manage	여러 화폐의 최대 수량(500)을 동시에 넘었을 때, 복수의 오류 메시지를 출력하지 않는다	각각의 화폐 수량 초과에 대하여 오류메시지를 출력하도록 변경	pass

- Failed한 4가지 case에 대해서 3개 수정, 1개 수정 X
- 코드 수정 후 다른 모든 test case도 다시 pass 여부 확인 완료

1. System Testing Response – Brute Force Testing

4.1. Testing Result

Test Suite	Test Case	Result
Manager	관리자 프로세스에서 5번 반복하여 입/출금 할 수 있다.	Passed
Manager	관리자 프로세스에서 10번 반복하여 입/출금 할 수 있다.	Passed
User	프로그램 초기 실행 시 난수가 제대로 생성된다.	Passed
User	출금 10번 이내에 Jackpot이 터지는 지 확인한다.	Passed
User	Jackpot이 터지면 해당 User에게 5만원을 추가 인출하며 ATM의 잔고에서 5만원이 빠져나간다.	Passed
User	1일 1명만 Jackpot이 터진다.	Passed
User	출금 시에만 Jackpot이 터진다.	Passed
User	하나의 통장으로 여러 번 출금할 시 Jackpot 당첨이 여러 번 되지 않게 한다.	제외
User	모든 입력에 대한 기기의 반응은 1초 이내로 이루어진다.	Passed
User	거래는 1분 이내에 이루어져야 한다.	Passed
User	사용자가 보기 편한 화면을 제공해야 한다.	Passed
User	세 번 이상 연속적으로 출금할 수 있다.	Passed
User	세 번 이상 연속적으로 입금할 수 있다.	Passed
User	세 번 이상 연속적으로 송금할 수 있다.	Passed
User	세 번 이상 연속적으로 조회할 수 있다.	Passed
User	출금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Passed
User	입금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Passed
User	송금 시 금액 입력 값에 minus 정수를 입력하면 다음 단계로 넘어가지 않거나, 오류 메시지가 출력된다.	Passed
Manager	ATM 잔고가 일정수준(50만원) 이하가 되면 잔고부족 메시지를 출력한다.	Passed
User	출금 시 수수료 1200원을 더하여 출금한다.	Passed
User	송금 시 수수료 1200원을 더하여 출금한다.	Passed
User	입금 시 ATM 잔고 지폐 별 500개를 초과하면 오류 메시지를 출력한다.	Passed
User	입금 시 계좌 잔고 10억원을 초과하면 오류 메시지를 출력한다.	Passed
User	송금 시 상대방 계좌 잔고 10억원을 초과하면 오류 메시지를 출력한다.	Passed
User	Jackpot이 터질 때 ATM 잔고가 부족하면 사용자의 계좌로 5만원을 입금하고 알림 메시지를 출력한다.	Passed
User	복수의 화폐에서 최대치(500매)를 넘어선 매수를 입금하려고 하면, 초과된 모든 지폐에 대해 오류 메시지를 출력한다.	Passed
Manager	복수의 화폐에서 최대치(500매)를 넘어선 매수를 입금하려고 하면, 초과된 모든 지폐에 대해 오류 메시지를 출력한다.	Passed

1. System Testing Response – Category-partition Testing

Test Suite	Test Case	Latest Exec Result
User	atm-5:Test Case 1	Passed
User	atm-6:Test Case 2	Passed
User	atm-7:Test Case 3	Passed
User	atm-8:Test Case 4	Passed
User	atm-9:Test Case 5	Passed
User	atm-10:Test Case 6	Passed
User	atm-11:Test Case 7	Passed
User	atm-12:Test Case 8	Passed
User	atm-13:Test Case 9	Passed
User	atm-14:Test Case 10	Passed
User	atm-15:Test Case 11	Passed
User	atm-16:Test Case 12	Passed
User	atm-17:Test Case 13	Passed
User	atm-18:Test Case 14	Passed
User	atm-19:Test Case 15	Passed
User	atm-20:Test Case 16	Passed
User	atm-21:Test Case 17	Passed
User	atm-22:Test Case 18	Passed
User	atm-23:Test Case 19	Passed
User	atm-24:Test Case 20	Passed
User	atm-25:Test Case 21	Passed
User	atm-26:Test Case 22	Passed
User	atm-27:Test Case 23	Passed
User	atm-28:Test Case 24	Passed
User	atm-29:Test Case 25	Passed
User	atm-30:Test Case 26	Passed
User	atm-31:Test Case 27	Passed
User	atm-32:Test Case 28	Passed
User	atm-33:Test Case 29	Passed
User	atm-34:Test Case 30	Passed
User	atm-39:Test Case 31	Passed
User	atm-40:Test Case 32	Passed
Manager	atm-41:Test Case 1	Passed
Manager	atm-42:Test Case 2	Passed
Manager	atm-43:Test Case 3	Passed
Manager	atm-44:Test Case 4	Passed
Manager	atm-45:Test Case 5	Passed
Manager	atm-46:Test Case 6	Passed
Manager	atm-47:Test Case 7	Passed

64/64 = 100% Pass

- 64가지 모든 case에 대해 pass, 수정 진행하지 않음

1. System Testing Response

- Category Partition Test

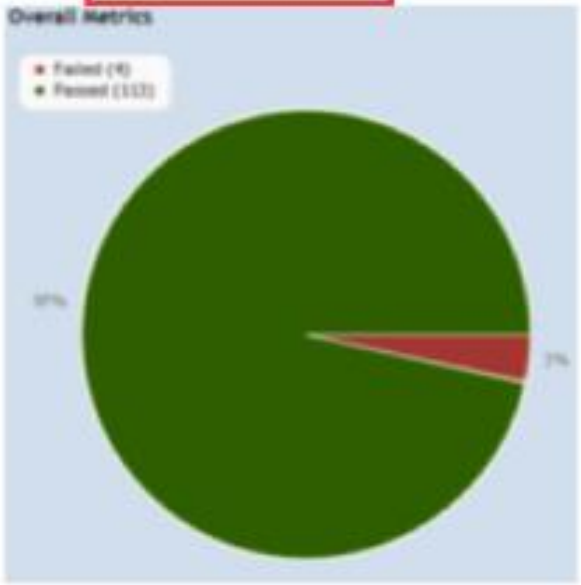
  64/64 = 100% Pass

- Pairwise Test

  26/26 = 100% Pass

- Brute Force Test

  23/27 = 85% Pass



-Category Partition Test

64/64 = 100% Pass

-Pairwise Test

26/26 = 100% Pass

-Brute Force Test

26/26 = 100% Pass

Overall Metrics(116)

2. Static Analysis Response

- 설명 주석 내용이 비어 있음. 의미 없는 주석 과다

```
8 : public class Manager {
```
[JAVA_71] 클래스의 설명 주석이 없거나, 설명 주석 내용에 모든 태그가 포함되어있지 않음

[OBJ04-J] mutable 클래스가 신뢰할 수 없는 코드에 객체를 안전하게 전달하기 위한 복사 기능을 제공하지 않음.

```
9 :
10 :     /**
11 :      * Default constructor
12 :      */
13 :     public Manager(int id, int pw) {
```
[JAVA_72] 메서드의 설명 주석이 없거나, 설명 주석 내용에 모든 태그가 포함되어있지 않음

```
14 :             ManagerID=id;
15 :             ManagerPW=pw;
16 :     }
17 :
18 :     /**
19 :      *
20 :      */
21 :     private int ManagerID;
```
[EGOV_26_VariableNamingConventions] 필드의 이름(ManagerID)이 규칙(^[a-z]([a-z]|[A-Z]|[0-9])*)에 맞지 않음.

[Sun_20] 필드의 이름(ManagerID)이 규칙(^[a-z]([a-z]|[A-Z]|[0-9])*)에 맞지 않음.

[Sun_05] 필드의 선언문이 메서드 선언문보다 코드상 뒤 쪽에 존재함

```
22 :
23 :     /**
24 :      *
25 :      */
26 :     private int ManagerPW;
```
[EGOV_26_VariableNamingConventions] 필드의 이름(ManagerPW)이 규칙(^[a-z]([a-z]|[A-Z]|[0-9])*)에 맞지 않음.

[Sun_20] 필드의 이름(ManagerPW)이 규칙(^[a-z]([a-z]|[A-Z]|[0-9])*)에 맞지 않음.

[Sun_05] 필드의 선언문이 메서드 선언문보다 코드상 뒤 쪽에 존재함

```
27 :
28 :
29 :     /**
30 :      * @param inputID
31 :      * @return
32 :      */
33 :     public Boolean checkID(int inputID) {
```
[JAVA_72] 메서드의 설명 주석이 없거나, 설명 주석 내용에 모든 태그가 포함되어있지 않음

Modified :
Code generation 후 건드릴 필요가 없다고 생각해서 두었던 부분.
의미 없는 주석 제거함

2. Static Analysis Response

수정전 :

```
/**
 *
 */
private int TotalMoney;

/**
 *
 */
private int inputID;

/**
 *
 */
private int inputPW;

/**
 *
 */
private int inputMoney;

/**
 *
 */
private int UserCount;

/**
 *
 */
private static int inputMenu;

/**
 *
 */
private int inputPOM;

/**
 *
 */
private int input50000;
```

수정후:

```
private int TotalMoney;

private int inputID;

private int inputPW;

private int inputMoney;

private int UserCount;

private static int inputMenu;

private int inputPOM;

private int input50000;

private int input10000;

private int input5000;
```

2. Static Analysis Response

- non-static 필드가 public 으로 선언

```
132 :     public Set<Card> Have;
```

[EGOV_26_VariableNamingConventions] 필드의 이름(Have)이 규칙(^[a-z]([a-z]|[A-Z]|[0-9])*)에 맞지 않음.

[Sun_20] 필드의 이름(Have)이 규칙(^[a-z]([a-z]|[A-Z]|[0-9])*)에 맞지 않음.

[Sun_05] 필드의 선언문이 메서드 선언문보다 코드상 뒤 쪽에 존재함

[Sun_22] non-static 필드가 public으로 선언.

Modified : 사용되지 않는 필드라서 제거

수정전:

```
private Card currentcard;
/**
 *
 */
public Set<Card> Have;

private int Code;
```

수정후 :

```
private Card currentcard;

private int code;
```

2. Static Analysis Response

- Exception catch & printStackTrace 사용
  (시스템 내부 데이터나 디버깅 관련 정보가 공개되면 공격의 빌미가 될 수 있다.
따라서 예외 발생 시 스택 정보를 출력하지 말아야 한다.)

```
16 :          } catch (Exception e) {
```
[ERR08-J] (Exception)을 catch함.
```
17 :                  e.printStackTrace();
```
[JAVA_44] 메서드 (printStackTrace)가 사용됨.

Modified : 스택 정보를 출력하지 않도록 코드를 수정

수정전 :

```
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

수정후 :

```
} catch (IOException e) {
}
```

2. Static Analysis Response

- switch 문은 두 개 이상의 case 를 가져야 함
  이 경우 if 문을 사용할 것을 권장함

```
122         switch (inputMenu) {
123         case 5:
124             if(manager.checkID(input)) {
125                 return true;
126             }
127             return false;
128
129         default:
130             for(int index=0; index<bank.size(); index++) {
131                 int check=bank.get(index).searchID(input,1);
132                 if(check==1) {
133                     currentbank=bank.get(index);
134                     return true;
135                 }
136             }
137             break;
138         }
```

Modified : case 문을 if 문으로 변경

수정전 :

```
switch (inputMenu) {

case 5:
    if(manager.checkID(input)) {
        return true;
    }
    return false;

default:
    for(int index=0; index<bank.size(); index++) {
        int check=bank.get(index).searchID(input,1);
        if(check==1) {
            currentbank=bank.get(index);
            return true;
        }
    }
    break;
}
```

수정후 :

```
if(inputMenu==5) {
    if(manager.checkID(input)) {
        return true;
    }
    return false;
}
else {
    for(int index=0; index<bank.size(); index++) {
        int check=bank.get(index).searchID(input,1);
        if(check==1) {
            currentbank=bank.get(index);
            return true;
        }
    }
}
```

2. Static Analysis Response

- identifier(frame)가 이미 필드로 선언되어 있어 hide 가 발생함

```
6 :     private static MainFrame frame;
7 :     public static void main(String[] args) {
```

[JAVA_72] 메서드의 설명 주석이 없거나, 설명 주석 내용에 모든 태그가 포함되어있지 않음

```
8 :         // TODO Auto-generated method stub
9 :
10 :        try {
11 :            Main_controller main = new Main_controller();
12 :            MainFrame frame = main.getMainFrame();
```

[Sun_09] identifier(frame)가 이미 필드로 선언되어 있어 hide가 발생함

[Sun_23] 클래스 object를 이용하여 static 멤버에접근.

[JAVA_70] identifier(frame)가 이미 필드로 선언되어 있어 hide가 발생함

Modified : 이름이 같은 상단의 frame 변수를 제거

수정전 :

```
public class Main_controller {
    private static MainFrame frame;
    public static void main(String[] args) {

        try {
            Main_controller main = new Main_controller();
            MainFrame frame = main.getMainFrame();
            frame=new MainFrame();
            frame.setVisible(true);
```

수정후 :

```
public class Main_controller {
    public static void main(String[] args) {

        try {
            MainFrame frame=new MainFrame();
            frame=new MainFrame();
            frame.setVisible(true);
```

2. Static Analysis Response

```
public Boolean ATMBalanceUpdate(int inputMoney, int inputcount, int inputPOM) {
    // TODO implement here
    switch (inputMoney) {
    case 50000:
        if(inputPOM < 0) {
            if(Count50000>=inputcount) {
                Count50000+=inputcount*inputPOM;
                return true;
            }
            else return false;
        }
        else {
            Count50000+=inputcount*inputPOM;
            return true;
        }
    case 10000:
        if(inputPOM < 0) {
            if(Count10000>=inputcount) {
                Count10000+=inputcount*inputPOM;
                return true;
            }
            else return false;
        }
        else {
            Count10000+=inputcount*inputPOM;
            return true;
        }
    case 5000:
        if(inputPOM < 0) {
            if(Count5000>=inputcount) {
                Count5000+=inputcount*inputPOM;
                return true;
            }
            else return false;
        }
        else {
            Count5000+=inputcount*inputPOM;
            return true;
        }
    case 1000:
        if(inputPOM < 0) {
            if(Count1000>=inputcount) {
                Count1000+=inputcount*inputPOM;
                return true;
            }
            else return false;
        }
        else {
            Count1000+=inputcount*inputPOM;
            return true;
        }
    default:
        break;
    }
    return false;

}
```

Feedback
- 함수는 최소의 exit point를 가져야 함

Response
- 1000, 5000, 10000, 50000원을 모두 다루어야 하기에 해당 코드가 최소한의 exit point를 가지고 있다고 생각하여 수정하지 않음

2. Static Analysis Response

2. 정적 분석 후 추가된 일감(Redmine)

일감

새 일감만들기

검색조건

상태 진행중

유형 이다 Bug

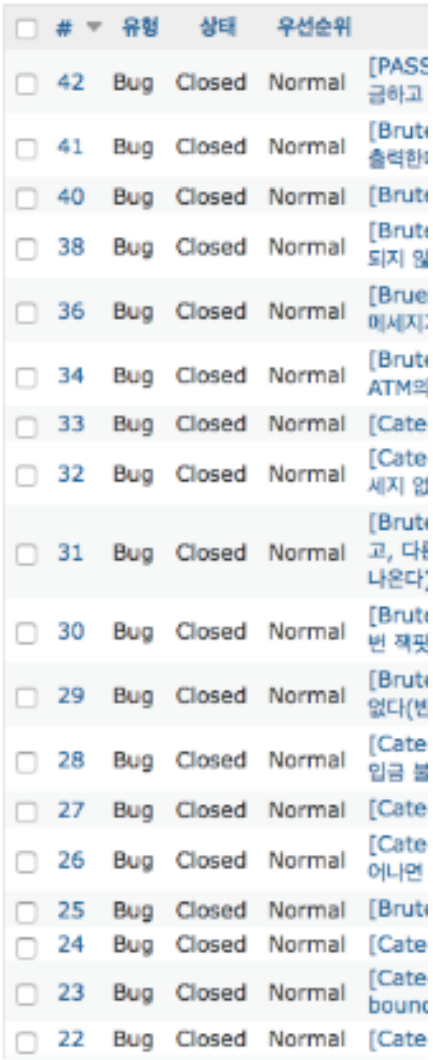
검색조건 추가

옵션

적용 지우기 저장

#	유형	상태	우선순위	제목	담당자	변경
42	Bug	New	Normal	[PASS] Jackpot이 터질 때 ATM 잔고가 부족하면 사용자의 계좌로 5만원을 입금하고 알림 메시지를 출력한다.		2018/06/01 18:16
41	Bug	New	Normal	[Brute Force Test] 송금 시 상대방 계좌 잔고 10억원을 초과하면 오류 메시지를 출력한다.		2018/06/01 18:01
40	Bug	New	Normal	[Brute Force Test] Jackpot이 1일 1명만 터지지 않는다 >>> PASS		2018/06/02 02:30
38	Bug	New	Normal	[Brute Force Test]하나의 통장으로 여러 번 출금할 시 Jackpot 당첨이 여러 번 되지 않게 한다.		2018/06/02 02:28
36	Bug	New	Normal	[BrueForce] 입금, 관리자 모드에서 여러 화폐의 최대 수량을 넘어섰을 때 에러 메세지가 하나만 뜬다		2018/06/02 01:18

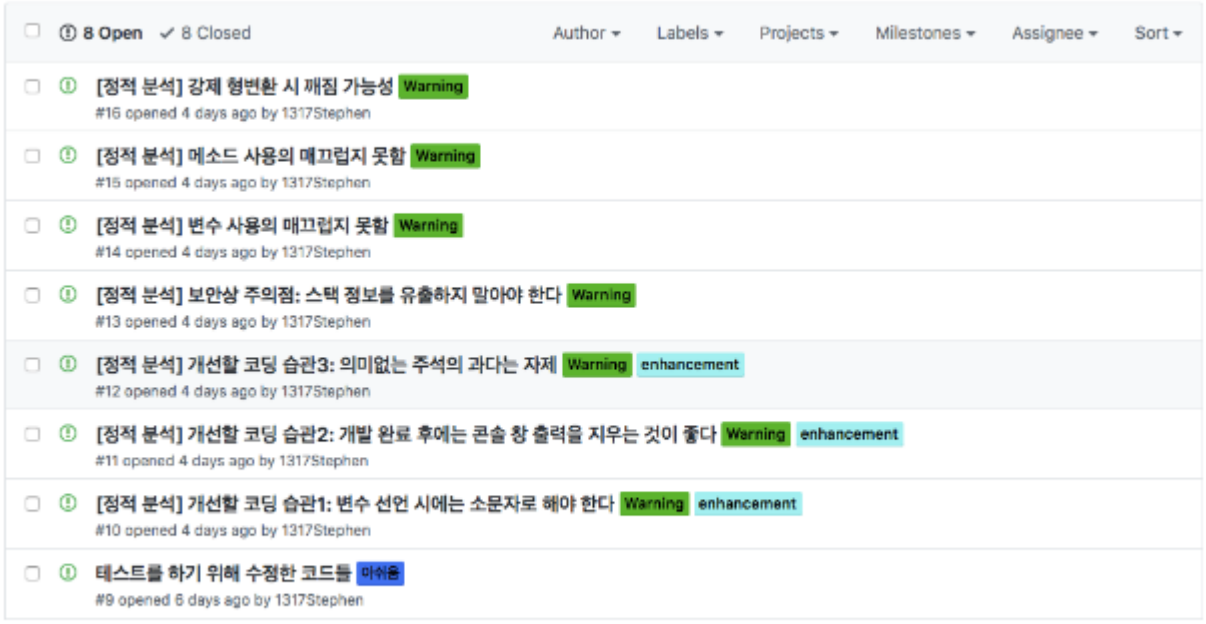
반영하여 총 42개 버그에 대한 수정이 진행되었으며 Closed 상태로 변경되었음을 확인

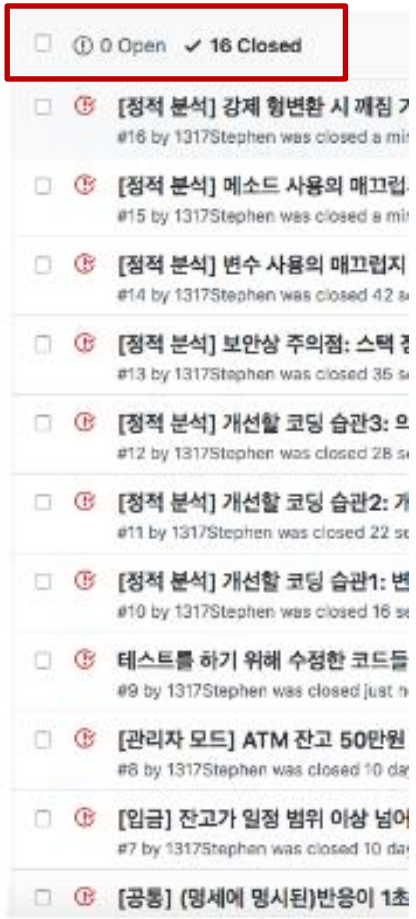


#	유형	상태	우선순위	
42	Bug	Closed	Normal	[PASS 금하고
41	Bug	Closed	Normal	[Brut 출력한
40	Bug	Closed	Normal	[Brut
38	Bug	Closed	Normal	[Brut 되지 않
36	Bug	Closed	Normal	[Brue 메세지
34	Bug	Closed	Normal	[Brut ATM의
33	Bug	Closed	Normal	[Cate
32	Bug	Closed	Normal	[Cate 세지 않
31	Bug	Closed	Normal	[Brut 고, 다 나온다
30	Bug	Closed	Normal	[Brut 번 잭팟
29	Bug	Closed	Normal	[Brut 없다(번
28	Bug	Closed	Normal	[Cate 입금 불
27	Bug	Closed	Normal	[Cate
26	Bug	Closed	Normal	[Cate 어니면
25	Bug	Closed	Normal	[Brut
24	Bug	Closed	Normal	[Cate
23	Bug	Closed	Normal	[Cate boun
22	Bug	Closed	Normal	[Cate

2. Static Analysis Response

3. 정적 분석 후 추가된 일감(Github)





반영하여 총 16개 버그에 대한 수정이 진행되었으며 Closed 상태로 변경되었음을 확인

3. Opinion

- 팀워크
- 의사소통
- OO 방법론
- 새로운 도구

- 팀원들과 소통하며 맡은 역할을 책임감 있게 수행
- OO 방법론을 심도 있게 배우고 적용할 수 있었던 좋은 기회
- 새로운 도구(ex. starUML, JUNIT, swing window builder 등..)를 사용하며 익힐 수 있었음

Q & A