Payback ATM

Team: T7
Name: 문기태, 한상민

CHART


Analyze Brute Force Testing Reports
Modified factors by Brute Testing Reports
Modified Files
Implementation

Analyze Brute Force Testing Reports

4.2 Failed Case Report

Num	Report
1	페이백 번호를 어떻게 설정해도 페이백 종류가 일정하다.
2	페이백 종류를 안넣어도, 페이백 종류 2가 아닌 다른 수를 넣어도 2로 고정된다
3	페이백의 조건은 출금/송금 거래시 인데, 잔액조회 이후에도 페이백 여부를 물어봄
4	페이백으로 제시된 번호 이외의 숫자를 넣어도 에러출력 없이 거래 종료
5	영수증 출력 옵션(y/n)에 따른 결과 변화가 없음
6	로그인 계좌와 송금할 계좌가 같은 경우의 예외처리가 안되어있음
7	출금/송금/입금 거래시 음수값 입력에 대한 예외처리가 안되어있음
8	DB내의 한도값이 음수인 경우에 대한 예외처리가 안되어있음
9	영수증 출력 옵션/페이백 번호 중 하나만 입력되어도 거래가 정상종료된것으로 처리
10	계좌번호 검색 알고리즘에 문제가 있는 것으로 보임
11	수수료 기능 미구현으로 보임
12	입금 거래시 5자리 이상의 큰 값이 들어올 경우 DB에서 잔액의 변화가 없음
13	Actor가 Admin인 상태가 구현이 안되어, DB를 직접 조작하지 않는 한 사용자들 한도 변경 불가능 -> 1억 이상의 잔고를 가져도 한도가 0이면 거래 불가능
14	이전 실행에서의 입력사항(String)값의 찌꺼기가 다음 거래에도 남아있음
15	데이터의 저장 및 변화가 이루어지지 않음.
16	문서상 분류했던 4개의 은행이 아닌 2개의 은행에 대한 DB만 존재함

4 Brute Force Testing Report

4.1 Testing Result

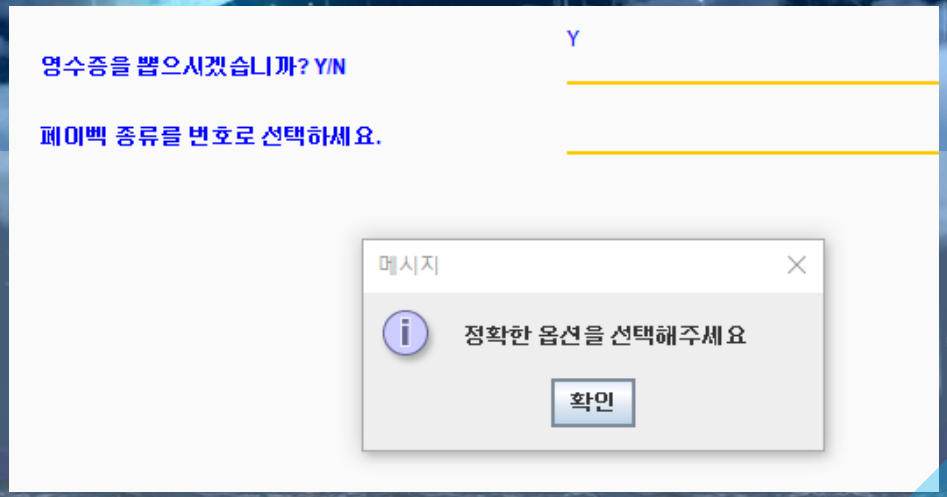
Test Case Num	Test Case	Result
1	페이백 종류 범위값 내 번호 입력	F
2	페이백 종류 입력하지 않음	F
3	페이백 종류 범위값 외 번호 입력	F
4	Printstatement의 입력으로 임의의 스트링	F
5	로그인 계좌와 동일한 계좌로 송금 요청	F
6	입금/출금/송금 금액에 음수값 입력	F
7	100000이상의 입금을 수행하고 DB파일 잔액 변화유무 확인	F
8	DB파일 한도를 음수값으로 설정한 후 프로그램 실행	F
9	영수증 출력(y/n) 입력하지 않음	F
10	이체시 DB에 없는 계좌번호값 입력	F
11	페이백 종류, 영수증 출력값 둘 다 입력하지 않음	P
12	입금-금액입력시 특정 값("333")입력	F
13	입력값이 잘못된 경우에 다음으로 진행하는데(이미 Fail인 상태), 그 때 영수증, 페이백 입력	F
14	수수료 발생 유무 확인	F

1/14 = 7% Pass

Modified factors by Brute Testing Reports

Test Case No	2
Problem	Payback 종류 값을 입력하지 않았을 경우 예외처리를 해주지 않고 넘어감
Reason	유효범위 이외의 값에 대한 예외처리를 구현해주지 않음
Solution	소스 코드 수정

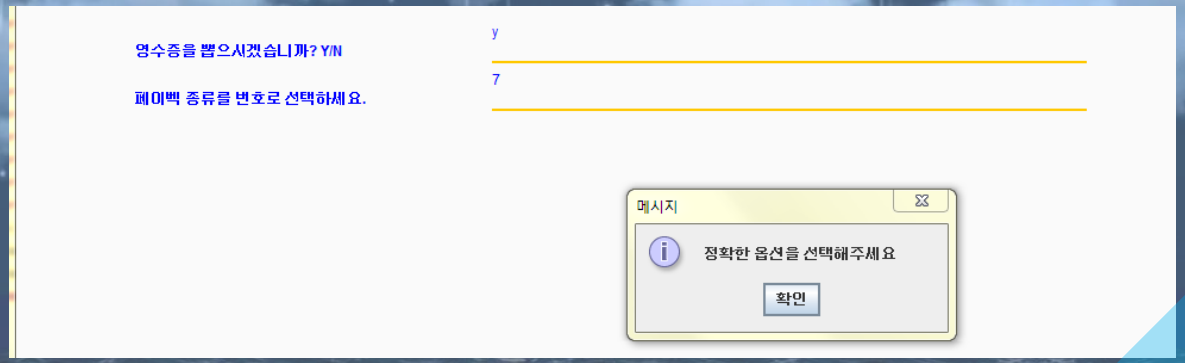
```
long result=pay.Check_Payback((controller.getUfreq()-1));
if(mode==3)
{
    result=0;
}
if(result>=1)
{

    if(choice.getText().equals("1")||choice.getText().equals("2"))
    {
        pay.Choose_Gift_code(result,Long.parseLong(choice.getText()));
        process=1;
    }
    else if(choice.getText()==""){
        process=2;
        nextPanel("statement");
    }
    else
    {
        process=2;
        Mainview.show("정확한 옵션을 선택해주세요");
        nextPanel("statement");
    }

}
```



Modified factors by Brute Testing Reports

Test Case No	3
Problem	Payback 유효범위 이외의 값 입력 시 예외처리를 해주지 않고 넘어감
Reason	유효범위 이외의 값에 대한 예외처리를 구현해주지 않음
Solution	소스 코드 수정

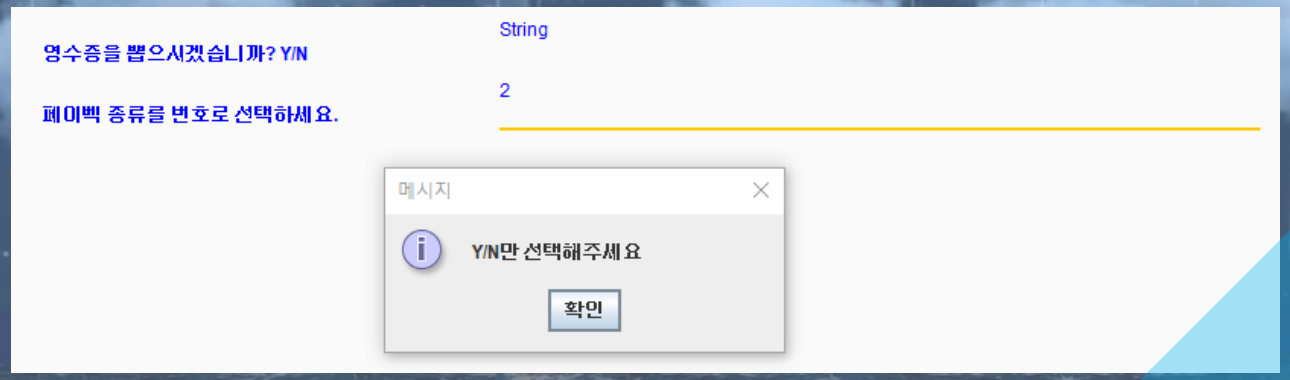
```
long result=pay.Check_Payback((controller.getUfreq()-1));
if(mode==3)
{
    result=0;
}
if(result>=1)
{

    if(choice.getText().equals("1")||choice.getText().equals("2"))
    {
        pay.Choose_Gift_code(result,Long.parseLong(choice.getText()));
        process=1;
    }
    else if(choice.getText()==""){
        process=2;
        nextPanel("statement");
    }
    else
    {
        process=2;
        Mainview.show("정확한 옵션을 선택해주세요");
        nextPanel("statement");
    }

}
```



Modified factors by Brute Testing Reports

Test Case No	4
Problem	Print Statement의 입력 값에 다른 형태의 데이터 입력 시 오류 없이 넘어감
Reason	입력 값에 대한 데이터 형에 따라 예외처리를 해주지 않음
Solution	소스 코드 수정

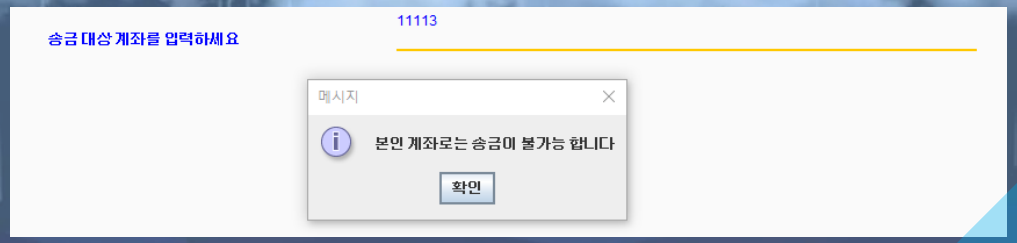
```
else if(YN==2)
{
    if(result>=1)
    {
        Mainview.show("페이백이 있으므로 영수증이 자동으로 뽑힙니다");
        nextPanel("print");
    }
    else
    {
    Mainview.this.show("거래가 끝났습니다");
    update_setter=1;
    nextPanel("selectMenu");
    }

}
else
{
    Mainview.show("Y/N만 선택해주세요");
    nextPanel("statement");
}
}
```



Modified factors by Brute Testing Reports

Test Case No	5
Problem	로그인 계좌와 동일한 계좌로 송금 요청 시 오류 없이 넘어감
Reason	User와 Receiver 계정을 비교하여 예외처리해주는 부분을 구현하지 않음
Solution	소스 코드 수정

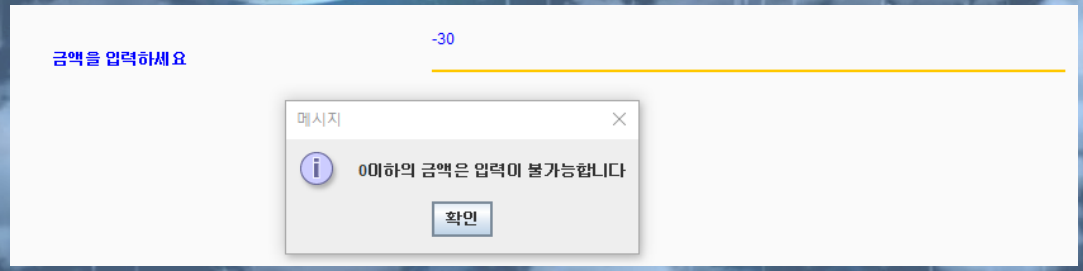
```
try {
    check=controller.get_ReceiverAccount(accountinput.getText());
} catch (Exception e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}
if(check==true)
{
if(accountinput.getText().equals(controller.get_UAccount()))
{

    Mainview.show("본인 계좌로는 송금이 불가능 합니다");
    nextPanel("inputreceiver");
}
```



Modified factors by Brute Testing Reports

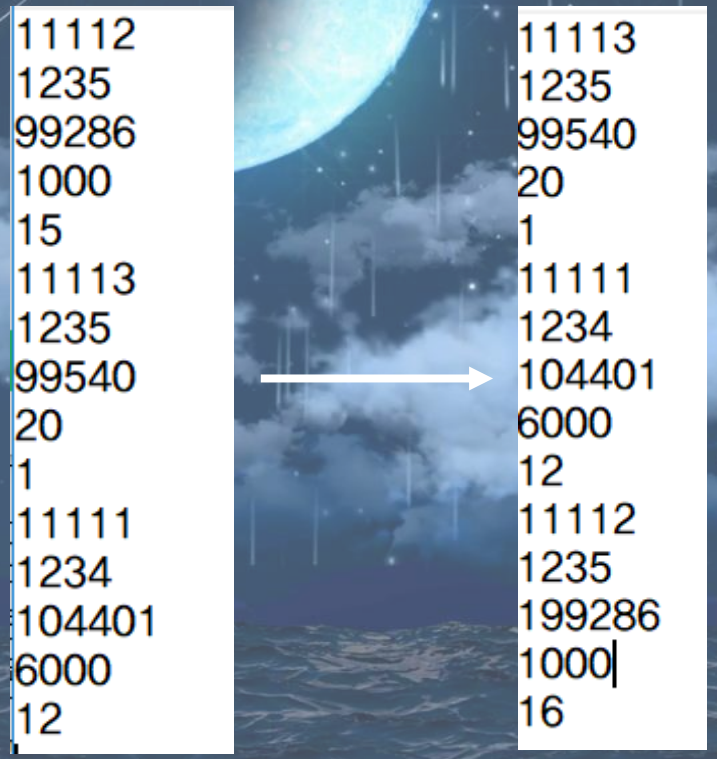
Test Case No	6
Problem	각 거래의 Input Amount에 음수 값을 입력 시 오류 없이 넘어감
Reason	Input Amount에 음수에 대한 예외처리를 구현하지 않음
Solution	소스 코드 수정

```
try{
if(Long.parseLong(amount.getText())<=0)
{
    Mainview.show("0이하의 금액은 입력이 불가능합니다");
    nextPanel("inputamount");
}
```



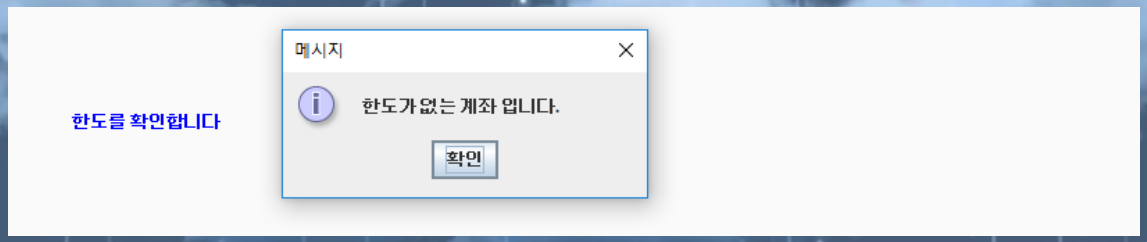
Test Case No	7
Problem	1000000이상의 입금 수행 시 DB파일의 잔액 변화가 이루어지지 않음
Reason	Update Account에서 10000000이상의 값에 대한 값이 처리되지 않음
Solution	소스 코드 수정

```
    void set_Amount(long amount)
    {
        this.Amount=amount;
    }

    long get_Amount()
    {
        return -this.Amount;
    }

}
```



Test Case No	8
Problem	DB 파일의 한도를 음수 값으로 설정한 경우 오류 발생
Reason	DB 파일의 한도가 0 이하의 경우 한도액이 0으로 처리하는 부분이 구현되어 있지 않음
Solution	소스 코드 수정

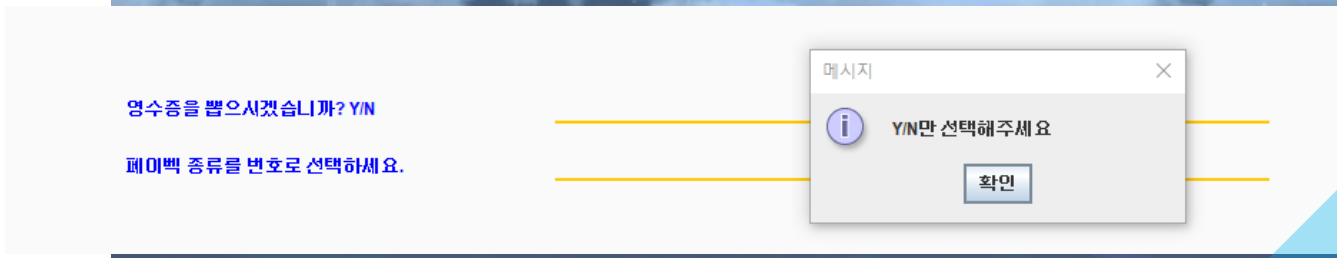
```
update_setter=0;
boolean limitcheck=controller.Limit_Amount();
boolean max=controller.max();
if(controller.getUlimit()<=0)
{
    Mainview.show("한도가 없는 계좌 입니다.");
```



Modified factors by Brute Testing Reports

Test Case No	9
Problem	영수증 출력 입력 칸에 값을 입력하지 않았을 경우 오류 없이 넘어감
Reason	영수증 입력 값을 받지 않았을 경우의 예외처리를 해주지 않음
Solution	소스 코드 수정

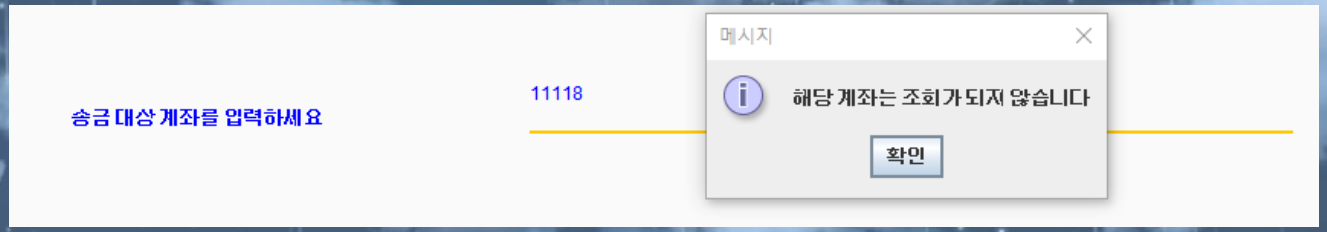
```
else if(YN==2)
{
    if(result>=1)
    {
        Mainview.show("페이백이 있으므로 영수증이 자동으로 뽑힙니다");
        nextPanel("print");
    }
    else
    {
    Mainview.this.show("거래가 끝났습니다");
    update_setter=1;
    nextPanel("selectMenu");
    }

}
else
{
    Mainview.show("Y/N만 선택해주세요");
    nextPanel("statement");
}
}
```



Modified factors by Brute Testing Reports

Test Case No	10
Problem	이체 거래 시 DB에 없는 계좌번호 입력 시 오류 없이 넘어감
Reason	txt파일에 없는 값 입력 시 예외처리를 해주지 않음
Solution	소스 코드 수정

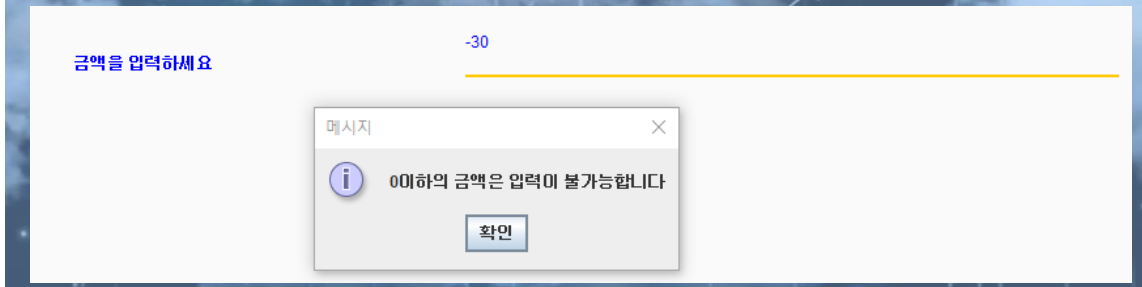
```
            }
            }
            else
            {
                Mainview.show("해당 계좌는 조회가 되지 않습니다");
            }
            }
        }catch(java.lang.NumberFormatException e4)
        {
            Mainview.show("입력하신 값이 너무 큽니다 다시 입력하세요");
            nextPanel("inputreceiver");
        }
```



Modified factors by Brute Testing Reports

Test Case No	13
Problem	입력 값이 잘못된 경우에도 다음으로 진행됨
Reason	입력 값 오류에 대해 예외처리 후 그 다음 코드로 진행하게 설정함
Solution	소스 코드 수정

```
try{
if(Long.parseLong(amount.getText())<=0)
{
    Mainview.show("0이하의 금액은 입력이 불가능합니다");
    nextPanel("inputamount");
}
else{
if(mode==1)
{
Send send=new Send();
controller.SetInput_Amount(send.get_Amount(controller.input_Amount(amount.getText())));
nextPanel("limit");
}
else if(mode==2)
{
Withdraw withdraw=new Withdraw();
controller.SetInput_Amount(withdraw.get_Amount(controller.input_Amount(amount.getText())));
nextPanel("limit");
}
else if(mode==3)
{
Deposit deposit=new Deposit();
deposit.set_Amount(controller.input_Amount(amount.getText()));
controller.SetInput_Amount(deposit.get_Amount());
nextPanel("statement");
}
}
}catch(java.lang.NumberFormatException e4)
{
    Mainview.show("입력하신 값이 너무 큽니다 다시 입력하세요");
    nextPanel("inputamount");
```



Modified Files

Stage 1000	
수정 전	수정 후
Describe Use Case에서 Actor에 대한 구분에 대한 명세가 필요함	Actor에 실제로 이용하는 고객으로 Customer로 수정함
Use Case 번호가 이름이 일치하지 않음	보고서 내에 모든 use case 번호와 순서 일치시킴

Stage 2030	
수정 전	수정 후
Send의 Typical Courses of Events의 7번 항목에 대해 어법상 어색	'입력한 출금액과 수수료가 더한 값이 정해 놓은 한도액을 초과하는지에 대한 부분을 검사한다'고 수정
Sequence Diagram에서 매개변수에 대한 정의가 필요 없음	매개변수 부분 제거
Operation Contracts에서 실제 사용할 함수의 이름과 같게 작성(Send!=Send Money)	Operation Name과 Use Case의 이름을 모두 실제 사용할 함수인 Send, Withdraw...형식으로 수정
State Diagram에서 send / withdraw 에서 Amount나 Payback 대상이 맞는지 확인하는 부분이 없음	과정 중에 한도 및 Payback 대상 맞는지 확인하는 Sequence 추가
State Diagram에서 check remain 에서 Password가 맞는지 확인하는 Sequence 없음	추가함


Q&A