A New Reliable ATM

대응 보고서

3rd Cycle

Project Team T6

201411140 권성완
201511247 김선정
201510436 허윤아
201510285 조수빈

Date

2018-06-10

1. 문서 수정 피드백

1. OOPT Stage 1000문서 수정

- Activity 1006 의 Use case 중 Withdraw

1st Cycle 피드백

- 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.

2nd Cycle 피드백

Use Case	3. Withdraw
Actors	Customer
Description	-Customer can withdraw cash from account by using check card or bankbook. -Customer can only withdraw money in unit of 10000₩ and 50000₩ under balance. -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 database.(Hidden)

➔ 시스템 보고서 v1 내용이 반영되지 않았다.

대응 : ATM에서 인출할 수 있는 현금을 무한대라고 가정하고, Withdraw 의 Description 중 limit a day라는 부분을 뺐습니다. 또한, ATM에서 기본적으로 관리하는 은행의 계좌를 제외하고는 인출 시 수수료가 붙는다는 Description을 추가했습니다.

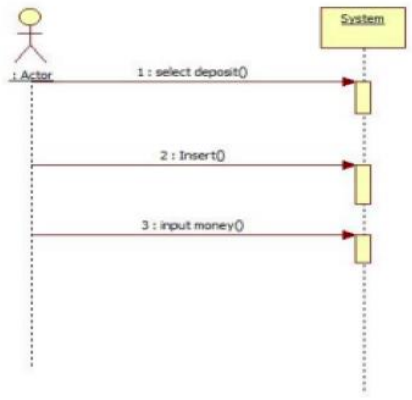
Use Case	3. Withdraw
Actors	Customer
Description	-Customer can withdraw cash from account. -Customer can only withdraw money in unit of 10000₩, 50000₩ under balance. -Customer needs to choose the number of 50000₩. -Customer needs to know password of an account. -If customer is withdrawing cash from other bank's account(not a default bank), ATM takes charge of certain percentage. -If customer successfully withdraws, ATM requests Offer to update DB.

2. OOPT Stage 2030 문서 수정

- Activity 2035 의 System Sequence Diagram 중 Deposit

1st Cycle 피드백

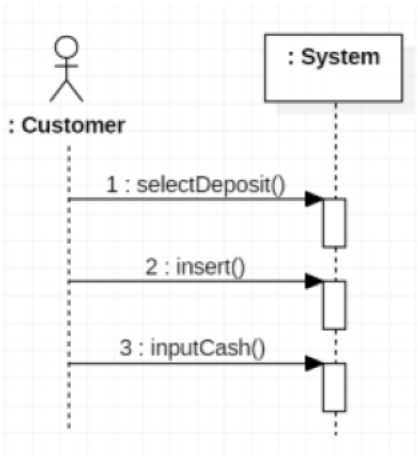
Activity 2035. Define System Sequence Diagrams

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.

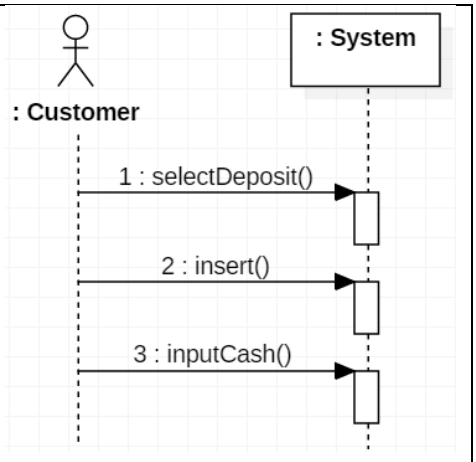
2nd Cycle 피드백

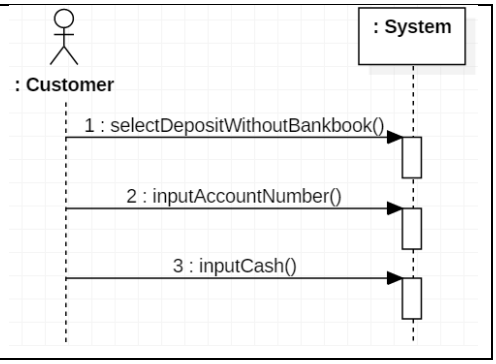
Activity 2035. Define System Sequence Diagrams

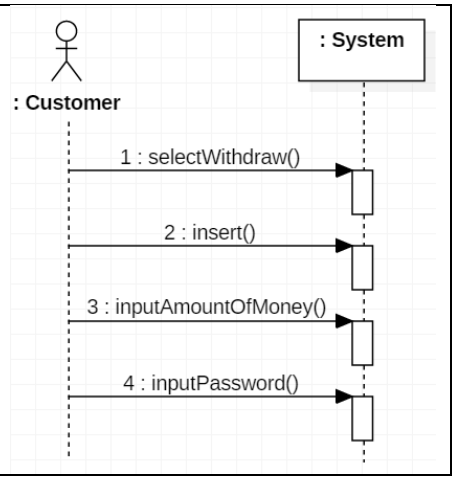
Use Case : Deposit	
1. Customer selects deposit menu from the basic screen. 2. System prints out "Insert check card or bankbook or credit card". 3. Customer inserts credit card, check card or bankbook("Insert"). 4. Invoke "Check Validation". If inserted method is valid, ask customer to input cash. 5. Customer Inputs cash in unit of 10000₩ and 50000₩. 6. System checks amount of cash inputted. 7. If total amount of money is correct, invoke "Print transaction receipt". 8. If Transaction Receipt is successfully printed, invoke "Update Database".	

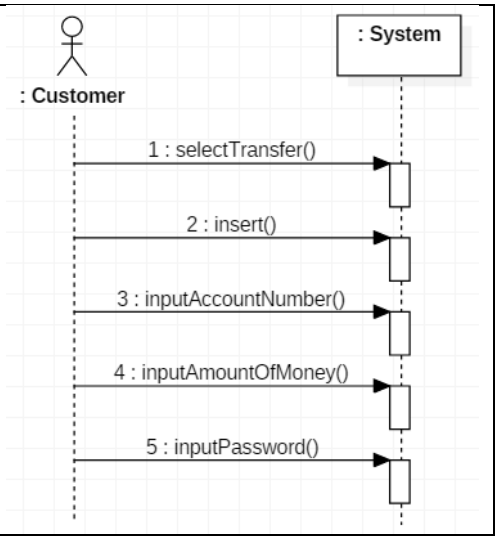
➔ 시스템 보고서 v1 내용이 반영되지 않았다.

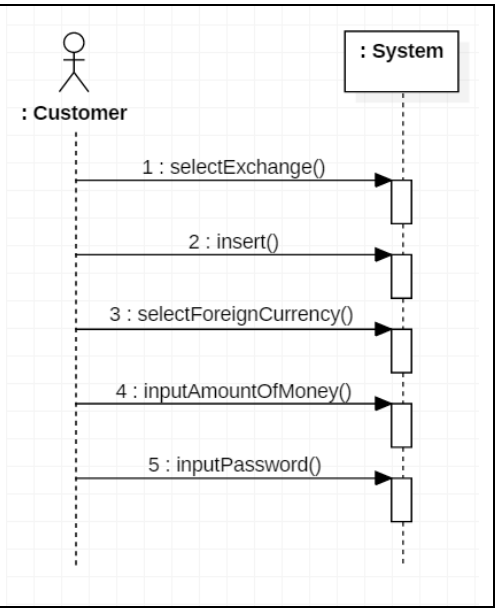
대응 : 기존의 너무 구체적이었던 Description을 간결하게 수정했습니다. Print Transaction Receipt의 경우, sequence상 필요하다고 느껴져 빼지 않았지만, system operation이 아니므로 diagram에는 반영하지 않았습니다. 이 외의 다른 Use case들도 System sequence diagram과 일관성을 가지도록 description을 수정했습니다.

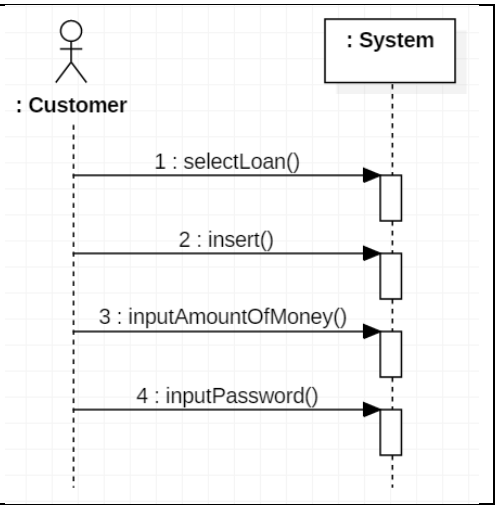
Use Case : Deposit	
1. Customer selects deposit menu from the basic screen. 2. Customer inputs account or credit card number. 3. Customer Inputs cash in unit of 10000₩, 50000₩. 4. System prints out transaction receipt.	

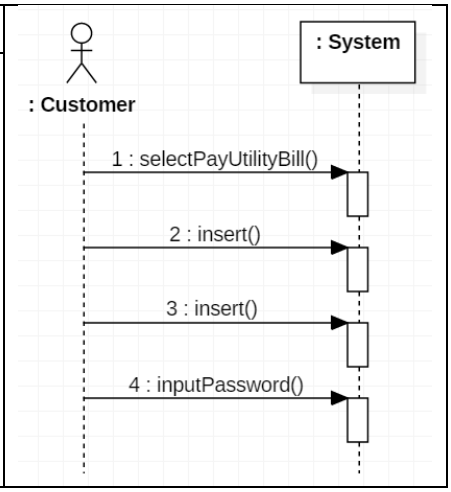
Use Case : Deposit without Bankbook	
1. Customer selects Deposit Without Bankbook menu from screen. 2. Customer inputs bank account number. 4. Customer inputs cash in unit of 10000₩, 50000₩. 4. system prints out transaction receipt.	

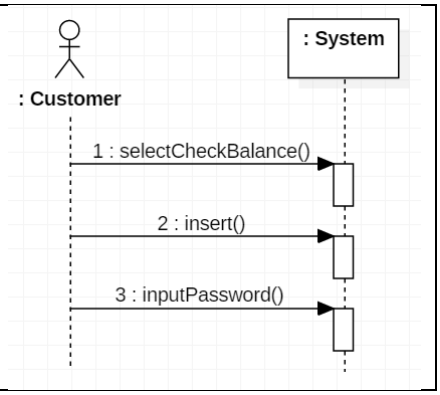
Use Case : Withdraw	
1. Customer selects withdraw menu from the basic screen. 2. Customer inputs account. 3. Customer inputs amount of money in unit of 10000₩, 50000₩ to withdraw from account. 4. Customer inputs password for the account. 5. If balance is enough, System withdraws cash and prints out transaction receipt.	

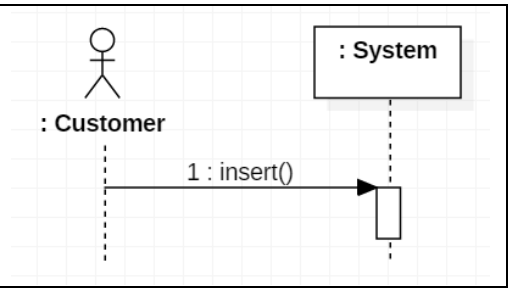
Use Case : Transfer	
1. Customer selects transfer menu from the basic screen. 2. Customer inputs account. 3. Customer inputs account number to transfer money. 4. Customer inputs amount of money to transfer. 5. Customer inputs password for the account. 6. If password is correct, system prints out transaction receipt.	

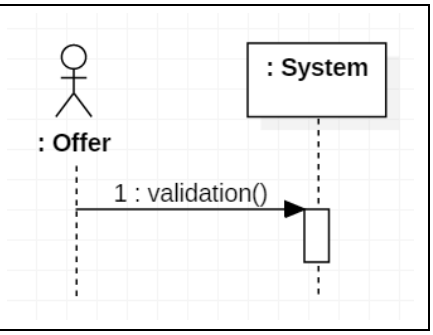
Use Case : Exchange	
1. Customer selects exchange menu from basic screen. 2. Customer inputs account. 3. Customer selects foreign currency to exchange. 4. Customer inputs amount of money to exchange. 5. Customer inputs password. 6. If password is correct, and If withdrawn correctly, system prints out transaction receipt.	

Use Case : Loan	
1. Customer chooses loan menu from basic screen. 2. Customer inputs account. 3. Customer inputs amount of money to loan in unit of 10000₩, 50000₩. 4. Customer inputs password for the credit card. 5. If password is correct, money to loan is under loan limit, withdrawn correctly, system prints out transaction receipt.	

Use Case : Pay Utility Bill	
1. Customer chooses pay utility bill menu from basic screen. 2. Customer inputs account written in giro bill. 3. Customer inputs own account. 4. Customer inputs password for the account. 5. If password is correct, balance is enough, system prints out transaction receipt.	

Use Case : Check Balance	
1. Customer chooses check balance menu from basic screen. 2. Customer inputs account. 3. Customer inputs password for the account. 4. If password is correct, system prints out recent transactional information and balance after each transactional process.	

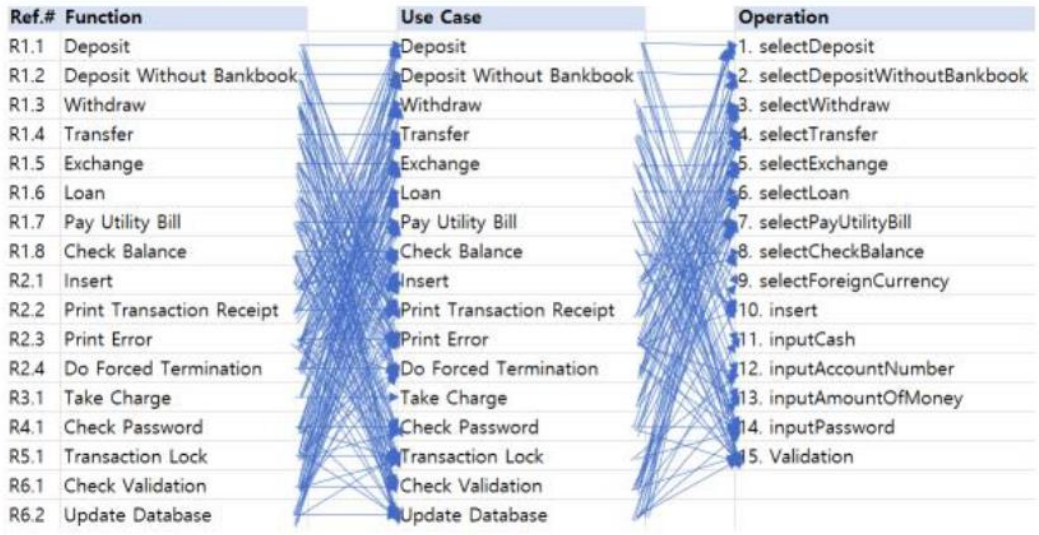
Use Case : Insert	
1. Customer inserts method for transaction, such as account of credit card.	

Use Case : Check Validation	
1. System requests to check validation of inserted method to Offer. 2. Offer compares information of inserted method between information stored in Database. 3. If inserted method is valid, Offer gives "valid" to System, and system goes on a process.	

- Activity 2039 Traceability Analysis

2nd Cycle 피드백

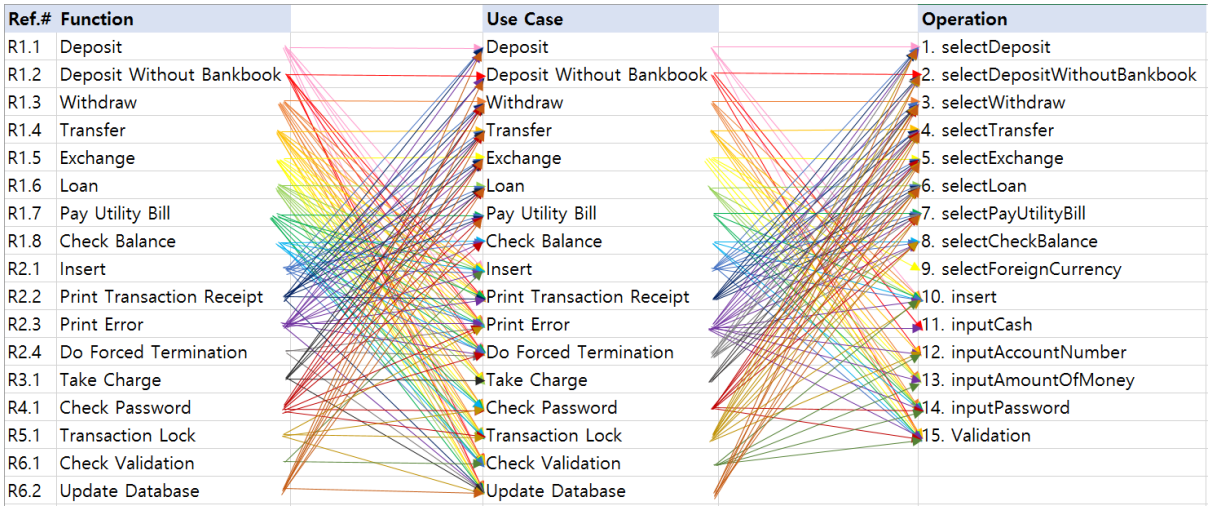
Activity 2039. Analyze (2030) Traceability Analysis



Ref.#	Function	Use Case	Operation
R1.1	Deposit	Deposit	1. selectDeposit
R1.2	Deposit Without Bankbook	Deposit Without Bankbook	2. selectDepositWithoutBankbook
R1.3	Withdraw	Withdraw	3. selectWithdraw
R1.4	Transfer	Transfer	4. selectTransfer
R1.5	Exchange	Exchange	5. selectExchange
R1.6	Loan	Loan	6. selectLoan
R1.7	Pay Utility Bill	Pay Utility Bill	7. selectPayUtilityBill
R1.8	Check Balance	Check Balance	8. selectCheckBalance
R2.1	Insert	Insert	9. selectForeignCurrency
R2.2	Print Transaction Receipt	Print Transaction Receipt	10. insert
R2.3	Print Error	Print Error	11. inputCash
R2.4	Do Forced Termination	Do Forced Termination	12. inputAccountNumber
R3.1	Take Charge	Take Charge	13. inputAmountOfMoney
R4.1	Check Password	Check Password	14. inputPassword
R5.1	Transaction Lock	Transaction Lock	15. Validation
R6.1	Check Validation	Check Validation	
R6.2	Update Database	Update Database	

➔ 모든 Function 과 Use case Operation의 관계들이 그려졌지만 정확한 관계가 보여지지 않는다. 추가적인 수정이 필요하다..

대응 : 서로 다른 색상을 이용해 작업함으로써, 좀 더 보기 쉽도록 수정했습니다.



Ref.#	Function	Use Case	Operation
R1.1	Deposit	Deposit	1. selectDeposit
R1.2	Deposit Without Bankbook	Deposit Without Bankbook	2. selectDepositWithoutBankbook
R1.3	Withdraw	Withdraw	3. selectWithdraw
R1.4	Transfer	Transfer	4. selectTransfer
R1.5	Exchange	Exchange	5. selectExchange
R1.6	Loan	Loan	6. selectLoan
R1.7	Pay Utility Bill	Pay Utility Bill	7. selectPayUtilityBill
R1.8	Check Balance	Check Balance	8. selectCheckBalance
R2.1	Insert	Insert	9. selectForeignCurrency
R2.2	Print Transaction Receipt	Print Transaction Receipt	10. insert
R2.3	Print Error	Print Error	11. inputCash
R2.4	Do Forced Termination	Do Forced Termination	12. inputAccountNumber
R3.1	Take Charge	Take Charge	13. inputAmountOfMoney
R4.1	Check Password	Check Password	14. inputPassword
R5.1	Transaction Lock	Transaction Lock	15. Validation
R6.1	Check Validation	Check Validation	
R6.2	Update Database	Update Database	

2. Brute Force Test

Test Case No.	2
문제	무통장 입금시, 현대카드로 카드사 변경이 되지 않는다.
대응	신한은행 ATM기로 지정해놨으므로 카드사 변경이 불가능하도록 설계되었으므로 수정의 필요성이 없음.

Test Case No.	14
문제	단위가 맞지 않는다는 안내가 나오지 않는다.
대응	안내가 나오도록 수정.

Test Case No.	17
문제	수수료가 처리되지 않는다.
대응	출금, 대출 시 수수료가 처리되도록 수정.

Test Case No.	18
문제	신용 등급이 제공되지 않는다.
대응	대출 거래 진행시 신용 등급이 제공되도록 수정.

Test Case No.	19
문제	확인 가능한 명세표가 제공되지 않는다.
대응	거래의 마지막에 명세표 제공.

Test Case No.	20
문제	1만,10만,100만이 잘 표시되지 않는다.
대응	환전에서 1만, 10만, 100만 버튼이 제대로 작동되도록 수정

3. Category Partitioning Test

Test Case No.	10
문제	존재하지 않는 계좌 번호를 입력해도 진행이 된다.
대응	존재하지 않는 계좌 번호는 거래가 되지 않도록 수정.

Test Case No.	14,18,22,26,38,42,46,58
문제	입금거래를 진행하며 0~10000원 사이의 금액을 입력할 수 없다.
대응	시스템상에서 10000원 단위로만 입출금이 가능하도록 설정해놨기 때문에 수정의 필요성이 없다.

Test Case No.	13,15,16
문제	입금거래를 진행하며 만원 단위의 적절한 금액을 입력, 송신 계좌 입력이 되지 않는다.
대응	코드를 수정

Test Case No.	17,18,19,20,41,42,43,44,49,50,51,52,57,58,59,60
문제	거래가 정지된 계좌번호일 시에 금액, 계좌 입력이 되지 않는다.
대응	거래가 정지된 계좌번호는 다음 단계로 진행할 수 없도록 설계했으므로 수정의 필요성이 없음.

Test Case No.	21,23,24,25
문제	무통장 입금 거래 진행시 정지되지 않은 계좌번호 입력과 적절한 금액 입력이 되지않는다.
대응	무통장 입금이 되도록 코드 수정

Test Case No.	39,40,41
문제	거래가 정지되지 않은 계좌로 대출이 되지 않는다.
대응	대출이 가능하도록 코드 수정.

Test Case No.	45,47,48
문제	송금 거래 진행이 되지 않는다.
대응	송금이 가능하도록 코드 수정

Test Case No.	53,55,56
문제	환전 거래 진행이 되지 않는다.
대응	환전이 가능하도록 코드 수정

4. Static Analysis

```
@@ -1,11 +1,17 @@
 package ATM;

-import java.text.SimpleDateFormat;
 import java.util.Date;
-import java.util.Scanner;

 import OFFER.Offer;

+//TODO 1: 중복되는 코드 정리
+//TODO 2: 체크카드0, 통장1, 신용카드 2... 이런 magic number를 enum으로 정리
+//TODO 3: if-else 정리
+//TODO 4: if문 조건식 정리. ex)if (isBig(String.valueOf((won = Integer.parseInt(money) * exchangeRate[i])), this.account.getBalance(
+//TODO 5: GUI문-  switch에 default 문이 없음.
+//TODO 6: 주석추가.
+//수정했으면 수정했다고 써 놓겠음.
+
 public class MainSystem {
     private Account account;
     public String card;
@@ -14,7 +20,24 @@
     public String b5, b1;
     public int errorType, ch;

+    // ----------------------------------------------------------------
+    // String[] list_bank: 메소드마다 들어가 있던거 global로 뺌.
+    String[] list_bank = { "국민은행", "기업은행", "농협은행", "신한은행", "씨티은행", "우리은행", "한국은행", "삼성카드", "현대카드", "롯
+
+    // 코드마다 return 2; return 3; return 1; 이런거 뭔지 못알아먹겠어서 가독성과 코드의 재활용성을 위해 상수로 따로 뺌.
+    static final int NO_ERROR = 0;
+    static final int NOT_ENOUGH_BALANCE = 1;
+    static final int SERVER_NOT_RESPONSE = 2;
+    static final int WRONG_ACCOUNT = 3;
+    static final int WRONG_PASSWORD_OVER_3 = 4;
+    static final int ACCOUNT_LOCKED = 5;
+    static final int DEPOSIT_OVER_DEPT = 6;
+    static final int OVER_LIMIT = 7;
+    static final int SAME_SENDER_RECEIVER = 8;
+    static final int UB_ALREADY_PAYED = 9;
+
```

```
17  39      public MainSystem() {
    40  +       // main system. system initial.
18  41          account = new Account();
19  42          card = null;
20  43          offer = new Offer[10];
    ❄   @@ -30,210 +53,230 @@ public void setAccount(Account account) {
30  53          this.account = account;
31  54      }
32  55
    56  +   private int findWhichBank() {
    57  +       // 수정: 메소드 추가.
    58  +       // 어떤 은행/카드사인지 찾고 은행종류를 int로 리턴.
    59  +       int bank;
    60  +       for (bank = 0; bank < list_bank.length; bank++) {
    61  +           if (list_bank[bank].equals(account.getBank()))
    62  +               break;
    63  +       }
    64  +       return bank;
    65  +   }
    66  +
    67  +   private int checkUpdateDB(Offer offer) { // update Database 성공여부 리턴.
    68  +       if (offer.updateDatabase(account))
    69  +           return NO_ERROR;
    70  +       else
    71  +           return SERVER_NOT_RESPONSE;
    72  +   }
    73  +
33  74      // 체크카드 0, 통장 1, 신용카드 2, 지로용지3 // 은행 // 계좌정보
34  75      public void insert(String str) {
35      -       String[] list_bank = { "국민은행", "기업은행", "농협은행", "신한은행", "씨티은행", "우리은행", "한국은행", "삼성카드", "현대카드"
    76  +       // 계좌번호를 insert받음. GUI에서 사용.
36  77          this.card = str;
37  78          this.account.setBank(list_bank[Integer.parseInt(card.charAt(0) + "")]);
38  79          this.account.setAccountNumber(card.substring(1));
39  80      }
40  81
```

```
41      -   public int lockAccount() {
42      -       String[] list_bank = { "국민은행", "기업은행", "농협은행", "신한은행", "씨티은행", "우리은행", "한국은행", "삼성카드", "현대카드"
43      -       int bank;
44      -       for (bank = 0; bank < list_bank.length; bank++) {
45      -           if (list_bank[bank].equals(account.getBank()))
46      -               break;
        82  +   public int isUnderBalance(String money) { // 잔액이 충분한지 확인하는 메소드
        83  +
        84  +       // 신한은행
        85  +       if ((account.getBank().equals("신한은행") && isBig(money, account.getBalance()))
        86  +               || (!account.getBank().equals("신한은행") && isBig(plus(money, "1300"), account.getBalance()))) {
        87  +           return 1;
47      88          }
48      -       if ((this.errorType = offer[bank].readDatabase(account)) == 0) {
49      -           account.setIsLocked(true);
50      -           if (offer[bank].updateDatabase(account)) {
51      -               return 0;
52      -           } else
53      -               return 2;
54      -       } else
55      -           return 3;
        89  +
        90  +       // 신한은행
        91  +       return 11111;
56      92      }
57      93
58      -   public int deposit(String money) {
59      -       String[] list_bank = { "국민은행", "기업은행", "농협은행", "신한은행", "씨티은행", "우리은행", "한국은행", "삼성카드", "현대카드"
        94  +   public boolean isWrongAccount(int bankIndex) { // 잘못된 계좌: true 리턴.
        95  +       this.errorType = offer[bankIndex].readDatabase(account);
        96  +       boolean readWrongDB = (this.errorType != 0);
        97  +       return readWrongDB;
        98  +   }
        99  +
        100 +   public int lockAccount() { // account를 lock하는 메소드.
        101 +
        102 +       // 머느 은행인지 찾는다.
```

```
61  -       for (bank = 0; bank < list_bank.length; bank++) {
62  -           if (list_bank[bank].equals(account.getBank()))
63  -               break;
64  -       }
    104 +       bank = findWhichBank();
    105 +
    106 +       // 계좌가 비정상적인 계좌인지 확인.
    107 +       if (isWrongAccount(bank))
    108 +           return WRONG_ACCOUNT;
    109 +
    110 +       // lock을 건다.
    111 +       account.setIsLocked(true);
    112 +
    113 +       // DB에 update
    114 +       return checkUpdateDB(offer[bank]);
    115 +
    116 +   }
    117 +
    118 +   public int deposit(String money) { // 입금 기능. return하는 에러들은 GUI에서 frame만들 때 쓰인다.
    119 +
    120 +       int bank;
    121 +       bank = findWhichBank();
    122 +       if (isWrongAccount(bank))
    123 +           return WRONG_ACCOUNT;
    124 +
    125 +       // 입력한 계좌가 카드번호인경우 빚 상환.
65  126         if (this.account.getBank().substring(2, 4).equals("카드")) {
66  -           if ((this.errorType = offer[bank].readDatabase(account)) == 0) {
67  -               if (isBig(money, account.getDept())) {
68  -                   return 6;
69  -               } else {
70  -                   account.setDept(minus(account.getDept(), money));
71  -                   String newLog = new Date() + " " + money + " 상환 \t( 대출금 : " + account.getDept() + " )\n";
72  -                   newLog = newLog + account.getLog();
73  -                   account.setLog(newLog);
74  -                   if (offer[bank].updateDatabase(account)) {
75  -                       return 0;
76  -                   } else {
77  -                       return 2;
78  -                   }
79  -               }
```

```
127 +            // 입금하는 돈 > 빚이면 입금불가.
128 +            if (isBig(money, account.getDept())) {
129 +                return DEPOSIT_OVER_DEPT;
130              } else {
81  -                return 3;
82  -            }
83  -        } else {
84  -            if ((this.errorType = offer[bank].readDatabase(account)) == 0) {
85  -                account.setBalance(plus(account.getBalance(), money));
86  -                String newLog = new Date() + " " + money + " 입금 \t( 잔액 : " + account.getBalance() + " )\n";
131 +                account.setDept(minus(account.getDept(), money));
132 +                String newLog = new Date() + " " + money + " 상환 \t( 대출금 : " + account.getDept() + " )\n";
133              newLog = newLog + account.getLog();
134              account.setLog(newLog);
89  -                if (offer[bank].updateDatabase(account)) {
90  -                    return 0;
91  -                } else {
92  -                    return 2;
93  -                }
94  -            } else {
95  -                return 3;
135 +                return checkUpdateDB(offer[bank]);
136          }
137 +
138 +        } else { // 입력한 계좌가 은행 계좌일 경우.
139 +
140 +            account.setBalance(plus(account.getBalance(), money));
141 +            String newLog = new Date() + " " + money + " 입금 \t( 잔액 : " + account.getBalance() + " )\n";
142 +            newLog = newLog + account.getLog();
143 +            account.setLog(newLog);
144 +            return checkUpdateDB(offer[bank]);
145 +
146          }
147      }
148
100 -    public int withdraw(String money) {
101 -        String[] list_bank = { "국민은행", "기업은행", "농협은행", "신한은행", "씨티은행", "우리은행", "한국은행", "삼성카드", "현대카드"
149 +    public int withdraw(String money) { // 출금한다.
150          int bank;
```

```
103     -       for (bank = 0; bank < list_bank.length; bank++) {
104     -           if (list_bank[bank].equals(account.getBank()))
105     -               break;
106     -       }
107     -       if ((this.errorType = offer[bank].readDatabase(account)) == 0) {
108     -           if ((account.getBank().equals("신한은행") && isBig(money, account.getBalance()))
109     -                   || (!account.getBank().equals("신한은행") && isBig(plus(money, "1300"), account.getBalance()))) {
110     -               return 1;
111     -           } else {
112     -               String newLog;
113     -               if (!account.getBank().equals("신한은행")) {
114     -                   this.takeCharge(account);
115     -               }
116     -               account.setBalance(minus(account.getBalance(), money));
117     -               newLog = new Date() + " " + money + " 출금 \t( 잔액 : " + account.getBalance() + " )\n" + account.getLog();
118     -               account.setLog(newLog);
        151 +       bank = findWhichBank();
119     152
120     -               if (offer[bank].updateDatabase(account)) {
121     -                   return 0;
122     -               } else {
123     -                   return 2;
124     -               }
        153 +       // account 확인.
        154 +       if (isWrongAccount(bank))
        155 +           return WRONG_ACCOUNT;
        156 +
        157 +       // 잔액 충분한지 확인
        158 +       if ((account.getBank().equals("신한은행") && isBig(money, account.getBalance()))
        159 +               || (!account.getBank().equals("신한은행") && isBig(plus(money, "1300"), account.getBalance()))) {
        160 +           return 1;
        161 +       } else { // 돈 출금
        162 +           String newLog;
        163 +           if (!account.getBank().equals("신한은행")) {
        164 +               this.takeCharge(account);
125     165             }
126     -       } else {
127     -           return 3;
```

```diff
+            account.setBalance(minus(account.getBalance(), money));
+            newLog = new Date() + " " + money + " 출금 \t( 잔액 : " + account.getBalance() + " )\n" + account.getLog();
+            account.setLog(newLog);
+
+            return checkUpdateDB(offer[bank]);
         }
     }

-    public int transfer(String money, Account newAccount) {
-        String[] list_bank = { "국민은행", "기업은행", "농협은행", "신한은행", "씨티은행", "우리은행", "한국은행", "삼성카드", "현대카드"
+    public int transfer(String money, Account newAccount) { // 송금 시도하고 error code를 리턴.
         int bank;
-        for (bank = 0; bank < list_bank.length; bank++) {
-            if (list_bank[bank].equals(account.getBank()))
-                break;
-        }
+        bank = findWhichBank();
+
         if (offer[bank].checkValid(account)) {
-            if ((this.errorType = offer[bank].readDatabase(account)) == 0) {
-                int temp_bank = 20;
-                for (int i = 0; i < 10; i++) {
-                    if (newAccount.getBank().equals(list_bank[i])) {
-                        temp_bank = i;
-                        break;
-                    }
-                }
-                if (temp_bank == 20) {
-                    return 2;
+            // ----------------------------------------------------------------
+            this.errorType = offer[bank].readDatabase(account);
+            boolean readWrongDB = !(this.errorType == 0);
+            if (readWrongDB)
+                return WRONG_ACCOUNT;
+            // if ((this.errorType = offer[bank].readDatabase(account)) == 0)
+            // {
+
+            int temp_bank = 20;
+            for (int i = 0; i < 10; i++) {
+                if (newAccount.getBank().equals(list_bank[i])) {
+                    temp_bank = i;
+                    break;
```

```
150     -               if ((account.getBank().equals("신한은행") && isBig(money, account.getBalance()))
151     -                       || (!account.getBank().equals("신한은행") && isBig(plus(money, "1300"), account.getBalance()))) {
152     -                   return 1;
153     -               } else {
154     -                   if (newAccount.getBank().substring(2, 4).equals("카드")) {
155     -                       if ((this.errorType = offer[bank].readDatabase(newAccount)) == 0) {
156     -                           if (isBig(money, newAccount.getDept())) {
157     -                               return 6;
193     +           }
194     +           if (temp_bank == 20) {
195     +               return 2;
196     +           }
197     +
198     +           if ((account.getBank().equals("신한은행") && isBig(money, account.getBalance()))
199     +                   || (!account.getBank().equals("신한은행") && isBig(plus(money, "1300"), account.getBalance()))) {
200     +               return 1;
201     +           } else {
202     +               if (newAccount.getBank().substring(2, 4).equals("카드")) {
203     +                   if ((this.errorType = offer[bank].readDatabase(newAccount)) == 0) {
204     +                       if (isBig(money, newAccount.getDept())) {
205     +                           return 6;
206     +                       } else {
207     +                           newAccount.setDept(minus(newAccount.getDept(), money));
208     +                           String newLog = new Date() + " " + money + " 상환 \t( 대출금 : " + newAccount.getDept() + " )\n";
209     +                           newLog = newLog + newAccount.getLog();
210     +                           newAccount.setLog(newLog);
211     +                           if (!offer[bank].updateDatabase(newAccount)) {
212     +                               return 2;
213     +                           }
214     +                           offer[bank].readDatabase(account);
215     +                           if (!account.getBank().equals("신한은행") || !newAccount.getBank().equals("신한은행"))
216     +                               takeCharge(account);
217     +                           account.setBalance(minus(account.getBalance(), money));
218     +                           newLog = new Date() + " " + money + " 송금 " + newAccount.getBank() + " "
219     +                                   + newAccount.getAccountNumber() + "\t( 잔액 : " + account.getBalance() + " )\n";
220     +                           newLog = newLog + account.getLog();
221     +                           account.setLog(newLog);
222     +                           if (offer[bank].updateDatabase(account)) {
223     +                               return 0;
158     224                           } else {
```

```
159     -                   newAccount.setDept(minus(newAccount.getDept(), money));
160     -                   String newLog = new Date() + " " + money + " 상환 \t( 대출금 : " + newAccount.getDept()
        225 +               newAccount.setDept(plus(newAccount.getDept(), money));
        226 +               newLog = new Date() + " " + money + " 기존거래 취소 \t( 대출금 : " + newAccount.getDept()
161     227                             + " )\n";
162     228                 newLog = newLog + newAccount.getLog();
163     229                 newAccount.setLog(newLog);
164     230                 if (!offer[bank].updateDatabase(newAccount)) {
165     231                     return 2;
166     232                 }
167     -                   offer[bank].readDatabase(account);
168     -                   if (!account.getBank().equals("신한은행") || !newAccount.getBank().equals("신한은행"))
169     -                       takeCharge(account);
170     -                   account.setBalance(minus(account.getBalance(), money));
171     -                   newLog = new Date() + " " + money + " 송금 " + newAccount.getBank() + " "
172     -                           + newAccount.getAccountNumber() + "\t( 잔액 : " + account.getBalance() + " )\n";
173     -                   newLog = newLog + account.getLog();
174     -                   account.setLog(newLog);
175     -                   if (offer[bank].updateDatabase(account)) {
176     -                       return 0;
177     -                   } else {
178     -                       newAccount.setDept(plus(newAccount.getDept(), money));
179     -                       newLog = new Date() + " " + money + " 기존거래 취소 \t( 대출금 : " + newAccount.getDept()
180     -                               + " )\n";
181     -                       newLog = newLog + newAccount.getLog();
182     -                       newAccount.setLog(newLog);
183     -                       if (!offer[bank].updateDatabase(newAccount)) {
184     -                           return 2;
185     -                       }
186     -                       return 2;
187     -                   }
        233 +               return 2;
188     234             }
189     -           } else {
190     -               return 3;
191     235         }
        236 +   } else {
        237 +       return 3;
192     238     }
        239 + }
```

```
239 +                }
240 +                newAccount.setBalance(plus(newAccount.getBalance(), money));
241 +                String tempLog = new Date() + " " + money + " 입금 " + account.getBank() + " "
242 +                        + account.getAccountNumber() + "\t( 잔액 : " + newAccount.getBalance() + " )\n";
243 +                tempLog = tempLog + newAccount.getLog();
244 +                newAccount.setLog(tempLog);
245 +                if (!offer[temp_bank].updateDatabase(newAccount)) {
246 +                    return 2;
247 +                }
248 +                offer[bank].readDatabase(account);
249 +                if (!account.getBank().equals("신한은행") || !newAccount.getBank().equals("신한은행"))
250 +                    takeCharge(account);
251 +                account.setBalance(minus(account.getBalance(), money));
252 +                String newLog = new Date() + " " + money + " 송금 " + newAccount.getBank() + " "
253 +                        + newAccount.getAccountNumber() + "\t( 잔액 : " + account.getBalance() + " )\n";
254 +                newLog = newLog + account.getLog();
255 +                account.setLog(newLog);
256 +                if (offer[bank].updateDatabase(account)) {
257 +                    return 0;
258 +                } else {
193 259                     newAccount.setBalance(plus(newAccount.getBalance(), money));
194     -                    String tempLog = new Date() + " " + money + " 입금 " + account.getBank() + " "
    260 +                    tempLog = new Date() + " " + money + " 기존거래 취소 " + account.getBank() + " "
195 261                              + account.getAccountNumber() + "\t( 잔액 : " + newAccount.getBalance() + " )\n";
196 262                     tempLog = tempLog + newAccount.getLog();
197 263                     newAccount.setLog(tempLog);
198 264                     if (!offer[temp_bank].updateDatabase(newAccount)) {
199 265                         return 2;
200 266                     }
201     -                    offer[bank].readDatabase(account);
202     -                    if (!account.getBank().equals("신한은행") || !newAccount.getBank().equals("신한은행"))
203     -                        takeCharge(account);
204     -                    account.setBalance(minus(account.getBalance(), money));
205     -                    String newLog = new Date() + " " + money + " 송금 " + newAccount.getBank() + " "
206     -                            + newAccount.getAccountNumber() + "\t( 잔액 : " + account.getBalance() + " )\n";
207     -                    newLog = newLog + account.getLog();
208     -                    account.setLog(newLog);
209     -                    if (offer[bank].updateDatabase(account)) {
210     -                        return 0;
211     -                    } else {
212     -                        newAccount.setBalance(plus(newAccount.getBalance(), money));
213     -                        tempLog = new Date() + " " + money + " 기존거래 취소 " + account.getBank() + " "
214     -                                + account.getAccountNumber() + "\t( 잔액 : " + newAccount.getBalance() + " )\n";
```

```
215     -                           tempLog = tempLog + newAccount.getLog();
216     -                           newAccount.setLog(tempLog);
217     -                           if (!offer[temp_bank].updateDatabase(newAccount)) {
218     -                                return 2;
219     -                           }
220     -                           return 2;
221     -                      }
        267 +                   return 2;
222     268                    }
223     -              } else {
224     -                   return 3;
225     269                }
        270 +          } else {
        271 +              return 3;
226     272            }
227     -          return 3;
228     273        }
        274 +      // return 3;
        275 +      // }
229     276
230     277        public int exchange(String money, String str) {
231     -          String[] list_bank = { "국민은행", "기업은행", "농협은행", "신한은행", "씨티은행", "우리은행", "한국은행", "삼성카드", "현대카드"
232     278            int bank;
233     -          for (bank = 0; bank < list_bank.length; bank++) {
234     -              if (list_bank[bank].equals(account.getBank()))
235     -                  break;
236     -          }
        279 +          bank = findWhichBank();
237     280            int[] exchangeRate = { 1100, 10, 1280, 200 };
238     281            String[] country = { "USD", "JPY", "EUR", "CNY" };
239     282            int i;
                   @@ -250,21 +293,13 @@ public int exchange(String money, String str) {
250     293                            + " )\n";
251     294                newLog = newLog + account.getLog();
252     295                account.setLog(newLog);
253     -              if (offer[bank].updateDatabase(account)) {
254     -                  return 0;
255     -              } else {
256     -                  return 2;
257     -              }
```

```
296 +             return checkUpdateDB(offer[bank]);
297           }
298       }
299
300       public int loan(String money) {
262 -         String[] list_bank = { "국민은행", "기업은행", "농협은행", "신한은행", "씨티은행", "우리은행", "한국은행", "삼성카드", "현대카드"
301           int bank;
264 -         for (bank = 0; bank < list_bank.length; bank++) {
265 -             if (list_bank[bank].equals(account.getBank()))
266 -                 break;
267 -         }
302 +         bank = findWhichBank();
303           if (isBig(plus(plus(account.getDept(), money), "1300"), account.getLimit())) {
304               return 7;
305           } else {
@@ -275,11 +310,7 @@ public int loan(String money) {
310                       + minus(account.getLimit(), account.getDept()) + " )\n";
311               newLog = newLog + account.getLog();
312               account.setLog(newLog);
278 -             if (offer[bank].updateDatabase(account)) {
279 -                 return 0;
280 -             } else {
281 -                 return 2;
282 -             }
313 +             return checkUpdateDB(offer[bank]);
314           }
315       }
316
@@ -290,11 +321,12 @@ public void takeCharge(Account account) {
321           String newLog;
322           if (account.getBank().substring(2, 4).equals("카드")) {
323               account.setDept(plus(account.getDept(), "1300"));
293 -             newLog = new Date() + "1300" + "수수료 \t( 한도 : " + minus(account.getLimit(), account.getDept()) + " )\n" + account.
324 +             newLog = new Date() + "1300" + "수수료 \t( 한도 : " + minus(account.getLimit(), account.getDept()) + " )\n"
325 +                     + account.getLog();
326           } else {
327               account.setBalance(minus(account.getBalance(), "1300"));
328               newLog = new Date() + " 1300" + " 수수료 \t( 잔액 : " + account.getBalance() + " )\n" + account.getLog();
297 -         }
329 +         }
```

```
298 330         account.setLog(newLog);
299 331     }
300 332
    ❄   @@ -307,12 +339,9 @@ public String checkBalance() {
307 339     }
308 340
309 341     public int payUtilityBill(Account newAccount) {
310     -       String[] list_bank = { "국민은행", "기업은행", "농협은행", "신한은행", "씨티은행", "우리은행", "한국은행", "삼성카드", "현대카드"
311     -       int bank;
312     -       for (bank = 0; bank < list_bank.length; bank++) {
313     -           if (list_bank[bank].equals(account.getBank()))
314     -               break;
315     -       }
    342 +
    343 +       int bank = findWhichBank();
    344 +
316 345         if (offer[bank].checkValid(account)) {
317 346             String money = newAccount.getBalance();
318 347             if ((this.errorType = offer[bank].readDatabase(account)) != 0) {
    ❄   @@ -340,18 +369,14 @@ public int payUtilityBill(Account newAccount) {
340 369                             + newAccount.getAccountNumber() + "\t( 잔액 : " + account.getBalance() + " )\n";
341 370                 newLog = newLog + account.getLog();
342 371                 account.setLog(newLog);
343     -               if (offer[bank].updateDatabase(account)) {
344     -                   return 0;
345     -               } else {
346     -                   return 2;
347     -               }
    372 +               return checkUpdateDB(offer[bank]);
348 373             }
349 374         } else {
350 375             return 3;
351 376         }
352 377     }
353 378
354     -   public boolean isBig(String n1, String n2) {
    379 +   public boolean isBig(String n1, String n2) { // n1 > n2면 true 리턴.
355 380         if (n1.length() > n2.length())
356 381             return true;
357 382         else if (n1.length() == n2.length()) {
```

```
@@ -59,7 +59,7 @@ public void setAccount(Account account)
59 59          this.account = account;
60 60      }
61 61
62    -     private int findWhichBank()
   62 +     private int findWhichBank(Account account)
63 63      {
64 64          //수정: 메소드 추가.
65 65          //어떤 은행/카드사인지 찾고 은행종류를 int로 리턴.
@@ -73,7 +73,7 @@ private int findWhichBank()
73 73      }
74 74
75 75
76    -     private int checkUpdateDB(Offer offer)
   76 +     private int checkUpdateDB(Offer offer, Account account)
77 77      {   //update Database 성공여부 리턴.
78 78          if (offer.updateDatabase(account)) return NO_ERROR;
79 79          else return SERVER_NOT_RESPONSE;
@@ -113,7 +113,7 @@ public int lockAccount()
113 113
114 114         //어느 은행인지 찾는다.
115 115         int bank;
116    -         bank = findWhichBank();
   116 +         bank = findWhichBank(account);
117 117
118 118         //계좌가 비정상적인 계좌인지 확인.
119 119         if (isWrongAccount(bank,account)) return WRONG_ACCOUNT;
@@ -122,7 +122,7 @@ public int lockAccount()
122 122         account.setIsLocked(true);
123 123
124 124         //DB에 update
125    -         return checkUpdateDB(offer[bank]);
   125 +         return checkUpdateDB(offer[bank], account);
126 126
127 127     }
128 128
```

```
@@ -131,202 +131,178 @@ public int deposit(String money)
131 131     {   //입금 기능. return하는 에러들은 GUI에서 frame만들 때 쓰인다.
132 132
133 133         int bank;
134     -       bank = findWhichBank();
    134 +       bank = findWhichBank(account);
135 135         if (isWrongAccount(bank,account)) return WRONG_ACCOUNT;
136 136
137 137         //입력한 계좌가 카드번호인경우 빛 상환.
138 138         if (this.account.getBank().substring(2, 4).equals("카드"))
139 139         {
140     -           return deposit_CARD(money,account, bank);
    140 +           return deposit_CARD(money, this.account, bank);
141 141         } else
142 142         { //입력한 계좌가 은행 계좌일 경우.
143     -           return deposit_BANK(money,account, offer[bank]);
144     -
    143 +           return deposit_BANK(money, this.account, bank);
145 144         }
146 145     }
147 146
148     -   private int deposit_BANK(String money, Account account, Offer offer)
149     -   {
150     -       account.setBalance(plus(this.account.getBalance(), money));
    147 +   private int deposit_BANK(String money, Account account, int bank)
    148 +   {   //통장으로 입금하기. 입금+로그저장+checkUpdateDB까지.
    149 +
    150 +       account.setBalance(plus(account.getBalance(), money));
151 151         String newLog = new Date() + " " + money + " 입금 \t( 잔액 : " + account.getBalance() + " )\n";
152     -       newLog = newLog + this.account.getLog();
    152 +       newLog = newLog + account.getLog();
153 153         account.setLog(newLog);
154     -       return checkUpdateDB(offer);
    154 +       return checkUpdateDB(offer[bank],account );
155 155     }
156 156
157 157     private int deposit_CARD(String money, Account account, int bank)
158     -   {
    158 +   {   //카드로 입금하기. 입금+로그저장+checkUpdateDB까지.
    159 +
```

```
159 160         // 입금하는 돈 > 빚이면 입금불가.
160     -       if (isBig(money, this.account.getDept()))
    161 +       if (isBig(money, account.getDept()))
161 162         {
162 163             return DEPOSIT_OVER_DEPT;
163 164         } else
164 165         {
165     -           this.account.setDept(minus(this.account.getDept(), money));
166     -           String newLog = new Date() + " " + money + " 상환 \t( 대출금 : " + this.account.getDept() + " )\n";
167     -           newLog = newLog + this.account.getLog();
168     -           this.account.setLog(newLog);
169     -           return checkUpdateDB(offer[bank]);
    166 +           account.setDept(minus(account.getDept(), money));
    167 +           String newLog = new Date() + " " + money + " 상환 \t( 대출금 : " + account.getDept() + " )\n";
    168 +           newLog = newLog + account.getLog();
    169 +           account.setLog(newLog);
    170 +           return checkUpdateDB(offer[bank],account );
170 171         }
171 172     }
172 173
173 174
174 175     public int withdraw(String money)
175 176     {   //출금한다.
    177 +
    178 +       //지역변수 bank. 어디 은행인지 알기 위함.
176 179         int bank;
177     -       bank = findWhichBank();
    180 +       bank = findWhichBank(account);
178 181
179 182         //account 확인.
180 183         if(isWrongAccount(bank, account)) return WRONG_ACCOUNT;
181 184
182 185         //잔액 충분한지 확인
183 186             if(isUnderBalance(money)) return NOT_ENOUGH_BALANCE;
184 187             else
185     -           {   //돈 출금
186     -               String newLog;
187     -               if (!account.getBank().equals("신한은행"))
188     -               {
189     -                   this.takeCharge(account);
190     -               }
191     -               account.setBalance(minus(account.getBalance(), money));
192     -               newLog = new Date() + " " + money + " 출금 \t( 잔액 : " + account.getBalance() + " )\n" + account.getLog();
```

```
193     -               account.setLog(newLog);
194     -
195     -               return checkUpdateDB(offer[bank]);
        188 +           {
        189 +               return withdraw_Money(money, offer[bank], account, "출금");
196     190             }
        191 +   }
        192 +
        193 +   private int withdraw_Money(String money, Offer offer, Account account, String withdrawType)
        194 +   {
        195 +       //돈 출금기능(수수료까지 출금o) + log업데이트 +  checkUpdateDB
        196 +
        197 +       String newLog;
        198 +       if (!account.getBank().equals("신한은행"))
        199 +       {
        200 +           this.takeCharge(account);
        201 +       }
        202 +       account.setBalance(minus(account.getBalance(), money));
        203 +       newLog = new Date() + " " + money + withdrawType+ "\t( 잔액 : " + account.getBalance() + " )\n" + account.getLog();
        204 +       account.setLog(newLog);
        205 +
        206 +       return checkUpdateDB(offer,account );
        207 +   }
        208 +
        209 +   private int withdraw_Money(String money, Offer offer, Account account, String withdrawType, Account receiverAccount)
        210 +   {  //Overloading
        211 +       //돈 출금기능(수수료까지 출금o) + log업데이트 +  checkUpdateDB + 수신자정보까지.
        212 +
        213 +       String newLog;
        214 +       if (!this.account.getBank().equals("신한은행"))
        215 +       {
        216 +           this.takeCharge(this.account);
197     217             }
        218 +       this.account.setBalance(minus(this.account.getBalance(), money));
        219 +       newLog = new Date() + " " + money + withdrawType+ receiverAccount.getBank() + " " + receiverAccount.getAccountNumber() + "
        220 +       this.account.setLog(newLog);
        221 +
        222 +       return checkUpdateDB(offer, account);
        223 +   }
        224 +
```

```
198  225
199  226      public int transfer(String money, Account receiveAccount)
200  227      {   //송금 시도하고 error code를 리턴.
     228  +
201  229          int bank;
202       -      bank = findWhichBank();
     230  +      bank = findWhichBank(this.account);
     231  +
     232  +      //송금하려는 account가 valid하지 않으면 오류.
     233  +      if (!offer[bank].checkValid(account)) return WRONG_ACCOUNT;
     234  +      if (isWrongAccount(bank, account)) return WRONG_ACCOUNT;
     235  +      offer[bank].readDatabase(account);
     236  +
     237  +
     238  +      //receiver의 bank 얻어오고 valid하지 않으면 오류.
     239  +      int receiver_bank;
     240  +      receiver_bank = findWhichBank(receiveAccount);
     241  +      if(isWrongAccount(receiver_bank, receiveAccount)) return WRONG_ACCOUNT;
     242  +      offer[receiver_bank].readDatabase(receiveAccount);
     243  +
     244  +
     245  +      //송금계좌의 잔액 확인
     246  +      if(isUnderBalance(money)) return NOT_ENOUGH_BALANCE;
     247  +
     248  +      int receiverUpdateCheck;
     249  +      int senderUpdateCheck;
     250  +
     251  +      if (receiveAccount.getBank().substring(2, 4).equals("카드"))
     252  +      {   //수신자가 카드인 경우: 수신자 빚 상환
     253  +          receiverUpdateCheck = deposit_CARD(money, receiveAccount, receiver_bank);
     254  +          //송금자 잔액 차감
     255  +          senderUpdateCheck = withdraw_Money(money,offer[bank],account,"송금 ", receiveAccount);
     256  +
     257  +          if(receiverUpdateCheck ==2 || senderUpdateCheck ==2)
     258  +          {//updateDB시 오류가 난 경우 거래 취소.
     259  +
     260  +              //수신자 거래 취소
     261  +              receiveAccount.setDept(plus(receiveAccount.getDept(), money));
     262  +              String newLog = new Date() + " " + money + " 기존거래 취소 \t( 대출금 : " + receiveAccount.getDept()
     263  +                      + " )\n";
     264  +              newLog = newLog + receiveAccount.getLog();
     265  +              receiveAccount.setLog(newLog);
     266  +
```

```
267 +                //송신자 거래 취소
268 +                account.setBalance(plus(account.getBalance(), money));
269 +                String tempLog = new Date() + " " + money + " 기존거래 취소 " + account.getBank() + " "
270 +                        + account.getAccountNumber() + "\t( 잔액 : " + account.getBalance() + " )\n";
271 +                tempLog = tempLog + account.getLog();
272 +                account.setLog(tempLog);
273 +                return SERVER_NOT_RESPONSE;
274 +            }
275 +            return NO_ERROR;
276 +        }
277 +        else
278 +        {   //수신자가 일반 은행 계좌인 경우: 입금.
279 +            receiverUpdateCheck = deposit_BANK(money, receiveAccount, receiver_bank);
203 280
204     -       if (! offer[bank].checkValid(account)) return WRONG_ACCOUNT;
205     -       //{
206     -           if (isWrongAccount(bank, account)) return WRONG_ACCOUNT;
281 +           //송신자에서 차감.
282 +           senderUpdateCheck = withdraw_Money(money,offer[bank],account,"송금 ", receiveAccount);
207 283
208     -           //receiverAccount가 존재하는지 확인.
209     -           int temp_bank = 0;
210     -           boolean new_account_exist = false;
211     -           while(temp_bank < 10)
284 +
285 +           if (receiverUpdateCheck == 2 || senderUpdateCheck == 2)
212 286            {
213     -               if (receiveAccount.getBank().equals(list_bank[temp_bank]))
214     -               {
215     -                   new_account_exist = true;
216     -                   break;
217     -               }
218     -               temp_bank ++;
219     -           }
220     -           if(!new_account_exist) return SERVER_NOT_RESPONSE;
287 +               //updateDB시 오류가 난 경우 거래 취소.
```

```
221 288
222     -           //------
223     -           //송금인 잔액확인
224     -           if(isUnderBalance(money)) return NOT_ENOUGH_BALANCE;
    289 +               //수신자 거래 취소
    290 +               withdraw_Money(money, offer[receiver_bank], receiveAccount, "기존 거래 취소");
    291 +               //송신자 거래 취소
    292 +               withdraw_Money(money, offer[bank], account, "기존 거래 취소");
225 293
226     -           else
227     -           {// 여기 else삭제해도 됨. 송금인 잔액이 충분하면.
228     -               if (receiveAccount.getBank().substring(2, 4).equals("카드"))
229     -               {   //수신자가 카드면?
230     -                   if(isWrongAccount(bank, receiveAccount)) return WRONG_ACCOUNT;
231     -                   //if ((this.errorType = offer[bank].readDatabase(newAccount)) != 0)
232     -                   //{
233     -                   //    return 3;
234     -                   //}
235     -                   else //여기 else삭제해도 됨.
236     -                   {
237     -                       if (isBig(money, receiveAccount.getDept()))
238     -                       {
239     -                           return DEPOSIT_OVER_DEPT;
240     -                       } else
241     -                       {   // 여기 else삭제해도 됨.그냥 여기 Else 놔두자.
242     -                           receiveAccount.setDept(minus(receiveAccount.getDept(), money));
243     -                           String newLog = new Date() + " " + money + " 상환 \t( 대출금 : " + receiveAccount.getDept()
244     -                                   + " )\n";
245     -                           newLog = newLog + receiveAccount.getLog();
246     -                           receiveAccount.setLog(newLog);
247     -
248     -                           //checkUpdateDB
249     -
250     -                           //update db 안되면 2리턴.
251     -                           // return checkUpdateDB(bank,newAccount)
252     -                           if (!offer[bank].updateDatabase(receiveAccount))
253     -                           {
254     -                               return 2;
255     -                           }
256     -
257     -
258     -                           offer[bank].readDatabase(account);
259     -                           if (!account.getBank().equals("신한은행") || !receiveAccount.getBank().equals("신한은행"))
```

```
258 -                           offer[bank].readDatabase(account);
259 -                           if (!account.getBank().equals("신한은행") || !receiveAccount.getBank().equals("신한은행"))
260 -                               takeCharge(account);
261 -                           account.setBalance(minus(account.getBalance(), money));
262 -                           newLog = new Date() + " " + money + " 송금 " + receiveAccount.getBank() + " "
263 -                                   + receiveAccount.getAccountNumber() + "\t( 잔액 : " + account.getBalance() + " )\n";
264 -                           newLog = newLog + account.getLog();
265 -                           account.setLog(newLog);
266 -                           if (offer[bank].updateDatabase(account))
267 -                           {
268 -                               return 0;
269 -                           } else
270 -                           {
271 -                               receiveAccount.setDept(plus(receiveAccount.getDept(), money));
272 -                               newLog = new Date() + " " + money + " 기존거래 취소 \t( 대출금 : " + receiveAccount.getDept()
273 -                                       + " )\n";
274 -                               newLog = newLog + receiveAccount.getLog();
275 -                               receiveAccount.setLog(newLog);
276 -                               if (!offer[bank].updateDatabase(receiveAccount))
277 -                               {
278 -                                   return 2;
279 -                               }
280 -                               return 2;
281 -                           }
282 -                       }
283 -                   }
284 -               }
285 -               receiveAccount.setBalance(plus(receiveAccount.getBalance(), money));
286 -               String tempLog = new Date() + " " + money + " 입금 " + account.getBank() + " "
287 -                       + account.getAccountNumber() + "\t( 잔액 : " + receiveAccount.getBalance() + " )\n";
288 -               tempLog = tempLog + receiveAccount.getLog();
289 -               receiveAccount.setLog(tempLog);
290 -               if (!offer[temp_bank].updateDatabase(receiveAccount))
291 -               {
292 -                   return 2;
293 -               }
294 -               offer[bank].readDatabase(account);
295 -               if (!account.getBank().equals("신한은행") || !receiveAccount.getBank().equals("신한은행"))
296 -                   takeCharge(account);
297 -               account.setBalance(minus(account.getBalance(), money));
298 -               String newLog = new Date() + " " + money + " 송금 " + receiveAccount.getBank() + " "
299 -                       + receiveAccount.getAccountNumber() + "\t( 잔액 : " + account.getBalance() + " )\n";
300 -               newLog = newLog + account.getLog();
301 -               account.setLog(newLog);
```

```
302     -               if (offer[bank].updateDatabase(account))
303     -               {
304     -                   return 0;
305     -               } else
306     -               {
307     -                   receiveAccount.setBalance(plus(receiveAccount.getBalance(), money));
308     -                   tempLog = new Date() + " " + money + " 기존거래 취소 " + account.getBank() + " "
309     -                           + account.getAccountNumber() + "\t( 잔액 : " + receiveAccount.getBalance() + " )\n";
310     -                   tempLog = tempLog + receiveAccount.getLog();
311     -                   receiveAccount.setLog(tempLog);
312     -                   if (!offer[temp_bank].updateDatabase(receiveAccount))
313     -                   {
314     -                       return 2;
315     -                   }
316     -                   return 2;
317     -               }
        294 +               return SERVER_NOT_RESPONSE;
318     295             }
319     -       // } else
320     -        //{
321     -        //    return WRONG_ACCOUNT;
322     -       // }
        296 +           return NO_ERROR;
        297 +       }
323     298     }
324     299
325     300
326     -   public int exchange(String money, String str)
        301 +
        302 +       public int exchange(String money, String str)
327     303     {
328     304         int bank;
329     -       bank = findWhichBank();
        305 +       bank = findWhichBank(account);
330     306         int[] exchangeRate = {1100, 10, 1280, 200};
331     307         String[] country = {"USD", "JPY", "EUR", "CNY"};
332     308         int i;
            @@ -346,14 +322,14 @@ public int exchange(String money, String str)
346     322                     + " )\n";
347     323             newLog = newLog + account.getLog();
348     324             account.setLog(newLog);
349     -           return checkUpdateDB(offer[bank]);
        325 +           return checkUpdateDB(offer[bank],account );
```

```
350 326             }
351 327         }
352 328
353 329         public int loan(String money)
354 330         {
355 331             int bank;
356     -           bank = findWhichBank();
    332 +           bank = findWhichBank(account);
357 333             if (isBig(plus(plus(account.getDept(), money), "1300"), account.getLimit()))
358 334             {
359 335                 return 7;
    @@ -366,7 +342,7 @@ public int loan(String money)
366 342                         + minus(account.getLimit(), account.getDept()) + " )\n";
367 343                 newLog = newLog + account.getLog();
368 344                 account.setLog(newLog);
369     -               return checkUpdateDB(offer[bank]);
    345 +               return checkUpdateDB(offer[bank],account );
370 346             }
371 347         }
372 348
    @@ -402,7 +378,7 @@ public int payUtilityBill(Account newAccount)
402 378         {
403 379
404 380
405     -           int bank = findWhichBank();
    381 +           int bank = findWhichBank(account);
406 382
407 383             if (offer[bank].checkValid(account))
408 384             {
    @@ -437,7 +413,7 @@ public int payUtilityBill(Account newAccount)
437 413                             + newAccount.getAccountNumber() + "\t( 잔액 : " + account.getBalance() + " )\n";
438 414                     newLog = newLog + account.getLog();
439 415                     account.setLog(newLog);
440     -                   return checkUpdateDB(offer[bank]);
    416 +                   return checkUpdateDB(offer[bank],account );
441 417                 }
442 418             } else
443 419             {
```