

**객체지향 프로그래밍**

**CSE 201**

**Term project (Fall 2023)**

Automated Teller Machine (ATM) System

202011125 유선우

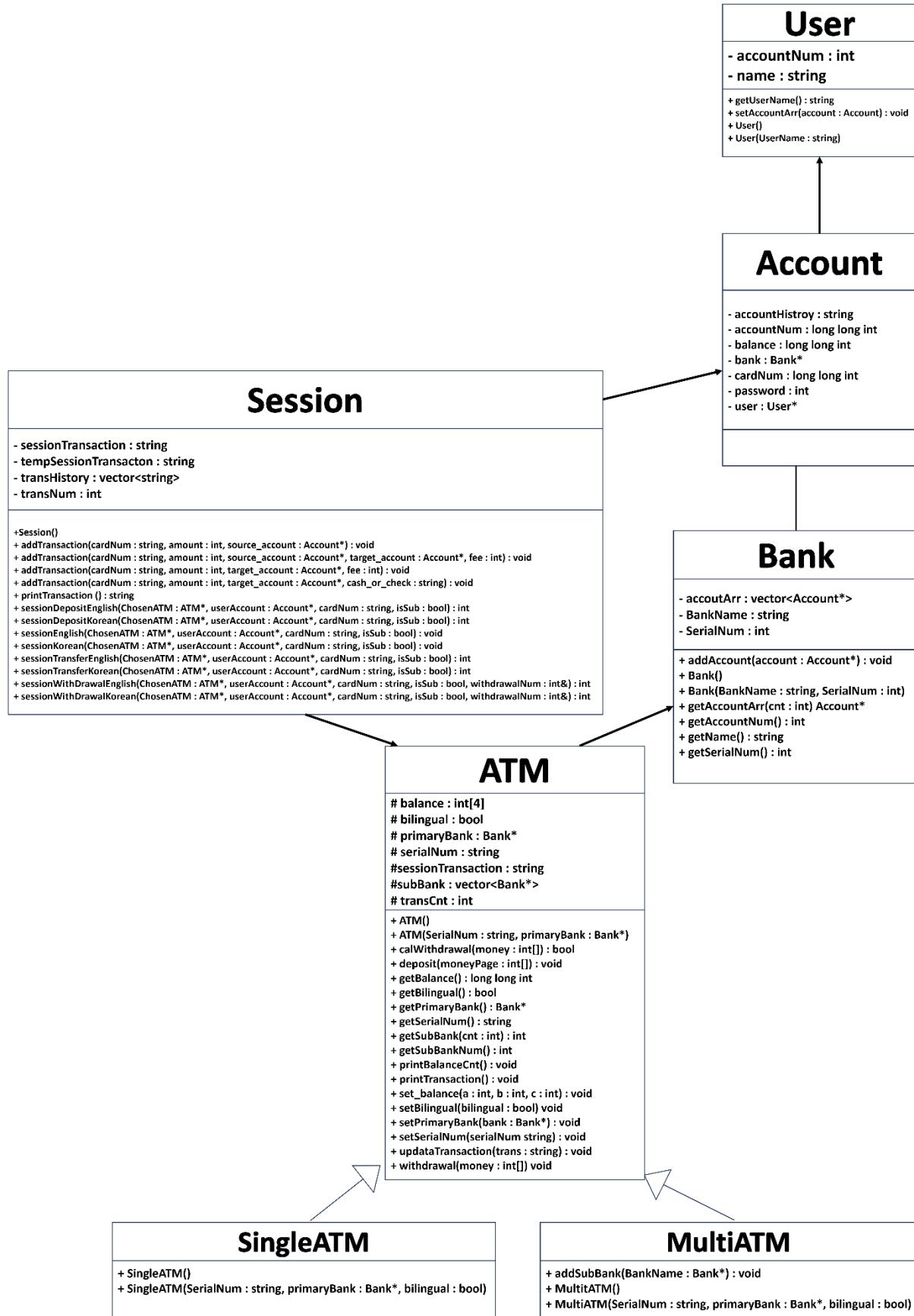
202011168 이재현

202011175 이창석

# 목 차

- 1. Class diagram design**
- 2. Implemented requirement & Console screenshot**
- 3. The list of concepts of objected-oriented programming**
- 4. Instruction**
- 5. Source code**
- 6. Member student contribution**

# 1. Class diagram design



## 2. Implemented requirement & Console screenshot

설명의 편의성을 위해 계좌를 아래와 같이 설정.

User	주거래 은행	계좌번호	카드 번호	비밀번호
철수	국민	111-111-111111	1111-1111-1111-1111	1111
철수	국민	222-222-222222	2222-2222-2222-2222	2222
철수	국민	333-333-333333	3333-3333-3333-3333	33333
영희	신한	444-444-444444	4444-4444-4444-4444	4444
영희	신한	555-555-555555	5555-5555-5555-5555	5555
철수	신한	666-666-666666	6666-6666-6666-6666	6666

ATM1 : type : Single,

language : Unilingual,

serial number : 111111

ATM2 : type : Multi,

language : Bilingual,

serial number : 222222

### REQ 1.1

- ✓ An ATM has a 6-digit serial number that can be uniquely identified among all ATMs.
  - 싱글 or 멀티 ATM에서 6자리의 시리얼 넘버를 정확히 입력한 경우

```
1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 : 1
*----- ATM 생성 -----*
1. SingleATM
2. MultiATM
1
1. Unilingual
2. Bilingual
1
ATM Serial Number를 입력해주세요 : 111111
메인 은행을 입력해주세요 : kookmin
ATM이 소유한 deposit을 입력하시오.
50,000 : 10
10,000 : 10
5,000 : 10
1,000 : 10
*----- ATM을 생성하였습니다. -----*
```

- 싱글 or 멀티 ATM에서 6자리 수 이외의 것을 입력한 경우(6자리보다 작은 경우)

```
1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 : 1

*----- ATM 생성 -----*
1. SingleATM
2. MultiATM
1
1. Unilingual
2. Bilingual
1
ATM Serial Number를 입력해주세요 : 111111
*----- 6자리 외의 Serial Number는 입력이 불가능합니다. -----*
```

- 싱글 or 멀티 ATM에서 6자리 수 이외의 것을 입력한 경우(6자리보다 큰 경우)

```
1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 : 1

*----- ATM 생성 -----*
1. SingleATM
2. MultiATM
1
1. Unilingual
2. Bilingual
1
ATM Serial Number를 입력해주세요 : 1111111
*----- 6자리 외의 Serial Number는 입력이 불가능합니다. -----*
```

- 중복되는 시리얼 넘버를 입력하는 경우

```
1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 : 1

*----- ATM 생성 -----*
1. SingleATM
2. MultiATM
1
1. Unilingual
2. Bilingual
1
ATM Serial Number를 입력해주세요 : 111111
*----- 이미 존재하는 Serial Number입니다. -----*
```

## REQ 1.2

- ✓ An ATM is set to one of the following types: (1) Single Bank ATM, (2) Multi-Bank ATM.
  - For single Bank ATM, the ATM is belonged to a primary bank, only a card issued by the primary bank is considered valid.
  - For Multi-Bank ATM, there is a primary bank that manages the ATM, but a card issued by any other bank is considered valid.
- 싱글 ATM이 제대로 생성되는 경우

```
1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 : 1  
*----- ATM 생성 -----*  
1. SingleATM  
2. MultiATM  
1  
1. Unilingual  
2. Bilingual  
1  
ATM Serial Number를 입력해주세요 : 111111  
메인 은행을 입력해주세요 : kookmin  
ATM이 소유한 deposit을 입력하시오.  
50,000 : 10  
10,000 : 10  
5,000 : 10  
1,000 : 10  
*----- ATM을 생성하였습니다. -----*
```

- 멀티 ATM이 제대로 생성되는 경우

```
1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 : 1  
*----- ATM 생성 -----*  
1. SingleATM  
2. MultiATM  
2  
1. Unilingual  
2. Bilingual  
2  
ATM Serial Number를 입력해주세요 : 222222  
메인 은행을 입력해주세요 : kookmin  
연결되어 있는 Bank를 입력해주세요 (끝나면 C입력) : sinhan  
sinhan은행이 ATM의 Sub은행으로 추가되었습니다.  
연결되어 있는 Bank를 입력해주세요 (끝나면 C입력) : c  
*----- ATM생성을 취소합니다. -----*  
ATM이 소유한 deposit을 입력하시오.  
50,000 : 10  
10,000 : 10  
5,000 : 10  
1,000 : 10  
*----- ATM을 생성하였습니다. -----*
```

### REQ 1.3

- ✓ An ATM may support either unilingual or bilingual languages.
  - When an ATM is configured unilingual, all information is displayed in English only.
  - When an ATM is configured bilingual, a user can choose if the information is to be displayed either English or Korean (Note: if you know only one of the languages, consider using a language translation service, such as Google Translation)
- 단일 언어 ATM을 선택하였기 때문에 영어가 바로 출력되는 경우

1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 : 3

\*----- ATM을 실행하였습니다. -----\*

1. Primary Bank : kookmin, Serial Number : 111111  
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 1  
Please enter your card number : 1111111111111111  
Please enter your password : 1111  
1. Deposit  
2. Withdrawal  
3. Transfer

- 이중 언어 ATM일 때 한국어나 영어가 선택이 가능하고 출력되는 경우(한글, 영어 순)

1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 : 3

\*----- ATM을 실행하였습니다. -----\*

1. Primary Bank : kookmin, Serial Number : 111111  
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2  
1. 한국어  
2. English  
1  
카드 번호를 입력해주세요 : 1111111111111111  
비밀번호를 입력해주세요 : 1111  
1. 입금  
2. 출금  
3. 송금

```

1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 : 3

*----- ATM을 실행하였습니다. -----
1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2
1. 한국어
2. English
2
Please enter your card number : 1111111111111111
Please enter your password : 1111
1. Deposit
2. Withdrawal
3. Transfer

```

#### REQ 1.4

- ✓ A Bank deposits a certain amount of cashes to an ATM to serve users.
- ATM이 갖고 있는 금액을 설정

```

*----- ATM 생성 -----
1. SingleATM
2. MultiATM
2
1. Unilingual
2. Bilingual
2
ATM Serial Number를 입력해주세요 : 222222
메인 은행을 입력해주세요 : kookmin
연결되어 있는 Bank를 입력해주세요 (끝나면 C입력) :
sinhan
sinhan은행이 ATM의 Sub은행으로 추가되었습니다.

연결되어 있는 Bank를 입력해주세요 (끝나면 C입력) :
C
*----- ATM생성을 취소합니다. -----
ATM의 소유한 deposit을 입력하시오.
50,000 : 10
10,000 : 10
5,000 : 10
1,000 : 10

```

#### REQ 1.5

- ✓ A Bank can open an Account for user with necessary information to perform bank services.
- Bank name (e.g, Kakao, Shinhan), User name, Account number (12-digit), Available

funds, Transaction histories.

- 은행은 계좌에 들어 있는 유저 정보를 열람 가능

Bank.h

```
vector<Account*> accountArr;
```

뱅크 클래스에서 계좌번호를 저장하는 vector 존재. 각 계좌 열람 가능.

### REQ 1.6

- ✓ A user may have multiple Accounts in a Bank.
- 동일한 유저가 한 은행에서 여러 개의 계좌 개설이 가능한 경우
- 유저 철수가 kookmin에서 2개의 계좌를 갖는 case

```
*----- 계좌 생성 -----*
이름을 입력하세요 : 철수
사용하실 계좌번호를 입력하세요 : 111111111111
사용하실 카드번호를 입력하세요 : 1111111111111111
사용하실 비밀번호를 입력해주세요 : 1111
전용 은행을 입력하세요 : kookmin
최초 입금액을 입력하세요 : 100000
*----- 계좌를 생성하였습니다. -----*

*----- 계좌 생성 -----*
이름을 입력하세요 : 철수
사용하실 계좌번호를 입력하세요 : 222222222222
사용하실 카드번호를 입력하세요 : 2222222222222222
사용하실 비밀번호를 입력해주세요 : 2222
전용 은행을 입력하세요 : kookmin
최초 입금액을 입력하세요 : 100000
*----- 계좌를 생성하였습니다. -----*

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 100000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 100000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 100000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 100000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 100000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 100000
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*
```

### REQ 1.7

- ✓ A user may have Accounts in multiple Banks.
- 동일한 유저가 여러 은행에서 각각의 은행 계좌를 개설할 수 있는 경우
- 유저 철수가 kookmin과 sinhan 각각의 계좌를 갖는 경우

```

*----- 계좌 생성 -----*
이름을 입력하세요 : 철수
사용하실 계좌번호를 입력하세요 : 111111111111
사용하실 카드번호를 입력하세요 : 1111111111111111
사용하실 비밀번호를 입력해주세요 : 1111
전용 은행을 입력하세요 : kookmin
최초 입금액을 입력하세요 : 100000
*----- 계좌를 생성하였습니다. -----*

*----- 계좌 생성 -----*
이름을 입력하세요 : 철수
사용하실 계좌번호를 입력하세요 : 666666666666
사용하실 카드번호를 입력하세요 : 6666666666666666
사용하실 비밀번호를 입력해주세요 : 6666
전용 은행을 입력하세요 : sinhan
최초 입금액을 입력하세요 : 100000
*----- 계좌를 생성하였습니다. -----*

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 100000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 100000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 100000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 100000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 100000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 100000
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*

```

### REQ 1.8

- ✓ Each ATM have several types of transaction fees, and paid as follows:
- 수수료 정책에 맞게 수수료가 정확하게 지불되고 있는지 여부
- (싱글atm) 주거래은행일 때 입금 수수료 0원

```
*----- ATM을 실행하였습니다. -----*
1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 1
Please enter your card number : 1111111111111111
Please enter your password : 1111
1. Deposit
2. Withdrawal
3. Transfer
1

Please choose whether to use cash or checks.
1. Cash
2. Check
1
Please deposit the cash. If it's not a bankkookmin, a fee of 1,000 won will be charged.
50,000 : 0
10,000 : 0
5,000 : 2
1,000 : 0
*----- The deposit has been completed. -----*

1. Deposit
2. Withdrawal
3. Transfer
x

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 110000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 100000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 100000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 100000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 100000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 100000
[ATM 1] Remaining Cash : 670000 (50000원X10장, 10000원X10장, 5000원X12장, 1000원X10장)
[ATM 2] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*
```

- (멀티 atm) 주거래은행일 때 입금 수수료 0원

\*----- ATM을 실행하였습니다. -----\*

1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2

1. 한국어
2. English

1

카드 번호를 입력해주세요 : 1111111111111111

비밀번호를 입력해주세요 : 1111

1. 입금
2. 출금
3. 송금

1

현금을 사용할지 수표를 사용할지 골라주세요.

1. 현금
2. 수표

1

현금을 입금해주세요. kookmin은행이 아니면 수수료 1000원이 부과됩니다.

- 50,000 : 0  
10,000 : 1  
5,000 : 0  
1,000 : 0

\*----- 입금이 완료되었습니다. -----\*

\*----- SnapShot -----\*

```
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 120000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 100000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 100000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 100000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 100000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 100000
[ATM 1] Remaining Cash : 670000 (50000원X10장, 10000원X10장, 5000원X12장, 1000원X10장)
[ATM 2] Remaining Cash : 670000 (50000원X10장, 10000원X11장, 5000원X10장, 1000원X10장)
```

\*----- SnapShot Finish -----\*

- (멀티 atm) 타행 카드 입금 수수료 1000원

- (싱글 atm) 주거래 은행 카드 출금 수수료 1000원

```
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 15000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 50000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 50000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 50000
[ATM 1] Remaining Cash : 930000 (50000원X15장, 10000원X9장, 5000원X15장, 1000원X15장)
[ATM 2] Remaining Cash : 122000 (50000원X2장, 10000원X1장, 5000원X2장, 1000원X2장)
```

\*----- SnapShot Finish -----\*

1. Deposit
2. Withdrawal
3. Transfer

2

###(The main bank is charged 1000 won and the sub-bank is charged 2000 won automatically.)###  
Please enter the amount to withdraw :

- 50,000 : 0  
10,000 : 0  
5,000 : 1  
1,000 : 0

\*----- Withdrawal is completed. -----\*

```

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 1569000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 9000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 50000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 50000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 50000
[ATM 1] Remaining Cash : 925000 (50000원X15장, 10000원X9장, 5000원X14장, 1000원X15장)
[ATM 2] Remaining Cash : 122000 (50000원X2장, 10000원X1장, 5000원X2장)
*----- SnapShot Finish -----*

```

- Single ATM(ATM 1)은 반드시 주거래 은행이므로 1,000원 수수료 발생  
2번 계좌(15,000원)에서 5,000원 출금 시, 1,000원 수수료 포함 6,000원 차감.  
2번 계좌 잔액 15,000 – 6,000 = 9,000원.

- (멀티 atm) 주거래 은행 카드 출금 수수료 1000원

```

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2229000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 10000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 50000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 50000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 50000
[ATM 1] Remaining Cash : 925000 (50000원X15장, 10000원X9장, 5000원X14장, 1000원X15장)
[ATM 2] Remaining Cash : 785000 (50000원X12장, 10000원X11장, 5000원X12장, 1000원X15장)
*----- SnapShot Finish -----*

```

1. 입금  
2. 출금  
3. 종금  
2

###(주은행은 1000원 서비스은행은 2000원의 수수료가 자동으로 부과됩니다.)###  
출금할 금액을 입력해주세요 :

50,000 : 0  
10,000 : 0  
5,000 : 1  
1,000 : 0

\*----- 출금이 완료되었습니다. -----\*

\*----- SnapShot -----\*

```

[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2229000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 4000

```

- Multi ATM(ATM 2)은 주거래 은행인 2번에서 출금 시 1,000원 수수료 발생  
2번 계좌(10,000원)에서 5,000원 출금 시, 1,000원 수수료 포함 6,000원 차감.  
2번 계좌 잔액 10,000 – 6,000 = 4,000원.

- (멀티 atm) 타행 카드 출금 수수료 2000원

```

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 50000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 50000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 50000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 38000
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 650000 (50000원X10장, 10000원X9장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*

1. 입금
2: 출금
3. 송금
2

###(주은행은 1000원, 서브은행은 2000원의 수수료가 자동으로 부과됩니다.)###
출금할 금액을 입력해주세요 :
50,000 : 0
10,000 : 1
5,000 : 0
1,000 : 0
*----- 출금이 완료되었습니다. -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 50000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 50000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 50000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 26000

```

- Multi ATM(ATM 2)은 서브 은행인 4번에서 출금 시 2,000원 수수료 발생
- 4번 계좌(38,000원)에서 10,000원 출금 시, 2,000원 수수료 포함 12,000원 차감.
- 2번 계좌 잔액  $38,000 - 12,000 = 26,000$ 원.
- 계좌 이체 수수료(주거래 은행 계좌간 이체: 2000원, 주거래와 타행일 경우: 3000원, 타행 간의 이체: 4000원)

```

1. Transfer Cash
2. Transfer account balance
2
Please enter the account number of the account to receive the remittance.
111111111111
Please enter the account number of the account you want to transfer.
222222222222
Please enter the amount to be transferred.
10000
You will be charged a fee of 2,000 won.
Transfer? [y/n]
-
```

- Single ATM(ATM 1)의 주거래 은행 계좌 간의 이체, 수수료 2000원 발생  
(타행 계좌 번호 입력 시 exception handling 참조)

```
1. 현금 전송  
2. 계좌 잔액 전송  
2  
송금받을 계좌의 계좌번호를 입력해주세요.  
111111111111  
송금할 계좌의 계좌번호를 입력해주세요.  
222222222222  
송금할 금액을 입력해주세요.  
10000  
2000원의 수수료가 부과되었습니다.  
송금하시겠습니까? [y/n]  
송금받을 계좌의 계좌번호를 입력해주세요.  
444444444444  
송금할 계좌의 계좌번호를 입력해주세요.  
111111111111  
송금할 금액을 입력해주세요.  
50000  
3000원의 수수료가 부과됩니다.  
송금하시겠습니까? [y/n]  
y  
송금받을 계좌의 계좌번호를 입력해주세요.  
555555555555  
송금할 계좌의 계좌번호를 입력해주세요.  
444444444444  
송금할 금액을 입력해주세요.  
10000  
4000원의 수수료가 부과됩니다.  
송금하시겠습니까? [y/n]  
y
```

- Multi ATM(ATM 2)

위에서부터 주거래 은행 계좌 간의 이체, 주거래와 타행의 이체, 타행 간의 이체 각각 2000원, 3000원, 4000원 수수료 부과.

#### REQ 6.6

- ✓ The inserted cash for transfer increase available cash in ATM that can be used by other users.

```
[ATM 1] Remaining Cash : 1212000 (50000원X21장, 10000원X10장, 5000원X11장, 1000원X7장)
[ATM 2] Remaining Cash : 640000 (50000원X10장, 10000원X8장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*
```

1. Deposit  
2. Withdrawal  
3. Transfer  
3

1. Transfer Cash  
2. Transfer account balance  
1

Please enter the account number of the account to receive the remittance.  
333333333333

A fee of 5,000 won will be incurred. Do you want to proceed? [y/n]  
y

Please enter the amount to be transferred :

50,000 : 1  
10,000 : 0  
5,000 : 0  
1,000 : 0

Please choose how to pay the fee of 5000 won.  
1. 5000Won X 1  
2. 1000Won X 5  
1

```
[ATM 1] Remaining Cash : 1267000 (50000원X22장, 10000원X10장, 5000원X12장, 1000원X7장)
[ATM 2] Remaining Cash : 640000 (50000원X10장, 10000원X8장, 5000원X10장, 1000원X10장)
```

Please choose how to pay the fee of 5000 won.

1. 5000Won X 1  
2. 1000Won X 5  
2

```
[ATM 1] Remaining Cash : 1322000 (50000원X23장, 10000원X10장, 5000원X12장, 1000원X12장)
[ATM 2] Remaining Cash : 640000 (50000원X10장, 10000원X8장, 5000원X10장, 1000원X10장)
```

- Single ATM(ATM 1)에 현금 이체로 입금된 현금은 atm의 잔고 반영(수수료 포함)  
50,000원 입금 시 수수료 5,000원 포함 입금  
(5,000원으로 지불 시) ATM 50,000원 21장 -> 22장, 5,000원 11장 -> 12장  
(1,000원 X 5으로 지불 시) ATM 50,000원 22장 -> 23장, 1,000원 7장 -> 12장

```

[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 675000 (50000원X10장, 10000원X11장, 5000원X11장, 1000원X10장)
*----- SnapShot Finish -----*
1. 입금
2. 출금
3. 송금
3

1. 현금 전송
2. 계좌 잔액 전송
1
송금받을 계좌의 계좌번호를 입력해주세요.
111111111111
5000원의 수수료가 발생합니다. 진행하시겠습니까? [y/n]
y
송금할 금액을 입력해주세요 :
50,000 : 1
10,000 : 0
5,000 : 0
1,000 : 0

수수료 5000원을 지불할 방식을 선택해주세요.
1. 5000원 X 1
2. 1000원 X 5
1
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 730000 (50000원X11장, 10000원X11장, 5000원X12장, 1000원X10장)

```

#### 같은 방식으로 50,000원 추가 입금 시

```

[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 785000 (50000원X12장, 10000원X11장, 5000원X12장, 1000원X15장)

- Multi ATM(ATM 2)에 현금 이체로 입금된 현금은 atm의 잔고 반영(수수료 포함)
    (5,000원으로 지불 시) ATM 50,000원 10장 -> 11장, 5,000원 11장 -> 12장
    (1,000원 X 5으로 지불 시) ATM 50,000원 11장 -> 12장, 1,000원 10장 -> 15장

```

#### REQ. 1.9

- ✓ An admin can access the menu of "Transaction History" via an admin card (See REQ Display of Transaction History).
- 관리자 카드 번호(0000) 입력 시 거래 내역 출력

```

1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 : 3
*----- ATM을 실행하였습니다. -----
1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2
1. 한국어
2. English
1
카드 번호를 입력해주세요 : 0000
1. Transaction History
1
*----- Transaction History -----
1. [Deposit Cash] Card Number : 1111111111111111, Deposit amount : 66000, Balance after Deposit : 166000
2. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 66000, Balance after Withdrawal : 99000
3. [Deposit Cash] Card Number : 1111111111111111, Deposit amount : 10000, Balance after Deposit : 109000
*----- Transaction History Finish -----

```

### REQ 1.10

- ✓ An ATM only accepts and returns the following types of cashes and checks.
  - (Cash type) KRW 1,000, KRW 5,000, KRW 10,000, KRW 50,000
  - (Check type) Any amount over KRW 100,000 check (e.g., KRW 100,000, 100,001, 234,567 are all valid checks)
- 현금의 경우 오만원권, 만원권, 오천원권, 천원권만 지정해서 입출금 해야 함.

```

현금을 사용할지 수표를 사용할지 골라주세요.
1. 현금
2. 수표
1
현금을 입금해주세요. kookmin은행이 아니면 수수료 1000원이 부과됩니다.
50,000 : 0
10,000 : 0
5,000 : 0
1,000 : 0
*----- 입금이 완료되었습니다 -----

```

- 수표는 십만원 이상의 금액만 가능
- 십만원 아래의 수표로 입금되지 않고 오류메시지를 출력하는 경우

```

1. 원금
2. 통장
3. 충전
1
현금을 사용할지 수표를 사용할지 골라주세요.
1. 현금
2. 수표
2
각 수표의 값을 입력해주세요. C가 입력되면 입금이 종료됩니다.
10000
*----- 입금하신 금액은 수표에 맞지 않는 금액입니다. 100,000원 이상의 수표를 입금해주세요. -----

```

- 십만원 이상의 수표를 올바르게 입금 받는 경우

```

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2109000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 100000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 100000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 100000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 100000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 100000
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 670000 (50000원X10장, 10000원X11장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*

1. 입금
2. 출금
3. 송금
1

현금을 사용할지 수표를 사용할지 골라주세요.
1. 현금
2. 수표
2
각 수표의 값을 입력해주세요. C가 입력되면 입금이 종료됩니다.
200000
C
*----- 입금이 완료되었습니다. -----*

1. 입금
2. 출금
3. 송금
x

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2309000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 100000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 100000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 100000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 100000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 100000
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 670000 (50000원X10장, 10000원X11장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*

```

### REQ 1.11

- ✓ All accounts and ATMs shall be created and initialized during the program execution.
  - During the program execution, the necessary information to create accounts and ATMs shall be given from a user via console input (i.e., hard coding of account and ATM information is not allowed).
  - The accounts and ATMs shall be created and initialized based on the user input.
    - 계좌와 ATM 초기화는 콘솔 입력을 통해 이루어져야 함
    - ATM 생성 과정

1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 : 1

\*----- ATM 생성 -----\*

1. SingleATM  
2. MultiATM  
1  
1. Unilingual  
2. Bilingual  
1  
ATM Serial Number를 입력해주세요 : 111111  
메인 은행을 입력해주세요 : kookmin  
ATM이 소유한 deposit을 입력하시오.  
50,000 : 10  
10,000 : 10  
5,000 : 10  
1,000 : 10  
\*----- ATM을 생성하였습니다. -----\*

- 계좌 생성 과정

1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 : 2

\*----- 계좌 생성 -----\*

이름을 입력하세요 : 철수  
사용하실 계좌번호를 입력하세요 : 111111111111  
사용하실 카드번호를 입력하세요 : 1111111111111111  
사용하실 비밀번호를 입력해주세요 : 1111  
전용 은행을 입력하세요 : kookmin  
최초 입금액을 입력하세요 : 100000  
\*----- 계좌를 생성하였습니다. -----\*

1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 : 2

\*----- 계좌 생성 -----\*

이름을 입력하세요 : 철수  
사용하실 계좌번호를 입력하세요 : 222222222222  
사용하실 카드번호를 입력하세요 : 2222222222222222  
사용하실 비밀번호를 입력해주세요 : 2222  
전용 은행을 입력하세요 : kookmin  
최초 입금액을 입력하세요 : 100000  
\*----- 계좌를 생성하였습니다. -----\*

## REQ 2.1

- ✓ A session starts when a user inserts a card.
- ATM 시작 이후 카드 입력란 출력 여부

```
1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 : 3

----- ATM을 실행하였습니다. -----
1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2
1. 한국어
2. English
1
카드 번호를 입력해주세요 :
```

## REQ 2.2

- ✓ A session ends whenever a user wishes (e.g., by choosing a cancel button) or there are some exceptional conditions detected by the ATM (e.g., no cash available).
- 사용자가 원할 때, 예외 발생시 종료(메뉴, 언어 선택, 거래종류 선택, 현금/수표 선택)

```
1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 : c

----- 프로그램이 종료됩니다. -----

1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 : 3

----- ATM을 실행하였습니다. -----
1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2
1. 한국어
2. English
c
----- ATM이 종료됩니다. -----*
```

\*----- ATM을 실행하였습니다. -----\*

1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2

1. 한국어
2. English

1

카드 번호를 입력해주세요 : 1111111111111111

비밀번호를 입력해주세요 : 1111

1. 입금

2. 출금

3. 송금

1

현금을 사용할지 수표를 사용할지 골라주세요.

1. 현금

2. 수표

c

잘못된 입력입니다.

\*----- ATM이 종료됩니다. -----\*

\*----- Session Transaction -----\*

\*----- Session Transaction Finish -----\*

1. ATM 생성

2. 계좌 생성

3. ATM 실행

실행 메뉴를 선택해주세요 : 3

\*----- ATM을 실행하였습니다. -----\*

1. Primary Bank : kookmin, Serial Number : 111111

2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2

1. 한국어

2. English

1

카드 번호를 입력해주세요 : 1111111111111111

비밀번호를 입력해주세요 : 1111

1. 입금

2. 출금

3. 송금

c

\*----- ATM이 종료됩니다. -----\*

\*----- Session Transaction -----\*

\*----- Session Transaction Finish -----\*

### REQ 2.3

- ✓ When a session ends, the summary of all transactions performed in a session must be displayed.
  - Account/card info, transaction types (deposit, transfer, withdrawal), and their amount, ...
- (싱글 atm) ATM이 종료될 때 모든 거래(입금, 이체, 출금)의 기록이 출력되어야 함

```
1. 입금  
2. 출금  
3. 송금  
c  
*----- ATM이 종료됩니다. -----*  
*----- Session Transaction -----*  
3. [Deposit Cash] Card Number : 1111111111111111, Deposit amount : 10000, Balance after Deposit : 113000  
4. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 10000, Balance after Withdrawal : 103000  
5. [Withdrawal] Card Number : 1111111111111111, Reciever Account Number : 222222222222, Transfer amount : 10000, Fee : 5000  
*----- Session Transaction Finish -----*
```

- (멀티 atm) ATM이 종료될 때 모든 거래(입금, 이체, 출금)의 기록이 출력되어야 함

```
1. 입금  
2. 출금  
3. 송금  
c  
*----- ATM이 종료됩니다. -----*  
*----- Session Transaction -----*  
6. [Deposit Cash] Card Number : 1111111111111111, Deposit amount : 10000, Balance after Deposit : 112000  
7. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 10000, Balance after Withdrawal : 101000  
8. [Withdrawal] Card Number : 1111111111111111, Reciever Account Number : 222222222222, Transfer amount : 10000, Fee : 5000  
*----- Session Transaction Finish -----*
```

### REQ 2.4

- ✓ Each transaction has a unique identifier across all sessions.
- 각각의 거래에 거래 번호가 붙어야 함. 모든 atm을 포함하여 붙는 번호

```
*----- Transaction History -----*  
1. [Deposit Cash] Card Number : 1111111111111111, Deposit amount : 10000, Balance after Deposit : 110000  
2. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 6000, Balance after Withdrawal : 103000  
3. [Deposit Cash] Card Number : 1111111111111111, Deposit amount : 10000, Balance after Deposit : 113000  
4. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 10000, Balance after Withdrawal : 102000  
5. [Withdrawal] Card Number : 1111111111111111, Reciever Account Number : 222222222222, Transfer amount : 10000, Fee : 5000  
6. [Deposit Cash] Card Number : 1111111111111111, Deposit amount : 10000, Balance after Deposit : 112000  
7. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 10000, Balance after Withdrawal : 101000  
8. [Withdrawal] Card Number : 1111111111111111, Reciever Account Number : 222222222222, Transfer amount : 10000, Fee : 5000  
*----- Transaction History Finish -----*
```

### REQ 3.1

- ✓ An ATM checks if the inserted card is valid for the current type of ATM.
  - See REQ in System Setup which card is considered valid
- (싱글 atm) 카드번호가 정확히 기입된 경우

```
1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 : 3
```

```
*----- ATM을 실행하였습니다. -----*  
1. Primary Bank : kookmin, Serial Number : 111111  
2. Primary Bank : kookmin, Serial Number : 222222  
3. Primary Bank : kookmin, Serial Number : 333333
```

```
Enter the number of the ATM you want to use : 1  
Please enter your card number : 1111111111111111  
Please enter your password : 1111  
1. Deposit  
2. Withdrawal  
3. Transfer
```

- (멀티 atm) 주거래은행 카드번호가 정확히 기입된 경우

```
1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 : 3
```

```
*----- ATM을 실행하였습니다. -----*  
1. Primary Bank : kookmin, Serial Number : 111111  
2. Primary Bank : kookmin, Serial Number : 222222  
3. Primary Bank : kookmin, Serial Number : 333333
```

```
Enter the number of the ATM you want to use : 2  
1. 한국어  
2. English  
1  
카드 번호를 입력해주세요 : 1111111111111111  
비밀번호를 입력해주세요 : 1111  
1. 입금  
2. 출금  
3. 송금
```

- (멀티 atm) 타행 카드번호가 정확히 기입된 경우

```
1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 : 3  
----- ATM을 실행하였습니다. -----  
1. Primary Bank : kookmin, Serial Number : 111111  
2. Primary Bank : kookmin, Serial Number : 222222  
3. Primary Bank : kookmin, Serial Number : 333333  
  
Enter the number of the ATM you want to use : 2  
1. 한국어  
2. English  
1  
카드 번호를 입력해주세요 : 6666666666666666  
비밀번호를 입력해주세요 : 6666  
1. 입금  
2. 출금  
3. 송금
```

### REQ 3.2

- ✓ If an invalid card is inserted, the ATM shall display an appropriate error message (e.g., Invalid Card).
  - (싱글 atm) 카드번호가 잘못 기입된 경우

```
1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 : 3  
----- ATM을 실행하였습니다. -----  
1. Primary Bank : kookmin, Serial Number : 111111  
2. Primary Bank : kookmin, Serial Number : 222222  
3. Primary Bank : kookmin, Serial Number : 333333  
  
Enter the number of the ATM you want to use : 1  
Please enter your card number : 1111  
----- Invalid card number. -----
```

- (멀티 atm) 카드 번호가 잘못 기입된 경우

```
1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 : 3  
----- ATM을 실행하였습니다. -----  
1. Primary Bank : kookmin, Serial Number : 111111  
2. Primary Bank : kookmin, Serial Number : 222222  
3. Primary Bank : kookmin, Serial Number : 333333  
Enter the number of the ATM you want to use : 2  
1. 한국어  
2. English  
1  
카드 번호를 입력해주세요 : 1111  
----- 유효하지 않은 카드번호입니다. -----
```

### REQ 3.3

- ✓ An ATM shall ask a user to enter the password (e.g., Enter Password), and verify if the password is correct.
  - An ATM does not maintain any user information, so the card and password information need to be passed to the bank; then, the bank verifies the password, and return the authorization result.
- (싱글 atm)비밀번호 입력을 성공했을 때(순서대로1회차 성공, 2회차 성공, 3회차 성공)

```
1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 : 3  
----- ATM을 실행하였습니다. -----  
1. Primary Bank : kookmin, Serial Number : 111111  
2. Primary Bank : kookmin, Serial Number : 222222  
3. Primary Bank : kookmin, Serial Number : 333333  
Enter the number of the ATM you want to use : 1  
Please enter your card number : 1111111111111111  
Please enter your password : 1111  
1. Deposit  
2. Withdrawal  
3. Transfer
```

- 1. ATM 생성
- 2. 계좌 생성
- 3. ATM 실행

실행 메뉴를 선택해주세요 : 3

\*----- ATM을 실행하였습니다. -----\*

- 1. Primary Bank : kookmin, Serial Number : 111111
- 2. Primary Bank : kookmin, Serial Number : 222222
- 3. Primary Bank : kookmin, Serial Number : 333333

Enter the number of the ATM you want to use : 1

Please enter your card number : 1111111111111111

Please enter your password : 1112

Incorrect. You have 2 chances left.

Please enter your password : 1111

- 1. Deposit
- 2. Withdrawal
- 3. Transfer

- 1. ATM 생성
- 2. 계좌 생성
- 3. ATM 실행

실행 메뉴를 선택해주세요 : 3

\*----- ATM을 실행하였습니다. -----\*

- 1. Primary Bank : kookmin, Serial Number : 111111
- 2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 1

Please enter your card number : 1111111111111111

Please enter your password : 1112

Incorrect. You have 2 chances left.

Please enter your password : 1113

Incorrect. You have 1 chances left.

Please enter your password : 1111

- 1. Deposit
- 2. Withdrawal
- 3. Transfer

- (멀티 atm)비밀번호 입력을 성공했을 때(순서대로1회차 성공, 2회차 성공, 3회차 성공)

1. ATM 생성
  2. 계좌 생성
  3. ATM 실행
- 실행 메뉴를 선택해주세요 : 3

\*----- ATM을 실행하였습니다. -----\*

1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2

1. 한국어
  2. English
- 1

카드 번호를 입력해주세요 : 1111111111111111

비밀번호를 입력해주세요 : 1111

1. 입금
2. 출금
3. 송금

\*----- ATM을 실행하였습니다. -----\*

1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2

1. 한국어
  2. English
- 1

카드 번호를 입력해주세요 : 1111111111111111

비밀번호를 입력해주세요 : 1112

틀렸습니다. 2번의 기회가 남았습니다.

비밀번호를 입력해주세요 : 1111

1. 입금
2. 출금
3. 송금

```
1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 : 3  
----- ATM을 실행하였습니다. -----  
1. Primary Bank : kookmin, Serial Number : 111111  
2. Primary Bank : kookmin, Serial Number : 222222  
Enter the number of the ATM you want to use : 2  
1. 한국어  
2. English  
1  
카드 번호를 입력해주세요 : 1111111111111111  
비밀번호를 입력해주세요 : 1112  
틀렸습니다. 2번의 기회가 남았습니다.  
비밀번호를 입력해주세요 : 1113  
틀렸습니다. 1번의 기회가 남았습니다.  
비밀번호를 입력해주세요 : 1111  
1. 입금  
2. 출금  
3. 송금
```

#### REQ 3.4

- ✓ If the entered password is incorrect, the ATM shall display an appropriate error message (e.g., Wrong Password).
- (싱글 atm)잘못된 비밀번호를 입력했을 경우 적절한 오류 메시지 출력(순서대로 1,2 회 실패)

```
1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 : 3  
----- ATM을 실행하였습니다. -----  
1. Primary Bank : kookmin, Serial Number : 111111  
2. Primary Bank : kookmin, Serial Number : 222222  
Enter the number of the ATM you want to use : 1  
Please enter your card number : 1111111111111111  
Please enter your password : 1112  
Incorrect. You have 2 chances left.  
Please enter your password :
```

```
1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 : 3

*----- ATM을 실행하였습니다. -----
1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 1
Please enter your card number : 1111111111111111
Please enter your password : 1112
Incorrect. You have 2 chances left.
Please enter your password : 1113
Incorrect. You have 1 chances left.
Please enter your password :
```

- (멀티 atm)잘못된 비밀번호를 입력했을 경우 적절한 오류 메시지 출력(순서대로 1,2 회 실패)

```
1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 : 3

*----- ATM을 실행하였습니다. -----
1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2
1. 한국어
2. English
1
카드 번호를 입력해주세요 : 1111111111111111
비밀번호를 입력해주세요 : 1112
틀렸습니다. 2번의 기회가 남았습니다.
비밀번호를 입력해주세요 :
```

```
*----- ATM을 실행하였습니다. -----*
1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2
1. 한국어
2. English
1
카드 번호를 입력해주세요 : 1111111111111111
비밀번호를 입력해주세요 : 1112
틀렸습니다. 2번의 기회가 남았습니다.
비밀번호를 입력해주세요 : 1113
틀렸습니다. 1번의 기회가 남았습니다.
비밀번호를 입력해주세요 :
```

### REQ 3.5

- ✓ If a user enters wrong passwords 3 times in a row, a session is aborted, and return the card to the user.
- (싱글 atm)비밀번호 3회 실패 시 세션이 중단되고 출력되는 메시지

```
1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 : 3

*----- ATM을 실행하였습니다. -----*
1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 1
Please enter your card number : 1111111111111111
Please enter your password : 1112
Incorrect. You have 2 chances left.
Please enter your password : 1113
Incorrect. You have 1 chances left.
Please enter your password : 1113
*----- Invalid, ATM execution will end. -----*
```

- (멀티 atm)비밀번호 3회 실패 시 세션이 중단되고 출력되는 메시지

```
*----- ATM을 실행하였습니다. -----*
1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2
1. 한국어
2. English
1
카드 번호를 입력해주세요 : 1111111111111111
비밀번호를 입력해주세요 : 1112
틀렸습니다. 2번의 기회가 남았습니다.
비밀번호를 입력해주세요 : 1113
틀렸습니다. 1번의 기회가 남았습니다.
비밀번호를 입력해주세요 : 1114
*----- 틀렸습니다. ATM 실행이 종료됩니다. -----*
```

#### REQ 4.1

- ✓ An ATM shall take either cash or check from a user.
- (싱글 atm) ATM 실행 시 입금 종류를 묻는 상황

```
1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 : 3

*----- ATM을 실행하였습니다. -----*
1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 1
Please enter your card number : 1111111111111111
Please enter your password : 1111
1. Deposit
2. Withdrawal
3. Transfer
1

Please choose whether to use cash or checks.
1. Cash
2. Check
```

- (멀티 atm) ATM 실행 시 입금 종류를 묻는 상황

```

*----- ATM을 실행하였습니다. -----*
1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2
1. 한국어
2. English
1
카드 번호를 입력해주세요 : 1111111111111111
비밀번호를 입력해주세요 : 1111
1. 입금
2. 출금
3. 송금
1

현금을 사용할지 수표를 사용할지 골라주세요.
1. 현금
2. 수표

```

#### REQ 4.2

- ✓ An ATM shall display an appropriate error message if the number of the inserted cash or checks exceed the limit allowed by the ATM.
- 현금의 경우 총 50장 이내, 수표는 30장 이내로 입금 가능. 이 양을 초과해서 입금 될 시 오류 메시지 출력
- 현금의 경우 각 지폐마다 50장이 넘는 경우, 모든 지폐의 합이 50장이 넘는 경우

```

1. 입금
2. 출금
3. 송금
1

현금을 사용할지 수표를 사용할지 골라주세요.
1. 현금
2. 수표
1
현금을 입금해주세요. sinhan은행이 아니면 수수료 1000원이 부과됩니다.
50,000 : 51
10,000 : 0
5,000 : 0
1,000 : 0
*----- 너무 많은 현금이 입금되었습니다. 입금 실패!! -----*
*----- ATM이 종료됩니다. -----*
*----- Session Transaction -----*
*----- Session Transaction Finish -----*

1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 :

```

1. 입금  
2. 출금  
3. 송금  
1

현금을 사용할지 수표를 사용할지 골라주세요.

1. 현금  
2. 수표  
1

현금을 입금해주세요. sinhan은행이 아니면 수수료 1000원이 부과됩니다.

50,000 : 0  
10,000 : 51  
5,000 : 0  
1,000 : 0

\*----- 너무 많은 현금이 입금되었습니다. 입금 실패!! -----\*

\*----- ATM이 종료됩니다. -----\*

\*----- Session Transaction -----\*

\*----- Session Transaction Finish -----\*

1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 :

1. 입금  
2. 출금  
3. 송금  
1

현금을 사용할지 수표를 사용할지 골라주세요.

1. 현금  
2. 수표  
1

현금을 입금해주세요. sinhan은행이 아니면 수수료 1000원이 부과됩니다.

50,000 : 0  
10,000 : 0  
5,000 : 51  
1,000 : 0

\*----- 너무 많은 현금이 입금되었습니다. 입금 실패!! -----\*

\*----- ATM이 종료됩니다. -----\*

\*----- Session Transaction -----\*

\*----- Session Transaction Finish -----\*

1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행 메뉴를 선택해주세요 :

```
1. 입금  
2. 출금  
3. 송금  
1  
  
현금을 사용할지 수표를 사용할지 골라주세요.  
1. 현금  
2. 수표  
1  
현금을 입금해주세요. sinhan은행이 아니면 수수료 1000원이 부과됩니다.  
50,000 : 0  
10,000 : 0  
5,000 : 0  
1,000 : 51  
*----- 너무 많은 현금이 입금되었습니다. 입금 실패!! -----*  
*----- ATM이 종료됩니다. -----*  
*----- Session Transaction -----*  
  
*----- Session Transaction Finish -----*  
  
1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
실행메뉴를 선택해주세요 :  
1. 입금  
2. 출금  
3. 송금  
1  
  
현금을 사용할지 수표를 사용할지 골라주세요.  
1. 현금  
2. 수표  
1  
현금을 입금해주세요. sinhan은행이 아니면 수수료 1000원이 부과됩니다.  
50,000 : 13  
10,000 : 13  
5,000 : 13  
1,000 : 13  
*----- 너무 많은 현금이 입금되었습니다. 입금 실패!! -----*  
*----- ATM이 종료됩니다. -----*  
*----- Session Transaction -----*  
  
*----- Session Transaction Finish -----*  
  
1. ATM 생성  
2. 계좌 생성  
3. ATM 실행  
시작 메뉴를 선택해주세요 :
```

- 수표의 경우 십만원 이상의 금액들이 30장을 초과하는 경우

```
*----- ATM을 실행하였습니다. -----*
1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2
1. 한국어
2. English
1
카드 번호를 입력해주세요 : 1111111111111111
비밀번호를 입력해주세요 : 1111
1. 입금
2. 출금
3. 송금
1

현금을 사용할지 수표를 사용할지 골라주세요.
1. 현금
2. 주표
2
각 수표의 값을 입력해주세요. C가 입력되면 입금이 종료됩니다.

100000
100000
100000
100000
100005
100006
100007
100008
100009
1
*----- 입금하신 금액은 수표에 맞지 않는 금액입니다. 100,000원 이상의 수표를 입금해주세요. -----*

100010
100011
100012
100013
100014
100015
100016
100017
100018
100019
100020
100021
100022
100023
100024
100025
100026
100027
100028
100029
100030
100031
100032
100033
C
*----- 입금이 완료되었습니다. -----*

*----- 너무 많은 수표가 입금되었습니다. 입금 실패!! -----*
*----- ATM이 종료됩니다. -----*
*----- Session Transaction -----*

*----- Session Transaction Finish -----*

1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 :
```

REQ 4.3

- ✓ Once cash or checks are accepted by ATM, the transaction must be reflected to the bank account as well (i.e., the same amount of fund must be added to the corresponding bank account).

- 현금 입금 시 계좌 반영 여부

```
1. 입금  
2. 출금  
3. 송금  
1  
  
현금을 사용할지 수표를 사용할지 골라주세요.  
1. 현금  
2. 수표  
1  
현금을 입금해주세요. sinhan은행이 아니면 수수료 1000원이 부과됩니다.  
50,000 : 2  
10,000 : 2  
5,000 : 2  
1,000 : 2  
1000원 : 3장, 5000원 : 3장, 10000원 : 3장, 50000원 : 3장  
1. 입금  
2. 출금  
3. 송금  
x  
----- SnapShot -----  
[Account 1] Balance : 232000  
[ATM 1] Remaining Cash : 198000 (50000원X3장, 10000원X3장, 5000원X3장, 1000원X3장)  
----- SnapShot Finish -----
```

- 수표 입금 시 계좌 반영 여부

```

*----- Snapshot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2309000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 100000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 150000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 100000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 100000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 100000
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 720000 (50000원X11장, 10000원X11장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*

1. 입금
2. 출금
3. 송금
1

현금을 사용할지 수표를 사용할지 골라주세요.
1. 현금
2. 수표
2
각 수표의 값을 입력해주세요. 0가 입력되면 입금이 종료됩니다.
200000
C
*----- 입금이 완료되었습니다. -----*

1. 입금
2. 출금
3. 송금
x

*----- Snapshot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2309000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 100000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 350000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 100000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 100000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 100000
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 720000 (50000원X11장, 10000원X11장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*

1. 입금
2. 출금
3. 송금

```

#### REQ 4.4

- ✓ Some deposit fee may be charged (See REQ in System Setup)
  - 입금 수수료 : 프라이머리 은행 카드일 경우 없음, 타행 카드일 경우 1000원
  - (싱글 atm) 주거래 은행카드 입금 수수료 0원

```
*----- SnapShot -----*
[Account 1] Balance : 1000000
[ATM 1] Remaining Cash : 1320000 (50000원X20장, 10000원X20장, 5000원X20장, 1000원X20장)
*----- SnapShot Finish -----*

1. 입금
2. 출금
3. 종료
1

현금을 사용할지 수표를 사용할지 골라주세요.
1. 현금
2. 수표
1
현금을 입금해주세요. sinhan은행이 아니면 수수료 1000원이 부과됩니다.
50,000 : 0
10,000 : 0
5,000 : 0
1,000 : 1
1000원 : 21장, 5000원 : 20장, 10000원 : 20장, 50000원 : 20장
1. 입금
2. 출금
3. 종료
x

*----- SnapShot -----*
[Account 1] Balance : 1001000
[ATM 1] Remaining Cash : 1321000 (50000원X20장, 10000원X20장, 5000원X20장, 1000원X21장)
*----- SnapShot Finish -----*
```

#### - (멀티 atm) 주거래 은행카드 입금 수수료 0원

```
*----- ATM을 실행하였습니다. -----*
1. Primary Bank : kookmin, Serial Number : 111111
2. Primary Bank : kookmin, Serial Number : 222222

Enter the number of the ATM you want to use : 2
1. 한국어
2. English
1
카드 번호를 입력해주세요 : 1111111111111111
비밀번호를 입력해주세요 : 1111
1. 입금
2. 출금
3. 종료
1

현금을 사용할지 수표를 사용할지 골라주세요.
1. 현금
2. 수표
1
현금을 입금해주세요. kookmin은행이 아니면 수수료 1000원이 부과됩니다.
50,000 : 0
10,000 : 1
5,000 : 0
1,000 : 0
*----- 입금이 완료되었습니다. -----*
```

```

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 120000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 100000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 100000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 100000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 100000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 100000
[ATM 1] Remaining Cash : 670000 (50000원X10장, 10000원X10장, 5000원X12장, 1000원X10장)
[ATM 2] Remaining Cash : 670000 (50000원X10장, 10000원X11장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*

```

- (멀티 atm) 주거래 은행카드 입금 수수료 1000원

#### REQ 4.5

- ✓ The deposited cash increase available cash in ATM that can be used by other users.
- 현금 입금 시 그 금액에 맞게 ATM의 잔고도 증가했는지 확인
- (싱글 atm) 주거래은행 카드 입금 수수료 0원 atm 반영 여부

```

*----- SnapShot -----*
[Account 1] Balance : 232000
[ATM 1] Remaining Cash : 198000 (50000원X3장, 10000원X3장, 5000원X3장, 1000원X3장)
*----- SnapShot Finish -----*

1. 일금
2. 출금
3. 종금
1

현금을 사용할지 수표를 사용할지 골라주세요.
1. 현금
2. 주표
1

현금을 입금해주세요. sinhan은행이 아니면 수수료 1000원이 부과됩니다.
50,000 : 1
10,000 : 1
5,000 : 1
1,000 : 1
1000원 : 4장, 5000원 : 4장, 10000원 : 4장, 50000원 : 4장
1. 일금
2. 출금
3. 종금
x

*----- SnapShot -----*
[Account 1] Balance : 298000
[ATM 1] Remaining Cash : 264000 (50000원X4장, 10000원X4장, 5000원X4장, 1000원X4장)
*----- SnapShot Finish -----*

```

- (멀티 atm) 주거래은행 카드 입금 수수료 0원 atm 반영 여부

```
*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2309000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 100000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 360000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 100000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 100000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 109000
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 740000 (50000원X11장, 10000원X13장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*
```

1. 입금  
2. 출금  
3. 종료  
1

현금을 사용할지 수표를 사용할지 골라주세요.

1. 현금  
2. 주표  
1

현금을 입금해주세요. kookmin은행이 아니면 수수료 1000원이 부과됩니다.

50,000 : 0  
10,000 : 1  
5,000 : 0  
1,000 : 0

```
*----- 입금이 완료되었습니다. -----*
```

1. 입금  
2. 출금  
3. 종료  
x

```
*----- SnapShot -----*
```

```
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2309000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 110000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 360000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 100000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 100000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 109000
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 750000 (50000원X11장, 10000원X14장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*
```

- (멀티 atm) 타행카드 입금 수수료 1000원 atm 반영 여부

```

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2309000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 100000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 350000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 100000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 100000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 100000
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 720000 (50000원X11장, 10000원X11장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*

1. 입금
2. 출금
3. 종금
1

현금을 사용할지 수표를 사용할지 골라주세요.
1. 현금
2. 수표
1
현금을 입금해주세요. kookmin은행이 아니면 수수료 1000원이 부과됩니다.
50,000 : 0
10,000 : 1
5,000 : 0
1,000 : 0
*----- 입금이 완료되었습니다. -----*

1. 입금
2. 출금
3. 종금
x

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2309000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 100000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 350000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 100000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 100000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 109000
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 730000 (50000원X11장, 10000원X12장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*

```

## REQ 4.6

- ✓ The deposited check does not increase available cash in ATM that can be used by other users.
- 수표 입금 시 ATM의 잔고는 변동 없음.

```

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2309000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 110000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 360000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 100000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 100000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 109000
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 750000 (50000원X11장, 10000원X14장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*

1. 입금
2. 출금
3. 송금
1

현금을 사용할지 수표를 사용할지 골라주세요.
1. 현금
2. 수표
2
각 수표의 값을 입력해주세요. C가 입력되면 입금이 종료됩니다.
100000
C
*----- 입금이 완료되었습니다. -----*

1. 입금
2. 출금
3. 송금
x

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2309000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 210000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 360000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 100000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 100000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 109000
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 750000 (50000원X11장, 10000원X14장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*

```

## REQ 5.1

- ✓ An ATM shall ask a user to enter the amount of fund to withdraw.

```

*----- ATM을 실행하였습니다. -----*
1. Primary Bank : 국민, Serial Number : 111111
2. Primary Bank : 국민, Serial Number : 222222

Enter the number of the ATM you want to use : 1
Please enter your card number : 1111111111111111
Please enter your password : 1111
1. Deposit
2. Withdrawal
3. Transfer
2

###(The main bank is charged 1000 won and the sub-bank is charged 2000 won automatically.)###
Please enter the amount to withdraw :
50,000 : -

```

Single ATM(ATM 1) 인출 금액 입력 창 출력

\* ----- ATM을 실행하였습니다. -----\*

1. Primary Bank : 국민, Serial Number : 111111  
2. Primary Bank : 국민, Serial Number : 222222

Enter the number of the ATM you want to use : 2

1. 한국어

2. English

1

카드 번호를 입력해주세요 : 1111111111111111

비밀번호를 입력해주세요 : 1111

1. 입금

2. 출금

3. 종료

2

###(주은행은 1000원 서브은행은 2000원의 수수료가 자동으로 부과됩니다.)###  
출금할 금액을 입력해주세요 :

50,000 : -

Multi ATM(ATM 2) 인출 금액 입력 창 출력

## REQ 5.2

- ✓ An ATM shall display an appropriate error message if there is insufficient fund in the account or insufficient cash in the ATM.

[Account 2] Name : 철수, Account Number : 222222222222 Balance : 49000  
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 50000  
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 50000  
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 50000  
[ATM 1] Remaining Cash : 290000 (50000원X5장, 10000원X1장, 5000원X5장, 1000원X5장)  
[ATM 2] Remaining Cash : 132000 (50000원X2장, 10000원X2장, 5000원X2장, 1000원X2장)

\*----- SnapShot Finish -----\*

1. Deposit  
2. Withdrawal  
3. Transfer  
2

###(The main bank is charged 1000 won and the sub-bank is charged 2000 won automatically.)###  
Please enter the amount to withdraw :

50,000 : 1

10,000 : 0

5,000 : 0

1,000 : 0

You cannot withdraw more than the amount you have in your account.

\*----- The ATM will be terminated. -----\*

- Single ATM(ATM 1) 계좌에 금액이 부족할 경우 오류메시지 출력

철수의 2번 계좌 (49,000원)에서 50,000원 출력 시 잔액 부족

```
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 49000  
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 50000  
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 50000  
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 50000  
[ATM 1] Remaining Cash : 290000 (50000원X5장, 10000원X1장, 5000원X5장, 1000원X5장)  
[ATM 2] Remaining Cash : 132000 (50000원X2장, 10000원X2장, 5000원X2장, 1000원X2장)  
*----- SnapShot Finish -----*
```

```
1. 입금  
2. 출금  
3. 송금  
2
```

###(주은행은 1000원 서브은행은 2000원의 수수료가 자동으로 부과됩니다.)###  
출금할 금액을 입력해주세요 :

50,000 : 1  
10,000 : 0  
5,000 : 0  
1,000 : 0

계좌에 보유한 금액 이상의 금액은 출금이 불가능합니다.

```
*----- ATM01 종료됩니다. -----*
```

- Multi ATM(ATM 2) 계좌에 금액이 부족할 경우 오류메시지 출력

철수의 2번 계좌 (49,000원)에서 50,000원 출력 시 잔액 부족

```
[ATM 1] Remaining Cash : 290000 (50000원X5장, 10000원X1장, 5000원X5장, 1000원X5장)  
[ATM 2] Remaining Cash : 330000 (50000원X5장, 10000원X5장, 5000원X5장, 1000원X5장)  
*----- SnapShot Finish -----*
```

```
1. Deposit  
2. Withdrawal  
3. Transfer  
2
```

###(The main bank is charged 1000 won and the sub-bank is charged 2000 won automatically.)###  
Please enter the amount to withdraw :

50,000 : 6  
10,000 : 0  
5,000 : 0  
1,000 : 0

It is not possible to withdraw more than the amount currently held by the ATM.

- Single ATM(ATM 1) ATM에 금액이 부족할 경우 오류메시지 출력

현재 29만원 ATM에 존재. 30만원 출금 불가

```
[ATM 1] Remaining Cash : 290000 (50000원X5장, 10000원X1장, 5000원X5장, 1000원X5장)  
[ATM 2] Remaining Cash : 132000 (50000원X2장, 10000원X2장, 5000원X2장, 1000원X2장)  
*----- SnapShot Finish -----*
```

```
1. 입금  
2. 출금  
3. 송금  
2
```

###(주은행은 1000원 서브은행은 2000원의 수수료가 자동으로 부과됩니다.)###  
출금할 금액을 입력해주세요 :

50,000 : 3  
10,000 : 0  
5,000 : 0  
1,000 : 0

ATM기가 현재 보유한 금액 이상의 출금이 불가능합니다.

```
*----- ATM01 종료됩니다. -----*
```

- Multi ATM(ATM 2) ATM에 금액이 부족할 경우 오류메시지 출력

현재 132,000원 ATM에 존재. 15만원 출금 불가

### REQ 5.3

- ✓ Once the withdrawal is successful, the transaction must be reflected to the bank account as well (i.e., the same amount of fund must be deducted from the corresponding bank account).

```
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 48000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 50000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 50000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 50000
[ATM 1] Remaining Cash : 290000 (5000원X5장, 10000원X1장, 5000원X5장, 1000원X5장)
[ATM 2] Remaining Cash : 132000 (5000원X2장, 10000원X2장, 5000원X2장, 1000원X2장)
*----- SnapShot Finish -----*

1. Deposit
2. Withdrawal
3. Transfer
2

###(The main bank is charged 1000 won and the sub-bank is charged 2000 won automatically.)###
Please enter the amount to withdraw :
50,000 : 0
10,000 : 1
5,000 : 0
1,000 : 0
*----- Withdrawal is completed. -----*
*----- SnapShot -----*

[Account 1] Name : 철수, Account Number : 111111111111 Balance : 909000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 37000
- Single ATM(ATM1) 출금 내역 계좌 잔고에 반영
2번 계좌(48,000원)에서 10,000원 출금 시, 1,000원 수수료 포함 11,000원 차감.
2번 계좌 잔액 48,000 – 11,000 = 37,000원.

*----- SnapShot -----*

[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2229000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 10000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 50000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 50000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 50000
[ATM 1] Remaining Cash : 925000 (5000원X15장, 10000원X9장, 5000원X14장, 1000원X15장)
[ATM 2] Remaining Cash : 785000 (5000원X12장, 10000원X11장, 5000원X12장, 1000원X15장)
*----- SnapShot Finish -----*

1. 입금
2. 출금
3. 송금
2

###(주은행은 1000원, 서브은행은 2000원의 수수료가 자동으로 부과됩니다.)###
출금할 금액을 입력해주세요 :
50,000 : 0
10,000 : 0
5,000 : 1
1,000 : 0
*----- 출금이 완료되었습니다. -----*
*----- SnapShot -----*

[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2229000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 4000
- Multi ATM(ATM2) 출금 내역 계좌 잔고에 반영
```

2번 계좌(10,000원)에서 5,000원 출금 시, 1,000원 수수료 포함 6,000원 차감.  
2번 계좌 잔액 10,000 – 6,000 = 4,000원.

#### REQ 5.4

- ✓ Some withdrawal fee may be charged (See REQ in System Setup).

- 출금 수수료: 주거래은행시 1,000원, 타행 카드일시 2,000원

```
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 15000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 50000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 50000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 50000
[ATM 1] Remaining Cash : 930000 (50000원X15장, 10000원X9장, 5000원X15장, 1000원X15장)
[ATM 2] Remaining Cash : 122000 (50000원X2장, 10000원X1장, 5000원X2장, 1000원X2장)
*----- SnapShot Finish -----*

1. Deposit
2. Withdrawal
3. Transfer
2

###(The main bank is charged 1000 won and the sub-bank is charged 2000 won automatically.)###
Please enter the amount to withdraw :
50,000 : 0
10,000 : 0
5,000 : 1
1,000 : 0
*----- Withdrawal is completed. -----*
*----- SnapShot -----*

[Account 1] Name : 철수, Account Number : 111111111111 Balance : 1569000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 9000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 50000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 50000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 50000
[ATM 1] Remaining Cash : 925000 (50000원X15장, 10000원X9장, 5000원X14장, 1000원X15장)
[ATM 2] Remaining Cash : 122000 (50000원X2장, 10000원X1장, 5000원X2장, 1000원X2장)
*----- SnapShot Finish -----*
```

- Single ATM(ATM 1)은 반드시 주거래 은행이므로 1,000원 수수료 발생

2번 계좌(15,000원)에서 5,000원 출금 시, 1,000원 수수료 포함 6,000원 차감.

2번 계좌 잔액 15,000 – 6,000 = 9,000원.

```
*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2229000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 10000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 50000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 50000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 50000
[ATM 1] Remaining Cash : 925000 (50000원X15장, 10000원X9장, 5000원X14장, 1000원X15장)
[ATM 2] Remaining Cash : 785000 (50000원X12장, 10000원X11장, 5000원X12장, 1000원X15장)
*----- SnapShot Finish -----*
```

1. 입금  
2. 출금  
3. 송금  
2

###(주은행은 1000원 서브은행은 2000원의 수수료가 자동으로 부과됩니다.)###  
출금할 금액을 입력해주세요 :

50,000 : 0  
10,000 : 0  
5,000 : 1  
1,000 : 0

\*----- 출금이 완료되었습니다. -----\*

```
*----- SnapShot -----*
```

```
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 2229000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 4000
```

- Multi ATM(ATM 2)은 주거래 은행인 2번에서 출금 시 1,000원 수수료 발생

2번 계좌(10,000원)에서 5,000원 출금 시, 1,000원 수수료 포함 6,000원 차감.

2번 계좌 잔액 10,000 – 6,000 = 4,000원.

```
*----- SnapShot -----*
```

```
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 50000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 50000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 50000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 38000
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 650000 (50000원X10장, 10000원X9장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*
```

1. 입금  
2. 출금  
3. 송금  
2

###(주은행은 1000원 서브은행은 2000원의 수수료가 자동으로 부과됩니다.)###  
출금할 금액을 입력해주세요 :

50,000 : 0  
10,000 : 1  
5,000 : 0  
1,000 : 0

\*----- 출금이 완료되었습니다. -----\*

```
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 50000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 50000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 50000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 26000
```

- Multi ATM(ATM 2)은 서브 은행인 4번에서 출금 시 2,000원 수수료 발생

4번 계좌(38,000원)에서 10,000원 출금 시, 2,000원 수수료 포함 12,000원 차감.

2번 계좌 잔액 38,000 – 12,000 = 26,000원.

## REQ 5.5

- ✓ The cash withdrawal lower available cash in the ATM that can be used by other users.

```
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 640000 (50000원X10장, 10000원X8장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*
```

1. Deposit  
2. Withdrawal  
3. Transfer  
2

###(The main bank is charged 1000 won and the sub-bank is charged 2000 won automatically.)###  
Please enter the amount to withdraw :  
50,000 : 0  
10,000 : 1  
5,000 : 0  
1,000 : 0  
\*----- Withdrawal is completed. -----\*

```
[ATM 1] Remaining Cash : 650000 (50000원X10장, 10000원X9장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 640000 (50000원X10장, 10000원X8장, 5000원X10장, 1000원X10장)
```

- Single ATM(ATM 1)에서 현금 인출 시 ATM 잔고 감소 확인  
10,000원 출금 후, 사용 가능 금액 660,000 -> 650,000으로 감소  
10,000원 10장 -> 9장 감소

```
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 650000 (50000원X10장, 10000원X9장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*
```

1. 입금  
2. 출금  
3. 송금  
2

###(주은행은 1000원, 서브은행은 2000원의 수수료가 자동으로 부과됩니다.)###  
출금할 금액을 입력해주세요 :  
50,000 : 0  
10,000 : 1  
5,000 : 0  
1,000 : 0  
\*----- 출금이 완료되었습니다. -----\*

```
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 640000 (50000원X10장, 10000원X8장, 5000원X10장, 1000원X10장)
```

- Multi ATM(ATM 2)에서 현금 인출 시 ATM 잔고 감소 확인  
10,000원 출금 후, 사용 가능 금액 650,000 -> 640,000으로 감소  
10,000원 9장 -> 8장 감소

## REQ 5.6

- ✓ The maximum number of withdrawals per each session is 3.
  - If a user wants to withdraw four times, it needs to end the current session after withdrawing three times and restart another session for one more withdrawal.

```
Enter the number of the ATM you want to use : 1
Please enter your card number : 1111111111111111
Please enter your password : 1111
1. Deposit
2. Withdrawal
3. Transfer
2

###(The main bank is charged 1000 won and the sub-bank is charged 2000 won automatically.)###
Please enter the amount to withdraw :
50,000 : 0
10,000 : 0
5,000 : 0
1,000 : 1
----- Withdrawal is completed. -----*

1. Deposit
2. Withdrawal
3. Transfer
2

###(The main bank is charged 1000 won and the sub-bank is charged 2000 won automatically.)###
Please enter the amount to withdraw :
50,000 : 0
10,000 : 0
5,000 : 0
1,000 : 1
----- Withdrawal is completed. -----*

1. Deposit
2. Withdrawal
3. Transfer
2

###(The main bank is charged 1000 won and the sub-bank is charged 2000 won automatically.)###
Please enter the amount to withdraw :
50,000 : 0
10,000 : 0
5,000 : 0
1,000 : 1
----- Withdrawal is completed. -----*

1. Deposit
2. Withdrawal
3. Transfer
2

No more withdrawals are available from this session.
```

- Single ATM(ATM 1) 한 세션 당 최대 출금 횟수는 3회, 3회 초과시 오류메시지 출력

```
카드 번호를 입력해주세요 : 1111111111111111
비밀번호를 입력해주세요 : 1111
1. 입금
2. 출금
3. 송금
2

####(주은행은 1000원 서보은행은 2000원의 수수료가 자동으로 부과됩니다.)####
출금할 금액을 입력해주세요 :
50,000 : 0
10,000 : 0
5,000 : 0
1,000 : 1
----- 출금이 완료되었습니다. -----*

1. 입금
2. 출금
3. 송금
2

####(주은행은 1000원 서보은행은 2000원의 수수료가 자동으로 부과됩니다.)####
출금할 금액을 입력해주세요 :
50,000 : 0
10,000 : 0
5,000 : 0
1,000 : 1
----- 출금이 완료되었습니다. -----*

1. 입금
2. 출금
3. 송금
2

####(주은행은 1000원 서보은행은 2000원의 수수료가 자동으로 부과됩니다.)####
출금할 금액을 입력해주세요 :
50,000 : 0
10,000 : 0
5,000 : 0
1,000 : 1
----- 출금이 완료되었습니다. -----*

1. 입금
2. 출금
3. 송금
2

해당 세션에서 더 이상 출금이 불가능합니다.
```

- Multi ATM(ATM 1) 한 세션 당 최대 출금 횟수는 3회, 3회 초과시 오류메시지 출력

#### REQ 5.7

- ✓ The maximum amount of cash withdrawal per transaction is KRW 500,000.

```
1. Deposit  
2. Withdrawal  
3. Transfer  
2  
###(The main bank is charged 1000 won and the sub-bank is charged 2000 won automatically.)###  
Please enter the amount to withdraw :  
50,000 : 10  
10,000 : 0  
5,000 : 0  
1,000 : 0  
*----- Withdrawal is completed. -----*  
###(The main bank is charged 1000 won and the sub-bank is charged 2000 won automatically.)###  
Please enter the amount to withdraw :  
50,000 : 10  
10,000 : 0  
5,000 : 0  
1,000 : 1  
It is not possible to withdraw more than 500,000 won from one transaction.  
*----- The ATM will be terminated. -----*
```

- Single ATM(ATM 1) 1회 출금의 최대 한도는 500,000원
- 500,000원 출금 가능, 하지만 501,000원부터 출금 불가능

```
*----- ATM을 실행하였습니다. -----*  
1. Primary Bank : 국민, Serial Number : 111111  
2. Primary Bank : 국민, Serial Number : 222222  
  
Enter the number of the ATM you want to use : 2  
1. 한국어  
2. English  
1  
카드 번호를 입력해주세요 : 1111111111111111  
비밀번호를 입력해주세요 : 1111  
1. 일금  
2. 출금  
3. 송금  
2  
###(주은행은 1000원 서보은행은 2000원의 수수료가 자동으로 부과됩니다.)###  
출금할 금액을 입력해주세요 :  
50,000 : 10  
10,000 : 0  
5,000 : 0  
1,000 : 0  
*----- 출금이 완료되었습니다. -----*  
  
1. 일금  
2. 출금  
3. 송금  
2  
###(주은행은 1000원 서보은행은 2000원의 수수료가 자동으로 부과됩니다.)###  
출금할 금액을 입력해주세요 :  
50,000 : 10  
10,000 : 0  
5,000 : 0  
1,000 : 1  
한번의 거래에서 500,000원 이상의 금액은 출금이 불가능합니다.  
*----- ATM이 종료됩니다. -----*
```

- Multi ATM(ATM 2) 1회 출금의 최대 한도는 500,000원  
500,000원 출금 가능, 하지만 501,000원부터 출금 불가능

#### REQ 6.1

- ✓ An ATM shall ask a user to choose the transfer types either cash transfer or account fund transfer.

```
Enter the number of the ATM you want to use : 1
Please enter your card number : 1111111111111111
Please enter your password : 1111
1. Deposit
2. Withdrawal
3. Transfer
3

1. Transfer Cash
2. Transfer account balance
```

- Single ATM(ATM 1) 현금 이체인지, 계좌 이체인지 묻는 입력 창 출력

```
Enter the number of the ATM you want to use : 2
1. 한국어
2. English
1
카드 번호를 입력해주세요 : 1111111111111111
비밀번호를 입력해주세요 : 1111
1. 입금
2. 출금
3. 송금
3

1. 현금 전송
2. 계좌 잔액 전송
```

- Multi ATM(ATM 2) 현금 이체인지, 계좌 이체인지 묻는 입력 창 출력

#### REQ 6.2

- ✓ For both cash and account transfers, an ATM shall ask the destination account number where the fund is to be transferred.

```
1. Transfer Cash
2. Transfer account balance
1
Please enter the account number of the account to receive the remittance.
```

```
1. Transfer Cash  
2. Transfer account balance  
2
```

Please enter the account number of the account to receive the remittance.

- Single ATM(ATM 1) 이체 받을 계좌번호를 묻는 입력 창 출력  
현금 이체, 혹은 계좌이체 둘 다 이체 받을 계좌번호 입력 창 출력

```
1. 현금 전송  
2. 계좌 잔액 전송  
1
```

송금받을 계좌의 계좌번호를 입력해주세요.

- Multi ATM(ATM 2) 이체 받을 계좌번호를 묻는 입력 창 출력  
현금 이체, 혹은 계좌이체 둘 다 이체 받을 계좌번호 입력 창 출력

### REQ 6.3

- ✓ For cash transfer, an ATM shall ask user to insert the cash including the transaction fees, and verify if the amount of the inserted cash is correct. All inserted cash excluding the transaction fee shall be transferred.

```
1. Transfer Cash  
2. Transfer account balance  
1
```

Please enter the account number of the account to receive the remittance.  
333333333333

A fee of 5,000 won will be incurred. Do you want to proceed? [y/n]

y  
Please enter the amount to be transferred :  
50,000 : 0  
10,000 : 1  
5,000 : 0  
1,000 : 0

Please choose how to pay the fee of 5000 won.

1. 5000Won X 1  
2. 1000Won X 5  
1

\*----- I have completed the payment of the fee. -----\*

- Single ATM(ATM 1) 현금 이체는 수수료(5,000원)를 포함하여 입금 받고, 수수료를 제외한 금액은 상대 계좌로 이체

```

1. 현금 전송
2. 계좌 잔액 전송
1
송금받을 계좌의 계좌번호를 입력해주세요.
333333333333
5000원의 수수료가 발생합니다. 진행하시겠습니까? [y/n]
y
송금할 금액을 입력해주세요 :
50,000 : 0
10,000 : 1
5,000 : 0
1,000 : 0

수수료 5000원을 지불할 방식을 선택해주세요.
1. 5000원 X 1
2. 1000원 X 5
1
*----- 수수료 지불을 완료하였습니다. -----*

```

- Multi ATM(ATM 2) 현금 이체는 수수료(5,000원)를 포함하여 입금 받고, 수수료를 제외한 금액은 상대 계좌로 이체

#### REQ 6.4

- ✓ For account transfer, an ATM shall ask the source account number, and the amount of fund to be transferred.

```

1. Transfer Cash
2. Transfer account balance
2
Please enter the account number of the account to receive the remittance.
333333333333
Please enter the account number of the account you want to transfer.
222222222222
Please enter the amount to be transferred.
10000
You will be charged a fee of 3,000 won.
Transfer? [y/n]
y

```

- Single ATM(ATM 1) 이체할 계좌 번호와 이체 금액을 적는 입력 창 출력

```

1. 현금 전송
2. 계좌 잔액 전송
2
송금받을 계좌의 계좌번호를 입력해주세요.
222222222222
송금할 계좌의 계좌번호를 입력해주세요.
111111111111
송금할 금액을 입력해주세요.
50000
3000원의 수수료가 부과됩니다.
송금하시겠습니까? [y/n]
y

```

- Multi ATM(ATM 2) 이체할 계좌 번호와 이체 금액을 적는 입력 창 출력

## REQ 6.5

- ✓ Some transfer fee may be charged (See REQ in System Setup).
- 계좌 이체 수수료(주거래 은행 계좌간 이체: 2000원, 주거래와 타행일 경우: 3000원, 타행 간의 이체: 4000원)

```
1. Transfer Cash  
2. Transfer account balance  
2  
Please enter the account number of the account to receive the remittance.  
111111111111  
Please enter the account number of the account you want to transfer.  
222222222222  
Please enter the amount to be transferred.  
10000  
You will be charged a fee of 2,000 won.  
Transfer? [y/n]
```

- Single ATM(ATM 1)의 주거래 은행 계좌 간의 이체, 수수료 2000원 발생  
(타행 계좌 번호 입력 시 exception handling 참조)

```
1. 현금 전송  
2. 계좌 잔액 전송  
2  
송금받을 계좌의 계좌번호를 입력해주세요.  
111111111111  
송금할 계좌의 계좌번호를 입력해주세요.  
222222222222  
송금할 금액을 입력해주세요.  
10000  
2000원의 수수료가 부과되었습니다.  
송금하시겠습니까? [y/n]  
송금받을 계좌의 계좌번호를 입력해주세요.  
444444444444  
송금할 계좌의 계좌번호를 입력해주세요.  
111111111111  
송금할 금액을 입력해주세요.  
50000  
3000원의 수수료가 부과됩니다.  
송금하시겠습니까? [y/n]  
y  
송금받을 계좌의 계좌번호를 입력해주세요.  
555555555555  
송금할 계좌의 계좌번호를 입력해주세요.  
444444444444  
송금할 금액을 입력해주세요.  
10000  
4000원의 수수료가 부과됩니다.  
송금하시겠습니까? [y/n]  
y
```

- Multi ATM(ATM 2)

위에서부터 주거래 은행 계좌 간의 이체, 주거래와 타행의 이체, 타행 간의 이체

각각 2000원, 3000원, 4000원 수수료 부과.

### REQ 6.6

- ✓ The inserted cash for transfer increase available cash in ATM that can be used by other users.

```
[ATM 1] Remaining Cash : 1212000 (50000원X21장, 10000원X10장, 5000원X11장, 1000원X7장)
[ATM 2] Remaining Cash : 640000 (50000원X10장, 10000원X8장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*

1. Deposit
2. Withdrawal
3. Transfer
3

1. Transfer Cash
2. Transfer account balance
1
Please enter the account number of the account to receive the remittance.
333333333333
A fee of 5,000 won will be incurred. Do you want to proceed? [y/n]
y
Please enter the amount to be transferred :
50,000 : 1
10,000 : 0
5,000 : 0
1,000 : 0

Please choose how to pay the fee of 5000 won.
1. 5000Won X 1
2. 1000Won X 5
1
[ATM 1] Remaining Cash : 1267000 (50000원X22장, 10000원X10장, 5000원X12장, 1000원X7장)
[ATM 2] Remaining Cash : 640000 (50000원X10장, 10000원X8장, 5000원X10장, 1000원X10장)

Please choose how to pay the fee of 5000 won.
1. 5000Won X 1
2. 1000Won X 5
2
[ATM 1] Remaining Cash : 1322000 (50000원X23장, 10000원X10장, 5000원X12장, 1000원X12장)
[ATM 2] Remaining Cash : 640000 (50000원X10장, 10000원X8장, 5000원X10장, 1000원X10장)
```

- Single ATM(ATM 1)에 현금 이체로 입금된 현금은 atm의 잔고 반영(수수료 포함)  
50,000원 입금 시 수수료 5,000원 포함 입금  
(5,000원으로 지불 시) ATM 50,000원 21장 -> 22장, 5,000원 11장 -> 12장  
(1,000원 X 5으로 지불 시) ATM 50,000원 22장 -> 23장, 1,000원 7장 -> 12장

```
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 675000 (50000원X10장, 10000원X11장, 5000원X11장, 1000원X10장)
*----- SnapShot Finish -----*
```

1. 입금  
2. 출금  
3. 송금  
3

1. 현금 전송  
2. 계좌 잔액 전송

1

송금받을 계좌의 계좌번호를 입력해주세요.

111111111111

5000원의 수수료가 발생합니다. 진행하시겠습니까? [y/n]

y

송금할 금액을 입력해주세요 :

50,000 : 1  
10,000 : 0  
5,000 : 0  
1,000 : 0

수수료 5000원을 지불할 방식을 선택해주세요.

1. 5000원 X 1  
2. 1000원 X 5  
1

```
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 730000 (50000원X11장, 10000원X11장, 5000원X12장, 1000원X10장)
```

같은 방식으로 50,000원 추가 입금 시

```
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 785000 (50000원X12장, 10000원X11장, 5000원X12장, 1000원X15장)
```

- Multi ATM(ATM 2)에 현금 이체로 입금된 현금은 atm의 잔고 반영(수수료 포함)  
(5,000원으로 지불 시) ATM 50,000원 10장 -> 11장, 5,000원 11장 -> 12장  
(1,000원 X 5으로 지불 시) ATM 50,000원 11장 -> 12장, 1,000원 10장 -> 15장

## REQ 6.7

- ✓ Once the transfer is successful, the transaction must be reflected to the bank account as well (i.e., the same amount of fund must be deducted from the source bank account, and then added to the destination bank account).

```
*-----[Account 3] Name : 철수, Account Number : 333333333333 Balance : 130000  
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 26000  
[ATM 1] Remaining Cash : 1267000 (50000원X22장, 10000원X10장, 5000원X12장, 1000원X7장)  
[ATM 2] Remaining Cash : 640000 (50000원X10장, 10000원X8장, 5000원X10장, 1000원X10장)  
*----- SnapShot Finish -----*
```

```
1. Deposit  
2. Withdrawal  
3. Transfer  
3
```

```
1. Transfer Cash  
2. Transfer account balance
```

```
1
```

```
Please enter the account number of the account to receive the remittance.
```

```
333333333333
```

```
A fee of 5,000 won will be incurred. Do you want to proceed? [y/n]
```

```
y
```

```
Please enter the amount to be transferred :
```

```
50,000 : 1  
10,000 : 0  
5,000 : 0  
1,000 : 0
```

```
*-----[Account 1] Name : 철수, Account Number : 111111111111 Balance : 579000  
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 24000  
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 180000  
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 26000
```

```
- Single ATM(ATM 1) 현금 이체 성공 시, 이체 받는 계좌 잔고 반영
```

```
*----- SnapShot -----*
```

```
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 50000  
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 50000  
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 50000  
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 38000  
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 50000  
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)  
[ATM 2] Remaining Cash : 320000 (50000원X5장, 10000원X4장, 5000원X5장, 1000원X5장)
```

```
*----- SnapShot Finish -----*
```

```
Please enter the account number of the account to receive the remittance.
```

```
222222222222
```

```
Please enter the account number of the account you want to transfer.
```

```
111111111111
```

```
Please enter the amount to be transferred.
```

```
10000
```

```
You will be charged a fee of 2,000 won.
```

```
Transfer? [y/n]
```

```
y
```

```
*----- SnapShot -----*
```

```
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 38000  
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 60000
```

```
- Single ATM(ATM 1) 계좌 이체 성공 시, 1번 -> 2번 10,000원 송금 시
```

```
50,000 - 10,000 - 2,000 = 38,000원, 50,000 + 10,000 = 60,000원
```

```
1번 계좌 50,000 -> 38,000원
```

```
2번 계좌 50,000 -> 60,000원
```

```
1. Transfer Cash  
2. Transfer account balance  
2  
Please enter the account number of the account to receive the remittance.  
333333333333  
Please enter the account number of the account you want to transfer.  
222222222222  
Please enter the amount to be transferred.  
10000  
You will be charged a fee of 2,000 won.  
Transfer? [y/n]  
y  
*----- SnapShot -----*  
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 38000  
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 48000  
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 60000  
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 38000  
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 50000  
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)  
[ATM 2] Remaining Cash : 320000 (5000원X5장, 10000원X4장, 5000원X5장)  
*----- SnapShot Finish -----*
```

- Single ATM(ATM 1) 계좌 이체 성공 시, 2번 -> 3번 10,000원 송금 시  
 $60,000 - 10,000 - 2,000 = 48,000\text{원}$ ,  $50,000 + 10,000 = 60,000\text{원}$   
2번 계좌 50,000 -> 48,000원 2번계좌 50,000 -> 60,000원

```
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 380000  
y  
송금할 금액을 입력해주세요 :  
50,000 : 1  
10,000 : 0  
5,000 : 0  
1,000 : 0  
  
수수료 5000원을 지불할 방식을 선택해주세요.  
1. 5000원 X 1  
2. 1000원 X 5
```

```
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 430000  
- Multi ATM(ATM 2) 현금 이체 시 이체 받는 계좌 잔고 반영
```

```
1. 현금 전송  
2. 계좌 잔액 전송  
2  
송금받을 계좌의 계좌번호를 입력해주세요.  
111111111111  
송금할 계좌의 계좌번호를 입력해주세요.  
333333333333  
송금할 금액을 입력해주세요.  
10000  
2000원의 수수료가 부과되었습니다.  
송금하시겠습니까? [y/n]  
y
```

```

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 48000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 48000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 48000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 38000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 50000
[ATM 1] Remaining Cash : 660000 (50000원X10장, 10000원X10장, 5000원X10장, 1000원X10장)
[ATM 2] Remaining Cash : 320000 (50000원X5장, 10000원X4장, 5000원X5장)
*----- SnapShot Finish -----*

```

- 타행, 주거래 은행간 거래 시

3 -> 1번 계좌로 10,000원 계좌 이체 시 2,000원 수수료 포함 12,000원 차감.

3번 계좌 :  $60,000 - 10,000 - 2,000 = 48,000$ 원

1번 계좌 :  $38,000 + 10,000 = 48,000$ 원

```

[Account 1] Name : 철수, Account Number : 111111111111 Balance : 430000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 155000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 110000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 86000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 110000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 50000

```

송금할 금액을 입력해주세요.

1000

3000원의 수수료가 부과됩니다.

송금하시겠습니까? [y/n]

y

1. 입금

2. 출금

3. 송금

x

```

*----- SnapShot -----*
```

```

[Account 1] Name : 철수, Account Number : 111111111111 Balance : 430000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 155000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 111000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 86000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 110000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 46000

```

- 타행, 주거래 은행간 거래 시

6 -> 3번 계좌로 1,000원계좌 이체 시 3,000원 수수료 포함 4,000원 차감.

6번 계좌 :  $50,000 - 1,000 - 3,000 = 46,000$ 원

3번 계좌 :  $110,000 + 1,000 = 111,000$ 원

송금받을 계좌의 계좌번호를 입력해주세요.

444444444444

송금할 계좌의 계좌번호를 입력해주세요.

666666666666

송금할 금액을 입력해주세요.

10000

4000원의 수수료가 부과됩니다.

송금하시겠습니까? [y/n]

y

```
*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 430000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 155000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 111000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 96000
[Account 5] Name : 영희, Account Number : 555555555555 Balance : 110000
[Account 6] Name : 철수, Account Number : 666666666666 Balance : 32000
```

- 타 은행간 거래 시

6 -> 4번 계좌로 10,000원을 이체 시 4,000원 수수료 포함 14,000원 차감

6번 계좌 :  $46,000 - 10,000 - 4,000 = 32,000$ 원

4번 계좌 :  $86,000 + 10,000 = 96,000$ 원

### REQ 7.1

- ✓ When a session is started by an admin by inserting an admin card (See REQ in System Setup), an ATM displays a menu of "Transaction History" only.

```
----- Transaction History -----
1. [Deposit Cash] Card Number : 1111111111111111, Deposit amount : 500000, Balance after Deposit : 550000
2. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 150000, Balance after Withdrawal : 399000
3. [Withdrawal] Card Number : 1111111111111111, Sender Account Number : 111111111111, Reciever Account Number : 222222222222, Transfer amount : 50000, Fee : 2000
----- Transaction History Finish -----*
```

- Single ATM(ATM 1)에 관리자 카드 번호(0000) 입력 시 거래 내역 메뉴 등장

각 ATM 별 거래만 출력, 해당 거래에 대한 구별은 Transaction ID를 통해 구별

```
----- Transaction History -----
4. [Deposit Check] Card Number : 1111111111111111, Deposit amount : 100000, Balance after Deposit : 447000
5. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 50000, Balance after Withdrawal : 399000
6. [Withdrawal] Card Number : 1111111111111111, Sender Account Number : 111111111111, Reciever Account Number : 444444444444, Transfer amount : 50000, Fee : 3000
----- Transaction History Finish -----*
```

- Multi ATM(ATM 2)에 관리자 카드 번호(0000) 입력 시 거래 내역 메뉴 등장

각 ATM 별 거래만 출력, 해당 거래에 대한 구별은 Transaction ID를 통해 구별

### REQ 7.2

- ✓ When the "Transaction History" menu is selected, an ATM display the information of all transactions from all users from the beginning of the system start.
- Transaction ID, Card Number, Transaction Types, Amount, other transaction-specific information
- Each transaction may have different types of information, so they need to be appropriately displayed (e.g., a deposit transaction does not have the source account information in a transfer transaction).

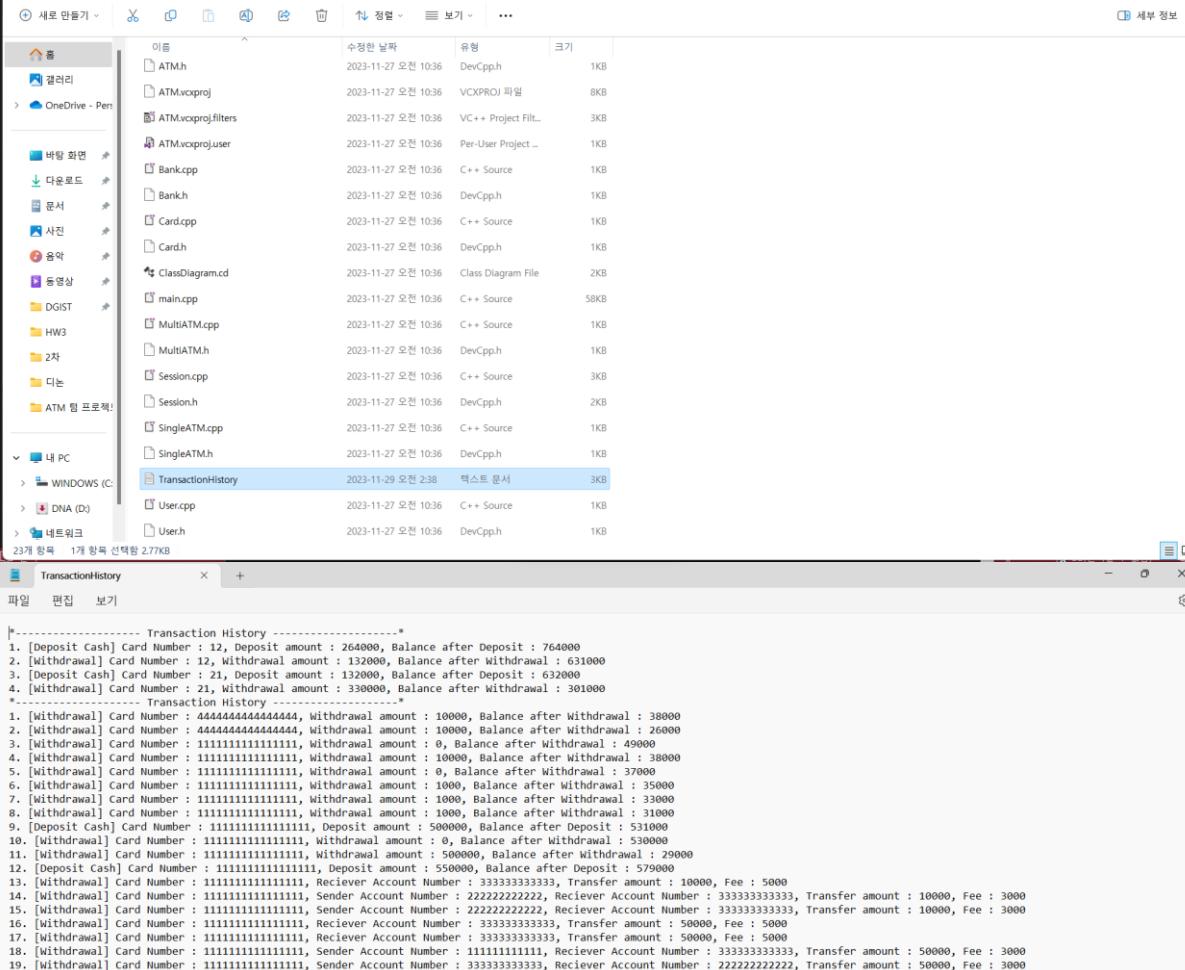
```
카드 번호를 입력해주세요 : 0000
1. Transaction History
----- Transaction History -----
1. [Withdrawal] Card Number : 1111111111111111, Reciever Account Number : 333333333333, Transfer amount : 10000, Fee : 5000
2. [Withdrawal] Card Number : 1111111111111111, Sender Account Number : 111111111111, Reciever Account Number : 222222222222, Transfer amount : 50000, Fee : 3000
3. [Withdrawal] Card Number : 1111111111111111, Sender Account Number : 111111111111, Reciever Account Number : 333333333333, Transfer amount : 50000, Fee : 3000
4. [Withdrawal] Card Number : 1111111111111111, Sender Account Number : 111111111111, Reciever Account Number : 444444444444, Transfer amount : 50000, Fee : 3000
5. [Withdrawal] Card Number : 4444444444444444, Sender Account Number : 444444444444, Reciever Account Number : 555555555555, Transfer amount : 10000, Fee : 4000
6. [Withdrawal] Card Number : 1111111111111111, Sender Account Number : 111111111111, Reciever Account Number : 222222222222, Transfer amount : 50000, Fee : 3000
7. [Withdrawal] Card Number : 1111111111111111, Sender Account Number : 111111111111, Reciever Account Number : 222222222222, Transfer amount : 50000, Fee : 3000
8. [Withdrawal] Card Number : 1111111111111111, Reciever Account Number : 111111111111, Transfer amount : 50000, Fee : 5000
9. [Withdrawal] Card Number : 1111111111111111, Reciever Account Number : 111111111111, Transfer amount : 50000, Fee : 5000
10. [Deposit Cash] Card Number : 1111111111111111, Deposit amount : 0, Balance after Deposit : 380000
----- Transaction History Finish -----*
```

- Transaction History가 뜨고 1을 선택 시, 앞에서부터

거래 번호, 거래 내역, 카드 번호, 계좌 번호, 거래 금액, 수수료 출력한다.

## REQ 7.3

- ✓ The “Transaction History” information shall be outputted to the external file (e.g., txt file)



```

----- Transaction History -----
1. [Deposit Cash] Card Number : 12, Deposit amount : 264000, Balance after Deposit : 764000
2. [Withdrawal] Card Number : 12, Withdrawal amount : 132000, Balance after Withdrawal : 631000
3. [Deposit Cash] Card Number : 21, Deposit amount : 132000, Balance after Deposit : 632000
4. [Withdrawal] Card Number : 21, Withdrawal amount : 330000, Balance after Withdrawal : 301000
----- Transaction History -----
1. [Withdrawal] Card Number : 4444444444444444, Withdrawal amount : 10000, Balance after Withdrawal : 38000
2. [Withdrawal] Card Number : 4444444444444444, Withdrawal amount : 10000, Balance after Withdrawal : 26000
3. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 0, Balance after Withdrawal : 49000
4. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 10000, Balance after Withdrawal : 38000
5. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 0, Balance after Withdrawal : 37000
6. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 1000, Balance after Withdrawal : 35000
7. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 1000, Balance after Withdrawal : 33000
8. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 1000, Balance after Withdrawal : 31000
9. [Deposit Cash] Card Number : 1111111111111111, Deposit amount : 500000, Balance after Deposit : 531000
10. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 0, Balance after Withdrawal : 530000
11. [Withdrawal] Card Number : 1111111111111111, Withdrawal amount : 500000, Balance after Withdrawal : 290000
12. [Deposit Cash] Card Number : 1111111111111111, Deposit amount : 550000, Balance after Deposit : 579000
13. [Withdrawal] Card Number : 1111111111111111, Reciever Account Number : 333333333333, Transfer amount : 10000, Fee : 5000
14. [Withdrawal] Card Number : 1111111111111111, Sender Account Number : 222222222222, Reciever Account Number : 333333333333, Transfer amount : 10000, Fee : 3000
15. [Withdrawal] Card Number : 1111111111111111, Sender Account Number : 222222222222, Reciever Account Number : 333333333333, Transfer amount : 10000, Fee : 3000
16. [Withdrawal] Card Number : 1111111111111111, Reciever Account Number : 333333333333, Transfer amount : 50000, Fee : 5000
17. [Withdrawal] Card Number : 1111111111111111, Reciever Account Number : 333333333333, Transfer amount : 50000, Fee : 5000
18. [Withdrawal] Card Number : 1111111111111111, Sender Account Number : 111111111111, Reciever Account Number : 333333333333, Transfer amount : 50000, Fee : 3000
19. [Withdrawal] Card Number : 1111111111111111, Sender Account Number : 333333333333, Reciever Account Number : 222222222222, Transfer amount : 50000, Fee : 3000

```

- ATM의 파일 내에 TransactionHistory.txt 새로 생성
- 파일 확인 시 현재까지의 모든 거래 내역 확인 가능  
(ex. 거래 번호, 카드 번호, 거래 유형, 금액, 거래 후 잔액)

## REQ 8.1

- ✓ An ATM that is configured with the bilingual support shall provide an option for a user to choose the preferred language either English or Korean.

```
*----- ATM을 실행하였습니다. -----*
1. Primary Bank : 국민, Serial Number : 111111
2. Primary Bank : 국민, Serial Number : 222222

Enter the number of the ATM you want to use : 2
1. 한국어
2. English
1
카드 번호를 입력해주세요 :
```

- Bilingual ATM(ATM 2)의 경우 한국어 혹은 영어를 선택하는 창 존재.

#### REQ 8.2

- ✓ Once a certain language is chosen, all menus must be displayed using the chosen language.

```
*----- ATM을 실행하였습니다. -----*
1. Primary Bank : 국민, Serial Number : 111111
2. Primary Bank : 국민, Serial Number : 222222

Enter the number of the ATM you want to use : 2
1. 한국어
2. English
1
카드 번호를 입력해주세요 :

*----- ATM을 실행하였습니다. -----*
1. Primary Bank : 국민, Serial Number : 111111
2. Primary Bank : 국민, Serial Number : 222222

Enter the number of the ATM you want to use : 2
1. 한국어
2. English
2
Please enter your card number :
```

Bilingual ATM(ATM 2)의 경우 한국어 선택 시 한국어 출력, 영어 선택 시 영어 출력.

#### REQ 9.1

- ✓ An ATM shall display an appropriate message for each exception scenario (both explicitly stated in this document and implicitly assumed ones), and take an appropriate action (e.g., print an appropriate error message, end a session).

##### 1. 초기 설정 오류 시

```
*----- Bank Initialization -----*
은행 생성 도중 취소하면 은행 생성이 되지 않습니다.
취소를 원하시면 C를 눌러주세요.
은행 이름을 입력해주세요 : 국민
은행 시리얼 넘버를 입력해주세요 : 1
국민은행이 생성되었습니다.

은행 이름을 입력해주세요 : 신한
은행 시리얼 넘버를 입력해주세요 : 2
신한은행이 생성되었습니다.

은행 이름을 입력해주세요 : c
2
ATM Serial Number를 입력해주세요 : 111111
메인 은행을 입력해주세요 : 국민
연결되어 있는 Bank를 입력해주세요 (끝나면 C입력) :
카카오
###해당 Bank가 존재하지 않습니다. 다시 입력해주세요###
```

- 초기 은행 설정에서 만들어두지 않은 은행 Multi ATM에 연결 시,  
"해당 Bank가 존재하지 않습니다. 다시 입력해주세요." 출력

```
*----- ATM을 실행하였습니다. -----
1. Primary Bank : 국민, Serial Number : 111111
2. Primary Bank : 국민, Serial Number : 222222

Enter the number of the ATM you want to use : 3
*----- This ATM does not exist. -----*
```

- 존재하지 않는 ATM 입력시 오류 메시지 출력

```
*----- 계좌 생성 -----
이름을 입력하세요 : 홍길동
*----- There is no user name. -----*
```

- 존재하지 않는 User 이용 시 오류 메시지 출력

## 2. ATM 실행 시

```
카드 번호를 입력해주세요 : 1111111111111111
비밀번호를 입력해주세요 : 1111
1. 입금
2. 출금
3. 송금
4

1. 입금
2. 출금
3. 송금
```

- 잘못된 실행 버튼을 누르면 다시 입력

```
*----- ATM을 실행하였습니다. -----*
1. Primary Bank : 국민, Serial Number : 111111
2. Primary Bank : 국민, Serial Number : 222222

Enter the number of the ATM you want to use : 2
1. 한국어
2. English
1
카드 번호를 입력해주세요 : 1
*----- 유효하지 않은 카드번호입니다. -----*
```

- Multi ATM(ATM 2)에 유효하지 않는 카드 입력 시,  
"유효하지 않은 카드 번호입니다."(혹은 "Invalid card number.") 출력

```
송금받을 계좌의 계좌번호를 입력해주세요.
11
*----- 해당 계좌가 존재하지 않습니다. -----*

*----- ATM이 종료됩니다. -----*

1. 현금 전송
2. 계좌 잔액 전송
2
송금받을 계좌의 계좌번호를 입력해주세요.
444444444444
송금할 계좌의 계좌번호를 입력해주세요.
1
*----- 해당 계좌가 존재하지 않습니다. -----*

*----- ATM이 종료됩니다. -----*
```

- 송금 받을 계좌 존재하지 않은 경우, 혹은 송금할 계좌가 존재하지 않는 경우  
"해당 계좌가 존재하지 않습니다." 출력 후, 종료

```
송금받을 계좌의 계좌번호를 입력해주세요.
111111111111
5000원의 수수료가 발생합니다. 진행하시겠습니까? [y/n]
n
*----- 송금을 취소합니다. ATM이 종료됩니다. -----*
```

- 수수료를 원치 않는 경우 "송금을 취소합니다. ATM이 종료됩니다." 출력 후 종료

```
1. 입금
2. 출금
3. 송금
3
1. 현금 전송
2. 계좌 잔액 전송
3
3은 잘못된 입력입니다.
*----- ATM이 종료됩니다. -----*
```

- 송금 중 잘못된 실행 번호 입력 시 오류 메시지 출력 후 ATM 종료.

```
송금받을 계좌의 계좌번호를 입력해주세요.
111111111111
5000원의 수수료가 발생합니다. 진행하시겠습니까? [y/n]
z
z는 잘못된 입력입니다.
*----- ATM이 종료됩니다. -----*
```

- 송금 수수료 y,n 가 아닌 다른 문자 들어오면 오류 메시지 반환 후 ATM 종료

```
수수료 5000원을 지불할 방식을 선택해주세요.
1. 5000원 X 1
2. 1000원 X 5
3
3는 잘못된 입력입니다.
*----- ATM이 종료됩니다. -----*
```

- 수수료 선택시도 동일. 다른 문자 들어오면 오류 메시지 반환 후 ATM 종료

```
*----- Session Transaction Finish -----*
1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 : c
*----- 프로그램이 종료됩니다. -----*
C:\Users\LG\Desktop\DGIST\2학년_2학기\액프\ATM 팀 프로젝트\ATM_영어 수정\ATM_영어 수정\x64\Debug\ATM.exe(프로세스 2791
개)이(가) 종료되었습니다.(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사
하도록 설정합니다.
0 창을 닫으려면 아무 키나 누르세요...*
```

- 초기화면에서 c 입력 시 "프로그램이 종료됩니다." 출력 후 프로그램 종료

## REQ 10.1

- ✓ When a particular character (e.g. 'x') is given as a console input during the program execution, the following information shall be displayed to the console.
  - All ATMs' information: Remaining cash
    - (e.g., ATM [SN: 111111] remaining cash: 7000, ATM [SN: 222222] remaining cash: 4000, ATM [SN: 333333] remaining cash: 2000)
  - All accounts' information: Remaining balance
    - (e.g., Account [Bank: Kakao, No: 111111111111, Owner: Jenny] balance: 7000, Account [Bank: Daegu, No: 222222222222, Owner: Tom] balance: 1000, Account [Bank: Shinhan, No: 333333333333, Owner: Jenny] balance: 2000)
- 프로그램 실행 중 x or X 입력 시 다음과 같은 스냅샷 출력

```

1. Deposit
2. Withdrawal
3. Transfer
x

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 526000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 74000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 177000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 26000
[ATM 1] Remaining Cash : 1322000 (50000원X23장, 10000원X10장, 5000원X12장, 1000원X12장)
[ATM 2] Remaining Cash : 640000 (50000원X10장, 10000원X8장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*

```

- 거래 종류 출력 창에서 SnapShot 출력 가능

```

1. ATM 생성
2. 계좌 생성
3. ATM 실행
실행 메뉴를 선택해주세요 : x

*----- SnapShot -----*
[Account 1] Name : 철수, Account Number : 111111111111 Balance : 526000
[Account 2] Name : 철수, Account Number : 222222222222 Balance : 74000
[Account 3] Name : 철수, Account Number : 333333333333 Balance : 177000
[Account 4] Name : 영희, Account Number : 444444444444 Balance : 26000
[ATM 1] Remaining Cash : 1322000 (50000원X23장, 10000원X10장, 5000원X12장, 1000원X12장)
[ATM 2] Remaining Cash : 640000 (50000원X10장, 10000원X8장, 5000원X10장, 1000원X10장)
*----- SnapShot Finish -----*

```

- 실행 창에서 SnapShot 출력 가능

### 3. The list of concepts of objected-orient programming

- ◆ Encapsulation
  - **개념:** 객체의 데이터(속성)와 그 데이터를 조작하는 메서드를 하나의 단위로 묶는 것입니다. 이를 통해 데이터의 보호 및 은닉화가 가능합니다.
  - **구현:**
- ◆ Abstraction system
  - **개념:** 복잡한 내부 구현을 숨기고, 필요한 부분만을 간략하게 표현하는 것. 이를 통해 사용자는 복잡한 내부 구현을 알 필요 없이 간단한 인터페이스를 통해 기능을 사용할 수 있음.
  - **구현:**

```

*----- ATM을 실행하였습니다. -----*
1. Primary Bank : 국민, Serial Number : 111111
2. Primary Bank : 국민, Serial Number : 222222

Enter the number of the ATM you want to use : 1
Please enter your card number : 1111111111111111
Please enter your password : 1111
1. Deposit
2. Withdrawal
3. Transfer
2

###(The main bank is charged 1000 won and the sub-bank is charged 2000 won automatically)
Please enter the amount to withdraw :
50,000 : 0
10,000 : 1
5,000 : 0
1,000 : 0
*----- Withdrawal is completed. -----*

1. Deposit
2. Withdrawal
3. Transfer
1

Please choose whether to use cash or checks.
1. Cash
2. Check
1
Please deposit the cash. If it's not a bank 국민, a fee of 1,000 won will be charged.
50,000 : 5
10,000 : 0
5,000 : 0
1,000 : 0
*----- The deposit has been completed. -----*

1. Deposit
2. Withdrawal
3. Transfer
3

1. Transfer Cash
2. Transfer account balance
1
Please enter the account number of the account to receive the remittance.
-
```

### User가 이용할 화면

- ATM 선택 창부터 시작해서 User는 내부 구현에 대한 이해 없이 요구하는 정보 입력으로 시행 가능.

## ◆ Inheritance

- 개념:** 한 클래스가 다른 클래스의 속성과 메서드를 물려받는 것. 이를 통해 코드의 재사용성과 관리 용이성이 증가.

### 구현:

Single ATM과 Multi ATM이 각각 ATM에서부터 상속 받음.

ATM에 protected로 구현된 함수들을 Single ATM과 Multi ATM에서 다시 구현할 필요 X

- ATM의 protected 변수를 public으로 상속받아오면서 그대로 protected 변수를 사용 가능.

- SingleATM, MultiATM constructor 모두 ATM 변수 활용하여 변수 전달.

```
class SingleATM : public ATM {
public:
    SingleATM();
    SingleATM(string SerialNum, Bank* primaryBank, bool bilingual);
};

class MultiATM : public ATM {
public:
    MultiATM();
    MultiATM(string SerialNum, Bank* primaryBank, bool bilingual);
    void addSubBank(Bank* BankName);
};

class ATM {
protected:
    Bank* primaryBank;
    vector<Bank*> subBank;
    int balance[4] = { 0,0,0,0 };
    string serialNum;
    string sessionTransaction;
    bool bilingual;
    int transCnt;

public:
    ATM();
    ATM(string SerialNum, Bank* primaryBank);
    Bank* getPrimaryBank();
    Bank* getSubBank(int cnt);
    long long int getBalance();
    void setPrimaryBank(Bank* bank);
    void setBilingual(bool bilingual);
    bool getBilingual();
    //balance에 moneyPage 넣어주기
    void deposit(int moneyPage[]);
    bool calWithdrawal(int money[]);
    void withdrawal(int money[]);
    string getSerialNum();
    int getSubBankNum();
    void set_balance(int a, int b, int c, int d);
    void setSerialNum(string serialNum);
    void printBalanceCnt();
    void updateTransaction(string trans);
    void printTransaction();
};

};
```

## ◆ Polymorphism

- **개념:** 같은 이름의 메서드가 다른 클래스에서 다양한 방식으로 동작할 수 있게 하는 것. 오버로딩(Overloading)과 오버라이딩(Overriding)을 통해 구현. 오버로딩은 같은 이름의 메서드가 매개변수의 타입이나 개수에 따라 다르게 동작할 수 있음을 의미하고, 오버라이딩은 상속받은 메서드의 기능을 하위 클래스에서 재정의하는 것.

### □ 구현:

Overloading을 사용하여, 매개변수를 달리 만들며, 4가지 case를 구현.

입금, 출금, 송금 (현금 이체), 송금 (계좌 이체)

모든 경우 카드 번호와 거래 금액 필요.

- 입금의 경우 입금할 계좌와 수표인지 현금인지 묻는 변수 필요
- 출금의 경우 출금할 계좌 변수 필요
- 송금 (현금)의 경우 송금할 계좌와 수수료 변수 필요
- 송금 (계좌)의 경우 출금할 계좌와 입금할 계좌, 수수료 변수 필요.

```
// 입금
void addTransaction(string cardNum, int amount, Account* target_account, string cash_or_check);
// 출금
void addTransaction(string cardNum, int amount, Account* source_account);
// 송금 (현금)
void addTransaction(string cardNum, int amount, Account* target_account, int fee);
// 송금 (계좌->계좌)
void addTransaction(string cardNum, int amount, Account* source_account, Account* target_account, int fee);
```

## ◆ Exception handling

- **개념:** 예외 처리(Exception Handling)는 프로그램 실행 중에 발생할 수 있는 예외적인 상황(예외)을 관리하는 프로그래밍 방법론. 이를 통해 프로그램의 정상적인 흐름을 방해하는 이벤트를 효과적으로 처리하고, 프로그램이 안정적으로 동작하도록 함.

### □ 구현:

수표의 금액을 입력 받는 상황 (아래 그림 참조)

- SnapShot을 위한 x 혹은 X 입력 상황 (SnapShot 함수 실행)
- 금액을 그만 받기 위해 종료하는 c 또는 C 입력 상황 (throw)
- 10만원 미만의 금액이 입금된 상황 (throw)
  - c 또는 C의 Catch의 경우, 입금 완료 메시지 출력
- 10만원 미만의 Catch의 경우, 10만원 이상의 수표 입금 요구 메시지 출력

```

while (1) {
    // Exception Handling (예외 처리)
    try {
        cin >> check;
        // (REQ1.1) SnapShot
        if (check == "x" or check == "X") {
            snapshot();
        }
        if (check == "C" or check == "c") {
            throw check;
        }
        // (REQ1.10) 수표는 100,000원 이상의 금액이어야한다.
        if ((long long int)stoll(check) < 100000) {
            throw (long long int)stoll(check);
        }
        totalCheck += (long long int)stoll(check);
        checkNum++;
    }
    catch (long long int check) {
        state("입금하신 금액은 수표에 맞지 않는 금액입니다. 100,000원 이상의 수표를 입금해주세요.");
        continue;
    }
    catch (string check) {
        state("입금이 완료되었습니다.");
        break;
    }
}

```

수수료 발생에 대한 고지, 및 선택권 부여.

- 수수료 지불을 원치 않을 경우 n 또는 N 입력, return 1을 통해 종료
- Y 또는 y 입력 시, 계속 진행, 그 외의 문자열이 입력된 경우 에러 메시지 출력

```

// Exception Handling (예외 처리)
try {
    cout << "5000원의 수수료가 발생합니다. 진행하시겠습니까? [y/n]" << endl;
    cin >> go;
    if (go == 'n' or go == 'N') {
        state("송금을 취소합니다. ATM이 종료됩니다.");
        return 1;
    }
    else if (go != 'y' and go != 'Y') {
        throw(go);
    }
}
catch (char go) {
    cout << go << "는 잘못된 입력입니다." << endl;
    return 1;
}

```

## ◆ STL

- **개념:** C++ 표준 템플릿 라이브러리(Standard Template Library, STL)는 C++의 표준 라이브러리의 일부로, 일반적인 프로그래밍 작업에 사용할 수 있는 템플릿 기반의 여러 컨테이너, 알고리즘, 반복자 등을 제공. STL은 고성능과 재사용 가능한 소프트웨어 컴포넌트를 개발할 수 있도록 함.

#### □ 구현:

STL vector를 사용

계좌에 대한 정보, ATM에 대한 정보를 저장하는 vector 생성 및 관리

```
// STL vector 사용
vector<Account*> AccountArr(100);
vector<ATM*> ATMArr(100);
```

- Vector의 Capacity를 100으로 설정, Size를 조절하면서 활용.
- Vector 상에 해당 계좌 혹은 ATM이 존재하는지 for문을 활용하여 파악.

```
for (int i = 0; i < ChosenATM->getPrimaryBank()->getAccountNum(); i++) {
    if (accountNum == ChosenATM->getPrimaryBank()->getAccountArr(i)->getAccountNum()) {
        reciever = ChosenATM->getPrimaryBank()->getAccountArr(i);
        exist = true;
        recieverPrimary = true;
    }
}
for (int i = 0; i < ChosenATM->getSubBankNum(); i++) {
    for (int j = 0; j < ChosenATM->getSubBank(i)->getAccountNum(); j++) {
        if (accountNum == ChosenATM->getSubBank(i)->getAccountArr(j)->getAccountNum()) {
            reciever = ChosenATM->getSubBank(i)->getAccountArr(j);
            exist = true;
        }
    }
}
```

Vector 활용하여 ATM 존재 여부 및 primary or sub bank 여부 파악.

## 4. Instruction

main.cpp 실행

-> 응행 초기화

-> 사용자 초기화

-> 1. ATM 생성

-> 2. 계좌 생성

-> 3. 원하는 ATM 실행

-> 1. 입금

-> 2. 출금

-> 3. 송금

-> 1. 현금 이체

-> 2. 계좌 이체

C 누르면 종료, X 누르면 현황 확인

## 5. Source code

main.cpp

```
#pragma once
#include <iostream>
#include <string>
#include <vector>
```

```

#include <fstream>
#include "ATM.h"
#include "Bank.h"
#include "User.h"
#include "Account.h"
#include "SingleATM.h"
#include "MultiATM.h"
#include "Session.h"

class ATM;
class Bank;
class User;
class Account;
class Session;

using namespace std;

static int BankCnt = 0;
static int UserCnt = 0;
static int AccountCnt = 0;
static int ATMCnt = 0;
// STL vector 사용
vector<Account*> AccountArr(100);
vector<ATM*> ATMArr(100);

int Session::transNum = 1;

// Admin 카드 번호 : 0000

void state(string statement) {
    cout << "-----" << statement << " -----" << endl <<
endl;
}

// ----- ATM, User balance 확인 -----
void snapshot() {
    state("SnapShot");
    for (int i = 0; i < AccountCnt; i++) {
        cout << "[Account " << i + 1 << "] Name : " << AccountArr[i]->getUserName() << ", "
        Account Number : " << AccountArr[i]->getAccountNum() << " Balance : " << AccountArr[i]-
        >getBalance() << endl;
    }
    for (int i = 0; i < ATMCnt; i++) {
        cout << "[ATM " << i + 1 << "] Remaining Cash : " << ATMArr[i]->getBalance() << "
";
        ATMArr[i]->printBalanceCnt();
    }
    state("SnapShot Finish");
}

// 세션 설정 (한국어)
void Session::sessionKorean(ATM* ChosenATM, Account* userAccount, string cardNum, bool

```

```

isSub) {
    tempSessionTransaction = "";
    int withdrawalNum = 0;
    while (1) {
        char operation;
        cout << "1. 입금" << endl;
        cout << "2. 출금" << endl;
        cout << "3. 송금" << endl;
        cin >> operation;
        cout << endl;

        // (REQ10.1) SnapShot
        if (operation == 'x' or operation == 'X') {
            snapshot();
        }

        // ATM 종료!
        if (operation == 'c' or operation == 'C') {
            state("ATM이 종료됩니다.");
            break;
        }

        // (REQ2.2) 비정상적인 작동이나 사용자가 종료하면 입,출,송금에서 1을 return해 ATM
        종료 (C입력 받으면 세션 종료)
        if (operation == '1') {
            int stop = sessionDepositKorean(ChosenATM, userAccount, cardNum, isSub);
            if (stop == 1) {
                state("ATM이 종료됩니다.");
                break;
            }
        }
        else if (operation == '2') {
            int stop = sessionWithdrawalKorean(ChosenATM, userAccount, cardNum, isSub,
            withdrawalNum);
            if (stop == 1) {
                state("ATM이 종료됩니다.");
                break;
            }
        }
        else if (operation == '3') {
            int stop = sessionTransferKorean(ChosenATM, userAccount, cardNum, isSub);
            if (stop == 1) {
                state("ATM이 종료됩니다.");
                break;
            }
        }
    }
}

// (REQ2.3) summary of all transactions is displayed
state("Session Transaction");
cout << tempSessionTransaction << endl;
ChosenATM->updateTransaction(tempSessionTransaction);
state("Session Transaction Finish");

}

```

```

// 입금 (한국어)
int Session::sessionDepositKorean(ATM* ChosenATM, Account* userAccount, string cardNum, bool
isSub) {
    // cash = 1 , check = 2
    char cashOrCheck;

    // (REQ4.1) ATM take either cash or check from a user
    cout << "현금을 사용할지 수표를 사용할지 골라주세요." << endl;
    cout << "1. 현금" << endl;
    cout << "2. 수표" << endl;
    cin >> cashOrCheck;

    // (REQ10.1) SnapShot
    if (cashOrCheck == 'x' or cashOrCheck == 'X') {
        snapshot();
    }

    // ATM 종료
    if (cashOrCheck != '1' and cashOrCheck != '2') {
        cout << "잘못된 입력입니다." << endl;
        return 1;
    }
    if (cashOrCheck == 'C' or cashOrCheck == 'c') {
        return 1;
    }

    if (cashOrCheck == '1') {
        cout << "현금을 입금해주세요. " << ChosenATM->getPrimaryBank()->getName() << "은행이
아니면 수수료 1000원이 부과됩니다." << endl;
        // Deposit 입력받는다.
        int moneyPage[4];
        cout << "50,000 : ";
        cin >> moneyPage[3];
        cout << "10,000 : ";
        cin >> moneyPage[2];
        cout << "5,000 : ";
        cin >> moneyPage[1];
        cout << "1,000 : ";
        cin >> moneyPage[0];
        int totalCash = 50000 * moneyPage[3] + 10000 * moneyPage[2] + 5000 * moneyPage[1] +
1000 * moneyPage[0];

        // (REQ4.2) ATM display error. Number of inserted cash exceed.
        if (moneyPage[0] + moneyPage[1] + moneyPage[2] + moneyPage[3] > 50) {
            state("너무 많은 현금이 입금되었습니다. 입금 실패!!");
            return 1;
        }
        else {
            // (REQ1.8) Non-primary bank deposit fee : 1000
            // (REQ1.8) primary bank deposit fee : 0
            // (REQ4.3) Transaction updates bank account, too.
            // (REQ4.4) Deposit fee is charged.
            userAccount->deposit(50000 * moneyPage[3] + 10000 * moneyPage[2] + 5000 *
moneyPage[1] + 1000 * (moneyPage[0] - (int)isSub));
            // (REQ4.5) Deposit increase available cash in ATM.
        }
    }
}

```

```

ChosenATM->deposit(moneyPage);
this->addTransaction(cardNum, totalCash, userAccount, "Cash");
state("입금이 완료되었습니다.");
return 0;
}
}
else {
    string check;
    int totalCheck = 0;
    int checkNum = 0;
    cout << "각 수표의 값을 입력해주세요. C가 입력되면 입금이 종료됩니다." << endl;
    while (1) {
        // Exception Handling (예외 처리)
        try {
            cin >> check;
            // (REQ10.1) SnapShot
            if (check == "x" or check == "X") {
                snapshot();
            }
            if (check == "C" or check == "c") {
                throw check;
            }
            // (REQ1.10) 수표는 100,000원 이상의 금액이어야한다.
            if ((long long int)stoll(check) < 100000) {
                throw (long long int)stoll(check);
            }
            totalCheck += (long long int)stoll(check);
            checkNum++;
        }
        catch (long long int check) {
            state("입금하신 금액은 수표에 맞지 않는 금액입니다. 100,000원 이상의 수표를
입금해주세요.");
            continue;
        }
        catch (string check) {
            state("입금이 완료되었습니다.");
            break;
        }
    }
}

// (REQ4.2) ATM display error. Number of inserted check exceed.
if (checkNum > 30) {
    state("너무 많은 수표가 입금되었습니다. 입금 실패!!");
    return 1;
}
else {
    // (REQ4.3) Transaction updates bank account, too.
    // (REQ4.4) Deposit fee is charged.
    // (REQ4.6) Deposit check do not increase available cash in ATM.
    userAccount->deposit(totalCheck - 1000 * (int)isSub);
    this->addTransaction(cardNum, totalCheck, userAccount, "Check");
    return 0;
}
}
}

```

```

// 출금 (한국어)
int Session::sessionWithdrawalKorean(ATM* ChosenATM, Account* userAccount, string cardNum,
bool isSub, int& withdrawalNum) {
    // (REQ5.6) 해당 세션에서 3번 출금한 경우 출금 불가능
    if (withdrawalNum >= 3) {
        cout << "해당 세션에서 더 이상 출금이 불가능합니다." << endl << endl;
        return 0;
    }

    int moneyPage[4] = { 0, };
    cout << "###(주은행은 1000원 서브은행은 2000원의 수수료가 자동으로 부과됩니다.)###" <<
endl;

    // (REQ5.1) ATM ask a user to enter the amount of fund to withdraw
    cout << "출금할 금액을 입력해주세요 : " << endl;
    cout << "50,000 : ";
    cin >> moneyPage[3];
    cout << "10,000 : ";
    cin >> moneyPage[2];
    cout << "5,000 : ";
    cin >> moneyPage[1];
    cout << "1,000 : ";
    cin >> moneyPage[0];

    int total_money = 50000 * moneyPage[3] + 10000 * moneyPage[2] + 5000 * moneyPage[1] +
1000 * moneyPage[0];

    // (REQ5.7) 500,000원 이상의 금액은 출금 불가능
    if (total_money > 500000) {
        cout << "한번의 거래에서 500,000원 이상의 금액은 출금이 불가능합니다." << endl <<
endl;
        return 1;
    }
    // (REQ5.2) ATM이 해당 금액을 출금할 경우의 수가 안될 경우 출금 불가능
    else if (ChosenATM->calWithdrawal(moneyPage)) {
        cout << "ATM기가 현재 보유한 금액 이상의 출금이 불가능합니다." << endl << endl;
        return 1;
    }
    // (REQ1.8) primary bank withdrawal fee : 1000
    // (REQ1.8) Non-primary bank withdrawal fee : 2000
    // (REQ5.2) 계좌에 돈이 충분하지 않으면 에러 출력
    else if (userAccount->getBalance() < total_money + 1000 * (1 + (int)isSub)) {
        cout << "계좌에 보유한 금액 이상의 금액은 출금이 불가능합니다." << endl << endl;
        return 1;
    }
    else {
        // (REQ5.3) Transaction is reflected to the bank account.
        // (REQ5.4) 출금할 때 수수료 부과 (주은행이면 1000원 sub은행이면 2000원)
        userAccount->withdrawal(total_money + 1000 * (1 + (int)isSub));
        // (REQ5.5) Withdrawal lower available cash in ATM.
        ChosenATM->withdrawal(moneyPage);
        this->addTransaction(cardNum, total_money, userAccount);
        withdrawalNum += 1;
        state("출금이 완료되었습니다.");
        return 0;
    }
}

```

```

        }
    }

// 송금 (한국어)
int Session::sessionTransferKorean(ATM* ChosenATM, Account* userAccount, string cardNum,
bool isSub) {
    // (REQ6.1) 사용자에게 transfer type을 물어본다.
    char transfer_type;
    // Exception Handling (예외 처리)
    try {
        cout << "1. 현금 전송" << endl;
        cout << "2. 계좌 잔액 전송" << endl;
        cin >> transfer_type;
        // (REQ10.1) SnapShot
        if (transfer_type == 'x' or transfer_type == 'X') {
            snapshot();
        }
        if (transfer_type != '1' and transfer_type != '2') {
            throw(transfer_type);
        }
    }
    catch (char transfer_type) {
        cout << transfer_type << "은 잘못된 입력입니다." << endl;
        return 1;
    }

    // (REQ6.2) 전송받을 사람의 account number는 항상 확인한다.
    long long int accountNum;
    Account* reciever = ChosenATM->getPrimaryBank()->getAccountArr(0);
    bool exist = false;
    bool recieverPrimary = false;
    cout << "송금받을 계좌의 계좌번호를 입력해주세요." << endl;
    cin >> accountNum;
    for (int i = 0; i < ChosenATM->getPrimaryBank()->getAccountNum(); i++) {
        if (accountNum == ChosenATM->getPrimaryBank()->getAccountArr(i)->getAccountNum()) {
            reciever = ChosenATM->getPrimaryBank()->getAccountArr(i);
            exist = true;
            recieverPrimary = true;
        }
    }
    for (int i = 0; i < ChosenATM->getSubBankNum(); i++) {
        for (int j = 0; j < ChosenATM->getSubBank(i)->getAccountNum(); j++) {
            if (accountNum == ChosenATM->getSubBank(i)->getAccountArr(j)->getAccountNum()) {
                reciever = ChosenATM->getSubBank(i)->getAccountArr(j);
                exist = true;
            }
        }
    }
}

// 틀리면 ATM 종료
if (exist == false) {
    state("해당 계좌가 존재하지 않습니다.");
    return 1;
}

```

// (REQ6.3) 현금 전송의 경우, transaction fee를 공지하고, 현금을 정확히 넣었는지 확인한다.

```
if (transfer_type == '1') {
    int moneyPage[4] = { 0, };
    char go;
    // Exception Handling (예외 처리)
    try {
        cout << "5000원의 수수료가 발생합니다. 진행하시겠습니까? [y/n]" << endl;
        cin >> go;
        if (go == 'n' or go == 'N') {
            state("송금을 취소합니다. ATM이 종료됩니다.");
            return 1;
        }
        else if (go != 'y' and go != 'Y') {
            throw(go);
        }
    }
    catch (char go) {
        cout << go << "는 잘못된 입력입니다." << endl;
        return 1;
    }

    cout << "송금할 금액을 입력해주세요 : " << endl;
    cout << "50,000 : ";
    cin >> moneyPage[3];
    cout << "10,000 : ";
    cin >> moneyPage[2];
    cout << "5,000 : ";
    cin >> moneyPage[1];
    cout << "1,000 : ";
    cin >> moneyPage[0];

    int total_money = 50000 * moneyPage[3] + 10000 * moneyPage[2] + 5000 * moneyPage[1]
+ 1000 * moneyPage[0];
    char howtopay;
    // Exception Handling (예외 처리)
    try {
        cout << endl << "수수료 5000원을 지불할 방식을 선택해주세요." << endl;
        cout << "1. 5000원 X 1" << endl;
        cout << "2. 1000원 X 5" << endl;
        cin >> howtopay;
        // (REQ10.1) SnapShot
        if (howtopay == 'x' or howtopay == 'X') {
            snapshot();
        }
        if (howtopay != '1' and howtopay != '2') {
            throw(howtopay);
        }
    }
    catch (char howtopay) {
        cout << howtopay << "는 잘못된 입력입니다." << endl;
        return 1;
    }
}
```

```

// (REQ1.8) Cash transfer fee : 5000
if (howtopay == '1' or howtopay == '2') {
    state("수수료 지불을 완료하였습니다.");
    // (REQ6.7) Transaction is reflected to the bank account (destination account)
    reciever->deposit(total_money);
    // (REQ6.6) ATM의 사용가능한 현금이 증가한다.
    if (howtopay == '1') {
        moneyPage[1] += 1;
        ChosenATM->deposit(moneyPage);
        this->addTransaction(cardNum, total_money, reciever, 5000);
        return 0;
    }
    else if (howtopay == '2') {
        moneyPage[0] += 5;
        ChosenATM->deposit(moneyPage);
        this->addTransaction(cardNum, total_money, reciever, 5000);
        return 0;
    }
}
else if (transfer_type == '2') {

    // (REQ6.4) 돈을 보낼 계좌의 계좌번호를 입력받는다.
    long long int accountNum;
    Account* sender = ChosenATM->getPrimaryBank()->getAccountArr(0);
    bool exist = false;
    bool senderPrimary = false;
    cout << "송금할 계좌의 계좌번호를 입력해주세요." << endl;
    cin >> accountNum;
    for (int i = 0; i < ChosenATM->getPrimaryBank()->getAccountNum(); i++) {
        if (accountNum == ChosenATM->getPrimaryBank()->getAccountArr(i)->getAccountNum()) {
            sender = ChosenATM->getPrimaryBank()->getAccountArr(i);
            exist = true;
            senderPrimary = true;
        }
    }
    for (int i = 0; i < ChosenATM->getSubBankNum(); i++) {
        for (int j = 0; j < ChosenATM->getSubBank(i)->getAccountNum(); j++) {
            if (accountNum == ChosenATM->getSubBank(i)->getAccountArr(j)->getAccountNum()) {
                sender = ChosenATM->getSubBank(i)->getAccountArr(j);
                exist = true;
            }
        }
    }
}

// 틀리면 ATM 종료
if (exist == false) {
    state("해당 계좌가 존재하지 않습니다.");
    return 1;
}

int sendMoney;
cout << "송금할 금액을 입력해주세요." << endl;

```

```

cin >> sendMoney;

// (REQ1.8) Account transfer fee between primary banks : 2000
// (REQ1.8) Account transfer fee between primary bank and non-primary bank : 3000
// (REQ1.8) Account transfer fee between non-primary banks : 4000
// (REQ6.5) 수수료 설정
int MoneyWithFee;
string go;
if (recieverPrimary == true and senderPrimary == true) {
    cout << "2000원의 수수료가 부과되었습니다." << endl;
    cout << "송금하시겠습니까? [y/n]" << endl;
    cin >> go;
    MoneyWithFee = sendMoney + 2000;
}
else if (recieverPrimary == true or senderPrimary == true) {
    cout << "3000원의 수수료가 부과됩니다." << endl;
    cout << "송금하시겠습니까? [y/n]" << endl;
    cin >> go;
    MoneyWithFee = sendMoney + 3000;
}
else if (recieverPrimary == false and senderPrimary == false) {
    cout << "4000원의 수수료가 부과됩니다." << endl;
    cout << "송금하시겠습니까? [y/n]" << endl;
    cin >> go;
    MoneyWithFee = sendMoney + 4000;
}

if (go == "n" or go == "N") {
    return 1;
}

// 계좌에 돈이 충분하지 않으면 에러 출력
if (sender->getBalance() < MoneyWithFee) {
    cout << "계좌에 보유한 금액 이상의 금액은 출금이 불가능합니다." << endl << endl;
    return 1;
}
else {
    // (REQ6.7) Transaction is reflected to the bank account (source, destination
account)
    reciever->deposit(sendMoney);
    sender->withdrawal(MoneyWithFee);
    this->addTransaction(cardNum, sendMoney, sender, reciever, MoneyWithFee -
sendMoney);
    return 0;
}
}

// 세션 설정 (영어)
void Session::sessionEnglish(ATM* ChosenATM, Account* userAccount, string cardNum, bool
isSub) {
    tempSessionTransaction = "";
    int withdrawalNum = 0;
    while (1) {
        char operation;

```

```

cout << "1. Deposit" << endl;
cout << "2. Withdrawal" << endl;
cout << "3. Transfer" << endl;
cin >> operation;
cout << endl;

// (REQ10.1) SnapShot
if (operation == 'x' or operation == 'X') {
    snapshot();
}

// ATM 종료!
if (operation == 'c' or operation == 'C') {
    state("The ATM will be terminated.");
    break;
}

// (REQ2.2) 비정상적인 작동이나 사용자가 종료하면 입,출,송금에서 1을 return해 ATM
종료 (C입력 받으면 세션 종료)
if (operation == '1') {
    int stop = sessionDepositEnglish(ChosenATM, userAccount, cardNum, isSub);
    if (stop == 1) {
        state("The ATM will be terminated.");
        break;
    }
}
else if (operation == '2') {
    int stop = sessionWithdrawalEnglish(ChosenATM, userAccount, cardNum, isSub,
withdrawalNum);
    if (stop == 1) {
        state("The ATM will be terminated.");
        break;
    }
}
else if (operation == '3') {
    int stop = sessionTransferEnglish(ChosenATM, userAccount, cardNum, isSub);
    if (stop == 1) {
        state("The ATM will be terminated.");
        break;
    }
}
}

// (REQ2.3) summary of all transactions is displayed
state("Session Transaction");
cout << tempSessionTransaction << endl;
ChosenATM->updateTransaction(tempSessionTransaction);
state("Session Transaction Finish");

}

// 입금 (영어)
int Session::sessionDepositEnglish(ATM* ChosenATM, Account* userAccount, string cardNum,
bool isSub) {
    // cash = 1 , check = 2
    char cashOrCheck;

```

```

// (REQ4.1) ATM take either cash or check from a user
cout << "Please choose whether to use cash or checks." << endl;
cout << "1. Cash" << endl;
cout << "2. Check" << endl;
cin >> cashOrCheck;

// (REQ10.1) SnapShot
if (cashOrCheck == 'x' or cashOrCheck == 'X') {
    snapshot();
}

// ATM 종료
if (cashOrCheck != '1' and cashOrCheck != '2') {
    cout << "Invalid input." << endl;
    return 1;
}
if (cashOrCheck == 'C' or cashOrCheck == 'c') {
    return 1;
}

if (cashOrCheck == '1') {
    cout << "Please deposit the cash. If it's not a bank" << ChosenATM-
>getPrimaryBank()->getName() << ", a fee of 1,000 won will be charged." << endl;
    // Deposit 입력받는다.
    int moneyPage[4];
    cout << "50,000 : ";
    cin >> moneyPage[3];
    cout << "10,000 : ";
    cin >> moneyPage[2];
    cout << "5,000 : ";
    cin >> moneyPage[1];
    cout << "1,000 : ";
    cin >> moneyPage[0];
    int totalCash = 50000 * moneyPage[3] + 10000 * moneyPage[2] + 5000 * moneyPage[1] +
1000 * moneyPage[0];

    // (REQ4.2) ATM display error. Number of inserted cash exceed.
    if (moneyPage[0] + moneyPage[1] + moneyPage[2] + moneyPage[3] > 50) {
        state("Too much cash has been deposited. Deposit failed!!");
        return 1;
    }
    else {
        // (REQ1.8) Non-primary bank deposit fee : 1000
        // (REQ1.8) primary bank deposit fee : 0
        // (REQ4.3) Transaction updates bank account, too.
        // (REQ4.4) Deposit fee is charged.
        userAccount->deposit(50000 * moneyPage[3] + 10000 * moneyPage[2] + 5000 *
moneyPage[1] + 1000 * (moneyPage[0] - (int)isSub));
        // (REQ4.5) Deposit increase available cash in ATM.
        ChosenATM->deposit(moneyPage);
        this->addTransaction(cardNum, totalCash, userAccount, "Cash");
        state("The deposit has been completed.");
        return 0;
    }
}

```

```

    }
    else {
        string check;
        int totalCheck = 0;
        int checkNum = 0;
        cout << "Please enter a value for each check. When C is entered, the deposit will be
terminated." << endl;
        while (1) {
            // Exception Handling (예외 처리)
            try {
                cin >> check;
                // (REQ10.1) SnapShot
                if (check == "x" or check == "X") {
                    snapshot();
                }
                if (check == "C" or check == "c") {
                    throw check;
                }
                // (REQ1.10) 수표는 100,000원 이상의 금액이어야한다.
                if ((long long int)stoll(check) < 100000) {
                    throw (long long int)stoll(check);
                }
                totalCheck += (long long int)stoll(check);
                checkNum++;
            }
            catch (long long int check) {
                state("The amount you deposited does not fit the check. Please deposit a
check of 100,000 won or more.");
                continue;
            }
            catch (string check) {
                state("The deposit has been completed.");
                break;
            }
        }
        // (REQ4.2) ATM display error. Number of inserted check exceed.
        if (checkNum > 30) {
            state("Too many checks have been deposited. Deposit failed!!");
            return 1;
        }
        else {
            // (REQ4.3) Transaction updates bank account, too.
            // (REQ4.4) Deposit fee is charged.
            // (REQ4.6) Deposit check do not increase available cash in ATM.
            userAccount->deposit(totalCheck - 1000 * (int)isSub);
            this->addTransaction(cardNum, totalCheck, userAccount, "Check");
            return 0;
        }
    }
}

// 출금 (영어)
int Session::sessionWithDrawalEnglish(ATM* ChosenATM, Account* userAccount, string cardNum,
bool isSub, int& withdrawalNum) {

```

```

// (REQ5.6) 해당 세션에서 3번 출금한 경우 출금 불가능
if (withdrawalNum >= 3) {
    cout << "No more withdrawals are available from this session." << endl << endl;
    return 0;
}

int moneyPage[4] = { 0, };
cout << "###(The main bank is charged 1000 won and the sub-bank is charged 2000 won
automatically.)###" << endl;
// (REQ5.1) ATM ask a user to enter the amount of fund to withdraw
cout << "Please enter the amount to withdraw : " << endl;
cout << "50,000 : ";
cin >> moneyPage[3];
cout << "10,000 : ";
cin >> moneyPage[2];
cout << "5,000 : ";
cin >> moneyPage[1];
cout << "1,000 : ";
cin >> moneyPage[0];

int total_money = 50000 * moneyPage[3] + 10000 * moneyPage[2] + 5000 * moneyPage[1] +
1000 * moneyPage[0];

// (REQ5.7) 500,000원 이상의 금액은 출금 불가능
if (total_money > 500000) {
    cout << "It is not possible to withdraw more than 500,000 won from one transaction."
<< endl << endl;
    return 1;
}
// (REQ5.2) ATM이 해당 금액을 출금할 경우의 수가 안될 경우 출금 불가능
else if (ChosenATM->calWithdraw(moneyPage)) {
    cout << "It is not possible to withdraw more than the amount currently held by the
ATM." << endl << endl;
    return 1;
}
// (REQ1.8) primary bank withdrawal fee : 1000
// (REQ1.8) Non-primary bank withdrawal fee : 2000
// (REQ5.2) 계좌에 돈이 충분하지 않으면 예외 출력
else if (userAccount->getBalance() < total_money + 1000 * (1 + (int)isSub)) {
    cout << "You cannot withdraw more than the amount you have in your account." << endl
<< endl;
    return 1;
}
else {
    // (REQ5.3) Transaction is reflected to the bank account.
    // (REQ5.4) 출금할 때 수수료 부과 (주은행이면 1000원 sub은행이면 2000원)
    userAccount->withdrawal(total_money + 1000 * (1 + (int)isSub));
    // (REQ5.5) Withdrawal lower available cash in ATM.
    ChosenATM->withdrawal(moneyPage);
    this->addTransaction(cardNum, total_money, userAccount);
    withdrawalNum += 1;
    state("Withdrawal is completed.");
    return 0;
}
}

```

```

// 송금 (영어)
int Session::sessionTransferEnglish(ATM* ChosenATM, Account* userAccount, string cardNum,
bool isSub) {
    // (REQ6.1) 사용자에게 transfer type을 물어본다.
    char transfer_type;
    // Exception Handling (예외 처리)
    try {
        cout << "1. Transfer Cash" << endl;
        cout << "2. Transfer account balance" << endl;
        cin >> transfer_type;
        // (REQ10.1) SnapShot
        if (transfer_type == 'x' or transfer_type == 'X') {
            snapshot();
        }
        if (transfer_type != '1' and transfer_type != '2') {
            throw(transfer_type);
        }
    }
    catch (char transfer_type) {
        cout << transfer_type << "is an invalid input." << endl;
        return 1;
    }

    // (REQ6.2) 전송받을 사람의 account number는 항상 확인한다.
    long long int accountNum;
    Account* reciever = ChosenATM->getPrimaryBank()->getAccountArr(0);
    bool exist = false;
    bool recieverPrimary = false;
    cout << "Please enter the account number of the account to receive the remittance." <<
    endl;
    cin >> accountNum;
    for (int i = 0; i < ChosenATM->getPrimaryBank()->getAccountNum(); i++) {
        if (accountNum == ChosenATM->getPrimaryBank()->getAccountArr(i)->getAccountNum()) {
            reciever = ChosenATM->getPrimaryBank()->getAccountArr(i);
            exist = true;
            recieverPrimary = true;
        }
    }
    for (int i = 0; i < ChosenATM->getSubBankNum(); i++) {
        for (int j = 0; j < ChosenATM->getSubBank(i)->getAccountNum(); j++) {
            if (accountNum == ChosenATM->getSubBank(i)->getAccountArr(j)->getAccountNum()) {
                reciever = ChosenATM->getSubBank(i)->getAccountArr(j);
                exist = true;
            }
        }
    }
    // 틀리면 ATM 종료
    if (exist == false) {
        state("This account does not exist.");
        return 1;
    }

    // (REQ6.3) 현금 전송의 경우, transaction fee를 공지하고, 현금을 정확히 넣었는지
    확인한다.

```

```

if (transfer_type == '1') {
    int moneyPage[4] = { 0, };
    char go;
    // Exception Handling (예외 처리)
    try {
        cout << "A fee of 5,000 won will be incurred. Do you want to proceed? [y/n]" <<
    endl;
        cin >> go;
        if (go == 'n' or go == 'N') {
            state("Cancel the remittance. The ATM will be terminated.");
            return 1;
        }
        else if (go != 'y' and go != 'Y') {
            throw(go);
        }
    }
    catch (char go) {
        cout << go << "is an invalid input." << endl;
        return 1;
    }

    cout << "Please enter the amount to be transferred : " << endl;
    cout << "50,000 : ";
    cin >> moneyPage[3];
    cout << "10,000 : ";
    cin >> moneyPage[2];
    cout << "5,000 : ";
    cin >> moneyPage[1];
    cout << "1,000 : ";
    cin >> moneyPage[0];

    int total_money = 50000 * moneyPage[3] + 10000 * moneyPage[2] + 5000 * moneyPage[1]
+ 1000 * moneyPage[0];
    char howtopay;
    // Exception Handling (예외 처리)
    try {
        cout << endl << "Please choose how to pay the fee of 5000 won." << endl;
        cout << "1. 5000Won X 1" << endl;
        cout << "2. 1000Won X 5" << endl;
        cin >> howtopay;
        // (REQ10.1) SnapShot
        if (howtopay == 'x' or howtopay == 'X') {
            snapshot();
        }
        if (howtopay != '1' and howtopay != '2') {
            throw(howtopay);
        }
    }
    catch (char howtopay) {
        cout << howtopay << "is an invalid input." << endl;
        return 1;
    }

    // (REQ1.8) Cash transfer fee : 5000
}

```

```

if (howtopay == '1' or howtopay == '2') {
    state("I have completed the payment of the fee.");
    // (REQ6.7) Transaction is reflected to the bank account (destination account)
    reciever->deposit(total_money);
    // (REQ6.6) ATM의 사용가능한 현금이 증가한다.
    if (howtopay == '1') {
        moneyPage[1] += 1;
        ChosenATM->deposit(moneyPage);
        this->addTransaction(cardNum, total_money, reciever, 5000);
        return 0;
    }
    else if (howtopay == '2') {
        moneyPage[0] += 5;
        ChosenATM->deposit(moneyPage);
        this->addTransaction(cardNum, total_money, reciever, 5000);
        return 0;
    }
}
else if (transfer_type == '2') {
    // (REQ6.4) 돈을 보낼 계좌의 계좌번호를 입력받는다.
    long long int accountNum;
    Account* sender = ChosenATM->getPrimaryBank()->getAccountArr(0);
    bool exist = false;
    bool senderPrimary = false;
    cout << "Please enter the account number of the account you want to transfer." <<
endl;
    cin >> accountNum;

    for (int i = 0; i < ChosenATM->getPrimaryBank()->getAccountNum(); i++) {
        if (accountNum == ChosenATM->getPrimaryBank()->getAccountArr(i)->getAccountNum()) {
            sender = ChosenATM->getPrimaryBank()->getAccountArr(i);
            exist = true;
            senderPrimary = true;
        }
    }
    for (int i = 0; i < ChosenATM->getSubBankNum(); i++) {
        for (int j = 0; j < ChosenATM->getSubBank(i)->getAccountNum(); j++) {
            if (accountNum == ChosenATM->getSubBank(i)->getAccountArr(j)->getAccountNum()) {
                sender = ChosenATM->getSubBank(i)->getAccountArr(j);
                exist = true;
            }
        }
    }
}

// 틀리면 ATM 종료
if (exist == false) {
    state("This account does not exist.");
    return 1;
}

int sendMoney;
cout << "Please enter the amount to be transferred." << endl;

```

```

cin >> sendMoney;

// (REQ1.8) Account transfer fee between primary banks : 2000
// (REQ1.8) Account transfer fee between primary bank and non-primary bank : 3000
// (REQ1.8) Account transfer fee between non-primary banks : 4000
// (REQ6.5) 수수료 설정
int MoneyWithFee;
string go;
if (recieverPrimary == true and senderPrimary == true) {
    cout << "You will be charged a fee of 2,000 won." << endl;
    cout << "Transfer? [y/n]" << endl;
    cin >> go;
    MoneyWithFee = sendMoney + 2000;
}
else if (recieverPrimary == true or senderPrimary == true) {
    cout << "You will be charged a fee of 3,000 won." << endl;
    cout << "Transfer? [y/n]" << endl;
    cin >> go;
    MoneyWithFee = sendMoney + 3000;
}
else if (recieverPrimary == false and senderPrimary == false) {
    cout << "You will be charged a fee of 4,000 won." << endl;
    cout << "Transfer? [y/n]" << endl;
    cin >> go;
    MoneyWithFee = sendMoney + 4000;
}

if (go == "n" or go == "N") {
    return 1;
}

// 계좌에 돈이 충분하지 않으면 에러 출력
if (sender->getBalance() < MoneyWithFee) {
    cout << "You cannot transfer more than the amount you have in your account." <<
endl << endl;
    return 1;
}
else {
    // (REQ6.7) Transaction is reflected to the bank account (source, destination
account)
    reciever->deposit(sendMoney);
    sender->withdrawal(MoneyWithFee);
    this->addTransaction(cardNum, sendMoney, sender, reciever, MoneyWithFee -
sendMoney);
    return 0;
}
}

int main()
{
Session session;
// Bank Initialization
// STL vector 사용

```

```

vector<Bank*> BankArr(100);
cout << "----- Bank Initialization -----" << endl;
cout << "은행 생성 도중 취소하면 은행 생성이 되지 않습니다." << endl;
cout << "취소를 원하시면 C를 눌러주세요." << endl;
while (1)
{
    string BankName;
    string SerialNum;
    cout << "은행 이름을 입력해주세요 : ";
    cin >> BankName;
    if (BankName == "C" or BankName == "c")
        break;
    cout << "은행 시리얼 넘버를 입력해주세요 : ";
    cin >> SerialNum;
    BankArr[BankCnt] = new Bank(BankName, (long long int)stoll(SerialNum));
    cout << BankArr[BankCnt]->getName() << "은행이 생성되었습니다." << endl << endl;
    BankCnt++;
}

cout << "----- Bank Initialization Finish -----" << endl
<< endl;

// User Initialization
// STL vector 사용
vector<User*> UserArr(100);
cout << "----- User Initialization -----" << endl;
cout << "User 생성 도중 취소하면 User 생성이 되지 않습니다." << endl;
cout << "취소를 원하시면 C를 눌러주세요." << endl;
while (1)
{
    string UserName;
    cout << "이름을 입력해주세요 : ";
    cin >> UserName;
    if (UserName == "C" or UserName == "c")
        break;
    UserArr[UserCnt] = new User(UserName);
    cout << UserArr[UserCnt]->getUserName() << " User가 생성되었습니다." << endl <<
endl;
    UserCnt++;
}
cout << "----- User Initialization Finish -----" << endl
<< endl;

// User Selection Page
while (1)
{
selection:
    char SelectMenu;
    cout << "1. ATM 생성" << endl;
    cout << "2. 계좌 생성" << endl;
    cout << "3. ATM 실행" << endl;
    cout << "실행 메뉴를 선택해주세요 : ";
    cin >> SelectMenu;
}

```

```

cout << endl;

// (REQ10.1) SnapShot
if (SelectMenu == 'x' or SelectMenu == 'X') {
    snapshot();
}

// 종료 코드
if (SelectMenu == 'C' or SelectMenu == 'c') {
    state("프로그램이 종료됩니다.");
    break;
}

// (REQ1.11) All ATM is created by user input
// ----- ATM 생성 코드 -----
if (SelectMenu == '1')
{
    cout << "----- ATM 생성 -----" << endl;
    // (REQ1.2) single ATM인지 multi ATM인지 입력받아서 생성
    char SingleORMulti;
    cout << "1. SingleATM" << endl;
    cout << "2. MultiATM" << endl;
    cin >> SingleORMulti;

    // (REQ10.1) SnapShot
    if (SingleORMulti == 'x' or SingleORMulti == 'X') {
        snapshot();
    }

    // 종료 코드
    if (SingleORMulti == 'C' or SingleORMulti == 'c') {
        state("프로그램이 종료됩니다.");
        break;
    }

    char Bilingual;
    cout << "1. Unilingual" << endl;
    cout << "2. Bilingual" << endl;
    cin >> Bilingual;
    bool bil;
    if (Bilingual == '2') {
        bil = true;
    }
    else if (Bilingual=='1') {
        bil = false;
    }

    // (REQ10.1) SnapShot
    if (Bilingual == 'x' or Bilingual == 'X') {
        snapshot();
    }

    // 종료 코드
    if (Bilingual == 'C' or Bilingual == 'c') {
        state("프로그램이 종료됩니다.");
        break;
    }
}

```

```

string SerialNum;
cout << "ATM Serial Number를 입력해주세요 : ";
cin >> SerialNum;

// (REQ10.1) SnapShot
if (SerialNum == "x" or SerialNum == "X") {
    snapshot();
}
// 종료 코드
if (SerialNum == "C" or SerialNum == "c") {
    state("프로그램이 종료됩니다.");
    break;
}

// (REQ1.1) ATM이 6자리가 아니면 생성불가
if (strlen(SerialNum.c_str()) != 6) {
    state("6자리 외의 Serial Number는 입력이 불가능합니다.");
    goto selection;
}
// (REQ1.1) ATM이 unique한 Serial Num을 가져야함
for (int i = 0; i < ATMCnt; i++) {
    if (ATMArr[i]->getSerialNum() == (SerialNum)) {
        state("이미 존재하는 Serial Number입니다.");
        goto selection;
    }
}

string primaryBank;
cout << "메인 은행을 입력해주세요 : ";
cin >> primaryBank;

// (REQ10.1) SnapShot
if (primaryBank == "x" or primaryBank == "X") {
    snapshot();
}
// 종료 코드
if (primaryBank == "C" or primaryBank == "c") {
    state("프로그램이 종료됩니다.");
    break;
}

// (REQ1.2) single ATM인지 multi ATM인지 입력받아서 생성
for (int i = 0; i < BankCnt; i++) {
    if (BankArr[i]->getName() == primaryBank) {
        // Single ATM 생성
        if (SingleORMulti == '1') {
            SingleATM singleATM = SingleATM((SerialNum), BankArr[i], bil);
            ATM* single = new SingleATM(singleATM);
            ATMArr[ATMCnt] = single;
            break;
        }
        // Multi ATM 생성
        else if (SingleORMulti == '2') {
            MultiATM multi = MultiATM((SerialNum), BankArr[i], bil);
        }
    }
}

```

```

        string subName;

        while (1)
        {
            cout << "연결되어 있는 Bank를 입력해주세요 (끝나면 C입력) : " <<
            endl;
            cin >> subName;
            // (REQ10.1) SnapShot
            if (subName == "x" or subName == "X") {
                snapshot();
            }
            if (subName == "C" or subName == "c") {
                state("Bank 입력을 종료합니다.");
                break;
            }
            for (int j = 0; j < BankCnt; j++) {
                if (BankArr[j]->getName() == subName) {
                    multi.addSubBank(BankArr[j]);
                    cout << subName << "은행이 ATM의 Sub은행으로
                    추가되었습니다." << endl << endl;
                    break;
                }
                else if (j == BankCnt - 1) {
                    cout << "##해당 Bank가 존재하지 않습니다. 다시
                    입력해주세요##" << endl << endl;
                    break;
                }
            }
            ATM* multi_to_ATM = new MultiATM(multi);
            ATMArr[ATMCnt] = multi_to_ATM;
            break;
        }
    }
    else if (i == BankCnt - 1) {
        state("해당 Bank가 존재하지 않습니다.");
        goto selection;
    }
}

// (REQ1.10) ATM only accepts 1000, 5000, 10000, 50000
// Deposit 입력받는다.
cout << "ATM이 소유한 deposit을 입력하시오." << endl;
int moneyPage[4] = { 0, };
cout << "50,000 : ";
cin >> moneyPage[3];
cout << "10,000 : ";
cin >> moneyPage[2];
cout << "5,000 : ";
cin >> moneyPage[1];
cout << "1,000 : ";
cin >> moneyPage[0];

// (REQ1.4) ATM 생성시, user가 사용하기 위한 초기금을 deposit

```

```

ATMArr[ATMCnt]->deposit(moneyPage);
ATMCnt++;
cout << "----- ATM을 생성하였습니다. -----" <<
endl;
}

// (REQ1.11) All Account is created by user input
// ----- 계좌 생성 코드 -----
if (SelectMenu == '2')
{
    cout << "----- 계좌 생성 -----" << endl;
    string tempAccountNum;
    long long int AccountNum;
    string tempcardNum;
    long long int cardNum;
    string UserName;
    cout << "이름을 입력하세요 : ";
    cin >> UserName;

    // (REQ1.6) 유저가 같은 은행 계좌 생성하는 것에 제한 X
    // (REQ1.7) 다양한 은행에 계좌 생성해도 문제 X
    //User loop를 돌면서 해당 이름과 동일한 사람이 없으면 오류 있으면 그 사람으로
계좌 생성
    for (int i = 0; i < UserCnt; i++) {
        if (UserArr[i]->getUserName() == UserName) {

            cout << "사용하실 계좌번호를 입력하세요 : ";
            cin >> tempAccountNum;

            // (REQ10.1) SnapShot
            if (tempAccountNum == "x" or tempAccountNum == "X") {
                snapshot();
            }
            // 종료 코드
            if (tempAccountNum == "C" or tempAccountNum == "c") {
                state("ATM이 종료됩니다.");
                break;
            }
            // 계좌번호길이는 무조건 12
            if (strlen(tempAccountNum.c_str()) != 12) {
                state("계좌번호는 길이가 12여야 합니다. ATM을 종료합니다.");
                goto selection;
            }
        }
    }

    AccountNum = (long long int)stoll(tempAccountNum);

    cout << "사용하실 카드번호를 입력하세요 : ";
    cin >> tempcardNum;

    // (REQ10.1) SnapShot
    if (tempcardNum == "x" or tempcardNum == "X") {
        snapshot();
    }
    // 종료 코드
    if (tempcardNum == "C" or tempcardNum == "c") {

```

```

        state("ATM이 종료됩니다.");
        break;
    }

cardNum = (long long int)stoll(tempcardNum);

int password;
cout << "사용하실 비밀번호를 입력해주세요 : ";
cin >> password;

string accountBank;
cout << "전용 은행을 입력하세요 : ";
cin >> accountBank;

for (int j = 0; j < BankCnt; j++) {
    if (BankArr[j]->getName() == accountBank) {
        AccountArr[AccountCnt] = new Account(BankArr[j], UserArr[i],
AccountNum, cardNum, password);

        int money;
        cout << "최초 입금액을 입력하세요 : ";
        cin >> money;
        AccountArr[AccountCnt]->deposit(money);
        BankArr[j]->addAccount(AccountArr[AccountCnt]);

        AccountCnt++;
        break;
    }
    else if ((j == BankCnt - 1) and (BankArr[j]->getName() != accountBank)) {
        state("There is no bank");
        goto selection;
    }
}
break;
}

// 여기서 cout가 계속 실행되는 error가 있다.
else if (i == UserCnt - 1) {
    state("There is no user name.");
    goto selection;
}
}

//위의 User Loop에서 찾은 user에 계좌 생성
cout << "----- 계좌를 생성하였습니다. -----" <<
endl << endl;
}

// ATM 실행 (카드 삽입)
if (SelectMenu == '3')
{
    cout << "----- ATM을 실행하였습니다. -----" <<
endl;

// 사용할 ATM 선택

```

```

int whatATM;
for (int i = 0; i < ATMCnt; i++) {
    cout << i + 1 << ". Primary Bank : " << ATMArr[i]->getPrimaryBank()-
>getName() << ", Serial Number : " << ATMArr[i]->getSerialNum() << endl;
}
cout << endl << "Enter the number of the ATM you want to use : ";
cin >> whatATM;
if (whatATM <= ATMCnt and whatATM > 0) {
    whatATM -= 1;
}
else {
    state("This ATM does not exist.");
    goto selection;
}

ATM* ChosenATM = ATMArr[whatATM];

string SelectLang;

if (ChosenATM->getBilingual() == false) {
    SelectLang = "2";
}
else {
    cout << "1. 한국어" << endl;
    cout << "2. English" << endl;

    cin >> SelectLang;

    // (REQ10.1) SnapShot
    if (SelectLang == "x" or SelectLang == "X") {
        snapshot();
    }

    // ATM 종료
    if (SelectLang == "c" or SelectLang == "C") {
        state("ATM이 종료됩니다.");
        goto selection;
    }
}

// 한글 선택
if (SelectLang == "1") {

    bool isPrimary = false;
    bool isSub = false;
    Account* userAccount = NULL;

    // (REQ2.1) user가 card를 insert하고 나서 session이 시작된다.
    string cardNum;
    cout << "카드 번호를 입력해주세요 : ";
    cin >> cardNum;

    // (REQ10.1) SnapShot
    if (cardNum == "x" or cardNum == "X") {

```

```

        snapshot();
    }

    // (REQ1.9) Admin card 번호 : 0000 입력시 Transaction History 접근 가능
    // Admin 카드 사용시 Transaction 보여주기
    if (cardNum == "0000") {
        string select;
        // (REQ7.1) 선택창에 Transaction History를 보여주는 선택지만 보여줌
        cout << "1. Transaction History" << endl;
        cin >> select;
        if (select == "1") {
            state("Transaction History");
            // (REQ7.2) ATM class 내부 함수 printTransaction으로 Transaction
History 출력
            ChosenATM->printTransaction();
            state("Transaction History Finish");

            // (REQ7.3) Transaction History -> external file
            ofstream fout("TransactionHistory.txt", ios::app);
            fout << "----- Transaction History -----"
-----*Wn";
            fout << session.printTransaction();
        }
        continue;
    }

    // ATM 종료 케이스
    if (cardNum == "C" or cardNum == "c") {
        state("ATM기를 종료합니다.");
        goto selection;
    }

    // (REQ3.1)
    // 선택한 ATM에 입력한 카드가 주은행 카드인지, sub은행 카드인지, 둘다 아님
판별
    // (REQ1.2) Single ATM에서는 primary bank에 해당하는 카드만, Multi ATM에서는
primary, sub bank에 해당하는 카드만 사용 가능
    // (REQ1.5) Bank can open an Account for user (Bank has all Account array
using that bank)
        for (int i = 0; i < ChosenATM->getPrimaryBank()->getAccountNum(); i++) {
            if (ChosenATM->getPrimaryBank()->getAccountArr(i)->getCardNum() == (long
long int)stoll(cardNum)) {
                userAccount = ChosenATM->getPrimaryBank()->getAccountArr(i);
                isPrimary = true;
                break;
            }
        }

        if (isPrimary == false) {
            for (int i = 0; i < ChosenATM->getSubBankNum(); i++) {
                for (int j = 0; j < ChosenATM->getSubBank(i)->getAccountNum(); j++) {
                    if (ChosenATM->getSubBank(i)->getAccountArr(j)->getCardNum() ==
(long long int)stoll(cardNum)) {
                        userAccount = ChosenATM->getSubBank(i)->getAccountArr(j);
                        isSub = true;
                    }
                }
            }
        }
    }
}

```

```

        }
    }
}

if (isSub == false) {
    // (REQ3.2) ATM에서 primary 은행, sub 은행에 해당하는 카드가 아니면
    // 유효하지 않으므로 에러 발생!
    state("유효하지 않은 카드번호입니다.");
    goto selection;
}
}

// (REQ3.5) 3번 연속으로 비밀번호 틀리면 ATM 종료
// 비밀번호 입력받아 3번 이상 틀리면 ATM 종료!
int pwFailCnt = 3;
int gotPW;
while (pwFailCnt > 0) {
    try{
        // (REQ3.3) 비밀번호 입력받고 검증
        cout << "비밀번호를 입력해주세요 : ";
        cin >> gotPW;
        if (gotPW != userAccount->getPassword()) {
            throw(gotPW);
        }
        else if (gotPW == userAccount->getPassword()) {
            break;
        }
    }
    catch ( int gotPW) {
        --pwFailCnt;
        if (pwFailCnt == 0) {
            state("틀렸습니다. ATM 실행이 종료됩니다.");
            goto selection;
        }
        // (REQ3.4) 비밀번호를 틀렸을 때, 에러 발생하고, 남은 기회 알려준다.
        cout << "틀렸습니다. " << pwFailCnt << "번의 기회가 남았습니다. " <<
    endl;
    }
}

// 입금, 출금, 송금, 영수증 출력 선택 페이지
session.sessionKorean(ChosenATM, userAccount, cardNum, isSub);
}

// 영어 선택
else if (SelectLang == "2") {

    bool isPrimary = false;
    bool isSub = false;
    Account* userAccount = NULL;

    // (REQ2.1) user가 card를 insert하고 나서 session이 시작된다.
    string cardNum;
    cout << "Please enter your card number : ";
    cin >> cardNum;
}

```

```

// (REQ10.1) SnapShot
if (cardNum == "x" or cardNum == "X") {
    snapshot();
}

// (REQ1.9) Admin card 번호 : 0000 입력시 Transaction History 접근 가능
// Admin 카드 사용시 Transaction 보여주기
if (cardNum == "0000") {
    string select;
    // (REQ7.1) 선택창에 Transaction History를 보여주는 선택지만 보여줌
    cout << "1. Transaction History" << endl;
    cin >> select;
    if (select == "1") {
        state("Transaction History");
        // (REQ7.2) ATM class 내부 함수 printTransaction으로 Transaction
History 출력
        ChosenATM->printTransaction();
        state("Transaction History Finish");

        // (REQ7.3) Transaction History -> external file
        ofstream fout("TransactionHistory.txt", ios::app);
        fout << "----- Transaction History -----"
-----*Wn";
        fout << session.printTransaction();
    }
    continue;
}

// ATM 종료 케이스
if (cardNum == "C" or cardNum == "c") {
    state("Shut down the ATM.");
    goto selection;
}

// (REQ3.1)
// 선택한 ATM에 입력한 카드가 주은행 카드인지, sub은행 카드인지, 둘다 아님
판별
// (REQ1.2) Single ATM에서는 primary bank에 해당하는 카드만, Multi ATM에서는
primary, sub bank에 해당하는 카드만 사용 가능
// (REQ1.5) Bank can open an Account for user (Bank has all Account array
using that bank)
for (int i = 0; i < ChosenATM->getPrimaryBank()->getAccountNum(); i++) {
    if (ChosenATM->getPrimaryBank()->getAccountArr(i)->getCardNum() == (long
long int)stoll(cardNum)) {
        userAccount = ChosenATM->getPrimaryBank()->getAccountArr(i);
        isPrimary = true;
        break;
    }
}

if (isPrimary == false) {
    for (int i = 0; i < ChosenATM->getSubBankNum(); i++) {
        for (int j = 0; j < ChosenATM->getSubBank(i)->getAccountNum(); j++)
{

```

```

        if (ChosenATM->getSubBank(i)->getAccountArr(j)->getCardNum() ==
(long long int)stoll(cardNum)) {
            userAccount = ChosenATM->getSubBank(i)->getAccountArr(j);
            isSub = true;
        }
    }
    if (isSub == false) {
        // (REQ3.2) ATM에서 primary 은행, sub 은행에 해당하는 카드가 아니면
        유효하지 않으므로 에러발생!
        state("Invalid card number.");
        goto selection;
    }
}

// (REQ3.5) 3번 연속으로 비밀번호 틀리면 ATM 종료
// 비밀번호 입력받아 3번 이상 틀리면 ATM 종료!
int pwFailCnt = 3;
int gotPW;
while (pwFailCnt > 0) {
    try {
        // (REQ3.3) 비밀번호 입력받고 검증
        cout << "Please enter your password : ";
        cin >> gotPW;
        if (gotPW != userAccount->getPassword()) {
            throw(gotPW);
        }
        else if (gotPW == userAccount->getPassword()) {
            break;
        }
    }
    catch (int gotPW) {
        --pwFailCnt;
        if (pwFailCnt == 0) {
            state("Invalid, ATM execution will end.");
            goto selection;
        }
        // (REQ3.4) 비밀번호를 틀렸을 때, 에러 발생하고, 남은 기회 알려준다.
        cout << "Incorrect. You have " << pwFailCnt << " chances left." <<
endl;
    }
}

// 입금, 출금, 송금, 영수증 출력 선택 페이지
session.sessionEnglish(ChosenATM, userAccount, cardNum, isSub);
}
}
}

```

### **Account.cpp**

```

#include "Account.h"
#include <iostream>
#include <string>
#include "User.h"
#include "Bank.h"

```

```

using namespace std;

class Account;
class Bank;

Account::Account() {
}

Account::Account(Bank* bank, User* user, long long int accountNum, long long int cardNum,
int password) {
    this->bank = bank;
    this->user = user;
    this->accountNum = accountNum;
    this->cardNum = cardNum;
    this->password = password;
}

void Account::deposit(int cash) {
    balance += cash;
}

long long int Account::getBalance() {
    return balance;
}

void Account::withdrawal(int cash) {
    balance -= cash;
}

long long int Account::getCardNum() {
    return cardNum;
}

long long int Account::getAccountNum() {
    return accountNum;
}

string Account::getUserName() {
    return user->getUserName();
}

string Account::snapShot() {
    return "Hello";
    //스냅샷 출력
}

int Account::getPassword() {
    return password;
}

```

**ATM.cpp**

```

#pragma once
#include "ATM.h"
#include <iostream>
#include <string>

```

```

#include <algorithm>
#include "Account.h"

using namespace std;

void states(string statement) {
    cout << "-----*-----" << statement << "-----*-----" << endl;
}

ATM::ATM() {
    set_balance(0, 0, 0, 0);
    transCnt += 1;
    sessionTransaction = "";
}

ATM::ATM(string SerialNum, Bank* primaryBank) {
    setSerialNum(SerialNum);
    setPrimaryBank(primaryBank);
    set_balance(0, 0, 0, 0);
    transCnt += 1;
    sessionTransaction = "";
}

Bank* ATM::getPrimaryBank() {
    return primaryBank;
}

Bank* ATM::getSubBank(int cnt) {
    return subBank[cnt];
}

long long int ATM::getBalance() {
    return 1000 * balance[0] + 5000 * balance[1] + 10000 * balance[2] + 50000 *
balance[3];
}

void ATM::setPrimaryBank(Bank* bank) {
    primaryBank = bank;
}

void ATM::setBilingual(bool bilingual) {
    this->bilingual = bilingual;
}

bool ATM::getBilingual() {
    return bilingual;
}

void ATM::deposit(int moneyPage[]) {
    for (int i = 0; i < 4; i++) {
        this->balance[i] += moneyPage[i];
    }
}

```

```

bool ATM::calWithdrawal( int money[ ] ) {
    for ( int i = 3; i >= 0; i-- ) {
        if ( balance[ i ] < money[ i ] ) {
            // 인출이 불가능할 때 true return
            return true;
        }
    }
    return false;
}

void ATM::withdrawal( int money[ ] ) {
    for ( int i = 3; i >= 0; i-- ) {
        balance[ i ] -= money[ i ];
    }
}

string ATM::getSerialNum() {
    return serialNum;
}

int ATM::getSubBankNum() {
    return subBank.size();
}

void ATM::set_balance( int a, int b, int c, int d ) {
    this->balance[ 0 ] = a;
    this->balance[ 1 ] = b;
    this->balance[ 2 ] = c;
    this->balance[ 3 ] = d;
}

void ATM::setSerialNum( string serialNum ) {
    this->serialNum = serialNum;
}

void ATM::printBalanceCnt() {
    cout << "(";
    if ( balance[ 3 ] != 0 ) {
        cout << "5000원X" << balance[ 3 ] << "장";
    }
    if ( balance[ 2 ] != 0 ) {
        if ( balance[ 3 ] > 0 ) {
            cout << ", ";
        }
        cout << "10000원X" << balance[ 2 ] << "장";
    }
    if ( balance[ 1 ] != 0 ) {
        if ( balance[ 3 ] + balance[ 2 ] > 0 ) {
            cout << ", ";
        }
        cout << "5000원X" << balance[ 1 ] << "장";
    }
    if ( balance[ 0 ] != 0 ) {
        if ( balance[ 3 ] + balance[ 2 ] + balance[ 1 ] > 0 ) {
            cout << ", ";
        }
    }
}

```

```

        }
        cout << "1000원X" << balance[0] << "장";
    }
    cout << ")" << endl;
}

```

```

void ATM::updateTransaction(string trans) {
    sessionTransaction += trans;
}
void ATM::printTransaction() {
    cout << sessionTransaction << endl;
}

```

### **Bank.cpp**

```

#include "Bank.h"
#include "Account.h"
#include <iostream>
#include <string>

class Account;

using namespace std;

Bank::Bank() {

}

Bank::Bank(string BankName, int SerialNum) {
    this->BankName = BankName;
    this->SerialNum = SerialNum;
}

string Bank::getName() {
    return this->BankName;
}

int Bank::getSerialNum() {
    return this->SerialNum;
}

int Bank::getAccountNum() {
    return accountArr.size();
}

void Bank::addAccount(Account* account) {
    accountArr.push_back(account);
}

Account* Bank::getAccountArr(int cnt) {
    return accountArr[cnt];
}

```

### **MultiATM.cpp**

```
#include <iostream>
```

```

#include <string>
#include "Bank.h"
#include "ATM.h"
#include "MultiATM.h"

using namespace std;

class Bank;
class ATM;

MultiATM::MultiATM() {
}

MultiATM::MultiATM(string SerialNum, Bank* primaryBank, bool bilingual) {
    setBilingual(bilingual);
    setSerialNum(SerialNum);
    setPrimaryBank(primaryBank);
}

void MultiATM::addSubBank(Bank* BankName) {
    this->subBank.push_back(BankName);
}

```

### **SingleATM.cpp**

```

#include <iostream>
#include <string>
#include "Bank.h"
#include "ATM.h"
#include "SingleATM.h"

using namespace std;

class Bank;
class ATM;

SingleATM::SingleATM(){
}

SingleATM::SingleATM(string SerialNum, Bank* primaryBank, bool bilingual){
    setBilingual(bilingual);
    setSerialNum(SerialNum);
    setPrimaryBank(primaryBank);
}

```

### **User.cpp**

```

#include "User.h"
#include <iostream>
#include <string>
#include "Account.h"

using namespace std;

class User;

User::User() {

```

```

}

User::User(string UserName) {
    this->name = UserName;
}

string User::getUserName() {
    return this->name;
}

void User::setAccountArr(Account account) {
    //this->accountArr[accountCnt] = account;
    accountCnt += 1;
}

Session.cpp

#pragma once
#include "ATM.h"
#include <iostream>
#include <string>
#include <algorithm>
#include "Session.h"

using namespace std;

Session::Session() {

}

void Session::addTransaction(string cardNum, int amount, Account* target_account, string cash_or_check) {
    // (REQ2.4) Each transaction has a unique identifier (transCnt)
    string inform = to_string(transNum) + ". [Deposit " + cash_or_check + "] Card Number : " + cardNum + ", Deposit amount : " + to_string(amount) + ", Balance after Deposit : " + to_string(target_account->getBalance());
    transHistory.push_back(inform);
    sessionTransaction += inform + "\n";
    tempSessionTransaction += inform + "\n";
    transNum += 1;
}

void Session::addTransaction(string cardNum, int amount, Account* source_account) {
    // (REQ2.4) Each transaction has a unique identifier (transCnt)
    string inform = to_string(transNum) + ". [Withdrawal] Card Number : " + cardNum + ", Withdrawal amount : " + to_string(amount) + ", Balance after Withdrawal : " + to_string(source_account->getBalance());
    transHistory.push_back(inform);
    sessionTransaction += inform + "\n";
    tempSessionTransaction += inform + "\n";
    transNum += 1;
}

// 송금 (현금)
void Session::addTransaction(string cardNum, int amount, Account* target_account, int fee) {
}

```

```

// (REQ2.4) Each transaction has a unique identifier (transCnt)
    string inform = to_string(transNum) + ". [Withdrawal] Card Number : " + cardNum + ", "
Reciever Account Number : " + to_string(target_account->getAccountNum()) + ", Transfer
amount : " + to_string(amount) + ", Fee : " + to_string(fee);
    transHistory.push_back(inform);
    sessionTransaction += inform + "\n";
    tempSessionTransaction += inform + "\n";
    transNum += 1;
}

// 송금 (계좌->계좌)
void Session::addTransaction(string cardNum, int amount, Account* source_account, Account*
target_account, int fee) {
    // (REQ2.4) Each transaction has a unique identifier (transCnt)
    string inform = to_string(transNum) + ". [Withdrawal] Card Number : " + cardNum + ", "
Sender Account Number : " + to_string(source_account->getAccountNum()) + ", Reciever Account
Number : " + to_string(target_account->getAccountNum()) + ", Transfer amount : " +
to_string(amount) + ", Fee : " + to_string(fee);
    transHistory.push_back(inform);
    sessionTransaction += inform + "\n";
    tempSessionTransaction += inform + "\n";
    transNum += 1;
}

string Session::printTransaction() {
    /*string trans = "";
    for (int i = 0; i < transNum; i++) {
        trans += (transHistory[i] + "\n");
    }*/
    return sessionTransaction;
}

```

## Account.h

```

#pragma once
#include <iostream>
#include <string>
#include "User.h"
#include "Bank.h"

class Bank;

using namespace std;

class Account{
private:
    User *user;
    Bank *bank;
    long long int accountNum;
    long long int cardNum;
    long long int balance = 0;
    int password;
    string accountHistory;

public:
    Account();
    Account(Bank* bank, User* user, long long int accountNum, long long int cardNum, int

```

```
password);
    //생성자 실행할 때, 해당 User의 accountArr에 Account 객체도 저장해야함.
    void deposit(int cash);
    long long int getBalance();
    void withdrawal(int cash);
    long long int getCardNum();
    long long int getAccountNum();
    string getUserName();
    string snapShot();
    int getPassword();
};
```

### ATM.h

```
#pragma once
#include <iostream>
#include <string>
#include <vector>
#include "Bank.h"

using namespace std;

class ATM {
protected:
    Bank* primaryBank;
    vector<Bank*> subBank;
    int balance[4] = { 0,0,0,0 };
    string serialNum;
    string sessionTransaction;
    bool bilingual;
    int transCnt;

public:
    ATM();
    ATM(string SerialNum, Bank* primaryBank);
    Bank* getPrimaryBank();
    Bank* getSubBank(int cnt);
    long long int getBalance();
    void setPrimaryBank(Bank* bank);
    void setBilingual(bool bilingual);
    bool getBilingual();
    //balance에 moneyPage 넣어주기
    void deposit(int moneyPage[]);
    bool calWithdrawal(int money[]);
    void withdrawal(int money[]);
    string getSerialNum();
    int getSubBankNum();
    void set_balance(int a, int b, int c, int d);
    void setSerialNum(string serialNum);
    void printBalanceCnt();
    void updateTransaction(string trans);
    void printTransaction();

};
```

### Bank.h

```
#pragma once
```

```

#include <iostream>
#include <string>
#include "Account.h"
#include <vector>

using namespace std;

class Account;

class Bank {
private:
    string BankName;
    int SerialNum;
    vector<Account*> accountArr;

public:
    Bank();
    Bank(string BankName, int SerialNum);
    string getName();
    int getSerialNum();
    int getAccountNum();
    void addAccount(Account* account);
    Account* getAccountArr(int cnt);
};

```

### **MultiATM.h**

```

#pragma once
#include <iostream>
#include <string>
#include "Bank.h"
#include "ATM.h"

using namespace std;

class MultiATM : public ATM {
public:
    MultiATM();
    MultiATM(string SerialNum, Bank* primaryBank, bool bilingual);
    void addSubBank(Bank* BankName);
};

```

### **SingleATM.h**

```

#pragma once
#include <iostream>
#include <string>
#include "Bank.h"
#include "ATM.h"

using namespace std;

class SingleATM : public ATM {
public:
    SingleATM();
    SingleATM(string SerialNum, Bank* primaryBank, bool bilingual);
};

```

### **User.cpp**

```

#pragma once
#include <iostream>
#include <string>

using namespace std;

class Account;

class User {
private:
    int accountCnt = 0;
    string name;
    //Account accountArr[100];

public:
    User();
    User(string UserName);
    string getUserName();
    void setAccountArr(Account account);
    //accountArr에 append 해준다.
};

Session.h
#pragma once
#include <iostream>
#include <string>
#include <vector>
#include "Bank.h"
#include "ATM.h"

using namespace std;

class Session {
private:
    static int transNum;
    string sessionTransaction;
    string tempSessionTransaction;
    vector<string> transHistory;
public:
    // 세션 설정
    Session();

    // Polymorphism
    // 입,출금, 송금 내역 Transaction 저장

    // 입금
    void addTransaction(string cardNum, int amount, Account* target_account, string cash_or_check);
    // 출금
    void addTransaction(string cardNum, int amount, Account* source_account);
    // 송금 (현금)
    void addTransaction(string cardNum, int amount, Account* target_account, int fee);
    // 송금 (계좌->계좌)
    void addTransaction(string cardNum, int amount, Account* source_account, Account* target_account, int fee);
    string printTransaction();
}

```

```
    void sessionKorean(ATM* ChosenATM, Account* userAccount, string cardNum, bool
isSub);
    int sessionDepositKorean(ATM* ChosenATM, Account* userAccount, string cardNum, bool
isSub);
    int sessionWithdrawKorean(ATM* ChosenATM, Account* userAccount, string cardNum,
bool isSub, int& withdrawalNum);
    int sessionTransferKorean(ATM* ChosenATM, Account* userAccount, string cardNum, bool
isSub);

    void sessionEnglish(ATM* ChosenATM, Account* userAccount, string cardNum, bool
isSub);
    int sessionDepositEnglish(ATM* ChosenATM, Account* userAccount, string cardNum, bool
isSub);
    int sessionWithdrawEnglish(ATM* ChosenATM, Account* userAccount, string cardNum,
bool isSub, int& withdrawalNum);
    int sessionTransferEnglish(ATM* ChosenATM, Account* userAccount, string cardNum,
bool isSub);
};


```

## 6. Member student contribution

All students equally contributed.

유선우 : 이재현 : 이창석 = 1 : 1 : 1