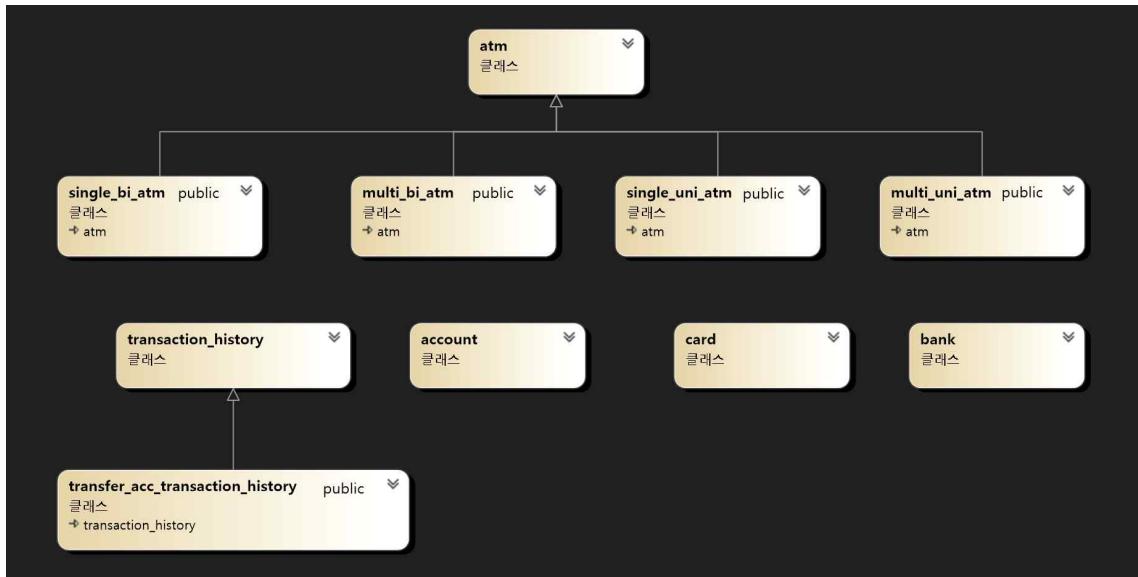


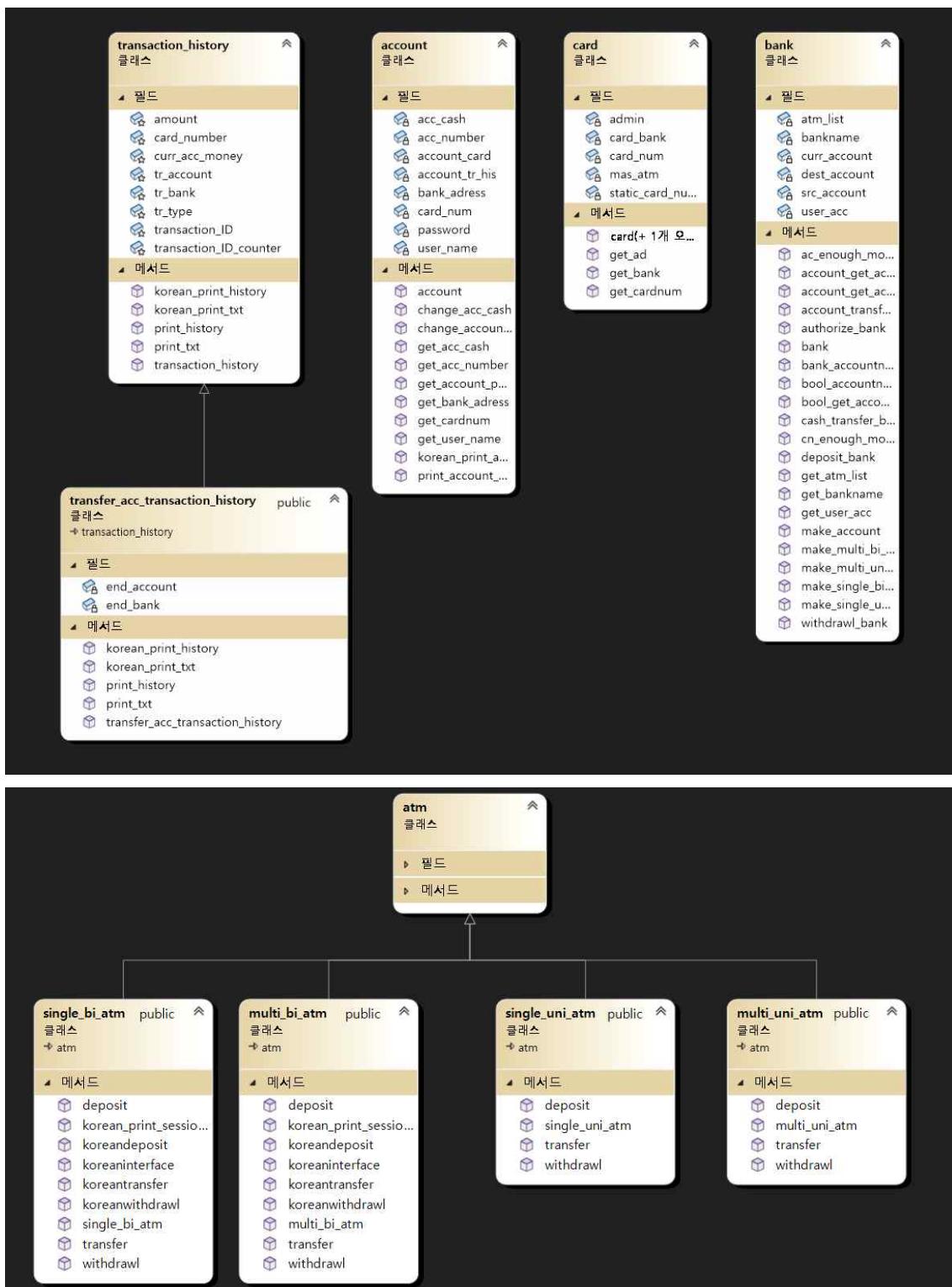
객체지향 프로그래밍 CSE201 Term Project REPORT

202011003 강승수, 202011133 윤현서, 202011197 정세훈

Classs UML







1. System Setup

(REQ 1.1) An ATM has a 6-digit serial number that can be uniquely identified among all ATMs(e.g., 315785).

1) serial number가 315785인 ATM 만들기

```
<ATM 만들기>
ATM의 SERIAL NUMBER을 입력해주세요 315785
ATM type에 해당하는 번호를 입력해주세요
0. Single Bank ATM 1. Multi-Bank ATM 1
ATM의 언어 지원 여부에 해당하는 번호를 입력해주세요
0. Unilingual 1. Bilingual 1
돈은 얼마나 들어있게 할까요?
천원을 몇 장 넣으시겠습니까? 10
오천원을 몇 장 넣으시겠습니까? 10
만원을 몇 장 넣으시겠습니까? 10
오만원을 몇 장 넣으시겠습니까? 10
admin card가 생성되었습니다. 카드의 은행은 한국은행입니다.
atm 이 생성되었습니다. atm의 serial number은 315785입니다. 현재 돈은 660000보유하고 있습니다.
```

2) 6자리가 아닐 경우

```
<ATM 만들기>
ATM의 SERIAL NUMBER을 입력해주세요 123
잘못된 ATM SERIAL NUMBER의 유형입니다. 6자리로 입력해주세요.
ATM의 SERIAL NUMBER을 입력해주세요
```

3) 이미 serial number가 존재할 경우

```
<ATM 만들기>
ATM의 SERIAL NUMBER을 입력해주세요 123
잘못된 ATM SERIAL NUMBER의 유형입니다. 6자리로 입력해주세요.
ATM의 SERIAL NUMBER을 입력해주세요 315785
이미 존재하는 ATM SERIAL NUMBER입니다. 다시 입력해주세요.
ATM의 SERIAL NUMBER을 입력해주세요
```

(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, and only a card issued by the primary bank is considered valid.

<한국은행 single atm 생성>

```
ATM의 SERIAL NUMBER을 입력해주세요 456789
ATM type에 해당하는 번호를 입력해주세요
0. Single Bank ATM 1. Multi-Bank ATM 0
ATM의 언어 지원 여부에 해당하는 번호를 입력해주세요
0. Unilingual 1. Bilingual 1
돈은 얼마나 들어있게 할까요?
천원을 몇 장 넣으시겠습니까? 10
오천원을 몇 장 넣으시겠습니까? 10
만원을 몇 장 넣으시겠습니까? 10
오만원을 몇 장 넣으시겠습니까? 10
admin card가 생성되었습니다. 카드의 은행은 한국은행입니다.
atm 이 생성되었습니다. atm의 serial number은 456789입니다. 현재 돈은 660000보유하고 있습니다.
```

<한국은행 single atm에 대구은행 카드를 넣을 경우>

<ATM 선택하기>
선택하고 싶은 atm의 번호를 기입해주세요. 이전으로 가고 싶으시면 -1을 입력해주세요.
0. atm serial number : 315785
1. atm serial number : 456789
1
어떤 언어를 고르시겠습니까? Which language do you prefer?
0. 영어 1. 한국어
1
카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요.1001
주은행 계좌가 아닙니다.

<한국은행 single atm에 한국은행 카드를 넣을 경우>

<ATM 선택하기>
선택하고 싶은 atm의 번호를 기입해주세요. 이전으로 가고 싶으시면 -1을 입력해주세요.
0. atm serial number : 315785
1. atm serial number : 456789
1
어떤 언어를 고르시겠습니까? Which language do you prefer?
0. 영어 1. 한국어
1
카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요.1002
비밀번호를 입력해주세요.
1234
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 :

- For Multi-Bank ATM, there is a primary bank that manages the ATM, but a card issued by any other banks is considered valid.

<한국은행 multi atm 생성>

<ATM 만들기>
ATM의 SERIAL NUMBER를 입력해주세요. 315785
ATM type에 해당하는 번호를 입력해주세요
0. Single Bank ATM 1. Multi-Bank ATM 1
ATM의 언어 지원 여부에 해당하는 번호를 입력해주세요
0. Unilingual 1. Bilingual 1
돈은 얼마나 들어있게 할까요?
천원을 몇 장 넣으시겠습니까? 10
오천원을 몇 장 넣으시겠습니까? 10
만원을 몇 장 넣으시겠습니까? 10
오만원을 몇 장 넣으시겠습니까? 10
admin card가 생성되었습니다. 카드의 은행은 한국은행입니다.
atm 이 생성되었습니다. atm의 serial number은 315785입니다. 현재 돈은 660000보유하고 있습니다.

<한국은행 multi atm에 대구은행 카드를 넣을 경우>

<ATM 선택하기>
선택하고 싶은 atm의 번호를 기입해주세요. 이전으로 가고 싶으시면 -1을 입력해주세요.
0. atm serial number : 315785
1. atm serial number : 456789
0
어떤 언어를 고르시겠습니까? Which language do you prefer?
0. 영어 1. 한국어
1
카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요.1001
비밀번호를 입력해주세요.
1234
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 :

<한국은행 multi atm에 한국은행 카드를 넣을 경우>

```
<ATM 선택하기>
선택하고 싶은 atm의 번호를 기입해주세요. 이전으로 가고 싶으시면 -1을 입력해주세요.
0. atm serial number : 315785
1. atm serial number : 456789
0
어떤 언어를 고르시겠습니까? Which language do you prefer?
0. 영어 1. 한국어
1
카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요. 1002
비밀번호를 입력해주세요.
1234
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 :
```

(REQ1.3) An ATM may support either unilingual or bilingual languages.

- When an ATM is configured unilingual, all information is displayed in English only.

<unilingual ATM 만들기>

```
<ATM 만들기>
ATM의 SERIAL NUMBER을 입력해주세요 123456
ATM type에 해당하는 번호를 입력해주세요
0. Single Bank ATM 1. Multi-Bank ATM 1
ATM의 언어 지원 여부에 해당하는 번호를 입력해주세요
0. Unilingual 1. Bilingual 0
돈은 얼마나 들어 있게 할까요?
천원을 몇 장 넣으시겠습니까? 10
오천원을 몇 장 넣으시겠습니까? 10
만원을 몇 장 넣으시겠습니까? 10
오만원을 몇 장 넣으시겠습니까? 10
admin card가 생성되었습니다. 카드의 은행은 한국은행입니다.
atm 이 생성되었습니다. atm의 serial number은 123456입니다. 현재 돈은 660000보유하고 있습니다.
```

<ATM 이용하기> - english only >

```
<ATM 선택하기>
선택하고 싶은 atm의 번호를 기입해주세요. 이전으로 가고 싶으시면 -1을 입력해주세요.
0. atm serial number : 315785
1. atm serial number : 456789
2. atm serial number : 123456
2
Please insert the card.(Please enter the card number). If you want to exit, enter -1. 1001
Please enter the password.
1234
What kind of work would you like to do? Please enter the number.
1. deposit 2. withdrawl 3. transfer 4. end session:
```

- 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).

<bilingual ATM 만들기>

```
<ATM 만들기>
ATM의 SERIAL NUMBER를 입력해주세요 315785
ATM type에 해당하는 번호를 입력해주세요
0. Single Bank ATM 1. Multi-Bank ATM 1
ATM의 언어 지원 여부에 해당하는 번호를 입력해주세요
0. Unilingual 1. Bilingual 1
돈은 얼마나 들어있게 할까요?
천원을 몇 장 넣으시겠습니까? 10
오천원을 몇 장 넣으시겠습니까? 10
만원을 몇 장 넣으시겠습니까? 10
오만원을 몇 장 넣으시겠습니까? 10
admin card가 생성되었습니다. 카드의 은행은 한국은행입니다.
atm 이 생성되었습니다. atm의 serial number은 315785입니다. 현재 돈은 660000보유하고 있습니다.
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
```

<언어 고르기-영어>

```
<ATM 선택하기>
선택하고 싶은 atm의 번호를 기입해주세요. 이전으로 가고 싶으시면 -1을 입력해주세요.
0. atm serial number : 315785
1. atm serial number : 456789
2. atm serial number : 123456
0
어떤 언어를 고르시겠습니까? Which language do you prefer?
0. 영어 1. 한국어
0
Please insert the card.(Please enter the card number). If you want to exit, enter -1.1001
Please enter the password.
1234
What kind of work would you like to do? Please enter the number.
1. deposit 2. withdrawl 3. transfer 4. end session: 4
```

<언어 고르기-한국어>

```
<ATM 선택하기>
선택하고 싶은 atm의 번호를 기입해주세요. 이전으로 가고 싶으시면 -1을 입력해주세요.
0. atm serial number : 315785
1. atm serial number : 456789
2. atm serial number : 123456
0
어떤 언어를 고르시겠습니까? Which language do you prefer?
0. 영어 1. 한국어
1
카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요. 1001
비밀번호를 입력해주세요.
1234
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 :
```

(REQ1.4) A Bank deposits a certain amount of cashes to an ATM to serve users.

```
<ATM 만들기>
ATM의 SERIAL NUMBER를 입력해주세요 315785
ATM type에 해당하는 번호를 입력해주세요
0. Single Bank ATM 1. Multi-Bank ATM 1
ATM의 언어 지원 여부에 해당하는 번호를 입력해주세요
0. Unilingual 1. Bilingual 1
돈을 얼마나 들어있게 할까요?
천원을 몇 장 넣으시겠습니까? 10
오천원을 몇 장 넣으시겠습니까? 10
만원을 몇 장 넣으시겠습니까? 10
오만원을 몇 장 넣으시겠습니까? 10
admin card가 생성되었습니다. 카드의 은행은 한국은행입니다.
atm 이 생성되었습니다. atm의 serial number은 315785입니다. 현재 돈은 660000보유하고 있습니다.
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
```

(REQ1.5) A Bank can open an Account for user with necessary information to perform bank services.

- (e.g.) Bank name (e.g, Kakao, Shinhan), User name, Account number (12-digit), Available funds, Transaction histories.

```
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 은행 만들기 2. 은행 선택하기 3. 정보 출력하기 : 2
<은행 선택하기>
가고자 하는 은행 번호를 입력해주세요. 이전 화면으로 돌아가고자 하면 -1을 입력해주세요.
0. 한국은행 1. 대구은행
0
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 은행 계좌 만들기 2. ATM 만들기 3. ATM 이용하기 4. 정보 출력하기 5. 계좌 정보 접근6. 이전 화면으로 돌아가기 : 5
계좌번호를 입력해주세요 : 111111111111
거래 ID: 1
거래 유형: 계좌_계좌이체
카드 번호: 1003
계좌_계좌이체금액: 1000원
이체 계좌 -> 대구은행은행: 333333333333, 소유주: 김민관
입금 계좌 -> 한국은행은행: 111111111111, 소유주: 강승수
거래 후 금액: 96000원

거래 ID: 2
거래 유형: 현금_계좌이체
카드 번호: 1003
현금_계좌이체 금액: 10000원
현금_계좌이체 계좌 -> 한국은행은행: 111111111111, 소유주: 강승수
거래 후 금액: 111000원

거래 ID: 3
거래 유형: 입금
카드 번호: 1001
입금 금액: 10000원
입금 계좌 -> 한국은행은행: 111111111111, 소유주: 강승수
거래 후 금액: 121000원

거래 ID: 4
거래 유형: 출금
카드 번호: 1001
출금 금액: 10000원
출금 계좌 -> 한국은행은행: 111111111111, 소유주: 강승수
거래 후 금액: 110000원
```

(REQ1.6) A user may have multiple Accounts in a Bank

<은행에 강승수 user가 여러 개의 계좌를 가진 경우>

<은행 계좌 만들기>
당신의 이름을 입력해주세요 : 강승수
계좌번호를 입력해주세요 : 111111111111
계좌에 가능한 돈이 얼마인지를 입력해주세요 : 100000
계좌의 비밀번호는 무엇으로 할지 입력해주세요 : 1234
card가 생성되었습니다. 카드의 은행은 대구은행이며 카드 번호는 1001입니다.
계좌가 개설되었습니다. 계좌의 은행은 대구은행이며 유저 이름은 강승수입니다. 계좌번호는 111111111111입니다.

<은행 계좌 만들기>
당신의 이름을 입력해주세요 : 강승수
계좌번호를 입력해주세요 : 333333333333
계좌에 가능한 돈이 얼마인지를 입력해주세요 : 100000
계좌의 비밀번호는 무엇으로 할지 입력해주세요 : 1234
card가 생성되었습니다. 카드의 은행은 대구은행이며 카드 번호는 1002입니다.
계좌가 개설되었습니다. 계좌의 은행은 대구은행이며 유저 이름은 강승수입니다. 계좌번호는 333333333333입니다.

(REQ1.7) A user may have Accounts in multiple Banks.

<대구은행과 한국은행에 계좌가 있는 경우>

<은행 계좌 만들기>
당신의 이름을 입력해주세요 : 강승수
계좌번호를 입력해주세요 : 111111111111
계좌에 가능한 돈이 얼마인지를 입력해주세요 : 100000
계좌의 비밀번호는 무엇으로 할지 입력해주세요 : 1234
card가 생성되었습니다. 카드의 은행은 대구은행이며 카드 번호는 1001입니다.
계좌가 개설되었습니다. 계좌의 은행은 대구은행이며 유저 이름은 강승수입니다. 계좌번호는 111111111111입니다.

<은행 계좌 만들기>
당신의 이름을 입력해주세요 : 강승수
계좌번호를 입력해주세요 : 444444444444
계좌에 가능한 돈이 얼마인지를 입력해주세요 : 100000
계좌의 비밀번호는 무엇으로 할지 입력해주세요 : 1234
card가 생성되었습니다. 카드의 은행은 한국은행이며 카드 번호는 1002입니다.
계좌가 개설되었습니다. 계좌의 은행은 한국은행이며 유저 이름은 강승수입니다. 계좌번호는 444444444444입니다.

(REQ1.8) Each ATM have several types of transaction fees, and paid as follows:

- Deposit fee for non-primary banks: KRW 1,000; the fee is paid by inserting additional cash.

카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요. 1001
비밀번호를 입력해주세요.
1234
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 1
타 은행 입금에는 수수료 1000원이 발생합니다. 수수료를 포함해서 입금하세요
천원을 몇 장 넣으시겠습니까? 10
오천원을 몇 장 넣으시겠습니까? 0
만원을 몇 장 넣으시겠습니까? 0
오만원을 몇 장 넣으시겠습니까? 0
수표를 몇 장 넣으시겠습니까? 0

9000원이 입금되었습니다.
잔액은 109000

- Deposit fee for primary banks: KRW 0

카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요. 1001
비밀번호를 입력해주세요.
1234
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 1
천원을 몇 장 넣으시겠습니까? 10
오천원을 몇 장 넣으시겠습니까? 0
만원을 몇 장 넣으시겠습니까? 0
오만원을 몇 장 넣으시겠습니까? 0
수표를 몇 장 넣으시겠습니까? 0

10000원이 입금되었습니다.
잔액은 119000

- Withdrawal fee for a primary bank: KRW 1,000; the fee is paid from the withdrawal account.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2
자 은행 출금에는 수수료 1000원이 발생합니다.
얼마를 출금하시겠습니까?(수수료 제외) 3000

은행에서 4000원이 출금되었고 가져가실 현금은 총 3000원입니다.
천원 3장 / 오천원 0장 / 만원 0장 / 오만원 0장
잔액은 115000

- Withdrawal fee for non-primary banks: KRW 2,000; the fee is paid from the withdrawal account.

카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요.1001
비밀번호를 입력해주세요.
1234
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2
타 은행 출금에는 수수료 2000원이 발생합니다.
얼마를 출금하시겠습니까?(수수료 제외) 3000

은행에서 5000원이 출금되었고 가져가실 현금은 총 3000원입니다.
천원 3장 / 오천원 0장 / 만원 0장 / 오만원 0장
잔액은 110000

- Account transfer fee between primary banks: KRW 2,000; the fee is paid from the source account.

카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요.1001
비밀번호를 입력해주세요.
1234
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3
번호를 고르시오. 1. 현금 이체 2. 계좌 이체: 2
돈을 받을 계좌의 계좌번호를 입력하세요: 666666666666
본인의 계좌번호를 입력하세요: 111111111111
송금하실 금액을 알려주세요: 3000
3000+2000 원이 계좌에서 출금되었습니다.
계좌이체가 완료되었습니다.

- Account transfer fee between primary and non-primary banks: KRW 3,000; the fee is paid from the source account.

은행에서 1000원이 출금되었고 가져가실 현금은 총 0원입니다.
천원 0장 / 오천원 0장 / 만원 0장 / 오만원 0장
잔액은 104000
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3
번호를 고르시오. 1. 현금 이체 2. 계좌 이체: 2
돈을 받을 계좌의 계좌번호를 입력하세요: 555555555555
본인의 계좌번호를 입력하세요: 111111111111
송금하실 금액을 알려주세요: 3000
3000+3000 원이 계좌에서 출금되었습니다.
계좌이체가 완료되었습니다.

- Account transfer fee between non-primary banks: KRW 4,000; the fee is paid from the source account.

카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요. 1001
비밀번호를 입력해주세요.

1234

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3

번호를 고르시오. 1. 현금 이체 2. 계좌 이체: 2

돈을 받을 계좌의 계좌번호를 입력하세요: 777777777777

본인의 계좌번호를 입력하세요: 111111111111

송금하실 금액을 알려주세요: 3000

3000+4000 원이 계좌에서 출금되었습니다.

계좌이체가 완료되었습니다.

- Cash transfer fee to any bank type: KRW 5,000; the fee is paid by inserting additional cash.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3

번호를 고르시오. 1. 현금 이체 2. 계좌 이체: 1

돈을 받고자 하는 계좌의 계좌번호를 입력하시오. : 777777777777

돈을 얼마나 보내겠습니까? 30000

5000원의 수수료가 드니, 수수료를 포함해서 현금을 넣어주세요

1000원 몇 장 넣겠습니까? 35

5000원 몇 장 넣겠습니까? 0

10000원 몇 장 넣겠습니까? 0

50000원 몇 장 넣겠습니까? 0

이체가 완료되었습니다.

(REQ1.9) An admin can access the menu of “Transaction History” via an admin card (See REQ Display of Transaction History).

Admin card number 7777777을 입력하면 Transaction History에 access 할 수 있다.

<ATM 선택하기>
선택하고 싶은 atm의 번호를 기입해주세요. 이전으로 가고 싶으시면 -1을 입력해주세요.
0. atm serial number : 123456
0
어떤 언어를 고르시겠습니까? Which language do you prefer?
0. 영어 1. 한국어
1
카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요. 7777777
선택하실 메뉴를 골라주세요.
메뉴
1. 거래 기록
1
거래 ID: 1
거래 유형: 계좌_계좌이체
카드 번호: 1003
계좌_계좌이체금액: 1000원
이체_계좌 -> 대구은행은행: 333333333333, 소유주: 김민관
입금_계좌 -> 한국은행은행: 111111111111, 소유주: 강승수
거래 후 금액: 96000원

거래 ID: 2
거래 유형: 현금_계좌이체
카드 번호: 1003
현금_계좌이체 금액: 10000원
현금_계좌이체 계좌 -> 한국은행은행: 111111111111, 소유주: 강승수
거래 후 금액: 111000원

거래 ID: 3
거래 유형: 입금
카드 번호: 1001
입금 금액: 10000원
입금_계좌 -> 한국은행은행: 111111111111, 소유주: 강승수
거래 후 금액: 121000원

거래 ID: 4
거래 유형: 출금
카드 번호: 1001
출금 금액: 10000원
출금_계좌 -> 한국은행은행: 111111111111, 소유주: 강승수
거래 후 금액: 110000원

세션을 종료합니다.

(REQ1.10) 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)

<입금>

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 1
타 은행 입금에는 수수료 1000원이 발생합니다. 수수료를 포함해서 입금하세요
천원을 몇 장 넣으시겠습니까? 10
오천원을 몇 장 넣으시겠습니까? 10
만원을 몇 장 넣으시겠습니까? 10
오만원을 몇 장 넣으시겠습니까? 10
수표를 몇 장 넣으시겠습니까? 1

1번째 수표의 금액은 얼마입니까? 5000

잘못된 수표 금액입니다.

1번째 수표의 금액은 얼마입니까? 234567

893567원이 입금되었습니다.

잔액은 1406567

<출금>

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2
타 은행 출금에는 수수료 2000원이 발생합니다.
얼마를 출금하시겠습니까?(수수료 제외) 10000

은행에서 12000원이 출금되었고 가져가실 현금은 총 10000원입니다.
천원 0장 / 오천원 0장 / 만원 1장 / 오만원 0장

잔액은 1394567

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2
타 은행 출금에는 수수료 2000원이 발생합니다.
얼마를 출금하시겠습니까?(수수료 제외) 124000

은행에서 126000원이 출금되었고 가져가실 현금은 총 124000원입니다.
천원 4장 / 오천원 0장 / 만원 2장 / 오만원 2장

잔액은 1268567

<현금 이체>

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3
번호를 고르시오. 1. 현금 이체 2. 계좌 이체: 1
돈을 받고자 하는 계좌의 계좌번호를 입력하시오. : 777777777777
돈을 얼마나 보내겠습니까? 30000
돈을 얼마나 보내시겠습니까? 5000원의 수수료가 듭니다. 수수료 제외해서 입력해주세요.
1000원 몇 장 넣겠습니까? 20
5000원 몇 장 넣겠습니까? 3
10000원 몇 장 넣겠습니까? 0
50000원 몇 장 넣겠습니까? 0
이체가 완료되었습니다.

(REQ1.11) 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.

<은행 계좌 생성>

```
<은행 계좌 만들기>
당신의 이름을 입력해주세요 : 강승수
계좌번호를 입력해주세요 : 111111111111
계좌에 가능한 돈이 얼마인지를 입력해주세요 : 100000
계좌의 비밀번호는 무엇으로 할지 입력해주세요 : 1234
card가 생성되었습니다. 카드의 은행은 대구은행이며 카드 번호는 1001입니다.
계좌가 개설되었습니다. 계좌의 은행은 대구은행이며 유저 이름은 강승수입니다. 계좌번호는 111111111111입니다.
```

<은행 ATM 생성>

```
<ATM 만들기>
ATM의 SERIAL NUMBER를 입력해주세요 123456
ATM type에 해당하는 번호를 입력해주세요
0. Single Bank ATM 1. Multi-Bank ATM 1
ATM의 언어 지원 여부에 해당하는 번호를 입력해주세요
0. Unilingual 1. Bilingual 1
돈은 얼마나 들어있게 할까요?
천원을 몇 장 넣으시겠습니까? 10
오천원을 몇 장 넣으시겠습니까? 10
만원을 몇 장 넣으시겠습니까? 10
오만원을 몇 장 넣으시겠습니까? 10
admin card가 생성되었습니다. 카드의 은행은 한국은행입니다.
atm 이 생성되었습니다. atm의 serial number은 123456입니다. 현재 돈은 660000보유하고 있습니다.
```

2. ATM Session

(REQ2.1) A session starts when a user inserts a card.

```
std::cout << "Please insert the card.(Please enter the card number). If you want to exit, enter -1.";
std::cin >> number; // 카드 번호 받기
set_card_insert_slot(number);
if (get_card_insert_slot() == -1) {
    return;
}
if (get_card_insert_slot() == 7777777) {
    admin_interface();
    goto end;
}
if (valid_cardnumber_account(get_card_insert_slot()) == 0) {
    cout << "Invalid card number." << endl;
    goto main1;
}
increase_session(); //session start
```

(REQ2.2) 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).

<세션 종료하기>

```
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 4
세션을 종료합니다.
```

<atm에 충분한 현금이 없으면, session end 대신 다시 출금 화면으로 돌아가게 하였다.>

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2
자 은행 출금에는 수수료 1000원이 발생합니다.
얼마를 출금하시겠습니까?(수수료 제외) 500000

ATM에 충분한 현금이 존재하지 않습니다
얼마를 출금하시겠습니까?(수수료 제외)

<비밀번호 3번 틀리면 세션 종료>

카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요. 1001
비밀번호를 입력해주세요.
1564
비밀번호를 입력해주세요.
1646
비밀번호를 입력해주세요.
3546
비밀번호를 3번 틀리셨습니다.
세션을 종료합니다.

(REQ2.3) When a session ends, the summary of all transactions performed in a session must be displayed.

- (e.g.) Account/card info, transaction types (deposit, transfer, withdrawal), and their amount, ...

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 4
세션을 종료합니다.
거래 ID: 20
거래 유형: 입금
카드 번호: 1001
입금 금액: 9000원
입금 계좌 -> 대구은행은행: 111111111111, 소유주: 강승수
거래 후 금액: 2183567원

거래 ID: 21
거래 유형: 출금
카드 번호: 1001
출금 금액: 3000원
출금 계좌 -> 대구은행은행: 111111111111, 소유주: 강승수
거래 후 금액: 2178567원

거래 ID: 22
거래 유형: 계좌_계좌이체
카드 번호: 1001
계좌_계좌이체금액: 3000원
이체 계좌 -> 대구은행은행: 111111111111, 소유주: 강승수
입금 계좌 -> 한국은행은행: 555555555555, 소유주: 윤현서
거래 후 금액: 2172567원

(REQ2.4) Each transaction has a unique identifier across all sessions.

<거래 ID를 통해 식별할 수 있다.>

```
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.  
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 4  
세션을 종료합니다.  
거래 ID: 20  
거래 유형: 입금  
카드 번호: 1001  
입금 금액: 9000원  
입금 계좌 -> 대구은행은행: 111111111111, 소유주: 강승수  
거래 후 금액: 2183567원  
  
거래 ID: 21  
거래 유형: 출금  
카드 번호: 1001  
출금 금액: 3000원  
출금 계좌 -> 대구은행은행: 111111111111, 소유주: 강승수  
거래 후 금액: 2178567원  
  
거래 ID: 22  
거래 유형: 계좌_계좌이체  
카드 번호: 1001  
계좌_계좌이체금액: 3000원  
이체 계좌 -> 대구은행은행: 111111111111, 소유주: 강승수  
입금 계좌 -> 한국은행은행: 555555555555, 소유주: 윤현서  
거래 후 금액: 2172567원
```

3. User Authorization

(REQ3.1) An ATM checks if the inserted card is valid for the current type of ATM.

```
<ATM 만들기>  
ATM의 SERIAL NUMBER를 입력해주세요 111111  
ATM type에 해당하는 번호를 입력해주세요  
0. Single Bank ATM 1. Multi-Bank ATM 0  
ATM의 언어 지원 여부에 해당하는 번호를 입력해주세요  
0. Unilingual 1. Bilingual 1  
돈은 얼마나 들어있게 할까요?  
천원을 몇 장 넣으시겠습니까? 5  
오천원을 몇 장 넣으시겠습니까? 5  
만원을 몇 장 넣으시겠습니까? 0  
오만원을 몇 장 넣으시겠습니까? 0  
admin card가 생성되었습니다. 카드의 은행은 대구은행입니다.  
atm 이 생성되었습니다. atm의 serial number은 111111입니다. 현재 돈은 30000보유하고 있습니다.
```

대구은행 Single.bi ATM을 생성하였다.

```
<은행 계좌 만들기>  
당신의 이름을 입력해주세요 : Kate  
계좌번호를 입력해주세요 : 200000000000  
계좌에 가능한 돈이 얼마인지를 입력해주세요 : 50000  
계좌의 비밀번호는 무엇으로 할지 입력해주세요 : 1234  
card가 생성되었습니다. 카드의 은행은 대구은행이며 카드 번호는 1002입니다.  
계좌가 개설되었습니다. 계좌의 은행은 대구은행이며 유저 이름은 Kate입니다. 계좌번호는 200000000000입니다.
```

대구은행에서 Kate라는 사람의 계좌를 만들었고 카드번호는 1002이다.

```
카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요. 1002  
비밀번호를 입력해주세요.  
1234  
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.  
1. 입금 2. 출금 3. 이체 4. 세션 종료 :
```

Kate의 카드의 은행과 ATM의 은행이 대구은행으로 같기 때문에 비밀번호를 입력받아 본인확인을 한 뒤 ATM 이용을 할 수 있도록 하였다.

(REQ3.2) If an invalid card is inserted, the ATM shall display an appropriate error message (e.g., Invalid Card).

<은행 계좌 만들기>
당신의 이름을 입력해주세요 : David
계좌번호를 입력해주세요 : 100000000000
계좌에 가능한 돈이 얼마인지를 입력해주세요 : 5000
계좌의 비밀번호는 무엇으로 할지 입력해주세요 : 1234
card가 생성되었습니다. 카드의 은행은 한국은행이며 카드 번호는 1001입니다.
계좌가 개설되었습니다. 계좌의 은행은 한국은행이며 유저 이름은 David입니다. 계좌 번호는 100000000000입니다.

한국은행에 David라는 사람의 계좌를 만들었고 카드 번호는 1001이다.

카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요. 1001
주은행 계좌가 아닙니다.

대구은행 Single.bi ATM에 한국은행 카드를 넣으면 주거래 은행이 아닌 카드로는 거래가 불 가능하기 때문에 경고문을 띄우고 다시 ATM 초기화면으로 돌아가도록 하였다.

(REQ3.3) An ATM shall ask a user to enter the password (e.g., Enter Password), and verify if the password is correct.

카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요. 1001
비밀번호를 입력해주세요.
1234
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 :

카드번호를 입력받은 뒤 비밀번호를 입력하여 맞다면 ATM을 이용할 수 있게 구성하였다.

(REQ3.4) If the entered password is incorrect, the ATM shall display an appropriate error message(e.g., Wrong Password).

카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요. 1001
비밀번호를 입력해주세요.
1111
비밀번호가 틀립니다.
비밀번호를 입력해주세요.

비밀번호를 1234로 설정한 다음 1111을 입력하면 ‘비밀번호가 틀렸다고 뜬 뒤 다시 비밀번호를 입력받도록 하였다.

(REQ3.5) If a user enters wrong passwords 3 times in a row, a session is aborted, and return the card to the user.

카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요. 1001
비밀번호를 입력해주세요.
1111
비밀번호가 틀립니다.
비밀번호를 입력해주세요.
2222
비밀번호가 틀립니다.
비밀번호를 입력해주세요.
3333
비밀번호가 틀립니다.
비밀번호를 3번 틀리셨습니다.
세션을 종료합니다.
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 은행 만들기 2. 은행 선택하기 3. 정보 출력하기 :

비밀번호를 3번 틀린 경우, session이 종료되고 card slot을 초기화하여 카드를 사용자에게 돌려주었다. 그리고 ATM의 초기화면으로 돌아가게 하였다.

4. Deposit

(REQ4.1) An ATM shall take either cash or check from a user.

현금을 입금받는다.

1. 입금 2. 출금 3. 이제 4. 세션 종료 : 1
타 은행 입금에는 수수료 1000원이 발생합니다. 수수료를 포함해서 입금하세요
천원을 몇 장 넣으시겠습니까? 11
오천원을 몇 장 넣으시겠습니까? 10
만원을 몇 장 넣으시겠습니까? 10
오만원을 몇 장 넣으시겠습니까? 10
수표를 몇 장 넣으시겠습니까? 0

660000원이 입금되었습니다.

수표를 입금받는다.

1. 입금 2. 출금 3. 이제 4. 세션 종료 : 1
타 은행 입금에는 수수료 1000원이 발생합니다. 수수료를 포함해서 입금하세요
천원을 몇 장 넣으시겠습니까? 0
오천원을 몇 장 넣으시겠습니까? 0
만원을 몇 장 넣으시겠습니까? 0
오만원을 몇 장 넣으시겠습니까? 0
수표를 몇 장 넣으시겠습니까? 1

1번째 수표의 금액은 얼마입니까? 100000
99000원이 입금되었습니다.

(REQ4.2) An ATM shall display an appropriate error message if the number of the inserted cash or checks exceed the limit allowed by the ATM.

현금이 50장을 초과하면 가능한 현금 장 수를 초과했다고 안내하고 다시 현금을 입력받는다.

천원을 몇 장 넣으시겠습니까? 30
오천원을 몇 장 넣으시겠습니까? 30
만원을 몇 장 넣으시겠습니까? 30
오만원을 몇 장 넣으시겠습니까? 30
가능한 현금 장 수를 초과하셨습니다.
천원을 몇 장 넣으시겠습니까?

수표가 30장을 초과하면 가능한 수표 수를 초과했다고 안내하고 다시 수표 수를 입력받는다.

수표를 몇 장 넣으시겠습니까? 50

가능한 수표 장 수를 초과하셨습니다.
수표를 몇 장 넣으시겠습니까?

(REQ4.3) 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).

1. 은행 계좌 만들기 2. ATM 만들기 3. ATM 이용하기 4. 정보 출력하기 5. 계좌 정보 접근 6. 이전 화면으로 돌아가기 : 4
[Account 0] Balance:1000000
[Account 1] Balance:1000000
[ATM 0] Remaining Cash: 660000(1000*10장 5000*10장 10000*10장 50000*10장)

1. 입금 2. 출금 3. 이제 4. 세션 종료 : 1
타 은행 입금에는 수수료 1000원이 발생합니다. 수수료를 포함해서 입금하세요
천원을 몇 장 넣으시겠습니까? 11
오천원을 몇 장 넣으시겠습니까? 10
만원을 몇 장 넣으시겠습니까? 10
오만원을 몇 장 넣으시겠습니까? 10
수표를 몇 장 넣으시겠습니까? 0

660000원이 입금되었습니다.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 4
세션을 종료합니다.
거래 ID: 1
거래 유형: 입금
카드 번호: 1001
입금 금액: 660000원
입금 계좌 -> 대구은행: 100000000001, 소유주: 정세훈
거래 후 금액: 1660000원

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 은행 계좌 만들기 2. ATM 만들기 3. ATM 이용하기 4. 정보 출력하기 5. 계좌 정보 접근 6. 이전 화면으로 돌아가기 : 4
[Account 0] Balance:1660000
[Account 1] Balance:1000000
[ATM 0] Remaining Cash: 1321000(1000*21장 5000*20장 10000*20장 50000*20장)

입금 금액이 [Account 0]에 반영되었다.

(REQ4.4) Some deposit fee may be charged (See REQ in System Setup)

ATM의 primary bank와 카드의 은행이 다르면 수수료 1000원이 부과된다.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 1
타 은행 입금에는 수수료 1000원이 발생합니다. 수수료를 포함해서 입금하세요
천원을 몇 장 넣으시겠습니까? 11
오천원을 몇 장 넣으시겠습니까? 10
만원을 몇 장 넣으시겠습니까? 10
오만원을 몇 장 넣으시겠습니까? 10
수표를 몇 장 넣으시겠습니까? 0

660000원이 입금되었습니다.

(REQ4.5) The deposited cash increase available cash in ATM that can be used by other users.

기존 ATM이 가지고 있던 현금은 다음과 같다.

1. 은행 계좌 만들기 2. ATM 만들기 3. ATM 이용하기 4. 정보 출력하기 5. 계좌 정보 접근 6. 이전 화면으로 돌아가기 : 4
[Account 0] Balance:1000000
[Account 1] Balance:1000000
[ATM 0] Remaining Cash: 660000(1000*10장 5000*10장 10000*10장 50000*10장)

유저가 현금을 투입한다.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 1
타 은행 입금에는 수수료 1000원이 발생합니다. 수수료를 포함해서 입금하세요
천원을 몇 장 넣으시겠습니까? 11
오천원을 몇 장 넣으시겠습니까? 10
만원을 몇 장 넣으시겠습니까? 10
오만원을 몇 장 넣으시겠습니까? 10
수표를 몇 장 넣으시겠습니까? 0

660000원이 입금되었습니다.

투입한 현금이 ATM에 반영된다.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 은행 계좌 만들기 2. ATM 만들기 3. ATM 이용하기 4. 정보 출력하기 5. 계좌 정보 접근 6. 이전 화면으로 돌아가기 : 4
[Account 0] Balance:1660000
[Account 1] Balance:1000000
[ATM 0] Remaining Cash: 1321000(1000*21장 5000*20장 10000*20장 50000*20장)

(REQ4.6) The deposited check does not increase available cash in ATM that can be used by other users.

기존 ATM이 가지고 있던 현금은 다음과 같다.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 은행 계좌 만들기 2. ATM 만들기 3. ATM 이용하기 4. 정보 출력하기 5. 계좌 정보 접근 6. 이전 화면으로 돌아가기 : 4
[Account 0] Balance:1660000
[Account 1] Balance:1000000
[ATM 0] Remaining Cash: 1321000(1000*21장 5000*20장 10000*20장 50000*20장)

유저가 수표를 투입한다.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 1

타 은행 입금에는 수수료 1000원이 발생합니다. 수수료를 포함해서 입금하세요

천원을 몇 장 넣으시겠습니까? 0

오천원을 몇 장 넣으시겠습니까? 0

만원을 몇 장 넣으시겠습니까? 0

오만원을 몇 장 넣으시겠습니까? 0

수표를 몇 장 넣으시겠습니까? 1

1번 째 수표의 금액은 얼마입니까? 100000

99000원이 입금되었습니다.

잔액은 1759000

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 4

세션을 종료합니다.

거래 ID: 2

거래 유형: 입금

카드 번호: 1001

입금 금액: 99000원

입금 계좌 -> 대구은행: 100000000001, 소유주: 정세훈

거래 후 금액: 1759000원

투입한 수표가 ATM에 반영되지 않는다.

1. 은행 계좌 만들기 2. ATM 만들기 3. ATM 이용하기 4. 정보 출력하기 5. 계좌 정보 접근 6. 이전 화면으로 돌아가기 : 4
[Account 0] Balance:1759000
[Account 1] Balance:1000000
[ATM 0] Remaining Cash: 1321000(1000*21장 5000*20장 10000*20장 50000*20장)

5. Withdrawal

(REQ5.1) An ATM shall ask a user to enter the amount of fund to withdraw.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2

자 은행 출금에는 수수료 1000원이 발생합니다.

얼마를 출금하시겠습니까?(수수료 제외)

출금이 실행되면 수수료를 안내하고 얼마나 출금할지 금액을 입력받는다.

(REQ5.2) An ATM shall display an appropriate error message if there is insufficient fund in the account or insufficient cash in the ATM.

계좌에 돈이 충분하지 않을 경우 에러 메시지를 출력한다.

당신의 이름을 입력해주세요 : 구세운
계좌 번호를 입력해주세요 : 100000000002
계좌에 가능한 돈이 얼마인지를 입력해주세요 : 100000
계좌의 비밀번호는 무엇으로 할지 입력해주세요 : 1234
card가 생성되었습니다. 카드의 은행은 현풍이며 카드 번호는 1002입니다.
계좌가 개설되었습니다. 계좌의 은행은 현풍이며 유저 이름은 구세운입니다. 계좌 번호는 100000000002입니다.

요청된 금액은 500000원이나 계좌가 소유하고 있는 금액은 100000원이다. 따라서 에러 메시지를 출력한다.

얼마를 출금하시겠습니까?(수수료 제외) 500000

계좌에 돈이 충분하지 않습니다

ATM에 돈이 충분하지 않을 경우 에러 메시지를 출력한다.

<ATM 만들기>
ATM의 SERIAL NUMBER을 입력해주세요 3
잘못된 ATM SERIAL NUMBER의 유형입니다. 6자리로 입력해주세요.
ATM의 SERIAL NUMBER을 입력해주세요 100002
ATM type에 해당하는 번호를 입력해주세요
0. Single Bank ATM 1. Multi-Bank ATM 1
ATM의 언어 지원 여부에 해당하는 번호를 입력해주세요
0. Unilingual 1. Bilingual 1
돈은 얼마나 들어있게 할까요?
천원을 몇 장 넣으시겠습니까? 5
오천원을 몇 장 넣으시겠습니까? 5
만원을 몇 장 넣으시겠습니까? 5
오만원을 몇 장 넣으시겠습니까? 5
admin card가 생성되었습니다. 카드의 은행은 현풍입니다.
atm 이 생성되었습니다. atm의 serial number은 100002입니다. 현재 돈은 330000보유하고 있습니다.

ATM이 가지고 있는 돈은 330000원이나 요청된 금액은 400000원이다. 따라서 에러 메시지를 출력한다.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2
타 은행 출금에는 수수료 2000원이 발생합니다.
얼마를 출금하시겠습니까?(수수료 제외) 400000

ATM에 충분한 현금이 존재하지 않습니다

(REQ5.3) 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).

계좌가 가지고 있던 돈은 100000원 이었다.

당신의 이름을 입력해주세요 : 구세운
계좌번호를 입력해주세요 : 100000000002
계좌에 가능한 돈이 얼마인지를 입력해주세요 : 100000
계좌의 비밀번호는 무엇으로 할지 입력해주세요 : 1234
card가 생성되었습니다. 카드의 은행은 현풍이며 카드 번호는 1002입니다.
계좌가 개설되었습니다. 계좌의 은행은 현풍이며 유저 이름은 구세운입니다. 계좌 번호는 100000000002입니다.

10000원을 출금하자 수수료 1000원을 포함한 11000원이 출금된다.

얼마를 출금하시겠습니까?(수수료 제외) 10000

은행에서 11000원이 출금되었고 가져가실 현금은 총 10000원입니다.
천원 0장 / 오천원 0장 / 만원 1장 / 오만원 0장
잔액은 89000

100000원에서 11000을 제한 89000원이 계좌에 반영되었다.

세션을 종료합니다.
거래 ID: 1
거래 유형: 출금
카드 번호: 1002
출금 금액: 10000원
출금 계좌 -> 현풍은행: 100000000002, 소유주: 구세운
거래 후 금액: 89000원

(REQ5.4) Some withdrawal fee may be charged (See REQ in System Setup).

ATM의 primary bank가 사용하려는 계좌의 bank인 경우 1000원의 수수료가 발생한다.

```
당신의 이름을 입력해주세요 : 구세운  
계좌번호를 입력해주세요 : 100000000002  
계좌에 가능한 돈이 얼마인지를 입력해주세요 : 100000  
계좌의 비밀번호는 무엇으로 할지 입력해주세요 : 1234  
card가 생성되었습니다. 카드의 은행은 현풍이며 카드 번호는 1002입니다.  
계좌가 개설되었습니다. 계좌의 은행은 현풍이며 유저 이름은 구세운입니다. 계좌  
번호는 100000000002입니다.
```

수수료를 포함한 11000원이 계좌에서 출금되어 잔액이 89000원이 되었다.

얼마를 출금하시겠습니까?(수수료 제외) 10000

```
은행에서 11000원이 출금되었고 가져가실 현금은 총 10000원입니다.  
천원 0장 / 오천원 0장 / 만원 1장 / 오만원 0장  
잔액은 89000
```

ATM의 primary bank가 사용하려는 계좌의 bank가 아닌 경우 2000원의 수수료가 발생한다.

```
<은행 계좌 만들기>  
당신의 이름을 입력해주세요 : 구세운  
계좌번호를 입력해주세요 : 100000000002  
계좌에 가능한 돈이 얼마인지를 입력해주세요 : 100000  
계좌의 비밀번호는 무엇으로 할지 입력해주세요 : 1234  
card가 생성되었습니다. 카드의 은행은 현풍이며 카드 번호는 1002입니다.  
계좌가 개설되었습니다. 계좌의 은행은 현풍이며 유저 이름은 구세운입니다. 계좌  
번호는 100000000002입니다.
```

수수료를 포함한 12000원이 계좌에서 출금되어 잔액이 988000원이 되었다.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2

타 은행 출금에는 수수료 2000원이 발생합니다.

얼마를 출금하시겠습니까?(수수료 제외) 400000

ATM에 충분한 현금이 존재하지 않습니다

얼마를 출금하시겠습니까?(수수료 제외) 10000

```
은행에서 12000원이 출금되었고 가져가실 현금은 총 10000원입니다.  
천원 0장 / 오천원 0장 / 만원 1장 / 오만원 0장  
잔액은 988000
```

(REQ5.5) The cash withdrawal lower available cash in the ATM that can be used by other users.

ATM 0이 원래 소유하고 있었던 50000원은 20장 이었다.

1. 은행 만들기 2. 은행 선택하기 3. 정보 출력하기 : 3

[Account 0] Balance:988000

[Account 1] Balance:89000

[ATM 0]Remaining Cash: 1310000(1000*20장 5000*20장 10000*19장 50000*20장)

[ATM 1]Remaining Cash: 320000(1000*5장 5000*5장 10000*4장 50000*5장)

500000원을 출금해 50000원 10장을 가져갔다.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2

타 은행 출금에는 수수료 2000원이 발생합니다.

얼마를 출금하시겠습니까?(수수료 제외) 501000

출금 가능하신 금액을 초과하셨습니다

얼마를 출금하시겠습니까?(수수료 제외) 500000

은행에서 502000원이 출금되었고 가져가실 현금은 총 500000원입니다.

천원 0장 / 오천원 0장 / 만원 0장 / 오만원 10장

잔액은 486000

ATM에 남은 50000원은 10장이다.

1. 은행 만들기 2. 은행 선택하기 3. 정보 출력하기 : 3

[Account 0] Balance:486000

[Account 1] Balance:89000

[ATM 0] Remaining Cash: 810000(1000*20장 5000*20장 10000*19장 50000*10장)

[ATM 1] Remaining Cash: 320000(1000*5장 5000*5장 10000*4장 50000*5장)

(REQ5.6) 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.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2

타 은행 출금에는 수수료 2000원이 발생합니다.

얼마를 출금하시겠습니까?(수수료 제외) 10000

은행에서 12000원이 출금되었고 가져가실 현금은 총 10000원입니다.

천원 0장 / 오천원 0장 / 만원 1장 / 오만원 0장

잔액은 474000

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2

타 은행 출금에는 수수료 2000원이 발생합니다.

얼마를 출금하시겠습니까?(수수료 제외) 10000

은행에서 12000원이 출금되었고 가져가실 현금은 총 10000원입니다.

천원 0장 / 오천원 0장 / 만원 1장 / 오만원 0장

잔액은 462000

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2

타 은행 출금에는 수수료 2000원이 발생합니다.

얼마를 출금하시겠습니까?(수수료 제외) 10000

은행에서 12000원이 출금되었고 가져가실 현금은 총 10000원입니다.

천원 0장 / 오천원 0장 / 만원 1장 / 오만원 0장

잔액은 450000

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2

더 출금하시려면 세션을 초기화 하셔야 합니다.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 :

출금을 3번이상 실행하자 세션을 초기화해야 한다는 안내문이 나오고 다시 업무를 선택하도록 한다.

(REQ5.7) The maximum amount of cash withdrawal per transaction is KRW 500,000

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2
타 은행 출금에는 수수료 2000원이 발생합니다.
얼마를 출금하시겠습니까?(수수료 제외) 501000

출금 가능하신 금액을 초과하셨습니다
얼마를 출금하시겠습니까?(수수료 제외) 500000

은행에서 502000원이 출금되었고 가져가실 현금은 총 500000원입니다.
천원 0장 / 오천원 0장 / 만원 0장 / 오만원 10장
잔액은 486000

500000원을 1000원 초과하는 금액을 요구하자 출금이 불가능하다고 안내한다. 그리고 500000원을 정확히 입력하자 출금을 해준다.

6. Transfer

(REQ6.1) An ATM shall ask a user to choose the transfer types either cash transfer or account fund transfer.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3
번호를 고르시오. 1. 현금 이체 2. 계좌 이체 :

이체를 선택하였을 때, 현금 이체를 할 것인지, 계좌 이체를 할 것인지 물어본다.

(REQ6.2) For both cash and account transfers, an ATM shall ask the destination account number where the fund is to be transferred.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3
번호를 고르시오. 1. 현금 이체 2. 계좌 이체: 1
돈을 받고자 하는 계좌의 계좌번호를 입력하시오. :

1번 현금 이체를 선택하였을 때, 돈을 받고자 하는 계좌의 계좌번호를 물어본다.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3
번호를 고르시오. 1. 현금 이체 2. 계좌 이체: 2
돈을 받을 계좌의 계좌번호를 입력하세요:

2번 계좌 이체를 선택하였을 때도 마찬가지로 돈을 받고자 하는 계좌의 계좌번호를 물어본다.

(REQ6.3) 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.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3
번호를 고르시오. 1. 현금 이체 2. 계좌 이체: 1
돈을 받고자 하는 계좌의 계좌번호를 입력하시오. : 100000000000
돈을 얼마나 보내겠습니까? 5000
5000원의 수수료가 드니, 수수료를 포함해서 현금을 넣어주세요
1000원 몇 장 넣겠습니까? 5
5000원 몇 장 넣겠습니까? 1
10000원 몇 장 넣겠습니까? 0
50000원 몇 장 넣겠습니까? 0
이체가 완료되었습니다.

현금 계좌이체 시 이체할 돈 5000원과 수수료 5000원을 포함하여 입력받도록 하며 실제 10000원이 입력되었을 때 이체가 정상적으로 완료된 모습을 볼 수 있다.

돈을 얼마나 보내겠습니까? 5000
5000원의 수수료가 드니, 수수료를 포함해서 현금을 넣어주세요
1000원 몇 장 넣겠습니까? 5
5000원 몇 장 넣겠습니까? 0
10000원 몇 장 넣겠습니까? 0
50000원 몇 장 넣겠습니까? 0
요청하신 금액과 지불하신 현금 액수가 다릅니다.
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 :

만약 5000원을 보내려고 하는데 수수료 5000원이 더한 10000원을 입력하지 않았을 경우에는 경고 메시지를 뜨게하고 ATM 초기화면으로 돌아가게 하였다.

(REQ6.4) For account transfer, an ATM shall ask the source account number, and the amount of fund to be transferred.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3
번호를 고르시오. 1. 현금 이체 2. 계좌 이체: 2
돈을 받을 계좌의 계좌번호를 입력하세요: 200000000000
본인의 계좌번호를 입력하세요: 100000000000
송금하실 금액을 알려주세요: 5000

계좌 이체를 선택하였을 때는 돈을 보낼 자신의 계좌와 보낼 돈의 액수를 입력하게 했다.

(REQ6.5) Some transfer fee may be charged (See REQ in System Setup).

위 요구사항을 검토하기 위해 다음과 같은 상황설정을 하였다.

한국은행 : David, Tom / 대구은행: Kate, Andy / ATM : multi_bi_ATM, 한국은행

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3

번호를 고르시오. 1. 현금 이체 2. 계좌 이체: 2

돈을 받을 계좌의 계좌번호를 입력하세요: 200000000000

본인의 계좌번호를 입력하세요: 100000000000

송금하실 금액을 알려주세요: 5000

5000+3000 원이 계좌에서 출금되었습니다.

계좌이체가 완료되었습니다.

한국은행 ATM 이용시 주 은행(David: 한국은행)에서 타 은행(Kate 대구은행)으로 계좌이체를 할 시 3000원의 수수료가 붙은 금액이 계좌 총액에서 빠지고 돈을 받는 계좌는 수수료를 제외한 금액을 받도록 하였다.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3

번호를 고르시오. 1. 현금 이체 2. 계좌 이체: 2

돈을 받을 계좌의 계좌번호를 입력하세요: 400000000000

본인의 계좌번호를 입력하세요: 100000000000

송금하실 금액을 알려주세요: 5000

5000+2000 원이 계좌에서 출금되었습니다.

계좌이체가 완료되었습니다.

한국은행 ATM 이용시 주 은행(David: 한국은행)에서 주 은행(Tom 한국은행)으로 계좌이체를 할 시 2000원의 수수료가 붙은 금액이 계좌 총액에서 빠지고 돈을 받는 계좌는 수수료를 제외한 금액을 받도록 하였다.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3

번호를 고르시오. 1. 현금 이체 2. 계좌 이체: 2

돈을 받을 계좌의 계좌번호를 입력하세요: 300000000000

본인의 계좌번호를 입력하세요: 200000000000

송금하실 금액을 알려주세요: 5000

5000+4000 원이 계좌에서 출금되었습니다.

계좌이체가 완료되었습니다.

한국은행 ATM 이용시 타 은행(Kate: 대구은행)에서 타 은행(Andy 대구은행)으로 계좌이체를 할 시 4000원의 수수료가 붙은 금액이 계좌 총액에서 빠지고 돈을 받는 계좌는 수수료를 제외한 금액을 받도록 하였다.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요 .
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3
번호를 고르시오. 1. 현금 이체 2. 계좌 이체: 1
돈을 받고자 하는 계좌의 계좌번호를 입력하시오. : 200000000000
돈을 얼마나 보내겠습니까? 5000
5000원의 수수료가 드니, 수수료를 포함해서 현금을 넣어주세요
1000원 몇 장 넣겠습니까? 5
5000원 몇 장 넣겠습니까? 1
10000원 몇 장 넣겠습니까? 0
50000원 몇 장 넣겠습니까? 0

현금 이체를 할때에는 은행의 종류에 상관없이 항상 수수료가 5000원이 부과되도록 하였다.

(REQ6.6) The inserted cash for transfer increase available cash in ATM that can be used by other users.

1. 은행 만들기 2. 은행 선택하기 3. 정보 출력하기 : 3
[Account 0] Balance:500000
[Account 1] Balance:500000
[Account 2] Balance:500000
[Account 3] Balance:500000
[ATM 0]Remaining Cash: 160000(1000*10장 5000*10장 10000*10장)
[ATM 1]Remaining Cash: 160000(1000*10장 5000*10장 10000*10장)

초기 ATM0에는 천원 10장, 오천원 10장, 만원 10장 총 160000원을 보유하고 있었다.

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 4
세션을 종료합니다.
거래 ID: 1
거래 유형: 현금_계좌이체
카드 번호: 1001
현금_계좌이체 금액: 5000원
현금_계좌이체 계좌 -> 한국은행은행: 100000000000, 소유주: David
거래 후 금액: 505000원

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 은행 만들기 2. 은행 선택하기 3. 정보 출력하기 : 3
[Account 0] Balance:505000
[Account 1] Balance:500000
[Account 2] Balance:500000
[Account 3] Balance:500000
[ATM 0] Remaining Cash: 170000(1000*15장 5000*11장 10000*10장)
[ATM 1] Remaining Cash: 160000(1000*10장 5000*10장 10000*10장)
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

현금 계좌이체를 위해 투입된 현금은 ATM0의 사용가능한 현금을 증가시켰으며 1000원 5장, 5000원 1장이 입력되었으므로 ATM0 금액도 그 만큼 증가하였다.

(REQ6.7) 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).

거래 ID: 1
거래 유형: 현금_계좌이체
카드 번호: 1001
현금_계좌이체 금액: 5000원
현금_계좌이체 계좌 -> 한국은행은행: 100000000000, 소유주: David
거래 후 금액: 505000원

David 계좌에 5000원이 추가되어야 하고 ATM에도 5000원이 추가되어야 한다.

거래 ID: 2
거래 유형: 계좌_계좌이체
카드 번호: 1001
계좌_계좌이체금액: 5000원
이체 계좌 -> 한국은행은행: 100000000000, 소유주: David
입금 계좌 -> 대구은행은행: 200000000000, 소유주: Kate
거래 후 금액: 497000원

David 계좌로부터 5000+3000(수수료)원이 빠지고 Kate 계좌에는 5000원이 증가되어야 한다.

```

거래 ID: 3
거래 유형: 계좌_계좌이체
카드 번호: 1001
계좌_계좌이체금액: 5000원
이체 계좌 -> 한국은행은행: 100000000000, 소유주: David
입금 계좌 -> 한국은행은행: 400000000000, 소유주: Tom
거래 후 금액: 490000원

```

David 계좌에서 5000+2000(수수료)원이 빠지고 Tom의 계좌에는 5000원이 증가되어야 한다.

```

거래 ID: 4
거래 유형: 계좌_계좌이체
카드 번호: 1002
계좌_계좌이체금액: 5000원
이체 계좌 -> 대구은행은행: 200000000000, 소유주: Kate
입금 계좌 -> 대구은행은행: 300000000000, 소유주: Andy
거래 후 금액: 496000원

```

Kate 계좌에서 5000+4000(수수료)원이 빠지고 Andy의 계좌에는 5000원이 증가되어야 한다.

```

[Account 0] Balance:490000
[Account 1] Balance:496000
[Account 2] Balance:505000
[Account 3] Balance:505000
[ATM 0] Remaining Cash: 170000(1000*15장 5000*11장 10000*10장 )
[ATM 1] Remaining Cash: 160000(1000*10장 5000*10장 10000*10장 )

```

David - Account 0(거래 ID 1,2) : $500000 + 5000 - (5000 + 3000) - (5000 + 2000) = 490000$ 원

Kate - Account 1(거래 ID 1,3,4) : $500000 + 5000 - (5000 + 4000) = 496000$ 원

Andy - Account 2(거래 ID 3) : $500000 + 5000 = 505000$ 원

Tom - Account 3(거래 ID 2) : $500000 + 5000 = 505000$ 원

현금 이체와 계좌 이체 후 돈을 보낸 계좌와 돈을 받은 계좌의 잔액이 정상적으로 업데이트 된 것을 확인할 수 있다.

7. Display of Transaction History(Admin Menu)

(REQ7.1) 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.

```

카드를 넣어 주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요. 7777777
선택하실 메뉴를 골라주세요.
메뉴
1. 거래 기록

```

ATM에 카드를 넣을 때 Admin 카드 '7777777'을 넣으면 '거래 기록' 메뉴만 화면에 나오도록 하였다.

(REQ7.2) 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.

```
메뉴
1. 거래 기록
1
거래 ID: 1
거래 유형: 현금_계좌이체
카드 번호: 1001
현금_계좌이체 금액: 5000원
현금_계좌이체 계좌 -> 한국은행은행: 100000000000, 소유주: David
거래 후 금액: 505000원

거래 ID: 2
거래 유형: 계좌_계좌이체
카드 번호: 1001
계좌_계좌이체금액: 5000원
이체 계좌 -> 한국은행은행: 100000000000, 소유주: David
입금 계좌 -> 대구은행은행: 200000000000, 소유주: Kate
거래 후 금액: 497000원

거래 ID: 3
거래 유형: 계좌_계좌이체
카드 번호: 1001
계좌_계좌이체금액: 5000원
이체 계좌 -> 한국은행은행: 100000000000, 소유주: David
입금 계좌 -> 한국은행은행: 400000000000, 소유주: Tom
거래 후 금액: 490000원

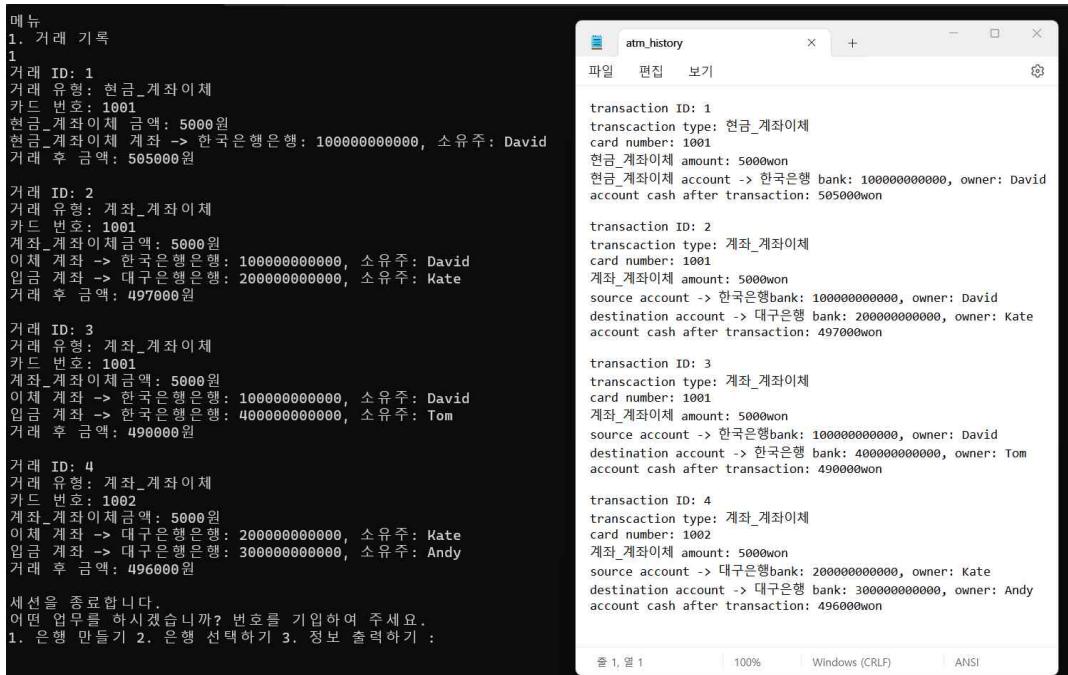
거래 ID: 4
거래 유형: 계좌_계좌이체
카드 번호: 1002
계좌_계좌이체금액: 5000원
이체 계좌 -> 대구은행은행: 200000000000, 소유주: Kate
입금 계좌 -> 대구은행은행: 300000000000, 소유주: Andy
거래 후 금액: 496000원

세션을 종료합니다.
```

Admin 메뉴의 “1. 거래 기록” 메뉴가 선택되면 시스템이 시작되고 발생한 거래 내역이 모두 출력되도록 하였다.

(REQ7.3) The “Transaction History” information shall be outputted to the external file (e.g., txt file).

📁 Debug	2023-11-28 오후 12:21	파일 폴더
📄 atm_history	2023-11-28 오후 12:29	텍스트 문서 1KB



1. 거래 내역 메뉴를 선택한다면 transaction history가 출력되고 txt 파일이 추가적으로 생성되는 것을 확인할 수 있다.

8. Multi-language support

(REQ8.1) 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.

```

<ATM 만들기>
ATM의 SERIAL NUMBER을 입력해주세요 100001
ATM type에 해당하는 번호를 입력해주세요
0. Single Bank ATM 1. Multi-Bank ATM 1
ATM의 언어 지원 여부에 해당하는 번호를 입력해주세요
0. Unilingual 1. Bilingual 1
돈은 얼마나 들어있게 할까요?
천원을 몇 장 넣으시겠습니까? 20
오천원을 몇 장 넣으시겠습니까? 20
만원을 몇 장 넣으시겠습니까? 20
오만원을 몇 장 넣으시겠습니까? 20
admin card가 생성되었습니다. 카드의 은행은행은행입니다.
atm 이 생성되었습니다. atm의 serial number은 100001입니다. 현재 돈은 1320000
보유하고 있습니다.

```

```

<ATM 선택하기>
선택하고 싶은 atm의 번호를 기입해주세요. 이전으로 가고 싶으시면 -1을 입력해주세요.
0. atm serial number : 100001
0
어떤 언어를 고르시겠습니까? Which language do you prefer?
0. 영어 1. 한국어

```

(REQ8.2) Once a certain language is chosen, all menus must be displayed using the chosen language.

영어를 고르자 오로지 영어만 나온다.

```
ATM의 언어 지원 여부에 해당하는 번호를 입력해주세요
0. Unilingual 1. Bilingual
돈은 얼마나 들어 있게 할까요?
전월을 몇 장 넣으시겠습니까? 10
오천월을 몇 장 넣으시겠습니까? 10
만월을 몇 장 넣으시겠습니까? 10
오만월을 몇 장 넣으시겠습니까? 10
admin card가 생성되었습니다. 카드의 은행은 대한입니다.
atm 이 생성되었습니다. atm의 serial number은 100001입니다. 현재 돈은 660000보유하고 있습니다.
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 은행 계좌 만들기 2. ATM 만들기 3. ATM 이용하기 4. 정보 출력하기 5. 계좌 정보 접근 6. 이전 화면으로 돌아가기 : 3
<ATM 선택하기>
선택하고 싶은 atm의 번호를 기입해주세요. 이전으로 가고 싶으시면 -1을 입력해주세요.
0. atm serial number : 100001
0
어떤 언어를 고르시겠습니까? Which language do you prefer?
0. 영어 1. 한국어
0
Please insert the card.(Please enter the card number). If you want to exit, enter -1.1001
Please enter the password.
1234
What kind of work would you like to do? Please enter the number.
1. deposit 2. withdrawl 3. transfer 4. end session: 1
How many 1,000 won would you like to put in? 10
How many 5,000 won would you like to put in? 10
How many 10,000 won would you like to put in? 10
How many 50,000 won would you like to put in? 10
How many checks would you like to write? 1

1th check, how much? 1000000
1660000Won has been deposited.
The balance is 1760000
What kind of work would you like to do? Please enter the number.
1. deposit 2. withdrawl 3. transfer 4. end session: 2
There is a fee of 1,000 won for withdrawal.
How much do you want to withdraw (excluding fees) : 100000

In the bank 101000won was withdrawn, Total cash to bring is 100000Won.
1000 Won 0sheets / 5000Won 0sheets / 10000 Won 0sheets / 50000 Won 2sheets.
The balance is 1659000
```

```
What kind of work would you like to do? Please enter the number.
1. deposit 2. withdrawl 3. transfer 4. end session: 3
Please choose between 1 Cash transfer or 2 Account transfer : 1
Please enter the Destination Account : 100000000002
Please enter the cash to send: 10000
Please enter the amount you wish to send along with the 5,000 won fee.
How many 1,000 won would you like to put in? 0
How many 5,000 won would you like to put in? 3
How many 10,000 won would you like to put in? 0
How many 50,000 won would you like to put in? 0
Cash transfer has been completed.

What kind of work would you like to do? Please enter the number.
1. deposit 2. withdrawl 3. transfer 4. end session: 3
Please choose between 1 Cash transfer or 2 Account transfer : 2
Please enter the Destination Account : 100000000002
Enter your account number : 100000000001
Please enter the cash to send : 10000
10000+2000 Won was deducted from your account.
The account transfer has been completed.
```

```
What kind of work would you like to do? Please enter the number.
1. deposit 2. withdrawl 3. transfer 4. end session: 4
Ends the session.
transaction ID: 1
transaction type: deposit
card number: 1001
deposit amount: 1660000won
deposit account -> 대한 bank: 100000000001, owner: 정
account cash after transaction: 1760000won
```

```
transaction ID: 2
transaction type: Withdrawl
card number: 1001
Withdrawl amount: 100000won
Withdrawl account -> 대한 bank: 100000000001, owner: 정
account cash after transaction: 1659000won
```

```
transaction ID: 3
transaction type: cash_transfer
card number: 1001
```

```
cash_transferamount: 10000won
source account -> 대한bank: 100000000001, owner: 정
destination account -> 대한 bank: 100000000002, owner: 강
account cash after transaction: 1659000won

transaction ID: 4
transaction type: account_transfer
card number: 1001
account_transferamount: 10000won
source account -> 대한bank: 100000000001, owner: 정
destination account -> 대한 bank: 100000000002, owner: 강
account cash after transaction: 1647000won
```

한국어를 고르자 오로지 한국어만 나온다.

어떤 언어를 고르시겠습니까? Which language do you prefer?
0. 영어 1. 한국어
1
카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요. 1001
비밀번호를 입력해주세요.
1234
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 1
천원을 몇 장 넣으시겠습니까? 10
오천원을 몇 장 넣으시겠습니까? 10
만원을 몇 장 넣으시겠습니까? 10
오만원을 몇 장 넣으시겠습니까? 10
수표를 몇 장 넣으시겠습니까? 1

1번째 수표의 금액은 얼마입니까? 10000000
10660000원이 입금되었습니다.
잔액은 12307000
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2
자 은행 출금에는 수수료 1000원이 발생합니다.
얼마를 출금하시겠습니까?(수수료 제외) 10000000

출금 가능하신 금액을 초과하셨습니다
얼마를 출금하시겠습니까?(수수료 제외) 10000000

은행에서 101000원이 출금되었고 가져가실 현금은 총 100000원입니다.
천원 0장 / 오천원 0장 / 만원 0장 / 오만원 2장
잔액은 12206000
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3
번호를 고르시오. 1. 현금 이체 2. 계좌 이체: 1
돈을 받고자 하는 계좌의 계좌번호를 입력하시오. : 100000000002
돈을 얼마나 보내겠습니까? 100000
5000원의 수수료가 드니, 수수료를 포함해서 현금을 넣어주세요
10000원 몇 장 넣겠습니까? 5
5000원 몇 장 넣겠습니까? 0
10000원 몇 장 넣겠습니까? 10
50000원 몇 장 넣겠습니까? 0

이체가 완료되었습니다.
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3
번호를 고르시오. 1. 현금 이체 2. 계좌 이체: 2
돈을 받을 계좌의 계좌번호를 입력하세요: 100000000002
본인의 계좌번호를 입력하세요: 100000000001
송금하실 금액을 알려주세요: 10000
10000+2000 원이 계좌에서 출금되었습니다.
계좌이체가 완료되었습니다.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 4
세션을 종료합니다.
거래 ID: 5
거래 유형: 입금
카드 번호: 1001
입금 금액: 10660000원
입금 계좌 -> 대한은행: 100000000001, 소유주: 정
거래 후 금액: 12307000원

거래 ID: 6
거래 유형: 출금
카드 번호: 1001
출금 금액: 100000원
출금 계좌 -> 대한은행: 100000000001, 소유주: 정
거래 후 금액: 12206000원

거래 ID: 7
거래 유형: 현금_계좌이체
카드 번호: 1001
현금_계좌이체금액: 100000원
이체 계좌 -> 대한은행: 100000000001, 소유주: 정
입금 계좌 -> 대한은행: 100000000002, 소유주: 강
거래 후 금액: 12206000원

거래 ID: 8
거래 유형: 계좌_계좌이체
카드 번호: 1001
계좌_계좌이체금액: 10000원
이체 계좌 -> 대한은행: 100000000001, 소유주: 정

입금 계좌 -> 대한은행: 100000000002, 소유주: 강
거래 후 금액: 12194000원

9. Exception Handling

(REQ9.1) 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).

입금, 출금, 계좌 이체에서 유효하지 않은 입력에 대한 처리는 위에서 다루었다. 잘못된 비밀번호, 계좌번호, 동작 번호등이 들어오면 아래와 같이 에러 메시지를 출력하고 이전 메뉴로 되돌아가는 것을 확인할 수 있다.

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 은행 만들기 2. 은행 선택하기 3. 정보 출력하기 : 4
번호를 잘못 입력하셨습니다.
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 은행 만들기 2. 은행 선택하기 3. 정보 출력하기 : 2
<은행 선택하기>
가고자 하는 은행 번호를 입력해주세요. 이전 화면으로 돌아가고자 하면 -1을 입력해주세요.
0. 대한
0
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 은행 계좌 만들기 2. ATM 만들기 3. ATM 이용하기 4. 정보 출력하기 5. 계좌 정보 접근6. 이전 화면으로 돌아가기 : 7
번호를 잘못 입력하셨습니다.
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 은행 계좌 만들기 2. ATM 만들기 3. ATM 이용하기 4. 정보 출력하기 5. 계좌 정보 접근6. 이전 화면으로 돌아가기 : 3
<ATM 선택하기>
선택하고 싶은 atm의 번호를 기입해주세요. 이전으로 가고 싶으시면 -1을 입력해 주세요.
0. atm serial number : 100001
1
잘못된 ATM 번호입니다. 다시 입력해주세요.
<ATM 선택하기>
선택하고 싶은 atm의 번호를 기입해주세요. 이전으로 가고 싶으시면 -1을 입력해 주세요.
0. atm serial number : 100001
0
어떤 언어를 고르시겠습니까? Which language do you prefer?
0. 영어 1. 한국어
2
잘못된 번호를 입력하셨습니다
어떤 언어를 고르시겠습니까? Which language do you prefer?
0. 영어 1. 한국어
0
Please insert the card.(Please enter the card number). If you want to exit, enter -1.10003
Invalid card number.
Please insert the card.(Please enter the card number). If you want to exit, enter -1.1002
Please enter the password.
14231
Wrong Password
Please enter the password.
1234
What kind of work would you like to do? Please enter the number.
1. deposit 2. withdrawl 3. transfer 4. end session: 5

You have selected the wrong menu.
What kind of work would you like to do? Please enter the number.

1. deposit 2. withdrawl 3. transfer 4. end session: 2

There is a fee of 1,000 won for withdrawal.

How much do you want to withdraw (excluding fees) : 50000000

You have exceeded the amount you can withdraw.

How much do you want to withdraw (excluding fees) : 1000

In the bank 2000won was withdrawn, Total cash to bring is 1000Won.

1000 Won 1sheets / 5000Won 0sheets / 10000 Won 0sheets / 50000 Won 0sheets.

The balance is 228000

What kind of work would you like to do? Please enter the number.

1. deposit 2. withdrawl 3. transfer 4. end session: 4

Ends the session.

transaction ID: 9

transcation type: Withdrawl

card number: 1002

Withdrawl amount: 1000won

Withdrawl account -> 대한 bank: 100000000002, owner: 강

account cash after transaction: 228000won

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 은행 만들기 2. 은행 선택하기 3. 정보 출력하기 :

```

<은행 계좌 만들기>
당신의 이름을 입력해주세요 : 정
계좌번호를 입력해주세요 : 1001
계좌번호 양식이 잘못되었습니다. 12자리로 입력해주세요.
<은행 계좌 만들기>
당신의 이름을 입력해주세요 : 100000000003
계좌번호를 입력해주세요 : 1
계좌번호 양식이 잘못되었습니다. 12자리로 입력해주세요.
<은행 계좌 만들기>
당신의 이름을 입력해주세요 : 정
계좌번호를 입력해주세요 : 1000000003
계좌번호 양식이 잘못되었습니다. 12자리로 입력해주세요.
<은행 계좌 만들기>
당신의 이름을 입력해주세요 : 정
계좌번호를 입력해주세요 : 100000000002
이미 해당 계좌번호가 존재합니다!
<은행 계좌 만들기>
당신의 이름을 입력해주세요 : 100000000003
계좌번호를 입력해주세요 : 1
계좌번호 양식이 잘못되었습니다. 12자리로 입력해주세요.
<은행 계좌 만들기>
당신의 이름을 입력해주세요 : 정
계좌번호를 입력해주세요 : 100000000003
계좌번호를 입력해주세요 : 1234
card가 생성되었습니다. 카드의 은행은 대한이며 카드 번호는 1003입니다.
계좌가 개설되었습니다. 계좌의 은행은 대한이며 유저 이름은 정입니다. 계좌번호는 100000000003입니다.
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 은행 계좌 만들기 2. ATM 만들기 3. ATM 이용하기 4. 정보 출력하기 5. 계좌 정보 접근 6. 이전 화면으로 돌아가기 : 2
<ATM 만들기>
ATM의 SERIAL NUMBER를 입력해주세요 101
잘못된 ATM SERIAL NUMBER의 유형입니다. 6자리로 입력해주세요.
ATM의 SERIAL NUMBER를 입력해주세요 100001
이미 존재하는 ATM SERIAL NUMBER입니다. 다시 입력해주세요.
ATM의 SERIAL NUMBER를 입력해주세요 100002
ATM type에 해당하는 번호를 입력해주세요
0. Single Bank ATM 1. Multi-Bank ATM 2
잘못된 번호입니다.
ATM type에 해당하는 번호를 입력해주세요
0. Single Bank ATM 1. Multi-Bank ATM 0

```

10. Display of Account/ATM snapshot

(REQ10.1) 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.

```

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 은행 만들기 2. 은행 선택하기 3. 정보 출력하기 : 3
[Account 0] Balance:500000
[Account 1] Balance:500000
[Account 2] Balance:500000
[ATM 0] Remaining Cash: 160000(1000*10장 5000*10장 10000*10장 )
[ATM 1] Remaining Cash: 160000(1000*10장 5000*10장 10000*10장 )
[ATM 2] Remaining Cash: 160000(1000*10장 5000*10장 10000*10장 )

```

은행을 선택하기 전 3.정보 출력하기를 누르면 현재 생성된 계좌와 잔액, ATM별로 남아있는 금액을 출력한다.

ATM Project에 사용된 OOP 개념

1. inheritance

```
class atm {
private:
    int serial_number; //atm 일련번호
    int atm_cash[4] = { 0,0,0,0 }; // atm 안에 남아 있는 돈 // 1000,5000,10000,50000
    int card_insert_slot;//현재 꽂힌 card의 번호
    int session = 0; //session
    int number_of_withdrawl = 0; //출금 횟수
    int limit_of_number_of_withdrawl = 3;//출금 횟수 제한
    int get_money = 0;//출금한 돈
    int limit_of_get_money;//출금한 돈의 제한
    int single_or_multi; //single은 0, multi는 1
    int uni_or_bi; //uni는 1, bi는 0
    bank* primary_bank;//atm의 주 은행
    //static-정체론>>시리얼넘버>>bank make atm 구현하기.
    card* admin_card;
    vector<transaction_history*> temp_tr_his;
    vector<transaction_history*> atm_tr_his;
};

public:
    atm(int single_or_multi, int uni_or_bi, int one_thou, int five_thou, int ten_thou, int fifty_thou, bank* pb, int serialnumber); //constructor
    int get_serial_number() { return serial_number; } //serial number 얻어 오는 거
    int get_atm_cash() { //atm에 돈이 얼마 있는지 얻어 오는 함수.
        int cash = 1000 * atm_cash[0] + 5000 * atm_cash[1] + 10000 * atm_cash[2] + 50000 * atm_cash[3];
        return cash;
    }
    int get_1000() { return atm_cash[0]; }
    int get_5000() { return atm_cash[1]; }
    int get_10000() { return atm_cash[2]; }
    int get_50000() { return atm_cash[3]; }
    int get_card_insert_slot() { return card_insert_slot; } //현재 꽂힌 card의 번호를 가져오고 오는 함수
    void set_card_insert_slot(int card_num) { card_insert_slot = card_num; }
    void increase_session() { session += 1; } //세션을 1 추가하는 함수.
    void decrease_session() { session -= 1; } //세션을 1 감소하는 함수.
    void increase_number_of_withdrawl() { number_of_withdrawl += 1; } //출금 횟수를 1 증가하는 함수.
    void zero_number_of_withdrawl() { number_of_withdrawl = 0; } //출금 횟수를 0으로 초기화하는 함수.
    void increase_get_money(int money) { get_money += money; } //출금한 돈을 증가하는 함수
    bank* find_cardnumber_bank(int cardnumber); //카드 번호로 은행 찾기
    bool valid_cardnumber_account(int cardnumber); //카드번호에 해당하는 계좌가 있는지 없는지
    bank* find_bank_by_account(long long int accountnumber); //계좌번호로 은행찾기
    bool valid_find_bank_by_account(long long int accountnumber); //계좌번호에 해당하는 계좌가 있는지 없는지
    virtual void transfer(); //계좌이체
    virtual void deposit(); //입금
    virtual void withdrawl(); //출금
    virtual void koreaninterface(); //인터페이스(한국어 버전)
    virtual void interface(); //인티페이스
    virtual void koreantransfer(); //계좌이체
    virtual void koreandeposit(); //입금
    virtual void koreanwithdrawl(); //출금
    //virtual void koreaninterface(); //인티페이스
    bool check_password(int cardnumber, int password);
    void print_session_history(); //강승수 세션 총료 main 추가
    void print_atm_history();
    virtual void admin_interface();
    virtual void admin_interface_korean();
    virtual void mini_interface();
    bank* get_primary_bank() { return primary_bank; }
    void change_atm_cash(int a, int b, int c, int d) {
        atm_cash[0] += a;
        atm_cash[1] += b;
        atm_cash[2] += c;
        atm_cash[3] += d;
    }
}
```

```

    vector <transaction_history> get_temp() {
        return temp_tr_his;
    }
    vector <transaction_history> get_atm_tr() {
        return atm_tr_his;
    }
    void change_temp(transaction_history* tr) {
        temp_tr_his.push_back(tr);
    }
    void change_atm_tr(transaction_history* tr) {
        atm_tr_his.push_back(tr);
    }
    void clear_temp() {
        temp_tr_his.clear();
    }
    void change_transfer_temp(transfer_acc_transaction_history* tr) {
        temp_tr_his.push_back(tr);
    }
    void set_number_of_withdrawl(int number) {
        number_of_withdrawl = number;
    }
    int get_number_of_withdrawl() { return number_of_withdrawl; }
    int get_limit_of_number_of_withdrawl() { return limit_of_number_of_withdrawl; }
    void korean_print_atm_history();
};


```

사진_ATM class 코드 부분

```

class single_uni_atm : public atm {
private:
public:
    single_uni_atm(int type, int language_type, int one_thou, int five_thou, int ten_thou, int fifty_thou, bank* pb, int serialnumber) :atm(type, language_type, one_thou, five_thou, ten_thou, fifty_thou, pb, serialnumber) {}
    void transfer(); //계좌이체
    void deposit(); // 입금
    void withdrawl(); //출금
    void interface(); //인터넷뱅킹
};

class multi_uni_atm : public atm {
private:
public:
    multi_uni_atm(int type, int language_type, int one_thou, int five_thou, int ten_thou, int fifty_thou, bank* pb, int serialnumber) :atm(type, language_type, one_thou, five_thou, ten_thou, fifty_thou, pb, serialnumber) {}
    void transfer(); //계좌이체
    void deposit(); // 입금
    void withdrawl(); //출금
    void interface(); //인터넷뱅킹
};

class single_bi_atm : public atm {
private:
public:
    single_bi_atm(int type, int language_type, int one_thou, int five_thou, int ten_thou, int fifty_thou, bank* pb, int serialnumber) :atm(type, language_type, one_thou, five_thou, ten_thou, fifty_thou, pb, serialnumber) {}
    void transfer(); //계좌이체
    void deposit(); // 입금
    void withdrawl(); //출금
    void interface(); //인터넷뱅킹
    void koreantransfer(); //계좌이체
    void koreandeposit(); // 입금
    void koreanwithdrawl(); //출금
    void koreaninterface(); //인터넷뱅킹
    void koreanprint_session_history();
};

class multi_bi_atm : public atm {
private:
public:
    multi_bi_atm(int type, int language_type, int one_thou, int five_thou, int ten_thou, int fifty_thou, bank* pb, int serialnumber) :atm(type, language_type, one_thou, five_thou, ten_thou, fifty_thou, pb, serialnumber) {}
    void transfer(); //계좌이체
    void deposit(); // 입금
    void withdrawl(); //출금
    void interface(); //인터넷뱅킹
    void koreantransfer(); //계좌이체
    void koreandeposit(); // 입금
    void koreanwithdrawl(); //출금
    void koreaninterface(); //인터넷뱅킹
    void koreanprint_session_history();
};

```

사진_single_uni_ATM, single_bi_ATM, multi_uni_ATM, multi_bi_ATM class 캡쳐 사진

```

class transaction_history {
protected:
    int transaction_ID;
    static int transaction_ID_counter;
    int card_number;
    string tr_type; //출금, 입금, 계좌이체
    int amount; //양
    bank* tr_bank;//현재 작업한 은행
    account* tr_account; // 작업한 계좌
    int curr_acc_money;
public:
    transaction_history(int cardnum, string type, int am, bank* bank, account* account); //생성자
    virtual void print_history(); //print 하는 함수.
    virtual void korean_print_history();
    virtual string print_txt();
    virtual string korean_print_txt();
};

int transaction_history::transaction_ID_counter = 1;

```

사진_Base class인 transaction history를 캡처한 사진이다.

```
class transfer_acc_transaction_history : public transaction_history {  
private:  
    bank* end_bank;  
    account* end_account;  
public:  
    transfer_acc_transaction_history(int cardnum, string type, int am, bank* start_bank, bank* end_bank, account* start_account, account* end_account);  
    void print_history();  
    void korean_print_history();  
    string print_txt();  
    string korean_print_txt();  
};
```

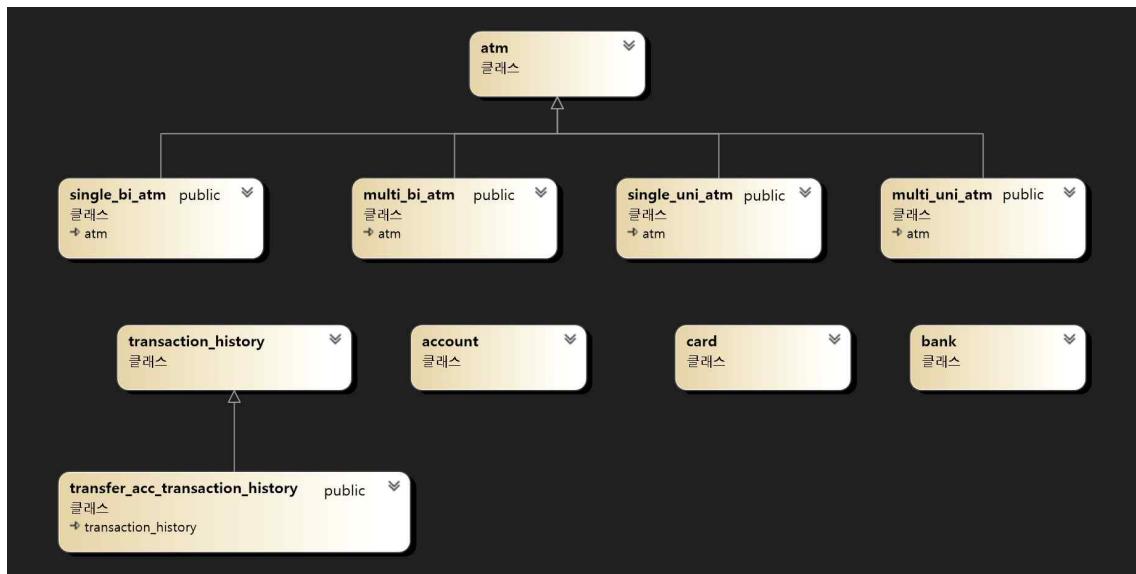
사진_child class인 transfer_acc_transaction_history를 캡처한 사진이다.

Inheritance 개념을 ATM class와 Transaction_history class에 적용하였다.

사진 1, 2, 3은 Base class인 ATM class이고 사진4는 child class인 single_uni_ATM, single_bi_ATM, multi_uni_ATM, multi_bi_ATM의 class이다. 4개의 ATM type이 공통적으로 가지는 property와 method를 ATM class에 넣어서 만들어 주었다. 그리고 모두 ATM class를 상속받아서 구현이 달라져야 하는 method와 새롭게 추가되어야 할 method를 function overloading을 하여 구체화 시켜주었다.

사진 5는 Base class인 transaction_history class이고 사진 6은 child class인 transfer_acc_transaction_history class이다. 4가지 거래 종류인 deposit, withdraw, cash_transfer, account_transfer에 대해 deposit과 withdraw, cash_transfer는 계좌의 종류가 1개만 기록되면 되므로 형태가 비슷하지만 계좌끼리의 이체는 돈을 받는 계좌와 돈을 보내는 계좌 두 종류의 계좌가 기록되어야 하므로 형태가 달라져야 한다. 이 또한 inheritance의 개념을 사용하여 동일한 property와 method들은 transaction_history class에 넣어주고, 계좌끼리의 계좌이체에 필요한 추가적인 정보들은 transfer_acc_transaction_history class에 function overloading을 이용해 구현해 주었다.

2.abstraction



사진_ATM Class diagram

ATM(single_bi, multi_bi, single_uni, multi_uni)와 transaction

history(transfer_transaction_history), account, card, bank에 대해 abstraction 개념을 적용하여 class로 표현해 주었다. 그리고 실제 객체를 사용하고 싶을 때는 해당 클래스를 이용해 인스턴스화를 해서 사용하였다.

3.encapsulation

```
class account { //계좌
private:
    bank* bank_adress; //은행 주소
    string user_name; // 사람 이름
    long long int acc_number;// 계좌 번호 6자리
    int acc_cash; // 계좌 돈
    int password; // 계좌 비밀번호
    int card_num; // 카드 번호 1001 부터 시작.
    card* account_card; // 계좌와 연결된 카드
    vector<transaction_history*> account_tr_his;//account의 거래기록
public:
    account(long long int acc_n, bank* b_adress, string u_name, int pd, int a_cash); // constructor
    bank* get_bank_adress() { return bank_adress; } // 은행 주소 받아오기
    string get_user_name() { return user_name; } // 사람 이름 받아 오기
    int get_account_password() { return password; } // 계좌 비밀번호 받아오기
    long long int get_acc_number() { return acc_number; } // 계좌 번호 받아오기
    int get_acc_cash() { return acc_cash; } //계좌에 돈이 얼마 있는지 알아오기
    int get_cardnum() { return card_num; } // 카드 번호 받아오기
    void change_acc_cash(int ch_cash) { acc_cash = ch_cash; } // 계좌 안에 있는 돈 갱신하기
    void change_account_tr(transaction_history* tr) {
        account_tr_his.push_back(tr);
    }
    void print_account_history();
    void korean_print_account_history();
};
```

account class 구현 부분

계좌 class에서 사용자 정보, 계좌번호, 비밀번호, 카드 숫자 등은 외부에서 쉽게 접근하거나 수정되면 안 되는 정보이기 때문에 접근 지정자를 private로 선언해주었다. 다른 class 내에서도 class 변수들에 대해 외부에서 접근하면 안 되는 변수들에 대해서는 Private, 상속받은 class에서만 사용할 수 있는 변수는 Protected 그리고 외부에서 접근 가능한 Public으로 접근 지정자를 통해 encapsulation을 구현하였다.

4.polymorphism

```
virtual void transfer(); //계좌이체  
virtual void deposit(); // 입금  
virtual void withdraw(); //출금  
virtual void koreaninterface(); //인터페이스(한국어 버전)  
virtual void interface(); //인터페이스  
virtual void koreantransfer(); //계좌이체  
virtual void koreandeposit(); // 입금  
virtual void koreanwithdrawl(); //출금
```

ATM class의 public method 부분

ATM의 종류에 따라 3가지 거래 종류인 transfer, deposit, withdraw가 다르게 구현되어야 한다. 우선 base class인 ATM에서는 interface, transfer, deposit, withdraw 함수가 multi_bit로 실행되도록 만들었고 virtual로 선언해주었다. Single ATM에는 각 거래에서 interface 함수에서 입력받은 카드가 ATM의 은행과 다를 때 에러가 발생하도록 overriding 하였고 transfer, deposit, withdraw 함수에서도 입력받은 계좌가 ATM의 은행과 다를 경우 에러가 overriding하였다. 또한 bilingual ATM에서는 koreantransfer, koreandeposit, koreanwithdrawl을 사용하여 single인 경우 ATM의 은행과 입력받은 계좌가 다를 때 거래가 진행되지 않도록 함수를 overriding 해주었다.

```
virtual void print_history(); //print 하는 함수.  
virtual void korean_print_history();  
virtual string print_txt();  
virtual string korean_print_txt();
```

transaction_history의 public method 부분

transaction_history 또한 계좌끼리의 이체에서는 정보가 다르게 저장되어야 한다. 우선 Base class인 transaction_histroy의 public 함수 중 print_history, korean_print_history, print_txt, korean_print_txt는 deposit, withdrawl, cash_transfer에서 사용할 수 있게 계좌 정보를 1개만 저장하도록 만든 뒤 virtual로 선언해주었다. 이후 child class인 transfer_acc_transaction_history에서는 돈을 받을 계좌와 돈을 보낼 계좌를 모두 입력받고 출력할 수 있도록 위 4개 함수를 overriding 해주었다.

객체 지향 프로그래밍 term project instruction

<초기 화면>

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 은행 만들기 2. 은행 선택하기 3. 정보 출력하기 :

초기화면이다. 하고 싶은 업무의 번호를 입력한다.

<1. 은행 만들기>

<은행 만들기>

은행 이름을 기입하여 주세요. 이전 화면으로 돌아가고자 하면 back을 입력해주세요 :

만들려는 은행의 이름을 적는다. back을 적으면 이전 화면으로 돌아간다.

은행 이름을 기입하여 주세요. 이전 화면으로 돌아가고자 하면 back을 입력해주세요 : 한국은행
은행이 생성되었습니다. 은행 이름은 한국은행입니다.

은행이 생성된다.

<2. 은행 선택하기>

<은행 선택하기>

가고자 하는 은행 번호를 입력해주세요. 이전 화면으로 돌아가고자 하면 -1을 입력해주세요.

0. 한국은행

0

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 은행 계좌 만들기 2. ATM 만들기 3. ATM 이용하기 4. 정보 출력하기 5. 계좌 정보 접근 6. 이전 화면으로 돌아가기 :

은행 선택하기를 고르면, 해당 은행에서 수행할 수 있는 업무들이 나온다.

<2-1 은행 계좌 만들기>

<은행 계좌 만들기>

당신의 이름을 입력해주세요 : 강승수

계좌번호를 입력해주세요 : 111111111111

계좌에 가능한 돈이 얼마인지를 입력해주세요 : 100000

계좌의 비밀번호는 무엇으로 할지 입력해주세요 : 1234

card가 생성되었습니다. 카드의 은행은 한국은행이며 카드 번호는 1001입니다.

계좌가 개설되었습니다. 계좌의 은행은 한국은행이며 유저 이름은 강승수입니다. 계좌 번호는 111111111111입니다.

이름과 계좌번호(12자리), 가능한 돈, 비밀번호를 입력하면 계좌가 생성된다.

<2-2 ATM 만들기>

<ATM 만들기>

ATM의 SERIAL NUMBER를 입력해주세요 111111

ATM type에 해당하는 번호를 입력해주세요

0. Single Bank ATM 1. Multi-Bank ATM 1

ATM의 언어 지원 여부에 해당하는 번호를 입력해주세요

0. Unilingual 1. Bilingual 1

돈은 얼마나 들어있게 할까요?

천원을 몇 장 넣으시겠습니까? 10

오천원을 몇 장 넣으시겠습니까? 10

만원을 몇 장 넣으시겠습니까? 10

오만원을 몇 장 넣으시겠습니까? 10

admin card가 생성되었습니다. 카드의 은행은 한국은행입니다.

atm 이 생성되었습니다. atm의 serial number은 111111입니다. 현재 돈은 660000보유하고 있습니다.

ATM의 SERIAL NUMBER, TYPE, 언어 지원 여부와 들어있는 돈을 입력하면 ATM이 생성된다.

<2-3 ATM 이용하기>

<ATM 선택하기>

선택하고 싶은 atm의 번호를 기입해주세요. 이전으로 가고 싶으시면 -1을 입력해주세요.

0. atm serial number : 111111

0

어떤 언어를 고르시겠습니까? Which language do you prefer?

0. 영어 1. 한국어

1

카드를 넣어주세요.(카드 번호를 입력해주세요). 종료하고 싶으시면 -1을 입력해주세요. 1001
비밀번호를 입력해주세요.

1234

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.

1. 입금 2. 출금 3. 이체 4. 세션 종료 :

선택하고 싶은 atm의 번호를 기입한다. atm의 종류에 따라 언어 지원 여부를 물기도 하고
안 물기도 한다. 카드 번호와 카드의 비밀번호를 기입한다. 그러면, 어떤 업무를 할지 고르는
창이 뜬다.

<2-3-1 입금>

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 1

천원을 몇 장 넣으시겠습니까? (종료하려면 -1을 입력하세요) 10
오천원을 몇 장 넣으시겠습니까? 10

만원을 몇 장 넣으시겠습니까? 0

오만원을 몇 장 넣으시겠습니까? 0

수표를 몇 장 넣으시겠습니까? 1

1번째 수표의 금액은 얼마입니까? 100000

160000원이 입금되었습니다.

잔액은 320000

천원, 오천원, 만원, 오만원, 수표를 금액에 맞게 입력하면, 입금이 완료된다.

수표를 넣게 되면 금액을 물어본다.

<2-3-2 출금>

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 2

자 은행 출금에는 수수료 1000원이 발생합니다.

얼마를 출금하시겠습니까?(수수료 제외) (종료하려면 -1을 입력하세요) 10000

은행에서 11000원이 출금되었고 가져가실 현금은 총 10000원입니다.

천원 0장 / 오천원 0장 / 만원 1장 / 오만원 0장

잔액은 309000

출금하고 싶은 금액을 입력하면 출금이 완료된다.

<2-3-3 이체>

번호를 고르시오. 1. 현금 이체 2. 계좌 이체 :

현금 이체

번호를 고르시오. 1. 현금 이체 2. 계좌 이체 : 1
돈을 받고자 하는 계좌의 계좌번호를 입력하시오. : 222222222222
돈을 얼마나 보내겠습니까? 10000
5000원의 수수료가 드니, 수수료를 포함해서 현금을 넣어주세요
1000원 몇 장 넣겠습니까? 15
5000원 몇 장 넣겠습니까? 0
10000원 몇 장 넣겠습니까? 0
50000원 몇 장 넣겠습니까? 0
이체가 완료되었습니다.

받고자 하는 계좌의 계좌번호와 금액을 입력한 후, 지폐의 장수를 금액과 수수료에 맞게 입력하면 이체가 완료된다.

계좌 이체

1. 입금 2. 출금 3. 이체 4. 세션 종료 : 3
번호를 고르시오. 1. 현금 이체 2. 계좌 이체 : 2
돈을 받을 계좌의 계좌번호를 입력하세요 : 222222222222
본인의 계좌번호를 입력하세요 : 111111111111
송금하실 금액을 알려주세요 : 10000
10000+2000 원이 계좌에서 출금되었습니다.
계좌이체가 완료되었습니다.

받고자 하는 계좌의 계좌번호를 입력한다. 본인이 해당 은행에 가지고 있는 계좌 중에 돈이 빠지길 원하는 계좌를 입력한다. 송금할 금액을 입력한다. 계좌이체가 완료된다.

<2-3-4> 세션 종료

어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.
1. 입금 2. 출금 3. 이체 4. 세션 종료 : 4
세션을 종료합니다.
거래 ID: 4
거래 유형: 현금_계좌이체
카드 번호: 1001
현금_계좌이체 금액: 10000원
현금_계좌이체 계좌 -> 한국은행은행: 222222222222, 소유주: 윤현서
거래 후 금액: 110000원

거래 ID: 5
거래 유형: 계좌_계좌이체
카드 번호: 1001
계좌_계좌이체 금액: 10000원
이체 계좌 -> 한국은행은행: 111111111111, 소유주: 강승수
입금 계좌 -> 한국은행은행: 222222222222, 소유주: 윤현서
거래 후 금액: 297000원

세션이 종료되면 자신이 한 거래가 나오고, 초기 화면으로 돌아간다.

<2-4. 정보 출력하기>

```
[Account 0] Balance:100000  
[ATM 0] Remaining Cash: 660000(1000*10장 5000*10장 10000*10장 50000*10장 )
```

정보 출력하기 버튼은 지금까지 만든 atm과 계좌의 정보에 대해서 출력된다.

<2-5. 계좌정보 접근>

```
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.  
1. 은행 계좌 만들기 2. ATM 만들기 3. ATM 이용하기 4. 정보 출력하기 5. 계좌 정보 접근6. 이전 화면으로 돌아가기 : 5  
계좌번호를 입력해주세요 : 111111111111  
거래 ID: 1  
거래 유형: 입금  
카드 번호: 1001  
입금 금액: 60000원  
입금 계좌 -> 한국은행은행: 111111111111, 소유주: 강승수  
거래 후 금액: 160000원  
  
거래 ID: 2  
거래 유형: 입금  
카드 번호: 1001  
입금 금액: 160000원  
입금 계좌 -> 한국은행은행: 111111111111, 소유주: 강승수  
거래 후 금액: 320000원  
  
거래 ID: 3  
거래 유형: 출금  
카드 번호: 1001  
출금 금액: 10000원  
출금 계좌 -> 한국은행은행: 111111111111, 소유주: 강승수  
거래 후 금액: 30000원  
  
거래 ID: 5  
거래 유형: 계좌_계좌이체  
카드 번호: 1001  
계좌_계좌이체금액: 10000원  
이체 계좌 -> 한국은행은행: 111111111111, 소유주: 강승수  
입금 계좌 -> 한국은행은행: 222222222222, 소유주: 윤현서  
거래 후 금액: 297000원
```

은행에서 계좌번호를 입력하면, 해당 계좌의 거래 기록과 현재 거래 후 금액이 나온다.

<2-6. 이전 화면으로 돌아가기>

```
어떤 업무를 하시겠습니까? 번호를 기입하여 주세요.  
1. 은행 만들기 2. 은행 선택하기 3. 정보 출력하기 :
```

<3. 정보 출력하기>

2-4의 정보 출력하기와 동일하다.

Final source code

```
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#define _MAX_(x, y) ((x > y) ? x : y) //MAX 함수
#define _MIN_(x, y) ((x < y) ? x : y) //MIN 함수
using namespace std;

class bank;
class atm;
class account;

vector <bank*> banklist; //bank list global variable
vector <account*> accountlist; // account list global variable
vector <atm*> atmlist; //atm list global variable

class card { // card 클래스 // 계좌가 만들어질 때 자동으로 일반용 construct 불리야한다.
    atm이 만들어질때마다 admin 카드번호가 불러져야 되고.>> 강승수 구현함
private:
    bool admin; // admin 인지 아닌지.
    bank* card_bank; // 카드가 어느 bank 인지
    int card_num; // 카드 번호
    atm* mas_atm; // 관리자 역할을 할 수 있는 atm
    static int static_card_number;
public:
    card(bank* b, atm* master_atm); // admin 용 constructor
    card(bank* b); // 일반용 constructor
    bool get_ad() { return admin; } // admin인지 아닌지
    bank* get_bank() { return card_bank; } // bank 주소 받아 오는거
    int get_cardnum() { return card_num; } // card number 받아 오는거
};

int card::static_card_number = 1000;

class bank { // 은행
private:
    string bankname; // 은행 이름
    vector<account*> user_acc; // 계좌들을 담아 놓은 vector
    account* curr_account; // 현재 은행내에서 사용하고 있는 계좌 주소
    account* src_account; // 계좌 이체시 사용할 주소(꼭 이 은행일 필요 없음!)
    account* dest_account; // 돈을 보낼 계좌 주소
```

```

vector <atm*> atm_list; // atm을 담아놓은 리스트
public:
    bank(string b_name); //bank 생성자
    string get_bankname() { return bankname; } //은행 이름 받아오는 거
    vector<account*> get_user_acc(); // 유저 어카운트 벡터
    bool bool_get_account(int card_num);// 카드 번호를 가지고 계좌 찾는 거.
    account* account_get_account(int card_num);// 카드 번호를 가지고 계좌 찾는 거.
    account* account_get_accountnum(long long int account_num);// 계좌 번호로 계좌
    찾는 거.
    bool authorize_bank(int card_num, int password);// card_num과 password가 같은지
    확인하기
    bool ac_enough_money_in_acc_bank(long long int acc_num, int m);// 매개변수로
    받은 계좌번호에 돈이 충분히 있는지 확인하기
    bool cn_enough_money_in_acc_bank(int card_num, int m);// 매개변수로 받은
    카드번호에 돈이 충분히 있는지 확인하기
    void account_transfer_bank(long long int src_acc_num, bank* dest_bank, long long
    int dest_acc_num, int m, int fee);// 계좌에서 계좌로 계좌이체하기
    void cash_transfer_bank(bank* dest_bank, long long int dest_acc_num, int m);// 현금으로
    계좌에 계좌이체하기
    void withdrawl_bank(int cardnum, int m);//출금함수 뱅크용
    void deposit_bank(int cardnum, int m); //카드번호로 접근해서 돈 넣기
    bool bool_accountnumber_in_bank(long long int accountnumber);// 계좌번호가 이
    은행에 존재하는지 bool로 확인
    bank* bank_accountnumber_in_bank(int accountnumber);// 계좌번호가 이 은행에
    존재한다면 은행 주소 반환
    void make_multi_uni_atm(int type, int language_type, int one_thou, int five_thou, int
    ten_thou, int fifty_thou, int serialnumber);
    void make_multi_bi_atm(int type, int language_type, int one_thou, int five_thou, int
    ten_thou, int fifty_thou, int serialnumber);
    void make_single_uni_atm(int type, int language_type, int one_thou, int five_thou,
    int ten_thou, int fifty_thou, int serialnumber);
    void make_single_bi_atm(int type, int language_type, int one_thou, int five_thou, int
    ten_thou, int fifty_thou, int serialnumber);
    void make_account(long long int ac_n, string user_name, int password, int cash);
    vector <atm*> get_atm_list();
};

class transaction_history {
protected:
    int transaction_ID;
    static int transaction_ID_counter;
    int card_number;
    string tr_type; //출금, 입금, 계좌이체
    int amount; //양
    bank* tr_bank;//현재 작업한 은행
    account* tr_account; // 작업한 계좌
    int curr_acc_money;
}

```

```

public:
    transaction_history(int cardnum, string type, int am, bank* bank, account*
account); //생성자
    virtual void print_history(); //print 하는 함수.
    virtual void korean_print_history();
    virtual string print_txt();
    virtual string korean_print_txt();
};

int transaction_history::transaction_ID_counter = 1;

class transfer_acc_transaction_history : public transaction_history {
private:
    bank* end_bank;
    account* end_account;
public:
    transfer_acc_transaction_history(int cardnum, string type, int am, bank* start_bank,
bank* end_bank, account* start_account, account* end_account);
    void print_history();
    void korean_print_history();
    string print_txt();
    string korean_print_txt();
};

class account { //계좌
private:
    bank* bank_adress; //은행 주소
    string user_name; // 사람 이름
    long long int acc_number;// 계좌 번호 6자리
    int acc_cash; // 계좌 돈
    int password; // 계좌 비밀번호
    int card_num; // 카드 번호 1001 부터 시작.
    card* account_card; // 계좌와 연결된 카드
    vector<transaction_history*> account_tr_his://account의 거래기록
public:
    account(long long int acc_n, bank* b_adress, string u_name, int pd, int a_cash); //constructor
    bank* get_bank_adress() { return bank_adress; } // 은행 주소 받아오기
    string get_user_name() { return user_name; } // 사람 이름 받아 오기
    int get_account_password() { return password; } // 계좌 비밀번호 받아오기
    long long int get_acc_number() { return acc_number; } // 계좌 번호 받아오기
    int get_acc_cash() { return acc_cash; } //계좌에 돈이 얼마 있는지 알아오기
    int get_cardnum() { return card_num; } // 카드 번호 받아오기
    void change_acc_cash(int ch_cash) { acc_cash = ch_cash; } // 계좌 안에 있는 돈
    [생신하기]
    void change_account_tr(transaction_history* tr) {
        account_tr_his.push_back(tr);

```

```

    }

void print_account_history();
void korean_print_account_history();
};

class atm {
private:
    int serial_number; //atm 일련번호
    int atm_cash[4] = { 0,0,0,0 }; // atm 안에 남아 있는 돈 // 1000,5000,10000,50000
    int card_insert_slot;//현재 꽂힌 card의 번호
    int session = 0; //session
    int number_of_withdrawl = 0; //출금 횟수
    int limit_of_number_of_withdral = 3;//출금 횟수 제한
    int get_money = 0;//출금한 돈
    int limit_of_get_money;//출금한 돈의 제한
    int single_or_multi; //single은 0, multi는 1
    int uni_or_bi;//uni는 0, bi는 1
    bank* primary_bank;//atm의 주 은행
    //static-정세훈>>시리얼넘버>>bank make atm 구현하기.
    card* admin_card;
    vector<transaction_history*> temp_tr_his;
    vector<transaction_history*> atm_tr_his;

public:
    atm(int single_or_multi, int uni_or_bi, int one_thou, int five_thou, int ten_thou, int
fifty_thou, bank* pb, int serialnumber);//constructor
    int get_serial_number() { return serial_number; } //serial number 얻어 오는 거
    int get_atm_cash() { //atm에 돈이 얼마 있는지 얻어 오는 함수.
        int cash = 1000 * atm_cash[0] + 5000 * atm_cash[1] + 10000 * atm_cash[2]
+ 50000 * atm_cash[3];
        return cash;
    }
    int get_1000() { return atm_cash[0]; }
    int get_5000() { return atm_cash[1]; }
    int get_10000() { return atm_cash[2]; }
    int get_50000() { return atm_cash[3]; }
    int get_card_insert_slot() { return card_insert_slot; } //현재 꽂힌 card의 번호를
    가지지고 오는 함수
    void set_card_insert_slot(int card_num) { card_insert_slot = card_num; }
    void increase_session() { session += 1; } // 세션을 1 추가하는 함수.
    void decrease_session() { session -= 1; } // 세션을 1 감소하는 함수.
    void increase_number_of_withdrawl() { number_of_withdrawl += 1; } //출금 횟수를 1
    증가하는 함수.
    void zero_number_of_withdrawl() { number_of_withdrawl = 0; } // 출금 횟수를 0으로
    초기화하는 함수.
    void increase_get_money(int money) { get_money += money; }//출금한 돈을 증가하는
    함수
}

```

```

bank* find_cardnumber_bank(int cardnumber); //카드 번호로 은행 찾기
bool valid_cardnumber_account(int cardnumber); //카드번호에 해당하는 계좌가 있는지
없는지
    bank* find_bank_by_account(long long int accountnumber); //계좌번호로 은행찾기
    bool valid_find_bank_by_account(long long int accountnumber); //계좌번호에 해당하는
계좌가 있는지 없는지
        virtual void transfer(); //계좌이체
        virtual void deposit(); // 입금
        virtual void withdrawl(); //출금
        virtual void koreaninterface(); //인터페이스(한국어 버전)
        virtual void interface(); //인터페이스
        virtual void koreantransfer(); //계좌이체
        virtual void koreandeposit(); // 입금
        virtual void koreanwithdrawl(); //출금
        //virtual void koreaninterface(); //인터페이스
        bool check_password(int cardnumber, int password);
        void print_session_history(); //강승수 세션 종료 main 추가
        void print_atm_history();
        virtual void admin_interface();
        virtual void admin_interface_korean();
        virtual void mini_interface();
        bank* get_primary_bank() { return primary_bank; }
        void change_atm_cash(int a, int b, int c, int d) {
            atm_cash[0] += a;
            atm_cash[1] += b;
            atm_cash[2] += c;
            atm_cash[3] += d;
        }
        vector <transaction_history*> get_temp() {
            return temp_tr_his;
        }
        vector <transaction_history*> get_atm_tr() {
            return atm_tr_his;
        }
        void change_temp(transaction_history* tr) {
            temp_tr_his.push_back(tr);
        }
        void change_atm_tr(transaction_history* tr) {
            atm_tr_his.push_back(tr);
        }
        void clear_temp() {
            temp_tr_his.clear();
        }
        void change_transfer_temp(transfer_acc_transaction_history* tr) {
            temp_tr_his.push_back(tr);
        }
    }
}

```

```

        void set_number_of_withdrawl(int number) {
            number_of_withdrawl = number;
        }
        int get_number_of_withdrawl() { return number_of_withdrawl; }
        int get_limit_of_number_of_withdrawl() { return limit_of_number_of_withdrawl; }
        void korean_print_atm_history();
    };

    class single_uni_atm : public atm {
    private:
    public:
        single_uni_atm(int type, int language_type, int one_thou, int five_thou, int ten_thou,
int fifty_thou, bank* pb, int serialnumber) :atm(type, language_type, one_thou, five_thou,
ten_thou, fifty_thou, pb, serialnumber) {};
        void transfer(); //계좌이체
        void deposit(); // 입금
        void withdrawl();//출금
        void interface();//인터페이스
    };
    class multi_uni_atm : public atm {
    private:
    public:
        multi_uni_atm(int type, int language_type, int one_thou, int five_thou, int ten_thou,
int fifty_thou, bank* pb, int serialnumber) :atm(type, language_type, one_thou, five_thou,
ten_thou, fifty_thou, pb, serialnumber) {};
        void transfer(); //계좌이체
        void deposit(); // 입금
        void withdrawl();//출금
        void interface();//인터페이스
    };
    class single_bi_atm : public atm {
    private:
    public:
        single_bi_atm(int type, int language_type, int one_thou, int five_thou, int ten_thou,
int fifty_thou, bank* pb, int serialnumber) :atm(type, language_type, one_thou, five_thou,
ten_thou, fifty_thou, pb, serialnumber) {};
        void transfer(); //계좌이체
        void deposit(); // 입금
        void withdrawl();//출금
        void interface();//인터페이스
        void koreantransfer(); //계좌이체
        void koreandeposit(); // 입금
        void koreanwithdrawl();//출금
        void koreaninterface();//인터페이스
        void korean_print_session_history();
    };

```

```

class multi_bi_atm : public atm {
private:
public:
    multi_bi_atm(int type, int language_type, int one_thou, int five_thou, int ten_thou,
int fifty_thou, bank* pb, int serialnumber) :atm(type, language_type, one_thou, five_thou,
ten_thou, fifty_thou, pb, serialnumber) {};
    void transfer(); //계좌이체
    void deposit(); // 입금
    void withdrawl(); //출금
    void interface(); //인터페이스
    void koreantransfer(); //계좌이체
    void koreandeposit(); // 입금
    void koreanwithdrawl(); //출금
    void koreaninterface(); //인터페이스
    void korean_print_session_history();

};

//카드 부분
card::card(bank* b, atm* master_atm) { //admin용 constructor
    admin = true; //admin이다
    card_bank = b; //bank 어딘지
    card_num = 7777777; // admin의 card_num은 7777777이다.
    mas_atm = master_atm; //atm 마다 마스터 카드
    std::cout << "admin card가 생성되었습니다. 카드의 은행은" << b->get_bankname() <<
"입니다." << endl;
}
card::card(bank* b) //일반용 constructor
{
    static_card_number++;
    admin = false; //admin 이] 아니다.
    card_bank = b; // bank 주소
    card_num = static_card_number; // cardnumber
    std::cout << "card가 생성되었습니다. 카드의 은행은" << b->get_bankname() << "이며 "
<< "카드 번호는" << card_num << "입니다." << endl;
}

//은행 부분

bank::bank(string b_name) { // 은행 생성자
    bankname = b_name;
    banklist.push_back(this);
    std::cout << "은행이 생성되었습니다. 은행 이름은" << bankname << "입니다." << endl;
}

vector<account*> bank::get_user_acc() { return user_acc; }

bool bank::bool_get_account(int card_num) { // 카드 번호를 가지고 계좌 찾는 거 //참 거짓

```

```

        for (int i = 0; i < user_acc.size(); i++) { // user acc 내를 돈다.
            if (user_acc[i]->get_cardnum() == card_num) { //같은 카드 번호를 찾았다.
                curr_account = user_acc[i]; //현재 계좌번호에 넣음
                return true; //return true 해준다.
            }
        }
        return false;
    }

account* bank::account_get_account(int card_num) { // 카드 번호를 가지고 계좌 찾는 거.
    int count = 0;
    for (int i = 0; i < user_acc.size(); i++)
    { // user acc 내를 돈다.
        count += 1;
        if (user_acc[i]->get_cardnum() == card_num)
        { //같은 카드 번호를 찾았다.
            account* ac = user_acc[i]; //현재 계좌번호에 넣음
            return ac;// 현재 계좌 인스턴스를 반환
        }
    }
}

account* bank::account_get_accountnum(long long int account_num) { // 계좌 번호로 계좌 찾는 거.
    int count = 0;
    for (int i = 0; i < user_acc.size(); i++) {
        count += 1;
        if (user_acc[i]->get_acc_number() == account_num) { // 같은 계좌 번호를 찾았다.
            curr_account = user_acc[i]; //현재 계좌번호에 넣음
            return curr_account; // 현재 계좌 인스턴스를 반환
        }
    }
}

bool bank::authorize_bank(int card_num, int password) { // card_num과 password가 같은지 확인하기
    if (this->account_get_account(card_num)->get_account_password() == password) // card_num의 계좌의 password 받아와서 받은 password랑 비교하기
        return true; // 같으면 true
    else return false; // 다르면 false
}

bool bank::ac_enough_money_in_acc_bank(long long int acc_num, int m) { // 매개변수로 받은 계좌번호에 돈이 충분히 있는지 확인하기
    account* acc = account_get_accountnum(acc_num); // 계좌번호로 계좌 인스턴스 찾기
    if (acc->get_acc_cash() < m) { return false; } // 계좌의 잔여금액이 m보다 작다면 false를 반환
    else { return true; } // 아니라면 true를 반환
}

```

```

bool bank::cn_enough_money_in_acc_bank(int card_num, int m) { // 매개변수로 받은
    카드번호에 돈이 충분히 있는지 확인하기
        account* acc = account_get_account(card_num); // 카드번호로 계좌 인스턴스 찾기
        if (acc->get_acc_cash() < m) { return false; } // 계좌의 잔여금액이 m보다
    작다면 false를 반환
        else { return true; } // 아니라면 true를 반환
    }

void bank::account_transfer_bank(long long int src_acc_num, bank* dest_bank, long long int
dest_acc_num, int m, int fee) { // 계좌에서 계좌로 계좌이체하기
    account* src_account = account_get_accountnum(src_acc_num); // 돈을 보낼 계좌를
src_account에 저장함
    src_account->change_acc_cash(src_account->get_acc_cash() - m - fee); // src_account 계좌에서 m원을 뺌
    dest_account = dest_bank->account_get_accountnum(dest_acc_num); // 돈을 받는
계좌를 dest_account에 저장함
    dest_account->change_acc_cash(dest_account->get_acc_cash() + m); // dest_account 계좌에서 m원을 더함
}

void bank::cash_transfer_bank(bank* dest_bank, long long int dest_acc_num, int m) { // 현금으로 계좌에 계좌이체하기
    dest_account = dest_bank->account_get_accountnum(dest_acc_num); // 돈을 받는
계좌를 dest_account에 저장함
    dest_account->change_acc_cash(dest_account->get_acc_cash() + m); // dest_account 계좌에서 m원을 더함
}

void bank::withdrawl_bank(int cardnum, int m) { //출금함수 뱅크용
    account* acc = account_get_account(cardnum); //카드넘으로 어카운트
    acc->change_acc_cash(acc->get_acc_cash() - m);
}

void bank::deposit_bank(int cardnum, int m) { //카드번호로 접근해서 돈 넣기
    account* acc = account_get_account(cardnum);
    acc->change_acc_cash(acc->get_acc_cash() + m);
}

bool bank::bool_accountnumber_in_bank(long long int accountnumber) { // 계좌번호가 이
은행에 존재하는지 bool로 확인
    int number_of_account = user_acc.size();
    for (int i = 0; i < number_of_account; i++) {
        if (accountnumber == user_acc[i]->get_acc_number()) { return true; }
    }
    return false;
}

bank* bank::bank_accountnumber_in_bank(int accountnumber) { // 계좌번호가 이 은행에
존재한다면 은행 주소 반환
    int number_of_account = user_acc.size();
    for (int i = 0; i < number_of_account; i++) {
        if (accountnumber == user_acc[i]->get_acc_number()) { return this; }
    }
}

```

```

        }
    }

void bank::make_multi_uni_atm(int type, int language_type, int one_thou, int five_thou, int
ten_thou, int fifty_thou, int serial_number) {
    atm_list.push_back(new multi_uni_atm(type, language_type, one_thou, five_thou,
ten_thou, fifty_thou, this, serial_number));
}

void bank::make_multi_bi_atm(int type, int language_type, int one_thou, int five_thou, int
ten_thou, int fifty_thou, int serial_number) {
    atm_list.push_back(new multi_bi_atm(type, language_type, one_thou, five_thou,
ten_thou, fifty_thou, this, serial_number));
}

void bank::make_single_uni_atm(int type, int language_type, int one_thou, int five_thou, int
ten_thou, int fifty_thou, int serial_number) {
    atm_list.push_back(new single_uni_atm(type, language_type, one_thou, five_thou,
ten_thou, fifty_thou, this, serial_number));
}

void bank::make_single_bi_atm(int type, int language_type, int one_thou, int five_thou, int
ten_thou, int fifty_thou, int serial_number) {
    atm_list.push_back(new single_bi_atm(type, language_type, one_thou, five_thou,
ten_thou, fifty_thou, this, serial_number));
}

void bank::make_account(long long int ac_n, string user_name, int password, int cash) {
    user_acc.push_back(new account(ac_n, this, user_name, password, cash));
}

vector <atm*> bank::get_atm_list() { return atm_list; }

//transaction_history
transaction_history::transaction_history(int cardnum, string type, int am, bank* bank,
account* account) {
    transaction_ID = transaction_ID_counter++;
    card_number = cardnum;
    tr_type = type;
    amount = am;
    tr_bank = bank;
    tr_account = account;
    curr_acc_money = tr_account->get_acc_cash();
}

void transaction_history::print_history() {
    std::cout << "transaction ID: " << transaction_ID << endl;
    std::cout << "transcation type: " << tr_type << endl;
    std::cout << "card number: " << card_number << endl;
}

```

```

        std::cout << tr_type << " amount: " << amount << "won" << endl;
        std::cout << tr_type << " account -> " << tr_bank->get_bankname() << " bank: " <<
tr_account->get_acc_number() << ", owner: " << tr_account->get_user_name() << endl;
        std::cout << "account cash after transaction: " << curr_acc_money << "won" << endl
<< endl;
    }
void transaction_history::korean_print_history() {
    std::cout << "거래 ID: " << transaction_ID << endl;
    std::cout << "거래 유형: " << tr_type << endl;
    std::cout << "카드 번호: " << card_number << endl;
    std::cout << tr_type << " 금액: " << amount << "원" << endl;
    std::cout << tr_type << " 계좌 -> " << tr_bank->get_bankname() << "은행: " <<
tr_account->get_acc_number() << ", 소유주: " << tr_account->get_user_name() << endl;
    std::cout << "거래 후 금액: " << curr_acc_money << "원" << endl << endl;
}
string transaction_history::print_txt() {
    string t = "transaction ID: " + to_string(transaction_ID) + '\n'
        + "transcation type: " + tr_type + '\n'
        + "card number: " + to_string(card_number) + '\n'
        + tr_type + " amount: " + to_string(amount) + "won" + '\n'
        + tr_type + " account -> " + tr_bank->get_bankname() + " bank: " +
to_string(tr_account->get_acc_number()) + ", owner: " + tr_account->get_user_name() + '\n'
        + "account cash after transaction: " + to_string(curr_acc_money) + "won" +
'\n' + '\n';
    return t;
}
string transaction_history::korean_print_txt() {
    string t = "거래 ID: " + to_string(transaction_ID) + '\n'
        + "거래 유형: " + tr_type + '\n'
        + "카드 번호: " + to_string(card_number) + '\n'
        + tr_type + " 금액: " + to_string(amount) + "원" + '\n'
        + tr_type + " 계좌 -> " + tr_bank->get_bankname() + "은행: " +
to_string(tr_account->get_acc_number()) + ", 소유주: " + tr_account->get_user_name() + '\n'
        + "거래 후 금액: " + to_string(curr_acc_money) + "원" + '\n' + '\n';
    return t;
}

transfer_acc_transaction_history::transfer_acc_transaction_history(int cardnum, string type,
int am, bank* start_bank, bank* eb, account* start_account, account* ea) :
transaction_history(cardnum, type, am, start_bank, start_account) {
    end_bank = eb;
    end_account = ea;
}
void transfer_acc_transaction_history::print_history() {
    std::cout << "transaction ID: " << transaction_ID << endl;
    std::cout << "transcation type: " << tr_type << endl;

```

```

        std::cout << "card number: " << card_number << endl;
        std::cout << tr_type << "amount: " << amount << "won" << endl;
        std::cout << "source account -> " << tr_bank->get_bankname() << "bank: " <<
tr_account->get_acc_number() << ", owner: " << tr_account->get_user_name() << endl;
        std::cout << "destination account -> " << end_bank->get_bankname() << " bank: " <<
end_account->get_acc_number() << ", owner: " << end_account->get_user_name() << endl;
        std::cout << "account cash after transaction: " << curr_acc_money << "won" << endl
<< endl;
    }
void transfer_acc_transaction_history::korean_print_history() {
    std::cout << "거래 ID: " << transaction_ID << endl;
    std::cout << "거래 유형: " << tr_type << endl;
    std::cout << "카드 번호: " << card_number << endl;
    std::cout << tr_type << "금액: " << amount << "원" << endl;
    std::cout << "이체 계좌 -> " << tr_bank->get_bankname() << "은행: " <<
tr_account->get_acc_number() << ", 소유주: " << tr_account->get_user_name() << endl;
    std::cout << "입금 계좌 -> " << end_bank->get_bankname() << "은행: " <<
end_account->get_acc_number() << ", 소유주: " << end_account->get_user_name() << endl;
    std::cout << "거래 후 금액: " << curr_acc_money << "원" << endl << endl;
}
string transfer_acc_transaction_history::print_txt() {
    string t = "transaction ID: " + to_string(transaction_ID) + '\n'
        + "transcation type: " + tr_type + '\n'
        + "card number: " + to_string(card_number) + '\n'
        + tr_type + " amount: " + to_string(amount) + "won" + '\n'
        + "source account -> " + tr_bank->get_bankname() + "bank: " +
to_string(tr_account->get_acc_number()) + ", owner: " + tr_account->get_user_name() + '\n'
        + "destination account -> " + end_bank->get_bankname() + " bank: " +
to_string(end_account->get_acc_number()) + ", owner: " + end_account->get_user_name() +
'\n'
        + "account cash after transaction: " + to_string(curr_acc_money) + "won" +
'\n' + '\n';
    return t;
}
string transfer_acc_transaction_history::korean_print_txt() {
    string t = "거래 ID: " + to_string(transaction_ID) + '\n'
        + "거래 유형: " + tr_type + '\n'
        + "카드 번호: " + to_string(card_number) + '\n'
        + tr_type + " 금액: " + to_string(amount) + "원" + '\n'
        + "이체 계좌 -> " + tr_bank->get_bankname() + "은행: " +
to_string(tr_account->get_acc_number()) + ", 소유주: " + tr_account->get_user_name() + '\n'
        + "입금 계좌 -> " + end_bank->get_bankname() + "은행: " +
to_string(end_account->get_acc_number()) + ", 소유주: " + end_account->get_user_name() +
'\n'
        + "거래 후 금액: " + to_string(curr_acc_money) + "원" + '\n' + '\n';
    return t;
}

```

```

}

//계좌부분
account::account(long long int acc_n, bank* b_adress, string u_name, int pd, int a_cash) {
    bank_adress = b_adress; //bank 주소
    user_name = u_name; // 유저 이름
    acc_number = acc_n; // 계좌 번호
    acc_cash = a_cash; // 계좌 돈
    password = pd; // 비밀번호
    account_card = new card(this->get_bank_adress());
    card_num = account_card->get_cardnum();
    accountlist.push_back(this);
    std::cout << "계좌가 개설되었습니다. 계좌의 은행은 " << bank_adress->get_bankname()
<< "이며 " << "유저 이름은 " << user_name << "입니다. 계좌번호는 " << acc_number << "입니다."
<< endl;
}
void account::print_account_history() {
    for (int i = 0; i < account_tr_his.size(); i++) {
        account_tr_his[i]->print_history();
    }
}
void account::korean_print_account_history() {
    for (int i = 0; i < account_tr_his.size(); i++) {
        account_tr_his[i]->korean_print_history();
    }
}
atm::atm(int type, int language_type, int one_thou, int five_thou, int ten_thou, int fifty_thou,
bank* pb, int serialnumber) { //constructor
    single_or_multi = type;
    uni_or_bi = language_type;
    serial_number = serialnumber;
    atm_cash[0] = one_thou;
    atm_cash[1] = five_thou;
    atm_cash[2] = ten_thou;
    atm_cash[3] = fifty_thou;
    primary_bank = pb;
    admin_card = new card(this->primary_bank, this);
    atmlist.push_back(this);

    std::cout << "atm 이 생성되었습니다. atm의 serial number은 " << serial_number <<
    "입니다." << "현재 돈은 " << get_atm_cash() << "보유하고 있습니다." << endl;
}
bool atm::valid_cardnumber_account(int cardnumber) { //카드 번호에 해당하는 계좌가 있는지
    없는지
}

```

```

int banknum = banklist.size();
for (int i = 0; i < banknum; i++)
{
    if ((banklist[i]->bool_get_account(cardnumber)) == true)
    {
        return true;
    }
}
return false;
}

bank* atm::find_cardnumber_bank(int cardnumber) { // 카드 번호에 해당하는 은행을 찾기
    int banknum = banklist.size();
    for (int i = 0; i < banknum; i++) {
        if ((banklist[i]->bool_get_account(cardnumber)) == true)
        {
            return banklist[i];
        }
    }
}
bool atm::valid_find_bank_by_account(long long int accountnumber) { //계좌번호가 유효한지를
true false return
    int banknum = banklist.size();
    for (int i = 0; i < banknum; i++) {
        if ((banklist[i]->bool_accountnumber_in_bank(accountnumber)) == true)
        {
            return true;
        }
    }
    return false;
}

bank* atm::find_bank_by_account(long long int accountnumber) { //계좌번호로 은행 찾기
    int banknum = banklist.size();
    for (int i = 0; i < banknum; i++) {
        if ((banklist[i]->bool_accountnumber_in_bank(accountnumber)) == true)
        {
            return banklist[i];
        }
    }
}
bool atm::check_password(int cardnumber, int password) {
    return find_cardnumber_bank(cardnumber)->authorize_bank(cardnumber, password);
}
void atm::print_session_history() {
    for (int i = 0; i < temp_tr_his.size(); i++) {
        temp_tr_his[i]->print_history();
    }
}

```

```

        atm_tr_his.push_back(temp_tr_his[i]);
    }
    temp_tr_his.clear();//정세훈 찾아보고 벡터 초기화
}
void atm::print_atm_history() {
    std::ofstream outFile("atm_history.txt", std::ios::trunc);
    for (int i = 0; i < atm_tr_his.size(); i++) {
        atm_tr_his[i]->print_history();
        outFile.write(atm_tr_his[i]->print_txt().c_str(), atm_tr_his[i]->print_txt().size());
    }
    outFile.close();
}

void atm::transfer() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int money = 0;
    int money_sum = 0;
    int c1;
    int fee;
    bank* curr_bank = find_cardnumber_bank(card_insert_slot);
    long long int dest_account;
    bank* dest_bank;
    long long int src_account;
    bank* src_bank;
    string u_name =
curr_bank->account_get_account(card_insert_slot)->get_user_name();

    while (1) {
        std::cout << "Please choose between 1 Cash transfer or 2 Account transfer
: ";
        std::cin >> c1;
        if (c1 == 1) { //현금 계좌이체
            std::cout << "Please enter the Destination Account : ";
            std::cin >> dest_account;
            dest_bank = find_bank_by_account(dest_account);
            if (!valid_find_bank_by_account(dest_account)) {
                std::cout << "There is no account you entered." << endl;
                return;
            }
            std::cout << "Please enter the cash to send: ";
            std::cin >> money;
            std::cout << "Please enter the amount you wish to send along with
the 5,000 won fee." << endl;

```

```

        std::cout << "How many 1,000 won would you like to put in? ";
        std::cin >> one_thou;
        std::cout << "How many 5,000 won would you like to put in? ";
        std::cin >> five_thou;
        std::cout << "How many 10,000 won would you like to put in? ";
        std::cin >> ten_thou;
        std::cout << "How many 50,000 won would you like to put in?";
        std::cin >> fifty_thou;
        money_sum = one_thou * 1000 + five_thou * 5000 + ten_thou *
10000 + fifty_thou * 50000;
        if (money_sum != money + 5000) {
            std::cout << "The money you requested and the money
you entered are different." << endl;
            return;
        };
        change_atm_cash(one_thou, five_thou, ten_thou, fifty_thou);
        curr_bank->cash_transfer_bank(dest_bank, dest_account, money);
        std::cout << "Cash transfer has been completed." << endl;
        transaction_history* th = new transaction_history(card_insert_slot,
"현금_계좌이체", money, dest_bank, dest_bank->account_get_accountnum(dest_account));
        temp_tr_his.push_back(th);

dest_bank->account_get_accountnum(dest_account)->change_account_tr(th);

        return;
    }

    else if (c1 == 2) { // 계좌끼리 계좌이체
        std::cout << "Please enter the Destination Account : ";
        std::cin >> dest_account;
        dest_bank = find_bank_by_account(dest_account);
        if (!valid_find_bank_by_account(dest_account)) {
            std::cout << "There is no account you entered." << endl;
            return;
        }
        std::cout << "Enter your account number : ";
        std::cin >> src_account;
        src_bank = find_bank_by_account(src_account);
        if (!valid_find_bank_by_account(src_account)) {
            std::cout << "There is no account you entered." << endl;
            return;
        }

        if (u_name !=
src_bank->account_get_accountnum(src_account)->get_user_name() or src_bank !=
curr_bank) {

```

```

        std::cout << "You cannot send money from this account.
Please insert another bank card!" << endl;
        return;
    }

    if (src_bank == primary_bank && dest_bank == primary_bank) {
fee = 2000; }
    else if ((src_bank != primary_bank && dest_bank == primary_bank)
or (src_bank == primary_bank && dest_bank != primary_bank)) { fee = 3000; }
    else if (src_bank != primary_bank && dest_bank != primary_bank)
{ fee = 4000; }

    std::cout << "Please enter the cash to send : ";
    std::cin >> money;
    if (!curr_bank->cn_enough_money_in_acc_bank(card_insert_slot,
money + fee)) {
        std::cout << "There is not enough money in the account.";
        return;
    }
    curr_bank->account_transfer_bank(src_account, dest_bank,
dest_account, money, fee);
    std::cout << money << "+" << fee << " Won was deducted from
your account." << endl;
    cout << "The account transfer has been completed." << endl <<
endl;
    transfer_acc_transaction_history* th = new
transfer_acc_transaction_history(card_insert_slot, "계좌_계좌이체", money, src_bank,
dest_bank, src_bank->account_get_accountnum(src_account),
dest_bank->account_get_accountnum(dest_account));
    temp_tr_his.push_back(th);

src_bank->account_get_accountnum(src_account)->change_account_tr(th);

dest_bank->account_get_accountnum(dest_account)->change_account_tr(th);
    return;
}
else {
    std::cout << "Choose between 1 and 2!!!" << endl;
    return;
}
}

void atm::deposit() {
    int one_thou = 0;
    int five_thou = 0;
}

```

```

int ten_thou = 0;
int fifty_thou = 0;
int check_num = 0;
long long int check_sum = 0;
int withdrawl_count = 0;
bank* b = find_cardnumber_bank(card_insert_slot);
if (b != primary_bank) {
    std::cout << "타 은행 입금에는 수수료 1000원이 발생합니다. 수수료를 포함해서
입금하세요" << endl;
}
while (true) {
    std::cout << "천원을 몇 장 넣으시겠습니까? (종료하려면 -1을 입력하세요)";
    std::cin >> one_thou;
    if (one_thou == -1) {
        return;
    }
    std::cout << "오천원을 몇 장 넣으시겠습니까? ";
    std::cin >> five_thou;
    std::cout << "만원을 몇 장 넣으시겠습니까? ";
    std::cin >> ten_thou;
    std::cout << "오만원을 몇 장 넣으시겠습니까? ";
    std::cin >> fifty_thou;
    if (one_thou + five_thou + ten_thou + fifty_thou > 50) {
        std::cout << "가능한 현금 장 수를 초과하셨습니다." << endl;
    }
    else { break; }
}
while (1) {
    std::cout << "수표를 몇 장 넣으시겠습니까? ";
    std::cin >> check_num;
    std::cout << endl;
    if (check_num > 30) {
        std::cout << "가능한 수표 장 수를 초과하셨습니다." << endl;
    }
    else { break; }
}
while (1) {
    int flag = 0; //flag = 0 일때 정상
    check_sum = 0;
    for (int i = 0; i < check_num; i++) {
        long long int temp = 0;
        std::cout << i + 1 << "번째 수표의 금액은 얼마입니까? ";
        std::cin >> temp;
        if (temp < 100000 || temp > 999999999) { // 100억 이상은 불가
            flag = 1; //flag = 1이면 에러 발생
            break;
        }
        check_sum += temp;
    }
    if (flag == 0) {
        std::cout << "입금 완료되었습니다." << endl;
    }
}

```

```

        }
        else {
            check_sum += temp;
        }
    }
    if (flag == 1) {
        std::cout << "잘못된 수표 금액입니다." << endl;
    }
    else { break; }
}
atm_cash[0] += one_thou;
atm_cash[1] += five_thou;
atm_cash[2] += ten_thou;
atm_cash[3] += fifty_thou;
long long int m = 1000 * one_thou + 5000 * five_thou + 10000 * ten_thou + 50000
* fifty_thou + check_sum;
if (b != primary_bank) { m = m - 1000; }
b->deposit_bank(card_insert_slot, m);
std::cout << m << "원이 입금되었습니다." << endl;
std::cout << "잔액은 " << b->account_get_account(card_insert_slot)->get_acc_cash() <<
endl;
transaction_history* th;
th = new transaction_history(card_insert_slot, "입금", m, b,
b->account_get_account(card_insert_slot));
temp_tr_his.push_back(th);
b->account_get_account(card_insert_slot)->change_account_tr(th);
return;
}

void atm::withdrawl() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int check_num = 0;
    int flag = 0;
    bank* b = find_cardnumber_bank(card_insert_slot);
    if (number_of_withdrawl >= limit_of_number_of_withdral) { //withdrawl 초기화 필요.
atm에 카드를 넣으면 withdrawl을 초기화 해줘야함.
        std::cout << "더 출금하시려면 세션을 초기화 하셔야 합니다." << endl;
        return;
    }
    if (b != primary_bank) {
        std::cout << "타 은행 출금에는 수수료 2000원이 발생합니다." << endl;
        flag = 1;
    }
}

```

```

else {
    std::cout << "자 은행 출금에는 수수료 1000원이 발생합니다." << endl;
}
while (1) {
    std::cout << "얼마를 출금하시겠습니까?(수수료 제외) (종료하려면 -1을
입력하세요):";
    std::cin >> check_num;
    if (check_num == -1) {
        return;
    }
    std::cout << endl;
    if (check_num > 500000) {//출금 가능한 금액을 초과하였는지 조사
        std::cout << "출금 가능하신 금액을 초과하셨습니다" << endl;
    }
    else if (check_num % 1000 != 0) {
        std::cout << "화폐 최소 단위는 1000원 입니다" << endl;
    }
    else if (get_atm_cash() < check_num) {
        std::cout << "ATM에 충분한 현금이 존재하지 않습니다" << endl;
    }
    else if (!(b->cn_enough_money_in_acc_bank(card_insert_slot, (flag ?
(check_num + 2000) : (check_num + 1000))))) {
        std::cout << "계좌에 돈이 충분하지 않습니다" << endl;
    }
    else {
        int m = check_num;
        fifty_thou = _MIN_((m / 50000), atm_cash[3]);
        m = m - 50000 * fifty_thou;
        ten_thou = _MIN_((m / 10000), atm_cash[2]);
        m = m - 10000 * ten_thou;
        five_thou = _MIN_((m / 5000), atm_cash[1]);
        m = m - 5000 * five_thou;
        one_thou = _MIN_((m / 1000), atm_cash[0]);
        m = m - 1000 * one_thou;

        if (m != 0) {
            std::cout << "ATM에 지폐의 개수가 부족합니다" << endl;
        }
        else {
            //atm_cash 조정
            atm_cash[0] -= one_thou;
            atm_cash[1] -= five_thou;
            atm_cash[2] -= ten_thou;
            atm_cash[3] -= fifty_thou;
            b->withdrawl_bank(card_insert_slot, (flag ? (check_num +
2000) : (check_num + 1000)));
        }
    }
}

```

```

        std::cout << "은행에서 " << (flag ? (check_num + 2000) :
(check_num + 1000)) << "원이 출금되었고 가져가실 현금은 총 " << check_num << "원 입니다." <<
endl;

        std::cout << "천원 " << one_thou << "장 / " << "오천원 " <<
five_thou << "장 / " << "만원 " << ten_thou << "장 / " << "오만원 " << fifty_thou << "장" <<
endl;

        std::cout << "잔액은 " <<

b->account_get_account(card_insert_slot)->get_acc_cash() << endl;
        number_of_withdrawl += 1;
        return;
        transaction_history* th;
        th = new transaction_history(card_insert_slot, "출금",
check_num, b, b->account_get_account(card_insert_slot));
        temp_tr_his.push_back(th);

b->account_get_account(card_insert_slot)->change_account_tr(th);
        return;
    }
}
}

void atm::korean_print_atm_history() {
    std::ofstream outFile("atm_history.txt", std::ios::trunc);
    for (int i = 0; i < get_atm_tr().size(); i++) {
        get_atm_tr()[i]->korean_print_history();
        outFile.write(get_atm_tr()[i]->print_txt().c_str(),
get_atm_tr()[i]->print_txt().size());
    }
    outFile.close();
}

void atm::interface() {
    increase_session();
    set_number_of_withdrawl(0);
    int password;
    int work_type;
    int password_count = 0;
    int number;//추가
main1:
    std::cout << "Please insert the card.(Please enter the card number). If you want to
exit, enter -1.";
    std::cin >> number; //변경
    set_card_insert_slot(number);//추가
    if (get_card_insert_slot() == -1) {
        return;

```

```

    }
    if (get_card_insert_slot() == 7777777) {
        admin_interface();
        goto end;
    }
    if (valid_cardnumber_account(get_card_insert_slot()) == 0) {
        cout << "Invalid card number." << endl;
        goto main1;
    }
    while (1) {
        if (password_count == 3) {
            break;
        }
        std::cout << "Please enter the password." << endl;
        std::cin >> password;
        if (check_password(get_card_insert_slot(), password)) {
            break;
        }
        cout << "Wrong Password" << endl;
        password_count++;
    }
    if (password_count == 3) {
        cout << "You got the password wrong 3 times. " << endl;
        goto end;//세션 end
    }
    while (1) {
        std::cout << "What kind of work would you like to do? Please enter the
number." << endl;
        std::cout << "1. deposit " << "2. withdrawl " << "3. transfer " << "4. end
session: ";
        std::cin >> work_type;
        if (work_type == 1) { deposit(); }
        else if (work_type == 2) { withdrawl(); }
        else if (work_type == 3) { transfer(); }
        else if (work_type == 4) { break; }
        else { std::cout << "You have selected the wrong menu." << endl; }
    }
end:
    cout << "Ends the session." << endl;
    print_session_history();
    decrease_session();
}

void atm::koreaninterface() {
    increase_session();
    number_of_withdrawl = 0;
}

```

```

int password;
int work_type;
int password_count = 0;

main1:
    std::cout << "카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을
    입력해주세요.";
    std::cin >> card_insert_slot;
    if (card_insert_slot == -1) {
        return;
    }
    if (card_insert_slot == 7777777) {
        admin_interface_korean();
        goto end;
    }
    if (valid_cardnumber_account(card_insert_slot) == 0) {
        cout << "유효하지 않은 카드 번호입니다. " << endl;
        goto main1;
    }
    while (1) {
        if (password_count == 3) {
            break;
        }
        std::cout << "비밀번호를 입력해주세요." << endl;
        std::cin >> password;
        if (check_password(card_insert_slot, password)) {
            break;
        }
        cout << "잘못된 비밀번호입니다." << endl;
        password_count++;
    }
    if (password_count == 3) {
        cout << "비밀번호를 3번 틀리셨습니다. " << endl;
        goto end;//세션 end
    }
    while (1) {
        std::cout << "어떤 업무를 하시겠습니까? 번호를 기입하여 주세요." << endl;
        std::cout << "1. 입금 " << "2. 출금 " << "3. 이체 " << "4. 세션 종료 : ";
        std::cin >> work_type;
        if (work_type == 1) { deposit(); }
        else if (work_type == 2) { withdrawl(); }
        else if (work_type == 3) { transfer(); }
        else if (work_type == 4) { break; }
        else { std::cout << "잘못된 메뉴를 선택하셨습니다." << endl; }
    }
end:
    cout << "세션을 종료합니다." << endl;

```

```

        print_session_history();
        decrease_session();
    }

void atm::admin_interface() {
    int number;
    cout << "Choose the menu" << endl;
    cout << "Menu" << endl;
    cout << "1. Transcation History" << endl;
menu1:
    cin >> number;
    if (number != 1) {
        cout << "Wrong number. Please enter it one more time." << endl;
        goto menu1;
    }
    else {
        print_atm_history();
        // transaction_history 출력해줄수 있 나요?
        //정세훈 여기다가 fileout
    }
    return;
}

void atm::admin_interface_korean() {
    int number;
    cout << "선택하실 메뉴를 골라주세요." << endl;
    cout << "메뉴" << endl;
    cout << "1. 거래 기록" << endl;
menu1:
    cin >> number;
    if (number != 1) {
        cout << "잘못된 번호를 입력하셨습니다. 다시 입력해주세요." << endl;
        goto menu1;
    }
    else {
        korean_print_atm_history();
        // transaction_history 출력해줄수 있 나요?
        //정세훈 여기다가 fileout
    }
    return;
}

void show_information() {
    int account_number = accountlist.size();
    int atm_number = atmlist.size();
    for (int i = 0; i < account_number; i++) {

```

```

        cout << "[Account " << i << "] Balance:" << accountlist[i]->get_acc_cash() <<
endl;
    }
    for (int i = 0; i < atm_number; i++) {
        cout << "[ATM " << i << "]";
        cout << "Remaining Cash: " << atmlist[i]->get_atm_cash();
        if (atmlist[i]->get_atm_cash() != 0) {
            cout << "(";
        }
        if (atmlist[i]->get_1000() >= 1) {
            cout << "1000*" << atmlist[i]->get_1000() << "장 ";
        }
        if (atmlist[i]->get_5000() >= 1) {
            cout << "5000*" << atmlist[i]->get_5000() << "장 ";
        }
        if (atmlist[i]->get_10000() >= 1) {
            cout << "10000*" << atmlist[i]->get_10000() << "장 ";
        }
        if (atmlist[i]->get_50000() >= 1) {
            cout << "50000*" << atmlist[i]->get_50000() << "장 ";
        }
        if (atmlist[i]->get_atm_cash() != 0) {
            cout << ")";
        }
        cout << endl;
    }
}
void atm::mini_interface() {
    int number;
    if (uni_or_bi == 0) {
        interface();
        return;
    }
    else if (uni_or_bi == 1) {
menu1:
        cout << "어떤 언어를 고르시겠습니까? Which language do you prefer?" <<
endl;
        cout << "0. 영어 \t 1. 한국어" << endl;
        cin >> number;
        if (number == 0) {
            interface();
        }
        else if (number == 1) {
            koreaninterface();
        }
        else {

```

```

        cout << "잘못된 번호를 입력하셨습니다" << endl;
        goto menu1;
    }
}

void atm::koreantransfer() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int money = 0;
    int money_sum = 0;
    int c1;
    int fee;
    bank* curr_bank = find_cardnumber_bank(card_insert_slot);
    long long int dest_account;
    bank* dest_bank;
    long long int src_account;
    bank* src_bank;
    string u_name =
curr_bank->account_get_account(card_insert_slot)->get_user_name();

    while (1) {
        std::cout << "어떤 업무를 하시겠습니까? 번호를 기입해주세요. 1. 현금 이체 2.
계좌 이체 : ";
        std::cin >> c1;
        if (c1 == 1) { //현금 계좌이체
            std::cout << "누구에게 계좌 이체하시겠습니까? 송금 받을 계좌를
입력해주세요. : ";
            std::cin >> dest_account;
            dest_bank = find_bank_by_account(dest_account);
            if (!valid_find_bank_by_account(dest_account)) {
                std::cout << "해당하는 은행계좌는 존재하지 않습니다." <<
endl;
                return;
            }
            std::cout << "얼마를 송금하시겠습니까?: ";
            std::cin >> money;
            std::cout << "5000원의 수수료가 드니, 수수료를 포함해서 현금을
넣어주세요" << endl;
            std::cout << "1000원 짜리 지폐를 몇 장 넣겠습니까? ";
            std::cin >> one_thou;
            std::cout << "5000원 짜리 지폐를 몇 장 넣겠습니까? ";
            std::cin >> five_thou;
            std::cout << "10000원 짜리 지폐를 몇 장 넣겠습니까? ";

```

```

        std::cin >> ten_thou;
        std::cout << "50000원 짜리 지폐를 몇 장 넣겠습니까? ";
        std::cin >> fifty_thou;
        money_sum = one_thou * 1000 + five_thou * 5000 + ten_thou *
10000 + fifty_thou * 50000;
        if (money_sum != money + 5000) {
            std::cout << "당신이 입력한 돈과 넣은 돈이 다릅니다." <<
endl;
            return;
        }
        atm_cash[0] += one_thou;
        atm_cash[1] += five_thou;
        atm_cash[2] += ten_thou;
        atm_cash[3] += fifty_thou;
        curr_bank->cash_transfer_bank(dest_bank, dest_account, money);
        std::cout << "현금 이체가 완료되었습니다." << endl << endl;
        return;
    }

else if (c1 == 2) { // 계좌끼리 계좌이체
    std::cout << "돈을 받을 계좌의 계좌번호를 입력하세요 : ";
    std::cin >> dest_account;
    dest_bank = find_bank_by_account(dest_account);
    if (!valid_find_bank_by_account(dest_account)) {
        std::cout << "해당 계좌는 유효하지 않습니다." << endl; //이게
맞아?
    }
    std::cout << "당신의 계좌번호를 입력해주세요 : ";
    std::cin >> src_account;
    src_bank = find_bank_by_account(src_account);
    if (!valid_find_bank_by_account(src_account)) {
        std::cout << "해당하는 계좌가 없습니다." << endl;
        return;
    }

    if (src_bank == primary_bank && dest_bank == primary_bank) {
fee = 2000; }

    else if ((src_bank != primary_bank && dest_bank == primary_bank)
or (src_bank == primary_bank && dest_bank != primary_bank)) { fee = 3000; }

    else if (src_bank != primary_bank && dest_bank != primary_bank)
{ fee = 4000; }

    if (u_name !=
src_bank->account_get_accountnum(src_account)->get_user_name() or src_bank !=
curr_bank) {

```

```

        std::cout << "해당 계좌에서 돈을 이체할 수 없습니다." <<
endl;
        return;
    }

    std::cout << "얼마나 보낼 지 입력하세요. : ";
    std::cin >> money;
    if (!curr_bank->cn_enough_money_in_acc_bank(card_insert_slot,
money + fee)) {
        std::cout << "계좌에 충분한 돈이 있지 않습니다.";
        return;
    }
    curr_bank->account_transfer_bank(src_account, dest_bank,
dest_account, money, fee);
    std::cout << money << "+" << fee << "원이 계좌에서 빠집니다." <<
endl;
cout << "계좌 이체가 완료되었습니다." << endl << endl;
return;
}
else {
    std::cout << "1번과 2번 사이에 고르시오." << endl;
    return;
}
}
}
}

```

```

void atm::koreandeposit() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int check_num = 0;
    long long int check_sum = 0;
    int withdrawl_count = 0;
    bank* b = find_cardnumber_bank(card_insert_slot);
    if (b != primary_bank) {
        std::cout << "타 은행 입금에는 수수료 1000원이 발생합니다. 수수료를 포함해서
입금하세요" << endl;
    }
    while (true) {
        std::cout << "천원을 몇 장 넣으시겠습니까? (종료하려면 -1을 입력하세요)";
        std::cin >> one_thou;
        if (one_thou == -1) {

```

```

        return;
    }
    std::cout << "오천원을 몇 장 넣으시겠습니까? ";
    std::cin >> five_thou;
    std::cout << "만원을 몇 장 넣으시겠습니까? ";
    std::cin >> ten_thou;
    std::cout << "오만원을 몇 장 넣으시겠습니까? ";
    std::cin >> fifty_thou;
    if (one_thou + five_thou + ten_thou + fifty_thou > 50) {
        std::cout << "가능한 현금 장 수를 초과하셨습니다." << endl;
    }
    else { break; }
}
while (1) {
    std::cout << "수표를 몇 장 넣으시겠습니까? ";
    std::cin >> check_num;
    std::cout << endl;
    if (check_num > 30) {
        std::cout << "가능한 수표 장 수를 초과하셨습니다." << endl;
    }
    else { break; }
}
while (1) {
    int flag = 0; //flag = 0 일때 정상
    for (int i = 0; i < check_num; i++) {
        long long int temp = 0;
        std::cout << i + 1 << "번째 수표의 금액은 얼마입니까? ";
        std::cin >> temp;
        if (temp < 100000 || temp > 9999999999) {
            flag = 1; //flag = 1이면 에러 발생
            break;
        }
        else {
            check_sum += temp;
        }
    }
    if (flag == 1) {
        std::cout << "잘못된 수표 금액입니다." << endl;
    }
    else { break; }
}
atm_cash[0] += one_thou;
atm_cash[1] += five_thou;
atm_cash[2] += ten_thou;
atm_cash[3] += fifty_thou;
long long int m = 1000 * one_thou + 5000 * five_thou + 10000 * ten_thou + 50000

```

```

* fifty_thou + check_sum;
    if (b != primary_bank) { m = m - 1000; }
    b->deposit_bank(card_insert_slot, m);
    std::cout << m << "원이 입금되었습니다." << endl;
    std::cout << "잔액은 " << b->account_get_account(card_insert_slot)->get_acc_cash() <<
endl;
    transaction_history* th;
    th = new transaction_history(card_insert_slot, "입금", m, b,
b->account_get_account(card_insert_slot));
    temp_tr_his.push_back(th);
    b->account_get_account(card_insert_slot)->change_account_tr(th);
    return;
}

void atm::koreanwithdrawl() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int check_num = 0;
    int flag = 0;
    bank* b = find_cardnumber_bank(card_insert_slot);
    if (number_of_withdrawl > limit_of_number_of_withdral) { //withdrawl 초기화 필요.
atm에 카드를 넣으면 withdrawl을 초기화 해줘야함.
        std::cout << "더 출금하시려면 세션을 초기화 하셔야 합니다." << endl;
        return;
    }
    if (b != primary_bank) {
        std::cout << "타 은행 출금에는 수수료 2000원이 발생합니다." << endl;
        flag = 1;
    }
    else {
        std::cout << "자 은행 출금에는 수수료 1000원이 발생합니다." << endl;
    }
    while (1) {
        std::cout << "천원을 몇 장 넣으시겠습니까? (종료하려면 -1을 입력하세요)";
        std::cin >> one_thou;
        if (one_thou == -1) {
            return;
        }
        std::cout << endl;
        if (check_num > 500000) {//출금 가능한 금액을 초과하였는지 조사
            std::cout << "출금 가능하신 금액을 초과하셨습니다" << endl;
        }
        else if (check_num % 1000 != 0) {

```

```

        std::cout << "화폐 최소 단위는 1000원 입니다" << endl;
    }
    else if (get_atm_cash() < check_num) {
        std::cout << "ATM에 충분한 현금이 존재하지 않습니다" << endl;
    }
    else if (!(b->cn_enough_money_in_acc_bank(card_insert_slot, (flag ? (check_num + 2000) : (check_num + 1000))))) {
        std::cout << "계좌에 돈이 충분하지 않습니다" << endl;
    }
    else {
        int m = check_num;
        fifty_thou = _MIN_((m / 50000), atm_cash[3]);
        m = m - 50000 * fifty_thou;
        ten_thou = _MIN_((m / 10000), atm_cash[2]);
        m = m - 10000 * ten_thou;
        five_thou = _MIN_((m / 5000), atm_cash[1]);
        m = m - 5000 * five_thou;
        one_thou = _MIN_((m / 1000), atm_cash[0]);
        m = m - 1000 * one_thou;

        if (m != 0) {
            std::cout << "ATM에 지폐의 개수가 부족합니다" << endl;
        }
        else {
            //atm_cash 조정
            atm_cash[0] -= one_thou;
            atm_cash[1] -= five_thou;
            atm_cash[2] -= ten_thou;
            atm_cash[3] -= fifty_thou;
            b->withdrawl_bank(card_insert_slot, (flag ? (check_num + 2000) : (check_num + 1000)));
            std::cout << "은행에서 " << (flag ? (check_num + 2000) : (check_num + 1000)) << "원이 출금되었고 가져가실 현금은 총 " << check_num << "원입니다." << endl;
            std::cout << "천원 " << one_thou << "장 / " << "오천원 " << five_thou << "장 / " << "만원 " << ten_thou << "장 / " << "오만원 " << fifty_thou << "장" << endl;
            std::cout << "잔액은 " <<
            b->account_get_account(card_insert_slot)->get_acc_cash() << endl;
            number_of_withdrawl += 1;
            transaction_history* th;
            th = new transaction_history(card_insert_slot, "출금",
            check_num, b, b->account_get_account(card_insert_slot));
            temp_tr_his.push_back(th);
        }
    }
    b->account_get_account(card_insert_slot)->change_account_tr(th);
}

```

```

                    return;
                }
            }
        }
    }
//윤현서 시작

// single_uni

void single_uni_atm::transfer() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int money = 0;
    int money_sum = 0;
    int c1;
    int fee;
    bank* curr_bank = find_cardnumber_bank(get_card_insert_slot());
    long long int dest_account;
    bank* dest_bank;
    long long int src_account;
    bank* src_bank;
    string u_name =
curr_bank->account_get_account(get_card_insert_slot())->get_user_name();

    while (1) {
        std::cout << "Please choose between 1 Cash transfer or 2 Account transfer
: ";
        std::cin >> c1;
        if (c1 == 1) {      //현금 계좌이체
            std::cout << "Please enter the Destination Account : ";
            std::cin >> dest_account;
            dest_bank = find_bank_by_account(dest_account);
            if (dest_bank != get_primary_bank()) {
                cout << "This is not primary bank account." << endl;
                return;
            }
            if (!valid_find_bank_by_account(dest_account)) {
                std::cout << "There is no account you entered." << endl;
                return;
            }
            std::cout << "Please enter the cash to send: ";
            std::cin >> money;
            std::cout << "Please enter the amount you wish to send along with
the 5,000 won fee." << endl;

```

```

        std::cout << "How many 1,000 won would you like to put in? ";
        std::cin >> one_thou;
        std::cout << "How many 5,000 won would you like to put in? ";
        std::cin >> five_thou;
        std::cout << "How many 10,000 won would you like to put in? ";
        std::cin >> ten_thou;
        std::cout << "How many 50,000 won would you like to put in?";
        std::cin >> fifty_thou;
        money_sum = one_thou * 1000 + five_thou * 5000 + ten_thou *
10000 + fifty_thou * 50000;
        if (money_sum != money + 5000) {
            std::cout << "The money you requested and the money
you entered are different." << endl;
            return;
        }
        change_atm_cash(one_thou, five_thou, ten_thou, fifty_thou);
        curr_bank->cash_transfer_bank(dest_bank, dest_account, money);
        std::cout << "Cash transfer has been completed." << endl;
        transaction_history* th = new
transaction_history(get_card_insert_slot(), "cash_transfer", money, dest_bank,
dest_bank->account_get_accountnum(dest_account));
        change_temp(th);

dest_bank->account_get_accountnum(dest_account)->change_account_tr(th);
        return;
    }

else if (c1 == 2) {    // 계좌끼리 계좌이체
    std::cout << "Please enter the Destination Account : ";
    std::cin >> dest_account;
    dest_bank = find_bank_by_account(dest_account);
    if (!valid_find_bank_by_account(dest_account)) {
        std::cout << "There is no account you entered." << endl;
        return;
    }
    if (dest_bank != get_primary_bank()) {
        cout << "This is not primary bank account." << endl;
        return;
    }
    std::cout << "Enter your account number : ";
    std::cin >> src_account;
    src_bank = find_bank_by_account(src_account);
    if (!valid_find_bank_by_account(src_account)) {
        std::cout << "There is no account you entered." << endl;
        return;
    }
}

```

```

        if (src_bank != get_primary_bank()) {
            cout << "This is not primary bank account." << endl;
            return;
        }
        if (u_name !=

src_bank->account_get_accountnum(src_account)->get_user_name() or src_bank !=
curr_bank) {
            std::cout << "You cannot send money from this account.
Please insert another bank card!" << endl;
            return;
}

        if (src_bank == get_primary_bank() && dest_bank ==
get_primary_bank()) { fee = 2000; }
        else if ((src_bank != get_primary_bank() && dest_bank ==
get_primary_bank()) or (src_bank == get_primary_bank() && dest_bank !=
get_primary_bank())) { fee = 3000; }
        else if (src_bank != get_primary_bank() && dest_bank !=
get_primary_bank()) { fee = 4000; }

        std::cout << "Please enter the cash to send : ";
        std::cin >> money;
        if
(!curr_bank->cn_enough_money_in_acc_bank(get_card_insert_slot(), money + fee)) {
            std::cout << "There is not enough money in the account.";
            return;
}
        curr_bank->account_transfer_bank(src_account, dest_bank,
dest_account, money, fee);
        std::cout << money << "+" << fee << " Won was deducted from
your account." << endl;
        cout << "The account transfer has been completed." << endl <<
endl;
        transfer_acc_transaction_history* th = new
transfer_acc_transaction_history(get_card_insert_slot(), "account_transfer", money, src_bank,
dest_bank, src_bank->account_get_accountnum(src_account),
dest_bank->account_get_accountnum(dest_account));
        change_transfer_temp(th);

src_bank->account_get_accountnum(src_account)->change_account_tr(th);

dest_bank->account_get_accountnum(dest_account)->change_account_tr(th);
        return;
}
else {
        std::cout << "Choose between 1 and 2!!!" << endl;
}

```

```

                return;
            }
        }
    }

void single_uni_atm::deposit() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int check_num = 0;
    long long int check_sum = 0;
    int withdrawl_count = 0;
    bank* b = find_cardnumber_bank(get_card_insert_slot());
    if (b != get_primary_bank()) {
        std::cout << "A fee of 1,000 won will be charged for deposits to other
banks. Deposit including fees" << endl;
    }
    while (true) {
        std::cout << "How many 1,000 won would you like to put in? (Enter -1 to
exit)";
        std::cin >> one_thou;
        if (one_thou == -1) {
            return;
        }
        std::cout << "How many 5,000 won would you like to put in? ";
        std::cin >> five_thou;
        std::cout << "How many 1,0000 won would you like to put in? ";
        std::cin >> ten_thou;
        std::cout << "How many 5,0000 won would you like to put in? ";
        std::cin >> fifty_thou;
        if (one_thou + five_thou + ten_thou + fifty_thou > 50) {
            std::cout << "You have exceeded the number of cash cards
available." << endl;
        }
        else { break; }
    }
    while (1) {
        std::cout << "How many checks would you like to write? ";
        std::cin >> check_num;
        std::cout << endl;
        if (check_num > 30) {
            std::cout << "You have exceeded the number of checks available."
<< endl;
        }
        else { break; }
    }
}

```

```

    }
    while (1) {
        int flag = 0; //flag = 0 일때 정상
        check_sum = 0;
        for (int i = 0; i < check_num; i++) {
            long long int temp = 0;
            std::cout << i + 1 << "th check, how much? ";
            std::cin >> temp;
            if (temp < 100000 || temp > 999999999) { // 100억 이상은 불가
                flag = 1; //flag = 1이면 에러 발생
                break;
            }
            else {
                check_sum += temp;
            }
        }
        if (flag == 1) {
            std::cout << "The check amount is incorrect." << endl;
        }
        else { break; }
    }
    change_atm_cash(one_thou, five_thou, ten_thou, fifty_thou);
    long long int m = 1000 * one_thou + 5000 * five_thou + 10000 * ten_thou + 50000
* fifty_thou + check_sum;
    if (b != get_primary_bank()) { m = m - 1000; }
    b->deposit_bank(get_card_insert_slot(), m);
    std::cout << m << "Won has been deposited." << endl;
    std::cout << "The balance is " <<
    b->account_get_account(get_card_insert_slot())->get_acc_cash() << endl;
    transaction_history* th;
    th = new transaction_history(get_card_insert_slot(), "deposit", m, b,
b->account_get_account(get_card_insert_slot()));
    change_temp(th);
    b->account_get_account(get_card_insert_slot())->change_account_tr(th);
    return;
}

void single_uni_atm::withdrawl() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int check_num = 0;
    int flag = 0;
    bank* b = find_cardnumber_bank(get_card_insert_slot());
    if (get_number_of_withdrawl() >= get_limit_of_number_of_withdrawl()) {

```

```

        std::cout << "If you want to withdraw more, you must reset your session."
<< endl;
        return;
    }
    if (b != get_primary_bank()) {
        std::cout << "A fee of 2,000 won is charged for withdrawals from other
banks." << endl;
        flag = 1;
    }
    else {
        std::cout << "There is a fee of 1,000 won for withdrawal." << endl;
    }
    while (1) {
        std::cout << "How much do you want to withdraw (excluding fees)(Enter -1
to exit) : ";
        std::cin >> check_num;
        if (check_num == -1) {
            return;
        }
        std::cout << endl;
        if (check_num > 500000) { //출금 가능한 금액을 초과하였는지 조사
            std::cout << "You have exceeded the amount you can withdraw."
        }
        << endl;
    }
    else if (check_num % 1000 != 0) {
        std::cout << "The minimum unit of currency is 1,000 won." <<
endl;
    }
    else if (get_atm_cash() < check_num) {
        std::cout << "There is not enough cash in the ATM." << endl;
    }
    else if (!(b->cn_enough_money_in_acc_bank(get_card_insert_slot(), (flag ?
(check_num + 2000) : (check_num + 1000))))) {
        std::cout << "There is not enough money in the account" << endl;
    }
    else {
        int m = check_num;
        fifty_thou = _MIN_((m / 50000), get_50000());
        m = m - 50000 * fifty_thou;
        ten_thou = _MIN_((m / 10000), get_10000());
        m = m - 10000 * ten_thou;
        five_thou = _MIN_((m / 5000), get_5000());
        m = m - 5000 * five_thou;
        one_thou = _MIN_((m / 1000), get_1000());
        m = m - 1000 * one_thou;
    }
}

```

```

        if (m != 0) {
            std::cout << "There are not enough bills in the ATM" <<
endl;
        }
        else {
            //atm_cash 조정
            change_atm_cash(-1 * (one_thou), -1 * (five_thou), -1 *
(ten_thou), -1 * (fifty_thou));
            b->withdrawl_bank(get_card_insert_slot(), (flag ?
(check_num + 2000) : (check_num + 1000)));
            std::cout << "In the bank " << (flag ? (check_num + 2000)
: (check_num + 1000)) << "won was withdrawn, Total cash to bring is " << check_num <<
"Won." << endl;
            std::cout << "1000 Won " << one_thou << "sheets / " <<
"5000Won " << five_thou << "sheets / " << "10000 Won " << ten_thou << "sheets / " << "50000
Won " << fifty_thou << "sheets." << endl;
            std::cout << "The balance is " <<
b->account_get_account(get_card_insert_slot())->get_acc_cash() << endl;
            set_number_of_withdrawl(get_number_of_withdrawl() + 1);
            transaction_history* th = new
transaction_history(get_card_insert_slot(), "Withdrawl", check_num, b,
b->account_get_account(get_card_insert_slot()));
            change_temp(th);

b->account_get_account(get_card_insert_slot())->change_account_tr(th);
            return;
        }
    }
}

void single_uni_atm::interface() {
    increase_session();
    set_number_of_withdrawl(0);
    int password;
    int work_type;
    int password_count = 0;
    int number;//추가
main1:
    std::cout << "Please insert the card.(Please enter the card number). If you want to
exit, enter -1.";
    std::cin >> number; //변경
    set_card_insert_slot(number);//추가
    if (get_card_insert_slot() == -1) {
        return;
    }
}

```

```

if (get_card_insert_slot() == 7777777) {
    admin_interface();
    goto end;
}
if (find_cardnumber_bank(get_card_insert_slot()) != get_primary_bank()) {
    cout << " This is not primary bank account." << endl;
    return;
}
if (valid_cardnumber_account(get_card_insert_slot()) == 0) {
    cout << "Invalid card number." << endl;
    goto main1;
}
while (1) {
    if (password_count == 3) {
        break;
    }
    std::cout << "Please enter the password." << endl;
    std::cin >> password;
    if (check_password(get_card_insert_slot(), password)) {
        break;
    }
    cout << "Wrong Password" << endl;
    password_count++;
}
if (password_count == 3) {
    cout << "You got the password wrong 3 times. " << endl;
    goto end;//종료 end
}
while (1) {
    std::cout << "What kind of work would you like to do? Please enter the
number." << endl;
    std::cout << "1. deposit " << "2. withdrawl " << "3. transfer " << "4. end
session: ";
    std::cin >> work_type;
    if (work_type == 1) { deposit(); }
    else if (work_type == 2) { withdrawl(); }
    else if (work_type == 3) { transfer(); }
    else if (work_type == 4) { break; }
    else { std::cout << "You have selected the wrong menu." << endl; }
}
end:
cout << "Ends the session." << endl;
print_session_history();
decrease_session();
}

```

```

// multi_uni
void multi_uni_atm::transfer() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int money = 0;
    int money_sum = 0;
    int c1;
    int fee;
    bank* curr_bank = find_cardnumber_bank(get_card_insert_slot());
    long long int dest_account;
    bank* dest_bank;
    long long int src_account;
    bank* src_bank;
    string u_name =
curr_bank->account_get_account(get_card_insert_slot())->get_user_name();

    while (1) {
        std::cout << "Please choose between 1 Cash transfer or 2 Account transfer
: ";
        std::cin >> c1;
        if (c1 == 1) {      //현금 계좌이체
            std::cout << "Please enter the Destination Account : ";
            std::cin >> dest_account;
            dest_bank = find_bank_by_account(dest_account);
            if (!valid_find_bank_by_account(dest_account)) {
                std::cout << "There is no account you entered." << endl;
                return;
            }
            std::cout << "Please enter the cash to send: ";
            std::cin >> money;
            std::cout << "Please enter the amount you wish to send along with
the 5,000 won fee." << endl;
            std::cout << "How many 1,000 won would you like to put in? ";
            std::cin >> one_thou;
            std::cout << "How many 5,000 won would you like to put in? ";
            std::cin >> five_thou;
            std::cout << "How many 10,000 won would you like to put in? ";
            std::cin >> ten_thou;
            std::cout << "How many 50,000 won would you like to put in?";
            std::cin >> fifty_thou;
            money_sum = one_thou * 1000 + five_thou * 5000 + ten_thou *
10000 + fifty_thou * 50000;
        }
    }
}

```

```

        if (money_sum != money + 5000) {
            std::cout << "The money you requested and the money
you entered are different." << endl;
            return;
        }
        change_atm_cash(one_thou, five_thou, ten_thou, fifty_thou);
        curr_bank->cash_transfer_bank(dest_bank, dest_account, money);
        std::cout << "Cash transfer has been completed." << endl;
        transaction_history* th = new
transaction_history(get_card_insert_slot(), "cash_transfer", money, dest_bank,
dest_bank->account_get_accountnum(dest_account));
        change_temp(th);

dest_bank->account_get_accountnum(dest_account)->change_account_tr(th);
        return;
    }

else if (c1 == 2) { // 계좌끼리 계좌이체
    std::cout << "Please enter the Destination Account : ";
    std::cin >> dest_account;
    dest_bank = find_bank_by_account(dest_account);
    if (!valid_find_bank_by_account(dest_account)) {
        std::cout << "There is no account you entered." << endl;
        return;
    }
    std::cout << "Enter your account number : ";
    std::cin >> src_account;
    src_bank = find_bank_by_account(src_account);
    if (!valid_find_bank_by_account(src_account)) {
        std::cout << "There is no account you entered." << endl;
        return;
    }
    if (u_name !=
src_bank->account_get_accountnum(src_account)->get_user_name() or src_bank !=
curr_bank) {
        std::cout << "You cannot send money from this account.
Please insert another bank card!" << endl;
        return;
    }

    if (src_bank == get_primary_bank() && dest_bank ==
get_primary_bank()) { fee = 2000; }
    else if ((src_bank != get_primary_bank() && dest_bank ==
get_primary_bank()) or (src_bank == get_primary_bank() && dest_bank !=
get_primary_bank())) { fee = 3000; }
    else if (src_bank != get_primary_bank() && dest_bank !=

```

```

get_primary_bank()) { fee = 4000; }

        std::cout << "Please enter the cash to send : ";
        std::cin >> money;
        if
(!curr_bank->cn_enough_money_in_acc_bank(get_card_insert_slot(), money + fee)) {
                std::cout << "There is not enough money in the account.";
                return;
}
        curr_bank->account_transfer_bank(src_account, dest_bank,
dest_account, money, fee);
        std::cout << money << "+" << fee << " Won was deducted from
your account." << endl;
        cout << "The account transfer has been completed." << endl <<
endl;
        transfer_acc_transaction_history* th = new
transfer_acc_transaction_history(get_card_insert_slot(), "account_transfer", money, src_bank,
dest_bank, src_bank->account_get_accountnum(src_account),
dest_bank->account_get_accountnum(dest_account));
        change_transfer_temp(th);

src_bank->account_get_accountnum(src_account)->change_account_tr(th);

dest_bank->account_get_accountnum(dest_account)->change_account_tr(th);
        return;
}
else {
        std::cout << "Choose between 1 and 2!!!" << endl;
        return;
}
}

void multi_uni_atm::deposit() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int check_num = 0;
    long long int check_sum = 0;
    int withdrawl_count = 0;
    bank* b = find_cardnumber_bank(get_card_insert_slot());
    if (b != get_primary_bank()) {
            std::cout << "A fee of 1,000 won will be charged for deposits to other
banks. Deposit including fees" << endl;
}
}

```

```

while (true) {
    std::cout << "How many 1,000 won would you like to put in? (Enter -1 to
exit)";
    std::cin >> one_thou;
    if (one_thou == -1) {
        return;
    }
    std::cout << "How many 5,000 won would you like to put in? ";
    std::cin >> five_thou;
    std::cout << "How many 1,0000 won would you like to put in? ";
    std::cin >> ten_thou;
    std::cout << "How many 5,0000 won would you like to put in? ";
    std::cin >> fifty_thou;
    if (one_thou + five_thou + ten_thou + fifty_thou > 50) {
        std::cout << "You have exceeded the number of cash cards
available." << endl;
    }
    else { break; }
}
while (1) {
    std::cout << "How many checks would you like to write? ";
    std::cin >> check_num;
    std::cout << endl;
    if (check_num > 30) {
        std::cout << "You have exceeded the number of checks available."
<< endl;
    }
    else { break; }
}
while (1) {
    int flag = 0; //flag = 0 일때 정상
    check_sum = 0;
    for (int i = 0; i < check_num; i++) {
        long long int temp = 0;
        std::cout << i + 1 << "th check, how much? ";
        std::cin >> temp;
        if (temp < 100000 || temp > 999999999) { // 100억 이상은 불가
            flag = 1; //flag = 1이면 에러 발생
            break;
        }
        else {
            check_sum += temp;
        }
    }
    if (flag == 1) {
        std::cout << "The check amount is incorrect." << endl;
    }
}

```

```

        }
        else { break; }
    }
change_atm_cash(one_thou, five_thou, ten_thou, fifty_thou);
long long int m = 1000 * one_thou + 5000 * five_thou + 10000 * ten_thou + 50000
* fifty_thou + check_sum;
if (b != get_primary_bank()) { m = m - 1000; }
b->deposit_bank(get_card_insert_slot(), m);
std::cout << m << "Won has been deposited." << endl;
std::cout << "The balance is " <<
b->account_get_account(get_card_insert_slot())->get_acc_cash() << endl;
transaction_history* th;
th = new transaction_history(get_card_insert_slot(), "deposit", m, b,
b->account_get_account(get_card_insert_slot()));
change_temp(th);
b->account_get_account(get_card_insert_slot())->change_account_tr(th);
return;
}

void multi_uni_atm::withdrawl() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int check_num = 0;
    int flag = 0;
    bank* b = find_cardnumber_bank(get_card_insert_slot());
    if (get_number_of_withdrawl() >= get_limit_of_number_of_withdral()) {
        std::cout << "If you want to withdraw more, you must reset your session."
<< endl;
        return;
    }
    if (b != get_primary_bank()) {
        std::cout << "A fee of 2,000 won is charged for withdrawals from other
banks." << endl;
        flag = 1;
    }
    else {
        std::cout << "There is a fee of 1,000 won for withdrawal." << endl;
    }
    while (1) {
        std::cout << "How much do you want to withdraw (excluding fees)(Enter -1
to exit) : ";
        std::cin >> check_num;
        if (check_num == -1) {
            return;
        }
}
}

```

```

    }
    std::cout << endl;
    if (check_num > 500000) {//출금 가능한 금액을 초과하였는지 조사
        std::cout << "You have exceeded the amount you can withdraw."
    << endl;
}
else if (check_num % 1000 != 0) {
    std::cout << "The minimum unit of currency is 1,000 won." <<
endl;
}
else if (get_atm_cash() < check_num) {
    std::cout << "There is not enough cash in the ATM." << endl;
}
else if (!(b->cn_enough_money_in_acc_bank(get_card_insert_slot(), (flag ?
(check_num + 2000) : (check_num + 1000))))) {
    std::cout << "There is not enough money in the account" << endl;
}
else {
    int m = check_num;
    fifty_thou = _MIN_((m / 50000), get_50000());
    m = m - 50000 * fifty_thou;
    ten_thou = _MIN_((m / 10000), get_10000());
    m = m - 10000 * ten_thou;
    five_thou = _MIN_((m / 5000), get_5000());
    m = m - 5000 * five_thou;
    one_thou = _MIN_((m / 1000), get_1000());
    m = m - 1000 * one_thou;

    if (m != 0) {
        std::cout << "There are not enough bills in the ATM" <<
endl;
    }
    else {
        //atm_cash 조정
        change_atm_cash(-1 * (one_thou), -1 * (five_thou), -1 *
(ten_thou), -1 * (fifty_thou));
        b->withdrawl_bank(get_card_insert_slot(), (flag ?
(check_num + 2000) : (check_num + 1000)));
        std::cout << "In the bank " << (flag ? (check_num + 2000)
: (check_num + 1000)) << "won was withdrawn. Total cash to bring is " << check_num <<
"Won." << endl;
        std::cout << "1000 Won " << one_thou << "sheets / " <<
"5000Won " << five_thou << "sheets / " << "10000 Won " << ten_thou << "sheets / " << "50000
Won " << fifty_thou << "sheets." << endl;
        std::cout << "The balance is " <<
b->account_get_account(get_card_insert_slot())->get_acc_cash() << endl;

```

```

        set_number_of_withdrawl(get_number_of_withdrawl() + 1);
        transaction_history* th = new
transaction_history(get_card_insert_slot(), "Withdrawl", check_num, b,
b->account_get_account(get_card_insert_slot()));
        change_temp(th);

b->account_get_account(get_card_insert_slot())->change_account_tr(th);
        return;
    }
}
}

void multi_uni_atm::interface() {
    increase_session();
    set_number_of_withdrawl(0);
    int password;
    int work_type;
    int password_count = 0;
    int number;//추가

main1:
    std::cout << "Please insert the card.(Please enter the card number). If you want to
exit, enter -1.";
    std::cin >> number; //변경
    set_card_insert_slot(number);//추가
    if (get_card_insert_slot() == -1) {
        return;
    }
    if (get_card_insert_slot() == 7777777) {
        admin_interface();
        goto end;
    }
    if (valid_cardnumber_account(get_card_insert_slot()) == 0) {
        cout << "Invalid card number." << endl;
        goto main1;
    }
    while (1) {
        if (password_count == 3) {
            break;
        }
        std::cout << "Please enter the password." << endl;
        std::cin >> password;
        if (check_password(get_card_insert_slot(), password)) {
            break;
            cout << "Wrong Password" << endl;
        }
        password_count++;
    }
}

```

```

    }
    if (password_count == 3) {
        cout << "You got the password wrong 3 times. " << endl;
        goto end;//세션 end
    }
    while (1) {
        std::cout << "What kind of work would you like to do? Please enter the
number." << endl;
        std::cout << "1. deposit " << "2. withdrawl " << "3. transfer " << "4. end
session: ";
        std::cin >> work_type;
        if (work_type == 1) { deposit(); }
        else if (work_type == 2) { withdrawl(); }
        else if (work_type == 3) { transfer(); }
        else if (work_type == 4) { break; }
        else { std::cout << "You have selected the wrong menu." << endl; }
    }
end:
    cout << "Ends the session." << endl;
    print_session_history();
    decrease_session();
}

```

//강승수 시작

```

void multi_bi_atm::transfer() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int money = 0;
    int money_sum = 0;
    int c1;
    int fee;
    bank* curr_bank = find_cardnumber_bank(get_card_insert_slot());
    long long int dest_account;
    bank* dest_bank;
    long long int src_account;
    bank* src_bank;
    string u_name =
curr_bank->account_get_account(get_card_insert_slot())->get_user_name();

```

```

while (1) {
    std::cout << "Please choose between 1 Cash transfer or 2 Account transfer
: ";
    std::cin >> c1;
    if (c1 == 1) {      //현금 계좌이체
        std::cout << "Please enter the Destination Account : ";
        std::cin >> dest_account;
        dest_bank = find_bank_by_account(dest_account);
        if (!valid_find_bank_by_account(dest_account)) {
            std::cout << "There is no account you entered." << endl;
            return;
        }
        std::cout << "Please enter the cash to send: ";
        std::cin >> money;
        std::cout << "Please enter the amount you wish to send along with
the 5,000 won fee." << endl;
        std::cout << "How many 1,000 won would you like to put in? ";
        std::cin >> one_thou;
        std::cout << "How many 5,000 won would you like to put in? ";
        std::cin >> five_thou;
        std::cout << "How many 10,000 won would you like to put in? ";
        std::cin >> ten_thou;
        std::cout << "How many 50,000 won would you like to put in?";
        std::cin >> fifty_thou;
        money_sum = one_thou * 1000 + five_thou * 5000 + ten_thou *
10000 + fifty_thou * 50000;
        if (money_sum != money + 5000) {
            std::cout << "The money you requested and the money
you entered are different." << endl;
            return;
        }
        change_atm_cash(one_thou, five_thou, ten_thou, fifty_thou);
        curr_bank->cash_transfer_bank(dest_bank, dest_account, money);
        std::cout << "Cash transfer has been completed." << endl;
        transaction_history* th = new
transaction_history(get_card_insert_slot(), "cash_transfer", money, dest_bank,
dest_bank->account_get_accountnum(dest_account));
        change_temp(th);

dest_bank->account_get_accountnum(dest_account)->change_account_tr(th);
        return;
    }

    else if (c1 == 2) {    // 계좌끼리 계좌이체
        std::cout << "Please enter the Destination Account : ";
        std::cin >> dest_account;

```

```

        dest_bank = find_bank_by_account(dest_account);
        if (!valid_find_bank_by_account(dest_account)) {
            std::cout << "There is no account you entered." << endl;
            return;
        }
        std::cout << "Enter your account number : ";
        std::cin >> src_account;
        src_bank = find_bank_by_account(src_account);
        if (!valid_find_bank_by_account(src_account)) {
            std::cout << "There is no account you entered." << endl;
            return;
        }
        if (u_name !=

src_bank->account_get_accountnum(src_account)->get_user_name() or src_bank !=
curr_bank) {
            std::cout << "You cannot send money from this account.
Please insert another bank card!" << endl;
            return;
}

        if (src_bank == get_primary_bank() && dest_bank ==
get_primary_bank()) { fee = 2000; }
        else if ((src_bank != get_primary_bank() && dest_bank ==
get_primary_bank()) or (src_bank == get_primary_bank() && dest_bank !=
get_primary_bank())) { fee = 3000; }
        else if (src_bank != get_primary_bank() && dest_bank !=
get_primary_bank()) { fee = 4000; }

        std::cout << "Please enter the cash to send : ";
        std::cin >> money;
        if
(!curr_bank->cn_enough_money_in_acc_bank(get_card_insert_slot(), money + fee)) {
            std::cout << "There is not enough money in the account.";
            return;
}
        curr_bank->account_transfer_bank(src_account, dest_bank,
dest_account, money, fee);
        std::cout << money << "+" << fee << " Won was deducted from
your account." << endl;
        cout << "The account transfer has been completed." << endl <<
endl;
        transfer_acc_transaction_history* th = new
transfer_acc_transaction_history(get_card_insert_slot(), "account_transfer", money, src_bank,
dest_bank, src_bank->account_get_accountnum(src_account),
dest_bank->account_get_accountnum(dest_account));
        change_transfer_temp(th);

```

```

src_bank->account_get_accountnum(src_account)->change_account_tr(th);

dest_bank->account_get_accountnum(dest_account)->change_account_tr(th);
    return;
}
else {
    std::cout << "Choose between 1 and 2!!!" << endl;
    return;
}
}

void multi_bi_atm::deposit() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int check_num = 0;
    long long int check_sum = 0;
    int withdrawl_count = 0;
    bank* b = find_cardnumber_bank(get_card_insert_slot());
    if (b != get_primary_bank()) {
        std::cout << "A fee of 1,000 won will be charged for deposits to other
banks. Deposit including fees" << endl;
    }
    while (true) {
        std::cout << "How many 1,000 won would you like to put in? (Enter -1 to
exit)":
        std::cin >> one_thou;
        if (one_thou == -1) {
            return;
        }
        std::cout << "How many 5,000 won would you like to put in? ";
        std::cin >> five_thou;
        std::cout << "How many 1,0000 won would you like to put in? ";
        std::cin >> ten_thou;
        std::cout << "How many 5,0000 won would you like to put in? ";
        std::cin >> fifty_thou;
        if (one_thou + five_thou + ten_thou + fifty_thou > 50) {
            std::cout << "You have exceeded the number of cash cards
available." << endl;
        }
        else { break; }
    }
    while (1) {

```

```

        std::cout << "How many checks would you like to write? ";
        std::cin >> check_num;
        std::cout << endl;
        if (check_num > 30) {
            std::cout << "You have exceeded the number of checks available."
<< endl;
        }
        else { break; }
    }
    while (1) {
        int flag = 0; //flag = 0 일때 정상
        check_sum = 0;
        for (int i = 0; i < check_num; i++) {
            long long int temp = 0;
            std::cout << i + 1 << "th check, how much? ";
            std::cin >> temp;
            if (temp < 100000 || temp > 999999999) { // 100억 이상은 불가
                flag = 1; //flag = 1이면 에러 발생
                break;
            }
            else {
                check_sum += temp;
            }
        }
        if (flag == 1) {
            std::cout << "The check amount is incorrect." << endl;
        }
        else { break; }
    }
    change_atm_cash(one_thou, five_thou, ten_thou, fifty_thou);
    long long int m = 1000 * one_thou + 5000 * five_thou + 10000 * ten_thou + 50000
* fifty_thou + check_sum;
    if (b != get_primary_bank()) { m = m - 1000; }
    b->deposit_bank(get_card_insert_slot(), m);
    std::cout << m << "Won has been deposited." << endl;
    std::cout << "The balance is " <<
    b->account_get_account(get_card_insert_slot())->get_acc_cash() << endl;
    transaction_history* th;
    th = new transaction_history(get_card_insert_slot(), "deposit", m, b,
b->account_get_account(get_card_insert_slot()));
    change_temp(th);
    b->account_get_account(get_card_insert_slot())->change_account_tr(th);
    return;
}

void multi_bi_atm::withdrawl() {

```

```

int one_thou = 0;
int five_thou = 0;
int ten_thou = 0;
int fifty_thou = 0;
int check_num = 0;
int flag = 0;
bank* b = find_cardnumber_bank(get_card_insert_slot());
if (get_number_of_withdrawl() >= get_limit_of_number_of_withdrawl()) {
    std::cout << "If you want to withdraw more, you must reset your session."
<< endl;
    return;
}
if (b != get_primary_bank()) {
    std::cout << "A fee of 2,000 won is charged for withdrawals from other
banks." << endl;
    flag = 1;
}
else {
    std::cout << "There is a fee of 1,000 won for withdrawal." << endl;
}
while (1) {
    std::cout << "How much do you want to withdraw (excluding fees)(Enter -1
to exit) : ";
    std::cin >> check_num;
    if (check_num == -1) {
        return;
    }
    std::cout << endl;
    if (check_num > 500000) {//출금 가능한 금액을 초과하였는지 조사
        std::cout << "You have exceeded the amount you can withdraw."
    }
    else if (check_num % 1000 != 0) {
        std::cout << "The minimum unit of currency is 1,000 won." <<
endl;
    }
    else if (get_atm_cash() < check_num) {
        std::cout << "There is not enough cash in the ATM." << endl;
    }
    else if (!(b->cn_enough_money_in_acc_bank(get_card_insert_slot(), (flag ?
(check_num + 2000) : (check_num + 1000))))) {
        std::cout << "There is not enough money in the account" << endl;
    }
    else {
        int m = check_num;
        fifty_thou = _MIN_((m / 50000), get_50000());

```

```

        m = m - 50000 * fifty_thou;
        ten_thou = _MIN_((m / 10000), get_10000());
        m = m - 10000 * ten_thou;
        five_thou = _MIN_((m / 5000), get_5000());
        m = m - 5000 * five_thou;
        one_thou = _MIN_((m / 1000), get_1000());
        m = m - 1000 * one_thou;

        if (m != 0) {
            std::cout << "There are not enough bills in the ATM" <<
endl;
        }
        else {
            //atm_cash 조정
            change_atm_cash(-1 * (one_thou), -1 * (five_thou), -1 *
(ten_thou), -1 * (fifty_thou));
            b->withdrawl_bank(get_card_insert_slot(), (flag ?
(check_num + 2000) : (check_num + 1000)));
            std::cout << "In the bank " << (flag ? (check_num + 2000)
: (check_num + 1000)) << "won was withdrawn. Total cash to bring is " << check_num <<
"Won." << endl;
            std::cout << "1000 Won " << one_thou << "sheets / " <<
"5000Won " << five_thou << "sheets / " << "10000 Won " << ten_thou << "sheets / " << "50000
Won " << fifty_thou << "sheets." << endl;
            std::cout << "The balance is " <<
b->account_get_account(get_card_insert_slot())->get_acc_cash() << endl;
            set_number_of_withdrawl(get_number_of_withdrawl() + 1);
            transaction_history* th = new
transaction_history(get_card_insert_slot(), "Withdrawl", check_num, b,
b->account_get_account(get_card_insert_slot()));
            change_temp(th);

            b->account_get_account(get_card_insert_slot())->change_account_tr(th);
            return;
        }
    }
}

void multi_bi_atm::interface() {
    increase_session();
    set_number_of_withdrawl(0);
    int password;
    int work_type;
    int password_count = 0;
    int number;//추가
}

```

```

main1:
    std::cout << "Please insert the card.(Please enter the card number). If you want to
exit, enter -1.";
    std::cin >> number; //변경
    set_card_insert_slot(number); //추가
    if (get_card_insert_slot() == -1) {
        return;
    }
    if (get_card_insert_slot() == 7777777) {
        admin_interface();
        goto end;
    }
    if (valid_cardnumber_account(get_card_insert_slot()) == 0) {
        cout << "Invalid card number." << endl;
        goto main1;
    }
    while (1) {
        if (password_count == 3) {
            break;
        }
        std::cout << "Please enter the password." << endl;
        std::cin >> password;
        if (check_password(get_card_insert_slot(), password)) {
            break;
        }
        cout << "Wrong Password" << endl;
        password_count++;
    }
    if (password_count == 3) {
        cout << "You got the password wrong 3 times. " << endl;
        goto end; //종료
    }
    while (1) {
        std::cout << "What kind of work would you like to do? Please enter the
number." << endl;
        std::cout << "1. deposit " << "2. withdrawl " << "3. transfer " << "4. end
session: ";
        std::cin >> work_type;
        if (work_type == 1) { deposit(); }
        else if (work_type == 2) { withdrawl(); }
        else if (work_type == 3) { transfer(); }
        else if (work_type == 4) { break; }
        else { std::cout << "You have selected the wrong menu." << endl; }
    }
end:
    cout << "Ends the session." << endl;

```

```

        print_session_history();
        decrease_session();
    }

void multi_bi_atm::koreantransfer() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int money = 0;
    int money_sum = 0;
    int c1;
    int fee;
    bank* curr_bank = find_cardnumber_bank(get_card_insert_slot());
    long long int dest_account;
    bank* dest_bank;
    long long int src_account;
    bank* src_bank;
    string u_name =
curr_bank->account_get_account(get_card_insert_slot())->get_user_name();

    while (1) {
        std::cout << "번호를 고르시오. " << "1. 현금 이체 2. 계좌 이체: ";
        std::cin >> c1;
        if (c1 == 1) { //현금 계좌이체
            std::cout << "돈을 받고자 하는 계좌의 계좌번호를 입력하시오. : ";
            std::cin >> dest_account;
            dest_bank = find_bank_by_account(dest_account);
            if (!valid_find_bank_by_account(dest_account)) {
                std::cout << "당신이 입력한 계좌번호가 존재하지 않습니다." <<
endl;
                return;
            }
            std::cout << "돈을 얼마나 보내겠습니까? ";
            std::cin >> money;
            std::cout << "5000원의 수수료가 드니, 수수료를 포함해서 현금을
넣어주세요" << endl;
            std::cout << "1000원 몇 장 넣겠습니까? ";
            std::cin >> one_thou;
            std::cout << "5000원 몇 장 넣겠습니까? ";
            std::cin >> five_thou;
            std::cout << "10000원 몇 장 넣겠습니까? ";
            std::cin >> ten_thou;
            std::cout << "50000원 몇 장 넣겠습니까? ";
            std::cin >> fifty_thou;
        }
    }
}

```

```

        money_sum = one_thou * 1000 + five_thou * 5000 + ten_thou *
10000 + fifty_thou * 5000;
        if (money_sum != money + 5000) {
            std::cout << "요청하신 금액과 지불하신 현금 액수가 다릅니다."
<< endl;
            return;
        }
        change_atm_cash(one_thou, five_thou, ten_thou, fifty_thou);
        curr_bank->cash_transfer_bank(dest_bank, dest_account, money);
        std::cout << "이체가 완료되었습니다." << endl;
        transaction_history* th = new
transaction_history(get_card_insert_slot(), "현금_계좌이체", money, dest_bank,
dest_bank->account_get_accountnum(dest_account));
        change_temp(th);

dest_bank->account_get_accountnum(dest_account)->change_account_tr(th);
        return;
    }

else if (c1 == 2) { // 계좌끼리 계좌이체
    std::cout << "돈을 받을 계좌의 계좌번호를 입력하세요: ";
    std::cin >> dest_account;
    dest_bank = find_bank_by_account(dest_account);
    if (!valid_find_bank_by_account(dest_account)) {
        std::cout << "입력하신 계좌번호의 계좌가 존재하지 않습니다."
<< endl;
        return;
    }
    std::cout << "본인의 계좌번호를 입력하세요: ";
    std::cin >> src_account;
    src_bank = find_bank_by_account(src_account);
    if (!valid_find_bank_by_account(src_account)) {
        std::cout << "입력하신 계좌번호의 계좌가 존재하지 않습니다."
<< endl;
        return;
    }
    if (u_name !=
src_bank->account_get_accountnum(src_account)->get_user_name() or src_bank !=
curr_bank) {
        std::cout << "해당 계좌에서 돈을 이체할 수 없습니다." <<
endl;
        return;
    }

    if (src_bank == get_primary_bank() && dest_bank ==
get_primary_bank()) { fee = 2000; }

```

```

        else if ((src_bank != get_primary_bank() && dest_bank == get_primary_bank()) || (src_bank == get_primary_bank() && dest_bank != get_primary_bank())) { fee = 3000; }

        else if (src_bank != get_primary_bank() && dest_bank != get_primary_bank()) { fee = 4000; }

        std::cout << "송금하실 금액을 알려주세요: ";
        std::cin >> money;
        if (!curr_bank->cn_enough_money_in_acc_bank(get_card_insert_slot(), money + fee)) {
            std::cout << "계좌에 돈이 충분하지 않습니다.";
            return;
        }
        curr_bank->account_transfer_bank(src_account, dest_bank, dest_account, money, fee);
        std::cout << money << "+" << fee << " 원이 계좌에서 출금되었습니다."
<< endl;
        cout << "계좌이체가 완료되었습니다." << endl << endl;
        transfer_acc_transaction_history* th = new
        transfer_acc_transaction_history(get_card_insert_slot(), "계좌_계좌이체", money, src_bank,
        dest_bank, src_bank->account_get_accountnum(src_account),
        dest_bank->account_get_accountnum(dest_account));
        change_transfer_temp(th);

src_bank->account_get_accountnum(src_account)->change_account_tr(th);

dest_bank->account_get_accountnum(dest_account)->change_account_tr(th);
        return;
    }
    else {
        std::cout << "1과 2중에 고르세요." << endl;
        return;
    }
}

void multi_bi_atm::koreanwithdrawl() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int check_num = 0;
    int flag = 0;
    bank* b = find_cardnumber_bank(get_card_insert_slot());
    if (get_number_of_withdrawl() >= get_limit_of_number_of_withdrawl()) {
        std::cout << "더 출금하시려면 세션을 초기화 하셔야 합니다." << endl;
        return;
    }
}

```

```

    }

    if (b != get_primary_bank()) {
        std::cout << "타 은행 출금에는 수수료 2000원이 발생합니다." << endl;
        flag = 1;
    }
    else {
        std::cout << "자 은행 출금에는 수수료 1000원이 발생합니다." << endl;
    }
    while (1) {
        std::cout << "얼마를 출금하시겠습니까?(수수료 제외) (종료하려면 -1을
입력하세요):";
        std::cin >> check_num;
        if (check_num == -1) {
            return;
        }
        std::cout << endl;
        if (check_num > 500000) {//출금 가능한 금액을 초과하였는지 조사
            std::cout << "출금 가능하신 금액을 초과하셨습니다" << endl;
        }
        else if (check_num % 1000 != 0) {
            std::cout << "화폐 최소 단위는 1000원 입니다" << endl;
        }
        else if (get_atm_cash() < check_num) {
            std::cout << "ATM에 충분한 현금이 존재하지 않습니다" << endl;
        }
        else if (!(b->cn_enough_money_in_acc_bank(get_card_insert_slot(), (flag ?
(check_num + 2000) : (check_num + 1000))))) {
            std::cout << "계좌에 돈이 충분하지 않습니다" << endl;
        }
        else {
            int m = check_num;
            fifty_thou = _MIN_((m / 5000), get_50000());
            m = m - 50000 * fifty_thou;
            ten_thou = _MIN_((m / 10000), get_10000());
            m = m - 10000 * ten_thou;
            five_thou = _MIN_((m / 5000), get_5000());
            m = m - 5000 * five_thou;
            one_thou = _MIN_((m / 1000), get_1000());
            m = m - 1000 * one_thou;

            if (m != 0) {
                std::cout << "ATM에 지폐의 개수가 부족합니다" << endl;
            }
            else {
                //atm_cash 조정
                change_atm_cash(-1 * (one_thou), -1 * (five_thou), -1 *

```

```

(ten_thou), -1 * (fifty_thou));
                                b->withdrawl_bank(get_card_insert_slot(), (flag ?
(check_num + 2000) : (check_num + 1000)));
                                std::cout << "은행에서 " << (flag ? (check_num + 2000) :
(check_num + 1000)) << "원이 출금되었고 가져가실 현금은 총 " << check_num << "원 입니다." <<
endl;
                                std::cout << "천원 " << one_thou << "장 / " << "오천원 " <<
five_thou << "장 / " << "만원 " << ten_thou << "장 / " << "오만원 " << fifty_thou << "장" <<
endl;
                                std::cout << "잔액은 " <<
b->account_get_account(get_card_insert_slot())->get_acc_cash() << endl;
                                set_number_of_withdrawl(get_number_of_withdrawl() + 1);
                                transaction_history* th = new
transaction_history(get_card_insert_slot(), "출금", check_num, b,
b->account_get_account(get_card_insert_slot()));
                                change_temp(th);

b->account_get_account(get_card_insert_slot())->change_account_tr(th);
                                return;
}
}
}
}

void multi_bi_atm::koreandeposit() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int check_num = 0;
    long long int check_sum = 0;
    int withdrawl_count = 0;
    bank* b = find_cardnumber_bank(get_card_insert_slot());
    if (b != get_primary_bank()) {
        std::cout << "타 은행 입금에는 수수료 1000원이 발생합니다. 수수료를 포함해서
입금하세요" << endl;
    }
    while (true) {
        std::cout << "천원을 몇 장 넣으시겠습니까? (종료하려면 -1을 입력하세요)";
        std::cin >> one_thou;
        if (one_thou == -1) {
            return;
        }
        std::cout << "오천원을 몇 장 넣으시겠습니까? ";
        std::cin >> five_thou;
        std::cout << "만원을 몇 장 넣으시겠습니까? ";
        std::cin >> ten_thou;

```

```

        std::cout << "오만원을 몇 장 넣으시겠습니까? ";
        std::cin >> fifty_thou;
        if (one_thou + five_thou + ten_thou + fifty_thou > 50) {
            std::cout << "가능한 현금 장 수를 초과하셨습니다." << endl;
        }
        else { break; }
    }
    while (1) {
        std::cout << "수표를 몇 장 넣으시겠습니까? ";
        std::cin >> check_num;
        std::cout << endl;
        if (check_num > 30) {
            std::cout << "가능한 수표 장 수를 초과하셨습니다." << endl;
        }
        else { break; }
    }
    while (1) {
        int flag = 0; //flag = 0 일때 정상
        check_sum = 0;
        for (int i = 0; i < check_num; i++) {
            long long int temp = 0;
            std::cout << i + 1 << "번째 수표의 금액은 얼마입니까? ";
            std::cin >> temp;
            if (temp < 100000 || temp > 999999999) { // 100억 이상은 불가
                flag = 1; //flag = 1이면 에러 발생
                break;
            }
            else {
                check_sum += temp;
            }
        }
        if (flag == 1) {
            std::cout << "잘못된 수표 금액입니다." << endl;
        }
        else { break; }
    }
    change_atm_cash(one_thou, five_thou, ten_thou, fifty_thou);
    long long int m = 1000 * one_thou + 5000 * five_thou + 10000 * ten_thou + 50000
    * fifty_thou + check_sum;
    if (b != get_primary_bank()) { m = m - 1000; }
    b->deposit_bank(get_card_insert_slot(), m);
    std::cout << m << "원이 입금되었습니다." << endl;
    std::cout << "잔액은 " <<
    b->account_get_account(get_card_insert_slot())->get_acc_cash() << endl;
    transaction_history* th;
    th = new transaction_history(get_card_insert_slot(), "입금", m, b,

```

```

b->account_get_account(get_card_insert_slot());
    change_temp(th);
    b->account_get_account(get_card_insert_slot())->change_account_tr(th);
    return;
}

void multi_bi_atm::korean_print_session_history() {
    for (int i = 0; i < get_temp().size(); i++) {
        get_temp()[i]->korean_print_history();
        change_atm_tr(get_temp()[i]);
    }
    clear_temp(); //정세훈 찾아보고 벡터 초기화
}

void multi_bi_atm::koreaninterface() {
    increase_session();
    set_number_of_withdrawl(0);
    int password;
    int work_type;
    int password_count = 0;
    int number;//추가
main1:
    std::cout << "카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을
    입력해주세요.:";

    std::cin >> number; //변경
    set_card_insert_slot(number); //추가
    if (get_card_insert_slot() == -1) {
        return;
    }
    if (get_card_insert_slot() == 7777777) {
        admin_interface_korean();
        goto end;
    }
    if (valid_cardnumber_account(get_card_insert_slot()) == false) {
        cout << "유효하지 않은 카드 번호입니다. " << endl;
        goto main1;
    }
    while (1) {
        if (password_count == 3) {
            break;
        }
        std::cout << "비밀번호를 입력해주세요." << endl;
        std::cin >> password;
        if (check_password(get_card_insert_slot(), password)) {
            break;
        }
    }
}

```

```

        cout << "비밀번호가 틀립니다." << endl;
        password_count++;
    }
    if (password_count == 3) {
        cout << "비밀번호를 3번 틀리셨습니다. " << endl;
        goto end;//세션 end
    }
    while (1) {
        std::cout << "어떤 업무를 하시겠습니까? 번호를 기입하여 주세요." << endl;
        std::cout << "1. 입금 " << "2. 출금 " << "3. 이체 " << "4. 세션 종료 : ";
        std::cin >> work_type;
        if (work_type == 1) { koreandeposit(); }
        else if (work_type == 2) { koreanwithdrawl(); }
        else if (work_type == 3) { koreantransfer(); }
        else if (work_type == 4) { break; }
        else { std::cout << "잘못된 메뉴를 선택하셨습니다." << endl; }
    }
end:
    cout << "세션을 종료합니다." << endl;
    korean_print_session_history();
    decrease_session();
}

```

//single bi atm

```

void single_bi_atm::transfer() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int money = 0;
    int money_sum = 0;
    int c1;
    int fee;
    bank* curr_bank = find_cardnumber_bank(get_card_insert_slot());
    long long int dest_account;
    bank* dest_bank;
    long long int src_account;
    bank* src_bank;
    string u_name =
curr_bank->account_get_account(get_card_insert_slot())->get_user_name();

    while (1) {

```

```

        std::cout << "Please choose between 1 Cash transfer or 2 Account transfer
        : ";
        std::cin >> c1;
        if (c1 == 1) {          //현금 계좌이체
            std::cout << "Please enter the Destination Account : ";
            std::cin >> dest_account;
            dest_bank = find_bank_by_account(dest_account);
            if (dest_bank != get_primary_bank()) {
                cout << "This is not primary bank account." << endl;
                return;
            }
            if (!valid_find_bank_by_account(dest_account)) {
                std::cout << "There is no account you entered." << endl;
                return;
            }
            std::cout << "Please enter the cash to send: ";
            std::cin >> money;
            std::cout << "Please enter the amount you wish to send along with
the 5,000 won fee." << endl;
            std::cout << "How many 1,000 won would you like to put in? ";
            std::cin >> one_thou;
            std::cout << "How many 5,000 won would you like to put in? ";
            std::cin >> five_thou;
            std::cout << "How many 10,000 won would you like to put in? ";
            std::cin >> ten_thou;
            std::cout << "How many 50,000 won would you like to put in?";
            std::cin >> fifty_thou;
            money_sum = one_thou * 1000 + five_thou * 5000 + ten_thou *
10000 + fifty_thou * 50000;
            if (money_sum != money + 5000) {
                std::cout << "The money you requested and the money
you entered are different." << endl;
                return;
            }
            change_atm_cash(one_thou, five_thou, ten_thou, fifty_thou);
            curr_bank->cash_transfer_bank(dest_bank, dest_account, money);
            std::cout << "Cash transfer has been completed." << endl;
            transaction_history* th = new
transaction_history(get_card_insert_slot(), "cash_transfer", money, dest_bank,
dest_bank->account_get_accountnum(dest_account));
            change_temp(th);

dest_bank->account_get_accountnum(dest_account)->change_account_tr(th);
        return;
    }
}

```

```

else if (c1 == 2) {    // 계좌끼리 계좌이체
    std::cout << "Please enter the Destination Account : ";
    std::cin >> dest_account;
    dest_bank = find_bank_by_account(dest_account);
    if (!valid_find_bank_by_account(dest_account)) {
        std::cout << "There is no account you entered." << endl;
        return;
    }
    if (dest_bank != get_primary_bank()) {
        cout << "This is not primary bank account." << endl;
        return;
    }
    std::cout << "Enter your account number : ";
    std::cin >> src_account;
    src_bank = find_bank_by_account(src_account);
    if (!valid_find_bank_by_account(src_account)) {
        std::cout << "There is no account you entered." << endl;
        return;
    }
    if (src_bank != get_primary_bank()) {
        cout << "This is not primary bank account." << endl;
        return;
    }
    if (u_name !=

src_bank->account_get_accountnum(src_account)->get_user_name() or src_bank !=
curr_bank) {
        std::cout << "You cannot send money from this account.
Please insert another bank card!" << endl;
        return;
    }

    if (src_bank == get_primary_bank() && dest_bank ==
get_primary_bank()) { fee = 2000; }
    else if ((src_bank != get_primary_bank() && dest_bank ==
get_primary_bank()) or (src_bank == get_primary_bank() && dest_bank !=
get_primary_bank())) { fee = 3000; }
    else if (src_bank != get_primary_bank() && dest_bank !=
get_primary_bank()) { fee = 4000; }

    std::cout << "Please enter the cash to send : ";
    std::cin >> money;
    if
(!curr_bank->cn_enough_money_in_acc_bank(get_card_insert_slot(), money + fee)) {
        std::cout << "There is not enough money in the account.";
        return;
}

```

```

curr_bank->account_transfer_bank(src_account, dest_bank,
dest_account, money, fee);
    std::cout << money << "+" << fee << " Won was deducted from
your account." << endl;
    cout << "The account transfer has been completed." << endl <<
endl;
    transfer_acc_transaction_history* th = new
transfer_acc_transaction_history(get_card_insert_slot(), "account_transfer", money, src_bank,
dest_bank, src_bank->account_get_accountnum(src_account),
dest_bank->account_get_accountnum(dest_account));
    change_transfer_temp(th);

src_bank->account_get_accountnum(src_account)->change_account_tr(th);

dest_bank->account_get_accountnum(dest_account)->change_account_tr(th);
    return;
}
else {
    std::cout << "Choose between 1 and 2!!!" << endl;
    return;
}
}

void single_bi_atm::deposit() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int check_num = 0;
    long long int check_sum = 0;
    int withdrawl_count = 0;
    bank* b = find_cardnumber_bank(get_card_insert_slot());
    if (b != get_primary_bank()) {
        std::cout << "A fee of 1,000 won will be charged for deposits to other
banks. Deposit including fees" << endl;
    }
    while (true) {
        std::cout << "How many 1,000 won would you like to put in? (Enter -1 to
exit)";
        std::cin >> one_thou;
        if (one_thou == -1) {
            return;
        }
        std::cout << "How many 5,000 won would you like to put in? ";
        std::cin >> five_thou;
    }
}

```

```

        std::cout << "How many 1,0000 won would you like to put in? ";
        std::cin >> ten_thou;
        std::cout << "How many 5,0000 won would you like to put in? ";
        std::cin >> fifty_thou;
        if (one_thou + five_thou + ten_thou + fifty_thou > 50) {
            std::cout << "You have exceeded the number of cash cards
available." << endl;
        }
        else { break; }
    }
    while (1) {
        std::cout << "How many checks would you like to write? ";
        std::cin >> check_num;
        std::cout << endl;
        if (check_num > 30) {
            std::cout << "You have exceeded the number of checks available."
<< endl;
        }
        else { break; }
    }
    while (1) {
        int flag = 0; //flag = 0 일때 정상
        check_sum = 0;
        for (int i = 0; i < check_num; i++) {
            long long int temp = 0;
            std::cout << i + 1 << "th check, how much? ";
            std::cin >> temp;
            if (temp < 100000 || temp > 999999999) { // 100억 이상은 불가
                flag = 1; //flag = 1이면 에러 발생
                break;
            }
            else {
                check_sum += temp;
            }
        }
        if (flag == 1) {
            std::cout << "The check amount is incorrect." << endl;
        }
        else { break; }
    }
    change_atm_cash(one_thou, five_thou, ten_thou, fifty_thou);
    long long int m = 1000 * one_thou + 5000 * five_thou + 10000 * ten_thou + 50000
* fifty_thou + check_sum;
    if (b != get_primary_bank()) { m = m - 1000; }
    b->deposit_bank(get_card_insert_slot(), m);
    std::cout << m << "Won has been deposited." << endl;
}

```

```

        std::cout << "The balance is " <<
b->account_get_account(get_card_insert_slot())->get_acc_cash() << endl;
        transaction_history* th;
        th = new transaction_history(get_card_insert_slot(), "deposit", m, b,
b->account_get_account(get_card_insert_slot()));
        change_temp(th);
        b->account_get_account(get_card_insert_slot())->change_account_tr(th);
        return;
    }

void single_bi_atm::withdrawl() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int check_num = 0;
    int flag = 0;
    bank* b = find_cardnumber_bank(get_card_insert_slot());
    if (get_number_of_withdrawl() >= get_limit_of_number_of_withdrawl()) {
        std::cout << "If you want to withdraw more, you must reset your session."
<< endl;
        return;
    }
    if (b != get_primary_bank()) {
        std::cout << "A fee of 2,000 won is charged for withdrawals from other
banks." << endl;
        flag = 1;
    }
    else {
        std::cout << "There is a fee of 1,000 won for withdrawal." << endl;
    }
    while (1) {
        std::cout << "How much do you want to withdraw (excluding fees)(Enter -1
to exit) : ";
        std::cin >> check_num;
        if (check_num == -1) {
            return;
        }
        std::cout << endl;
        if (check_num > 500000) {//출금 가능한 금액을 초과하였는지 조사
            std::cout << "You have exceeded the amount you can withdraw."
<< endl;
        }
        else if (check_num % 1000 != 0) {
            std::cout << "The minimum unit of currency is 1,000 won." <<
endl;
    }
}

```

```

    }

    else if (get_atm_cash() < check_num) {
        std::cout << "There is not enough cash in the ATM." << endl;
    }

    else if (!(b->cn_enough_money_in_acc_bank(get_card_insert_slot(), (flag ?
(check_num + 2000) : (check_num + 1000))))) {
        std::cout << "There is not enough money in the account" << endl;
    }

    else {
        int m = check_num;
        fifty_thou = _MIN_((m / 50000), get_50000());
        m = m - 50000 * fifty_thou;
        ten_thou = _MIN_((m / 10000), get_10000());
        m = m - 10000 * ten_thou;
        five_thou = _MIN_((m / 5000), get_5000());
        m = m - 5000 * five_thou;
        one_thou = _MIN_((m / 1000), get_1000());
        m = m - 1000 * one_thou;

        if (m != 0) {
            std::cout << "There are not enough bills in the ATM" <<
endl;
        }
        else {
            //atm_cash 조정
            change_atm_cash(-1 * (one_thou), -1 * (five_thou), -1 *
(ten_thou), -1 * (fifty_thou));
            b->withdrawl_bank(get_card_insert_slot(), (flag ?
(check_num + 2000) : (check_num + 1000)));
            std::cout << "In the bank " << (flag ? (check_num + 2000)
: (check_num + 1000)) << "won was withdrawn, Total cash to bring is " << check_num <<
"Won." << endl;
            std::cout << "1000 Won " << one_thou << "sheets / " <<
"5000Won " << five_thou << "sheets / " << "10000 Won " << ten_thou << "sheets / " << "50000
Won " << fifty_thou << "sheets." << endl;
            std::cout << "The balance is " <<
b->account_get_account(get_card_insert_slot())->get_acc_cash() << endl;
            set_number_of_withdrawl(get_number_of_withdrawl() + 1);
            transaction_history* th = new
transaction_history(get_card_insert_slot(), "Withdrawl", check_num, b,
b->account_get_account(get_card_insert_slot()));
            change_temp(th);

b->account_get_account(get_card_insert_slot())->change_account_tr(th);
return;
}

```

```

        }
    }
}

void single_bi_atm::korean_print_session_history() {
    for (int i = 0; i < get_temp().size(); i++) {
        get_temp()[i]->korean_print_history();
        change_atm_tr(get_temp()[i]);
    }
    clear_temp();//정세훈 찾아보고 벡터 초기화
}

void single_bi_atm::interface() {
    increase_session();
    set_number_of_withdrawl(0);
    int password;
    int work_type;
    int password_count = 0;
    int number;//추가

main1:
    std::cout << "Please insert the card.(Please enter the card number). If you want to
exit, enter -1.";
    std::cin >> number; //변경
    set_card_insert_slot(number);//추가
    if (get_card_insert_slot() == -1) {
        return;
    }
    if (get_card_insert_slot() == 7777777) {
        admin_interface();
        goto end;
    }

    if (find_cardnumber_bank(get_card_insert_slot()) != get_primary_bank()) {
        cout << " This is not primary bank account." << endl;
        return;
    }
    if (valid_cardnumber_account(get_card_insert_slot()) == 0) {
        cout << "Invalid card number." << endl;
        goto main1;
    }
    while (1) {
        if (password_count == 3) {
            break;

```

```

        }
        std::cout << "Please enter the password." << endl;
        std::cin >> password;
        if (check_password(get_card_insert_slot(), password)) {
            break;
        }
        cout << "Wrong Password" << endl;
        password_count++;
    }
    if (password_count == 3) {
        cout << "You got the password wrong 3 times. " << endl;
        goto end;//종료 end
    }
    while (1) {
        std::cout << "What kind of work would you like to do? Please enter the
number." << endl;
        std::cout << "1. deposit " << "2. withdrawl " << "3. transfer " << "4. end
session: ";
        std::cin >> work_type;
        if (work_type == 1) { deposit(); }
        else if (work_type == 2) { withdrawl(); }
        else if (work_type == 3) { transfer(); }
        else if (work_type == 4) { break; }
        else { std::cout << "You have selected the wrong menu." << endl; }
    }
end:
    cout << "Ends the session." << endl;
    print_session_history();
    decrease_session();
}

```

```

void single_bi_atm::koreantransfer() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int money = 0;
    int money_sum = 0;
    int c1;
    int fee;
    bank* curr_bank = find_cardnumber_bank(get_card_insert_slot());
    long long int dest_account;
    bank* dest_bank;
    long long int src_account;
    bank* src_bank;

```

```

        string u_name =
curr_bank->account_get_account(get_card_insert_slot())->get_user_name();

while (1) {
    std::cout << "번호를 고르시오. " << "1. 현금 이체 2. 계좌 이체: ";
    std::cin >> c1;
    if (c1 == 1) {          //현금 계좌이체
        std::cout << "돈을 받고자 하는 계좌의 계좌번호를 입력하시오. : ";
        std::cin >> dest_account;
        dest_bank = find_bank_by_account(dest_account);
        if (dest_bank != get_primary_bank()) {
            cout << "이 은행은 주 은행 계좌가 아닙니다." << endl;
            return;
        }
        if (!valid_find_bank_by_account(dest_account)) {
            std::cout << "당신이 입력한 계좌번호가 존재하지 않습니다." <<
endl;
            return;
        }
        std::cout << "돈을 얼마나 보내시겠습니까? " << endl;
        std::cin >> money;
        std::cout << "5000원의 수수료가 드니. 수수료를 포함해서 현금을
넣어주세요" << endl;
        std::cout << "1000원 몇 장 넣겠습니까? ";
        std::cin >> one_thou;
        std::cout << "5000원 몇 장 넣겠습니까? ";
        std::cin >> five_thou;
        std::cout << "10000원 몇 장 넣겠습니까? ";
        std::cin >> ten_thou;
        std::cout << "50000원 몇 장 넣겠습니까? ";
        std::cin >> fifty_thou;
        money_sum = one_thou * 1000 + five_thou * 5000 + ten_thou *
10000 + fifty_thou * 50000;
        if (money_sum != money + 5000) {
            std::cout << "요청하신 금액과 지불하신 현금 액수가 다릅니다."
<< endl;
            return;
        }
        change_atm_cash(one_thou, five_thou, ten_thou, fifty_thou);
        curr_bank->cash_transfer_bank(dest_bank, dest_account, money);
        std::cout << "이체가 완료되었습니다." << endl;
        transaction_history* th = new
transaction_history(get_card_insert_slot(), "현금_계좌이체", money, dest_bank,
dest_bank->account_get_accountnum(dest_account));
        change_temp(th);
    }
}

```

```

dest_bank->account_get_accountnum(dest_account)->change_account_tr(th);
    return;
}

else if (c1 == 2) { // 계좌끼리 계좌이체
    std::cout << "돈을 받을 계좌의 계좌번호를 입력하세요: ";
    std::cin >> dest_account;
    dest_bank = find_bank_by_account(dest_account);
    if (!valid_find_bank_by_account(dest_account)) {
        std::cout << "입력하신 계좌번호의 계좌가 존재하지 않습니다."
<< endl;
    }
    return;
}
if (dest_bank != get_primary_bank()) {
    cout << "주 은행 계좌가 아닙니다." << endl;
    return;
}
std::cout << "본인의 계좌번호를 입력하세요: ";
std::cin >> src_account;
src_bank = find_bank_by_account(src_account);
if (!valid_find_bank_by_account(src_account)) {
    std::cout << "입력하신 계좌번호의 계좌가 존재하지 않습니다."
<< endl;
}
if (src_bank != get_primary_bank()) {
    cout << "주계좌 은행이 아닙니다." << endl;
    return;
}
if (u_name !=

src_bank->account_get_accountnum(src_account)->get_user_name() or src_bank !=
curr_bank) {
    std::cout << "해당 계좌에서 돈을 이체할 수 없습니다." <<
endl;
    return;
}

if (src_bank == get_primary_bank() && dest_bank ==
get_primary_bank()) { fee = 2000; }
else if ((src_bank != get_primary_bank() && dest_bank ==
get_primary_bank()) or (src_bank == get_primary_bank() && dest_bank !=
get_primary_bank())) { fee = 3000; }
else if (src_bank != get_primary_bank() && dest_bank !=
get_primary_bank()) { fee = 4000; }

std::cout << "송금하실 금액을 알려주세요: ";

```

```

        std::cin >> money;
        if
(!curr_bank->cn_enough_money_in_acc_bank(get_card_insert_slot(), money + fee)) {
            std::cout << "계좌에 돈이 충분하지 않습니다.";
            return;
        }
        curr_bank->account_transfer_bank(src_account, dest_bank,
dest_account, money, fee);
        std::cout << money << "+" << fee << " 원이 계좌에서 출금되었습니다."
<< endl;
        cout << "계좌이체가 완료되었습니다." << endl << endl;
        transfer_acc_transaction_history* th = new
transfer_acc_transaction_history(get_card_insert_slot(), "계좌_계좌이체", money, src_bank,
dest_bank, src_bank->account_get_accountnum(src_account),
dest_bank->account_get_accountnum(dest_account));
        change_transfer_temp(th);

src_bank->account_get_accountnum(src_account)->change_account_tr(th);

dest_bank->account_get_accountnum(dest_account)->change_account_tr(th);
        return;
    }
    else {
        std::cout << "1과 2중에 고르세요." << endl;
        return;
    }
}
}

void single_bi_atm::koreanwithdrawl() {
    int one_thou = 0;
    int five_thou = 0;
    int ten_thou = 0;
    int fifty_thou = 0;
    int check_num = 0;
    int flag = 0;
    bank* b = find_cardnumber_bank(get_card_insert_slot());
    if (get_number_of_withdrawl() >= get_limit_of_number_of_withdral()) {
        std::cout << "더 출금하시려면 세션을 초기화 하셔야 합니다." << endl;
        return;
    }
    if (b != get_primary_bank()) {
        std::cout << "타 은행 출금에는 수수료 2000원이 발생합니다." << endl;
        flag = 1;
    }
    else {
        std::cout << "자 은행 출금에는 수수료 1000원이 발생합니다." << endl;
    }
}

```

```

    }
    while (1) {
        std::cout << "얼마를 출금하시겠습니까?(수수료 제외) (종료하려면 -1을
입력하세요):";
        std::cin >> check_num;
        if (check_num == -1) {
            return;
        }
        std::cout << endl;
        if (check_num > 500000) {//출금 가능한 금액을 초과하였는지 조사
            std::cout << "출금 가능하신 금액을 초과하셨습니다" << endl;
        }
        else if (check_num % 1000 != 0) {
            std::cout << "화폐 최소 단위는 1000원 입니다" << endl;
        }
        else if (get_atm_cash() < check_num) {
            std::cout << "ATM에 충분한 현금이 존재하지 않습니다" << endl;
        }
        else if (!(b->cn_enough_money_in_acc_bank(get_card_insert_slot(), (flag ?
(check_num + 2000) : (check_num + 1000))))) {
            std::cout << "계좌에 돈이 충분하지 않습니다" << endl;
        }
        else {
            int m = check_num;
            fifty_thou = _MIN_((m / 50000), get_50000());
            m = m - 50000 * fifty_thou;
            ten_thou = _MIN_((m / 10000), get_10000());
            m = m - 10000 * ten_thou;
            five_thou = _MIN_((m / 5000), get_5000());
            m = m - 5000 * five_thou;
            one_thou = _MIN_((m / 1000), get_1000());
            m = m - 1000 * one_thou;

            if (m != 0) {
                std::cout << "ATM에 지폐의 개수가 부족합니다" << endl;
            }
            else {
                //atm_cash 조정
                change_atm_cash(-1 * (one_thou), -1 * (five_thou), -1 *
(ten_thou), -1 * (fifty_thou));
                b->withdrawl_bank(get_card_insert_slot(), (flag ?
(check_num + 2000) : (check_num + 1000)));
                std::cout << "은행에서 " << (flag ? (check_num + 2000) :
(check_num + 1000)) << "원이 출금되었고 가져가실 현금은 총 " << check_num << "원 입니다." <<
endl;
                std::cout << "천원 " << one_thou << "장 / " << "오천원 " <<

```

```

five_thou << "장 / " << "만원 " << ten_thou << "장 / " << "오만원 " << fifty_thou << "장" <<
endl;
std::cout << "잔액은 " <<
b->account_get_account(get_card_insert_slot())->get_acc_cash() << endl;
set_number_of_withdrawl(get_number_of_withdrawl() + 1);
transaction_history* th = new
transaction_history(get_card_insert_slot(), "출금", check_num, b,
b->account_get_account(get_card_insert_slot()));
change_temp(th);

b->account_get_account(get_card_insert_slot())->change_account_tr(th);
return;
}
}
}
}

void single_bi_atm::koreandeposit() {
int one_thou = 0;
int five_thou = 0;
int ten_thou = 0;
int fifty_thou = 0;
int check_num = 0;
long long int check_sum = 0;
int withdrawl_count = 0;
bank* b = find_cardnumber_bank(get_card_insert_slot());
if (b != get_primary_bank()) {
    std::cout << "타 은행 입금에는 수수료 1000원이 발생합니다. 수수료를 포함해서
입금하세요" << endl;
}
while (true) {
    std::cout << "천원을 몇 장 넣으시겠습니까? (종료하려면 -1을 입력하세요)";
    std::cin >> one_thou;
    if (one_thou == -1) {
        return;
    }
    std::cout << "오천원을 몇 장 넣으시겠습니까? ";
    std::cin >> five_thou;
    std::cout << "만원을 몇 장 넣으시겠습니까? ";
    std::cin >> ten_thou;
    std::cout << "오만원을 몇 장 넣으시겠습니까? ";
    std::cin >> fifty_thou;
    if (one_thou + five_thou + ten_thou + fifty_thou > 50) {
        std::cout << "가능한 현금 장 수를 초과하셨습니다." << endl;
    }
    else { break; }
}

```

```

while (1) {
    std::cout << "수표를 몇 장 넣으시겠습니까? ";
    std::cin >> check_num;
    std::cout << endl;
    if (check_num > 30) {
        std::cout << "가능한 수표 장 수를 초과하셨습니다." << endl;
    }
    else { break; }
}
while (1) {
    int flag = 0; //flag = 0 일때 정상
    check_sum = 0;
    for (int i = 0; i < check_num; i++) {
        long long int temp = 0;
        std::cout << i + 1 << "번째 수표의 금액은 얼마입니까? ";
        std::cin >> temp;
        if (temp < 100000 || temp > 999999999) { // 100억 이상은 불가
            flag = 1; //flag = 1이면 에러 발생
            break;
        }
        else {
            check_sum += temp;
        }
    }
    if (flag == 1) {
        std::cout << "잘못된 수표 금액입니다." << endl;
    }
    else { break; }
}
change_atm_cash(one_thou, five_thou, ten_thou, fifty_thou);
long long int m = 1000 * one_thou + 5000 * five_thou + 10000 * ten_thou + 50000
* fifty_thou + check_sum;
if (b != get_primary_bank()) { m = m - 1000; }
b->deposit_bank(get_card_insert_slot(), m);
std::cout << m << "원이 입금되었습니다." << endl;
std::cout << "잔액은 " <<
b->account_get_account(get_card_insert_slot())->get_acc_cash() << endl;
transaction_history* th;
th = new transaction_history(get_card_insert_slot(), "입금", m, b,
b->account_get_account(get_card_insert_slot()));
change_temp(th);
b->account_get_account(get_card_insert_slot())->change_account_tr(th);
return;
}

void single_bi_atm::koreaninterface() {

```

```

increase_session();
set_number_of_withdrawl(0);
int password;
int work_type;
int password_count = 0;
int number;//추가

main1:
    std::cout << "카드를 넣어주세요.(카드번호를 입력해주세요). 종료하고 싶으시면 -1을
입력해주세요.";
    std::cin >> number; //변경
    set_card_insert_slot(number);//추가
    if (get_card_insert_slot() == -1) {
        return;
    }
    if (get_card_insert_slot() == 7777777) {
        admin_interface_korean();
        goto end;
    }
    if (find_cardnumber_bank(get_card_insert_slot()) != get_primary_bank()) {
        // 윤현서가 추가한 부분(정세훈 확인필요)
        cout << " 주은행 계좌가 아닙니다." << endl;
        return;
    }
    if (valid_cardnumber_account(get_card_insert_slot()) == 0) {
        cout << "유효하지 않은 카드 번호입니다. " << endl;
        goto main1;
    }
    while (1) {
        if (password_count == 3) {
            break;
        }
        std::cout << "비밀번호를 입력해주세요." << endl;
        std::cin >> password;
        if (check_password(get_card_insert_slot(), password)) {
            break;
        }
        cout << "비밀번호가 틀립니다." << endl;
        password_count++;
    }
    if (password_count == 3) {
        cout << "비밀번호를 3번 틀리셨습니다. " << endl;
        goto end;//세션 end
    }
    while (1) {
        std::cout << "어떤 업무를 하시겠습니까? 번호를 기입하여 주세요." << endl;
        std::cout << "1. 입금 " << "2. 출금 " << "3. 이체 " << "4. 세션 종료 : ";

```

```

        std::cin >> work_type;
        if (work_type == 1) { koreandeposit(); }
        else if (work_type == 2) { koreanwithdrawl(); }
        else if (work_type == 3) { koreantransfer(); }
        else if (work_type == 4) { break; }
        else { std::cout << "잘못된 메뉴를 선택하셨습니다." << endl; }
    }

end:
    cout << "세션을 종료합니다." << endl;
    korean_print_session_history();
    decrease_session();
}

bool check_atm_serial_number(int number) { // atm serial number 사용할 수 있는지 확인하는
    함수
    int size = atmlist.size();
    for (int i = 0; i < size; i++) {
        if (number == atmlist[i]->get_serial_number()) {
            return false; //이미 존재하면 false
        }
    }
    return true; //존재하지않으면 사용할수 있으면 true
}

int main() {
    int number; // 변수
    int size; // 변수
    int banknumber; // 은행 어디 선택했는지.
    string bank_name; // 은행 이름
    string user_name; // 유저 이름
    int initial_money = 0; // 돈 초기화
    int password; //비밀번호
    int type; //타입 번호
    int language_type;// 언어 타입 번호
    int cardnumber; //카드 번호
    int password_count = 0;//비밀번호를 얼마나 틀렸는지.
    long long int accountnumber;
    int count = 0;
    int number_of_atm;
    int test_num;
    int atm_serial_number;

main1: //메인 1 은행만들기 or 은행 선택하기
    std::cout << "어떤 업무를 하시겠습니까? 번호를 기입하여 주세요." << endl;
    std::cout << "1. 은행 만들기" << " " << "2. 은행 선택하기" << " " << "3. 정보 출력하기 :
";
}

```

```

std::cin >> number;
if (number == 1) {
    std::cout << "<은행 만들기>" << endl;
    std::cout << "은행 이름을 기입하여 주세요. 이전 화면으로 돌아가고자 하면
back을 입력해주세요 : ";
    std::cin >> bank_name;
    if (bank_name == "back") { goto main1; } // 이전 화면 돌아가기
    else {
        new bank(bank_name);
        goto main1;
    }
}
else if (number == 2) {
    std::cout << "<은행 선택하기>" << endl;
    std::cout << "가고자 하는 은행 번호를 입력해주세요. 이전 화면으로 돌아가고자
하면 -1을 입력해주세요." << endl;
    size = banklist.size();
    for (int i = 0; i < size; i++) {
        std::cout << i << "." << banklist[i]->get_bankname() << " "; // ex) 0.
    }
    std::cout << endl;
}

sub_main_1_2:
    std::cin >> banknumber; // banklist의 index를 입력 받은 것임.
    if (banknumber == -1) { goto main1; } // 이전 화면 돌아가기
    else if (0 <= banknumber and banknumber < size) {
        goto main2;
    }
    else {
        cout << "잘못된 번호를 입력하셨습니다. 다시 입력해주세요. " << endl;
        goto sub_main_1_2;
    } //
}

else if (number == 3) {
    show_information(); goto main1;
}
else {
    std::cout << "번호를 잘못 입력하셨습니다." << endl;
    goto main1;
}

main2: // 은행 계좌 만들기 or ATM 만들기 or ATM 선택하기
    std::cout << "어떤 업무를 하시겠습니까? 번호를 기입하여 주세요." << endl;
    std::cout << "1. 은행 계좌 만들기 " << "2. ATM 만들기 " << "3. ATM 이용하기 " << "4.
정보 출력하기 " << "5. 계좌 정보 접근" << "6. 이전 화면으로 돌아가기 : ";
    std::cin >> number;
    if (number == 1) {

```

```

sub_main2_0://은행 계좌 만들기
    std::cout << "<은행 계좌 만들기>" << endl;
    std::cout << "당신의 이름을 입력해주세요 : ";
    std::cin >> user_name;
    std::cout << "계좌번호를 입력해주세요 : ";
    std::cin >> accountnumber;
    if ((accountnumber < 9999999999) or (accountnumber > 100000000000)) {
        cout << "계좌번호 양식이 잘못되었습니다. 12자리로 입력해주세요." <<
    endl;
        goto sub_main2_0;
    }
    for (int i = 0; i < banklist.size(); i++) {
        if (banklist[i]->bool_accountnumber_in_bank(accountnumber) ==
true) {
            cout << "이미 해당 계좌번호가 존재합니다! " << endl;
            goto sub_main2_0;
        }
    }
    //은행은 banklist[banknumber] 임.
    std::cout << "계좌에 가능한 돈이 얼마인지를 입력해주세요 : ";
    std::cin >> initial_money;
    std::cout << "계좌의 비밀번호는 무엇으로 할지 입력해주세요 : ";
    std::cin >> password;
    banklist[banknumber]->make_account(accountnumber, user_name,
password, initial_money);
    goto main2;
}
else if (number == 2) { //ATM 만들기
    std::cout << "<ATM 만들기>" << endl;
sub_main_2_1:
    std::cout << "ATM의 SERIAL NUMBER을 입력해주세요 ";
    cin >> atm_serial_number;
    if (atm_serial_number < 100000 or atm_serial_number>999999) {
        cout << "잘못된 ATM SERIAL NUMBER의 유형입니다. 6자리로
입력해주세요." << endl;
        goto sub_main_2_1;
    }
    if (!check_atm_serial_number(atm_serial_number)) {
        cout << "이미 존재하는 ATM SERIAL NUMBER 입니다. 다시
입력해주세요." << endl;
        goto sub_main_2_1;
    }
sub_main_2_2:
    std::cout << "ATM type에 해당하는 번호를 입력해주세요" << endl;
    std::cout << "0. Single Bank ATM ";
    std::cout << "1. Multi-Bank ATM ";

```

```

        std::cin >> type;
        if (type != 1 and type != 0) {
            cout << "잘못된 번호입니다." << endl;
            goto sub_main_2_2;
        }
    sub_main_2_3:
        std::cout << "ATM의 언어 지원 여부에 해당하는 번호를 입력해주세요" << endl;
        std::cout << "0. Unilingual ";
        std::cout << "1. Bilingual ";
        std::cin >> language_type;
        if (language_type != 1 and language_type != 0) {
            cout << "잘못된 번호입니다." << endl;
            goto sub_main_2_3;
        }
        std::cout << "돈은 얼마나 들어있게 할까요?" << endl;
        int one_thou = 0;
        int five_thou = 0;
        int ten_thou = 0;
        int fifty_thou = 0;
        int check_num = 0;
        std::cout << "천원을 몇 장 넣으시겠습니까? ";
        std::cin >> one_thou;
        std::cout << "오천원을 몇 장 넣으시겠습니까? ";
        std::cin >> five_thou;
        std::cout << "만원을 몇 장 넣으시겠습니까? ";
        std::cin >> ten_thou;
        std::cout << "오만원을 몇 장 넣으시겠습니까? ";
        std::cin >> fifty_thou;
        if (type == 1 && language_type == 1) //multi_bi
            banklist[banknumber]->make_multi_bi_atm(1, 1, one_thou,
five_thou, ten_thou, fifty_thou, atm_serial_number);
        else if (type == 1 && language_type == 0) //multi_uni
            banklist[banknumber]->make_multi_uni_atm(1, 0, one_thou,
five_thou, ten_thou, fifty_thou, atm_serial_number);
        else if (type == 0 && language_type == 1) //single_bi
            banklist[banknumber]->make_single_bi_atm(0, 1, one_thou,
five_thou, ten_thou, fifty_thou, atm_serial_number);
        else if (type == 0 && language_type == 0) //single_uni
            banklist[banknumber]->make_single_uni_atm(0, 0, one_thou,
five_thou, ten_thou, fifty_thou, atm_serial_number);
        goto main2;
    }
else if (number == 3) {// ATM 선택하기
    // atm 선택하기. 어떤 atm을 선택하시겠습니까? 과정 필요.
    sub_main2:
        std::cout << "<ATM 선택하기>" << endl;

```

```

        std::cout << "선택하고 싶은 atm의 번호를 기입해주세요. 이전으로 가고 싶으시면
-1을 입력해주세요." << endl;
        size = banklist[banknumber]->get_atm_list().size();
        for (int i = 0; i < size; i++)
        {
            cout << i << ". atm serial number : " <<
banklist[banknumber]->get_atm_list()[i]->get_serial_number() << endl;
        }
        cin >> number_of_atm;
        if (number_of_atm == -1) {
            goto main2;
        }

        if (number_of_atm >= 0 and size > number_of_atm) {
            goto main3;
        }
        else {
            cout << "잘못된 ATM 번호입니다. 다시 입력해주세요." << endl;
            goto sub_main2;
        }

    }
else if (number == 4) {
    show_information(); goto main2;
}
else if (number == 5) {
    std::cout << "계좌번호를 입력해주세요 : ";
    std::cin >> accountnumber;
    if (banklist[banknumber]->bool_accountnumber_in_bank(accountnumber)) {

banklist[banknumber]->account_get_accountnum(accountnumber)->korean_print_account_hist
ory();
    }
    else {
        cout << "계좌번호가 은행에 존재하지 않습니다." << endl;
    }
    goto main2;
}
else if (number == 6) {
    goto main1;
}
else {
    cout << "번호를 잘못 입력하셨습니다." << endl;
    goto main2;
}

```

```
main3://ATM 이용하기
    (banklist[banknumber]->get_atm_list())[number_of_atm]->mini_interface();
    goto main1;
}
```

Member student contribution table

강승수	33.34%
윤현서	33.33%
정세훈	33.33%