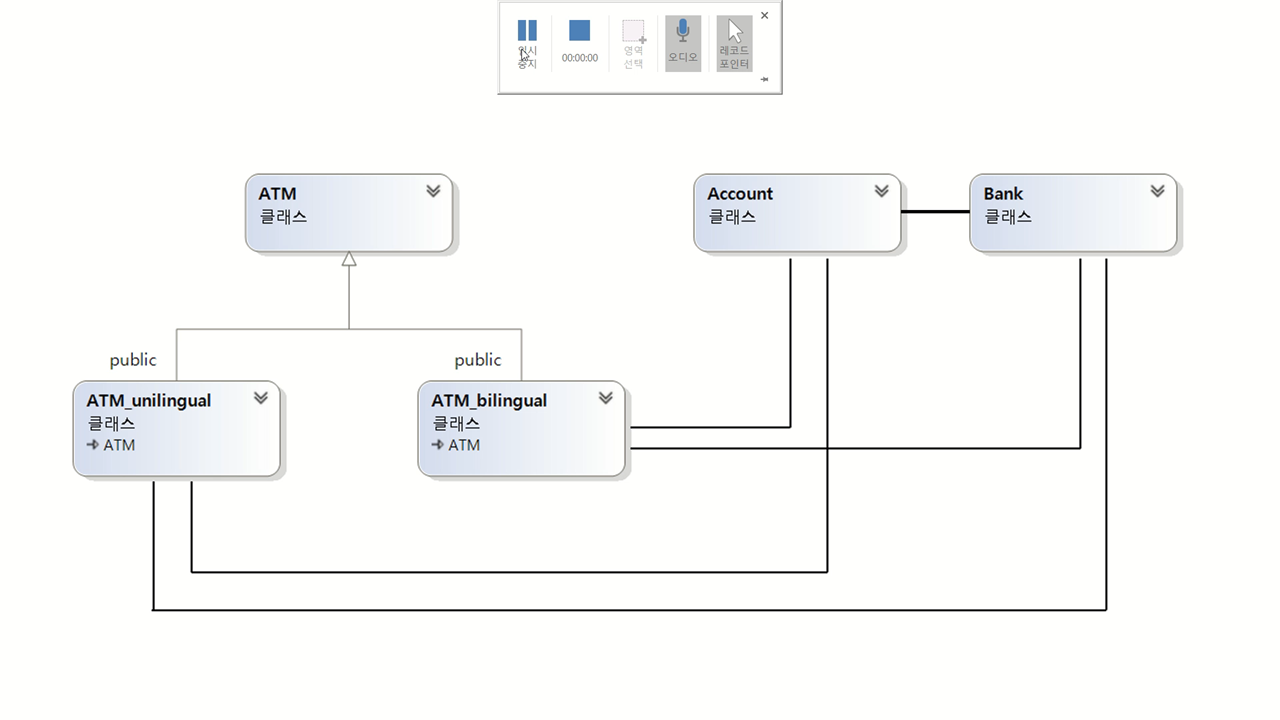
CSE201 Term Project

-Automated Teller Machine(ATM) System-

201911099 오선무, 202011023 김나영 ,202011212 진호진, 202011213 천규빈

1. Final Class Diagram

각 클래스에 대한 UML 다이어그램을 압축된 형태로 표현한 그림이다. ATM\_unilingual과 ATM\_bilingual 은 각각 ATM class를 상속받고 있다. 그리고 Account, Bank 클래스는 ATM\_bilingual, ATM\_unilingual과 association 관계를 맺고 있다.



아래는 각 클래스에 포함된 method, variable에 대한 정보를 나타내고 있다.

테이블이(가) 표시된 사진

자동 생성된 설명

1. List of requirements that have been successfully implemented

|  |  |
| --- | --- |
| Requirement | 구현여부 |
| 1.1 | O |
| 1.2 | O |
| 1.3 | O |
| 1.4 | O |
| 1.5 | O |
| 1.6 | O |
| 1.7 | O |
| 1.8 | O |
| 1.9 | O |
| 2.1 | O |
| 2.2 | O |
| 2.3 | O |
| 2.4 | O |
| 3.1 | O |
| 3.2 | O |
| 3.3 | O |
| 3.4 | O |
| 3.5 | O |
| 4.1 | O |
| 4,2 | O |
| 4.3 | O |
| 4.4 | O |
| 4.5 | O |
| 4.6 | O |
| 5.1 | O |
| 5.2 | O |
| 5.3 | O |
| 5.4 | O |
| 5.5 | O |
| 6.1 | O |
| 6.2 | O |
| 6.3 | O |
| 6.4 | O |
| 6.5 | O |
| 6.6 | O |
| 6.7 | O |
| 7.1 | O |
| 7.2 | O |
| 7.3 | O |
| 8.1 | O |
| 8.2 | O |

1. Console screenshots to demonstrate the successfully implemented

|  |  |  |
| --- | --- | --- |
|  | Requirement  1.1 | 1.1 ->  Main에서 ATM a1,a2 에 대한 Serial number가 생성되는 것을 확인할 수 있다. |
| 텍스트이(가) 표시된 사진  자동 생성된 설명   * Multi ATM에서 primary bank account 사용   텍스트이(가) 표시된 사진  자동 생성된 설명   * Multi ATM에서 primary bank를 제외한 account 사용   텍스트이(가) 표시된 사진  자동 생성된 설명  -Single ATM에서 primary bank를 제외한 account 사용  텍스트이(가) 표시된 사진  자동 생성된 설명  -Single ATM에서 primary bank card로 거래를 하려는 경우 | 1.2,  3.1,  3.3,  8.1,  8.2 | 1.2- >  Single ATM에서 Primary Bank를 제외한 나머지 Bank Card를 이용한 거래는 허용되지 않음을 확인할 수 있었다.  Multi ATM에서는 모든 경우가 가능함을 확인할 수 있었다.  3.1- >올바른 카드가 입력되면 다음 단계로 진행이 된다.  3.3->Password를 물어보고 이것이 옳은지 아닌지 확인한다.  8.1->Choose language에서 0,1를 고르면서 언어를 선택할 수 있다.  8.2-> 언어를 영어로 선택한 후, 영어로 거래가 진행된다. |
| 텍스트이(가) 표시된 사진  자동 생성된 설명  -Bilingual에서 한국어 기능을 선택한 경우  텍스트이(가) 표시된 사진  자동 생성된 설명  -Unilingual인 경우 | 1.3 | 1.3 -> Unilingual과 Bilingual에서 출력결과가 어떻게 나타나는지 확인할 수 있었다. |
| * Main에서 생성한 Account 정보     -Bank에서 출력한 Account 정보 | 1.5 | 1.5-> Bank에 포함되어 있는 Account 정보를 열람할 수 있다. |
| -Main 함수    -1.6, 1.7 출력결과 | 1.6  1.7 | 1.6,1.7-> 사용자가 동일한 은행에서 여러개의 계좌를 개설할 수 있고, 여러 은행에서 계좌를 생성할 수 있다. |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 2.3  2.4 | 2.3,2.4.-> 거래 완료시 transaction에 대한 정보가 표시되는 것과 각각의 거래를 분별하기 위한 Transaction ID가 출력되는 것을 확인할 수 있다. |
| -Main 함수  텍스트이(가) 표시된 사진  자동 생성된 설명  -출력 결과 | 3.2 | 3.2-> 잘못된 카드가 입력되는 경우 invalid atm use라는 메세지를 출력한다. |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 3.4  3.5 | 3.4-> 잘못된 password가 입력되면 invalid password라는 문구가 출력된다.  3.5-> 3번 틀릴 때 Program이 종료된다. |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 1.4  2.1  2.2  4.1  4.2  9.1 | 1.4-> ATM 잔액보다 많은 금액을 넣어서 error가 발생하였다.  2.1-> 카드를 넣어야 세션이 시작된다.  2.2-> ATM 잔액보다 많은 금액을 넣어서 error가 발생하여서 종료되었다.  4.1-> cash와 check 모두 받는다.  4.2-> ATM 잔액보다 많은 금액을 넣어서 error가 발생하였다.  9.1-> 예외 상황에서 “limit error” message와 session 종료 |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 1.8  4.3  4.4  4.5 | 1.8,4.4-> 수수료 500원이 적용되었다  4.3,-> account balance가 적절하게 바뀌었다.  4.5-> cash로 넣는 경우 ATM 잔액을 증가시킨다. |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 4.6 | 4.6->cash를 넣으면 ATM balance가 변하지 않는다. |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 5.1  5.3  5.4  5.5 | 5.1->인출 금액을 “How much money to withdrawl?”로 물어본다.  5.3-> withdrawl을 5000원 만큼 했을 때, 5000원과 수수료 1000원이 account에서 출금되고 ATM에서 5000원만큼 감소하고 |
| 텍스트이(가) 표시된 사진  자동 생성된 설명  텍스트이(가) 표시된 사진  자동 생성된 설명 | 5.2 | 5.2-> account의 돈이 입력한 값보다 적어서 “Money is too low”라고 출력된다. ATM돈이 입력된 돈보다 적어서 Money is too low라고 말해준다. |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 6.1  6.2  6.4  6.5  6.7 | 6.1 -> transfer를 선택한 후 송금할 내 계좌의 잔액과 지금 쓰는 ATM의 잔액을 보여준 후 어떤 타입의 송금(cash, account)을 할지 보여준다.  6.2 -> “Account Number of recipient “ 라는 문장으로 송금 될 계좌의 정보를 묻는다.  6.4 -> “ Source Account “ , “ Money “로 송금 할 계좌의 정보와 송금될 금액을 확인한다.  6.5, 6.7 -> “After my account balance” 를 보면 20000원을 송금한 후 내 계좌의 돈이 -20000, -2000(primary bank인 Kookmin 과 non primary인 대구의 수수료)원이 된 128000원이 된 것을 알 수 있다.  6.7 - > “Before transfer balance”는 송금 될 계좌의 이전 잔액 50000원, after는 송금 된 계좌의 송금 후 잔액 70000원을 보여준다. |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 6.3  6.6 | 6.3  > ATM에서 송긂할 account number을 물어보고 “Check if Bank account and Amount of Money is right?”로 물어본 뒤, Y/N로 정보가 정확한지 확인한다.  6.6 - > “After ATM balance”는 cash로 송금했으니 ATM 자체의 잔액이 20000원 늘어났음을 잘 보여준다. |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 1.9  7.1  7.2 | 1,9,7.1,7.2 -> Admin 카드 넘버인 0을 입력한 후 “Open Transaction History” Menu를 display 하고, Y 하면 이전 거래 내역 (transaction history)를 다 Display 한다.  우리는 프로그램(세션) 종료 후 새로운 거래를 시작하기 전까지 그 전 프로그램(세션) 정보를 저장하고 있는다. |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 7.3 | 7.3 -> transaction\_info.txt를 만들어 transaction history를 잘 저장하고 있음을 보여준다. |

1. The list of concepts of object-oriented programming

Encapsulation

ATM을 통해 여러 종류의 거래를 하기 위해 encapsulation을 이용하였다. ATM 클래스, Account 클래스, Bank 클래스, ATM\_unilingual 클래스, ATM\_bilingual 클래스로 5개의 클래스를 생성하고 이용하였다.

대표적으로 encapsulation 실행한 것은 Account 클래스이고 코드는 다음과 같다.

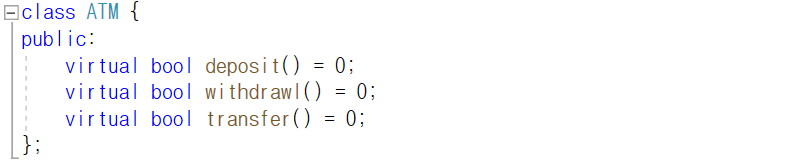
텍스트이(가) 표시된 사진

자동 생성된 설명

Abstract 클래스

Interface 클래스인 ATM class를 상위 클래스로 사용하였다.

Abstract 클래스의 종류 중 하나인 Interface 클래스를 사용하였으므로 Abstract 클래스를 사용하였다. 또한 총 5개의 클래스를 사용하였으므로 Abstract 클래스를 만들었다고 이야기할 수 있다.



Inheritance

영어만 사용할 수 있는 ATM인 ATM\_unilingual 클래스와 한국어와 영어를 둘 다 사용할 수 있는 ATM\_bilingual 클래스를 생성하였다.

이 두 클래스는 ATM 클래스에 inheritance되어 있어 ATM 클래스의 하위 클래스를 구성하고 있다.

대표적으로 inheritance되어 있는 ATM\_unilingual 클래스는 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Polymorphism

Account 클래스의 constuctor을 두 종류로 만들었다. 각각은 변수로 아무것도 받지 않는 constuctor와 string type 2개, int type 4개의 변수를 받는 constuctor이다. 이 두 constuctor는 함수의 이름이 같지만 받는 변수가 다르므로 polymorphism 이라고 할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

ATM\_unilingual 클래스를 통해서 overriding을 진행했다. ATM에서 virtual로 정의한 함수인 withdrawl, deposit, transfer에 대해 overriding 하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Exception handling

if와 else를 통해 error가 발생하는 상황을 조정해 주었다. 이 과정을 통해 exception handling을 잘 수행한 것을 알 수 있다.

대표적으로 deposit 함수에서 single atm에 primary bank의 카드가 아닌 다른 카드가 들어왔을 때 error가 발생하는데 이를 if 문을 이용해서 error 처리를 하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

STL

Bank class, ATM\_unilingual, ATM\_bilingual의 경우 account 정보를 stl을 통해 받았고 이를 통해서 account의 정보를 변경/이용하였다.

대표적으로 Bank class는 자신의 은행 이름으로 만들어진 Account들을 stl로 받았는데 이 코드는 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. Instruction to run the source code

현재 은행은 2개 (대구, 국민), 계좌는 4개 (각 은행 별 2개씩) 은행과 계좌의 한도는 int가 선언할 수 있는 최대 숫자만큼으로 설정하였다.

1) 언어 선택 – 한국어, 영어

2) 카드 번호 입력 (나의 카드 – 1,2,3,4가 있음)

3) 카드 비밀 번호 입력 (카드 번호, 계좌번호, 비밀번호 다 같음)

4) 거래 type 선택

- 입금 : 입금 종류(cash, check), 금액, paper number 입력

- 출금 : 출금 금액 입력

- 송금 :

송금 종류 입력 (그 후 입력해야 할 것들)

1) Account

a) 송금 금액

b) 송금 수신인 계좌번호

c) 송금 수신인 은행

d) 송금 정보 확인 (N,Y)

2) Cash

a) 송금 금액

b) 송금 수신인 계좌번호

c) 송금 수신인 은행

d) 송금 할 paper number

e) 송금 정보 확인 (N,Y)

- Exit ( session 종료 후 summary 확인 가능)

5) Admin mode

- Card number 0(admin card number) 입력

- Transaction History open (Y,N) 입력

1. The final version of source code

#include <iostream>

#include <vector>

#include <cstdlib>

#include <ctime>

#include <fstream>

#include <string>

#include <sstream>

#include <random>

**using** **namespace** std;

**class** Account {

**private**:

string name;

**int** password;

**int** account\_number;

**int** card\_number;

**int** balance;

string bankname;

**public**:

Account();

Account(string name, **int** password, **int** account\_number, **int** card\_number, **int** balance, string bankname);

string get\_name();

**int** get\_password();

**int** get\_account\_number();

**int** get\_card\_number();

**int** get\_balance();

string get\_bank\_name();

**int** change\_balance(**int** money);

};

**class** Bank {

**private**:

vector<Account\*> account\_stl; // Vector type account information

**int** bank\_balance;

string bankname;

**public**:

string get\_bankname();

Bank(vector<Account\*>, **int**, string);

vector<Account\*> get\_account\_stl();

**int** get\_bankbalance();

**int** change\_balance(**int** money);

**void** get\_account\_info(**int** i){

cout<<"User name: "<<account\_stl[i]->get\_name()<<" Account number: "<<account\_stl[i]->get\_account\_number()<<" Available fund: "<<account\_stl[i]->get\_balance()<<endl;

}

};

**class** ATM{

**public**:

**virtual** **bool** deposit()=0;

**virtual** **bool** withdrawl()=0;

**virtual** **bool** transfer()=0;

};

**class** ATM\_unilingual:**public** ATM{

**private**:

**int** balance;

**int** limit\_check = 30;

**int** limit\_cash = 50;

**int** error\_count = 0;

**int** admin\_card;

string trsac\_info;

**int** transaction\_id;

vector<Account\*> Account\_array[2];

Account\* newacc;

Account\* reciacc;

**int** cmp\_cardnum;

**public**:

Bank\* bank;

**int** serialnumber;

string atm\_type;

//int select\_language;

**int** max\_transaction = 1000000;

ATM\_unilingual(Bank\* primary\_bank, vector<Account\*> \*Account\_arrays, string type, **int** admin, **int** atm\_balance);

**int** check\_account(); // 0 (admin) 1 (존재) 2(존재X)

**bool** check\_password(); //

**bool** check\_bank(); // true (사용 가능) false ( 사용 불가능 )

**int** change\_balance(**int** money);

**int** get\_atm\_balance();

**int** get\_admin\_card();

**bool** deposit();

**bool** withdrawl();

**bool** transfer();

**bool** stop(**bool** check);

**void** use();

};

**class** ATM\_bilingual:**public** ATM{

**private**:

**int** balance;

**int** limit\_check = 30;

**int** limit\_cash = 50;

**int** error\_count = 0;

**int** admin\_card;

string trsac\_info;

**int** transaction\_id;

vector<Account\*> Account\_array[2];

Account\* newacc;

Account\* reciacc;

**int** cmp\_cardnum;

**public**:

Bank\* bank;

**int** serialnumber;

string atm\_type;

**int** select\_language;

**int** max\_transaction = 1000000;

ATM\_bilingual(Bank\* primary\_bank, vector<Account\*> \*Account\_arrays, string type, **int** admin, **int** atm\_balance);

**int** check\_account(); // 0 (admin) 1 (존재) 2(존재X)

**bool** check\_password(); //

**bool** check\_bank(); // true (사용 가능) false ( 사용 불가능 )

**int** change\_balance(**int** money);

**int** get\_atm\_balance();

**int** get\_admin\_card();

**bool** deposit();

**bool** withdrawl();

**bool** transfer();

**bool** stop(**bool** check);

**void** use();

};

//Account

Account::Account(string name, **int** password, **int** account\_number, **int** card\_number, **int** balance, string bankname) {

**this**->name = name;

**this**->password = password;

**this**->account\_number = account\_number;

**this**->card\_number = card\_number;

**this**->balance = balance;

**this**->bankname = bankname;

}

Account::Account() {

}

**int** Account::change\_balance(**int** money) {

balance = balance + money;

**return** balance;

}

string Account::get\_name() {

**return** name;

}

**int** Account::get\_password() {

**return** password;

}

**int** Account::get\_account\_number() {

**return** account\_number;

}

**int** Account::get\_card\_number() {

**return** card\_number;

}

**int** Account::get\_balance() {

**return** balance;

}

string Account::get\_bank\_name() {

**return** bankname;

}

//Bank

Bank::Bank(vector<Account\*> a, **int** b, string c) {

**for** (**int** i = 0; i < a.size(); i++)

{

Account\* a1 = a.at(i);

**this**->account\_stl.push\_back(a1);

}

**this**->bank\_balance = b;

**this**->bankname = c;

}

vector<Account\*> Bank::get\_account\_stl() {

**return** account\_stl;

}

**int** Bank::get\_bankbalance() {

**return** bank\_balance;

}

string Bank::get\_bankname() {

**return** bankname;

}

**int** Bank::change\_balance(**int** money) {

bank\_balance = bank\_balance + money;

**return** 0;

}

// ATM\_unilingual

ATM\_unilingual::ATM\_unilingual(Bank\* primary\_bank1, vector<Account\*> \*Account\_arrays, string type, **int** admin, **int** atm\_balance){

**for**( **int** i = 0; i < 2; i++)

{

Account\_array[i] = Account\_arrays[i];

}

transaction\_id = 0;

bank = primary\_bank1;

atm\_type = type;

admin\_card = admin;

balance = atm\_balance;

serialnumber = rand();

cout<<"Serial number:"<<serialnumber<<endl;

}

**int** ATM\_unilingual::check\_account(){

**int** input\_cardnumber;

**char** select;

cout << "Write the card number" << endl;

cin >> input\_cardnumber;

cmp\_cardnum = input\_cardnumber;

**if** (input\_cardnumber == admin\_card) {

cout << "Open Transaction history Y/N" << endl;

cin >> select;

**if**(select == 'Y'){

ifstream readFile;

readFile.open("transaction\_info.txt");

**if** (readFile.is\_open())

{

**while** (!readFile.eof())

{

string str;

getline(readFile, str);

cout << str << endl; //지금은 읽은 문자열 바로 출력.

}

readFile.close(); //파일 닫아줍니다.

}

}

**else** **if**(select == 'N'){

cout << "exit" << endl;

}

**else**{

cout << "invalid selection(select between Y/N)" << endl;

}

**return** 0;

}

**for**( **int** i = 0; i < 2; i++)

{

vector<Account\*> vector\_account = Account\_array[i];

**for** (**int** j = 0; j < 2 ; j++)

{

**if** (vector\_account.at(j)->get\_card\_number() == input\_cardnumber)

{

newacc = vector\_account.at(j);

**return** 1;

}

}

}

cout << "invalid card" << endl;

**return** 2;

}

**bool** ATM\_unilingual::check\_password(){

**int** input\_password;

**while**( error\_count < 3){

cout << "Write password" << endl;

cin >> input\_password;

**if** (newacc->get\_password() == input\_password) {

cout << "valid password" << endl;

**return** **true**;

}

**else** {

error\_count = error\_count + 1;

cout << "invalid password" << endl;

}

}

**return** **false**;

}

**bool** ATM\_unilingual::check\_bank(){

**if** (newacc->get\_bank\_name() == bank->get\_bankname()) {

cout << "main bank" << endl;

**return** **true**;

}

**else** {

**if** (atm\_type == "Single") {

cout << "invalid atm use" << endl;

**return** **false**;

}

**else** {

**return** **true**;

}

}

}

**int** ATM\_unilingual::change\_balance(**int** money){

balance = balance + money;

**return** balance;

}

**int** ATM\_unilingual::get\_atm\_balance(){

**return** balance;

}

**int** ATM\_unilingual::get\_admin\_card(){

**return** admin\_card;

}

**bool** ATM\_unilingual::deposit(){

string money\_type;

**int** money;

**int** money\_number;

**bool** returnname = **true**;

cout << "What type of money are you depoisting? \nHow much money are you depositing? \nHow many cash or checks are you depositing?" << endl;

cin >> money\_type;

cin >> money;

cin >> money\_number;

**int** transition\_fee = 0;

**if** (bank->get\_bankname() == newacc->get\_bank\_name()) {

transition\_fee = 0;

}

**else** **if** (atm\_type == "single") {

cout << "This atm is single." << endl;

returnname = **false**;

}

**else** {

transition\_fee = 500;

}

**if**( returnname == **true**)

{

**if** (money < transition\_fee)

{

cout << "Money is too low." << endl;

returnname = **false**;

}

}

**if**( returnname == **true**)

{

**if** (money\_type == "check"){

**if** (money\_number > limit\_check)

{

cout << "There are too many checks" << endl;

returnname = **false**;

}

**else**

{

newacc->change\_balance(money-transition\_fee);

}

}

**else**{

**if** (money\_number > limit\_cash)

{

cout << "There are too many cashs" << endl;

returnname = **false**;

}

**else**

{

change\_balance(money);//For ATM

newacc->change\_balance(money-transition\_fee);

}

}

}

ofstream writeFile;

**if**(transaction\_id == 0){

writeFile.open("transaction\_info.txt",ios::out);

}

**else**{

writeFile.open("transaction\_info.txt",ios::app);

}

transaction\_id++;

trsac\_info = "transsaction id : " + to\_string(transaction\_id) + ", card number : " + to\_string(newacc -> get\_card\_number()) + ", Transaction Type : deposit, Amount : " + to\_string(money) + '\n';

writeFile.write(trsac\_info.c\_str(), trsac\_info.size());

writeFile.close();

**return** returnname;

}

**bool** ATM\_unilingual::withdrawl(){

**int** money;

**bool** returnname = **true**;

cout << "How much money to withdrawl?" << endl;

cin >> money;

**int** transition\_fee = 0;

**if** (bank->get\_bankname() == newacc->get\_bank\_name()) {

transition\_fee = 500;

}

**else** **if** (atm\_type == "single") {

cout << "This atm is single." << endl;

returnname = **false**;

}

**else** {

transition\_fee = 1000;

}

**if**(returnname == **true**){

**if** (money < transition\_fee)

{

cout << "Money is too low." << endl;

returnname = **false**;

}

}

**if**(returnname == **true**){

**if** (balance > money && newacc->get\_balance() > money + transition\_fee && max\_transaction > money)

{

newacc->change\_balance(-(money + transition\_fee));

change\_balance(-money);//For ATM

}

**else**

{

cout << "Money is too low" << endl;

returnname = **false**;

}

}

ofstream writeFile;

**if**(transaction\_id == 0){

writeFile.open("transaction\_info.txt",ios::out);

}

**else**{

writeFile.open("transaction\_info.txt",ios::app);

}

transaction\_id++;

trsac\_info = "transsaction id : " + to\_string(transaction\_id) + ", card number : " + to\_string(newacc -> get\_card\_number()) + ", Transaction Type : withdrawl, Amount : " + to\_string(money) + '\n';

writeFile.write(trsac\_info.c\_str(), trsac\_info.size());

writeFile.close();

**return** returnname;

}

**bool** ATM\_unilingual::transfer(){

**bool** returnname = **true**;

**int** select\_1;

**int** transfer\_money;

**int** Account\_Number;

**int** Cash\_number;

**char** Check\_transfer;

string Bank\_name;

**int** Transition\_fee = 0;

cout << "1. Transfer from your account\n 2. Transfer with cash" << endl;

cin >> select\_1;

cout << "How much money you will transfer?\n" << endl;

cin >> transfer\_money;

cout << "Account Number of recipient" << endl;

cin >> Account\_Number;

cout << "Bank Name of recipient" << endl;

cin >> Bank\_name;

**for**( **int** i = 0; i < 2; i++)

{

vector<Account\*> vector\_account = Account\_array[i];

**for** (**int** j = 0; j < 2; j++)

{

**if** (vector\_account.at(j)->get\_bank\_name() == Bank\_name && vector\_account.at(j)->get\_account\_number() == Account\_Number)

{

reciacc = vector\_account.at(j);

}

}

}

**if** (reciacc == **NULL**) {

cout << "wrong account information" << endl;

exit(-1);

}

**if**(select\_1 == 1){

cout << "Check if Bank account and Amount of Money is right\n Source Bank name:"<<newacc -> get\_bank\_name() << "\nSource Bank account:" << newacc -> get\_account\_number() <<"\nMoney:"<< transfer\_money<<endl;

cout << "Y/N" << endl;

cin >> Check\_transfer;

**if** (Check\_transfer == 'Y'){

newacc ->change\_balance(-transfer\_money);

}

**else**{

returnname = **false**;

}

}

**else** **if**(select\_1 == 2){

cout << "How many number of cash are you transfering?" << endl;

cin >> Cash\_number;

**if**( Cash\_number > 50){

cout << "Cash number error" << endl;

returnname = **false**;

}

**else**{

cout << "Check if Bank account and Amount of Money is right \n Bank name:"<<Bank\_name<< "\nBank account:" << Account\_Number <<"\nMoney:"<< transfer\_money<<endl;

cout << "Y/N" << endl;

cin >> Check\_transfer;

**if** (Check\_transfer == 'Y'){

change\_balance(transfer\_money);

}

**else**{

returnname = **false**;

}

}

}

**else**{

cout << "key error" << endl;

returnname = **false**;

}

cout << "Before transfer balance: " << reciacc->get\_balance() << endl;

reciacc -> change\_balance(transfer\_money);

**if**(newacc -> get\_bank\_name() == reciacc ->get\_bank\_name()){

**if**(check\_bank()){

Transition\_fee = 1500;

}

**else**{

Transition\_fee = 2500;

}

}

**else**{

Transition\_fee = 2000;

}

newacc ->change\_balance(-Transition\_fee);

ofstream writeFile;

**if**(transaction\_id == 0){

writeFile.open("transaction\_info.txt",ios::out);

}

**else**{

writeFile.open("transaction\_info.txt",ios::app);

}

transaction\_id++;

trsac\_info = "transaction id : " + to\_string(transaction\_id) + ", card number : " + to\_string(newacc -> get\_card\_number()) + ", Transaction Type : withdrawl, Amount : " + to\_string(transfer\_money) + " recipient account number : " + to\_string(reciacc->get\_account\_number()) + '\n';

writeFile.write(trsac\_info.c\_str(), trsac\_info.size());

writeFile.close();

**return** returnname;

}

**bool** ATM\_unilingual::stop(**bool** check)

{

**if**(check) {

**return** **true**;

}

**else** {

**if**( cmp\_cardnum != admin\_card ){

ifstream readFile;

readFile.open("transaction\_info.txt");

**if** (readFile.is\_open())

{

**while** (!readFile.eof())

{

string str;

getline(readFile, str);

cout << str << endl; //지금은 읽은 문자열 바로 출력.

}

readFile.close(); //파일 닫아줍니다.

}

}

exit(-1);

}

}

**void** ATM\_unilingual::use()

{

/\*

cout << "Chose language : (0) Korea (1) English" << endl;

cin >> select\_language;

\*/

**int** select;

**bool** check;

cout << "Welcome" << endl;

cout << "If you want to stop the session press Ctrl+C" << endl;

cout << "To start, please inset your debit card" << endl;

**int** a = check\_account(); // 0 관리자 1 정상 2 비정상

**if** (a == 1 ) { check = **true**; }

**else**{ check = **false**; }

stop(check);

check = check\_bank();

stop(check);

check = check\_password();

stop(check);

**while** (**true**)

{

cout << " Select: (1)Deposit (2)Withdrawl (3) Transfer (4) Exit" << endl;

cin >> select;

**if** (select == 1)

{

cout << "Before ATM balance: " << get\_atm\_balance() << endl;

cout << "After my account balance: " << newacc->get\_balance() << endl;

check = deposit();

cout << "Before ATM balance: " << get\_atm\_balance() << endl;

cout << "After my account balance: " << newacc->get\_balance() << endl;

stop(check);

}

**else** **if** (select == 2)

{

cout << "Before ATM balance: " << get\_atm\_balance() << endl;

cout << "Before my account balance: " << newacc->get\_balance() << endl;

check = withdrawl();

cout << "After ATM balance: " << get\_atm\_balance() << endl;

cout << "After my account balance: " << newacc->get\_balance() << endl;

stop(check);

}

**else** **if**(select==3)

{

cout << "Before ATM balance: " << get\_atm\_balance() << endl;

cout << "Before my account balance: " << newacc->get\_balance() << endl;

check = transfer();

cout << "After ATM balance: " << get\_atm\_balance() << endl;

cout << "After my account balance: " << newacc->get\_balance() << endl;

cout << "After transfer balance: " << reciacc->get\_balance() << endl;

stop(check);;

}

**else** **if**(select==4){

stop(**false**);

}

}

ifstream readFile;

readFile.open("transaction\_info.txt");

**if** (readFile.is\_open())

{

**while** (!readFile.eof())

{

string str;

getline(readFile, str);

cout << str << endl; //지금은 읽은 문자열 바로 출력.

}

readFile.close(); //파일 닫아줍니다.

}

}

//ATM bilingual

// ATM

ATM\_bilingual::ATM\_bilingual(Bank\* primary\_bank1, vector<Account\*> \*Account\_arrays, string type, **int** admin, **int** atm\_balance){

**for**( **int** i = 0; i < 2; i++)

{

Account\_array[i] = Account\_arrays[i];

}

transaction\_id = 0;

bank = primary\_bank1;

atm\_type = type;

admin\_card = admin;

balance = atm\_balance;

serialnumber = rand();

cout<<"Serial number:"<<serialnumber<<endl;

}

**int** ATM\_bilingual::check\_account(){

**int** input\_cardnumber;

**char** select;

**if**(select\_language){

cout << "Write the card number" << endl;

}

**else**{

cout<<"카드 번호를 입력하세요"<<endl;

}

cin >> input\_cardnumber;

cmp\_cardnum = input\_cardnumber;

**if** (input\_cardnumber == admin\_card) {

**if**(select\_language){

cout << "Open Transaction history Y/N" << endl;

}

**else**{

cout <<"거래 내역을 여시겠습니까? Y/N"<<endl;

} cin >> select;

**if**(select == 'Y'){

ifstream readFile;

readFile.open("transaction\_info.txt");

**if** (readFile.is\_open())

{

**while** (!readFile.eof())

{

string str;

getline(readFile, str);

cout << str << endl; //지금은 읽은 문자열 바로 출력.

}

readFile.close(); //파일 닫아줍니다.

}

}

**else** **if**(select == 'N'){

cout << "exit" << endl;

}

**else**{

**if**(select\_language){

cout << "invalid selection(select between Y/N)" << endl;

}

**else**{

cout<<"허용되지 않은 선택입니다."<<endl;

}

}

**return** 0;

}

**for**( **int** i = 0; i < 2; i++)

{

vector<Account\*> vector\_account = Account\_array[i];

**for** (**int** j = 0; j < 2 ; j++)

{

**if** (vector\_account.at(j)->get\_card\_number() == input\_cardnumber)

{

newacc = vector\_account.at(j);

**return** 1;

}

}

}

**if**(select\_language){

cout << "invalid card" << endl;

}

**else**{

cout<<"허용되지 않은 카드입니다."<<endl;

}

**return** 2;

}

**bool** ATM\_bilingual::check\_password(){

**int** input\_password;

**while**( error\_count < 3){

**if**(select\_language){

cout << "Write password" << endl;

}

**else**{

cout<<"비밀번호를 입력하세요"<<endl;

}

cin >> input\_password;

**if** (newacc->get\_password() == input\_password) {

**if**(select\_language){

cout << "valid password" << endl;

}

**else**{

cout<<"옳은 비밀번호입니다."<<endl;

}

**return** **true**;

}

**else** {

error\_count = error\_count + 1;

**if**(select\_language){

cout << "invalid password" << endl;

}

**else**{

cout<<"옳지 않은 비밀번호입니다."<<endl;

}

}

}

**return** **false**;

}

**bool** ATM\_bilingual::check\_bank(){

**if** (newacc->get\_bank\_name() == bank->get\_bankname()) {

**if**(select\_language){

//cout << "main bank" << endl;

}

**else**{

//cout<<"주거래 은행입니다."<<endl;

}

**return** **true**;

}

**else** {

**if** (atm\_type == "Single") {

**if**(select\_language){

cout << "invalid atm use" << endl;

}

**else**{

cout<<"허용되지 않은 atm 사용입니다."<<endl;

}

**return** **false**;

}

**else** {

**return** **true**;

}

}

}

**int** ATM\_bilingual::change\_balance(**int** money){

balance = balance + money;

**return** balance;

}

**int** ATM\_bilingual::get\_atm\_balance(){

**return** balance;

}

**int** ATM\_bilingual::get\_admin\_card(){

**return** admin\_card;

}

**bool** ATM\_bilingual::deposit(){

string money\_type;

**int** money;

**int** money\_number;

**bool** returnname = **true**;

**if**(select\_language){

cout << "What are you depoisting?cash/check \n How much money are you depositing? \n How many cash or checks are you depositing?" << endl;

}

**else**{

cout << "어떤 화폐로 입금하나요? 현금(cash)/수표(check) \n 얼만큼 돈을 입금하나요 \n 몇장의 화폐를 입금하시나요?" << endl;

}

cin >> money\_type;

cin >> money;

cin >> money\_number;

**int** transition\_fee = 0;

**if** (bank->get\_bankname() == newacc->get\_bank\_name()) {

transition\_fee = 0;

}

**else** **if** (atm\_type == "single") {

**if**(select\_language){

cout<< "unallowed transaction" <<endl;

}

**else**{

cout <<"허용되지 않은 거래입니다."<< endl;

}

returnname = **false**;

}

**else** {

transition\_fee = 500;

}

**if**( returnname == **true**)

{

**if** (money < transition\_fee)

{

returnname = **false**;

**if**(select\_language){

cout << "not enough money" << endl;

}

**else**{

cout<<"돈이 충분하지 않습니다."<<endl;

}

}

}

**if**( returnname == **true**)

{

**if** (money\_type == "check"){

**if** (money\_number > limit\_check)

{

**if**(select\_language){

cout << "limit error" << endl;

}

**else**{

cout << "화폐 수 초과 에러" << endl;

}

returnname = **false**;

}

**else**

{

newacc->change\_balance(money-transition\_fee);

}

}

**else**{

**if** (money\_number > limit\_cash)

{

**if**(select\_language){

cout << "limit error" << endl;

}

**else**{

cout << "화폐 수 초과 에러" << endl;

}

returnname = **false**;

}

**else**

{

change\_balance(money);//For ATM

newacc->change\_balance(money-transition\_fee);

}

}

}

ofstream writeFile;

**if**(transaction\_id == 0){

writeFile.open("transaction\_info.txt",ios::out);

}

**else**{

writeFile.open("transaction\_info.txt",ios::app);

}

transaction\_id++;

trsac\_info = "transsaction id : " + to\_string(transaction\_id) + ", card number : " + to\_string(newacc -> get\_card\_number()) + ", Transaction Type : deposit, Amount : " + to\_string(money) + '\n';

writeFile.write(trsac\_info.c\_str(), trsac\_info.size());

writeFile.close();

**return** returnname;

}

**bool** ATM\_bilingual::withdrawl(){

**int** money;

**bool** returnname = **true**;

**if**(select\_language){

cout << "How much money to withdrawl?" << endl;

}

**else**{

cout<<"얼마나 출금하시겠습니까?"<<endl;

}

cin >> money;

**int** transition\_fee = 0;

**if** (bank->get\_bankname() == newacc->get\_bank\_name()) {

transition\_fee = 500;

}

**else** **if** (atm\_type == "single") {

**if**(select\_language){

cout << "This atm is single." << endl;

}

**else**{

cout<<"이 atm은 단독으로만 사용됩니다."<<endl;

}

returnname = **false**;

}

**else** {

transition\_fee = 1000;

}

**if**(returnname == **true**){

**if** (money < transition\_fee)

{

**if**(select\_language){

cout << "Money is too low." << endl;

}

**else**{

cout<<"돈이 너무 적습니다."<<endl;

}

returnname = **false**;

}

}

**if**(returnname == **true**){

**if** (balance > money && newacc->get\_balance() > money + transition\_fee && max\_transaction > money)

{

newacc->change\_balance(-(money + transition\_fee));

change\_balance(-money);//For ATM

}

**else**

{

**if**(select\_language){

cout << "Money is too low" << endl;

}

**else**{

cout<<"돈이 너무 적습니다."<<endl;

}

returnname = **false**;

}

}

ofstream writeFile;

**if**(transaction\_id == 0){

writeFile.open("transaction\_info.txt",ios::out);

}

**else**{

writeFile.open("transaction\_info.txt",ios::app);

}

transaction\_id++;

trsac\_info = "transsaction id : " + to\_string(transaction\_id) + ", card number : " + to\_string(newacc -> get\_card\_number()) + ", Transaction Type : withdrawl, Amount : " + to\_string(money) + '\n';

writeFile.write(trsac\_info.c\_str(), trsac\_info.size());

writeFile.close();

**return** returnname;

}

**bool** ATM\_bilingual::transfer(){

**bool** returnname = **true**;

**int** select\_1;

**int** transfer\_money;

**int** Account\_Number;

**int** Cash\_number;

**char** Check\_transfer;

string Bank\_name;

**int** Transition\_fee = 0;

**if**(select\_language){

cout << "1. Transfer from your account\n 2. Transfer with cash" << endl;

}

**else**{

cout << "1. 계좌로부터 송금\n 2. 현금으로 송금" << endl;

}

cin >> select\_1;

**if**(select\_language){

cout << "How much money you will transfer?\n" << endl;

}

**else**{

cout << "얼마나 송금하실 건가요?\n" << endl;

}

cin >> transfer\_money;

**if**(select\_language){

cout << "Account Number of recipient" << endl;

}

**else**{

cout<<"송금하려는 상대의 계좌번호를 입력하세요?"<<endl;

}

cin >> Account\_Number;

**if**(select\_language){

cout << "Bank Name of recipient" << endl;

}

**else**{

cout<<"송금 상대의 은행을 입력하세요"<<endl;

}

cin >> Bank\_name;

**for**( **int** i = 0; i < 2; i++)

{

vector<Account\*> vector\_account = Account\_array[i];

**for** (**int** j = 0; j < 2; j++)

{

**if** (vector\_account.at(j)->get\_bank\_name() == Bank\_name && vector\_account.at(j)->get\_account\_number() == Account\_Number)

{

reciacc = vector\_account.at(j);

}

}

}

**if** (reciacc == **NULL**) {

cout << "wrong account information" << endl;

exit(-1);

}

**if**(select\_1 == 1){

**if**(select\_language){

cout << "Check if Bank account and Amount of Money is right\n Source Bank name:"<<newacc -> get\_bank\_name() << "\nSource Bank account:" << newacc -> get\_account\_number() <<"\nMoney:"<< transfer\_money<<endl;

cout << "Y/N" << endl;

}

**else**{

cout << "은행과 거래하려는 돈이 알맞은지 확인하세요\n 소스 은행 이름:"<< newacc -> get\_bank\_name() << "\n 소스 계좌번호:"<< newacc -> get\_account\_number() <<"\n 거래 금액:"<< transfer\_money<<endl;

cout << "Y/N" << endl;

}

cin >> Check\_transfer;

**if** (Check\_transfer == 'Y'){

newacc ->change\_balance(-transfer\_money);

}

**else**{

returnname = **false**;

}

}

**else** **if**(select\_1 == 2){

**if**(select\_language){

cout << "How many number of cash are you transfering?" << endl;

}

**else**{

cout<<" 몇장의 현금을 송금할 것입니까? "<<endl;

}

cin >> Cash\_number;

**if**( Cash\_number > 50){

**if**(select\_language){

cout << "Cash number error" << endl;

}

**else**{

cout<<"현금 수 오류"<<endl;

}

returnname = **false**;

}

**else**{

**if**(select\_language){

cout << "Check if Bank account and Amount of Money is right\n Bank name:"<<Bank\_name<< "\nBank account:" << Account\_Number <<"\nMoney:"<< transfer\_money<<endl;

cout << "Y/N" << endl;

}

**else**{

cout << "은행과 거래하려는 돈이 알맞은지 확인하세요\n 은행 이름:"<<Bank\_name<< "\n 계좌번호:"<< Account\_Number <<"\n 거래 금액:"<< transfer\_money<<endl;

cout << "Y/N" << endl;

}

cin >> Check\_transfer;

**if** (Check\_transfer == 'Y'){

change\_balance(transfer\_money);

}

**else**{

returnname = **false**;

}

}

}

**else**{

**if**(select\_language){

cout << "key error" << endl;

}

**else**{

cout<<"입력 오류"<<endl;

}

returnname = **false**;

}

cout << "Before transfer balance: " << reciacc->get\_balance() << endl;

reciacc -> change\_balance(transfer\_money);

**if**(newacc -> get\_bank\_name() == reciacc ->get\_bank\_name()){

**if**(check\_bank()){

Transition\_fee = 1500;

}

**else**{

Transition\_fee = 2500;

}

}

**else**{

Transition\_fee = 2000;

}

newacc ->change\_balance(-Transition\_fee);

ofstream writeFile;

**if**(transaction\_id == 0){

writeFile.open("transaction\_info.txt",ios::out);

}

**else**{

writeFile.open("transaction\_info.txt",ios::app);

}

transaction\_id++;

trsac\_info = "transsaction id : " + to\_string(transaction\_id) + ", card number : " + to\_string(newacc -> get\_card\_number()) + ", Transaction Type : transfer, Amount : " + to\_string(transfer\_money) + " recipient account number : " + to\_string(reciacc->get\_account\_number()) + '\n';

writeFile.write(trsac\_info.c\_str(), trsac\_info.size());

writeFile.close();

**return** returnname;

}

**bool** ATM\_bilingual::stop(**bool** check)

{

**if**(check) {

**return** **true**;

}

**else** {

**if**( cmp\_cardnum != admin\_card ){

ifstream readFile;

readFile.open("transaction\_info.txt");

**if** (readFile.is\_open())

{

**while** (!readFile.eof())

{

string str;

getline(readFile, str);

cout << str << endl; //지금은 읽은 문자열 바로 출력.

}

readFile.close(); //파일 닫아줍니다.

}

}

exit(-1);

}

}

**void** ATM\_bilingual::use()

{

cout << "Choose language : (0) Korea (1) English" << endl;

cin >> select\_language;

**int** select;

**bool** check;

**if** (select\_language) {

cout << "Welcome" << endl;

cout << "If you want to stop the session press Ctrl+C" << endl;

cout << "To start, please inset your debit card" << endl;

}

**else** {

cout << "환영합니다" << endl;

cout << "세션을 중단하고 싶다면 Ctrl+C를 누르세요" << endl;

cout << "체크카드를 넣어서 세션을 시작하세요" << endl;

}

**int** a = check\_account(); // 0 관리자 1 정상 2 비정상

**if** (a == 1) { check = **true**; }

**else** { check = **false**; }

stop(check);

check = check\_bank();

stop(check);

check = check\_password();

stop(check);

**while** (**true**)

{

**if** (select\_language) {

cout << " Select: (1)Deposit (2)Withdrawl (3) Transfer (4)Exit" << endl;

}

**else** {

cout << "선택하세요:(1) 입금 (2) 출금 (3) 송금 (4) 나가기" << endl;

}

cin >> select;

**if** (select == 1)

{

cout << "Before ATM balance: " << get\_atm\_balance() << endl;

cout << "After my account balance: " << newacc->get\_balance() << endl;

check = deposit();

cout << "Before ATM balance: " << get\_atm\_balance() << endl;

cout << "After my account balance: " << newacc->get\_balance() << endl;

stop(check);

}

**else** **if** (select == 2)

{

cout << "Before ATM balance: " << get\_atm\_balance() << endl;

cout << "Before my account balance: " << newacc->get\_balance() << endl;

check = withdrawl();

cout << "After ATM balance: " << get\_atm\_balance() << endl;

cout << "After my account balance: " << newacc->get\_balance() << endl;

stop(check);

}

**else** **if**(select==3)

{

cout << "Before ATM balance: " << get\_atm\_balance() << endl;

cout << "Before my account balance: " << newacc->get\_balance() << endl;

check = transfer();

cout << "After ATM balance: " << get\_atm\_balance() << endl;

cout << "After my account balance: " << newacc->get\_balance() << endl;

cout << "After transfer balance: " << reciacc->get\_balance() << endl;

stop(check);

}

**else** **if**(select==4){

stop(**false**);

}

**else**{

cout << "wrong choose" << endl;

exit(-1);

}

}

ifstream readFile;

readFile.open("transaction\_info.txt");

**if** (readFile.is\_open())

{

**while** (!readFile.eof())

{

string str;

getline(readFile, str);

cout << str << endl; //지금은 읽은 문자열 바로 출력.

}

readFile.close(); //파일 닫아줍니다.

}

}

**int** main()

{

//Creating two Daegu account

Account c1 = Account("Tom", 1, 1, 1, 100000, "Daegu");

Account c2 = Account("Tom", 2, 2, 2, 50000, "Daegu");

//Account c3 = Account("Amy", 3, 3, 3, 150000, "Kookmin");

vector<Account\*> Daegu\_account = { &c1, &c2 };

//Creating two Kookmin account

Account c3 = Account("Tom", 3, 3, 3, 150000, "Kookmin");

Account c4 = Account("Tom", 4, 4, 4, 50000, "Kookmin");

vector<Account\*> Kookmin\_account = { &c3, &c4 };

//Creating Bank instance

Bank Daegu(Daegu\_account, 100000000, "Daegu");

Bank Kookmin(Kookmin\_account, 100000000, "Kookmin");

Kookmin.get\_account\_info(0);

Kookmin.get\_account\_info(1);

Daegu.get\_account\_info(0);

vector<Account\*> Account\_array[]={Daegu\_account, Kookmin\_account};

ATM\_bilingual a1 = ATM\_bilingual(&Kookmin, Account\_array, "Single", 0, 100000);

ATM\_bilingual a2 = ATM\_bilingual(&Kookmin, Account\_array, "Multi", 0, 100000);

ATM\_unilingual a3 = ATM\_unilingual(&Kookmin, Account\_array, "Multi", 0, 100000);

a2.use();

}

1. Member student contribution table

모든 조원이 동일하게 기여했습니다.