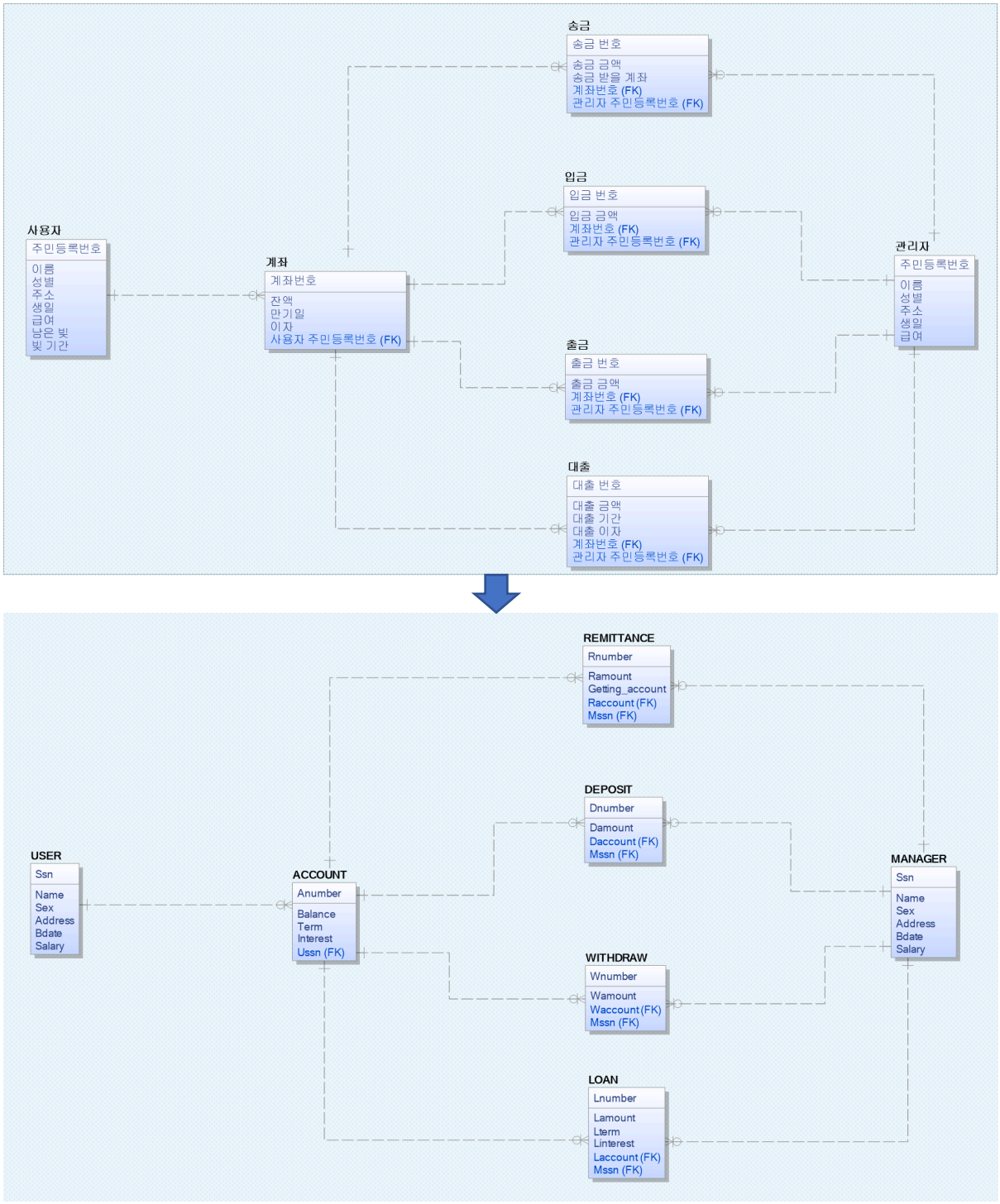
프로젝트 4: DBMS 프로그램 개발

2015005078 정진교



사용자 entity의 attribute 중에 남은 빚, 빚 기간을 제거하였습니다. 사용자, 계좌, 대출 entity를 이용하여 SQL문을 사용하는 것으로 대체하겠습니다. 또한 각 entity와 attribute의 이름을 영어로 바꾸었고 그에 따라 relation model을 첨부하였습니다.

다음과 같은 SQL문으로 테이블 스키마를 만들었습니다.

- CREATE TABLE USER

```
(Name        varchar(15)    not null,
Ssn          char(9)        not null,
Sex          char,
Address      varchar(30),
Bdate        date,
Salary       decimal(8),
Primary key (Ssn));
```

- CREATE TABLE ACCOUNT

```
(Anumber     char(12)       not null,
Balance      decimal(10)    not null,
Term         date,
Interest     decimal(3,2),
Ussn         char(9)        not null,
Primary key (Anumber),
Foreign key (Ussn) references USER(Ssn));
```

- CREATE TABLE MANAGER

```
(Name        varchar(15)    not null,
Ssn          char(9)        not null,
Sex          char,
Address      varchar(30),
Bdate        date,
Salary       decimal(8),
Primary key (Ssn));
```

- CREATE TABLE REMITTANCE

```
(Rnumber         decimal(5)     not null,
Ramount          decimal(10)    not null,
Getting_account  char(12)       not null,
Raccount.        char(12)       not null,
Mssn             char(9)        not null,
Primary key (Rnumber),
Foreign key (Raccount) references
ACCOUNT(Anumber),
Foreign key (Mssn) references MANAGER(Ssn));
```

- CREATE TABLE DEPOSIT

```
(Dnumber     decimal(5)     not null,
Damount.     decimal(10)    not null,
Daccount     char(12)       not null,
Mssn         char(9)        not null,
Primary key (Dnumber),
Foreign key (Daccount) references
ACCOUNT(Anumber),
Foreign key (Mssn) references MANAGER(Ssn));
```

- CREATE TABLE WITHDRAW

```
(Wnumber     decimal(5)     not null,
Wamount      decimal(10)    not null,
Waccount     char(12)       not null,
Mssn         char(9)        not null,
Primary key (Wnumber),
Foreign key (Waccount) references
ACCOUNT(Anumber),
Foreign key (Mssn) references MANAGER(Ssn));
```

- CREATE TABLE LOAN

```
(Lnumber     decimal(5)     not null,
Lamount      decimal(10)    not null,
Lterm        date           not null,
Linterest    decimal(3,2)   not null,
Laccount     char(12)       not null,
Mssn         char(9)        not null,
Primary key (Lnumber),
Foreign key (Laccount) references
ACCOUNT(Anumber),
Foreign key (Mssn) references MANAGER(Ssn));
```

샘플 데이터

User

Name	Ssn	Sex	Address	Bdate	Salary
John	123456789	M	731 Fondren, Houston, TX	1965.1.9	30000
Franklin	333445555	M	638 Voss, Houston, TX	1955.12.8	40000
Alicia	999887777	F	3321 Castle, Spring, TX	1968.1.19	25000
Jennifer	987654321	F	291 Berry, Bellaire, TX	1941.6.20	43000
Ramesh	666884444	M	975 Fire Oak, Humble, TX	1962.9.15	38000
Joyce	453453453	F	5631 Rice, Houston, TX	1972.7.31	25000
Ahmad	987987987	M	980 Dallas, Houston, TX	1969.3.29	25000
James	888665555	M	450 Stone, Houston, TX	1937.11.10	55000

Account

Anumber	Balance	Term	Interest	Ussn
964189181156	1000000	2020.5.14	1.07	453453453
273986092469	1250000	2020.5.9	1.07	987987987
713682887887	1380000	2020.2.13	1.2	999887777
478347894945	1400000	2020.9.1	1.13	123456789
943110567309	1550000	2020.1.4	1.12	666884444
216942137264	2430000	2020.3.1	1.07	333445555
599168678006	3250000	2020.5.1	1.03	987654321
539134653046	4250000	2020.2.5	1.04	888665555
578345624945	250000	NULL	NULL	123456789
948482887887	250000	NULL	NULL	999887777
943113467309	250000	NULL	NULL	666884444
366947237264	300000	NULL	NULL	333445555
539608253046	380000	NULL	NULL	888665555
729641348006	400000	NULL	NULL	987654321
964462351156	430000	NULL	NULL	453453453
850986092469	550000	NULL	NULL	987987987

Withdraw

Wnumber	Wamount	Waccount	Mssn
1	50	850986092469	222444888
2	50	964462351156	222444888
3	50	729641348006	222444888
4	250	539608253046	666333111
5	450	366947237264	666333111
6	650	943113467309	666333111
7	450	948482887887	222444888
8	450	578345624945	666333111
9	450	850986092469	222444888
10	450	964462351156	234234234
11	650	729641348006	234234234
12	650	539608253046	666333111
13	650	366947237264	666333111
14	650	943113467309	666333111
15	650	948482887887	222444888
16	250	578345624945	666333111
17	250	850986092469	666333111
18	250	964462351156	234234234
19	50	729641348006	234234234
20	50	539608253046	234234234
21	50	366947237264	234234234
22	250	943113467309	234234234
23	450	948482887887	234234234
24	650	578345624945	234234234
25	450	850986092469	222444888
26	450	964462351156	666333111
27	450	729641348006	666333111
28	450	539608253046	222444888
29	650	366947237264	222444888
30	650	943113467309	222444888
31	650	948482887887	222444888
32	650	578345624945	234234234

Manager

Name	Ssn	Sex	Address	Bdate	Salary
David	234234234	M	713 Fondren, Houston, TX	1951.12.15	50000
Fred	666333111	F	1455 Voss, Houston, TX	1995.8.13	15000
George	222444888	M	1121 Castle, Spring, TX	1960.11.11	40000

Remittance

Rnumber	Ramount	Getting_account	Raccount	Mssn
1	25	964462351156	850986092469	222444888
2	25	964462351156	850986092469	666333111
3	25	539608253046	964462351156	222444888
4	125	943113467309	964462351156	234234234
5	225	964462351156	729641348006	222444888
6	325	964462351156	729641348006	234234234
7	225	578345624945	539608253046	666333111
8	225	943113467309	539608253046	234234234
9	225	539608253046	366947237264	666333111
10	225	578345624945	366947237264	234234234
11	325	578345624945	943113467309	666333111
12	325	539608253046	943113467309	234234234
13	325	943113467309	948482887887	222444888
14	325	539608253046	948482887887	234234234
15	325	539608253046	578345624945	666333111
16	125	943113467309	578345624945	234234234
17	125	539608253046	850986092469	222444888
18	125	943113467309	850986092469	222444888
19	25	578345624945	964462351156	234234234
20	25	850986092469	964462351156	666333111
21	25	539608253046	729641348006	234234234
22	125	943113467309	729641348006	666333111
23	225	850986092469	539608253046	666333111
24	325	729641348006	539608253046	222444888
25	225	850986092469	366947237264	666333111
26	225	943113467309	366947237264	222444888
27	225	578345624945	943113467309	666333111
28	225	729641348006	943113467309	222444888
29	325	729641348006	948482887887	222444888
30	325	578345624945	948482887887	222444888
31	325	539608253046	578345624945	666333111
32	325	729641348006	578345624945	234234234

Deposit

Dnumber	Damount	Daccount	Mssn
1	100	578345624945	234234234
2	100	948482887887	222444888
3	100	943113467309	222444888
4	200	366947237264	222444888
5	300	539608253046	222444888
6	400	729641348006	666333111
7	300	964462351156	666333111
8	300	850986092469	222444888
9	300	578345624945	234234234
10	300	948482887887	234234234
11	400	943113467309	234234234
12	400	366947237264	234234234
13	400	539608253046	234234234
14	400	729641348006	234234234
15	400	964462351156	234234234
16	200	850986092469	666333111
17	200	578345624945	666333111
18	200	948482887887	222444888
19	100	943113467309	666333111
20	100	366947237264	666333111
21	100	539608253046	666333111
22	200	729641348006	234234234
23	300	964462351156	234234234
24	400	850986092469	222444888
25	300	578345624945	666333111
26	300	948482887887	222444888
27	300	943113467309	666333111
28	300	366947237264	666333111
29	400	539608253046	666333111
30	400	729641348006	222444888
31	400	964462351156	222444888
32	400	850986092469	222444888

Loan

Lnumber	Lamount	Lterm	Linterest	Laccount	Mssn
1	300000	2023.4.1	1.19	964462351156	222444888
2	530000	2022.6.9	1.29	539608253046	666333111
3	123000	2021.8.27	1.17	366947237264	666333111
4	240000	2023.10.3	1.25	948482887887	222444888
5	500000	2020.12.13	1.2	578345624945	234234234

메소드 설명

```
public static int menu() {

    System.out.println("0. Exit");
    System.out.println("1. User Menu");
    System.out.println("2. Manager Menu");
    System.out.print("Input : ");

    int temp = keyboard.nextInt();

    return temp;
}
public static int userMenu() {

    System.out.println("0. Return to previous menu");
    System.out.println("1. View account");
    System.out.println("2. REMITTANCE");
    System.out.println("3. DEPOSIT");
    System.out.println("4. WITHDRAW");
    System.out.println("5. LOAN");
    System.out.println("6. View transaction history");
    System.out.println("7. Make an account");
    System.out.println("8. Remove an account");
    System.out.print("Input : ");

    int temp = keyboard.nextInt();

    return temp;
}
public static int userSsn() {

    System.out.print("User ssn : ");

    int temp = keyboard.nextInt();

    return temp;
}
```

- menu()

```
0. Exit
1. User Menu
2. Manager Menu
Input :
```

=> 초기 메뉴화면입니다.

- userMenu()

```
0. Return to previous menu
1. View account
2. REMITTANCE
3. DEPOSIT
4. WITHDRAW
5. LOAN
6. View transaction history
7. Make an account
8. Remove an account
Input :
```

=> 유저 메뉴화면입니다.

- userSsn()

```
User ssn :
```

=> 유저의 ssn을 입력받는 화면입니다.

```
public static long account() {

    System.out.print("Account : ");

    long temp = keyboard.nextLong();

    return temp;
}
```

- account()

```
Account :
```

=> 계좌 번호를 입력받는 화면입니다.

```java
public static long getting_account() {

    System.out.print("Getting_account : ");

    long temp = keyboard.nextLong();

    return temp;
}
public static int amount() {

    System.out.print("Amount : ");

    int temp = keyboard.nextInt();

    return temp;
}
public static Date term() {

    System.out.print("Term(20xx-xx-xx) : ");

    String temp = keyboard.next();

    Date temp2 = Date.valueOf(temp);

    return temp2;
}
```

- getting_account()

```
Getting_account :
```

=> 송금받을 계좌 번호를 입력받는 화면입니다.

- amount()

```
Amount :
```

=> 거래를 할 돈의 양을 입력받는 화면입니다.

- term()

```
Term(20xx-xx-xx) :
```

=> 대출의 마감기한을 입력받는 화면입니다.

```java
public static float interest() {

    System.out.print("Interest(1.xx) : ");

    float temp = keyboard.nextFloat();

    return temp;
}
public static int mgrMenu() {

    System.out.println("0. Return to previous menu");
    System.out.println("1. View all users' transaction history");
    System.out.println("2. View a user's transaction history");
    System.out.println("3. View all users' account");
    System.out.println("4. View a user's account");
    System.out.print("Input : ");

    int temp = keyboard.nextInt();

    return temp;
}
public static int mgrSsn() {

    System.out.print("Manager ssn : ");

    int temp = keyboard.nextInt();

    return temp;
}
```

- interest()

```
Interest(1.xx) :
```

=> 대출의 이자율을 입력받는 화면입니다.

- mgrMenu()

```
0. Return to previous menu
1. View all users' transaction history
2. View a user's transaction history
3. View all users' account
4. View a user's account
Input :
```

=> 관리자 메뉴화면입니다.

- mgrSsn()

```
Manager ssn :
```

=> 매니저의 ssn을 입력받는 화면입니다.

- main()

```
public static void main(String[] args) {
    try {

        con = DriverManager.getConnection("jdbc:mariadb://localhost:3306/bank", "jinkyo422", "qkqhek123");

        stmt = con.createStatement();

        while(true) {
            menu = menu();

            if(menu == 0)   break;
            else if(menu == 1) {
                while(true) {
                    usermenu = userMenu();

                    if(usermenu == 0)   break;
                    else if(usermenu == 1)  userssn = userSsn();
                    else if(usermenu == 2)  userssn = userSsn();
                    else if(usermenu == 3)  userssn = userSsn();
                    else if(usermenu == 4)  userssn = userSsn();
                    else if(usermenu == 5)  userssn = userSsn();
                    else if(usermenu == 6)  userssn = userSsn();
                    else if(usermenu == 7)  userssn = userSsn();
                    else if(usermenu == 8)  userssn = userSsn();
                    else    System.out.println("Input Error..");
                }
            }
            else if(menu == 2) {
                while(true) {
                    mgrmenu = mgrMenu();

                    if(mgrmenu == 0)        break;
                    else if(mgrmenu == 1)   mgrssn = mgrSsn();
                    else if(mgrmenu == 2)   mgrssn = mgrSsn();
                    else if(mgrmenu == 3)   mgrssn = mgrSsn();
                    else if(mgrmenu == 4)   mgrssn = mgrSsn();
                    else {
                        System.out.println("Input Error..");
                    }
                }
            }
            else {
                System.out.println("Input Error..");
            }
        }

        stmt.close();
        con.close();
        }
    catch ( Exception e ) {
        e.printStackTrace();
    }
}
```

프로그램이 실행되면 maria DB에 연결한 후 초기 메뉴를 통해 종료, 유저 메뉴, 매니저 메뉴를 선택합니다. 유저 메뉴로 들어가면 9가지 메뉴 중에 하나를 선택하여 해당되는 메뉴에 들어가 0번 메뉴를 제외하고 나머지 메뉴들은 유저ssn을 입력받게 됩니다. 매니저 메뉴로 들어가면 5가지 메뉴중에 하나를 선택하여 해당되는 메뉴에 들어가 0번 메뉴를 제외하고 나머지 메뉴들은 매니저 ssn을 입력받게 됩니다. 0번 메뉴를 제외한 나머지 메뉴들에 대한 코드 설명을 하도록 하겠습니다.

- 유저메뉴 1번 View account

```
else if(usermenu == 1) {

    userssn = userSsn();

    rs = stmt.executeQuery( "select * from account where ussn=" + userssn);

    System.out.println("*If term is null and interest is 0.0, account is ordinary deposit.");
    System.out.println("*If not, account is time deposit.");
    System.out.println("-------------------------------------------");
    System.out.println("   anumber  \tbalance\t   term\t\tinterest");
    System.out.println("-------------------------------------------");
    while( rs.next() ) {
        long anumber = rs.getLong(1);
        long balance = rs.getLong(2);
        Date term = rs.getDate(3);
        float interest = rs.getFloat(4);
        if(term == null) {
            System.out.println(anumber + "\t" + balance + "\t   " + term + "\t\t" + interest);
        }
        else{
            System.out.println(anumber + "\t" + balance + "\t   " + term + "\t" + interest);
        }
    }
    System.out.println("-------------------------------------------");
    rs.close();
}
```

=> 한 유저가 가지고 있는 계좌정보를 출력하는 메뉴입니다.

=> 만약 term이 null이고 interest가 0.0이면 그 계좌는 보통예금이고 그렇지 않으면 정기예금입니다.

=> SQL

`"select * from account where ussn=" + userssn`

-> account 테이블에 저장된 ussn와 입력받은 userssn이 같은 터플들을 모두 뽑아옵니다. 만약 계좌가 없는 사람이라면 아무런 터플도 나오지 않습니다.

=> 실행 예시

```
User ssn : 123456789
*If term is null and interest is 0.0, account is ordinary deposit.
*If not, account is time deposit.
-------------------------------------------
   anumber      balance    term         interest
-------------------------------------------
478347894945    1400000    2020-09-01   1.13
578345624945    250000     null         0.0
-------------------------------------------
```

```
User ssn : 11
*If term is null and interest is 0.0, account is ordinary deposit.
*If not, account is time deposit.
-------------------------------------------
   anumber      balance    term         interest
-------------------------------------------
-------------------------------------------
```

- 유저메뉴 2번 REMITTANCE

```
else if(usermenu == 2) {

    userssn = userSsn();
    account = account();

    Date term = null, term2 = null;
    int balance = -1, balance2 = -1;

    rs = stmt.executeQuery("select balance, term from account where anumber=" + account
            + " and ussn=" + userssn);

    if(rs.next()) {
        balance = rs.getInt(1);
        term = rs.getDate(2);
    }
    if(balance == -1) {
        System.out.println("*The account does not exist");
        rs.close();
        continue;
    }
    if(term != null) {
        System.out.println("*The account is time deposit.");
        rs.close();
        continue;
    }

    get_account = getting_account();

    rs = stmt.executeQuery("select balance, term from account where anumber=" + get_account);

    if(rs.next()) {
        balance2 = rs.getInt(1);
        term2 = rs.getDate(2);
    }
    if(balance2 == -1) {
        System.out.println("*The getting account does not exist");
        rs.close();
        continue;
    }
    if(term2 != null) {
        System.out.println("*The getting account is time deposit.");
        rs.close();
        continue;
    }

    amount = amount();
```

```
    if(balance < amount) {
        System.out.println("*There is no money in the account.");
        rs.close();
        continue;
    }

    rs = stmt.executeQuery("select ssn from manager");
    if(rs.next())    mgrssn = rs.getInt(1);
```

```
    int rnumber = 0;
    rs = stmt.executeQuery("select max(rnumber) from remittance");
    if(rs.next())       rnumber = rs.getInt(1);
    rnumber++;
```

```
     stmt.executeUpdate("insert into remittance values('"
             + rnumber + "','" + amount +  "','" + get_account + "','"
             + account + "','" + mgrssn + "')");

     balance = balance - amount;
     stmt.executeUpdate("update account set balance=" + balance
             + " where anumber=" + account);

     balance2 = balance2 + amount;
     stmt.executeUpdate("update account set balance=" + balance2
             + " where anumber=" + get_account);

     System.out.println("*REMITTANCE success!");
     rs.close();
}
```

=> 한 유저가 본인의 보통계좌에서 다른 보통계좌로 송금하는 메뉴입니다.

=> 만약 계좌에 있는 잔액보다 더 많은 돈을 송금하면 오류메시지(*There is no money in the account.)를 출력합니다.

=> 정상적으로 송금이 완료되면 성공메시지(*REMITTANCE success!)를 출력합니다.

=> SQL

"select balance, term from account where anumber=" + *account* + " and ussn=" + *userssn*

-> account 테이블에 저장된 anumber, ussn와 입력받은 account, userssn이 같은 터플을 하나 뽑아옵니다. 만약 유저가 본인이 가지고 있지 않은 계좌번호를 입력하면 오류메시지(*The account does not exist.)를 출력합니다. 만약 유저가 본인의 정기예금을 입력하면 오류메시지(*The account is time deposit.)를 출력합니다.

"select balance, term from account where anumber=" + *get_account*

-> account 테이블에 저장된 anumber와 입력받은 get_account가 같은 터플을 하나 뽑아옵니다. 만약 account테이블에 없는 계좌번호를 입력하게 되면 송금받을 계좌가 DB에 없는 것이므로 오류메시지(*The account does not exist.)를 출력합니다. 만약 유저가 입력한 송금받을 계좌가 정기예금이면 오류메시지(*The account is time deposit.)를 출력합니다.

"select ssn from manager"

-> remittance 테이블에 송금내역을 터플로 저장할 때 같이 저장할 mgrssn을 불러옵니다.

"select max(rnumber) from remittance"

-> remittance 테이블에 송금내역을 터플로 저장할 때 같이 저장할 rnumber을 계산하기 위해 현재 remittance 테이블에 저장된 rnumber의 max를 불러옵니다.

"insert into remittance values('"+ rnumber + "','" + *amount* + "','" + *get_account* + "','" + *account* + "','" + *mgrssn* + "')"

-> 쿼리를 통해 불러온 rnumber의 가장 큰 값에서 1을 더한 값과 입력받거나 쿼리를 통해 불러온 amount, get_account, account, mgrssn을 하나의 터플로 만들어 remittance테이블에 삽입합니다.

"update account set balance=" + balance + " where anumber=" + *account*

-> 유저의 계좌의 balance에서 amount만큼을 뺀 balance로 업데이트합니다.

"update account set balance=" + balance2+ " where anumber=" + *get_account*

-> 유저가 송금한 amount만큼을 송금받는 계좌에 더하여 balance로 업데이트합니다.

=> 실행 예시

```
User ssn : 123456789
Account : 578345624940
*The account does not exist
```

```
User ssn : 123456789
Account : 478347894945
*The account is time deposit.
```

```
User ssn : 123456789
Account : 578345624945
Getting_account : 578345624940
*The getting account does not exist
```

```
User ssn : 123456789
Account : 578345624945
Getting_account : 599168678006
*The getting account is time deposit.
```

```
User ssn : 123456789
Account : 578345624945
Getting_account : 539608253046
Amount : 300000
*There is no money in the account.
```

```
User ssn : 123456789
Account : 578345624945
Getting_account : 539608253046
Amount : 100
*REMITTANCE success!
```

- 유저메뉴 3번 DEPOSIT

```
userssn = userSsn();
account = account();

Date term = null;
int balance = -1;

rs = stmt.executeQuery("select balance, term from account where anumber="
        + account + " and ussn=" + userssn);

if(rs.next()) {
    balance = rs.getInt(1);
    term = rs.getDate(2);
}
if(balance == -1) {
    System.out.println("*The account does not exist");
    rs.close();
    continue;
}
if(term != null) {
    System.out.println("*The account is time deposit.");
    rs.close();
    continue;
}

amount = amount();

rs = stmt.executeQuery("select ssn from manager");
if(rs.next())    mgrssn = rs.getInt(1);
```

```
int dnumber = 0;
rs = stmt.executeQuery("select max(dnumber) from deposit");
if(rs.next())        dnumber = rs.getInt(1);
dnumber++;
```

```
stmt.executeUpdate("insert into deposit values('"
        + dnumber + "','" + amount + "','" + account + "','" + mgrssn
        + "')");

balance = balance + amount;
stmt.executeUpdate("update account set balance=" + balance
        + " where anumber=" + account);

System.out.println("*DEPOSIT success!");
rs.close();
```

=> 한 유저가 본인의 보통계좌에 돈을 입금하는 메뉴입니다.

=> 정상적으로 입금이 완료되면 성공메시지(*DEPOSIT success!)를 출력합니다

=> SQL

"select balance, term from account where anumber=" + *account* + " and ussn=" + *userssn*

-> account 테이블에 저장된 anumber, ussn와 입력받은 account, userssn이 같은 터플을 하나 뽑아옵니다. 만약 유저가 본인이 가지고 있지 않은 계좌번호를 입력하면 오류메시지(*The account does not exist.)를 출력합니다. 만약 유저가 본인의 정기예금을 입력하면 오류메시지(*The account is time deposit.)를 출력합니다.

"select ssn from manager"

-> deposit 테이블에 입금내역을 터플로 저장할 때 같이 저장할 mgrssn을 불러옵니다.

"select max(dnumber) from deposit"

-> deposit 테이블에 입금내역을 터플로 저장할 때 같이 저장할 dnumber을 계산하기 위해 현재 deposit 테이블에 저장된 dnumber의 max를 불러옵니다.

"insert into deposit values('"+ dnumber + "','" + *amount* + "','" + *account* + "','" + *mgrssn* + "')"

-> 쿼리를 통해 불러온 dnumber의 가장 큰 값에서 1을 더한 값과 입력받거나 쿼리를 통해 불러온 amount, account, mgrssn을 하나의 터플로 만들어 deposit테이블에 삽입합니다

"update account set balance=" + balance + " where anumber=" + *account*

-> 유저의 계좌의 balance에서 amount만큼을 더한 balance로 업데이트합니다.

=> 실행 예시

```
User ssn : 123456789
Account : 578345624940
*The account does not exist
```

```
User ssn : 123456789
Account : 478347894945
*The account is time deposit.
```

```
User ssn : 123456789
Account : 578345624945
Amount : 500
*DEPOSIT success!
```

- 유저메뉴 4번 WITHDRAW

```
userssn = userSsn();
account = account();

Date term = null;
int balance = -1;

rs = stmt.executeQuery("select balance, term from account where anumber=" + account
        + " and ussn=" + userssn);

if(rs.next()) {
    balance = rs.getInt(1);
    term = rs.getDate(2);
}
if(balance == -1) {
    System.out.println("*The account does not exist");
    rs.close();
    continue;
}
if(term != null) {
    System.out.println("*The account is time deposit.");
    rs.close();
    continue;
}

amount = amount();

if(balance < amount) {
    System.out.println("*There is no money in the account.");
    rs.close();
    continue;
}

rs = stmt.executeQuery("select ssn from manager");
if(rs.next())    mgrssn = rs.getInt(1);
```

```
int wnumber = 0;
rs = stmt.executeQuery("select max(wnumber) from withdraw");
if(rs.next())       wnumber = rs.getInt(1);
wnumber++;
```

```
stmt.executeUpdate("insert into withdraw values('"
        + wnumber + "','" + amount + "','" + account + "','" + mgrssn
        + "')");

balance = balance - amount;
stmt.executeUpdate("update account set balance=" + balance
        + " where anumber=" + account);

System.out.println("*WITHDRAW success!");
rs.close();
```

=> 한 유저가 본인의 보통계좌에 돈을 출금하는 메뉴입니다.

=> 만약 계좌에 있는 잔액보다 더 많은 돈을 출금하면 오류메시지(*There is no money in the

account.)를 출력합니다.

=> 정상적으로 송금이 완료되면 성공메시지(*WITHDRAW success!)를 출력합니다.

=> SQL

```
"select balance, term from account where anumber=" + account + " and ussn="
+ userssn
```

-> account 테이블에 저장된 anumber, ussn와 입력받은 account, userssn이 같은 터플을 하나 뽑아옵니다. 만약 유저가 본인이 가지고 있지 않은 계좌번호를 입력하면 오류메시지(*The account does not exist.)를 출력합니다. 만약 유저가 본인의 정기예금을 입력하면 오류메시지(*The account is time deposit.)를 출력합니다.

```
"select ssn from manager"
```

-> withdraw 테이블에 출금내역을 터플로 저장할 때 같이 저장할 mgrssn을 불러옵니다.

```
"select max(wnumber) from withdraw"
```

-> withdraw 테이블에 출금내역을 터플로 저장할 때 같이 저장할 wnumber을 계산하기 위해 현재 withdraw 테이블에 저장된 wnumber의 max를 불러옵니다.

```
"insert into withdraw values('" + wnumber + "','" + amount + "','" +
account + "','" + mgrssn + "')"
```

-> 쿼리를 통해 불러온 wnumber의 가장 큰 값에서 1을 더한 값과 입력받거나 쿼리를 통해 불러온 amount, account, mgrssn을 하나의 터플로 만들어 withdraw테이블에 삽입합니다

```
"update account set balance=" + balance + " where anumber=" + account
```

-> 유저의 계좌의 balance에서 amount만큼을 뺀 balance로 업데이트합니다.

=> 실행 예시

```
User ssn : 123456789
Account : 578345624940
*The account does not exist
```

```
User ssn : 123456789
Account : 478347894945
*The account is time deposit.
```

```
User ssn : 123456789
Account : 578345624945
Amount : 300000
*There is no money in the account.
```

```
User ssn : 123456789
Account : 578345624945
Amount : 250
*WITHDRAW success!
```

- 유저메뉴 5번 LOAN

```
userssn = userSsn();

int loan = 0;

rs = stmt.executeQuery("select lnumber from account, loan where ussn = " + userssn
        + " and laccount = anumber");

if(rs.next()) {
    loan = rs.getInt(1);
}
if(loan != 0) {
    System.out.println("*You already have a loan");
    rs.close();
    continue;
}
```

```
account = account();

Date term = null;
int balance = -1;

rs = stmt.executeQuery("select balance, term from account where anumber=" + account
        + " and ussn=" + userssn);

if(rs.next()) {
    balance = rs.getInt(1);
    term = rs.getDate(2);
}
if(balance == -1) {
    System.out.println("*The account does not exist");
    rs.close();
    continue;
}
if(term != null) {
    System.out.println("*The account is time deposit.");
    rs.close();
    continue;
}

loan_term = term();
loan_interest = interest();
amount = amount();

rs = stmt.executeQuery("select ssn from manager");
if(rs.next())    mgrssn = rs.getInt(1);
```

```
int lnumber = 0;
rs = stmt.executeQuery("select max(lnumber) from loan");
if(rs.next())       lnumber = rs.getInt(1);
lnumber++;
```

```
stmt.executeUpdate("insert into loan values('"
        + lnumber + "','" + amount + "','" + loan_term + "','" + loan_interest + "','" +
        account + "','" + mgrssn + "')");

balance = balance + amount;
stmt.executeUpdate("update account set balance=" + balance
        + " where anumber=" + account);

System.out.println("*LOAN success!");
rs.close();
```

=> 한 유저가 대출을 받아 본인의 보통계좌에 대출받은 돈을 넣는 메뉴입니다.

=> 정상적으로 송금이 완료되면 성공메시지(*LOAN success!)를 출력합니다.

=> SQL

```
"select lnumber from account, loan where ussn = " + userssn + " and
laccount = anumber"
```

-> account, loan 테이블에 저장된 laccount 와 anumber가 같으면서 account 테이블에 저장된 ussn와 입력받은 userssn이 같은 터플을 하나 뽑아옵니다. 만약 이미 대출을 받은 유저는 다시

대출을 받을 수 없기 때문에 오류메시지(*You already have a loan)를 출력합니다.

```
"select balance, term from account where anumber=" + account + " and ussn="
+ userssn
```

-> account 테이블에 저장된 anumber, ussn와 입력받은 account, userssn이 같은 터플을 하나 뽑아옵니다. 만약 유저가 본인이 가지고 있지 않은 계좌번호를 입력하면 오류메시지(*The account does not exist.)를 출력합니다. 만약 유저가 본인의 정기예금을 입력하면 오류메시지(*The account is time deposit.)를 출력합니다.

```
"select ssn from manager"
```

-> loan 테이블에 대출내역을 터플로 저장할 때 같이 저장할 mgrssn을 불러옵니다.

```
"select max(lnumber) from loan"
```

-> loan 테이블에 대출내역을 터플로 저장할 때 같이 저장할 lnumber을 계산하기 위해 현재 loan 테이블에 저장된 lnumber의 max를 불러옵니다.

```
"insert into loan values('" + lnumber + "','" + amount + "','" + loan_term
+ "','" + loan_interest + "','" + account + "','" + mgrssn + "')"
```

-> 쿼리를 통해 불러온 lnumber의 가장 큰 값에서 1을 더한 값과 입력받거나 쿼리를 통해 불러온 amount, loan_term, loan_interest, account, mgrssn을 하나의 터플로 만들어 loan테이블에 삽입합니다

```
"update account set balance=" + balance + " where anumber=" + account
```

-> 유저의 계좌의 balance에서 amount만큼을 더한 balance로 업데이트합니다.

=> 실행 예시

```
User ssn : 123456789
*You already have a loan
```

```
Input : 5
User ssn : 987654321
Account : 729641348000
*The account does not exist
```

```
Input : 5
User ssn : 987654321
Account : 599168678006
*The account is time deposit.
```

```
User ssn : 987654321
Account : 729641348006
Term(20xx-xx-xx) : 2019-1-1
Interest(1.xx) : 1.05
Amount : 500
*LOAN success!
```

- 유저메뉴 6번 View transaction history

```
userssn = userSsn();

System.out.println("*REMITTANCE history");
rs = stmt.executeQuery( "select ramount, getting_account, raccount from account, remittance "
        + "where anumber = raccount and ussn=" + userssn);

System.out.println("-----------------------------------------");
System.out.println("   anumber  \tamount\t   getting_account");
System.out.println("-----------------------------------------");
while( rs.next() ) {
    long anumber = rs.getLong(3);
    int amount = rs.getInt(1);
    long getting_account = rs.getLong(2);
    System.out.println(anumber + "\t" + amount + "\t   " + getting_account);
}
System.out.println("-----------------------------------------");
```

```
System.out.println("*Getting history");
rs = stmt.executeQuery( "select ramount, getting_account, raccount from account, remittance "
        + "where anumber = getting_account and ussn=" + userssn);

System.out.println("-----------------------------------------");
System.out.println("   anumber  \tamount\t   sent_account");
System.out.println("-----------------------------------------");
while( rs.next() ) {
    long anumber = rs.getLong(2);
    int amount = rs.getInt(1);
    long sent_account = rs.getLong(3);
    System.out.println(anumber + "\t" + amount + "\t   " + sent_account);
}
System.out.println("-----------------------------------------");


System.out.println("*DEPOSIT history");
rs = stmt.executeQuery( "select damount, daccount from account, deposit "
        + "where anumber = daccount and ussn=" + userssn);

System.out.println("-----------------------------------------");
System.out.println("   anumber  \tamount\t   ");
System.out.println("-----------------------------------------");
while( rs.next() ) {
    long anumber = rs.getLong(2);
    int amount = rs.getInt(1);
    System.out.println(anumber + "\t" + amount);
}
System.out.println("-----------------------------------------");

System.out.println("*WITHDRAW history");
rs = stmt.executeQuery( "select wamount, waccount from account, withdraw "
        + "where anumber = waccount and ussn=" + userssn);

System.out.println("-----------------------------------------");
System.out.println("   anumber  \tamount\t   ");
System.out.println("-----------------------------------------");
while( rs.next() ) {
    long anumber = rs.getLong(2);
    int amount = rs.getInt(1);
    System.out.println(anumber + "\t" + amount);
}
System.out.println("-----------------------------------------");
```

```
System.out.println("*LOAN history");
rs = stmt.executeQuery( "select lamount, lterm, linterest, laccount from account, loan "
        + "where anumber = laccount and ussn=" + userssn);

System.out.println("-----------------------------------------");
System.out.println("   anumber  \tamount\t   term\t\tinterest");
System.out.println("-----------------------------------------");
while( rs.next() ) {
    long anumber = rs.getLong(4);
    int amount = rs.getInt(1);
    Date term = rs.getDate(2);
    float interest = rs.getFloat(3);
    System.out.println(anumber + "\t" + amount + "\t   " + term + "\t" + interest);
}
System.out.println("-----------------------------------------");
rs.close();
```

=> 한 유저에 대한 거래내역(송금내역, 송금받은 내역, 입금내역, 출금내역, 대출내역)을 불러오는 메뉴입니다.

=> SQL

```
"select ramount, getting_account, raccount from account, remittance " +
"where anumber = raccount and ussn=" + userssn
```

-> account, remittance 테이블에 저장된 anumber와 raccount가 같고 account에 저장된 ussn과 입력받은 userssn이 같은 터플들을 뽑아옵니다.

```
"select ramount, getting_account, raccount from account, remittance " +
"where anumber = getting_account and ussn=" + userssn
```

-> account, remittance 테이블에 저장된 anumber와 getting_account가 같고 account에 저장된 ussn과 입력받은 userssn이 같은 터플들을 뽑아옵니다.

```
"select damount, daccount from account, deposit " + "where anumber =
daccount and ussn=" + userssn
```

-> account, deposit 테이블에 저장된 anumber와 daccount가 같고 account에 저장된 ussn과 입력받은 userssn이 같은 터플들을 뽑아옵니다.

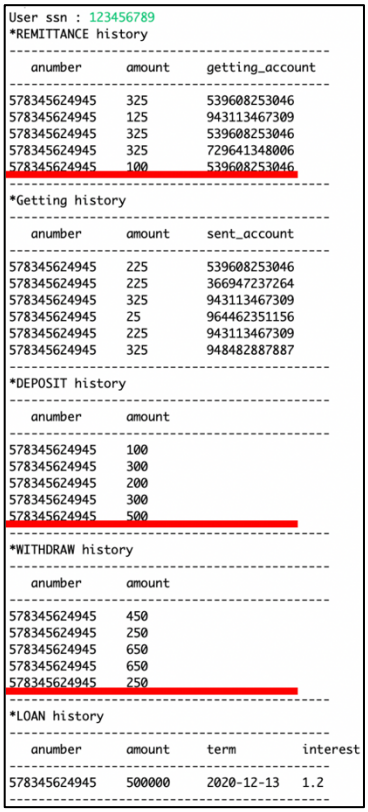
```
"select wamount, waccount from account, withdraw " + "where anumber =
waccount and ussn=" + userssn
```

-> account, withdraw 테이블에 저장된 anumber와 waccount가 같고 account에 저장된 ussn과 입력받은 userssn이 같은 터플들을 뽑아옵니다.

```
"select lamount, lterm, linterest, laccount from account, loan " + "where
anumber = laccount and ussn=" + userssn
```

-> account, loan 테이블에 저장된 anumber와 laccount가 같고 account에 저장된 ussn과 입력받은 userssn이 같은 터플들을 뽑아옵니다.

=> 실행 예시



User ssn : 123456789
*REMITTANCE history

anumber	amount	getting_account
578345624945	325	539608253046
578345624945	125	943113467309
578345624945	325	539608253046
578345624945	325	729641348006
578345624945	100	539608253046

*Getting history

anumber	amount	sent_account
578345624945	225	539608253046
578345624945	225	366947237264
578345624945	325	943113467309
578345624945	25	964462351156
578345624945	225	943113467309
578345624945	325	948482887887

*DEPOSIT history

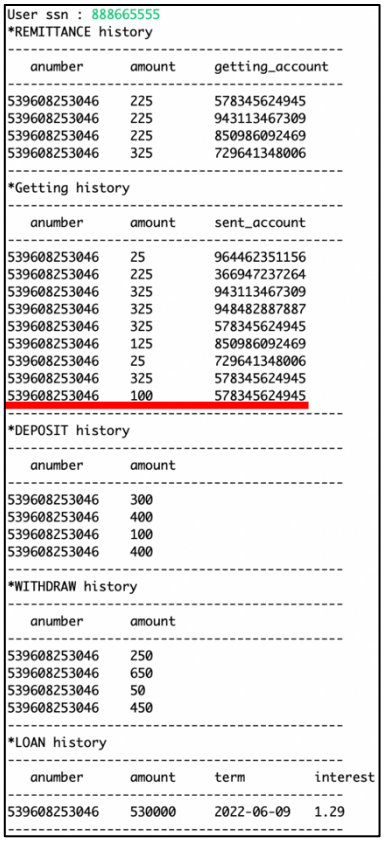
anumber	amount
578345624945	100
578345624945	300
578345624945	200
578345624945	300
578345624945	500

*WITHDRAW history

anumber	amount
578345624945	450
578345624945	250
578345624945	650
578345624945	650
578345624945	250

*LOAN history

anumber	amount	term	interest
578345624945	500000	2020-12-13	1.2



User ssn : 888665555
*REMITTANCE history

anumber	amount	getting_account
539608253046	225	578345624945
539608253046	225	943113467309
539608253046	225	850986092469
539608253046	325	729641348006

*Getting history

anumber	amount	sent_account
539608253046	25	964462351156
539608253046	225	366947237264
539608253046	325	943113467309
539608253046	325	948482887887
539608253046	325	578345624945
539608253046	125	850986092469
539608253046	25	729641348006
539608253046	325	578345624945
539608253046	100	578345624945

*DEPOSIT history

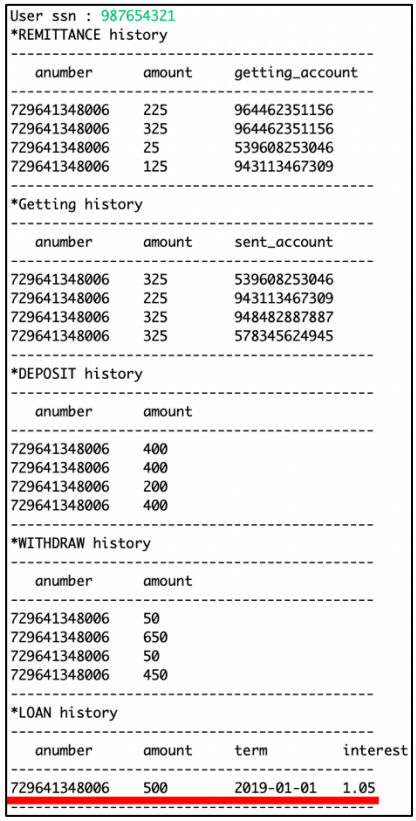
anumber	amount
539608253046	300
539608253046	400
539608253046	100
539608253046	400

*WITHDRAW history

anumber	amount
539608253046	250
539608253046	650
539608253046	50
539608253046	450

*LOAN history

anumber	amount	term	interest
539608253046	530000	2022-06-09	1.29



User ssn : 987654321
*REMITTANCE history

anumber	amount	getting_account
729641348006	225	964462351156
729641348006	325	964462351156
729641348006	25	539608253046
729641348006	125	943113467309

*Getting history

anumber	amount	sent_account
729641348006	325	539608253046
729641348006	225	943113467309
729641348006	325	948482887887
729641348006	325	578345624945

*DEPOSIT history

anumber	amount
729641348006	400
729641348006	400
729641348006	200
729641348006	400

*WITHDRAW history

anumber	amount
729641348006	50
729641348006	650
729641348006	50
729641348006	450

*LOAN history

anumber	amount	term	interest
729641348006	500	2019-01-01	1.05

- 유저메뉴 7번 Make an account

```
userssn = userSsn();

int salary = 0;

rs = stmt.executeQuery("select salary from user where ssn=" + userssn);

if(rs.next())    salary = rs.getInt(1);
if(salary == 0) {
    System.out.println("*The user does not exist.");
    rs.close();
    continue;
}
System.out.println("*Account number must be 12 digits.");

account = account();
int lenth = (int)(Math.log10(account)+1);

if(lenth != 12) {
    System.out.println("*The account number is not 12 digits.");
    rs.close();
    continue;
}

int balance = -1;

rs = stmt.executeQuery("select balance from account where anumber=" + account);

if(rs.next())    balance = rs.getInt(1);
if(balance != -1) {
    System.out.println("*The account number already exists.");
    rs.close();
    continue;
}

String temp = "2020-1-1";
loan_term = Date.valueOf(temp);
loan_interest = (float) 1.01;

System.out.println("*You can make a time deposit or an ordinary deposit.");
System.out.println("*A time depoisit : term is " + loan_term + " and interest is " + "loan_interest");
System.out.println("*If you want, T or t. If not, O or o");
System.out.print("Account type : ");

String type = keyboard.next();
```

```
if(type.equals("T") || type.equals("t")) {
    System.out.println("*How much do you want?");
    System.out.print("Money : ");
    balance = keyboard.nextInt();

    rs = stmt.executeQuery("select ssn from manager");
    if(rs.next())    mgrssn = rs.getInt(1);
```

```
     int dnumber = 0;
     rs = stmt.executeQuery("select max(dnumber) from deposit");
     if(rs.next())        dnumber = rs.getInt(1);
     dnumber++;
```

```
    stmt.executeUpdate("insert into account values('"
                + account + "','" + balance +  "','" + loan_term + "','" + loan_interest + "','" + userssn
                + "')");

    stmt.executeUpdate("insert into deposit values('"
                + dnumber + "','" + balance + "','" + account + "','" + mgrssn
                + "')");
```

```
    System.out.println("*Success to make a time deposit!");
}
else if(type.equals("0") || type.equals("o")){
    stmt.executeUpdate("insert into account values('"
                + account + "','0', null, null,'" + userssn
                + "')");
    System.out.println("*Success to make a ordinary deposit!");
}
else {
    System.out.println("Input Error..");
}
rs.close();
```

=> 한 유저가 계좌를 만드는 메뉴입니다.

=> 어떠한 계좌번호로 계좌를 만들지 입력을 받고 그 계좌번호가 12자리인지 확인합니다. 만약 12자리가 아니라면 오류메시지(*The account number is not 12 digits.)를 출력합니다. 그 후, 정기예금이나 보통예금을 선택할 수 있습니다.

=> 정기예금일 경우 초기에 입금할 금액을 입력받아 계좌에 입금하고 계좌를 만듭니다. 또한, 입금내역에 새로운 터플을 추가합니다. 그 후, 성공메시지(*Success to make a time deposit!)를 출력합니다.

=> 보통예금일 경우 초기 잔액을 0원으로 설정하여 계좌를 만들고 성공메시지(*Success to make a ordinary deposit!)를 출력합니다.

=> SQL

```
"select salary from user where ssn=" + userssn
```

-> 현재 DB에 저장된 사람인지 확인하기 위한 query입니다. 만약 정보가 없다면 오류메시지(*The user does not exist.)를 출력합니다.

```
"select balance from account where anumber=" + account
```

-> 입력받은 account가 이미 DB에 저장된 값인지 확인하기 위한 query입니다. 만약 이미 존재하는 값이라면 오류메시지(*The account number already exists.)를 출력합니다.

```
"select ssn from manager"
```

-> deposit 테이블에 입금내역을 터플로 저장할 때 같이 저장할 mgrssn을 불러옵니다.

```
"select max(dnumber) from deposit"
```

-> deposit 테이블에 입금내역을 터플로 저장할 때 같이 저장할 dnumber을 계산하기 위해 현재 deposit 테이블에 저장된 dnumber의 max를 불러옵니다.

```
"insert into account values('" +  account + "','" + balance +  "','" +
loan_term + "','" + loan_interest + "','" + userssn + "')"
```

-> 입력받은 account, balance, userssn과 코드에서 설정되어있는 loan_term, loan_interest를 하나의 터플로 만들어 account테이블에 삽입합니다.

```
"insert into deposit values('"+ dnumber + "','" + balance + "','" + account
+ "','" + mgrssn + "')"
```

-> 쿼리를 통해 불러온 dnumber의 가장 큰 값에서 1을 더한 값과 입력받거나 쿼리를 통해 불러온 balance, account, mgrssn을 하나의 터플로 만들어 deposit테이블에 삽입합니다.

```
"insert into account values('" + account + "','0', null, null,'" + userssn
+ "')"
```

-> 입력받은 account, userssn과 나머지 값들은 0이나 null로 하나의 터플을 만들어 account테이블에 삽입합니다.

=> 실행 예시

```
Input : 7
User ssn : 123
*The user does not exist.
```

```
User ssn : 123456789
*Account number must be 12 digits.
Account : 111
*The account number is not 12 digits.
```

```
User ssn : 123456789
*Account number must be 12 digits.
Account : 111111111111
*You can make a time deposit or an ordinary deposit.
*A time depoisit : term is 2020-01-01 and interest is loan_interest
*If you want, T or t. If not, O or o
Account type : t
*How much do you want?
Money : 9999
*Success to make a time deposit!
```

```
User ssn : 123456789
*Account number must be 12 digits.
Account : 222222222222
*You can make a time deposit or an ordinary deposit.
*A time depoisit : term is 2020-01-01 and interest is loan_interest
*If you want, T or t. If not, O or o
Account type : o
*Success to make a ordinary deposit!
```

```
User ssn : 123456789
*Account number must be 12 digits.
Account : 578345624945
*The account number already exists.
```

=> 실행 완료 후 유저 1번메뉴를 통해 계좌가 DB에 잘 저장되었음을 확인 수 있습니다.

```
*If term is null and interest is 0.0, account is ordinary deposit.
*If not, account is time deposit.
---------------------------------------------
   anumber      balance    term         interest
---------------------------------------------
111111111111    9999       2020-01-01   1.01
222222222222    0          null         0.0
478347894945    1400000    2020-09-01   1.13
578345624945    250150     null         0.0
---------------------------------------------
```

- 유저메뉴 8번 Remove an account

```
userssn = userSsn();

int salary = 0;

rs = stmt.executeQuery("select salary from user where ssn=" + userssn);

if(rs.next())    salary = rs.getInt(1);
if(salary == 0) {
    System.out.println("*The user does not exist.");
    rs.close();
    continue;
}
System.out.println("*Do you want to remove your account?");
System.out.println("*Although it is a time deposit, you cannot take the interest");
System.out.println("*If you want, Y or y. If not, N or n");
System.out.print("Answer: ");

String ans = keyboard.next();

if(ans.equals("Y") || ans.equals("y")) {
    System.out.println("*Which account do you want to remove?");
    account = account();

    int balance = -1;

    rs = stmt.executeQuery("select balance from account where anumber=" + account
            + " and ussn=" + userssn);

    if(rs.next())    balance = rs.getInt(1);

    if(balance == -1) {
        System.out.println("*The account does not exist.");
        continue;
    }

    int lnumber = 0;

    rs = stmt.executeQuery("select lnumber from loan where laccount = " + account);

    if(rs.next())    lnumber = rs.getInt(1);

    if(lnumber != 0) {
        System.out.println("*The account has a loan.");
        continue;
    }
    stmt.executeUpdate("delete from remittance where rnumber="+ account);
    stmt.executeUpdate("delete from remittance where getting_account="+ account);
    stmt.executeUpdate("delete from deposit where dnumber="+ account);
    stmt.executeUpdate("delete from withdraw where wnumber="+ account);
    stmt.executeUpdate("delete from account where anumber="+ account);
    System.out.println("*You take " + balance + ".");
    System.out.println("*Success to remove the account!");
}
```

```
else if(ans.equals("N") || ans.equals("n")) {
    System.out.println("*It's a good choice.");
}
else {
    System.out.println("Input Error..");
}
rs.close();
```

=> 한 유저가 본인의 계좌를 삭제하는 메뉴입니다.

=> 계좌의 종류에 관계없이 현재 잔액을 유저가 가질 수 있습니다. 계좌가 삭제되면 성공메시지(*Success to remove the account!)를 출력합니다.

=> SQL

`"select salary from user where ssn=" + userssn`

-> 현재 DB에 저장된 사람인지 확인하기 위한 query입니다. 만약 정보가 없다면 오류메시지(*The user does not exist.)를 출력합니다.

`"select balance from account where anumber=" + account + " and ussn=" + userssn`

-> account 테이블에 저장된 anumber, ussn와 입력받은 account, userssn이 같은 터플을 하나 뽑아옵니다. 만약 유저가 본인이 가지고 있지 않은 계좌번호를 입력하면 오류메시지(*The account does not exist.)를 출력합니다.

`"select lnumber from loan where laccount = " + account`

-> loan 테이블에 저장된 laccount와 입력받은 account가 같은 터플을 하나 뽑아옵니다. 만약 존재한다면, 그 계좌는 대출을 가지고 있는 것이므로 삭제될 수 없기 때문에 오류메시지(*The account has a loan.)를 출력합니다.

```
delete from remittance where rnumber="+ account
delete from remittance where getting_account="+ account
delete from deposit where dnumber="+ account
delete from withdraw where wnumber="+ account
delete from account where anumber="+ account
```

-> remittance, deposit, withdraw테이블에 있는 입력받은 account의 거래내역이 있는 터플들을 지우고 마지막에 account테이블에서 계좌를 삭제합니다.

=> 실행 예시

```
Input : 8
User ssn : 123
*The user does not exist.
```

```
User ssn : 123456789
*Do you want to remove your account?
*Although it is a time deposit, you cannot take the interest
*If you want, Y or y. If not, N or n
Answer: y
*Which account do you want to remove?
Account : 578345624940
*The account does not exist.
```

```
User ssn : 123456789
*Do you want to remove your account?
*Although it is a time deposit, you cannot take the interest
*If you want, Y or y. If not, N or n
Answer: y
*Which account do you want to remove?
Account : 578345624945
*The account has a loan.
```

```
User ssn : 987987987
*Do you want to remove your account?
*Although it is a time deposit, you cannot take the interest
*If you want, Y or y. If not, N or n
Answer: y
*Which account do you want to remove?
Account : 850986092469
*You take 550000.
*Success to remove the account!
```

=> 실행 완료 후 유저 1번메뉴를 통해 계좌가 DB에서 삭제되었음을 확인 수 있습니다.

```
User ssn : 987987987
*If term is null and interest is 0.0, account is ordinary deposit.
*If not, account is time deposit.
-------------------------------------------
   anumber       balance     term          interest
-------------------------------------------
273986092469     1250000     2020-05-09    1.07
-------------------------------------------
```

- 매니저메뉴 1번 View all users' transaction history

```
mgrssn = mgrSsn();

int salary = 0;

rs = stmt.executeQuery("select salary from manager where ssn=" + mgrssn);

if(rs.next())    salary = rs.getInt(1);
if(salary == 0) {
    System.out.println("You are not manager.");
    rs.close();
    continue;
}

System.out.println("*REMITTANCE history");
rs = stmt.executeQuery( "select name, ssn, ramount, getting_account, raccount"
        + " from user, account, remittance "
        + " where anumber = raccount and ussn=ssn");

System.out.println("-------------------------------------------");
System.out.println("name\t ssn\t\tanumber  \tamount\t   getting_account");
System.out.println("-------------------------------------------");
while( rs.next() ) {
```

```
    String name = rs.getString(1);
    int ssn = rs.getInt(2);
    long anumber = rs.getLong(5);
    int amount = rs.getInt(3);
    long getting_account = rs.getLong(4);
    System.out.println(name + "\t " + ssn + "\t" + anumber + "\t" + amount + "\t   " + getting_account);
}
System.out.println("-----------------------------------------");

System.out.println("*Getting history");
rs = stmt.executeQuery( "select name, ssn, ramount, getting_account, raccount"
        + " from user, account, remittance "
        + " where anumber = getting_account and ussn=ssn");

System.out.println("-----------------------------------------");
System.out.println("name\t ssn\t\tanumber  \tamount\t   sent_account");
System.out.println("-----------------------------------------");
while( rs.next() ) {
    String name = rs.getString(1);
    int ssn = rs.getInt(2);
    long anumber = rs.getLong(4);
    int amount = rs.getInt(3);
    long sent_account = rs.getLong(5);
    System.out.println(name + "\t " + ssn + "\t" + anumber + "\t" + amount + "\t   " + sent_account);
}
System.out.println("-----------------------------------------");
```

```
System.out.println("*DEPOSIT history");
rs = stmt.executeQuery( "select name, ssn, damount, daccount "
        + " from user, account, deposit "
        + " where anumber = daccount and ussn=ssn");

System.out.println("-----------------------------------------");
System.out.println("name\t ssn\t\tanumber  \tamount\t   ");
System.out.println("-----------------------------------------");
while( rs.next() ) {
    String name = rs.getString(1);
    int ssn = rs.getInt(2);
    long anumber = rs.getLong(4);
    int amount = rs.getInt(3);
    System.out.println(name + "\t " + ssn + "\t" + anumber + "\t" + amount);
}
System.out.println("-----------------------------------------");

System.out.println("*WITHDRAW history");
rs = stmt.executeQuery( "select name, ssn, wamount, waccount "
        + " from user, account, withdraw "
        + " where anumber = waccount and ussn=ssn");

System.out.println("-----------------------------------------");
System.out.println("name\t ssn\t\tanumber  \tamount\t   ");
System.out.println("-----------------------------------------");
while( rs.next() ) {
    String name = rs.getString(1);
    int ssn = rs.getInt(2);
    long anumber = rs.getLong(4);
    int amount = rs.getInt(3);
    System.out.println(name + "\t " + ssn + "\t" + anumber + "\t" + amount);
}
System.out.println("-----------------------------------------");
```

```
System.out.println("*LOAN history");
rs = stmt.executeQuery( "select name, ssn, lamount, lterm, linterest, laccount "
        + " from user, account, loan "
        + " where anumber = laccount and ussn=ssn");

System.out.println("-----------------------------------------");
System.out.println("name\t ssn\t\tanumber  \tamount\t   term\t\tinterest");
System.out.println("-----------------------------------------");
while( rs.next() ) {
```

```
    String name = rs.getString(1);
    int ssn = rs.getInt(2);
    long anumber = rs.getLong(6);
    int amount = rs.getInt(3);
    Date term = rs.getDate(4);
    float interest = rs.getFloat(5);
    System.out.println(name + "\t " + ssn + "\t" + anumber + "\t" + amount + "\t   " + term + "\t" + interest);
}
System.out.println("-----------------------------------------");
rs.close();
```

=> 모든 유저의 거래내역을 보는 메뉴입니다.

=> 모든 유저의 송금내역, 송금받은 내역, 입금내역, 출금내역, 대출내역을 조회합니다.

=> SQL

`"select salary from manager where ssn=" + mgrssn`

-> manager 테이블에 저장된 ssn과 입력받은 mgrssn이 같은 사람을 찾습니다. 만약 없다면 그 사람은 매니저가 아니므로 오류메시지(You are not manager.)를 출력합니다.

`"select name, ssn, ramount, getting_account, raccount" + " from user, account, remittance " + " where anumber = raccount and ussn=ssn"`

-> user, account, remittance 테이블에 저장된 anumber와 raccount가 같고 ussn과 ssn이 같은 터플들을 뽑아냅니다.

`"select name, ssn, ramount, getting_account, raccount" + " from user, account, remittance " + " where anumber = getting_account and ussn=ssn"`

-> user, account, remittance 테이블에 저장된 anumber와 getting_account가 같고 ussn과 ssn이 같은 터플들을 뽑아냅니다.

`"select name, ssn, damount, daccount " + " from user, account, deposit " + " where anumber = daccount and ussn=ssn"`

-> user, account, deposit 테이블에 저장된 anumber와 daccount가 같고 ussn과 ssn이 같은 터플들을 뽑아냅니다.

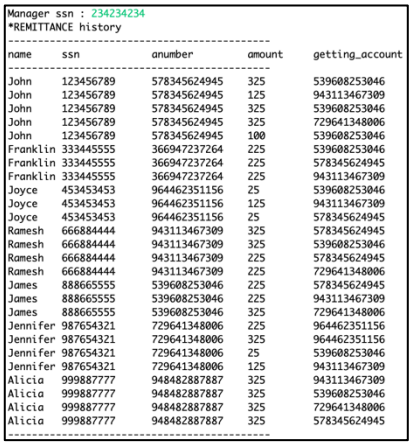
`"select name, ssn, wamount, waccount " + " from user, account, withdraw " + " where anumber = waccount and ussn=ssn"`

-> user, account, withdraw 테이블에 저장된 anumber와 waccount가 같고 ussn과 ssn이 같은 터플들을 뽑아냅니다.

`"select name, ssn, lamount, lterm, linterest, laccount " + " from user, account, loan " + " where anumber = laccount and ussn=ssn"`

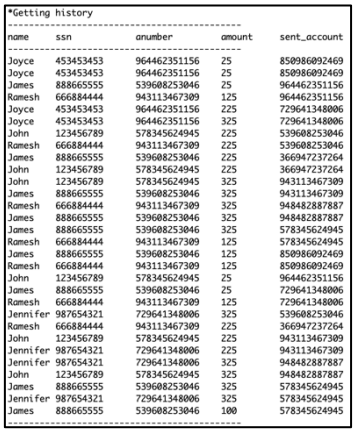
-> user, account, loan 테이블에 저장된 anumber와 laccount가 같고 ussn과 ssn이 같은 터플들을 뽑아냅니다.

=> 실행 예시



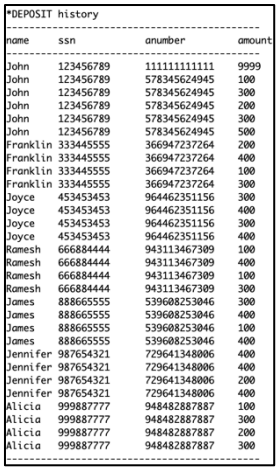
Manager ssn : 234234234
*REMITTANCE history

name	ssn	anumber	amount	getting_account
John	123456789	578345624945	325	539608253046
John	123456789	578345624945	125	943113467309
John	123456789	578345624945	325	539608253046
John	123456789	578345624945	325	729641348006
John	123456789	578345624945	100	539608253046
Franklin	333445555	366947237264	225	539608253046
Franklin	333445555	366947237264	225	578345624945
Franklin	333445555	366947237264	225	943113467309
Joyce	453453453	964462351156	25	539608253046
Joyce	453453453	964462351156	125	943113467309
Joyce	453453453	964462351156	25	578345624945
Ramesh	666884444	943113467309	325	578345624945
Ramesh	666884444	943113467309	325	539608253046
Ramesh	666884444	943113467309	225	578345624945
Ramesh	666884444	943113467309	225	729641348006
James	888665555	539608253046	225	578345624945
James	888665555	539608253046	225	943113467309
James	888665555	539608253046	325	729641348006
Jennifer	987654321	729641348006	225	964462351156
Jennifer	987654321	729641348006	325	964462351156
Jennifer	987654321	729641348006	25	539608253046
Jennifer	987654321	729641348006	125	943113467309
Alicia	999887777	948482887887	325	943113467309
Alicia	999887777	948482887887	325	539608253046
Alicia	999887777	948482887887	325	578345624945



*Getting history

name	ssn	anumber	amount	sent_account
Joyce	453453453	964462351156	25	850986092469
Joyce	453453453	964462351156	25	850986092469
James	888665555	539608253046	25	964462351156
Ramesh	666884444	943113467309	125	964462351156
Joyce	453453453	964462351156	225	729641348006
Joyce	453453453	964462351156	325	729641348006
John	123456789	578345624945	225	539608253046
Ramesh	666884444	943113467309	225	539608253046
James	888665555	539608253046	225	366947237264
John	123456789	578345624945	225	366947237264
John	123456789	578345624945	325	943113467309
James	888665555	539608253046	325	943113467309
Ramesh	666884444	943113467309	325	948482887887
James	888665555	539608253046	325	948482887887
James	888665555	539608253046	325	578345624945
Ramesh	666884444	943113467309	125	578345624945
James	888665555	539608253046	125	850986092469
Ramesh	666884444	943113467309	125	850986092469
John	123456789	578345624945	25	964462351156
James	888665555	539608253046	25	729641348006
Ramesh	666884444	943113467309	125	729641348006
Jennifer	987654321	729641348006	325	539608253046
Ramesh	666884444	943113467309	225	366947237264
John	123456789	578345624945	225	943113467309
Jennifer	987654321	729641348006	225	943113467309
John	123456789	578345624945	325	948482887887
James	888665555	539608253046	325	578345624945
Jennifer	987654321	729641348006	325	578345624945
James	888665555	539608253046	100	578345624945



*DEPOSIT history

name	ssn	anumber	amount
John	123456789	111111111111	9999
John	123456789	578345624945	100
John	123456789	578345624945	300
John	123456789	578345624945	200
John	123456789	578345624945	300
John	123456789	578345624945	500
Franklin	333445555	366947237264	200
Franklin	333445555	366947237264	400
Franklin	333445555	366947237264	100
Franklin	333445555	366947237264	300
Joyce	453453453	964462351156	300
Joyce	453453453	964462351156	400
Joyce	453453453	964462351156	300
Joyce	453453453	964462351156	400
Ramesh	666884444	943113467309	100
Ramesh	666884444	943113467309	400
Ramesh	666884444	943113467309	100
Ramesh	666884444	943113467309	300
James	888665555	539608253046	300
James	888665555	539608253046	400
James	888665555	539608253046	100
James	888665555	539608253046	400
Jennifer	987654321	729641348006	400
Jennifer	987654321	729641348006	400
Jennifer	987654321	729641348006	200
Jennifer	987654321	729641348006	400
Alicia	999887777	948482887887	100
Alicia	999887777	948482887887	300
Alicia	999887777	948482887887	200
Alicia	999887777	948482887887	300

```
*WITHDRAW history
-----------------------------------------
name     ssn          anumber         amount
-----------------------------------------
John     123456789    578345624945    450
John     123456789    578345624945    250
John     123456789    578345624945    650
John     123456789    578345624945    650
John     123456789    578345624945    250
Franklin 333445555    366947237264    450
Franklin 333445555    366947237264    650
Franklin 333445555    366947237264    50
Franklin 333445555    366947237264    650
Joyce    453453453    964462351156    50
Joyce    453453453    964462351156    450
Joyce    453453453    964462351156    250
Joyce    453453453    964462351156    450
Ramesh   666884444    943113467309    650
Ramesh   666884444    943113467309    650
Ramesh   666884444    943113467309    250
Ramesh   666884444    943113467309    650
James    888665555    539608253046    250
James    888665555    539608253046    650
James    888665555    539608253046    50
James    888665555    539608253046    450
Jennifer 987654321    729641348006    50
Jennifer 987654321    729641348006    650
Jennifer 987654321    729641348006    50
Jennifer 987654321    729641348006    450
Alicia   999887777    948482887887    450
Alicia   999887777    948482887887    650
Alicia   999887777    948482887887    450
Alicia   999887777    948482887887    650
-----------------------------------------
```

```
*LOAN history
-------------------------------------------
name     ssn          anumber         amount     term         interest
-------------------------------------------
Joyce    453453453    964462351156    300000     2023-04-01   1.19
James    888665555    539608253046    530000     2022-06-09   1.29
Franklin 333445555    366947237264    123000     2021-08-27   1.17
Alicia   999887777    948482887887    240000     2023-10-03   1.25
John     123456789    578345624945    500000     2020-12-13   1.2
Jennifer 987654321    729641348006    500        2019-01-01   1.05
-------------------------------------------
```

```
Manager ssn : 234234
You are not manager.
```

- 매니저메뉴 2번 View a user's transaction history

```
mgrssn = mgrSsn();

int salary = 0;

rs = stmt.executeQuery("select salary from manager where ssn=" + mgrssn);

if(rs.next())    salary = rs.getInt(1);
if(salary == 0) {
    System.out.println("You are not manager.");
    rs.close();
    continue;
}

userssn = userSsn();

salary = 0;

rs = stmt.executeQuery("select salary from user where ssn=" + userssn);

if(rs.next())    salary = rs.getInt(1);
if(salary == 0) {
    System.out.println("The user does not exist.");
    rs.close();
    continue;
}

System.out.println("*REMITTANCE history");
rs = stmt.executeQuery( "select name, ramount, getting_account, raccount "
        + " from user, account, remittance "
        + " where anumber = raccount and ussn=ssn and ssn=" + userssn);

System.out.println("-------------------------------------------");
System.out.println("name\t anumber  \tamount\t   getting_account");
System.out.println("-------------------------------------------");
while( rs.next() ) {
    String name = rs.getString(1);
    long anumber = rs.getLong(4);
    int amount = rs.getInt(2);
    long getting_account = rs.getLong(3);
    System.out.println(name + "\t " + anumber + "\t" + amount + "\t    " + getting_account);
```

```
}
System.out.println("-----------------------------------------");
```

```
System.out.println("*Getting history");
rs = stmt.executeQuery( "select name, ssn, ramount, getting_account, raccount"
        + " from user, account, remittance "
        + " where anumber = getting_account and ussn=ssn and ssn=" + userssn);

System.out.println("-----------------------------------------");
System.out.println("name\t anumber  \tamount\t   sent_account");
System.out.println("-----------------------------------------");
while( rs.next() ) {
    String name = rs.getString(1);
    long anumber = rs.getLong(2);
    int amount = rs.getInt(3);
    long sent_account = rs.getLong(4);
    System.out.println(name + "\t " + anumber + "\t" + amount + "\t   " + sent_account);
}
System.out.println("-----------------------------------------");

System.out.println("*DEPOSIT history");
rs = stmt.executeQuery( "select name, damount, daccount from user, account, deposit "
        + "where anumber = daccount and ussn=ssn and ssn=" + userssn);

System.out.println("-----------------------------------------");
System.out.println("name\t anumber  \tamount\t   ");
System.out.println("-----------------------------------------");
while( rs.next() ) {
    String name = rs.getString(1);
    long anumber = rs.getLong(3);
    int amount = rs.getInt(2);
    System.out.println(name + "\t " + anumber + "\t" + amount);
}
System.out.println("-----------------------------------------");

System.out.println("*WITHDRAW history");
rs = stmt.executeQuery( "select name, wamount, waccount from user, account, withdraw "
        + "where anumber = waccount and ussn=ssn and ssn=" + userssn);

System.out.println("-----------------------------------------");
System.out.println("name\t anumber  \tamount\t   ");
System.out.println("-----------------------------------------");
while( rs.next() ) {
    String name = rs.getString(1);
    long anumber = rs.getLong(3);
    int amount = rs.getInt(2);
    System.out.println(name + "\t " + anumber + "\t" + amount);
}
System.out.println("-----------------------------------------");
```

```
System.out.println("*LOAN history");
rs = stmt.executeQuery( "select name, lamount, lterm, linterest, laccount "
        + " from user, account, loan "
        + " where anumber = laccount and ussn=ssn and ssn=" + userssn);

System.out.println("-----------------------------------------");
System.out.println("name\t anumber  \tamount\t   term\t\tinterest");
System.out.println("-----------------------------------------");
```

```
while( rs.next() ) {
    String name = rs.getString(1);
    long anumber = rs.getLong(5);
    int amount = rs.getInt(2);
    Date term = rs.getDate(3);
    float interest = rs.getFloat(4);
    System.out.println(name + "\t " + anumber + "\t" + amount + "\t   " + term + "\t" + interest);
}
System.out.println("-----------------------------------------");
rs.close();
```

=> 한 유저의 거래내역을 보는 메뉴입니다.

=> 한 유저의 송금내역, 송금받은 내역, 입금내역, 출금내역, 대출내역을 조회합니다.

=> SQL

"select salary from manager where ssn=" + *mgrssn*

-> manager 테이블에 저장된 ssn과 입력받은 mgrssn이 같은 사람을 찾습니다. 만약 없다면 그 사람은 매니저가 아니므로 오류메시지(You are not manager.)를 출력합니다.

"select salary from user where ssn=" + *userssn*

-> user 테이블에 저장된 ssn과 입력받은 ssn이 같은 사람을 찾습니다. 만약 없다면 오류메시지(The user does not exist.)를 출력합니다.

"select name, ramount, getting_account, raccount " + " from user, account, remittance " + " where anumber = raccount and ussn=ssn and ssn=" + *userssn*

-> user, account, remittance 테이블에 저장된 anumber와 raccount가 같고 ussn과 ssn이 같고 ssn이 입력받은 userssn과 같은 터플들을 뽑아냅니다.

"select name, ssn, ramount, getting_account, raccount" + " from user, account, remittance " + " where anumber = getting_account and ussn=ssn and ssn=" + *userssn*

-> user, account, remittance 테이블에 저장된 anumber와 getting_account가 같고 ussn과 ssn이 같고 ssn이 입력받은 userssn과 같은 터플들을 뽑아냅니다.

"select name, damount, daccount from user, account, deposit " + "where anumber = daccount and ussn=ssn and ssn=" + *userssn*

-> user, account, deposit 테이블에 저장된 anumber와 daccount가 같고 ussn과 ssn이 같고 ssn이 입력받은 userssn과 같은 터플들을 뽑아냅니다.

"select name, wamount, waccount from user, account, withdraw " + "where anumber = waccount and ussn=ssn and ssn=" + *userssn*

-> user, account, withdraw 테이블에 저장된 anumber와 waccount가 같고 ussn과 ssn이 같고 ssn이 입력받은 userssn과 같은 터플들을 뽑아냅니다.

"select name, lamount, lterm, linterest, laccount " + " from user, account, loan " + " where anumber = laccount and ussn=ssn and ssn=" + *userssn*

-> user, account, loan 테이블에 저장된 anumber와 laccount가 같고 ussn과 ssn이 같고 ssn이 입력받은 userssn과 같은 터플들을 뽑아냅니다.

=> 실행 예시

```
Manager ssn : 234234
You are not manager.
```

```
Manager ssn : 234234234
User ssn : 33
The user does not exist.
```

```
Manager ssn : 234234234
User ssn : 123456789
*REMITTANCE history
-------------------------------------------
name     anumber        amount     getting_account
-------------------------------------------
John     578345624945   325        539608253046
John     578345624945   125        943113467309
John     578345624945   325        539608253046
John     578345624945   325        729641348006
John     578345624945   100        539608253046
-------------------------------------------
*Getting history
-------------------------------------------
name     anumber        amount     sent_account
-------------------------------------------
John     123456789      225        578345624945
John     123456789      225        578345624945
John     123456789      325        578345624945
John     123456789      25         578345624945
John     123456789      225        578345624945
John     123456789      325        578345624945
-------------------------------------------
```

```
*DEPOSIT history
-------------------------------------------
name     anumber        amount
-------------------------------------------
John     111111111111   9999
John     578345624945   100
John     578345624945   300
John     578345624945   200
John     578345624945   300
John     578345624945   500
-------------------------------------------
*WITHDRAW history
-------------------------------------------
name     anumber        amount
-------------------------------------------
John     578345624945   450
John     578345624945   250
John     578345624945   650
John     578345624945   650
John     578345624945   250
-------------------------------------------
*LOAN history
-------------------------------------------
name     anumber        amount     term         interest
-------------------------------------------
John     578345624945   500000     2020-12-13   1.2
-------------------------------------------
```

- 매니저메뉴 3번 View all users' account

```
mgrssn = mgrSsn();

int salary = 0;

rs = stmt.executeQuery("select salary from manager where ssn=" + mgrssn);

if(rs.next())    salary = rs.getInt(1);
if(salary == 0) {
    System.out.println("You are not manager.");
    rs.close();
    continue;
}

rs = stmt.executeQuery( "select name, ssn, anumber, balance, term, interest"
        + " from user, account where ussn=ssn");

System.out.println("*If term is null and interest is 0.0, account is ordinary deposit.");
System.out.println("*If not, account is time deposit.");
System.out.println("-------------------------------------------");
System.out.println("name\t ssn\t\tanumber  \tbalance\t   term\t\tinterest");
System.out.println("-------------------------------------------");
while( rs.next() ) {
    String name = rs.getString(1);
    int ssn = rs.getInt(2);
    long anumber = rs.getLong(3);
    long balance = rs.getLong(4);
    Date term = rs.getDate(5);
    float interest = rs.getFloat(6);
    if(term == null) {
        System.out.println(name + "\t " + ssn + "\t" + anumber + "\t" + balance + "\t    "
                                + term + "\t\t" + interest);
    }
    else{
```

```
        System.out.println(name + "\t " + ssn + "\t" + anumber + "\t" + balance + "\t    "
                                + term + "\t" + interest);
    }
}
System.out.println("-----------------------------------------");
rs.close();
```

=> 모든 유저의 계좌를 보는 메뉴입니다.

=> SQL

"select salary from manager where ssn=" + *mgrssn*

-> manager 테이블에 저장된 ssn과 입력받은 mgrssn이 같은 사람을 찾습니다. 만약 없다면 그 사람은 매니저가 아니므로 오류메시지(You are not manager.)를 출력합니다.

"select name, ssn, anumber, balance, term, interest" + " from user, account where ussn=ssn"

-> user, account 테이블에 저장된 ussn과 ssn이 같은 터플들을 뽑아냅니다.

=> 실행 예시

```
Manager ssn : 234234234
*If term is null and interest is 0.0, account is ordinary deposit.
*If not, account is time deposit.
-----------------------------------------
name     ssn            anumber         balance    term         interest
-----------------------------------------
John     123456789      111111111111    9999       2020-01-01   1.01
John     123456789      222222222222    0          null         0.0
John     123456789      478347894945    1400000    2020-09-01   1.13
John     123456789      578345624945    250150     null         0.0
Franklin 333445555      216942137264    2430000    2020-03-01   1.07
Franklin 333445555      366947237264    300000     null         0.0
Joyce    453453453      964189181156    1000000    2020-05-14   1.07
Joyce    453453453      964462351156    430000     null         0.0
Ramesh   666884444      943110567309    1550000    2020-01-04   1.12
Ramesh   666884444      943113467309    250000     null         0.0
James    888665555      539134653046    4250000    2020-02-05   1.04
James    888665555      539608253046    380100     null         0.0
Jennifer 987654321      599168678006    3250000    2020-05-01   1.03
Jennifer 987654321      729641348006    400500     null         0.0
Ahmad    987987987      273986092469    1250000    2020-05-09   1.07
Alicia   999887777      713682887887    1380000    2020-02-13   1.2
Alicia   999887777      948482887887    250000     null         0.0
-----------------------------------------
```

- 매니저메뉴 4번 View a user's account

```
mgrssn = mgrSsn();

int salary = 0;

rs = stmt.executeQuery("select salary from manager where ssn=" + mgrssn);

if(rs.next())    salary = rs.getInt(1);
if(salary == 0) {
    System.out.println("You are not manager.");
    rs.close();
    continue;
}
```

```
userssn = userSsn();

salary = 0;

rs = stmt.executeQuery("select salary from user where ssn=" + userssn);

if(rs.next())    salary = rs.getInt(1);
if(salary == 0) {
    System.out.println("The user does not exist.");
    rs.close();
    continue;
}

rs = stmt.executeQuery( "select name, anumber, balance, term, interest"
        + " from user, account where ussn=ssn and ssn = " + userssn);

System.out.println("*If term is null and interest is 0.0, account is ordinary deposit.");
System.out.println("*If not, account is time deposit.");
System.out.println("-----------------------------------------");
System.out.println("name\t anumber  \tbalance\t   term\t\tinterest");
System.out.println("-----------------------------------------");
while( rs.next() ) {
    String name = rs.getString(1);
    long anumber = rs.getLong(2);
    long balance = rs.getLong(3);
    Date term = rs.getDate(4);
    float interest = rs.getFloat(5);
    if(term == null) {
        System.out.println(name + "\t " + anumber + "\t" + balance + "\t   "
                           + term + "\t\t" + interest);
    }
    else{
        System.out.println(name + "\t " + anumber + "\t" + balance + "\t   "
                           + term + "\t" + interest);
    }
}
System.out.println("-----------------------------------------");
rs.close();
```

=> 한 유저의 계좌를 보는 메뉴입니다.

=> SQL

`"select salary from manager where ssn=" + mgrssn`

-> manager 테이블에 저장된 ssn과 입력받은 mgrssn이 같은 사람을 찾습니다. 만약 없다면 그 사람은 매니저가 아니므로 오류메시지(You are not manager.)를 출력합니다.

`"select salary from user where ssn=" + userssn`

-> user 테이블에 저장된 ssn과 입력받은 ssn이 같은 사람을 찾습니다. 만약 없다면 오류메시지(The user does not exist.)를 출력합니다.

`"select name, anumber, balance, term, interest" + " from user, account where ussn=ssn and ssn = " + userssn`

-> user, account 테이블에 저장된 ussn과 ssn이 같고 ssn과 입력받은 userssn이 같은 터플들을 뽑아냅니다.

=> 실행 예시

```
Manager ssn : 234234234
User ssn : 123456789
*If term is null and interest is 0.0, account is ordinary deposit.
*If not, account is time deposit.
-----------------------------------------
name     anumber        balance    term         interest
-----------------------------------------
John     111111111111   9999       2020-01-01   1.01
John     222222222222   0          null         0.0
John     478347894945   1400000    2020-09-01   1.13
John     578345624945   250150     null         0.0
-----------------------------------------
```

컴파일 방법

1. mariadb에서 bank 데이터베이스를 dump파일로 추출합니다.
   => mysqldump -u jinkyo422 -p bank > bank.sql

2. 실행하려는 pc(windows)의 mariadb에 bank 데이터베이스를 만듭니다.
   => create database bank;

3. 추출한 dump파일을 사용하여 mariadb로 복구합니다.
   => mysql -u jinkyo422 -p bank < C:₩Users₩jinkyo₩Desktop₩bank.sql

4. dbproj.java와 mariadb-java-client-2.3.0.jar를 컴파일합니다.
   => javac -cp C:₩Users₩jinkyo₩Desktop₩mariadb-java-client-2.3.0.jar dbproj.java

5. 생성된 dbproj.class를 실행합니다.
   => java -cp C:₩Users₩jinkyo₩Desktop₩mariadb-java-client-2.3.0.jar;. dbproj

   과제에서 요구하는 CLI환경을 구현하였습니다.

```
C:₩Users₩jinkyo₩Desktop>java -cp C:₩Users₩jinkyo₩Desktop₩mariadb-java-client-2.3.0.jar;. dbproj
0. Exit
1. User Menu
2. Manager Menu
Input : 1
0. Return to previous menu
1. View account
2. REMITTANCE
3. DEPOSIT
4. WITHDRAW
5. LOAN
6. View transaction history
7. Make an account
8. Remove an account
Input : 1
```