JAVA 프로젝트

통합계좌관리 프로그램

데이터분석과 구본성

CONTENTS

CONTENTS

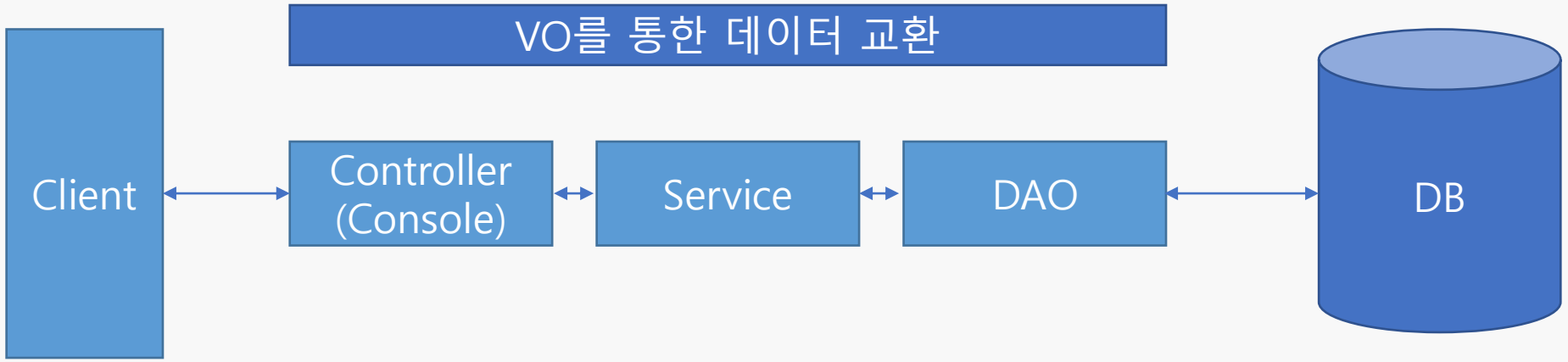
프로젝트 개요

- ✓ **통합 계좌 관리 프로그램**
  - ✓ 언어 : JAVA
  - ✓ DB : ORACLE DB 12c
  - ✓ 개발자 : 데이터분석과 구본성
  - ✓ 개발기간 : 2021. 05. 31 ~ 2021. 06. 04

프로젝트 주제

- ✓ 정해진 은행에 속해 있는 계좌를 통합관리 할 수 있는 프로그램
- ✓ 어떤 은행인지에 대해 구애 받지 않고 계좌 생성, 수정, 삭제가 용이
- ✓ 입금, 출금, 계좌이체 구현

프로그램 구성



DB 설계

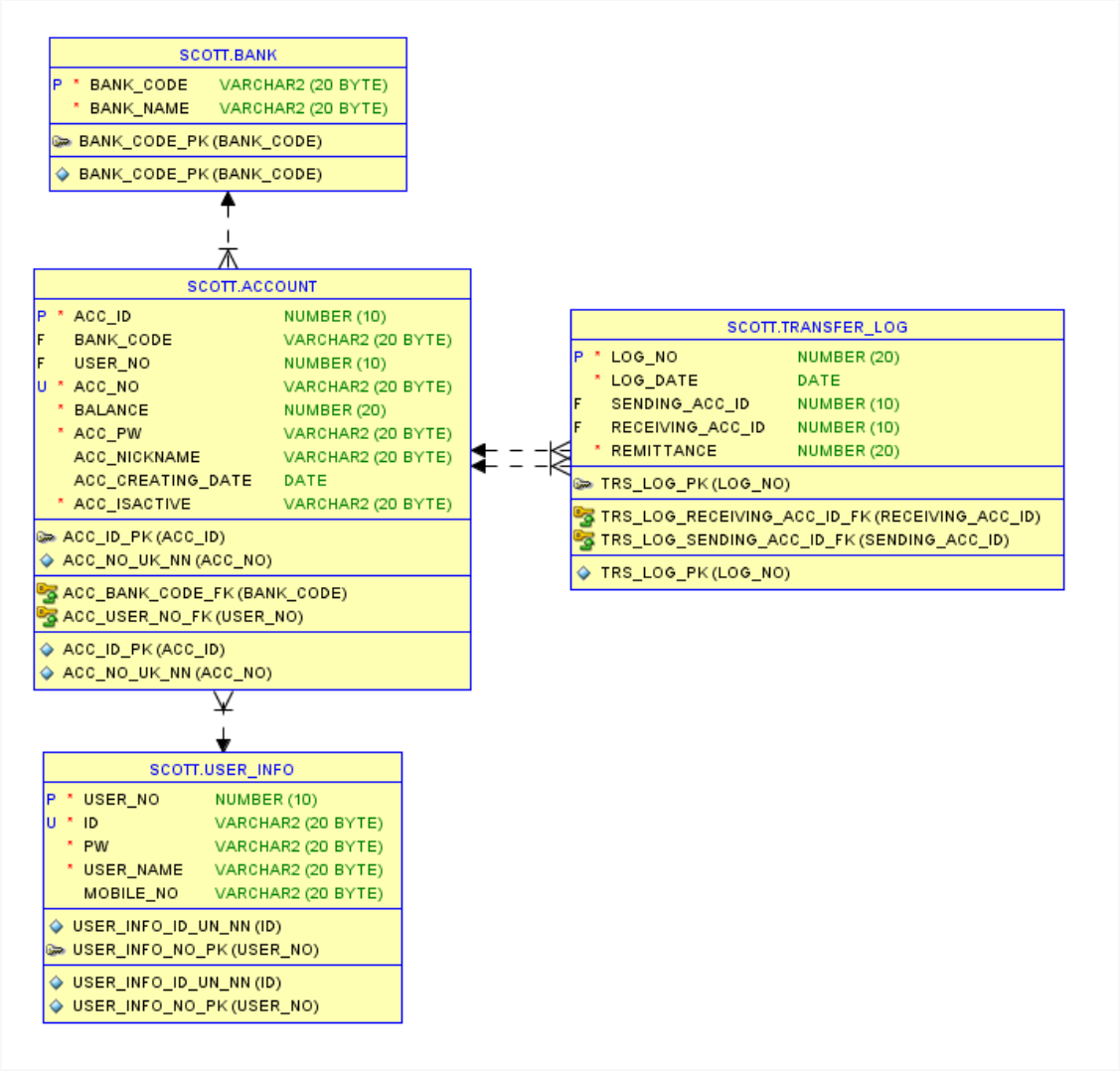
- ✓ 필요한 요소
  - ✓ 유저의 정보
  - ✓ 유저가 가지고 있는 계좌에 대한 정보
  - ✓ 계좌 이체 시 보내는 사람과 받는 사람의 정보와 계좌에 대한 정보
  - ✓ App이 허용하는 은행에 대한 정보

- ✓ 총 4개의 테이블로 DB 구성

DB 설계

- ✓ USER_INFO : 유저의 정보
- ✓ ACCOUNT : 유저가 가지고 있는 계좌에 대한 정보
- ✓ TRANSFER_LOG : 계좌 이체 시 보내는 사람과 받는 사람의 정보와 계좌에 대한 정보
- ✓ BANK : App이 허용하는 은행에 대한 정보

- ✓ 각 테이블은 FK를 통해 참조



USER_INFO

- ✓ USER_NO (PK) : 유저에 대한 순번
  - ✓ SEQUENCE
- ✓ ID, PW : 로그인에 필요한 정보
- ✓ USER_NAME : 이름
- ✓ MOBILE_NO : 휴대전화 번호

	USER_NO	ID	PW	USER_NAME	MOBILE_NO
1	1	test	test	구본성	010-4773-9853
2	2	test1	test1	유저1	010-1234-5678
3	3	test2	test2	유저2	010-2345-6789
4	4	test3	test3	유저3	010-3456-7890
5	5	test4	test4	유저4	010-4567-8901
6	6	test5	test5	유저5	010-4567-8901

ACCOUNT

- ✓ ACC_ID (PK) : 유저에 대한 순번 (SEQUENCE)
- ✓ BANK_CODE (FK) : 은행코드, 허용한 은행만 DB INSERT 가능
- ✓ USER_NO (FK) : USER 테이블에 존재하는 인원에 대한 USER 번호
- ✓ BALANCE : 잔액 / ACC_PW : 계좌 비밀번호
- ✓ ACC_NICKNAME : 계좌 별칭 / ACC_CREATING_DATE : 계좌 생성일시
- ✓ ACC_ISACTIVE : 계좌 상태 (활성화 상태 / 삭제 상태)

	ACC_ID	BANK_CODE	USER_NO	ACC_NO	BALANCE	ACC_PW	ACC_NICKNAME	ACC_CREATING_DATE	ACC_ISACTIVE
1	1	1000	1	111-222-333	1000000	1234	주택청약	16/01/01	ACTIVE
2	2	2000	2	222-555-666	2000000	4321	적금	17/01/01	ACTIVE
3	3	3000	3	333-888-999	3000000	1234	비트코인	18/01/01	ACTIVE
4	4	3000	3	333-345-678	3000000	1111	비상금	19/01/01	ACTIVE
5	5	4000	4	444-645-545	4000000	1111	밥값	21/06/01	ACTIVE
6	6	4000	1	444-625-545	4000000	1111	확인용	21/06/02	ACTIVE

BANK

- ✓ BANK_CODE (PK) : 은행 고유 코드
  - ✓ 임의로 부여
  - ✓ 하나 : 1000 / 신한 : 2000 / KB : 3000 / 우리 : 4000
- ✓ BANK_NAME : 은행 이름

	BANK_CODE	BANK_NAME
1	1000	HANA
2	2000	SHINHAN
3	3000	KB
4	4000	WOORI

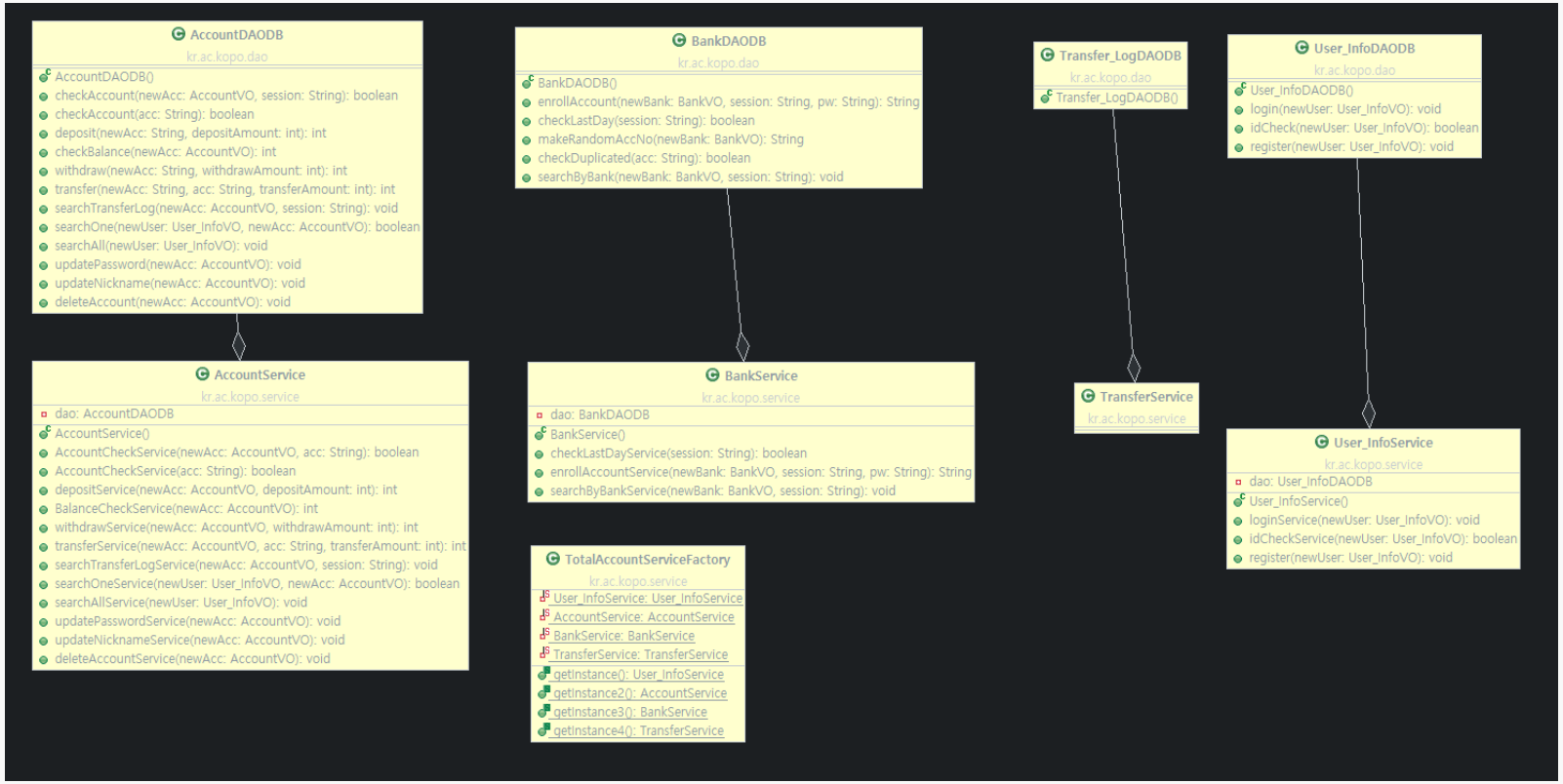
TRANSFER_LOG

- ✓ LOG_NO (PK) : 이체기록에 대한 로그 번호
  - ✓ SEQUENCE
- ✓ LOG_DATE : 로그 기록 일시 (이체일시)
- ✓ SENDING_ACC_ID (FK)
  : 보내는 사람의 계좌에 대한 ID
- ✓ RECEIVING_ACC_ID (FK)
  : 받는 사람의 계좌에 대한 ID
- ✓ REMITTANCE : 송금액

	LOG_NO	LOG_DATE	SENDING_ACC_ID	RECEIVING_ACC_ID	REMITTANCE
1	1	21/06/04	1	2	10000
2	2	21/06/04	1	2	20000
3	3	21/06/04	1	3	50000
4	4	21/06/04	2	3	40000
5	5	21/06/04	4	1	50000

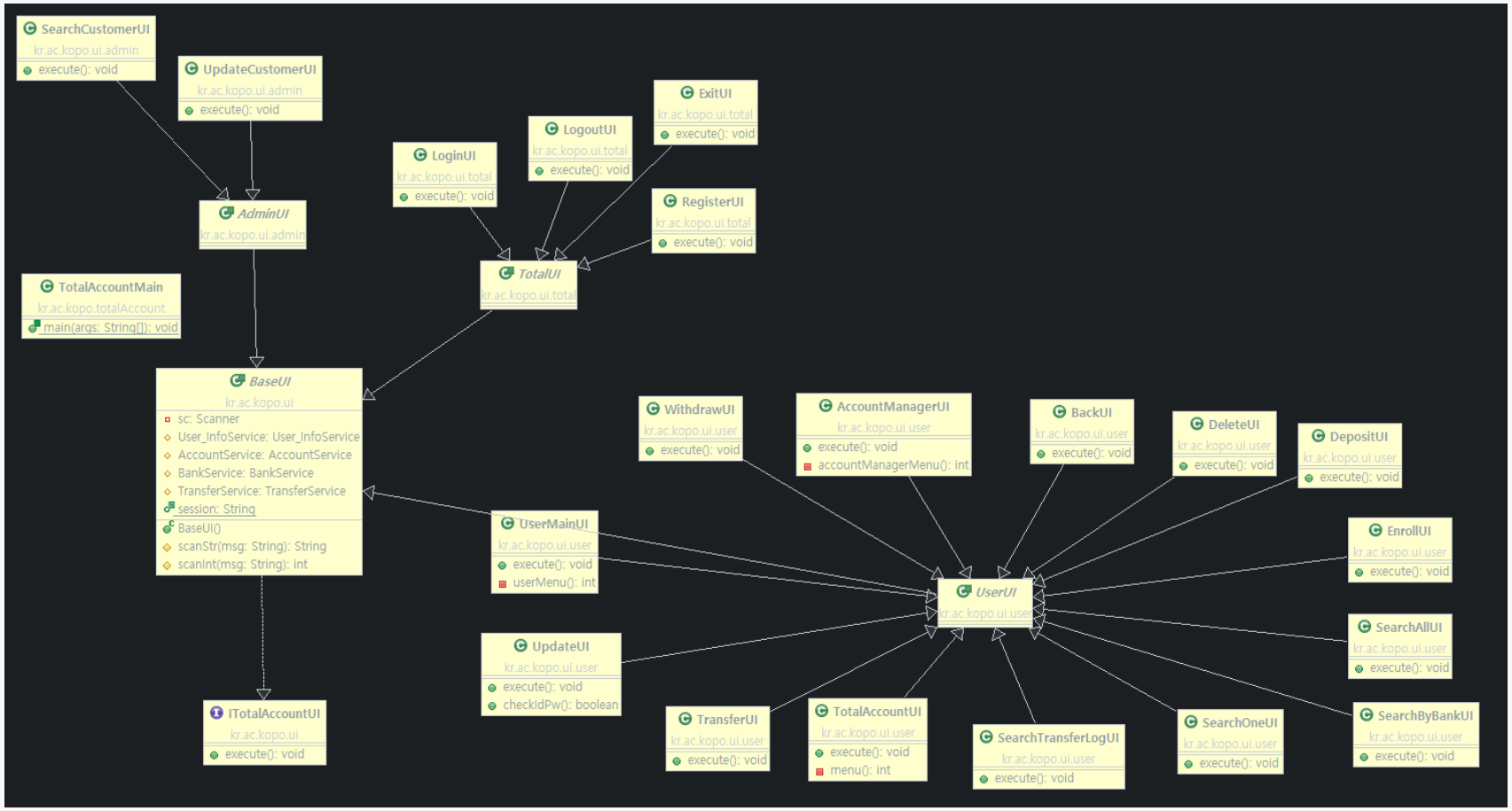
SERVICE-DAO

- ✓ 서비스에서 적절한 비즈니스 로직 처리
- ✓ DAO에서 서비스로부터 VO를 통해 데이터를 전달받고 DB에 접근한 뒤 적절한 데이터를 반환



UI 설계

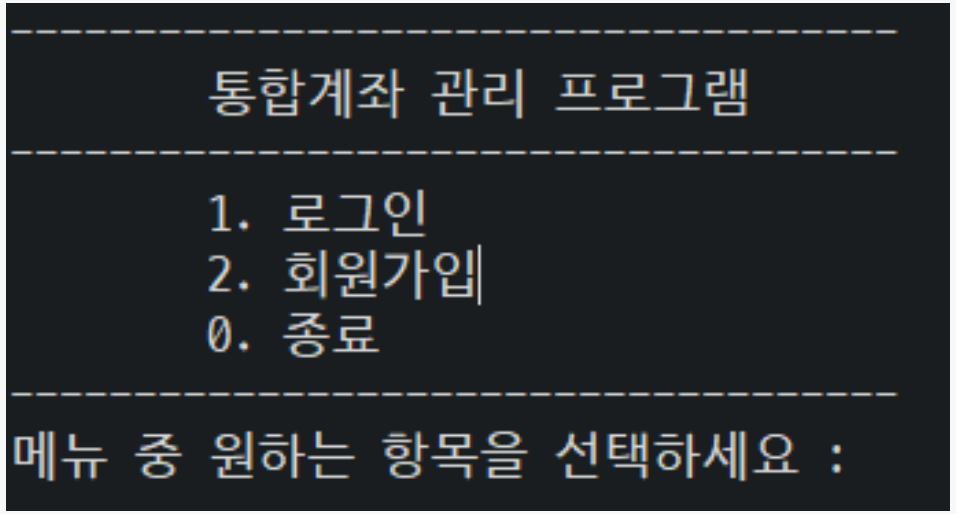
- ✓ 공통으로 구현하는 메소드나 정의 부분을 상위 클래스에 배치
- ✓ 공통 UI, 유저UI, 관리자 UI로 구분
  (관리자 미구현)



메인 UI

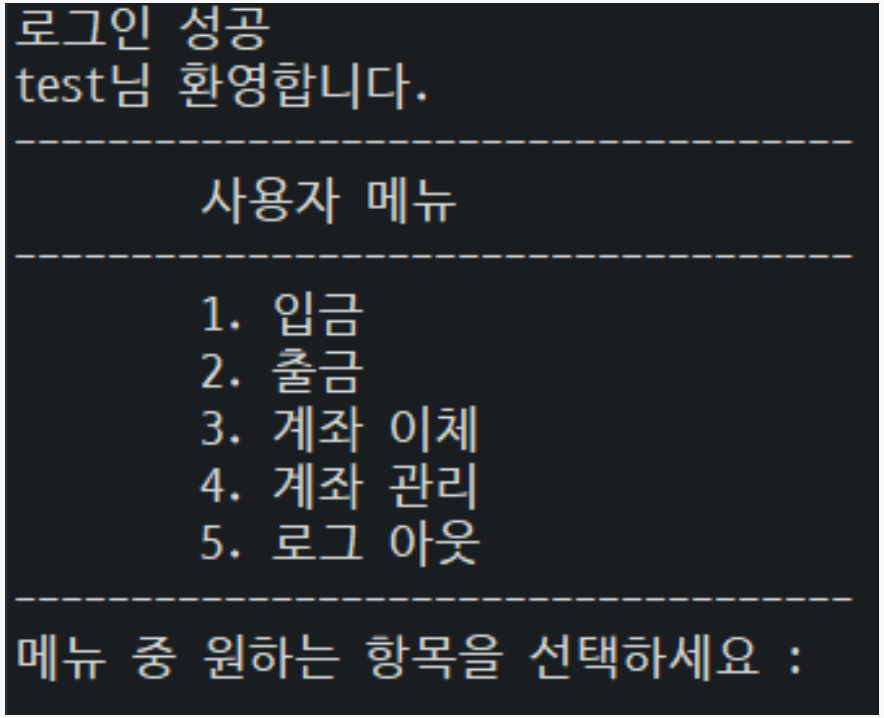
- ✓ [로그인]
  - ✓ USER DB의 정보와 일치하면 로그인 가능

- ✓ [회원가입]
  - ✓ 겹치지 않는 ID에 한해
    USER DB에 ID와 PW 정보를 입력



사용자 UI

- ✓ [입금], [출금]
- ✓ [계좌 이체]
- ✓ [계좌 관리]
  - ✓ [계좌 생성]
  - ✓ [계좌 조회] (상세, 은행별, 전체 계좌, 계좌 이체 내역)
  - ✓ [계좌 수정]
  - ✓ [계좌 삭제]
- ✓ [로그 아웃] : 세션을 종료하고 메인 화면으로 돌아 감
- ✓ 모든 트랜잭션은 에러, 예외 시 ROLLBACK



사용자 UI

- ✓ [입금]
  - ✓ 자신의 계좌 중 원하는 계좌에 돈을 넣음

  - ✓ 불가한 경우
    - ✓ 해당 계좌가 DB에 없거나
      본인 소유의 계좌가 아닌 경우
    - ✓ 0 이하의 값을 입력한 경우

```
입금 서비스 입니다.
계좌번호를 입력해주세요 ( - 포함 ) : 111-222-333
계좌 비밀번호를 입력해주세요 : 1234
해당 계좌가 존재합니다
얼마나 입금하시겠습니까? : 10000
입금이 완료되었습니다.
```

```
입금 서비스 입니다.
계좌번호를 입력해주세요 ( - 포함 ) : 123
계좌 비밀번호를 입력해주세요 : 1
계좌가 존재하지 않습니다.
```

```
입금 서비스 입니다.
계좌번호를 입력해주세요 ( - 포함 ) : 111-222-333
계좌 비밀번호를 입력해주세요 : 1234
해당 계좌가 존재합니다
얼마나 입금하시겠습니까? : -100
입력하신 값이 올바르지 않습니다.
```

사용자 UI

- ✓ [출금]
  - ✓ 자신의 계좌 중 원하는 계좌에 돈을 빼냄
  - ✓ 불가한 경우
    - ✓ 해당 계좌가 DB에 없거나
      본인 소유의 계좌가 아닌 경우
    - ✓ 출금액이 잔액보다 많은 경우
    - ✓ 0 이하의 값을 입력한 경우

```
출금 서비스 입니다.
계좌번호를 입력해주세요 ( - 포함) : 111-222-333
계좌 비밀번호를 입력해주세요 : 1234
해당 계좌가 존재합니다
얼마나 출금하시겠습니까? : 10000
출금이 완료되었습니다.
```

```
출금 서비스 입니다.
계좌번호를 입력해주세요 ( - 포함) : 12341234
계좌 비밀번호를 입력해주세요 : 1111
해당 계좌가 존재하지 않습니다.
```

```
출금 서비스 입니다.
계좌번호를 입력해주세요 ( - 포함) : 111-222-333
계좌 비밀번호를 입력해주세요 : 1234
해당 계좌가 존재합니다
얼마나 출금하시겠습니까? : 10000
잔액이 부족합니다
```

```
출금 서비스 입니다.
계좌번호를 입력해주세요 ( - 포함) : 111-222-333
계좌 비밀번호를 입력해주세요 : 1234
해당 계좌가 존재합니다
얼마나 출금하시겠습니까? : -300
입력하신 값이 올바르지 않습니다.
```

사용자 UI

- ✓ [계좌 이체]
  - ✓ 자신의 계좌 중 원하는 계좌에서 출금하여 상대방의 계좌에 입금함
  - ✓ 불가한 경우
    - ✓ 해당 계좌가 DB에 없거나 본인 소유의 계좌가 아닌 경우 출금을 할 수 없음
    - ✓ 동일한 계좌를 입력할 경우
    - ✓ 이체액이 잔액보다 많은 경우

```
계좌이체 서비스 입니다.
계좌번호를 입력해주세요 ( - 포함) : 111-222-333
계좌 비밀번호를 입력해주세요 : 1234
해당 계좌가 존재합니다
이체할 계좌번호를 입력해주세요 ( - 포함) : 222-555-666
얼마나 이체하시겠습니까? : 4321
이체에 성공했습니다.
```

```
계좌이체 서비스 입니다.
계좌번호를 입력해주세요 ( - 포함) : 111-222-333
계좌 비밀번호를 입력해주세요 : 1233
본인 소유의 계좌가 존재하지 않습니다.
```

```
계좌이체 서비스 입니다.
계좌번호를 입력해주세요 ( - 포함) : 111-222-333
계좌 비밀번호를 입력해주세요 : 1234
해당 계좌가 존재합니다
이체할 계좌번호를 입력해주세요 ( - 포함) : 111-222-333
동일한 계좌에는 이체하실 수 없습니다.
```

```
계좌이체 서비스 입니다.
계좌번호를 입력해주세요 ( - 포함) : 111-222-333
계좌 비밀번호를 입력해주세요 : 1234
해당 계좌가 존재합니다
이체할 계좌번호를 입력해주세요 ( - 포함) : 222-555-666
얼마나 이체하시겠습니까? : 50000000
잔액이 부족합니다
```

사용자 UI

- ✓ [계좌 관리] – [계좌 생성]
  - ✓ 원하는 은행의 계좌를 생성함
  - ✓ 계좌 번호는 [은행별 고유 계좌 앞자리] – [무작위 3자리] – [무작위 3자리] 총 9자리로 구성
    (하나 : 111 신한 : 222 KB : 333 우리 : 444)

  - ✓ 불가한 경우
    - ✓ 비밀번호가 일치하지 않는 경우
    - ✓ 최근 30일 이내에 개설한 계좌가 있는 경우

```
계좌 생성 서비스 입니다.
원하는 은행명을 입력해주세요 :
1. 하나  2. 신한  3. KB  4. 우리 : 1
원하는 비밀번호를 입력해주세요 : 1234
다시 한번 입력해주세요 : 1234
고객님의 계좌번호는 [ 111-681-125 ] 입니다.
```

```
계좌 생성 서비스 입니다.
원하는 은행명을 입력해주세요 :
1. 하나  2. 신한  3. KB  4. 우리 : 1
원하는 비밀번호를 입력해주세요 : 1234
다시 한번 입력해주세요 : 4321
비밀번호가 일치하지 않습니다.
```

```
메뉴 중 원하는 항목을 선택하세요 : 1
고객님의 최근 30일 이내에 개설한 계좌가 존재합니다.
서비스를 이용할 수 없습니다.
```

사용자 UI

- ✓ [계좌 관리] – [계좌 상세 조회]
  - ✓ 자신의 계좌 중 원하는 계좌의 상세 정보를 조회할 수 있음

  - ✓ 불가한 경우
    - ✓ 해당 계좌가 DB에 없거나 본인 소유의 계좌가 아닌 경우

```
메뉴 중 원하는 항목을 선택하세요 : 2
계좌 상세 조회 서비스입니다.
계좌 번호를 입력해주세요 : 111-222-333
비밀번호를 입력해주세요 : 1234
은행명 : HANA
고객명 : 구본성
계좌번호 : 111-222-333
잔액 : 95679
계좌별칭 : 주택청약
계좌생성일자 : 2016-01-01
```

```
계좌 상세 조회 서비스입니다.
계좌 번호를 입력해주세요 : 222-333-444
비밀번호를 입력해주세요 : 4321
입력하신 본인 소유 계좌가 존재하지 않습니다.
```

사용자 UI

- ✓ [계좌 관리] – [은행별 계좌 조회]
  - ✓ 자신의 계좌 중 원하는 은행에 소속되어 있는 계좌를 조회할 수 있음
  - ✓ 계좌 번호, 잔액 등 간략한 정보만 표시
  - ✓ 불가한 경우
    - ✓ 해당 계좌가 DB에 없거나 본인 소유의 계좌가 아닌 경우

```
메뉴 중 원하는 항목을 선택하세요 : 3
은행별 계좌 조회 서비스입니다.
원하는 은행명을 입력해주세요 :
1. 하나  2. 신한  3. KB  4. 우리 : 1
은행명  계좌번호          잔액
HANA    111-451-680       1000
HANA    111-222-333       95679
HANA    111-222-444       1000000
HANA    111-222-555       1000000
HANA    111-681-125       1000
```

사용자 UI

- ✓ [계좌 관리] – [전체 계좌 조회]
  - ✓ 자신이 소유하고 있는 모든 계좌 정보 조회
    계좌 번호, 잔액 등 간략한 정보만 표시

  - ✓ 불가한 경우
    - ✓ 해당 계좌가 DB에 없거나
      본인 소유의 계좌가 아닌 경우

```
메뉴 중 원하는 항목을 선택하세요 : 4
전체 계좌 조회 서비스입니다.
은행명   계좌번호          잔액
HANA    111-451-680      1000
HANA    111-222-333      95679
HANA    111-222-444      1000000
HANA    111-222-555      1000000
HANA    111-681-125      1000
WOORI   444-625-545      4000000
```

사용자 UI

- ✓ [계좌 관리] – [거래 내역 조회]
  - ✓ 자신의 계좌 중 원하는 계좌의 모든 계좌 이체 내역 조회

```
거래 내역 조회 서비스입니다.
계좌 번호를 입력해주세요 : 111-222-333
비밀번호를 입력해주세요 : 1234
거래일자              보내는사람      보내는계좌       받는사람      받는계좌       송금액
2021-06-04 01:58:30   유저2           333-345-678      구본성        111-222-333    50000
2021-06-04 01:58:30   구본성          111-222-333      유저1         222-555-666    10000
2021-06-04 01:58:30   구본성          111-222-333      유저2         333-888-999    50000
2021-06-04 01:58:30   구본성          111-222-333      유저1         222-555-666    20000
2021-06-04 09:43:29   구본성          111-222-333      유저1         222-555-666    4321
```

  - ✓ 불가한 경우
    - ✓ 해당 계좌가 DB에 없거나 본인 소유의 계좌가 아닌 경우

사용자 UI

- ✓ [계좌 관리] – [계좌 수정]
  - ✓ 원하는 계좌의
    계좌 비밀번호, 계좌 별칭 수정
  - ✓ 불가한 경우
    - ✓ 해당 계좌가 DB에 없거나
      본인 소유의 계좌가 아닌 경우
      수정 할 수 없음
    - ✓ 입력한 정보가 틀릴 경우

```
계좌 수정 서비스 입니다.
변경하실 정보를 선택해주세요
1. 계좌비밀번호  2. 계좌별칭 :
```

```
비밀번호를 변경할 계좌번호를 입력해주세요 : 111-222-333
비밀번호를 입력해주세요 : 1234
변경할 비밀번호를 입력해주세요 : 4321
비밀번호를 다시 입력해주세요 : 4321
비밀번호가 변경되었습니다.
```

```
계좌 수정 서비스 입니다.
변경하실 정보를 선택해주세요
1. 계좌비밀번호  2. 계좌별칭 : 2
정보를 변경할 계좌번호를 입력해주세요 : 111-222-333
비밀번호를 입력해주세요 : 4321
변경할 별칭을 입력하세요 : 밥값
별칭이 변경되었습니다.
```

```
정보를 변경할 계좌번호를 입력해주세요 : 11111111
비밀번호를 입력해주세요 : 111
입력하신 정보가 올바르지 않습니다.
```

사용자 UI

- ✓ [계좌 관리] – [계좌 삭제]
  - ✓ 원하는 계좌의 정보를 비활성화 함
  - ✓ 계좌는 삭제되어도 이체 로그는 삭제되지 않음
  - ✓ 불가한 경우
    - ✓ 해당 계좌가 DB에 없거나 본인 소유의 계좌가 아닌 경우
    - ✓ 잔액이 0원이 아닌 경우

```
계좌 삭제 서비스 입니다.
계좌는 잔액이 없는 계좌만 삭제할 수 있습니다.
계좌번호를 입력해주세요 ( - 포함) : 111-222-333
계좌 비밀번호를 입력해주세요 : 4321
해당 계좌가 존재합니다
계좌를 삭제했습니다.
```

```
계좌 삭제 서비스 입니다.
계좌는 잔액이 없는 계좌만 삭제할 수 있습니다.
계좌번호를 입력해주세요 ( - 포함) : 111-222-333
계좌 비밀번호를 입력해주세요 : 4321
해당 계좌가 존재합니다
잔액이 있어 삭제할 수 없습니다.
```

```
계좌 삭제 서비스 입니다.
계좌는 잔액이 없는 계좌만 삭제할 수 있습니다.
계좌번호를 입력해주세요 ( - 포함) : 111111
계좌 비밀번호를 입력해주세요 : 1111
해당 계좌가 존재하지 않습니다.
```

사용자 UI

- ✓ [계좌 이체]
  - ✓ 자신의 계좌 중 원하는 계좌에서 출금하여 상대방의 계좌에 입금함
  - ✓ 불가한 경우
    - ✓ 해당 계좌가 DB에 없거나 본인 소유의 계좌가 아닌 경우 출금을 할 수 없음
    - ✓ 동일한 계좌를 입력할 경우
    - ✓ 이체액이 잔액보다 많은 경우

```
계좌이체 서비스 입니다.
계좌번호를 입력해주세요 ( - 포함) : 111-222-333
계좌 비밀번호를 입력해주세요 : 1234
해당 계좌가 존재합니다
이체할 계좌번호를 입력해주세요 ( - 포함) : 222-555-666
얼마나 이체하시겠습니까? : 4321
이체에 성공했습니다.
```

```
계좌이체 서비스 입니다.
계좌번호를 입력해주세요 ( - 포함) : 111-222-333
계좌 비밀번호를 입력해주세요 : 1233
본인 소유의 계좌가 존재하지 않습니다.
```

```
계좌이체 서비스 입니다.
계좌번호를 입력해주세요 ( - 포함) : 111-222-333
계좌 비밀번호를 입력해주세요 : 1234
해당 계좌가 존재합니다
이체할 계좌번호를 입력해주세요 ( - 포함) : 111-222-333
동일한 계좌에는 이체하실 수 없습니다.
```

```
계좌이체 서비스 입니다.
계좌번호를 입력해주세요 ( - 포함) : 111-222-333
계좌 비밀번호를 입력해주세요 : 1234
해당 계좌가 존재합니다
이체할 계좌번호를 입력해주세요 ( - 포함) : 222-555-666
얼마나 이체하시겠습니까? : 50000000
잔액이 부족합니다
```

미흡한 부분

- ✓ DB 테이블 설계
  - ✓ 테이블을 단순화하다 보니, 테이블 내에 구체적인 정보를 담지 못함
  - ✓ JOIN에 대해 미숙해 성능이 좋지 않은 기능 위주의 쿼리 작성

```
sql.append(" SELECT B.BANK_NAME, U.USER_NAME, A.ACC_NO, A.BALANCE, A.ACC_NICKNAME,"
        + "TO_CHAR(A.ACC_CREATING_DATE, 'yyyy-mm-dd') AS ACC_CREATING_DATE ");
sql.append(" FROM ACCOUNT A, BANK B, USER_INFO U ");
sql.append(" WHERE A.ACC_ISACTIVE = 'ACTIVE' AND A.BANK_CODE = B.BANK_CODE AND A.USER_NO = U.USER_NO ");
sql.append(" AND U.ID = ? AND A.ACC_PW = ? AND A.ACC_NO = ? ");
```

04 프로젝트 보완점

미흡한 부분 및 보완

미흡한 부분

- ✓ 개발 Logic에 대한 이해
  - ✓ Service와 DAO에 대한 구분 기준
  - ✓ 어떤 식으로 Service, DAO 내에서 메소드를 구현해야 하는지…?
- ✓ 메소드 위주의 개발
  - ✓ 과도한 if문 남발로 기능은 구현했을지라도 코드 가독성과 성능에서 차이를 보임

```
        String accNo = scanStr("계좌번호를 입력해주세요 ( - 포함) : ");
        String accPw = scanStr("계좌 비밀번호를 입력해주세요 : ");

        newAcc.setAccNo(accNo);
        newAcc.setAccPw(accPw);

        boolean bool = AccountService.AccountCheckService(newAcc, session);

        if (bool) {
            System.out.println("해당 계좌가 존재합니다");
            String acc = scanStr("이체할 계좌번호를 입력해주세요 ( - 포함) : ");

            boolean bool_2 = AccountService.AccountCheckService(acc);

            if (!newAcc.getAccNo().equals(acc)) { // 동일한 계좌 입력했는지
                if (bool_2) {
                    int transferAmount = scanInt("얼마나 이체하시겠습니까? : ");
                    int balance = AccountService.BalanceCheckService(newAcc);

                    if (balance > 0 && transferAmount <= balance) {
                        int chk = AccountService.transferService(newAcc, acc, transferAmount);
                        if (chk == 1) {
                            System.out.println("이체에 성공했습니다.");
                        } else {
                            System.out.println("이체에 실패했습니다.");
                        }
                    } else { // 잔액이 부족할 경우
                        System.out.println("잔액이 부족합니다");
                    }
                } else {
                    System.out.println("상대 계좌가 존재하지 않습니다.");
                }
            } else {
                System.out.println("동일한 계좌에는 이체하실 수 없습니다.");
            }
        } else {
            System.out.println("본인 소유의 계좌가 존재하지 않습니다.");
        }

    }
}
```

04 프로젝트 보완점

미흡한 부분 및 보완

보완점

✓ DB 테이블 설계는 아무리 강조해도 지나치지 않다

→ 불필요한 데이터는 제외하되, 테이블 간 적절한 관계를 가질 수 있도록 하자

✓ 비즈니스 로직 + 패턴을 가진 개발

→ SERVICE, DAO, VO 등 기능을 확실하게 구분하고

일정한 흐름에 의해 프로그램이 진행될 수 있도록 하자

✓ 메소드 위주의 개발

→ 추후 유지 보수에 원활하도록