DB 2. JDBC DAO 만들기

| Background

- ✓ Transaction 원리

| Goal

- ✓ 여러 개의 SQL 구문을 더 이상 나눌 수 없는 Transaction단위로 처리한다.
- ✓ try~catch~finally 구문을 잘 활용해서 transaction처리구문과 연결한다.

1. eclipse 에 프로젝트를 새로 구성한다. (프로젝트명 : hw02_jdbc)
2. 오라클 드라이버를 BuildPath로 잡는다.
3. 제공된 소스코드를 생성한 프로젝트에 잘 배포합니다.
4. 해당 코드의 테이블과 시퀀스는 아래 부분을 긁어서 생성한 후 배포된 코드를 실행, Transaction 원리를 팀원과 함께 정리하도록 합니다.

```
CREATE TABLE bank(
        id number primary key,
        name varchar2(20),
        bankname varchar2(20),
        balance number(10));


CREATE SEQUENCE bank_seq
    INCREMENT BY 1
    START WITH 1;


INSERT INTO bank (id, name, bankname, balance) VALUES(bank_seq.nextVal, '동은','국민',1000000);
INSERT INTO bank (id, name, bankname, balance) VALUES(bank_seq.nextVal, '연진','국민',2000000);

COMMIT;
```

DB 2. JDBC DAO 만들기

| Background

- ✓ Transaction 원리

| Goal

- ✓ 여러 개의 SQL 구문을 더 이상 나눌 수 없는 Transaction단위로 처리한다.
- ✓ try~catch~finally 구문을 잘 활용해서 transaction처리구문과 연결한다.

SELECT id, name, bankname, balance FROM bank;

동은 – 국민은행 계좌번호 1, 현재 잔액 1백만원
연진 – 국민은행 계좌번호 2, 현재 잔액 2백만원

ID	NAME	BANKNAME	BALANCE
1	동은	국민	1000000
2	연진	국민	2000000

코드분석 주의할 점

1. 서버정보와 SQL Query문 모두 메타 데이타화 시킴
2. 50만원씩 연진은 동은에게 계속 돈을 뜯기는 상황을 연출했다.
   (계속 계좌이체 기능을 호출함)
   이때 연진의 잔고가 이체하려는 금액보다 적을 경우 이체를 멈추도록 한다

3. 여러 SQL문을 하나로 묶는 기능은 setAutocommt(false) 함수를 이용
   try 구문에서는 commit
   catch 구문에서는 rollback
   finally 구문에서는 close와 함께 setAutocommt(true)를 사용해서 하나의 묶음을 마무리 한다.

4. JDBC 미니 프로젝트 비지니스 로직 구현시 해당 Transaction원리를 추가할 경우 가산점 부여~!!

팀원과 해당 코드를 공유하였다면 **Manage** 사이트에 코드를 압축해서 제출기한 내에 올려주세요