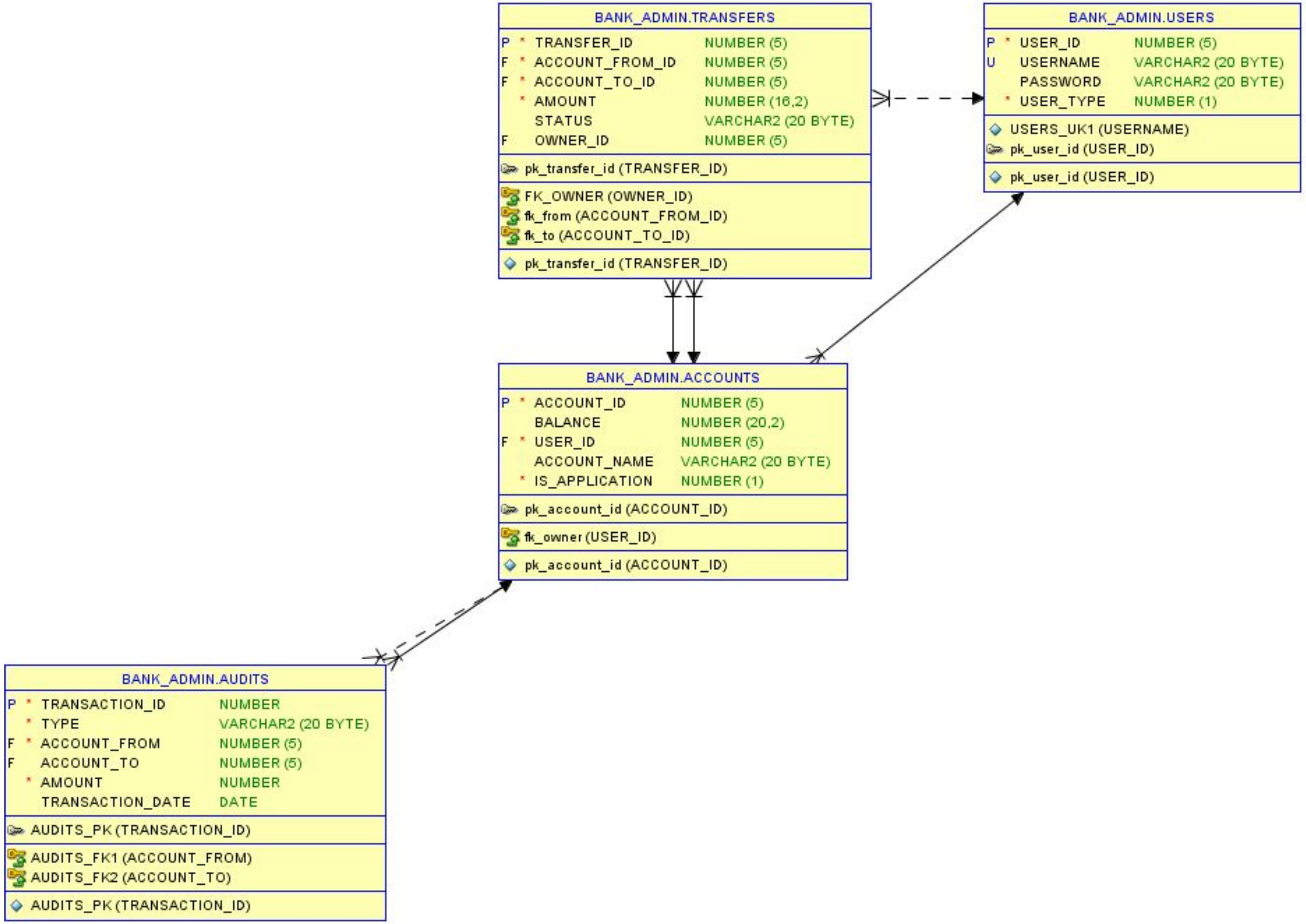
Project 0
Banking System

George Viccaro

ERD



BANK_ADMIN.TRANSFERS

P	*	TRANSFER_ID	NUMBER (5)
F	*	ACCOUNT_FROM_ID	NUMBER (5)
F	*	ACCOUNT_TO_ID	NUMBER (5)
	*	AMOUNT	NUMBER (16,2)
		STATUS	VARCHAR2 (20 BYTE)
F		OWNER_ID	NUMBER (5)

pk_transfer_id (TRANSFER_ID)

FK_OWNER (OWNER_ID)
fk_from (ACCOUNT_FROM_ID)
fk_to (ACCOUNT_TO_ID)

pk_transfer_id (TRANSFER_ID)

BANK_ADMIN.USERS

P	*	USER_ID	NUMBER (5)
U		USERNAME	VARCHAR2 (20 BYTE)
		PASSWORD	VARCHAR2 (20 BYTE)
	*	USER_TYPE	NUMBER (1)

USERS_UK1 (USERNAME)
pk_user_id (USER_ID)

pk_user_id (USER_ID)

BANK_ADMIN.ACCOUNTS

P	*	ACCOUNT_ID	NUMBER (5)
		BALANCE	NUMBER (20,2)
F	*	USER_ID	NUMBER (5)
		ACCOUNT_NAME	VARCHAR2 (20 BYTE)
	*	IS_APPLICATION	NUMBER (1)

pk_account_id (ACCOUNT_ID)

fk_owner (USER_ID)

pk_account_id (ACCOUNT_ID)

BANK_ADMIN.AUDITS

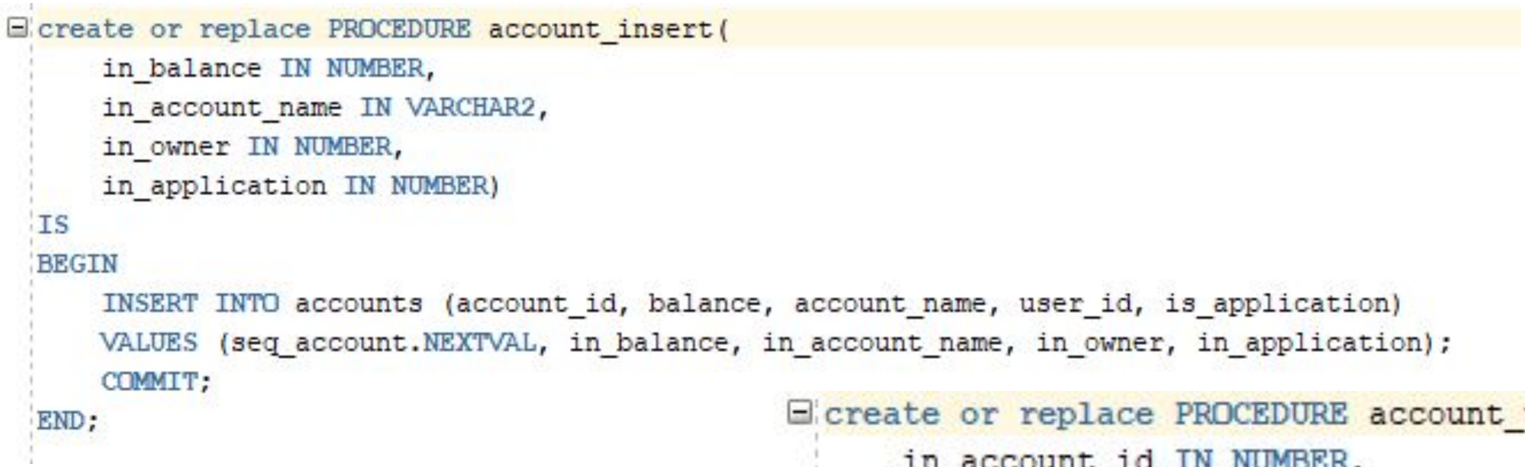
P	*	TRANSACTION_ID	NUMBER
	*	TYPE	VARCHAR2 (20 BYTE)
F	*	ACCOUNT_FROM	NUMBER (5)
F		ACCOUNT_TO	NUMBER (5)
	*	AMOUNT	NUMBER
		TRANSACTION_DATE	DATE

AUDITS_PK (TRANSACTION_ID)

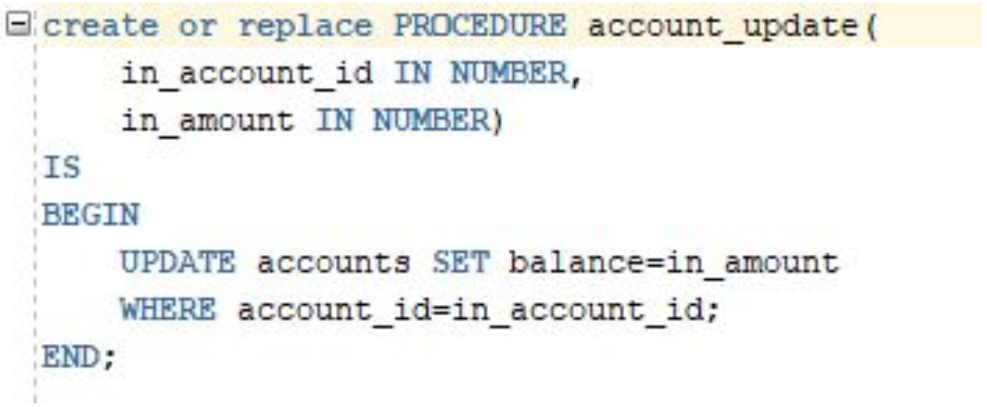
AUDITS_FK1 (ACCOUNT_FROM)
AUDITS_FK2 (ACCOUNT_TO)

AUDITS_PK (TRANSACTION_ID)

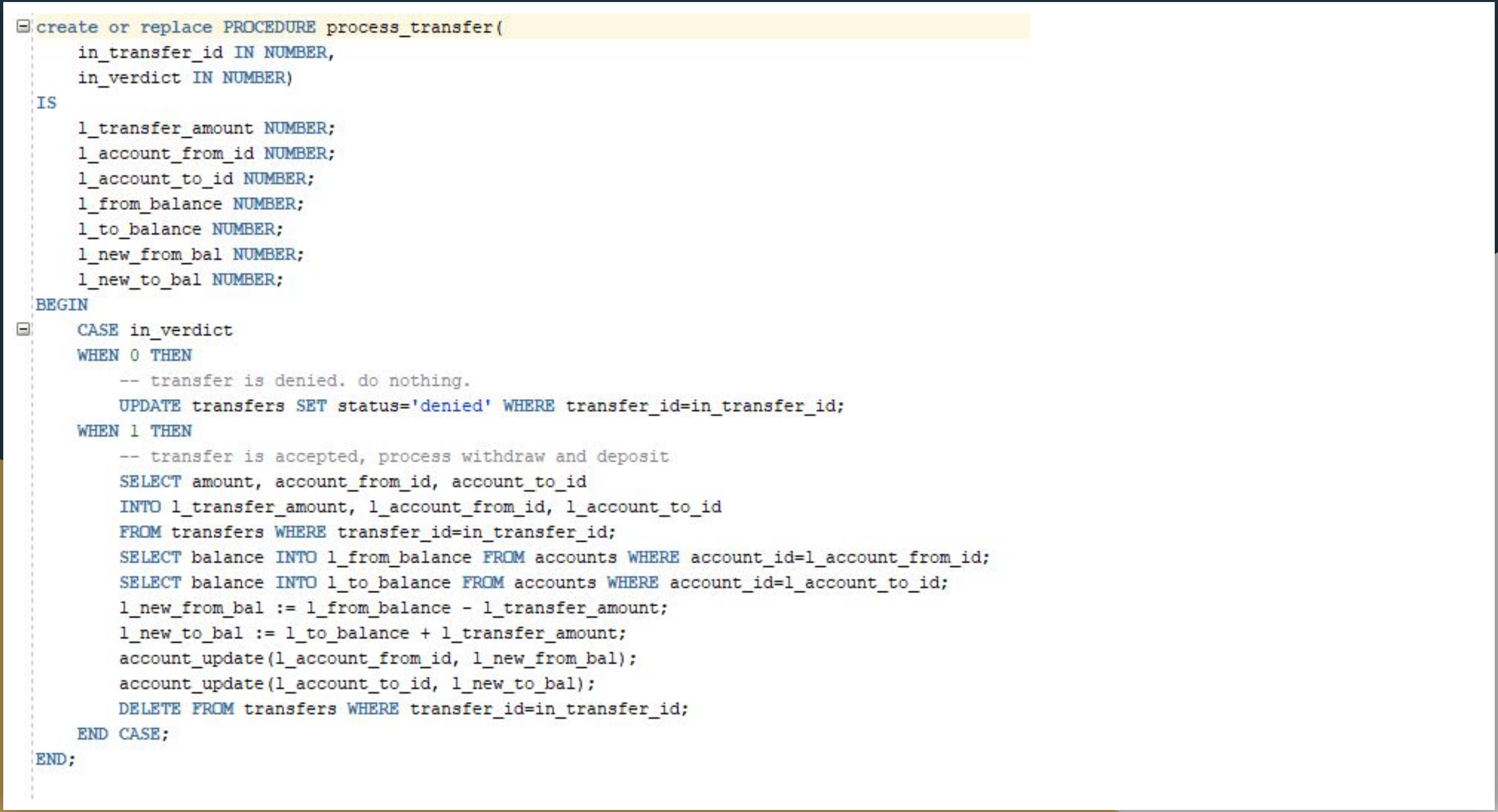
PL/SQL Procedures



```
create or replace PROCEDURE account_insert(
    in_balance IN NUMBER,
    in_account_name IN VARCHAR2,
    in_owner IN NUMBER,
    in_application IN NUMBER)
IS
BEGIN
    INSERT INTO accounts (account_id, balance, account_name, user_id, is_application)
    VALUES (seq_account.NEXTVAL, in_balance, in_account_name, in_owner, in_application);
    COMMIT;
END;
```

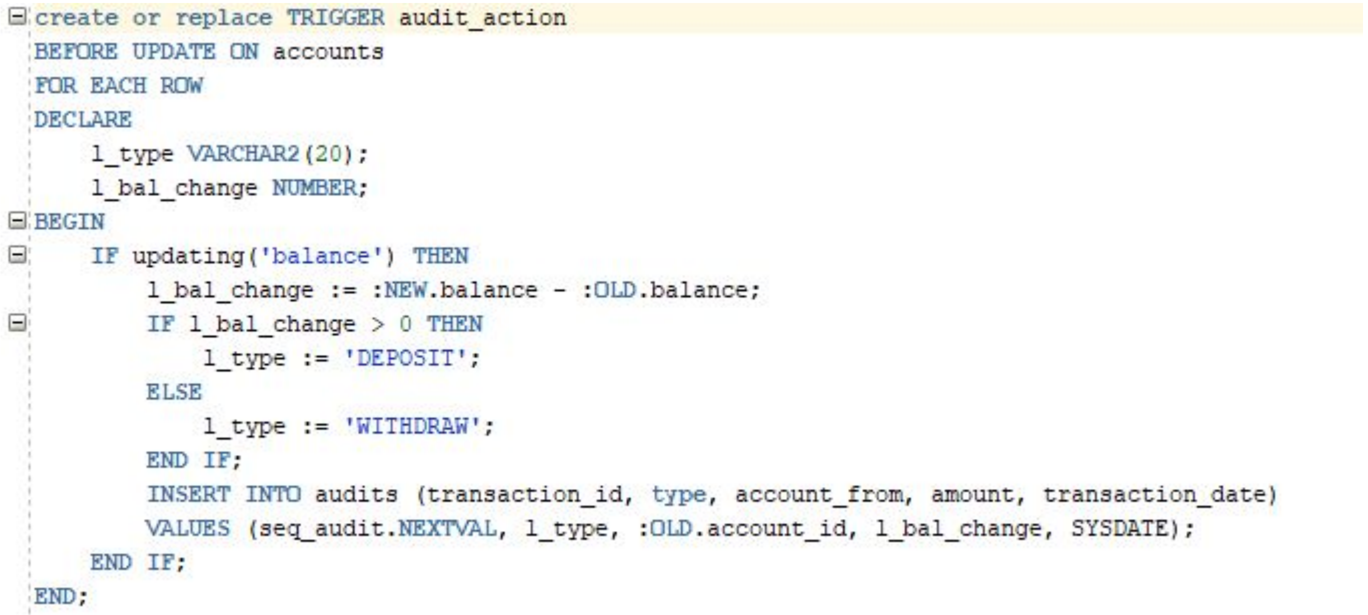


```
create or replace PROCEDURE account_update(
    in_account_id IN NUMBER,
    in_amount IN NUMBER)
IS
BEGIN
    UPDATE accounts SET balance=in_amount
    WHERE account_id=in_account_id;
END;
```

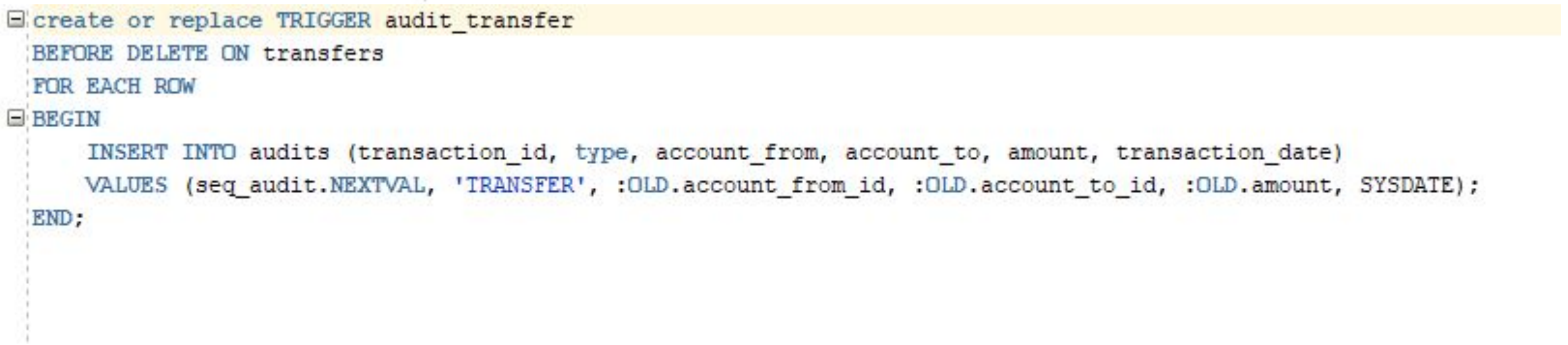


```
create or replace PROCEDURE process_transfer(
    in_transfer_id IN NUMBER,
    in_verdict IN NUMBER)
IS
    l_transfer_amount NUMBER;
    l_account_from_id NUMBER;
    l_account_to_id NUMBER;
    l_from_balance NUMBER;
    l_to_balance NUMBER;
    l_new_from_bal NUMBER;
    l_new_to_bal NUMBER;
BEGIN
    CASE in_verdict
    WHEN 0 THEN
        -- transfer is denied. do nothing.
        UPDATE transfers SET status='denied' WHERE transfer_id=in_transfer_id;
    WHEN 1 THEN
        -- transfer is accepted, process withdraw and deposit
        SELECT amount, account_from_id, account_to_id
        INTO l_transfer_amount, l_account_from_id, l_account_to_id
        FROM transfers WHERE transfer_id=in_transfer_id;
        SELECT balance INTO l_from_balance FROM accounts WHERE account_id=l_account_from_id;
        SELECT balance INTO l_to_balance FROM accounts WHERE account_id=l_account_to_id;
        l_new_from_bal := l_from_balance - l_transfer_amount;
        l_new_to_bal := l_to_balance + l_transfer_amount;
        account_update(l_account_from_id, l_new_from_bal);
        account_update(l_account_to_id, l_new_to_bal);
        DELETE FROM transfers WHERE transfer_id=in_transfer_id;
    END CASE;
END;
```

PL/SQL Triggers



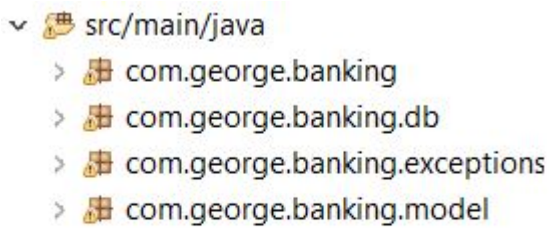
```
create or replace TRIGGER audit_action
BEFORE UPDATE ON accounts
FOR EACH ROW
DECLARE
    l_type VARCHAR2(20);
    l_bal_change NUMBER;
BEGIN
    IF updating('balance') THEN
        l_bal_change := :NEW.balance - :OLD.balance;
        IF l_bal_change > 0 THEN
            l_type := 'DEPOSIT';
        ELSE
            l_type := 'WITHDRAW';
        END IF;
        INSERT INTO audits (transaction_id, type, account_from, amount, transaction_date)
        VALUES (seq_audit.NEXTVAL, l_type, :OLD.account_id, l_bal_change, SYSDATE);
    END IF;
END;
```



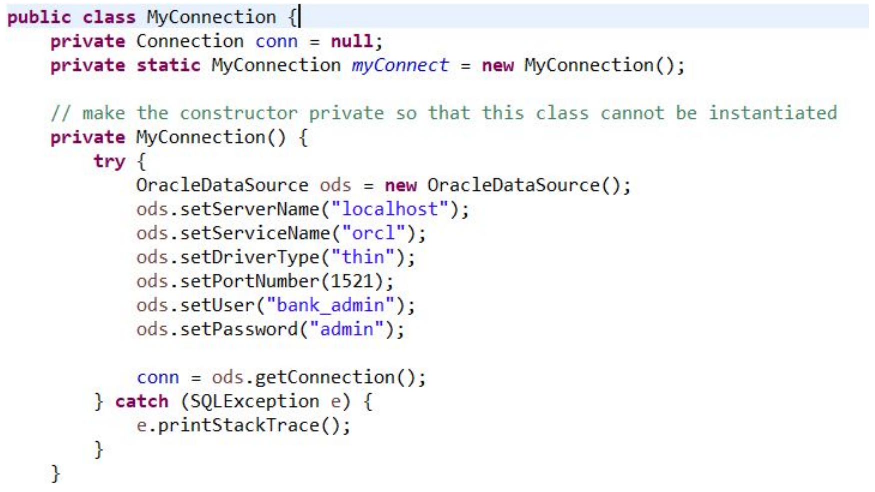
```
create or replace TRIGGER audit_transfer
BEFORE DELETE ON transfers
FOR EACH ROW
BEGIN
    INSERT INTO audits (transaction_id, type, account_from, account_to, amount, transaction_date)
    VALUES (seq_audit.NEXTVAL, 'TRANSFER', :OLD.account_from_id, :OLD.account_to_id, :OLD.amount, SYSDATE);
END;
```

Implementation Challenges

- Learning PL/SQL
- Implementing design patterns appropriately
- Proper file structure



```
src/main/java
    com.george.banking
    com.george.banking.db
    com.george.banking.exceptions
    com.george.banking.model
```



```java
public class MyConnection {
    private Connection conn = null;
    private static MyConnection myConnect = new MyConnection();

    // make the constructor private so that this class cannot be instantiated
    private MyConnection() {
        try {
            OracleDataSource ods = new OracleDataSource();
            ods.setServerName("localhost");
            ods.setServiceName("orcl");
            ods.setDriverType("thin");
            ods.setPortNumber(1521);
            ods.setUser("bank_admin");
            ods.setPassword("admin");

            conn = ods.getConnection();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

Demo!