```
-- Project 2 DDLs: create.clp
--
connect to cs157a;
--
-- drop previous definition first
drop specific function p2.encrypt;
drop specific function p2.decrypt;
drop table p2.account;
drop table p2.customer;
--
-- Without column constraint on Age & Pin, need logic in application to handle
--
create table p2.customer
(
  ID            integer generated always as identity (start with 100, increment by 1),
  Name          varchar(15) not null,
  Gender        char not null check (Gender in ('M','F')),
  Age           integer not null,
  Pin           integer not null check (Pin >= 0),
  primary key (ID)
);
--
-- Without column constraint on Balance, Type, Status, need logic in application to handle
--
create table p2.account
(
  Number        integer generated always as identity (start with 1000, increment by 1),
  ID            integer not null references p2.customer (ID),
  Balance       integer not null,
  Type              char not null,
  Status char not null,
  primary key (Number)
);
--
-- p2.encrypt takes in an integer and output an "encrypted" integer
--
CREATE FUNCTION p2.encrypt ( pin integer )
  RETURNS integer
  SPECIFIC p2.encrypt
  LANGUAGE SQL
  DETERMINISTIC
  NO EXTERNAL ACTION
  READS SQL DATA
  RETURN
    CASE
      WHEN
        pin >= 0
      THEN
        pin * pin + 1000
      ELSE
        -1
    END;
--
-- p2.decrypt takes in an integer and output an "unencrypted" integer
--
CREATE FUNCTION p2.decrypt ( pin integer )
  RETURNS integer
  SPECIFIC p2.decrypt
  LANGUAGE SQL
  DETERMINISTIC
  NO EXTERNAL ACTION
  READS SQL DATA
  RETURN
    CASE
      WHEN
        pin >= 0
      THEN
        SQRT(pin - 1000)
      ELSE
        -1
    END;
--
commit;
terminate;

//*******************************************************************
```

```
--

-- db2 -td"@" -f P2.clp

--

CONNECT TO CS157A@

--

--

DROP PROCEDURE P2.CUST_CRT@

DROP PROCEDURE P2.CUST_LOGIN@

DROP PROCEDURE P2.ACCT_OPN@

DROP PROCEDURE P2.ACCT_CLS@

DROP PROCEDURE P2.ACCT_DEP@

DROP PROCEDURE P2.ACCT_WTH@

DROP PROCEDURE P2.ACCT_TRX@

DROP PROCEDURE P2.ADD_INTEREST@

DROP PROCEDURE P2.IsOwned@

--

--create customer

CREATE PROCEDURE P2.CUST_CRT

(IN p_name CHAR(15), IN p_gender CHAR(1), IN p_age INTEGER, IN p_pin INTEGER, OUT id INTEGER,
OUT sql_code INTEGER, OUT err_msg CHAR(100))

LANGUAGE SQL

  BEGIN

    DECLARE pinCode INTEGER;

    IF p_gender != 'M' AND p_gender != 'F' THEN

      SET sql_code = -100;

      SET err_msg = 'Invalid gender';

    ELSEIF p_age <= 0 THEN

      SET sql_code = -100;

      SET err_msg = 'Invalid age';
```

```
   ELSEIF p_pin < 0 THEN
    SET sql_code = -100;
    SET err_msg = 'Invalid pin!!!';
   ELSE
    SET pinCode = p2.encrypt(p_pin);
    INSERT INTO P2.Customer (Name, Gender, Age, Pin) VALUES (p_name, p_gender, p_age, pinCode);
    SET id = IDENTITY_VAL_LOCAL();
    SET err_msg = '-';
    SET sql_code = 0;
   END IF;
  END@
--
--customer login
CREATE PROCEDURE P2.CUST_LOGIN
(IN p_id INTEGER, IN p_pin INTEGER, OUT valid INTEGER, OUT sql_code INTEGER, OUT err_msg
CHAR(100))
LANGUAGE SQL
   BEGIN
      DECLARE pinCode INTEGER;
      SET pinCode = (SELECT pin FROM p2.customer WHERE ID = p_id);
      IF EXISTS(SELECT * FROM p2.customer WHERE ID=p_id) AND (P2.decrypt(pinCode) = p_pin) THEN
         SET valid = 1;
         SET sql_code = 0;
         SET err_msg = '-';
      ELSE
         SET valid = 0;
         SET sql_code = -100;
         SET err_msg = 'Incorrect id or pin';
      END IF;
    END@
--
--open account
CREATE PROCEDURE P2.ACCT_OPN
(IN p_id INTEGER, IN p_balance INTEGER, IN p_type CHAR(1), OUT number INTEGER, OUT sql_code
INTEGER, OUT err_msg CHAR(100))
LANGUAGE SQL
  BEGIN
    IF NOT EXISTS (SELECT * FROM P2.Customer WHERE ID = p_id) THEN
      SET sql_code = -100;
      SET err_msg = 'Invalid customer id';
    ELSEIF p_balance < 0 THEN
      SET sql_code = -100;
      SET err_msg = 'Invalid balance';
    ELSEIF p_type != 'S' AND p_type != 'C' THEN
      SET sql_code = -100;
      SET err_msg = 'Invalid type';
    ELSE
      INSERT INTO P2.account (ID, Balance, Type, Status) VALUES (p_id, p_balance, p_type, 'A');
      SET number = IDENTITY_VAL_LOCAL();
      SET sql_code = 0;
      SET err_msg = '-';
    END IF;
  END@
--
--close account
CREATE PROCEDURE P2.ACCT_CLS
(IN p_number INTEGER, OUT sql_code INTEGER, OUT err_msg CHAR(100))
LANGUAGE SQL
```

```
    BEGIN
        IF NOT EXISTS (SELECT * FROM p2.account WHERE number = p_number) THEN
           SET sql_code = -100;
           SET err_msg = 'Invalid account number';
        ELSE
           UPDATE p2.account set balance=0, status='I' where number= p_number;
           SET sql_code = 0;
           SET err_msg = '-';
        END IF;
    END@
--
--deposit into account
CREATE PROCEDURE P2.ACCT_DEP
(IN p_number INTEGER, IN p_amt INTEGER, OUT sql_code INTEGER, OUT err_msg CHAR(100))
LANGUAGE SQL
    BEGIN
        IF NOT EXISTS(SELECT * FROM p2.account WHERE number = p_number AND status = 'A') THEN
            SET sql_code = -100;
            SET err_msg = 'Invalid account number';
        ELSEIF p_amt < 0 THEN
            SET sql_code = -100;
            SET err_msg = 'Invalid amount';
        ELSE
            UPDATE p2.account SET balance = balance + p_amt WHERE number = p_number;
            SET sql_code = 0;
            SET err_msg = '-';
        END IF;
    END@
--
```

```
--withdraw from account
CREATE PROCEDURE P2.ACCT_WTH
(IN p_number INTEGER, IN p_amt INTEGER, OUT sql_code INTEGER, OUT err_msg CHAR(100))
LANGUAGE SQL
  BEGIN
    IF NOT EXISTS(SELECT * FROM p2.account WHERE number = p_number AND status = 'A') THEN
      SET sql_code = -100;
      SET err_msg = 'Invalid account number';
    ELSEIF p_amt < 0 THEN
      SET sql_code = -100;
      SET err_msg = 'Invalid amount';
    ELSEIF NOT EXISTS(SELECT * FROM p2.account WHERE number=p_number AND status='A' AND
balance>=p_amt) THEN
      SET sql_code = -100;
      SET err_msg = 'Not enough funds';
    ELSE
      UPDATE p2.account SET balance = balance - p_amt WHERE number = p_number;
      SET sql_code = 0;
      SET err_msg = '-';
    END IF;
  END@
--
--transfer to another account
CREATE PROCEDURE P2.ACCT_TRX
(IN src_acct INTEGER, IN dest_acct INTEGER, IN p_amt INTEGER, OUT sql_code INTEGER, OUT err_msg
CHAR(100))
LANGUAGE SQL
  BEGIN
--      IF NOT EXISTS(SELECT * FROM p2.account WHERE number = src_acct AND status = 'A') THEN
--          SET sql_code = -100;
--          SET err_msg = 'Invalid source account number';
--      ELSEIF NOT EXISTS(SELECT * FROM p2.account WHERE number = dest_acct AND status = 'A') THEN
--          SET sql_code = -100;
--          SET err_msg = 'Invalid destination account number';
--      ELSEIF NOT (SELECT balance FROM p2.account WHERE number = src_acct AND status = 'A') >=
p_amt THEN
--          SET sql_code = -100;
--          SET err_msg = 'Not enough funds';
--      ELSE
--          UPDATE p2.account SET balance=balance - p_amt where number= src_acct;
--          UPDATE p2.account SET balance=balance + p_amt where number= dest_acct;
--          SET sql_code = 0;
--          SET err_msg = '-';
--      END IF;
      CALL P2.ACCT_WTH(src_acct, p_amt, sql_code, err_msg);
      CALL P2.ACCT_DEP(dest_acct, p_amt, sql_code, err_msg);
   END@
--
--interest
CREATE PROCEDURE P2.ADD_INTEREST
(IN savings_rate REAL, IN checking_rate REAL, OUT sql_code INTEGER, OUT err_msg CHAR(100))
LANGUAGE SQL
   BEGIN
       UPDATE p2.account SET balance=balance * (1+savings_rate) WHERE type='S' AND status='A';
       UPDATE p2.account SET balance=balance * (1+checking_rate) WHERE type='C' AND status='A';
       SET sql_code = 0;
       SET err_msg = '-';
   END@
```

```
--

--to check whether the customer owner this account

CREATE PROCEDURE P2.IsOwned

(IN p_id INTEGER, IN p_number INTEGER, OUT is_owned INTEGER, OUT sql_code INTEGER, OUT err_msg
CHAR(100))

LANGUAGE SQL

  BEGIN

    IF EXISTS(SELECT * FROM p2.account WHERE id=p_id AND number=p_number) THEN

      SET is_owned = 1;

      SET sql_code = 0;

      SET err_msg = '-';

    ELSE

      SET is_owned = 0;

      SET sql_code = -100;

      SET err_msg = 'The customer ID does not owned this account.';

    END IF;

  END@

--

--

TERMINATE@

--

--


//*******************************************************************
```

```
import java.io.FileInputStream;
import java.sql.DriverManager;
import java.util.ArrayList;
import java.util.Properties;
import java.util.Scanner;

public class P2 {
    public static void main(String args[]){
            System.out.println(":: PROGRAM START");
            if (args.length < 1) {
                System.out.println("Need database properties filename");
            } else {
                BankingSystem.init(args[0]);
                BankingSystem.testConnection();
                System.out.println();
                ArrayList<String> methodParams = new ArrayList<String>();
                String selection, selection_c, selection_0;
                do{
                    System.out.println("Main Menu--Welcome to the Self 
Services banking System!");
                    System.out.println("1. New Customer\n2. Customer 
Login\n3. Exit");
                    Scanner in = new Scanner(System.in);
                    selection = in.next();
                    switch (selection){
                        case "1":
                            System.out.println("To create a new customer");
                            System.out.println("Please input your information 
follow the order: Name->Gender->Age->Pin:");
                            System.out.println("Name(should be letters or 
space), Gender(M or F), Age(numbers), Pin(numbers)");
                            Scanner input = new Scanner(System.in);
                            String name, gender, age, pin;
                            name = input.nextLine();
                            gender = input.nextLine();
                            age = input.nextLine();
                            pin = input.nextLine();
                            methodParams.add(name);
                            methodParams.add(gender);
                            methodParams.add(age);
                            methodParams.add(pin);
                            executeMethod("#newCustomer", methodParams);
                            methodParams.clear();
                            break;
                        case "2":
                            System.out.println("Please input your Customer ID 
and PIN:");
                            Scanner input_ln = new Scanner(System.in);
                            String id_ln, pin_ln;
                            int isLogin=0;
                            id_ln=input_ln.nextLine();
                            pin_ln=input_ln.nextLine();
                            if(id_ln.equals("0") && 
pin_ln.equals("0")){    //Screen #4
                                do{
                                System.out.println("Administrator Main 
Menu");
```

```
                                System.out.println("1. Account Summary for a
Customer\n" +
                                        "2. Report A::Customer Information
with Total Balance in Decreasing Order\n" +
                                        "3. Report B::Find the Average Total
Balance Between Age Groups\n"+
                                        "4. Exit");
                                Scanner input_0 = new Scanner(System.in);
                                selection_0 = input_0.next();
                                switch(selection_0){
                                    case "1":
                                        System.out.println("Account
summary");
                                        System.out.println("Please input
Customer ID:");
                                        Scanner input_A = new
Scanner(System.in);
                                        String cusID;
                                        cusID = input_A.nextLine();
                                        methodParams.add(cusID);
                                        executeMethod("#accountSummary",
methodParams);
                                        methodParams.clear();
                                        break;
                                    case "2":
                                        System.out.println("Report
A::Customer Information with Total Balance in Decreasing Order");
                                        executeMethod("#reportA",
methodParams);
                                        methodParams.clear();
                                        break;
                                    case "3":
                                        System.out.println("Report B::Find
the Average Total Balance Between Age Groups");
                                        System.out.println("Please input the
Min age, then Max age. They should be numbers:");//String min, String max
                                        Scanner input_3 = new
Scanner(System.in);
                                        String min, max;
                                        min = input_3.nextLine();
                                        max = input_3.nextLine();
                                        methodParams.add(min);
                                        methodParams.add(max);
                                        executeMethod("#reportB",
methodParams);
                                        methodParams.clear();
                                        break;
                                    default:
                                        break;
                                }
                                }while (!selection_0.equals("4"));       //
exit do-while for Administrator Main Menu
                            }
                            methodParams.add(id_ln);
                            methodParams.add(pin_ln);
                            isLogin=BankingSystem.logIn(methodParams.get(0),
methodParams.get(1));
```

```
                            methodParams.clear();
                            if(isLogin==0){
                                System.out.println("Your customer ID and PIN
do not exist or do not match. Try again.");
                                break;
                            }
                        do{
                            System.out.println("Customer Main Menu");
                            System.out.println("1. Open Account\n2. Close
Account\n3. Deposit\n4. Withdraw\n5. Transfer\n6. Account Summary\n7. Exit");
                            Scanner input_c = new Scanner(System.in);
                            selection_c=input_c.next();
                            switch (selection_c){
                                case "1":
                                    System.out.println("To open a new
account");
                                    System.out.println("Please input the
information follow the order: Customer ID->Initial amount->Type:");
                                    System.out.println("Customer ID(Numbers.
If no, to open a new customer), Initial amount(Numbers. >=0), Type(C for
Checking; S for Saving)");
                                    Scanner input_o = new Scanner(System.in);
                                    String id, amount, type;
                                    id = input_o.nextLine();
                                    amount = input_o.nextLine();
                                    type = input_o.nextLine();
                                    methodParams.add(id);
                                    methodParams.add(amount);
                                    methodParams.add(type);
                                    executeMethod("#openAccount",
methodParams);
                                    methodParams.clear();
                                    break;
                                case "2":
                                    System.out.println("To close an
account");
                                    System.out.println("Please input your
Account Number:");
                                    Scanner input_cc = new
Scanner(System.in);
                                    String accNum;
                                    accNum = input_cc.nextLine();
                                   // methodParams.add(id_ln);
                                    methodParams.add(id_ln);
                                    methodParams.add(accNum);
                                    int
isOwned=BankingSystem.IsOwned(methodParams.get(0), methodParams.get(1));
                                    methodParams.clear();
                                    if(isOwned==0){
                                        System.out.println("This account is
not under your Customer ID. You cannot close this account");
                                        break;
                                    }
                                    methodParams.add(accNum);
                                    executeMethod("#closeAccount",
methodParams);
                                    methodParams.clear();
                                    break;
                                case "3":
                                    System.out.println("To deposit money to 
account");
                                    System.out.println("Please input your 
information follow the order: Account number->Deposit Amount:");
                                    System.out.println("Account number(your 
account number), Amount(numbers, >=0)");
                                    Scanner input_d = new Scanner(System.in);
                                    String accNum_d, amount_d;
                                    accNum_d = input_d.nextLine();
                                    amount_d = input_d.nextLine();
                                    methodParams.add(accNum_d);
                                    methodParams.add(amount_d);
                                    executeMethod("#deposit", methodParams);
                                    methodParams.clear();
                                    break;
                                case "4":
                                    System.out.println("To withdraw money 
from account");
                                    System.out.println("Please input your 
information follow the order: Account number->Withdraw Amount:");
                                    System.out.println("Account number(your 
account number), Amount(numbers, >=0)");
                                    Scanner input_w = new Scanner(System.in);
                                    String accNum_w, amount_w;
                                    accNum_w = input_w.nextLine();
                                    amount_w = input_w.nextLine();
                                    //-------------------------------------
                                    methodParams.add(id_ln);
                                    methodParams.add(accNum_w);
                                    int 
isOwned_w=BankingSystem.IsOwned(methodParams.get(0), methodParams.get(1));
                                    methodParams.clear();
                                    if(isOwned_w==0){
                                        System.out.println("This account is 
not under your Customer ID. You cannot withdraw from this account");
                                        break;
                                    }
                                    //-------------------------------------
                                    methodParams.add(accNum_w);
                                    methodParams.add(amount_w);
                                    executeMethod("#withdraw", methodParams);
                                    methodParams.clear();
                                    break;
                                case "5":
                                    System.out.println("To transfer money 
from source account to destination account");
                                    System.out.println("Please input your 
information follow the order: Source account->Destination account->Transfer 
amount:");
                                    Scanner input_t = new Scanner(System.in);
                                    String srcAccNum_t, destAccNum_t, 
amount_t;
                                    srcAccNum_t = input_t.nextLine();
                                    destAccNum_t = input_t.nextLine();
                                    amount_t = input_t.nextLine();
                                    //----------------------------------------
------------
                                    methodParams.add(id_ln);
                                    methodParams.add(srcAccNum_t);
                                    int
isOwned_t=BankingSystem.IsOwned(methodParams.get(0), methodParams.get(1));
                                    methodParams.clear();
                                    if(isOwned_t==0){
                                        System.out.println("This source
account is not under your Customer ID. You cannot transfer from this
account");
                                        break;
                                    }
                                    //----------------------------------------
------------
                                    methodParams.add(srcAccNum_t);
                                    methodParams.add(destAccNum_t);
                                    methodParams.add(amount_t);
                                    executeMethod("#transfer", methodParams);
                                    methodParams.clear();
                                    break;
                                case "6":
                                    System.out.println("Account summary");
                                    System.out.println("Please input your
Customer ID:");

                                    Scanner input_A = new Scanner(System.in);
                                    String cusID;
                                    cusID = input_A.nextLine();
                                    methodParams.add(cusID);
                                    executeMethod("#accountSummary",
methodParams);

                                    methodParams.clear();
                                    break;
                                default:
                                    break;
                            }
                        }while (!selection_c.equals("7"));  // exit do-while
for Customer Main Menu
                        break;                              // Welcome menu,
case "2"-break
                        default:
                            break;                          // Welcome menu,
default-break
                    }
                }while (!selection.equals("3"));            // exit for
Welcome menu
            }

            System.out.println(":: PROGRAM END");
    }



//      /**
//       * Run batch input using properties file.
//       * @param filename properties filename
//       */
//    public static void run(String filename) {
//        String methodName = "";
//        ArrayList<String> methodParams = new ArrayList<String>();
//        try {
//            // Extract batch input from property file.
//            Properties props = new Properties();
//            FileInputStream input = new FileInputStream(filename);
//            props.load(input);
//            String value = props.getProperty("p1.batch.input");
//            // Parse input for method names and parameters.//   not mine
//            String[] tokens = value.split(",");
//            for (int i = 0; i < tokens.length; i++) {
//                if (tokens[i].charAt(0) == '#' && methodName == "") {
//                    methodName = tokens[i];
//                }
//                else if (tokens[i].charAt(0) == '#' && methodName != "") {
//                    for(String s: methodParams){
//                        System.out.println("?^"+s);
//                    }
//                    executeMethod(methodName, methodParams);
//* Execute when meet next '#'.
//                    methodName = tokens[i];
//                    methodParams.clear();
//                }
//                else {
//                    methodParams.add(tokens[i]);
//                }
//            }
//            if (methodName != "") {
//                executeMethod(methodName, methodParams);
//            }
//        } catch (Exception e) {
//            e.printStackTrace();
//        }
//    }

    /**
     * Execute method with name and parameters.
     * @param methodName name of method to execute
     * @param methodParams list of parameters to pass to method
     */
    private static void executeMethod(String methodName, ArrayList<String>
methodParams) {
        switch (methodName) {
            case "#newCustomer":
                BankingSystem.newCustomer(methodParams.get(0),
methodParams.get(1), methodParams.get(2), methodParams.get(3));
                break;
            case "#openAccount":
                BankingSystem.openAccount(methodParams.get(0),
methodParams.get(1), methodParams.get(2));
                break;
            case "#closeAccount":
                BankingSystem.closeAccount(methodParams.get(0));
                break;
            case "#deposit":
                BankingSystem.deposit(methodParams.get(0),
methodParams.get(1));
                break;
            case "#withdraw":
                BankingSystem.withdraw(methodParams.get(0),
methodParams.get(1));
                break;
            case "#transfer":
                BankingSystem.transfer(methodParams.get(0),
methodParams.get(1), methodParams.get(2));
                break;
            case "#accountSummary":
                BankingSystem.accountSummary(methodParams.get(0));
                break;
            case "#reportA":
                BankingSystem.reportA();
                break;
            case "#reportB":
                BankingSystem.reportB(methodParams.get(0),
methodParams.get(1));
                break;
            default:
                System.out.println("Could not find method to execute");
                break;
        }
        System.out.println();
    }
}
```

//**

```
import java.io.FileInputStream;
import java.sql.*;
import java.util.Properties;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * Manage connection to database and perform SQL statements.
 */
public class BankingSystem {
    // Connection properties
    private static String driver;
    private static String url;
    private static String username;
    private static String password;

    // JDBC Objects
    private static Connection con;
    private static Statement stmt;
    private static ResultSet rs;

    /**
     * Initialize database connection given properties file.
     * @param filename name of properties file
     */
    public static void init(String filename) {
        try {
            Properties props = new Properties();                  // Create a
new Properties object
            FileInputStream input = new FileInputStream(filename); // Create
a new FileInputStream object using our filename parameter
            props.load(input);                                    // Load the file
contents into the Properties object
            driver = props.getProperty("jdbc.driver");           // Load the
driver
            url = props.getProperty("jdbc.url");                   // Load the
url
            username = props.getProperty("jdbc.username");        // Load the
username
            password = props.getProperty("jdbc.password");        // Load the
password
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    /**
     * Test database connection.
     */
    public static void testConnection() {
        System.out.println(":: TEST - CONNECTING TO DATABASE");
        try {
            Class.forName(driver);
            con = DriverManager.getConnection(url, username, password);
            con.close();
            System.out.println(":: TEST - SUCCESSFULLY CONNECTED TO
DATABASE");
        } catch (Exception e) {
            System.out.println(":: TEST - FAILED CONNECTED TO DATABASE");
            e.printStackTrace();
        }
    }

    /**
     * Create a new customer.
     * @param name customer name
     * @param gender customer gender
     * @param age customer age
     * @param pin customer pin
     */
    public static void newCustomer(String name, String gender, String age,
String pin)
    {
        System.out.println(":: CREATE NEW CUSTOMER - RUNNING");
        /* insert your code here */
        Pattern pattern = Pattern.compile("[0-9]*");
        Matcher isNum_pin = pattern.matcher(pin);
        if( !isNum_pin.matches()){
            System.out.println("Fail! The input are not numbers");
            return;
        }
        ////////////////////////////////////////////////////////////
        try{
            Class.forName(driver);
            con = DriverManager.getConnection(url, username, password);
//Create the connection

            CallableStatement storeProc = con.prepareCall("CALL
p2.CUST_CRT(?,?,?,?,?,?,?)");
            storeProc.setString(1, name);
            storeProc.setString(2, gender);
            storeProc.setInt(3, Integer.valueOf(age));
            storeProc.setInt(4, Integer.valueOf(pin));
            storeProc.registerOutParameter(5, Types.INTEGER);
            storeProc.registerOutParameter(6, Types.INTEGER);
            storeProc.registerOutParameter(7, Types.CHAR);
            storeProc.executeUpdate();
            int p_id = storeProc.getInt(5);
            int sql_code = storeProc.getInt(6);
            String err_msg = storeProc.getString(7);
            if (sql_code==0)
                System.out.println("Your Customer ID is: " + p_id);
            else
                System.out.println("Fail to create a new customer. The error
message is "+err_msg);
        } catch (Exception e){
            System.out.println("Exception in main()");
            e.printStackTrace();
        }
        ///////////////////////////////////////////////////////////////
        System.out.println(":: CREATE NEW CUSTOMER - SUCCESS");
    }

    /**
     * Open a new account.
     * @param id customer id
     * @param balance initial deposit amount
     * @param type type of account
     */
    public static void openAccount(String id, String balance, String type)
    {
        System.out.println(":: OPEN ACCOUNT - RUNNING");
        /* insert your code here */
        ////////////////////////////////////////////////////////////
        try{
            Class.forName(driver);
            con = DriverManager.getConnection(url, username, password);
//Create the connection

            CallableStatement storeProc = con.prepareCall("CALL
p2.ACCT_OPN(?,?,?,?,?,?)");
            storeProc.setInt(1, Integer.valueOf(id));
            storeProc.setInt(2, Integer.valueOf(balance));
            storeProc.setString(3, type);
            storeProc.registerOutParameter(4, Types.INTEGER);
            storeProc.registerOutParameter(5, Types.INTEGER);
            storeProc.registerOutParameter(6, Types.CHAR);
            storeProc.executeUpdate();
            int accNumber = storeProc.getInt(4);
            int sql_code = storeProc.getInt(5);
            String err_msg = storeProc.getString(6);
            if (sql_code==0)
                System.out.println("Your account ID is: " + accNumber);
            else
                System.out.println("Fail to create a new account. The error
message is "+err_msg);
        } catch (Exception e){
            System.out.println("Exception in main()");
            e.printStackTrace();
        }
        ///////////////////////////////////////////////////////////////
        System.out.println(":: OPEN ACCOUNT - SUCCESS");
    }

    /**
     * Close an account.
     * @param accNum account number
     */
    public static void closeAccount(String accNum)
    {
        System.out.println(":: CLOSE ACCOUNT - RUNNING");
        /* insert your code here */
        ////////////////////////////////////////////////////////////
        try{
            Class.forName(driver);
            con = DriverManager.getConnection(url, username, password);
//Create the connection
//            PreparedStatement stat = con.prepareStatement("update
p1.account set balance=0, status='I' where number= ?");
//            stat.setInt(1, Integer.valueOf(accNum));
//            stat.executeUpdate();
            CallableStatement storeProc = con.prepareCall("CALL
P2.ACCT_CLS(?,?,?)");
            storeProc.setInt(1, Integer.valueOf(accNum));
            storeProc.registerOutParameter(2, Types.INTEGER);
            storeProc.registerOutParameter(3, Types.CHAR);
            storeProc.executeUpdate();
            int sql_code = storeProc.getInt(2);
            String err_msg = storeProc.getString(3);
            if (sql_code==0)
                System.out.println("Close account successfully.");
            else
                System.out.println("Fail to create a new account. The error
message is "+err_msg);

        } catch (Exception e){
            System.out.println("Exception in main()");
            e.printStackTrace();
        }
        ////////////////////////////////////////////////////////////////
        System.out.println(":: CLOSE ACCOUNT - SUCCESS");
    }

    /**
     * Deposit into an account.
     * @param accNum account number
     * @param amount deposit amount
     */
    public static void deposit(String accNum, String amount)
    {
        System.out.println(":: DEPOSIT - RUNNING");
        /* insert your code here */
        Pattern pattern = Pattern.compile("[0-9]*");
        Matcher isNum_amount = pattern.matcher(amount);
        Matcher isNum_accNum = pattern.matcher(accNum);
        if( !(isNum_amount.matches() && isNum_accNum.matches() )){
            System.out.println("Fail! The input are not numbers");
            return;
        }
        /////////////////////////////////////////////////////////////
        try{
            Class.forName(driver);
            con = DriverManager.getConnection(url, username, password);
//Create the connection
//              PreparedStatement stat = con.prepareStatement("update
p1.account set balance=balance + ? where number= ?");
//              stat.setInt(1, Integer.valueOf(amount));
//              stat.setInt(2, Integer.valueOf(accNum));
//              stat.executeUpdate();
            CallableStatement storeProc = con.prepareCall("CALL
P2.ACCT_DEP(?,?,?,?)");
            storeProc.setInt(1, Integer.valueOf(accNum));
            storeProc.setInt(2, Integer.valueOf(amount));
            storeProc.registerOutParameter(3, Types.INTEGER);
            storeProc.registerOutParameter(4, Types.CHAR);
            storeProc.executeUpdate();
            int sql_code = storeProc.getInt(3);
            String err_msg = storeProc.getString(4);
            if (sql_code==0)
```

```
                System.out.println("Deposit successfully.");
            else
                System.out.println("Fail to deposit. The error message is
"+err_msg);
        } catch (Exception e){
            System.out.println("Exception in main()");
            e.printStackTrace();
        }
        ////////////////////////////////////////////////////////////////
        System.out.println(":: OPEN ACCOUNT - SUCCESS");
    }

    /**
     * Withdraw from an account.
     * @param accNum account number
     * @param amount withdraw amount
     */
    public static void withdraw(String accNum, String amount)
    {
        System.out.println(":: WITHDRAW - RUNNING");
        /* insert your code here */
        Pattern pattern = Pattern.compile("[0-9]*");
        Matcher isNum_amount = pattern.matcher(amount);
        Matcher isNum_accNum = pattern.matcher(accNum);
        if( !(isNum_amount.matches() && isNum_accNum.matches() )){
            System.out.println("Fail! The input are not numbers");
            return;
        }
        /////////////////////////////////////////////////////////////
        try{
            Class.forName(driver);
            con = DriverManager.getConnection(url, username, password);
//Create the connection
//              PreparedStatement stat = con.prepareStatement("update
p1.account set balance=balance - ? where number= ? AND balance>=?");
//              stat.setInt(1, Integer.valueOf(amount));
//              stat.setInt(2, Integer.valueOf(accNum));
//              stat.setInt(3, Integer.valueOf(amount));
//              stat.executeUpdate();
            CallableStatement storeProc = con.prepareCall("CALL
P2.ACCT_WTH(?,?,?,?)");
            storeProc.setInt(1, Integer.valueOf(accNum));
            storeProc.setInt(2, Integer.valueOf(amount));
            storeProc.registerOutParameter(3, Types.INTEGER);
            storeProc.registerOutParameter(4, Types.CHAR);
            storeProc.executeUpdate();
            int sql_code = storeProc.getInt(3);
            String err_msg = storeProc.getString(4);
            if (sql_code==0)
                System.out.println("Withdraw successfully.");
            else
                System.out.println("Fail to withdraw. The error message is
"+err_msg);

        } catch (Exception e){
            //System.out.println("Fail to withdraw! Please check whether the
amount is bigger than the balance.");
```

```
            e.printStackTrace();
        }
        ////////////////////////////////////////////////////////////////
        System.out.println(":: WITHDRAW - SUCCESS");
    }

    /**
     * Transfer amount from source account to destination account.
     * @param srcAccNum source account number
     * @param destAccNum destination account number
     * @param amount transfer amount
     */
    public static void transfer(String srcAccNum, String destAccNum, String
amount)
    {
        System.out.println(":: TRANSFER - RUNNING");
        /* insert your code here */
        /////////////////////////////////////////////////////////////
        try{
            Class.forName(driver);
            con = DriverManager.getConnection(url, username, password);
//Create the connection
            CallableStatement storeProc = con.prepareCall("CALL
P2.ACCT_TRX(?,?,?,?,?)");
            storeProc.setInt(1, Integer.valueOf(srcAccNum));
            storeProc.setInt(2, Integer.valueOf(destAccNum));
            storeProc.setInt(3, Integer.valueOf(amount));
            storeProc.registerOutParameter(4, Types.INTEGER);
            storeProc.registerOutParameter(5, Types.CHAR);
            storeProc.executeUpdate();
            int sql_code = storeProc.getInt(4);
            String err_msg = storeProc.getString(5);
            if (sql_code==0)
                System.out.println("Transfer successfully.");
            else
                System.out.println("Fail to transfer. The error message is
"+err_msg);
        } catch (Exception e){
            //System.out.println("Fail to transfer! Please check whether the
amount is bigger than the balance.");
            e.printStackTrace();
        }
        System.out.println(":: TRANSFER - SUCCESS");
    }

    /**
     * Display account summary.
     * @param cusID customer ID
     */
    public static void accountSummary(String cusID)
    {
        System.out.println(":: ACCOUNT SUMMARY - RUNNING");
        try{
            Class.forName(driver);
            con = DriverManager.getConnection(url, username, password);
//Create the connection
            stmt = con.createStatement();
            String query0 = "select number, balance from p2.account where
id=" + cusID+" AND status='A'";
            rs = stmt.executeQuery(query0);
            System.out.println("NUMBER" + ",\t" + "BALANCE");
            System.out.println("-----------------------------");
            while(rs.next())
{                                                                    //Loop
through result set and retrieve contents of each row
                int number = rs.getInt(1);
                int balance = rs.getInt(2);

                System.out.println(number + ",\t" + balance);        //Print
out each row's values to the screen
            }
            rs.close();
            System.out.println("-----------------------------");

            String query = "select Name, p2.customer.ID, Sum(balance) as
Total from p2.account, p2.customer where (p2.account.id=p2.customer.id AND
p2.customer.id=" + cusID + ") group by name, p2.customer.ID Order by Total";
            rs = stmt.executeQuery(query);
            System.out.println("Name" + ",\t" + "ID" + ",\t" + "Total");
            while(rs.next())
{                                                                    //Loop
through result set and retrieve contents of each row

                String Name = rs.getString(1);
                int ID = rs.getInt(2);
                int Total = rs.getInt(3);

                System.out.println(Name + ",  " + ID + ",\t" + Total);
//Print out each row's values to the screen
            }
            rs.close();

        } catch (Exception e){
            System.out.println("Exception in main()");
            e.printStackTrace();
        }
        ///////////////////////////////////////////////////////////////
        System.out.println(":: ACCOUNT SUMMARY - SUCCESS");
    }

    /**
     * Display Report A - Customer Information with Total Balance in
Decreasing Order.
     */
    public static void reportA()
    {
        System.out.println(":: REPORT A - RUNNING");
        /* insert your code here */
////////////////////////////////////////////////////////////
        try{
            Class.forName(driver);
            con = DriverManager.getConnection(url, username, password);
//Create the connection
            stmt = con.createStatement();
            String query = "select p2.customer.ID,name,gender,age, 
Sum(balance) as Total from p2.account, p2.customer " +
                    "where p2.account.id=p2.customer.id group by 
p2.customer.ID, name, gender, age Order by Total DESC";
            rs = stmt.executeQuery(query);
            System.out.println("ID" + "\t" + "NAME" + "\t" + "GENDER" + "\t" 
+ "AGE" + "\t" + "TOTAL" );
            System.out.println("--------------------------------------------
------------------------");
            while(rs.next())
{                                                                           //Loop 
through result set and retrieve contents of each row
                int id = rs.getInt(1);
                String name = rs.getString(2);
                String gender = rs.getString(3);
                int age = rs.getInt(4);
                int total = rs.getInt(5);

                System.out.println(id + "\t" + name + "\t" + gender + "\t" + 
age + "\t" + total );        //Print out each row's values to the screen
            }
            rs.close();

        } catch (Exception e){
            System.out.println("Exception in main()");
            e.printStackTrace();
        }
        ///////////////////////////////////////////////////////////////
        System.out.println(":: REPORT A - SUCCESS");
    }

    /**
     * Display Report B - Customer Information with Total Balance in 
Decreasing Order.
     * @param min minimum age
     * @param max maximum age
     */
    public static void reportB(String min, String max)
    {
        System.out.println(":: REPORT B - RUNNING");
        /* insert your code here */
        ////////////////////////////////////////////////////////////
        try{
            Class.forName(driver);
            con = DriverManager.getConnection(url, username, password);
//Create the connection
            stmt = con.createStatement();

            String query = "select AVG (Total) from (select 
p2.customer.ID,name,gender,age, Sum(balance) as Total from p2.account, 
p2.customer " +
                    "where p2.account.id=p2.customer.id group by 
p2.customer.ID, name, gender, age) where (age>="+min+" AND age<="+max+")";
            rs = stmt.executeQuery(query);

            System.out.println( "AVERAGE" );
            System.out.println("---------------------------------------------
");
            while(rs.next())
{                                                                                   //Loop
through result set and retrieve contents of each row
            int average = rs.getInt(1);
            System.out.println("\t" + average );        //Print out each
row's values to the screen
            }
            rs.close();

        } catch (Exception e){
            System.out.println("Exception in main()");
            e.printStackTrace();
        }
        ////////////////////////////////////////////////////////////////
        System.out.println(":: REPORT B - SUCCESS");
    }



    /**
     * Customer log in
     * @param cusID customer ID
     * @param pin maximum age
     */
    public static int logIn(String cusID, String pin)
    {
////////////////////////////////////////////////////////////////
        int valid=0;
        try{
            Class.forName(driver);
            con = DriverManager.getConnection(url, username, password);
//Create the connection
//            stmt = con.createStatement();
//
//            String query = "select count(*) from p1.customer where
id="+cusID+" AND pin = "+pin;
//            rs = stmt.executeQuery(query);
//            while(rs.next())
{                                                                                   //Loop
through result set and retrieve contents of each row
//                count = rs.getInt(1);
//            }
//            rs.close();

            CallableStatement storeProc = con.prepareCall("CALL
P2.CUST_LOGIN(?,?,?,?,?)");
            storeProc.setInt(1, Integer.valueOf(cusID));
            storeProc.setInt(2, Integer.valueOf(pin));
            storeProc.registerOutParameter(3, Types.INTEGER);
            storeProc.registerOutParameter(4, Types.INTEGER);
            storeProc.registerOutParameter(5, Types.CHAR);
            storeProc.executeUpdate();
            valid = storeProc.getInt(3);
            int sql_code = storeProc.getInt(4);
            String err_msg = storeProc.getString(5);
            if(sql_code!=0)
                System.out.println("Fail log in. The error message is 
"+err_msg);
        } catch (Exception e){
            System.out.println("Exception in main()");
            e.printStackTrace();
        }
        return valid;
        ////////////////////////////////////////////////////////////////
    }

    /**
     * Check whether the Customer owns the account
     * @param cusID customer ID
     * @param accNum account number
     */
    public static int IsOwned(String cusID, String accNum)
    {
////////////////////////////////////////////////////////////////
        int is_owned=0;
        try{
//            Class.forName(driver);
//            con = DriverManager.getConnection(url, username, password);
//Create the connection
//            stmt = con.createStatement();
//            String query = "select count(*) from p1.account where 
id="+cusID+" AND number = "+accNum;
//            rs = stmt.executeQuery(query);
//            while(rs.next())
{                                                                       //Loop 
through result set and retrieve contents of each row
//                count = rs.getInt(1);
//            }
//            rs.close();
            CallableStatement storeProc = con.prepareCall("CALL 
P2.IsOwned(?,?,?,?,?)");
            storeProc.setInt(1, Integer.valueOf(cusID));
            storeProc.setInt(2, Integer.valueOf(accNum));
            storeProc.registerOutParameter(3, Types.INTEGER);
            storeProc.registerOutParameter(4, Types.INTEGER);
            storeProc.registerOutParameter(5, Types.CHAR);
            storeProc.executeUpdate();
            is_owned = storeProc.getInt(3);
            int sql_code = storeProc.getInt(4);
            String err_msg = storeProc.getString(5);
            if(sql_code==0)
                is_owned = 1;
            else{
                is_owned=0;
                System.out.println("Fail. The error message is "+err_msg);
            }

        } catch (Exception e){
            System.out.println("Exception in main()");
            e.printStackTrace();
        }
        return is_owned;
```

```
            ////////////////////////////////////////////////////////////////////////
        }
    }
```