**CS157A: Spring 2021**

**Project #1: Self Services Banking System (130 Points)**

**Due Date: Apr 29, 2021**

Description:

You are responsible to write a database application which acts as a simple banking system. This application must be able to do basic banking functions according to the specifications given below. This project must be implemented using DB2, Java, & JDBC.

Project Specification:

Section A: Schema Definition

P1.Customer (*ID*, Name, Gender, Age, Pin)

P1.Account (*Number*, ID, Balance, Type, Status)

-- All the attributes cannot be NULL.

-- Underlined attributes is denoted as the primary key of that relation.

-- Italicized attributes is system generated.

-- Attribute types and ranges:

- ID: integer (system generate starting from 100)

- Name: varchar(15)

- Gender: char (must be M or F only !!!)

- Age: integer (>= 0)

- Pin: integer (>= 0)

- Number: integer (system generate starting from 1000)

- Balance: integer (>= 0)

- Type: char (C for Checking, S for Saving)

- Status: char (A for Active, I for Inactive)

Section B: Data Administration & Manipulation

Screen # 1 (Title – Welcome to the Self Services Banking System! – Main Menu)

1. New Customer
2. Customer Login
3. Exit

For #1, prompt for Name, Gender, Age, and Pin. System will return a customer ID if successful.

For #2, prompt for customer ID and pin to authenticate the customer. If user enters 0 for both customer ID & pin, then you will go straight to Screen #4.

Screen # 3 (Title – Customer Main Menu)

1. Open Account
2. Close Account
3. Deposit
4. Withdraw
5. Transfer
6. Account Summary
7. Exit

For #1, prompt for customer ID, account type, and balance (Initial deposit). System will return an account number if successful.

For #2, prompt for account #, change the status attribute to ‘I’ and empty the balance for that account.

For #3, prompt for account # and deposit amount.

For #4, prompt for account # and withdraw amount.

For #5, prompt for the source and destination account #s and transfer amount.

For #6, display each account # and its balance for same customer and the total balance of all accounts.

For #7, go back to the previous menu.

Screen # 4 (Title – Administrator Main Menu)

1. Account Summary for a Customer
2. Report A :: Customer Information with Total Balance in Decreasing Order
3. Report B :: Find the Average Total Balance Between Age Groups
4. Exit

Note: The only way you can get to Screen #4 is by entering 0 as the ID and 0 as the pin in the customer login screen.

For #1, same function as #6 above except that you would need to input the customer ID explicitly.

For #2, you would display the customer ID, Name, Age, Gender, and total balance in decreasing order.

For #3, you prompt for a min & max age to compute and display Average Balance.

For #4, go back to the previous menu.

Section C: User Interfaces

1. Command line interface described in Section B.
2. If you have extra time, you can add GUI panels on top of the command line interfaces.

Section D: Additional Notes:

1. The special administrator ID and Pin (0,0) can be hardcoded in your program as a special ID/PIN.
2. Customer IDs are system generated and initiated by the “New Customer“ operation.
3. The customer and administrator main menus are the default top level menu after an operation.
4. You can open an account for someone else but you cannot close someone else’s account.
5. You can deposit into other people’s accounts but you can’t withdraw from them.
6. You can transfer money from your account to someone else but not the reverse.
7. All the range checking need to be handled both in DDLs and your application.
8. Customer account summary should not include accounts in the closed state.
9. For all the administrator reports, closed accounts will not be part of the reports.
10. For database connections information, use a properties file.
11. Must handle error condition gracefully (e.g. should not crash and exit because of any exceptions).
12. You will be given Java code framework and you need to put your logic inside.

1. Download all these file (along with the new driver file above db2jcc4.jar into a single directory).

2. Compile all the \*.java files  (e.g. javac \*.java)

3. Copy p1\_create.sql to your container directory ( e.g. docker cp ./p1\_create.sql 92a7b62a44b1:./database/config/db2inst1/. )

4. Execute p1\_create.sql inside the Db2 container (e.g. db2 -tvf p1\_create.sql )

5. Execute the test framework by java -cp ":./db2jcc4.jar" ProgramLauncher ./db.properties >test1.out

This db.properties file contain the usual 4 lines + a set of test data for  9 methods defined in BankingSystem.java:

public static void newCustomer(String name, String gender, String age, String pin)

public static void openAccount(String id, String type, String amount)

public static void closeAccount(String accNum)

public static void deposit(String accNum, String amount)

public static void withdraw(String accNum, String amount)

public static void transfer(String srcAccNum, String destAccNum, String amount)

public static void accountSummary(String cusID)

public static void reportA()

public static void reportB(String min, String max)

The only method that is not there is the login method which you will define only if you have an UI.  After you execute the command above.  test1.out contains the output of the run.  It should match the test\_empty.out above.  Once you implemented the 9 methods, your output should looks like test\_full.out.

6.  Due to time constraint, I move out the due date and late submission date by 2 days.  The command line UI is now OPTIONAL and it will count as extra credit (+20 points).  Command line main class should be p1.java.  If you plan to submit the UI for extra credit, make sure you have Clear instructions in your readme file so the TA can follow.

7.  One important note - you cannot use/call any of the stored procedures from P2.  That means all the logic will be Java/JDBC code inside each method.

8.  Without extra credit, you need to submit BankingSystem.java and a readme file.  If you have other files, make sure you described them in your readme.

9. Grading will be based on the 9 methods inside the BankingSystem.java  (login will only be use by the UI).  The batch testing will use mainly "good" data with a few error testcases.  However if you submit an UI, there will be lots more error testing (e.g. input validation, ... etc)

10.  In the original p1.doc, it said to create a view in Notes: #12.  DON'T do that, just query the tables directly.

TIPS:  You should always run p1\_create.sql each time before you run the Java program, otherwise your output may not sync up correctly due to the auto-generated IDs.

Note:   Even though I came up with the idea of this batch testing framework.  You are welcome to use it in future classes.  This tool is a good way to test each individual Java class method without have a front-end or any user interface.