COMP4003 Assignment #4

JDBC Programming

Due: November 16

Instruction

1. You should do the assignments independently. Copying is not allowed.
2. Submit your assignment as a single word/PDF document on culearn.
3. Put the programs in Parts 1, 2, 4 into the file to be submitted.
4. Put the screenshots of the execution in Parts 3 and 4 into the file to be submitted.
5. The total mark is 106.

Part 1 Database Creation (10)

Write one JDBC program to create a database for a bank application with 3 relations: branch, customer, account with proper primary keys, foreign key and other constraints.

- A branch has a 3 digit branch number starting from '000', and an address.
- A customer has a 5 digit customer number starting from '00000', a name that is unique for simplicity, and a status between 0 and 3, which is based on the total balance the customer has in the bank:
  - 0 if balance is 0,
  - 1 if balance is below 1000,
  - 2 if balance is below 2000, and
  - 3 if otherwise.
- An account has a 7 digit account number that includes 3 digit branch number and 4 digit local account number starting from '0000000', a customer number, and a balance which must never be negative.

Put the program into the file to be submitted.

Part 2 User interface programming (22)

Write one JDBC program that allows bank employees to perform the following basic tasks. The program should provide a simple menu to allow the user to select the options and should perform necessary checks to prevent errors. For each task, you should write a Java method with the name given. You need to properly use transactions for these operations to gurantee the consistency of the database.

1. open_branch: When the user gives an address for the new branch, it first checks if there is a branch number unused (closed branch). If so, just uses the first unused number; otherwise, generates a branch number that is the highest branch number plus one. Then adds the tuple in the corresponding table and tells the user the new branch number.
2. close_branch: When the user gives an address or a branch number, it deletes the branch row. If the branch does not exists, it should let the user know.
3. setup_account: When given a customer name, a branch address or branch number, and an initial amount if any, it first checks if the customer and branch exist. If so, it obtains an account number whose first 3 digits are branch number and whose remaining 4 digits are local account number, and then inserts the row into the account table. If there is an unused account number, it should use it. If not, it generates a number that is the highest local account number plus one.
4. setup_customer: When given a customer name, a branch address or branch number, it first generates a new customer number and a status and then inserts the customer row. Then it invokes setup_account method to set up an account. If the customer has already been setup, generate error message.
5. close_account: Given a customer name, and a branch address or branch number, it deletes the row in the account table if the balance is 0. If this is the customer's last account, delete the customer as well. If this is the last account in the branch, close the branch.
6. withdraw: Given a customer name, account number, and the amount to be withdraw, it withdraws the amount from the customer's account if it has enough money and updates the customer status accordingly. Assume that customer name is unique.

7. deposit: Given a customer name, account number, and the amount to be deposited, it deposits the amount to the customer's account and updates the customer status accordingly. So just transfer from one account to another owned by the same customer.
8. transfer: Given a customer name, two account numbers, and the amount, it transfers the amount from the first account to the second account if the first account has enough money.
9. show_branch: Given a branch name or number, it displays the accounts in the branch and the balance of each account and the total balance.
10. show_all_branches: It invokes show_branch method for all branches in the system.
11. show_customer: Given a customer name, it displays customer number, status, and account numbers and balance in each account and the total balance.

Part 3 Database Populating and Testing (26)

1. open a branch in London
2. open a branch in Paris
3. open a branch in Toronto
4. open a branch in New York
5. show all branches
6. setup a customer called John in London branch
7. setup an account for John in Toronto branch with initial deposit of $500
8. setup an account for John in Paris branch with initial deposit of $1,000
9. setup an account for John in New York branch with initial deposit of $1,000
10. show customer John
11. setup a customer called Joan in Toronto branch with initial deposit of $1,000
12. setup a customer called Mary in Paris branch
13. deposit $1,000 to Mary's Paris branch
14. setup a customer called Mary in New York branch with initial deposit of $1,000
15. setup an account for Mary in New York branch with initial deposit of $1,000
16. show customer Mary
17. setup a customer called Sean in Toronto branch with initial deposit of $1,000
18. setup a customer called Tony in Ottawa branch
19. setup a customer called Tony in Toronto branch
20. transfer $1000 from John's Toronto account to his Paris account
21. transfer $1000 from John’s New York branch to his Paris account
22. show the balances of all John’s account.
23. close every John's account that has 0 balance.
24. withdraw $1000 from Sean's Toronto account
25. show all accounts in Toronto branch
26. show all branches of the bank.

Part 4 JDBC Object programming (22)

Write one JDBC program using Strongly Typed Objects that allows users to perform the same basic tasks on the database as in Part 3.

Part 5 Database Populating and Testing (26)

Delete the tuples in tables branch, customer, account and redo Part 3 using the methods in Part 4.