Assignment 2: Take-Home Assignment

Instructions

You will be creating an account class, a bank class and a runner class

Account Class

The account class will need to store

- An account number of alphanumerical characters.
- An account balance representing a quantity of cash.
- An account name.

The above information must not be directly accessible; methods must be used to access and manipulate them.

The following limitations must be imposed via methods:

- The account number can only contain alphanumerical characters
- The account name can only contain alphabetical characters, as well as spaces and hyphens
- The account balance can only contain non-negative values.

Do not allow changes and output an appropriate error message if the requirements are NOT met. If the requirements are met, output no message.

Bank Class

After creating the above class, you are to create a Bank class. The Bank class will need to store

- It's Bank name
- It's Branch location (from a list of pre-determined options)
- A collection of all the accounts created (from the Account class) at this Bank's branch. There is no limit on how many accounts can be created.
- A method to get a specific account by its account number
- A method to parse a command from a string argument
- Multiple methods to execute various functions stated below

Runner Class

This runner class will interact with the user. It will complete the following;

- Ask the user to input a Bank name and branch location
- Output a welcome message in the format: Welcome to BRANCH_LOCATION of BANK_NAME
- Ask the user what they would like to do next of the following options:
  - Add Account
  - View Accounts
  - Account Details
  - Modify Account
  - Delete Account
  - Summary
  - Help

Below is more information on what the process of the actions should resemble.

Add Account

Ask the user for all the requirement account class information. Only add the account if all of the account information is valid.

If all the account information is valid, create the account object and add this account to the Bank's collection of accounts.

View Accounts

Output all account information for each of the Banks' account.

Account Details

The user should be able to type "view ACCOUNT_NUMBER details" and the user will get a summary of the specific account details.

Modify Account

The user should be able to type "modify ACCOUNT_NUMBER" and get a list of options they can modify of the account object (name, number, balance)

Or the user can type "modify ACCOUNT_NUMBER OPTION" and the user will a prompt to enter a new value. Ensure the Account class limitations are enforced.

Delete Account

The user should be able to type "delete ACCOUNT_NUMBER" and the account will be removed from the bank. A confirmation message will be needed to complete the deletion.

Help

By typing help, the user gets a summary of how to user the program, including which arguments are possible and what options are available for each argument. Consider this like a MAN command in linux.

Summary

By typing summary, a display of the number of accounts created, the sum of all balances and the average balance for all accounts is generated (3 pieces of information)

Evaluation

Task #	Task Description	Task Weight
1	Account Class has appropriate instance variables	3
2	Account Class methods are correctly implemented	8
3	Bank Class has appropriate instance variables	3
4	Bank Class methods are correctly implemented	12
5	Runner class allows user to instantiate bank object	3
6	Runner class allows user to execute appropriate command	12
7	Runner class add account command	5
8	Runner class view accounts command	7
9	Runner class account details command	5
10	Runner class modify account command	25
11	Runner class delete account command	3
12	Runner class help command	10
13	Runner class summary command	4
	Total	100

Submission

Submit one file named Assignment2_DDDDDDDDD.zip, where DDDDDDDDD represents your student number. The zip file should only contain .java source files.
Your package name should be Assignment2_DDDDDDDDD

Deductions

The following deductions will apply if applicable

Name	Percentage	Description
Late Submission	10% per day	If assignment is not handed in by the due date, this penalty will be applied every 24 hours
File Not Named Correctly	5%	Case-Sensitive naming convention must be followed
Package Not Named Correctly	5%	Case-Sensitive naming convention must be followed
Submission contains files or folders that are not .java source files	2% per non-java source file	Any extra folder or file found will be penalized at the specified rate