CS 3331 – Advanced Object-Oriented Programming, Fall 2019
Programming Assignment 2

Instructor: Daniel Mejia

Instructions:
This assignment should be done individually. Your code must be written in Java. You must submit your assignment through Blackboard. In the comments (Javadoc) of your source code (at the beginning), write your given name, alias, date, course, instructor, lab assignment, lab description, and honesty statement.

Scenario:
You have begun organizing your bank. You now have a few customers who are using the bank. Refactor your existing code to meet the following requirements.

1. Create abstract classes
    a. Account
        i. Mickey & Friends (and enemies) have accounts
    b. Person
        i. Create methods for these classes
        ii. Mickey & Friends (and enemies) are persons
2. Create an interface (and implement it)
    a. Log
        i. Log inquiry
        ii. Log withdrawal
        iii. Log deposit
        iv. Etc.
3. Create classes that (may) inherit from the abstract class:
    a. Checking
    b. Savings
    c. Customer
    d. Credit
        i. Assume no minimum payment
        ii. Assume no interest
        iii. Assume all payments are towards the principle (Which is a negative number)
    e. RunBank (Where you have your Main Method)
        i. Classes should have appropriate methods
        ii. Classes should have appropriate fields

4. Read a file with information, and store it appropriately

    a. Pick a data structure that is appropriate
        i. Consider the time complexity
    b. Consider how your objects will interact with each other
5. Your system should be able to handle the following:
    a. Inquire a balance
    b. Deposit money
    c. Withdraw money
    d. Transfer money (i.e. from checking to credit account)
    e. Pay someone (i.e. Mickey pays Donald)
    f. Print out a response for all of the mentioned tasks
        i. Credit accounts should not be able to accept more than the balance
6. Allow for user interaction
    a. Ask if the user is an individual person (i.e. Mickey Mouse, Donald Duck)
        i. Transactions for a single person (i.e. Mickey Inquires balance)
        ii. Transactions between any two people (i.e. Mickey pays Donald)
    b. Ask if the user is a Bank Manager (Don't implement bank manager, simply introduce functionality)
        i. Have access to inquire about any account chosen (hint: method overloading)
            1. Example:

Console: "A. Inquire account by name"
"B. Inquire account by type/number"
User: "A"
Console: "Who's account would you like to inquire about?"
User: "Mickey Mouse"
<List All Mickey Information>

            2. Example:

Console: "A. Inquire account by name"
"B. Inquire account by type/number"
User: "B"
Console: "What account type?"
User: "Checking" (Or have it as an option)
Console: "What is the account number?"
User: "12345"
<List All Mickey Information>

        ii. Inquire and print all accounts (everything from the input file should be printed with updated balances) in a list

7. When the user has decided to exit, write a new account balance sheet (similar to the original input, except with the new values) as a text file
8. Log transactions
    a. Log should keep track of transactions only for that particular running session
    b. Sample (not necessarily the only way – make it better if you can) of the output file is as follows:

"Mickey Mouse made a balance inquiry on Checking-12345. Mickey Mouse's Balance for Checking-12345: $150"
"Mickey Mouse paid Donald Duck $50 from Checking-12345. Mickey Mouse's New Balance for Checking-12345: $100"
"Donald Duck received $50 from Mickey Mouse. Donald Duck's New Balance for Checking-12346: $75"
"Donald Duck made a balance inquiry on Savings-11112. Donald Duck's Balance for Savings-11112: $25"
"Donald Duck transferred $20 from Checking-12346 to Savings-11112. Donald Duck's Balance for Checking-12346: $55. Donald Duck's Balance for Savings-11112: $55"
"Mickey Mouse withdrew $20 in cash from Checking-12345. Mickey Mouse's Balance for Checking-12345: $80"

9. Handle all exceptions appropriately
    a. User cannot have negative money in an account (unless its credit)
10. Write the Javadoc for your system
11. Write the lab report describing your work (template provided)
    a. Any assumptions made should be precisely commented in the source code and described in the lab report.
12. Schedule a demo time with the TA.

To turn in (Blackboard) October 6, 2019 by 11:59pm:

1. Source code
2. Lab report
3. Updated Balance Sheet
4. Javadoc
5. Transaction log