The University of Texas at El Paso
Department of Computer Science
CS 3331 – Advanced Object-Oriented Programming, Summer 2020
Programming Assignment 3

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, date, course, instructor, lab assignment, lab description, and honesty statement.

Scenario:
You have set up the foundation of your bank. You now have many customers who are using the bank.

1. Refactor your existing code.
    a. Your code should handle all functionality from Programming Assignment 2 (PA2)
    b. Fix anything that should be corrected
        i. Are you using a dynamic data structure?
        ii. Are you using objects?
            1. Do these objects also have relationships with other classes?
        iii. Are you using appropriate algorithms?
            1. Are the runtimes of these algorithms at minimum O(n)?
        iv. Does it meet the open-closed principle?
        v. Etc.

2. Add new bank user functionality (create users from console)
    a. Users do not necessarily have all fields. The fields that are required for all users include: Name, DOB, Address, Phone Number, and at least a savings account
    b. Users should have the option to create a checking/credit account
    c. Identification numbers should increment by one from the last user created

3. Handle individuals with the same first name or same last name (assume that users will have either same first names or same last names, not both)

    a. Handle looking up users by name and returning the correct user

4. Handle receiving an input file that does not have the information in the same column order as PA2
    a. Name is not in column 1, last name is not in column 2, etc. (see input file)

5. User Interaction
    a. Single User Interaction Functionality (From PA2)
    b. Bank Manager Functionality (From PA2)
    c. Transaction Reader
        i. Handle all the transactions from the input file (Transaction Action)
        Format:

From First Name	From Last Name	From Where	Action	To First Name	To Last Name	To Where	Action Amount
Mickey	Mouse	Checking	pays	Donald	Duck	Checking	100

            Action Words (not changing):
            1. pays
                a. person pays person
            2. transfers
                a. person transfers from one account to another (savings -> checking, etc.)
            3. inquires
                a. Inquires only has "From" components (on a specific account)
            4. withdraws
                a. Withdraws only has "From" components (on a specific account)
            5. deposits
                a. Deposits only have "To" components and amounts (like cash)
        ii. Your code should handle the input of the file
            Example: when it reads "pays", your system should call the appropriate methods to pay someone
        iii. Successfully complete all transactions from input file
        iv. Consider the possibility of having transactions that will fail (i.e. withdrawing more than account balance)
            1. Write a message that this transaction failed in the console and why
            2. Continue working on the remaining transactions

6. Extend Bank Manager Functionality
    a. Keep all functionality from PA2
    b. Write a Bank Statement file for a specific user
        i. Choose a user by name
        ii. The formatting is up to you (Google sample bank statements for inspiration.) – Does not have to be fancy, but functional
        iii. All information about the user should be on the statement
            1. Name, address, phone, etc.
        iv. All transactions should be written
            1. For a particular session of running the code
        v. Only the bank manager can write a bank statement

7. Create a class BankStatement
    a. Customer Information
    b. Account Information
    c. Starting Balance (Beginning of session)
    d. Ending Balance (At the end of the program/at time requested)
    e. All transactions for that person
    f. Date of transaction (date of running the code)

8. 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 csv file
    a. Do not overwrite the initial input file (this new file may be overwritten each time the code is run)

9. Handle all exceptions appropriately
    a. User cannot have negative money in an account (unless its credit)
    b. A user cannot pay themselves (Example, Mickey Mouse should not be able to “pay” or “transfer” from his checking account to his checking account)
    c. A credit account should not be able to accept more money than the balance. For example, if there is a balance of –$20.00 in a credit account it should not be able to accept a $20.01 payment
    d. All transactions should notify the user if they succeed or fail
    e. All other common exceptions should be handled

10. Write the Javadoc for your system
    a. Generate the doc folder and include it with your submission

11. Demo with a partner, only functionality – don’t share code
    a. For doing this online, MS Teams allows you to take control of another user’s computer when sharing screens
    b. Make sure that they understand what you’re talking about
    c. Make sure that they cannot break your code
    d. Partners should focus on ensuring all requirements are met (functionality)
    e. Partners should try to break the system

12. 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.
    b. Write an additional section describing the demo of your partner
        i. What questions did you have about your partners functionality?
        ii. What concerns do you have about your partners functionality?
        iii. How did you try to break it?
            1. What test cases did you use?
            2. Explain why and how this is a black or white box texting

13. Use a design pattern in your code. Explain how your code follows this pattern in the report and why you chose this pattern.

14. Schedule a demo time with the TA\IA.

15. (If submission is passed the deadline) Your report must have an additional section entitled “Why I submitted late”. In that section, explain the reason why your submission was late. (Note: you will still be penalized the typical late penalty)

To turn in (Blackboard) October 19, 2020 by 11:59pm (Zip):

1. Source code (.java files)
2. Lab report (.pdf)
3. New Balance Sheet (.csv)
4. Javadoc (doc folder)
5. Transaction log (.txt)
6. Bank Statement for three users (You, Your favorite Disney Character on the list, a new user not yet on the list) (.txt)