The University of Texas at El Paso
Department of Computer Science
CS 3331 – Advanced Object-Oriented Programming
Instructor: Dr. Bhanukiran Gurijala
Fall 2024

Project Part 1

Academic Integrity Statement:
This work is to be done as a team. It is not permitted to share, reproduce, or alter any part of this assignment for any purpose. Students are not permitted to share code, upload this assignment online in any form, or view/receive/modify code written by anyone else. This assignment is part of an academic course at The University of Texas at El Paso and a grade will be assigned for the work produced individually by the student.

Instructions:
Your code must be written in Java. In the comment heading of your source code, you should write your name(s), date, course, instructor, programming assignment 1, lab description, and honesty statement. The honesty statement must state that you completed this work entirely on your own without any outside sources including peers, experts, online sources, or the like. Only assistance from the instructor, TA, or IA will be permitted. Generate Javadoc for your complete code.

Scenario:
You have decided to begin a brand-new bank called – *El Paso Miners Bank*. Your bank securely maintains the money and provides few financial services to the customers. You have a few customers who use the bank.

Part A:
Read the requirements described in Part B to complete Part A. Part A must be completed before implementing the requirements in Part B

1. Write a UML Use Case Diagram (Level II) for your system. With at least the following:
    a. 3 actors
    b. 3 Use Cases
    c. 1 includes

    d. 1 extends
2. Write 2 use case scenarios based on Part B.
3. Write a UML Class Diagram to structure your code using the classes, requirements, and concepts described in Part B

Part B

1. Create the following classes (Note: some may be abstract)
    a. Account
    b. Person
    c. Checking
    d. Saving
    e. Customer
    f. Credit
        i. Assume no minimum payment.
        ii. Assume no interest.
        iii. Assume all payments are towards the principle (which is a negative number).
            1. Note: If a credit starting balance states -10 this is the money that they have already charged to their account.
            2. They cannot go over their credit limit (credit max).
    g. RunBank (Where you have your Main Method)
    h. Any other additional Classes that you believe will be beneficial to help with successfully implementing this program.
        i. All Classes should have appropriate methods.
        ii. All Classes should have appropriate fields.
2. Read files with information and store the information appropriately.
    a. Pick a data structure that is appropriate.
        i. Consider the time complexity.
        ii. Consider space complexity.
    b. Consider the use of objects and how your objects will interact with each other.
3. Your system should be able to handle the following:
    a. Inquire about a balance.
    b. Deposit money to an account.
    c. Withdraw money from an account.
    d. Transfer money between accounts.
    e. Pay someone (i.e., Mickey pays Donald).
    f. Pick an informative response for all the mentioned tasks (i.e., Deposit of $XX successful, etc.)
        i. Credit accounts should not be able to accept more than the balance.

4. 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:	"Whose account would you like to inquire about?"
User:	"Mickey Mouse" <List All Mickey's Account Information>

            2. Example:

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

5. Log all transactions.
    a. Log should keep track of transactions for all sessions (i.e., the log file is not overwritten for every session, it appends transactions of a particular session to the log entries written in the previous session).
    b. A sample of how a log file can look is provided below (not necessarily the only way – feel free to make it better):

"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"

6. The user can exit the program by writing "EXIT" while in the main menu only. When the user exits the program
    a. Write a new updated Bank Users sheet (similar to the original input, except with the new values)
7. Handle all exceptions appropriately.
8. Write Javadoc for your system.
9. Write a lab report describing your work (template provided)
    a. Any assumptions made should be precisely mentioned in the source code and described in the lab report.
    b. The lab report should contain sample screenshots of the program being run in different circumstances, including successful and failing changes.
10. Complete an individual code review on your code (template provided)
11. Schedule a demo with TA/IA for both check-in and final deliverables
12. **If the submission is past 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)

Deadlines:

Check-in on October 16, 2024, by 11:59 pm:

1. UML Class Diagram Progress (.pdf)
2. UML Use Case Diagram Progress (.pdf)
3. Current Progress Source Code (.java) – Commit current progress up to this point.

For each item (1-3)

a. Does not have to be complete. Ensure there are no compilation errors.
b. Should be a significant amount of work done (as determined by instructional team).
c. Demo with TA/IA after 10/16. All the check-in demos should be completed by 10/16. The TA/IA will review for progress and provide informal feedback. No grade will be assigned.
d. The TA/IA working with the team will announce their availability for check-ins on Blackboard.

Final Deliverables on October 23, 2024, by 11:59 pm:

1. UML Class Diagram (.pdf)
2. UML Use Case Diagram (.pdf)
3. Use case Scenarios (.pdf)
4. Source code (.java files)
5. Lab report (.pdf file)
6. Javadoc (entire doc folder)
7. Updated Bank Users Sheet (.csv)
8. Log (.txt)