INFO C210
Course Project Phase 1: Simple Banking System
Project Phase 1

1. Introduction

You are asked to implement a simple Bank Management System for an *unrealistic* bank so that all information on customers and transactions are stored and accessed by bank operators. The system supports interactive inquiries from bank staff through a friendly user interface. You are required to create a Java application that implements the system.

For this project, you must work with your group to complete both phases of this project.

Your project will be graded based on the correctness and efficiency of your implementation along with the documentation and readability of your code. A grading rubric is attached with this project.

2. Specifications

System Overview

You will write a Java application to simulate a simple (unrealistic) bank system. This bank supports three different kinds of accounts: Checking, Gold, and Regular. Information common to all account types include account number, balance, and customer information (customer ID, customer name, etc) of the customer who owns the account.
The Checking account is interest free and charges transaction fees. The first two monthly transactions are free. It charges a $3 fee for every extra transaction (deposit, withdrawal).
The Gold account gives a fixed interest at 5% while the Regular account gives fixed interest at 6%, less a fixed charge of $10.

Whenever a withdrawal from a Regular or a Checking account is attempted and the given value is higher than the account's current balance, only the money currently available on the account is withdrawn. Unlike all other accounts, a Gold account holder can withdraw indefinitely. There is no transaction fee for customers with Gold or Regular accounts.

The system is host-centered program currently supporting only one kind of user – a bank operator who accesses the system to perform regular system administrative work. The following tasks are usually performed by bank operators and must be implemented in your system:

1. Create a Checking account
2. Create a Gold account
3. Create a Regular account
4. Deposit a specified amount of money to a given account
5. Withdraw money from a given account
6. Display account information
7. Remove an account
8. Apply end of month account updates. This function should be used once every end of month and will apply interest to the Regular and Gold accounts and deduct transaction fees from Checking accounts whenever applicable
9. Display Bank statistics: this feature will display a simple report for bank administrators that include things like the total sum of all accounts in the bank, number of zero-balance accounts, average balance of accounts, the account with largest balance, etc.
10. Exit: quits the system.

A sample run is attached as a reference

<u>Design and Implementation</u>

You need to design and implement a working solution for this problem. The design will include documentation for each of the classes that are part of the application and the methods in each class. You need to document your design and implementation in a readme file (see below). After completing your design, you will implement it. The number of classes you decide to use is up to you; however, your design must follow the good object oriented design practices and system restrictions.

3. System versions

The system should be implemented in two phases; the first phase is described above and will result in a working system that can be run from the command line. Upon completion of this phase, you will be asked to enhance and possibly redesign your system in a second phase.

4. Phase 1 Simplifying Assumptions

This is a simple system with a simple set of constraints described above. Uniqueness of account numbers and customer IDs are not accounted for. Moreover, there is no limit of the number of accounts of the same type a customer may have. You may have noticed also that the system currently does not provide an interface for customers to perform basic transactions such as deposit, withdrawal, check balance, etc. In this phase, your code is not required to handle exceptions resulted from faulty user inputs. This is something I will ask you to implement in the next phase

5. Submission

Submit a zip archive of all your Java files (only the ***.java** files, no .class files) by the due date. Notice that, each phase should be submitted separately. A readme file should be included in the Zip archive with each phase.
Your project's readme file must include a complete description of your design and implementation. Things to consider when writing the readme file:

- Include a UML diagram for the complete system
- Any design and implementation assumptions
- The objects in your system, what they do, and how they interact (including any patterns that you've used)
- Any challenges that you faced in the design or implementation of your program (this part is very important to me).
- Which parts you didn't do
- Known bugs (if I find bugs in your code that you did not report, I will assume you didn't test your code well enough to find the bug)
- Any specific procedures to running the system.

6. Questions about the Project

For any questions about this assignment, please post your questions in the Project's Open Discussion forum in Canvas. Please check this forum regularly and feel free to respond to other student's questions (I highly encourage and appreciate active participation in discussion forums). You are not allowed to include your entire code in the forum, if you feel that you must show your progress in a particular question, you can include your file(s) and submit that to me directly.