BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Department of Computer Science and Engineering

July 2023 CSE 308 Offline Assignment on

Basics of Object Oriented Programming (OOP)

In this assignment, you will implement a simple banking system using the basic concepts of Object Oriented Programming. You are allowed to use only Java as your programming language. You will also have to submit the corresponding detailed UML class diagram of your implementation in pdf format.

Accounts: In our banking system, there will be three types of accounts: *Savings*, *Student*, and *Fixed deposit*. For the sake of simplicity, assume an user can open one account only. You have to implement the following functionalities:

1. **Create Account:** After entering the system, users can request to open a new account with their name, account type, and initial deposit. A fixed deposit account must ensure the first deposit is at least 100,000$. After successful opening of the account, the user will be notified accordingly. If an account already exists against that name, an error message will be shown. Assume the names to be primary keys.

2. **Deposit:** An user can deposit any sum of money into an account, except for a fixed deposit account. The deposit amount must not be less than 50,000$ for the latter case.

3. **Withdraw:** Withdrawing is different across different account types:

   (a) A student account cannot withdraw more than 10,000$ in one transaction.

   (b) A fixed deposit account cannot withdraw if it has not reached a maturity period of one year.

   (c) A savings account cannot withdraw if the withdrawal results in a deposit of less than 1,000$.

   A transaction will be deemed invalid if the amount to withdraw exceeds the deposited sum.

4. **Request Loan:** Any user can request a loan. The maximum allowable loan for savings, student, and fixed deposit accounts are 10,000$, 1,000$, 100,000$, respectively. All loans have a fixed 10% interest rate, which will be deducted after one year. The loans must be approved by an employee of the bank.

5. **Query Deposit:** An user can query the amount of deposit at any time.

Interest rates on deposit for savings, student, and fixed deposit accounts are 10%, 5%, 15%, respectively. All accounts except the student accounts will be deducted an amount of 500$ of service charge after one year.

Employees: There will be three types of employees in a banking system: *Managing Director*, *Officer*, and *Cashier*. There are different operations of employees based on their roles:

1. **Lookup:** Any employee can see the deposit sum of any user account.
2. **Approve Loan:** The Managing director and officer can approve loan requests of users.
3. **Change Interest Rate:** The managing director has the discretion to change the interest rates of different types of accounts.
4. **See Internal Fund:** The managing director can see the internal funds.

Maintain a bank class for handling the accounts and employees, and the internal fund. The initial fund is 1,000,000$. Assume the bank class will not be instantiated more than once, and the following employees will be created at the time of bank instantiation: 1 Managing Director (MD), 2 Officers (O_1, O_2), 5 Cashiers ($C_1, \cdots, C_5$). Also maintain a clock variable to increment the year count of operation (assume all accounts are created at the same time).

Some sample test inputs and corresponding outputs are shown in Table 1.

Input	**Output**
	Bank Created; $MD, S_1, S_2, C_1, C_2, C_3, C_4, C_5$ created
`Create Alice Student 1000`	Student account for Alice Created; initial balance 1,000$
`Deposit 20000`	20,000$ deposited; current balance 21,000$
`Withdraw 12,000`	Invalid transaction; current balance 21,000$
`Query`	Current balance 21,000
`Request 500`	Loan request successful, sent for approval
`Close`	Transaction Closed for Alice
`Open S1`	S_1 active, there are loan approvals pending
`Approve Loan`	Loan for Alice approved
`Change Student 7.50`	You don't have permission for this operation
`Lookup Alice`	Alice's current balance 21,500$
`See`	You don't have permission for this operation
`Close`	Operations for S_1 closed
`Open Alice`	Welcome back, Alice
`Query`	Current Balance 21,500$, loan 500$
`Close`	Transaction Closed for Alice
`INC`	1 year passed
`Open Alice`	Welcome back, Alice
`Query`	Current balance, 22,525$, loan 500$
`Close`	Transaction Closed for Alice

Table 1: A sample input for banking system simulation

Your implementation must abide by the encapsulation, inheritance, and polymorphism properties of OOP. Make the design choices accordingly. Remember, the sole purpose of this sessional is to improve your design choices. A mere match with the output will not ensure good marks.

Please DO NOT COPY solutions from anywhere (e.g., your friends, seniors, internet). Any form of plagiarism, irrespective of source or destination, will result in -100% marks in the online/offline.

Submission Guideline:

1. Create a directory with your seven digit id as its name.
2. Put the source files (java and pdf) only into the directory created in step 1.
3. Don't forget about the UML class diagram pdf.
4. Compress the directory in .zip format. Submit the .zip in the moodle.

Submission Deadline: 11:55 PM, November 25, 2023