Course Code: M251

Course Title: Object Oriented Programming Using Java

Tutor Marked Assignment

Cut-Off Date: TBA Total Marks: 100

This TMA covers the first 7 sessions of M251. It is required to do the following:

- Create a Java project using your IDE and name it using your name and your student ID
- Implement all the required classes in the default package
- Create a Word file contains the following:
  - Answer of the given questions
  - A copy (~~not a snapshot~~) of the code of each class
  - A complete snapshot of the output
  - If part of the code does not work, try to explain your idea
  - The *header* of each page should contain your name and your student ID
- On LMS page of M251, there are 2 links and you should submit 2 files (one file per link):
  - The Word file you created
  - A compressed file contains the folder of your Java project

Plagiarism Warning:

As per AOU rules and regulations, all students are required to submit their own TMA work and avoid plagiarism. The AOU has implemented sophisticated techniques for plagiarism detection. You must provide all references in case you use and quote another person's work in your TMA. You will be penalized for any act of plagiarism as per the AOU's rules and regulations.

Declaration of No Plagiarism by Student (to be signed and submitted by student with TMA work):

I hereby declare that this submitted TMA work is a result of my own efforts and I have not plagiarized any other person's work. I have provided all references of information that I have used and quoted in my TMA work.

Name of Student:

Signature:

Date:

Overview:

A new bank wants to make a simple application to keep track of all accounts and transactions. In this TMA, it is required to help the bank manager implement the required application.

Requirements:

After a quick meeting with the bank manager, you got the following information:

- It is required to store all bank accounts in one collection and all the transactions happened in another collection.
- Each account has a unique account number, a holder and balance. There is a specific prefix (common for all accounts) that should be added to the holder's civil id to create the unique account number. In addition, it is not allowed for a holder to have more than one account. Furthermore, only three transactions are allowed on any account: deposit, withdrawal and transfer money to another account.
- Each holder has a unique civil ID (national id), a name and other attributes (add at least 2 attributes from your choice).
- For each transaction, it is required to store the account(s) affected, amount of money, date, and if they are successful. There are 3 types of transactions: deposit, withdrawal and transfer.

Moreover, you have been informed that the following operations happen frequently:

- Creating a new account
- Deposit money into a specified account
- Withdrawal money from a specified account
- Transfer money between two specified accounts
- Printing details of the transaction where the maximum amount of money has been transferred between 2 accounts
- Saving all accounts and transactions into a text file.
  For accounts, they should be sorted in ascending order (according to the civil id)

Analysis:

Q1: There are common attributes and methods between the three types of transactions. What is the best choice for designing and writing their codes? Explain your answer.

Q2: Draw a simple class diagram showing only relationships between the classes.

Implementation:

After analysing the given requirements, implement the required application:

- with **Object Oriented Programming** style
- following the rules of good programming style (e.g. adding comments, etc.)
- using **only the material covered in M251** (and its prerequisites)

Hints:

- For each class, it is required to implement constructors, setters, getters, toString() method, and any other necessary method
- If the user tries to do an operation that could violate the state of objects, the operation should be ignored and the application should display an error message (e.g. creating an account for the same holder twice, etc.)
- Checking equality of any 2 objects should be done via the equals() method
- There is a class that will do the main job of the bank as follows:
  - It has a collection to store the accounts and another one to store the transactions
  - It has static methods, one for each operation happens frequently
  - For each operation, a message should be displayed to the user to explain the status of the operation (i.e. if it was successful or not)

Testing:

After implementing the required classes, design and implement a testing class to test them as follows:

- Create at least 3 bank accounts and do some transactions on them
- Try to violate the state of the objects and show that your code prevents all violations
- Show that the other operations that happen frequently are working fine
- At the end, the required data should be stored into a text file and this file should be saved automatically inside the folder contains your Java project

Marks distribution:

- 22 marks for the essential classes
- 23 marks for the class represents Account
- 37 marks for the class that will do the main job
- 9 marks for the testing class
- 3 marks for following the given instructions
- 6 marks for answering the two questions

Important notes on penalties:

- Penalty on late submission: 10% per day
- Penalty for not including the Java project: 50% of the achieved mark.
- Penalty for not including the Word file: 50% of the achieved mark (after deducting any mark related to the Word file).
- Penalty on similarity percentage: according to AOU rules.

End of Assessment