Assignment 4

Deadline: Friday, 13 November 2020 at 6pm

Before you start: create a new folder, called "*Assignment_4*" for this assignment in your GitHub repository.

Exercise 1 (5 points)

In this assignment, you are going to create the structure of a bank management system.
The system must include the following entities:

Customer: The system must keep track of the data of a customer. The data to save are the customer's name, surname, age, bank account number and their savings (i.e., the amount of money available to the customer). Moreover, each customer is identified by a unique Identification Number (ID).
Each customer has a different type of credit card (depending on the customer's level). The customer can deposit and withdraw cash, use their credit cards, and pay the bills using a bank transfer. The customer's actions follow these rules:

- Depositing money simply adds the sum of money deposited to the customer's savings.
- Withdrawing money removes the specified sum to the customer's savings. A customer can not withdraw more than the amount of their savings. Withdrawing money returns the amount of money withdrawn.
- Paying with bank transfer is only allowed if the customer has enough savings to pay the specified amount of money. However, the method subtracts the paid amount from the customer's savings, but it does not return the amount of money paid.
- Paying by credit cards is always allowed, regardless of the amount of money available in the customer's savings. However, a customer can not pay more than the amount allowed by its credit card for each transaction (e.g., 2000 CHF for a regular credit card, as specified below).

There are three possible level of customers, which can perform different actions:

1. **Regular customer**: they have a Regular Credit Card and can pay only up to 2000 CHF by credit card per transaction. Payments with bank transfer do not have an amount limit.
2. **Golden customer**: they have a Gold Credit Card and can pay up to 5000 CHF by credit card per transaction. Payments with bank transfer do not have an amount limit.
3. **Platinum customer**: they have a Platinum Credit Card and can pay up to 10000 CHF by credit card per transaction. Payments with bank transfer do not have an amount limit.

To successfully complete a payment by credit card, the credit card must not be expired.

Credit card: A credit card entity should contain the data of the owner (name and surname), a serial number, a security code of three digits, and an expiration date. There are three different types of credit cards: Regular credit card, Gold credit card, and Platinum credit card.

Bank employee: bank employees supervise all the operations in the bank. Each employee has a name, surname, identification number, and a list of customers assigned to them. They can upgrade a customer from Regular level to Golden level given the Customer ID.
Employees are divided in three levels:

1) **Regular employee**: they do not have any special characteristic.
2) **Section chief**: each Section chief is assigned to a different city. In addition to being able to perform any regular employee's actions, a section chief can downgrade the level of a customer from Gold to Regular given the customer ID. Furthermore, differently from a regular employee, a section chief can upgrade a customer level from Golden to Platinum.
3) **Main Chief:** The main chief can perform all actions of a section chief but they are not assigned to any city. Moreover, they can downgrade a customer from Platinum level to any previous level (Golden or Regular).

Technicians: the bank has also a team of technicians at its disposal to fix technical issues in the system and support customers. There are two types of technicians:

1) **Web-technician:** they are in charge of managing the website. Each web-technician has a name, surname, and unique identification number. The only action they can perform is to fix the website. This method will wait for 30 seconds, print a message "reparations complete" and then conclude.
2) **Backend-technician**: they are in charge of managing the backend. They have a name, a surname and a unique Identification Number (ID). They can fix the back-end. This method will take as parameter the technician ID and print the String "fixed!" and then conclude.

Implement the above-mentioned functionalities using inheritance, interfaces, and abstract classes when appropriate.

Motivate your design choices in a **ANSWERS.md** file.

Draw a UML class diagram comprising all the classes in your code. Write a description of the class diagram (if needed) to further describe your choices. Address these points in the **ANSWERS.md** file in the assignment folder.

You can use tools to help you draw the class diagram, but you are **not** allowed to automatically generate it.

Exercise 2 (5 points)

Considering the code written in Exercise 1, complete the following points:

1. Implement unit test cases for all the functionalities of your code. *Think about **how** each step of your code should work and **what** would break it.*
2. Document all test cases using Javadoc. The Javadocs can be auto-generated, but you need to complete it with all method information and what the test is covering.

To complete the exercise all tests must pass. Getter and setter methods do not need to be tested.

A guide to Javadoc can be found here:
https://www.tutorialspoint.com/java/java_documentation.htm

The tags @param and @return should always be indicated (when applicable).

Please note: Auto-generated tests and incomplete Javadocs will not be accepted.

Important notes

Write all your answers in the **ANSWERS.md** file. Do not use a file with a different name. Also, please ensure that it is clear which answers belong to each exercise of the assignment.

Please note that no commits can be added to the assignment folder on GitHub after the deadline. Any commit added after the deadline **will not** be counted in the evaluation of the assignment.