Object Oriented Programming

Seminar 2

Bank application

Goals and Learning outcomes

The goal of this seminar is that the student:

• Develops knowledge and skills within object-oriented concepts

• Develops an application with an object-oriented approach

Introduction

In this seminar you shall implement a java console application, to be used in a bank, to manage the customers and their bank accounts. The bank has some customers, and each customer can have one or more bank accounts.

The bank offers two types of accounts. Saving account and Credit account.

Saving account:

A saving account has account number, account type, balance, and interest rate. The bank is offering an interest rate of 1% for all saving accounts. It shall be possible to perform transactions (deposit / withdraw), retrieve account number, calculate interest (balance * interest rate / 100) and present the account (account number, balance, account type, interest rate).

Credit account:

A credit account has account number, account type, balance, and interest rate (0.5% on the deposited money), and credit limit. The default credit limit is 5000 SEK.

When the credit account is created, the balance is 0 SEK, and the credit limit is set to 5000 SEK. The customer can withdraw money as long as the balance does not exceed the credit limit (-5000), as the credit limit is 5000.

Regarding the customers:

Each customer shall have a unique id number, name, and a list of the customer's bank accounts.

Requirements

You can use your creativity and get the output as you like, as long as the application does the minimum requirements below:

• Once you run the application, it shall ask for a customer id. The user inputs the id and the application will either type “Customer not found” if the customer is not in the system or, if the customer is in the system, the application will print the customer name, followed by a menu with the following options:

1. Create a saving account for the customer
2. Create a credit account for the customer
3. View all accounts
4. Deposit money
5. Withdraw money

• Choosing case 3, a list of all customer’s accounts, including account number, account type, current balance, etc. will be printed.

• Choosing case 4 or 5, the application will first ask the user to input the account number and also amount of money, and then it will do the action accordingly. In case of withdraw money, if there is not enough money/credit, the application shall print an error message.

Analyzing the task:

The following classes can be needed:

1. A super class Account and two sub-classes, Saving Account and Credit account (Generalization). Do we need to use the abstract keyword? An abstract class? Some abstract methods?
2. A Customer class (note that in the customer class, we need to have an ArrayList of type Account, to save the customer’s account objects in).
3. We can have all user interactions and the related methods in a separate class, like BankLogic. Our BankLogic shall have an ArrayList of type Customer, and we can create (hardcode) some Customer-objects and add them to that ArrayList. In the Banklogic we can create a method like startUp(), or any similar name. In that method we can have the user interactions. Then we can provide at least 5 other methods in that class (one for each of the above menu options (1-5)), that can be called from the startUp method depending on the customer’s choice.
4. A Main class with a main method. In the main method we just create an object of the BankLogic and call its startUp method.