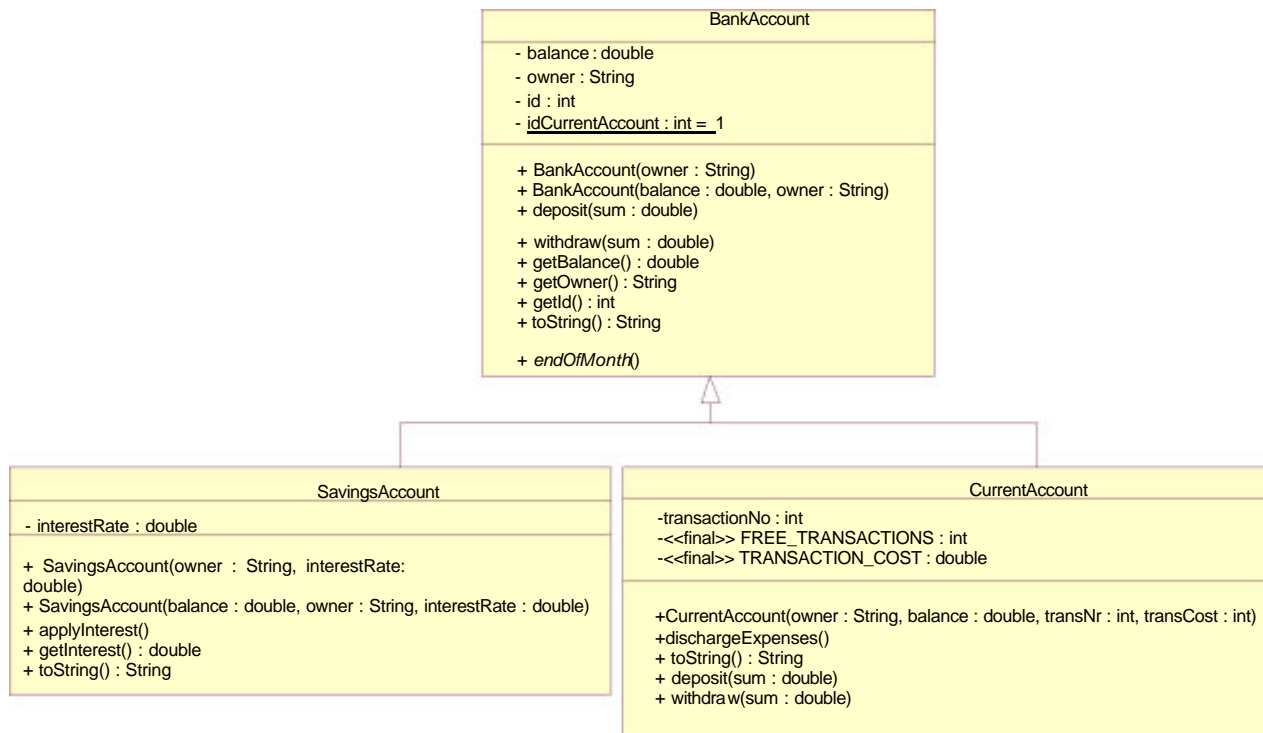


Abstract classes. Object class

Problem 1.

Let us consider the classes BankAccount, SavingsAccount introduced in the Lab2 and improve the conceptual model as presented in the following diagram:



A. The Account objects represents bank accounts internally described by a number (`id`), a balance in euro (`balance`) and the owner's name (`owner`).

B. An user of Account objects may:

1. ask the account number (`getId()`),
2. request the balance (`getBalance()`),
3. ask the owner (`getOwner()`)
4. deposit a money amount (`deposit()`),
5. withdraw money (`withdraw()`),
6. request information about the account (`toString()`).

In the case of withdrawing, your program should verify that the requested amount is available in the account. In the negative case, the operation will not withdraw the amount and the program will display an error message (without exiting from the program).

C. Bank accounts are created by providing either the owner's name and the initial balance or only the owner's name (so, the initial balance will be 0). When an account is created, its `id` will be automatically generated. This number is unique for each account. In order to do this, you should use a class variable (that is a variable declared static!) named `idCurrentAccount` and initialized to 1. Each time an account is created, the constructor will initialize the variable `id` with the current value of the variable `idCurrentAccount` and then increment this variable.

D. The class **SavingsAccount** is a bank account characterized by an interest that can be visualized at request. The interest will be applied at the end of the month when the interest will be added to balance. Let call `applyInterest()` this operation.

Finally, the information about the account, including the interest, may be displayed calling the method `toString()` .

E. The CurrentAccount class is a bank account that facilitates deposit and withdraw operations generically called transactions. The account owner has a number of free transactions (FREE_TRANSACTIONS_NR) which will be set at the current account creation time. All the other transactions will be paid with an amount of money memorized in the variable TRANSACTION_COST. This value will be set at the program loading time (the value is a constant for all current accounts) and used by the method dischargeExpenses() for operating expenses deduction.

The class diagram introduces an abstract method endOfMonth() for updating the accounts in the last day of each month.

Requirements:

- A. Write the program such that the methods applyInterest() and dischargeExpenses() will be called only at the end of the current month. In order to do this, in the class BankAccount you will add an abstract method called endOfMonth(), that, called for an BankAccount object, will verify if the current day is the last one of month and if so, it will call applyInterest() or dischargeExpenses() depending on the actual class of the object: SavingsAccount or CurrentAccount.

Note: the method toString() will return a string which contains the account's type (savings or current), besides other information. In order to do this, you may call the method getClass() of the class Object.

- B. Write the class TestAccount such that:

1. Create two current accounts and two savings accounts.
2. Print all relevant information in the four accounts.
3. Execute some transactions, printing the current balance after each transaction: deposit or withdraw.
4. Call the method endOfMonth().
5. Print once again all relevant information in the four accounts.