Computing 2 Homework Problems

Problem Set 5 (Inheritance using Bank Accounts)

Implement a base class `Account` and the classes `Savings` and `Checking` that are derived from it. To this end, adjust a sketch of an UML diagram that shows the base class and the two derived classes following the descriptions below first. Start by adding additional member variables and member functions to the appropriate classes.

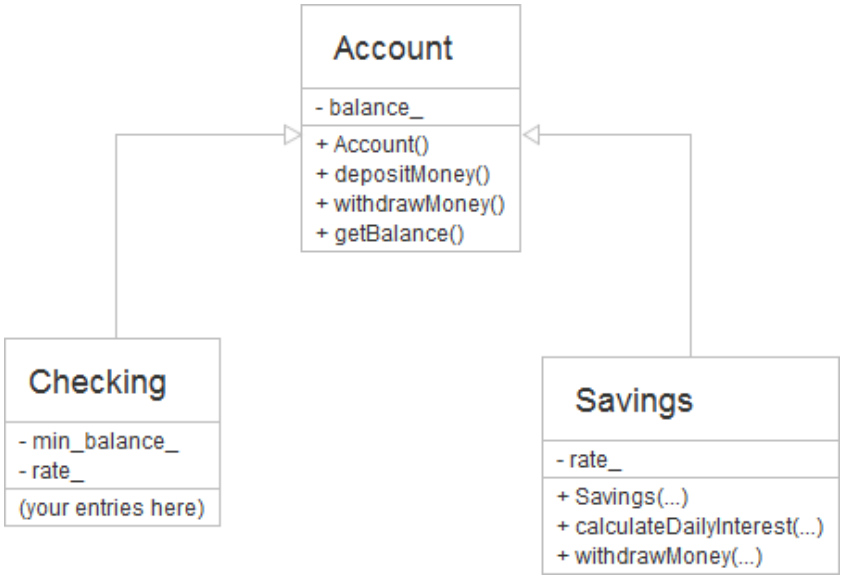


Figure 1: Sketch of an UML diagram for the inheritance hierachy starting with the base class `Account`. Use the Violet UML editor to open provided to you. You can either double click violetumleditor-3.0.0.exe or violetumleditor-3.0.0.jar. Both should start the application. Then open the file Account_Class_Diagramm.class.violet.html from within Violet. Finally, fill in the missing details, for example, data types, member functions, input arguments, return values..

In the next step, implement member functions `depositMoney()`, `withdrawMoney()` and `getBalance()`where necessary. Provide a function `calculateDailyInterest()` for both the class `Savings` and for the class `Checking`. This function computes and adds the daily interest. For calculations, assume that every month has 30 days and that the year has 12 months. You can calculate the daily interest as `balance * rate /kDaysPerMonth / kMonthsPerYear`. The checking account yields interest of 1 percent annually on *balances over $1,000*. The savings account yields interest of 2 percent annually on *the entire balance*. Please declare appropriate variables for the days per month, months per year, and interest rates to consider this.

In what follows, overload the functions `depositMoney()` and `withdrawMoney()` as follows:

1. Overlad `depositMoney()` with `operator+=(..)`
2. Overlad `withdrawMoney()` with `operator-=(..)`

Challenge Problems:

1. Change the `Checking` class so that a 1,- Euro fee is levied for deposits or withdrawals if more than three monthly transactions happen. Place the code for computing the fee into a private member function, `checkForFee()`, that you call from the member functions `depositMoney` and `withdrawMoney` functions of the `Checking` class.
2. Provide a public element function `resetTransactions()` of `Checking` class objects with which you can reset the number of transactions to zero.

Below you find part of the initial header for the Account base class.

Account.h (in part)

```
const int kDaysPerMonth = 30;
const int kMonthsPerYear = 12;


class Account
{
public:
   Account();
   Account(double b);
   void depositMoney(double amount);
   void withdrawMoney(double amount);
   double getBalance() const;
   void setBalance(double amount);
private:
   double balance;
};
```

There is also a part of a driver program and its output. The driver program already shows how the overloaded operators are to be used once they have been implemented (see comments).

```
int main()
{
   Checking c = Checking(1000.0);
   Savings s = Savings(1000.0);
   for (int i = 1; i <= kDaysPerMonth; i++)
   {
      c.depositMoney(i * 5.0);
         //c += i * 5.0;
      c.withdrawMoney(i * 2.0);
         //c -= i*2.0;
      s.depositMoney(i * 5.0);
         //s += i * 5.0;
      s.withdrawMoney(i * 2.0);
         //s -= i*2.0;
      c.calculateDailyInterest();
      s.calculateDailyInterest();
      if (i % 10 == 0)
      {
         cout << "day " << i << endl;
         cout << "Checking balance: " << c.getBalance() << endl;
         cout << "Savings balance: " << s.getBalance() << endl;
      }
   }
   c.resetTransactions();
   c.withdrawMoney(100);
   //c -= 100.0;
   cout << "Free withdrawal from checking account, that's nice: "
        << c.getBalance() << endl;
   return 0;
}
```

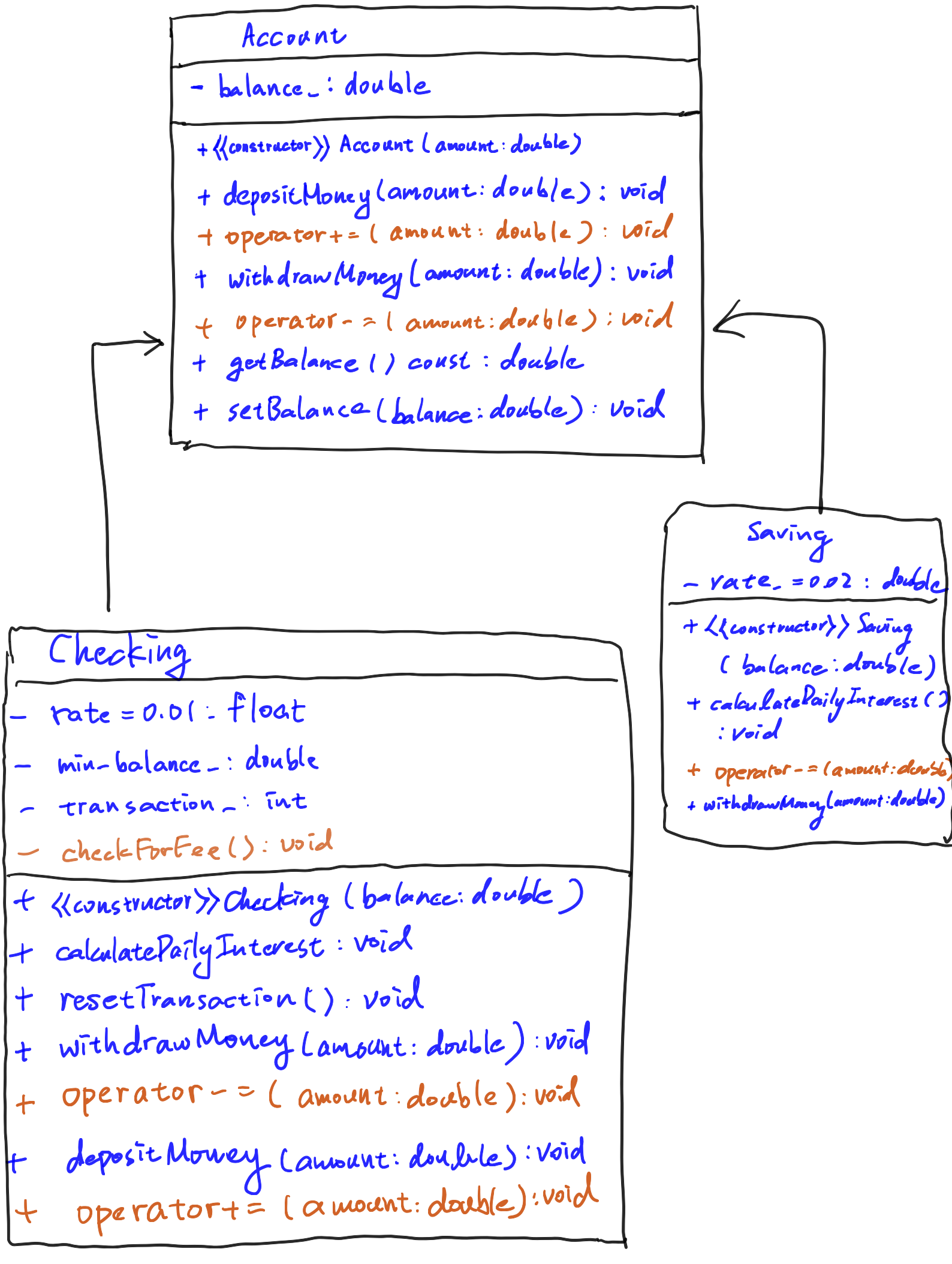
Outputs of main.cpp without checkforFee()

```
day 10
Checking balance: 1165.02
Savings balance: 1165.59
day 20
Checking balance: 1630.13
Savings balance: 1631.37
day 30
Checking balance: 2395.41
Savings balance: 2397.50
Free withdrawal from checking account, that's nice: 2295.41
```

Figure 2 Screen output after implementing the first part of the problem.

```
day 10
Checking balance: 1148.02
Savings balance: 1165.59
day 20
Checking balance: 1593.12
Savings balance: 1631.37
day 30
Checking balance: 2338.39
Savings balance: 2397.50
Free withdrawal from checking account, that's nice: 2238.39
```

Figure 3 Screen output after implementing the challenge part of the problem. In this case, fees are charged to the checking account when there are more than a minimum number of three checking account transactions.


Account
- balance_ : double
+ <<constructor>> Account (amount : double)
+ depositMoney (amount : double) : void
+ operator+= (amount : double) : void
+ withdrawMoney (amount : double) : void
+ operator-= (amount : double) : void
+ getBalance ( ) const : double
+ setBalance (balance : double) : void
Saving
- rate_ = 0.02 : double
+ <<constructor>> Saving (balance : double)
+ calculateDailyInterest ( ) : void
+ operator-= (amount : double)
+ withdrawMoney (amount : double)
Checking
- rate = 0.01 : float
- min_balance_ : double
- transaction_ : int
- checkForFee ( ) : void
+ <<constructor>> Checking (balance : double)
+ calculateDailyInterest : void
+ resetTransaction ( ) : void
+ withdrawMoney (amount : double) : void
+ operator-= (amount : double) : void
+ depositMoney (amount : double) : void
+ operator+= (amount : double) : void