Programming in C++: Exam

Candidate No.253074

January 13, 2022

1 Intro

For this task/ exam the questions relate to C++ classes, their concepts and inheritance, which will allow me to show my ability with these features. As such the two questions pertain to Banking and Savings accounts.

The first question wants the definition of a class BankAccounts, with required ability to take a range of inputs to define the classes private members and allow for the overload of operators to allow output of private members also (both as cout« or through the use of a file). Class functions were also made use of in this question. The second question wants the use of sub-classes and the use of class level functions to edit private members of these sub-classes, as well as their output.

2 Bank Account

For this program (BankAccount.cpp and Bank.hpp) I first defined and then imported/ included the BankAccount class, with said class holding the funds within the account as well as the interest rate. The inputs to create a variable of this class are two strings, funds and interest rate, or numerical inputs of pounds, pennies and interest rate or pounds and interest rate, as well as allowing for the input of nothing; this leading to an account with funds of £0.00 and an interest rate of 0.00%.

After this I had the '«' operator overwritten to allow for the output of the funds and interest rate directly from a variable of the BankAccount class. This allowed for both the writing to files of, and output of, results. I also made a class function .next_years_funds() which calculates and outputs the expected funds next year at the current interest rate, given interest each month; $\left[\text{fund} = \text{fund} * \left(1 + \frac{\text{Interest Rate}(\%)}{100}\right)^{12}\right]$. The results of this can be seen below for a range of inputs:

```
Balance: £ 159.29
Rate (%): 0.00
Next Years Funds (£): 159.29

Balance: £ 159.29
Rate (%): 1.00
Next Years Funds (£): 179.49

Balance: £ 159.00
Rate (%): 2.00
Next Years Funds (£): 201.65

Balance: £ 159.00
Rate (%): 2.00
Next Years Funds (£): 201.65

Balance: £ 0.00
Rate (%): 0.00
Next Years Funds (£): 0.00
```

Figure 1: This figure depicts the results from running BankAccount.cpp

This output is also identical to that of the files contents and is all to two decimal points, meaning the accuracy and overload of the '«' operator to be correct, as well as the function used to calculate the funds after a year of interest being correct; find it calculated to be correct, and match 'Next Years Funds' in figure 1, below:

$$159.29 + 159.29 * (1.00)^{12} = 159.29 \tag{1}$$
$$159.29 + 159.29 * (1.01)^{12} = 179.49 \tag{2}$$
$$159.00 + 159.00 * (1.02)^{12} = 201.65 \tag{3}$$
$$159.00 + 159.00 * (1.02)^{12} = 201.65 \tag{4}$$
$$0.00 + 0.00 * (1.000)^{12} = 0.00 \tag{5}$$

3 Savings Account

For this program (SavingsAccount.cpp and Savings.hpp) I first re-defined the BankAccount class to use protected members and then defined the SavingsAccount sub-class. This meant that to define a variable to the SavingsAccount class the same inputs as seen for the BankAccount class are accepted. It also shared the members and functions of the parent class (BankAccount) and allowed for the defining of functions for specific use on the SavingsAccount class.

These functions were .deposit(n) and .withdraw(n), of which added or removed (with a fee) £n of money from the classes funds; said fee was calculated using a portion of the expected gained interest, 60 days of the expected 365 days worth times the twelve months interest accumulates, the fee is therefore $\left[\text{Fee (£)} = n * \left(1 + \frac{\text{Interest Rate(\%)}}{100}\right)^{\frac{60*12}{365}} \approx n * \left(1 + \frac{\text{Interest Rate(\%)}}{100}\right)^{2}\right]$. These functions both print the initial balance, new balance and the interest rate of the account. When this is run on the same inputs as in the first question the following output is recorded.

```
Balance: £ 159.29
Rate (%): 0.00

Deposit £10:
Original funds (£): 159.29
New Funds (£):169.29
Interest rate (%): 0.00

Withdraw £20
Original funds (£): 169.29
New Funds (£):149.29
Interest rate (%): 0.00
-----------

Balance: £ 159.29
Rate (%): 1.00

Deposit £10:
Original funds (£): 159.29
New Funds (£):169.29
Interest rate (%): 1.00

Withdraw £20
Original funds (£): 169.29
New Funds (£):148.89
Interest rate (%): 1.00
-----------

Balance: £ 159.00
Rate (%): 2.00

Deposit £10:
Original funds (£): 159.00
New Funds (£):169.00
Interest rate (%): 2.00

Withdraw £20
Original funds (£): 169.00
New Funds (£):148.20
Interest rate (%): 2.00
-----------
```

```
Balance: £ 159.00
Rate (%): 2.00

Deposit £10:
Original funds (£): 159.00
New Funds (£):169.00
Interest rate (%): 2.00

Withdraw £20
Original funds (£): 169.00
New Funds (£):148.20
Interest rate (%): 2.00
-----------

Balance: £ 0.00
Rate (%): 0.00

Deposit £10:
Original funds (£): 0.00
New Funds (£):10.00
Interest rate (%): 0.00

Withdraw £20
Original funds (£): 10.00
New Funds (£):-10.00
Interest rate (%): 0.00
-----------
```

Figure 2: This figure depicts the results from running SavingsAccount.cpp

As it can be seen in figure 2, the functions state what is taking place, the original funds, the new funds and the interest rate as required. The .deposit(n) function can be seen to be correct through the use of simple addition, whereas we can do the calculations to show the 'New Funds' amount is correct, as is done below:

$$169.29 - 20 * \Big(1.00\Big)^2 = 149.29 \tag{6}$$

$$169.29 - 20 * \Big(1.01\Big)^2 = 148.89 \tag{7}$$

$$169.00 - 20 * \Big(1.02\Big)^2 = 148.20 \tag{8}$$

$$169.00 - 20 * \Big(1.02\Big)^2 = 148.20 \tag{9}$$

$$10.00 - 20 * \Big(1.00\Big)^2 = -10 \tag{10}$$

4 Conclusion

Each class and sub-class acted as expected and wanted, taking a range of inputs and being able to output themselves as needed; either through operators or functions as wanted. This led to both my programs being effective as accounts allowing for the holding of given funds and allowing for the calculation of said funds after interest, as well as the addition and removal of funds.

As a result of this each program met their requirements and worked as wanted, using the required features of classes and inheritance effectively. Therefore, I can say this task/ exam was a success, having allowed me to demonstrate my ability with said features.