Exercise 7

Implementing Classes 2: Simple Bank Accounts Hierarchy

Introduction:

Code reuse is an essential component when utilizing classes. In C++ this component among classes is called inheritance. This means that an existing class can be a base class for a new (derived) class. By using inheritance, a hierarchy of related classes can be created that share the same code and interface [1]. The syntax for class inheritance is as follows

```
class derivedclassID:
accessSpecifier baseClassID
{

};
//accessSpecifier – public, private or protected
```

The access specifier is the means in which a base class can be accessed by the derived class. For instance all the public members of a public class can be accessed directly by the derived class, while the private members cannot be accessed.

Learning Outcomes:

- Create a class for a simple bank account with member variables and functions.
- Derive this class into other bank account types
- Test these classes on the client code.

Problem background

1. Define the class bankAccount to store a bank customer's account number and balance. Suppose that account number is of type int, and balance is of type double. Your class should, at least, provide the following operations: set the account number, retrieve the account number, retrieve the balance, deposit and withdraw money, and print account information. Add appropriate constructors. Separate the implementation details and class definition
2. Every bank offers a checking account. Derive the class checkingAccount from the class bankAccount (designed in part (1)). This class inherits members to store the account number and the balance from the base class. A customer with a checking account typically receives interest, maintains a minimum balance, and pays service charges if the balance falls below the minimum balance. Add member variables to store this additional information. In addition to the operations inherited from the base class, this class should provide the following operations: set interest rate, retrieve interest rate, set minimum balance, retrieve minimum balance, set service charges, retrieve service charges, post interest, verify if the balance is less than the minimum balance, write a check, withdraw (override the method of the base class), and print account information. Add appropriate constructors. Separate the implementation details and class definition.
3. Every bank offers a savings account. Derive the class savingsAccount from the class bankAccount (designed in part (a)). This class inherits members to store the account

number and the balance from the base class. A customer with a savings account typically receives interest, makes deposits, and withdraws money. In addition to the operations inherited from the base class, this class should provide the following operations: set interest rate, retrieve interest rate, post interest, withdraw (override the method of the base class), and print account information. Add appropriate constructors. Separate the implementation details and class definition.

4. Write a program (client code) to test your classes designed in parts (2) and (3).