Homework 4: Inheritance & Polymorphism

Due Date: 24th November, 11:59 PM

Objective The goal of this assignment is to design and implement a C++ class hierarchy using **Inheritance** and **Polymorphism**. You will create a base **Account** class to manage generic banking data, and derive two specific classes—**SavingsAccount** and **CheckingAccount**—to handle specialized behaviors. You will also implement file processing to dynamically instantiate objects based on input data.

Background In Object-Oriented Programming, Inheritance allows for code reusability by defining a general base class and extending it with derived classes. Polymorphism allows a program to process objects of different derived classes through a single interface (pointers to the base class), ensuring that the correct specific behavior is executed at runtime.

Task: The Banking Hierarchy You will implement a complete banking system consisting of three classes (**Account**, **SavingsAccount**, **CheckingAccount**) and a **main** driver program that processes a data file.

Part 1: Account.h (The Base Class) Create a header file named **Account.h**. This class represents a generic bank account.

Protected Members:

- int accountNo;
- string name;
- double balance;

Public Members:

- **Constructor:**
  - Account(int accNo, string name, double balance);
  - Initializes the protected member variables.
- **Destructor:**
  - virtual ~Account();
  - Required to ensure correct memory deallocation for derived objects.
- **Methods:**
  - virtual string toString();
  - Returns a string formatted as: Account No: [AccNo] | Name: [Name] | Balance: $[Balance]

Part 2: Derived Classes (Savings & Checking) Create two new classes that inherit publicly from Account.

Class 1: SavingsAccount

- **Private Member:** double interestRate; (e.g., 0.05 represents 5%)
- **Constructor:**
    - SavingsAccount(int accNo, string name, double balance, double rate);
    - Pass the generic data to the Account constructor and initialize the interest rate.
- **Method Override:**
    - string toString();
    - Returns the string from the base class, concatenated with: " [Type: Savings, Rate: X%]"

Class 2: CheckingAccount

- **Private Member:** double fee; (e.g., 2.50 represents $2.50)
- **Constructor:**
    - CheckingAccount(int accNo, string name, double balance, double fee);
    - Pass the generic data to the Account constructor and initialize the fee.
- **Method Override:**
    - string toString();
    - Returns the string from the base class, concatenated with: " [Type: Checking, Fee: $X]"

Part 3: Main.cpp (File Processing) You must create a file named **clients.txt** and write a main program to process it.

Input File Format (clients.txt): The first character on each line indicates the account type:

- **'S'**: Savings Account (The last number is the Interest Rate)
- **'C'**: Checking Account (The last number is the Fee)

File Content:

```
S 101 Alice 1000.0 0.05
C 102 Bob 500.0 2.0
S 103 Charlie 2000.0 0.03
C 104 David 150.0 5.0
```

Requirements for Main:

1. **Data Structure:** Use a **vector<Account*>** to store the list of accounts.
2. **File Reading:**
    - Open and read **clients.txt**.
    - Determine the account type based on the first character ('S' or 'C').
    - Dynamically allocate memory (using **new**) for the specific derived class (**SavingsAccount** or **CheckingAccount**) using the data read from the line.

    - Store the pointer in the vector.
3. **Output:** Iterate through the vector and print the details of each account by calling **toString()**.
4. **Memory Management:** Iterate through the vector and delete all pointers before the program terminates.

Submission Guidelines

1. Add the following header comment block to the top of each file:
   // Name: Your Full Name
   // Student ID: YourID
   // Assignment: Homework 4: Inheritance & Polymorphism

2. Create a `.zip` archive containing the following files:
Account.h, Account.cpp, SavingsAccount.h, SavingsAccount.cpp, CheckingAccount.h, CheckingAccount.cpp, Main.cpp, clients.txt and
A **screenshot** of your complete program run from `main.cpp`.

3. Name your archive file in the format: **Lastname_hw4.zip** (e.g., `Smith_hw4.zip`).

4. Submit the `.zip` file to Canvas