Banking system

Design class for 'Account' and 'Bank' Include function for depositing, withdrawing, checking account balance and creating new accounts.

```
#include <iostream>
using namespace std;
class Account {
public:
   int accno;
   string accName;
   double balance;
};
class Bank : public Account {
public:
   Bank(int acc_no, string acc_name, double initialBalance)
   {
      accno = acc_no;
      balance = initialBalance;
      accName = acc_name;
   }
   void deposit() {
      double depositAmount;
      cout << "Enter Deposit Amount: ";
      cin >> depositAmount;
      balance += depositAmount;
   }
   void withdraw() {
      double withdrawAmount;
      cout << "Enter Withdraw Amount: ";
```

```
    cin >> withdrawAmount;
    if (withdrawAmount > balance)
    {
        cout << "Insufficient" << endl;
    }
    else
    {
        balance -= withdrawAmount;
    }
}
void checkingAcc() {
    cout << "Account No: " << accno << endl;
    cout << "Name: " << accName << endl;
    cout << "Balance: " << balance << endl;
}
void createNewAcc() {
    int acc_num;
    string acc_naming;
    double initial_balance;
    cout << "Enter Account Number: ";
    cin >> acc_num;
    cout << "Enter Account Name: ";
    cin >> acc_naming;
    cout << "Enter Initial Balance: ";
    cin >> initial_balance;
    accno = acc_num;
    balance = initial_balance;
    accName = acc_naming;
```

```
    cout << "Account created successfully. Account number: " << acc_num << endl;
}
void operateAccount() {
    char choice;
    do {
       cout << "Select operation: (D)eposit, (W)ithdraw, (C)heck Balance, (N)ew Account, (E)xit: ";
        cin >> choice;
        switch (choice) {
            case 'D':
            case 'd':
                deposit();
                break;

            case 'W':
            case 'w':
                withdraw();
                break;

            case 'C':
            case 'c':
                checkingAcc();
                break;

            case 'N':
            case 'n':
                createNewAcc();
                break;
```

```
            case 'E':
            case 'e':
                cout << "Exiting program." << endl;
                break;
            default:
                cout << "Invalid operation. Please try again." << endl;
                break;
        }
    } while (choice != 'E' && choice != 'e');
  }
};
int main()
{
Bank b(23456781, "NBwi", 200.00); // Default values, as these will be set in the createNewAcc function
  b.operateAccount();
  return 0;
}
```

Code Explanation

This system allows users to perform operations such as deposit, withdraw, check account balance, create a new account, and exit the program.

Class **Account**

This class represents a basic bank account with attributes **accno** (account number) , **accName** (account name), and **balance** (account balance).

Class **Bank**

- The **Bank** class is derived from the **Account** class using public inheritance.
- The constructor **Bank** initializes **accno** (account number), **accName** (account name), and **balance** (initial balance) based on the parameters provided.

Member Function of Bank class

void deposit( )

- Takes user input for a deposit amount and adds it to the account balance.

void withdraw( )

- Takes user input for a withdrawal amount and checks if it's greater than the account balance. If yes, displays "Insufficient." Otherwise, subtracts the withdrawal amount from the balance.

void checkingAcc( )

- Displays the account number, account name, and balance.

void createNewAcc( )

- Takes user input for a new account's details (account number, account name, initial balance) and sets them. Displays a success message.

void operateAccount( )

- Provides a menu-driven interface for the user to choose operations (Deposit, Withdraw, Check Balance, New Account, Exit) in a loop until the user chooses to exit.

“main” function (int main)

- creating the object of the class **bank** with arguments.
- Invokes the **operateAccount** method to start the banking program.

In summary, this program allows the user to interactively perform banking operations on a single account. The user can deposit, withdraw, check the balance, create a new account, or exit the program. The account details are managed using object-oriented principles with classes and inheritance.