banco.cpp

```cpp
#include "banco.h"

#include <iostream>

using namespace std;

Banco::Banco()
{

    this -> numeroAtualdeContas = 4;
    this-> pContas = new Conta[this -> numeroAtualdeContas];

    this -> pContas[0] = {1234, 1, "Joao", "Corrente", 300};
    this -> pContas[1] = {4567, 2, "Jose", "Poupanca", 800};
    this -> pContas[2] = {7890, 3, "Maria", "Corrente", 1000};
    this -> pContas[3] = {8956, 4, "Madalena", "Poupanca", 2000};




    this->gerentes[0] = Gerente(6655,1,"Bernardo");
    this->gerentes[1] = {0405,2,"Moyses"};


}

Banco::~Banco()
{
}

Conta *Banco::buscaConta(int numero)
{
    for (int i = 0; i < this -> numeroAtualdeContas; i++)
    {
        if (numero == this->pContas[i].numero)
        {
            return &this->pContas[i];
        }
    }

    return nullptr;
}

Gerente* Banco::buscaGerente(int numero)
{
    for (int i = 0; i < NUMGERENTES; i++)
    {
        if (numero == this->gerentes[i].numero)
        {
            return &this->gerentes[i];
        }
    }

    return nullptr;
}
```



```cpp
void Banco::atendimentoCliente()
{
    Conta *contaCliente;
    int numC = 0;
    int senhain;
    bool atendimento = true;
    int numcontadestino;
    Conta *contaDestino;

    cout << "Bem vindo ao sistema de atendimento ao Cliente" << endl;
    cout << "Digite o numero da sua conta: ";
    cin >> numC;

    contaCliente = this->buscaConta(numC);

    if (contaCliente == nullptr)
    {
        cout << "Conta invalida" << endl;
    }
    else
    {
        cout << "Digite a sua senha: ";
        cin >> senhain;

        if (contaCliente->validaSenha(senhain))
        {
            cout << "Ola " << contaCliente->titular << endl;
            while (atendimento)
            {
                int op;
                double valor;
                bool status_transf = true;
                cout << "Qual operacao deseja fazer? (1 - Saque, 2 - Deposito, 3 - Ver Saldo,
4 - Transferencia, 5 - Sair): ";
                cin >> op;
                switch (op)
                {
                case 1:
                    cout << "Digite o valor: ";
                    cin>>valor;
                    contaCliente->saque(senhain,valor);
                    break;
                case 2:
                    cout << "Digite o valor: ";
                    cin>>valor;
                    contaCliente->deposito(valor);
                    break;
                case 3:
                    cout << "Saldo: R$ "<<contaCliente->getSaldo(senhain)<<endl;
                    break;
                case 4:
                    cout<< "Digite o número da conta de destino: ";
                    cin>>numcontadestino;
                    if(numcontadestino <= this->numeroAtualdeContas){
                        contaDestino = this->buscaConta(numcontadestino);
                        cout<< "Digite o valor a ser transferido: ";
                        cin>>valor;
```

```
                        status_transf = contaCliente-
>transferencia(valor,senhain,contaDestino);
                        if(status_transf){
                            cout<<"Transferencia de R$"<<valor<<" realizada com sucesso para
conta "<<numcontadestino<<"."<<endl;
                        }
                        else{
                            cout<<"Valor de Transferencia inválido "<<endl;
                        }
                        break;
                    }
                    else{
                        cout<<"O número da conta é inválido. "<<endl;
                        break;
                    }

                case 5:
                    atendimento = false;
                    break;
                }
            }
        }
        else
        {
            cout << "Senha invalida" << endl;
        }
    }
}
void Banco::atendimentoGerente()
{
    int numerogerente;
    int senhain;
    bool atendimento = true;
    Gerente* contaGerente;

    cout<<"Bem vindo Gerente!"<<endl;
    cout<<"Digite o seu número de funcionário: "<<endl;
    cin>>numerogerente;

    contaGerente = this->buscaGerente(numerogerente);

    if (contaGerente == nullptr)
    {
        cout << "Conta invalida" << endl;
    }
    else
    {
        cout << "Digite a sua senha: ";
        cin >> senhain;

        if (contaGerente->validaSenha(senhain))
        {
            cout << "Ola " << contaGerente->nome << endl;
            while (atendimento)
            {
                int op;
                cout << "Qual operacao deseja fazer? (1 - Criar nova Conta, 2 - Sair): ";
                cin >> op;
                switch (op)
                {
```



```
                case 1:
                    this ->criaContaNova(senhain,contaGerente);
                    break;
                case 2:
                    atendimento = false;
                    break;
                }
            }
        }
        else
        {
            cout << "Senha invalida" << endl;
        }
    }
}

void Banco::atendimento(){
    int escolha;
    bool status_atendimento = true;
    bool atendimento_geral = true;
    while(atendimento_geral){
        cout<<"Bem vindo ao sistema de atendimento do banco "<<endl;
        cout<<"O que você deseja: 1- Atendimento ao Cliente, 2- Atendimento ao Gerente, 3 -
Sair"<<endl;
        cin>>escolha;

        switch(escolha){
            while(status_atendimento){
                case 1:
                this -> atendimentoCliente();
                break;
            case 2:
                this -> atendimentoGerente();
                break;
            case 3:
                status_atendimento = false;
                atendimento_geral = false;
                break;
            }
        }
    }
}

int Banco::getNumeroAtualContas(int senha,Gerente* contaGerente){
    if (contaGerente->validaSenha(senha)){
        return this->numeroAtualdeContas;
    }
    else{
        cout<<"Senha inválida";
        return -10000;
    }
}

void Banco::criaContaNova(int senha, Gerente* contaGerente){
    int senhaConta;
    string nometitular;
    string tipoConta;
    double saldoConta;
    int numeroContas;
```

```
    numeroContas = getNumeroAtualContas(senha,contaGerente);
    pcopiaContas = new Conta[(numeroContas) + 1];

    for(int i = 0; i<numeroContas ; i++){
        pcopiaContas[i] = pContas[i];
    }

    delete[] this->pContas;
    cout<<"Entre com os dados da nova conta: "<<endl;
    cout<<"Digite o titular: "<<endl;
    cin>>nometitular;



    pcopiaContas[numeroContas].titular = nometitular;


    cout<<"Digite o tipo de conta: "<<endl;
    cin>>tipoConta;

    pcopiaContas[numeroContas].tipo = tipoConta;

    cout<<"Digite o saldo da conta: "<<endl;
    cin>>saldoConta;

    pcopiaContas[numeroContas].setSaldo(saldoConta);


    cout<<"Digite a senha da conta: "<<endl;
    cin>>senhaConta;

    pcopiaContas[numeroContas].setSenha(senhaConta);


    pcopiaContas[numeroContas].numero = (numeroContas + 1);

    cout<<"Conta criada com sucesso "<<endl;
    cout<<"Exibindo dados públicos da nova conta criada: "<<endl;
    pcopiaContas[numeroContas].exibeDados();

    pContas = new Conta[(numeroContas) + 1];

    for(int i = 0; i<(numeroContas + 1) ; i++){
        pContas[i] = pcopiaContas[i];
    }

    delete[] this->pcopiaContas;

    this->numeroAtualdeContas++;
    cout<<"Número atual de contas: "<<this->numeroAtualdeContas<<endl;
}
```

banco.h

```
#ifndef BANCO_H
#define BANCO_H

#include "conta.h"
#include "gerente.h"
#include <string>


#define NUMGERENTES 5

class Banco
{
private:

    Conta* pContas;
    Conta* pcopiaContas;

    Gerente gerentes[NUMGERENTES];
    int numeroAtualdeContas;
public:
    Banco();
    ~Banco();
    Conta* buscaConta(int numero);
    Gerente* buscaGerente(int numero);
    void atendimento();
    void atendimentoCliente();
    void atendimentoGerente();
    int getNumeroAtualContas(int senha,Gerente* contagerente);
    void criaContaNova(int senha, Gerente* contaGerente);
};


#endif
```

conta.cpp

```cpp
#include "conta.h"
#include <iostream>

Conta::Conta()
{
    this->numero = 0;
    this->senha = 1111;
    this->titular = "Nenhum";
    this->saldo = 0;
}

Conta::Conta(int senha, int numero, std::string titular, std::string tipo, double saldo)
{
    this->senha = senha;
    this->numero = numero;
    this->titular = titular;
    this->tipo = tipo;
    if(saldo>0)
    {
        this->saldo = saldo;
    }
    else
    {
        std::cout<<"Saldo inicial invalido"<<std::endl;
    }
}

Conta::~Conta()
{
}

void Conta::exibeDados()
{
    std::cout<< "Titular: "<<this->titular<<std::endl;
    std::cout<< "Numero: "<<this->numero<<std::endl;
    std::cout<< "Tipo: "<<this->tipo<<std::endl;
}

double Conta::getSaldo(int senha)
{
    if(senha==this->senha)
    {
        return this->saldo;
    }
    else
    {
        std::cout<<"Senha inválida"<<std::endl;
        return -10000000;
    }

}

void Conta::setSaldo(double valor)
{
    this->saldo = valor;
}

void Conta::setSenha(int novaSenha)
{
    this->senha = novaSenha;
}

void Conta::deposito(double valor)
```



```cpp
{
    if(valor>0)
    {
        this->saldo+=valor;
    }
    else
    {
        std::cout<<"Valor invalido"<<std::endl;
    }

}

void Conta::saque(int senha, double valor)
{
    if(senha==this->senha)
    {
        if(this->saldo>valor)
        {
            this->saldo-=valor;
            std::cout<<"Saque de R$"<<valor<<" realizado com sucesso."<<std::endl;
        }
        else
        {
            std::cout<<"Saldo insuficiente"<<std::endl;
        }
    }
    else
    {
        std::cout<<"Senha invalida"<<std::endl;
    }

}

bool Conta:: transferencia(double valor, int senha, Conta* cdestino){
    if(valor >0){
      if(senha == this ->senha){
        if(this-> saldo >= valor){
            this->saldo -= valor;
            cdestino->deposito(valor);
            return true;
        }
        else
        {
                return false;
        }
      }
   }
}

bool Conta::validaSenha(int senha)
{
    if(this->senha == senha)
    {
        return true;
    }
    else
    {
        return false;
    }

}
```

conta.h

```
#ifndef CONTA_H
#define CONTA_H
#include <string>


class Conta
{
private:
    double saldo;
    int senha;
public:
    Conta();
    Conta(int senha, int numero, std::string titular, std::string tipo, double saldo);
    ~Conta();
    int numero;
    std::string titular;
    std::string tipo;
    void exibeDados();
    double getSaldo(int senha);
    void setSaldo(double valor);
    void setSenha(int novaSenha);
    void deposito(double valor);
    void saque(int senha, double valor);
    bool validaSenha(int senha);
    bool transferencia(double,int,Conta*);

};


#endif
```

gerente.cpp

```cpp
#include "gerente.h"
#include <iostream>
#include <string>

using namespace std;

Gerente::Gerente()
{
    this->senha = 1111;
    this->numero = 0;
    this->nome = "Nenhum";
}

Gerente::Gerente(int senha, int numero, string nome)
{
    this->senha = senha;
    this->numero = numero;
    this->nome = nome;
}

Gerente::~Gerente()
{
}

void Gerente::exibeDados()
{
    std::cout<< "Nome: "<<this->nome<<std::endl;
    std::cout<< "Numero: "<<this->numero<<std::endl;
}

void Gerente::setSenha(int novaSenha)
{
    this->senha = novaSenha;
}

bool Gerente::validaSenha(int senha)
{
return (this->senha == senha);
}
```

gerente.h

```
#ifndef GERENTE_H
#define GERENTE_H

#include <string>


class Gerente
{

private:
    int senha;

public:
    Gerente();
    ~Gerente();
    Gerente(int senha, int numero, std::string nome);
    int numero;
    std::string nome;
    void setSenha(int novaSenha);
    bool validaSenha(int senha);
    void exibeDados();

};
#endif
```

main.cpp

```cpp
#include <iostream>
#include "banco.h"
using namespace std;

int main()
{
    Banco b1;
    b1.atendimento();

    return 0;
}
```