#include <iostream>

#include <vector>

#include <string>

#include <ctime>

#include <iomanip>

using namespace std;

// Transaction class

class Transaction {

public:

string type;

double amount;

string date;

string note;

Transaction(const string &t, double a, const string &n = "")

: type(t), amount(a), note(n) {

// store current date-time

time\_t now = time(0);

char buf[80];

strftime(buf, sizeof(buf), "%Y-%m-%d %H:%M:%S", localtime(&now));

date = buf;

}

void display() const {

cout << setw(10) << type

<< " | Amount: " << setw(8) << fixed << setprecision(2) << amount

<< " | Date: " << date

<< " | " << note << "\n";

}

};

// Account class

class Account {

private:

int accountNumber;

double balance;

vector<Transaction> transactions;

public:

Account(int accNo) : accountNumber(accNo), balance(0.0) {}

int getAccountNumber() const { return accountNumber; }

double getBalance() const { return balance; }

void deposit(double amount, const string &note = "") {

balance += amount;

transactions.emplace\_back("Deposit", amount, note);

}

bool withdraw(double amount, const string &note = "") {

if (amount > balance) {

cout << "Insufficient balance!\n";

return false;

}

balance -= amount;

transactions.emplace\_back("Withdraw", amount, note);

return true;

}

bool transfer(Account &to, double amount) {

if (withdraw(amount, "Transfer to " + to.getAccountNumberStr())) {

to.deposit(amount, "Transfer from " + getAccountNumberStr());

return true;

}

return false;

}

string getAccountNumberStr() const {

return to\_string(accountNumber);

}

void displayTransactions() const {

if (transactions.empty()) {

cout << "No transactions found.\n";

return;

}

cout << "\n--- Transaction History for Account " << accountNumber << " ---\n";

for (const auto &t : transactions) {

t.display();

}

}

};

// Customer class

class Customer {

private:

string name;

string address;

vector<Account> accounts;

public:

Customer(const string &n, const string &a) : name(n), address(a) {}

string getName() const { return name; }

Account& createAccount(int accNo) {

accounts.emplace\_back(accNo);

return accounts.back();

}

Account\* getAccount(int accNo) {

for (auto &acc : accounts) {

if (acc.getAccountNumber() == accNo) return &acc;

}

return nullptr;

}

void displayAccounts() const {

cout << "\n--- Accounts for " << name << " ---\n";

for (const auto &acc : accounts) {

cout << "Account No: " << acc.getAccountNumber()

<< " | Balance: " << fixed << setprecision(2) << acc.getBalance() << "\n";

}

}

};

// Main Banking System

class BankingSystem {

private:

vector<Customer> customers;

int nextAccountNumber = 1001;

public:

Customer& createCustomer(const string &name, const string &address) {

customers.emplace\_back(name, address);

return customers.back();

}

Customer\* findCustomer(const string &name) {

for (auto &c : customers) {

if (c.getName() == name) return &c;

}

return nullptr;

}

int generateAccountNumber() {

return nextAccountNumber++;

}

};

int main() {

BankingSystem bank;

while (true) {

cout << "\n==== Banking System Menu ====\n";

cout << "1. Create Customer\n";

cout << "2. Create Account\n";

cout << "3. Deposit\n";

cout << "4. Withdraw\n";

cout << "5. Transfer\n";

cout << "6. View Accounts\n";

cout << "7. View Transactions\n";

cout << "0. Exit\n";

cout << "Enter choice: ";

int choice;

cin >> choice;

if (choice == 0) break;

string name, address;

Customer \*customer;

int accNo;

double amount;

switch (choice) {

case 1:

cout << "Enter customer name: ";

cin.ignore();

getline(cin, name);

cout << "Enter address: ";

getline(cin, address);

bank.createCustomer(name, address);

cout << "Customer created successfully!\n";

break;

case 2:

cout << "Enter customer name: ";

cin.ignore();

getline(cin, name);

customer = bank.findCustomer(name);

if (customer) {

accNo = bank.generateAccountNumber();

customer->createAccount(accNo);

cout << "Account created successfully! Account No: " << accNo << "\n";

} else {

cout << "Customer not found.\n";

}

break;

case 3:

cout << "Enter customer name: ";

cin.ignore();

getline(cin, name);

customer = bank.findCustomer(name);

if (customer) {

cout << "Enter account number: ";

cin >> accNo;

Account \*acc = customer->getAccount(accNo);

if (acc) {

cout << "Enter deposit amount: ";

cin >> amount;

acc->deposit(amount);

cout << "Deposit successful.\n";

} else {

cout << "Account not found.\n";

}

} else {

cout << "Customer not found.\n";

}

break;

case 4:

cout << "Enter customer name: ";

cin.ignore();

getline(cin, name);

customer = bank.findCustomer(name);

if (customer) {

cout << "Enter account number: ";

cin >> accNo;

Account \*acc = customer->getAccount(accNo);

if (acc) {

cout << "Enter withdrawal amount: ";

cin >> amount;

acc->withdraw(amount);

} else {

cout << "Account not found.\n";

}

} else {

cout << "Customer not found.\n";

}

break;

case 5: {

string nameTo;

cout << "Enter sender name: ";

cin.ignore();

getline(cin, name);

customer = bank.findCustomer(name);

if (!customer) {

cout << "Sender not found.\n";

break;

}

cout << "Enter sender account number: ";

cin >> accNo;

Account \*senderAcc = customer->getAccount(accNo);

if (!senderAcc) {

cout << "Sender account not found.\n";

break;

}

cout << "Enter receiver name: ";

cin.ignore();

getline(cin, nameTo);

Customer \*receiver = bank.findCustomer(nameTo);

if (!receiver) {

cout << "Receiver not found.\n";

break;

}

int accNoTo;

cout << "Enter receiver account number: ";

cin >> accNoTo;

Account \*receiverAcc = receiver->getAccount(accNoTo);

if (!receiverAcc) {

cout << "Receiver account not found.\n";

break;

}

cout << "Enter transfer amount: ";

cin >> amount;

if (senderAcc->transfer(\*receiverAcc, amount)) {

cout << "Transfer successful!\n";

}

break;

}

case 6:

cout << "Enter customer name: ";

cin.ignore();

getline(cin, name);

customer = bank.findCustomer(name);

if (customer) {

customer->displayAccounts();

} else {

cout << "Customer not found.\n";

}

break;

case 7:

cout << "Enter customer name: ";

cin.ignore();

getline(cin, name);

customer = bank.findCustomer(name);

if (customer) {

cout << "Enter account number: ";

cin >> accNo;

Account \*acc = customer->getAccount(accNo);

if (acc) {

acc->displayTransactions();

} else {

cout << "Account not found.\n";

}

} else {

cout << "Customer not found.\n";

}

break;

default:

cout << "Invalid choice!\n";

}

}

cout << "Exiting Banking System...\n";

return 0;

}