**OOP (Object Oriented Paradigm)**

Muhammad Ahmed Hassan (4263)

Assignment No 3

Inheritance

**Qno6:** Using classes, design an online address book to keep track of the names, addresses, phone numbers, and dates of birth of family members, close friends, and certain business associates. Your program should be able to handle a maximum of 500 entries.

a. Define a class, addressType, that can store a street address, city, state, and ZIP code. Use the appropriate functions to print and store the address. Also, use constructors to automatically initialize the member variables.

b. Define a class extPersonType using the class personType (as defined in Example 12-9, Chapter 12), the class dateType (as designed in this chapter’s Programming Exercise 2), and the class addressType. Add a member variable to this class to classify the person as a family 788 | Chapter 13: Inheritance and Composition member, friend, or business associate. Also, add a member variable to store the phone number. Add (or override) the functions to print and store the appropriate information. Use constructors to automatically initialize the member variables.

c. Define the class addressBookType using the previously defined classes. An object of the type addressBookType should be able to process a maximum of 500 entries. The program should perform the following operations:

i. Load the data into the address book from a disk.

ii. Sort the address book by last name.

iii. Search for a person by last name.

iv. Print the address, phone number, and date of birth (if it exists) of a given person.

v. Print the names of the people whose birthdays are in a given month.

vi. Print the names of all of the people between two last names.

vii. Depending on the user’s request, print the names of all family members, friends, or business associates.

**Solution:**

#include <iostream>

#include <string>

#define MAX 500

using namespace std;

class addressType {

protected:

string add, city, state; //Street address

unsigned int zipCode; //zipcode

public:

addressType() : add(" "), city(" "), state(" "), zipCode(0) {}

void addSetter();

void displayAdd()const;

};

////////////Definitions of Member Functions //////////////

void addressType::addSetter() {

cout << "Enter the address: ";

getline(cin, add);

cin.ignore(0, '\n');

cout << "Enter the city name: ";

getline(cin, city);

cin.ignore(0, '\n');

cout << "Enter the name of State: ";

getline(cin, state);

cin.ignore(0, '\n');

cout << "Enter the Zip code: ";

cin >> zipCode;

cout << endl;

}

////////////////// Display ///////////////////////

void addressType::displayAdd()const {

cout << "Address:" << add << endl;

cout << "City:" << city << endl;

cout << "State:" << state << endl;

cout << "Zip Code:" << zipCode << endl;

}

///////////////////Date Type Class ////////////////

class dateType {

protected:

unsigned int dd, mm, yy;

long int ph; //dd->date, mm->month, yy->year and ph->phone no

public:

dateType() :dd(0), mm(0), yy(0), ph(0) {}

void setdate();

void getdate()const;

};

/////////////////Defintion of Member Functions //////////////////

void dateType::setdate() {

cout << "Enter the date of birth" << endl;

cout << "Enter the date: ";

cin >> dd;

cout << "Enter the month: ";

cin >> mm;

cout << "Enter the year: ";

cin >> yy;

cout << "Enter the Phone Number:" << endl;

cin >> ph;

}

void dateType::getdate()const {

cout << "The date of Birth is :" << dd << "/" << mm << "/" << yy << endl;

cout << "The Phone Number is:" << ph << endl;

}

///////////////////PersonType Class ////////////////

class personType :public addressType {

protected:

string firstN, lastN, relation;

public:

personType() {

firstN = lastN = relation = " ";

add = city = state = " ";

zipCode = 0;

}

void setname();

void getname()const;

};

/////////////////Defintion of Member Functions //////////////////

void personType::setname() {

cout << "Enter your First Name: ";

cin.ignore(20, '\n');

cin >> firstN;

cout << "Enter your Last Name: ";

cin.ignore(20, '\n');

cin >> lastN;

cout << "Enter the relation: ";

cin.ignore(20, '\n');

getline(cin, relation);

}

void personType::getname()const {

cout << "Your first name :" << firstN << endl;

cout << "Your last name:\n" << lastN << endl;

cout << "Your full name:" << firstN << " " << lastN << endl;

}

/////////////////// AddressType Class ////////////////////////

class addressBooktype :public personType, public dateType {

public:

addressBooktype() {

firstN = lastN = relation = " ";

dd = mm = yy = 0;

ph = 0;

add = city = state = " ";

zipCode = 0;

}

int search(string d, string l);

int birthsearch(int m);

};

/////////////////Defintion of Member Functions //////////////////

int addressBooktype::search(string d, string l) {

if (firstN.compare(d) == 0 && lastN.compare(l) == 0)

{

return 0;

}

else

{

return -1;

}

}

///////////////// Birth Search //////////////////

int addressBooktype::birthsearch(int m) {

if (m == mm) {

return 0;

}

else {

return -1;

}

}

int main()

{

addressBooktype f[MAX], r[MAX], m[MAX]; // f[] for freinds, r[] for relatives, m[] for family

char opt;

string d, l;

int b, found = -1, i = 0, n = 0;

do {

cout << "\t\t\t\t\t\t------------------------------ \n"

<< "\t\t\t\t\t\t| Address Book |\n"

<< "\t\t\t\t\t\t| 1. Enter Record |\n"

<< "\t\t\t\t\t\t| 2. Show records |\n"

<< "\t\t\t\t\t\t| 3. Search Record |\n"

<< "\t\t\t\t\t\t| 4. Search Birthday |\n"

<< "\t\t\t\t\t\t| 5. Exit |\n"

<< "\t\t\t\t\t\t ----------------------------- \n"

<< "Select Option: ";

cin >> opt;

switch (opt) {

case '1':

cout << " 1. Enter data in Relative book" << endl;

cout << " 2. Enter data in Bussiness Associate book." << endl;

cout << " 3. Enter data in friends book." << endl;

cin >> opt;

switch (opt) {

case '1':

if (n < MAX) {

m[i].setname();

m[i].addSetter();

m[i].setdate();

i++;

n++;

}

break;

case '2':

if (n < MAX) {

r[i].setname();

r[i].addSetter();

r[i].setdate();

n++;

i++;

}

break;

case '3':

if (n < MAX) {

f[i].addSetter();

f[i].setdate();

n++;

i++;

}

break;

}

break;

case '2':

cout << " 1. Display data in relative book" << endl;

cout << " 2. Display data in bussiness associate book." << endl;

cout << " 3. Display data in friends book." << endl;

cout << "Select Option: ";

cin >> opt;

switch (opt) {

case '1':

for (i = 0; i <= n; i++) {

m[i].getname();

m[i].displayAdd();

m[i].getdate();

}

break;

case '2':

for (i = 0; i <= n; i++) {

r[i].getname();

r[i].displayAdd();

r[i].getdate();

}

break;

case '3':

for (i = 0; i <= n; i++) {

f[i].getname();

f[i].displayAdd();

f[i].getdate();

}

break;

}

break;

case '3':

cout << " 1.Search a relative." << endl;

cout << " 2.Search a bussiness associate." << endl;

cout << " 3.Search a friend." << endl;

cout << "Select Option: ";

cin >> opt;

switch (opt) {

case '1':

cout << "Enter the first Name of relative:" << endl;

cin.ignore(20, '\n');

getline(cin, d);

cout << "Enter the last of family relative: " << endl;

cin.ignore(20, '\n');

getline(cin, l);

for (i = 0; i < n; i++) {

found = m[i].search(d, l);

if (found == 0) {

m[i].getname();

m[i].getdate();

m[i].displayAdd();

}

}

break;

case '2':

cout << "Enter the first Name of bussiness associates:" << endl;

cin.ignore(20, '\n');

getline(cin, d);

cout << "Enter the last of bussiness associate: " << endl;

cin.ignore(20, '\n');

getline(cin, l);

for (int i = 0; i < n; i++) {

found = r[i].search(d, l);

if (found == 0) {

r[i].getname();

r[i].getdate();

r[i].displayAdd();

}

}

break;

case '3':

cout << "Enter the first Name of friend:" << endl;

cin.ignore(20, '\n');

getline(cin, d);

cout << "Enter the last of friend: " << endl;

cin.ignore(20, '\n');

getline(cin, l);

for (int i = 0; i < n; i++) {

found = f[i].search(d, l);

if (found == 0) {

f[i].getname();

f[i].getdate();

f[i].displayAdd();

}

}

break;

}

break;

case '4':

cout << " 1. Search Birthday of relative." << endl;

cout << " 2. Search a birthday of associate." << endl;

cout << " 3. Search a birthday of friend." << endl;

cin >> opt;

switch (opt) {

case '1':

cout << "Enter the month of birthday of your relative (1-12):" << endl;

cin >> b;

for (i = 0; i < n; i++) {

found = m[i].birthsearch(b);

if (found == 0) {

m[i].getname();

m[i].getdate();

}

}

break;

case '2':

cout << "Enter the month of birthday of your bussiness associate (1-12):" << endl;

cin >> b;

for (int i = 0; i < n; i++) {

found = r[i].birthsearch(b);

if (found == 0) {

r[i].getname();

r[i].getdate();

r[i].displayAdd();

}

}

break;

case '3':

cout << "Enter the month of birthday of your friend (1-12):" << endl;

cin >> b;

for (int i = 0; i < n; i++) {

found = f[i].search(d, l);

if (found == 0) {

f[i].getname();

f[i].getdate();

}

}

break;

}

break;

case '5':

exit(0);

}

} while (true);

}

**Output:**

****

**Qno12:** In this exercise, you will design various classes and write a program to computerize the billing system of a hospital.

a. Design the class doctorType, inherited from the class personType, defined in Chapter 12, with an additional data member to store a doctor’s speciality. Add appropriate constructors and member functions to initialize, access, and manipulate the data members.

b. Design the class billType with data members to store a patient’s ID and a patient’s hospital charges, such as pharmacy charges for medicine, doctor’s fee, and room charges. Add appropriate constructors and member functions to initialize and access and manipulate the data members.

c. Design the class patientType, inherited from the class personType, defined in Chapter 12, with additional data members to store a patient’s ID, age, date of birth, attending physician’s name, the date when the patient was admitted in the hospital, and the date when the patient was discharged from the hospital. (Use the class dateType to store the date of birth, admit date, discharge date, and the class doctorType to store the attending physician’s name.) Add appropriate constructors and member functions to initialize, access, and manipulate the data members. Write a program to test your classes.

**Solution:**

#include<iostream>

#include<string>

using namespace std;

class PersonType

{

string FirstName, LastName;

public:

void SetName();

void getName() const;

};

void PersonType::getName() const

{

cout << FirstName << " " << LastName << endl;

}

void PersonType::SetName()

{

cout << "Please Enter Your FIRST Name.\n";

cin >> FirstName;

cout << "Please Enter Your LAST Name.\n";

cin >> LastName;

}

class DoctorType : public PersonType

{

public:

void SetSpeciality();

void getSpeciality() const;

private:

string FirstName, LastName, Speciality;

};

void DoctorType::SetSpeciality()

{

SetName();

cout << "Please Enter Your SPECIALITY.\n";

cin >> Speciality;

}

void DoctorType::getSpeciality() const

{

getName();

cout << "Your speciality :" << Speciality << endl;

}

class DateType

{

public:

void getDate() const;

void SetDate();

private:

int TheDay, TheYear, TheMonth;

};

void DateType::SetDate()

{

cout << "Please Enter The Day (DD).\n";

cin >> TheDay;

cout << "Please Enter The Month (MM).\n";

cin >> TheMonth;

cout << "Please Enter The Year (YYYY).\n";

cin >> TheYear;

}

void DateType::getDate() const

{

cout << TheDay << " " << TheMonth << " " << TheYear << endl;

}

class DateOfBirthType : public DateType

{

public:

void printDOB() const;

void SetDOB();

private:

int TheDay, TheYear, TheMonth;

};

void DateOfBirthType::printDOB() const

{

cout << "Patients Date Of Birth: " << TheDay << "/" << TheMonth << "/" <<

TheYear << endl;

}

void DateOfBirthType::SetDOB()

{

SetDate();

}

class AdmittanceDateType : public DateType

{

public:

void printAdmittanceDate() const;

void SetAdmittanceDate();

private:

int TheDay, TheYear, TheMonth;

};

void AdmittanceDateType::printAdmittanceDate() const

{

cout << "The Patients Admittance Date: " << TheDay << " " << TheMonth << " "

<< TheYear << endl;

}

void AdmittanceDateType::SetAdmittanceDate()

{

getDate();

}

class DischargeDateType : public DateType

{

public:

void printDischargeDate() const;

void SetDischargeDate();

private:

int TheDay, TheYear, TheMonth;

};

void DischargeDateType::printDischargeDate() const

{

cout << "The Patients Discharge Date: " << TheDay << " " << TheMonth << " "

<< TheYear << endl;

}

void DischargeDateType::SetDischargeDate()

{

getDate();

}

class PatientType : public PersonType

{

public:

void Disp() const;

void SetPatient();

private:

int PatientID, PatientAge;

};

void PatientType::SetPatient() {

SetName();

cout << "Enter Patient ID" << endl;

cin >> PatientID;

cout << "Enter Patient Age" << endl;

cin >> PatientAge;

}

void PatientType::Disp()const {

cout << "Patient Name:" << endl;

getName();

cout << "Patient ID:" << PatientID << endl;

cout << "Patient Age:" << PatientAge << endl;

}

class billType

{

private:

int PtID;

double MedChg;

double DrF;

double RmChg;

public:

billType(int pid = 0, double mchg = 0, double df = 0, double rc = 0) {

PtID = pid;

MedChg = mchg;

DrF = df;

RmChg = rc;

}

void setChg();

void getChg() const;

};

void billType::setChg() {

cout << "Enter your patient ID" << endl;

cin >> PtID;

cout << "Enter hospital fee" << endl;

cin >> MedChg;

cout << "Enter pharmacy fee" << endl;

cin >> DrF;

cout << "Enter room fee" << endl;

cin >> RmChg;

}

void billType::getChg() const {

cout << "Patient ID" << PtID << endl;

cout << "Hospital Charges" << MedChg << endl;

cout << "Doctor's Fee" << DrF << endl;

cout << "Room charges" << RmChg << endl;

int total = PtID + MedChg + DrF + RmChg;

cout << "Total fee =" << total << endl;

}

int main() {

DoctorType d1;

PatientType p1;

DateOfBirthType b1{};

AdmittanceDateType o1{};

DischargeDateType c1{};

int choice;

do {

cout << "Who would you like to input information for?\n";

cout << " 1 - Doctor\n";

cout << " 2 - Patient\n";

cin >> choice;

switch (choice)

{

case 1:

do {

cout << "Who would you like to input information for?\n";

cout << " 1 - set info\n";

cout << " 2 - view info\n";

cout << " 3 - Add Patient charges\n";

cout << " 4 - Exit \n";

cin >> choice;

switch (choice)

{

case 1:

d1.SetSpeciality();

break;

case 2:

d1.getSpeciality();

break;

case 3:

case 4:

exit(0);

break;

}

} while (true);

break;

case 2:

do {

cout << "Who would you like to input information for?\n";

cout << " 1 - set info\n";

cout << " 2 - view info\n";

cout << " 3 - Exit \n";

cin >> choice;

switch (choice)

{

case 1:

p1.SetPatient();

cout << "Enter date of birth" << endl;

b1.SetDate();

cout << "Enter date of Admission" << endl;

o1.SetDate();

cout << "Enter date of Discharge" << endl;

c1.SetDate();

break;

case 2:

p1.Disp();

cout << "Date of birth" << endl;

b1.getDate();

cout << "Date of Admission" << endl;

o1.getDate();

cout << "Date of Discharge" << endl;

c1.getDate();

break;

case 3:

exit(0);

break;

}

} while (true);

}

} while (true);

}

**Output:**

Text

Description automatically generated

**Qno14:**

a. 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. 1 3 Programming Exercises | 791

b. Every bank offers a checking account. Derive the class checkingAccount 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 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.

c. 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.

d. Write a program to test your classes designed in parts (b) and (c).

**Solution:**

#include <iostream>

using namespace std;

class BankAccount

{

protected:

int accountNumber;

float intrst;

double Balance;

public:

BankAccount(int Accno, int intst, double Bal)

{

accountNumber = Accno;

intrst = intst;

Balance = Bal;

}

void Setacc();

void Dispacc();

double depAmnt(double a);

double wdrawAmnt(double a);

};

void BankAccount::Setacc()

{

cout << "Enter your Acc no" << endl;

cin >> accountNumber;

cout << "Enter your Balance" << endl;

cin >> Balance;

cout << "Enter your Interest rate" << endl;

cin >> intrst;

}

void BankAccount::Dispacc() {

cout << " ---------------------------------" << endl;

cout << " || Account Info " << endl;

cout << " ---------------------------------" << endl;

cout << " || Account Number :" << accountNumber << endl;

cout << " || Account Balance:" << Balance << endl;

cout << " || Interest Rate :" << intrst << endl;

cout << " ---------------------------------" << endl;

cout << endl;

}

double BankAccount::depAmnt(double dAmnt)

{

Balance = Balance + dAmnt;

return(Balance);

}

double BankAccount::wdrawAmnt(double wAmnt)

{

if (wAmnt <= Balance)

Balance = Balance - wAmnt;

else

cout << "You have insufficient funds!" << endl;

return(Balance);

}

class CheckingAccount : public BankAccount

{

double minBal;

public:

CheckingAccount(int a, int b, double c) :BankAccount(a, b, c)

{

minBal = 500; //by default

}

void showAcc();

};

void CheckingAccount::showAcc() {

if (Balance < 500)

{

Balance = Balance - 1;

}

Dispacc();

}

class SavingAccount :public BankAccount

{

double minBal;

public:

SavingAccount(int a, int b, double c) :BankAccount(a, b, c)

{

minBal = 500;

}

void showAcc();

void postintst();

};

void SavingAccount::showAcc() {

Dispacc();

}

void SavingAccount::postintst() {

cout << "Your current Interest Rate is:" << intrst << endl;

cout << "You can only post interest After every one week" << endl;

cout << "Enter the new interest rate" << endl;

cin >> intrst;

}

int main()

{

CheckingAccount acc1(0, 0, 500);

SavingAccount acc2(0, 0, 500);

double dAmnt;

double amount;

int choice;

do

{

cout << endl;

cout << " Create an Account " << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \*\* 1 - Checking Account \*\*" << endl;

cout << " \*\* 2 - Saving Account \*\*" << endl;

cout << " \*\* 3 - Exit \*\*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cin >> choice;

cout << endl;

if (choice == 1)

{

acc1.Setacc();

do

{

cout << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \*\* 1 - Deposit \*\*" << endl;

cout << " \*\* 2 - Withdraw \*\*" << endl;

cout << " \*\* 3 - Account Info \*\*" << endl;

cout << " \*\* 4 - Exit \*\*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cin >> choice;

cout << endl;

switch (choice)

{

case 1:

cout << "Enter amount: " << endl;

cin >> amount;

dAmnt = acc1.depAmnt(amount);

cout << "Available balance: " << dAmnt <<

endl;

break;

case 2:

cout << "Enter amount: " << endl;

cin >> amount;

dAmnt = acc1.wdrawAmnt(amount);

cout << "Available balance: " << dAmnt <<

endl;

break;

case 3:

acc1.showAcc();

break;

}

} while (choice != 4);

}

if (choice == 2)

{

acc2.Setacc();

do

{

cout << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \*\* 1 - Deposit \*\*" << endl;

cout << " \*\* 2 - Withdraw \*\*" << endl;

cout << " \*\* 3 - Interest \*\*" << endl;

cout << " \*\* 4 - Account Info \*\*" << endl;

cout << " \*\* 5 - Exit \*\*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cin >> choice;

cout << endl;

switch (choice)

{

case 1:

cout << "Enter amount: " << endl;

cin >> amount;

dAmnt = acc2.depAmnt(amount);

cout << "Available balance: " << dAmnt <<

endl;

break;

case 2:

cout << "Enter amount: " << endl;

cin >> amount;

dAmnt = acc2.wdrawAmnt(amount);

cout << "Available balance: " << dAmnt <<

endl;

break;

case 3:

acc2.postintst();

break;

case 4:

acc2.showAcc();

break;

}

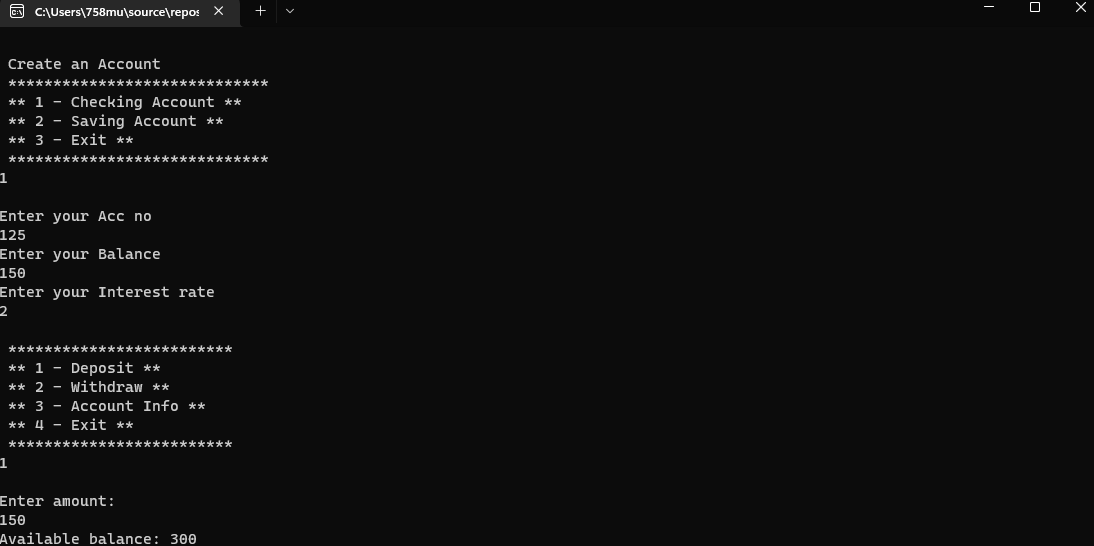
} while (choice != 5);

}

} while (choice != 3);

}

**Output :**

****