OOPs LAB 04

NAME: Abdul Rahim Dawra RollNo:24K-0633  
Task 01

#include <iostream>

#include <string>

using namespace std;

class MobileAccount {

private:

double balance;

string phoneNumber;

public:

MobileAccount() : balance(50.0), phoneNumber("") {}

MobileAccount(const string& phone, double bal) : balance(bal), phoneNumber(phone) {}

void display() const {

cout << "MobileAccount - Phone: " << phoneNumber << ", Balance: " << balance << " PKR\n";

}

};

int main() {

MobileAccount m1;

MobileAccount m2("03001234567", 200.0);

m1.display();

m2.display();

}

  
Task 02

#include <iostream>

#include <string>

using namespace std;

class NUCES\_Student {

private:

int id;

string name;

public:

NUCES\_Student() : id(0), name("Not Registered") {}

NUCES\_Student(int id) : id(id), name("Name Pending") {}

NUCES\_Student(int id, const string& name) {

this->id = id;

this->name = name;

}

void display() const {

cout << "NUCES\_Student - ID: " << id << ", Name: " << name << '\n';

}

};

int main(){

NUCES\_Student s1;

NUCES\_Student s2(12345);

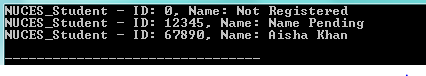
NUCES\_Student s3(67890, "Aisha Khan");

s1.display();

s2.display();

s3.display();

}

  
  
Task 03

#include <iostream>

#include <string>

using namespace std;

class PropertyDeed {

private:

int\* ownerID;

public:

PropertyDeed(int id = 0) {

ownerID = new int;

\*ownerID = id;

}

PropertyDeed(const PropertyDeed &other, bool shallow) {

if (shallow) {

ownerID = other.ownerID;

} else {

ownerID = new int;

\*ownerID = \*(other.ownerID);

}

}

PropertyDeed(const PropertyDeed &other) {

ownerID = new int;

\*ownerID = \*(other.ownerID);

}

void setOwnerID(int id) {

if (ownerID) \*ownerID = id;

}

void display() const {

cout << "PropertyDeed - ownerID value: " << \*ownerID

<< ", address: " << static\_cast<const void\*>(ownerID) << '\n';

}

~PropertyDeed() {}

void freeOwner() {

if (ownerID) {

delete ownerID;

ownerID = nullptr;

}

}

int\* getPointer() const { return ownerID; }

};

int main(){

PropertyDeed original(555);

cout << "Original: ";

original.display();

PropertyDeed shallowCopy(original, true);

cout << "Shallow copy (after creation): ";

shallowCopy.display();

PropertyDeed deepCopy(original);

cout << "Deep copy (after creation): ";

deepCopy.display();

cout << "-- Change original's ownerID to 999 --\n";

original.setOwnerID(999);

cout << "Original: "; original.display();

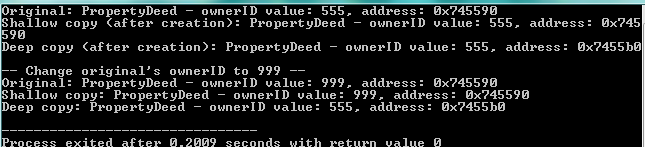
cout << "Shallow copy: "; shallowCopy.display();

cout << "Deep copy: "; deepCopy.display();

deepCopy.freeOwner();

original.freeOwner();

}

  
  
Task 04

#include <iostream>

#include <string>

using namespace std;

class ServerSession {

private:

int sessionID;

public:

ServerSession(int id) : sessionID(id) {

cout << "ServerSession " << sessionID << " started. Resource acquired.\n";

}

~ServerSession() {

cout << "ServerSession " << sessionID << " ending. Resource released.\n";

}

};

int main(){

{

ServerSession sess1(101);

}

}  
  
Task 05

#include <iostream>

#include <string>

using namespace std;

class CurrencyExchange {

private:

const double PKR\_to\_USD\_Rate;

string &branchName;

int totalTransactions;

public:

CurrencyExchange(double rate, string &branch, int transactions)

: PKR\_to\_USD\_Rate(rate), branchName(branch), totalTransactions(transactions) {}

double convertPKRtoUSD(double pkr) const {

return pkr \* PKR\_to\_USD\_Rate;

}

void display() const {

cout << "CurrencyExchange - Branch: " << branchName

<< ", Rate: " << PKR\_to\_USD\_Rate

<< ", TotalTransactions: " << totalTransactions << '\n';

}

};

int main(){

string branch = "Karachi Main Branch";

CurrencyExchange ex(0.0036, branch, 1200);

ex.display();

cout << "Convert 10000 PKR -> " << ex.convertPKRtoUSD(10000) << " USD";

}  
  
  
Task 06

#include <iostream>

#include <string>

using namespace std;

class OrderDetails {

private:

int orderID;

string itemName;

const double deliveryFee;

public:

OrderDetails(int orderID, const string &itemName, double deliveryFee)

: deliveryFee(deliveryFee) {

this->orderID = orderID;

this->itemName = itemName;

}

~OrderDetails() {

cout << "OrderDetails - Order " << orderID << " Archived and memory cleared.\n";

}

void display() const {

cout << "OrderDetails - ID: " << orderID << ", Item: " << itemName << ", DeliveryFee: " << deliveryFee << '\n';

}

};

int main(){

OrderDetails order(4001, "Wireless Mouse", 250.0);

order.display();

}  
  
  
Task 07

#include <iostream>

#include <string>

using namespace std;

class SecureBankAccount {

private:

int accountNumber;

double balance;

const string accountType;

int\* logHistory;

public:

SecureBankAccount(int acctNum, double initialBalance, const string &acctType, int logValue)

: accountType(acctType) {

this->accountNumber = acctNum;

this->balance = initialBalance;

logHistory = new int;

\*logHistory = logValue;

if (initialBalance < 500.0) {

cout << "Error: Initial balance (" << initialBalance << " PKR) is less than minimum (500 PKR).\n";

}

}

SecureBankAccount(const SecureBankAccount &other)

: accountType(other.accountType) {

this->accountNumber = other.accountNumber;

this->balance = other.balance;

logHistory = new int;

\*logHistory = \*(other.logHistory);

}

~SecureBankAccount() {

delete logHistory;

}

void display() const {

cout << "SecureBankAccount - Account#: " << accountNumber << ", Balance: " << balance << ", Type: " << accountType << ", LogHistory value: " << \*logHistory << ", LogHistory addr: " << static\_cast<const void\*>(logHistory) << '\n';

}

void setLogValue(int v) { if (logHistory) \*logHistory = v; }

};

int main(){

SecureBankAccount acct1(9001, 1000.0, "Current", 5);

acct1.display();

cout << "Create deep copy of acct1 into acct2 and change acct1's log value to 99\n";

SecureBankAccount acct2 = acct1;

acct1.setLogValue(99);

cout << "acct1: "; acct1.display();

cout << "acct2 (deep copy): "; acct2.display();

return 0;

}

