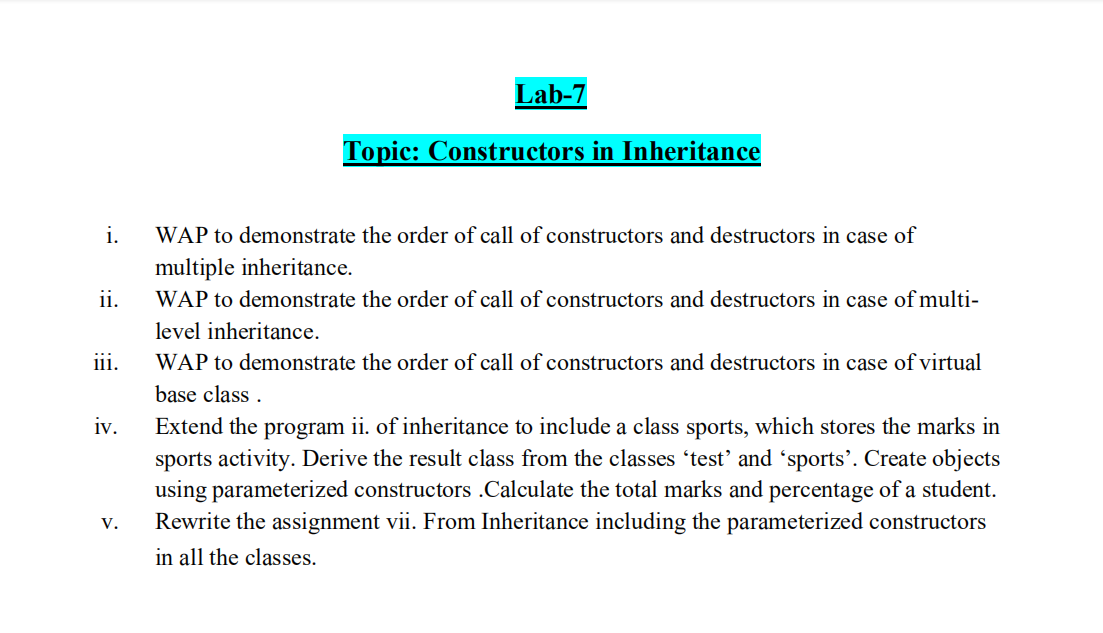
## **CHAUDHARY HAMDAN**

**1905387**

**OOP LAB-7**

**Date : 11-09-2020**



1.

#include<iostream>

using namespace std;

class A

{

protected:

int a;

public:

A()

{

cout << "Constructor of Base Class A Called " << endl;

}

~A()

{

cout << "Destructor of Base Class A Called " << endl;

}

};

class A1

{

protected:

int a1;

public:

A1()

{

cout << "Constructor of Base Class A1 Called " << endl;

}

~A1()

{

cout << "Destructor of Base Class A1 Called " << endl;

}

};

class E : public A, public A1

{

protected:

int e;

public:

E()

{

cout << "Constructor of Derived Class E Called " << endl;

}

~E()

{

cout << "Destructor of Derived Class E Called " << endl;

}

};

int main()

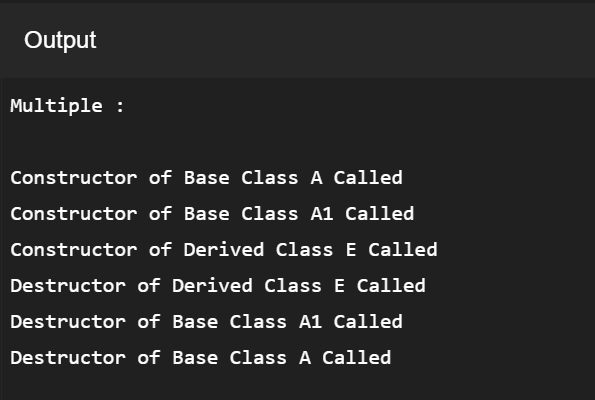
{

cout << "Multiple : \n" << endl;

E obe;

return 0;

}



2.

#include<iostream>

using namespace std;

class A

{

protected:

int a;

public:

A()

{

cout << "Constructor of Grandparent Class A Called " << endl;

}

~A()

{

cout << "Destructor of Grandparent Class A Called " << endl;

}

};

class B : public A

{

protected:

int b;

public:

B()

{

cout << "Constructor of Parent Class B Called " << endl;

}

~B()

{

cout << "Destructor of Parent Class B Called " << endl;

}

};

class D : public B

{

protected:

int d;

public:

D()

{

cout << "Constructor of Child Class D Called " << endl;

}

~D()

{

cout << "Destructor of Child Class D Called " << endl;

}

};

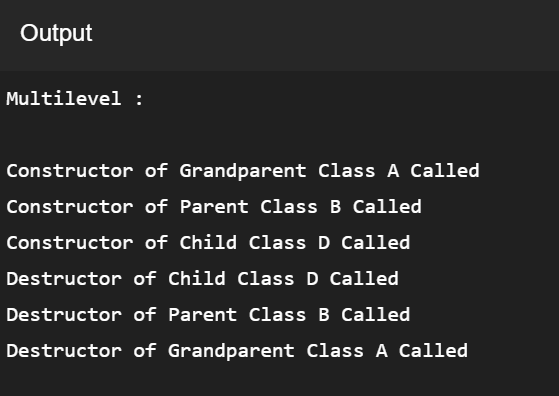
int main()

{

cout << "Multilevel : \n" << endl;

D obd;

}



3.

#include <iostream>

using namespace std;

class A

{

public:

A()

{

cout << "Constructor of Grandparent Class A Called " << endl;

}

~A()

{

cout << "Destructor of Grandparent Class A Called " << endl;

}

};

class B : public virtual A

{

public:

B()

{

cout << "Constructor of Parent Class B Called " << endl;

}

~B()

{

cout << "Destructor of Parent Class B Called " << endl;

}

};

class C : virtual public A

{

public:

C()

{

cout << "Constructor of Parent Class C Called " << endl;

}

~C()

{

cout << "Destructor of Parent Class C Called " << endl;

}

};

class D : public B, public C

{

public:

D()

{

cout << "Constructor of Child Class D Called " << endl;

}

~D()

{

cout << "Destructor of Child Class D Called " << endl;

}

};

int main()

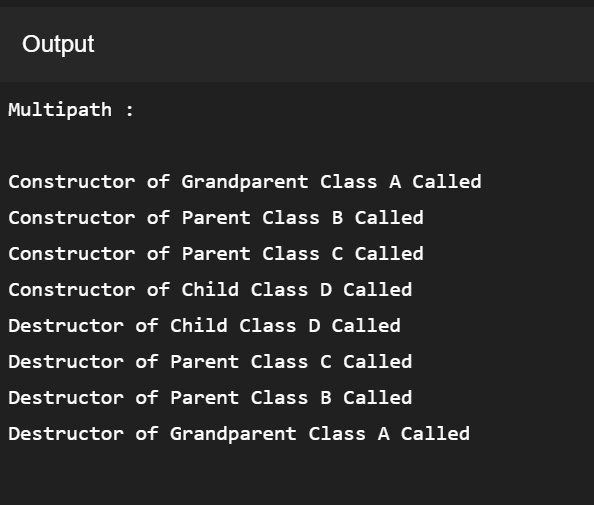
{

cout << "Multipath :\n" << endl;

D obj;

return 0;

}



4.

#include<iostream>

using namespace std;

class student

{

protected:

char name[20];

int roll;

public:

void getdata()

{

cout << "Enter roll and name " << endl;

cin >> roll >> name;

}

};

class test : public virtual student

{

protected:

int sub1;

int sub2;

int sub3;

int sub4;

int sub5;

public:

void getmark()

{

cout << "Enter 5 subjects marks : " << endl;

cin >> sub1 >> sub2>> sub3 >> sub4 >> sub5;

}

void details()

{

cout << "\n\nName : " << name << " Roll number : " << roll << endl;

cout << "Marks in 5 subjects : " << sub1 << ", " << sub2 << ", " << sub3 << ", " << sub4 << ", " << sub5 << endl;

}

};

class sports : public virtual student

{

protected:

int msports;

public:

void getspo()

{

cout << "Enter marks in sports : ";

cin >> msports;

}

};

class result : public sports, public test

{

int total;

float percent;

public:

void display()

{

cout << "Marks in sports = " << msports << endl;

total = sub1+sub2+sub3+sub4+sub5+msports;

percent = (total\*100)/60;

cout << "Total marks : " << total << " Percent = " << percent << endl;

}

};

int main()

{

result ob1;

ob1.getdata();

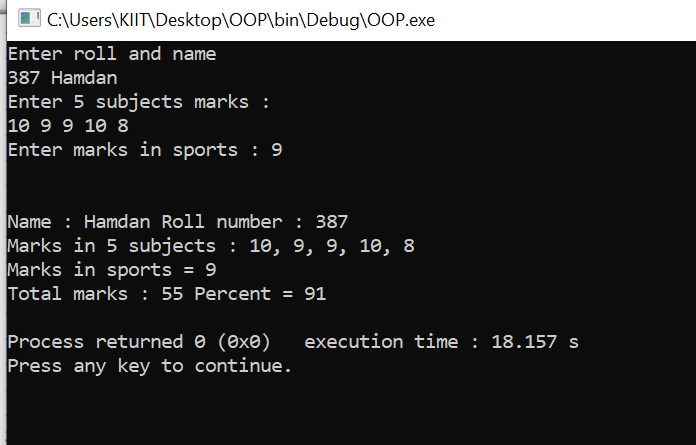
ob1.getmark();

ob1.getspo();

ob1.details();

ob1.display();

}



5.

#include<iostream>

using namespace std;

class Account

{

protected:

int custno;

string custname;

int balance;

public:

Account(int cno,string cname,int bal)

{

custno = cno;

custname = cname;

balance = bal;

}

};

class Savings : public Account

{

protected:

int minbalance;

public:

Savings(int minbal, int cno, string cna, int bal): Account(cno,cna,bal)

{

minbalance = minbal;

}

void depositSavings()

{

int dep;

cout << "Enter Amount to deposit in Savings Account : ";

cin >> dep;

balance += dep;

}

void withdraw()

{

int with;

cout << "Enter Amount you want to Withdraw from Savings Account : ";

cin >> with;

if(balance-with < minbalance)

cout << "Amount can't be withdrawn as you will not be left with minimum balance ... " << endl;

else

{

balance -= with;

cout << "Amount Withdrawn Successfully... Collect your Cash... \nRemaining Balance = " << balance << endl;

}

}

void Display()

{

cout << "\nSavings Account :--\nCustomer number = " << custno << " Name = " << custname << " Balance Remaining = " << balance << endl;

}

};

class Current : public Account

{

protected:

int overdue;

public:

Current(int ovdue, int cno, string cna, int bal): Account(cno,cna,bal)

{

Account(387,"Hamdan",1000);

overdue = ovdue;

}

void depositCurrent()

{

int dep;

cout << "Enter Amount to deposit in Savings Account : ";

cin >> dep;

balance += dep;

}

void withdraw()

{

int with;

cout << "Enter Amount you want to Withdraw from Current Account : ";

cin >> with;

if(balance-with < overdue)

cout << "Amount can't be withdrawn as you will not be left with Over-due amount ... " << endl;

else

{

balance -= with;

cout << "Amount Withdrawn Successfully... Collect your Cash... \nRemaining Balance = " << balance << endl;

}

}

void Display()

{

cout << "\nCurrent Account :--\nCustomer number = " << custno << " Name = " << custname << " Balance Remaining = " << balance << endl;

}

};

int main()

{

int cno, bal;

string nam;

cout << "Enter Customer number, Name and Balance : ";

cin >> cno >> nam >> bal;

Savings obs(500,cno,nam,bal);

Current obc(500,cno,nam,bal);

obs.depositSavings();

obs.withdraw();

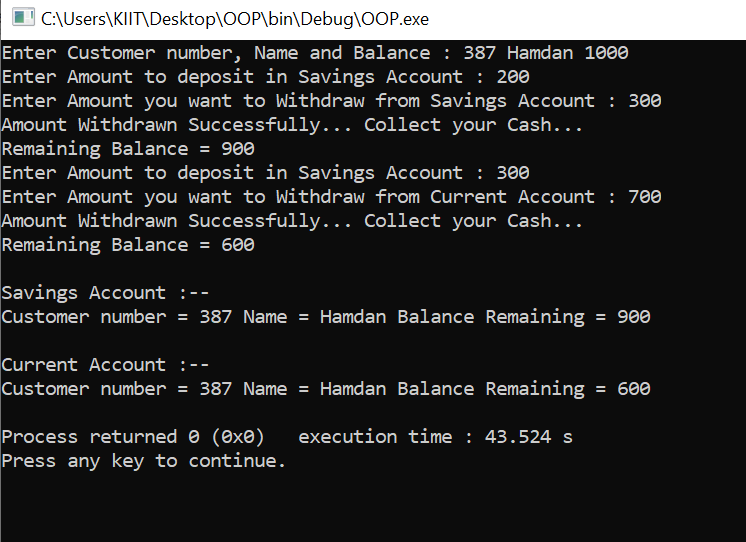
obc.depositCurrent();

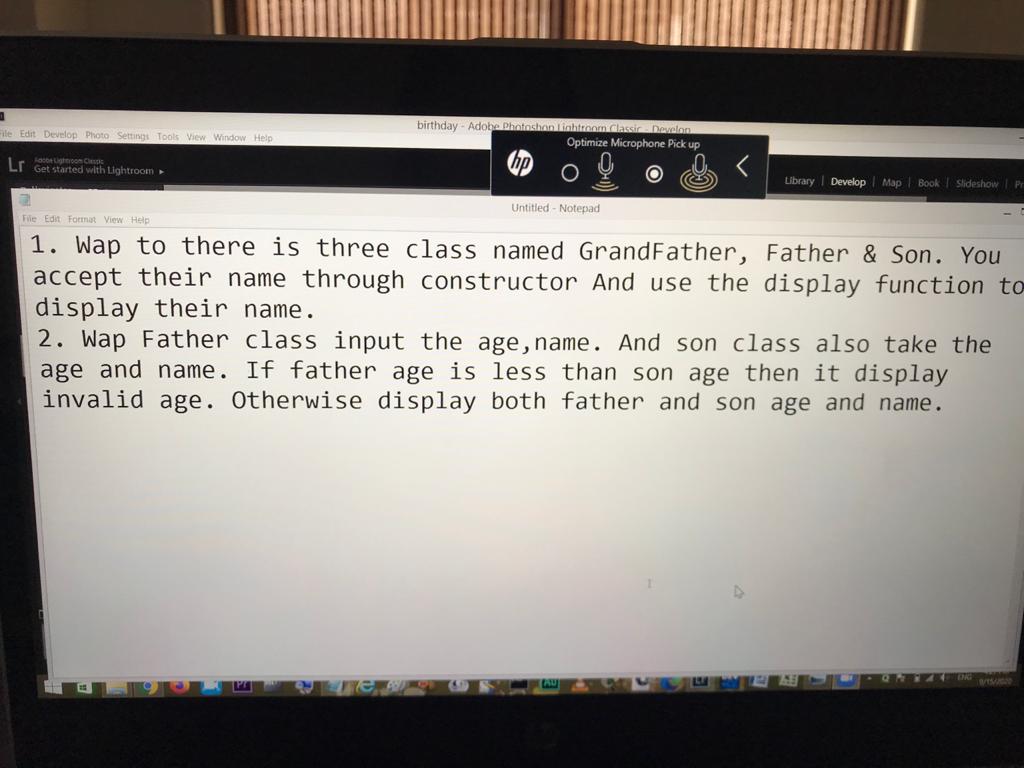
obc.withdraw();

obs.Display();

obc.Display();

}







#include<iostream>

using namespace std;

class grandfather

{

string name;

public:

grandfather()

{

cout << "Enter name of Grandfather : " ;

cin >> name;

}

void display()

{

cout << "\n\nGrandfather's name : " << name << endl;

}

};

class father : public grandfather

{

string name;

public:

father() : grandfather()

{

cout << "Enter name of Father : " ;

cin >> name;

}

void display()

{

cout << "Father's name : " << name << endl;

}

};

class son : public father

{

string name;

public:

son() : father()

{

cout << "Enter name of Son : " ;

cin >> name;

}

void display()

{

cout << "Son's name : " << name << endl;

}

};

int main()

{

son ob;

ob.father :: grandfather :: display();

ob.father :: display();

ob. display();

return 0;

}

OUTPUT :

Enter name of Grandfather : Siraj

Enter name of Father : Hammad

Enter name of Son : Hamdan

Grandfather's name : Siraj

Father's name : Hammad

Son's name : Hamdan

2.

#include<iostream>

using namespace std;

class father

{

public :

int age;

string name;

father()

{

cout << "Enter Father's name and age : ";

cin >> name >> age;

}

};

class son : public father

{

public :

int age;

string name;

son()

{

cout << "Enter Son's name and age : ";

cin >> name >> age;

}

};

int main()

{

cout << "\nINPUT : \n" << endl;

son ob;

cout << "\n\nOUTPUT : \n" << endl;

if(ob.age >= ob.father :: age)

{

cout << "Invalid age entered..." << endl;

return 0;

}

cout << "Father's Name : " << ob.father :: name << endl;

cout << "Father's Age : " << ob.father :: age << endl;

cout << "Son's Name : " << ob.name << endl;

cout << "Son's Age : " << ob.age << endl;

return 0;

}

OUTPUT :

INPUT :

Enter Father's name and age : Hammad 50

Enter Son's name and age : Hamdan 20

OUTPUT :

Father's Name : Hammad

Father's Age : 50

Son's Name : Hamdan

Son's Age : 20