**EVEREST ENGINEERING COLLEGE**

**SANEPA, LALITPUR**



(AFFILIATED TO POKHARA UNIVERSITY)

AN

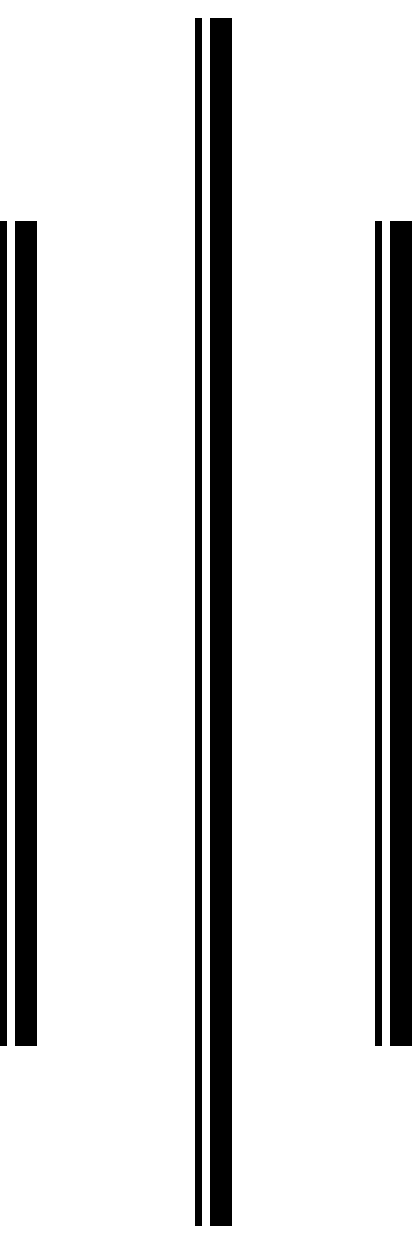
LAB REPORT

ON

**Object Oriented Programming In C++**

***[Object Inheritance & Reusability]***

**Lab Sheet: 7**



**SUBMITTED BY SUBMITTED TO**

Name:-**SAKCHYAM ACHARYA** Er.Pradip Paudel

Roll No:-40 Everest Engineering College

Department of Computer Engineering

Faculty:-BE CMP

Year/Semester: - 2nd SEM

Batch: 2021

Verified By:-

Student's Signature

**Index Page:-**

Topics Page No

* Title, Objective....................................................................................3
* Theory……………………………………………………………………………….2-3

• Introduction to inheritance & its type

• Ambiguity in inheritance

• Virtual Base Class

* Composition
* Questions with Source Code & Output…………………………………………...3-25
* Discussion & Conclusion ...................................................................................25

**Title:** ***Object Inheritance & Reusability*.**

**Objective:**

• To be familiar with inheritance & composition.

• To understand how inheritance support reusability.

* To understand about ambiguity in inheritance & virtual base class.
* **Theory:**

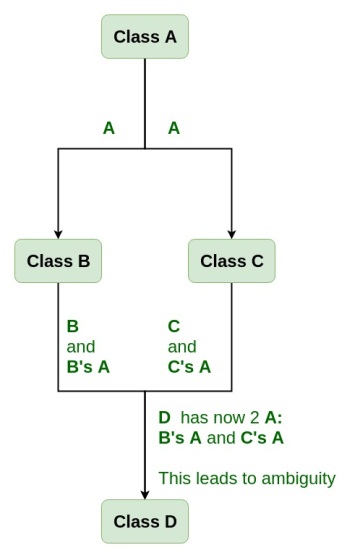
• **Introduction to inheritance & its type:-** Inheritance is **a feature or a process in which, new classes are created from the existing classes**. The new class created is called “derived class” or “child class” and the existing class is known as the “base class” or “parent class”. The derived class now is said to be inherited from the base class.

**5 Types of Inheritance in C++**

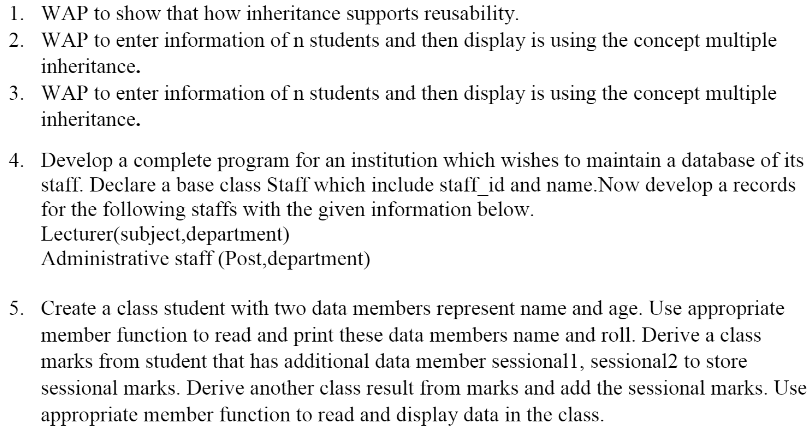
* Single Inheritance.
* Multiple Inheritance.
* Multilevel Inheritance.
* Hierarchical Inheritance.
* Hybrid Inheritance

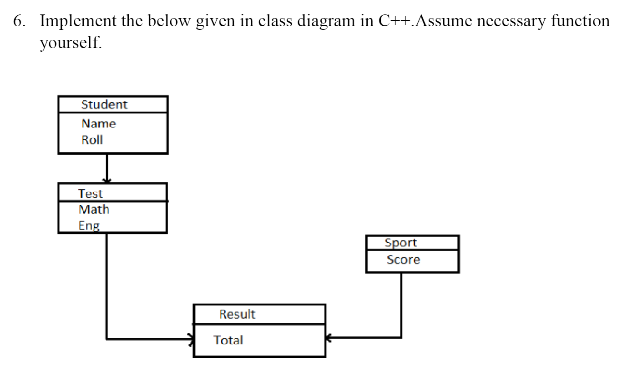
• **Ambiguity in inheritance:-** Ambiguity in inheritance can be defined as **when one class is derived for two or more base classes then there are chances that the base classes have functions with the same name**. So it will confuse derived class to choose from similar name functions.

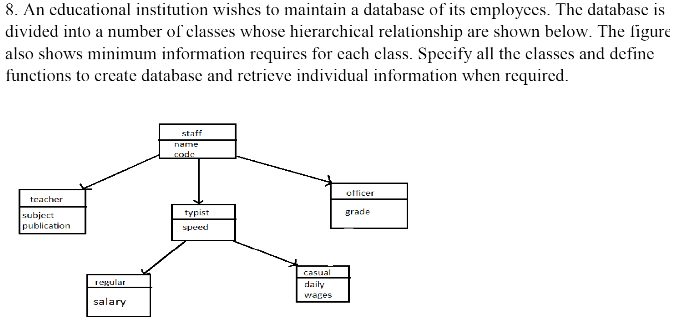
• **Virtual Base Class:-** Virtual base classes are **used in virtual inheritance in a way of preventing multiple “instances” of a given class appearing in an inheritance hierarchy when using multiple inheritances**. Need for Virtual Base Classes: Consider the situation where we have one class A.

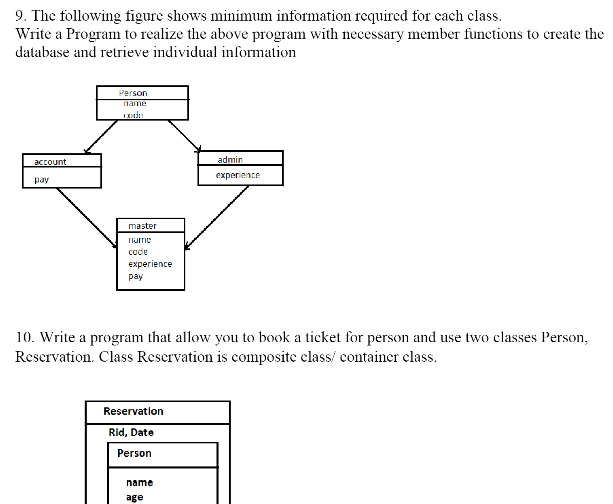


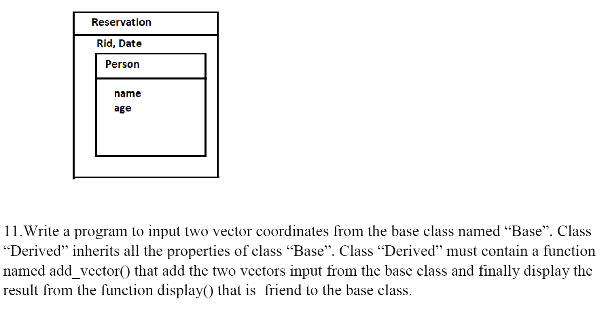
* **Composition:-** Composition in C++ is also referred to as **object composition**. The object composition concept work on the model of has-a relationship among two different objects. Complex objects are often referred to as parent components while objects which are simpler or smaller are often referred to as child components.
* **Lab Problems:**











**///Prob1)**

**//Source Code:**

//Inheritence Support Resuability of Code

//Using Multiple Inheritence

//Calculator Banaidim haitw hybrid...haha

#include <iostream>

#include <cmath>

using namespace std;

class SimpleCalculator

{

int a,b;

public:

void get\_dataSimple()

{

cout<<"Enter the value of a:"<< endl;

cin>>a;

cout<<"Enter the value of b:"<< endl;

cin>>b;

}

void performOpertationsSimple()

{

cout << "The value of a + b =: " << a + b << endl;

cout << "The value of a - b =: " << a - b << endl;

cout << "The value of a \* b =: " << a \* b << endl;

cout << "The value of a / b =: " << a / b << endl;

}

};

class ScientificCalculator{

int a,b;

public:

void get\_dataScientific()

{

cout<<"Enter the value of a:"<<endl;

cin>>a;

cout<<"Enter the value of b:"<<endl;

cin>>b;

}

void performOpertationsScientific()

{

cout << "The value of cos(a) =" << cos(a) << endl;

cout << "The value of sin(a) =" << sin(a) << endl;

cout << "The value of exp(a) =" << exp(a) << endl;

cout << "The value of tan(a) =" << tan(a) << endl;

cout << "The value of cos(b) =" << cos(b) << endl;

cout << "The value of sin(b) =" << sin(b) << endl;

cout << "The value of exp(b) =" << exp(b) << endl;

cout << "The value of tan(b) =" << tan(b) << endl;

}

};

class HybridCalculator : public SimpleCalculator, public ScientificCalculator

{

};

int main(){

cout<<"\t\t\tSAK's Hybrid Calculator"<<endl;

HybridCalculator calc;

calc.get\_dataScientific();

calc.performOpertationsScientific();

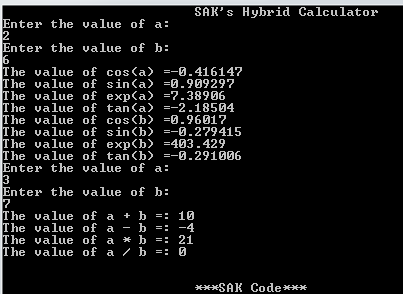
calc.get\_dataSimple();

calc.performOpertationsSimple();

cout<<"\n\n\t\t\t\*\*\*SAK Code\*\*\*";

return 0;

}



**///Prob2)**

**//Source Code:**

//Multilevel Inheritence

#include<iostream>

using namespace std;

class A{

protected:

int idno,rollno;

char name[20];

public:

void inputdata(){

//cout<<"\t\tEnter the details of the student:"<<endl;

cout<<"\nName?:";

//cin.ignore();

cin>>name;

cout<<"Roll NO?:";

cin>>rollno;

cout<<"ID No?:";

cin>>idno;

}

void display(){

cout<<"\nName="<<name<<"\nRollNo="<<rollno<<"\nIDNo="<<idno<<endl;

}

};

class B:public A{

protected:

char semester[20],program[20];

public:

void inputdata(){

//cin.ignore();

cout<<"Program?:";

cin>>program;

//cin.ignore();

cout<<"Semester?:";

cin>>semester;

}

void display(){

cout<<"\nProgram="<<program;

cout<<"\nSemester="<<semester;

}

};

class derived:public B{

protected:

unsigned long int phn;

public:

void inputdata(){

cout<<"Phone No:";

cin>>phn;

}

void display(){

A::display();

B::display();

cout<<"\nPhone Number="<<phn<<endl;

}

};

int main(){

int n;

cout<<"Enter the number of students:";

cin>>n;

derived \*d=new derived[n];

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

cout<<"\n\t\tDetails of "<<(i+1)<<" Students:"<<endl;

d[i].A::inputdata();

d[i].B::inputdata();

d[i].inputdata();

d[i].display();

}

delete d;

return 0;

}

**///Prob3)**

**//Source Code:**

//Multiple Inheritence

#include<iostream>

using namespace std;

class A{

protected:

int idno,rollno;

char name[20];

public:

void inputdata(){

//cout<<"\t\tEnter the details of the student:"<<endl;

cout<<"\nName?:";

//cin.ignore();

cin>>name;

cout<<"Roll NO?:";

cin>>rollno;

cout<<"ID No?:";

cin>>idno;

}

void display(){

cout<<"\nName="<<name<<"\nRollNo="<<rollno<<"\nIDNo="<<idno<<endl;

}

};

class B{

protected:

char semester[20],program[20];

public:

void inputdata(){

//cin.ignore();

cout<<"Program?:";

cin>>program;

//cin.ignore();

cout<<"Semester?:";

cin>>semester;

}

void display(){

cout<<"\nProgram="<<program;

cout<<"\nSemester="<<semester;

}

};

class derived:public A,public B{

protected:

unsigned long int phn;

public:

void inputdata(){

A::inputdata();

B::inputdata();

cout<<"Phone No:";

cin>>phn;

}

void display(){

A::display();

B::display();

cout<<"Phone Number="<<phn<<endl;

}

};

int main(){

int n;

cout<<"Enter the number of students:";

cin>>n;

derived \*d=new derived[n];

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

cout<<"\n\t\tDetails of "<<(i+1)<<" Students:"<<endl;

// d[i].A::inputdata();

// d[i].B::inputdata();

d[i].inputdata();

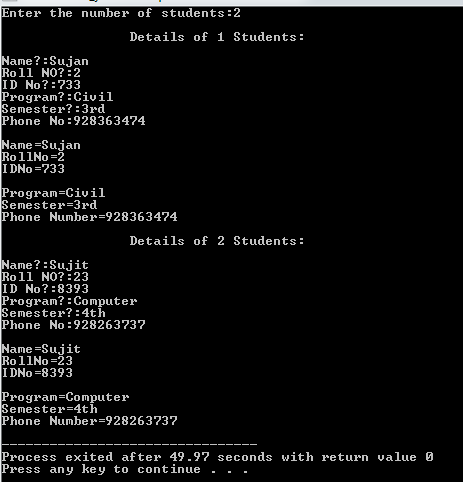
d[i].display();

}

delete d;

return 0;

}



**///Prob4)**

**//Source Code:**

//QN 4

#include<iostream>

using namespace std;

class Staff{

protected:

int staff\_id;

char name[20];

public:

void getdata(){

cout<<"\t\tEnter the details:"<<endl;

cout<<"Staff Id?:";

cin>>staff\_id;

cin.ignore();

cout<<"Staff Name?:";

cin.get(name,20);

}

void showdata(){

cout<<"\nStaff Idno="<<staff\_id<<endl<<"Staff Name="<<name;

}

};

class Lecturer:public Staff{

protected:

char subject[15],department[14];

public:

void getdata(){

cout<<"\nSubject?:";

cin>>subject;

cout<<"Department?:";

cin>>department;

}

void showdata(){

cout<<"\nSubject="<<subject<<endl<<"Department="<<department;

}

};

class Administrative:public Staff{

protected:

char post[15],department[15];

public:

void getdata(){

cout<<"\nPost?:";

cin>>post;

cout<<"Department?:";

cin>>department;

}

void showdata(){

cout<<"\nPost="<<post<<endl<<"Department="<<department;

}

};

int main(){

Lecturer obj1;

Administrative obj2;

//getdata

obj1.Staff::getdata();

obj1.getdata(); //lecturer

obj2.getdata(); //administrative

//showdata

cout<<"\n\t\tDisplaying Details";

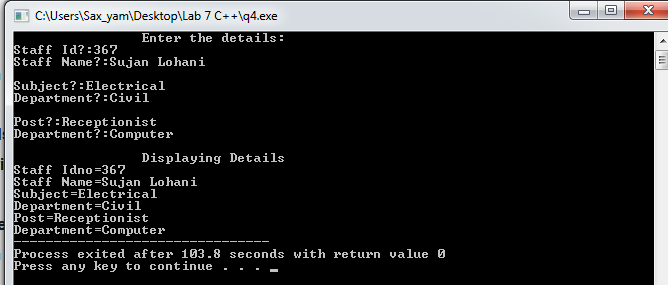
obj1.Staff::showdata();

obj1.showdata(); //lecturer

obj2.showdata(); //administrative

return 0;

}



**///Prob5)**

**//Source Code:**

//QN 5

#include<iostream>

using namespace std;

class Student{

protected:

int age,roll\_no;

char name[20];

public:

void getdata(){

cout<<"\t\tEnter the details of STUDENT:"<<endl;

cout<<"RollNo:";

cin>>roll\_no;

cout<<"Age:";

cin>>age;

cin.ignore();

cout<<"Name?:";

cin.get(name,20);

}

void showdata(){

cout<<"\nRoll No="<<roll\_no<<"\nAge="<<age<<"\nName="<<name<<endl;

}

};

class Mark:public Student{

protected:

int sessone,sesstwo;

public:

void getdata(){

cout<<"\nMarks of Sessional1?:";

cin>>sessone;

cout<<"Marks of Sessional2?:";

cin>>sesstwo;

}

void showdata(){

cout<<"\n Marks of Sessional1="<<sessone<<endl<<"Marks of Sessional2="<<sesstwo<<endl;

}

};

class Addmark:public Mark{

public:

void show(){

cout<<"Total Marks Obtained in Sessional1 & Sessional2="<<sessone+sesstwo;

}

};

int main(){

Addmark obj;

//getdata

obj.Student::getdata();

obj.Mark::getdata(); //Mark

//showdata

cout<<"\n\t\tDisplaying Details";

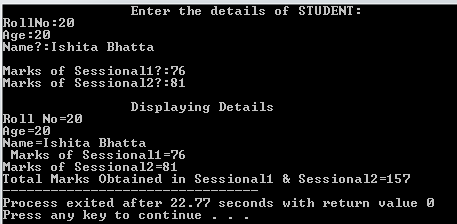
obj.Student::showdata();

obj.showdata(); //mark

obj.show(); //addmark

return 0;

}



**///Prob6)**

**//Source Code:**

//QN 6

#include<iostream>

using namespace std;

class Student{

protected:

int roll\_no;

char name[20];

public:

void getdata(){

cout<<"\t\tEnter the details of STUDENT:"<<endl;

cout<<"\*For STUDENT\*";

cout<<"\nRollNo:";

cin>>roll\_no;

cin.ignore();

cout<<"Name?:";

cin.get(name,20);

}

void showdata(){

cout<<"\nRoll No="<<roll\_no<<"\nName="<<name<<endl;

}

};

class Test:public Student{

protected:

int math,eng;

public:

void getdata(){

cout<<"\nMarks of Maths?:";

cin>>math;

cout<<"Marks of Eng.Drawing?:";

cin>>eng;

}

void showdata(){

cout<<"\n Marks of Maths="<<math<<endl<<"Marks of Eng.Drawing="<<eng<<endl;

}

};

class Sport{

protected:

int score;

public:

void getdata(){

cout<<"\nSport Score?:";

cin>>score;

}

void showdata(){

cout<<"Sport Score="<<score;

}

};

class Result:public Test,public Sport{

public:

void showdata(){

cout<<"\nTotal Score="<<(math+eng+score);

}

};

int main(){

Result obj1;

//getdata

obj1.Student::getdata();

cout<<"\n\*For TEST\*";

obj1.Test::getdata(); //test

cout<<"\n\*For SPORTS\*";

obj1.Sport::getdata(); //sport

//showdata

cout<<"\n\t\tDisplaying Details";

obj1.Student::showdata();

obj1.Test::showdata(); //test

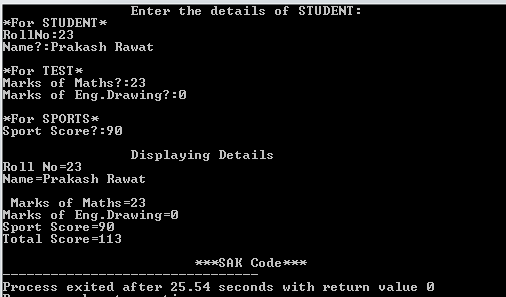
obj1.Sport::showdata(); //sport

obj1.Result::showdata(); //result

cout<<"\n\n\t\t\t\*\*\*SAK Code\*\*\*";

return 0;

}



**///Prob8)**

**//Source Code:**

//Q 8

#include<iostream>

using namespace std;

class Staff{

protected:

int code;

char name[20];

public:

void getdata(){

cout<<"\t\tEnter the details:"<<endl;

cout<<"\*For STAFF\*";

cout<<"\nCode?:";

cin>>code;

cin.ignore();

cout<<"Staff Name?:";

cin.get(name,20);

}

void showdata(){

cout<<"\nCode="<<code<<"\nStaff Name="<<name;

}

};

class Teacher:public Staff{

char subject[20],publication[20];

public:

void getdata(){

cout<<"\nSubject?:";

cin>>subject;

cout<<"Publication Name?:";

cin>>publication;

}

void showdata(){

cout<<"\nSubject="<<subject<<"\nPublication="<<publication<<endl;

}

};

class Typist:public Staff{

int speed;

public:

void getdata(){

cout<<"\nSpeed?:";

cin>>speed;

}

void showdata(){

cout<<"\nSpeed="<<speed<<endl;

}

};

class Officer:public Staff{

char grade[7];

public:

void getdata(){

cout<<"\nGrade?:";

cin>>grade;

}

void showdata(){

cout<<"\nGrade="<<grade<<endl;

}

};

class Regular:public Typist{

double salary;

public:

void getdata(){

cout<<"\nSalary?:";

cin>>salary;

}

void showdata(){

cout<<"\nSalary="<<salary<<endl;

}

};

class Casual:public Typist{

long int daily\_wages;

public:

void getdata(){

cout<<"\nDaily Wages?:";

cin>>daily\_wages;

}

void showdata(){

cout<<"\nDaily Wages="<<daily\_wages<<endl;

}

};

int main(){

Teacher T1;

Officer O1;

Regular R1;

Casual C1;

T1.Staff::getdata();

cout<<"\n\*For TEACHER\*";

T1.Teacher::getdata();

cout<<"\n\*For OFFICER\*";

O1.getdata();

cout<<"\n\*For TYPIST\*";

R1.Typist::getdata();

cout<<"\n\*For REGULAR\*";

R1.Regular::getdata();

cout<<"\n\*For CASUAL\*";

C1.getdata();

cout<<"\n\t\t\tDisplaying the details:";

T1.Staff::showdata();

T1.Teacher::showdata();

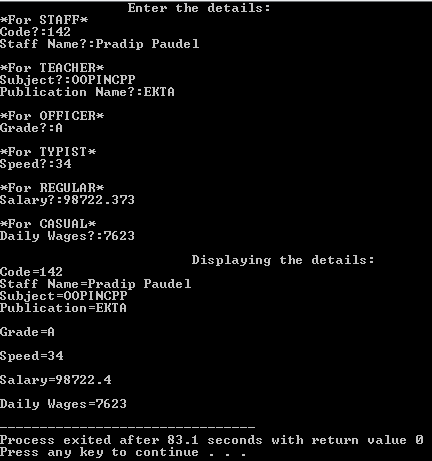
O1.showdata();

R1.Typist::showdata();

R1.Regular::showdata();

C1.showdata();

return 0;

}

**///Prob9)**

**//Source Code:**

//QN 9

#include<iostream>

using namespace std;

class Person{

protected:

int code;

char name[20];

public:

void getdata(){

cout<<"\t\tEnter the details of PERSON:"<<endl;

cout<<"Code:";

cin>>code;

cin.ignore();

cout<<"Name?:";

cin.get(name,20);

}

void showdata(){

cout<<"\nName="<<name<<"\nCode="<<code<<endl;

}

};

class Account:virtual public Person{

protected:

unsigned long int pay;

public:

void getdata(){

cout<<"\nPayment?:";

cin>>pay;

}

void showdata(){

cout<<"Payment="<<pay<<endl;

}

};

class Admin:virtual public Person{

protected:

int experience;

public:

void getdata(){

cout<<"Experience?:";

cin>>experience;

}

void showdata(){

cout<<"Experience="<<experience;

}

};

class Master:public Account,public Admin{

public:

void showdata(){

Person::showdata();

Account::showdata();

Admin::showdata();

cout<<"\n\nName="<<name<<"\nCode="<<code<<"\nPayment="<<pay<<"\nExperience="<<experience<<endl;

}

};

int main(){

Master M1;

//getdata

M1.Person::getdata();

M1.Account::getdata();

M1.Admin::getdata();

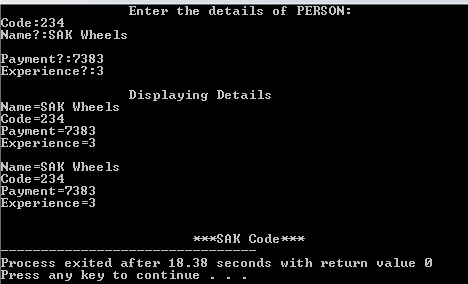
//showdata

cout<<"\n\t\tDisplaying Details";

M1.showdata();

cout<<"\n\n\t\t\t\*\*\*SAK Code\*\*\*";

return 0;

}

**///Prob10)**

**//Source Code:**

//Q 10

#include <iostream>

#include<string.h>

using namespace std;

class Person{

protected:

char name[30];

int age;

public:

void getD(){

cout<<"\t\t\tEnter the DETAILS of Customer\n";

cout<<"Name:? ";

cin.get(name,20);

cout<<"Age:? ";

cin>>age;

}

void display(void){

cout<<"Name: "<<name<<"\nAge: "<<age<<endl;

}

};

//has a - relationship

class Reservation: public Person{

private:

unsigned int rid;

short int yy,dd,mm;

Person P; // Composition

public:

Reservation(){

P.getD();

cout<<"\nRegistration ID(RID) Number:?";

cin>>rid;

cout<<"\t\t\tEnter the Date:\n";

cout<<"Day:? ";

cin>>dd;

cout<<"Month:? ";

cin>>mm;

cout<<"Year:? ";

cin>>yy;

}

void display(void){

cout<<"\n\t\tDisplaying the confirmation of the CUSTOMER:\n";

P.display();

cout<<"Registration ID: "<<rid<<"\nDate(dd/mm/yy): "<<dd<<"/"<<mm<<"/"<<yy<<endl;

}

};

int main()

{

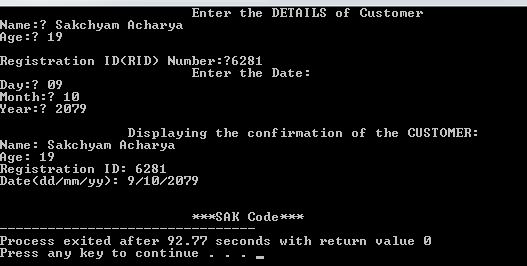
Reservation R;

R.display();

cout<<"\n\n\t\t\t\*\*\*SAK Code\*\*\*";

return 0;

}



**///Prob11)**

**//Source Code:**

//Q 11

#include<iostream>

using namespace std;

class derived;

class base{

protected:

int x,y;

public:

void getvec\_cor(){

cout<<"Enter X.Co-ordinate: ";

cin>>x;

cout<<"Enter Y.Co-ordinate: ";

cin>>y;

}

friend void display(derived);

};

class derived:public base{

public:

void add\_vector(derived d1, derived d2){

x=(d1.x+d2.x);

y=(d1.y+d2.y);

}

};

void display(derived d){

cout<<"\nX Co-ordinate="<<d.x<<endl<<"Y Co-ordinate="<<d.y<<endl;

}

int main(){

derived d1,d2,d3;

cout<<"\t\tInput 1st Vector Co-ordinate:"<<endl;

d1.getvec\_cor();

cout<<"\n\t\tInput 2nd Vector Co-ordinate:"<<endl;

d2.getvec\_cor();

d3.add\_vector(d1,d2);

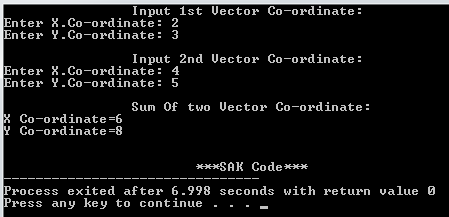
cout<<"\n\t\tSum Of two Vector Co-ordinate:";

display(d3);

cout<<"\n\n\t\t\t\*\*\*SAK Code\*\*\*";

return 0;

}



**Discussion & Conclusion:-**

The program is focused on various tasks on “**Different Types of Inheritance & its Reusability**”. From this program I understood how new classes(“derived class” or “child class”) are created from the existing classes(“base class” or “parent class”) & I have also learnt about how Inheritance support Reusability of Code.

**\* Thank You\*[SAK Wheels]**