**1 Solve this.**

**Fresh business scenario to apply inheritance , polymorphism to emp based organization scenario.**

**Create Emp based organization structure --- Emp , Mgr , Worker**

**1.1 Emp state--- id(int), name, deptId , basicSalary(double)**

**Accept all of above in constructor arguments.**

**Methods ---**

**1.2. compute net salary ---ret 0**

**(eg : public double computeNetSalary(){return 0;})**

**1.2 Mgr state ---id,name,basic,deptId , perfBonus**

**Add suitable constructor**

**Methods ----**

**1. compute net salary (formula: basic+perfBonus) -- override computeNetSalary**

**1.3 Worker state --id,name,basic,deptId,hoursWorked,hourlyRate**

**Methods :**

**1. compute net salary (formula: = basic+(hoursWorked\*hourlyRate) --override computeNetSalary**

**2. get hrlyRate of the worker -- add a new method to return hourly rate of a worker.(getter)**

**Create suitable array to store organization details.**

**Provide following options**

**1. Hire Manager**

**I/P : all manager details**

**2. Hire Worker**

**I/P : all worker details**

**3. Display information of all employees net salary (by invoking computeNetSal),**

**4. Exit**

#include<iostream>

#include<typeinfo>

#include<vector>

#include<fstream>

using namespace std;

class employee {

protected : int id, deptid;

string name;

double salary;

fstream file;

public :

employee()

{

id = 101;

deptid = 200;

name = "Rahul";

salary = 5000.0;

}

employee(int id,int deptid,string name, double salary)

{

this->id = id;

this->name = name;

this->salary = salary;

this->deptid = deptid;

}

virtual double netsalary(){

return 0;

}

void display(){

cout<<"Employee "<<name<<" having id - "<<id<<" and department id - "<<deptid<<" has a basic salary - "<<salary<<endl;

}

};

class manager : public employee

{

private : double perfbonus;

int x;

public :

manager()

{

perfbonus = 4000.0;

}

manager(int id , int deptid , string name ,double salary , double perfbonus) : employee(id,deptid,name,salary) , perfbonus(perfbonus) {}

double netsalary(){

return (salary+perfbonus);

}

void display(){

cout<<"Employee "<<name<<" having id - "<<id<<" and department id - "<<deptid<<" has a basic salary - "<<salary+perfbonus<<endl;

file.open("info.txt",ios::app);

file<<"Employee "<<name<<" ID "<<id<<" has a salary - Rs."<<salary<<endl;

file.close();

}

};

class worker : public employee

{

private : int workinghour , hourrate;

int x;

public :

worker(int id , int deptid ,string name , double salary, int workinghour , int hourrate) : employee(id ,deptid,name,salary) , workinghour(workinghour) , hourrate(hourrate) {}

double netsalary(){

return salary+(workinghour\*hourrate);

}

void display(){

cout<<"Employee "<<name<<" having id - "<<id<<" and department id - "<<deptid<<" has a basic salary - "<<salary+(workinghour\*hourrate)<<endl;

file.open("info.txt",ios::app);

file<<"Employee "<<name<<" ID "<<id<<" has a salary - Rs."<<salary<<endl;

file.close();

}

int hourlyrate()

{

return hourrate;

}

};

int main() {

// employee e(234,456,"Ashay",80000);

// e.display();

//

// manager m(345,678,"Nakul",89000,1000);

// m.netsalary();

// m.display();

//

// worker w(988,43,"Mitesh",17000,8,500);

// w.netsalary();

// w.display();

// by creating array display all the data

int choice ;

int id , deptid,perfbonus;

string name;

int workinghour,hourrate;

double salary;

vector<employee> emp;

employee e;

cout<<"\n1.Hire Manager\n2.Hire Worker\n3.Display all employee net salary\n4.EXIT"<<endl;

do

{

cout<<"Enter your choice - "; cin>>choice;

switch(choice)

{

case 1 : {

cout<<"\nEnter the employee id - ";cin>>id;

cout<<"\nEnter the department id of an manager - ";cin>>deptid;

cout<<"\nEnter the name of an manager - ";cin>>name;

cout<<"\nEnter the salary of an manager - ";cin>>salary;

cout<<"\nEnter the performance bonus of an manager - "; cin>>perfbonus;

manager \*m = new manager(id,deptid,name,salary,perfbonus);

emp.push\_back(\*m);

break;

}

case 2 :

{

cout<<"\nEnter the employee id - ";cin>>id;

cout<<"\nEnter the department id of an worker - ";cin>>deptid;

cout<<"\nEnter the name of an worker - ";cin>>name;

cout<<"\nEnter the salary of an worker - ";cin>>salary;

cout<<"\nEnter working hour of an worker - "; cin>>workinghour;

cout<<"\nEnter the hour rate of an worker - "; cin>>hourrate;

worker \*w = new worker(id,deptid,name,salary,workinghour,hourrate);

emp.push\_back(\*w);

cout<<"New student added"<<endl;

break;

}

case 3 :

{

for(int i=0;i<emp.size();i++)

{

emp[i].display();

}

break;

}

case 4: break;

default : cout<<"Invalid input..!!"<<endl;

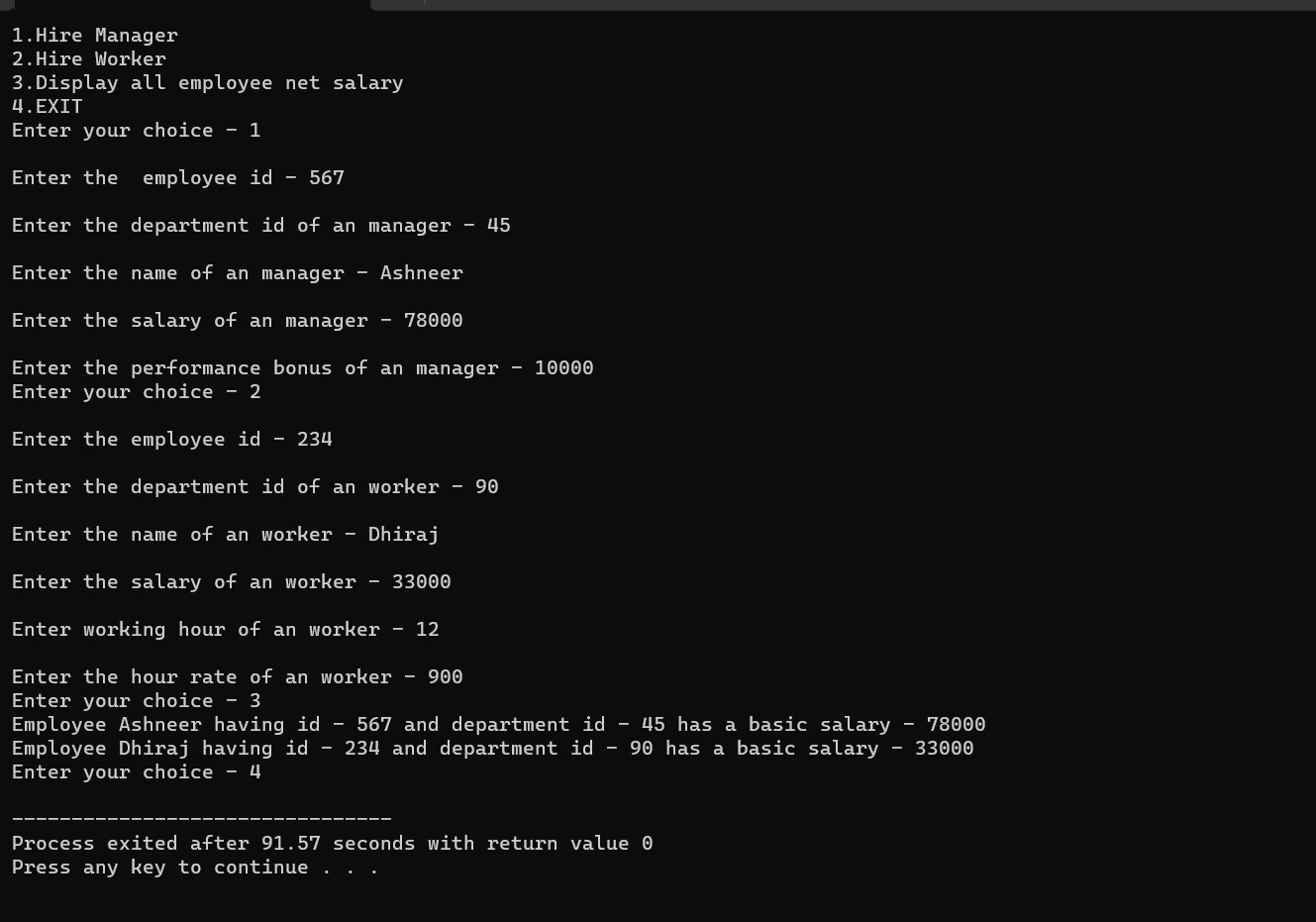
}

}

while(choice != 4);

return 0;

}



----------------------------------------------------

**2:Create cpp application for bank account handling.**

**2.1. Create a class BankAccount -- acct no(int),customer name(string),balance(double)**

**Add constr. (2 constrs : first to accept all details )**

**2.2 Add Business logic methods**

**Methods**

**public void withdraw(double amt)**

**public void deposit(double amt)**

**2.3: Create object of account class and test withdraw and deposit methods.**

Library :-

#include<iostream>

using namespace std;

class bankAccount

{

private : string name;

double balance;

int actid;

public : bankAccount()

{

name = "Ram";

this->balance = 12000.00;

actid = 101;

}

bankAccount(string name , int actid)

{

this->name = name;

this->actid = actid;

this->balance = 12000.00;

}

void deposit(double amt)

{

this->balance = this->balance + amt;

}

void withdraw(double amt)

{

this->balance = this->balance - amt;

}

void displaybalance()

{

cout<<"\nAccount balance is : "<<this->balance<<endl;

}

};

CPP FILE :-

#include<iostream>

#include "bankapp.h"

using namespace std;

int main()

{

string name;

double amount;

int actid;

cout<<"------ Bank Account details ------"<<endl;

cout<<"\n\nEnter your name - ";cin>>name;

cout<<"\n\nEnter your bank account id - "; cin>>actid;

bankAccount b(name,actid);

cout<<endl;

int choice;

cout<<"\n1.Withdraw money \n2.Deposit money \n3.Check & Display balance \n4.EXIT"<<endl;

double w,d;

do

{

cout<<"\nEnter your choice "; cin>>choice;

switch(choice)

{

case 1 : cout<<"Enter the amount you want to withdraw - ";cin>>w;

b.withdraw(w);

break;

case 2 : cout<<"Enter the amount you want to deposit - "; cin>>d;

b.deposit(d);

break;

case 3 : b.displaybalance();

break;

case 4 : break;

default : cout<<"\nInvalid input ..!!"<<endl; break;

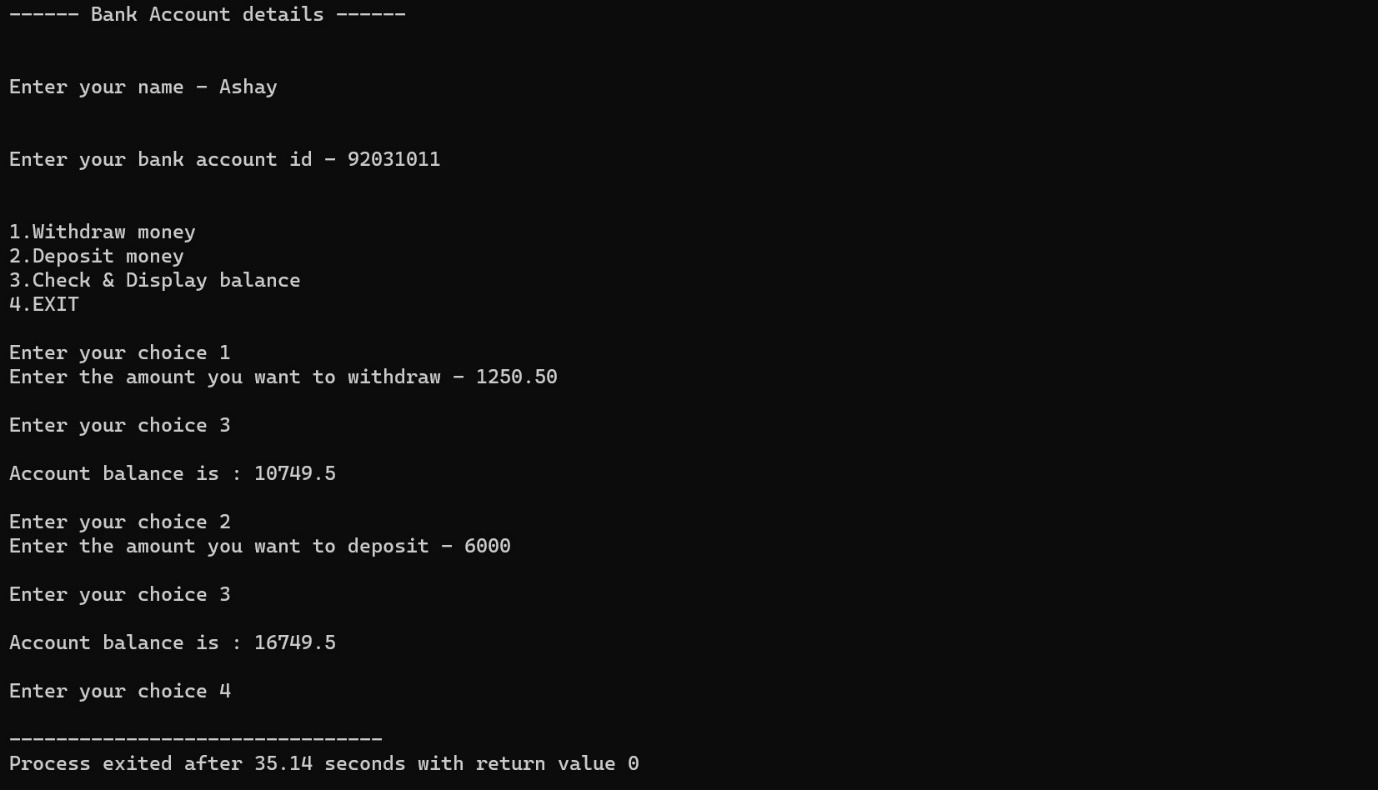
}

}

while(choice != 4);

return 0;

}



---------------------------------------------------------------------------------

**3:Create a abstract class Shape with pure virtual method area;**

**Create Rectangle,Circle,Square class..inherit them from Shape class..Override area method.**

**Test these all classes by creating object of respective class.**

#include<iostream>

using namespace std;

class SHAPE

{

public :

virtual void area() = 0;

virtual void perimeter() = 0;

};

class CIRCLE : public SHAPE

{

private : double radius ;

public :

CIRCLE(){

radius = 1;

}

CIRCLE(double radius){

this->radius = radius;

}

void area(){

cout<<"Area of circle for "<<radius<<" cm is = "<<(3.14\*radius\*radius)<<endl;

}

void perimeter(){

cout<<"Perimeter of circle for "<<radius<<" cm is "<<(2\*3.14\*radius)<<endl;

}

};

class SQUARE : public SHAPE

{

private : double side;

public :

SQUARE(){

side = 1;

}

SQUARE(double side){

this->side = side;

}

void area(){

cout<<"Area of square for "<<side<<" cm is "<<side\*side<<endl;

}

void perimeter(){

cout<<"Perimeter of square for "<<side<<" cm is "<<4\*side<<endl;

}

} ;

class RECTANGLE : public SHAPE

{

private : double length , breadth;

public :

RECTANGLE(){

length = 1;

breadth = 1;

}

RECTANGLE(double length , double breadth){

this->length = length;

this->breadth = breadth;

}

void area(){

cout<<"Area of Rectangle for length "<<length<<" cm and breadth "<<breadth<<" cm is = "<<length\*breadth;

}

void perimeter(){

cout<<"Perimeter of Rectangle for length "<<length<<" cm and breadth "<<breadth<<" cm is = "<<2\*(length+breadth)<<endl;

}

};

int main()

{

double radius,side,length,breadth;

int choice ;

cout<<"\nEnter the radius of circle - "; cin>>radius;

CIRCLE r(radius);

cout<<"\nEnter the side of square - " ;cin>>side;

SQUARE s(side);

cout<<"\nEnter the length and breadth of Rectangle - "; cin>>length>>breadth;

RECTANGLE t(length,breadth);

cout<<"\n1.Area of Circle\n2.Perimeter of Circle\n3.Area of Square \n4.Perimeter of Square\n5.Area of Rectangle\n6.Perimeter of Rectangle\n7.EXIT"<<endl;

do

{

cout<<"\nEnter your choice - "<<endl;

cin>>choice;

switch(choice){

case 1 : r.area(); break;

case 2 : r.perimeter(); break;

case 3 : s.area(); break;

case 4 : s.perimeter(); break;

case 5: t.area(); break;

case 6 : t.perimeter();break;

case 7 : break;

default : cout<<"Invalid input..!!"<<endl; break;

}

}

while(choice != 7 );

}

