National University of Computer and Emerging Sciences



Lab Exercise 11

For

Object Oriented Programming Lab

|  |  |
| --- | --- |
| Name | Muhammad Zain |
| Roll No | 19F-0228 |
| Course Instructor(s) | Dr. Danish |
| Lab Instructor(s) | Mr. Mughees Ismail |
| Semester | Spring 2020 |

**FAST School of Computing**

|  |
| --- |
| **Question#1** |

# Source Code;

#include<iostream>

#include<string>

#include<iomanip>

using namespace std;

class Employee {

private:

string Name;

string Rank;// of employee (Basic pay scale)

double Basic;// pay

double Medical;// (Medical allowance)

double House;//(House Rent)

double Gross;// pay (total pay)

void Display()

{

cout << endl << "Name of employee : " << this->Name;

cout << endl << "Rank of employee : " << this->Rank;

cout << endl << "Basic Pay of employee : " << this->Basic;

cout << endl << "Medical allowance : " << this->Medical;

cout << endl << "House Rent : " << this->House;

cout << endl << "Gross Pay : " << this->House;

}

public:

Employee()

{

cout << "This is Employee Class constructor" << endl;

Name = "-";

Rank = "-";

Basic = 0;

Medical = 0.0;

House = 0.0;

Gross = 0.0;

}

~Employee()

{

cout << "“Software Developed by{ Muhammad Zain}”" << endl;

}

friend void Read\_Record(Employee&);

friend void Gross\_Pay(Employee&);

friend double Annual\_Increment(Employee&);

friend void Output(Employee&);

};

void Read\_Record(Employee &rec)

{

cout << "Enter Name : ";

cin >> rec.Name;

while (rec.Name == "")

{

cin >> rec.Name;

}

cout << "Enter the Rank :";

cin >> rec.Rank;

while (rec.Rank == "")

{

cin >> rec.Rank;

}

cout << "Enter the the Basic Pay :";

cin >> rec.Basic;

while (rec.Basic == 0.0)

{

cin >> rec.Basic;

}

cout << "60 % Medical Allowance is Deducted as per the Company Policy" << endl;

rec.Medical = (rec.Basic \* 60) / 100;

cout << "28.9% House Rent is Deducted as per the Company Policy" << endl;

rec.House = (rec.Basic\*28.9) / 100;

}

void Gross\_Pay(Employee& gro) {

gro.Gross = gro.Basic - gro.Medical - gro.House;//gross pay of employee using basic pay and allowances.

}

double Annual\_Increment(Employee& inc)

{

inc.Basic += (inc.Basic\*20) / 100;

return inc.Basic;

}

//In the next year the pay of employee increases 20% of basic pay.

void Output(Employee& disp) {

disp.Display();

}

int main()

{

cout << "\tSitara Private Limited" << endl;

Employee object;

Read\_Record(object);

Gross\_Pay(object);

Output(object);

cout << endl << endl << "After one year There will be 20% Increase in Basic Pay " << endl;

cout << "Increamented Pay will be " << Annual\_Increment(object);

system("pause>0");

}

# Snip:

A screenshot of a computer

Description automatically generated

|  |
| --- |
| **Question#2** |

# Source Code:

#pragma warning(disable:4996)

#include<iostream>

#include<time.h>

#include<string>

#include<cstdlib>

#include<string.h>

using namespace std;

class Sender;

class Receiver;

struct tm PcTime;

class Channel {

private:

int frequency;

time\_t now;

string port;

public:

Channel()

{

cout << "This is Constructor of Channel Class" << endl;

}

void inputDetails()

{

cout << "Please input the frequency" << endl;

cin >> frequency;

cout << "Please input Port" << endl;

cin >> port;

}

void getTime()

{

now = time(0);//this will get the time from the operating system

PcTime = \*localtime(&now);

cout << "Date is :" << PcTime.tm\_mday << " / " << PcTime.tm\_mon + 1 << " / " << PcTime.tm\_year + 1900 << endl;

cout << "Time is :" << PcTime.tm\_hour << " : " << PcTime.tm\_min << " : " << PcTime.tm\_sec << endl;

}

void printDetails()

{

cout << "Frequency : " << frequency << " Hz" << endl;

cout << "Port : " << port << endl;

getTime();

}

};

struct message { string mess; int mess\_id; };

class Sender {

friend void path(Sender&, Receiver&);

private:

message obj\_M;

Channel obj\_C;

public:

Sender()

{

cout << "This is Construcotr of sender" << endl;

}

void LoadMessage()

{

cout << "Type the message" << endl;

cin>>obj\_M.mess;

obj\_M.mess\_id = rand()%10;

obj\_C.inputDetails();

}

};

class Receiver {

friend void path(Sender&, Receiver&);

private:

message object;

bool received;

public:

void printMessage()

{

cout << "Recived Message : " << object.mess << endl;

cout << "Message ID : " << object.mess\_id << endl;

}

};

void path(Sender &obj\_S, Receiver &obj\_R)

{

obj\_R.object.mess = obj\_S.obj\_M.mess;

obj\_R.object.mess\_id = obj\_S.obj\_M.mess\_id;

cout << endl << endl << "These detils are of sent message " << endl;

obj\_S.obj\_C.printDetails();

}

int main()

{

srand(time(0));

int temp;

Sender Obj\_S;

Receiver Obj\_R;

while (int x = 1)

{

cout << "Press 1 to Load Message" << endl;

cout << "\t\t (channel details and message will be inputted after this selection)" << endl;

cout << "Press 2 to Print Message" << endl;

cout << "\t\t(the path() will be called and then message will be printed with the details of the channel)" << endl;

cout << "Press 3 to Exit" << endl << endl;

cin >> temp;

//system("cls");

if (temp == 1)

{

Obj\_S.LoadMessage();

path(Obj\_S, Obj\_R);

}

else if (temp == 2)

{

cout << endl << endl;

cout << "These are Received message details" << endl;

Obj\_R.printMessage();

}

else if (temp == 3)

{

break;

}

}

system("Pause>0");

# }

# Snip;

|  |
| --- |
| **Question#3** |

# Source Code:

#include<iostream>

using namespace std;

class Friend\_Class;//forward declaration

class Friend\_Class2;//forward declaration

class Base {

int Base3;

public:

int Base1;

int Base2;

Base()

{

cout << "This is Base class Constructor" << endl << endl;

Base1 = 0;

Base2 = 0;

Base3 = 0;

}

Base(int a, int b, int c)

{

cout << "This is BAse Class overloaded constructor" << endl;

Base1 = a;

Base2 = b;

Base3 = c;

}

void Display()

{

cout << "Base 1 : " << Base1 << endl;

cout << "Base 2 : " << Base2 << endl;

cout << "Base 3 : " << Base3 << endl;

}

friend Friend\_Class;

};

class Friend\_Class

{ //if we make obj of class then we will unable to access it from nxt friend class

public:

int a;

int b;

int c;

void Access(Base obj\_B)

{

a = obj\_B.Base1;

b = obj\_B.Base2;

c = obj\_B.Base3;

cout << "1st Variable of base class : " << a << endl;

cout << "2nd Variable of base class : " << b << endl;

cout << "3rd Variable of base class : " << c << endl;

}

friend class Friend\_class2;

};

class Friend\_Class2 {

int x, y, z;

public:

void access1(Friend\_Class obj\_F1)// asscessing friend class

{

int x = obj\_F1.a;

int y = obj\_F1.b;

int z = obj\_F1.c;

cout << "We are Able to access all the variables of friend class in friend 2 class" << endl << endl;

cout << "1st Variable of base class in Second Friend Class via 1st Friend Class : " << x << endl;

cout << "2nd Variable of base class in Second Friend Class via 1st Friend Class : " << y << endl;

cout << "3rd Variable of base class in Second Friend Class via 1st Friend Class : " << z << endl << endl << endl;

}

void access2(Base obj\_B)//asscessing base class

{

int x = obj\_B.Base1;

int y = obj\_B.Base2;

cout << "We are unable to access base 3 (Private VAriable) of base class directlyin friend 2 class" << endl;

cout << "1st Variable of base class directly in Second Friend Class " << x << endl;

cout << "2nd Variable of base class directly in Second Friend Class " << y << endl;

}

};

int main()

{

int a, b, c;

cout << "Input the Value of Base 1" << endl;

cin >> a;

cout << "Input the Value of Base 2" << endl;

cin >> b;

cout << "Input the Value of Base 3" << endl;

cin >> c;

Base obj\_B(a, b, c);

//values to overloaded constructor

Friend\_Class obj\_F1;

obj\_F1.Access(obj\_B);

//Passing object of base class to friend class 1

//through which we can acess its attributes

Friend\_Class2 obj\_F2;

obj\_F2.access1(obj\_F1);

//object of friend class to accesssing friend class attributes

obj\_F2.access2(obj\_B);

//object of base class to acces base class attributes

//access directl from friend 2 class

cout << endl << endl << "\t\t\t\t\tHAPPY CODING" << endl;

system("pause>0");

}

# SnipA screenshot of a computer Description automatically generated

**HAPPY CODING**