National University of Computer and Emerging Sciences



Task given in Class2

Object Oriented Programming

|  |  |
| --- | --- |
| Name | Muhammad Zain |
| Roll No. | 19F-0228 |
| INSTRUCTOR | Dr Danish Shehzad |
| Semester | Spring 2020 |

Task 1:

# Source code:

#include<iostream>

#include<string>

using namespace std;

class B;//forward declaration

class A {

friend void fun(A, B);

int x;

public:

A()

{

cout << "\tThis is A class constructor " << endl;

cout << endl << endl << "Enter the value of X" << endl;

cin >> x;

}

};

class B {

friend void fun(A, B);

int y;

public:

B()

{

cout << "\tThis is B class constructor " << endl;

cout << endl << endl << "Enter the value of Y" << endl;

cin>>y;

}

};

void fun(A obj\_A, B obj\_B)

{

if (obj\_A.x > obj\_B.y)

{

cout << "Check = " << obj\_A.x << " > " << obj\_B.y;

cout << "The value of X in class A is greater than the value of variable y in B class" << endl;

}

else if (obj\_A.x<obj\_B.y)

{

cout << "Check = " << obj\_A.x << " < " << obj\_B.y;

cout << "The value of X in class A is less than the value of variable y in B class" << endl;

}

else if (obj\_A.x == obj\_B.y)

{

cout << "Check = " << obj\_A.x << " = " << obj\_B.y;

cout << "Both Values are equal" << endl;

}

}

int main()

{

A obj\_A;

B obj\_B;

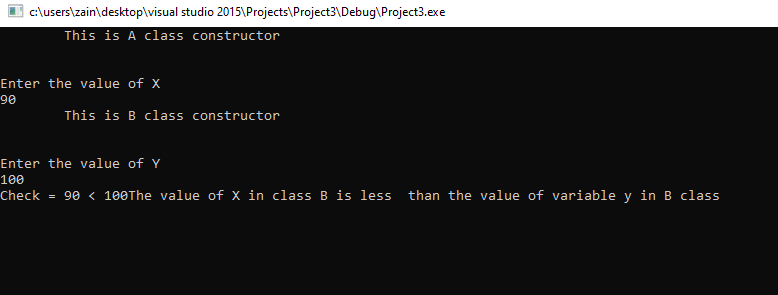
fun(obj\_A, obj\_B);

system("Pause>0")

;

}

# Snip;



Task 2:

# Source code:

#include<iostream>

using namespace std;

class B;

class A {

int x;

public:

A()

{

cout << "This is the constructor of A class" << endl;

x = 0;

}

friend class B;

};

class B {

int y;

A obj;

public:

B()

{

cout << "This is the constructor of B class" << endl;

y= 0;

}

void value\_A()

{

cout << "Input the value of variable present in class A" << endl;

cin >> obj.x;

}

void value\_B() {

cout << "Input the value of variable present in class B" << endl;

cin >> y;

}

void get\_Both() {

cout <<endl<< "The value of x in A class = " << obj.x;

cout << endl<<"The value of y in B class = " << y;

}

};

int main()

{

B obj;

obj.value\_A();

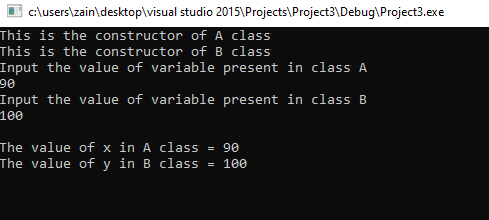
obj.value\_B();

obj.get\_Both();

system("pause>0");

}

# Snip:



Task 3:

Simple Destructor

# Source code:

#include<iostream>

using namespace std;

class A

{

public:

~A()

{

cout << "This is destructor in A class "<<endl;

}

};

class B :public A

{

public:

~B()

{

cout << "This is destructor in B class";

}

};

int main()

{

cout << "This destructors are without virtual keyword" << endl;

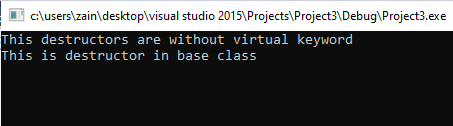
A \*ptr = new B;

delete ptr;

system("pause>0");

}

# Snip:



With Virtual Destructor

# Source Code:

#include<iostream>

using namespace std;

class A

{

public:

virtual~A()

{

cout << "This is Virtual Destructor of A Class "<<endl;

}

};

class B :public A

{

public:

~B()

{

cout << "This is the Virtual Destructor of B Class""<<endl;

}

};

int main()

{//without pointer we are not able to see destructor

//dynamic allocation wala scene hay

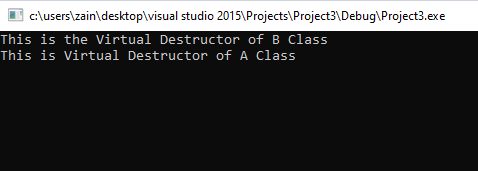
A \*ptr = new B;

delete ptr;

system("pause>0");

}

# Snip:



Task 4:

# Source Code:

#include<iostream>

#include<string>

#include<string.h>

using namespace std;

class Employee {

protected:

char \*firstName;

char \*lastName;

public:

Employee(const char \*, const char \*);

~Employee();

char \*getFirstName() const;

char \*getLastName() const;

// Pure virtual functions make Employee abstract base

//class.

virtual float earnings() const = 0; // pure virtual

virtual void print() const = 0; // pure virtual

};

Employee::Employee(const char \*first, const char \*last)

{

firstName = new char[strlen(first) + 1];

strcpy(firstName, first);

lastName = new char[strlen(last) + 1];

strcpy(lastName, last);

}

// Destructor deallocates dynamically allocated memory

Employee::~Employee() {

delete[] firstName; delete[] lastName;

}

// Return a pointer to the first name

char \*Employee::getFirstName() const {

return firstName; // caller must delete memory

}

char \*Employee::getLastName() const {

return lastName; // caller must delete memory

}

class SalariedEmployee : public Employee {

public:

SalariedEmployee(const char \*, const char \*, float =

0.0);

void setWeeklySalary(float);

virtual float earnings() const;

virtual void print() const;

private:

float weeklySalary;

};

// Constructor function for class

SalariedEmployee::SalariedEmployee(const char \*first,

const char \*last, float s)

: Employee(first, last) // call base-class constructor

{

weeklySalary = s > 0 ? s : 0;

}

// Set the SalariedEmployee’s salary

void SalariedEmployee::setWeeklySalary(float s)

{

weeklySalary = s > 0 ? s : 0;

}

// Get the SalariedEmployee’s pay

float SalariedEmployee::earnings() const { return weeklySalary; }

// Print the SalariedEmployee’s name

void SalariedEmployee::print() const

{

cout << endl << " Salaried Employee: " << getFirstName()

<< ' ' << getLastName();

}

class CommissionWorker : public Employee {

public:

CommissionWorker(const char \*, const char \*, float = 0.0, unsigned =

0);

void setCommission(float);

void setQuantity(unsigned);

virtual float earnings() const;

virtual void print() const;

private:

float commission; // amount per item sold

unsigned quantity; // total items sold for week

};

CommissionWorker::CommissionWorker(const char \*first,

const char \*last, float c, unsigned q)

: Employee(first, last) // call base-class constructor

{

commission = c > 0 ? c : 0;

quantity = q > 0 ? q : 0;

}

void CommissionWorker::setCommission(float c)

{

commission = c > 0 ? c : 0;

}

void CommissionWorker::setQuantity(unsigned q)

{

quantity = q > 0 ? q : 0;

}

float CommissionWorker::earnings() const

{

return commission \* quantity;

}

void CommissionWorker::print() const

{

cout << endl << "Commission worker: " << getFirstName()

<< ' ' << getLastName();

}

class HourlyWorker : public Employee {

public:

HourlyWorker(const char \*, const char \*,

float = 0.0, float = 0.0);

void setWage(float);

void setHours(float);

virtual float earnings() const;

virtual void print() const;

private:

float wage; // wage per hour

float hours; // hours worked for week

};

HourlyWorker::HourlyWorker(const char \*first, const char \*last,

float w, float h)

: Employee(first, last) // call base-class constructor

{

wage = w > 0 ? w : 0;

hours = h >= 0 && h < 168 ? h : 0;

}

void HourlyWorker::setWage(float w) { wage = w > 0 ? w : 0; }

// Set the hours worked

void HourlyWorker::setHours(float h)

{

hours = h >= 0 && h < 168 ? h : 0;

}

// Get the HourlyWorker's pay

float HourlyWorker::earnings() const { return wage \* hours; }

// Print the HourlyWorker's name

void HourlyWorker::print() const

{

cout << endl << " Hourly worker: " << getFirstName()

<< ' ' << getLastName();

}

class BasePlusCommissionEmployee :public

CommissionWorker

{

private:

float baseSalary;

public:

BasePlusCommissionEmployee(const char\*,

const char\*, float = 0.0, unsigned = 0, float = 0.0);

void setBaseSalary(float sal) {

baseSalary = sal;

}

float getBaseSalary(void) const {

return baseSalary;

}

void print() const;

float earnings() const;

};

BasePlusCommissionEmployee::BasePlusCommissionEmployee(const char\* first, const char\* last, float c, unsigned q, float sal) :CommissionWorker(first, last, c, q)

{

baseSalary = (sal);

}

void BasePlusCommissionEmployee::print() const

{

cout << "\nbase-salaried commission employee: ";

CommissionWorker::print(); // code reuse

} // end function print

float BasePlusCommissionEmployee::earnings() const

{

return getBaseSalary() + CommissionWorker::earnings();

} // end function earnings

int main()

{

Employee \*ptr; // base-class pointer

SalariedEmployee b("Nauman", "Sarwar", 800.00);

ptr = &b;

ptr->print();

cout << " earned $" << ptr->earnings();

b.print();

cout << " earned $" << b.earnings();

CommissionWorker c("Qasim", "Ali", 3.0, 150);

ptr = &c;

ptr->print();

cout << " earned $" << ptr->earnings();

c.print();

cout << " earned $" << c.earnings();

BasePlusCommissionEmployee p("Mehshan", "Mustafa", 2.5, 200, 1000.0);

ptr = &p;

ptr->print();

cout << " earned $" << ptr->earnings();

p.print();

cout << " earned $" << p.earnings();

HourlyWorker h("Samer", "Tufail", 13.75, 40);

ptr = &h;

ptr->print();

cout << " earned $" << ptr->earnings();

h.print();

cout << " earned $" << h.earnings();

cout << endl;

return 0;

}

# Snip:

