**LAB 10:  
SUBMITTED BY: NOOR-UL-AIN SAGHEER (BCS223020)**

**SUBMITTED TO: MAM SARA IBRAHIM**

**THEORY:**

* When we use virtual base inheritance, the object of the virtual base class is not stored in each object of the derived class.
* The object of the virtual base class is stored separately, and each derived class has a pointer to this object.

**DIAMOND PROBLEM SOLUTION CODE:**

#include<iostream>

#include<string>

using namespace std;

class person

{

protected:

string name;

public:

person(string n)

{

name = n;

}

void display()

{

cout << " --Person-- " << endl;

cout << " Name:" << name << endl;

}

};

class student :virtual public person

{

protected:

double gpa;

public:

student(string n, double g) :person(n)

{

gpa = g;

}

void display()

{

cout << " --Student-- " << endl;

cout << " Name: " << name << endl;

cout << " gpa: " << gpa << endl;

}

};

class professor :virtual public person

{

protected:

double salary;

public:

professor(string n, double sal) :person(n)

{

salary = sal;

}

void display()

{

cout << " --professor-- " << endl;

cout << " Name: " << name << endl;

cout << " salary: " << salary << endl;

}

};

class TA :public student, public professor

{

public:

TA(string n, double g, double sal) :person(n), student(n, g), professor(n, sal)

{

}

void display()

{

cout << " --Teaching assistant-- " << endl;

cout << " Name: " << name << endl;

cout << " GPA: " << gpa << endl;

cout << " Salary: " << salary << endl;

}

};

void main()

{

//testing person class

person p("noor");

p.display();

// testing student class

student s("noor", 3.87);

s.display();

//testing professor class

professor pro("noor", 20000);

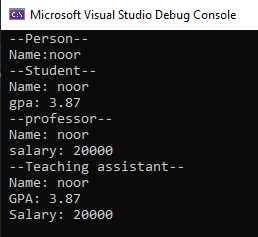
pro.display();

//testing teaching assistant class

TA obj("noor", 3.87, 20000);

obj.display();

} **OUTPUT:**



**PRACTICE TASK 2:**

**CODE:**

#include<iostream>

#include<string>

using namespace std;

class person

{

protected:

string name;

int age;

public:

person(string n, int a) : name(n), age(a) {}

virtual void print() const = 0;

};

class student : public person

{

protected:

string id;

double cgpa;

public:

student(string n, int a, string ID, double CGPA) : person(n, a), id(ID), cgpa(CGPA) {}

void print() const override

{

cout << "-- Student --" << endl;

cout << "Name: " << name << endl;

cout << "Age: " << age << endl;

cout << "ID: " << id << endl;

cout << "CGPA: " << cgpa << endl;

}

};

class regular : public student

{

protected:

string school\_name;

public:

regular(string n, int a, string Id, double Cgpa, string name) : student(n, a, Id, Cgpa), school\_name(name) {}

void print() const override

{

student::print();

cout << "School Name: " << school\_name << endl;

}

};

class privatee : public regular

{

protected:

string address;

int fee;

public:

privatee(string n, int a, string id, double cgpa, string name, string adr, int f)

: regular(n, a, id, cgpa, name), address(adr), fee(f) {}

void print() const override

{

regular::print();

cout << "Address: " << address << endl;

cout << "Fee: " << fee << endl;

}

};

int main()

{

student stu("Alice", 20, "123", 3.8);

regular reg("Bob", 22, "456", 3.5, "XYZ School");

privatee pri("Charlie", 25, "789", 3.9, "school", "123 Main St", 2500);

person\* ptr;

ptr = &stu;

ptr->print();

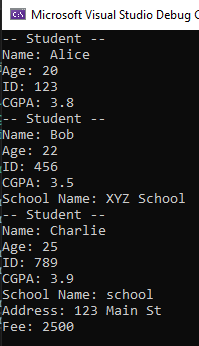
ptr = &reg;

ptr->print();

ptr = &pri;

ptr->print();

return 0;

} **OUTPUT:**  


**PRACTISE TASK 3:  
CODE:**

#include <iostream>

#include <iomanip>

using namespace std;

// Base Class: GeometricShape

class GeometricShape {

public:

// Pure virtual function

virtual void show() const = 0;

};

// Derived Class: Rectangle

class Rectangle : public GeometricShape {

protected:

double length;

double width;

public:

// Constructor

Rectangle(double l, double w) : length(l), width(w) {}

// Accessor functions

void setDimensions(double l, double w) {

length = l;

width = w;

}

// Overriding function to compute area for a rectangle

double computeArea() const {

return length \* width;

}

// Overriding function to display details for a rectangle

void show() const override {

cout << "-- Rectangle --" << endl;

cout << "Length: " << length << ", Width: " << width << endl;

cout << "Area: " << computeArea() << endl;

}

};

// Derived Class: Cuboid

class Cuboid : public Rectangle {

protected:

double height;

public:

// Constructor

Cuboid(double l, double w, double h) : Rectangle(l, w), height(h) {}

// Mutator function for height

void setHeight(double h) {

height = h;

}

// Overriding function to compute area for a cuboid

double computeArea() const override {

return 2 \* (length \* width + length \* height + width \* height);

}

// Overriding function to display details for a cuboid

void show() const override {

cout << "-- Cuboid --" << endl;

cout << "Length: " << length << ", Width: " << width << ", Height: " << height << endl;

cout << "Surface Area: " << computeArea() << endl;

}

};

int main() {

// Creating instances of derived classes

Rectangle rectangle(5.0, 3.0);

Cuboid cuboid(4.0, 6.0, 2.0);

// Accessing respective show() functions using dynamic binding

GeometricShape\* ptrShape = &rectangle;

ptrShape->show();

ptrShape = &cuboid;

ptrShape->show();

return 0;

}