**Lab 16 - Write a C++ program to implement virtual functions**

#include <iostream>

#include <string>

using namespace std;

class Person

{

protected:

string name;

int age;

public:

void setPersonDetails(string n, int a)

{

name = n;

age = a;

}

void displayPersonDetails()

{

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

}

};

class AcademicRecord

{

protected:

float gpa;

public:

void setAcademicRecord(float g)

{

gpa = g;

}

void displayAcademicRecord()

{

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

}

};

class ExtraCurricular

{

protected:

string activity;

public:

virtual void setActivity(string act)

{

activity = act;

}

virtual void displayActivity()

{

cout << "Extra-Curricular Activity: " << activity << endl;

}

};

class SportsActivity : public ExtraCurricular

{

public:

void displayActivity() override

{

cout << "Sports Activity: " << activity << endl;

}

};

class ArtsActivity : public ExtraCurricular

{

public:

void displayActivity() override

{

cout << "Arts Activity: " << activity << endl;

}

};

class Student : public Person, public AcademicRecord

{

protected:

int studentID;

float attendance;

public:

virtual float calculateFee()

{

return 0.0;

}

void setStudentDetails(int id, string n, int a, float g, float att)

{

studentID = id;

setPersonDetails(n, a);

setAcademicRecord(g);

attendance = att;

}

bool isEligibleForExam()

{

return attendance >= 75.0;

}

void displayStudentDetails()

{

cout << "Student ID: " << studentID << endl;

displayPersonDetails();

displayAcademicRecord();

cout << "Attendance: " << attendance << "%" << endl;

cout << "Exam Eligibility: " << (isEligibleForExam() ? "Eligible" : "Not Eligible") << endl;

}

};

class Undergraduate : public Student

{

public:

float calculateFee() override

{

return 1500.0;

}

};

class Postgraduate : public Student

{

public:

float calculateFee() override

{

return 2500.0;

}

};

class profile

{

private:

Student\* studentType;

ExtraCurricular\* extraCurricular;

public:

void setProfile(Student\* student, ExtraCurricular\* activityType)

{

studentType = student;

extraCurricular = activityType;

}

void displayProfile()

{

studentType->displayStudentDetails();

cout << "Fee: $" << studentType->calculateFee() << endl;

extraCurricular->displayActivity();

}

~profile()

{

delete studentType;

delete extraCurricular;

}

};

int main()

{

profile info;

int id, age;

float gpa, attendance;

string name;

Student\* studentType;

ExtraCurricular\* activityType;

cout << "Enter Student ID: ";

cin >> id;

cout << "Enter Name: ";

cin.ignore();

getline(cin, name);

cout << "Enter Age: ";

cin >> age;

cout << "Enter GPA: ";

cin >> gpa;

cout << "Enter Attendance Percentage: ";

cin >> attendance;

int studentChoice;

cout << "Select Student Type (1 for Undergraduate, 2 for Postgraduate): ";

cin >> studentChoice;

if (studentChoice == 1)

{

studentType = new Undergraduate();

}

else

{

studentType = new Postgraduate();

}

studentType->setStudentDetails(id, name, age, gpa, attendance);

int activityChoice;

cout << "Select Extra-Curricular Activity Type (1 for Sports, 2 for Arts): ";

cin >> activityChoice;

if (activityChoice == 1)

{

activityType = new SportsActivity();

activityType->setActivity("Football");

}

else

{

activityType = new ArtsActivity();

activityType->setActivity("Painting");

}

info.setProfile(studentType, activityType);

cout << "\nStudent Full Profile:\n";

info.displayProfile();

return 0;

}