Scenario:

Design a class hierarchy to represent various entities in a university system.

Base Class (Person):

Data members: name (string), age (int)

Member functions: getDetails(), a virtual function to print basic person details

Derived Class (Student): (Single Inheritance)

Inherits from Person

Data members: studentId (int), major (string)

Member functions:

setMajor(string) to set the student's major

getMajor() to retrieve the major

Override getDetails() to include student-specific information

Derived Class (Faculty): (Single Inheritance)

Inherits from Person

Data members: department (string), employeeId (int)

Member functions:

setDepartment(string) to set the faculty member's department

getDepartment() to retrieve the department

Override getDetails() to include faculty-specific information

Derived Class (TeachingAssistant): (Multilevel Inheritance)

Inherits from Student (inherits indirectly from Person as well)

Data member: coursesTeaching (array/vector of strings)

Member functions:

setCoursesTeaching(string[]) to set the courses the TA is teaching

getCoursesTeaching() to retrieve the list of courses

Override getDetails() to include TA-specific information (e.g., courses)

Derived Class (ResearchAssistant): (Hierarchical Inheritance)

Inherits from Person (separate inheritance from Student)

Data members: researchArea (string), supervisor (string)

Member functions:

setResearchArea(string) to set the research area

getResearchArea() to retrieve the research area

setSupervisor(string) to set the research supervisor

getSupervisor() to retrieve the supervisor

Override getDetails() to include RA-specific information

Derived Class (GraduateStudentTA): (Hybrid Inheritance)

Inherits from both Student and TeachingAssistant (combines functionality)

Might have additional data members or functions specific to graduate student Tas

Code:

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Person {

protected:

string name;

int age;

public:

Person(const string& n, int a) : name(n), age(a) {}

virtual void getDetails() {

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

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

}

};

class Student : public Person {

protected:

int studentId;

string major;

public:

Student(const string& n, int a, int id, const string& m)

: Person(n, a), studentId(id), major(m) {}

void getDetails() override {

Person::getDetails();

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

cout << "Major: " << major << endl;

}

};

class Faculty : public Person {

protected:

string department;

int employeeId;

public:

Faculty(const string& n, int a, const string& dept, int empId)

: Person(n, a), department(dept), employeeId(empId) {}

void getDetails() override {

Person::getDetails();

cout << "Department: " << department << endl;

cout << "Employee ID: " << employeeId << endl;

}

};

class TeachingAssistant : public Student {

protected:

vector<string> coursesTeaching;

public:

TeachingAssistant(const string& n, int a, int id, const vector<string>& courses)

: Student(n, a, id, ""), coursesTeaching(courses) {}

void setCoursesTeaching(const vector<string>& courses) {

coursesTeaching = courses;

}

void getDetails() override {

Student::getDetails();

cout << "Courses Teaching: ";

for (const auto& course : coursesTeaching) {

cout << course << ", ";

}

cout << endl;

}

};

class ResearchAssistant : public Person {

protected:

string researchArea;

string supervisor;

public:

ResearchAssistant(const string& n, int a, const string& area, const string& sup)

: Person(n, a), researchArea(area), supervisor(sup) {}

void getDetails() override {

Person::getDetails();

cout << "Research Area: " << researchArea << endl;

cout << "Supervisor: " << supervisor << endl;

}

};

class GraduateStudentTA : public Student, public TeachingAssistant {

protected:

string researchTopic;

public:

GraduateStudentTA(const string& n, int a, int id, const string& m, const vector<string>& courses)

: Student(n, a, id, m), TeachingAssistant(n, a, id, courses), researchTopic("") {}

void setResearchTopic(const string& topic) {

researchTopic = topic;

}

void getDetails() override {

Student getDetails();

TeachingAssistant::getDetails();

cout << "Research Topic: " << researchTopic << endl;

}

};

int main() {

GraduateStudentTA gst("Souvik", 25, 3001, "ECE", {"ECE101", "ECE202"});

gst.setCoursesTeaching({"CS303", "CS404"});

gst.setResearchTopic("Natural Language Processing");

cout << "Graduate Student TA details:" << endl;

gst.getDetails();

cout << endl;

return 0;

}

Flow Chart:

| Start |

|

| Define Person Class |

|

v

| Define Student Class|

|

v

| Define Faculty Class|

|

v

| Define TeachingAssistant |

| Class |

|

v

| Define ResearchAssistant |

| Class |

|

v

| Define GraduateStudentTA |

| Class |

|

v

| Main Function |

|

v

|Create GraduateStudentTA Inst|

|

v

| Set CoursesTeaching |

|

v

| Set ResearchTopic |

|

v

| Call getDetails() |

|

v

| End |