#include <iostream>

#include <vector>

using namespace std;

class Individual {

int age, id;

long long phone;

string fullname;

public:

Individual(int a, int b, long long c, string d) {

assignAge(a);

id = b;

phone = c;

assignName(d);

}

void assignAge(int a) {

if (a > 0 && a < 80) age = a;

else cout << "Invalid age provided";

}

void assignName(string d) {

if (d.empty()) cout << "Invalid name provided";

else fullname = d;

}

virtual void showDetails() {

cout << "Name: " << fullname << endl;

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

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

cout << "Phone: " << phone << endl;

}

virtual float getPayment() {

return 0.0;

}

};

class Learner : public Individual {

string joinDate, course;

public:

float score;

Learner(int a, int b, long long c, string d, string e, string f, float g)

: Individual(a, b, c, d), joinDate(e), course(f) {

assignGPA(g);

}

void assignGPA(float g) {

if (g > 0.0 && g < 4.0) score = g;

else cout << "GPA not valid";

}

void showDetails() override {

Individual::showDetails();

cout << "Joined: " << joinDate << endl;

cout << "Course: " << course << endl;

cout << "GPA: " << score << endl;

}

float getPayment() override {

return 20000 - (score \* 1000);

}

};

class UGStudent : public Learner {

string majorField, minorField, gradTime;

public:

UGStudent(int a, int b, long long c, string d, string e, string f, float g, string h, string i, string j)

: Learner(a, b, c, d, e, f, g), majorField(h), minorField(i), gradTime(j) {}

void showDetails() override {

Learner::showDetails();

cout << "Major: " << majorField << endl;

cout << "Minor: " << minorField << endl;

cout << "Graduation: " << gradTime << endl;

}

float getPayment() override {

return 20000 - (score \* 1200);

}

};

class PGStudent : public Learner {

string topic, mentor, thesis;

public:

PGStudent(int a, int b, long long c, string d, string e, string f, float g, string h, string i, string j)

: Learner(a, b, c, d, e, f, g), topic(h), mentor(i), thesis(j) {}

void showDetails() override {

Learner::showDetails();

cout << "Research: " << topic << endl;

cout << "Mentor: " << mentor << endl;

cout << "Thesis: " << thesis << endl;

}

float getPayment() override {

return 25000 - (score \* 1500);

}

};

class Educator : public Individual {

string dept, field, startDate;

public:

Educator(int a, int b, long long c, string d, string e, string f, string g)

: Individual(a, b, c, d), dept(e), field(f), startDate(g) {}

void showDetails() override {

Individual::showDetails();

cout << "Department: " << dept << endl;

cout << "Field: " << field << endl;

cout << "Start: " << startDate << endl;

}

float getPayment() override {

float pay = 50000;

if (dept == "CSE") pay += 10000;

else if (dept == "ECE") pay += 8000;

return pay;

}

};

class JrEducator : public Educator {

int yearsInTrack;

public:

JrEducator(int a, int b, long long c, string d, string e, string f, string g, int h)

: Educator(a, b, c, d, e, f, g), yearsInTrack(h) {}

void showDetails() override {

Educator::showDetails();

cout << "Track Years: " << yearsInTrack << endl;

}

float getPayment() override {

return 55000 + (yearsInTrack > 5 ? 5000 : 0);

}

};

class MidEducator : public Educator {

bool publications;

public:

MidEducator(int a, int b, long long c, string d, string e, string f, string g, bool h)

: Educator(a, b, c, d, e, f, g), publications(h) {}

void showDetails() override {

Educator::showDetails();

cout << "Publications: " << (publications ? "Yes" : "No") << endl;

}

float getPayment() override {

return 60000 + (publications ? 7000 : 0);

}

};

class SrEducator : public Educator {

int yearsAtLevel;

public:

SrEducator(int a, int b, long long c, string d, string e, string f, string g, int h)

: Educator(a, b, c, d, e, f, g), yearsAtLevel(h) {}

void showDetails() override {

Educator::showDetails();

cout << "Years at Level: " << yearsAtLevel << endl;

}

float getPayment() override {

return 70000 + (yearsAtLevel > 10 ? 10000 : 0);

}

};

class Subject {

string code, title, desc;

int units;

public:

Subject(string a, string b, string c, int d) : code(a), title(b), desc(c) {

if (d > 0) units = d;

else cout << "Invalid credit units";

}

};

class Dept {

string deptName, block;

int funds;

public:

Dept(string a, string b, int c) : block(b), funds(c) {

if (!a.empty()) deptName = a;

else cout << "Invalid department name";

}

};

class Records {

vector<int> ids;

vector<float> marks;

public:

void pushMark(int id, float mark) {

if (mark < 0.0 || mark > 4.0) {

cout << "Invalid grade.\n";

return;

}

ids.push\_back(id);

marks.push\_back(mark);

}

float averageMark() {

if (marks.empty()) return 0.0;

float total = 0;

for (float m : marks) total += m;

return total / marks.size();

}

float maxMark() {

if (marks.empty()) return 0.0;

float high = 0;

for (float m : marks) if (m > high) high = m;

return high;

}

void showFails() {

for (int i = 0; i < marks.size(); i++) {

if (marks[i] < 1.0)

cout << "Student with ID " << ids[i] << " is failing.\n";

}

}

};

class EnrollmentSystem {

vector<string> courseRefs;

vector<vector<int>> enrolled;

int locate(string code) {

for (int i = 0; i < courseRefs.size(); i++) {

if (courseRefs[i] == code) return i;

}

return -1;

}

public:

void addEnrollment(string code, int sid) {

int idx = locate(code);

if (idx == -1) {

courseRefs.push\_back(code);

enrolled.push\_back({});

idx = courseRefs.size() - 1;

}

enrolled[idx].push\_back(sid);

}

void removeStudent(string code, int sid) {

int idx = locate(code);

if (idx == -1) {

cout << "Course not found.\n";

return;

}

for (int i = 0; i < enrolled[idx].size(); i++) {

if (enrolled[idx][i] == sid) {

enrolled[idx].erase(enrolled[idx].begin() + i);

cout << "Removed student ID: " << sid << endl;

return;

}

}

cout << "Student ID " << sid << " not enrolled.\n";

}

void showEnrollments(string code) {

int idx = locate(code);

if (idx == -1) {

cout << "No such course.\n";

return;

}

cout << "Enrollment count for " << code << ": " << enrolled[idx].size() << endl;

}

};

void showInfo(Individual\* ref) {

ref->showDetails();

}

int main() {

Individual ind(26, 3114, 9643247444, "Kanishk");

Individual\* learnerPtr = new Learner(24, 2222, 9876543210, "Ravi", "24/4/2025", "MBA", 3.67);

Learner learner(21, 3333, 9876543211, "Anuj", "31/1/2024", "Btech", 3.63);

showInfo(&ind);

showInfo(learnerPtr);

Records rec;

rec.pushMark(3114, 3.6);

rec.pushMark(3312, 2.2);

rec.pushMark(4321, 0.6);

cout << rec.averageMark() << endl;

rec.showFails();

cout << rec.maxMark() << endl;

cout << "\n-- Polymorphic Example --\n";

Individual\* group[3];

group[0] = new Learner(20, 1001, 9876543210, "Amit", "01/01/2023", "BCA", 3.5);

group[1] = new Educator(45, 1002, 9123456789, "Dr. Sharma", "CSE", "AI", "15/08/2010");

group[2] = new Learner(22, 1003, 9988776655, "Neha", "12/12/2022", "MBA", 3.8);

for (int i = 0; i < 3; ++i) {

group[i]->showDetails();

cout << "Payment: " << group[i]->getPayment() << endl << "----------" << endl;

delete group[i];

}

UGStudent ug(22, 4444, 9876543233, "Vikram", "01/01/2024", "Btech", 3.75, "CS", "Math", "05/2027");

PGStudent pg(25, 5555, 9876543234, "Sneha", "01/01/2022", "Mtech", 3.9, "AI", "Gupta", "Deep Learning");

JrEducator jr(40, 6666, 9876543235, "Mehta", "ECE", "Signals", "12/08/2015", 6);

MidEducator mid(50, 7777, 9876543236, "Sharma", "CSE", "ML", "15/05/2010", true);

SrEducator sr(55, 8888, 9876543237, "Gupta", "ECE", "DSP", "20/02/2005", 12);

ug.showDetails(); cout << "Payment: " << ug.getPayment() << endl;

pg.showDetails(); cout << "Payment: " << pg.getPayment() << endl;

jr.showDetails(); cout << "Payment: " << jr.getPayment() << endl;

mid.showDetails(); cout << "Payment: " << mid.getPayment() << endl;

sr.showDetails(); cout << "Payment: " << sr.getPayment() << endl;

Subject subj("CS101", "Intro", "Programming basics", 4);

Dept d("CS", "Block A", 500000);

EnrollmentSystem es;

es.addEnrollment("CS101", 1001);

es.addEnrollment("CS101", 1003);

es.showEnrollments("CS101");

es.removeStudent("CS101", 1003);

es.showEnrollments("CS101");

delete learnerPtr;

return 0;

}