TASK 01

#include <iostream>

#include <string>

using namespace std;

class Device {

protected:

int deviceID;

string deviceName;

bool status;

string location;

public:

Device(int id, string name, string loc = "")

: deviceID(id), deviceName(name), status(false), location(loc) {}

virtual void turnOn() {

status = true;

cout << deviceName << " turned ON\n";

}

virtual void turnOff() {

status = false;

cout << deviceName << " turned OFF\n";

}

virtual void getStatus() const {

cout << deviceName << " is " << (status ? "ON" : "OFF") << ".\n";

}

virtual void displayInfo() const {

cout << "Device ID: " << deviceID << "\n"

<< "Name: " << deviceName << "\n"

<< "Status: " << (status ? "ON" : "OFF") << "\n"

<< "Location: " << (location.empty() ? "Not Set" : location) << "\n";

}

virtual ~Device() {}

};

class Light : public Device {

private:

int brightnessLevel;

string colorMode;

public:

Light(int id, string name, int brightness, string color, string loc = "")

: Device(id, name, loc), brightnessLevel(brightness), colorMode(color) {}

void displayInfo() const override {

Device::displayInfo();

cout << "Brightness: " << brightnessLevel << "\n"

<< "Color Mode: " << colorMode << "\n";

}

};

class Thermostat : public Device {

private:

float temperature;

string mode;

float targetTemperature;

public:

Thermostat(int id, string name, float temp, string m, float target, string loc = "")

: Device(id, name, loc), temperature(temp), mode(m), targetTemperature(target) {}

void getStatus() const override {

cout << deviceName << " is set to " << mode

<< " mode at " << temperature << "°C (Target: " << targetTemperature << "°C).\n";

}

};

class SecurityCamera : public Device {

private:

int resolution;

bool recordingStatus;

bool nightVisionEnabled;

public:

SecurityCamera(int id, string name, int res, bool nightVision, string loc = "")

: Device(id, name, loc), resolution(res), recordingStatus(false), nightVisionEnabled(nightVision) {}

void turnOn() override {

status = true;

recordingStatus = true;

cout << deviceName << " is now recording.\n";

}

void displayInfo() const override {

Device::displayInfo();

cout << "Resolution: " << resolution << "p\n"

<< "Night Vision: " << (nightVisionEnabled ? "Enabled" : "Disabled") << "\n"

<< "Recording: " << (recordingStatus ? "Active" : "Inactive") << "\n";

}

};

class SmartPlug : public Device {

private:

float powerConsumption;

int timerSetting;

public:

SmartPlug(int id, string name, float power, int timer, string loc = "")

: Device(id, name, loc), powerConsumption(power), timerSetting(timer) {}

void turnOff() override {

status = false;

cout << deviceName << " turned OFF. Power consumption logged: " << powerConsumption << "W.\n";

}

};

int main() {

Light light(1, "Living Room Light", 75, "Warm White");

Thermostat thermostat(2, "Main Thermostat", 22.5, "Cooling", 24.0);

SecurityCamera camera(3, "Front Door Camera", 1080, true);

SmartPlug plug(4, "TV Plug", 150, 2);

cout << "\n--- Device Information ---\n";

light.displayInfo();

thermostat.displayInfo();

camera.displayInfo();

plug.displayInfo();

cout << "\n--- Testing Devices ---\n";

cout << "\nTurning on devices:\n";

light.turnOn();

thermostat.turnOn();

camera.turnOn();

plug.turnOn();

cout << "\nGetting device status:\n";

light.getStatus();

thermostat.getStatus();

camera.getStatus();

plug.getStatus();

cout << "\nTurning off devices:\n";

light.turnOff();

thermostat.turnOff();

camera.turnOff();

plug.turnOff();

cout << "\nGetting final status:\n";

light.getStatus();

thermostat.getStatus();

camera.getStatus();

plug.getStatus();

return 0;

}

TASK 02

#include <iostream>

#include <string>

using namespace std;

class Product {

protected:

int id;

string name;

float basePrice;

int quantityInStock;

public:

Product(int pid, string pname, float price, int stock = 0)

: id(pid), name(pname), basePrice(price), quantityInStock(stock) {}

virtual float applyDiscount() const {

return basePrice;

}

virtual float getTotalCost(int qty) const {

return qty \* applyDiscount();

}

virtual void showDetails() const {

cout << "Product ID: " << id << "\n"

<< "Product Name: " << name << "\n"

<< "Base Price: $" << basePrice << "\n"

<< "Available Stock: " << quantityInStock << "\n";

}

};

class Electronics : public Product {

private:

int warrantyMonths;

string brandName;

public:

Electronics(int pid, string pname, float price, int stock, int warranty, string brand)

: Product(pid, pname, price, stock), warrantyMonths(warranty), brandName(brand) {}

void showDetails() const override {

Product::showDetails();

cout << "Warranty Duration: " << warrantyMonths << " months\n"

<< "Brand: " << brandName << "\n";

}

};

class Clothing : public Product {

private:

string size, color, fabric;

public:

Clothing(int pid, string pname, float price, int stock, string s, string c, string f)

: Product(pid, pname, price, stock), size(s), color(c), fabric(f) {}

float applyDiscount() const override {

return basePrice \* 0.9f; // 10% off

}

};

class FoodItem : public Product {

private:

string expiryDate;

int calorieCount;

public:

FoodItem(int pid, string pname, float price, int stock, string exp, int cal)

: Product(pid, pname, price, stock), expiryDate(exp), calorieCount(cal) {}

float getTotalCost(int qty) const override {

if (qty > 5)

return qty \* (basePrice \* 0.85f); // 15% off

return qty \* basePrice;

}

void showDetails() const override {

Product::showDetails();

cout << "Expires On: " << expiryDate << "\n"

<< "Calories per item: " << calorieCount << "\n";

}

};

class Book : public Product {

private:

string authorName;

string bookGenre;

public:

Book(int pid, string pname, float price, int stock, string author, string genre)

: Product(pid, pname, price, stock), authorName(author), bookGenre(genre) {}

void showDetails() const override {

Product::showDetails();

cout << "Author: " << authorName << "\n"

<< "Genre: " << bookGenre << "\n";

}

Book operator+(const Book& other) {

return Book(id, name, basePrice + other.basePrice, quantityInStock, authorName, bookGenre);

}

friend ostream& operator<<(ostream& out, const Book& b) {

out << "Book ID: " << b.id << "\n"

<< "Title: " << b.name << "\n"

<< "Cost: $" << b.basePrice << "\n"

<< "Stock Left: " << b.quantityInStock << "\n"

<< "Author: " << b.authorName << "\n"

<< "Genre: " << b.bookGenre << "\n";

return out;

}

};

int main() {

Electronics e1(1, "Laptop", 800, 10, 12, "Dell");

Clothing c1(2, "Shirt", 20, 50, "L", "Blue", "Cotton");

FoodItem f1(3, "Apple", 2, 100, "2025-01-01", 52);

Book b1(4, "C++ Programming", 30, 5, "Bjarne Stroustrup", "Programming");

cout << "\n--- Product Listings ---\n";

e1.showDetails();

c1.showDetails();

f1.showDetails();

cout << b1;

Book b2(5, "Python Programming", 25, 3, "Guido van Rossum", "Programming");

Book comboBook = b1 + b2;

cout << "\n--- Combined Book Info ---\n";

cout << comboBook;

return 0;

}

TASK 03

#include <iostream>

#include <string>

using namespace std;

class Ticket {

protected:

int ticketID;

string passengerName;

double price;

string date;

string destination;

public:

Ticket(int id, string name, double p, string d, string dest)

: ticketID(id), passengerName(name), price(p), date(d), destination(dest) {}

virtual void reserve() {

cout << "Ticket reserved for " << passengerName << " to " << destination << " on " << date << ".\n";

}

virtual void cancel() {

cout << "Ticket for " << passengerName << " has been cancelled.\n";

}

virtual void displayTicketInfo() const {

cout << "Ticket ID: " << ticketID << "\n"

<< "Passenger Name: " << passengerName << "\n"

<< "Price: $" << price << "\n"

<< "Date: " << date << "\n"

<< "Destination: " << destination << "\n";

}

virtual ~Ticket() {}

};

class FlightTicket : public Ticket {

private:

string airlineName;

string flightNumber;

string seatClass;

public:

FlightTicket(int id, string name, double p, string d, string dest, string airline, string flight, string seat)

: Ticket(id, name, p, d, dest), airlineName(airline), flightNumber(flight), seatClass(seat) {}

void displayTicketInfo() const override {

Ticket::displayTicketInfo();

cout << "Airline: " << airlineName << "\n"

<< "Flight Number: " << flightNumber << "\n"

<< "Seat Class: " << seatClass << "\n";

}

};

class TrainTicket : public Ticket {

private:

string trainNumber;

string coachType;

string departureTime;

public:

TrainTicket(int id, string name, double p, string d, string dest, string train, string coach, string time)

: Ticket(id, name, p, d, dest), trainNumber(train), coachType(coach), departureTime(time) {}

void reserve() override {

cout << "Train ticket auto-assigned for " << passengerName << " on " << trainNumber << " at " << departureTime << ".\n";

}

void displayTicketInfo() const override {

Ticket::displayTicketInfo();

cout << "Train Number: " << trainNumber << "\n"

<< "Coach Type: " << coachType << "\n"

<< "Departure Time: " << departureTime << "\n";

}

};

class BusTicket : public Ticket {

private:

string busCompany;

string seatNumber;

public:

BusTicket(int id, string name, double p, string d, string dest, string company, string seat)

: Ticket(id, name, p, d, dest), busCompany(company), seatNumber(seat) {}

void cancel() override {

cout << "Bus ticket for " << passengerName << " with " << busCompany << " (Seat: " << seatNumber << ") has been cancelled with full refund.\n";

}

void displayTicketInfo() const override {

Ticket::displayTicketInfo();

cout << "Bus Company: " << busCompany << "\n"

<< "Seat Number: " << seatNumber << "\n";

}

};

class ConcertTicket : public Ticket {

private:

string artistName;

string venue;

string seatType;

public:

ConcertTicket(int id, string name, double p, string d, string dest, string artist, string v, string seat)

: Ticket(id, name, p, d, dest), artistName(artist), venue(v), seatType(seat) {}

void displayTicketInfo() const override {

Ticket::displayTicketInfo();

cout << "Artist: " << artistName << "\n"

<< "Venue: " << venue << "\n"

<< "Seat Type: " << seatType << "\n";

}

};

int main() {

FlightTicket flight(1, "Ahmed Khan", 500.0, "2025-04-10", "Karachi", "PIA", "PK786", "Business");

TrainTicket train(2, "Faik Hussain", 75.0, "2025-04-12", "Lahore", "GreenLine", "AC Sleeper", "08:30 AM");

BusTicket bus(3, "Umer Farooq", 25.0, "2025-04-14", "Multan", "Daewoo Express", "B12");

ConcertTicket concert(4, "Hassan Ali", 100.0, "2025-05-05", "Islamabad", "Atif Aslam", "Jinnah Convention Center", "VIP");

cout << "\n--- Displaying Ticket Info ---\n";

flight.displayTicketInfo();

cout << "\n";

train.displayTicketInfo();

cout << "\n";

bus.displayTicketInfo();

cout << "\n";

concert.displayTicketInfo();

cout << "\n--- Testing Reservation and Cancellation ---\n";

flight.reserve();

train.reserve();

bus.cancel();

concert.reserve();

cout << "\n--- Testing Complete ---\n";

return 0;

}

TASK 4

#include <iostream>

#include <string>

using namespace std;

class Person {

protected:

string fullName;

int personAge;

string phoneNumber;

string homeAddress;

public:

Person(string n, int age, string phone, string address = "")

: fullName(n), personAge(age), phoneNumber(phone), homeAddress(address) {}

virtual void showProfile() const {

cout << "Full Name: " << fullName << "\n"

<< "Age: " << personAge << "\n"

<< "Contact: " << phoneNumber << "\n"

<< "Address: " << (homeAddress.empty() ? "Not Provided" : homeAddress) << "\n";

}

virtual void updateProfile(string n, int age, string phone, string address = "") {

fullName = n;

personAge = age;

phoneNumber = phone;

homeAddress = address;

}

virtual ~Person() {}

};

class Patient : public Person {

private:

int patID;

string history;

string assignedDoc;

public:

Patient(string n, int age, string phone, int id, string medHistory, string doctor)

: Person(n, age, phone), patID(id), history(medHistory), assignedDoc(doctor) {}

void showProfile() const override {

Person::showProfile();

cout << "Patient ID: " << patID << "\n"

<< "Medical History: " << history << "\n"

<< "Assigned Doctor: " << assignedDoc << "\n";

}

};

class Doctor : public Person {

private:

string field;

double fee;

string patientNames;

public:

Doctor(string n, int age, string phone, string spec, double consultation, string patients)

: Person(n, age, phone), field(spec), fee(consultation), patientNames(patients) {}

void showProfile() const override {

Person::showProfile();

cout << "Specialization: " << field << "\n"

<< "Consultation Fee: $" << fee << "\n"

<< "Patients: " << patientNames << "\n";

}

};

class Nurse : public Person {

private:

string ward;

string shift;

public:

Nurse(string n, int age, string phone, string assignedWard, string shiftTime)

: Person(n, age, phone), ward(assignedWard), shift(shiftTime) {}

void showProfile() const override {

Person::showProfile();

cout << "Ward: " << ward << "\n"

<< "Shift: " << shift << "\n";

}

};

class Administrator : public Person {

private:

string dept;

double monthlySalary;

public:

Administrator(string n, int age, string phone, string department, double salary)

: Person(n, age, phone), dept(department), monthlySalary(salary) {}

void updateProfile(string n, int age, string phone, string department, double salary) {

Person::updateProfile(n, age, phone);

dept = department;

monthlySalary = salary;

}

void showProfile() const override {

Person::showProfile();

cout << "Department: " << dept << "\n"

<< "Salary: $" << monthlySalary << "\n";

}

};

int main() {

cout << "\n--- Patient Details ---\n";

Patient p1("Arslan Rashid", 30, "123-456-7890", 101, "Diabetes", "Dr. Arslan");

p1.showProfile();

cout << "\n--- Doctor Details ---\n";

Doctor d1("Dr. Rouf", 45, "456-789-0123", "Cardiology", 200.0, "Patient A, Patient B");

d1.showProfile();

cout << "\n--- Nurse Details ---\n";

Nurse n1("Emie", 28, "321-654-0987", "ICU", "Night");

n1.showProfile();

cout << "\n--- Admin Details ---\n";

Administrator admin("Sarah", 40, "555-123-4567", "HR", 6000.0);

admin.showProfile();

cout << "\n--- Updating Admin Info ---\n";

admin.updateProfile("Sarah", 42, "555-123-4567", "Operations", 7000.0);

admin.showProfile();

return 0;

}

TASK 05

#include <iostream>

#include <string>

using namespace std;

class Character {

protected:

int charID;

string charName;

int charLevel;

int hp;

string equippedWeapon;

public:

Character(int id, string name, int level, int health, string weapon = "")

: charID(id), charName(name), charLevel(level), hp(health), equippedWeapon(weapon) {}

virtual void performAttack() {

cout << charName << " attacks using " << (equippedWeapon.empty() ? "fists" : equippedWeapon) << ".\n";

}

virtual void performDefense() {

cout << charName << " blocks the incoming strike.\n";

}

virtual void showStats() const {

cout << "Character ID: " << charID << "\n"

<< "Name: " << charName << "\n"

<< "Level: " << charLevel << "\n"

<< "HP: " << hp << "\n"

<< "Weapon: " << (equippedWeapon.empty() ? "None" : equippedWeapon) << "\n";

}

virtual ~Character() {}

};

class Warrior : public Character {

private:

int defensePower;

int swordDamage;

public:

Warrior(int id, string name, int level, int health, int armor, int damage)

: Character(id, name, level, health, "Sword"), defensePower(armor), swordDamage(damage) {}

void performAttack() override {

cout << charName << " slashes with sword for " << swordDamage << " damage!\n";

}

void performDefense() override {

cout << charName << " uses shield to absorb " << defensePower << " damage!\n";

}

};

class Mage : public Character {

private:

int mana;

int magicPower;

public:

Mage(int id, string name, int level, int health, int mp, int spellStrength)

: Character(id, name, level, health, "Staff"), mana(mp), magicPower(spellStrength) {}

void performDefense() override {

cout << charName << " casts magic shield absorbing " << magicPower << " damage!\n";

}

};

class Archer : public Character {

private:

int arrows;

double aimAccuracy;

public:

Archer(int id, string name, int level, int health, int arrowCount, double accuracy)

: Character(id, name, level, health, "Bow"), arrows(arrowCount), aimAccuracy(accuracy) {}

void performAttack() override {

if (arrows > 0) {

cout << charName << " fires an arrow with " << (aimAccuracy \* 100) << "% precision!\n";

arrows--;

} else {

cout << charName << " has no arrows left!\n";

}

}

};

class Rogue : public Character {

private:

int stealth;

int speed;

public:

Rogue(int id, string name, int level, int health, int stealthLv, int agility)

: Character(id, name, level, health, "Daggers"), stealth(stealthLv), speed(agility) {}

void showStats() const override {

Character::showStats();

cout << "Stealth Level: " << stealth << "\n"

<< "Agility: " << speed << "\n";

}

};

int main() {

cout << "\n--- Warrior Profile ---\n";

Warrior w1(1, "Ali", 10, 150, 20, 25);

w1.showStats();

w1.performAttack();

w1.performDefense();

cout << "\n--- Mage Profile ---\n";

Mage m1(2, "Hamza", 15, 100, 50, 30);

m1.showStats();

m1.performDefense();

cout << "\n--- Archer Profile ---\n";

Archer a1(3, "Ahmed", 12, 120, 5, 0.95);

a1.showStats();

a1.performAttack();

a1.performAttack();

cout << "\n--- Rogue Profile ---\n";

Rogue r1(4, "Usman", 14, 110, 90, 85);

r1.showStats();

return 0;

}