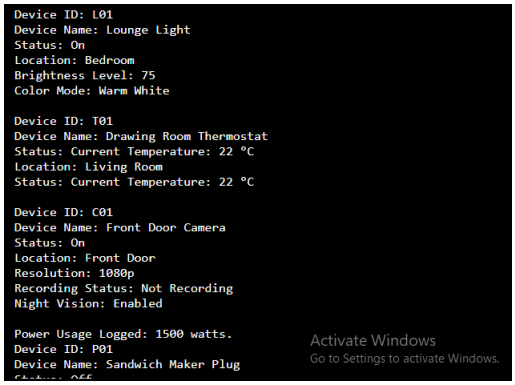
01)

```
#include <iostream>
#include <string>
using namespace std;

class Device {
protected:
    string deviceID;
    string deviceName;
    bool status;
    string location;

public:
    Device(string id, string name, bool stat = false, string loc = "") : deviceID(id),
deviceName(name), status(stat), location(loc) {}

    virtual void turnOn() {
        status = true;
    }

    virtual void turnOff() {
        status = false;
    }

    virtual string getStatus() const {
        return status ? "On" : "Off";
    }

    virtual void displayInfo() const {
        cout << "Device ID: " << deviceID << "\nDevice Name: " << deviceName << "\nStatus: " <<
getStatus() << "\nLocation: " << location << endl;
    }
};

class Light : public Device {
    int brightnessLevel;
    string colorMode;

public:
    Light(string id, string name, int brightness, string color, bool stat = false, string loc = "") :
Device(id, name, stat, loc), brightnessLevel(brightness), colorMode(color) {}

    void displayInfo() const override {
        Device::displayInfo();
```

```
        cout << "Brightness Level: " << brightnessLevel << "\nColor Mode: " << colorMode << endl;
    }
};

class Thermostat : public Device {
    int temperature;
    string mode;
    int targetTemperature;

public:
    Thermostat(string id, string name, int temp, string mod, int targetTemp, bool stat = false,
string loc = "") : Device(id, name, stat, loc), temperature(temp), mode(mod),
targetTemperature(targetTemp) {}

    string getStatus() const override {
        return "Current Temperature: " + to_string(temperature) + " °C";
    }
};

class SecurityCamera : public Device {
    string resolution;
    bool recordingStatus;
    bool nightVisionEnabled;

public:
    SecurityCamera(string id, string name, string res, bool nightVision, bool stat = false, string loc
= "") : Device(id, name, stat, loc), resolution(res), nightVisionEnabled(nightVision),
recordingStatus(false) {}

    void turnOn() override {
        status = true;
        recordingStatus = true;
    }

    void displayInfo() const override {
        Device::displayInfo();
        cout << "Resolution: " << resolution << "\nRecording Status: " << (recordingStatus ?
"Recording" : "Not Recording") << "\nNight Vision: " << (nightVisionEnabled ? "Enabled" :
"Disabled") << endl;
    }
};

class SmartPlug : public Device {
    int powerConsumption;
```

```
    int timerSetting;

public:
    SmartPlug(string id, string name, int power, int timer, bool stat = false, string loc = "") :
Device(id, name, stat, loc), powerConsumption(power), timerSetting(timer) {}

    void turnOff() override {
        status = false;
        cout << "Power Usage Logged: " << powerConsumption << " watts." << endl;
    }
};

int main() {
    Light light1("L01", "Lounge Light", 75, "Warm White", true, "Bedroom");
    Thermostat thermostat1("T01", "Drawing Room Thermostat", 22, "Cooling", 20, true, "Living
Room");
    SecurityCamera camera1("C01", "Front Door Camera", "1080p", true, true, "Front Door");
    SmartPlug plug1("P01", "Sandwich Maker Plug", 1500, 30, true, "Kitchen");

    light1.displayInfo();
    cout << endl;

    thermostat1.displayInfo();
    cout << "Status: " << thermostat1.getStatus() << endl << endl;

    camera1.displayInfo();
    cout << endl;

    plug1.turnOff();
    plug1.displayInfo();

    return 0;
}
```



03)

```
#include <iostream>
#include <string>
using namespace std;

class Ticket {
protected:
    string ticketID;
    string passengerName;
    double price;
    string date;
    string destination;

public:
    Ticket(string 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." << endl;
    }

    virtual void cancel() {
        cout << "Ticket cancelled." << endl;
    }

    virtual void displayTicketInfo() const {
        cout << "Ticket ID: " << ticketID << "\nPassenger Name: " << passengerName << "\nPrice:
" << price << "\nDate: " << date << "\nDestination: " << destination << endl;
    }
};

class FlightTicket : public Ticket {
    string airlineName;
    string flightNumber;
    string seatClass;

public:
    FlightTicket(string id, string name, double p, string d, string dest, string airline, string
flightNum, string seatC)
        : Ticket(id, name, p, d, dest), airlineName(airline), flightNumber(flightNum),
seatClass(seatC) {}

    void displayTicketInfo() const override {
        Ticket::displayTicketInfo();
```

```
        cout << "Airline Name: " << airlineName << "\nFlight Number: " << flightNumber << "\nSeat
Class: " << seatClass << endl;
    }
};

class TrainTicket : public Ticket {
    string trainNumber;
    string coachType;
    string departureTime;

public:
    TrainTicket(string id, string name, double p, string d, string dest, string trainNum, string coach,
string time)
        : Ticket(id, name, p, d, dest), trainNumber(trainNum), coachType(coach),
departureTime(time) {}

    void reserve() override {
        cout << "Seat automatically assigned and reserved." << endl;
    }
};

class BusTicket : public Ticket {
    string busCompany;
    int seatNumber;

public:
    BusTicket(string id, string name, double p, string d, string dest, string company, int seat)
        : Ticket(id, name, p, d, dest), busCompany(company), seatNumber(seat) {}

    void cancel() override {
        cout << "Last-minute refund processed. Ticket cancelled." << endl;
    }
};

class ConcertTicket : public Ticket {
    string artistName;
    string venue;
    string seatType;

public:
    ConcertTicket(string id, string name, double p, string d, string dest, string artist, string ven,
string seatT)
        : Ticket(id, name, p, d, dest), artistName(artist), venue(ven), seatType(seatT) {}
```

```
    void displayTicketInfo() const override {
        Ticket::displayTicketInfo();
        cout << "Artist Name: " << artistName << "\nVenue: " << venue << "\nSeat Type: " <<
seatType << endl;
    }
};

int main() {
    FlightTicket flight("FT001", "Sara", 500.0, "2025-03-30", "New York", "Airway Express",
"AX123", "Economy");
    TrainTicket train("TT001", "Ali", 150.0, "2025-03-30", "Chicago", "Train123", "First Class",
"10:00 AM");
    BusTicket bus("BT001", "Hasan", 50.0, "2025-03-30", "Los Angeles", "Bus Co.", 12);
    ConcertTicket concert("Maha", "Dave", 100.0, "2025-03-30", "New York", "The Weeknd",
"Madison Square Garden", "VIP");

    flight.displayTicketInfo();
    cout << endl;

    train.reserve();
    train.displayTicketInfo();
    cout << endl;

    bus.cancel();
    bus.displayTicketInfo();
    cout << endl;

    concert.displayTicketInfo();

    return 0;
}
```

```
Ticket ID: FT001
Passenger Name: Sara
Price: 500
Date: 2025-03-30
Destination: New York
Airline Name: Airway Express
Flight Number: AX123
Seat Class: Economy

Seat automatically assigned and reserved.
Ticket ID: TT001
Passenger Name: Ali
Price: 150
Date: 2025-03-30
Destination: Chicago

Last-minute refund processed. Ticket cancelled.
Ticket ID: BT001
Passenger Name: Hasan
Price: 50
Date: 2025-03-30
Destination: Los Angeles

Ticket ID: Maha
```

```
Ticket ID: Maha
Passenger Name: Dave
Price: 100
Date: 2025-03-30
Destination: New York
Artist Name: The Weeknd
Venue: Madison Square Garden
Seat Type: VIP
```

04)

```
#include <iostream>
#include <string>
using namespace std;

class Person {
protected:
    string name;
    int age;
    string contactNumber;
    string address;

public:
    Person(string n, int a, string c, string addr) : name(n), age(a), contactNumber(c),
address(addr) {}

    virtual void displayInfo() const {
        cout << "Name: " << name << "\nAge: " << age << "\nContact Number: " <<
contactNumber << "\nAddress: " << address << endl;
    }

    virtual void updateInfo(string newContact, string newAddress) {
        contactNumber = newContact;
        address = newAddress;
    }
};

class Patient : public Person {
    string patientID;
    string medicalHistory;
    string doctorAssigned;

public:
    Patient(string n, int a, string c, string addr, string id, string history, string doctor)
        : Person(n, a, c, addr), patientID(id), medicalHistory(history), doctorAssigned(doctor) {}

    void displayInfo() const override {
        Person::displayInfo();
        cout << "Patient ID: " << patientID << "\nMedical History: " << medicalHistory << "\nDoctor
Assigned: " << doctorAssigned << endl;
    }
};

class Doctor : public Person {
```

```
    string specialization;
    double consultationFee;
    string patientsList[10];
    int patientCount;

public:
    Doctor(string n, int a, string c, string addr, string spec, double fee)
        : Person(n, a, c, addr), specialization(spec), consultationFee(fee), patientCount(0) {}

    void addPatient(string patientName) {
        if (patientCount < 10) {
            patientsList[patientCount++] = patientName;
        } else {
            cout << "Patient list is full." << endl;
        }
    }

    void displayInfo() const override {
        Person::displayInfo();
        cout << "Specialization: " << specialization << "\nConsultation Fee: " << consultationFee
<< "\nPatients: ";
        for (int i = 0; i < patientCount; ++i) {
            cout << patientsList[i];
            if (i < patientCount - 1) cout << ", ";
        }
        cout << endl;
    }
};

class Nurse : public Person {
    string assignedWard;
    string shiftTimings;

public:
    Nurse(string n, int a, string c, string addr, string ward, string shift)
        : Person(n, a, c, addr), assignedWard(ward), shiftTimings(shift) {}

    void displayInfo() const override {
        Person::displayInfo();
        cout << "Assigned Ward: " << assignedWard << "\nShift Timings: " << shiftTimings << endl;
    }
};

class Administrator : public Person {
```

```
    string department;
    double salary;

public:
    Administrator(string n, int a, string c, string addr, string dept, double sal)
        : Person(n, a, c, addr), department(dept), salary(sal) {}

    void updateInfo(string newContact, string newAddress, double newSalary) {
        Person::updateInfo(newContact, newAddress);
        salary = newSalary;
    }

    void displayInfo() const override {
        Person::displayInfo();
        cout << "Department: " << department << "\nSalary: " << salary << endl;
    }
};

int main() {
    Patient patient("Alice", 30, "123-456-7890", "123 Street", "P001", "Diabetes", "Dr. Smith");
    Doctor doctor("Dr. Smith", 45, "987-654-3210", "456 Avenue", "Cardiology", 2000);
    Nurse nurse("Eve", 28, "111-111-1111", "789 Boulevard", "Ward A", "Day Shift");
    Administrator admin("Bob", 40, "444-444-4444", "321 Drive", "HR", 5000);

    doctor.addPatient("Alice");

    cout << "Patient Information:" << endl;
    patient.displayInfo();
    cout << "\nDoctor Information:" << endl;
    doctor.displayInfo();
    cout << "\nNurse Information:" << endl;
    nurse.displayInfo();
    cout << "\nAdministrator Information:" << endl;
    admin.displayInfo();

    return 0;
}
```

```
Patient Information:
Name: Alice
Age: 30
Contact Number: 123-456-7890
Address: 123 Street
Patient ID: P001
Medical History: Diabetes
Doctor Assigned: Dr. Smith

Doctor Information:
Name: Dr. Smith
Age: 45
Contact Number: 987-654-3210
Address: 456 Avenue
Specialization: Cardiology
Consultation Fee: 2000
Patients: Alice

Nurse Information:
Name: Eve
Age: 28
Contact Number: 111-111-1111
Address: 789 Boulevard
Assigned Ward: Ward A
Shift Timings: Day Shift

Administrator Information:
Name: Bob
Age: 40
Contact Number: 444-444-4444
Address: 321 Drive
Department: HR
Salary: 5000
```

05)

```
#include <iostream>
#include <string>
using namespace std;

class Character {
protected:
    string characterID;
    string name;
    int level;
    int healthPoints;
    string weaponType;

public:
    Character(string id, string n, int lvl, int hp, string weapon)
        : characterID(id), name(n), level(lvl), healthPoints(hp), weaponType(weapon) {}

    virtual void attack() const = 0;
    virtual void defend() const = 0;
    virtual void displayStats() const;
```

```
};

void Character::displayStats() const {
    cout << "Character ID: " << characterID << "\nName: " << name
         << "\nLevel: " << level << "\nHealth Points: " << healthPoints
         << "\nWeapon Type: " << weaponType << endl;
}

class Warrior : public Character {
    int armorStrength;
    int meleeDamage;

public:
    Warrior(string id, string n, int lvl, int hp, string weapon, int armor, int damage)
        : Character(id, n, lvl, hp, weapon), armorStrength(armor), meleeDamage(damage) {}

    void attack() const override {
        cout << name << " attacks with melee damage of " << meleeDamage << "." << endl;
    }

    void defend() const override {
        cout << name << " defends with armor strength of " << armorStrength << "." << endl;
    }
};

int main() {
    Warrior warrior("W001", "Aragorn", 10, 100, "Sword", 50, 30);

    cout << "Warrior Stats:" << endl;
    warrior.displayStats();
    warrior.attack();
    warrior.defend();

    return 0;
}
```

```
Warrior Stats:
Character ID: W001
Name: Aragorn
Level: 10
Health Points: 100
Weapon Type: Sword
Aragorn attacks with melee damage of 30.
Aragorn defends with armor strength of 50.

--------------------------------
Process exited after 0.3171 seconds with return value 0
Press any key to continue . . .
```