Introduzione al Progetto

Obiettivo

L'obiettivo del progetto SmartHouse è sviluppare un **modello di dominio in C#** in grado di simulare e gestire i dispositivi tipici di un'abitazione intelligente. Il sistema si concentra sull'astrazione, l'ereditarietà e il polimorfismo per garantire che nuove tipologie di dispositivi possano essere integrate facilmente nella struttura esistente.

Architettura

Il progetto è costruito su una gerarchia di classi ben definita:

`AbstractDevice`: La classe base per tutti i dispositivi.

Classi Intermedie/Modelli: Aggiungono specificità funzionali (es. `ClimateDevices`, `LampModel`).

Classi Concrete: Rappresentano i dispositivi reali (es. `Lamp`, `Thermostat`, `Door`, `Cctv`).

Classi di Aggregazione: Gestiscono e coordinano gruppi di dispositivi (`TwoLampDevice`, `LampRow`).

Dispositivo Base: `AbstractDevice`

La classe astratta `AbstractDevice` è il cuore del sistema e definisce le proprietà e i comportamenti comuni a **tutti** i dispositivi Smart House.

Proprietà Fondamentali

`Id`: Identificatore univoco globale.

`Name`: Nome assegnato dall'utente.

`IsOn`: Stato booleano di accensione.

`Status`: Stato operativo dettagliato (`DeviceStatus`).

`CreatedAtUtc` / **`LastModifiedAtUtc`**: Timestamp di creazione e ultima modifica.

Metodi Fondamentali

`TurnOn()`: Accende il dispositivo.

`TurnOff()`: Spegne il dispositivo.

`Toggle()`: Alterna lo stato di accensione/spegnimento.

`Rename(string newName)`: Modifica il nome del dispositivo.

Sottosistema Illuminazione (Luminous Devices)

Questo sottosistema gestisce i dispositivi che emettono luce.

Classe Base `LampModel`

Eredita da `AbstractDevice` e aggiunge la proprietà `Brightness` (livello di luminosità) e `IsEco` (flag per l'eco-compatibilità).

`increaseBrightness()`: Aumenta il livello di luminosità.

`decreaseBrightness()`: Diminuisce il livello di luminosità.

Implementazioni Concrete

`Lamp` (Standard): Implementa la regolazione della luminosità vincolata tra i limiti **MIN_BRIGHTNESS (1)** e **MAX_BRIGHTNESS (10)**.

`EcoLamp` (Risparmio Energetico):

`SwitchPowerSaveMode()`: Alterna la modalità di risparmio energetico, limitando la luminosità massima a 5.

`ShuldBeActivatedPowerSaveMode()`: Attiva automaticamente la modalità di risparmio energetico se il dispositivo è rimasto acceso per più di 180 minuti.

Sottosistema Clima

Questo sottosistema gestisce i dispositivi che regolano la temperatura ambientale.

Classe Base `ClimateDevices`

Eredità da `AbstractDevice` e aggiunge la proprietà `Temperature` (`double`).

`SetTemperature(double temperature)`: Imposta la temperatura target al valore specificato.

`IncreaseTemperature(double increment)`: Aumenta la temperatura target del valore dato.

`DecreaseTemperature(double decrement)`: Diminuisce la temperatura target del valore dato.

Implementazioni Concrete

`Thermostat` (Termostato): Offre un controllo più fine della temperatura.

`DimmerTemperatureUp()`: Aumenta la temperatura di **0.5** gradi (`DefaultDimmer`).

`DimmerTemperatureDown()`: Diminuisce la temperatura di **0.5** gradi.

`AirConditioner` (Condizionatore): Introduce la gestione delle modalità operative.

`SetAirConditionerStatus(AirConditionerStatus status)`: Imposta la modalità operativa (Heating, Cooling, Dry, Auto).

Sottosistema Sicurezza e Accesso

Telecamera (`Cctv`)

Metodo `SetCctvStatus(CctvStatus status)`: Imposta la modalità di visualizzazione della telecamera (Normal, Termic, NightVision).

Porta Intelligente (`Door`)

Gestisce lo stato fisico della porta e la serratura, basandosi sulle proprietà `IsOpen` e `IsLocked`.

`Open()`: Apre la porta (se accesa e sbloccata).

`Close()`: Chiude la porta (se accesa e sbloccata).

`Lock()`: Blocca la serratura (se accesa e chiusa).

`Unlock()`: Sblocca la serratura (se accesa e chiusa).

`SwitchOpenClose()`: Alterna lo stato di apertura/chiusura.

`SwitchLockUnlock()`: Alterna lo stato di blocco/sblocco.

Classi di Gestione

Queste classi sfruttano il **polimorfismo** per controllare gruppi di dispositivi.

Gestione di Coppia: `TwoLampDevice`

Controlla due lampade in modo sincrono.

`SwitchOnOffBothLamps()`: Alterna lo stato di accensione/spegnimento di entrambe le lampade.

`AreBothOn()`: Verifica se entrambe le lampade sono accese.

`IncreaseBrightnessBoth()` / **`DecreaseBrightnessBoth()`**: Regolano la luminosità di entrambe le lampade.

Gestione di Gruppo: `LampRow`

Controlla una lista dinamica di lampade (`List<LampModel>`).

`AddLamp`/ `AddEcoLamp()`: Aggiungono lampade alla lista.

`TurnOnAll()` / `TurnOffAll()`: Controllano lo stato di accensione/spegnimento di tutte le lampade nella riga.

`SetIntensityForAllLamps(int brightness)`: Imposta un livello di luminosità specifico per tutte le lampade.

`FindAllLampOn()` / `FindAllLampOff()`: Restituiscono le liste delle lampade accese o spente.

`FindLampById(Guid id)`: Cerca una lampada specifica tramite ID.