Practice 3 – Smart Home System

Błażej Drozd

Project repository: https://github.com/Tsugumik/programacion-uja

Changes and feedback implementation

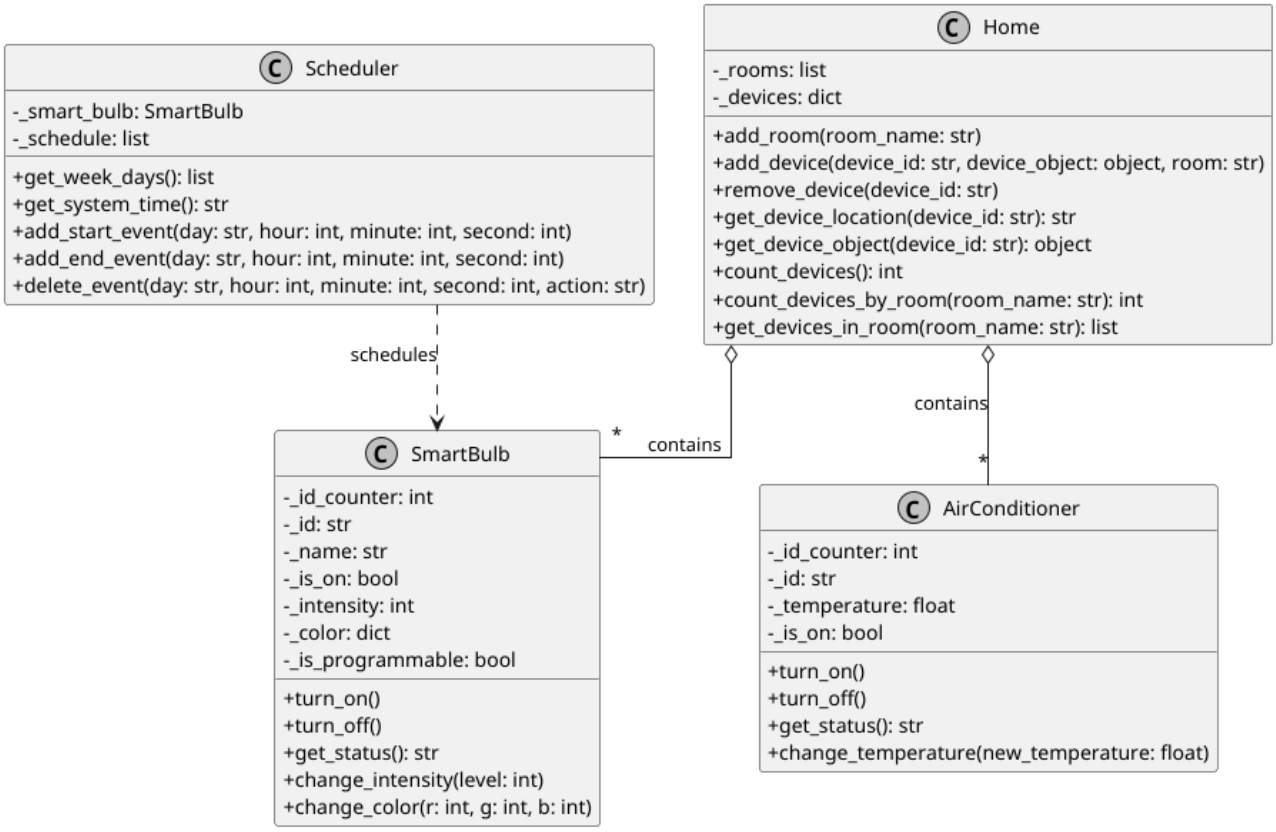
Based on the feedback, the following improvements have been made:

- Encapsulation
  - All class attributes in Home, SmartBulb and AirConditioner have been changed to private (e.g. _attribute). Access to these attributes is now managed through properties (@property), ensuring data protection.
- Project structure and imports
  - The module import problem has been fixed, the main.py file has been moved to the main project directory, and the import paths have been corrected to allow main.py correctly manage the Home class and its associated device classes.
- Translation to english
  - For better readability, the entire project (including file names, classes, methods and variables) have been translated from Spanish to English. The original Spanish was becoming a little bit confusing to me, and it is generally better practice to maintain software projects in english.

New features (Practice 3)

- Unique Ids
  - A mechanism for automatically generating unique Ids for devices (e.g. Bulb_0, AC_1) has been implemented using class counters. This ensures that each device has a unique id from the moment it is created.
- Scheduler
  - A new class has been added, which allows for the programming of events (turning on/off) for SmartBulb objects. This class is responsible for managing a schedule of events for a specific bulb.
- Exception Handling
  - An exception handling mechanism has been introduced. For example, the Scheduler class now throws a custom InvalidTimeError when a user tries to add an event with an incorrect day of the week or hour, which is then caught in main.py.

UML Diagram



The repository also includes a diagram.puml file containing PlantUML diagram for the entire system, illustrating the relationship between classes.

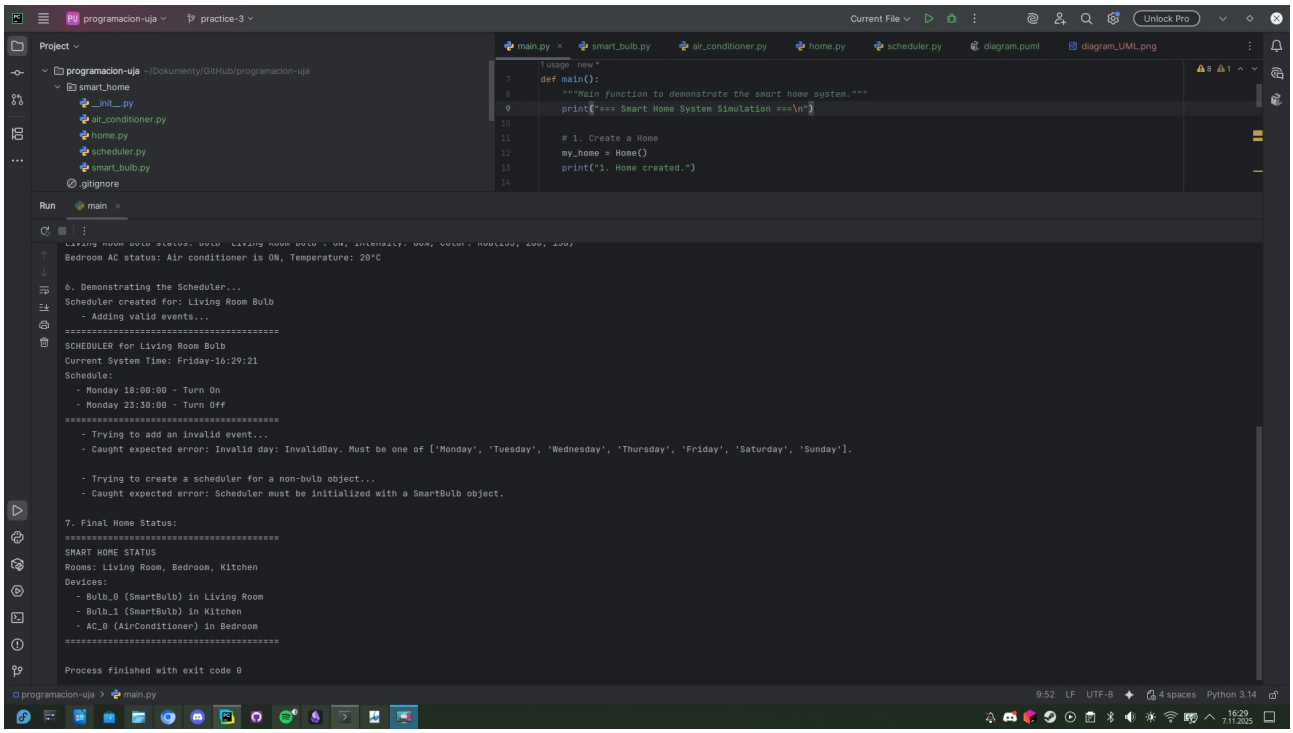
User Stories Validation

Also I verified that the current implementation of the Smart Home meets all the criteria defined in the user stories.

- HU01: Smart Bulb Management
  - Can be turned on and off:
    - Fulfilled: The SmartBulb class has the turn_on() and turn_off() methods.
  - Know its status:
    - Fulfilled: The get_status() method provides a full string representation of the bulb's status. Additionally, the is_on property can be checked for a boolean status
  - Change the intensity:
    - Fulfilled: The change_intensity(level) method allows changing the light intensity. It also includes validation to ensure the level is between 0 and 100.
  - Change the color:
    - Fulfilled: The change_color(r, g, b) method allows changing the RGB color. It includes validation to ensure the values are between 0 and 255.
- HU02: Air Conditioner Management

    - Be able to know the air temperature:
        - Fulfilled: The temperature property of the AirConditioner class returns the current temperature.
    - Be able to change the temperature:
        - Fulfilled: The change_temperature(new_temperature) method allows setting a new temperature. It includes validation to ensure the temperature is within a valid range (16-30°C).
    - Be able to know if it is off or on:
        - Fulfilled: The is_on property returns a boolean status, and the get_status() method provides a descriptive string.
    - Be able to turn it off or on:
        - Fulfilled: The AirConditioner class has the turn_on() and turn_off() methods.
- HU03: Device Distribution in the Home
    - Be able to tell how many rooms the home has and what rooms they are:
        - Fulfilled: The rooms property of the Home class returns a list of room names. The number of rooms can be obtained with len(home.rooms).
    - Be able to tell what devices are in the home:
        - Fulfilled: The devices property of the Home class returns a dictionary of all devices, with their IDs as keys.
    - Be able to add a device:
        - Fulfilled: The add_device(device_id, device_object, room) method allows adding a new device to a specific room.
    - Be able to remove a device:
        - Fulfilled: The remove_device(device_id) method allows removing a device from the home using its ID.
    - Be able to modify a device:
        - Fulfilled: This is achieved in a robust, object-oriented way. The user can retrieve the specific device object using home.get_device_object(device_id) and then call its own methods (e.g., bulb.change_intensity(80) or ac.change_temperature(21)). This is more secure and flexible than a generic modification method.
    - Be able to know the number of devices in the home and in each room:
        - Fulfilled: The count_devices() method returns the total number of devices, and the count_devices_by_room(room_name) method returns the count for a specific room.
    - Be able to identify the device that is in each location:
        - Fulfilled: The get_device_location(device_id) method returns the room where a specific device is located.

Main.py test



The application works without any problems.