Project Overview

You are required to implement an application that simulates a smart home management system. In this application the users must be able to create, manage and control electronic devices from different rooms of the house. A maximum allowed power consumption as well as day and night mode can also be set by the user. We will consider two types of users, admins and normal users. Admins of the system must be able to manage rooms and devices (add/delete) and change different settings (power mode – time mode). A normal users are only allowed to check the room and devices and control their functioning (search / turn on/ turn off….etc). We will consider two type of devices: lights and appliances.

Requirements

Implement the below classes by adding all the necessary constructor overload, getters and setters and various methods.

1. Device class:

Represents an electrical device:

- **Attributes**:
  - `int id:` unique value between 100 and 999 that represents the id of a given device. Default value is 0 and it indicates that the setup of the device has not been done correctly.
  - `String name:` the name of the device (ex: MainLight-MasterBedroom).
  - `int status:` 0 if the device is off – 1 if the device is on and 2 if the device is in standby mode.
  - `double maxPowerConsumption:` maximum power consumed by the device when it is on (in watts). Default is 50.
  - `boolean critical:` indicates if this is a critical device – critical devices can not be shut down without double confirmation. Default is false.
- **Constructors**:
  - `Device(int id, String name, double maxPowerConsumption)`
  - `Device(int id, String name, double maxPowerConsumption, boolean critical)`
- **Methods**:
  - `Getters and setters for all data fields`
  - `turnOn()`
  - `turnOff()`
  - `getCurrentConsumption()` : returns maxPowerConsumption if the device is on and 0 otherwise.
  - `String toString()`: returns a string representation of the device following format: id=[id] , name=[name], [status: On/ Off/Standby], maximum power consumption = [maxPowerConsumption], [critical/not critical]

2. Light Class (subclass of Device)

Represents a lighting device. Lights can be adjustable or not. Adjustable lights allows the user to adjust the light intensity percentage to a value between 0 and 100. Real power consumption of an adjustable light will be equal to the chosen level percentage multiplied by the maximum power consumption.

- **Attributes**:
  - `boolean adjustable`: indicates whether the light is adjustable or not.
  - `int level:` indicated the intensity level. Default is 100.
- **Constructors**:
  - `Light(int id, String name, double maxPowerConsumption)`
  - `Light(int id, String name, double maxPowerConsumption, boolean adjustable).`
  - `Light(int id, String name, double maxPowerConsumption, boolean critical, boolean adjustable).`
- **Methods**:
  - `Getters and setters for all data fields`
  - `turnOn():` override the turnOn method to turn on the light at 100% intesity.
  - `turnOn(level: int):` overloads the turnOn method to turn on the light on a specific level.
  - `getCurrentConsumption()`: if the light is on, returns the real power consumption of the light (according to the intensity level and maximum power consumption) . 0 otherwise.
  - `String toString()`: returns a string representation of the device following format: Light{ id=[id] , name=[name], [status: On/ Off/Standby], maximum power consumption = [maxPowerConsumption], [critical/not critical], [adjustable/not adjustable], level = [level] }

---

3. Appliance Class (Subclass of `Device`)

Represents a heavy appliance. Appliances can have one or multiple power levels (power consumption for each level is defined as a percentage of the maximum power consumption of the device). Different power levels are represented using an array of integers. The element at position i represents the i^{th} power level and its value represent the power consumption percentage for this level. The array must be sorted in acceding order.

For example the array [50, 100] indicates that the device can be activated with level 0 or level 1. At level 0 the device will consume 50% of its maximum power consumption and at level 1 the device will be consuming 100% of its maximum power consumption. The array or power levels can be of any size (1 or more).

In addition, noisy appliances are signaled by the `noisy` boolean variable in order to avoid their activation during night time.

- **Attributes**:
  - `int[] powerLevels:` array of power consumption percentage according to different power modes.
  - `int currentLevel :` current mode selected on the device. Default is 0.
  - `boolean noisy:` indicates if the device is noisy or not.
- **Constructors**:
  - `public Appliance(int id, String name, double maxPowerConsumption, int[] powerLevels, boolean noisy)`
  - `public Appliance(int id, String name, double maxPowerConsumption, boolean critical, int[]powerLevels, boolean noisy)`
- **Methods**:
  - `Getters and setters for all data fields`
  - `turnOn():` overides the turnOn method to turn on the appliance on power level 0.
  - `turnOn(level: int):` overloads the turnOn method to turn on the appliance on a specific power level.
  - `getCurrentConsumption()`: if the appliance is on, returns the real power consumption of the appliance (according to the selected mode and its percentage in the array of power levels and maximum power consumption) . 0 otherwise.
  - `String toString()`: returns a string representation of the device following format: Appliance{ id=[id] , name=[name], [status: On/ Off/Standby], maximum power consumption = [maxPowerConsumption], [critical/not critical],power Levels = {[powerlevel0, powerLevel1, …]} , level = [currentLevel], [noisy/not noisy] }

---

3. Room Class

A room object will represent one room in the house, to which a list of devices is associated.

- **Attributes**:
  - `String code:` unique code for each room. For example : “K1F”
  - `String description:` the description of the room for example “Kitchen - 1rst Floor”.
  - `ArrayList<Device> devicesList`: ArrayList including all the devices in the room.
- **Constructors**:
  - `public Room(String code, String description)`
- **Methods**:
  - `Getters and setters for all data fields`
  - `getNbLights():` returns the number of light devices in the room.
  - `getNbAppliances():` returns the number of Appliance devices in the room.

- `getCurrentComsuption ()`: returns the power currently consumed by all devices in the room.
- `addDevice(Device d)`: adds new device to the list of devices.
- `removeDevice(Device d)`: removes a given device from the list of devices.
- `searchDeviceById(int id)`: searches and returns a device from the list of devices according to its id. If the device does not exist in the list the methods returns null.
- `String toString()`: returns a string representation of the room with all the details of all devices.
- `String toBreifString()`: returns a string representation of the room with only the total number of devices and the number of lights and appliances in the room.

---

4. ManagementSystem Class

The ManagementSystem class is the class responsible of main operations in the smart home management system. The smart home system has 2 modes of operation: Settings and Control which requires 2 different passwords, admin password and user password. This class includes all information and data about the management system as well as the required methods for user interaction, basic setup and control. Devices can be managed (add/delete) by the admin and controlled (turn on/ turn off/ standby) by the user. The pre-set time mode (for noisy appliances) and power mode (for power consumption) must be respected at any moment. For this reason two waiting lists of standby devices are created (ArrayList). One for devices that needs to be turned on once the power consumption allows it and one for the noisy appliances that needs to be turned on once the day time is set. Check the following section and the notes at the end of this document for more information about the system functionalities.

- **Attributes**:
  - `String amdinPassword`: a String representing the admin password. It must include at least 8 characters with at least one upper case character, one lower case character and one digit.
  - `String userPassword`: a String representing the user password. It must include at least 8 characters with at least one upper case character, one lower case character and one digit.
  - `ArrayList<Room> rooms`: ArrayList including all the rooms in the system.
  - `double maxAllowedPower`: represents the maximum allowed power consumption. The power mode can be defined as one of 3 power modes LOW with max allowed power equal to 1000. NORMAL, which is the default mode, with max allowed power equal to 4000 and HIGH with max allowed power equal to 10000. Define the 3 modes as public static variables.
  - `boolean day`: represents the time mode. True for day time and false for night mode.

- `ArrayList<Device> waitingListDay`: ArrayList including all the devices put on standby until day time is set.
- `ArrayList<Device> waitingListPower`: ArrayList including all the devices put on standby until the power consumption allows it.

- **Constructors**:
  - `ManagmentSystem(String adminPassword, String userPassword)`
- **Some Methods and functionalities**:
  - `Setters for adminPassword and userPassword.`
  - `changeAdminPassword`
  - `changeUserPassword`
  - `displaySummaryAllRooms`
  - `displayDetailsOneRoom`
  - `addRoom`
  - `addDevice`
  - `removeRoom`
  - `removeDevice`
  - `setDayTime`
  - `setNightTime`
  - `turnOnDevice`
  - `turnOffDevice`
  - `shutDownOneRoom`
  - `shutDownAllDevices`
  - `searchRoomByCode`
  - `searchDeviceById`
  - `displayInfo`
  - You can add any method that you think useful for your application, for example: `run, displayMainMenu, displayAdminMenu, displayUserMenu…etc.`

5. More about the Application:

Your application must be user friendly. When lunched a main menu should be displayed where the user can: enter admin mode, enter user mode or exit. Below are the main functionalities for each mode as well as some important notes.

A. In admin mode the user must be able to:

1. Change admin and user passwords
2. Change power mode to one of the three possible modes
3. Set day/time mode
4. Add/Delete/Search a room
5. Add/Delete/Search a device
6. Exit admin mode

B. In control mode the user must be able to:

1. Check all rooms info
2. Check all devices info

3. Check all running devices
4. Check all standby devices in the day waiting list
5. Check all standby devices in the power waiting list
6. Search for a given room
7. Search for a given device
8. Turn on/ Turn off a device
9. Turn off all devices from one specific room
10. Turn off all devices in the house
11. Check current power consumption
12. Set day/night mode
13. Exit control mode

Important notes:

1- To turn on a device the user must select the room and then the device by providing the room code and device id. If the power consumption constraint allows it, and the device is not noisy or we are in day mode the device should be turned on. If the device is noisy and the system is set to night mode the user should receive a warning message and be able to either turn on the device anyway, put the device in standby mode and add it to the day waiting list or cancel the order.
2- At all moments, the total power consumption of all running devices must be less than the maximum allowed power by the system's power mode (LOW, HIGH or NORMAL). If a device is to be turned on and the maximum allowed power in not enough the user should have the option to add the device to the power waiting list or to cancel the order.
3- When one device is turned off the system should check for standby devices in the power waiting list and turn on any device(s) that can now be running without the violation the power constraint. The devices must be turned on according to their chronological order.
4- When night time is set the system should check for any running, noisy devices and ask the user if the device should be set off, standby in the day waiting list, or kept on.
5- When day time is set, all devices in the day waiting list should be turned on, unless restricted by the power consumption. In this cases the device is removed from the day waiting list and added to the power waiting list.
6- When day time is set, the user should be asked if he wants to turn off all the lights in the house or not.
7- If the user requires the turn off of any critical device at any moment the system should ask the user to double confirm the order by entering the admin password before shutting off the device.
8- When the user exit the admin mode or user mode the main menu should be displayed again.