HACETTEPE UNIVERSİTY
COMPUTER SCIENCE

BBM104 ASSIGNMENT 2
SMART HOME SYTEM

Latife Üstün
2220356121

INDEX:

Define the Problem:

The problem statement for this project is to control smart home accessories, which include Smart Lamp (with white-ambiance and color-white-ambiance variants), Smart Plug, and Smart Camera, based on commands received. The devices must always be sorted in ascending order according to their switch times, with the initial order preserved for devices that have the same switch time. If a device does not have a switch time, it is considered to be greater than all other devices. At initialization, each device is switched off and has no switch time unless specified otherwise. Additionally, changing the status of a device will delete its switch time information. The goal is to develop a code that can handle these requirements and ensure efficient and effective control of the smart home system.

Description of Solution:

In this project, a class was created for each type of device as there may be multiple devices for each device type. This decision was made to ensure modularity and maintainability of the code. Each class encapsulates the properties and behaviors of its respective device type, and has methods to set and get the switch time and status of the device.In order to achieve the desired functionality, a separate class was created for each smart home accessory. The necessary fields and methods were defined within each class to handle the control and time management of the respective device.In the main class, the incoming input was read and passed through checks based on the received commands. For example, whether the device already exists or not, or if the command is invalid. If the command is valid, the necessary methods were called to perform the desired command.
***For the device addition part*,** an object of the device type was created for each device to be added. In order to prevent the same device from being added again, lists were created for each device type that contained the names of the devices. Before each device addition operation, it was checked whether the device already exists in the corresponding list.
In the device removal part, it was checked if the device exists in the list of device names, and if the device exists, its object and name were removed.
For the change name method, the same approach was followed: the existence of the device was checked by searching the list of device names, and if the device exists and the new name is not already taken by another device, the name was changed.
For the Switch command, the current status of the device was checked. If the desired status was not equal to the device's current status, the device's command was changed. This was done by using the Switch On/Off method inside the device. This method compared the device's previous status with the desired status and made the necessary changes to the device's status.The 'switch on/off' method controls the time when a separate device is intended to be turned on and when it is not intended to be turned off."
For camera class the "switchOnOff" method is used to turn the camera on or off, and it takes in a string argument to indicate the desired action ("On" or "Off") and a

LocalDateTime object to indicate the time at which the action is taken. If the camera is switched off, the "usedStorage" method is called to calculate the storage used during the time it was switched on. For plug class the switchOnOff method allows for the SmartPlug to be switched on or off, depending on the input argument. If the input argument is "On" and the SmartPlug is already switched on, the method returns an error message. Similarly, if the input argument is "Off" and the SmartPlug is already switched off, the method returns an error message. If the SmartPlug is switched on or off, the method records the time of the switch.
The "Plugin"and "Plugout" command changes the state of the device being plugged in or not, which is implemented by the "plugIn" method in the "SmartPlug" class. Depending on the request, this method changes the state of the plug accordingly.
Set kelvin method sets the Kelvin value of a smart lamp or a color lamp based on the input command. It first checks if the command has three arguments and if the device name is valid. If the device name is valid, it sets the Kelvin value of the device to the input value. If the device is not a smart lamp, an error message is written. If the device name is invalid, an error message is also written.
Set brightness method checks the input command to see if it contains three arguments. If it doesn't, it writes an error message to the output. If the command has three arguments, it checks if the device specified in the second argument is a smart lamp or color lamp by searching through the device lists. If the device is found, it sets the brightness value of the lamp to the value specified in the third argument. If the device is not found or is not a smart lamp or color lamp, an appropriate error message is written to the output.
Set color code method sets the color code value of a smart color lamp device, and it checks whether the user input commands are valid and whether the specified device exists in the list of registered smart color lamps. If the specified device is a smart color lamp, it updates the color code value of the device, and if not, it returns an error message.
Set white method is responsible for setting the color temperature and brightness of a smart lamp or smart color lamp device. The user provides the device name, color temperature, and brightness as input through the command line. If the device is found and is a smart lamp or smart color lamp, the code sets the white light settings to the provided values. If the input command is incorrect or the device is not found, an appropriate error message is written to the output.
Set color method sets the color and brightness of the specified smart color lamp. It takes input in the form of device name, color code, and brightness. If the specified device is not a smart color lamp, an error message is printed. If the command index is out of bounds, an error message is printed.
Set time method sets the system time to the given time and updates the status of the devices according to their scheduled switch times. If the given time is before the current time, the devices are switched on or off according to their schedule. If the given time is the same as the current time, nothing is changed. If the given time is after the current time, an error message is printed. The code also handles errors related to incorrect time format.
Skip Minutes method skips a specified number of minutes from the current time and updates the status of devices based on their switch times if necessary. The method takes an ArrayList of Strings as input containing the command and its arguments. It iterates over a switch time list and checks if the initial time is after the switch time. If it is, it updates the status of the devices accordingly. If the specified number of minutes is negative, it returns an error message. If the input command is erroneous, it also returns an error message.

Switch time method sets the switch time for a device and adds the switch time to the switchTimeList. It also updates the switch time for the corresponding device in the allDevices list. The function takes in a list of commands entered by the user and throws a DateTimeParseException if the date format is incorrect. If the device is not found in the plugList, cameraList, lampList, or colorLampList, an error message is printed. If the switch time is in the past, an error message is also printed.
Nop sets the switch time of a device and adds it to a list of switch times. It checks if the switch time is in the past and if so, prints an error message. It then sorts the list of switch times, finds the next switch time after the initial time, and switches the status of the corresponding device. It also performs some device-specific operations and removes the switch time from the list of switch times. If there are no switch times in the list, it prints an error message.
Zreport method prints information about all devices in the system (smart cameras, smart lamps, smart color lamps, and smart plugs) using their respective methods for generating device information. It first writes the initial time of the system and then iterates through the list of devices, calling their specific methods for generating information and writing them to the output file.

Benefits of OOP:

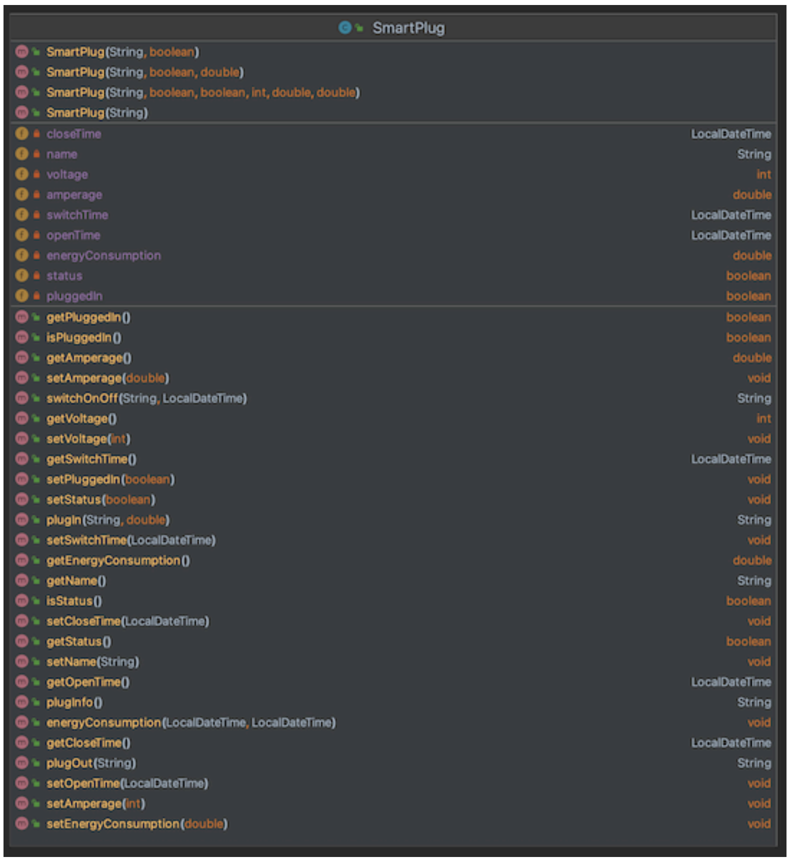
Class structures were used to avoid the need to store device properties in separate variables for each instance of the same type of device, and to reduce code redundancy by utilizing methods defined within the classes.The properties of multiple instances of the same type of device were not required to be stored in separate variables by using class structures, and code repetition was minimized through the use of methods contained within the classes.By utilizing class structures, the necessity of storing device properties in distinct variables for each device instance of the same type was eliminated, and the repetition of code was reduced through the implementation of methods within the classes.This approach also facilitates making changes directly to the class in case a new feature needs to be added to the device type, which can be very helpful for large systems.

The four pillars of OOP are:

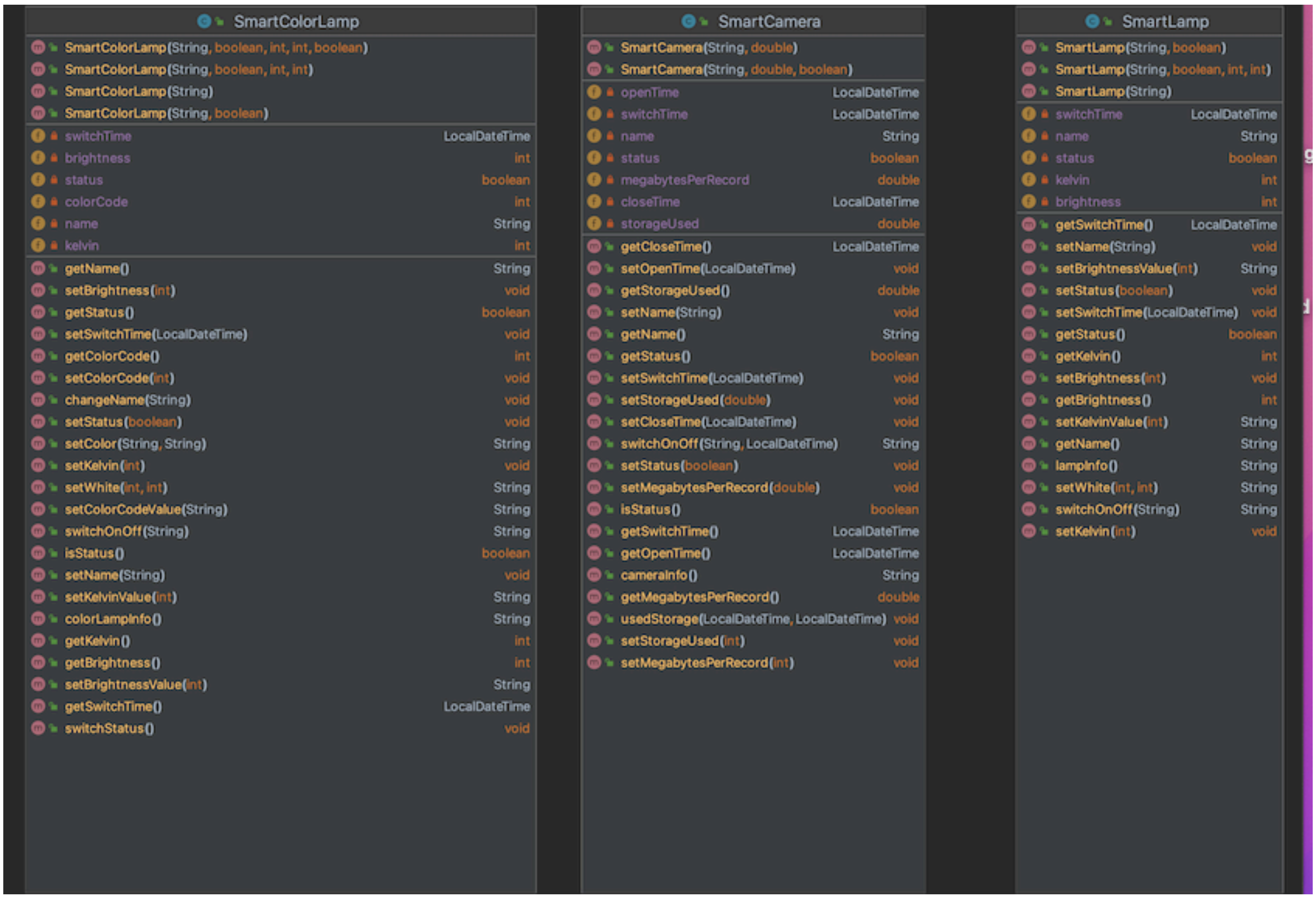
1. Encapsulation: This refers to the practice of hiding the internal details of an object and exposing only the relevant information through well-defined interfaces. This helps to improve the security and maintainability of the code.
2. Abstraction: This involves identifying the essential characteristics of an object and ignoring the rest, in order to simplify the programming process. Abstraction allows us to create complex systems by breaking them down into simpler, more manageable components.
3. Inheritance: This allows us to create new classes that inherit the properties and methods of existing classes, thus reducing code duplication and improving code reuse.
4. Polymorphism: This refers to the ability of objects to take on different forms or behaviors depending on the context in which they are used. Polymorphism allows us to write more flexible and reusable code.

UML (Unified Modeling Language) is a standardized graphical language used to represent software designs. It includes a variety of diagrams that can be used to model different aspects of a system, such as use cases, class structures, and interactions between objects. UML is used by developers to communicate and document their designs, and it helps to ensure that everyone involved in a project has a clear understanding of how the system is intended to work.

UML DIAGRAM:



```
SmartPlug
SmartPlug(String, boolean)
SmartPlug(String, boolean, double)
SmartPlug(String, boolean, boolean, int, double, double)
SmartPlug(String)
closeTime                                                LocalDateTime
name                                                            String
voltage                                                            int
amperage                                                        double
switchTime                                               LocalDateTime
openTime                                                 LocalDateTime
energyConsumption                                               double
status                                                         boolean
pluggedIn                                                      boolean
getPluggedIn()                                                 boolean
isPluggedIn()                                                  boolean
getAmperage()                                                   double
setAmperage(double)                                               void
switchOnOff(String, LocalDateTime)                              String
getVoltage()                                                       int
setVoltage(int)                                                   void
getSwitchTime()                                          LocalDateTime
setPluggedIn(boolean)                                             void
setStatus(boolean)                                                void
plugIn(String, double)                                          String
setSwitchTime(LocalDateTime)                                      void
getEnergyConsumption()                                          double
getName()                                                       String
isStatus()                                                     boolean
setCloseTime(LocalDateTime)                                       void
getStatus()                                                    boolean
setName(String)                                                   void
getOpenTime()                                            LocalDateTime
plugInfo()                                                      String
energyConsumption(LocalDateTime, LocalDateTime)                   void
getCloseTime()                                           LocalDateTime
plugOut(String)                                                 String
setOpenTime(LocalDateTime)                                        void
setAmperage(int)                                                  void
setEnergyConsumption(double)                                      void
```



```
SmartColorLamp
SmartColorLamp(String, boolean, int, int, boolean)
SmartColorLamp(String, boolean, int, int)
SmartColorLamp(String)
SmartColorLamp(String, boolean)
switchTime                                  LocalDateTime
brightness                                            int
status                                            boolean
colorCode                                             int
name                                               String
kelvin                                                int
getName()                                          String
setBrightness(int)                                   void
getStatus()                                       boolean
setSwitchTime(LocalDateTime)                         void
getColorCode()                                        int
setColorCode(int)                                    void
changeName(String)                                   void
setStatus(boolean)                                   void
setColor(String, String)                           String
setKelvin(int)                                       void
setWhite(int, int)                                 String
setColorCodeValue(String)                          String
switchOnOff(String)                                String
isStatus()                                        boolean
setName(String)                                      void
setKelvinValue(int)                                String
colorLampInfo()                                    String
getKelvin()                                           int
getBrightness()                                       int
setBrightnessValue(int)                            String
getSwitchTime()                             LocalDateTime
switchStatus()                                       void
```

```
SmartCamera
SmartCamera(String, double)
SmartCamera(String, double, boolean)
openTime                                    LocalDateTime
switchTime                                  LocalDateTime
name                                               String
status                                            boolean
megabytesPerRecord                                 double
closeTime                                   LocalDateTime
storageUsed                                        double
getCloseTime()                              LocalDateTime
setOpenTime(LocalDateTime)                           void
getStorageUsed()                                   double
setName(String)                                      void
getName()                                          String
getStatus()                                       boolean
setSwitchTime(LocalDateTime)                         void
setStorageUsed(double)                               void
setCloseTime(LocalDateTime)                          void
switchOnOff(String, LocalDateTime)                 String
setStatus(boolean)                                   void
setMegabytesPerRecord(double)                        void
isStatus()                                        boolean
getSwitchTime()                             LocalDateTime
getOpenTime()                               LocalDateTime
cameraInfo()                                       String
getMegabytesPerRecord()                            double
usedStorage(LocalDateTime, LocalDateTime)            void
setStorageUsed(int)                                  void
setMegabytesPerRecord(int)                           void
```

```
SmartLamp
SmartLamp(String, boolean)
SmartLamp(String, boolean, int, int)
SmartLamp(String)
switchTime                                  LocalDateTime
name                                               String
status                                            boolean
kelvin                                                int
brightness                                            int
getSwitchTime()                             LocalDateTime
setName(String)                                      void
setBrightnessValue(int)                            String
setStatus(boolean)                                   void
setSwitchTime(LocalDateTime)                         void
getStatus()                                       boolean
getKelvin()                                           int
setBrightness(int)                                   void
getBrightness()                                       int
setKelvinValue(int)                                String
getName()                                          String
lampInfo()                                         String
setWhite(int, int)                                 String
switchOnOff(String)                                String
setKelvin(int)                                       void
```