HACETTEPE UNIVERSITY COMPUTER ENGINEERING DEPARTMENT

2023 Spring Term Assignment 2

Smart Home System

NAME: Zeynep Nisa KARATAŞ

NUMBER: 2210356066

INSTRUCTOR: Görkem AKYILDIZ

b2210356066@cs.hacettepe.edu.tr

CONTENTS

PROBLEM

In today's world, people always seek ways to make their lives easier, even if it means doing complex jobs. Automation is one of the things that have been invented for this purpose. This project aims to develop a smart home system that includes Smart Lamp, Smart Lamp with Color, Smart Plug, and Smart Camera to provide ease to the user's life. The system will obey the four pillars of Object-Oriented Programming (OOP): Inheritance, Polymorphism, Abstraction, and Encapsulation.

The system is an autonomous system that will execute commands given by the user while controlling the time when no command is given. The devices in the system must always be in ascending order according to their switch times. If two or more devices have the same switch time, their initial order concerning each other will be preserved while sorting. In case a device does not have any switch time, it will be considered greater than all the other devices. By default, each device is switched off, and it has no switch time at initialization.

The Smart Lamp is a lamp with adjustable Kelvin and brightness values in the range of [2000K, 6500K] and [0%, 100%], respectively. Both Kelvin and brightness are integers with default values of 4000K and 100%, respectively.

The Smart Lamp with Color is similar to the Smart Lamp, with an additional feature of having a color mode. The color code is in the range of [0x000000, 0xFFFFFF] and is an integer in hexadecimal base. The Smart Lamp with Color has two modes: color mode and white mode. If the color mode is activated, the Kelvin value is not essential. Otherwise, the color code is not important. The default value for this lamp is the same as the Smart Lamp (4000K, 100%).

The Smart Plug is a wall plug that can calculate its total energy consumption. The system will update the consumption at events such as switching on/off or plugging in/out. Note that switching off the Smart Plug does not mean that it is unplugged. The default voltage value for the Smart Plug is 220 Volts, and the ampere value is in positive number format. Initially, there is nothing plugged into the Smart Plug.

The Smart Camera provides information about its storage usage. The system will update the storage information at events such as switching on/off. The number of megabytes per minute is in non-negative number format.

In conclusion, this project aims to develop a smart home system that includes Smart Lamp, Smart Lamp with Color, Smart Plug, and Smart Camera. The system will obey the four pillars of OOP, and the devices will always be in ascending order according to their switch times. The Smart Lamp and Smart Lamp with Color have adjustable Kelvin and brightness values, while the Smart Lamp with Color has an additional color mode. The Smart Plug can calculate its total energy consumption, and the Smart Camera provides information about its storage usage.

SOLUTION APPROACH

The first step is to parse the user inputs and validate them. The program must check whether the input is in the correct format and contains the required parameters. If the input is not in the correct format or contains missing parameters, an appropriate error message will be displayed.

The SetInitialTime command is the first command that the program expects to receive. This command sets the initial time of the system. The program must validate the input, check if the time is within the valid range, and set the system time accordingly.If the SetInitialTime command is not in first line of input file it won’t continue it’s work and this command will only work when its in the first line.

The SetTime command sets the time of the system. This command must be used after the SetInitialTime command. The program must validate the input, check if the time is within the valid range, and set the system time accordingly. If time format is not correct or the time that we get is before the current time it will print error message.

The SkipMinutes command skips the given number of minutes. The program must validate the input, check if the minutes are within the valid range, and add the given number of minutes to the system time.

The Nop command does nothing but waits until the next switch event occurs. If there is no switch event, the program must display an appropriate error message.

The program must provide the ability to add new devices to the system using the Add command. The program must validate the input, check if the device name is unique, and add the device to the system. The program must also display a success message if the device is added successfully.

The program must provide the ability to control the devices using specific commands. For example, the user can set the brightness of a color lamp or the megabyte per minutes of an SmartCamera. The program must validate the input, check if the device exists in the system, and perform the required operation on the device.

In conclusion, the smart home automation system is developed using Java programming language. The system includes multiple smart devices such as smart lamps,smart plugs and smart cameras, , which can be controlled by a central control unit using specific commands. The program validates the user inputs, checks if the commands are within the valid range, and performs the required operations on the devices. The program provides a simulation of a real-world smart home automation system.

PROBLEMS AND SOLUTIONS

I faced plenty of problems in this project since there are a lot of possiblities for each command.Mostly I struggled with Nop command .In Nop command we were expected to switch status of a certain device by its switch time.At first I record all the switch times in a list and in a map I stored the devices will be switched as keys and switch times as values.I get time values one by one and if the closest time was found I added that time into another list and after finishing the for loop I removed the times that will be switched from time list and also I removed the entry that has the value of the times that will be switched.

BENEFITS OF THE SYSTEM

There are plenty of benefits of this system.

Convenience: With smart home accessories, users can control various devices from a central hub or even remotely from their smartphones, making it more convenient to manage their home environment.

Energy savings: Smart home devices can be programmed to turn off or reduce energy consumption when not in use, potentially leading to lower energy bills and reduced environmental impact.

Increased security: Smart cameras and other security devices can provide real-time monitoring and notifications, allowing homeowners to take action in the event of a security breach or emergency.

Better health and wellbeing: Smart lamps can be adjusted to mimic natural lighting patterns, which can have a positive impact on sleep quality and overall health.

Data insights: Smart home devices can provide valuable data on energy consumption, occupancy patterns, and other aspects of home life, which can be used to optimize systems and improve overall efficiency.

Overall, a well-designed smart home system can provide a range of benefits to homeowners, making it a worthwhile investment for those looking to improve their quality of life and reduce their environmental impact.

BENEFITS OF OOP

There are many benefits of object-oriented programming (OOP), some of which are:

Reusability: OOP allows the creation of reusable code modules that can be used in different parts of an application or in different applications altogether. This means that developers don't have to start from scratch every time they need to create a new program.

Modularity: OOP promotes modular design, which means that complex systems can be broken down into smaller, more manageable parts. This makes it easier to understand and maintain the code.

Encapsulation: OOP encapsulates the data and the code that operates on that data, which means that the internal workings of an object can be hidden from other objects. This makes the code more secure and reduces the likelihood of errors caused by unintended changes to the data.

Polymorphism: OOP allows the creation of polymorphic objects, which can take on different forms depending on the context in which they are used. This makes the code more flexible and adaptable to changing requirements.

Inheritance: OOP allows the creation of classes that inherit properties and methods from other classes. This makes it easier to create new classes that are similar to existing ones, and reduces the amount of code that needs to be written.

Better code organization: OOP encourages the use of well-defined classes and objects, which makes the code more organized and easier to understand. This makes it easier to maintain and modify the code over time.

In conclusion, OOP provides a powerful set of tools for creating flexible, modular, and maintainable software applications

FOUR PILLARS OF OOP AND UML

Object-Oriented Programming (OOP) is a programming paradigm that uses the concept of objects to represent and manipulate data. OOP focuses on creating of reusable code, modularity, and flexibility. There are four main pillars of OOP that are important for creating well-organized and easily maintainable code: Inheritance, Polymorphism, Abstraction, and Encapsulation.

Abstraction: Abstraction is the process of focusing on essential features of an object while ignoring the non-essential details. It allows us to the creation of complex systems by breaking them down into smaller, more manageable components. Abstraction is achieved through abstract classes and interfaces.

Encapsulation: Encapsulation is the practice of hiding the internal details of an object from the outside world, and only exposing the necessary information through methods and interfaces. Encapsulation allows for better control of access to an object's properties and methods, and protects the object from unintended modifications.

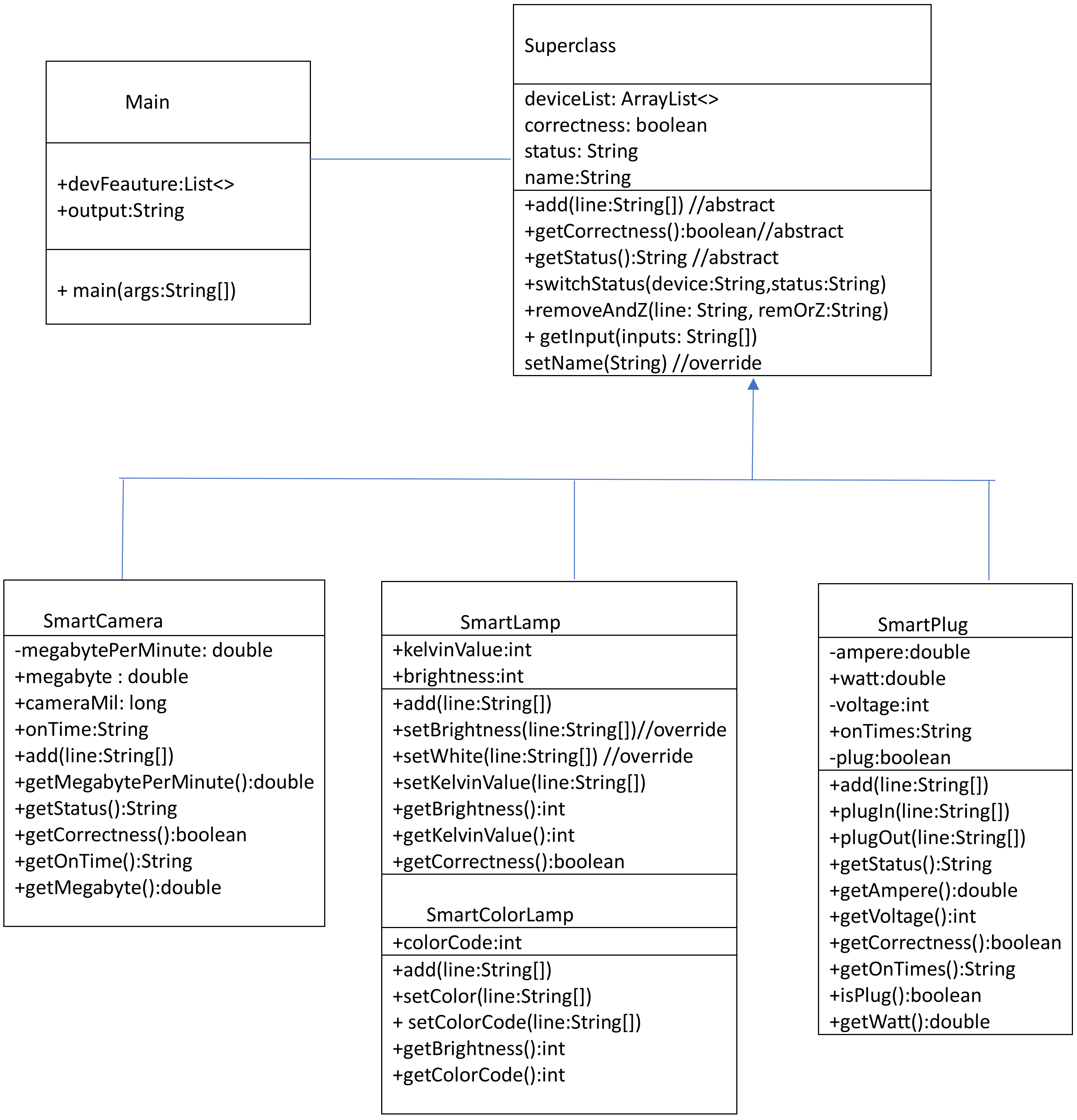
Inheritance: Inheritance is the process of creating new classes from existing classes. It allows a new class to reuse the variables,properties and methods of an existing class. Inheritance allows us to the creation of hierarchical structures that are based on common characteristics, making the code more organized and easier to maintain.

Polymorphism: Polymorphism is the ability of an object to take on multiple forms. It enables the same method or operator to be used with different objects, allowing for greater flexibility and reusability of code. Polymorphism is achieved through method overriding, method overloading, and interfaces.

Unified Modeling Language (UML) is a standardized visual language used to model and design software systems. It provides a set of tools and notations for creating diagrams that represent various aspects of a system. UML diagrams are used to make communication easier among stakeholders and to provide a common language for software developers.There are several types of UML diagrams ,including:Class diagrams,Sequence diagrams,Use case diagrams

In summary, the four pillars of OOP (Inheritance, Polymorphism, Abstraction, and Encapsulation) are essential concepts for creating well-organized and easily maintainable code. UML is a standardized visual language that provides a set of tools and notations for creating diagrams that represent various aspects of a software system. Together, they provide a framework for designing, implementing, and communicating software systems.

UML DIAGRAM OF MY OOP Design


Main
+devFeauture:List<>
+output:String
+ main(args:String[])
Superclass
deviceList: ArrayList<>
correctness: boolean
status: String
name:String
+add(line:String[]) //abstract
+getCorrectness():boolean//abstract
+getStatus():String //abstract
+switchStatus(device:String,status:String)
+removeAndZ(line: String, remOrZ:String)
+ getInput(inputs: String[])
setName(String) //override
SmartCamera
-megabytePerMinute: double
+megabyte : double
+cameraMil: long
+onTime:String
+add(line:String[])
+getMegabytePerMinute():double
+getStatus():String
+getCorrectness():boolean
+getOnTime():String
+getMegabyte():double
SmartLamp
+kelvinValue:int
+brightness:int
+add(line:String[])
+setBrightness(line:String[])//override
+setWhite(line:String[]) //override
+setKelvinValue(line:String[])
+getBrightness():int
+getKelvinValue():int
+getCorrectness():boolean
SmartColorLamp
+colorCode:int
+add(line:String[])
+setColor(line:String[])
+ setColorCode(line:String[])
+getBrightness():int
+getColorCode():int
SmartPlug
-ampere:double
+watt:double
-voltage:int
+onTimes:String
-plug:boolean
+add(line:String[])
+plugIn(line:String[])
+plugOut(line:String[])
+getStatus():String
+getAmpere():double
+getVoltage():int
+getCorrectness():boolean
+getOnTimes():String
+isPlug():boolean
+getWatt():double