COVER

Hacettepe University: Computer Engineering

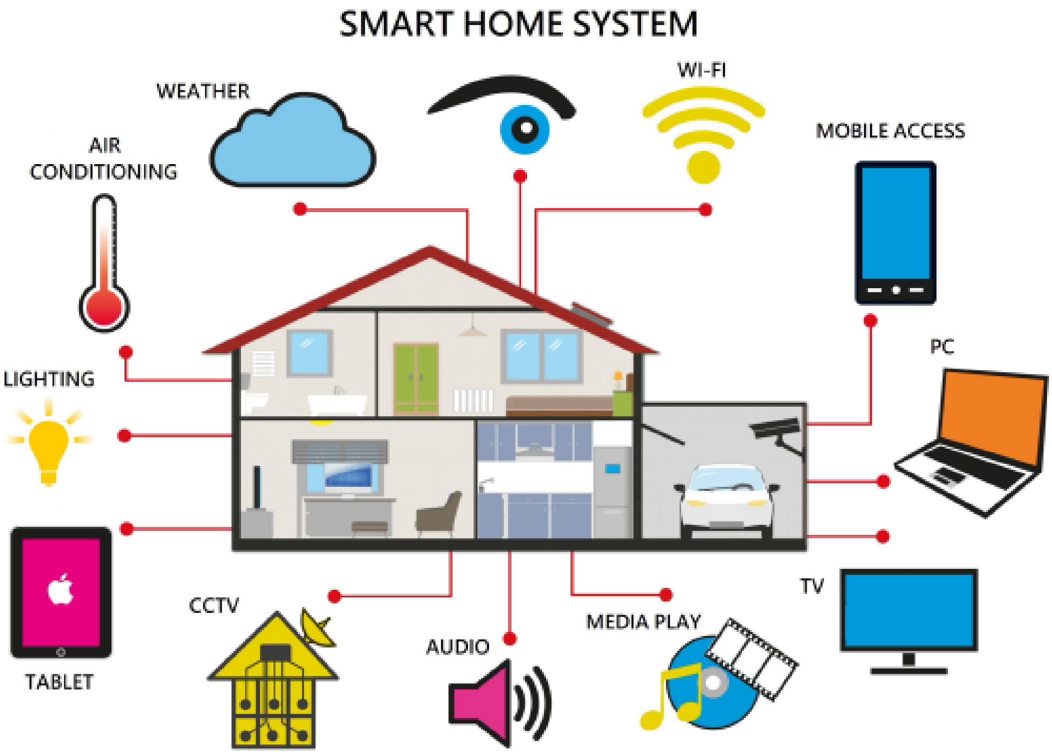
BBM102: Introduction to Programming Laboratory II

Assignment 2: Smart Home System

b2210356045

Berke Taşkın

21.04.2023



Indexing

Main Problem

Humankind is a lazy race unlike most animals. They always tried to do things with less effort. As a result of that, we invented the term "Automation". A source states that automation is the use of technology to perform tasks where human input is minimized. (This source is given in the reference part.) We use smart home systems in order to automate our lives. Complex systems make our lives easier and that is the irony.

Solution Approach

This project uses OOP logic in order to create a basic smart home system. There are 4 different types of smart device in this system. Smart Lamp, Smart Lamp with Color, Smart Plug, and Smart Camera are these devices. Number of devices was kept low since this system needs to be basic. Devices and time values will be controlled over commands. Each device will have their own switch time where they will automatically switch their status. Devices will be sorted according to their switch times in ascending order. If two devices have the same switch time, their initial order will be kept. No switch time means greater value in this sorted list. By default, each device is switched off and they don't have any switch time. Moreover, if the status of certain device changes, switch time information gets deleted.

Problems & Solutions (of mine)

I faced certain problems while writing this code as expected. It was hard for me to think of an OOP design as it is a new idea for all of us. It needed lots of planning before beginning to write the code. Each class has its own mission to accomplish in order to make this system run efficiently. There were some easy classes to be created but some of them really took lots of thinking. I used some research methods such as using the Internet in order to improve my usage of java. This project took many lines of coding, so the visuality of this code needed to be kept pleasant. Comment lines I used really helped me to achieve this certain visuality.

```
const fetch = require([illegible]
const log = require('./log');
let embed;

function transform([illegible]
  // Promise.resolve [illegible]
  return transformations[illegible]
}

function [illegible]
  return prev.then(($) => {
    $(':header').[illegible]
      const children = [illegible]
      if ($(children).length) {
        $(header).[illegible]
        $(children).[illegible]
      }
      return header;
    });
    return Promise.resolve[illegible]
});
```

Benefits / 4 Pillars of OOP

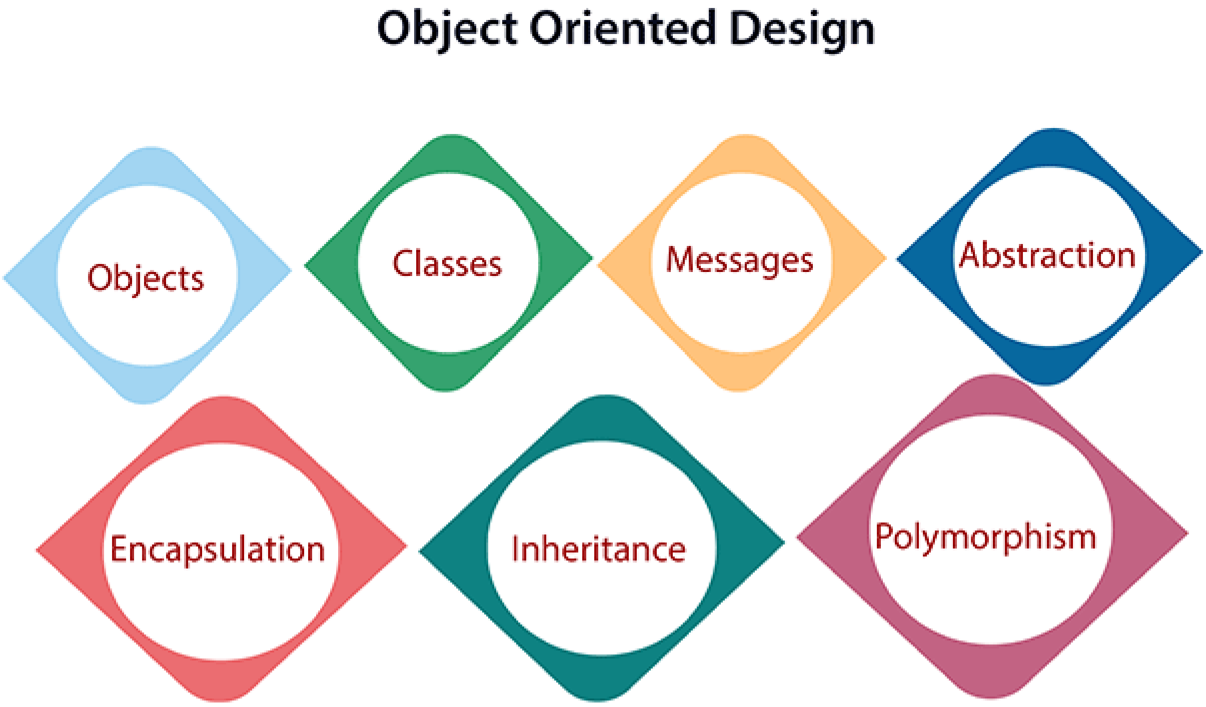
The Smart Home System already has crucial benefits as explained in the problem part. It helps us to waste very little time in order to control electrical devices in our home and it allows us to do what we want in this saved free time. Constructing this kind of a system may be complex but when it is done correctly, it really deletes a lot of complexity in our lives. OOP design basically helps us make this system a non-complex one since each device is an object of its own and it is easy to manipulate them if they are considered an object. OOP design has four pillars which helps us control these objects as we want.

1) **Inheritance**
- Mechanism of an object acquiring all properties of its parent object. Sub-classes also inherit properties from their super-class.

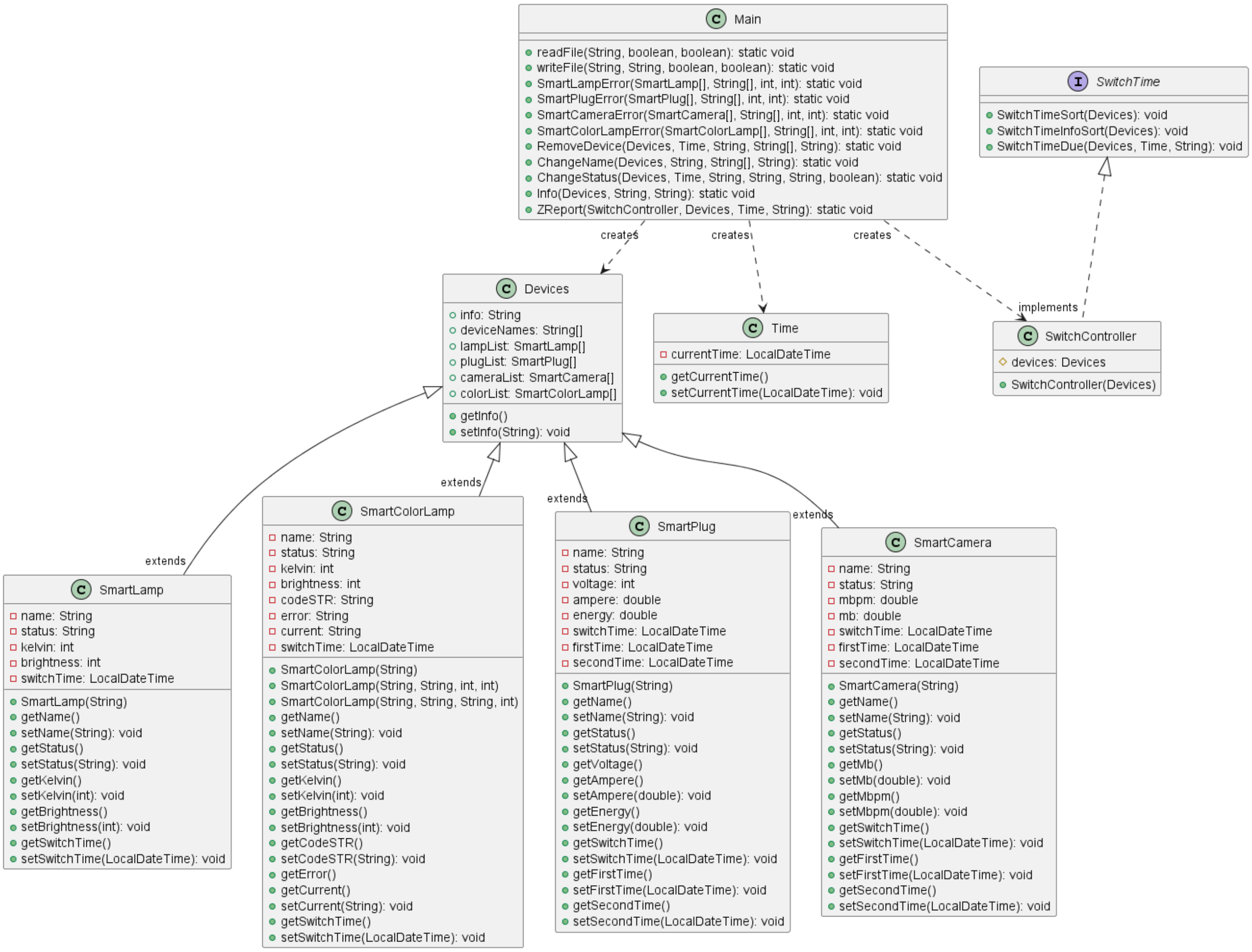
2) **Polymorphism**
- Creating a certain object with different forms using inheritance attributes. It enables objects to perform different tasks.

3) **Encapsulation**
- Setting the variables in a class as private so that data is hidden properly from outside use. It also grants us easy control over data, which makes testing simple.

4) **Abstraction**
- Hiding details about implementation and showing only essential things to the user. Classes can be defined as an abstract class. Interfaces also provide us with abstraction.



UML Diagram


Main
readFile(String, boolean, boolean): static void
writeFile(String, String, boolean, boolean): static void
SmartLampError(SmartLamp[], String[], int, int): static void
SmartPlugError(SmartPlug[], String[], int, int): static void
SmartCameraError(SmartCamera[], String[], int, int): static void
SmartColorLampError(SmartColorLamp[], String[], int, int): static void
RemoveDevice(Devices, Time, String, String[], String): static void
ChangeName(Devices, String, String[], String): static void
ChangeStatus(Devices, Time, String, String, String, boolean): static void
Info(Devices, String, String): static void
ZReport(SwitchController, Devices, Time, String): static void
SwitchTime
SwitchTimeSort(Devices): void
SwitchTimeInfoSort(Devices): void
SwitchTimeDue(Devices, Time, String): void
creates
creates
creates
implements
Devices
info: String
deviceNames: String[]
lampList: SmartLamp[]
plugList: SmartPlug[]
cameraList: SmartCamera[]
colorList: SmartColorLamp[]
getInfo()
setInfo(String): void
Time
currentTime: LocalDateTime
getCurrentTime()
setCurrentTime(LocalDateTime): void
SwitchController
devices: Devices
SwitchController(Devices)
extends
extends
extends
extends
SmartLamp
name: String
status: String
kelvin: int
brightness: int
switchTime: LocalDateTime
SmartLamp(String)
getName()
setName(String): void
getStatus()
setStatus(String): void
getKelvin()
setKelvin(int): void
getBrightness()
setBrightness(int): void
getSwitchTime()
setSwitchTime(LocalDateTime): void
SmartColorLamp
name: String
status: String
kelvin: int
brightness: int
codeSTR: String
error: String
current: String
switchTime: LocalDateTime
SmartColorLamp(String)
SmartColorLamp(String, String, int, int)
SmartColorLamp(String, String, String, int)
getName()
setName(String): void
getStatus()
setStatus(String): void
getKelvin()
setKelvin(int): void
getBrightness()
setBrightness(int): void
getCodeSTR()
setCodeSTR(String): void
getError()
getCurrent()
setCurrent(String): void
getSwitchTime()
setSwitchTime(LocalDateTime): void
SmartPlug
name: String
status: String
voltage: int
ampere: double
energy: double
switchTime: LocalDateTime
firstTime: LocalDateTime
secondTime: LocalDateTime
SmartPlug(String)
getName()
setName(String): void
getStatus()
setStatus(String): void
getVoltage()
getAmpere()
setAmpere(double): void
getEnergy()
setEnergy(double): void
getSwitchTime()
setSwitchTime(LocalDateTime): void
getFirstTime()
setFirstTime(LocalDateTime): void
getSecondTime()
setSecondTime(LocalDateTime): void
SmartCamera
name: String
status: String
mbpm: double
mb: double
switchTime: LocalDateTime
firstTime: LocalDateTime
secondTime: LocalDateTime
SmartCamera(String)
getName()
setName(String): void
getStatus()
setStatus(String): void
getMb()
setMb(double): void
getMbpm()
setMbpm(double): void
getSwitchTime()
setSwitchTime(LocalDateTime): void
getFirstTime()
setFirstTime(LocalDateTime): void
getSecondTime()
setSecondTime(LocalDateTime): void

8 classes and 1 interface are used for this project. Input for this system will be given as "input.txt", and output will be given as "output.txt". OOP implementation of the system is given below.

1)"Main" Class

-The Main class contains the main method and other different methods to process for the Smart Home System.

-11 Methods.

2)"Time" Class

-The Time class represents the time in the Smart Home System.

-It contains the current time value which is updated as the system runs.

-2 Methods.

3)”Devices” Class

-The Devices class represents a collection of smart devices in the Smart Home System.

-It contains arrays to store the name and switch time of each device.

-It also contains separate arrays for each type of device, including SmartLamps, SmartPlugs, SmartCameras, and SmartColorLamps.

-2 Methods.

4)”SmartLamp” Class (extends Devices)

-The SmartLamp class represents a smart lamp device that can be controlled remotely.

-It extends the Devices class and inherits the device-related properties.

-It stores information about the brightness, temperature (in kelvin), and switch time of the device. (Also the name and status.)

-11 Methods.

5)”SmartColorLamp” Class (extends Devices)

-The SmartLamp class represents a smart color lamp device that can be controlled remotely.

-It extends the Devices class and inherits the device-related properties.

-It stores information about the brightness, temperature (in kelvin or color code), and switch time of the device. (Also the name, status, and other string values.)

-18 Methods.

6)”SmartPlug” Class (extends Devices)

-The SmartPlug class represents a smart color lamp device that can be controlled remotely.

-It extends the Devices class and inherits the device-related properties.

-It stores information about the voltage, current (in amperes), energy usage, and switch time of the device. (Also the name, status, and other time values.)

-16 Methods.

7)”SmartCamera” Class (extends Devices)

-The SmartCamera class represents a smart color lamp device that can be controlled remotely.

-It extends the Devices class and inherits the device-related properties.

-It stores information about the total megabyte stored, megabyte per minute, and switch time of the device. (Also the name, status, and other time values.)

-15 Methods.

8)”SwitchController” Class

-The SwitchController class is responsible for controlling the switches of different devices based on their switch time.

-It implements the SwitchTime interface.

-1 Method

9)”SwitchTime” Interface

-The SwitchTime interface defines methods for sorting and managing the switch time of smart devices.

-3 Methods

Resources

-https://www.ibm.com/topics/automation (Definition of Automation)

-https://www.javatpoint.com/inheritance-in-java (Studied and used my own words for Inheritance)

-https://www.w3schools.com/java/java_polymorphism.asp (Studied and used my own words for Polymorphism)

-https://www.javatpoint.com/encapsulation (Studied and used my own words for Encapsulation)

-https://www.javatpoint.com/abstract-class-in-java (Studied and used my own words for Abstraction)

-BBM104_S23_PA2_v1.2.pdf