Hacettepe University

Department of Computer Engineering

BBM104 Assignment 2 Report

Smart Home System

İldeniz ÇELEBİ

b2210356013

21.04.2023

Index:

I. Introduction

II. Problem Definition

III. Solution Approach

IV. Challenges and Solutions

V. Benefits of the System

VI. Benefits of OOP

VII. Four Pillars of OOP and UML

VIII. UML Diagram of OOP Design and Explanation

I. Introduction:

This report describes the development of an Object-Oriented Programming (OOP) system for a business problem. The report covers the problem definition, solution approach, challenges faced, benefits of the system, benefits of OOP, the four pillars of OOP, and a UML diagram of the OOP design.

II. Problem Definition:

The problem was given as controlling smart home accessories including Smart Lamp, Smart Lamp with Color, Smart Plug, and Smart Camera, and managing their switch times and status based on user commands. The accessories have different functionalities and default values, such as adjusting kelvin and brightness values for lamps, calculating energy consumption for plugs, and storage information for cameras. The system sorts the devices based on their switch times in ascending order, and preserves their initial order if multiple devices have the same switch time. The commands include setting the initial time, setting time, skipping minutes, and adding new devices to the system. The program must handle errors for invalid commands and ranges, and provide appropriate error messages. The report is written in passive voice and is justified alignment.

III. Solution Approach:

An abstract class named "Super" was created for the Smart Home System, with the subclasses "SmartPlug," "SmartLamp," "SmartCamera," and "SmartColorLamp" derived from the Super class. These subclasses were designed to create the necessary objects in a smart home and perform the required commands. In addition to these, a class named "Time" was created to set the time, and a "Main" class was implemented.

IV. Challenges and Solutions:

In this assignment, separate devices were designed individually and within their own classes before creating a superclass by identifying common properties. It is thought that a more systematic approach could have been followed if the reverse had been done. The most challenging part was adjusting the timing and creating commands based on time. The research on which tense to use for this was carried out by me.

V. Benefits of the System:

The benefits of this system include the ability to control and automate smart home devices, such as lamps, plugs, and cameras. The system allows for easy adjustment of settings, such as brightness and color temperature, and provides information on energy consumption and storage usage. Additionally, the system can skip time and perform switch operations based on set schedules, providing convenience and efficiency for users. Errors are also detected and reported, ensuring proper functionality and preventing potential damage to devices. The system can be customized with the addition of new smart devices, and existing devices can be easily managed using simple commands.

VI. Benefits of OOP:

Benefits of object-oriented programming (OOP) include improved code organization, increased code reusability, enhanced code maintainability, and the ability to model complex systems. Encapsulation is achieved through the use of classes and objects, which can hide implementation details and provide a clear interface for interacting with the system. Inheritance allows for the creation of new classes that

inherit properties and behaviors from existing classes, reducing the need for redundant code. Polymorphism allows for the creation of classes that can behave differently depending on the context in which they are used. These benefits are widely recognized in the software development industry, and OOP continues to be a popular programming paradigm.

VII. Four Pillars of OOP and UML:

The four pillars of OOP are encapsulation, abstraction, inheritance, and polymorphism.

Encapsulation refers to the concept of hiding implementation details within a class, so that they cannot be accessed or modified from outside the class. This protects the internal workings of the class and makes it easier to change the implementation without affecting other parts of the code.

Abstraction is the process of creating a simplified representation of a complex system or concept. In OOP, this is achieved by defining abstract classes or interfaces that provide a general blueprint for how a class should behave, without specifying the exact implementation details.

Inheritance allows classes to inherit properties and methods from other classes, creating a hierarchy of related classes. This can help to reduce code duplication and improve code organization, as well as making it easier to add new features to existing classes.

Polymorphism refers to the ability of objects to take on different forms, depending on the context in which they are used. This is achieved through methods that can be overridden or overloaded, allowing different behaviors to be defined for the same method name depending on the input parameters or the class of the object.