Hacettepe University Department of Computer Engineering

BBM 104 – Assignment 2 Report

Ahmet Yavuz MUTLU- 2210356014

20.04.2023

INDEXES

1- WHAT IS THE PROBLEM OF THE EVOLVİNG WORLD OF TECHNOLOGY

Nowadays, technology is found all across the world, and it has even begun to go outside its borders. Life has changed, and the pace has accelerated. In today's fast-paced world, keeping up with technology provides several benefits. There is no time for accomplishing basic activities that require a significant amount of time. Those that can adapt and use technology effectively will be able to attain and maintain their goals more quickly. Managing our home in a more controlled way is among the tasks that we can facilitate with the introduction of technology into our lives. After all, we spend more than half of our time in our homes. The question we seek to answer here is: How can we make our homes more livable and enjoyable? This is the starting point for our Smart Home System.

2- ADDRESSING TECHNOLOGICAL CHALLENGES: OUR SOLUTION APPROACH

The Smart Home System is designed to make people's lives easier and allow them to use their valuable time to achieve their goals instead of spending it on daily tasks. With just one touch, it provides control over all smart devices in your home, shapes your home according to your lifestyle, and offers many methods to reduce your expenses by providing usage reports and giving you advice. It offers a solution for having a more enjoyable and livable home. To provide these features and keep the system adaptable to changes, it is necessary to follow the SOLID principles and use object-oriented programming.

3- FACED PROBLEMS AND THEIR SOLUTIONS

The first problem we need to solve is how to design an architecture to control different devices. To solve this problem, we need a class called SmartDevice that will contain all devices. All other smart devices will inherit the properties of this class, such as SmartLamp, SmartPlug, and SmartCamera. Since SmartColorLamp is essentially a lamp with additional features, it will inherit the properties of the SmartLamp class.

The second problem is taking input from the user and checking its validity. To solve this problem, first, we will define a class called Action to understand the inputs coming from the user in a way that can be used in the program, and then we will define a class called MethodClass to call the necessary methods according to these actions. Next, we can create a class called Validations to validate the data coming from the user and write the validation methods inside it, which can be called when necessary. If we receive an invalid input, we need to write a class called DeviceExceptions containing our error types to provide meaningful feedback to the user.

In the final part, using the Actions obtained from the inputs and the MethodClass, we can call the appropriate method in the SmartDeviceController class, which we created to manage all smart devices and contains the necessary methods. We can then display the response to the user.

4- BENEFITS OF THE SMART HOME SYSTEM

Smart Home System provides its users with an easier and more enjoyable life in daily routines. Thanks to its flexibility, you can easily leave your home without worrying if you unplugged your iron or not and have a great time outside. By using the feature to calculate the total energy consumption of each device, you can understand why your bill is high. You can adjust the temperature of your home to your liking before you even arrive. These are just some of the benefits provided by the Smart Home System.

5- BENEFITS OF THE USING OBJECT-ORIENTED PROGRAMMING

Using object-oriented programming helps us make our program sustainable and adaptable to different situations. It allows new features to be added to the program by simply adding new code without changing the old code. It greatly contributes to the readability and efficiency of the program. Finally, it enables the code to be written in a shorter form.

6- PRINCEPLES OF OBJECT-ORIENTED PROGRAMMING AND UML DIAGRAM

The 4 fundamental principles of object-oriented programming are Abstraction, Encapsulation, Inheritance, and Polymorphism.

Abstraction is related to simplifying things. We can write more understandable code by showing the necessary properties in our programs and hiding the others. An example of abstraction from everyday life is driving a car. When we press the gas pedal, the car accelerates, but we don't need to know the internal mechanism of the car.

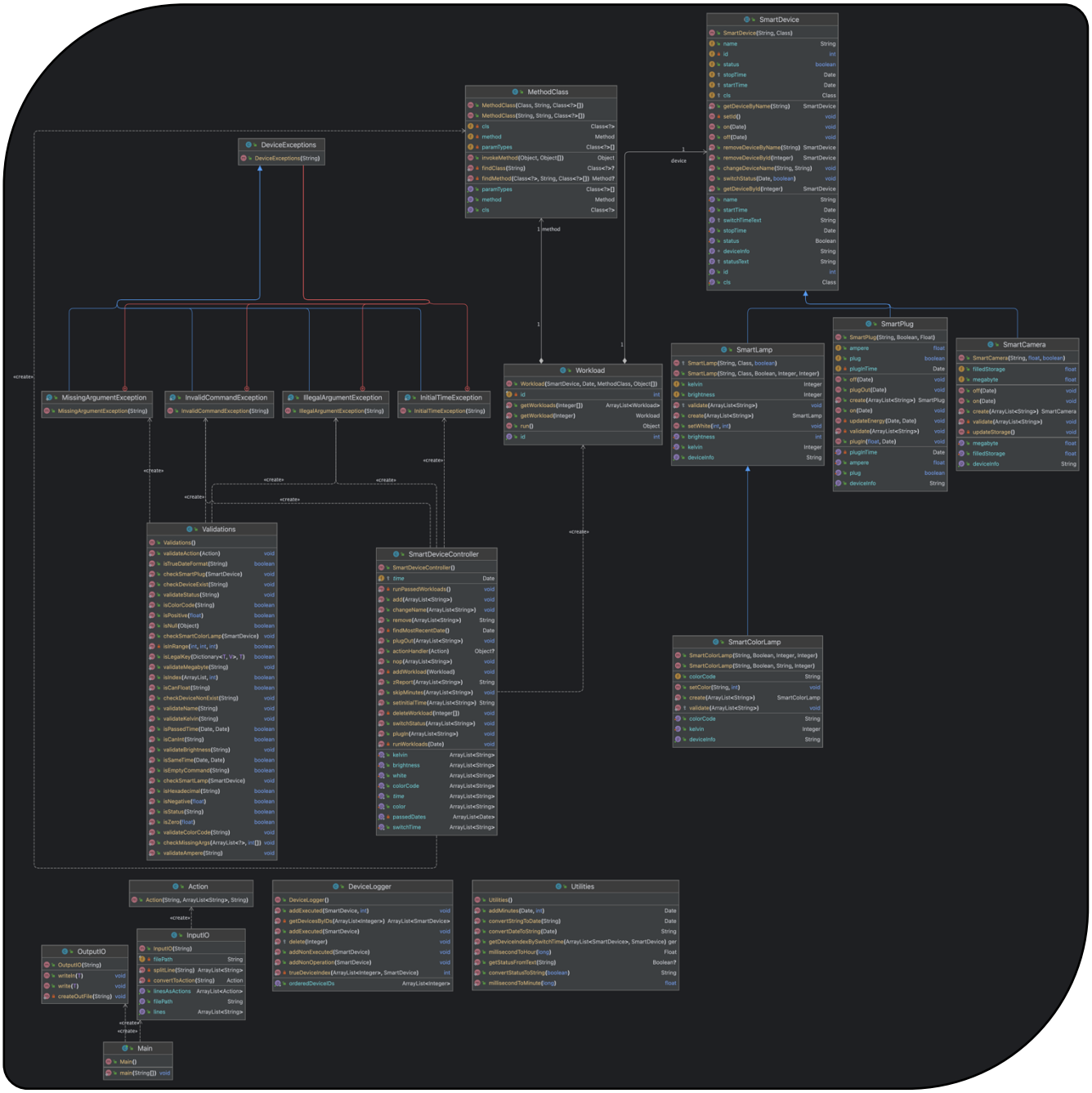
Encapsulation is related to access. We do not want some properties to be accessed from anywhere in our program. This can lead to errors and security vulnerabilities later on. We can make our program more secure by setting the level of access to a property with access specifiers such as private, public, and protected.

Inheritance is about using the inheritance method to write our code more efficiently and concisely. We can explain this with an example. A cat can carry common features of all animals, so it inherits them, but it can also have specific features such as meowing and scratching. Therefore, the cat has inherited from the animal class. We can use the same code we write for an animal for a cat without writing the same code again using the inheritance method.

Polymorphism means multiple forms and helps our code adapt to different situations. It allows us to use the same method in different ways. For example, we can use polymorphism when writing a program that calculates the salaries of employees in different departments of a company. Each department may have different salary calculation methods, but we can combine these methods into a single method to make our code more efficient.

UML stands for Unified Modeling Language and is used as a drawing language for object-oriented programming. UML diagrams visualize different parts and relationships of a program. UML facilitates communication between programmers and clients and helps to better understand the code.

7- UML DIAGRAM OF PROGRAM



The UML diagram of the program is shown above. The diagram includes all methods, constructors, and fields (public, private, and protected). The program starts from the Main class and, with the help of the InputIO class, receives Actions. These Actions are then converted to MethodClass objects to be invoked using the static Dictionary methodMapper in SmartDeviceController. The method in the MethodClass object is then called. The methods in the SmartDeviceController class validate Action parameters before executing the real work using the Validation class methods. If an error occurs, an error message is returned as a response. If all parameters are valid for the Action, the necessary operation is performed for the right SmartDevice object, and a response message is returned. If the user sends an Action to execute later, it goes through the same process, but after validation, SmartDeviceController does not execute the necessary methods. Instead, it creates a Workload object that contains the operation to be executed later. When the execution time comes, the necessary operation is performed using this Workload. After each Action is executed, a response message is shown to the user using the OutputIO class.