Object-Oriented Programming

BLG 252E

Exercise 2

Res. Assist. Mustafa İzzet Muştu
mustu18@itu.edu.tr

Instructors:
Assoc. Prof. Dr. Feza BUZLUCA
Assoc. Prof. Dr. Cihan TOPAL
Assis. Prof. Dr. Sanem KABADAYI

Faculty of Computer and Informatics Engineering
Department of Computer Engineering

1. Introduction

This exercise focuses on developing a smart home automation system that manages IoT devices such as smart lights, thermostats, and security cameras. The system will allow users to add and control these devices, turning them on or off while tracking their power consumption. It will also keep count of the total number of devices in the system.

The smart home system will include a central controller that manages multiple devices, providing functionalities for adding, removing, and displaying their statuses. Each device will have a unique ID, name, type, power consumption rating, and operational status. Devices will be stored within an internal collection and will support dynamic memory allocation for flexible management.

The exercise will demonstrate object-oriented programming concepts, including encapsulation, constructors, destructors, copy constructors, static members, and constant methods.

2. Implementation Details

2.1. SmartDevice Class

This class represents an individual IoT device in the smart home system, managing its identity, status, and power consumption.

2.1.1. Attributes

- ***deviceCount.*** Tracks the total number of devices currently in memory.
- ***name.*** Device name. For this type of data, std::string is used, but char* is required for educational purposes in this study.
- ***type.*** Device type (e.g., "Smart Light", "Thermostat"). For this type of data, std::string is used, but char* is required for educational purposes in this study.
- ***status.*** Indicates whether the device is ON or OFF.
- ***powerConsumption.*** Represents power usage in watts.
- ***usageCount.*** Tracks how many times the device status has been checked.

2.1.2. Methods

- ***Constructor (default).*** Initializes the device with a default name and type, increments the total device count, and sets the status to OFF and power consumption to zero.
- ***Constructor (parameterized).*** Sets custom name, type, and power consumption, initializes the device as OFF, and initializes usageCount.
- ***Copy Constructor.*** Creates a copy of another device, ensuring independent memory allocation for name and type.
- ***Destructor.*** Releases dynamically allocated memory to prevent memory leaks.
- ***togglePower().*** Toggles status between ON/OFF and displays the updated state.
- ***showStatus() const.*** Displays all details of the device, name, type, powerConsumption, and status, increments usageCount.
- ***getUsageCount() const.*** Returns the number of times showStatus() has been called.
- ***getDeviceCount().*** Returns the current number of SmartDevice objects in memory.

- ***getName() const.*** Returns the name of the device.
- ***setName().*** Assigns new name to the device.

2.2. SmartHome Class

This class represents the smart home system that manages multiple IoT devices.

2.2.1. Attributes

- ***homeName.*** Dynamically allocated name of the home.
- ***devices.*** Dynamically allocated array of SmartDevice objects.
- ***deviceIndex.*** Keeps track of how many devices have been added.
- ***maxDevices.*** Defines the maximum number of devices that can be added to the home.

2.2.2. Methods

- ***Constructor (parameterized).*** Assigns homeName, stores maxDevices to define the device limit, dynamically allocates an **array of SmartDevice*** with size maxDevices, and initializes deviceIndex to 0.
- ***Destructor.*** Releases dynamically allocated memory for homeName and devices.
- ***addDevice.*** Adds a new device if deviceIndex is less than maxDevices, uses the SmartDevice class to create a new device, prints a message confirming the addition, and if the maximum device limit is reached, prevents further additions.
- ***removeDevice()*** Searches for a device by name, if found, removes the device from the collection and shifts remaining devices to maintain order. Decrements deviceIndex and returns true if successful, false if the device is not found.
- ***showDevices() const.*** Displays details of all devices stored in the home.

2.3. Example

```
int main() {
    SmartHome myHome{"My Home", 5};
    SmartDevice device1{"Living Room Light 1", "Smart Light", 10};
    SmartDevice device2{device1};
    device2.setName("Living Room Light 2");
    SmartDevice device3{"Bedroom AC", "Smart Thermostat", 1500};
    SmartDevice device4{"Front Door Cam", "Security Camera", 5};

    myHome.addDevice(device1);
    myHome.addDevice(device2);
    myHome.addDevice(device3);
    myHome.addDevice(device4);

    std::cout << "\n--- All Devices ---\n";
    std::cout << "Device count: " << SmartDevice::getDeviceCount() << "\n";
    myHome.showDevices();

    std::cout << "\n--- Removing a Device ---\n";
    if(myHome.removeDevice("Living Room Light 2")) {
        std::cout << "Removed: " << "Living Room Light 2" << "\n";
    }
    std::cout << "Device count: " << SmartDevice::getDeviceCount() << "\n";
    myHome.showDevices();
    return 0;
}
```

```
Added: Living Room Light 1
Added: Living Room Light 2
Added: Bedroom AC
Added: Front Door Cam

--- All Devices ---
Device count: 8
Devices in My Home:
Name: Living Room Light 1
Type: Smart Light
Power Consumption: 10W
Status: OFF

Name: Living Room Light 2
Type: Smart Light
Power Consumption: 10W
Status: OFF
```

Name: Bedroom AC
Type: Smart Thermostat
Power Consumption: 1500W
Status: OFF

Name: Front Door Cam
Type: Security Camera
Power Consumption: 5W
Status: OFF

--- Removing a Device ---
Removed: Living Room Light 2
Device count: 7
Devices in My Home:
Name: Living Room Light 1
Type: Smart Light
Power Consumption: 10W
Status: OFF

Name: Bedroom AC
Type: Smart Thermostat
Power Consumption: 1500W
Status: OFF

Name: Front Door Cam
Type: Security Camera
Power Consumption: 5W
Status: OFF