

ให้นักศึกษา สร้าง class หาร สามารถใช้ บวก และ ลบ คุณ จาก class ก่อนหน้านี้ได้

```
Code C++
#include <iostream>
using namespace std;

class Calculator {
public:
    int add(int a, int b) {
        return a + b;
    }
};

class SubtractionCalculator : public Calculator {
public:
    int subtract(int a, int b) {
        return a - b;
    }
};

class MultiplicationCalculator : public SubtractionCalculator {
public:
    int multiply(int a, int b) {
        return a * b;
    }
};

class DivisionCalculator : public MultiplicationCalculator {
public:
    int divide(int a, int b) {
        if(b == 0) {
            cout << "Error: Cannot divide by zero." << endl;
            exit(1);
        }
        return a / b;
    }
};

int main() {
    DivisionCalculator mulCalc;

    cout << "Addition (10 + 5): " << mulCalc.add(10, 5) << endl;

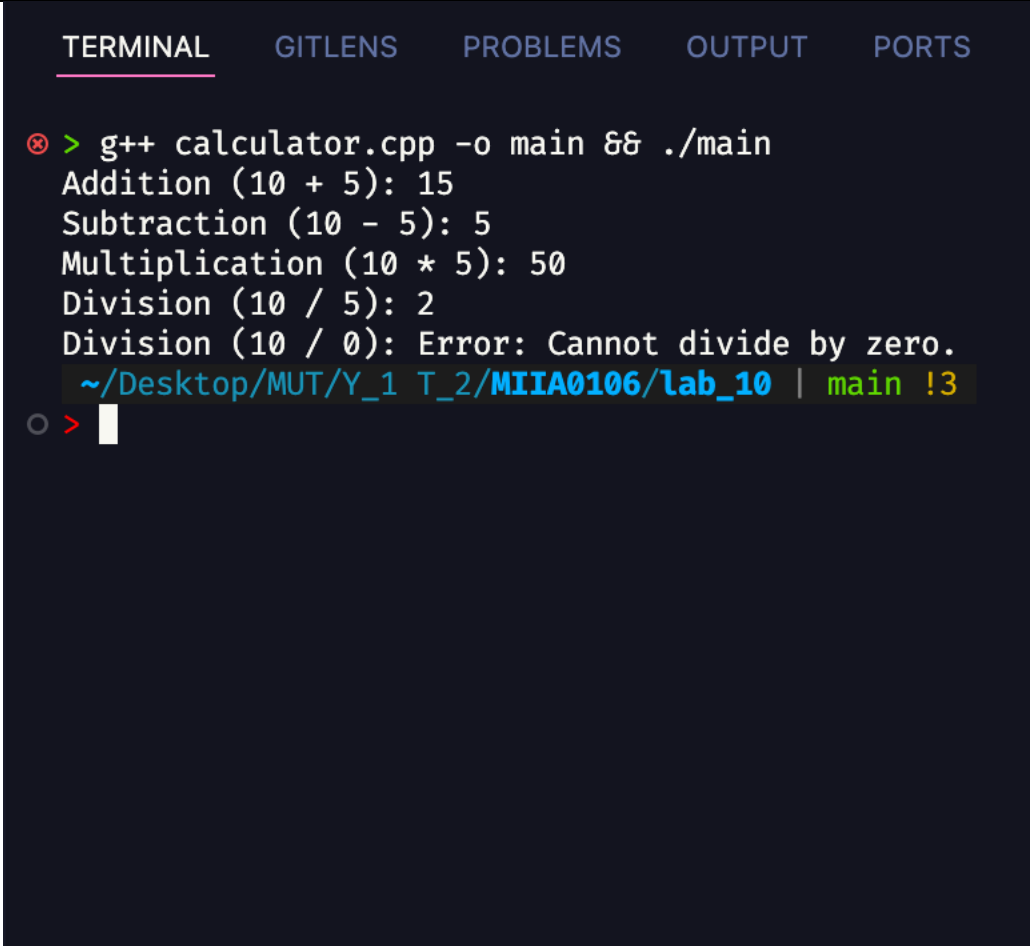
    cout << "Subtraction (10 - 5): " << mulCalc.subtract(10, 5) << endl;

    cout << "Multiplication (10 * 5): " << mulCalc.multiply(10, 5) << endl;

    cout << "Division (10 / 5): " << mulCalc.divide(10, 5) << endl;
```

```
cout << "Division (10 / 0): " << mulCalc.divide(10, 0) << endl;  
  
return 0;  
}
```

ผลการทดลอง



```
TERMINAL  GITLENS  PROBLEMS  OUTPUT  PORTS  
  
❌ > g++ calculator.cpp -o main && ./main  
Addition (10 + 5): 15  
Subtraction (10 - 5): 5  
Multiplication (10 * 5): 50  
Division (10 / 5): 2  
Division (10 / 0): Error: Cannot divide by zero.  
~/Desktop/MUT/Y_1 T_2/MIIA0106/lab_10 | main !3  
○ > █
```

Code Python

```
class Calculator:
    def add(self, a, b):
        return a + b

class SubtractionCalculator(Calculator):
    def subtract(self, a, b):
        return a - b

class MultiplicationCalculator(SubtractionCalculator):
    def multiply(self, a, b):
        return a * b

class DivisionCalculator(MultiplicationCalculator):
    def divide(self, a, b):
        if b == 0:
            return "Cannot divide by zero"
        return a / b

if __name__ == "__main__":
    mul_calc = DivisionCalculator()

    print(f"Addition (10 + 5): {mul_calc.add(10, 5)}")

    print(f"Subtraction (10 - 5): {mul_calc.subtract(10, 5)}")

    print(f"Multiplication (10 * 5): {mul_calc.multiply(10, 5)}")

    print(f"Division (10 / 5): {mul_calc.divide(10, 5)}")

    print(f"Division (10 / 0): {mul_calc.divide(10, 0)}")
```

ผลการทดลอง

--

TERMINAL

GITLENS

PROBLEMS

OUTPUT

PORTS

RO

```
● > python3 calculator.py
Addition (10 + 5): 15
Subtraction (10 - 5): 5
Multiplication (10 * 5): 50
Division (10 / 5): 2.0
Division (10 / 0): Cannot divide by zero
~/Desktop/MUT/Y_1 T_2/MIIA0106/lab_10 | main !3 ?1
○ > █
```

ให้นักศึกษา เลือกอุปกรณ์ไฟฟ้า 3 อุปกรณ์

1. การออกแบบคลาสลูก **Server**

คลาส **Server**: เป็นตัวแทนของ **server computer** ที่สามารถเปิด/ปิดได้เหมือนอุปกรณ์ทั่วไป แต่มีฟังก์ชันเพิ่มเติม:

- คุณสมบัติใหม่:
 - cpu: จำนวน core cpu ที่ต้องการ (int 1-16)
 - ram : จำนวน ram(GB) ที่ต้องการ (int 2-64)
 - storage : จำนวนพื้นที่เก็บข้อมูลถาวรที่ต้องการ(GB) (int 64-1024)
- พฤติกรรมเพิ่มเติม:
 - setCPU(int 1-16) ปรับ cpu core
 - setRam(int 2-64) ปรับ ram
 - setStorage(int 64-1024) ปรับ storage
 - getSpec() ดูข้อมูล spec ปัจจุบัน

2. การออกแบบคลาสลูก **Air Purifier**

คลาส **AirPurifier**: เป็นตัวแทนของ **เครื่องกรองฝุ่นในอากาศ** ที่สามารถเปิด/ปิดได้เหมือนอุปกรณ์ทั่วไป แต่มีฟังก์ชันเพิ่มเติม:

- คุณสมบัติใหม่:
 - Fan speed : ระดับความแรงของใบพัด (int 1-5)
 - Air score : ระดับฝุ่นในอากาศ (int 0-300)
- พฤติกรรมเพิ่มเติม:
 - setFanSpeed(int 1-5) : ปรับระดับความแรงของใบพัด
 - setAirScore(int 0-300) : ปรับระดับฝุ่นในอากาศ
 - getAirScore() : ดูค่าระดับฝุ่นในอากาศ

3. การออกแบบคลาสลูก **Speaker**

คลาส **Speaker**: เป็นตัวแทนของ **ลำโพง** ที่สามารถเปิด/ปิดได้เหมือนอุปกรณ์ทั่วไป แต่มีฟังก์ชันเพิ่มเติม:

- คุณสมบัติใหม่:
 - Volume : ระดับความดังของเสียง (int 0-100)
 - RGB light level : ระดับความสว่างของไฟแสดง visual ของคลื่นเสียงที่เล่นผ่านลำโพง(int 0-3)
- พฤติกรรมเพิ่มเติม:
 - setVolume(int 0-100) : ปรับระดับความดังของเสียง
 - setRGBLightLevel(int 0-3): ปรับระดับความสว่างของไฟแสดง visual ของคลื่นเสียงที่เล่นผ่านลำโพง
 - getConfig() : ดูข้อมูลการตั้งค่าลำโพง

Code

```
#include <iostream>
#include <string>
using namespace std;

class Device {
protected:
    string name;
    bool isOn;

public:
    Device(string deviceName) : name(deviceName), isOn(false) {}

    void turnOn() {
        isOn = true;
        cout << name << " is now ON." << endl;
    }

    void turnOff() {
        isOn = false;
        cout << name << " is now OFF." << endl;
    }

    void status() {
        cout << name << " is currently " << (isOn ? "ON" : "OFF") << "." << endl;
    }
};

class Server : public Device {
private:
    int cpu;
    int ram;
    int storage;

public:
    Server(string deviceName) : Device(deviceName), cpu(1), ram(2), storage(64) {}

    void setCPU(int core) {
        if (!isOn) {
            cout << name << " is OFF. Turn it ON to adjust CPU core." << endl;
            return;
        }
        if (core >= 1 && core <= 16) {
            cpu = core;
            cout << name << "'s CPU core is set to " << cpu << "." << endl;
        }
    }
};
```

```

    }
    else {
        cout << "Invalid CPU core! Please use a value between 1 and 16." << endl;
    }
}

void setRam(int gb) {
    if (!isOn) {
        cout << name << " is OFF. Turn it ON to adjust RAM." << endl;
        return;
    }
    if (gb >= 2 && gb <= 64) {
        ram = gb;
        cout << name << "'s RAM is set to " << ram << "GB." << endl;
    }
    else {
        cout << "Invalid RAM size! Please use a value between 2 and 64." << endl;
    }
}

void setStorage(int gb) {
    if (!isOn) {
        cout << name << " is OFF. Turn it ON to adjust storage." << endl;
        return;
    }
    if (gb >= 64 && gb <= 1024) {
        storage = gb;
        cout << name << "'s storage is set to " << storage << "GB." << endl;
    }
    else {
        cout << "Invalid storage size! Please use a value between 64 and 1024." << endl;
    }
}

void getSpec() {
    cout << name << "'s current spec: CPU core = " << cpu << ", RAM = " << ram << "GB, Storage = " <<
storage << "GB." << endl;
}
};

class AirPurifier : public Device {
private:
    int fanSpeed;
    int airScore;

public:
    AirPurifier(string deviceName) : Device(deviceName), fanSpeed(1), airScore(0) {}

```

```

void setFanSpeed(int speed) {
    if (!isOn) {
        cout << name << " is OFF. Turn it ON to adjust fan speed." << endl;
        return;
    }
    if (speed >= 1 && speed <= 5) {
        fanSpeed = speed;
        cout << name << "'s fan speed is set to level " << fanSpeed << "." << endl;
    }
    else {
        cout << "Invalid fan speed! Please use a value between 1 and 5." << endl;
    }
}

void setAirScore(int score) {
    if (!isOn) {
        cout << name << " is OFF. Turn it ON to adjust air score." << endl;
        return;
    }
    if (score >= 0 && score <= 300) {
        airScore = score;
        cout << name << "'s air score is set to " << airScore << "." << endl;
    }
    else {
        cout << "Invalid air score! Please use a value between 0 and 300." << endl;
    }
}

void getAirScore() {
    cout << name << "'s current air score: " << airScore << "." << endl;
}
};

class Speaker : public Device {
private:
    int volume;
    int rgbLightLevel;

public:
    Speaker(string deviceName) : Device(deviceName), volume(0), rgbLightLevel(0) {}

    void setVolume(int level) {
        if (!isOn) {
            cout << name << " is OFF. Turn it ON to adjust volume." << endl;
            return;
        }
    }
}

```



```

    if (level >= 0 && level <= 100) {
        volume = level;
        cout << name << "'s volume is set to " << volume << "." << endl;
    }
    else {
        cout << "Invalid volume level! Please use a value between 0 and 100." << endl;
    }
}

void setRGBLightLevel(int level) {
    if (!isOn) {
        cout << name << " is OFF. Turn it ON to adjust RGB light level." << endl;
        return;
    }
    if (level >= 0 && level <= 3) {
        rgbLightLevel = level;
        cout << name << "'s RGB light level is set to " << rgbLightLevel << "." << endl;
    }
    else {
        cout << "Invalid RGB light level! Please use a value between 0 and 3." << endl;
    }
}

void getConfig() {
    cout << name << "'s current configuration: Volume = " << volume << ", RGB light level = " <<
    rgbLightLevel << "." << endl;
}
};

int main() {

    Server webServer("Web Server");
    AirPurifier airPurifier("Air Purifier");
    Speaker musicSpeaker("Music Speaker");

    cout << "\n[Server Control]\n";
    cout << "| before turn on\n";
    webServer.status();
    webServer.setCPU(8);
    webServer.setRam(16);
    webServer.setStorage(256);
    webServer.getSpec();
    cout << "| after turn on\n";
    webServer.turnOn();
    webServer.status();
    webServer.setCPU(8);
    webServer.setRam(16);
    webServer.setStorage(256);

```

```
webServer.getSpec();  
cout << "| re-adjuste all properties\n";  
webServer.setCPU(16);  
webServer.setRam(32);  
webServer.setStorage(512);  
webServer.getSpec();  
webServer.turnOff();  
webServer.status();
```

```
cout << "\n[Air Purifier Control]\n";  
cout << "| before turn on\n";  
airPurifier.status();  
airPurifier.setFanSpeed(3);  
airPurifier.setAirScore(150);  
airPurifier.getAirScore();  
cout << "| after turn on\n";  
airPurifier.turnOn();  
airPurifier.status();  
airPurifier.setFanSpeed(3);  
airPurifier.setAirScore(150);  
airPurifier.getAirScore();  
cout << "| re-adjuste all properties\n";  
airPurifier.setFanSpeed(5);  
airPurifier.setAirScore(300);  
airPurifier.getAirScore();  
airPurifier.turnOff();  
airPurifier.status();
```

```
cout << "\n[Speaker Control]\n";  
cout << "| before turn on\n";  
musicSpeaker.status();  
musicSpeaker.setVolume(50);  
musicSpeaker.setRGBLightLevel(2);  
musicSpeaker.getConfig();  
cout << "| after turn on\n";  
musicSpeaker.turnOn();  
musicSpeaker.status();  
musicSpeaker.setVolume(50);  
musicSpeaker.setRGBLightLevel(2);  
musicSpeaker.getConfig();  
cout << "| re-adjuste all properties\n";  
musicSpeaker.setVolume(100);  
musicSpeaker.setRGBLightLevel(3);  
musicSpeaker.getConfig();  
musicSpeaker.turnOff();  
musicSpeaker.status();
```

```
    return 0;  
}
```

ผลการทดลอง

TERMINAL GITLENS PROBLEMS OUTPUT PORTS ROBOT OUTPUT COMMENTS DEBUG CONSOLE

```
● > g++ device.cpp -o main && ./main
```

```
[Server Control]
```

```
| before turn on
```

```
Web Server is currently OFF.
```

```
Web Server is OFF. Turn it ON to adjust CPU core.
```

```
Web Server is OFF. Turn it ON to adjust RAM.
```

```
Web Server is OFF. Turn it ON to adjust storage.
```

```
Web Server's current spec: CPU core = 1, RAM = 2GB, Storage = 64GB.
```

```
| after turn on
```

```
Web Server is now ON.
```

```
Web Server is currently ON.
```

```
Web Server's CPU core is set to 8.
```

```
Web Server's RAM is set to 16GB.
```

```
Web Server's storage is set to 256GB.
```

```
Web Server's current spec: CPU core = 8, RAM = 16GB, Storage = 256GB.
```

```
| re-adjuste all properties
```

```
Web Server's CPU core is set to 16.
```

```
Web Server's RAM is set to 32GB.
```

```
Web Server's storage is set to 512GB.
```

```
Web Server's current spec: CPU core = 16, RAM = 32GB, Storage = 512GB.
```

```
Web Server is now OFF.
```

```
Web Server is currently OFF.
```

```
[Air Purifier Control]
```

```
| before turn on
```

```
Air Purifier is currently OFF.
```

```
Air Purifier is OFF. Turn it ON to adjust fan speed.
```

```
Air Purifier is OFF. Turn it ON to adjust air score.
```

```
Air Purifier's current air score: 0.
```

```
| after turn on
```

```
Air Purifier is now ON.
```

```
Air Purifier is currently ON.
```

```
Air Purifier's fan speed is set to level 3.
```

```
Air Purifier's air score is set to 150.
```

```
Air Purifier's current air score: 150.
```

```
| re-adjuste all properties
```

```
Air Purifier's fan speed is set to level 5.
```

```
Air Purifier's air score is set to 300.
```

```
Air Purifier's current air score: 300.
```

```
Air Purifier is now OFF.
```

```
Air Purifier is currently OFF.
```

```
[Speaker Control]
```

```
| before turn on
```

```
Music Speaker is currently OFF.
```

```
Music Speaker is OFF. Turn it ON to adjust volume.
```

```
Music Speaker is OFF. Turn it ON to adjust RGB light level.
```

```
Music Speaker's current configuration: Volume = 0, RGB light level = 0.
```

```
| after turn on
```

```
Music Speaker is now ON.
```

```
Music Speaker is currently ON.
```

```
Music Speaker's volume is set to 50.
```

```
Music Speaker's RGB light level is set to 2.
```

```
Music Speaker's current configuration: Volume = 50, RGB light level = 2.
```

```
| re-adjuste all properties
```

```
Music Speaker's volume is set to 100.
```

```
Music Speaker's RGB light level is set to 3.
```

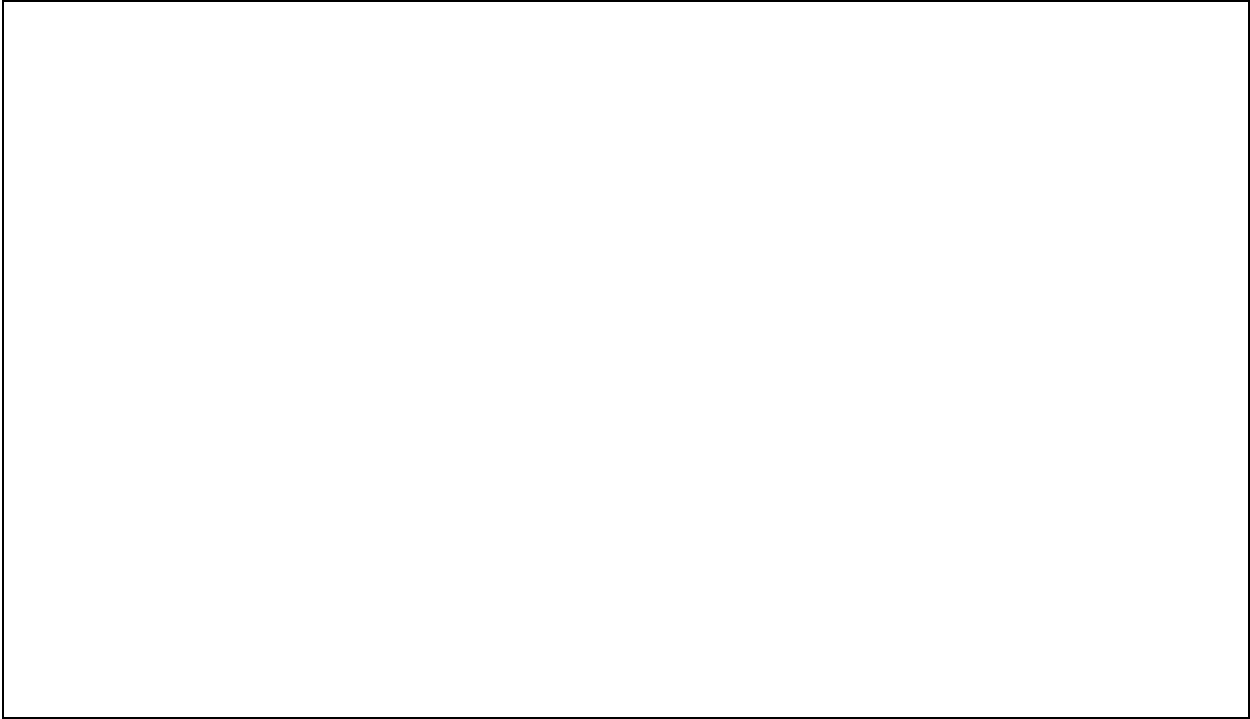
```
Music Speaker's current configuration: Volume = 100, RGB light level = 3.
```

```
Music Speaker is now OFF.
```

```
Music Speaker is currently OFF.
```

```
~/Desktop/MUT/Y_1 T_2/MIIA0106/lab_10 | main !4 ?2
```

```
○ > █
```



Code Python

```
class Device:
    def __init__(self, name):
        self.name = name
        self.is_on = False

    def turn_on(self):
        self.is_on = True
        print(f"{self.name} is now ON.")

    def turn_off(self):
        self.is_on = False
        print(f"{self.name} is now OFF.")

    def status(self):
        state = "ON" if self.is_on else "OFF"
        print(f"{self.name} is currently {state}.")

class Server(Device):
    def __init__(self, name, cpu=1, ram=2, storage=64):
        super().__init__(name)
        self.cpu = cpu
        self.ram = ram
        self.storage = storage

    def set_cpu(self, core):
        if not self.is_on:
            print(f"{self.name} is OFF. Turn it ON to adjust CPU core.")
            return
        if 1 <= core <= 16:
            self.cpu = core
            print(f"{self.name}'s CPU core is set to {self.cpu}.")
        else:
            print("Invalid CPU core! Please use a value between 1 and 16.")

    def set_ram(self, gb):
        if not self.is_on:
            print(f"{self.name} is OFF. Turn it ON to adjust RAM.")
            return
        if 2 <= gb <= 64:
            self.ram = gb
            print(f"{self.name}'s RAM is set to {self.ram}GB.")
        else:
            print("Invalid RAM size! Please use a value between 2 and 64.")

    def set_storage(self, gb):
        if not self.is_on:
```

```

        print(f'{self.name} is OFF. Turn it ON to adjust storage.')
        return
    if 64 <= gb <= 1024:
        self.storage = gb
        print(f'{self.name}'s storage is set to {self.storage}GB.")
    else:
        print("Invalid storage size! Please use a value between 64 and 1024.")

    def get_spec(self):
        print(f'{self.name}'s current spec: CPU core = {self.cpu}, RAM = {self.ram}GB, Storage = {self.storage}GB.")

class AirPurifier(Device):
    def __init__(self, name, fan_speed=1, air_score=0):
        super().__init__(name)
        self.fan_speed = fan_speed
        self.air_score = air_score

    def set_fan_speed(self, speed):
        if not self.is_on:
            print(f'{self.name} is OFF. Turn it ON to adjust fan speed.')
            return
        if 1 <= speed <= 5:
            self.fan_speed = speed
            print(f'{self.name}'s fan speed is set to level {self.fan_speed}.")
        else:
            print("Invalid fan speed! Please use a value between 1 and 5.")

    def set_air_score(self, score):
        if not self.is_on:
            print(f'{self.name} is OFF. Turn it ON to adjust air score.')
            return
        if 0 <= score <= 300:
            self.air_score = score
            print(f'{self.name}'s air score is set to {self.air_score}.")
        else:
            print("Invalid air score! Please use a value between 0 and 300.")

    def get_air_score(self):
        print(f'{self.name}'s current air score: {self.air_score}.")

class Speaker(Device):
    def __init__(self, name, volume=0, rgb_light_level=0):
        super().__init__(name)
        self.volume = volume
        self.rgb_light_level = rgb_light_level

    def set_volume(self, level):

```

```

    if not self.is_on:
        print(f"{self.name} is OFF. Turn it ON to adjust volume.")
        return
    if 0 <= level <= 100:
        self.volume = level
        print(f"{self.name}'s volume is set to {self.volume}.")
    else:
        print("Invalid volume level! Please use a value between 0 and 100.")

def set_rgb_light_level(self, level):
    if not self.is_on:
        print(f"{self.name} is OFF. Turn it ON to adjust RGB light level.")
        return
    if 0 <= level <= 3:
        self.rgb_light_level = level
        print(f"{self.name}'s RGB light level is set to {self.rgb_light_level}.")
    else:
        print("Invalid RGB light level! Please use a value between 0 and 3.")

def get_config(self):
    print(f"{self.name}'s current configuration: Volume = {self.volume}, RGB light level = {self.rgb_light_level}.")

def main():

    web_server = Server("Web Server")
    air_purifier = AirPurifier("Air Purifier")
    music_speaker = Speaker("Music Speaker")

    print("\n[Server Control]\n")
    print("| before turn on")
    web_server.status()
    web_server.set_cpu(8)
    web_server.set_ram(16)
    web_server.set_storage(256)
    web_server.get_spec()
    print("| after turn on")
    web_server.turn_on()
    web_server.status()
    web_server.set_cpu(8)
    web_server.set_ram(16)
    web_server.set_storage(256)
    web_server.get_spec()
    print("| re-adjuste all properties")
    web_server.set_cpu(16)
    web_server.set_ram(32)
    web_server.set_storage(512)

```



```
web_server.get_spec()
web_server.turn_off()
web_server.status()
```

```
print("\n[Air Purifier Control]\n")
print("| before turn on")
air_purifier.status()
air_purifier.set_fan_speed(3)
air_purifier.set_air_score(150)
air_purifier.get_air_score()
print("| after turn on")
air_purifier.turn_on()
air_purifier.status()
air_purifier.set_fan_speed(3)
air_purifier.set_air_score(150)
air_purifier.get_air_score()
print("| re-adjuste all properties")
air_purifier.set_fan_speed(5)
air_purifier.set_air_score(300)
air_purifier.get_air_score()
air_purifier.turn_off()
air_purifier.status()
```

```
print("\n[Speaker Control]\n")
print("| before turn on")
music_speaker.status()
music_speaker.set_volume(50)
music_speaker.set_rgb_light_level(2)
music_speaker.get_config()
print("| after turn on")
music_speaker.turn_on()
music_speaker.status()
music_speaker.set_volume(50)
music_speaker.set_rgb_light_level(2)
music_speaker.get_config()
print("| re-adjuste all properties")
music_speaker.set_volume(100)
music_speaker.set_rgb_light_level(3)
music_speaker.get_config()
```

```
if __name__ == "__main__":
    main()
```

ผลการทดลอง

TERMINAL GITLENS PROBLEMS OUTPUT PORTS ROBOT OUTPUT COMMENTS DE

● > python3 device.py

[Server Control]

| before turn on

Web Server is currently OFF.

Web Server is OFF. Turn it ON to adjust CPU core.

Web Server is OFF. Turn it ON to adjust RAM.

Web Server is OFF. Turn it ON to adjust storage.

Web Server's current spec: CPU core = 1, RAM = 2GB, Storage = 64GB.

| after turn on

Web Server is now ON.

Web Server is currently ON.

Web Server's CPU core is set to 8.

Web Server's RAM is set to 16GB.

Web Server's storage is set to 256GB.

Web Server's current spec: CPU core = 8, RAM = 16GB, Storage = 256GB.

| re-adjuste all properties

Web Server's CPU core is set to 16.

Web Server's RAM is set to 32GB.

Web Server's storage is set to 512GB.

Web Server's current spec: CPU core = 16, RAM = 32GB, Storage = 512GB.

Web Server is now OFF.

Web Server is currently OFF.

[Air Purifier Control]

| before turn on

Air Purifier is currently OFF.

Air Purifier is OFF. Turn it ON to adjust fan speed.

Air Purifier is OFF. Turn it ON to adjust air score.

Air Purifier's current air score: 0.

| after turn on

Air Purifier is now ON.

Air Purifier is currently ON.

Air Purifier's fan speed is set to level 3.

Air Purifier's air score is set to 150.

Air Purifier's current air score: 150.

| re-adjuste all properties

Air Purifier's fan speed is set to level 5.

Air Purifier's air score is set to 300.

Air Purifier's current air score: 300.

Air Purifier is now OFF.

Air Purifier is currently OFF.

[Speaker Control]

| before turn on

Music Speaker is currently OFF.

Music Speaker is OFF. Turn it ON to adjust volume.

Music Speaker is OFF. Turn it ON to adjust RGB light level.

Music Speaker's current configuration: Volume = 0, RGB light level = 0.

| after turn on

Music Speaker is now ON.

Music Speaker is currently ON.

Music Speaker's volume is set to 50.

Music Speaker's RGB light level is set to 2.

Music Speaker's current configuration: Volume = 50, RGB light level = 2.

| re-adjuste all properties

Music Speaker's volume is set to 100.

Music Speaker's RGB light level is set to 3.

Music Speaker's current configuration: Volume = 100, RGB light level = 3.

~/Desktop/MUT/Y_1 T_2/MIIA0106/lab_10 | main !4 ?2