**ให้นักศึกษา สร้าง class หาร สามารถใช้ บวก และ ลบ คูณ 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

|  |
| --- |
|  |

|  |
| --- |
| 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

|  |
| --- |
|  |

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

|  |
| --- |
|  |

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

|  |
| --- |
|  |