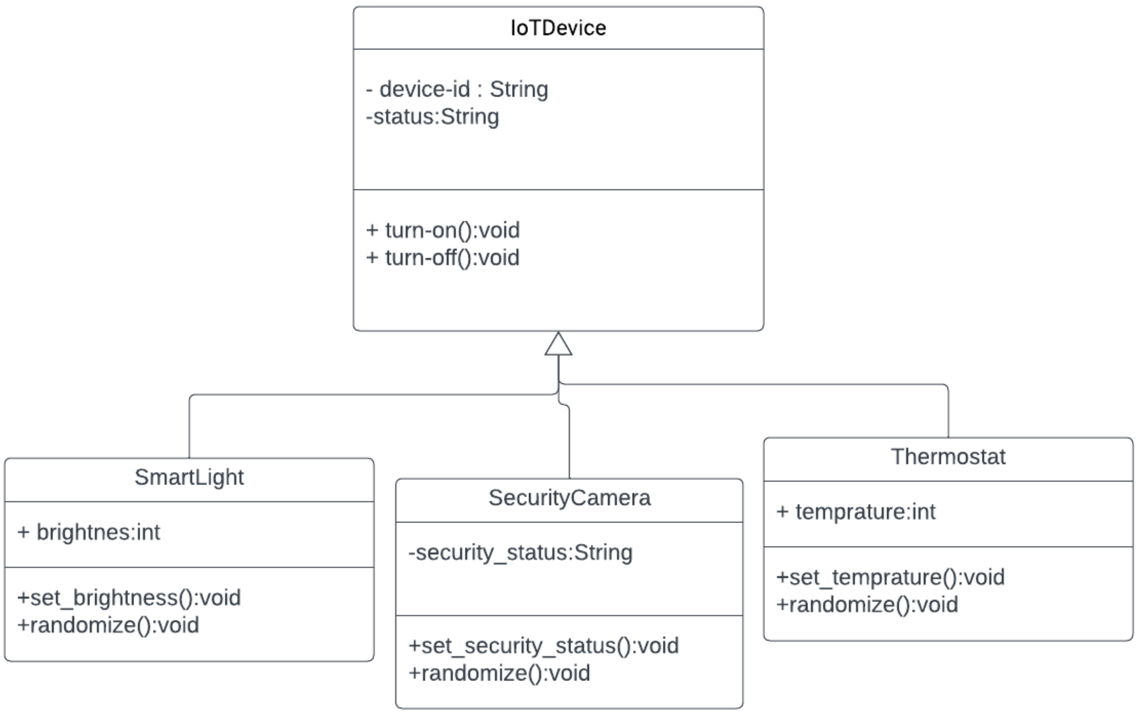
Python Assignment

Name: Abdel Hamid Khaled

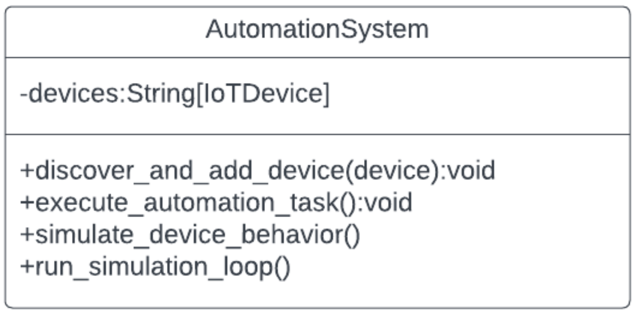
Code:UMI93X

Documentation:

Part1:



Part2:


AutomationSystem
-devices:String[IoTDevice]
+discover_and_add_device(device):void
+execute_automation_task():void
+simulate_device_behavior()
+run_simulation_loop()

Method Description:

Part 1:

IoTDevice Basics:

__init__: Just setting up our device with a name and making sure it's off to start.

turn_on: Makes the device go 'on'. Simple as flipping a switch.

turn_off: Makes the device go 'off'. Switching off the button.

SmartLight :

__init__: Gets our smart light ready with zero brightness (basically off).

set_brightness: Lets you slide the brightness up or down.

randomize: Randomly changes brightness.

Thermostat:

__init__: Sets up our thermostat to a standard temp.

set_temperature: You can change the temp to whatever you feel like.

randomize: Changes the temp by itself.

SecurityCamera:

__init__: Puts our camera in chill mode.

set_security_status: If it sees something odd, it goes into alert mode.

randomize: Sometimes decides to go into alert mode on its own.

Part 2:

AutomationSystem Class Methods:

__init__: Gets the system ready to add devices.

discover_and_add_device: Adds a new device to keep an eye on.

execute_automation_tasks: Goes through all devices to do things like save energy or record video if there's motion.

simulate_device_behavior: Makes each device act like someone's messing with it.

run_simulation_loop: Keeps checking on all devices, acting and reacting as needed, like a continuous check-up.

How to Use the Methods:

For Part 1, instantiate each device and test their methods to simulate turning devices on and off and adjusting properties like brightness and temperature.

For Part 2, create an instance of AutomationSystem and add your device instances to it. Then, start the simulation loop to see the automation system manage the devices. The loop will periodically update device states based on random behavior and predefined rules.

Test Cases:

Test Case for SmartLight:

Description: Ensure that the SmartLight can be turned on and have its brightness adjusted.

Steps:

1. Create an instance of SmartLight.
2. Call the turn_on() method.
3. Set the brightness to 50 using set_brightness(50).

Expected Outcome: The SmartLight instance should have a status of 'on' and a brightness level of 50.

Test Case for Thermostat:

Description: Ensure that the Thermostat can be turned off and its temperature adjusted.

Steps:

1. Create an instance of Thermostat.
2. Call the turn_off() method.
3. Set the temperature to 18 using set_temperature(18).

Expected Outcome: The Thermostat instance should have a status of 'off' and a temperature setting of 18.

Test Case for SecurityCamera:

Description: Ensure that the SecurityCamera can detect motion.

Steps:

1. Create an instance of SecurityCamera.
2. Simulate motion detection by calling set_security_status('alert').

Expected Outcome: The SecurityCamera instance should have a security status of 'alert'.

For Part2:

Automation Rule - Thermostat:

Description: Ensure that the AutomationSystem can adjust the Thermostat temperature based on rules.

Steps:

1. Create an instance of AutomationSystem.
2. Add a Thermostat instance with a temperature above the threshold.
3. Execute execute_automation_tasks().

Expected Outcome: The temperature of the Thermostat should be set to 20 if it was above 25.

Automation Rule - SmartLight:

Description: Ensure that the SecurityCamera starts recording when motion is detected.

Setup: Implement a start_recording method in the SecurityCamera class.

Steps:

1. Create an instance of AutomationSystem.
2. Add a SecurityCamera instance with security_status set to 'alert'.
3. Execute execute_automation_tasks().

Expected Outcome: start_recording method of SecurityCamera should be called.