Topic 1: Software Design 21P

Your task is to design the architecture for a smart home controller. Use design patterns and follow design principles to adhere to the description below. Your smart home controller solution should provide multiple components:

1. **Devices.** A smart home combines a variety of devices that can be controlled. For each device it should be possible to activate and de-activate the device and query the device whether it is currently active. For now, we only want to consider one kind of device, namely lights. Within our smart home, we want to be able to control different kinds of lights using the previously described interface. For example, we can have simple lights whose internal state can just be toggled

   ```
   class SimpleLight {
     var turnedOn: Boolean = false

     def toggle: Unit = {
       turnedOn = !turnedOn
     }
   }
   ```

   or we can have dimmable lights whose brightness can be adjusted in steps (`level = 0` indicates that the light is powered off, `level = 100` indicates that the light is fully powered on).

   ```
   class DimmableLight {
      var level = 0
    }
   ```

2. **Scenarios.** Within the smart home it should be possible to define multiple scenarios. Scenarios describe which actuators are to be activated based on some pre-condition, where a condition can be evaluated to some Boolean value. The simplest base scenario we want to support is to activate a light with no pre-condition. For example, activating the scenario "Kitchen" turns on all lights in the kitchen, while the scenario "Living Room" turns on all lights in the living room. A scenario has a name and comprises a set of pre-conditions and devices and provides a way to be activated, where the latter activates all devices that are part of the scenario if the pre-condition is satisfied.

   In addition to simple scenarios as described before, we also want to support complex scenarios that can additionally contain other scenarios. For example, the scenario "Party" can contain the scenarios "Kitchen" and "Living Room". When this scenario is activated, both sub-scenarios are activated, which means that both the lights in the kitchen and in the living room are powered on.

3. **Creating Scenarios.** Finally, your implementation should provide a way to create scenarios step-by-step, by adding arbitrary many devices, pre-conditions or scenarios.

Sketch your design as Scala code. Use expressive class and method names. **Shortly document the used patterns and the responsibilities of each class you add in the pattern. Briefly describe how the required extensibility was implemented in your design.**

The sketch has to contain code for:

- Representing devices
- Integrating devices with variable interfaces, using the example of lights
- Representing scenarios

For the "Creating Scenarios" requirement it is sufficient if you briefly state which design pattern can be used to provide the functionality. Also state which of your classes play which role in it.

General Hints:

- You can omit method implementations that are not necessary for your pattern(s) by using `???`.
- For the devices it is sufficient if you show how to integrate either `SimpleLight` or `DimmableLight`, however, your implementation should be open for extension with respect to other lights with different interfaces.
- For the scenarios, you can assume there is a method `isTrue` which returns true if all conditions contained in the set are satisfied